UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

JULIO CÉSAR SANTOS DOS ANJOS

Big Data Processing using Hybrid Infrastructures such as Cloud Computing and Desktop Grids

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Prof. Dr. Cláudio Fernando Resin Geyer
Advisor

Prof. Dr. Gilles Fedak
Coadvisor

Porto Alegre, February 2017

CIP – CATALOGING-IN-PUBLICATION

Santos dos Anjos, Julio César

Big Data Processing using Hybrid Infrastructures such as Cloud Computing and Desktop Grids / Julio César Santos dos Anjos. – Porto Alegre: PPGC da UFRGS, 2017.

150 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2017. Advisor: Cláudio Fernando Resin Geyer; Coadvisor: Gilles Fedak.

1. Big Data. 2. MapReduce. 3. Hybrid Infrastructures. 4. Distributed Systems. 5. Cloud Computing. 6. Desktop Grid. I. Resin Geyer, Cláudio Fernando. II. Fedak, Gilles. III. Título.

“Domine Iesu Christe, qui dixisti, apostolis tuis.
Pacem relinquo vobis, pacem meam do Vobis.
Ne Respiciais peccata Nostras, sed fidem
ecclésiae tuae.
Eam que secundum voluntatem
Tuam pacificare et co adunares dignéris;
Qui vivis et régnas in Saecula Saeculorum.
Pax Domini sit semper vobiscum,
et cum Spiritu tuo.”
— Papa João Paulo II

ACKNOWLEDGEMENTS

To God.
To my wife by time and patience.
To my parents.
To my advisor Prof. Claudio Geyer.
To my advisor Prof. Gilles Fedak.
To the Big Data team at UFRGS.
To CAPES and CAPES Process BEX 14966/13-1 by financial support.

LIST OF ABBREVIATIONS AND ACRONYMS

API	Application Program Interface.
AWS	Amazon Web Services.
BLOBs	(**B**inary **L**arge **O**bject**s**).
BOINC	Berkeley open infrastructure for network computing - a middleware to volunteer computing.
BoT	Bag-of- tasks. Applications that do not have a dependency between them.
B2B	Business-to-Business transaction.
B2C	Business-to-Consumer transaction.
CSP	Cloud service provider.
DFR	Dominant resource fairness algorithm.
DFRH	Dominant resource fairness for heterogeneous servers.
EC2	Elastic compute cloud from Amazon web service.
ERP	Enterprise resource planning.
GCE	Google compute engine.
HDFS	Hadoop distributed file system.
I/O	Input and output data.
IoT	Internet of Things.
LHC	Large Hadron Collider.
MPI	Message-passing interface.
NoSQL	Not only SQL.
QoS	Quality of services.
RDBMS	Relational database management system.
VC	Volunteer Computing.
SLA	Service-level agreement.
VCG	Vickrey-Clarke-Groves algorithm.
VM	Virtual machine.
WORM	Write-once-read-many. A data storage technology.

LIST OF ALGORITHMS

LIST OF FIGURES

LIST OF TABLES

CONTENTS

ABSTRACT

A huge volume of data is produced every day, from the information provided by social networks (such as Facebook, Instagram, Whatsapp, etc) or that generated by sensors on mobile devices, including Big Data applications like Google Searches. This deluge of data requires ever more computational resources to process the information more quickly. Although Cloud has grown rapidly in recent years, it still suffers from a lack of standardization and management resources. The users who need to execute applications may not know how to map their requirements to the available resources. This lack of knowledge about the Cloud provider infrastructure leads either to overestimating or underestimating the required processing capacity for tasks. This complex scenario raises enormous challenges for researchers of new systems and infrastructure kinds. On the other hand, it provides several opportunities for the researcher to find solutions for Big Data Analytics. This work establishes: i) a new platform called SMART which offers Big Data Analytics in a Lambda architecture within a hybrid infrastructure; ii) presents a simulator called BIGhybrid to be a toolkit for the study of Big Data Analytics in hybrid infrastructures. Its goal is to enable the user achieves the nearest configuration for Big Data applications into deployment in real-world environments. In addition, defines data distribution strategies in this complex scenario for reducing the risks of trouble caused by common configuration mistakes; iii) evaluates the use of the Dispatcher module in the SMART platform and iv) defines strategies for the use of Desktop Grid and Cloud Computing in a geo-distributed environment within a hybrid infrastructure. The boundaries to produce an acceptable quality of service (QoS) are presented. Such limits can be summarized as the relation between volunteer hosts and stable nodes, the shape of data distribution, load balancing strategies and relation Φ to resource allocation. Although it can be carried out in the real-world, an experimental evaluation on a large scale is only possible through simulation owing to the reproducibility and predictability of environmental features. These experiments indicate a good performance of the SMART platform in low and high-scale in simulated environments.

Keywords: Big Data, MapReduce, Hybrid Infrastructures, Distributed Systems, Cloud Computing, Desktop Grid.

Processamento Big Data usando Infra-estruturas Híbridas como Computação em Nuvem e Grade de Desktop

RESUMO

Um grande volume de dados é produzido todos os dias, desde informações fornecidas por redes sociais (tais como Facebook, Instagram, Whatsapp, etc) ou geradas por sensores em dispositivos móveis, até aplicações Big Data como a busca do Google. Esta inundação de dados requer cada vez mais recursos computacionais para processar informações mais rapidamente. Embora *Cloud* tenha crescido rapidamente nos últimos anos, ela ainda sofre com falta de padronização e gerenciamento de recursos adequados. Os usuários que necessitam executar aplicações podem não saber como mapear seus requisitos de sistemas para os recursos disponíveis. Esta falta de conhecimento sobre a infraestrutura dos provedores de nuvem leva a superestimar ou subestimar a capacidade de processamento necessária para as tarefas. Este cenário complexo apresenta enormes desafios para os pesquisadores em termos de sistemas e tipos de infraestruturas. Por outro lado, ele oferece várias oportunidades para o pesquisador encontrar soluções para a análise de Big Data. Este trabalho estabelece: i) uma nova plataforma chamada SMART que oferece a análise de Big Data em uma arquitetura Lambda sobre uma infraestrutura híbrida; ii) apresenta um simulador chamado BIGhybrid para ser um conjunto de ferramentas para o estudo da análise de Big Data em infraestruturas híbridas. Este permite que o usuário encontre as configurações mais próxima para as aplicações Big Data na implantação em ambientes reais. Ainda, define estratégias para a distribuição de dados neste cenário complexo para reduzir os riscos de problemas causados por erros comuns de configurações; iii) avalia o uso do módulo Despachante na plataforma SMART e iv) define estratégias para o uso de *Desktop Grid* e computação em nuvem em um ambiente geo-distribuído em uma infraestrutura híbrida. O objetivo é encontrar algumas das restrições a uma qualidade de serviços (QoS) aceitável. Tais restrições estão relacionadas com a relação entre máquinas voluntárias e nós estáveis, distribuição de dados, estratégias de balanceamento da carga e assim por diante. Embora isto possa ser construído em um ambiente real, uma avaliação experimental em larga escala é somente possível através de simulação devido às características de reprodutibilidade e previsibilidade de características ambientais. Os experimentos indicam um bom desempenho da plataforma SMART em baixa escala em um ambiente real.

Keywords: Big Data, MapReduce, Infra-estrutura Híbrida, Sistemas Distribuídos, Comptação em Nuvem, Grade de Desktop.

1 INTRODUCTION

Mankind is producing an ever increasing amount of data. According to the National Security Agency (NSA)[1], by 2020 there will be around 40 Zettabytes (40,000,000 Petabytes) of data that will require processing of some sort. This volume of data requires processing capabilities beyond those that the current IT infrastructure can provide. In addition, the data input can originate from different sources, such as social interaction, scientific research, business activities and government decisions (DOAN; HALEVY; IVES, 2012), (CHEN; ZHANG, 2014). Social customs have changed since the Internet revolution in the 1990s, with the growth of mobile devices in the last 20 years and the social network phenomenon of the last decade. Today, people are posting millions of photos per day on Facebook or Instagram that are shared with other people or groups of friends. Every day, billions of tweets are being exchanged on Twitter from mobile devices. The number of smartphones and access to broadband has been increasing steadily both in developed and emerging economies.

Although, Cloud Computing (Cloud) has grown rapidly in recent years, it still suffers from a lack of standardization and the availability of homogeneous management resources (TOOSI; CALHEIROS; BUYYA, 2014). *Private clouds* are used exclusively by a single organization, that keeps careful control of its performance, reliability and security, but might have low scalability for Big Data processing requirements. *Public clouds* have an infrastructure that is based on a specific Service Level Agreement (SLA) which provides services and quality assurance requirements with minimal resources in terms of processing, storage and bandwidth.

The Cloud Service Provider (CSP) manages its own physical resources, and only provides an abstraction layer for the user. This interface might vary depending on the provider, but maintains properties like elasticity, insulation and flexibility (SAKR et al., 2011). *Hybrid clouds* are a mix of the previous two systems and enable the *cloud bursting* application deployment model, where the excess of processing from *Private cloud* is forwarded to the *Public cloud* provider. Cloud providers can negotiate a special agreement as a means of forming a *Cloud federated* system, where providers that operate with low usage, might be able to lease a part of their resources to other federation member to avoid wasting their idle computational resources (TOOSI; CALHEIROS; BUYYA, 2014).

MapReduce (MR) (DEAN; GHEMAWAT, 2010) is a programming framework proposed by Google that is currently adopted by many large companies, and has been employed as a successful solution for data processing and analysis. Hadoop (WHITE, 2012) is the most popular open-source implementation of MR. Cloud computing has increasingly been used as a platform for business applications and data processing (PHAM et al.,

[1]The Next Wave, Vol. 20, No. 4, November 2014

2015). Cloud providers offer Virtual Machines (VMs), storage, communication and queue services to customers in a pay-as-you-go scheme. These resources are used for deploying Hadoop clusters as the basis of Big Data analytic processing.

Since there is a wide range of data sources, the collected datasets have different noise levels, redundancy and consistency (CHEN; MAO; LIU, 2014). Carrying out a Big Data analysis is still an arduous task (ASSUNCAO et al., 2015). Moreover, until now, the software infrastructure for Big Data has had features and tools that are insufficient to solve real problems, especially for the analysis of real-time applications (CHEN; ZHANG, 2014). Thus, it is necessary to find new ways of processing Big Data which can exploit idle computational resources and allow them to be combined with Cloud to add more scalability.

The emerging systems are highly heterogeneous environments with variable structures, where resources can be added or removed if necessary (VASILE et al., 2015). Desktop Grid (DG) is a platform that has received a good deal of attention from the scientific community because of its use in Big Data applications (LIN et al., 2010), (ANJOS et al., 2012) and (TANG; HE; FEDAK, 2015). The concept of a hybrid infrastructure in this Thesis can be defined as a mix of public or private Cloud with Desktop Grid, similar to the environment explored by (TANG; HE; FEDAK, 2015), which will be detailed in the next Chapters. This work explores these *hybrid infrastructures* to find the best alternatives for heterogeneous resource allocation in Big Data systems. A new platform for the *hybrid infrastructures*, called SMART, is proposed to enable the deployment of DG and Cloud. Also, a simulator called BIGhybrid was designed on the basis of the features required for this infrastructure. It is a toolkit for Big Data analytics. In addition, a set of strategies will be defined and evaluated for the use of DG and Cloud in hybrid infrastructures.

1.1 Motivation

The way of conducting science has changed throughout the centuries in different fields, such as, biology, physics, astronomy and meteorology (HEY; TANSLEY; TOLLE, 2009),(CHEN; ZHANG, 2014). In the 14^{th} Century, science was based on observations of physical phenomena and the precision of data was measured in terms of the size of the visual field. Today, scientists observe the universe with sophisticated telescopes which generate millions of images that need a lot of computation time for data analysis. For instance, the Large Synoptic Survey Telescope (LSST) produces 15 Terabyte of raw data per night (LSST, 2016). CERN's Large Hadron Collider (LHC) project is capable of capturing around 3 Petabyte of data per month. Collisions in the LHC have generated about 75 Petabyte of data in the past three years (CERN, 2016). In the recent past, finding information in large datasets was only possible through a relational database. The scientists had to choose the right query to obtain the correct result. In Big Data, the queries can include both structured, semi-structured or unstructured data, such as audio, video, web pages, text and so on, and it can originate from multiple data sources. Multimedia, social networks and Internet of Things (IoT) are collecting more and more information, which means that Big Data will have a growing prospect of being able to create value for businesses and consumers (CHEN; MAO; LIU, 2014). The purpose of Big Data is to amass a lot of data and find anomalies or patterns in it, so that value and significance can be added. However, it is common to find multiple data in different places, since the cost of data transfers for a single site is prohibitive owing to the limitations of size and bandwidth (JAYALATH; STEPHEN; EUGSTER, 2014), (HEINTZ et al., 2014).

The scientific community has proposed Cloud as an infrastructure for Big Data, with permanent storage and management facilities in large-scale disordered datasets, as well as Distributed File Systems (DFS) and NoSQL databases as a solution for data storage (CHEN; MAO; LIU, 2014). However, the CSPs in Big Data analytic applications still need to explore the issue of resource allocation in an effective manner so as to achieve economies of scale and elasticity. In the study of (ASSUNCAO et al., 2015) the authors state that it is possible to determine the main dependencies of five Big Data dimensions (*Volume, Velocity, Variety, Value* and *Veracity*). *Volume* depends on a hardware infrastructure to achieve scalability and *Value* depends on how much Big Data must be creatively and effectively exploited to improve efficiency and the quality needed to assign *Veracity* to information. The amount of data in *Variety* might originate from different sources, such as historical information, pictures, sensor information, satellite data and other structured or unstructured sources.

Both the Cloud Service Providers - CSPs - (public and private) maintain a service level based on the number of nines (99.999999....%) (MANSOURI; TOOSI; BUYYA, 2013); this refers to how much time the services are available for users. The service providers guarantee properties like flexibility, unlimited storage, system isolation guarantees and elasticity. These characteristics hide the hardware features from the users. However, the time and cost required for data transfers in Cloud, and the difficulty of maintaining a distribution with a fine-grained control of the hardware features, are an open problem in Big Data implementations (CHEN; ZHANG, 2014).

In addition to Cloud, several other types of infrastructure are able to support data-intensive applications. DGs, for instance, have a large number of users around the world who donate idle computing power to multiple projects (CÉRIN; FEDAK, 2012). DGs have been applied in several domains such as bio-medicine, weather forecasting, and natural disaster prediction. Merging DG with Cloud into hybrid infrastructures could provide a more affordable means of data processing. Several initiatives have implemented Big Data with Hadoop as a MR framework, for instance (LIN et al., 2010),(COSTA; SILVA; DAHLIN, 2011) and (LU et al., 2012). However, although MR has been designed to exploit the capabilities of commodity hardware, its use in a hybrid infrastructure is a complex task because of the resource heterogeneity and its high churn rate. This is usual for Desktop Grids but uncommon for Clouds. In addition, hybrid infrastructures are environments which have geographically distributed resources (HEINTZ et al., 2014) in heterogeneous platforms with mixing of Cloud, Grids and DG.

Before discussing the main goals and challenges of this Thesis, it is necessary to understand the nature of its adopted concept for hybrid environments. The virtual resources of Cloud are shared with different users of the same physical hardware and this leads to I/O competition when there are disk access and network traffic. In this context, although the CSPs make every effort to ensure uniform resources for all the users, unfortunately, the Cloud resources are in fact heterogeneous. On the other hand, DGs are heterogeneous resources when the task is processed with exclusive hardware through a Free Time window. Currently, the infrastructures provide limited or no assistance to the heterogeneous platform in Big Data analytical systems, (as the Related Work section seeks to demonstrate). However, a complex scenario like this can make several mistakes before an acceptable platform is able to execute Big Data applications with accuracy. Thus, it is necessary to build a minimal platform that can enable these different infrastructures to be merged and to provide a simulation tool that allows repeatability, reproducibility, and stability for the experiments conducted in these environments. These features are important to allow a set

of strategies to be defined that can assure the best load balancing possible for the Big Data applications and reduce the development time.

1.2 Problem Statement

The main goal of CSPs in public clouds is to show how to allocate all the free hardware slots to obtain the best use of resources. In view of this, although there are system isolation guarantees, an available resource can be shared with other users, when they are accessing the same physical hardware. The competition for resources creates a natural delay in the accesses to hardware which might lead to a poor performance and uncontrolled bottlenecks, especially with regard to disk performance. This delay will depend on the hardware features over which the user does not have any control in Cloud environments (XAVIER et al., 2015). In Big Data applications, "disk competition" is expected to create the "*disk contention phenomenon*", which was first observed by (ZAHARIA et al., 2008). The Cloud capabilities are organized by the providers to be as homogeneous as possible. However, in fact, the environment becomes heterogeneous when user applications with different execution profiles are carried out at the same time. Could cloud infrastructures that provide configurations that are sufficient for the users have a fine-grained application control ?

While in Cloud the capabilities are shared and stable, in DG the resources are dedicated and volatile. Owing to volatility problems, the approach to DG environments must involve of establishing a relation between stable and unstable nodes[2] that will be the nearest relation of 3 volatile nodes to 1 cloud node, that was demonstrated in the work of (KONDO et al., 2009). We can ask ourselves if this is feasible in the context of Big Data.

However, these environments are complex and hard to manage and adjust, for instance; the AWS (AMAZON, 2016) provides many instances where the user must choose between several hardware properties, such as CPU, memory, storage and network capacity. If on the one hand, it can be confusing for users without expertise, on the other hand, it may provide insufficient information for an expert administrator. The use of a testbed for evaluations is quite difficult owing to environmental factors and reproducibility problems. In DG, it is difficult to determine the exact instant that a node will have a shutdown, although it is possible that there is a probabilistic behavior for these environments. Could DG be a good alternative ? What are the conditions ?

Hence, it is necessary to achieve a balance between Cloud and DG when executing Big Data applications, before seeking answers to questions like the following: How is it possible to determine if a hybrid infrastructure is feasible? What are the main features of hybrid environments? What are the best strategies for data split or task distribution in a hybrid environment? How is it possible to ensure reproducibility when conducting the experiments? What is the ideal relation between volatile and stable nodes in hybrid infrastructures? Is it possible to exploit the low cost of DG environments for Big Data applications? The main goal is to find answers to the questions regarding the feasibility of adopting hybrid infrastructures for Big Data which impose acceptable constraints. These questions are summarized in the following hypothesis:

The Big Data applications can be feasible for hybrid infrastructures, such as Cloud and Desktop Grid, if a set of strategies are deployed for data and task distribution in these environments.

[2]To simplify this issue, from now on, node and machine will have the same meaning.

1.3 Thesis and Challenges

The computational resources available are too scarce to achieve trustworthy validations and there are several problems about carrying out reproducible experiments on a large scale. Because of this, a simulator was developed with the aim of enabling Big Data analytics on a large scale, based on a system shown in (ANTONIU et al., 2013) and after being extended to a geographically-distributed environment. Some strategies were evaluated for a hybrid infrastructure with a dispatcher layer (as discussed in Chapter 3). Three approaches make it possible to find answers and also determine the best strategies (and main restrictions) when implementing Big Data within hybrid infrastructures.

First, it is necessary to define minimal real-world functions that are required to reproduce this environment. Second, it is necessary to simulate the environments with a relative degree of accuracy, as well as compare the features of these environments with real-world experiments on a scale that can ensure a reliable analysis. Third, the two previous approaches can be combined, in a mechanism devised to produce the nearest real-world environmental configuration on the basis of the simulations and thus achieve the best execution time possible.

The environment simulation was the path chosen for finding answers to the questions raised in this Thesis. Moreover, in another scenario, it might be possible to include this simulation mechanism in the performance evaluation as a prediction tool for Big Data in hybrid environments. The challenge of making an accurate simulator is a) how to create a robust architecture for environment simulations through a careful study of the properties and characteristics of the real-world systems, and b) how to make use of the hardware setup from the Cloud and DGs to ensure the reproducibility of environment behavior. These goals can be achieved with the BIGhybrid simulator built during this work. BIGhybrid is a toolkit for MR simulation in hybrid environments where algorithms and strategies can be analyzed for studying hybrid platforms.

The aim of the SMART platform (Small & Medium-sized Enterprise Data Analytics in Real Time) is to set a platform for Big Data analysis which can be applied to a hybrid infrastructure. In comparison with the characteristics of the other works in the literature, this platform uses Cloud and DGs as its core infrastructure and is embedded in a Lambda Architecture for Big Data processing in batch mode and real-time. The platform will be evaluated with the BIGhybrid simulator through strategies which are defined for the hybrid infrastructures that are adopted in Big Data analytics. The evaluated layers in this work consists of the dispatcher (as a task scheduler and for data distribution) and core engine to Big Data analytics (for homogeneous and volatile environments). Its main purpose is to identify some operational constraints and to enable the Big Data analytics for hybrid infrastructures. Thus, some strategies will be employed for the evaluation of the Global Dispatcher in the SMART platform. The evaluation will make it possible to analyze the use and feasibility of homogeneous and volatile environments in the SMART platform in the context of Big Data applications. Its main purpose is to identify some appropriate boundaries for deployment.

1.4 Contributions

The main contributions of this research work are as follows:

- **To provide a hybrid infrastructure environment**.
  The desirable features for a hybrid infrastructure are generalized on the basis of the

main characteristics of the BlobSeer-Hadoop (NICOLAE et al., 2010) and BitDew-MapReduce (FEDAK; HE; CAPPELLO, 2009) environments. In addition, other related hybrid systems are evaluated to form the hybrid environment considered here.

- **To provide a platform that can allow the deployment of Big Data analytics within hybrid infrastructures**.
  The definition of the hybrid infrastructures enables a new platform to be designed for Big Data analytics by means of a Lambda architecture. This platform forecasts the execution of input data at the edges through a desktop grid environment.

- **To create a simulator as an analytical tool of hybrid infrastructures and for Big Data analytics**.
  A simulator was designed with the aim of being an analycal tool that could provide consistent studies for the Big Data algorithms in hybrid environments. This tool ensures an environment that can be controlled for the reproducibility and repeatability of the experiments.

- **To find an accurate method of conducting an analysis of a hybrid infrastructure**.
  A statistical study on the simulator is carried out to determine the accuracy and precision of the results in comparison with simulated and real-world experiments. The simulator can follow the execution traces of a real environment, such as volunteer computing, and thus be able to reproduce the execution behavior of the real-world.

- **To determine strategies for the use of hybrid infrastructures**.
  The strategies for data distribution and task allocation are important to ensure the Dispatcher layer has a satisfactory performance in Big Data executions within a *hybrid infrastructure*. These strategies will make it possible to identify some deployment boundaries.

1.5 Thesis Structure

After this first Chapter which contains a brief introduction and contextualization of this Thesis, the work is structured as follows:

Chapter 2 shows the concepts and the related work that are associated with this study. The works are set out in Section 2.1 which shows the concepts and assumptions adopted. In specific terms, the related work is structured in accordance with the following: Section 2.2 sets out the main approaches adopted to solve problems in heterogeneous environments with regard to Big Data. Section 2.3 describes some simulators for Big Data systems and explains the need to make a new simulator for hybrid systems. Section 2.4 shows the several frameworks that were established for multiple Clouds and that formed the basis of *MapReduce*. Section 2.5 shows the related work with the hybrid systems applied to Big Data. Section 2.6 summarizes the open-ended problems. Finally, Section 2.7 concludes with some final considerations.

Chapter 3 analyzes the main purpose of the hybrid infrastructure and establishes a platform to support this implementation. This Chapter reflects a collaborative network. Section 3.1 includes a brief introduction to this environment with a characterization for the hybrid infrastructures. In addition, shows the desirable features of a hybrid infrastructure for a Big Data environment. Section 3.2 examines SMART which is a hybrid platform

for Big Data. It introduces a use case for the implementation of SMART-Sent: a hybrid platform for Big Data that is integrated into IoT. Section 3.3 concludes with some final considerations about the hybrid infrastructure.

Chapter 4 introduces the BIGHybrid simulator that has been designed to simulation and evaluation of hybrid infrastructures. Section 4.1 describes the BIGHybrid Simulator and its operational strategies. Section 4.2 explains the main features of the BIGhybrid simulator. In addition, describes the volatility and mechanism for communication employed to reproduce the volatile environment in a Desktop Grid. Section 4.3 provides a detailed evaluation of the BIGhybrid Simulator including a simulation of *MapReduce* execution and its volatile behavior, with the operation of a failure tolerance mechanism. This evaluation makes it possible to validate the behavior pattern using the Grid5000 environment in real-world experiments. Section 4.4 includes a statistical evaluation and a study of the reproducibility of real-world experiments. Section 4.5 offers some final considerations about the BIGHybrid simulator.

Chapter 5 assesses the use of Hybrid infrastructures like Desktop Grid and Cloud to Big Data Analytics. Section 5.1 discuss about the strategies employed in the evaluation of hybrid infrastructures. Section 5.2 introduces the methodology applied to the evaluation and includes some use case studies. Section 5.3 includes an examination of the experiments and in Section 5.4 there is the analysis of the conclusions.

Finally, in Chapter 6 there are general conclusion, final considerations and future works.

2 BACKGROUND

The aim of this Chapter is to examine the perspective of concepts and infrastructure initiatives for Big Data implementations in homogeneous and heterogeneous environments. Some of these initiatives lead to hybrid infrastructures in private and public Cloud. However, it is very unusual to find an evaluation of hybrid environments which include a Desktop Grid platform, in the context to Big Data and particularly that of Geo-distributed environments. Thus, the strategies adopted for different Big Data platforms are summarized and evaluated with regard to related work that is closer to that of the hybrid model being considered. In addition, a comparison between the Big Data simulators is carried out to demonstrate the need to make more complex tools for Big Data analytics in hybrid environments.

2.1 General Concepts and Assumptions

This section outlines the main concepts of the MapReduce framework and other systems that have been used to create a Big Data ecosystem in hybrid infrastructures. The related work demonstrates the attempts made by the scientific community to find a solution for data-intensive computing in different platforms.

The Big Data implementation systems are grouped in accordance with their workload execution and include categories such as: batch, micro-batch, interactive, real-time, and near real-time (WU; BUYYA; RAMAMOHANARAO, 2016). In the workload with batch mode, the system knows both the data size and time execution. Thus, only an increase in computational capacity is required to obtain a better performance. The interactive mode explores datasets in a repeated manner while loading data of interest into memory over multiple parallel operations across the machines. However, the current Cloud environments have shortcomings when this interactive process is carried out. This means that new techniques should be developed to improve interactivity (ASSUNCAO et al., 2015). Some environments require the processing of data streaming in real-time to make complex decisions. Data pipes are treated at the moment of their arrival through real-time workloads.

2.1.1 Hybrid infrastructures

The concept of a hybrid infrastructure is defined in (TOOSI; CALHEIROS; BUYYA, 2014) as being a partnership between private and public cloud providers, thus enabling *cloud burst*. *Cloud burst* systems improve the executions in a private cloud when the computational capacity is increased and these systems enable on-demand task migrations from a private cloud to a public cloud. In this context, the environment maintains a mini-

mal SLA, as defined earlier. In the work of (SHARMA; WOOD; DAS, 2013) the authors consider a hybrid infrastructure to be an environment consisting of both native (*i.e.* only the use of physical machines) and virtualized clusters to exploit the benefits of both environments in Big Data applications.

The concept of a hybrid infrastructure in this Thesis can be defined as a mix of public or private Cloud with Desktop Grid, similar to the environment explored by (TANG; HE; FEDAK, 2015). The environment is based on a real-world experiment of two types of middleware from distinct infrastructures inspired by the work of (ANTONIU et al., 2013). Desktop Grids (ANDERSON, 2004) have been successfully employed in a wide range of projects, because they are able to take advantage of a large number of resources provided free of charge by volunteers. The resources are in fact heterogeneous in hybrid infrastructures because several cloud providers have an SLA guarantee that represents an average performance for hardware definitions which differ of those available to lease. The resources represent the most homogeneous environment possible, although, in fact, the resources have a heterogeneous behavior because they share virtual machines with other users at the same time. The DG environment is heterogeneous and consists of both stable and unstable nodes. A node turns unstable when a user requests the computational resources to the execution of their applications.

The unstable nodes can use a storage cloud, that is used to avoid data loss when a node shutdown occurs. The data is distributed geographically in different sites. The sites perform the same function for each processing to the workers. However, in Big Data analytic applications, it is necessary to combine each of the results from different sites, to a single *Reduce* function so as to achieve a final result.

The concept of *Multiple Cloud* is related to the use of distinct Cloud providers (both private and public), which also we adopt in this work. It should be noted that there is a fine grain in a private cloud with regard to the specifications for the hardware available, which is not present in public clouds. However, there is no guarantee that the resources will not be shared with other distinct users and applications.

2.1.2 MapReduce

MapReduce is a programming framework that abstracts the complexity of parallel applications. It is a batch processing system that partitions and scatters datasets across hundreds or thousands of machines, bringing the computation and data as close to each other as possible (WHITE, 2012). Figure 2.1, adapted from (WHITE, 2012), shows the *MapReduce* data flow. The *Map* and *Reduce* phases are handled by the programmer, whereas the *Shuffle* phase is created while the task is being carried out. The input data is split into smaller pieces called chunks, that normally have a size of 64 MB. The data is serialized and distributed across machines that form the Distributed File System (DFS).

When running an application, the master assigns tasks to workers and monitors the progress of each task. The machine that is assigned a *Map* task, executes a *Map* function and emits *key/value* pairs as intermediate results that are temporarily stored in the workers' disks, like *Map* function in 2.1.1. The execution model creates a computational *barrier*, which allows the tasks to be synchronized between the producers and consumers. A *Reduce* task does not start its processing until all the *Map* tasks have been completed. A hash function is applied to the intermediate data to determine which key will carry out a *Reduce* task. The group of selected keys forms a partition. Each partition is transferred to a single machine during the *Shuffle* phase, to execute the next phase. The shuffle is an overlapping stage with the *Map* phase in the first wave and non-overlapping stage in the

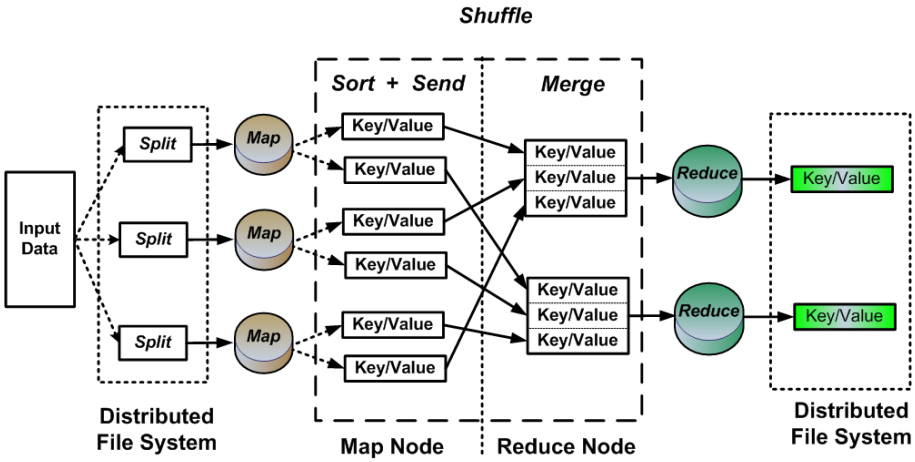


Figure 2.1: Model of the MapReduce data flowchart

last wave. Multiple waves increase the utilization of the disk I/O and reduce the performance (KHAN et al., 2016). After a *Reduce* function has been applied to the data, a new resulting *key/value* pair is issued, such as, *Reduce* function in Equation 2.1.1. Following this, the results are stored in the distributed file system and made available to the users.

$$\begin{aligned} map &: (k_1, v_1) \longrightarrow list(k_2, v_2) \\ reduce &: (k_2, list(v_2)) \longrightarrow list(k_3, v_3) \end{aligned} \tag{2.1.1}$$

MapReduce uses management systems for data replication and execution control. In addition, it has a management architecture based on the master/worker model, while a slave-to-slave data exchange requires a P2P model (WHITE, 2012). The worker is a node that can run a *Map* or *Reduce* functions in the *MapReduce* environment. Owing this, the scheduler launches a new task toward another node called speculative task during the last execution wave. A machine is characterized as a *straggler* when its task in progress has an above-average execution for the cluster. If a machine is characterized as a *straggler* after the first task distribution, it will not be assigned new tasks in its free slots.

2.1.3 BlobSeer-Hadoop

BlobSeer is a DFS that manages a huge amount of data in a flat sequence of bytes called BLOBs (**B**inary **L**arge **O**bject**s**). The data structure format allows a fine-grained access control. The existing storage file system has limited throughput under heavy access concurrency, for instance, the Hadoop file system (HDFS) does not support concurrent writes for the same file, and the data cannot be overwritten or appended. An unbalance workload is checked in the HDFS, when it receives new data from the incremental updates (NICOLAE et al., 2010). BlobSeer maintains the most recent version of a particular file in a Distributed Hash Table (DHT) to allow efficient concurrent access to metadata, which enables the incremental updating of database files, and a high throughput with concurrent reading, writing and updating of the data (ANTONIU et al., 2013). This is the main reason for using another file system like BlobSeer.

This data structure is completely transparent for the Hadoop users. The fault-tolerance mechanism is a simple data replication across the machines and enables the user to specify the replication level needed. The classical execution of *MapReduce* on Hadoop was not changed and explores data locality in a similar way to HDFS. In view of this, the BlobSeer

was the best choice to implement the features of the incremental update, without having to develop a new MapReduce framework for the Cloud implementation. The incremental update is also necessary for data management in a hybrid infrastructure.

2.1.4 BitDew-MapReduce

BitDew was developed to enable data management in large-scale, dynamic, heterogeneous,volatile and highly distributed Grids (FEDAK; HE; CAPPELLO, 2009). It is a middleware that exploits protocols like P2P, http, BitTorrent and ftp. The architecture is decentralized and has independent services. These services control the behavior of the data system, such as replication, fault-tolerance, data placement, incremental update, lifetime, protocols and event-driven programming facilities. BitDew-MapReduce (BitDew-MR) (LU et al., 2012) is a MapReduce implementation adapted to a volatile environment, that has already been combined with Cloud like a hybrid infrastructure (DELAMARE et al., 2012) to improve performance and reduce costs through the bag-of-tasks application.

The *Data Catalog* maintains a centralized and updated meta-data list for the whole system. The BitDew-MR model includes both stable and volatile storage. Stable storage is provided by stable machines or Cloud Storage like Dropbox and Google Drive, and volatile storage consists of local disks of volatile nodes. The *MapReduce* implementation is an API that controls the master and worker daemon programs. This *MapReduce* API can handle the *Map* and *Reduce* functions through BitDew services.

Result checking is controlled through a majority voting mechanism (MOCA; SILAGHI; FEDAK, 2011). In the Hadoop implementation when the network experiences unavailability, a heartbeat mechanism signals to the master that the host is dead. However, in BitDew the network can be temporarily offline without experiencing any failure. The fault-tolerance system needs a synchronization schema, as pointed out by (TANG; FEDAK, 2012), where transient and permanent failures can be handled. A barrier-free computation is implemented to mitigate the host churn behavior (LU et al., 2012). The computation of *Reduce* nodes starts as soon as the intermediate results are available.

These properties of BitDew-MR such as data placement, incremental update and fault-tolerance mechanism, are important for implementing a hybrid infrastructure. In addition, the computing power offered by the DG infrastructure is also of value to provide new infrastructures, starting from the allocation of free resources.

2.2 Big Data in a Heterogeneous Environment

The *MapReduce* model was originally conceived for large homogeneous cluster environments. As a result, some simplifications were adopted by the model with the aim of optimizing the task distribution. However, these simplifications may entail system degradation in heterogeneous environments. The work of (ZAHARIA et al., 2008) was the first study to detect these problems. Their study points out that there are some concerns over the simplification of the *MapReduce* model since this may lead to an excessive number of speculative tasks. These issues were observed among the executions of different applications in large *clusters* with virtual environments, *e.g.* Amazon *EC2* (AMAZON, 2016), owing to the competition for applications that provide access to hardware. To overcome this problem, the authors proposed LATE (*Longest Approximate Time to End*), a new task scheduler. Although LATE does not completely avoid speculative tasks, it considerably reduces performance degradation in heterogeneous environments. The results of an ex-

perimental evaluation results show that, compared with the native Hadoop scheduler in speculative mode, LATE achieves a gain ranging from 8.5% to 58%, depending on the application and number of working machines.

Some works focus on adapting *MapReduce* algorithms to volatile and heterogeneous environments. One of the first proposals was the MOON project (***M**apReduce **O**n **O**pportunistic e**N**vironments*) (LIN et al., 2010), a hybrid model for voluntary computing which considers that the system consists of volatile and reliable non-volatile machines. This approach was driven by the need to avoid the cost of data movement to and from the clouds across the wide area networks (WAN). The authors argue that to maintain one machine with an unavailability rate of 40%, eleven replicas are needed ($1 - 0.4^{11}$) to achieve an availability rate of 99.99% for a single data block in HDFS.

The authors had the aim of investigating the existence of a hybrid design which would be able to provide a high degree of data availability in volatile environments. Their solution involves applying the LATE algorithm. Loss of data from the volatility machines is overcome by replicating data in reliable machines. The data management of Hadoop has been extended to the "dedicated" Data Node and the volatile Data Node. The approach avoids putting replicas of chunks in volatile nodes. However, the number of reliable machines needed to replicate the data restrains scalability. In addition, a write request of a file might be declined if all the dedicated Data Nodes are close to saturation. The approach does not adapt the scheduling to the heterogeneous nature of the machines.

Other systems like the BOINC (ANDERSON, 2004), XtremWeb (FEDAK et al., 2001) and BitDew (FEDAK; HE; CAPPELLO, 2008) systems are successful implementations of DG environments. BOINC and XtremWeb have a centralized infrastructure for scheduling and management; in contrast, BitDew is an evolution of a distributed infrastructure designed for data management that supports well incremented updates and fault-tolerance mechanisms. Previous work (ANJOS et al., 2010) has shown that data redistribution based on the processing capacity of machines is suitable for the distribution of workload tasks in these environments.

Heterogeneous-Aware Tiered Storage (HATS) aims to improve I/O performance in Hadoop MR implementations (KRISH; ANWAR; BUTT, 2014). HATS performs data placement in accordance with I/O throughput and device capacity. Each different device is a HDFS instance in a *DataNode*. A *DataNode* with storage technologies that are different from usual, has a different data size, depending on its performance features. The data placement concept creates policies that take account of *network proximity*, *tier-awareness* and *hybrid* approaches. *Network proximity* involves retrieving replicas from the nearest rack to reduce network traffic. The *tier-aware* policy ensures that a node can store a single replica even if the node has multiple HDFS instances and retrieves data from the fastest available tier.

MRA++ (***M**ap**R**educe* with **A**dapted **A**lgorithms for **H**eterogeneous **E**nvironments) (ANJOS et al., 2015) introduced adapted *MapReduce* algorithms to heterogeneous environments with the aim of addressing the main problems originating from the simplification of the *MapReduce* model. Thus, the developed algorithms allow the use of data-intensive applications in large-scale environments with the use of the Internet.

The strategy adopted in this work is to examine three areas of the *MapReduce* implementation: grouping, data distribution and task scheduling. In *MapReduce*, the difference in computational power between machines in a heterogeneous environment causes an unbalanced load. The data is distributed in accordance with the heterogeneity of the machines to prevent a large increase of execution time. The machines are grouped according

to their computational capabilities; then they receive both data and tasks based on their own group. It does not need a history of executions as it creates a knowledge base of execution times during the setup phase before the *job* is executed. This allows scheduler configurations before the data splitting phase. The *MRA++* scheduler also avoids sending data to machines with low processing capability, which could be later characterized by the system as *stragglers*.

2.3 Simulators for Big Data Systems

GroudSim, a Grid and Cloud simulation toolkit for scientific applications, was introduced by (OSTERMANN et al., 2010). This simulator is based on a scalable simulation-independent discrete-event and is used for scientific applications. Here it was employed in an attempt to simulate two complex environments, like Grid and Cloud. *GroudSim* provides support for traces used for capturing both hosts and event traces.

Stochastic distributions make it possible to run deterministic and non-deterministic simulations. A failure rate model follows a stochastic distribution of failure properties like the size of the failure, the duration of the failure and the *Mean Time To Failure* (MTTF) for jobs and file transfers. However, it should be noted that the simulation architecture is composed of a single thread. The infrastructures are only very simple synthetic entities and, for this reason, it is difficult to capture discrete executions. Unlike GroudSim, BIGhybrid (which is characterized in Section 4.1) allows possible complex simulations including volatile environments.

CloudSim (CALHEIROS et al., 2011) is an extension of GridSim (SULISTIO et al., 2008) for Cloud simulation. The simulator supports the modeling of large-scale Cloud computing environments, including data centers, at a single physical computing node. This means that Clouds, service brokers, provisioning, and allocation policies can be modeled. The main features enable the creation and management of multiple, independent, and virtualized services in a data center.

The simulation is based on Java and has a dedicated management interface for VMs, memory, storage and bandwidth. A host can support multiple sets of VMs to simulate applications based on Software-as-a-Service providers. The authors assume that provisioned virtual machines are predictable and stable in their performance. There is an I/O contention that has been checked in read/write storage devices and has an impact on the performance (BUX; LESER, 2015). Although CloudSim can simulate Federated Cloud with a *Cloud Coordinator*, the simulator is not compatible with data-intensive applications (CALHEIROS et al., 2011) as in the model of the MapReduce framework.

CSPs, which has different geographical locations in the Internet, have to coordinate their load distribution across data centers. The study of (BUYYA; RANJAN; CALHEIROS, 2010) introduces the InterCloud simulator as a possible architecture that extends CloudSim to the Cloud Federation infrastructures. The main problem is that the service providers expect the users to choose the service that is nearest to their physical location. Otherwise, the clients have difficulty in determining the best location for hosting their services in advance, since a CSP may not know the origin of the consumers of their services. As a result, the CSP may not be able to provide the quality of service contracted in the locality if the customers originate from multiple geographical locations.

The InterCloud software architecture has a coordinator and brokers to locate resources for clients. The functions of the coordinator include scheduling, resource allocation, dynamic monitoring and application composition, although this architecture does not take

account of security mechanisms or a minimal SLA (KOHNE et al., 2014). In addition, the broker is not prepared for data-intensive management.

Kohne *et.al.* introduce a simulation of Cloud Federation (KOHNE et al., 2014) to reduce the complexity of the experiments called FederatedCloudSim. CSPs can use the resources of other CSPs with the aim of improving resource optimization while respecting the SLAs. The migration services must be executed automatically and a Service Level Agreement - SLA - has to be negotiated in advance. The purpose of this is to study standard interfaces to exchange services and establish an orchestration framework that creates and monitors distributed services based on SLAs. FederatedCloudSim is implemented with the CloudSim discussed here.

The scheduling process has several levels and invokes the brokers. It may be dedicated or employ a pass-through model. In the dedicated model, the tasks are executed by a broker locally, and in the pass-through model the tasks are passed on to a remote CSP member of the Cloud Federation. A special case is a virtual CSP that is outside the federation and can accept jobs from customers. Otherwise, the services will be "best-effort"and are described as Service Level Objectives - (SLO) in the SLA. Again, this implementation is not designed for data-intensive management.

The AweSim simulator is defined in (TANG et al., 2014) and based on a network simulation framework that involves a fine-grained simulation for workflow computation and data movement across multiple Clouds. The proposal attempts to overcome the problems of provisioning and allocating resources for multiple Cloud scientific workflows that require task placement and data movement between distributed multi-domain computing sites. AweSim is a client/server architecture. The implementation uses workload traces from a production data analysis service and is thus similar to the BIGhybrid simulator that adopts its behavior from traces in a real-world volatile environment.

The data-intensive approach avoids unnecessary data movements in the Workflow simulator. A ratio is calculated for the most expensive computational task (E_c), as $E_c = T_{run}/S_{in}$, where T_{run} is the runtime and S_{in} is the input data size. A historical "job" determines the average E_c that defines the most data-intensive task. The scheduling considers the distance between the server and computing resources. The CSP may have a different data size to adjust its distribution and explore the design of large-scale storage, network architecture and distributed data. The authors assume that the computing resources are homogeneous except for the network bandwidths for the data server; otherwise, the environment is different from the hybrid infrastructures where the workloads and resources are heterogeneous.

DynamicCloudSim is an extension of the popular simulation CloudSim toolkit that is used in the study of (BUX; LESER, 2015). The goal is to model the instability inherent in computational Clouds and similar distributed infrastructures. This instability is demonstrated in the study of (SCHAD; DITTRICH; QUIANÉ-RUIZ, 2010), where considerable performance variations were found; these fell into two bands, depending on the selected processor type. The simulator allocates resources to the VMs in terms of compute units, similar to Amazon EC2. Furthermore, in contrast with CloudSim, DynamicCloudSim, it does not assign new VMs to the host with the most available resources, but to a random machine within the data center. The heterogeneity is simulated through this random choice and represents permanent variance in the performance of VMs caused by differences in hardware. The stragglers (nodes with poor performance) are simulated through coefficient parameters of performance.

Table 2.1 summarizes the main simulators and features used to simulate environments.

The simulators can simulate Clouds and Federated Clouds although the Big Data support is limited to the AweSim simulator, but without providing the simulation of failures or trace support. A trace support is especially important to reproduce real behavior from the real-world environment. The hybrid infrastructures are an opportunity for new studies that still need to be carried out. This shows that there is an open space for the development of hybrid systems simulators. This simulation is important for determining constraints in complex environments, such as Cloud and DG. The BIGhybrid simulator, described in the next Chapters, explores these open issues, in especial, the fact of do not have a simulator to hybrid environments yet.

Table 2.1: Related work simulators

	Simulated Features							
Simulators	Grid	Cloud	Federated Cloud	Hybrid	Big Data Support	Failure Support	Trace Support	SLA Support
GroudSim	Yes	Yes	No	No	No	Yes	Yes	No
CloudSim	No	Yes	Yes	No	No	No	No	No
InterCloud	No	Yes	Yes	No	No	No	No	No
Federated CloudSim	No	Yes	Yes	No	No	No	No	Yes
AweSim	No	Yes	Yes	No	Yes	No	No	No
Dynamic CloudSim	No	Yes	Yes	No	No	Yes	No	No

2.4 Multiple Clouds and Frameworks

Organizations are increasingly relying on an infrastructure from multiple providers as a means of increasing fault tolerance and avoiding provider lock-in. When considering multiple Clouds (hereafter also described as *Multi-Cloud*), application deployment becomes complex. A *Multi-Cloud* infrastructure contains various configuration choices and can change its requirements and workloads dynamically at the time of execution. In view of this, solutions are needed for the automatic configuration of complex cloud services at different abstraction levels. In this context, multiple cloud infrastructures, like clouds in heterogeneous environments, need different configuration levels, such as the operational system, service containers and configuration capabilities (LE et al., 2014).

The allocation of resources from CSPs to users is carried out in terms of the execution time, number of virtual machines, data transfer and size of data storage. The users must map their computational resource needs before running their applications. This means that, if there is a lack of knowledge on the part of users about the CSP infrastructure or a real need for resource allocations, it can lead to an incorrect lease of CSP resources for the users and a higher cost than expected. However, an optimal allocation is difficult to achieve, and so strategies to obtain an approximation can be accepted (MASHAYEKHY; NEJAD; GROSU, 2014).

In the study of (MANSOURI; TOOSI; BUYYA, 2013) a brokering algorithm was employed for optimizing the storage availability and finding a placement of objects that was suitable for the required Quality of Service (QoS). The algorithm takes account of the cost of maintaining one object in a cloud provider, reduces the probability of failure and improves the associated QoS with each service-level agreement (SLA) contracted with a cloud provider. An object is a target data, without a particular size or defined type. The data is split into *chunks* and the main goal is to find the optimal *chunk* placement

depending on the user's needs and financial means.

A large number of transfers of objects from one cloud storage provider to another, takes up time and is often impossible during the execution time. An expected availability represents *M* objects in each data center, and this determines the expected failure of the object in each data center. The study evaluates two parameters to each cloud provider, the failure probability and the cost per object. The objects are replicated in multiple sites in accordance with these metrics. However, the proposed solution does not identify the network overhead generated by a large number of data transfers. The total size can achieve up to several exabytes, which can require a lot of time for these transfers.

SALSA (LE et al., 2014) is a framework for the orchestrated configuration of cloud services through multiple CSPs. This framework provides a model for application configurations and the deployment of different kinds of services. The information about the configuration supports each level of cloud service such as application levels, deployment relationships at multiples software stacks and the link between service units and configuration capabilities. The configuration capabilities are obtained from registered services (cloud services and specifications of topology services) or user specifications. SALSA has a *service unit orchestrator* for multiple configuration services for each configuration task group. Its purpose is to control the application deployments, movement of virtual instances among different cloud providers and the deployment of an environment like VM, library loads and support for multiple stack deployments of cloud.

The creation of VM is a separate process from other software levels. The configuration capabilities can be obtained via a registry service or from user specifications, to determine the relationship between the service units. A service orchestrator is generated for each service unit enable it to handle the tasks. *Meta Information* contains abstract nodes with generic types of service that implement the virtual nodes. SALSA adopts an approach where each *service unit orchestrator* runs independently and interacts with a cloud service orchestrator. Although the framework enables heterogeneous configurations, there is not a mechanism to evaluate the performance or the workloads used to adapt the load-balance in Cloud. The SALSA architecture, designed by (LE et al., 2014), is shown in Figure 2.2.

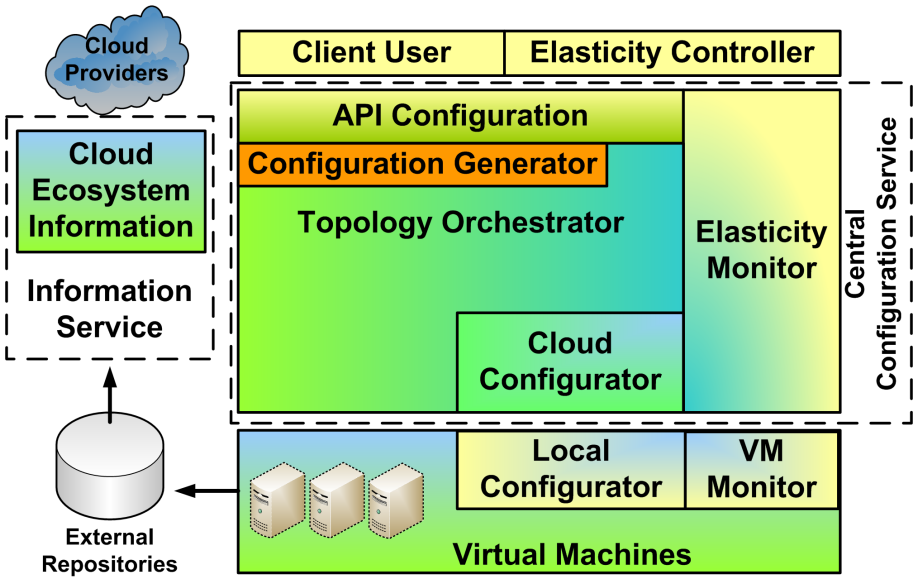


Figure 2.2: The Salsa architecture

The framework has a central configuration service that orchestrates the setup oper-

ation through the local configuration layer in the VM. The information service keeps a good deal of information about the Cloud infrastructure which is handled by a configuration generator. The topology orchestration layer creates a dependency graph and sets a configuration plan for a Cloud configuration system and the VMs are managed in this way. The monitor layer keeps the status of both VMs and the Cloud elasticity but it is necessary for the services to be already working before new service instances can be distributed.

HyMR (LORETI; CIAMPOLINI, 2015) is a framework for enabling an autonomic *cloud burst* for clusters of virtual machines that execute MapReduce jobs over *Multi-Cloud*. The authors implemented a Hybrid Infrastructure as a Service (HyIaaS) for the VM instance (partitions management) in *Multi-Cloud*. HyIaaS implements an OpenStack[1] extension. This partitioning is transparent to the users, since it allows them to have access to all the VMs in the same way, regardless of their physical allocation. HyIaaS receives the deadline specifications of the users that are stored in a user-policy for managing VM migrations. An external CSP will be responsible for receiving and launching the VMs across their *Cloud Controller* module.

A *Logical Node* monitors and analyses critical events from a physical machine. A *Logical Cloud* makes spawning/migration decisions based on *Logical Node* information. Figure 2.3 shows the HyMR architecture, where a HyIaaS orchestrates the application executions. The HyMR runs on the *Cloud Controller* and maintains data consistency in a part of HDFS. However, VM migrations have a poor performance when carrying out data copying operations from the HDFS.

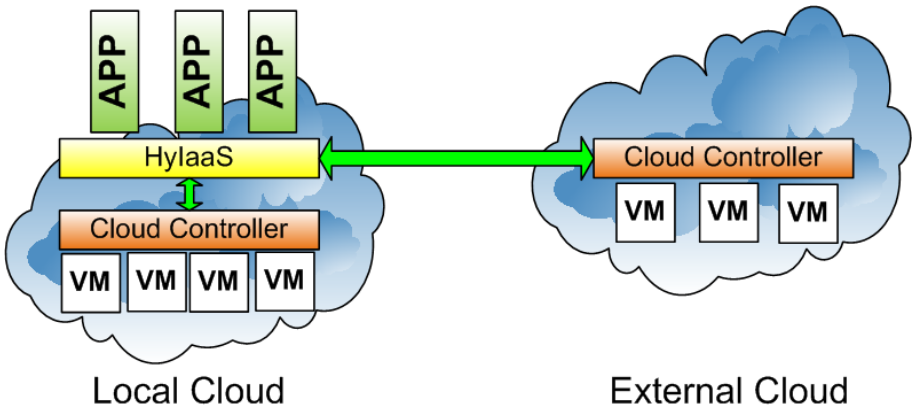


Figure 2.3: The HyMR architecture

2.4.1 MapReduce across Multiple Clouds

Twister (EKANAYAKE et al., 2010) supports direct intermediate data communication without saving the intermediate data in a local disk. This feature enables iterative *MapReduce* computation with the use of a Publish/Subscribe middleware such as a *Broker System*. The Twister computation follows a *MapReduce-Combine* model, where the *Combine* step is similar to *Merge* in *MapReduce* implementation. The *Merge* executes after the *Reduce* phase and the last phase is combined in a single result. Twister starts a daemon for each worker, to link a worker to a broker network for the purpose of receiving data and carrying out tasks. A driver provides an API to convert calls from Twister's API to commands and messages for the workers.

[1] https://www.openstack.org/

The Twister architecture has three functions: *Twister Client Driver* to guide the workers computation, *Twister Daemon* that manages the cache memory for each worker and *Broker network*. A master node maintains a broker server to allow intermediate data transference between workers. A *partition file* keeps the meta-data with the location of intermediate files for all the workers. The effectiveness of the cache memory is only achieved with a stable node, *i.e.*, without worker failures. For this reason, Twister implements a *checkpoint* of the computation state between interactions to enable a roll-back of few interaction failures but the architecture does not use a Cloud infrastructure.

Twister4Azure is a distributed iterative *MapReduce* runtime for cloud developed with the *Azure Cloud Infrastructure* service, without impairing the fault-tolerance, scalability, *MapReduce* APIs and the data model (GUNARATHNE et al., 2013). The Microsoft Azure platform implements a *Microsoft Message Queue* service - but without any guarantee of order or making all messages available for a single requests - and a Storage Blob service - *Page blobs* that are optimized for random read/write and *block Blobs* that are optimized for streaming. These services are based on the concept of an *affinity group* to optimize communication between accounts and services. The message Queue service has a scheduler function for *Map* and *Reduce* tasks (GUNARATHNE et al., 2013).

A *MRRoles4Azure* is a distributed *MapReduce* runtime that encapsulates *Map* and *Reduce* tasks within a message queue service to provide load balancing. The main role of *MRRoles4Azure* is to carry out the scheduling and monitoring of the computation. Re-execution and duplicate execution manage the task failure and slower execution respectively. The architecture is decentralized to avoid a single failure point. The *Merge* task introduced in (GUNARATHNE et al., 2013) is an extension of *MapReduce* for iterative computation. The broadcast operation spreads the output data of the *Merge* task to the next iteration. The data-flow is *Map* → *Combine* → *Shuffle* → *Sort* → *Reduce* → *Merge* → *Broadcast*. A *data cache* schema supports three level of data caching: (1) *instance storage* which stores all the files in a local disk, since the input data blobs do not change during the course of the computation; (2) *direct caching* in-memory ; and (3) *memory-mapped* file. The memory caching uses the *least recently used* (LRU) cache algorithm. It maintains a single instance of each data cache per worker-role and this enables a better use of the cache.

The operations of the layer, called *Map-Collective architecture*, replace multiple stages of the iterative *MapReduce* computation in *all-to-all* communication and the main API is the *MapReduce-MergeBroadcast* (MR-MB) (GUNARATHNE; QIU; GANNON, 2014). MR-MB provides *dynamic data*, including data broadcast for a list of key-value pairs such as the following:

```
Map(<key>, <value>, list_of<key,value>dynamicData)
Reduce(<key>,list_of<value>, list_of<key,value>dynamicData)
```

Merge improves *MapReduce* programming and provides support for iterative programming, which can be used for summarizing or aggregating the results of a single *MapReduce* iteration. The broadcast operations send data from Merge to the next iteration task. The MR-MB model implements the concept of *Map-Collective* with two communication primitives (*Map-AllGather* and *Map-AllReduce*). This concept is inspired by MPI 3.0 with collective communication primitives.

Map-Collective enables the applications to send data, as soon as *Map* has yielded its result. This characteristic helps to mitigate the effects of heterogeneous tasks by ensuring that there are no barriers in the multiple processing stages (this is also known as "*barrier-free*"). *Map-Collective* can support fault-tolerance by relaunching the iterations in case

of failures and only maintains the checkpoint obtained from the iteration results when the iterations are relatively fine-grained.

Map-AllGather broadcasts the *Map* task outputs to all the nodes. Each node transmits its results through a recipient, which *Map-AllGater* uses to send data to all the workers, once the *Map* task has been completed. This abstraction is an iterative *MapReduce* computation that eliminates the *reduce*, *merge* and broadcasting stages from the original *MapReduce* framework. *Map-AllReduce* is a collective which combines a set of values issued by all the workers and makes an aggregation of the results from the *Map* tasks. This model replaces the *Shuffle*, *Sort*, *Reduce*, *Merge* and *Broadcast* stages from MR-MB. *Map-AllReduce* implements a hierarchical method for reducing the number and size of the intermediate data. It is similar to the *Combine* function in the Hadoop implementation that leads to a reduction of intermediate keys in the *Map* phase.

These approaches show the need for a more fine-grained system of task management and data distribution across Big Data applications. Businesses and governments arrange their data in distributed cloud platforms for different reasons, such as, the need to maintain the proximity of resources; data storage with organizations that share common goals; and a desire to keep data replicas across regions for redundancy purposes. This data information must be analyzed on a global scale.

One possible way to do this is to merge all the data in a single data center, and another is to use a *Multiple Clouds* infrastructure to execute individual instances of *MapReduce* across each dataset separately and then aggregate the results. The study of (JAYALATH; STEPHEN; EUGSTER, 2014) suggests that this could be done by running jobs in a geo-distributed operation. The authors introduce the G-MR, a Hadoop implementation based on a geo-distributed dataset across multiple data centers. They state that, for instance, it is possible to have multiple execution paths for carrying out a *MapReduce* job in this scenario, although the performance may vary considerably for each path. Another problem is that popular *MapReduce* open sources, like Hadoop, do not support this feature. In addition, most CSPs do not usually provide a bandwidth guarantee for large-scale data transfers in execution time (ZHENG et al., 2014).

The G-MR has an algorithm called a *Data Transformation Graph* (DTG) which determines an execution path for performing a job sequence for *MapReduce*. The problem is how to decide which stage should derive partitions that must be moved and how to reduce costs by finding the best performance for *MapReduce* applications. Figure 2.4 shows the architecture from G-MR, adapted from (JAYALATH; STEPHEN; EUGSTER, 2014). The architecture consists of the following modules: a *Group Manager*, *Job Manager*, *Copy Manager* and *Aggregation*. The *Group Manager* optimizes the execution path and may instruct the *Job Manager* to copy data for a remote data center or aggregate multiple sub-datasets. The *Job Manager* performs the jobs over Hadoop which is deployed in each n data center. The *Copy Manager* is responsible for executing the data copy from one data center to another. However, the total number of nodes in a single job graph is $O(p^n)$ and can become huge when the number of p partitions grows. The *Aggregation* manager maintains the integrity of the results. The model shows that this architecture is feasible from the standpoint of data distribution and the integration of results.

The approach *Write Once Read Many* (WORM) is an accepted assumption for data access in *Big Data* applications like *MapReduce*. The handiest manner for *Big Data* processing across several data centers, is to use a data replication mechanism among different CSPs. However, the variability in the high-performance required for cloud operations leads to bottlenecks (IOSUP; YIGITBASI; EPEMA, 2011), (GROZEV; BUYYA, 2015).

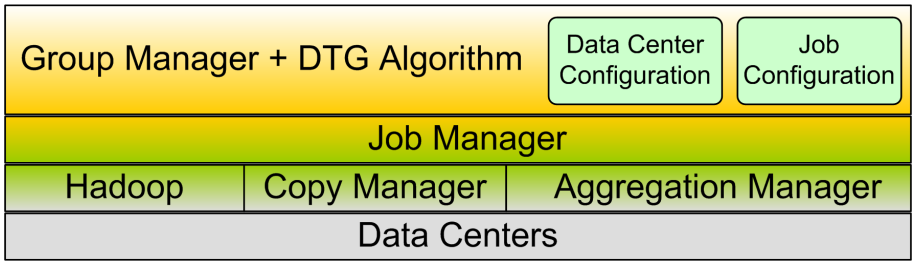


Figure 2.4: The G-MR architecture

Thus, the best strategy is to reduce data transfers.

The study of (TUDORAN et al., 2014) argues that there are two methods for modeling complex infrastructures. The *analytical models* use low-level details with workloads and are characterized by their ability to predict the performance. This means the wealth of detail is what will determine the best modeling. The *sampling method* is an active approach which does not require any previous knowledge of the infrastructure. The information about network bandwidth, topology and routing strategies is not available to the users in public clouds. Because of this, the authors introduce a sample-based category for modeling that monitors the environment with agents, called GEO-DMS. The agents carry out the monitoring for data transfers and geographically-distributed data management that is transferred across multiple clouds. The model registers the correlation between performance (execution time) and cost effectiveness (finance), and imposes budgetary constraints in the interests of safety.

The agents are implemented through VMs in each CSP where the applications are running. The *decision manager* is concerned with how the transfer paths are established between the source and destination. One way to achieve this is directly from the node to the data center or by using multiple paths across intermediate data centers. The data transfers are intra-site data replications that result from the presence of dedicated links among the data centers of the same CSP. The scientific applications interact with an API to provide data transfers over a WAN. A *monitor agent* carries out environmental monitoring and makes the measurements for the decision manager. The measurements include bandwidth throughput between data centers, and the CPU load, I/O speed and memory status of the VM nodes. The *decision manager* updates the weights of the paths periodically with the aid of these measurements.

The data management efficiency is measured through a *Transfer Time* as in Equation 5.1.5, where "ψ" is the throughput between the sites and "n" is the number of nodes. The amount of data to transfer (θ) is related to the gain (g) of parallel transfers, and is determined empirically as a value in the interval $[0,1]$.

$$T_t = \frac{\theta}{\psi} * \frac{1}{1+(n-1)*g} \tag{2.4.1}$$

The cost of a geographical transfer (C_{geo}) has three components, shown in Equation 2.4.2. The first is related to data output (ρ) from one CSP to another. The others correspond to lease costs of VMs and bandwidth. The parameter (η) represents the number of VMs. Both the lease cost from VM and bandwidth for each data transfer will define the cost of a geographical transfer. The machine performance (Vm_{cpu}) has an associated intrusiveness bias that reflects the influence of one or other processor use (I_{cpu}). The

bandwidth intrusiveness (I_{Bw}) reflects the impact of different bandwidths. This intrusiveness bias depends on the allocation resources and reveals to what extent an infrastructure is busy in each CSP. The network bandwidth (Vm_{Bw}) represents the degree to which a channel is free, and related to the data transfers ($\vartheta = \frac{\theta}{n} * T_t$).

$$C_{geo} = \eta * (\underbrace{T_t * Vm_{cpu} * I_{cpu}}_{\text{VM lease cost}} + \underbrace{\frac{\vartheta}{Vm_{Bw}} * I_{Bw}}_{\text{Bandwidth cost}}) + \rho * \theta \tag{2.4.2}$$

In the study of (PALANISAMY; SINGH; LIU, 2015), the authors argue that the users tend to choose resources based on their workload peak for *MapReduce* jobs in Cloud, due to their expertise in the use of dedicated clusters. In contrast, the submitted jobs are short, and have an average execution time of around 30 s. Despite this, the cloud solutions can be improved by means of per-job and per-customer optimization, which leads to a low utilization of cloud resources. These involved framework for cost-effective resource management called CURA to automatically create the cluster configurations for the MapReduce job. The aim is to optimize resource allocation to reduce the infrastructure costs in the cloud data center.

The user submits a job and can either define its execution profile or create a job performance modeling to make predictions of the job execution. CURA draws on this information to make decisions about scheduling and carry out performance predictions of tasks based on the input data size, VM types, cluster size and parameters for the job. This model implements its global resource allocations engine and is only used once when the customer's first job begins. The VM allocations will be maintained for the next jobs with the same configurations, and after that the Hadoop's instance is destroyed. This means that, the initialization to the next execution is faster, although the data upload to HDFS is still necessary.

The cost of cluster configuration ($C^{k,n}$) to run a i_{th} job (J_i) in a specific instance type (k) of a number of VMs (n) is defined in Equation 2.4.3. Where the number of physical resources (R_k) of a VM has a determined number of physical resources (M) like CPU, memory and disk . Additionally, if an execute prediction is also evaluated, an error for the execution time (t_{run}) need will be considered. This error can cause deviations when predicting the execution time because there are multiple *MapReduce* jobs.

$$C(J_i, C^{k,n}) = t_{run}(J_i, C^{k,n}) * \frac{n * R_k}{M} \tag{2.4.3}$$

Thus, the end time of job ($t_{end}(J_i)$) is similar to Equation 2.4.4, where $X_i^{k,n}$ is a Boolean variable that indicates if the scheduling uses the cluster configuration and $\sum_{k,n} X_i^{k,n} = 1, \forall i$. The time to end must be completed until it is achieved in $t_{deadline}(J_i)$. The way to meet the scheduled deadline is to find a minimum cost for all the job executions, where $Overallcost = \min \sum_{k,n} X_i^{k,n} * Cost(J_i, C^{k,n})$.

$$t_{end}(J_i) = t_{start}(J_i) + \sum_{k,n} X_i^{k,n} * t_{run}(J_i, C^{k,n}) \tag{2.4.4}$$

The CURA scheduler must make future reservations for the VM pool resources to avoid high latency when launching a new job. This can be achieved with a job prioritization scheme which attempts to create a minimum of idle instances. The job is considered

a priority job, if its *overall cost* is the cost of a higher usage of resources. CURA can bring about a reconfiguration of the VM instances of a VM pool, with the aim of changing the instance type from large to small or in contrast, on the basis of the workload analysis (PALANISAMY; SINGH; LIU, 2015).

The approach used for the analytics of geo-distributed data is a centralized approach that consumes a significant amount of bandwidth, and leads to a poor performance. In addition, this approach must consider new constraints, such as concerns about privacy. The distributed execution is a strategy that pushes computations to local data centers and then aggregates the intermediate result to do further processing. Thus, this execution method can lead to low latency since it involves a distributed execution. However, in several approaches the constraints on data movement constraints are not taken into account (JI; LI, 2016).

2.5 Related Work closer in Hybrid Systems

Hybrid systems have been described in some scientific papers as a mixture of public and private clouds. At the same time, this factor refers to the degree of availability in the resource policies. Factors regarding the deployment of native and virtualized clusters are evaluated in the work of (SHARMA; WOOD; DAS, 2013). In their analysis, the authors argue that generic benchmarks show an overhead of 5% and 15% for computation and I/O workloads respectively, when confronted with a non-virtualized system. In addition, the level of overhead may vary depending on the workload, availability of resources and programming of interactive jobs. The reduction of performance in virtualization systems has led companies like Google and Facebook to use physical machines.

In a *Hybrid Cloud* environment, the data centers are interconnected by means of slow links. The data is moved from the private to public Cloud when a new VM allocation is necessary to improve a task performance. The data locality and data movement are a challenge for accelerating iterative MapReduce in *Hybrid Clouds*. Iterative applications reuse invariant input data. Furthermore, since the extra resources represent an additional cost for data movement a trade-off between performance gains and benefits must be evaluated.

These issues are evaluated in the work by (CLEMENTE-CASTELLÓ et al., 2015) to address iterative MapReduce problems in a Hybrid IaaS Cloud environment. The authors argue that improving the ability to take advantage of data locality in a hybrid Cloud environment is critical. The aim of the strategy is to extend the original fault-tolerance mechanism of HDFS and deploy data replicas from an on-premise VM in a private Cloud to another allocated off-premise VM in a public Cloud as if it were an external rack over the HDFS. The off-premise VM initializes without data and need re-balance the initial data blocks with on-premise VM. A heuristic determines a re-balance factor from an I/O intensive benchmark to approximate the application behavior for the duration of the re-balancing. The scheduler waits for the off-premise VM to get a minimal replica number to start the task distribution.

In the work of (TANG; HE; FEDAK, 2015) the authors implement a layer to create a hybrid distributed file system (*HybridDFS*) under reliable data storage clusters in Cloud and unreliable data storage in desktop grid. Each data chunk in volatile nodes has at least one replica distributed for different volunteer PCs or cluster nodes. The volatile node employs a fault-tolerance mechanism that will be specified in Subsection 4.2.1. This means, it is possible to check the feasibility of using hybrid infrastructures such as Cloud and DGs.

The authors developed a Node Priority-Based Fair Scheduler (NPBFS) algorithm. This means the node with greater computational capacity processes more tasks. The system takes into account both data location and storage capacity, and the length of the task queues is related to the computational capacity of each node. A weight coefficient indicates the node priority for receiving a greater number of tasks.

There is an increasing need for a repeated analysis from Big Data in the Cloud with streaming characteristics. Streaming and data-intensive applications are often not the best profile for Cloud applications (TUDORAN et al., 2014). Streaming systems are event-driven and their behavior differs from batch systems like *MapReduce* (ZAHARIA et al., 2012). The *MapReduce* model lacks efficient support for real-time processing. The traditional system that has been developed to process static databases like Hadoop cannot provide a low latency response in real-time or streaming processing. The first requirement for a heterogeneous cluster to achieve load balancing is information such as that about the infrastructure, topology, and performance of their individual nodes. The second is knowledge about computational factors, such as, incoming tasks and input data. These requirements are more difficult to achieve in streaming than with batch workloads because of the continuous data streams (RYCHLY; KODA; SMRZ, 2014).

Hadoop Streaming is a utility included in the Hadoop distribution in an attempt to enable streaming in the MapReduce model through two standard Unix interfaces for stream processing, one *stdin* (input) and one *stdout* (output) (DING et al., 2011). These interfaces have two "executables" (one mapper and one reducer) that use a *Combiner* function. The *Combiner* is an implementation that enables a map output to run in memory. *Pipe operations* are created by the Unix system call to build a half-duplex communication channel for an external executable file. A *pipe call* is invoked to establish a communication channel for each interface. When a new pipe is created, two file descriptors are obtained. One of them is for reading and the other for writing. An *inode* identifies each pipe in the local file system to temporarily store data. The communication channel may generate system overhead and some critical fields of pipe can produce race conditions between the read/write operations.

The strategies for implementing streaming in Cloud is discussed in the work of (TUDORAN et al., 2014). The authors evaluate applications like the Ocean Observatory Initiative, where sensors send information that is collected by satellites for geo-spatial computations. The study shows how communication in the Cloud can interfere with computation. The approach uses persistent and ephemeral storage. In the first, called *Stream&Compute* (SC), the data is sent directly to VM computation without storage persistence. In the second, called *Copy&Compute* (CC), the data is first saved in an attached storage, so that it can be conveyed to VM computation afterwards. The system processes data similar to the BoT application. When the sensors produce new data, it is processed against existing features as in a temporal process. This eliminates, the need for communications between processes, but adds a huge volume of data that must be streamed for each worker.

The *Stream&Compute* provide a better response time, but when carried out on a large scale, there is a network saturation that leads to a need for redistribution across different CSPs. The *Copy&Compute* makes it easier to do repairs, when an unexpected stop occurs to the VM. In contrast, the date is near to the computation. However, a high variability in the remote copy phase causes a variation of around 20% in terms of data transfers per seconds. Disc contention is observed in the *Copy&Compute* approach, when all the workers are trying to access the data at the same time. The classic problem of CPU

utilization vs. I/O has a significant impact on the data throughput owing to virtualization.

In the work of (VASILE et al., 2015) the authors propose a resource-aware hybrid scheduling algorithm for batch jobs and workflows. The scheduler assigns tasks for groups, in accordance with the resource distribution strategy of each cluster. The algorithm considers a hierarchy of clusters to evaluate the lease resources. Heterogeneous clusters consist of different public and private systems. The new workloads involve on-demand provisioning. The study extends the CloudSim simulator (CALHEIROS et al., 2011) to four scheduling strategies.

The authors have shown the HySARC^2 architecture with three modules: monitoring services, an analysis module, and a scheduling mechanism. The analysis module implements a delay which makes it necessary to wait for a node's start-time without increasing the makespan. A graph represents a weight for each path where the shortest possible time is chosen for each task execution. The task without dependency is executed in the clusters. A DAG is created for the tasks with dependencies according to their distances. The distances are evaluated with a K-means algorithm and a ratio level between computation and communication. The evaluation has a scale of 1000 tasks versus 1000 resources, and 10 VM instances, which is a low scalability for the Big Data environment.

A next generation data processing engine has been created that is based on *Lambda Architecture* (WU; BUYYA; RAMAMOHANARAO, 2016). This architecture makes it possible to build Big Data systems as layers that satisfy the properties of a subset such as: having an internal code optimization; executing iterative algorithms; being able to achieve immutability and re-computation within the Big Data core system; getting low latency readings and updates without impairing the robustness of the system; having high scalability; and others (MARZ; WARREN, 2015). The architecture has three layers: the batch layer, serving layer and speed layer. The speed layer is related to the stream application where the data analytics involve real-time processing. The batch layer represents applications with a defined size of datasets without real-time constraints. The serving layer is a pre-processing in memory of immutable datasets and provides a speedup to an on-the-fly query analysis of both batch and stream environments. The user queries are expected to gather information about the batch view and real-time view that is similar to that of the merge abstraction. Figure 2.5 shows this architecture, adapted from (MARZ; WARREN, 2015).

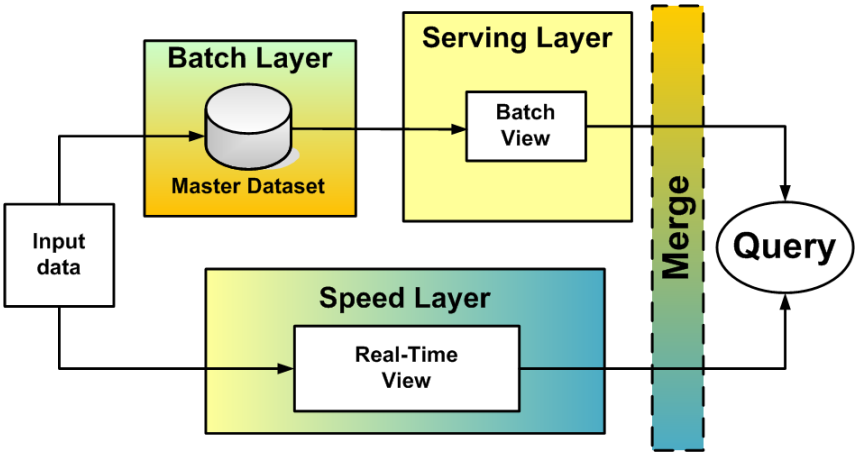


Figure 2.5: The Lambda architecture

The Apache Flink, previously called Stratosphere, is a data analytic framework that

follows the *Lambda Architecture* and enables the extraction, analysis and integration of heterogeneous datasets (ALEXANDROV et al., 2014). Flink has a flexible pipeline that enables several map-reduce and extended functions like Map, MapPartion, Reduce, Aggregate, Join and Iterative. It has two APIs, one for *DataSet* and other for *DataStream* respectively deployed on process batch and stream applications which constitute a hybrid programming environment. The core is a dataflow in a distributed streaming that does not store data but converts it into optimized binary formats, after their reading. It is extensible for traditional data warehousing queries such as textual data queries and information integration in a Table API library. The implementation supports iterative programs that allow an execution graph analysis and statistical applications inside the data processing engine. The architecture includes different deploying modes, such as local (in a single JVM), cluster (standalone and Hadoop-YARN environment) and Cloud computing (EC2 and GCE). Flink supports Java, Scala and Python programming languages (CHAUHAN et al., 2015).

The stack is built on top of each layer, and raises the abstraction level of the program that they accept. Figure 2.6 shows the main Flink architecture. An API layer implements multiple APIs that create DAG operators for its programs. Each API has to provide utilities (serializers, comparators) that describe the interaction between its data types and the runtime. All the programming APIs are translated to an intermediate program representation that is compiled and optimized via a cost-based optimizer.

Flink Common API and *Optimizer layer* takes programs in the form of operator DAGs. The operators are specific (*e.g.*, *Map, Join, Filter, Reduce, FlatMap, MapPartition, ReduceGroup, Aggregate, Union, Cross,* etc) and the data is in nonuniform type. The concrete types and their interaction with the runtime are specified by the higher layers. The *Flink Runtime layer* receives a program in the form of a *JobGraph*. A *Job Graph* is a generic parallel data flow with arbitrary tasks that consume and produce data streams. The runtime is designed to perform very well both in setups with abundant memory and in setups where memory is scarce. *Job Graph* is responsible for hardware resource allocations to run the *job* from a resource manager, scheduling the job's tasks, execution monitoring, managing the data flows between the tasks, and recovery failures (ALEXANDROV et al., 2014).

The Cloud-ware scheduling system (GHAFARIAN; JAVADI, 2015) has been proposed as a form of a data-intensive workflow scheduling in a hybrid infrastructure like VC systems and Cloud resources, and is employed as a means of keeping to a set deadline. The system has two phases: the first divides the workflow into sub-workflows to reduce the dependency of the data on the sub-workflows. The second phase schedules these sub-workflows to the VC on the basis of resource proximity and load balancing in accordance with the *Queue Theory*. If there is a probability that the deadline will not be met with VC resources, these workflows are re-scheduled to Cloud resources to find the set deadline. The system classifies the resources in accordance with a decision tree that has the following five attributes: *CPU speed, RAM size, hard disc space, an operational system* and *a processor model*.

The algorithm proposed assumes that each task has an estimated duration time and there is information about the deadline, the QoS constraints, minimum CPU speed, and minimum size of RAM or hard disc. Thus, each workflow has a fixed deadline. Since the workflow is a DAG, the partitioning into sub-workflows is computed as the sum of estimated execution times for their tasks. The authors defined that the communication delay can be computed by Equation 2.5.1, where Sp is a service time for each connection

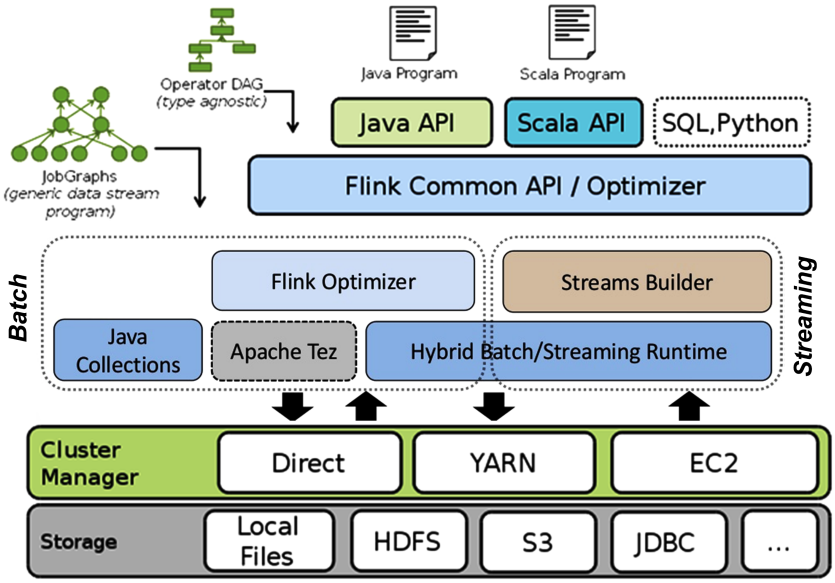


Figure 2.6: The Flink architecture

between two resources based on the store & forward flow control mechanism. The α_{net} is the network latency, β_{net} is the inverse of bandwidth and F is the volume data transfer. The σ_p^2 and λ_p are a variance and inter-arrival rate of traffic respectively, at source peer's queue.

$$R = (2S_p) + \left(\frac{\sigma_p^2 \lambda_p^2}{2((S_p^{-1} - \lambda_p)}\right) + \sum_{i=s+1}^{d} S_{p_i}, \quad \text{where } S_p = 0.5\alpha_{net} + F\beta_{net} \tag{2.5.1}$$

Cirus (PHAM et al., 2016) is a framework for *Ubilytics* which is a type of Big Data analytics applied to IoT. This provides a platform to both deploying and configuring the *Ubilytics* infrastructure in a Hybrid Cloud environment. The deployment supports heterogeneous environments based on brokers (IoT Edge). The IoT Edge tier is deployed through gateways in mobile and static sensors. The sensors and smartphones are implemented as a platform as a Service (PaaS) for IoT real-time applications. Figure 2.7, which is adapted from (PHAM et al., 2016), shows the architectural components.

The reconfiguration management is implemented by *Roboconf*, which dynamically adjusts the infrastructure to a throughput that is in accordance with the messages of the sensor. The Big Data computation is conducted through workloads in batch and real-time (using the Hadoop and Spark or Storm frameworks respectively). The platform has a Lambda architecture, although the batch processing layer has not been implemented in the prototype. Two different brokers are used to implement a message gateway and QoS properties, such as fault-tolerance, high-availability, elasticity, deterministic delivery time and other factors. The Mosquitto and RabbitMQ are brokers which offer support to the Message Queue Telemetry Transport (MQTT) protocol. Mosquitto is utilized to integrate IoT sensors and RabbitMQ into other cases. The monitoring layer runs the analysis and scaling to make decisions based on variations of the environment that can enable elasticity. A DSL-based plug-in layer provides a description of an abstraction component for Big Data applications in accordance with the use of the sensors.

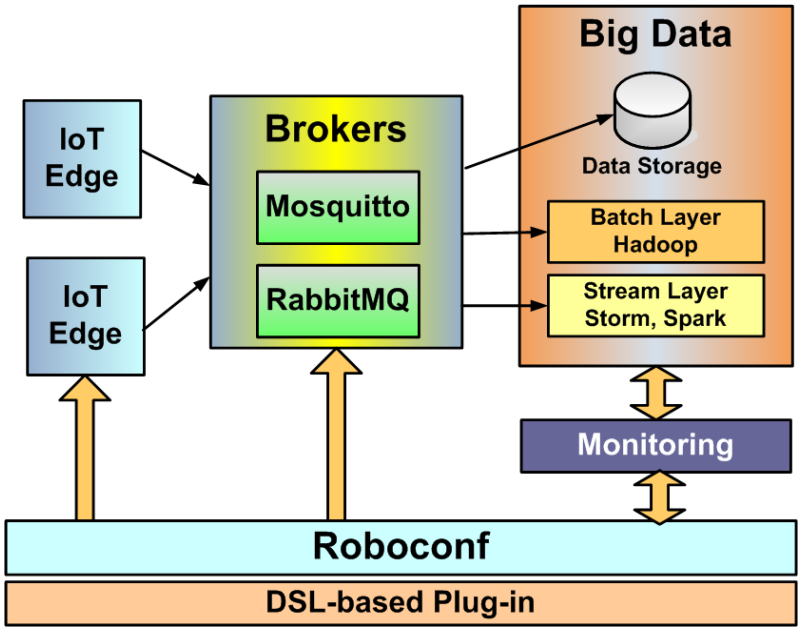


Figure 2.7: The Cirus architecture

2.6 Open-ended Problems and Related Work in Big Data Analysis

Tables 2.2 summarizes the main techniques and strategies used for the deploy of Big Data analysis. The relation of the related works refers to particular implementations, and only some works are generic systems like SALSA, Twister, BitDew-MR, Flink and CURA. Then, we will compare the strategies deployed on these Infrastructures that can be used for the solution of hybrid infrastructure in this work. An assessment of this sampling of related work found that 53% are Cloud or Multi-Cloud implementations, and 11% adopt a geo-distributed approach. Only two works employ the *Hybrid Infrastructure* as a Cloud and DG, but in these works one is an assessment of the distributed storage and the other is a scheduling algorithm. In addition, data and task distribution in a Big Data context are not examined. Thus, the study of hybrid platforms together with a "constraint analysis" are a way to find new solutions for Big Data analytic implementations. The main strategies adopted for the implementations were to evaluate the environment capacity, such as the device capacity like CPU, memory, and computational capacity to estimate the lower deployment cost for the user. 74% of the studies are using one or another approach, but only 16% include both, and the data replication strategy is the most widely used.

The network overhead is a problem because it reduces a chance to make use of *Geo-distributed infrastructures*, although the strategies adopted to reduce the number of the data transfers is not yet a policy that has been adopted in this cases. On the other hand, the implementation of groups with similar computational capacity is a strategy used by 21% of the studied implementations. The cost evaluation of the CSPs is the used strategy in 67% of implementations in Multi-Cloud environments. Only SALSA, Brokering, Twister and Cirus have implemented a broker system. The first of these orchestrates the stages of the execution phase in a systematic way; the second manages data storage and removes a single point of failure; the third creates a communication channel where the nodes can find the intermediate keys produced by *Map*; and the fourth uses the *Roboconf* as a simple orchestration mechanism to instantiate Vms.

Table 2.2: Frameworks and techniques for Big Data analysis

			Infrastructures							Strategies												Proc	
Area	Author	Proposal	Geo-Distributed	Cloud Infrastructure	Desktop Grid	Cluster	Multi-Cloud	Hybrid System	Lambda Architecture	Consider Cost	Failure Recovery	Network Overhead	I/O Throughput	Enviroment Capacity	Evaluates Storage	P/S or M_Queue	Replication	Minimize Transfers	Orchestration System	Processing in Memory	Group Strategy	Barrier Free	Batch	Event Stream
Frameworks	(EKANAYAKE et al., 2010)	Twister				X										X	X			X			X	
	(MANSOURI; TOOSI; BUYYA, 2013)	Brokering Alg.					X			X	X						X							
	(GUNARATHNE et al., 2013)	Twister4Azure		X												X	X			X		X	X	
	(LE et al., 2014)	SALSA		X										X					X					
	(JAYALATH; STEPHEN; EUGSTER, 2014)	G-MR	X				X			X		X											X	
	(TUDORAN et al., 2014)	GEO-DMS	X				X			X		X	X	X									X	
	(PALANISAMY; SINGH; LIU, 2015)	CURA		X						X														
Heterogeneous	(LIN et al., 2010)	Moon			X	X							X				X	X					X	
	(LU et al., 2012)	BitDew-MR			X						X		X				X	X				X	X	
	(KRISH; ANWAR; BUTT, 2014)	HATS				X						X	X	X	X								X	
	(ANJOS et al., 2015)	MRA++				X								X				X			X		X	
Hybrid System	(DING et al., 2011)	Hadoop Streaming				X														X				X
	(TUDORAN et al., 2014)	SC-CC		X										X									X	
	(ALEXANDROV et al., 2014)	FLINK				X			X		X	X				X				X	X		X	X
	(CLEMENTE-CASTELLÓ et al., 2015)	Hybrid IaaS Cloud					X				X												X	
	(TANG; HE; FEDAK, 2015)	HybridDFS		X	X			X		X	X	X		X	X		X					X	X	
	(VASILE et al., 2015)	HySARC^2					X			X											X			
	(GHAFARIAN; JAVADI, 2015)	Cloud-aware		X	X			X		X		X		X				X			X			
	(PHAM et al., 2016)	Cirus					X		X							X			X	X			X	X

The work of (TUDORAN et al., 2014) employs agents to collect information hardware from CSPs, but the problem remains of how it is possible to avoid excessive VM launching and thus reduce unnecessary allocation costs. In G-MR (JAYALATH; STEPHEN; EUGSTER, 2014) the performance can be very different for each execution path, thus there is still a problem of how to avoid massive data transfers if the CSPs cannot provide bandwidth guarantees. The implementation of Twister (EKANAYAKE et al., 2010) employs *Combiner* techniques to improve the task executions in Cluster, but it is not clear what the behavior of the broker systems in Cloud or Multi-clouds would be like. On the other hand, if Twister4Azure (GUNARATHNE et al., 2013) provides dynamic data with a Message Queue Infrastructure, it is not clear how it will behave when it has a finer grain for task management and data distribution. If MRA++ (ANJOS et al., 2015) could explore data distribution and tasks in accordance with its computational capabilities, it would be possible to employ this technique to improve the performance of the hybrid systems. However, if an iterative MapReduce environment (CLEMENTE-CASTELLÓ et al., 2015) extends the fault-tolerance mechanism of HDFS, how can one avoid data movements with iterative MapReduce in a Hybrid Cloud environment ?

The (PHAM et al., 2016) is the implementation that is the most closely related to our work regarding orchestration and publisher/subscriber mechanisms, although it is not clear how Hadoop (together with the other implementation layers) can provide a consistent architecture in an interactive manner. However, there is not yet any consensus of opinion about the allocation of resources in Cloud and there is no concern about the fact that there is I/O competition when multiple users are allocated for VMs using the same hardware.

Although there have been several works that address this issue, there are still many open problems about Big Data in Cloud environments. In addition, the Geo-distributed environment is not explored yet in hybrid systems. The combination of DG resource allocation with Cloud environments for data-intensive applications is a possible line of inquiry, that could be explored so that the available resources could be freely exploited. This Thesis will explore this possibility and define what are the reasonable constraints for the development of a hybrid platform to achieve this goal.

2.7 Final Considerations

This Chapter has shown the concepts and solutions of Big Data implementations in different environments, such as heterogeneous Cluster, Cloud and Multi-Cloud and hybrid infrastructures. In addition, the main open problems were discussed. The analysis of related work demonstrates that there are open-ended problems applied to Big Data. However, as these problems open up a broad spectrum, this Thesis will be confined to create a tool for analyzing a hybrid infrastructure through the BIGhybrid simulator; define a hybrid infrastructure architecture that is made possible through of the SMART platform, and establish strategies for the use of these environments.

3 SMART: A HYBRID INFRASTRUCTURE ARCHITECTURE

This Chapter shows the features of hybrid infrastructures by examining a geographically distributed model. Thus, the desirable features are displayed that are needed to achieve the definition of purpose of a minimal hybrid platform for Big Data analytics called SMART. The platform represents the cooperative scientific effort of several research laboratories for the building of a system like a hierarchical and modular infrastructure to provide a Big Data system through a lambda architecture within a hybrid infrastructure. In this context, our contribution is related to the specification of the several modules and the development a prototype to demonstrate the feasibility of this platform which we discussed in this work. Also, SMART platform provides a robust infrastructure for IoT applications through the Big Data analytics, which forms the SMART-Sent platform with the implementation of the sensors and where several students are working in different modules to make this work feasible.

3.1 Hybrid Infrastructure Characterization

The starter point of the SMART platform was the work of (ANTONIU et al., 2013). However, this initial model does not include the concepts of the dispatcher and aggregation modules, which are introduced like a contribution in this architecture. Also, the hybrid infrastructures characterization is needed for behaviour definition, which it is exhibited in this Section. The SMART platform, in Section 3.2, was born as a hybrid platform for Big Data analytics, thus the processing core is formed by the engines called Flink Apache and BitDew-MapReduce. These engines enable the Lambda architecture and desktop grid respectively to Big Data processing. The characterization this hybrid environment is the basis for the design in layers of SMART, which is extended as a use case for smart cities in the SMART-Sent specification in the following Sections.

Big Data applications can be implemented in several ways. Scattered data can be found in DNA research studies, where researchers need to investigate different databases, such as those in protein structure analysis. These applications seek a genetic mapping that requires a pre-existing reference genome to be employed for the read alignment of a gene (ZOU et al., 2014). Thus, the data processing is characterized by its ability to compare input data with different databases. This processing consists of several phases of search-merge-reduce, where the data are given an incremental update (MCKENNA et al., 2010). The solutions proposed in this work for the hybrid infrastructure consider this heterogeneous scenario and initially are based on the scope of the MapReduce ANR project[1].

[1]National Research Agency (ANR), ARPEGE 2010 call. Project number: ANR-10-SEGI-001

Some researchers (JAYALATH; STEPHEN; EUGSTER, 2014), (TUDORAN et al., 2014), (KRISH; ANWAR; BUTT, 2014) and (JI; LI, 2016) have put forward Hadoop implementation based on a geo-distributed dataset in multiple data centers. The authors state that, for instance, it is possible to have multiple execution paths for carrying out a MapReduce job in this scenario, and the performance can carry out a great deal. However, a popular MapReduce open source, like Hadoop, does not support this feature naturally, and the major Cloud Service Providers (CSPs) do not usually provide a bandwidth guarantee (ZHENG et al., 2014).

Figure 3.1 illustrates the scenario where dispersal data is used. Each locality is connected through slow links, where data transfers may not have a negligible cost. The data is scattered in the clusters. All the intermediate results must be combined to produce a single return for a Big Data analysis. These problems can be overcome by means of a hybrid infrastructure if there is a file system that supports the incremental updates and highly concurrent data sharing, for instance, in a BlobSeer deployment. A possible solution involves integrating a distributed file system like the HDFS with the use of a Cloud environment; otherwise, DGs are a large-scale infrastructure with specific characteristics in terms of volatility, reliability, connectivity, security and storage space. Both architectures are suitable for large-scale parallel processing. Finally, more complex combinations can be envisaged for platforms resulting from the use of multiple Clouds through an extension to a DG (ANTONIU et al., 2013).

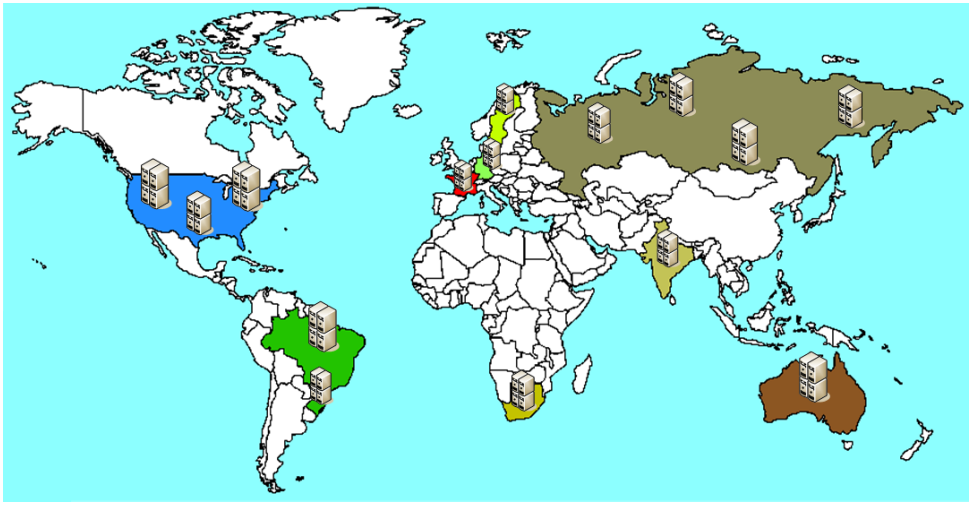


Figure 3.1: Geographically distributed data

3.1.1 Overview of the Hybrid Infrastructure Model

Different Cloud infrastructures have their own configuration parameters, and the availability and performance of offered resources can change dynamically due to several factors, including the degree of over-commitment that a provider employs. In this context, solutions are needed for the automatic configuration of complex cloud services. The Cloud infrastructure comprising heterogeneous hardware environments may need the specifications of configuration parameters at several levels such as the operating systems, service containers and network capabilities (LE et al., 2014). The users who need to execute applications may not know how to map their requirements to the available resources, this lack of knowledge about the cloud provider infrastructure will lead either to overestimating or underestimating the required capacity; both are equally bad and can lead to a

waste of resources.

A hybrid infrastructure, where there are many cloud providers with heterogeneous environments and configurations, often needs to use an orchestrator to manage the results and data input from users. The orchestrator must be decentralized (LE et al., 2014) to improve the data distribution in the network. This infrastructure enables the use of highly heterogeneous machines. Several scenarios and data strategies are possible when there is the need of the use of a public cloud for extending the capacity of a private cloud or a DG. The set scale of data distribution strategies that can be applicable to a given scenario depends on also the available bandwidth. In the case of the hybrid Big Data engine, which has two distinct DFS implementations, it may be necessary to handle data distribution in two scenarios, namely low and high-bandwidth.

With basis on previous work carried out in MR for hybrid environments (DELAMARE et al., 2012), (ANJOS; FEDAK; GEYER, 2014), Figure 3.2 illustrates the solution proposed here to a hybrid system model which depicts a *Global Dispatcher* and *Global Aggregator*. This concept that can be used as an infrastructure for services that uses multiple data abstractions. The *Global Dispatcher* located outside the cloud is a layer that handles task assignments and the management of user-provided data. The layer decouples the data storage system and manages policies for the splitting and distributing data in accordance with each system. These policies are defined in this work on the subsection 5.1. The working principle is similar to a publish/subscribe service in which the layer acts as a data producer that is afterwards consumed by workers (ANJOS; FEDAK; GEYER, 2014). The *Global Aggregator* obtains data output from both systems and merges them to obtain the final data set.

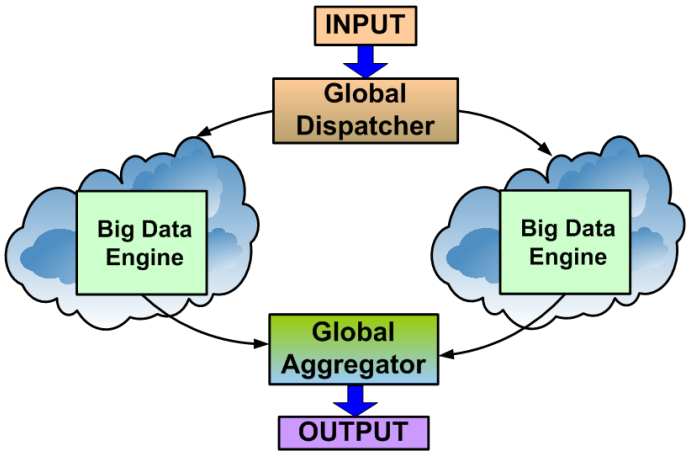


Figure 3.2: Hybrid infrastructure from a high-level view

The *Global Dispatcher* handles task assignments and input data from users. It is a centralized data-driven subsystem that manages remote data localization, policies for split and distribution of data, in accordance with the needs of each system. The working principle is similar to the publish/subscribe service where the system obtains data and publishes the computing results. This approach is simple, but risks causing a network bottleneck.

Global Aggregator receives all the *key/values* of *Reduce*, and the keys with the same index in each system are joined to the last *Reduce* function to obtain a consistent result. However, the iterative MR computations, that are undertaken by the *Global Aggregator*, are not supported by an original MR model. It is not an easy task to combine all the

Reduces from heterogeneous platforms, although it is possible to establish a new stage for MR (EKANAYAKE; PALLICKARA; FOX, 2008). One approach is to use the *MapIterativeReduce* (DOBRE; XHAFA, 2014) which creates an *Aggregator* to collect all the outputs of the *Reduce* tasks and combines them into a single result. At the end of each iteration, the reducer checks to find out whether or not it is the last, otherwise, according to (TUDORAN; COSTAN; ANTONIU, 2012), this schema might be ineffective for large workloads. The best choice is to produce a local aggregation of the keys, for instance, with a *Combiner function* to avoid unnecessary data transfers.

3.1.2 Desirable Features of the Hybrid Infrastructure for a Big Data Environment

The hybrid infrastructure uses an orchestrator to manage the results and data input for users. This must be decentralized to improve the data distribution in the network. In the particular case of Cloud and DG, fault-tolerance mechanisms adopt different policies to detect faults. A more specialized system is applied to DG due to its node volatility. It is important to define the main features of the hybrid infrastructures.

Table 3.1 summarizes the main architectural features of BlobSeer-Hadoop, BitDew-MR and the Hybrid MR environment. The hybrid infrastructure enables highly heterogeneous machines, with stable and volatile storage to avoid data loss. The extent to which a set of data-distribution strategies is applicable to a given scenario depends on how much bandwidth is available. Two independent DFS implementations are required to handle data distribution in two scenarios, namely low-bandwidth and high-bandwidth. The application profile is optimized for all file sizes in hybrid infrastructures, as the systems are independent and thus the different data sizes can be handled at the same time. The bandwidth and computational capacity of machines influence the initial assumptions for defining a straggler machine and because of this, each system must be treated in a different way.

3.2 SMART: A Hybrid Platform for Big Data

This work examines a model with various data sources, ranging from wireless sensor nodes to user interaction information in open environments, such as social networks, opendata information and other. The datasets consist of large corporate databases to broadcast media, where there is a clear need for standardization. The idea has evolved into a hybrid Big Data platform for applications processing in different domains. A “cooperation project” has been set up to achieve this vast domain. Several research institutes are involved, including the following:

- Inria, LIP, ENS Lyon, French, Ph.D Gilles Fedak;
- Universitat Politècnica de Catalunya, Spain, Ph.D Felix Freitag;
- Technische Universitat Berlin, Germany, Ph.D Volker Markl;
- JohnMoores University, Liverpool, England, Ph.D Rubem Pereira and Paul Fergus;
- University of Brasilia, Brazil, Ph.D João Paulo da Costa;
- Federal University of Rio Grande do Sul, Brazil, Ph.D Edison Pignaton;
- Belarusian State University, Belarus, Ph.D Tatiana Galibus;

Table 3.1: Comparison between the MR systems

Characteristics	**BlobSeer-Hadoop**	**BitDew-MR**	**Hybrid-MapReduce**
Heterogeneity	Moderate	High	High
Network	High Bandwidth	Low Bandwidth with distributed cache	Hybrid Bandwidth
Architecture	Decentralized	Decentralized	Decentralized
Storage	Distributed	Remote (Cloud Storage) + local	Distributed, Remote Cloud Storage and local
Management	Master/Slave	Master/Slave	Hierarchical Orchestrator
Metadata	Distributed by DHT	Centralized on Data Catalog	Distributed DHT/Data Catalog
App profile	Any	Low Communication in *Shuffle* phase	Optimized for all file sizes
File system API	Posix	Tuple Space model	Hybrid (Posix + Tuple Sace)
Data locality	Yes (One Rack only)	Yes (Affinity by node)	Implemented according to each platform
Chunk size	Fixed - 64MB	Fixed - 32MB	According each platform
Host model	Stable	Stable and Volatile	Stable and Volatile
FT mechanism	Data and Task Replication	Data Replication and transient failure support	Data and Task Replication, and transient failure support
Load balance	Strong Dynamic	Soft	Soft
Computation	Hadoop Compatilble	Barrier-free	Hybrid
Semantics of data concurrency	Multiple write, version update	Single write	Single write
Straggler management	Average execution task	Machine computational capacity	Hybrid
Storage Elasticity	High	High	High
MapReduce Semantic	More Compliance (Limited)	Restricted	Restrict

- Danube University Krems, Austria, Ph.D Thomas J. Lampoltshammer and Gabriela V. Pereira

This project aims at simplifying the deployment of Big Data services by SME. SMART, that was defined in the work of (ANJOS et al., 2015), takes advantage of the Cloud, multi-cloud and hybrid infrastructures to provide support for an SME service operation, and does not need to aggregate data in a single data center for Big Data analysis. It provides a secure and flexible cloud-based system that is capable of providing different types of services that can be combined to address the specific needs of multiple application domains. The approach for SMART is similar in high level to the work of (KRISH; ANWAR; BUTT, 2014)regarding the use of Orchestrator and publish/subscribe mechanisms. The data placement concept is to seek *network proximity* by retrieving replicas from the nearest rack to reduce network traffic. Our mission, in this project, is to define a model that enables hybrid infrastructures to be used in Big Data analysis, which is materialized in the *Hybrid Infrastructure Model*.

3.2.1 SMART Platform

The defining layers in a functional hierarchy are needed to build a platform for the SMART architecture. Figure 3.3 shows six modules: Global Collector, Global Dispatcher, Storage, Core Engine, Global Aggregator and Central Monitoring. The *Core Engine* must support the hybrid systems, *i.e.*, enable streaming and batch computations at the same time. Thus, the Flink framework is an important system that is worth considering. The BitDew-MR is another framework which can improve computational performance, with

the use of VC in a hybrid infrastructure. SMART allows computational resources to be taken from Cloud/Multi-Cloud, and Multi-Grid environments.

A *Client User API* provides an easy method for users to submit their applications and indicate the data sources. The *Client UI* is a security interface that provides a single-user identification through an encrypted key. A key which is keep by the users is employed in the *Encryption-Decryption Engine* to ensure the data is safe. The storage safety model is detailed in Annex B.1.

The *Global Collector* layer handles the management and coordination of the sensing modules. It is responsible for obtaining data from several sources and maintaining the data integrity mechanisms. The data integrity mechanisms filter possible noises in the hardware devices. The data is collected and serialized under a standard TCP/IP, which forms the communication stack for the *Global Dispatcher*.

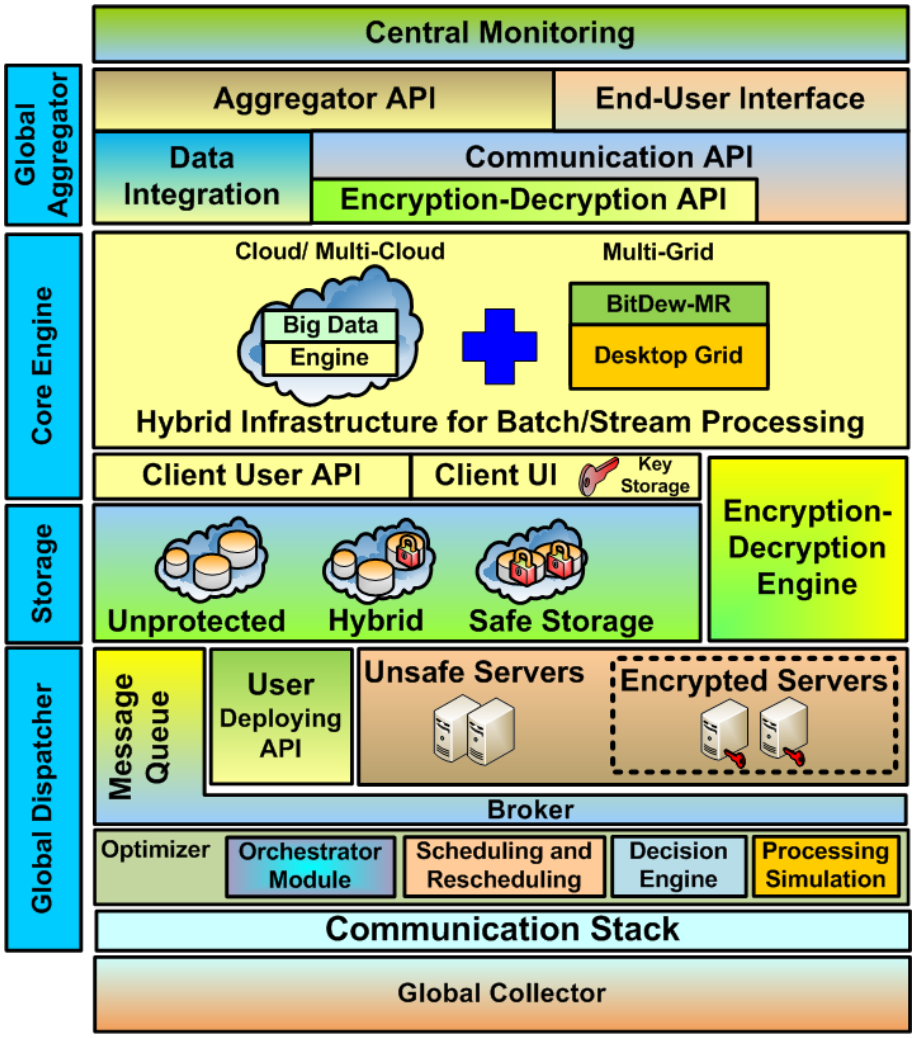


Figure 3.3: Smart platform

In the *Global Dispatcher*, the data is decoupled from the lower layers in the message queue mechanism like *Apache Kafka* (ZHANG, 2015). Data is put in a FIFO queue so that it can be distributed to machines in accordance the availability of their resources in both *Cloud/Multi-Cloud* and *Grid/Multi-Grid* environments. The *optimization* layer analyzes the volume of input data and employs the *Decision Engine* to make decisions about scheduling tasks and data through distinct environments. A simulation process

implements an execution time prediction that will be used by the *Decision Engine* to improve the accuracy of the scheduling mechanism. The user deploying module enables unsafe and encrypted data localization by the external server. The user must provide a key storage localization and the data path across the network before definitively attaching the data. In this way, the users must define the binding between the data stream and data source with the aid of the programming code. The storage-and-forward and pass-through protocols are implemented inside this layer according to the data source. The data transport endures a little delay and can be reallocated to Cloud or Desktop Grid. An orchestrator module like *Zookeeper* (WHITE, 2012) maintains distributed queues, the data structures coordination, and protocols among groups of peers. This module avoids a single point of failure and their participants do not need to know one another.

The intermediate results are processed in the *Core Engine*. These results will be serialized by the *Global Aggregator* what must carry out the data consolidation. The *Data Integration* module supports network overlays to build a pathway for operations among workers which need to execute an aggregation mathematical function of the similar keys in a Big Data application. The last phase of the data processing is designed to generate an iterative execution and provide the results of the consolidation. A *Communication API* is necessary to integrate the workers into a virtual network of data computation. The *Aggregator API* is a module that orchestrates the results of the aggregation and maintains the safety data mechanism for the end-users. The users submit their applications on *End-User Interface* and monitors the results in the *Central Monitoring* layer.

3.2.2 SMART-Sent: A Hybrid Platform for Big Data Integrated with IoT

As a use case, the **SMART** with **Sen**sor **T**echnology - *SMART-Sent Platform* - has been designed on the basis of two other systems: the SMART platform and the Urbosenti platform (ROLIM et al., 2015). The scenario involves the third digital technological revolution, the IoT (DUQUENNOY; GRIMAUD; VANDEWALLE, 2009). The evolution of mobile devices has enabled a high degree of Internet connectivity. The new range of Internet-connected devices available today in other scenarios, such as smart cities, can result in a better management of device resources, public infrastructure, transport, among other factors. The number of so called smart devices connected to the Internet is approaching 25 billion and they range from home sensors employed to automate daily home tasks to wearable devices such as smart watches.

The technologies developed within the scope of the SMART-Sent, are aimed at improving the functionality of IoT by improving the frameworks and mechanisms used for the analysis of data generated by connected devices. This can lead to important developments by introducing a plethora of new services by companies that can provide them more easily, as well as the integration of new methods for analyzing data, and managing the available Cloud infrastructure. In a similar way, this project seeks to integrate stream processing and optimized cloud features for Apache Flink. Other functions can be added, thus creating an "ecosystem" that allows practitioners and researchers to mutually benefit and make a contribution by devising new algorithms for data summarizing, curation, storage and analysis.

Figure 3.4 illustrates an architecture from a high-level view. A SMART-Sent Platform includes different sensor types, such as, social networks (Twitter, Facebook, LinkedIn, etc), *crowdsourcing* (online users with smartphone or tablet which send information to a central information collector) and sensor devices (sensors in smart cities). These sensors collect information about events in real-time (streaming processing). The streaming

processing has several data sources including small sensors spread across cities. The sensing devices send their information to the nearest desktop. A data collector application, deployed in each desktop, conveys the information from the sensors to the *Global Dispatcher*. Datasets with related information (in batch processing) and streaming processing (originated from sensors) are combined in Big Data analysis. Public or private agents receive compiled information which allows them to take action on the basis of defined thresholds in a central monitoring system.

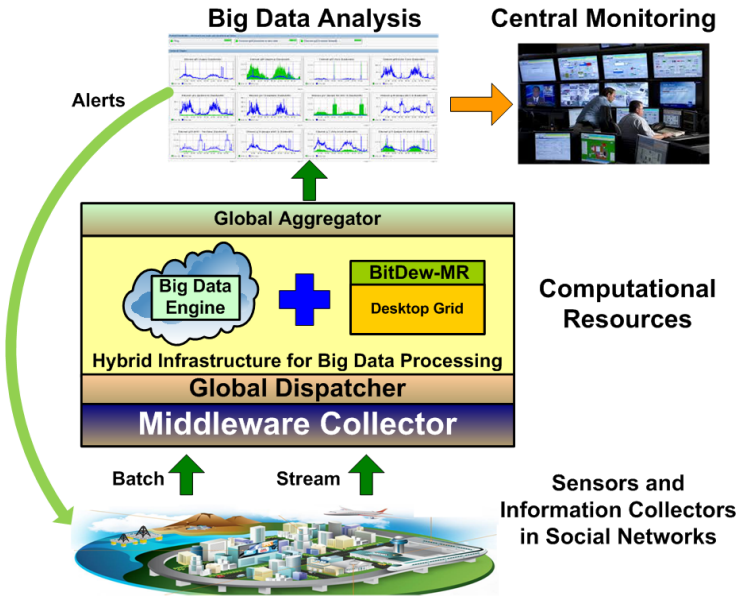


Figure 3.4: SMART-Sent architecture from a high-level view

Resource elasticity - the capacity to add/remove resources allocated to a service to match its workload - is one of the key selling points of cloud computing. As appealing as this feature may seem, it is hardly a result of magic. Providers often make available a much larger number of resources than what their clients demand, to maintain the illusion that the cloud is unlimited. Spare capacity that is not allocated to customers at the normal renting cost is provided at a discount price. However, studies of (NEJAD; MASHAYEKHY; GROSU, 2015) attempting to deconstruct the price of *Amazon Spot Instances* argue that the spot price is a result of random processes and does not reflect demand, a fact that leads researchers to believe that Amazon's infrastructure is considerably larger than what their customers in fact utilize. Hence, methods for handling elasticity in a more responsive manner from a cloud provider's perspective as well as from that of the customers' are essential to allow a more efficient use of the IT infrastructure. The SMART-sent platform can assist in the study of algorithms and offer new mechanisms for a complex event analysis.

3.2.3 The SMART-Sent Platform

There is a perfect link between the SMART and Urbosenti platforms, because from the high-level perspective of SMART, the *Collector* is a service to acquire information, whereas from the standpoint of a Urbosenti, Big Data analytics is a service infrastructure. Figure 3.5 shows the SMART Platform architecture. The *Global Collector* now represents a hierarchical infrastructure. A module, called *Runtime tools*, manages and monitors the information collector. Logs and performance indicators comply with the *policy enforce-*

ment to achieve SLA specifications. The *SOA Registry* module defines standards communication protocols like XML, Avro, and JSON that will be used for obtaining data from sensors. Data from a *Universal Description, Discovery, and Integration* UDDI manages the life-cycle and service discovery. A *Profiling* module controls the sensing modules and system users to manage access control and discovery information. The *Security API* maintains an interface with a security module to ensure data security.

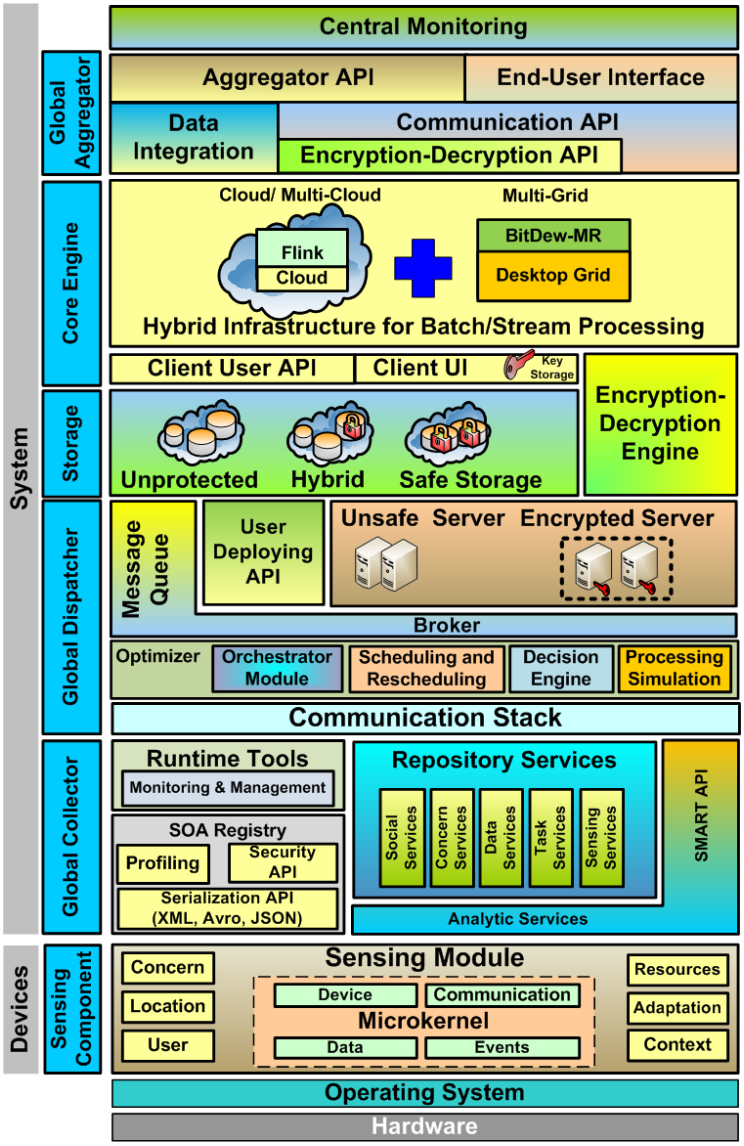


Figure 3.5: SMART-Sent platform

The *Analytic Services* layer enables data to be sent for Cloud and provides controls to access the Big Data analytics services through the SMART API. This API connects with the communication layer to provide a safe transmission channel. The *Repository Services* maintain all the available services so that they can carry out their functions properly. The services can be defined as follows: *Social Services* are responsible for *crowdsourcing* information; *Concern Services* manage security and privacy matters related to the users; *Data Services* is responsible for manipulating filters that eliminate noise from hardware devices and generate clean data for higher layers; *Task Services* coordinate the task of obtaining data from sensors, and interacting with other modules in this layer; *Sensing*

Services allows the interaction of several sensors types, through open standards running in the *Data Services* and *Concern Services* modules.

The *Sensing Component* layer is combined with mobile devices and fixed sensors and scattered across the different localities. They collect "intentional" information when the users are interacting with their devices or without an intention when an application obtains data directly from the environment. The *Microkernel* module is a core installed in each device to manage the basic services. The functions are structured as follows: *Device* provides information about a particular feature, such as the device name, Ip address, localization, and position, interfaces and sensors kinds, etc; *Communication* is related to TCP/IP standard protocols, such as IEEE 802.11 b/g/n, IEEE 802.3, GPRS/EDGE/3G, Bluetooth, etc; *Events* interface obtains a device status, the position changes of devices and traces of movements; *Data* handles storage operations and data retrieve.

The events obtained from *Microkernel* are available in components such as the following sensing modules. The *Concern* module manages the privacy, encryption and safety associated with an entity; The *Localization* component stores geolocation, points of interest (POI) and Location Based Services (LBS); The *User* module handles personal datasets like social network profiles and user preferences; The *Resource* module discovers available resources in the neighborhood and monitors the local resources; The *Adaptation* module controls devices and the behavior of applications. Basic policies are defined to forecast the device response for some of the defined thresholds; and the *Context* module is a mechanism to support reasoning, knowledge and discovery within a specific context.

However, it should be stressed that building a complex system such as SMART-Sent is an arduous task. The first stage of project development consists of defining a test environment where it would be possible reduce the fault risk. This environment must make it possible to evaluate the best strategies for data distribution and task allocation, and hence make it feasible to establish the platform for Big Data executions. As a result, the BIGhybrid simulator as a tool can help to analyze strategies for Big data in hybrid environments and must allow the SMART-Sent platform to be built with a reasonable performance.

3.3 Final Considerations

The challenges of developing hybrid infrastructures involve factors such as synchronization management, load balance control and achieving as low a rate of data transference as possible between the different sites, owing to the fact that CSPs do not provide bandwidth with a guarantee for data transfer in execution time, in their SLA agreements. In addition, managing hybrid infrastructures is a complex task since each CSP has few resource management tools available to the users. On the other hand, the users also have little expertise in the task of allocating computational resources in accordance with the needs of their applications. Thus, it is essential to offer tools that can provide the user with a wider choice of resources and make it easier to achieve an efficient fine-grain environment without the need for a knowledge of the available resources of CSP.

In this context, simulators can be added to the system architecture as an efficient tool for providing an interface that is easy for users, without losing a minimum degree of granularity. The BIGhybrid simulator shown in the next Chapter enables users to have a practical interface and a powerful tool for resource allocation which can be built into the SMART-Sent platform.

The SMART-Sent architecture represents an international attempt to bring about coop-

eration that is the result of researchers working together to find solutions for the development of the Big Data implementations for small and medium-sized companies. However, this platform is only a preliminary study and as it is still in its early stages, some further changes might be added in the future.

Currently, other two new Ph.D. students are conducting their Thesis which with basis on the SMART Platform, as well as Master's Degree students and four undergraduates are working on issues involving this platform. Two Master's Degree students at Belarus University are working on the security model for Big Data related to the SMART-Sent Platform. These activities illustrate the significance of this development.

4 SIMULATION OF HYBRID INFRASTRUCTURES

The use of real-world testbeds to evaluate MR applications is difficult since there is a lack of reproducibility in the experimental conditions for DG and fine-tuning of the Cloud software stacks is complex. This Chapter presents the BIGhybrid simulator, which is a simulation tool for *hybrid infrastructures* related to Cloud and Desktop Grid environments. The features of the simulator are defined carefully with the aim of reproducing the most important characteristics of environments that conduct executions similar to those found in the behavior of the real-world. The development of the BIGhydrid simulator as an analytical tool for *hybrid infrastructures* and Big Data analysis is a relevant contribution for this Thesis.

4.1 BIGhybrid Simulator for SMART Architecture

BIGhybrid is a toolkit for MR simulation in hybrid environments. It has the focus on Cloud and DG and is validated employing real-world experiments. The simulator itself is based on the SimGrid framework (CASANOVA et al., 2014). The main purpose of this Chapter is to demonstrate that the BIGhybrid simulator has features that allow it to carry out accurate simulations and that it is able to simulate the execution behavior of two types of middleware for two distinct infrastructures: BitDew-MR (FEDAK; HE; CAPPELLO, 2008; MOCA; SILAGHI; FEDAK, 2011) for Desktop Grid Computing and Hadoop-Blobseer (NICOLAE et al., 2010) for Cloud computing (called as Cloud-BlobSeer). BIGhybrid has several desirable features: a) it is built on top of SimGrid with two different simulators - *MapReduce over SimGrid* (MRSG), a validated Hadoop simulator (KOLBERG et al., 2013), and *MapReduce Adapted Algorithms to Heterogeneous Environments* (MRA++), a simulator used for heterogeneous environments (ANJOS et al., 2015); b) it has a trace toolkit that can assist in analysis and graphically shows the task executions; c) it is a trace-based simulator that is able to process real-world resource availability traces to implement realistic fault-tolerance scenarios. Some traces can be found on a website called Failure Trace Archive (FTA), which is a centralized public repository of resource availability traces for various parallel and distributed systems (KONDO et al., 2010); and d) its modular design allows for further extension.

BIGhybrid can be used for evaluating scheduling strategies for MR applications in *hybrid infrastructures*. We believe that this kind of tool is of great value to researchers and practitioners who are working on big data applications and scheduling. For validation purposes, the experiments are executed over Grid5000 (INRIA; CNRS, 2016). Grid5000 is an experimental testbed, supported by INRIA, CNRS, RENATER and several universities in France. The experiments demonstrate that there is a similarity between the simulations of BIGhybrid and those of the MapReduce real experiments, which can serve to validate

the simulator. BIGhybrid enables the study of variations and patterns for the deployment of the dispatcher and aggregation modules on the SMART platform.

4.1.1 Model of the BIGhybrid Simulator

The idea behind the BIGhybrid simulator is to optimize *hybrid infrastructure* environments such as Cloud services with the available resources of a DG system. BIGhybrid is modular and built on top of the SimGrid framework (CASANOVA et al., 2014). SimGrid is a simulation-based framework for evaluating clusters, Clouds, grid and P2P (peer-to-peer) algorithms and heuristics. SimGrid is responsible for the simulation of all the network communication and task processing in our implementation. Unlike other simulators, BIGhybrid has two independent systems. This enables it to use different configurations for DFS, schedulers, input/output data size, number of workers, homogeneous and heterogeneous environments, as well as combining two different platforms, and making use of parallel simulation into two environments. In view of this, it is possible to set up several types of network and architecture platforms with simple modifications in the BIGhybrid simulator, which can lead to a more generic *hybrid infrastructure*.

The BIGhydrid simulator generates traces from each system to allow an individual or collective analysis to be conducted within the same time frame. The simulator enables several strategies to be investigated to determine the best data distribution and resource allocation of MR applications in *hybrid infrastructures*, and thus address the bottleneck issues. The BIGhybrid simulator will allow finding the best strategies for using *hybrid infrastructure*.

BIGhybrid is built on two components described in previous work: MRSG which simulates Cloud-BlobSeer with Hadoop; and MRA++ which simulates BitDew-MR. Figure 4.1 illustrates the architecture of BIGhybrid, which comprises four main components: input data management (Global Dispather), the Cloud-BlobSeer module, the BitDew-MR module and an integration module for results (Global Aggregator). SimGrid simulates the platform, network and CPU computation on nodes. The communication between BIGhybrid and SimGrid is achieved through the use of MSG, one of the many application programming interfaces provided by SimGrid.

MapReduce has three main phases: 1) The *Map phase* reads the data from the distributed file system and calls on the user map function to emit (key, value) pairs as intermediate results; 2) In the *Shuffle phase*, the map nodes sort their output keys into partitions, that are then "pulled" by the *Reduce* nodes. This means that each reduce task will process the keys that belong to a specific partition. When all the data transfers have been completed, the *Reduce* nodes carry out the task of merging the data pulled from the map nodes; 3) Finally, the *Reduce phase* calls the user's reduce function and writes the output back into the DFS. These phases are simulated in the Cloud-BlobSeer and BitDew-MR simulations. More specific details about the MRSG simulator and MRA++ can be found in (KOLBERG et al., 2013) and (ANJOS et al., 2015).

A user can specify an input function for each system, as well as for individual *Map* and *Reduce* functions. With BIGhybrid it is possible to build platforms for real infrastructures through the platform description of configurations and real environments that use the FTA. This means that the BIGhybrid simulator can provide up to 256 settings of configurations in the same simulator, *i.e.*, 2^n setups with 8 different modules. In addition, it is possible to make adjustments to several kinds of strategies and configurations between the two environments, to find the best load balance without data loss.

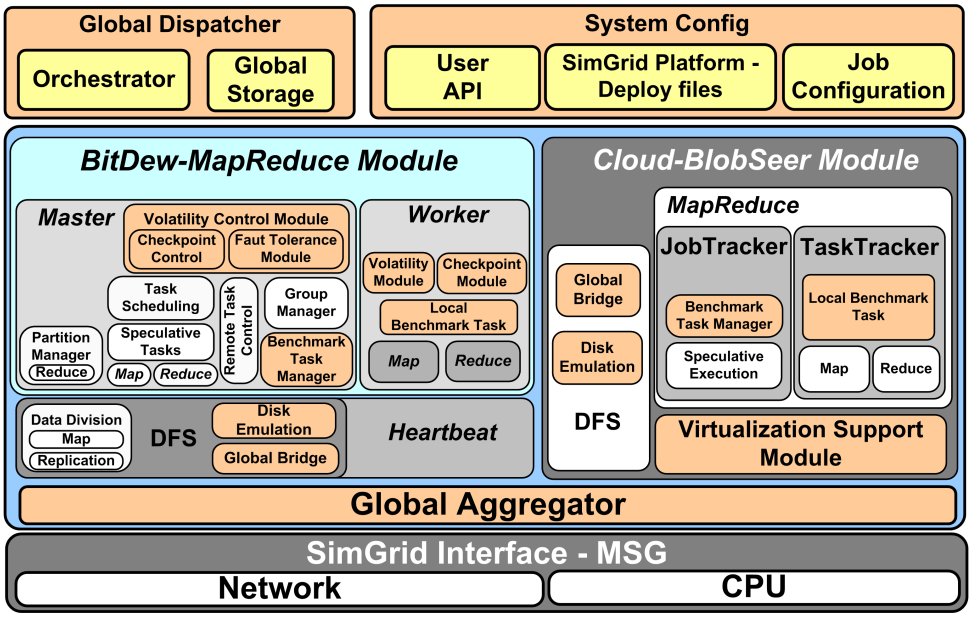


Figure 4.1: Architecture of the simulation

4.1.2 Cloud-BlobSeer Simulation Module

The Cloud-BlobSeer module reproduces the behavior of the MR framework and invokes SimGrid operations whenever a network transfer or processing task must- be performed. This simulation follows the Hadoop implementation, with a heartbeat mechanism to control the task execution. The architecture of Cloud-BlobSeer comprises the following modules: API of input users code, DFS, MapReduce functions, master (Jobtracker) and slaves (Tasktracker).

The DFS is implemented as a matrix that maps chunks to nodes. The master node knows where each chunk is "placed", as it occurs in the real implementation. Moreover, each chunk can be linked to more than one node, which allows a simulation of chunk replicas. The Cloud-BlobSeer simulation implements the node distribution in a single rack. The next version of BIGhybrid will use the storage simulation API of SimGrid, on *Disk Emulation Module*, to simulate the storage behavior. As at the time writing, disk simulation is specified as an I/O cost in the configuration file in the User API. The behavior of virtual machine is simulated as an additional task cost and implements disk contention. Disk contention represents an additional computational cost where a user is sharing the same hardware resources with another virtual machine.

4.1.3 BitDew-MapReduce Simulation Module

The implementation of MapReduce in BitDew is mainly targeted to the DG systems (LU et al., 2012) and employs mechanisms to alleviate the impact of host churn. The implementation relies on master and worker daemon programs. A MR API on top of BitDew-MR handles the *Map* and *Reduce* functions through BitDew-MR services. The data locality of Hadoop MR was implemented as a data attribute to support the separation of the input data distribution from the execution process. With the Hadoop implementation, when the network experiences unavailability, a heartbeat interval signals to the master that the host is dead, whereas in BitDew the network might be temporarily offline without undergoing any failure. The fault-tolerance mechanism (FTM) needs a

synchronization schema, as pointed out by (TANG; FEDAK, 2012), where transient and permanent failures can be handled. A barrier-free computation is implemented in the BitDew-MR simulation (as can be seen in Section 4.3).

BIGhybrid implements speculative tasks to create compatibility with the implementation of the MapReduce framework. The speculative task is launched at the execution end for both *Map* and *Reduce* phases to accelerate the executions of stragglers. The task scheduling is implemented through a *task scheduling* module in each simulator, which follows the locality principle described earlier. Hence, when the master node receives a heartbeat from a worker, and has checked the available slots for map processing, it will attempt to schedule a task in accordance with the following criteria:

1. An unassigned task that processes a chunk stored locally in the worker;
2. An unassigned task that is stored in another worker;
3. A speculative task that processes a local chunk;
4. A speculative task with non-local input.

A *Reduce* task does not have a locality and its input is spread among the workers that processed the *Map* tasks. For this reason, when assigning *Reduce* tasks, the scheduler distinguishes between unassigned and speculative tasks. In both the *Map* and *Reduce* phases, a speculative task is scheduled when all the regular tasks have already been processed or assigned to other workers.

4.2 Features of the BIGhybrid Simulator

The BIGhybrid simulator was developed to achieve an accurate simulation and maintain features compatible with Cloud-BlobSeer and BitDew-MR for a real hybrid environment. The Cloud-BlobSeer simulates Cloud environment, and BitDew-MR simulates Desktop Grid environment in the hybrid environment. The main features of the simulator are summarized in Table 4.1. In especial, the computational semantics follows the Hadoop-MapReduce compatibility in Cloud, and the semantic of the Barrier-free computation contends the Desktop Grid.

In BIGhybrid, the *Global Dispatcher* can be either manual or automatic. In the manual version, the user defines a function for data distribution and a job configuration. These configurations can explore the number of *Map* and *Reduce* tasks, input data, data size, chunk size and so on. In automatic release, an *Orchestrator* deals with user queries and distributes tasks to the systems. A *Global Storage* is used to maintain user-related data, so that the *Orchestrator* can initialize a new task, if necessary.

The results of the *Global Aggregator* module are implemented as a single Reduce task after the last current Reduce task has been completed. The processing results are tracked and saved in a file for future analysis. A toolkit for the system execution analysis was implemented to assist in creating both homogeneous and heterogeneous platforms, and make execution traces based on visualization traces supported by SimGrid. This toolkit enables users to analyze the whole execution system and change the strategies when needed. The traces can be individual, as well as for all the simulations in the system.

Table 4.1: BIGhybrid simulator features

Characteristics	Cloud simulation	Desktop Grid simulation
Behavior	Simulated or through trace reproducing	Simulated or through trace reproducing
Hardware platforms	Based on SimGrid	Based on SimGrid
Chunk-size	Defined by user	Defined by user
Computational Semantics	Hadoop-MapReduce Compatible	Barrier-free computation
Data distribution	Data locality	Data locality according to the computational capacity of the machines
Environment	Homogeneous or Heterogeneous	Homogeneous, Heterogeneous and Volatile
DFS	Simulated by Matrix	Simulated by Matrix
Fault-Tolerance Mechanism	Data and Task Replication	Data, Task Replication and Host Failure Recovery
Traces and Logs Generation	Yes	Yes
Storage	Simulated by cost or disk emulation	Simulated by cost or disk emulation
Straggler management	Average execution task	Machine computational capacity
Synchronization schema	Heartbeat by time stamp	Heartbeat by time stamp and Failure of Time-out Period
Virtualization Support	Disk contention	No
Network	Flat-Three, Ethernet, Token-Ring, P2P, Hierarchical and Non-Hierarchical	Flat-Three, Ethernet, Token-Ring, P2P, Hierarchical and Non-Hierarchical

4.2.1 Details of the Volatility Module and Communication Mechanism

The volatility control module implements the FTM, which is an environment designed to recover data and the task node failures. Figure 4.2, adapted from (TANG; FEDAK, 2012), shows the BitDew-MR synchronization schema that was implemented in the BIGhybrid simulator for failure detection. The node updates an *alivetime* variable at each *synchronization interval*. The *synchronization interval* is a period when the node synchronizes its state with the master. If the node goes offline, the status of the *alivetime* variable will be changed to unconnected and a failure will be detected. A period of failure-timeout is the time that the master waits to change the node status from online to offline.

This is necessary because in the DG environment, several hosts are behind firewalls and slow links that can cause a lack of momentaneous connection without the node having a shutdown. This is defined by the user in the configuration file of the simulator as a waiting time that begins with the last valid heartbeat. As each node is able to begin its

own processing in an undefined time, each node has its own *period of failure-timeout*. The master waits for a *transient period* during which it does not take any recovery action and does not send any new data or tasks to this node. When the *transient period* achieves a period of failure-timeout, the *alivetime* status becomes offline. At this moment, the system emits a backup task from the replicas in the FTM and removes the node from the database to avoid management overhead.

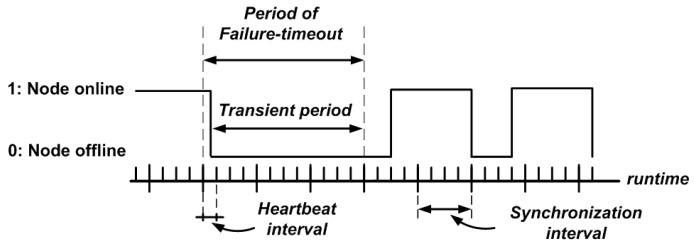


Figure 4.2: BitDew-MR synchronization schema

The communication occurs through a message exchange mechanism of SimGrid Mailbox API. Short messages (SMS) are sent by the BIGhybrid simulator to task control and data transfers in a full-duplex communication channel. Each node initializes the worker, heartbeat and mailbox processes.

Figure 4.3 shows the main communication schema between the master and workers. When a worker starts, it sends a message to become registered on master and receives an identification called “wid”. This SMS registration supplies information about the processor types/characteristics and local disk size of the worker. After this, the master determines the computational capacity of the nodes and how much data will be sent by the Data Transfer service, after the data has been distributed to each of them. The main communication mechanism is the heartbeat, which has a global time defined by the number of nodes.

After the node has received both the data and tasks, it begins the processing and the heartbeat goes to sleep until the next synchronization interval. When the heartbeat is turned on, the node sends a SMS_STATUS to the master which gives information about the progress of the task or conclusion of the execution. If the master does not receive a heartbeat, it calculates the period value of failure-timeout and wait, and during this period it does not send anything to the worker. If the worker sends a heartbeat before the waiting time has been completed, the period value of failure-timeout is reinitialized. On the other hand, if the master detects a failure, it triggers a FTM.

The FTM discovers which tasks have been completed by the worker shutdown, and finds a data replication to relaunch the task for another free node. The replication mechanism updates the data replication factor. The status of the worker is upgraded in the database and its “wid” is removed from the database. FTM is an important mechanism that is implemented in BIGhybrid and simulates real-world desktop grid environments. This behavior is demonstrated in Section 4.3.

The BIGhybrid simulator also implements a “slow beginner”. This means that if a node begins late, any task failure and data from the node failure, will be launched for this node to be executed. This behavior is checked in BitDew-MR if the failure occurs when the data is still being distributed. Otherwise, if the job executions have already begun, the node does not receive any data or task.

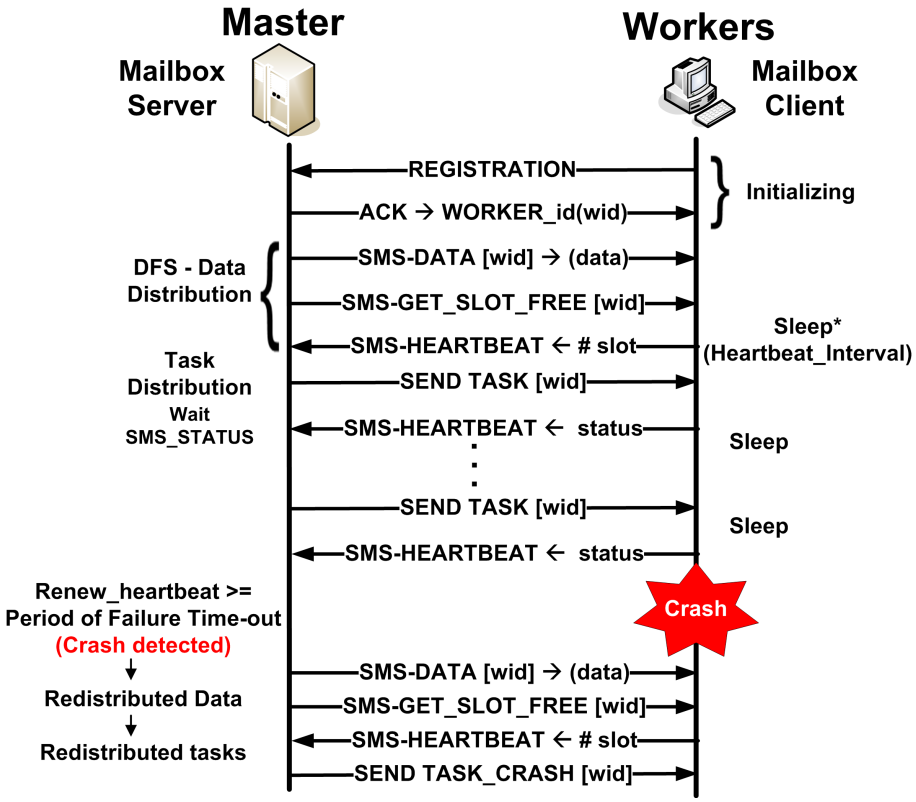


Figure 4.3: Communication schema

4.2.2 Related Work vs. BIGhybrid Features

Related work discussed in previous Sections and summarized in Table 4.2 shows that there is a long way to go to find solutions for Cloud and multiple Cloud environments. This illustrates the need to develop new tools to investigate such *hybrid infrastructures*. The BIGhybrid simulator makes it possible to study complex environments like hybrid environments and obtain more fine-grained runtime. The hardware infrastructure is modeled like real-world machines and the node behavior can be determined from real-world traces. These traces have origin from the resource availability in volatile environments and are obtained from the FTA website. The BIGhybrid simulator approach allows us to analyze generic data-intensive applications with MapReduce through traces of real executions (as in (CHEN et al., 2011), as demonstrated in Section 4.3).

Table 4.2: BIGhybrid vs. related work simulators

Simulators	Simulated Features							
	Grid	Cloud	Federated Cloud	Hybrid	Big Data Support	Failure Support	Trace Support	SLA Support
GroudSim	Yes	Yes	No	No	No	Yes	Yes	No
CloudSim	No	Yes	Yes	No	No	No	No	No
InterCloud	No	Yes	Yes	No	No	No	No	No
Federated CloudSim	No	Yes	Yes	No	No	No	No	Yes
AweSim	No	Yes	Yes	No	Yes	No	No	No
Dynamic CloudSim	No	Yes	Yes	No	No	Yes	No	No
BIGhybrid	No	Yes	No	Yes	Yes	Yes	Yes	No

4.3 Evaluation of the BIGhybrid Simulator

This section describes the environment setup and results of the evaluation so that the features and scalability of the simulator can be demonstrated. The experiments are separated into the execution time profile and behavior execution. The execution time profile shows how similar the simulation execution is to the real execution times and the behavior execution shows which features are reproduced in the simulator.The executions are repeated 30 times for each experiment in real-world experiments and a calculation is made of the means, standard deviation and coefficient of variation. Since the BIGhybrid is a deterministic simulator, its measurements are compared through a statistical evaluation based on the results of real-world experiments.

4.3.1 The Environmental Setup

Four environments have been considered. The first experiment has three different clusters in a simulated environment. One cluster simulates a homogeneous environment and has a 5-node cluster of 1 CPU with 2 cores each, 5.54 GFlops of processing capacity and 1 Gbps network. Two other clusters simulate the heterogeneous environment where one contains 5 heterogeneous machines with 1 CPU of 2 cores each and a network of 10 Mbps, and another contains 15 heterogeneous machines with 1 CPU of 2 cores each and a network of 10 Mbps, with 20% of volatile nodes. The machines in a heterogeneous environment have a processing capacity ranging from 4.76 GFlops to 6.89 GFlops, where this processing capacity is determined by a log-normal distribution according to (JAVADI et al., 2011). The second experiment is formed of clusters from the Grid5000 environment. This grid is an experimental testbed, carried out under the INRIA ALADDIN development plan with support from CNRS, RENATER and several universities in France. The experiments were divided into two environments: First, there was a homogeneous environment formed of clusters (as described in Table 4.3) with a 1 Gbps network. Second, there was a heterogeneous environment formed of machines (as described in Table 4.4).

Table 4.3: Grid5000 environment for homogeneous experiments

Site	# Host	Properties (# Processor,# Cores, RAM, HDD)	Performance (GFlops)
Sophia	16	2 x Intel Xeon E5520 @ 2.27 GHz, 4, 32GB, 557GB	55.46
Reims	32	2 x AMD Opteron 6164 HE @ 1.7 GHz, 12, 47GB, 232GB	121.30
Grenoble	64	2 x CPUs Intel Xeon E5520 @ 2.27 GHz, 4, 23GB, 119GB	55.45
Nancy	128	1 x Intel Xeon X3440 @ 2.53 GHz, 4, 16GB, 298GB	31.01

Table 4.4: Grid5000 environment for heterogeneous experiments

Site	# Host	Properties (# Processor,# Cores, RAM, HDD)	Performance (GFlops)
Sophia	30	2 x AMD Opteron 2218 @ 2.6 GHz, 2, 4 GB , 232 GB	16.80
Sophia	22	2 x Intel Xeon E5520 @ 2.27 GHz,4, 32 GB, 557 GB	55.46
Total	52	50 workers and 2 BitDew-MR servers	

The third experiment involves a cluster of 2,000 nodes and each node has 1 processor Intel Xeon X3440 @ 2.53 GHz, 4, 16 GB and a network of 1 Gbps. This configuration

represents a characterization of MR applications devised by Chen (CHEN et al., 2011) which was drawn on to define the large-scale setup. Chen examined the MR traces of two production environments from Yahoo and Facebook. The Yahoo traces were obtained from a 2,000 node cluster and contained 30,000 jobs spanning a period of over 3 weeks. The cluster was used to run applications that require batch, interactive and semi-streaming computations. For the purposes of this work, only "*aggregate, fast job*" applications characterized by Chen are considered. Table 4.5 shows the details of these applications, including the number of *Jobs*, input average data size for each *Job*, *Map* time and *Reduce* time. This *Job* has 568 GB of input and 9,088 tasks with an execution time of 322.64 seconds from *Map* and 703.32 seconds from *Reduce*.

Table 4.5: Yahoo traces (2,000 machines cluster)

# Jobs	Input	*Shuffle*	Output	*Map* Time	*Reduce* Time	Label
838	568GB	76GB	3.9GB	270,376	589,385	Aggregate, fast job
91	206GB	1.5TB	133MB	983,998	1,425,941	Expand and aggregate jobs
1,330	36GB	15GB	4GB	15,021	13,614	Data transformation

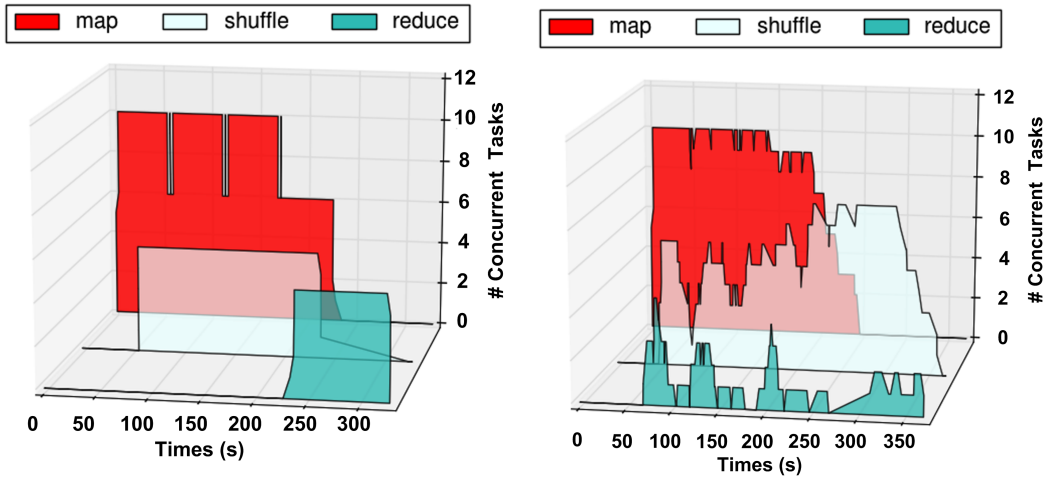
The operational system is Debian Wheezy-x64 with Hadoop 1.2 for homogeneous environments and Debian Wheezy-x64 with BitDew-MR 0.2.2 and Java SUN 1.6 for heterogeneous environments. The simulator is BIGhybrid version 1.0 - Build 3.11 with SimGrid 3.11.1 and is available in https://github.com/Julio-Anjos/Bighybrid.

4.3.2 Study of a Simulated MapReduce Execution

In this experiment, an attempt is made to evaluate if the BIGhybrid simulator is able to simulate the main features of two existing MapReduce runtime execution environments, namely, Cloud-BlobSeer and BitDew-MR. The experiment consists of the simulation of a MR execution using Cloud-BlobSeer and BitDew-MR in two different infrastructures, in homogeneous and heterogeneous clusters respectively. In this experiment, we seek to obtain the execution profile of the MR execution, *i.e.* the number of concurrent task executions during the *Map* and *Reduce* phases, and the number of data transfers during the *Shuffle* phase. The homogeneous environment was used to process 2GB of data, 5 mappers, 5 reducers and chunk size of 64 MB, while the heterogeneous environment was used to process 1.1 GB of data, 5 mappers, 5 reducers and chunk size of 16 MB.

Figure 4.4 and Figure 4.5 show the MR execution profile simulated by the BIGhybrid simulator. In Figure 4.4, the red, white and blue colors represent the *Map*, *Shuffle* and *Reduce* phases respectively. The execution time in the x-axis is measured in seconds, and the number of concurrent tasks for *Map*, *Reduce* and *Shuffle* in the y-axis are measured in units. Figure 4.4.a shows an execution of the Cloud-BlobSeer @ homogeneous environment. The *Map* tasks produce intermediary keys that are sent to the *reducers* during the *Shuffle* phase, and the *Reduce* tasks begin once the *Map* tasks have been completed. The *Map* tasks are restricted to 10 concurrent tasks (two tasks per node). The *Shuffle* begins after 5% of the *Map* tasks have been completed. The number of *Reduce* tasks is restricted to concurrent *Reduce* tasks, in this case 5 tasks. The *Reduce* phase begins after all the *Map* tasks have been completed. This shows the correct execution of the *barrier* implementation. This is a synchronization *barrier* between the *Map* and *Reduce* phase, in exactly the same way as was described in the execution of MR framework in Chapter 2. The BitDew-MR execution @ heterogeneous environment is shown in Figure 4.4.b.

The *Reduce* phase shows that the tasks start as soon as the machines have some data to process. This provides evidence that the *barrier-free* behavior that was implemented in the BitDew-MR can be reproduced. It should be noted that, as the link is 10 Mbps, the data transfers take longer to complete during the *Shuffle* phase.



(a) Cloud-BlobSeer @ homogeneous environment

(b) BitDew-MR @ heterogeneous environment

Figure 4.4: MapReduce execution profile simulated by BIGhybrid simulator

Figure 4.5 shows the *MapReduce* execution time in a hybrid environment simulated by a BIGhybrid simulator. The red, green and blue colors represent the *Map*, *Shuffle* and *Reduce* phases respectively. The execution time in the x-axis is measured in seconds, and the number of concurrent tasks for *Map*, *Reduce* and *Shuffle* in y-axis are measured in units. This shows a parallel execution of the MR execution in a Hybrid environment from the previous experiment. *Map* tasks have a restricted number of concurrent tasks for each system. The chart shows that *Reduce* tasks begin the data prefetching earlier in the *hybrid infrastructure*.

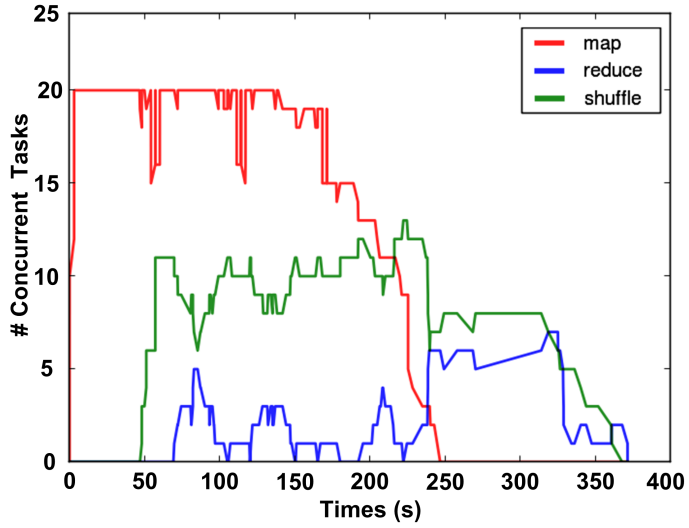


Figure 4.5: MapReduce execution profile in a hybrid environment simulated by a BIGhybrid simulator

4.3.3 Volatile Behavior and Failure Tolerance Mechanism

In this experiment, an attempt is made to evaluate if the BIGhybrid simulator is able to simulate accurately the Failure Tolerance Mechanism described in the BitDew-MR environment, in Section 4.2.1. The experiment consists of the simulation of the MR execution in a BitDew-MR @ heterogeneous and volatile environment. Fifteen machines were used to process 8GB of data, 120 maps, 30 reduces, with a chunk size of 64 MB, and 20% of these machines are volatile nodes. Figure 4.6 shows the time period when the node is on-line or off-line. This is a real volatile behavior of Boinc traces, with volatile behavior displayed by three hosts (A, B and C), represented in the y-axis. The x-axis shows the time period in seconds when a host is on-line, in the gray box, and when it is off-line without a box.

The experiment consists of Host A and B (MRA_Host 1 and MRA_Host 2 respectively). Host A and B begin the execution and then stop during a time period, and Host C (MRA_Host 3) begins a late execution. BIGhybrid obtains these traces from a trace file and reproduces the volatile behavior, by comparing the average execution time with the trace profile. The users can define three variables related to a volatile environment in the configured files. One is *mra_dfs_replication* which defines the replication factor for the data; the other is *perc_num_volatile_node* which is related to the number of volatile nodes (as a percentage) and determines how many nodes that have a volatile behavior will be read from the traces file; and finally there is *failure-timeout* which determines the *period of failure-timeout* for the FTM as shown in Section 4.2.1. This time-out is defined in terms of *n* heartbeat periods.

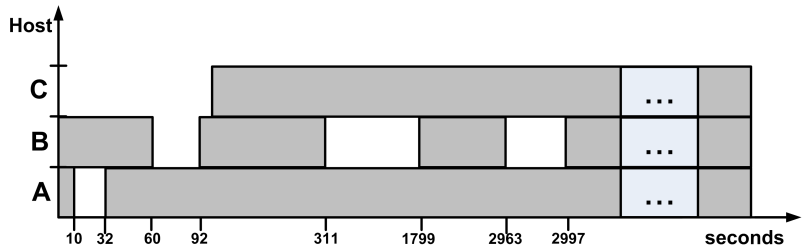


Figure 4.6: Time periods where a node is on-line in a volatile environment

Figure 4.7 shows the logs of the BIGhybrid simulator where this behavior is reproduced in the execution time. The MRA_Host 1 stops during a synchronization interval and the last valid heartbeat is considered to be the beginning of the period of failure-timeout, which in this case is 15 seconds and then the FTM system begins the failure recovery process. MRA_Host 2 stops before the synchronization interval when the next heartbeat detects the failure and, after the *period of failure-timeout*, the system begins the FTM recovery for *Map* and *Reduce* tasks. The BIGhybrid simulator also detects late nodes (those that begin the execution time after the other nodes); for instance, in MRA_Host 3, this host is available for execution. After that, it is necessary to move the data from the replica as a *map_task* 107 and 115 or to do a prefetching as in the case of *reduce_task* 0.

On the basis of these experiments, we concluded that the BIGhybrid simulator is able to simulate a MR execution accurately; in particular, it was confirmed that two distinct *barrier* and *barrier-free* features are correctly executed. As well as this, there was a failure recovery of the FTM in volatile environments.

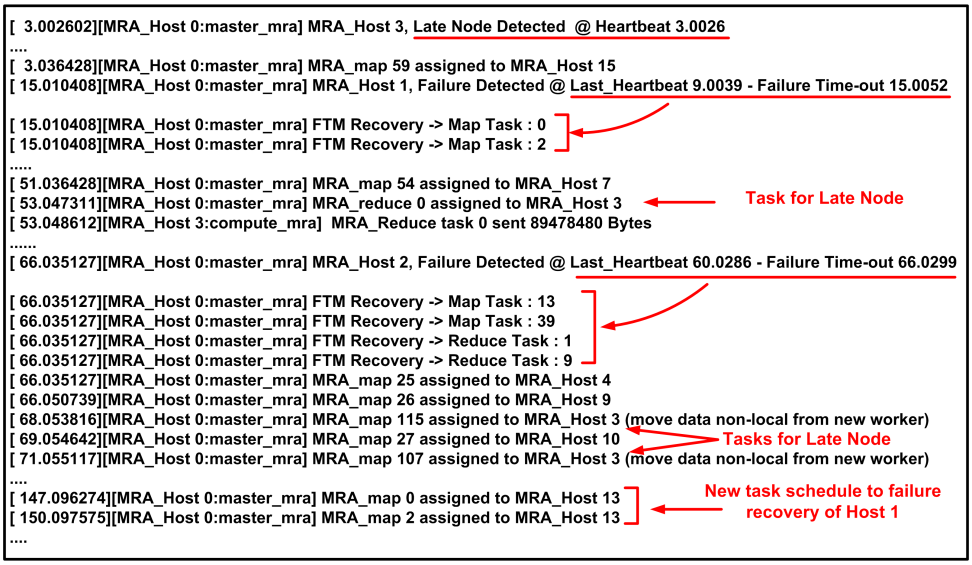


Figure 4.7: Logs of BIGhybrid simulator in execution time

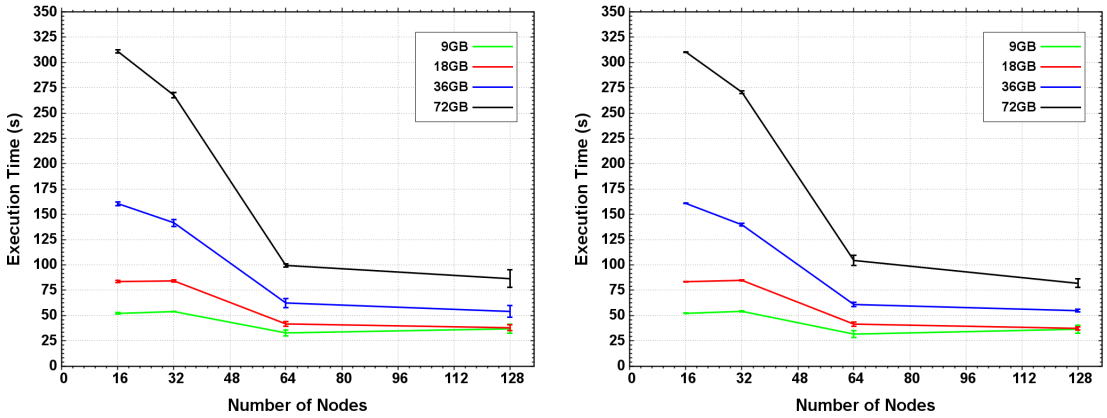
4.3.4 Study of the Behavior Profile in the Grid5000 Environment

In this experiment, our aim is to evaluate if the BIGhybrid simulator is able to simulate the *MapReduce* execution from Cloud-BlobSeer accurately in real environments with different workloads. The experiment consists of the execution in a homogeneous environment on Grid5000. In the case of Cloud-BlobSeer execution @ homogeneous environments, the experiment has 16, 32, 64 and 128 nodes, and the configuration of clusters is described in Table 4.3. In conducting this experiment, we seek to obtain the execution profile of the MR execution that is related to different workloads during the *Job*, *i.e.* the amount of time required for the execution of both the *Map* and *Reduce* phases.

The workload includes 9 GB - 141 chunks, 18 GB - 302 chunks, 36 GB - 571 chunks and 72 GB - 1143 chunks. Each Map processes 64 MB of data and produces an output of 42.85% of input data. The number of mappers and reducers is equal to the number of cluster nodes. The application is a function that calculates the average time when the nodes are free to execute a task. These times are available in a log file. The *Map* function reads the *id*, from each line of the input data. This *id* represents a node identification, which is related to the execution time. If the time is longer than 300 seconds, an intermediate key will be emitted with *id/time*. In the *Reduce* function, the average for each *id* is calculated and a new key *id/average* is emitted. Each experiment was executed 30 times and the result gave an average time. In the case of the BIGhybrid simulator, it is necessary to carry out a calibration procedure. This calibration is necessary to determine the simulation parameters, such as, task costs and network configuration, to create a platform with the same machine configuration as Grid5000, define the job configuration and so on. After this calibration, the number of machines needed to run the experiments is changed.

Figure 4.8 shows the Cloud-BlobSeer execution time in accordance with the number of nodes in the homogeneous environment. Figure 4.8.a shows the execution in Grid5000 and Figure 4.8.b shows the execution in a BIGhybrid simulator. The green, red, blue and black colors represent the workloads of 9GB, 18GB, 36GB and 72GB respectively. The execution time in the y-axis is measured in seconds, and the number of nodes in the x-axis is measured in units.

The BIGhybrid simulation has a similar execution time profile and a tendency to dis-



(a) G5000 environment (b) BIGhybrid simulator

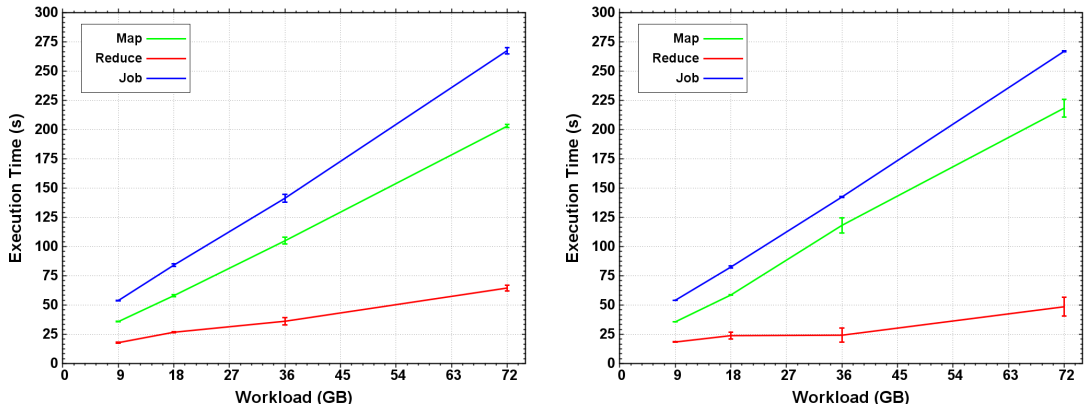
Figure 4.8: Cloud-BlobSeer execution time based on the number of nodes in the homogeneous environment

play the same behavior in the inclination curve for the *Job*. The execution on Grid5000, in Figure 4.8.a, has a standard deviation of 8.8% with a workload of 72 GB (more details are given in the statistical evaluation section - Section 4.4 -) that produces the highest data dispersion. The BIGhybrid simulation, in Figure 4.8.b, shows a simulation variation of $\approxeq 4\%$ for 72 and $\approxeq 5\%$ for 64 nodes, when compared with the Grid5000 execution, as a result of this data dispersion. This variation is a measure of absolute error in percentage terms (MAPE) and is calculated as MAPE = $\frac{1}{n}\sum\left|\frac{\overline{\mu}-X}{\overline{\mu}}\right| * 100$, where X is the measure of simulation, $\overline{\mu}$ is average of the measures and n is the sample size. This variation occurs because the simulation is an approximation of a real-world execution. This is acceptable when it is compared with the execution time in the Grid5000 environment (Figure 4.8.a). A statistical analysis is conducted in the next section that is based on these experiments.

In the next experiments, our aim is to evaluate if the BIGhybrid simulator is able to simulate the different phases of *MapReduce* execution accurately by means of Cloud-BlobSeer and BitDew-MR, in real environments with different workloads. The experiment involves executing two different applications in different infrastructures in homogeneous and heterogeneous environments in Grid5000. When conducting this experiment, it was necessary to evaluate if the execution time for *Map* and *Reduce* phases, (and the total execution time for the *Job*) requires the same amount of time as the Grid5000 and BIGhybrid simulator and take account of homogeneous and heterogeneous environments. The Cloud-BlobSeer execution @ homogeneous environments uses 32 nodes and the cluster configuration is described in Table 4.3. In the BitDew-MR execution @ heterogeneous environments 50 nodes as used for the workers and 2 nodes for the BitDew-MR services; this cluster is described in Table 4.4. The workload is 9 GB - 141 chunks, 18 GB - 302 chunks, 36 GB - 571 chunks and 72 GB - 1143 chunks, and the chunk size is of 64 MB for Cloud-BlobSeer. In BitDew-MR the workload is 3 GB - 192 chunks, 13 GB - 768 chunks, 26 GB -1536 chunks, and the chunk size is of 16 MB. The application for Cloud-BlobSeer was described in the previous experiment. The application in BitDew-MR execution is wordcount. Wordcount is a popular micro-benchmark that is widely used in the community, contained in the Hadoop distribution (HUANG et al., 2010). Each experiment was executed 30 times and the result is an average time.

Figure 4.9 and Figure 4.10 show the Cloud-BlobSeer and BitDew-MR execution times respectively. The green, red and blue colors represent the *Map* and *Reduce* phases and *Job*

respectively. The execution time in the y-axis is measured in seconds and the workload in the x-axis is measured in gigabytes (GB). Figure 4.9 shows the average execution time for *Map*, *Reduce* and *Job* executions for Cloud-BlobSeer @ homogeneous environments.



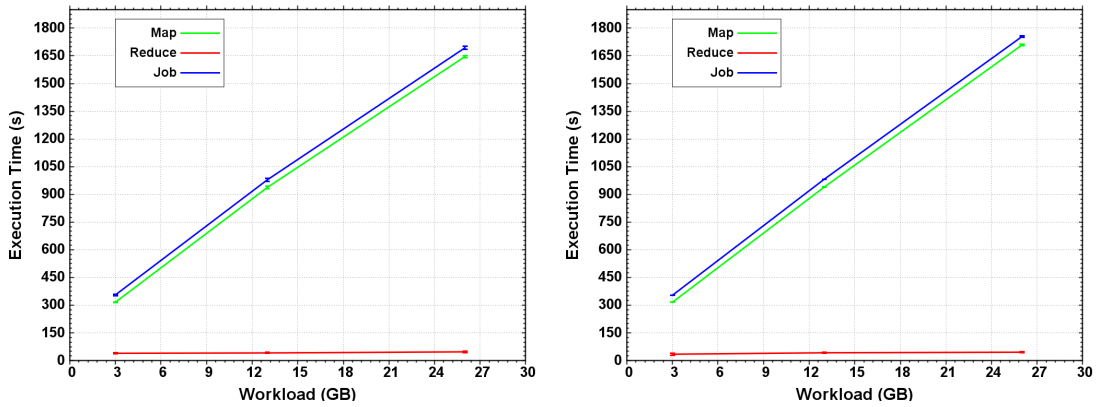
(a) G5000 environments - 32 nodes (b) BIGhybrid simulator - 32 nodes

Figure 4.9: Cloud-BlobSeer execution time @ homogeneous environments

Figure 4.9.b shows the simulation carried out by the Cloud-BlobSeer in BIGhybrid simulator. The behavior simulated is similar, but has a slight distortion in the *Reduce* phase execution that generates a MAPE of 0.58% in the worst case scenario. The distortion has little effect because the *Map* execution time is more significant than the *Reduce* execution time. The *Job* has ≅ 54 s @ 9 GB and ≅ 267 @ 72 GB, with a MAPE of 0.44% and 0.24% respectively. The *Map* phase has a greater weight in terms of number of tasks in the *Job* time, than the *Reduce* phase and, as a result, the distortion of *Reduce* time execution is reduced for the total time required for *Job*. The execution time is a reasonable approximation, if we consider the *Job* total time.

Figure 4.10 shows the execution time for the *Map*, *Reduce* and *Job* executions for BitDew-MR @ heterogeneous environment. Figure 4.10.a shows the average execution in Grid5000. The BitDew-MR execution processes the *Combine* function, that involves processing similar keys in the *Map* phase before the data is sent to the next phase. It is not possible to determine if a data partition will have more or fewer keys to join in the *Combine* function. The simulator has the highest execution time for the *Map* function to simulate this additional work, but it is not an approximation that is easy to define. The execution profile shows that the *Map* spends more execution time on this application and Figure 4.10.b shows that BitDew-MR simulation follows the same procedure and has a relative degree of precision in the BIGhybrid simulator. *Job* has ≅ 353 s @ 3 GB and ≅ 1754 @ 26 GB, with MAPE of 0.58% and 3.59% respectively; more details about *Reduce* are shown in Table 4.8, in Section 4.4. The *Map* phase has a greater weight in terms of task number and execution time (in the *Job* time), than the *Reduce* phase and because of this the *Job* total time is close to the *Map* time.

As a result of these experiments, we estimated that the BIGhybrid simulator is able to simulate the different phases of *MapReduce* execution from Cloud-BlobSeer and BitDew-MR. In Section 4.4, the experiments of Figure 4.8 and Figure 4.10, Cloud-BlobSeer and BitDew-MR executions, respectively, are analyzed from a statistical perspective.



(a) G5000 environments - 50 hosts

(b) BIGhybrid Simulator - 50 hosts

Figure 4.10: BitDew-MR execution time @ heterogeneous environments

4.4 Statistical Evaluation of the BIGhybrid Simulator

In this experiment, our aim is to evaluate the accuracy of the BIGhybrid simulator for the Cloud-BlobSeer execution. Table 4.6 shows the Cloud-BlobSeer execution time in accordance with the number of nodes in the homogeneous environment (as shown in Figure 4.8), which is described in the previous section. The measurements in the table are based on the Grid5000 and BIGhybrid experiments. These measurements are grouped in accordance with the node numbers, execution type and the workloads of 9GB, 18GB, 36GB and 72GB. First, in the Grid5000 execution, the average ($\overline{\mu}$) is the mean of the measured results in Grid5000. The standard deviation (σ) of these measures is $\sigma = \sqrt{\frac{\sum(x-\overline{\mu})^2}{n}}$, where x is the measure and n is the sample size. The coefficient of variation (CV) is the ratio of standard deviation to the average (in percentage terms). This value is a standardized measurement for analyzing the average dispersion of all measures together. The value is calculated as $CV = \frac{\sigma}{\overline{\mu}} * 100$. The T_{val} is a confidence interval for the standard deviation measures, in this case equal to 95%. Second, in the BIGhybrid execution, the measure for the simulation is printed in DUT (Device under test). The mean absolute percentage error (MAPE) is calculated as MAPE = $\frac{1}{n}\sum|\frac{\overline{\mu}-X}{\overline{\mu}}| * 100$, where X is the measure of simulation and MAPE is an accurate measurement that indicates how near a sample is to the average; if the value is low, it indicates a good degree of accuracy (MAKRIDAKIS, 1978).

The BIGhybrid execution measurements (DUT) for the Blobseer-Hadoop simulation shows a MAPE measure of 4.92% in the worst case scenario, which is very acceptable. A slight distortion (with 64 and 128 nodes and 72GB workload) is shown by a higher MAPE variation. One reason for this is network contention that is caused by the network being shared with other users, but this is a common resource share, as in the Internet. Otherwise, the simulator does not capture this variation. Better MAPE results are obtained in BIGhybrid experiments when the real-world experiments have a low standard deviation. This is because when there are only a few machines, there are more tasks to execute locally and the execution has less data movement. The BIGhybrid is a deterministic simulator and simulates a data locality feature that is reflected in lower variations for MAPE. Most executions of the BIGhybrid simulator are carried out within the standard deviation confidence interval of 95%. Hence, the results of the BIGhybrid simulator have a good rate of accuracy for executions of Blobseer-Hadoop in homogeneous environments.

It is necessary to evaluate if the simulations have a similar distribution to that found in

Table 4.6: Cloud-BlobSeer execution in a homogeneous environment

	Grid5000 measures					BIGhybrid measures	
# Nodes	# Input	$\overline{\mu}$	σ	CV(%)	T_{val}	DUT	MAPE(%)
16	9GB	52.05	0.82557	1.59612	51.66 - 52.44	52.15	0.19
	18GB	83.55	1.09904	1.31543	83.04 - 84.06	83.41	0.16
	36GB	160.55	1.93241	1.20361	159.65 - 161.45	160.77	0.17
	72GB	310.85	1.53125	0.49260	310.13 - 311.57	310.36	0.16
32	9GB	53.9	0.32544	0.61037	53.77 - 54.03	54.14	0.44
	18GB	84.23	0.86389	1.02559	83.90 - 84.56	84.80	0.67
	36GB	141.64	3.42900	2.42085	140.26 - 143.03	142.47	0.59
	72GB	267.67	2.68093	1.00159	266.58 - 268.75	267.01	0.24
64	9GB	32.83	2.98771	9.10130	40.81 - 42.58	31.65	3.59
	18GB	41.69	2.18752	5.24664	40.81 - 42.58	41.51	0.43
	36GB	62.39	4.45377	7.13804	60.70 - 64.09	60.97	2.27
	72GB	99.49	1.52223	1.53000	98.83 - 100.15	104.39	4.92
128	9GB	36.98	4.63255	12.53275	35.22 - 38.74	38.49	4.08
	18GB	37.95	2.87843	7.58405	36.64 - 39.26	37.33	1.63
	36GB	54.	5.80187	10.74092	51.51 - 56.52	54.78	1.41
	72GB	85.39	8.87972	10.39907	81.55 - 89.23	81.79	4.22

a real-world execution. This relation between the Grid5000 and BIGhybrid experiments is analyzed in accordance with the correlation coefficient ($\text{Corr}_{x,y}$) (JAIN, 1991). The $\text{Corr}_{x,y}$ is calculated by the Pearson Method, as in Equation 4.4.1, where "x" is the BIGhybrid measure, "y" is the Grid5000 measure and "n" is the number of measures. The confidence interval (T_{val}) for this analysis is equal to 95%. This coefficient is calculated on the basis of the Equation 4.4.1 and in Table 4.6, and the results are set out in Table 4.7. Figure 4.11 shows the dispersion chart based on the results from Table 4.7.

$$r = \frac{n\sum xy - (\sum x \cdot \sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}} \tag{4.4.1}$$

The measures of standard deviation and coefficient of variation, in Table 4.6, demonstrate that there is a little dispersion around the average. This is proved by the fact that there is a high correlation coefficient between the Grid5000 execution and the BIGhybrid simulation, in Table 4.7 (near to 99%). The analysis of dispersion (in the diagram) for 16, 32, 64 and 128 nodes, in Figure 4.11, shows a positive correlation. The red line represents the linear regression which is obtained from the calculation of the correlation. On the basis of these observations, it can be concluded that the BIGhybrid simulations achieve a good approximation to the MapReduce executions in hybrid environments for the Blobseer-Hadoop execution. This means that the BIGhybrid simulation can be used to simulate a hybrid environment and evaluate the likely behavior of the nodes in these

environments.

Table 4.7: Grid5000 vs BIGhybrid - correlation coefficient for Blobseer-Hadoop

# Nodes	$\text{Corr}_{x,y}$	T_{val}
16	0.9999981	0.9999064 - 1.0000000
32	0.9979155	0.9000929 - 0.9999586
64	0.9989227	0.9471165 - 0.9999786
128	0.9983309	0.9192125 - 0.9999669

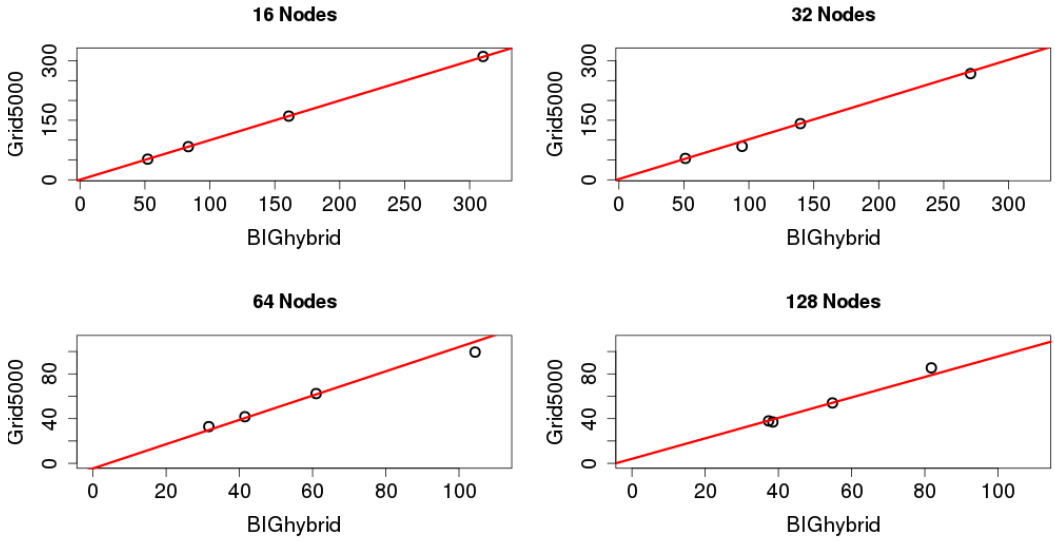


Figure 4.11: Dispersion chart - Grid5000 vs. BIGhybrid - to Blobseer-Hadoop

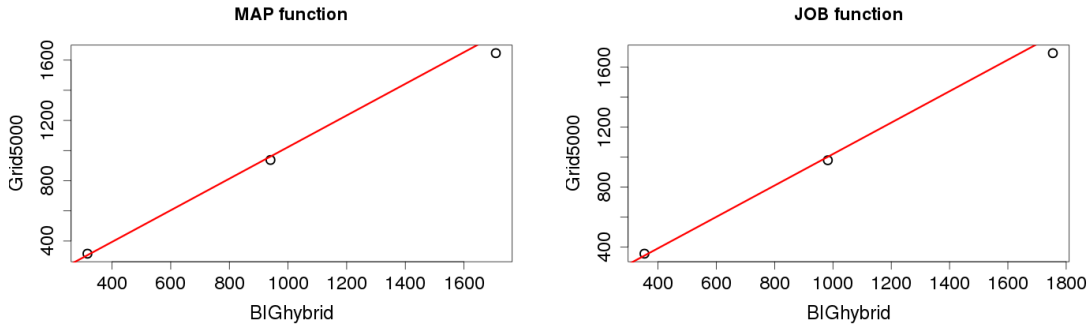
In the next analysis, our aim is to measure the accuracy of the BIGhybrid simulator for the BitDew-MR execution. The experiment was described in the previous section. The data in Figure4.10 for the BitDew-MR @ heterogeneous environment are set out in Table 4.8. The Table is formed with the aid of Grid5000 and BIGhybrid measures. These measures are grouped in accordance with their "function type" and workload input. The mean ($\overline{\mu}$) is the average of the measured results in the Grid5000 execution. The standard deviation (σ), coefficient of variation (CV), confidence interval T_{val} and the mean are evaluated in accordance with the *Map, Reduce* and *Job* executions. The BIGhybrid execution is the DUT measure and for each measure, it is linked to a mean absolute percentage error (MAPE), that is calculated as described earlier.

The execution in a real-world environment has a high dispersion rate for the *Reduce* function. It is related to the size of data transfers from intermediate data. This size depends on the generated partitions in the *Map* phase that can change as a result of the input data received. The bandwidth fluctuation is another factor to consider, because it has a bearing on time data transfers. However, the execution time takes up more that 80% of the *Map* function, and as a result, the dispersion effect is reduced. The DUT values within the standard deviation confidence interval show that the BIGhybrid simulation has a good degree of accuracy. The MAPE measure of 4.96% in the worst case scenario is very acceptable too, and the measures for *Map* have a maximum of 3.81%.

Table 4.8: BitDew-MR execution in the heterogeneous environment

		Grid5000 measures				BIGhybrid measures	
# Nodes	# Input	$\overline{\mu}$	σ	CV(%)	T_{val}	DUT	MAPE(%)
3GB	MAP	316.12	1.59708	0.50521	314.98 - 317.26	316.00	0.04
	REDUCE	39.48	3.02096	7.65188	37.32 - 41.64	37.52	4.96
	JOB	355.60	3.42085	0.96199	353.15 - 358.05	353.52	0.58
13GB	MAP	937.65	7.77135	0.82881	932.09 - 943.21	940.23	0.28
	REDUCE	41.39	2.80335	6.77301	39.38 - 43.39	42.30	2.20
	JOB	979.04	7.82037	0.79878	973.45 - 984.63	982.53	0.05
26GB	MAP	1,646.06	5.19790	0.31578	1,642.34 - 1,649.78	1,708.80	3.81
	REDUCE	47.35	4.80769	10.15352	43.91 - 50.79	45.44	4.03
	JOB	1,693.41	7.60722	0.44922	1,687.97 - 1,698.85	1,754.24	3.59

The correlation coefficient between the Grid5000 and BIGhybrid executions for BitDew-MR @ heterogeneous environment is 99.9% for the *Job*. The dispersion chart in Figure 4.12 shows that the dispersion has a positive correlation. The red line represents the linear regression which is obtained from the calculation of the correlation. The chart demonstrates that the values are near this line which suggests there is a good approximation to the simulation.



(a) Map Function - 50 hosts

(b) Job Function - 50 hosts

Figure 4.12: Dispersion chart - Grid5000 vs. BIGhybrid to BitDew-MR

The statistical evaluation for Cloud-BlobSeer and BitDew-MR simulations provides a simulation with a relative degree of accuracy. The mean absolute percentage error ($\approxeq$ 5% in the worst case scenario for heterogeneous and homogeneous environments) shows that the simulator can be an efficient evaluative instrument for *hybrid infrastructures*. The high correlation coefficient between the Grid5000 execution and the BIGhybrid simulation (around 99%) indicates that the real-world behavior is reproduced by the BIGhybrid simulator. It can thus be concluded that the simulator has a reproducible capability and is able to achieve a relative degree of accuracy in real-world experiments.

4.4.1 A Study of the Reproducibility of Real Experiments

In this experiment, our aim is to evaluate if the BIGhybrid simulator is able to reproduce the results obtained from the synthetic applications from real-world experiments. The experiment consists of simulating Cloud-BlobSeer from data collected from homogeneous executions carried out in a Yahoo cluster. In conducting this experiment, it was necessary to obtain the execution time for the *Map* and *Reduce* phases and to compare this with the times from Table 4.5 that consider a homogeneous environment in Cloud applications. With regard to the Cloud-BlobSeer execution @ homogeneous environments, 2000 nodes were used (as described in Section 4.3.1). The "*aggregated fast job*" applications characterized by Chen were taken into account. The workload has 568 GB of input and 9,088 tasks, and each *Job* has an execution time of 322.64 seconds from *Map* and 703.32 seconds from *Reduce*. The number of mappers is 2,000 and that of reducers is 1,000.

Figure 4.13 shows the Bighydrid simulation of the MapReduce execution from 2000 hosts @ Yahoo traces. The red, blue and green colors represent *Map*, *Reduce* and *Shuffle* phases respectively. In the y-axis, the number of concurrent tasks is measured in units and the time of the x-axis is measured in seconds. The number of *Map* and *Reduce* tasks is restricted to two task per node, *i.e.*, 4,000 tasks for the *Map* phase and 2,000 tasks for the *Reduce* phase. The execution time is 305.13 s for the *Map* phase and 673.32 s for the *Reduce* phase, and the simulation error is $\approxeq 5\%$ with regard to one *Job* in Table 4.5. This error is a calculation of $\text{ERROR} = \left|\frac{\vartheta - X}{\vartheta}\right| * 100$, where ϑ is the execution time of *aggregated fast job* executions and X is the measured time of the simulator.

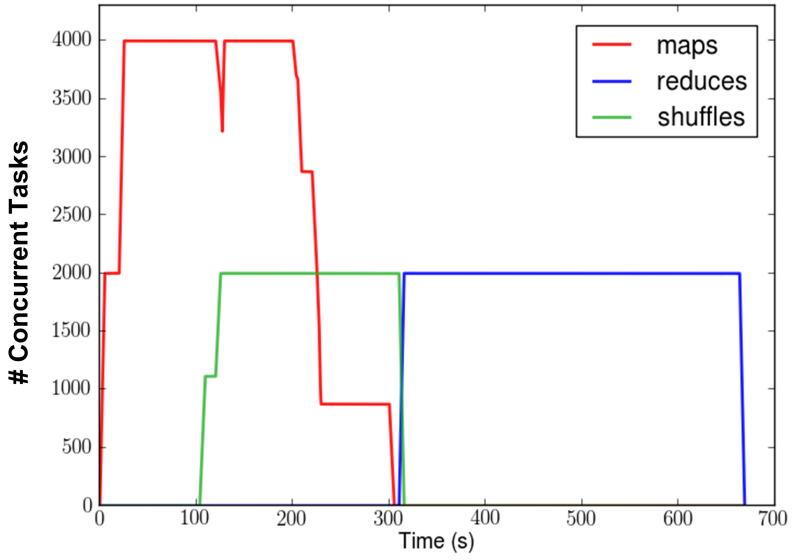


Figure 4.13: Simulation of MapReduce execution from 2000 hosts @ Yahoo traces

As a result of this experiment, we concluded that the BIGhybrid simulator is able to reproduce experiments from synthetic applications of real-world experiments. This shows the scalability of the simulator, its reproducible capabilities and its suitability for investigative research into new strategies based on a *hybrid infrastructure*.

4.5 Final Considerations

The rapid increase in the amount of data currently being produced will stretch the current infrastructure to its limits. Merging Cloud and DG into a hybrid infrastructure

might be a feasible low-cost alternative to simply using Cloud environments as a result of the cost-free DG resources available.

The characteristics of a hybrid infrastructure were introduced and the feasibility of integrating Cloud and DG was demonstrated by carrying out simulations to define the best strategies for the implementation. The experiments evaluated three different studies. In the study of the simulated *MapReduce* execution, the experiments showed that the BIGhybrid simulator is able to simulate a MR execution accurately. It was found that two distinct *barrier* and *barrier-free* features were correctly executed. As well as this, the implementation of the failure recovery behavior of the fault tolerance mechanism was carried out in volatile environments. In the study of the behavior profile in the Grid5000 environment, it was shown that the BIGhybrid simulator is able to simulate the different phases of the *MapReduce* execution from BlobSeer-Hadoop and BitDew-MR when this is compared with real environments and different workloads. In the study of the reproducibility of the real-world experiments, the BIGhybrid simulator is able to reproduce experiments involving synthetic applications from real-world experiments. This demonstrates the scalability of the simulator, its reproducible capability and its suitability for investigative research into new strategies to evaluate the implementation of a hybrid infrastructure with *MapReduce* applications.

The experiments and statistical evaluation proved that the simulator has a reproducible capability based on real-world experiments and provides evidence that the validation goals have been achieved. This means that, the BIGhybrid simulator makes it possible to evaluate MapReduce strategies that involve the adoption of hybrid infrastructures. It also suggests that it is possible to overcome problems through the adoption of determined strategies with a relative degree of accuracy. It is necessary to conduct further experiments to define what strategies are required for the adoption of hybrid infrastructures with the results obtained from real-world systems.

5 EVALUATION AND RESULTS

The adaptation of an existing MR framework or the development of new software for hybrid infrastructures raises some research questions: how to adopt efficient strategies for data splitting and distribution, how to keep communication between the infrastructures to a minimum, and how to deal with failures, sabotage, and data privacy. This Chapter shows the strategies that will be adopted for the deploying of SMART platform and their modules, in particular, the Global Dispatcher. Also, there will be an analysis of the use and feasibility of homogeneous and volatile environments in the SMART platform in the context of Big Data applications. The strategies, such as load balance control, resource allocation, volatile node range, cost model and bandwidth between nodes can establish some borders for use of hybrid environments.

5.1 Strategies for the Use and Evaluation of Hybrid Infrastructures

This Section sets out strategies for the use of the hybrid infrastructures that have a conceptual basis and emerged from "Related Work" which can be adapted to the context of a hybrid infrastructure. These strategies are evaluated in Section 5.3. The hybrid infrastructures have several features that are related to distinct resources. The user must launch its applications in an infrastructure where herself must choose the best resources from among two or more computational environments. In our case, for the hybrid infrastructure, the user must choose an amount of resources for Cloud and Volunteer Computing (VC).

A CSP offers resources that represent VM instances (i) of a hardware $H_{VM}(i)$, where $i \in VM$. Each VM represents a set of heterogeneous resources that are related to a number of CPU cores, memory, and storage. Each resource, called r_c, represents a VM and the total of the available resources represent a vector $\hat{R}_c(n_r) = \{r_{c_1}, r_{c_2}, ..., r_{c_n}\}$ where n_r is the number of available resources, *i.e.*, $n_r = [1, 2, ..., n]$ resources. The resource allocation defines the computational capacity in Flops/s of the machines set. The resources are homogeneous in user point view because the CSP enables only access to the virtual instances of machines. However, the hardware of the CSPs can be heterogeneous and the user, in general, does not have knowledge. Therefore, an allocated resource may have low performance compared with a single hardware component, when two or more users are running their applications on virtual machines over the same leased hardware. In the SMART, the user can allocate resources from different CSPs. As each resource can have an abstract specification different, the user will be able to know which resource is faster, through simulation environment, before choosing the resource. The performance can have a different time execution rate for each VM, called ρ, that is caused by the hardware contention. Its value can be between an interval of $(0, 1)$. Thus, the resource allocation ($A_{Vm} \subseteq \hat{R}_c$) of a

user (u) in Cloud is related to the Equation 5.1.1, where $n \leq n_r$.

$$A_{Vm}(u) = \sum_{i=1}^{n}(1-\rho_i)r_{ci} \tag{5.1.1}$$

In VC, a donor offers a hardware $H_l(t)$ that will be dedicated to the processing of tasks for an unspecified time period (t), where the number of offers to a volunteer resource pool are defined by $l = [1,2,3,....,m]$ resources. Any free hardware is related to heterogeneous resources that represent a number of CPU cores, memory, and storage available within a time period. The pool is a set of stable and unstable nodes. The volunteer resources are a vector $\hat{R}_v(H_l(t)) = \{h_1t_1, h_2t_2, ..., h_mt_k\}$ where t_k is the time availability in each machine for the users utilize a free hardware resource in the resource pool. The user cannot define an availability time for a resource because this issue is internally handled for the user by a fault-tolerance mechanism of the platform in a transparent way. However, the user can define a variable ξ which determines the number of unstable nodes in the pool. The variable ξ can vary between an interval $(0,1)$. The VC maintains connections through the Internet. In addition, data transference will depend on the bandwidth β_l between these nodes and the total bandwidth β_{ch} available on the CSP channel. The allocation of VC resources ($A_{VC} \subseteq \hat{R}_v$) of a user ($u$) is represented by Equation 5.1.2.

$$A_{VC}(u) = \sum_{k=1}^{m}(1-\xi)h_k \tag{5.1.2}$$

The performance model has a component that represents the Cloud Allocation with stable VMs and a VC allocation that depends on a rate corresponding to ξ that defines the unstable nodes ratio and can be influenced by the available bandwidth ratio. This model shows the effects of the job execution on the data-intensive applications. The best choice of platform characteristics and data distribution in accordance with the environment are crucial factors that are required to reduce these effects. Thus, the performance of an allocation $P(A_u)$ of a user u is defined in accordance with Equation 5.1.3.

$$P(A_u) = \underbrace{\sum_{i=1}^{n}(1-\rho_i)r_{ci}}_{\text{Cloud allocation}} + \underbrace{\sum_{k=1}^{m}(1-\xi)h_k}_{\text{VC allocation}} \tag{5.1.3}$$

The bandwidth affects the performance of data transference in each node over the VC environments. Javadi (JAVADI; ZHANG; TAUFER, 2014) has validated a bandwidth modeling for characterizing the correlation between upload and download in Big Data applications based on the Internet through Equation 5.1.4. The model regards the jobs as the incoming requests. Each host provides a service with a variable service time and the model makes it possible to analyze the bandwidth requirements for the Big Data applications for a hybrid infrastructure. The values for indexes a and b of Equation 5.1.4 were defined by Javadi in Table 5.1 and refer to job runtime with a normal distribution of $\mu = 1.2$hs and $\sigma = 0.9$hs. These parameters are related to mean time of jobs. However, this equation must take account of the bandwidth saturation.

$$f(t) = ae^{bt} \tag{5.1.4}$$

The Big Data algorithms, in SMART implementation, evaluate the bandwidth available for each host in the heterogeneous environments. Thus, when the scheduler instantiates a task, it identifies if is better to move data from one host to another faster for improving performance. Because of this, the cluster configurations can move more or fewer data with the same hardware configuration.

Table 5.1: Parameters for the fitted exponential model

Model	a	b
Mean download	5.698	$0.493e^{-3}$
Mean upload	0.1955	$0.404e^{-3}$

A hybrid infrastructure scenario in this work consists of five key components: workers, orchestrators, dispatcher, aggregators and CSPs. Figure 5.1 represents these components and a data flow for this scenario. The workers are grouped according to their computational capacity to integrate a DG that provides a pool of computational resources for a user in a private mode. This private mode means that no other user can employ the allocated resources until the user himself releases them to the resource pool. A CSP offers their cloud resources to users that must be allocated in accordance with an SLA. An orchestrator (built in the dispatcher module) receives data which is assigned as user tasks and transformed into a "job". Then, the dispatcher creates partitions of data and tasks based on computational capacities and assigns them to DG and Cloud. In a hybrid environment, each system processes its own data by grouping the intermediate data in a last *Reduce* phase within the Cloud to produce a single result in the aggregation stage. The *Shuffle* occurs in parallel with the *Map* phase. This begins when the *Map* tasks that have been completed achieve at least 5% (a Hadoop standard), assuming that the *Reduce* execution begins after all the *Map* tasks have been finished in the case of Cloud.

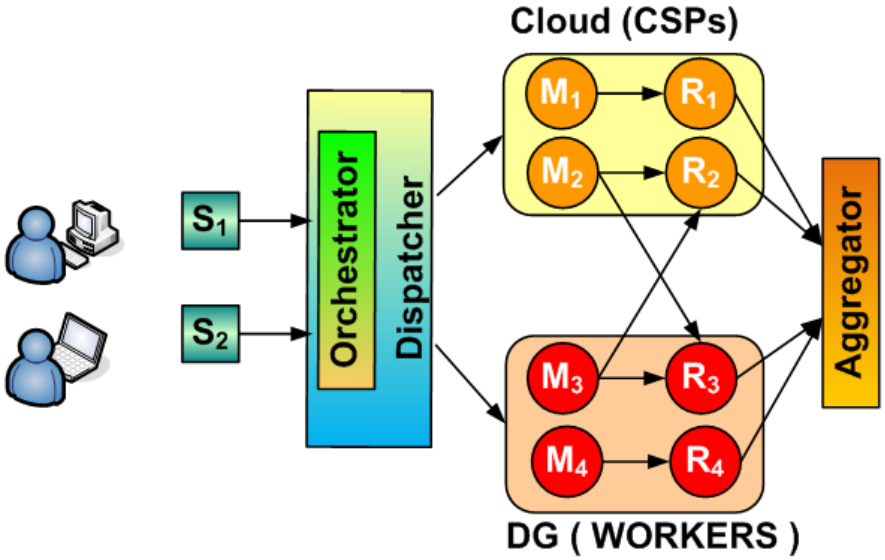


Figure 5.1: An example of the data flows in the hybrid environment

The time T_t (in Equation 5.1.5) of a *MapReduce* job in a hybrid environment (inspired on the equations of (OHNAGA; AIDA; ABDUL-RAHMAN, 2015) and (KHAN et al., 2016)) consists of maximum value between the execution time in Cloud (T_c) and DG (T_d) added to the time of the data integration (called T_i). The time T_i includes the transfers

of intermediate data (T_ε) from the DG to Cloud through an Internet link, and the time of the additional *Reduce* phase to produce a single *Reduce*. Our consideration is different from other approaches because it includes the integration time to produce a single and consistent result to Big Data processing.

$$T_t = \max(T_c, T_d) + T_i \tag{5.1.5}$$

The user submits a job (J) that will execute a workload across a set of selected resources (S) from the all available resources in Cloud and DG (R) (where $S \subseteq R$). When occurs a job launch pending tasks the system scheduling allocates one free slot per time in each machine under all resource pool, then after will allocate the other pending tasks in one free slot in each machine and so on. This process originates execution waves (w_j) which are limited to the maximum number of allocated resources since the workload is larger than the available computation. These waves are rounds of successive executions $G = \{1,2,3,4,5,...,n\}$ that continue until all the n tasks have been completed. The execution waves are defined as follows $w_j \subseteq G \subseteq J$. The concurrent tasks define execution spaces, called slots, in each machine and the number of free slots determines the amount of parallel work. As discussed previously, the *Shuffle* phase occurs concurrently with *Map* phase if there is more than one execution wave. Thus, the time spent to data transfer (called T_S) must be reduced of the max time of the last *Map* phase in the last wave. Therefore, the sum of the max time for the *Map* and *Reduce* phase in each wave are added to the T_S time.

Equation 5.1.5 summarizes these times on the basis of Equation 5.1.6 to 5.1.10 , where "τ" represents the number of the free slots for *Map* (M) or *Reduce* (R) in Cloud (c) or DG (d). Each machine processes an input data size (W_M in Bytes) in all the free slots in the *Map* phase. This number of free slots is related to the core features of CPU for the whole cluster. In addition, the *Combiner* function (like in Hadoop) does a preprocessing of intermediate keys, in the memory of the worker, before sending the data to the next stage. The *Shuffle* time (T_S) is related to the network bandwidth (B_l) between workers and the weight (K_i) of the *Map* output size. A computational capacity function (called $f_{C_{cap}}$) for each machine (W_i) is based on features, such as CPU, core number, memory and workload measured in Flops/Bytes. In our case, the function $f_{C_{cap}}$ is an empirical value measured as a computational cost for the *Map* and *Reduce* tasks in the simulator.

$$T_c = \underbrace{\sum_{i=1}^{A_{Vm}} \max_{w_i \in J} \left(\frac{W_M(i)}{\tau_c} f_{C_{cap}}(i) \right)^{-1}}_{\max Map \text{ time}} + T_{S_c} + \underbrace{\sum_{i=1}^{A_{Vm}} \max_{w_i \in J} \left(\frac{W_R(i)}{\tau_c} f_{C_{cap}}(i) \right)^{-1}}_{\max \ Reduce \text{ time}} \tag{5.1.6}$$

$$T_{S_c} = \int_{t_{lw-1}}^{t_{lw}} \frac{K_i}{B_l a e^{bt}} dt - \underbrace{\max_{lw \in J} \left(\frac{W_M(i)}{\tau_c} f_{C_{cap}}(i) \right)^{-1}}_{\max Map \text{ time in the last wave}} \tag{5.1.7}$$

$$T_d = \underbrace{\sum_{i=1}^{A_{VC}} \max_{w_i \in J} \left(\frac{W_M(i)}{\tau_d} f_{C_{cap}}(i) \right)^{-1}}_{\max Map \text{ time}} + T_{S_d} + \underbrace{\sum_{i=1}^{A_{VC}} \max_{w_i \in J} \left(\frac{W_R(i)}{\tau_d} f_{C_{cap}}(i) \right)^{-1}}_{\max \ Reduce \text{ time}} \tag{5.1.8}$$

$$T_{S_d} = \int_{t_{lw-1}}^{t_{lw}} \frac{K_i}{B_l a e^{bt}} dt - \underbrace{\max_{lw \in J} \left(\frac{W_M(i)}{\tau_d} f_{C_{cap}}(i) \right)^{-1}}_{\max Map \text{ time in the last wave}} \tag{5.1.9}$$

$$T_i = \underbrace{\frac{T_k^{Rc}}{\tau_{Rc}\eta_{Rc}} R(k)}_{T_\varepsilon} + \sum_{i=1}^{l} \frac{K_i}{B_l a e^{bt}} \tag{5.1.10}$$

The strategy of data distribution, in the Global Dispatcher module from SMART architecture, seeks to maximize the use of the DG resources, at the same time that minimizes the Cloud allocation resources without increasing the execution time in comparison with a single Cloud implementation. The data-split will consider the execution waves in the dispatcher to achieve this aim.

There are three types of storage technology (*Ephemeral, Persistent* and *Cloud storage*) that are linked to the allocation cost of a VM in Cloud (TUDORAN et al., 2014). The storage cost is only built into the VM in the *Ephemeral storage*, where the data only persist during the lifetime of the instances. *Persistent storage* is a network-attached device in block-level, where the data lifetime is independent of a VM instance and can be reused. The *Cloud storage* is a BLOB with a higher price and offers a service level as redundancy and privileged access time.

The Cloud resources model incurs a cost caused by the leasing of the number of instances found in Equation 2.4.3 (PALANISAMY; SINGH; LIU, 2015). Thus, the cost of submitting a job in Cloud $(J_i, C^{k,n})$ is defined as in Equation 5.1.11, where the index k represents the instance type (a price characteristic) and i is the job identification. The index β represents the total number of allocated resources. This cost depends on time duration (t_r) of $(J_i, C^{k,n})$ job related to the resources number (called n), and the price of the lease of a VM instance (R_{c_i}) in a determined number of physical resources (M_{c_i}) like CPU, memory and disk. On the other hand, the VC is free-cost. Thus, the strategy cost minimizes the Cloud allocation without increase the time execution improves the relation cost-benefit.

$$C(J_i, C^{k,n}) = t_r * (J_i, C^{k,n}) * \sum_{m=1}^{\beta} \frac{n * R_{c_m}}{M_{c_m}} \tag{5.1.11}$$

Without generalization loss, each machine represents a single VM deployed on each experiment. The disk type consists of a temporary storage in each VM, and additional costs like I/O contention on disks were not computed in these experiments. Thus, the cost model of Equation 5.1.11 can be simplified as follows, in Equation 5.1.12.

$$C(J_i, C^{k,n}) = t_r * (J_i, C^{k,n}) * \sum_{m=1}^{\beta} n * R_{c_m} \tag{5.1.12}$$

The work of (KONDO et al., 2009) discovered that the ratio of volunteer nodes needed to achieve the computing power of a small EC2 instance is about 2.83 active volunteer

hosts to 1. Thus, if they wish to achieve the best resource allocation for Desktop Grid volunteer and Cloud scenarios, the users must choose the resources on the basis of with data size, task numbers, low cost, the lowest possible time execution, minimizing data transference between sites, and minimizing the effects of parameters ρ and ξ. However, it is a hard task for users to conform to the definitions of these configurations.

Therefore, the BIGhybrid simulator can become a powerful tool to establish strategies for hybrid infrastructures. Table 5.2 summarizes the main strategies (on the basis of previous information) that allow the use of VC and Cloud in data-intensive applications in a realistic setting minimizing data transference.

Table 5.2: Strategies and targets

Parameters	Strategies	Targets
Volunteer hosts	Investigate a minimal ξ possible	Finding a relation between execution time and volatile nodes.
Cloud resources	Investigate the resource distribution	Finding a relation between the number of Cloud and VC resources.
I/O interference	Define an optimal ρ	Search for a relation between performance and interference in hybrid infrastructures.
Channel communication	Investigate data size	Conducting an analysis of bandwidth between nodes vs. Equation 5.1.4 to define a chunk size.
Data load balance	Evaluate the $P(A_u)$	Finding a relation of data distribution related to the number of Cloud and VC resources.
Cost model	Evaluate the $(J_i, C^{k,n})$	Seeking a relationship between Cloud and VC resource number, and cost model in Equation 5.1.12.

These strategies make it possible to find thresholds that make SMART platform feasible for using in hybrid infrastructures in a data-intensive scenario. These thresholds are based on the evaluation of parameters, such as the number of volunteer hosts, load balance control and cost model, in the context of Big Data synthetic applications. However, the evaluation of I/O interference was not possible to execute at this moment due modifications on disk structure of the BIGhybrid simulator, which was not implemented yet.

5.2 Methodology employed to Evaluation

This Section has the aims of defining the methodology used in the experiments and justify the setup definitions. The strategies in Table 5.2 were evaluated through an analysis of the workloads produced in the work of (CHEN et al., 2011) and (CHEN, 2012) which are summarized in Table 5.4. These workloads are real-world executions that have their origin in Big Data applications of enterprises like Yahoo and Facebook. The YH and FB clusters have 2,000 and 3,000 machines respectively. Although the job types at Facebook can change significantly from one year to another, the purpose of this scenario is to approach real-world environments. The simulation parameters are divided in two groups: one to homogeneous environment and other to heterogeneous environment summarized in Table 5.3.

The first calibration phase is conducted with the aim of defining some simulation parameters, such as the computational costs of *Map* and *Reduce* functions. The hardware

Table 5.3: Simulation Parameters

Homogeneous	Heterogeneous
# Hosts	# Hosts
Bandwidth	Bandwidth
Network Latency	Network Latency
# Reduces	# Reduces
Chunk size	Chunk size
# Replicas	# Replicas
# Map slots	# Map slots
# Reduce slots	# Reduce slots
% Intermediate data	% Intermediate data
Map task_cost	Map task_cost
Reduce task_cost	Reduce task_cost
	# volatile_node
	failure_timeout_period

configuration is based on the epoch of Yanpei working. On the basis of these clusters characteristics, the simulation was conducted in the homogeneous environment. The *Shuffle* size indicates the network overhead of the data transference between the *Map* and *Reduce* phases. After this, the parameters are extended to form the hybrid infrastructure under the BIGhybrid simulator. A similar technique was adopted in the work of (ROCHA; SENGER, 2013). The parameters of Table 5.2 will make it possible to find the constraints to validate these strategies.

A brief evaluation of input data in Table 5.4 demonstrates two possible scenarios. The workload of clusters YH2 and FB1 has a low-density of chunks per host, equal to 1.6 chunks/host and 1.2 chunks/host respectively, which suggest an inefficient cluster use. On the other hand, the workload of YH-1 has a density of 4.6 chunks/host, indicating a good cluster utilization. The little data output on YH-2 and FB-1 clusters demonstrates a high computational cost of *Reduce* phase. The experiments evaluate these two scenarios, in particular, the YH-1 cluster.

Table 5.4: Workload characterization for Big Data applications

Cluster	Input	*Shuffle*	Output	*Map* Time (s)	*Reduce* Time (s)	Label
YH-1	568GB	76GB	3.9GB	270.376	589.385	Aggregate, fast job
YH-2	206GB	1.5TB	133MB	983.998	1,425.941	Expand and aggregate jobs
FB-1	230GB	8.8GB	491MB	104.338	66.760	Aggregate, fast

Around 1,800 experiments of simulation were conducted in Nancy, Grenoble, Sophia and Rennes on Grid'5000 clusters with aims to find compatible patterns for low and high scale in hybrid environments. In addition, the Global Dispatcher layer was also evaluated on the basis of real-world experiments in a geo-distributed scenario. This experiments were conducted within the Cloud Azure infrastructure at three sites located in West US, North Europe, and East Japan, with one, two and three nodes respectively in each location. The environment is composed by external data sources, Global Dispatcher and one Big Data engine (Apache Flink). The Global Dispatcher system, in real-world experiments, is composed of elements, such one message queue service (Kafka) and one coordinating service or orchestrating system (Zookeeper). The experiment is structured in all the uncoupled systems.

5.2.1 Use Case Study

The use scenarios reflect how the devices interact with the systems. The *uncoupled system* is introduced in Figure 5.2. The data is collected through the orchestrators and can be geo-distributed in an uncoupled scenario. The different data sources are formed in an unstructured way as text (in the Twitter applications), photos (in the Facebook applications), signals (provided by sensors), social networks or other kinds of information. The sensor information must be pre-processed before it can be sent to the orchestrator, so as to avoid environmental noises.

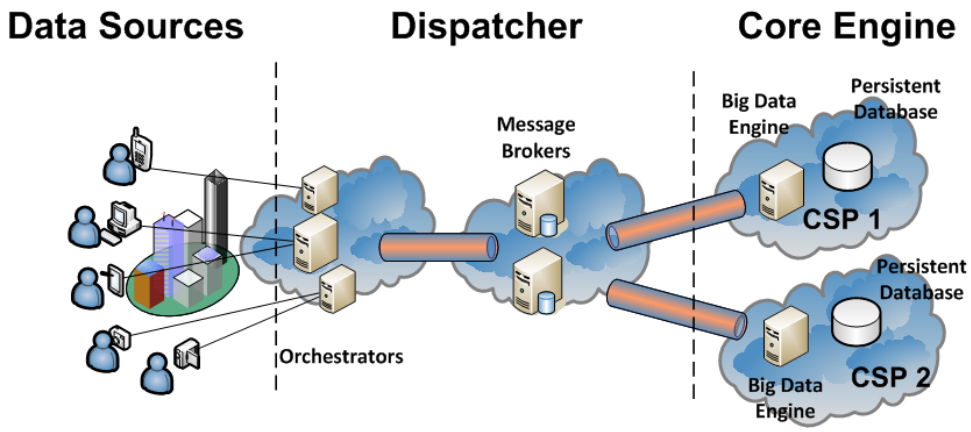


Figure 5.2: The uncoupled scenario for the SMART platform.

The orchestrator system is composed of one or more servers in each locality within a distributed system. In this context, the Zookeeper system maintains a fault tolerance mechanism. The message brokers receive the data in the form of topics that are immediately forwarded to the Core engine on two CSPs, in a storage-forward policy. The Dispatcher can swap the data destination to the Desktop Grid environment, for instance, or add more one CPS to the system based on the workload. In this scenario, all the systems are uncoupled from each other. Also, the dataset can be spread over several locations. This approach is adopted to avoid lengthy transfers of data between sites and allows the load balance. However, can be a management overhead. For instance, an application to avoid the market manipulation on the Stock Exchange with several users producing interactive data. In this example, the sensors are connectors like Twitter, Facebook, and real-time applications in the Stock Exchange which provide the latest price of the stock trading. The historical stock prices of the Stock Exchange are evaluated with information about sensors by means of a Big Data applications and analytics.

Most of the applications shown in the “Related Work” Section follow the coupled system model. Figure 5.3 shows this scenario. The model is hard to configure for new CSPs, for instance in the Dispatcher layer, although the SMART architecture can enable more complex executions due to the distributed nature of Zookeeper. On the other hand, another orchestrator can be implemented to provide a processing of Big Data within Desktop Grid environment before the message broker service.

The semi-coupled scenario is shown in Figure 5.4. This scenario has the advantage of avoiding slow links to data transfers between the orchestrators and message brokers. In addition, also it is possible to replicate the Dispatcher layer in several instances in other CSPs. The semi-coupled scenario represents a hybrid platform that can enable new VM launches in other CSPs to provide further computational capacity, as well as to allow the

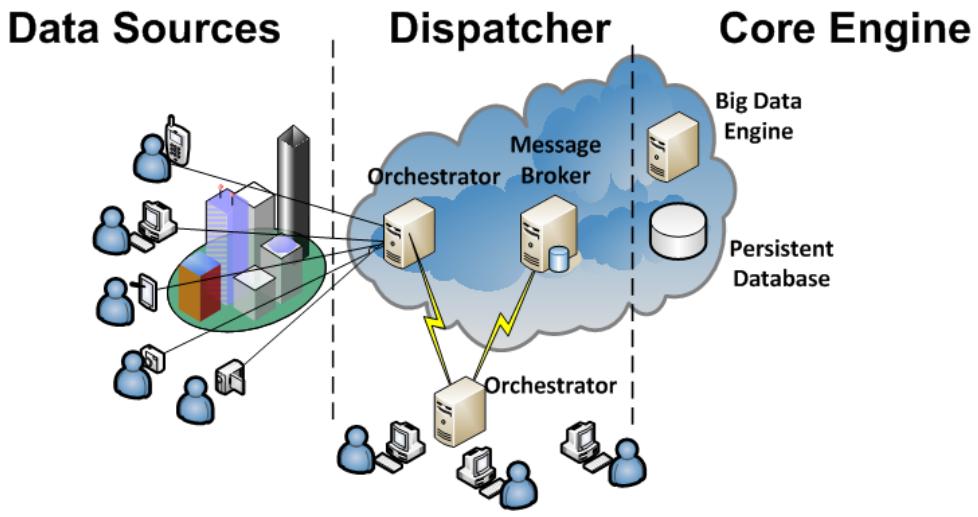


Figure 5.3: The coupled scenario for the SMART platform.

DG to be used like a Big Data processing engine. The semi-coupled scenario enables to evaluate Big Data in a distributed system environment in a hybrid platform. This last scenario will be simulated with the BIGhybrid to evaluate its constraints and also, the latency effect will be examined in an experiment in the Cloud Azure.

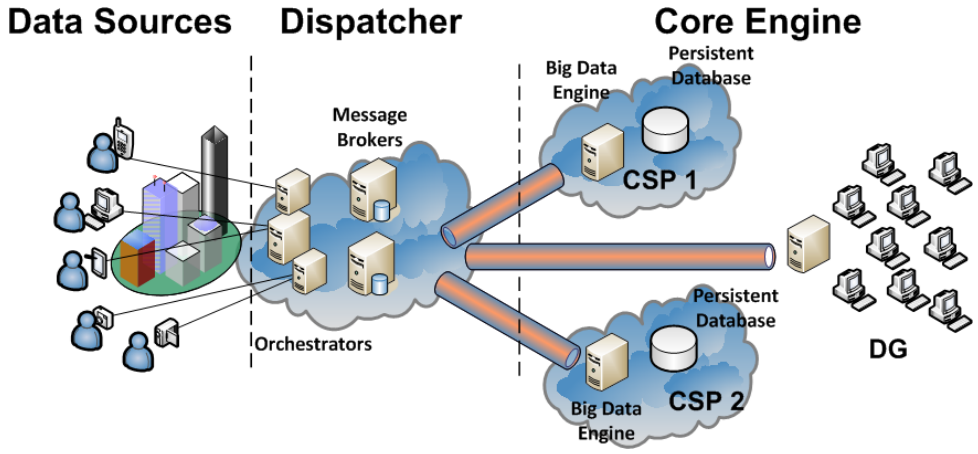


Figure 5.4: The semi-coupled scenario for the SMART platform.

5.3 Results

This Section shows the evaluations for finding behavior patterns to deploy on hybrid environments in low, medium and high scale and establishing the strategies of: a) A relation between execution time and volatile nodes; b) A relation to resource distribution in Cloud and DG; c) Investigating data size in function of available bandwidth; d) Determining a relationship between the load balance and number of machines; e) Evaluating the cost related to Cloud and DG. The conducted experiments are real-world in Cloud and aims to observe a behavior of real-time applications in an uncoupled scenario and simulated tests reproducing workloads from Big Data applications in a semi-coupled scenario.

5.3.1 Experiments in Real-time Execution to Dispatcher Evaluation

This experiment evaluates the uncoupled scenario deployed in the Microsoft Azure a CSP. Figure 5.5 illustrates the environment which is composed of three sites in a geo-distributed configuration. The sites are located in the West US (site A), North Europe (site B), and the East Japan (site C). The machines are of the Standard_A10 type and this is equivalent to Intel Xeon E5-2670 8 cores @ 2.6 GHz - 56 GB Ram DDR3 -1600 MHz. The inter-site bandwidth is 203 Mbps from site A-to-B and 83 Mbps from site B-to-C. The environment represents several users sending Tweets to the Message Broker (Apache Kafka) through Zookeeper. The data source producers are represented as near-real-time data streams. The data is arranged in a workload size of 100 to 100,000 messages of 2KB each. All the workload groups are transmitted 100,000 times when reproducing a Big Data environment. The experiment has 20 repetitions to each workload, with a total execution time of 12 hours.

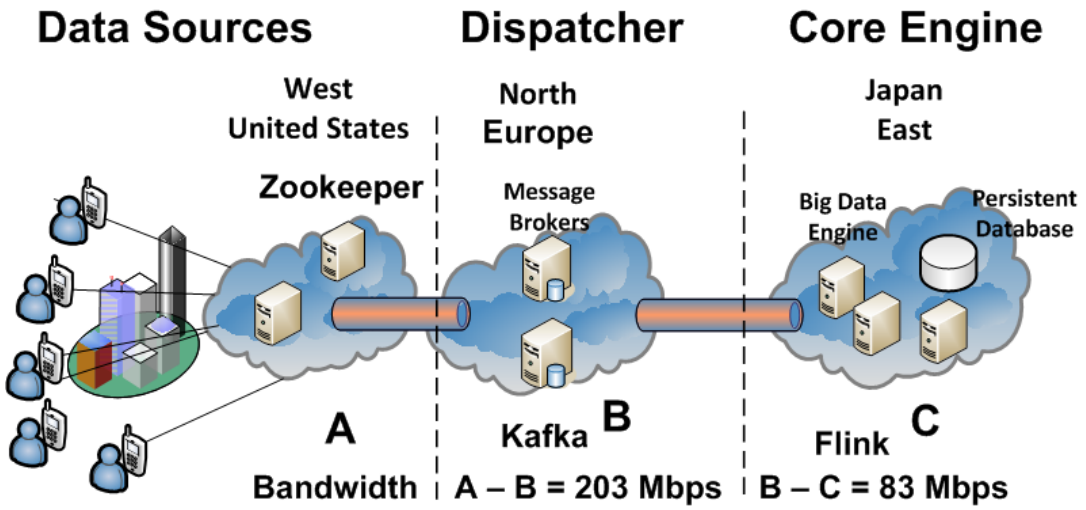


Figure 5.5: Application of a Sentiment Analysis

The application processes the sentiment analysis for tweets by means of a Natural Language Processing (NLP) technique. The Big Data engine receives a *stream of tweets* in JSON format where each event contains a relevance rate, an ID for the tweet identity, a language, and a message. The analysis is a sentence based on a positive or negative word that can be found a priori in a dictionary; afterward it will be decided whether a rating is favorable or not, and this solves the user's query.

Figure 5.6 presents the data generation to Cloud experiment.The data generation is equivalent to 30 machines that produce data streams simultaneously without interruption. The workloads are created first in little scale from 10 to 1,000 messages of 2KB each with 50 messages offset and immediately dispatched to Kafka. After, several message groups are created with workloads from 1,000 to 100,000 messages of 2KB with 1,000 messages offset. The first generation runs in memory and the second is a process where firstly it writes message blocks to disk and after sends the files on an RPC process for Kafka. The concurrent tweet messages in the x-axis are measured in message unit, and the execution time in the y-axis is measured in milliseconds. The line green indicates the tendency curve in a 2^{nd} order polynomial function. The red line represents the measured standard deviation.

The structure of the data generation produces data bursts, *i.e.*, periods with a massive

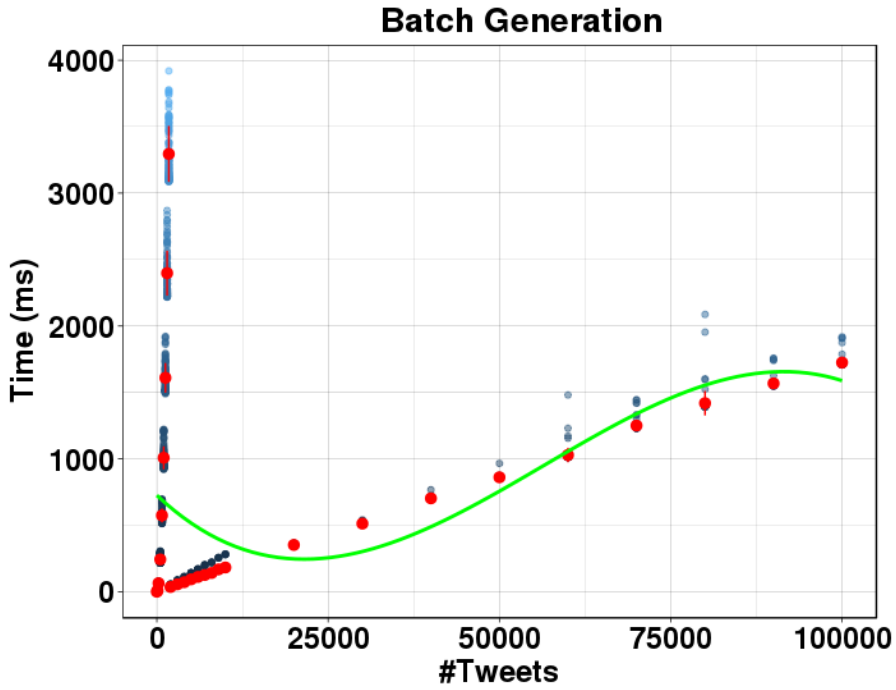
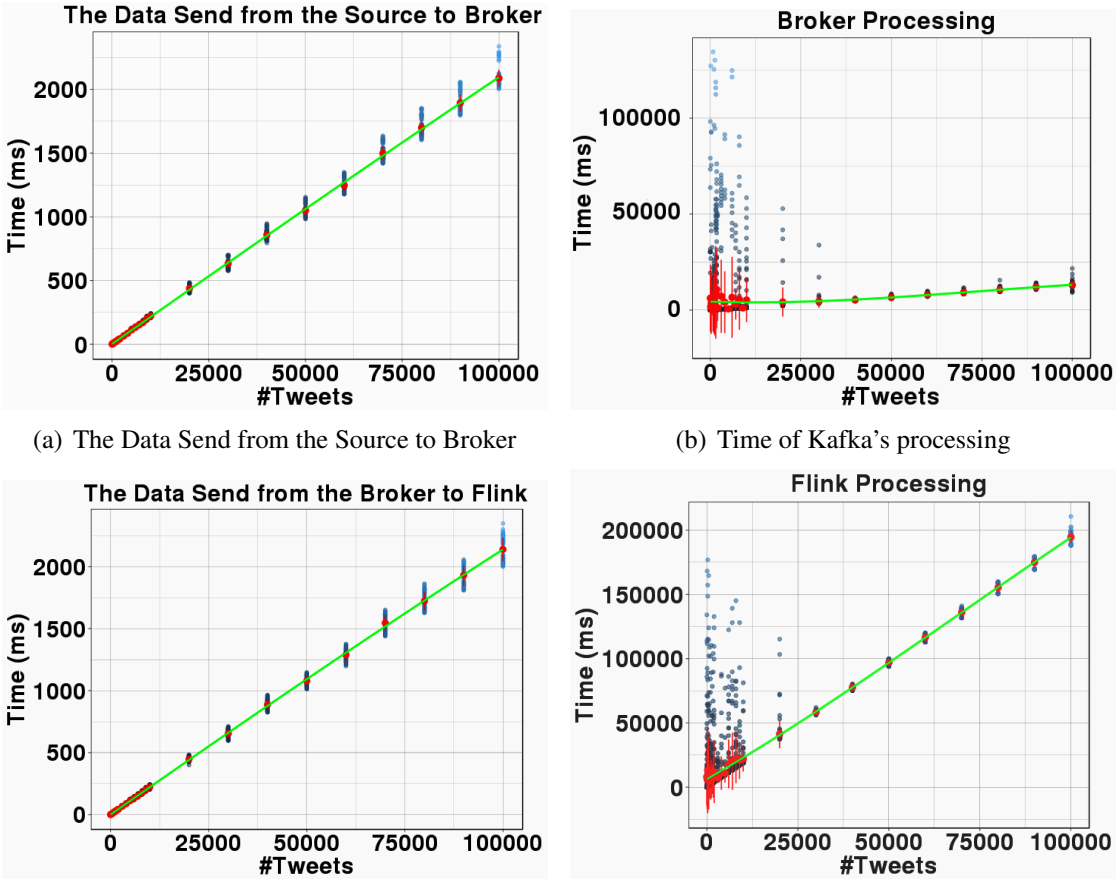


Figure 5.6: Data Generation to Cloud Experiments

transfer of data which can generate system overloads and others with little or without data traffic. This process is very common in applications like Twitter which occur in the interval of events like football games or events where there is a very high agglomeration of people. The generation manner has direct influence in the data processing by Big Data engines. If the data production is changed for a batch process of events with more data in higher time intervals, this problem that will be observed in the next experiments is minimized. However, the user could have a slow time to receive their answers until for nearest neighbors.

Figure 5.7 shows the results from the experiments with one machine in the A site, one machine in the B site and three machines in the C site. The experiment demonstrates the influence of the workload in each processing stage. The concurrent messages in the x-axis are measured in message unit, and the execution time in the y-axis is measured in milliseconds. The line green indicates the tendency curve. The red line represents the measured standard deviation.

The spent time for the data send from the source to broker (Kafka) and from the Kafka to the Flink, respectively, Figure 5.7.a and Figure 5.7.c, are equivalents as a result of large bandwidth inter-site available. The impact of the Dispatcher execution, in Figure 5.7.b, is lower than the Flink's processing, except those executions of smaller blocks. The data generation in Batch Generation module, in comparison between Figure 5.6 and Figure 5.7, does not impact in the communication channel because there are low latency and high bandwidth to data transfers. However, the size of input data affects the time execution in both process of Kafka and Flink that indicates the best performance when the size of input data has a larger block. The problem was previously discussed and represents a bottleneck that can be minimized but not avoided. This execution is a low-scale experiment, but the tendency for the data input also will be observed in large scale on the next Sections.



(a) The Data Send from the Source to Broker

(b) Time of Kafka's processing

(c) The Data Send from the Broker to Flink

(d) Time of Flink's processing

Figure 5.7: Workflow of Processing in a Hybrid Infrastructure

5.3.2 Experiments with the Hybrid Environments Simulation

This subsection describes an experimental environment setup to validate strategies in hybrid environments in a semi-coupled scenario. The experiments involve conducting discrete-event simulations on the environment of Grid'5000 in four sites (Sophia, Nancy, Rennes and Grenoble). The clusters located in Sophia and Grenoble, has 64 hosts and 50 hosts respectively, with 2 Intel Xeon E5520 processors of 2.27 GHz, with 4 cores, 24 GB of RAM, 119 GB of local disk and 1 Gbps network. The clusters in Nancy and Rennes have 40 and 70 hosts respectively, each one with 2 processor Intel Xeon E5-2630 v3 of 2.4 GHz, with 8 cores, 126 GB of RAM, 298 GB of local disk and 10 Gbps network. The simulations are deployed on each machine with a particular configuration with the aim of reducing the execution time from experiments. The evaluation of strategies for the use of hybrid environments in Big Data is divided into low-scale, middle-scale and high-scale experiments.

The *MapReduce* execution occurs in waves like in Figure 5.8. The example has 5 machines to process a *MapReduce* application with 20 chunks and 5 *Reduces* in the end. The concurrent tasks in the y-axis are represented in unities and the x-axis, on the top of Figure 5.8, measures execution time in seconds. The concept of available resources (R) is defined as *the number of machines multiplied by the number of slots* for one execution of the *MapReduce* tasks. Thus, a second wave begins if the first wave allocates all free computational resources. On the other hand, the lower time unity for a *Map* phase is equivalent to the execution time of one wave. These observations are very useful for

the analysis of resource distribution because the number of waves is independent of the execution time.

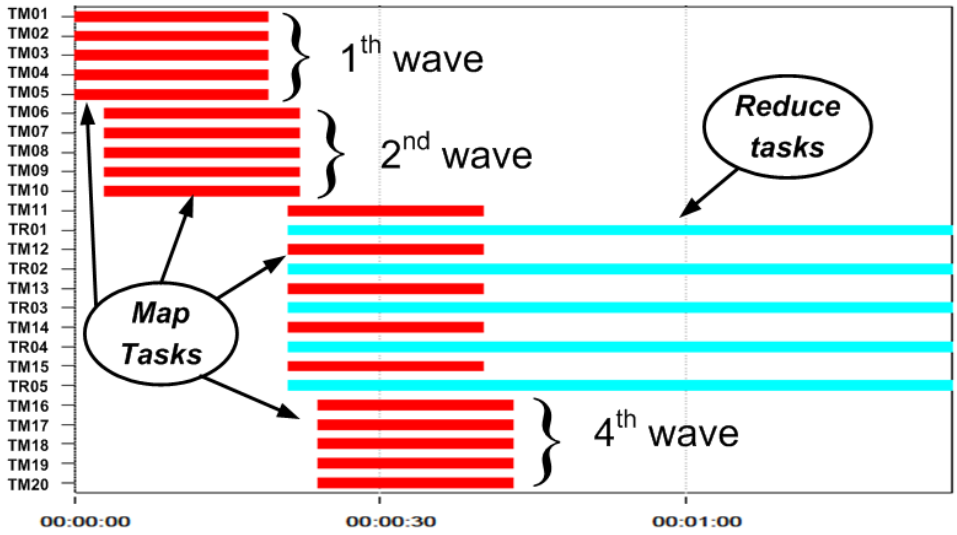


Figure 5.8: Waves in MapReduce Executions

The total resources R_{Tc} are the sum of Vm instances (Vm_c) multiplied by *Map* free slots of a machine (τ_{Mc}). However, the *MapReduce* scheduling first uses the first free slot of all machines and then distributes tasks for the other free slots forming an execution wave. Equation 5.3.1 introduces the relation Φ that is the result of total workload in Cloud (W_c) (equivalent to chunk numbers) divided by the available resources in Cloud (R_c). When $\Phi_C = 1$, then all resources are busy, and there is a full execution wave.

$$\Phi_C = \frac{W_c}{R_c}, \text{where } R_c = \sum_i^M Vm_c(i) \text{ and } R_c \subseteq R_{Tc} \tag{5.3.1}$$

For finding the best data distribution, we must take half of the machines to Cloud and DG. The job in hybrid systems has two executions and to achieve a good load balance the relation Φ_{Ch} in Cloud must be balanced to relation Φ_{DG}. The best strategy is to use the Equation 5.3.1 to find the load balancing. Equations 5.3.2 and 5.3.3 reflect the relation Φ in Cloud hybrid and DG, where Ck_{Ch} and Ck_{DG} are the chunk size on Cloud hybrid and DG respectively, and Ck_c the chunk size on single execution in Cloud. Finally, the best load balance there is when if verify the Equation 5.3.4, this relation defines the data distribution related to the number of Cloud and VC resources which is the goal of *data load balance* strategy.

$$\Phi_{Ch} = \frac{Ck_{Ch}}{Ck_c} * \frac{W_{Ch}}{\sum_i^M Vm_c(i)} \tag{5.3.2}$$

$$\Phi_{DG} = \frac{Ck_{DG}}{Ck_c} * \frac{W_{DG}}{\sum_i^M M_{DG}(i)} \tag{5.3.3}$$

$$\Phi_{Ch} \leq \Phi_C \Longleftrightarrow \Phi_{DG} \leq \Phi_C \tag{5.3.4}$$

The computational capacity of processors in the simulated experiments is equivalent to an Intel Xeon E5506 - 2 Cores, 4M Cache, 2.13 GHz ≈ 5 GFlops (BICAK, 2017) and the computational capacity in Volatility environment represents a distributed value between 4 to 6 GFlops, for all experiments. The configuration is similar that we found on Yahoo and Facebook by evaluation of Yanpei (CHEN, 2012). To analysis effect, the computational consumption is cpu_{64} = 416 GFlops, cpu_{32} = 208 GFlops and cpu_{16} = 104 GFlops for a 64, 32 and 16 MB chunk size respectively. The network, workload and the machines numbers vary according to each experiment. The amount of the *Reduce* tasks is equal to two times the machine number.

Initially, it is important to define an input workload. The experiments are available according to two environments: one has more tasks than resources available ($T > R$) such as the FB-1 and YH-2 clusters, in the Yanpei evaluation, and other has more resources than tasks ($R > T$) as the YH-1 cluster.

The first experiment in Figure 5.9 represents an execution in low-scale of a Hybrid environment, the aim is to obtain the best utilization of Cloud and distributing the most number of chunks on the DG adjusting the chunk size when necessary. The concurrent task execution in the y-axis is measured in seconds, and the number of the machines for Cloud and DG (Cloud/DG) in the x-axis is measured in units. The line red indicates the time execution of 200 tasks (chunk size 64 MB) on 128 machines in a single Cloud deploying, equivalent to 503 seconds (therefore, with basis on earlier concept of available resources, the environment has $R > T$). The chunk size and input data are described in the graphic to each execution for Cloud (C) and DG. Each execution is indicated with a character on top of the chart and represents a job in Hybrid environment where the first execution is in the Cloud and after on the DG. The networking for Cloud is 1 Gbps and to DG is 10 Mbps.

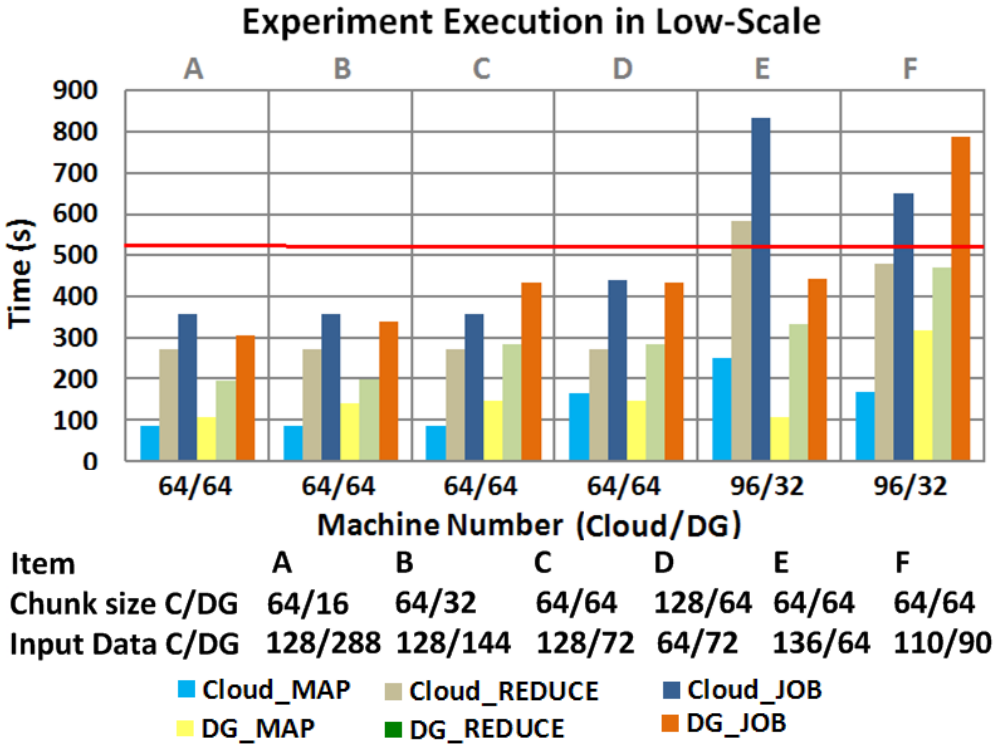
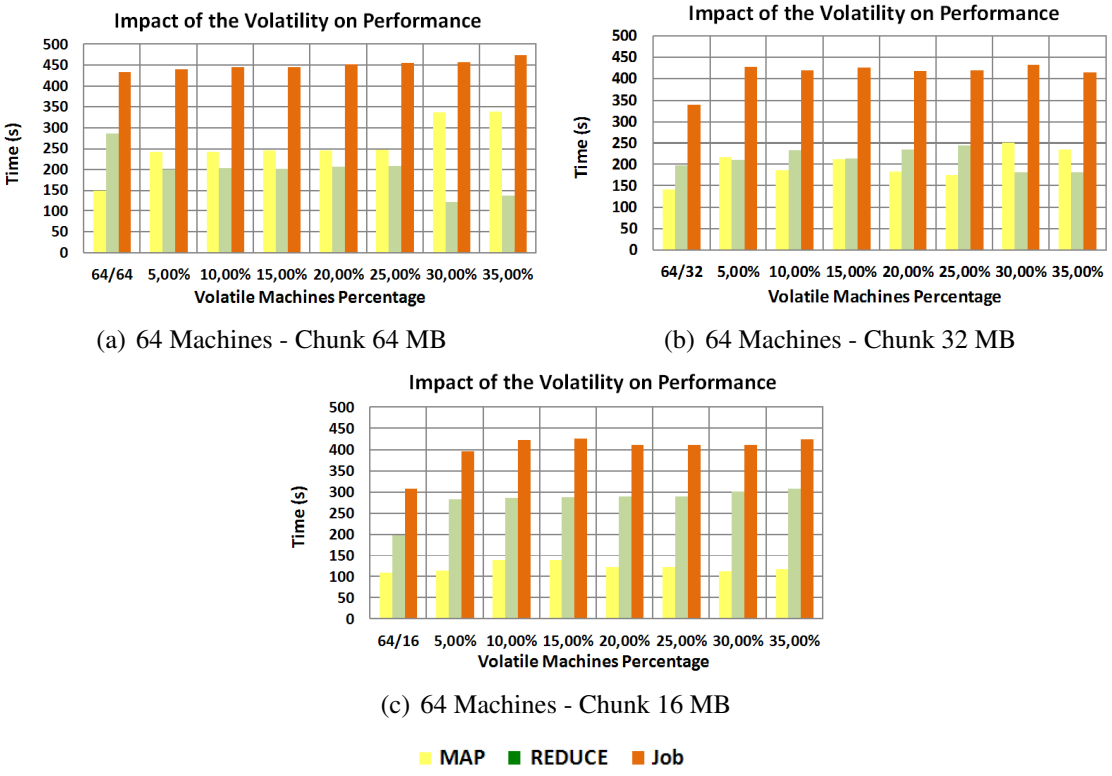


Figure 5.9: Low Scale - 128 Machines - 10 Mbps - Resources > Tasks

This execution environment has more resources than tasks ($R > T$) and the time is limited to 1.5 waves or a maximum of 2 waves. The time of Cloud job in **D** is longer than **C** in function to missing of task parallelism. On the other hand, the unbalanced data distribution results in more execution time like in **E** and **F**. The profile execution of DG in Figures 5.9 (**A**) and (**D**) are the same because the change of 128 MB chunk size occurs in Cloud and not in DG. Then, the strategy for this case is to use the greater amount of resources possible, like in the executions **A**, **B** and **C** of Figure 5.9. However, as $R > T$ is limited to 1.5 wave, we can only establish a inferior limit $\Phi_C = 1$ to make sense the next considerations.

Figure 5.10 presents the impact of volatility on performance where $R > T$. The letters (**a**), (**b**) and (**c**) correspond to executions **A**, **B** and **C** of Figure 5.9 respectively. The y-axis measures the time of concurrent tasks in seconds and the x-axis represents the percentage of volatile machines. The first execution on the left, for all Figures, is the implementation without volatility for comparison purposes which is the basis of the volatility increment made in offsets of 5% until 35%. The experiments show the behavior of the task executions considering 10 Mbps bandwidth.



(a) 64 Machines - Chunk 64 MB

(b) 64 Machines - Chunk 32 MB

(c) 64 Machines - Chunk 16 MB

Figure 5.10: Impact of Volatility on Performance - 10 Mbps - Resources > Tasks

This experiment demonstrates a relative resiliency of hybrid infrastructure with the volatile environment, but this can be an outcome of low-scale of machines and a small number of tasks compared with available resources. However, the experiment scale produces a little message exchanges between nodes due to having more resources than tasks and does not enable a definitive conclusion about this problem. This factor will be examined in the next experiments. On the other hand, experiment (**d**) demonstrates the problem described before about missing of task parallelism with the size task increase.

In this scenario, the best choice is to distribute data with the using of all slots in the

machines in Cloud which will maximize the use of resources and define a maximum number of DG machines to achieve the best execution in one wave. This means, chase the Φ lower limit.

The next experiment has more tasks than resources ($T > R$) with workload of 512 tasks (chunk size 64 MB) and 128 machines. The hardware and task configurations are the same that the earlier experiment. The aim this experiment is to consolidate the observations to the relation of data distribution between Cloud and DG. Figure 5.11 shows eight executions named from **A** to **H** on top of chart. The chunk size and input data are described in the graphic to each execution for Cloud (C) and DG. The line red indicates the time for a single Cloud deploying, equivalent to 1,232 seconds. The concurrent task execution in the y-axis is measured in seconds, and the number of the machines for Cloud and DG (Cloud/DG) in the x-axis is measured in units. The networking for Cloud is 1 Gbps and to DG is 10 Mbps.

The experiment of Figure 5.11 shows that data distribution for this scenario achieves the best load balancing when the Equation 5.3.4 is verified. Table 5.5 demonstrates that for this case when $T > R$ is possible to establish a superior limit for the relation Φ for Cloud and DG which is independent of the time execution and has a value of $\Phi \leq \Phi_C$. This, relationship for the resource distribution can be summarized in Equation 5.3.5. This relation enables to define the machine number in a hybrid environment through an execution simulation on Cloud and then to set the data distribution for the hybrid environment.

$$1 \leq \Phi \leq \Phi_C \tag{5.3.5}$$

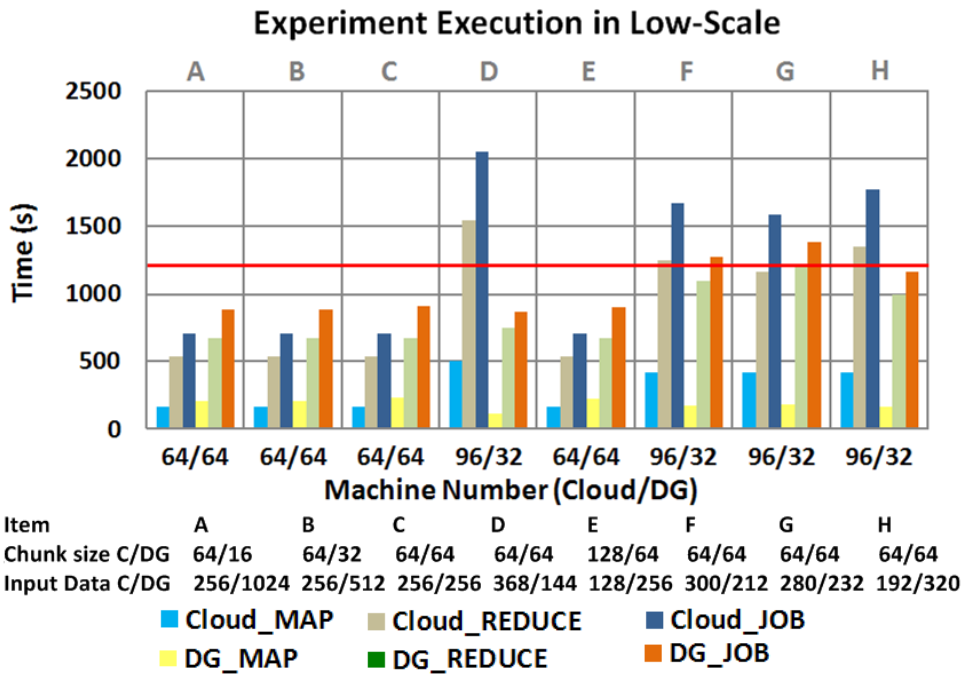


Figure 5.11: Low Scale - 128 Machines - 10 Mbps - Tasks > Resources

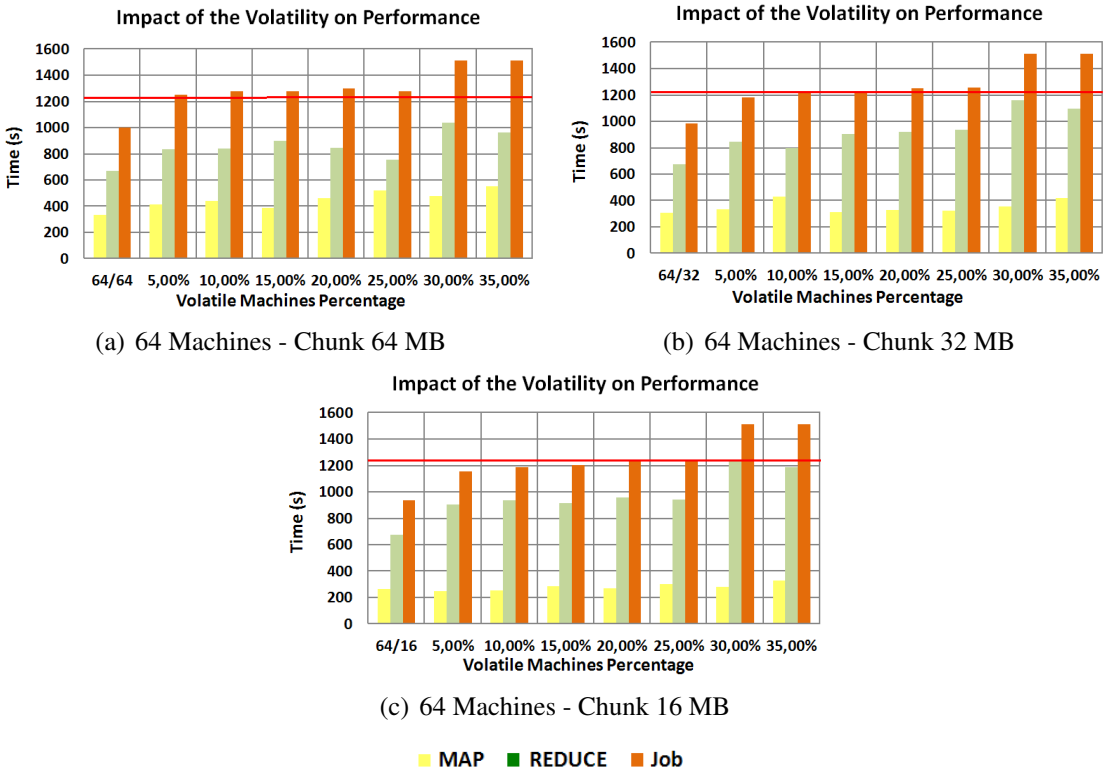
This relationship can be considered consistent, not only in low-scale but also in high-scale as will be demonstrated in the next experiments. However, this analysis was conducted with the nodes without the volatility effect. Because of this, the value of Φ can be lower than the superior limit in some cases. The executions **A**, **B**, **C** and **E** of Figure 5.11

Table 5.5: The relationship Φ for data distribution in hybrid environments

Relation/Experiment	Cloud	A	B	C	D	E	F	G	H
Φ_{Ch}/Φ_{DG}		4/4	4/4	4/4	3.83/4.5	4/4	3.13/6.63	2.91/7.25	2/10
Φ_C	4								

have a performance acceptable in this case. If the relation Φ_{Ch} would have a lower than Φ_C the number of machines in Cloud into the hybrid environment would be near from max number of hosts in a single Cloud execution and the using of DG will not produce a financial economy.

Figure 5.12 demonstrates the impact of volatility on performance where $T > R$. The letters (**a**), (**b**), (**c**) and (**d**) correspond to executions **A**, **B**, **C** and **E** of Figure 5.11 respectively. The y-axis measures the time in seconds and the x-axis represents the percentage of volatile machines. The first execution on the left, for all Figures, is the implementation without volatility for comparison purposes which is the basis of the volatility increment made in offsets of 5% until 35%. The experiments show the behavior of the task executions with networking of 10 Mbps bandwidth. The execution **E** has the same execution time from **C** in this case The line red represents the execution time of 1,232 seconds in a single Cloud.



(a) 64 Machines - Chunk 64 MB

(b) 64 Machines - Chunk 32 MB

(c) 64 Machines - Chunk 16 MB

Figure 5.12: Impact of Volatility on Performance - 10 Mbps - Tasks > Resources

The impact of volatility is sensitive for tasks with a chunk size from 16 to 64 MB due to the overhead of data copy when a machine has a shutdown. The low bandwidth of the network affects the data transfers considerably. The experiments suggest an operation flexibility in volatile environments when 20% to 25% of machines have a shutdown without important performance degradation. The chunk size for this scenario is 32 MB and 16 MB respectively.

However, losing 1/4 of machines in DG can produce a high latency in a slow Internet link. The DG environments can be until relatively stable in some scenarios but do not have a behavior predictable easily. Other consideration, it is to evaluate the strategy of the mechanism for improving performance in Big Data in hybrid infrastructure, discussed earlier. In a volatile environment, the machines can have an overhead with data copy to rebuild replicas that this mechanism could experiment long timeout periods, wich can take the FTM to consider false-negative in the execution behavior.

The next experiment reproduces the hardware configuration earlier of 128 machines with $R > T$, but in a network of 100 Mbps. The aim is to identify behavior changes related to bandwidth in this scenario. Figure 5.13 presents this experiment. In the y-axis the concurrent task execution is measured in seconds, and the number of the machines for Cloud and DG (Cloud/DG) in the x-axis is measured in units. The line red indicates the time execution of 200 tasks (chunk size 64 MB) on 128 machines in a single Cloud deploying, equivalent to 503 seconds. The chunk size and input data are described in the graphic to each execution for Cloud (C) and DG. Each execution is indicated with a character on top of the chart and represents a job in Hybrid environment where the first execution is in the Cloud and after on the DG. The networking for Cloud is 1 Gbps and to DG is 100 Mbps.

The results of Figure 5.13 suggest a similar behavior of executions in comparison with a low-bandwidth scenario. The executions **A**, **B**, **C** and **D** present the best performance in this case. This performance is related to the phenomenon of data locality which presents little data movement in *Map* phase. Thus, the bandwidth can modify the profile execution time only during *Shuffle* phase when occurs the data movement of intermediate pairs. However, this behavior changes in the presence of volatility due to data transfers for preserving the number of replicas in HDFS when occurs a node shutdown.

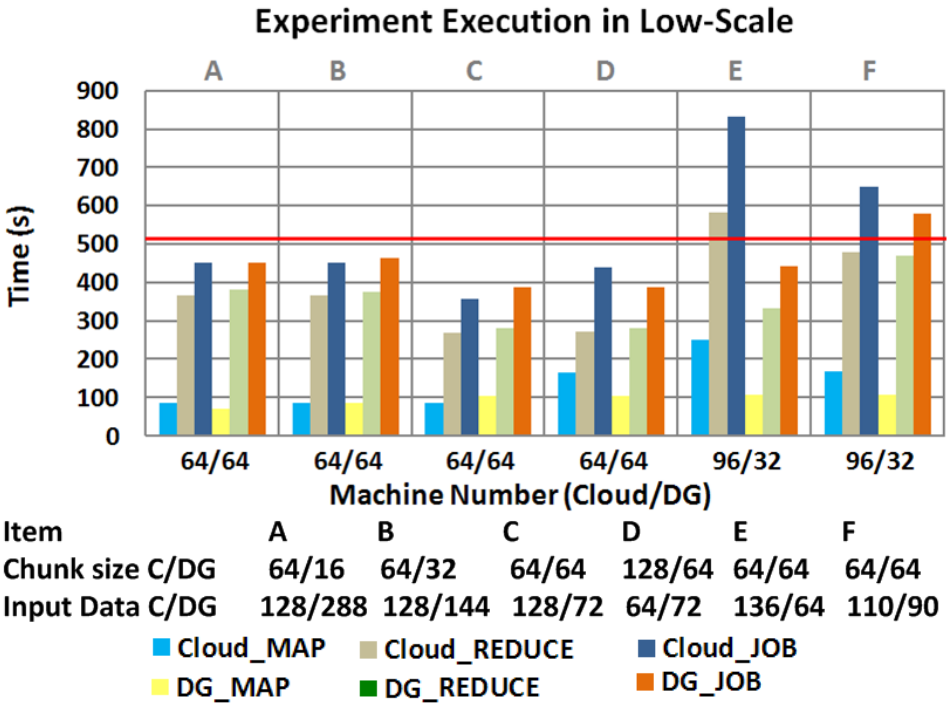
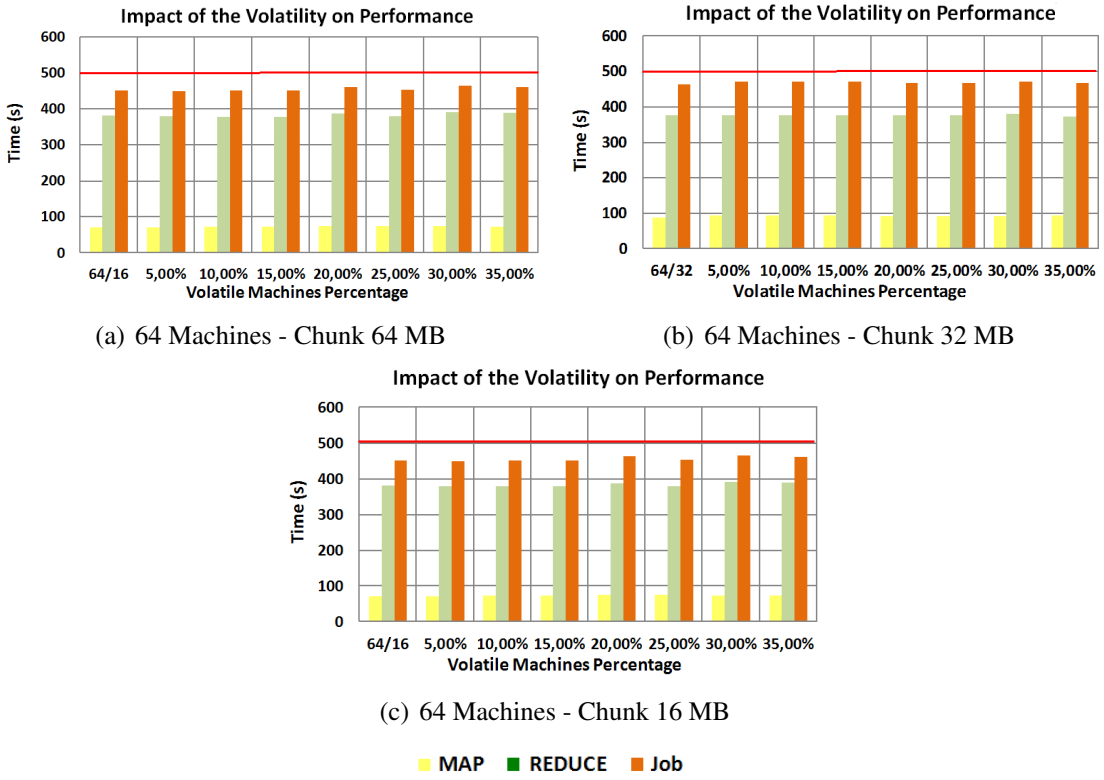


Figure 5.13: Low Scale - 128 Machines - 100 Mbps - Resources > Tasks

The next experiment demonstrates the profile changes in the function of a larger bandwidth in comparison with the experiment of Figure 5.10. Figure 5.14 shows the executions

in letters (**a**), (**b**) and (**c**) that correspond to executions with the best performance in Figure 5.13 respectively. The evaluation of execution **D** is not considered due to reasons exposed earlier. In the y-axis, the time of concurrent tasks is measures in seconds and the x-axis represents the percentage of volatile machines. The first execution on the left, for all Figures, represents the implementation without volatility for comparison purposes. The experiments show the behavior of the task executions considering 100 Mbps bandwidth. The line red represents the execution time of 503 seconds in a single execution in Cloud.



(a) 64 Machines - Chunk 64 MB

(b) 64 Machines - Chunk 32 MB

(c) 64 Machines - Chunk 16 MB

Figure 5.14: Impact of Volatility on Performance - 100 Mbps - Resources > Tasks

In fact, the impact of volatility in Figure 5.14 indicates be small with 100 Mbps. The data copies needed to establish the replica relation is faster in fast links than slow links what contributes to the best performance. Thus, the configurations of the hybrid environment with chunks of 64 MB, 32 MB and 16 MB are similar. However, in these experiments, there are few tasks in comparison with resources, and with the greater bandwidth, the problem is hidden. On the other hand, the environment with more resources than task is familiar in Cloud, because the CSPs need to have free computational resources to allocate their available resources for the user and to occupy the most hardware possible. The best strategy is the user allocate the exact slot of resources by its Job.

To complete our analysis in low-scale, also is necessary evaluate the environment with $T > R$ in a networking of 100 Mbps. Thus, the next experiment evaluate the execution in low-scale of a Hybrid environment, to confirm the strategies to obtain the best utilization of Cloud and distributing the most number of chunks on the DG. Figure 5.15 represents an execution in low-scale of a Hybrid environment, with a workload of 512 tasks with chunk size 64MB and 128 machines. Remembering, the hardware configuration is equivalent to an Intel Xeon E5506 - 2 Cores, 4M Cache, 2.13 GHz $\approx$ 5 GFlops and the computational capacity in Volatility environment represents a distributed value between 4 to 6 GFlops,

for all experiments. The computational consumption is cpu_{64} = 416 GFlops, cpu_{32} = 208 GFlops and cpu_{16} = 104 GFlops for a 64, 32 and 16 MB chunk size respectively.

The concurrent task execution in the y-axis is measured in seconds, and the number of the machines for Cloud and DG (Cloud/DG) in the x-axis is measured in units. The line red indicates the time execution in a single Cloud deploying, equivalent to 1,232 seconds. The chunk size and input data are described in Figure to each execution for Cloud (C) and DG. Each execution is indicated with a character on top of the chart and represents a job in Hybrid environment where the first execution is in the Cloud and after on the DG. The networking for Cloud is 1 Gbps and to DG is 100 Mbps.

It is important to observe that executions in Figure 5.15 follows the same execution profile from Figure 5.11. The evaluation of relative distance between the greater time of the best performance results in 10 Mbps is 18,74%, and 100 Mbps is 26,49%. Thus, we can imagine that improving the bandwidth from 10Mbps to 100Mbps must produce a performance gain near to 8% for the same application in this scenario. The result performance is also due to the data is ready to be consumed in the local disks in the Map phase. However, the approach of reducing the chunks from 64 MB to 16 MB to improve the executions with small bandwidth can not be sufficient to hybrid environments with high bandwidth. The next experiment with the use of volatility justifies these comments.

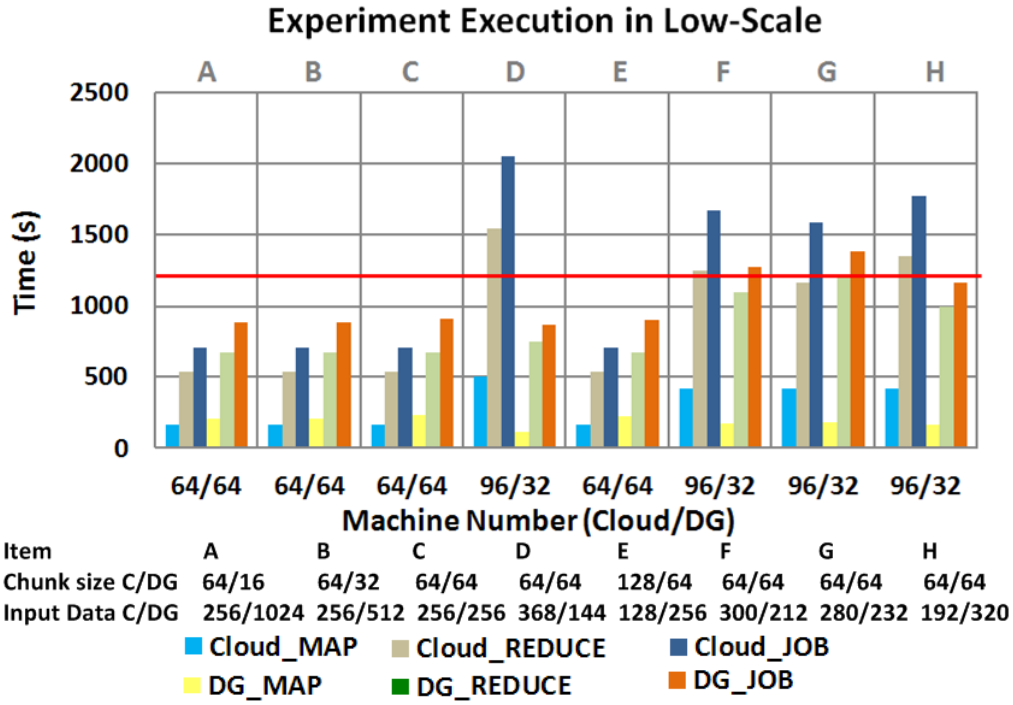
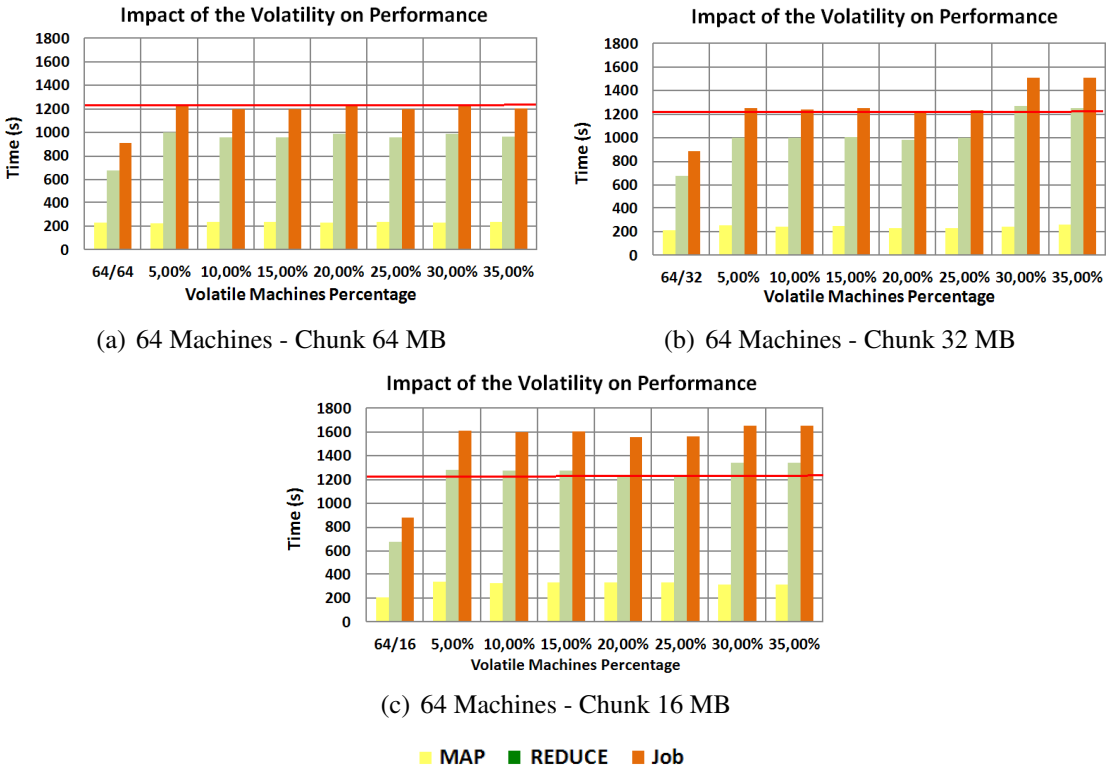


Figure 5.15: Low Scale - 128 Machines - 100 Mbps - Tasks > Resources

The next experiment in Figure 5.16 has aim of analyzing the impact of volatility in a hybrid environment, considering ($T > R$) in a 100 Mbps links. The execution is with basis on experiment earlier. The letters (**a**), (**b**) and (**c**) correspond to the best executions **A**, **B** and **C** of Figure 5.15 respectively. The execution of letter **D** is the same than the letter **A** in function of change chunk size from 64MB to 128 MB occurs only in Cloud. The y-axis measures the time of concurrent execution tasks in seconds and the x-axis represents the percentage of volatile machines. The first execution on the left side refers to deploying without volatility.

Figure 5.16 shows an impact significant of volatility under this setup. The experiments (**a**) and (**b**) with 64 MB and 32 MB chunk size, respectively, accept from 5% to 25% of volatile machines. However, the (**c**) experiment with a 16 MB chunk size demonstrates to be unfeasible in this scenario. The data transfers increase with volatility and the recurrent execution of Reduce phase, harms further the performance of the all job.

Until this moment, it is possible to determine some other limits for Big Data execution in hybrid environments. The first strategy to the relation between execution time and volatile nodes points to value $5\% < \xi < 25\%$, in agreement with these initial observations of the volatility on low-scale. The chunk size found to meet the requirements of channel communication is 32 MB and 64 MB, also considering the low-scale environment. Finally, the experiments suggest that the behavior profile of a workload in low-scale can be affected by the volatility presence and not by bandwidth which the environment is executed.



(a) 64 Machines - Chunk 64 MB

(b) 64 Machines - Chunk 32 MB

(c) 64 Machines - Chunk 16 MB

Figure 5.16: Impact of Volatility on Performance - 100 Mbps - Tasks > Resources

The next experiments are executed in medium-scale, which are nearest from academic environments. This experiments have aim to analysis the efficacy of data distribution relation and identifying the bandwidth impact in all hybrid infrastructure environment. The hardware configuration is equivalent to an Intel Xeon E5506 - 2 Cores, 4M Cache, 2.13 GHz $\approx$ 5 GFlops and the computational capacity in Volatility environment represents a distributed value between 4 to 6 GFlops, for all experiments. The computational consumption is cpu_{64} = 416 Gigaflops/Byte, cpu_{32} = 208 Gigaflops/Byte and cpu_{16} = 104 Gigaflops/Byte for a 64, 32 and 16 chunk size respectively. The single execution in Cloud processes 4608 chunks of 64 MB with 512 machines in 1,618 seconds. The bandwidth varies of 10 Mbps, 50 Mbps, 100 Mbps, 150 Mbps, 300 Mbps and 1 Gbps with latencies captured of the real-world environment. The bandwidth of 1 Gbps is considered only in

Cloud environment. Observes that a 1 Gbps bandwidth is possible, in this moment, only in Cloud.

In the first experiment, the number of the machines is divided into the half, 256 machines to Cloud and 256 machines to DG. The total tasks are equally divided in the half, 2,304 to the Cloud and DG. The job execution time in Cloud is equal to 1,616 seconds, in a network of 1 Gbps bandwidth, with a chunk size of 64 MB, for all executions. The Figure 5.17 presents the time results for the DG job execution. The color blue, green and yellow represent the executions with a chunk size of 64 MB, 32 MB, and 16 MB respectively. The red line represents the time execution for a single execution in Cloud equal to 1,618 seconds. The y-axis, on the left, measures the concurrent tasks in seconds. The y-axis, on the right, measures the DG workload in units for each execution with different chunk size. The workload is of 2,304, 4,608 and 9,216 chunks and has a chunk size of 64 MB, 32 MB, and 16 MB respectively. The x-axis measures the bandwidth. The aims is to verify if a single data division will be sufficient to data split, in a ($T > R$) scenario.

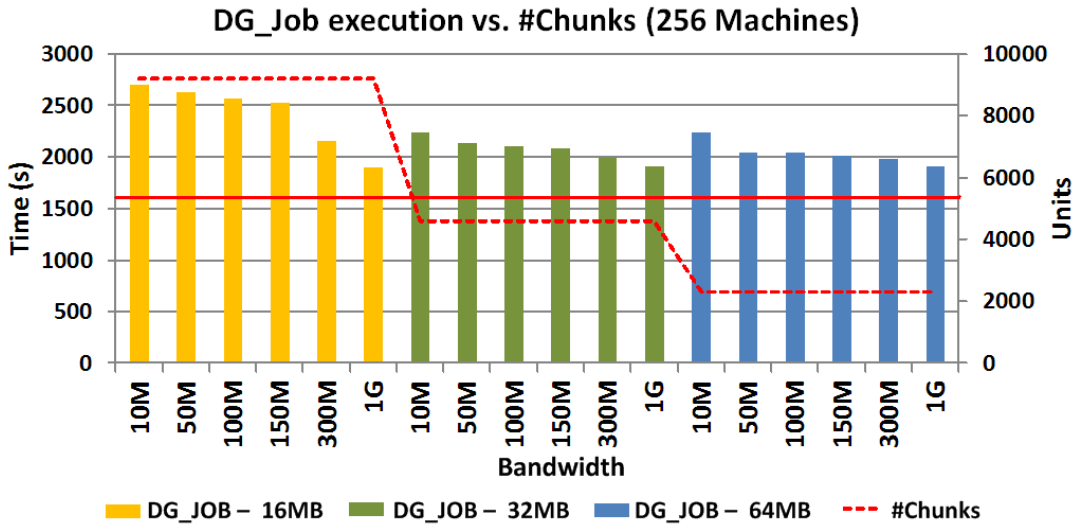


Figure 5.17: Medium-scale experiment - 256/256 machines (C/DG) - Tasks > Resources

It is possible notices that no execution achieved the time expected. The performance is worst than a single implementation in Cloud with all hosts. Thus, distributed the data and machines in half is not a good strategy to distribution data in a hybrid infrastructure, independently of bandwidth. Also, the experiment demonstrates that split the input in lower chunk sizes than 32 MB, like 16 MB can produce a terrible performance in this scenario. The drastic increase in the time execution instead of a job time decrease has as basis the false supposition of that the division of machines and data by the half (distributed in Cloud and DG) could represent a decreasing at half time in hybrid environments. In fact, this possibility is incorrect because, in the hybrid infrastructures, we must also consider the heterogeneity and volatility factors. However, this scenario can be evaluated in the other manner for understanding what occurs in fact with chunk size related to the bandwidth variation.

Figure 5.18 evaluates the impact of bandwidth with the basis in the normalization of job executions along these experiments in medium-scale. The aim is to identify patterns in the job execution related to the bandwidth and the chunk size. The y-axis measures the time of parallel tasks in a DG environment with normalized value in units for each execution with different chunk size. The x-axis measures the bandwidth.

The impact is linear and descending with a small inclination in the time from range 10 Mbps to 150 Mbps (in other words, increase 15 times) produces a decrease of 10% in the execution time. Similar behavior occurs from range 10 Mbps to 300 Mbps (in other words, increase 30 times) with a decrease 15% to 30% for the chunk size of 64 and 32 MB respectively. Although this results can seem fascinating, it in fact not sufficient for producing a improve to the job execution time as demonstrates the Figure 5.16. This means that the bandwidth is not the main factor to decrease the job time, but other factors like amount of machines, data distributions, number of volatile nodes, and so on. Thus, an approach based only on the control of chunk size to reduce the time of job execution can not obtain the desired results.

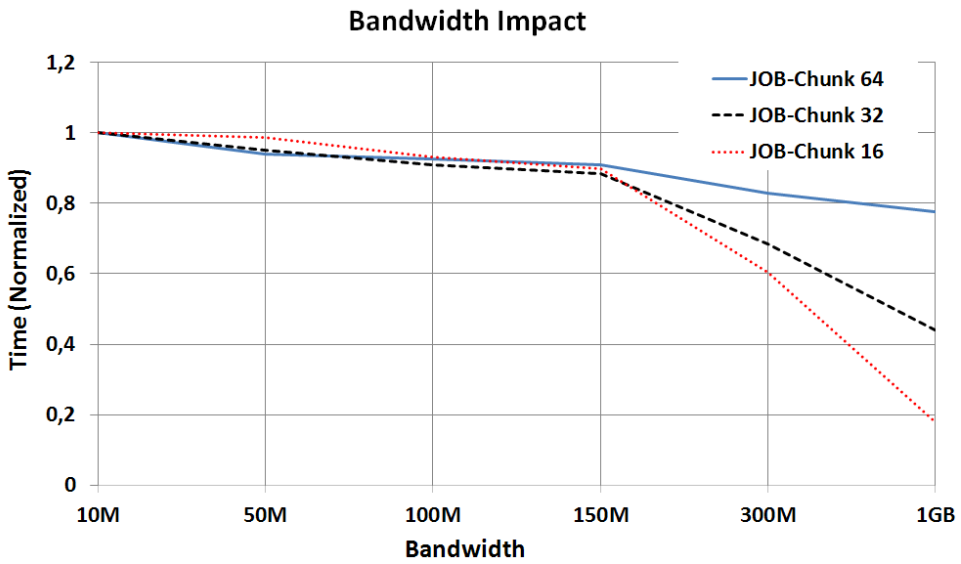


Figure 5.18: Impact of Bandwidth in a Job Execution

The previous experiments demonstrate that is needed the use of a technique for data distribution in hybrid environments. The next analysis is based in the earlier experiment of Figure 5.17 for 64 MB and 32 MB chunk size. The aims are to consolidate the earlier observations in low-scale and demonstrates that the using of Φ relation is a feasible strategy for the setup of machines and data distribution in hybrid environments.

Figure 5.19 presents the time execution of a job in a hybrid environment with a chunk size of 64 MB and 32 MB to a volatile environment with 256 machines. The Cloud job has the same runtime time of 1,618 seconds. The y-axis measures the concurrent tasks in a DG environment in seconds for each execution with different chunk size. The x-axis measures the bandwidth. The application processes a workload of 2304 and 4608 chunks with 64 MB and 32 MB chunk size, respectively.

It is possible to identify that the data distribution has a lower time than 1,618 seconds when the machine numbers are 461 and 922 (in the executions with a chunk size of 64 MB and 32 MB, respectively). The Φ relation produces a beneficial effect of decreasing the data transfers between machines from 39.1% to 57.14% in the worst and best case, respectively. The data transference decrease added to bandwidth impact produces the result needed to provide a proper load balancing to become feasible the use of hybrid infrastructure. These results demonstrate that the DG executes a greater number of local tasks with the relation Φ than without it and, thus, minimizes the data transfers in all the system.

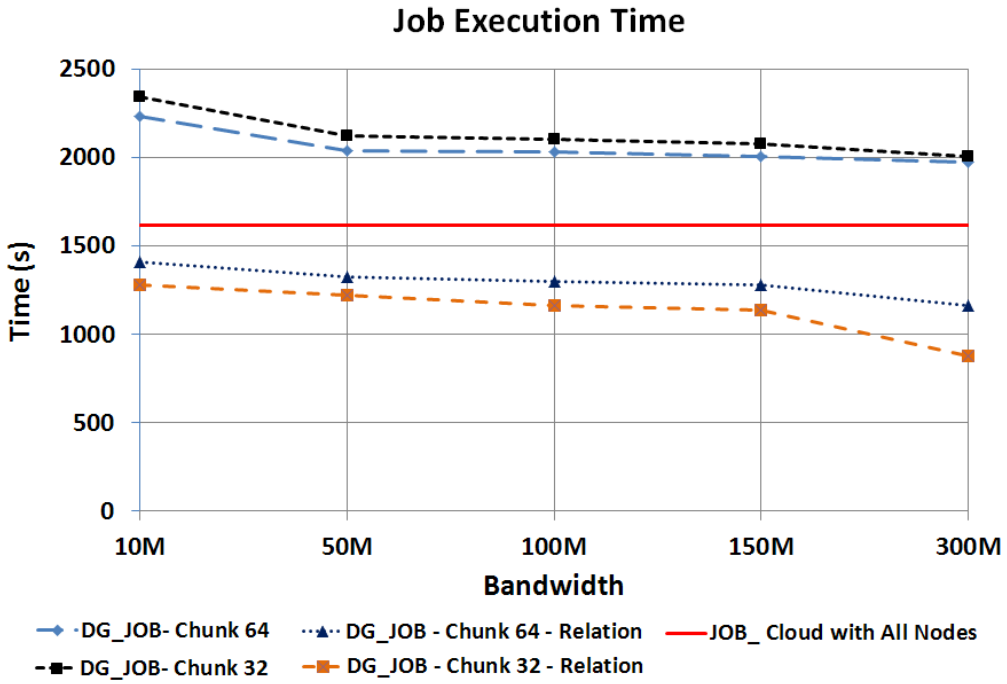


Figure 5.19: Data Distribution Relation

Figure 5.20 presents several DG job executions the aim verify behavior in high-scale. The executions run 64 MB chunk size. Table 5.6 shows the number of machines and workloads. The experiment in a single Cloud runs in 901 seconds on 2000 machines and a workload of 9088 chunks. The red line in the Figure 5.20 indicates the time of this execution in high-scale. The y-axis measures the parallel tasks in a DG environment in seconds for each execution and the x-axis measures the bandwidth. The different jobs are represented with colors indicates in the caption. The different jobs are represented with colors indicated in the caption of the Figure, and each Job shows their Φ relation to Cloud and DG respectively. The Φ_{DG} DG relation is preserved according to estimation earlier.

The experiment in high scale demonstrates behavior different from operations of medium and low-scale. This behavior is related to a large number of machines in the network and administrative overhead to manage the data transference over the Internet. The execution is possible in some cases from 50 MB to 300 MB bandwidth, but the best performance occurs with 300 MB bandwidth. However, a thorough cost analysis can determine other data distributions where the border line from cloud execution may be exceeded without the loss of the solution quality. The results indicate a cost decrease from 10% to 50% in the worst case and the best case respectively, in comparison with a single Cloud execution.

Table 5.6: Relation workload vs. number of machines

Relation (Φ_{Ch}/Φ_{DG})	**#Chunks (C/DG)**	**Machine (C/GD)**
DG_JOB (4.04/4.54)	7270/1818	1800/400
DG_JOB (4.03/2)	7270/1818	1800/909
DG_JOB (3.88/2)	7000/2088	1800/1044
DG_JOB (3.88/1.91)	7000/2088	1800/1800
DG_JOB (3.88/1.02)	7000/2088	1800/2044
DG_JOB (2.97/2)	5000/4088	1680/2044

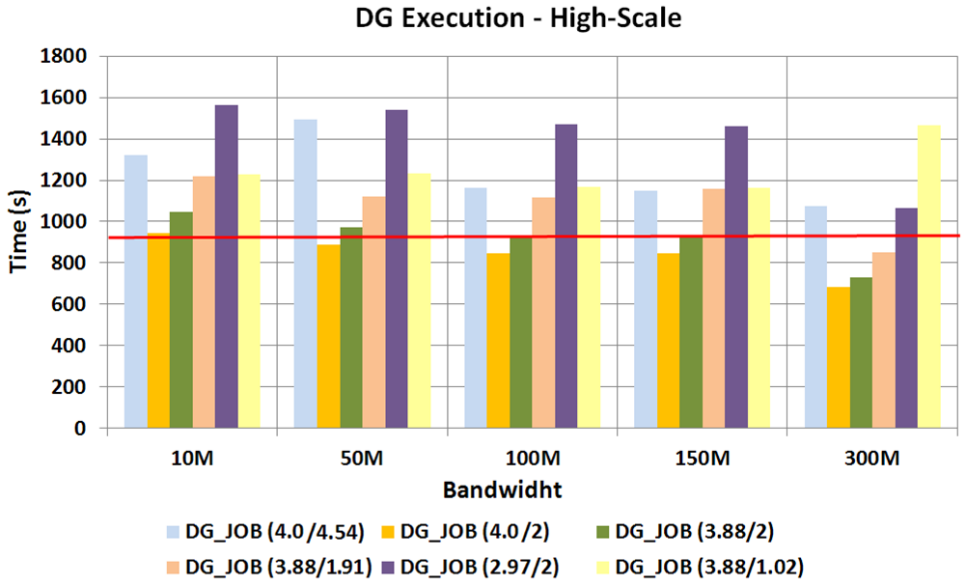


Figure 5.20: Simulation in High-Scale

This scenario demonstrates that a little workload variability can produce behavior changes in DG environment. Thus, hybrid environments with high workloads have the necessity of orchestrators to control of data transference and the task synchronization what justify our approach in the SMART Platform.

5.4 Discussion and Conclusions

This Chapter has shown real-world experiments and simulation experiments. These analyses evaluate two distinct scenarios, one uncoupled scenario deployed in the Mircosoft Azure and another semi-coupled scenario in hybrid environments with BIGhybrid simulator. The first evaluates only the execution of Dispatcher module in an application similar to the Twitter scenario. The second also evaluates the strategies to deploy of hybrid environments in Big Data analytics through of a synthetic application from Yahoo Cluster. Both scenarios represent Big Data applications in geographically distributed environments.

The works (JAYALATH; STEPHEN; EUGSTER, 2014) and (TUDORAN et al., 2014) are a Cloud-to-Cloud deployment, both provide support to the data transfers to reduce time execution in *MapReduce* jobs. The first focus on the cost of performance and the second observes I/O throughput and the environment capacity. In contrast, our work proposes a solution in a hybrid approach to applications real-time and batch. Also, it designs a platform in the Lambda architecture for solving this problem and implements built-in mechanisms to avoid data moving without needed.

The workflow of processing in a hybrid infrastructure for applications of the sentiment analysis show the opportunities to improve the execution of the Big Data engines in function of the found bottlenecks. Tudoran et.al. (TUDORAN et al., 2014) discusses that there is a potential to overlap computation and communication of messages and shortness of the time to compute a group of events in stream processing. The authors propose strategies such as *Stream&Compute* (SC) and *Copy&Compute* (CC) which were deployed in our experiment. However, they did not work well for this scenario, as can be seen in the Figure 5.7. A possible strategy is to create a dynamic scheduler to provide execution

priority and avoid the delay in the task flow.

The results of the simulated experiments establish strategies for using hybrid infrastructures based on behavior patterns of job executions in Cloud and DG described in Table 5.7. The strategies are a set of parameters and recommendations which can help in the setup of hybrid environments geographically distributed. The parameters indicate relations between execution time and volatile nodes to determine the number of unstable machines. It establishes a method for determining the resource allocation to Cloud and DG in hybrid environments. They point out ways to data distributing related to the number of Cloud and DG resources to produce a load balance. Also, the parameters establish the best chunk size in agreement with bandwidth of the communication channel and provide a method to find the best cost/benefit for a lower price of Cloud resources.

Table 5.7: Strategies and Achieved Aims

Parameters	Strategies	Aim
Volunteer hosts	$5\% < \xi < 25\%$	Achieved
Cloud resources	$\Phi_{Ch} = \frac{Ck_{Ch}}{Ck_c} * \frac{W_{Ch}}{\sum_i^M Vm_c(i)}$	Achieved
DG resources	$\Phi_{DG} = \frac{Ck_{DG}}{Ck_c} * \frac{W_{DG}}{\sum_i^M M_{DG}(i)}$	Achieved
Relation Φ between Cloud and DG	$1 \leq \Phi \leq \Phi_C$	Achieved
Channel communication	Chunk size 32 MB to 64 MB	Achieved
Data load balance	$\Phi_{Ch} \leq \Phi_C \Longleftrightarrow \Phi_{DG} \leq \Phi_C$	Achieved
Cost model	$C(J_i, C^{k,n}) = t_r * (J_i, C^{k,n}) * \sum_{m=1}^{\beta} n * R_{c_m}$	Need evaluation
I/O interference	Defining an optimal ρ	Open

The results point out that the hybrid environments show an operation continuity in an environment with until 25% of unstable nodes in the worst case without lost performance and with a replica number of 3 replicas which is the Hadoop standard. In contrast, the work of Lin (LIN et al., 2010) argues that one machine with an unavailability rate of 40% must have eleven replicas to achieve an availability rate of 99.99% for a single data block in HDFS. The experiments point a significant performance loss with rates 35% of node instability. Thus, with an unavailability rate of 40%, the hybrid environment could not operate well.

An optimal allocation is difficult to achieve, and so strategies to obtain an approximation can be accepted as argues Mashayekhy *et.al.* (MASHAYEKHY; NEJAD; GROSU, 2014). The parameters Φ_{Ch} and Φ_{DG} to find Cloud and DG resources establish a number of machines adequated to achieve acceptable performance. Also, these settings help inexperienced users to locate the number of Cloud and DG machines without a previous knowledge of the CSP infrastructure, which can be considered a significant contribution of Thesis.

Several authors like Mansouri *et.al.* (MANSOURI; TOOSI; BUYYA, 2013), Tudoran *et.al.* (TUDORAN et al., 2014), Balaji *et.al.* (PALANISAMY; SINGH; LIU, 2015) and others argue that the users tend to choose resources based on their workload peak, and the systems must find the optimal chunk placement depending on the user's needs. In contrast, the relation Φ between Cloud and DG can help the users to find resources adjusted to their workloads. Also, the recommendation of chunk size in channel communication can contribute to avoiding excessive data movement in Big Data applications over hybrid infrastructures.

The relation among workload, number of machines and load balance can be considered a significant contribution for providing data load balance and decreasing the data

transfers between machines from 39.1% to 57.14% in the worst and best case, respectively. These values are compatible with the work of Tudoran *et.al.* (TUDORAN et al., 2014) which achieved a decreasing of 50% with a relative error from 10% to 15%. The relation where $1 \leq \Phi \leq \Phi_C$ is particularly stable in all scales and also independent of bandwidth.

The reduction of chunk size can produce a little performance increase in Cloud with a 1 Gbps bandwidth. However, a decrease in the chunk size from 64MB to 16 MB generates a performance loss from 18.37% to 34.16% for executions with 300 Mbps and 50 Mbps respectively. These results demonstrate that data reduction results in a different behavior compared to a hybrid environment with the homogeneous environment. Thus, with the base on this results, the best chunk size for medium and high-scale is 64MB and on low-scale also is possible to use 32MB chunk sizes.

The use of uncoupled and semi-coupled scenarios enable to generalize this architecture design to a vast spectrum of Big Data implementations with data dispersal around the different sites, in particular, to geographically distributed datasets. These results represent a significant contribution for deployment of hybrid infrastructures in Big Data Analytics applications. The environment simulation was the path chosen for finding answers to the questions raised in this Thesis. However, some strategies as cost model and I/O interference need a careful study still in function of the high specificity of the theme. The cost model was implemented as part of the algorithm to data distribution in the Dispatcher module, as will be shown below, but needs of a particular evaluation. Thus, the conducting of this evaluation enables to consider achieved the aims of find strategies to make feasible the using of Big Data analytics applications in hybrid infrastructures.

5.4.1 Dispatcher Algorithm

This subsection presents the algorithm used in the previous experiments for the resource allocation in the Dispatcher module, with adaptations to the estimate of the reduction rating of cost. The reduction rating is a value range of (1,0] which is a user estimation in percentage. The user needs to provide information about their application, the chunk size and an initial estimate of the number of machines in Cloud. A minimal reference of hardware is needed, such as the number of slots that can be related with the thread number in a processor and a little execution of *MapReduce* with same chunks to determine the task cost of *Map* and *Reduce* phases, necessary to setup of simulator.

The Algorithm 5.4.1 computes the number of workers to Cloud and DG, and provide the amount of data for each environment with the base on an initial estimate of cost reduction provided by the user. This process can be automatized for the Dispatcher to collect information over different CSPs, seek the best price and then, to offer a lower cost for the user.

In comparison with the works of (OHNAGA; AIDA; ABDUL-RAHMAN, 2015) and (KHAN et al., 2016) which need of time reference for all executions in Dispatcher-Resources Algorithm this information is not necessary to determine the number of machines and workers in a hybrid environment. Thus, the users can evaluate a possible configuration before allocating any resource. Also, as the BIGhybrid simulator can provide a previous execution for adjusting the parameters, the user do not need to do an allocation in Cloud to test their configuration. This tool is an important characteristic of our work.

Algorithm 5.4.1 Dispatcher-Resources Algorithm - Pseudo-code

Require: $E \leftarrow$ reduction rating ▷ reduction rating in %
Require: $S \leftarrow$ free-slots number ▷ Machine free-slots
Require: $D_{in} \leftarrow$ Input size in MB ▷ Data input
Require: $C_C \leftarrow$ Chunk size in MB ▷ Chunk size to Cloud execution
Require: $C_{DG} \leftarrow$ Chunk size in MB ▷ Chunk size to DG execution
Require: $H_C \leftarrow$ Host estimation ▷ User host estimation in Cloud
Require: $C_S = 64$ ▷ Standard Chunk size
Require: Setup Phase of simulator
Require: Get BIGhybrid parameters in Calibration Phase

$D_T = \lceil \frac{D_{in}}{C_C} \rceil$
$U_e = 1 - E$
$\Phi = \frac{D_T}{H_C}$
if $\lfloor \Phi \rfloor == 1$ **then**
    $W_C = \lceil \frac{D_T * U_e}{\Phi} \rceil$ ▷ update number of workers in Cloud
    $W_{DG} = \lceil \frac{D_T}{\Phi} - W_C \rceil$ ▷ update number of workers in DG
else
    $\Phi_C = \lfloor \Phi * U_e * \frac{C_C}{C_S} \rfloor$
    $\Phi_{DG} = \lfloor \Phi_C * U_e * \frac{C_C}{C_{DG}} \rfloor$
    $D_C = \lceil D_T * U_e \rceil$ ▷ Data in Cloud
    $D_{DG} = \lceil D_T - D_C \rceil$ ▷ Data in DG
    $W_C = \lceil \frac{D_C}{\Phi_C} \rceil$ ▷ update number of workers in Cloud
    $W_{DG} = \lceil \frac{D_G}{\Phi_{DG}} \rceil$ ▷ update number of workers in DG
return

5.5 Comparison between SMART Platform and Related Work

Table 5.8 shows the SMART in comparison with Related Work. The platform enables the use of geographically distributed environments in hybrid infrastructures with Lambda architecture support. The main strategies considers cost, failure recovery, handles network overheads, distribute data for environment capacities, preserves replication mechanisms, minimizes data transfers, provide an orchestrator system, has buit-in a group strategy to data and task distribution and enable *barrier-free*.

These features are added to strategies to data distribution and definition of the number of machines from the system. The SMART platform definition provides a robust solution to the hybrid infrastructure specified in this Thesis. The Dispatcher algorithm helps users, without a previous knowledge of hardware infrastructure, to define the number of machines and data distribution. This property is an important competitive differential in comparison with the other works.

The BIGhybrid simulator use enables to the users an easy design of the hybrid infrastructures and a visualization of execution with little initial information for the simulation deploying. Thus, this avoids definition mistakes due to user inexperience and false premisses to data and resource allocations on the infrastructure.

Table 5.8: SMART versus Related Work

			Infrastructure							Strategies												Proc	
Area	Author	Proposal	Geo-Distributed	Cloud Infrastructure	Desktop Grid	Cluster	Multi-Cloud	Hybrid System	Lambda Architecture	Consider Cost	Failure Recovery	Network Overhead	I/O Throughput	Enviroment Capacity	Evaluates Storage	P/S or M_Queue	Replication	Minimize Transfers	Orchestration System	Processing in Memory	Group Strategy	Barrier Free	Batch	Event Stream
Frameworks	(EKANAYAKE et al., 2010)	Twister				X										X	X			X			X	
	(MANSOURI; TOOSI; BUYYA, 2013)	Brokering Alg.					X			X	X						X							
	(GUNARATHNE et al., 2013)	Twister4Azure		X												X	X			X		X	X	
	(LE et al., 2014)	SALSA		X										X					X					
	(JAYALATH; STEPHEN; EUGSTER, 2014)	G-MR	X				X			X		X											X	
	(TUDORAN et al., 2014)	GEO-DMS	X				X			X		X	X	X									X	
	(PALANISAMY; SINGH; LIU, 2015)	CURA		X						X														
Heterogeneous	(LIN et al., 2010)	Moon			X	X							X				X	X					X	
	(LU et al., 2012)	BitDew-MR			X						X		X				X	X				X	X	
	(KRISH; ANWAR; BUTT, 2014)	HATS				X						X	X	X	X								X	
	(ANJOS et al., 2015)	MRA++				X								X				X			X		X	
Hybrid System	(DING et al., 2011)	Hadoop Streaming				X														X				X
	(TUDORAN et al., 2014)	SC-CC		X										X									X	
	(ALEXANDROV et al., 2014)	FLINK				X			X		X	X				X				X	X		X	X
	(CLEMENTE-CASTELLÓ et al., 2015)	Hybrid IaaS Cloud					X				X												X	
	(TANG; HE; FEDAK, 2015)	HybridDFS		X	X			X		X	X	X		X	X		X					X	X	
	(VASILE et al., 2015)	HySARC^2					X			X											X			
	(GHAFARIAN; JAVADI, 2015)	Cloud-aware		X	X			X		X		X		X				X			X			
	(PHAM et al., 2016)	Cirus					X		X							X			X	X			X	X
	(Anjos, 2017)	**SMART**	X					X	X	X	X			X		X	X	X	X		X	X	X	X

5.6 Final Considerations

This Chapter enabled to evaluate the strategies for the using of hybrid infrastructures. In particular, the Dispatcher module was evaluated in a real world experiment wich con-

siders the SMART architecture. The results allow concluding that it is feasible to execute the Big Data analytics in hybrid infrastructures such as Cloud and Desktop Grid.

The strategies create to deployment in hybrid infrastructures are particularly compelling due to producing a load balance between the Cloud and DG environments oriented for reduction the data transfers without performance loss. Thus, with basis on the results of these experiments from this Chapter, it is possible to assume that these strategies enable to deploy the SMART platform in a hybrid infrastructure.

6 CONCLUSION AND FUTURE WORK

A huge volume of data is produced every day, from the information provided by social networks (such as Facebook, Instagram, Whatsapp, etc) or that generated by sensors on mobile devices, including Big Data applications like Google Searches. This deluge of data requires ever more computational resources to process the information more quickly.

This work establishes: i) a new platform called SMART which offers Big Data Analytics in a Lambda architecture within a hybrid infrastructure; ii) a simulator called BIGhybrid to be a toolkit for the study of Big Data Analytics in hybrid infrastructures. Its goal is to enable the user achieve the nearest configuration for Big Data applications into deployment in real-world environments; iii) evaluates the use of the Dispatcher module in the SMART platform and iv) defines strategies for the use of Desktop Grid and Cloud Computing in a geo-distributed environment within a hybrid infrastructure. In addition, to find the best data distribution strategies in this complex scenario for reducing the risks of trouble caused by common configuration mistakes.

In this chapter, there is a summary of the contributions made in previous chapters, as well as partner relationship developed in this work and the publications as result of Thesis development. Finally, our future works.

6.1 Contributions

The main contributions can be summarized as:

a) To provide a platform that can allow the deployment of Big Data analytics within hybrid infrastructures

A hybrid infrastructure environment was well defined. Then, the specifications gave origin to a new platform called SMART (Small & Medium-sized Enterprise Data Analytics in Real Time). The platform enables the Big Data analytics in a hybrid infrastructure. In comparison with other frameworks, this platform uses Cloud and DGs as its basic infrastructure and is embedded in a Lambda architecture for Big Data processing. The deploying approach is to use a geographically distributed system.

This platform is composed of several modules which depict a *Global Dispatcher* and *Global Aggregator*. This concept enables the use as an infrastructure for services that uses multiple data abstractions. The *Global Dispatcher* located outside the cloud is a layer that handles task assignments and the management of user-provided data. The layer decouples the data storage system and manages policies for the splitting and distributing data in accordance with each system. The working principle is similar to a publish/subscribe service in which the layer acts as a data producer that is afterwards consumed by workers. The *Global Aggregator* obtains data output from both systems and merges them to obtain the final data set.

b) To create a simulator as an analytical tool of hybrid infrastructures and for Big Data analytics

BIGhybrid is a toolkit for MR simulation in hybrid environments with a focus on Cloud and DG which was validated by means of real-world experiments. It can be used for evaluating scheduling strategies for MR applications in *hybrid infrastructures*. The idea behind the BIGhybrid simulator is to optimize *hybrid infrastructure* environments such as Cloud services with the available resources of a DG system. BIGhybrid is modular and built on top of the SimGrid framework.

The BIGhydrid simulator generates traces from each system to allow an individual or collective analysis to be conducted within the same time frame. The simulator enables several strategies to the data distribution and resource allocation of MR applications in *hybrid infrastructures*. In BIGhybrid, the *Global Dispatcher* can be either manual or automatic. In the manual version, the user defines a function for data distribution and a job configuration. These configurations explore the number of *Map* and *Reduce* tasks, input data, data size, chunk size and so on. The results of the *Global Aggregator* module are implemented as a single Reduce task after the last current Reduce task has been completed.

The statistical evaluations of the BIGhydrid simulator indicate a simulation with a relative degree of accuracy. The mean absolute percentage error ($\approxeq$ 5% in the worst case scenario for heterogeneous and homogeneous environments) shows that the simulator can be an efficient evaluative instrument for *hybrid infrastructures*.

c) To determine strategies for the use of hybrid infrastructures

The experiments are conducted in real-world and simulated environments. The real-world in Cloud had the aim to observe a behavior of real-time applications in an uncoupled scenario. The simulated environment had the goal of reproducing workloads from Big Data applications in a semi-coupled scenario.

The evaluations found behavior patterns which enable the deploy on hybrid environments in low, medium and high scale and establish a set of the strategies such as: i) The relation between execution time and the number of volatile nodes; ii) The relation of resource distribution in Cloud and DG; iii) The relation Φ between Cloud and DG; iv) The relation of data size in function of available bandwidth; v) The relationship between the load balance and number machines; vi) The cost related to Cloud and DG and vii) I/O interference. The results are summarized in Table 6.1.

Table 6.1: Strategies and Achieved Aims

Parameters	Strategies	Aim
Volunteer hosts	$5\% < \xi < 25\%$	Achieved
Cloud resources	$\Phi_{Ch} = \frac{Ck_{Ch}}{Ck_c} * \frac{W_{Ch}}{\sum_i^M Vm_c(i)}$	Achieved
DG resources	$\Phi_{DG} = \frac{Ck_{DG}}{Ck_c} * \frac{W_{DG}}{\sum_i^M M_{DG}(i)}$	Achieved
Relation Φ between Cloud and DG	$1 \leq \Phi \leq \Phi_C$	Achieved
Channel communication	Chunk size 32 MB to 64 MB	Achieved
Data load balance	$\Phi_{Ch} \leq \Phi_C \Longleftrightarrow \Phi_{DG} \leq \Phi_C$	Achieved
Cost model	$C(J_i, C^{k,n}) = t_r * (J_i, C^{k,n}) * \sum_{m=1}^{\beta} n * R_{c_m}$	Need evaluation
I/O interference	Defining an optimal ρ	Open

The results point out that the hybrid environments show an operation continuity in an

environment with until 25% of unstable nodes in the worst case without lost performance and with a replica number of 3 replicas which is the Hadoop standard.

The parameters Φ_{Ch} and Φ_{DG} to find Cloud and DG resources establish a number of machines adequated to achieve acceptable performance. Also, these settings help inexperienced users to locate the number of Cloud and DG machines without a previous knowledge of the CSP infrastructure, which can be considered a significant contribution of Thesis. The relation Φ between Cloud and DG can help the users to find resources adjusted to their workloads. Also, the recommendation of chunk size in channel communication can contribute to avoiding excessive data movement in Big Data applications over hybrid infrastructures.

The relation among workload, number of machines and load balance can be considered a significant contribution for providing data load balance and decreasing the data transfers between machines from 39.1% to 57.14% in the worst and best case, respectively. These values are compatible with works of literature.

The reduction of chunk size can produce a little performance increase in Cloud with a 1 Gbps bandwidth. However, a decrease in the chunk size from 64MB to 16 MB generates a performance loss from 18.37% to 34.16% for executions with 300 Mbps and 50 Mbps respectively. These results demonstrate that data reduction results in a different behavior compared to a hybrid environment with the homogeneous environment. Thus, with the base on this results, the best chunk size for medium and high-scale is 64MB and on low-scale also is possible to use 32MB chunk sizes.

The use of uncoupled and semi-coupled scenarios enable to generalize this architecture design to a vast spectrum of Big Data implementations with data dispersal around the different sites, in particular, to geographically distributed datasets. These results represent a significant contribution for deployment of hybrid infrastructures in Big Data Analytics applications. The results indicate a cost decrease from 10% to 50% in the worst case and the best case respectively, in comparison with a single Cloud execution. However, some strategies as cost model and I/O interference need a careful study still in function of the high specificity of the theme.

d) To find an accurate method of conducting an analysis of a hybrid infrastructure

An algorithm was implemented from the experiments to define the method of computation for the resource allocation in the Dispatcher module. Also, this algorithm has adaptations to estimate the cost in hybrid environments. However, the cost model needs a careful evaluation. The inexperienced users to locate the number of Cloud and DG machines without a previous knowledge of the CSP infrastructure.

The evaluations in the real-world experiments and simulated environment showed that using the *hybrid infrastructures* in Big Data applications can be feasible. The workflow of processing in a hybrid infrastructure for applications of the sentiment analysis show the opportunities to improve the execution of the Big Data engines in function of the found bottlenecks.

6.2 International Partners

Another contribution made is to establish a link between research groups that stimulate the cooperation and the development of a research network that is involved with similar problems.

This work can examine a model with various data sources, ranging from wireless sensor nodes to user interaction information in open environments, such as social networks, opendata information and other. The datasets consist of large corporate databases to broadcast media, where there is a clear need for standardization. The idea has evolved into a hybrid Big Data platform for applications processing in different domains. Because of this, some institutions have demonstrated interest in the SMART Platform. Thus, a "cooperation project" has been set up to achieve this vast domain. Several research institutes are involved, are summarized in able 6.2. It provides a list of the main researchers who are involved in work about the SMART platform.

Table 6.2: Collaborations

Institution	Group	Locality	Researcher
INRIA/ENS Lyon	LIP	Lyon, France	PhD. Gilles Fedak
INRIA/ENS Lyon	LIP	Lyon, France	PhD. Marcos Dias Assunção
TUB	DIMA	Germany	PhD. Volker Markl
JohnMoores University		Liverpool, England	PhD. Rubem Pereira
UnB	LASP	Brasilia, Brazil	PhD. João Paulo da Costa
UFRGS		Rio Grande do Sul, Brazil	PhD. Edison Pignaton
Belarusian State University		Minsk, Belarus	PhD. Tatiana Galibus
Danube University Krems		Austria	PhD. Thomas J. Lampoltshammer
Chinese Academy of Sciences	CNIC	Beijing, China	PhD. Haiwu HE

6.3 Publications

The publications of this Thesis are:

Journal:

Anjos, Julio C.S., Carrera, Iván, Kolberg, Wagner, Tibola, Andre Luis, Arantes, Luciana B., Geyer, Claudio R., ***MRA++: Scheduling and Data Placement on MapReduce for Heterogeneous Environments***, Future Generation Computer Systems Vol.42(0), 22–35, jan 2015

Anjos, Julio C. S., Fedak, Gilles, Geyer, Claudio F. R.: ***BIGhybrid: a simulator for MapReduce applications in hybrid distributed infrastructures validated with the Grid5000 experimental platform***, Concurrency and Computation: Practice and Experience 28(8), 2416–2439, June 2016, cpe.3665

Events:

Anjos, Julio C. S., Fedak, Gilles, Geyer, Claudio F.R.: ***BIGhybrid – A Toolkit for Simulating MapReduce in Hybrid Infrastructures***, Computer Architecture and High Performance Computing Workshop (SBAC-PADW), 2014 International Symposium on, UPMC - University Pierre et Marie Curie, 132–137, Oct 2014

Simonet, Anthony, Anjos, Julio, Fedak, Gilles, He, Haiwu, Tang, Bing, Lu, Lu, Jin, Hai, Shi, Xuanhua, Moca, Mircea, Silaghi, Gheorghe, Cheich, Asma Ben, Abbes, Heithem: ***D3-MapReduce: Towards MapReduce for Distributed and Dynamic Data Sets***, International Conference on Big Data Intelligence and Computing (DataCom 2015), 1 edition, IEEE, 20–30, Dec 2015

Anjos, Julio Cesar Santos, Filho, Bruno Reckziegel, Barros, Junior F., Schemmer, Raffael B., Geyer, Claudio, Matte, Ursula: ***Genetic Mapping of Diseases through Big Data Techniques***, Proceedings of the 17th International Conference on Enterprise Information Systems, 279–286, 2015

Anjos, Julio C. S., Assuncao, Marcos D., Bez, Jean, Geyer, Claudio F. R., de Freitas, Edison Pignaton, Carissimi, Alexandre, Costa, João Paulo C. L., Fedak, Gilles, Freitag, Felix, Markl, Volker, Fergus, Paul, Pereira, Rubem: ***SMART: An Application Framework for Real Time Big Data Analysis on Heterogeneous Cloud Environments***, Computer and Information Technology; Ubiquitous Computing and

Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/ DASC/PICOM), 2015 IEEE International Conference on, 199–206, Oct 2015

Galibus, Tatiana, Anjos, Julio C.S., de Freitas, Edison Pignaton, Geyer, Cláudio F. Resin, Fedak, Gilles, Jr., Rafael Timóteo de Sousa, Costa, João Paulo C. L., Pereira, Rubem, Fergus, Paul, Zaleski, Anton, Vissi, Herman, Markl, Volker: ***Security Framework for Distributed Data Processing***, International Conference on Pattern Recognition and Information Processing (PRIP'16), Belarusian State University, 13th edition, October 2016

da Silva, Veith Alexandre, dos, Anjos Julio C.S., Pignaton, de Freitas Edison, J., Lampoltshammer Thomas, F., Geyer Claudio: **Strategies for Big Data Analytics through Lambda Architectures in Volatile Environments**, IFAC-PapersOnLine 49(30), 114–119, 2016, 4th IFAC -Symposium on Telematics Applications TA- 2016 Porto Alegre, Brasil, 6—9 November 2016

6.4 Future Works

Several activities are needed to build the SMART platform in a real environment. First, it is necessary to conclude the cost model evaluations and the I/O interference of the strategies defined in this Thesis. The algorithms to *Global Aggregator* need to be defined and implemented. The *Global Dispatcher* stage need to be implemented and adjusted with the *Global Collector* and the Big Data Engines.

In particular, a possible strategy to apply on the Dispatcher module is to create a dynamic scheduler to provide an execution priority and avoid the delay in the task flow as shown earlier. The mechanism of decision engine can have some strategies of this Thesis deployed as a decision heuristic. The security module can have its algorithms studied for an increase of performance. First studies are in the Annex.

In the *Global Collector* module, the repository services need to be integrated with the SMART API and create a REST interface to sensing communication in low level. One study of information quality provided by sensors must enable the use of data from these devices grouped on zones in urban sensing to use with Big Data analysis in Smart Cities applications. This study is important for future project submissions with some partners.

The central monitoring and the end user interface must be specified by a multidisciplinary team, including designers and graphic computation specialists. Finally, the storage mechanism must be evaluated to include I/O interference and an evaluation of container techniques. Currently, we are writing an article to a Journal about the Thesis conclusions.

REFERENCES

ALEXANDROV, A. et al. The Stratosphere platform for big data analytics. **VLBD Journal**, [S.l.], v.23, n.6, p.939–964, 2014.

AMAZON. **Amazon EC2 Instances** . Available from Internet: <https://aws.amazon.com/ec2/instance-types/>.

ANDERSON, D. P. BOINC: a system for public-resource computing and storage. In: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2004. p.4–10. (GRID '04).

ANJOS, J. C. et al. MRA++: scheduling and data placement on mapreduce for heterogeneous environments. **Future Generation Computer Systems**, [S.l.], v.42, n.0, p.22–35, jan 2015.

ANJOS, J. C. S. et al. Estratégias para Uso de MapReduce em Ambientes Heterogêneos. In: CLCAR Conferência Latinoamericana de Computación de Alto Rendimiento. **Proceedings...** Evangraf, 2010. v.1, n.1, p.322–325.

ANJOS, J. C. S. et al. Addressing Data-Intensive Computing Problems with the Use of MapReduce on Heterogeneous Environments As Desktop Grid on Slow Links. In: Proceedings of the 2012 13th Symposium on Computing Systems, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2012. p.148–155. (WSCAD-SSC '12).

ANJOS, J. C. S. et al. SMART: an application framework for real time big data analysis on heterogeneous cloud environments. In: Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on. **Proceedings...** [S.l.: s.n.], 2015. p.199–206.

ANJOS, J. C. S.; FEDAK, G.; GEYER, C. F. BIGhybrid – A Toolkit for Simulating MapReduce in Hybrid Infrastructures. In: Computer Architecture and High Performance Computing Workshop (SBAC-PADW), 2014 International Symposium on, Paris, FR. **Proceedings...** [S.l.: s.n.], 2014. p.132–137.

ANTONIU, G. et al. Scalable Data Management for Map-Reduce-based Data-Intensive Applications: a view for cloud and hybrid infrastructures. **Int. Journal of Cloud Computing**, [S.l.], v.2, p.150–170, Feb. 2013.

ASSUNCAO, M. D. et al. Big Data computing and clouds: trends and future directions. **Journal of Parallel and Distributed Computing**, [S.l.], v.79–80, p.3–15, 2015. Special Issue on Scalable Systems for Big Data Management and Analytics.

BETHENCOURT, J.; SAHAI, A.; WATERS, B. Ciphertext-Policy Attribute-Based Encryption. In: Security and Privacy, 2007. SP '07. IEEE Symposium on. **Proceedings...** [S.l.: s.n.], 2007. p.321–334.

BICAK, M. **MaxxPI System Bench**. [S.l.]: MaxxPI.net, 2017. Available in http://www.maxxpi.net, accessed in 2017, January.

BUX, M.; LESER, U. DynamicCloudSim: simulating heterogeneity in computational clouds. **Future Generation Computer Systems**, Amsterdam, The Netherlands, The Netherlands, v.46, n.C, p.85–99, May 2015.

BUYYA, R.; RANJAN, R.; CALHEIROS, R. N. InterCloud: utility-oriented federation of cloud computing environments for scaling of application services. In: Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I, Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2010. p.13–31. (ICA3PP'10).

BV, B. D. S. **Storgrid EFSS: secure enterprise file sharing software**. Available from Internet: http://www.storgrid.com/.

CALHEIROS, R. N. et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. **Software: Practice and Experience**, [S.l.], v.41, n.1, p.23–50, 2011.

CASANOVA, H. et al. Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms. **Journal of Parallel and Distributed Computing**, [S.l.], v.74, n.10, p.2899–2917, June 2014.

CÉRIN, C.; FEDAK, G. (Ed.). **Desktop Grid Computing**. 1st.ed. [S.l.]: CRC Press, 2012. 362p. (Numerical Analysis and Scientific Computing).

CERN. **European Organization for Nuclear Research**. Available in <http://public.web.cern.ch/> Accessed in October 2016.

CHAUHAN, A. et al. **Apache Flink**. [S.l.]: The Apache Software Foundation, 2015.

CHEIKH, A. B.; ABBES, H.; FEDAK, G. Towards Privacy for MapReduce on Hybrid Clouds Using Information Dispersal Algorithm. In: Data Management in Cloud, Grid and P2P Systems. **Proceedings...** Springer International Publishing, 2014. p.37–48. (Lecture Notes in Computer Science, v.8648).

CHEN, C. P.; ZHANG, C.-Y. Data-intensive applications, challenges, techniques and technologies: a survey on big data. **Information Sciences**, [S.l.], v.275, p.314–347, 2014.

CHEN, M.; MAO, S.; LIU, Y. Big Data: a survey. **Mobile Networks and Applications**, [S.l.], v.19, n.2, p.171–209, April 2014.

CHEN, Y. **Workload-Driven Design and Evaluation of Large-Scale Data-Centric Systems**. 2012. 151p. PhD Thesis — Electrical Engineering and Computer Sciences.

CHEN, Y. et al. The Case for Evaluating MapReduce Performance Using Workload Suites. In: IEEE 19th Int. Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems, Singapore. **Proceedings...** IEEE Computer Society, 2011. p.390–399. ((MASCOTS)).

CLEMENTE-CASTELLÓ, F. J. et al. Enabling Big Data Analytics in the Hybrid Cloud Using Iterative MapReduce. In: Utility and Cloud Computing - UCC, 2015 IEEE/ACM 8th International Conference on . **Proceedings...** IEEE Computer Society, 2015. p.290–299.

COSTA, F.; SILVA, L.; DAHLIN, M. Volunteer Cloud Computing: mapreduce over the internet. In: Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on. **Proceedings...** [S.l.: s.n.], 2011. p.1855–1862.

DEAN, J.; GHEMAWAT, S. MapReduce - A Flexible Data Processing Tool. **Communications of the ACM**, New York, NY, USA, v.53, n.1, p.72–77, 2010.

DELAMARE, S. et al. SpeQuloS: a qos service for bot applications using best effort distributed computing infrastructures. In: Proceedings of the 21th international symposium on High-Performance Parallel and Distributed Computing, New York, NY, USA. **Proceedings...** ACM, 2012. p.173–186. (HPDC '12).

DING, M. et al. More Convenient More Overhead: the performance evaluation of hadoop streaming. In: Proceedings of the 2011 ACM Symposium on Research in Applied Computation, New York, NY, USA. **Proceedings...** ACM, 2011. p.307–313. (RACS '11).

DOAN, A.; HALEVY, A.; IVES, Z. **Principles of Data Integration**. 225 Wyman Street, Waltham, MA 02451, USA: Elsevier Publishers B. V., 2012. 497p. v.1, n.1.

DOBRE, C.; XHAFA, F. Parallel Programming Paradigms and Frameworks in Big Data Era. **Int. Journal of Parallel Programming**, [S.l.], v.42, n.5, p.710–738, 2014.

DUQUENNOY, S.; GRIMAUD, G.; VANDEWALLE, J.-J. The Web of Things: interconnecting devices with high usability and performance. In: Embedded Software and Systems, 2009. ICESS '09. International Conference on. **Proceedings...** [S.l.: s.n.], 2009. p.323–330.

EKANAYAKE, J. et al. Twister: a runtime for iterative mapreduce. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, New York, NY, USA. **Proceedings...** ACM, 2010. p.810–818. (HPDC '10).

EKANAYAKE, J.; PALLICKARA, S.; FOX, G. MapReduce for Data Intensive Scientific Analyses. In: IEEE Fourth Int. Conference on eScience, Indianapolis (Indiana) USA. **Proceedings...** [S.l.: s.n.], 2008. p.277 – 284. (eScience '08).

FEDAK, G. et al. XtremWeb: a generic global computing system. In: Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on, Brisbane, Australia. **Proceedings...** [S.l.: s.n.], 2001. p.582–587.

FEDAK, G.; HE, H.; CAPPELLO, F. BitDew: a programmable environment for large-scale data management and distribution. In: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, Piscataway, NJ, USA. **Proceedings...** IEEE Press, 2008. p.45:1–45:12. (SC '08).

FEDAK, G.; HE, H.; CAPPELLO, F. BitDew: a data management and distribution service with multi-protocol file transfer and metadata abstraction. **Journal of Network and Computer Applications**, [S.l.], v.32, n.5, p.961–975, 2009. Next Generation Content Networks.

GHAFARIAN, T.; JAVADI, B. Cloud-aware data intensive workflow scheduling on volunteer computing systems. **Future Generation Computer Systems**, [S.l.], v.51, p.87–97, Oct. 2015.

GHIT, B. et al. Balanced Resource Allocations Across Multiple Dynamic MapReduce Clusters. **SIGMETRICS Perform. Eval. Rev.**, New York, NY, USA, v.42, n.1, p.329–341, June 2014.

GHODSI, A. et al. Dominant Resource Fairness: fair allocation of multiple resource types. In: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, Berkeley, CA, USA. **Proceedings...** USENIX Association, 2011. p.323–336. (NSDI'11).

GOYAL, V. et al. Attribute-based Encryption for Fine-grained Access Control of Encrypted Data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, New York, NY, USA. **Proceedings...** ACM, 2006. p.89–98. (CCS '06).

GROZEV, N.; BUYYA, R. Performance Modelling and Simulation of Three-Tier Applications in Cloud and Multi-Cloud Environments. **The Computer Journal**, [S.l.], v.58, n.1, p.1–22, 2015.

GUNARATHNE, T. et al. Scalable parallel computing on clouds using Twister4Azure iterative MapReduce. **Future Generation Computer Systems**, [S.l.], v.29, n.4, p.1035–1048, 2013. Special Section: Utility and Cloud Computing.

GUNARATHNE, T.; QIU, J.; GANNON, D. Towards a Collective Layer in the Big Data Stack. In: Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, Chicago, IL, USA. **Proceedings...** [S.l.: s.n.], 2014. p.236–245.

HEINTZ, B. et al. End-to-end Optimization for Geo-Distributed MapReduce. **Cloud Computing, IEEE Transactions on**, [S.l.], v.PP, n.99, p.1–14, Sept. 2014.

HEY, T.; TANSLEY, S.; TOLLE, K. M. (Ed.). **The Fourth Paradigm: data-intensive scientific discovery**. 1.ed. [S.l.]: Microsoft Research, 2009. 252p.

HINDMAN, B. et al. Mesos: a platform for fine-grained resource sharing in the data center. In: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, Berkeley, CA, USA. **Proceedings...** USENIX Association, 2011. p.295–308. (NSDI'11).

HUANG, S. et al. The HiBench Benchmark Suite: characterization of the mapreduce-based data analysis. In: AGRAWAL, D.; CANDAN, K.; LI, W.-S. (Ed.). **New Frontiers in Information and Software as Services**. [S.l.: s.n.], 2010. p.41–51. (Lecture Notes in Business Information Processing).

INRIA; CNRS. **Grid5000**. [S.l.: s.n.], 2016. Available from Internet: <https://www.grid5000.fr>.

IOSUP, A.; YIGITBASI, N.; EPEMA, D. On the Performance Variability of Production Cloud Services. In: Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on. **Proceedings...** [S.l.: s.n.], 2011. p.104–113.

JAIN, R. **The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling**. Littleton (Massachusetts), USA: John Wiley & Sons Inc, 1991. 685 ppp. (Wiley professional computing).

JAVADI, B. et al. Discovering Statistical Models of Availability in Large Distributed Systems: an empirical study of seti@home. **IEEE Transactions on Parallel and Distributed Systems**, Los Alamitos, CA, USA, v.22, n.11, p.1896–1903, 2011.

JAVADI, B.; ZHANG, B.; TAUFER, M. Bandwidth Modeling in Large Distributed Systems for Big Data Applications. In: International Conference on Parallel and Distributed Computing, Applications and Technologies. **Proceedings...** IEEE Computer Society, 2014. p.21–27.

JAYALATH, C.; STEPHEN, J.; EUGSTER, P. From the Cloud to the Atmosphere: running mapreduce across data centers. **Computers, IEEE Transactions on**, [S.l.], v.63, n.1, p.74–87, Jan 2014.

JI, S.; LI, B. Wide area analytics for geographically distributed datacenters. **Tsinghua Science and Technology**, [S.l.], v.21, n.2, p.125–135, April 2016.

KHAN, M. et al. Hadoop Performance Modeling for Job Estimation and Resource Provisioning. **IEEE Transactions on Parallel and Distributed Systems**, [S.l.], v.27, n.2, p.441–454, Feb 2016.

KOHNE, A. et al. FederatedCloudSim: a sla-aware federated cloud simulation framework. In: Proceedings of the 2Nd International Workshop on CrossCloud Systems, New York, NY, USA. **Proceedings...** ACM, 2014. p.3:1–3:5. (CCB '14).

KOLBERG, W. et al. MRSG - A MapReduce simulator over SimGrid. **Parallel Computing**, Amsterdam, The Netherlands, The Netherlands, v.39, n.4-5, p.233–244, Apr. 2013.

KONDO, D. et al. Cost-benefit Analysis of Cloud Computing versus Desktop Grids. In: IPDPS '09 - Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2009. p.1–12.

KONDO, D. et al. The Failure Trace Archive: enabling comparative analysis of failures in diverse distributed systems. In: CCGRID - 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. **Proceedings...** IEEE Computer Society, 2010. p.398–407.

KRISH, K.; ANWAR, A.; BUTT, A. R. HATS: a heterogeneity-aware tiered storage for hadoop. In: Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, Chicago, IL, USA. **Proceedings...** [S.l.: s.n.], 2014. p.502–511.

LE, D.-H. et al. SALSA: a framework for dynamic configuration of cloud services. In: Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on. **Proceedings...** [S.l.: s.n.], 2014. p.146–153.

LIN, H. et al. MOON: mapreduce on opportunistic environments. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, New York, NY, USA. **Proceedings...** ACM, 2010. p.95–106. (HPDC '10).

LORETI, D.; CIAMPOLINI, A. A Hybrid Cloud Infrastructure for Big Data Applications. In: Proceedings of the 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conf on Embedded Software and Systems, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2015. p.1713–1718. (HPCC-CSS-ICESS '15).

LSST. **Data Management**. Available in <http://www.lsst.org/about/dm/> Accessed in October 2016.

LU, L. et al. Assessing MapReduce for Internet Computing: a comparison of hadoop and bitdew-mapreduce. In: Proceedings of the 2012 ACM/IEEE 13th Int. Conference on Grid Computing, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2012. p.76–84. (GRID '12).

MAKRIDAKIS, S. Time-Series Analysis and Forecasting: an update and evaluation. **International Statistical Review / Revue Internationale de Statistique**, [S.l.], v.46, n.3, p.pp. 255–278, 1978.

MANSOURI, Y.; TOOSI, A.; BUYYA, R. Brokering Algorithms for Optimizing the Availability and Cost of Cloud Storage Services. In: Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on. **Proceedings...** [S.l.: s.n.], 2013. v.1, p.581–589.

MARZ, N.; WARREN, J. . 1st.ed. [S.l.]: Manning Publications Co., 2015. 328p.

MASHAYEKHY, L.; NEJAD, M.; GROSU, D. A PTAS Mechanism for Provisioning and Allocation of Heterogeneous Cloud Resources. **Parallel and Distributed Systems, IEEE Transactions on**, [S.l.], v.PP, n.99, p.1–14, 2014.

MCKENNA, A. et al. The Genome Analysis Toolkit: a mapreduce framework for analyzing next-generation dna sequencing data. **Genome Research**, [S.l.], v.20, n.9, p.1297–1303, 2010.

MOCA, M.; SILAGHI, G.; FEDAK, G. Distributed Results Checking for MapReduce in Volunteer Computing. In: Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE Int. Symposium on. **Proceedings...** [S.l.: s.n.], 2011. p.1847–1854.

NEJAD, M.; MASHAYEKHY, L.; GROSU, D. Truthful Greedy Mechanisms for Dynamic Virtual Machine Provisioning and Allocation in Clouds. **Parallel and Distributed Systems, IEEE Transactions on**, [S.l.], v.26, n.2, p.594–603, Feb 2015.

NICOLAE, B. et al. BlobSeer: bringing high throughput under heavy concurrency to hadoop map-reduce applications. In: Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on. **Proceedings...** [S.l.: s.n.], 2010. p.1–11.

OHNAGA, H.; AIDA, K.; ABDUL-RAHMAN, O. Performance of Hadoop Application on Hybrid Cloud. In: Cloud Computing Research and Innovation (ICCCRI), 2015 International Conference on. **Proceedings...** IEEE Computer Society, 2015. p.130–138.

OSTERMANN, S. et al. GroudSim: an event-based simulation framework for computational grids and clouds. In: GUARRACINO, M. et al. (Ed.). **Euro-Par 2010 Parallel Processing Workshops**. [S.l.]: Springer Berlin Heidelberg, 2010. p.305–313. (Lecture Notes in Computer Science, v.6586).

PALANISAMY, B.; SINGH, A.; LIU, L. Cost-Effective Resource Provisioning for MapReduce in a Cloud. **Parallel and Distributed Systems, IEEE Transactions on**, [S.l.], v.26, n.5, p.1265–1279, May 2015.

PHAM, L. M. et al. An adaptable framework to deploy complex applications onto multi-cloud platforms. In: Computing Communication Technologies - Research, Innovation, and Vision for the Future (RIVF), 2015 IEEE RIVF International Conference on. **Proceedings...** [S.l.: s.n.], 2015. p.169–174.

PHAM, L. M. et al. CIRUS: an elastic cloud-based framework for ubilytics. **Annals of Telecommunications**, [S.l.], v.71, n.3, p.133–140, Apr. 2016.

ROCHA, F. d. G.; SENGER, H. **Análise de Ecalabilidade de Aplicações Hadoop/MapReduce por meio de Simulação**. 2013. 80p. Master Degree — .

ROLIM, C. O. et al. **Agent Technology for Intelligent Mobile Services and Smart Societies**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. p.1–10.

RYCHLY, M.; KODA, P.; SMRZ, P. Scheduling Decisions in Stream Processing on Heterogeneous Clusters. In: Complex, Intelligent and Software Intensive Systems (CISIS), 2014 Eighth International Conference on. **Proceedings...** [S.l.: s.n.], 2014. p.614–619.

SAKR, S. et al. A Survey of Large Scale Data Management Approaches in Cloud Environments. **Communications Surveys Tutorials, IEEE**, [S.l.], v.13, n.3, p.311–336, Third 2011.

SCHAD, J.; DITTRICH, J.; QUIANÉ-RUIZ, J.-A. Runtime Measurements in the Cloud: observing, analyzing, and reducing variance. In: VLDB-Conference on Very Large Data Bases, 36th International, Singapore. **Proceedings...** VLDB Endowment, 2010. v.3, p.460 – 471.

SHARMA, B.; WOOD, T.; DAS, C. HybridMR: a hierarchical mapreduce scheduler for hybrid data centers. In: Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on. **Proceedings...** [S.l.: s.n.], 2013. p.102–111.

SULISTIO, A. et al. A toolkit for modelling and simulating data Grids: an extension to gridsim. **Concurrency and Computation: Practice and Experience**, [S.l.], v.20, n.13, p.1591–1609, 2008.

TANG, B.; FEDAK, G. Analysis of Data Reliability Tradeoffs in Hybrid Distributed Storage Systems. In: Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International. **Proceedings...** [S.l.: s.n.], 2012. p.1546–1555.

TANG, B.; HE, H.; FEDAK, G. HybridMR: a new approach for hybrid mapreduce combining desktop grid and cloud infrastructures. **Concurrency and Computation: Practice and Experience**, [S.l.], v.27, n.16, p.4140–4155, 2015.

TANG, W. et al. Data-Aware Resource Scheduling for Multicloud Workflows: a fine-grained simulation approach. In: Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on, Singapore. **Proceedings...** [S.l.: s.n.], 2014. p.887–892.

TOOSI, A. N.; CALHEIROS, R. N.; BUYYA, R. Interconnected Cloud Computing Environments: challenges, taxonomy, and survey. **ACM Comput. Surv.**, New York, NY, USA, v.47, n.1, p.7:1–7:47, May 2014.

TUDORAN, R.; COSTAN, A.; ANTONIU, G. MapIterativeReduce: a framework for reduction-intensive data processing on azure clouds. In: Proceedings of 3rd Int. Workshop on MapReduce and Its Applications Date, New York, NY, USA. **Proceedings...** ACM, 2012. p.9–16. (MapReduce '12).

TUDORAN, R. et al. Bridging Data in the Clouds: an environment-aware system for geographically distributed data transfers. In: Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, Chicago, IL, USA. **Proceedings...** [S.l.: s.n.], 2014. p.92–101.

TUDORAN, R. et al. Evaluating Streaming Strategies for Event Processing Across Infrastructure Clouds. In: Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, Chicago, IL, USA. **Proceedings...** [S.l.: s.n.], 2014. p.151–159.

VASILE, M.-A. et al. Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing. **Future Generation Computer Systems**, [S.l.], v.51, p.61–71, 2015.

WANG, W.; LI, B.; LIANG, B. Multi-Resource Fair Allocation in Heterogeneous Cloud Computing Systems. **Parallel and Distributed Systems, IEEE Transactions on**, [S.l.], v.PP, n.99, p.1–1, 2014.

WHITE, T. **Hadoop - The Definitive Guide**. 3rd.ed. California, USA: O'Reilly Media, Inc., 2012. 688p. v.1.

WU, C.; BUYYA, R.; RAMAMOHANARAO, K. Big Data Analytics: machine learning plus cloud computing. In: Big Data: Principles and Paradigms, Burlington, Massachusetts, USA. **Proceedings...** Morgan Kaufmann, 2016. p.1–27.

XAVIER, M. et al. A Performance Isolation Analysis of Disk-Intensive Workloads on Container-Based Clouds. In: Parallel, Distributed and Network-Based Processing (PDP), 2015 23rd Euromicro International Conference on. **Proceedings...** [S.l.: s.n.], 2015. p.253–260.

ZAHARIA, M. et al. Improving MapReduce Performance in Heterogeneous Environments. **OSDI**, Berkeley, CA, USA, p.29–42, 2008.

ZAHARIA, M. et al. Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters. In: Proceedings of the 4th USENIX conference on Hot Topics in Cloud Computing, Berkeley, CA, USA. **Proceedings...** USENIX Association, 2012. p.10–10. (HotCloud'12).

ZHANG, C.; CHANG, E.-C.; YAP, R. H. Tagged-MapReduce: a general framework for secure computing with mixed-sensitivity data on hybrid clouds. In: Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, Chicago, IL, USA. **Proceedings...** [S.l.: s.n.], 2014. p.31–40.

ZHANG, T. Reliable Event Messaging in Big Data Enterprises: looking for the balance between producers and consumers. In: Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems, New York, NY, USA. **Proceedings...** ACM, 2015. p.226–233. (DEBS '15).

ZHANG, W. et al. Secure distributed keyword search in multiple clouds. In: Quality of Service (IWQoS), 2014 IEEE 22nd International Symposium of. **Proceedings...** [S.l.: s.n.], 2014. p.370–379.

ZHENG, Z. et al. STAR: strategy-proof double auctions for multi-cloud, multi-tenant bandwidth reservation. **Computers, IEEE Transactions on**, [S.l.], v.PP, n.99, p.1–14, 2014.

ZOU, Q. et al. Survey of MapReduce frame operation in bioinformatics. **Journal Briefings in Bioinformatics**, [S.l.], v.15, n.4, p.637–647, Feb. 2014.

Annex 1 Resource Allocation

A.1 Resource Allocation in Cloud Computing

The problem of heterogeneous resource allocation in Cloud can be defined as follows. If different users require similar resources from a CSP, and each user needs to meet heterogeneous demands, the system must decide which is the best resource assignment to one or another user. For example, one user might need CPU-intensive applications with a small data size, whereas another might need I/O-intensive applications with a Big Data size. The CSP can only provide one machine with two VMs for each user, or when the users with heterogeneous needs share a similar computer resource. What are the guarantees that the behavior of one application will not influence another in the Cloud ? On the other hand, the risk of the Cloud model only requires VMs so that it can have a provisioning based on demand and this creates an inefficient allocation of resources that results in higher costs for the cloud provider (PALANISAMY; SINGH; LIU, 2015).

The problem of multiple resource allocation has been generalized in (GHODSI et al., 2011) and is treated as a concept where each user receives a share of the resources proportional to its weight. Ghodsi proposes the Dominant Resource Fairness (DFR) algorithm, a generalization of the max-min fairness algorithm for multiple heterogeneous resource types, shown in Algorithm A.1.1. Each user receives resources based on a user's *dominant share*, that is the maximum share that the user has been allocated from any resource. The algorithm maximizes the minimum allocation received by a user of the system. This algorithm is implemented by the resource manager in the MESOS platform (HINDMAN et al., 2011), which has multiple cluster computing frameworks, such as Hadoop and MPI. The scheduling decision has an $O(\log n)$ time for n users, although the current implementation is a single cluster within a heterogeneous workloads environment. This algorithm has four properties for implementing a fair allocation policy:

1. *Sharing incentive*: Each n user should not be able to allocate more than $\frac{1}{n}$ tasks from all the resources in a cluster.

2. *Strategy-proofness*: A user cannot inflate her allocation by requests for more resources than necessary.

3. *Envy-freeness*: A user should not prefer the allocation of another user.

4. *Pareto efficiency*: The resource allocations are proportional. An increase in the user allocation implies a decrease of another user.

The DFR algorithm shares resources based on the user's dominant share (s_i). In this case, the user job does not have a weight related to prioritization of execution. A weight vector $\hat{W}_i = (\hat{w}_{i,1}, ..., \hat{w}_{i,m})$ is needed to define a prioritization, where the user's dominant share is defined as $s_i = max_{j=1}^{m}\{u_{i,j}/w_{i,j}\}$.

The study of the problem of multiple resource allocation in Cloud computing systems with heterogeneous servers is shown in the work of (WANG; LI; LIANG, 2014). Wang proposes the Dominant Resource Fairness for the Heterogeneous servers (DFRH) algorithm that is an extension of the DFR algorithm. The resources are pooled for they can support to a lot of heterogeneous servers. DFRH seeks an allocation that equalizes the

Algorithm A.1.1 DFR Algorithm - Pseudo-code

Require: $R = r_1, ..., r_m$ ▷ total available resources (CPU, memory)
Require: $C = c_1, ..., c_m$ ▷ consumed resources, Initially = 0
Require: $s_i, i \in (1, ..., n)$ ▷ user i's dominant shares, Initially = 0
Require: $\hat{U} = (\hat{u}_{i,1}, ..., \hat{u}_{i,m})$ ▷ resources given to user i, Initially = 0

1: **pick** user i with lowest dominant share s_i
2: $D_i \longleftarrow$ demand of user i's next task
3: **if** $C + D_i \leq R$ **then**
4: $\quad \hat{C} = C + D_i$ ▷ update consumed vector
5: $\quad \hat{U}_i = U_i + D_i$ ▷ update i's allocation vector
6: $\quad s_i = max_{j=1}^{m}\{u_{i,j}/r_j\}$
7: **else**
8: $\quad$ **return** ▷ cluster full

total value of every user's *global dominant share* in the entire Cloud resource pool. Currently, data centers are facilities built with servers from processors of different generations and heterogeneous resource groups of machines (with different types of processor, disk technologies and memory size). An algorithm pseudo-code is shown in Algorithm A.1.2. D_{ir} is the total amount of resources for each user i, and $D_{ir} > 0, \forall i, r \in R$. The resource r_i^* is called *global dominant resource* if Equation A.1.1

$$r_i^* \in \underset{r \in R}{\arg\max} D_{ir} \tag{A.1.1}$$

is true, *i.e.*, r_i^* is the most heavily requested computational resource, required by user i over the entire resource pool from the available hardware resources m. The allocation A is feasible if no server resource request is more than its total available resources. For each user i, the allocation A_{il} must be a $Nil(A_{il}) = \min_{r \in R}\{\frac{A_{ilr}}{D_{ir}}\}$. DFRH algorithm that is suitable for this environment where the users request a range of resources in heterogeneous profiles and workloads. The main problem of this model is that it does not define how the bandwidth problem can be handled among the multiple CSPs. The solution does not define any operation for Big Data environments in Cloud.

In the DFR, the allocations only depend on the total amount of pooled resources while in DFRH the resources are grouped among a large number of heterogeneous servers.The DFRH algorithm creates a user's *global dominant share* that is the maximum ratio of any resources from the user that has been allocated in the resource pool (in all the servers). The properties of DRFH allocations are defined below, and these are reduced to a single server as in the case of the DFR algorithm and are shown in (WANG; LI; LIANG, 2014):

1. *Sharing incentive*: The users can at least execute the number of scheduled tasks, when the entire pooled is already partitioned.

2. *Group Strategy-proofness*: A user cannot inflate her allocation by collaborating with other users to create misreports and request more resources than necessary.

3. *Envy-freeness*: A user should not prefer the allocation of another user.

4. *Pareto optimality*: The resource allocation of a user does not improve until at least another user has released a resource allocation.

Algorithm A.1.2 DFRH Algorithm - Pseudo-code

Require: $R = \{1, ..., k\}$ ▷ set of available resources (CPU, memory, storage,...)
Require: $S = \{s_1, ..., s_m\}$ ▷ set of heterogeneous servers.
Require: $\hat{c}_l = (\hat{c}_{l1}, ..., \hat{c}_{lm})^T$ ▷ resource capacity vector of server $l \in S$
Require: r_i^* ▷ user i's dominant resource, Initially = 0
Require: $U = \{1, ..., n\}$ ▷ set of Cloud users
Require: $\hat{D}_i = (D_{i1}, ..., D_{in})^T$ ▷ resource demand vector
Require: $\hat{A}_{il} = (A_{il1}, ..., A_{ilm})^T$ ▷ resource allocation vector
1: **if** $c_l + D_i \leq R$ **then**
2: $\hat{A}_{il} = c_{il} + \hat{D}_i$ ▷ user i's allocation vector
3: **while** $A_{ilr} \leq c_{lr}$ **do**
4: $\sum_{i \in U} A_{ilr}, \forall l \in S, r \in R$
5: **else**
6: **return** ▷ cluster full

A mechanism called FAWKES is a proposed scheduling system for the provision of dynamic resources with multiple MapReduce instances (GHIT et al., 2014). This mechanism decides how and which resources must be allocated across multiple MapReduce frameworks. The FAWKES receives calls to activate a MapReduce-cluster (MR-cluster) or to execute a MapReduce-job (MR-job). These requests are serviced in multiple queues with a FIFO policy. The main goal of this system is to have the load-balance from resources across all the active MR-clusters. The data locality model is relaxed to allow the replica reallocation. The goal is to allow data management and avoid data loss when a node is moved away from the MR-cluster. However, the node removed from the MR-cluster should only store a small amount of data to enable it to be reconfigured more rapidly.

The MR-cluster can improve the capacity with *transient* or *transient-core* nodes. The *transient* nodes are instantiated without local data, but can read and write data from or to the *transient-core* nodes. The *transient-core* requires data to process but it can write to its local disk. Thus, the FAWKES can improve the size of the MR-cluster without moving data. The model does not follow a discrete approach in time like the DRF and DRFH algorithms, but employs a *temporal discrimination* model represented by D_i to determine to what extent the system is balanced at each t moment, as in Equation A.1.2, where t_1 and t_2 represent the moments from the interval $[t_1, t_2]$ of requests for the deployment and removal of the MR-cluster. The proportion $c_i(t)$ of resources is shared on the basis of weight $w_i(t)$ at moment t. The weight mechanism of the MR-cluster is an average value of samples y_i collected during a time interval, defined by Equation A.1.3, where n is the number of active clusters.

$$D_i(t_1, t_2) = \int_{t_1}^{t_2} (c_i(t) - w_i(t))dt \tag{A.1.2}$$

$$w_i(t) = \frac{y_i(t)}{\sum_{k=1}^{n} y_k(t)} \tag{A.1.3}$$

A time interval updates the weights in each MR-cluster. The *instant preemption* and *delayed preemption* are two possible ways of shrinking when a node is removed from

an activated MR-cluster. In the *delayed preemption* a data movement is needed to the transient-core nodes before the task migrations. The weights will determine on-demand policies for the job, data and tasks. These policies manage the access to processor (P), disk (D) and a combination of these two resources, as in Equation A.1.4, where u_i^P and u_i^D resources are standardized to P and D. The $\psi \in (0,1)$ reflect the relative importance of these two resources:

$$u_i(t) = \psi \cdot u_i^P + (1 - \psi) \cdot u_i^D \tag{A.1.4}$$

CSPs allocate resources either statically, without taking account of the demands of the users or dynamically when evaluating the current users' demands. However, owing to variations in user workloads, the dynamic provision is a more efficient resource utilization mode, although it is harder to achieve optimal values. The studies of (MASHAYEKHY; NEJAD; GROSU, 2014) and (NEJAD; MASHAYEKHY; GROSU, 2015) propose a means of solving the dynamic provision problems with multiple resource types according to the requests of users.

The Polynomial-Time Approximation Scheme (PTAS) is a mechanism to facilitate Cloud resource provisioning based on users' on-demand requests and the availability of resources proposed in the study of (MASHAYEKHY; NEJAD; GROSU, 2014). The goal is to find an allocation of resources for the users, by maximizing the *social welfare*, where the *social welfare* is a metric of users' evaluations. This mechanism solves problems of provision of VM instances and resource allocation in the presence of multiple types of heterogeneous resources. The PTAS determines a near-optimal allocation while satisfying the strategy-proofness property and allowing a dynamic provisioning of VMs without requiring a VM pre-provisioning. They also take account of the problem of VM provisioning and allocation in Clouds and regard it as a relationship between price versus the user allocations. Each user first declares a request based on bundles and makes bids from different CSPs, and on the basis of these requests, the CSP determines the allocation of resources.

However, this relationship is one of conflict since the users want the highest number of resources at the lowest price, whereas the CSP would like to receive the highest price with the lowest number of allocated resources for one user. This problem can be solved with an algorithm called DP-VMPAC which is based on a dynamic programming approach, as shown in Algorithm A.1.3. The algorithm has two inputs, a user requests vector ($\hat{\theta}$) and a resource capacity vector ($\hat{C} = \{\hat{C}_1, ..., \hat{C}_R\}$). As output, the algorithm has an optimal *social welfare* (V^*), which is the VM value with the highest user aggregation expressed in Equation A.1.5, and optimal allocation of VM instances for the user (x^*).

$$V = \sum_{i \in U} v_i(S_i) x_i \tag{A.1.5}$$

Where $v_i(S_i) = b_i$ represents the maximum price that the user is willing to pay for using the requested bundle b_i and $x_i, i \in U$ are decision about the variables is defined as:

$$x_i = \begin{cases} 1 & \text{if bundle } S_i \text{ is allocated to user } i, \\ 0 & \text{otherwise} \end{cases} \tag{A.1.6}$$

The algorithm begin by determining the amount of each resources ($\hat{a}_{ir}$) of a specific type (w_{mr}) for a given user (i). DP-VMPAC algorithm solves the VM allocation problem

Algorithm A.1.3 DP-VMPAC Algorithm - Pseudo-code

Require: Input: $\hat{\theta} = \{\hat{\theta}_1, ..., \hat{\theta}_N\}$ ▷ vector of requests (bundle, bids).
Require: Input: $\hat{C} = \{\hat{C}_1, ..., \hat{C}_R\}$ ▷ vector of resources.
$\hat{S}_i = (\hat{k}_{i1}, ..., \hat{k}_{im})$ ▷ VM instances vector, where k_{im} is the VM number requested.
for all $i \in U$ **do**
  for all $r \in R$ **do**
    $\hat{a}_{ir} = \sum_{m \in VM} \hat{K}_{im} w_{mr}$, where $\hat{K}_{im} \in \hat{S}_i$
    $\hat{A}_i = (\hat{a}_{i1}, ..., \hat{a}_{iR})$ ▷ Amount of all resources types requested by user.
if $\hat{a}_{1r} \leq C_r, \forall r \in R$ **then**
  $V(1, \hat{C}) = \hat{b}_1$
  $\hat{C} = C - \hat{A}_1$
else
  $V(1, \hat{C}) = 0$
for all $j = 1, ..., N$ **do**
  $V(j, \hat{C}) = \max\{V(j-1, \hat{C}), V(j-1, \hat{C} - \hat{A}_j) + \hat{b}_j\}$
$V^* = V(N, \hat{C})$
Find x^* looking backward at $V(j, \hat{C})$
Output: V^*, x^*

in time $O(N(C_{max})^R)$, where $R = \{1, ..., M\}$ represents the different resource types and the value $C_{max} = max_{r \in R}\{C_r\}$ represents the maximum resource capacities, for N users. The Vickrey-Clark-Groves mechanism (VCG) determines a payment function, that is called DP-VMPAC and is designed to find the allocation and a *social welfare* without any intervention by the user. A function is a VCG mechanism if: $\hat{A}$ is an optimal allocation function and the provision of payment is defined in Equation A.1.7.

$$P_i(\hat{\theta}) = \underbrace{\sum_{j \in U \setminus \{i\}} v_j(A_j(\hat{\theta}_{-i}))}_{\text{optimal social welfare}} - \underbrace{\sum_{j \in U \setminus \{i\}} v_j(A_j(\hat{\theta}))}_{\text{all user valiations except for user } i} \tag{A.1.7}$$

This execution is prohibitive for large instances of VMPAC, because the problem is NP-hard. Therefore there is not a fully polynomial time approximation scheme (FPTAS) (MASHAYEKHY; NEJAD; GROSU, 2014). To solve this maximization problem the authors introduce a *strategy-proof* PTAS mechanism, where for all instances of I with associated error $\varepsilon > 0$, a solution S is found that satisfies the equation $S(i) = (1 - \varepsilon)S^*(I)$, where $S^*(I)$ is an optimal value of I. The algorithm called PTAS-ALLOC, is shown in Algorithm A.1.4, and has three inputs: the vector of user requests $\hat{\theta}_i = (\hat{S}_i, \hat{b}_i)$, the vector of capacities $\hat{C}$ and an integer q, where $q \leq N$. To solve the problem, the authors consider a subset $V(j, \hat{d})$, where j is the first remaining user with the available capacity $\hat{d}$. $V(j, \hat{d})$ is the optimal value to solve the problem, if the resource allocation for the jth user increases the value of $V(j-1, \hat{d})$, in this case, the bundle is allocated to the jth user. The maximum value is $V(j-1, \hat{d})$ and $V(j-1, \hat{d} - \tilde{A}_j) + \hat{b}_j$ results in an optimal value of $V(j, \hat{d})$, as in Equation A.1.8, where $\tilde{A}_j$ is the vector of the rounded size from requested resources of user j. However, when q is equal to the user's numbers, the algorithm degenerates into an exhaustive search (MASHAYEKHY; NEJAD; GROSU, 2014).

$$V(j, \hat{d}) = max\{V(j-1, \hat{d}), V(j-1, \hat{d} - \tilde{A}_j) + \hat{b}_j\} \tag{A.1.8}$$

Algorithm A.1.4 PTAS-ALLOC Algorithm - Pseudo-code

Require: Input: $\hat{\theta} = \{\hat{\theta}_1, ..., \hat{\theta}_N\}$ ▷ vector of requests (bundle, bids).
Require: Input: $\hat{C} = \{\hat{C}_1, ..., \hat{C}_R\}$ ▷ vector of resources.
Require: Input: q
$V = -\infty$
for all $\hat{U} \subseteq U : |\hat{U}| \leq q$ **do**
  $\hat{x} = 0$
  $\hat{V} = 0$
  $sum_r = 0, \forall r \in R$
  for all $i \in \hat{U}$ **do**
    $\hat{x} = 1$
    $\hat{V} = \hat{V} + \hat{b}_i$
    for all $r \in R$ **do**
      $sum_r = sum_r + \sum_{m \in VM} \hat{K}_{im} w_{mr} \hat{x}_i$
  if $C_r \geq sum_r, \forall \in R$ **then**
    $\hat{U} = U\ hatU$
    $\hat{q} = |hatU|$
    for all $r \in R$ **do**
      $d_r = C_r - \sum_{i \in U} \sum_{m \in VM} \hat{K}_{im} w_{mr} \hat{x}_i$
    $d = (d_1, ..., d_R)$
    for all $i \in U$ **do**
      for all $r \in R$ **do**
        $\hat{a}_{ir} = \sum_{m \in VM} \hat{K}_{im} w_{mr}$
        $\hat{a}_{ir} = \lceil \hat{a}_{ir} N^2 / d_r \rceil d_r / N^2$
      $\tilde{A} = (\tilde{a}_{i1}, ..., \tilde{a}_R)$
      {DP to find$(\tilde{V}, \tilde{x})$ for $(\tilde{U}, d)$ }
      $d = d$
      if $d_r \geqslant \tilde{a}_{1r}, \forall r \in R$ **then**
        $V(1, d) = \hat{b}_1$
        $\hat{d} = d - \tilde{A}_i$
      else
        $V(1, d) = 0$
      for all $j = 2, ..., N - \hat{q}$ **do**
        $V(j, d) = \max\{V(j-1, \hat{d}), V(j-1, \hat{d} - \tilde{A}_j) + \hat{b}_j\}$
      $\tilde{V} = V(N - \hat{q}, d)$
      Find $\tilde{x}$ by looking backward at $V(j, \hat{d}\ )$
      if $V < (\hat{V} + \tilde{V})$ **then**
        $V = \hat{V} + \tilde{V}$
        $x = \hat{x} + \tilde{x}$
Output: V, x

In the study of (NEJAD; MASHAYEKHY; GROSU, 2015) the problem for dynamic provisioning is viewed as a *multiple-unit combinatorial auction*. The goal is to design greedy mechanisms that determine the resource allocation problem in the presence of multiple types of resources (*i.e.* CPU, memory and storage). Each user has a personal value and a personal resource type linked to her request. The approach is *strategy-proofness* and the authors determine an approximation ratio of the proposed mechanisms, where there are very many guarantees for the obtained solutions. The mechanisms are *truthfully greedy*, although greedy algorithms do not necessarily satisfy properties required to guarantee truthfulness. The user request N resource types in a set U of users, that is denoted by $\hat{\theta}_i = (\hat{S}_i, \hat{b}_i)$. The allocation and payments depend on declarations of user resource types. The allocation function finds a subset $A(\hat{\theta}) \subseteq U$ for winning users, where A_i is the allocated resource to user i. A user *valuation* v_i from an allocated resource of a bundle b_i is defined in Equation A.1.9.

$$v_i(A_i) = \begin{cases} b_i & \text{if } S_i \subseteq A_i, \\ 0 & \text{otherwise} \end{cases} \tag{A.1.9}$$

The bundle of VM instances requested by a single-minded user is the minimum amount of resources that the user need to run her applications. The utility function is $u_i(\theta) = v_i(A_i(\theta)) - P_i(\theta)$, where $P_i(\theta)$ is the payment made by the leased resources of the CSP. The mechanism calculates the value of $P_i(\theta)$ on the basis of a payment rule P. The users declare different types of resources $\hat{\theta}_i = (\hat{S}_i, \hat{b}_i)$ and receive the resource $\theta_i = (S_i, b_i)$ from the CSP.

A user maximizes her resources due to truthful reporting regardless of other user reports. The authors implement a preference relation $\succeq$ to create a strategy-proof, as follows: $\hat{\theta}'_i \succeq \hat{\theta}_i$ **if** $\hat{b}'_i \geq \hat{b}_i$ **and** $\hat{S}_i =< \hat{k}_{i1}, \hat{k}_{i2}, \ldots, \hat{k}_{iM} >, \hat{S}'_i =< \hat{k}'_{i1}, \hat{k}'_{i2}, \ldots, \hat{k}'_{iM} >$ **such that** $\sum_{m \in VM} \hat{k}'_{im} w_{mr} \leq \sum_{m \in VM} \hat{k}_{im} w_{mr}, \forall r \in R$. The resources $\hat{\theta}'_i$ are preferred than $\hat{\theta}_i$ if the user requests fewer resources in her current bundle $\hat{b}'_i$ or submits a higher bid. The Amazon uses an approach called *spot market*, but only allows requests to individual VM instances and not to bundles of VM instances, as in this study (NEJAD; MASHAYEKHY; GROSU, 2015).

The authors propose the VCG-based truthful optimal mechanism, where the mechanism $M = (A, P)$ is a VCG mechanism if A maximizes the *social welfare*, and the payment P is defined by Equation A.1.10 as:

$$P_i(\hat{\theta}) = \underbrace{\underbrace{\sum_{j \in A(\hat{\theta}_{-i})} \hat{b}_j}_{\text{optimal social welfare}} - \underbrace{\sum_{j \in A(\hat{\theta}_{-j \neq i})} \hat{b}_j}_{\text{all users validations}}}_{\text{except those where user } i \text{ had participated}} \tag{A.1.10}$$

The G-VMPAC-ALLOC, shown in Algorithm A.1.5, defines a general *density* metric to determine how scarce a resource is, as defined in Equation A.1.11. This metric will identify users with high service demands. The factor $\hat{a}_{ir}$ represents the total resources of type r which the user i has requested. It is defined by $\sum_{m \in Vm} \hat{k}_{im} w_{mr}$. The f_r is a factor that characterizes the scarcity of resource r.

$$d_i = \frac{\hat{b}_i}{\sqrt{\sum_{r=1}^{R} f_r \, \hat{a}_{ir}}}, \forall i \in U \tag{A.1.11}$$

The authors argue that the f_r can assume three different values:

1. G-VMPAC-I-ALLOC: when $f_r = 1, \forall r \in R$, it may not work well where the VM instances are highly heterogeneous.

2. G-VMPAC-II-ALLOC: when $f_r = \frac{1}{C_r}, \forall r \in R$, this indicates that it has more idle resources than requests.

3. G-VMPAC-III-ALLOC: when $f_r = \frac{\sum_{i=1}^{N} \hat{a}_{ir} - C_r}{\sum_{i=1}^{N} \hat{a}_{ir}}$, $\forall r \in R$. This reflects the scarcity of resources, *i.e.*, users with high demands of resources have lower f_r and are less likely to receive their requested bundles.

Algorithm A.1.5 G-VMPAC-ALLOC Algorithm - Pseudo-code

Require: Input: $\hat{\theta} = \{\hat{\theta}_1, ..., \hat{\theta}_N\}$ ▷ vector of requests (bundle, bids).
Require: Input: $\hat{C} = \{\hat{C}_1, ..., \hat{C}_R\}$ ▷ vector of resources.
$V = 0$
$x \longleftarrow 0$
$\hat{C} = C$
for all $r \in R$ **do**
  $f_r \longleftarrow 1$, for G-VMPAC-I-ALLOC; or
  $f_r \longleftarrow \frac{1}{C_r}$, for G-VMPAC-II-ALLOC; or
  $f_r \longleftarrow \frac{\sum_{i=1}^{N} \hat{a}_{ir} - C_r}{\sum_{i=1}^{N} \hat{a}_{ir}}$, for G-VMPAC-III-ALLOC
for all $i \in U$ **do**
  $d_i = \frac{\hat{b}_i}{\sqrt{\sum_{r=1}^{R} f_r \, \hat{a}_{ir}}}$
Sort U in decreasing order of d_i
for all $i \in \hat{U}$ **do**
  $flag \longleftarrow$ TRUE
  for all $r \in R$ **do**
    $\tilde{C}_r = \hat{C}_r - \sum_{m \in VM} \hat{K}_{im} w_{mr}$
    if $\tilde{C}_r < 0$ **then**
      $flag \longleftarrow$ FALSE
      break;
  if $flag =$ TRUE **then**
    $V = V + \hat{b}_i$
    $x_i = 1$
    $\hat{C} = \tilde{C}$
Output: V, x

Annex 2 Data Privacy in Multiple Clouds

The data of the organizations might involve both sensitive and non-sensitive information. The computation of this profile should not be carried out in the public cloud without a safe environment. Many users and businesses hesitate to keep confidential data in a CSP because of data integrity and security problems, as described by Zhang (ZHANG et al., 2014). The study of Zhang entails conducting a secure search of distributed keywords across multiple clouds. The aim is to provide a secure search service for encrypted cloud data. This approach has two schemes: the first scheme is a *cross-store*, *i.e.* all *file slices*, keywords and keys are encrypted (this provides efficiency and anonymity for data owners). The second scheme involves constructing a *distribution strategy* for both keyword and files (this sets out the search and security requirements). These schemes are combined with *Shamir's secret scheme* to improve the distributed search system, where, all the files and keywords for different CSPs are encrypted with different secret keys.

In the *cross-store*, each file is encrypted through an encryption key before it is distributed to N cloud servers or for some server in a particular CSP. The user has one particular secret key $H_s(k_{w,j})$ that will be used to encrypt a file segment $C_{i,j} = (F_{i,j})_{H_s(K_{f,j})}$. The file distribution is generated from random *file slices* $(F_{i,1}, F_{i,2}, \ldots, F_{i,N-1})$ with the same length. Each file F_i carries out an exclusive OR operation $\oplus$ and $F_{i,N} = F_{i,1} \oplus F_{i,2} \oplus \ldots \oplus F_{i,N-1} \oplus F_i$. Then the secret keys ($k_{f,j}$, $k_{w,j}$) and *file slices* ($F_{i,N}$) are cross-stored among these CSPs. The authorized user retrieves the *file slices* and the secret keys when he wants to access the data directly.

In the *distribution strategy*, a secret key is used to encrypt the files and a secret key matrix shows the order of iterative encryption in the *file slices*. The file distribution is stored in three stages: a) Each file is encrypted and partitioned into N *file slices*. b) The system creates an iterative encryption matrix M with the *file slices* that are stored with the user. c) The encrypted file is distributed over CSPs without the providers knowing the secret matrix. When an authorized user wants to execute a security search, he issues a *trapdoor* to CSPs to retrieve the encrypted file slices. Following this, these file slices are decrypted and the original file is reconstructed. The *trapdoor* is an encrypted keyword without previous information from the destination. The computational complexity is $O((N!)^T)$, where N is the number of CSPs and T is the number of iterative encryption rounds.

Hybrid clouds uses local resources to process a determined number of workloads. When these resources dry up in a "have-tail consume", more availability is needed in a public cloud, and issues like security and privacy become a problem, as shown in the study of (ZHANG; CHANG; YAP, 2014). A safe computational approach to *hybrid cloud* is to separate sensitive from non-sensitive data. The authors introduce a framework to *MapReduce* for secure computing with mixed-sensitivity data in *hybrid cloud* called *Tagged-MapReduce*. The data receives sensitive or non-sensitive flags and the key/value pair has an equivalent tag.

The goal is to prevent sensitive data from leaving private cloud. The programmers can decide where the computation will be executed. As a result, the input data and the intermediate keys can be moved from a private cloud to a public cloud without compromising security. The *Map* and *Reduce* phases produce {key/value; tag} pairs when a sensitive input data is received. The scheduling of both HDFS and *Jobtracker* will identify the tag. The function of the tag is to indicate the execution path for the system. The scheduler em-

ploys all the intermediate results for private cloud to carry out the reduction, even though the intermediate transfers might overload the private servers and generate a high volume of data from clouds (whether public or private).

The study of (CHEIKH; ABBES; FEDAK, 2014) adopts an approach which involves breaking data down into meaningless chunks and spreading them in a hybrid environment. Meaningless data is obsolete and useless information, which if accessed by a malicious worker cannot be used to reconstruct the original information. The aim is to improve data privacy, through a combination of trusted and mistrusted infrastructures, like private and public Clouds. The authors implemented the Information Dispersal Algorithm (IDA), where after breaking and data scattering, each machine that seeks to access the original data, has to contact other machines to get some missing chunks to reconstruct the information. If m chunks are employed to reconstruct the data, $m-1$ chunks are deployed in a *hybrid cloud* infrastructure like public cloud and desktop grid, the remaining chunks are sent to a private cloud.

The scheme follows four phases: a) First, the input data is split to generate n chunks. A header maintains the positions of the rows and columns of a key matrix in a vector. The key vector indicates the i^{th} row of the matrix and a body stores the data produced. b) Second, the master sends these incomplete chunks to the mapper machines and waits for the next phase of the processing. c) Third, each mapper randomly chooses its friends from among all the mappers to receive their $m-1$ chunks and calculates an inverse matrix needed to retrieve the complete data. d) Fourth, a combining phase rebuilds the original chunk in memory. After this, the chunks are processed. The authors argue that the preparation phase can take a long time to complete. Hence, this approach might have a low performance with high workloads.

B.1 Overview of the SMART Safety Model

This model was proposed by the Belarus team in the context of the SMART platform. The proposed approach to secure Big Data systems in Cloud is an evolution of a cloud-based access control and privacy protection infrastructure which is currently implemented in a protected enterprise Cloud storage - Storgrid (BV, 2016). This infrastructure is the core of the protected environment in the Cloud and has heterogeneous devices and an attribute access-based policy. Thus, it is suitable for the SMART cloud-based architecture and corresponding Big Data processing services. The proposed security infrastructure includes the following components (as illustrated in Figure B.1).

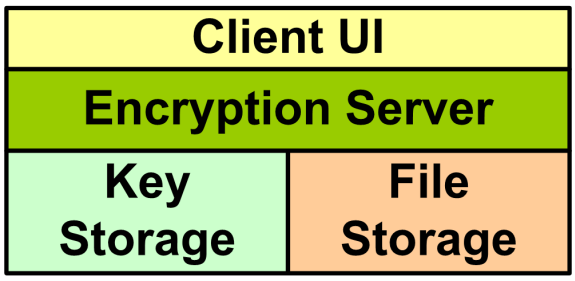


Figure B.1: Security infrastructure and components

Encryption server manages all the certified authentication (CA) and encryption operations and grants the user access to the data storage. This server can store the encryption keys and/or connect to a separate Key Storage server.

File storage is secure in the sense that some of the files specified by the domain administrator are stored, encrypted and have restricted access. Due to the fact that the file storage data is partially stored in the cloud, *i.e.*, externally, it is recommended this external part of the file storage is completely encrypted.

The *Client UI* can connect to the *Encryption server* and ask for permission to access the file storage to view/edit/upload specific files or folders.

All the components of the proposed system support the high-level cryptographic functions, including different types of encryption, timestamp verification and hashing. To improve efficiency, a hybrid encryption system is set up which combines both symmetric and attribute-based encryption. This configuration ensures the privacy of the users without compromising the overall encryption time. In addition, the basic ABE (BETHENCOURT; SAHAI; WATERS, 2007), (GOYAL et al., 2006) approach was modified to improve the configuration parameters and increase the speed of the encryption, for the purpose of setting up the validation period from user key and more sophisticated attributes corresponding to both the file shares and user groups. The basic functionality of the security components can be briefly described as follows:

File storage: The bulk data in the protected file storage is encrypted with the appropriate block cypher (AES, Blowfish, IDEA, Serpent). The key to the encrypted data is stored in the key storage and has an expiry period to provide better protection. Once the portion of data is sent to the client, it is decrypted by the server and re-encrypted with another single-use session key.

Key storage: The symmetric keys for the data in the file storage are kept in a separate storage system. The protection of the key storage is implemented by means of a secure authentication method, *i.e.*, two-factor authentication. Additionally, the following methods are employed to increase the security of the sensitive data:

a. Setting up the key expiry period;

b. Using separate keys for the different files;

c. Using a secret sharing mechanism for key storage with the most sensitive data.

Encryption server: The most important cryptography services are run by the Encryption server. This server generates the user keys and connects to the client UI, *i.e.*, a separate user of the system and decides whether access to the specific dataset should be granted to this user. In addition, the server runs the key renewal routines, stores the user public keys and attributes/audits the data.

Client UI: Client UI connects to the encryption server and checks the expiry period of the user keys (in case it has been configured) and permits the device user to view/edit/upload the data. Client UI stores the user keys for the ABE encryption and the single-use symmetric session keys which serve to restrict access to the downloaded files. The symmetric keys are encrypted with the ABE keys. The client allows the whole system to work in a heterogeneous environment as it supports different platforms and operating systems.

The described modular infrastructure allows different components to be set up separately and the security system to be configured for specific needs. The integration of the security infrastructure into the Big Data environment is possible because of its flexibility and scalable architecture. The main purpose of the proposed security infrastructure is to ensure the setup of the access control in the cloud-based protected environment. The access control mechanism performs the following tasks:

1. *Authentication of users in the cloud system:* The initial authentication is performed by the user password and email id. A hierarchical authentication system for highly sensitive data is implemented to increase security. This involves a more sophisticated two-factor authentication, as well as the password and access to e-mail; the possession of a specific device is checked too. This can be used by the government services or services that have the highly sensitive data.

2. *Provision of access control functions and protection of data from unauthorized access:* once the user has been authenticated, he/she can obtain access to the functions of the storage and then upload or download the data. The CA services are run by the encryption server. The server generates and distributes the user keys and keeps the group attributes along with the file sharing ids. Access control only allows data to be securely distributed and shown to the user (or accepted by the user)if he/she is permitted to view/edit it. This protection is ensured by a special version of ABE which is used with the implementation of both possible policies: key policy – KP, and ciphertext policy – CP, to support the attributes of the groups of users as well as the attributes of the file shares. This algorithm is developed specifically for the "access structure" of the proposed cloud architecture.

3. *Protection of user data privacy:* Once the user wishes to access a separate file that is downloaded on the user's device, the client uses his/her ABE key after performing the authentication to decrypt the symmetric session key and open the file.

B.1.1 Security Infrastructure of the SMART Cloud-Based Processing Service

The SMART architecture and the proposed hybrid CA system are combined by separating the functions of the encryption server, *i.e.*, the protection services that work once the data has been uploaded (authentication, CA, encryption). This must run before the data is sent to the Global Aggregator from the protection services that operate when the data is downloaded, and must be set up after the data passes the Global Dispatcher (CA, decryption, key refreshment). The encrypted cloud storage is also separated. The whole "ecosystem" is shown in Figure B.2.

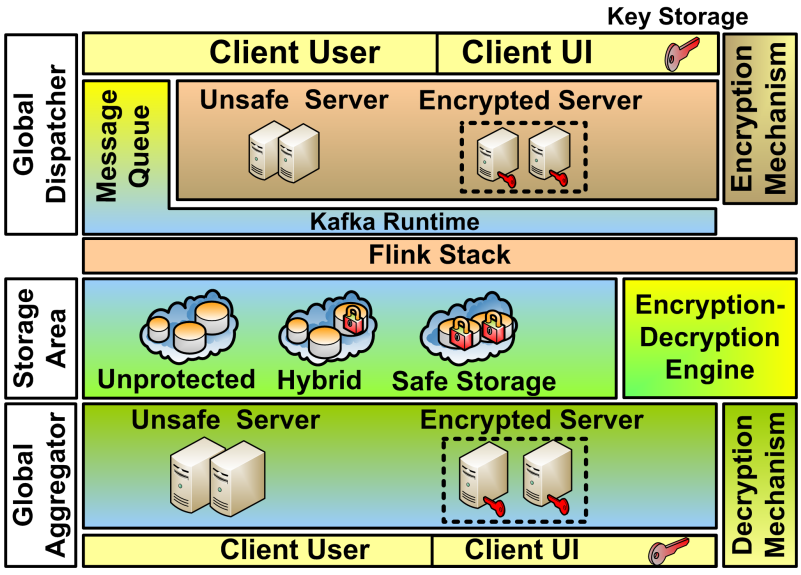


Figure B.2: Components of the security infrastructure and their interactions

This integration model makes it possible to implement the required level of security and user privacy in all types of government and corporate organizations and services (like health services) using the SMART infrastructure. The modular design of the implementation ensures protection of user privacy and controls access to sensitive data in the heterogeneous environment via the Client UI which is developed for a wide range of operational systems and platforms. Security is globally supervised with the aid of the Encryption server and separate key storage. Additionally, this architecture can be easily extended so that more sophisticated methods can be used for ensuring data and key protection, *i.e.*, secret sharing or complex attribute policies.

Annex 3 Principais Ideias da Tese e Trabalhos Futuros

Um enorme volume de dados é produzido todos os dias, a partir das informações fornecidas por redes sociais (tais como o Facebook, Instagram, Whatsapp, etc) ou gerados por sensores e dispositivos móveis, incluindo aplicações de Big Data, como as pesquisas no Google. Esse dilúvio de dados (também conhecido como *data deluge* requer cada vez mais recursos computacionais para processar as informações mais rapidamente.

Este trabalho estabelece: i) uma nova plataforma chamada SMART que oferece análise de Big Data em uma arquitetura Lambda sob uma infraestrutura híbrida; ii) um simulador chamado *BIGhybrid* destinado a ser um kit de ferramentas para o estudo de análise de Big Data em uma infraestrutura híbrida. Seu objetivo é permitir que o usuário possa encontrar a configuração mais próxima do real para a implantação de Big Data em ambientes híbridos; iii) avalia o uso do módulo *Dispatcher* na plataforma SMART e iv) define estratégias para o uso de grades de estações de trabalho *Desktop Grid* (DG) e computação em *Cloud Cloud Computing* (*Cloud*) em um ambiente geograficamente distribuído em uma infraestrutura híbrida. Além disso, busca encontrar as melhores estratégias de distribuição de dados neste cenário complexo para reduzir os riscos de problemas causados por erros de configuração comuns neste ambiente.

Neste capítulo, há um resumo das principais ideias e contribuições apresentadas nos capítulos anteriores, também apresenta as parcerias desenvolvidas durante este trabalho e as publicações como resultado do desenvolvimento da tese. Finalmente, os trabalhos futuros.

C.1 Contribuições

As principais contribuições podem ser resumidas como:

a) Fornecer uma plataforma que pode permitir a implantação da análise de Big Data em infraestruturas híbridas

Um ambiente de infraestrutura híbrida foi estabelecido. Em seguida, as especificações deram origem a uma nova plataforma chamada SMART (um acrônimo para análose de dados em tempo real para pequenas e médias empresas do inglês *Small & Medium-sized Enterprise Data Analytics in Real Time*). A plataforma permite a análise de Big Data em uma infraestrutura híbrida. Em comparação com outros *frameworks*, esta plataforma usa *Cloud* e DG, como sua infraestrutura básica e é incorporada em uma arquitetura Lambda para o processamento de Big Data. A abordagem de implantação segue a utilização de um sistema geograficamente distribuído .

Esta plataforma é composta de vários módulos que descrevem um *Dispatcher Global* e u *Global Aggregator*. Este conceito permite o uso de uma infraestrutura de serviços que usa várias abstrações de dados. O *Dispatcher Global* localizado fora da *Cloud* é uma camada que lida com atribuições e tarefas e o gerenciamento de dados fornecidos pelo usuário. A camada dissocia o sistema de armazenamento de dados e gerencia políticas para a divisão e distribuição dos dados, de acordo com cada sistema.

O princípio de funcionamento é semelhante a um serviço de publicação/assinatura (*publish/subscriber*), no qual a camada de publicação age como um produtor de dados que depois é consumido pelos trabalhadores. O *Global Aggregator* ou agregador global, obtém a saída de dados de ambos os sistemas e mescla as saídas para obter o conjunto de dados final.

b) Criar um simulador como uma ferramenta analítica de infraestrutura híbrida e para análise de Big Data

BIGhybrid é um conjunto de ferramentas para simulação de *MapReduce* em ambientes híbridos, como *Cloud* e DG, que foi validado por meio de experimentos reais. Ele pode ser usado para avaliar estratégias de escalonamento para aplicações *MapReduce* em infra-estruturas híbridas. A ideia por trás do simulador BIGhybrid é de otimizar ambientes de infra-estrutura híbridas com recursos disponíveis de um sistema DG. BIGhybrid é construído sobre o *framework* SimGrid (um simulador de propósito geral, construído em Grenoble/Franca) em uma estrutura modular.

O simulador de BIGhydrid gera traços de cada sistema para permitir uma análise individual ou coletiva a ser realizado dentro da mesma execução. O simulador permite alterar várias estratégias para a alocação de recursos e distribuição dados de aplicações *MapReduce* em infraestruturas híbridas. No BIGhybrid, o *Global Dispatcher* pode ser manual ou automático. Na versão manual, o usuário define uma função para distribuição de dados e uma configuração de trabalho. Essas configurações exploraram o número de tarefas *Map* e *Reduce*, dados de entrada, a carga de trabalho, divisão de dados e assim por diante.

Os resultados do módulo *Global Aggregator* são implementados como uma única tarefa de *Reduce* após a última fase de *Reduce* ser concluída. As avaliações estatísticas do simulador BIGhydrid indicam uma simulação com um relativo grau de precisão. O erro médio percentual absoluto de 5%, (na pior das hipóteses para ambientes heterogêneos e homogêneos) mostra que o simulador pode ser um instrumento eficiente para a avaliação de infraestruturas híbridas.

c) Determinar estratégias de utilização de infraestruturas híbridas

Os experimentos são realizados em ambientes reais e simulados. O experimentos reais em *Cloud* tem o objetivo de observar o comportamento de aplicações em tempo real em um cenário de desacoplado. O ambiente simulado tem o objetivo de reproduzir as cargas de trabalho de aplicações Big Data em um cenários semi-acoplados. As avaliações encontraram padrões de comportamento que permitem a implantação em ambientes híbridos em baixa, média e alta escala, e estabelecer um conjunto de estratégias, tais como: i) a relação entre o tempo de execução e o número de nós de voláteis; ii) a relação da distribuição de recursos em *Cloud* e DG; iii) a relação entre o número de máquinas de *Cloud* e DG; iv) a relação do tamanho dos dados em função da largura de banda disponível; v) a relação entre o balanceamento de carga e o número de máquinas; vi) o custo relacionado com a *Cloud* e o DG e o vii) a interferência I/O. Os resultados estão sumarizados na Tabela C.1.

Table C.1: Estratégias e Objetivos Atingidos

Parâmetros	Estratégias	Objetivos
Nós Voluntários	$5\% < \xi < 25\%$	Atingido
Recursos de *Cloud*	$\Phi_{Ch} = \frac{Ck_{Ch}}{Ck_c} * \frac{W_{Ch}}{\sum_i^M Vm_c(i)}$	Atingido
Recursos de DG	$\Phi_{DG} = \frac{Ck_{DG}}{Ck_c} * \frac{W_{DG}}{\sum_i^M M_{DG}(i)}$	Atingido
Relação Φ entre *Cloud* e DG	$1 \leq \Phi \leq \Phi_C$	Atingido
Canal de Comunicação	Tamanho de *Chunk* 32 MB a 64 MB	Atingido
Balancemaento de carga	$\Phi_{Ch} \leq \Phi_C \Longleftrightarrow \Phi_{DG} \leq \Phi_C$	Atingido
Modelo de Custo	$C(J_i, C^{k,n}) = t_r * (J_i, C^{k,n}) * \sum_{m=1}^{\beta} n * R_{c_m}$	Necessita avaliação
interferência de I/O	Definir um valor para ρ	Em aberto

Os resultados apontam que os ambientes híbridos mostram uma continuidade de op-

eração em um ambiente com até 25% de nós instáveis na pior das hipóteses, sem perder desempenho e com um número de 3 réplicas, a qual é o padrão do Hadoop. Os parâmetros de Φ_{Ch} e Φ_{DG} para encontrar recursos em *Cloud* e DG estabelecem um número de máquinas adequadas para atingir um desempenho aceitável. Além disso, essas configurações ajudam usuários inexperientes a localizar o número de máquinas *Cloud* e DG sem um conhecimento prévio da infraestrutura de provedor de *Cloud*, o que pode ser considerado uma contribuição significativa nesta Tese. A relação Φ entre a *Cloud* e a DG pode ajudar os usuários a encontrar recursos ajustados para suas cargas de trabalho. Também, a recomendação de tamanho de bloco na comunicação de canal pode contribuir para evitar a movimentação excessiva de dados em aplicações Big Data sobre as infraestruturas híbridas.

A relação entre a carga de trabalho, número de máquinas e balanceamento de carga pode ser considerada uma contribuição significativa para fornecer balanceamento de carga de dados e diminuir as transferências de dados entre máquinas de 39,1% para 57,14% no melhor e pior caso, respectivamente. Estes valores são compatíveis com obras da literatura.

A redução de tamanho de bloco pode produzir um pequeno aumento de desempenho em *Cloud* com uma largura de banda de 1 Gbps. No entanto, uma diminuição do tamanho de bloco de 64 MB para 16 MB gera uma perda de desempenho de 18.37% a 34.16% para execuções com 300 Mbps e 50 Mbps respectivamente. Estes resultados demonstram que a redução de dados resulta em um comportamento diferente em comparação entre um ambiente híbrido e um ambiente homogêneo. Assim, com base nestes resultados, o melhor tamanho de bloco para média e alta escala é de 64 MB e, em baixa escala, também é possível usar tamanhos de bloco de até 32 MB.

O uso de cenários semi-acoplados e desacoplados permitem generalizar este projeto de arquitetura para um vasto espectro de implementações de Big Data com dados dispersos em torno de diferentes locais, em particular, para dados geograficamente distribuídos. Estes resultados representam uma contribuição significativa para a implantação de infraestruturas de híbridas em aplicações de Big Data. Os resultados indicam uma redução do curso de 10% a 50% no pior caso e no melhor caso respectivamente, em comparação com somente uma execução em *Cloud*. No entanto, algumas estratégias como o modelo de custo e interferência de I/O precisam de um estudo cuidadoso ainda em função da alta especificidade do tema.

d) Encontrar um método preciso para realizar uma análise de uma infraestrutura híbrida

Um algoritmo foi implementado a partir dos experimentos para definir um método de cálculo para a alocação de recursos no módulo *Dispatcher*. Além disso, esse algoritmo tem adaptações para estimar o custo em ambientes híbridos. No entanto, o modelo de custo precisa de uma avaliação cuidadosa. Os usuários inexperientes podem alocar o número de máquinas para *Cloud* e DG sem um conhecimento prévio da infraestrutura do provedor de *Cloud*.

As avaliações no ambiente simulado e experimentos reais mostraram que usar infraestruturas híbridas em aplicações de Big Data pode ser viável. O fluxo de trabalho de processamento em uma infraestrutura híbrida para aplicações da análise de sentimento mostram as oportunidades para melhorar a execução dos motores de Big Data em função dos gargalos encontrados.

C.2 Comparação entre a plataforma SMART e trabalhos relacionados

Tabela C.2 mostra o SMART em comparação com trabalhos relacionados. A plataforma permite o uso de ambientes geograficamente distribuídos em infraestruturas híbrida com suporte a arquitetura Lambda. As principais estratégias considera o custo, recuperação de falhas, rede de alças despesas gerais, distribuir dados para capacidades de ambiente, preserva os mecanismos de replicação, minimiza a transferência dedados, fornecer um sistema orchestrator, tem buit-in uma estratégia de grupo a distribuição de dados e tarefas e habilitar sem barreiras. Esses recursos são adicionados às estratégias para distribuição dedados e definição do número de máquinas do sistema. Os usuários inexperientes para localizar nuvem e DG máquinas sem um conhecimento prévio da infra-estrutura do CSP.

Table C.2: *Frameworks* e técnicas de análise de Big Data

			Infraestrutura							Estratégias												Proc	
Area	Author	Proposta	Geo-distribuído	Infraestrutura de *Cloud*	*Desktop Grid*	*Cluster*	Multi-Cloud	Sistema Híbrido	Arquitetura Lambda	Considera Custo	Recuperação de Falhas	Sobrecarga de Rede	I/O Throughput	Capacidade do Ambiente	Avalia Armazenamento	P/S or M_Queue	Replicação	Minimiza Transferências	Sistemas de Orquestração	Processa em Memória	Estratégia de Grupos	Sem Barreiras	Batch	Orientado a Eventos
Frameworks	(EKANAYAKE et al., 2010)	Twister				X										X	X			X			X	
	(MANSOURI; TOOSI; BUYYA, 2013)	Brokering Alg.					X			X	X						X							
	(GUNARATHNE et al., 2013)	Twister4Azure		X												X	X			X		X	X	
	(LE et al., 2014)	SALSA		X										X					X					
	(JAYALATH; STEPHEN; EUGSTER, 2014)	G-MR	X				X			X		X											X	
	(TUDORAN et al., 2014)	GEO-DMS	X				X			X		X	X	X									X	
	(PALANISAMY; SINGH; LIU, 2015)	CURA		X						X														
Heterogêneo	(LIN et al., 2010)	Moon			X	X							X				X	X					X	
	(LU et al., 2012)	BitDew-MR			X						X		X				X	X				X	X	
	(KRISH; ANWAR; BUTT, 2014)	HATS				X						X	X	X	X								X	
	(ANJOS et al., 2015)	MRA++				X								X				X			X		X	
Sistemas Híbridos	(DING et al., 2011)	Hadoop Streaming				X														X				X
	(TUDORAN et al., 2014)	SC-CC		X										X									X	
	(ALEXANDROV et al., 2014)	FLINK				X			X		X	X				X				X	X		X	X
	(CLEMENTE-CASTELLÓ et al., 2015)	Hybrid IaaS Cloud					X				X												X	
	(TANG; HE; FEDAK, 2015)	HybridDFS		X	X			X		X	X	X		X	X		X					X	X	
	(VASILE et al., 2015)	HySARC^2					X			X											X			
	(GHAFARIAN; JAVADI, 2015)	Cloud-aware		X	X			X		X		X		X				X			X			
	(PHAM et al., 2016)	Cirus					X		X							X			X	X			X	X
	(Anjos, 2017)	**SMART**	X					X	X	X	X	X		X			X	X	X		X	X	X	X

C.3 Parcerias Internacionais

Outra contribuição foi estabelecer uma ligação entre grupos de pesquisa que estimulam a cooperação e o desenvolvimento de uma rede de pesquisa com pesquisadores envolvidos com problemas semelhantes. Este trabalho pode examinar um modelo com várias fontes de dados, variando de sensores sem fio com informações de interação de usuário em ambientes abertos, tais como redes sociais, informações de *open data* e outros. Os conjuntos de dados consistem de bases de dados de grandes corporações até mídias de difusão, onde há uma clara necessidade de normalização. A ideia vem evoluído

para uma plataforma de Big Data híbrida para o processamento de aplicações em diferentes domínios. Por causa disso, algumas instituições têm demonstrado interesse na plataforma SMART. Assim, um "projeto de cooperação" foi criado para alcançar este vasto domínio. Vários instituições envolvidos na pesquisa, estão resumidas em na Tabela C.3. Ela fornece uma lista dos principais pesquisadores que estão envolvidos no trabalho sobre a plataforma SMART.

Table C.3: Colaborações

instituições	Grupos	Localidade	Pesquisador
INRIA/ENS Lyon	LIP	Lyon, France	PhD. Gilles Fedak
INRIA/ENS Lyon	LIP	Lyon, France	PhD. Marcos Dias Assunção
TUB	DIMA	Germany	PhD. Volker Markl
JohnMoores University		Liverpool, England	PhD. Rubem Pereira
UnB	LASP	Brasilia, Brazil	PhD. João Paulo da Costa
UFRGS		Rio Grande do Sul, Brazil	PhD. Edison Pignaton
Belarusian State University		Minsk, Belarus	PhD. Tatiana Galibus
Danube University Krems		Austria	PhD. Thomas J. Lampoltshammer
Chinese Academy of Sciences	CNIC	Beijing, China	PhD. Haiwu HE

C.4 Publicações

As publicações relacionado com a Tese são:

Revistas:

Anjos, Julio C.S., Carrera, Iván, Kolberg, Wagner, Tibola, Andre Luis, Arantes, Luciana B., Geyer, Claudio R., ***MRA++: Scheduling and Data Placement on MapReduce for Heterogeneous Environments***, Future Generation Computer Systems Vol.42(0), 22–35, jan 2015

Anjos, Julio C. S., Fedak, Gilles, Geyer, Claudio F. R.: ***BIGhybrid: a simulator for MapReduce applications in hybrid distributed infrastructures validated with the Grid5000 experimental platform***, Concurrency and Computation: Practice and Experience 28(8), 2416–2439, June 2016, cpe.3665

Eventos:

Anjos, Julio C. S., Fedak, Gilles, Geyer, Claudio F.R.: ***BIGhybrid – A Toolkit for Simulating MapReduce in Hybrid Infrastructures***, Computer Architecture and High Performance Computing Workshop (SBAC-PADW), 2014 International Symposium on, UPMC - University Pierre et Marie Curie, 132–137, Oct 2014

Simonet, Anthony, Anjos, Julio, Fedak, Gilles, He, Haiwu, Tang, Bing, Lu, Lu, Jin, Hai, Shi, Xuanhua, Moca, Mircea, Silaghi, Gheorghe, Cheich, Asma Ben, Abbes, Heithem: ***D3-MapReduce: Towards MapReduce for Distributed and Dynamic Data Sets***, International Conference on Big Data Intelligence and Computing (DataCom 2015), 1 edition, IEEE, 20–30, Dec 2015

Anjos, Julio Cesar Santos, Filho, Bruno Reckziegel, Barros, Junior F., Schemmer, Raffael B., Geyer, Claudio, Matte, Ursula: ***Genetic Mapping of Diseases through Big Data Techniques***, Proceedings of the 17th International Conference on Enterprise Information Systems, 279–286, 2015

Anjos, Julio C. S., Assuncao, Marcos D., Bez, Jean, Geyer, Claudio F. R., de Freitas, Edison Pignaton, Carissimi, Alexandre, Costa, João Paulo C. L., Fedak, Gilles, Freitag, Felix, Markl, Volker, Fergus, Paul, Pereira, Rubem: ***SMART: An Application Framework for Real Time Big Data Analysis on Heterogeneous Cloud Environments***, Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/ DASC/PICOM), 2015 IEEE International Conference on, 199–206, Oct 2015

Galibus, Tatiana, Anjos, Julio C.S., de Freitas, Edison Pignaton, Geyer, Cláudio F. Resin, Fedak, Gilles, Jr., Rafael Timóteo de Sousa, Costa, João Paulo C. L., Pereira, Rubem, Fergus, Paul, Zaleski, Anton, Vissi,

Herman, Markl, Volker: ***Security Framework for Distributed Data Processing***, International Conference on Pattern Recognition and Information Processing (PRIP'16), Belarusian State University, 13th edition, October 2016

C.5 Trabalhos futuros

Diversas atividades são necessários para construir a plataforma SMART em um ambiente real. Em primeiro lugar, é necessário concluir as avaliações do modelo de custos e de interferência de I/O definido nesta tese. Os algoritmos para o *Global Aggregator* precisam ser definidos e implementado. O *Global Dispatcher* precisa ser implementado e então ajustar com o *Global Collector* e os motores grandes de dados. Em particular, uma possível estratégia para aplicar no módulo Dispatcher é criar um escalonador dinâmico para fornecer uma prioridade de execução e evitar o atraso no fluxo de tarefa como é apresentado anteriormente.

O mecanismo de tomada de decisão pode ter algumas estratégias desenvolvidas nesta Tese como uma heurística de decisão por exemplo. O módulo de segurança pode ter seus algoritmos estudados para um aumento de desempenho. Os primeiros estudos estão nos anexos. No módulo *Global Collector*, os serviços de repositório precisam ser integrados com a API do SMART e criar uma interface REST para a comunicação de sensores em baixo nível. Um estudo da qualidade das informações fornecida pelos sensores deve permitir o uso de dados de destes dispositivos agrupadas em zonas em sensoriamento urbana para usar com a análise Big Data em aplicações de cidades inteligentes. Este estudo é importante para a apresentação do projeto futuro com alguns parceiros.

A central de monitoramento e a interface de usuário final devem ser especificadas por uma equipe multidisciplinar, incluindo designers especialistas em computação gráfica. Finalmente, o mecanismo de armazenamento deve ser avaliado para incluir a interferência de I/O e uma avaliação de técnicas de *container*. Atualmente, estamos escrevendo um artigo para uma revista sobre as conclusões da tese.