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

EMMANUELL DIAZ CARREÑO

Migration and Evaluation of a Numerical Weather Prediction Application in a Cloud Computing Infrastructure

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

Advisor: Philippe Olivier Alexandre Navaux

Porto Alegre
September 2015

CIP – CATALOGING-IN-PUBLICATION

Diaz Carreño, Emmanuell

Migration and Evaluation of a Numerical Weather Prediction Application in a Cloud Computing Infrastructure / Emmanuell Diaz Carreño. – Porto Alegre: PPGC da UFRGS, 2015.

84 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2015. Advisor: Philippe Olivier Alexandre Navaux.

1. Cloud Computing. 2. High Performance Computing. 3. E-Science. 4. Numerical Weather Prediction. I. Alexandre Navaux, Philippe Olivier.

"The strongest of all warriors are
these two — Patience and Time."
— Leo Tolstoy

ACKNOWLEDGMENTS

First I want to thank my family for the support they have always given to me while I have been away.

I also thank my advisor Philippe Navaux for the opportunity he gave me to work with him and his research group, also by the advice and assistance received during the postgraduate course.

I want to express my sincere gratitude to my friends in the GPPD research group, for providing me a great company and helping me with suggestions for my research and my Portuguese.

I also like to extend my gratitude to the Federal University of Rio Grande do Sul (UFRGS), and the Institute of Informatics (INF) for the high academic quality. Also to the CNPq for the scholarship and the opportunity as a foreign student.

To my friends and colleagues in Colombia for keeping in touch and their words of support, a sincere and thankful hug.

LIST OF ABBREVIATIONS AND ACRONYMS

AFS	Azure File Service
BLAST	Basic Local Alignment Search Tool
BRAMS	Brasilian Regional Atmospheric Modeling System
CCATT	Coupled Aerosol and Tracer Transport
CPTEC	Centro de Previsão de Tempo e Estudos Climáticos
CLI	Command Line Interface
CMP	Cloud Migration Point
DNS	Domain Name System
FhGFS	Fraunhofer Parallel File System
FTP	File Transfer Protocol
GPPD	Grupo de Processamento Paralelo e Distribuído
GRS	Geo-redundant storage
HPC	High-Performance Computing
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IEEE	Institute of Electrical and Electronics Engineers
INPE	Instituto Nacional de Pesquisas Espaciais
ISO	International Standards Organization
LAN	Local Area Network
LRS	Local Redundant Storage
LTS	Long Term Support
MBRAMS	Multiple BRAMS deployments

MPI	Message Passing Interface
NAS	Numerical Aerodynamics Simulation
NIST	National Institute of Standards and Technology
NPB	NAS Parallel Benchmarks
NWP	Numerical Weather Prediction
OS	Operating System
QoS	Quality of Service
RAMS	Regional Atmospheric Modeling System
SaaS	Software as a Service
SMB	Server Message Block protocol
SSH	Secure Shell
URL	Uniform Resource Locator
USD	United States Dollar
VHD	Virtual Hard Disk
VM	Virtual Machine
VPN	Virtual Private Network

LIST OF FIGURES

LIST OF TABLES

CONTENTS

ABSTRACT

The usage of clusters and grids has benefited for years the High Performance Computing (HPC) community. These kind of systems have allowed scientists to use bigger datasets and to perform more intensive computations, helping them to achieve results in less time but has also increased the upfront costs associated with this area of science. As some e-Science projects are carried out also in highly distributed network environments or using immense data sets that sometimes require grid computing, they are good candidates for cloud computing initiatives. The Cloud Computing paradigm has emerged as a practical solution to perform large-scale scientific computing. The elasticity of the cloud and its pay-as-you-go model presents an attractive opportunity for applications commonly executed in clusters or supercomputers. In this context, the user does not need to buy infrastructure, the resources can be rented from a provider and used for a period of time. This thesis presents the challenges and solutions of migrating a numerical weather prediction (NWP) application to a cloud computing infrastructure. We performed the migration of this HPC application and evaluated its performance in a local cluster and the cloud using different instance sizes. We analyzed the main characteristics of the application running in the cloud. The experiments demonstrate that, although processing and networking create a limiting factor, storing input and output datasets in the cloud presents an attractive option to share results and ease the deployment of a test-bed for a weather research platform. Results show that cloud infrastructure can be used as a viable HPC alternative for numerical weather prediction software.

Keywords: Cloud Computing. High Performance Computing. e-Science. Numerical Weather Prediction.

Migração e Avaliação de uma Aplicação de Previsão Numérica do Tempo em uma Infrastructura de Computação em Nuvem

RESUMO

O uso de clusters e grids tem beneficiado durante anos a comunidade de computação de alto desempenho (HPC). O uso deste tipo de sistemas tem permitido aos cientistas usar conjuntos de dados maiores para executar cálculos mais complexos. A computação de alto desempenho tem ajudado para obter aqueles resultados em menos tempo, mas aumentou o custo das despesas de capital nesta área da ciência. Como alguns projetos de e-science são realizados também em ambientes de rede altamente distribuídos, ou usando conjuntos de dados imensos que muitas vezes requerem computação em grade, eles são muito bons candidatos para as iniciativas de computação em nuvem. O paradigma Cloud Computing surgiu como uma solução prática com foco comercial para realizar computação científica em larga escala. A elasticidade da nuvem e o modelo pay-as-you-go apresenta uma oportunidade interessante para aplicações comumente executados em supercomputadores ou clusters. Esta tese apresenta e avalia os desafios da migração e execução da previsão numérica de tempo (NWP) numa infra-estrutura de computação em nuvem. Foi realizada a migração desta aplicação HPC e foi avaliado o desempenho em um cluster local e na nuvem utilizando diferentes tamanhos de instâncias virtuais. Analisamos as principais características da aplicação executando na nuvem. As experiências demonstram que, embora o processamento e a rede criam um fator limitante, o armazenamento dos conjuntos de dados de entrada e saída na nuvem apresentam uma opção atraente para compartilhar resultados e facilitar a implantação de um ambiente de ensaio para investigação meteorológica. Os resultados mostram que a infraestrutura de nuvem pode ser usada como uma alternativa viável de HPC para software de previsão numérica do tempo.

Palavras-chave: Computação em Nuvem, Computação de Alto Desempenho, e-Science, Previsão Numérica do Tempo.

1 INTRODUCTION

Computer generated weather forecast is a continuous developing area that has been executed mostly on grids or supercomputers. Applications in climate research typically require huge amounts of data and computing power (SOUTO et al., 2007). This large-scale scientific computing has been performed per years in costly machines that are out of the possibilities of many research groups. High performance computing (HPC) requires a large number of processors interconnected and large data storage. With the developments in weather services, the demand for more accuracy and time reduction for weather forecast becomes higher, which as a result brings on the growth of computation complexity. Generally speaking, it is difficult to address such problems even by adopting high performance servers (IOSUP et al., 2011).

A common computing infrastructure for HPC is the grid, but this infrastructure is difficult to setup and maintain. For the execution of such applications in the context of a grid computing platform, it is necessary not only to share powerful computational resources, but also to have available an efficient mechanism to store and transfer required files related to the climate data sets. However, most of the time those systems are underutilized and produce a financial burden by upfront costs and electricity. Asking for access to utilize those HPC machines requires bureaucracy and reserving time slots that most of the time slow the pace of scientific research (VECCHIOLA; PANDEY; BUYYA, 2009). This computing approach has achieved much, but can also be a significant limitation to progress (EVANGELINOS; HILL, 2008).

On the other hand, the cloud computing paradigm provides an alternative to access large infrastructures and resources. In the cloud, the possibility of paying only for the used amount of resources brings convenience for academia. The pay-as-you-go concept could be transformed into a viable option for an institution lacking computing resources.

On-demand cloud computing offers an attractive new dimension to HPC, in which virtualized resources can be borrowed. These resources could be used in a customized manner to target a particular scenario, at the time and in the form the user desires them. However, it is clear that one of the major concerns for the HPC community is performance.

Some HPC initiatives have started utilizing Cloud Computing for researching. Nevertheless, performance concerns of HPC applications in the Cloud are considerable (BENEDICT, 2013). For weather research, the need to reduce the waiting time found in the batch jobs schedulers common in HPC infrastructure, the possibility of quickly access and share the processed data, and the development of newer simulations or ensemble forecasts has motivated them to use the Cloud. A trend already started as the request for funding of cloud storage resources are becoming increasingly common on grant proposals, rather than contributions to capital spending on hardware (YANG et al., 2014).

1.1 Motivation

The economics related to hardware investment budgets and the need for energy efficiency has motivated research in the usage of new technologies to optimize costs, cloud computing is one of them. Cloud computing provides cost-effective, fast, and a vast amount of virtualized resources for large-scale applications, it is used as utility computing where computing, storage, and networking services are provided on-demand.

Weather forecasting has been benefited by the computing area for years, the usage of computers has allowed them to use more variables in their simulations and more data sources to perform precise predictions. High performance computing had help to get those result in less time but has increased the cost associated with this area of science.

Previous work in the migration of HPC applications to the cloud looks at small HPC benchmarks, which are simple to port or run in most cases. We look at a full and complex HPC application, with difficult requirements as most of the software used in HPC has been optimized to run on specific architectures and require a set of conditions to run in a proper and efficient way. An ongoing problem with those legacy applications is the need to port them to new infrastructures. The usage of virtualization may solve some of the challenges. The migration of this scientific application is a great opportunity for the development of an e-Science weather service on a cloud platform, that could help weather research centers with their processing needs.

1.2 Objectives

The main purpose of this work is to migrate the Brazilian Regional Atmospheric Modeling System (BRAMS) to a new platform that uses the advantages provided by a cloud computing infrastructure and evaluate the challenges of this procedure. It is important to evaluate if cloud computing could be used as an environment for this scientific application. Also we developed a workflow to test the possibility of sharing research results between various weather research centers. To produce a meaningful result, we performed a series of experiments regarding processing and networking performance and storage availability.

In order to achieve the proposed goal, the main steps to be performed are:

- Migrate BRAMS to Infrastructure as a Service (IaaS) cloud model
- Measure the performance of the migrated version
- Analyze the cost of execution to achieve useful results
- Study the viability of using BRAMS in the cloud simultaneously by more than one deployment, sharing datasets between them

The main contributions of this thesis are, first, we share the experience obtained in porting a legacy application to a cloud environment. Second, we create a workflow explaining the modifications required to perform numerical weather simulations on a cloud infrastructure using BRAMS, introducing a cloud lifecycle control. Third we evaluate the performance of this ported application in the cloud by comparing it to a cluster of local machines. Finally, we developed a test bed to perform weather forecast using shared input datasets, evaluating the impact of this approach.

1.3 Text Organization

This document is organized as follows. The next Chapter presents a background on the topics related to this thesis and discusses related work in high performance using cloud computing. The scientific application migrated to the cloud is presented in Chapter 3. Chapter 4 describes the migration procedure and highlights the solutions proposed to achieve this objective. In Chapter 5, the proposed evaluation methodology is presented. Chapter 6 discusses the experimental results obtained. Finally, Chapter 7 presents the conclusion of this work and some insights on future work.

2 BACKGROUND

Some important concepts used in this thesis are explained in the following sections. These concepts are related to the cloud computing paradigm, high performance computing and related work about the approaches to executing HPC applications in the cloud.

2.1 Cloud Computing

Cloud computing is a recent paradigm of computing. It was developed with the combination and evolution of distributed computing and virtualization, with strong contributions from grid and parallel computing. There were many efforts to provide a definition of cloud computing, such as the work of Grid Computing and Distributed Systems Laboratory (VECCHIOLA; PANDEY; BUYYA, 2009) and the initiative from Berkeley University (FOX et al., 2009). In 2011, The National Institute of Standards and Technology (NIST), an American Institute, published its definition consolidating several studies (MELL; GRANCE, 2011) which was quickly adopted by the community. However, it was only in 2014 that *Cloud Computing* was finally accepted as an international standard and received a detailed and extended definition by the ISO (ISO/IEC, 2014).

As defined by the ISO standard, cloud computing is a paradigm for enabling network access to a scalable and elastic pool of shareable physical or virtual resources with self-service provisioning and administration on-demand. The categories and capabilities allow the selection of the subset of cloud services that could benefit the user projects. Table 2.1 shows the multiple options regarding cloud services and categories.

The cloud computing model seems the natural evolution for running scientific and high performance applications (FOSTER et al., 2008). An important issue is that because environments are virtualized, there is a performance loss problem due to this technology (RAMAKRISHNAN et al., 2011; HUBER et al., 2011). Performance degradation is more common in certain types of applications, such as applications

Table 2.1 – Cloud service categories and cloud capabilities types.

Cloud service categories	Cloud capabilities types		
	Infrastructure	Platform	Application
Compute as a Service	●		
Communications as a Service		●	●
Data Storage as a Service	●	●	
Infrastructure as a Service	●		
Network as a Service	●	●	●
Platform as a Service		●	
Software as a Service			●

with a large amount of communication between processes or threads. However, a new approach to encapsulate applications is growing since the last year. The technology is named virtualization containers. These containers share the physical resources in a simplified way, each container including a complete operating system and only the dependencies for the application. The container is a wrapper for the application, it does not try to emulate an actual machine. In the containers, everything happens at the application level, reducing the number of layers of complexity and with better performance than machine-level virtualization. It is, therefore, an attractive option for cloud deployments.

The economics related to hardware investment budgets and the need for energy efficiency has motivated research in the usage of new technologies to optimize costs, cloud computing is one of them. Another factor that must be considered is the effort to migrate applications to the cloud because they are mostly large applications with many lines of code and many specific settings and requirements (CARREÑO; ROLOFF; NAVAUX, 2015). Therefore, migrating an application before conducting a viability study of how it will behave in the new environment is not the best path to proceed in most cases.

2.2 Microsoft Azure

Multiple Cloud providers exist such as Amazon, Rackspace, Microsoft Azure. For this thesis, Microsoft Azure was chosen because previous work performed by (ROLOFF et al., 2012a) showed that performance and cost efficiency for HPC in Azure was better compared to Rackspace and Amazon. Microsoft started its initiative in Cloud Computing with the release of Windows Azure at its annual professional developer conference in 2008. The initial model was Platform as a Service, allowing developing and running applications written in the programming languages supported by the .NET framework. Nowadays, the Azure infrastructure covers all types of service models and have gained a significant market share. The number of datacenters is continuously growing as Microsoft has committed to the cloud as a part of its business strategy. Investments in datacenters and services from most of the cloud providers have create an intense competition for the cloud market, lowering the Cloud service prices in the last years. The datacenters locations actually in production and some planned for the next years are depicted in Figure 2.1. The service prices vary from location to location, because of this factor, selecting the best for each application could reduce or increase costs significantly.

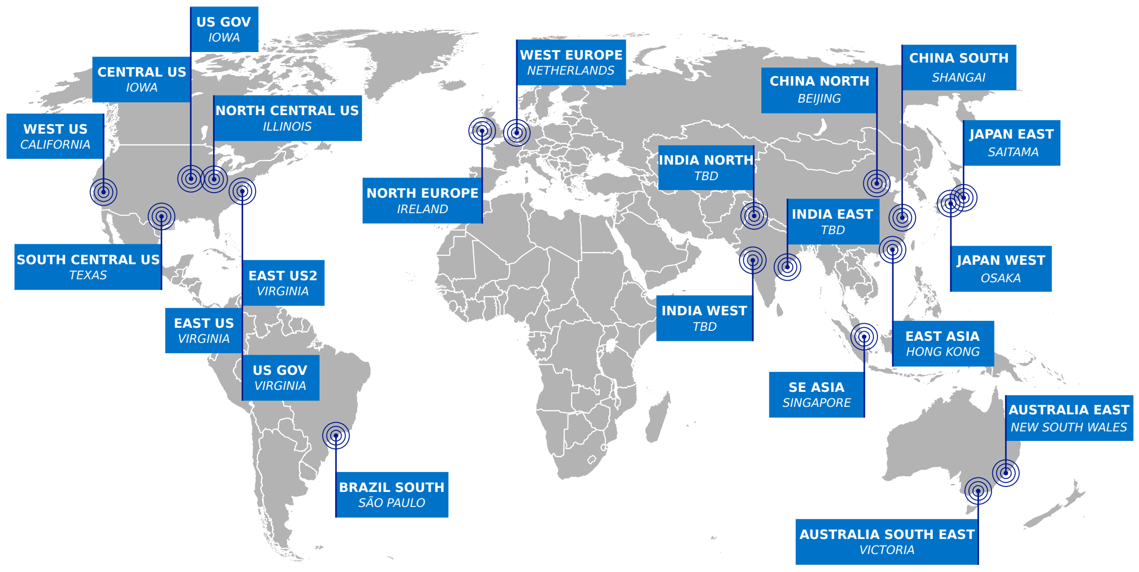


Figure 2.1 – Azure datacenter geographic locations. Some datacenters include more than one logical location name on Azure infrastructure. India datacenters are at the design stages. (Adapted from azure.com)

We are interested in IaaS model, one of the cloud service models provided to access and manage Virtual Machines (VMs) running on Microsoft's infrastructure. Azure offers multiple options regarding IaaS [1]. Each set of characteristics and configuration of a VM is called *instance size* in Azure services. Table 2.2 shows the options regarding the instance sizes. Some of the instances sizes are very interesting for HPC, especially the ones with Infiniband. Unfortunately, Infiniband was available only to Windows virtual machines using a proprietary MPI driver.

The user has a set of base images of Windows and Linux OS provided by Microsoft, but other images can be created and uploaded to the service. The user can also configure a virtual instance image directly into Azure and capture it to use locally or to redeploy it. Users can customize the environment for their application by installing specific software or by purchasing particular machine images. Developers

[1]http://msdn.microsoft.com/en-us/library/azure/dn197896.aspx

Table 2.2 – Instance sizes available on Microsoft Azure for IaaS. Some instances sizes are not available initially in all datacenters. Instances marked with an asterisk were not available when our experiments were performed.

Description	Name	CPU Cores	Memory [GB]	Disk Size [GB]
General purpose compute: Basic tier	A0	1	0.75	20
	A1	1	1.75	40
	A2	2	3.5	60
	A3	4	7	120
	A4	8	14	240
General purpose compute: Standard tier	A0	1	0.75	20
	A1	1	1.75	70
	A2	2	3.5	135
	A3	4	7	285
	A4	8	14	605
	A5	2	14	135
	A6	4	28	285
	A7	8	56	605
Fast networking: 10Gigabit + Infiniband	A8	8	56	382
	A9	16	112	382
Performance compute*: Latest CPUs, more memory, and local SSD	G1	2	28	412
	G2	4	56	825
	G3	8	112	1650
	G4	16	224	3300
	G5	32	448	6600
Compute and Memory: 60% faster CPUs, more memory, and local SSD	D1	1	3.5	50
	D2	2	7	100
	D3	4	14	200
	D4	8	28	400
	D11	2	14	100
	D12	4	28	200
	D13	8	56	400
	D14	16	112	800

can specify the number of VM instances at the deployment of their application or can dynamically adjust the number of instances at runtime.

Regarding resource allocation in Azure infrastructure, the service is limited by a *subscription*, defined by the paying capacity of the user. This subscription model allows setting quotas on the number of CPU cores, storage accounts and cloud services.The subscription also set some limitations on the parallel operations that could be performed while deploying or managing the operations of the instances.

Using IaaS requires SSH to access the virtual machines deployed using a Linux distribution. In that sense is not a big difference in usability as using a cluster or grid for users with some experience. For that reason, this aspect of the porting, usability would be almost the same for experienced BRAMS users.

2.3 High Performance Computing

High Performance Computing (HPC) is the use of state-of-the-art computational techniques to exploit the power of hardware to the limit of current processing capacity. It plays a crucial role in supporting computational science, now established as the "third pillar" of scientific inquiry alongside theory and experimentation (REED et al., 2005). According to the International Data Corporation, HPC is closely linked to scientific advances, industrial innovation, and economic competitiveness.

Simulation and modeling are routinely used to guide experiments and validate the theories. Moreover, they can predict phenomena outside the scope of current experimental methods and are in some cases a safer and cheaper alternative to experimentation. Some fields now rely entirely on HPC to provide basic research capability. The United States Government's identified big data as one of the eight great technologies that will further increase the need for HPC to perform large-scale data analysis.

Nowadays HPC forms part of the fundamental research infrastructure that the research committees, funding organizations, and potential staff and students expect that universities provide. The computational HPC infrastructure consists of a hierarchy of hardware resources where each level supports larger applications or more advanced models than the one below. Institutional HPC services play a significant role in bridging the gap between research group computing resources at the bottom and national and international HPC facilities at the top.

Currently, the computer system with greater processing capacity is the Chinese Tianhe-2, a machine that reaches 54 petaflops with its 3,120,000 cores and one petabyte of memory (TOP500, 2014). But the demand for more performance keeps growing (DONGARRA et al., 2011), and the HPC community expects to reach exascale computing performance by the year 2018 (BROWN et al., 2010). However, to achieve this level of performance, some significant challenges must be overcome (SIMON, 2010). Those challenges are the acquisition cost and energy requirements of such systems. The economics related to hardware investment budgets and the need for energy efficiency has motivated research in the usage of new technologies to optimize costs, cloud computing is one of them.

2.4 HPC Applications in the Cloud

Regarding the execution of HPC application using the cloud, (EVANGELINOS; HILL, 2008) used Amazon Elastic Computing Cloud (EC2) to run an atmosphere-ocean climate model in an on-demand cluster created for this task by them. They found that the performance in the cloud was below the level of a supercomputer, but comparable to a low-cost cluster system. As their primary concern was related to latency and bandwidth, they conclude that even if the results were encouraging, the future usage of Myrinet or Infiniband for the interconnections could help reduce this performance gap.

In another field, (LANGMEAD et al., 2009) executed a DNA sequencing algorithm in Amazon EC2, their work focused on executing a Hadoop application analyzing the costs and the scalability of the application. They concluded that by taking advantage of cloud computing services, it was possible to condense thousand of computing hours into a few hours without owning a cluster.

An evaluation of the performance of HPC applications on Microsoft Azure was performed by (ROLOFF et al., 2012b). Their work showed how a migrated suite of benchmarks performed in this cloud environment. They concluded that performance is acceptable in that cloud service, and that is a promising platform for science.

A case study of running data-intensive research in the cloud was performed by (LU; JACKSON; BARGA, 2010). They used a BLAST algorithm for bioinformatics. They found that, in some cases, having the ability to abort a long-running job is more desirable than waiting for a failing node to recover. They concluded that while high performance is often considered desirable, scalability and reliability are usually more important for scientific applications.

A satellite propagation and collision system was implemented by (JOHNSTON; COX; TAKEDA, 2011) in Microsoft Azure. They developed a scalable architecture that provided burst capabilities to their algorithm. They conclude that there is great potential for cloud-based architectures to deliver cost-effective solutions and confront large-scale problems in science, engineering, and business. (SIMALANGO; OH, 2010) performed a feasibility study on using the cloud to process large data sets. They concluded that cloud environments could provide some benefits like better resource utilization. They also raise a warning about the costs of performing massive computation on a public cloud.

(HERNÁNDEZ et al., 2013) evaluate whether cloud computing could be an alternative environment to annotate educational content. They remark that the cloud is useful for particular situations like fault recovery and dealing with demand peaks. They conclude that the usage of this paradigm give an opportunity to increase the reliability of scientific computations thanks to the high availability of the cloud, which is much higher than what a real-world cluster and grid provide.

(EVOY; SCHULZE, 2011) performed an analysis of the implications of the scheduling on the virtualized resources of a cloud service. They found that even when the virtualization overhead is minimal for raw computation combined workloads in virtualized hardware produce complex interactions between computational resources.

Regarding the deployment, performance and cost efficiency of HPC applications in the cloud, Roloff et al. (ROLOFF et al., 2012a) created a model to establish the breaking point to perform HPC in the cloud. They conclude that usage of cloud for this kind of experiments could be restricted by the actual amount of utilization compared to on-premises deployment.

(HUMPHREY et al., 2011) performed an analysis of the value of cloudbursting. Cloudbursting allows expand dynamically the local computing resources by temporary adding cloud resources for large-scale applications. The application processed environmental results from satellite imagery. They compared three versions of the application to identify which properties of the application determine the potential benefits of cloudbursting.

2.5 Migrating Applications to the Cloud

Migration in the cloud computing context means taking an application from an on-premises deployment to a cloud environment. Because scientific and HPC applications require specific settings to run, minimal modifications to the source code may be well adjusted to a migration process in order to prevent a significant impact on the development of the application. Menychtas et al. (MENYCHTAS et al., 2013) developed a migration model to automatize the migration analysis of business applications, mostly web applications. They used source code analysis and reverse engineering to determine how much does an application has to change to be migrated successfully to the cloud. Their methodology considers both technical and business aspects of the legacy applications. However, they focus on the cost of re-factoring and re-developing code, not the migration of applications as they are used in the original environments. Others take this refactoring approach too (LV; LIU, 2013; ANDRIKOPOULOS et al., 2013; WARD et al., 2010; TAK; TANG, 2014).

An economic approach for migration was developed by (TRAN et al., 2011), they proposed Cloud Migration Point, a modification of a well-known software size estimation model called Function Point for estimating the size of cloud migration projects. They validated their model as a predictor of effort estimation for cloud migration projects and concluded that software size is an important measure for legacy-to-cloud migration projects.

Numerous models deal with the economics of cloud adoption while disregarding the equally important performance evaluation. Others focus their work on the economic aspect of the migration (HAJJAT et al., 2010; BESERRA et al., 2012; HERNÁNDEZ et al., 2013; HOHENSTEIN et al., 2013; WU et al., 2014; SIMALANGO; OH, 2010).

There are currently no major migration approaches or models available that are sufficiently applicable to most HPC or eScience applications.

2.6 Summary

Cloud computing provides cost-effective, fast, and a considerable pool of virtualized resources for large-scale applications, it is used as utility computing where the computing, storage, and networking services are provided on-demand and as needed. The economics related to hardware investment budgets and the need for energy efficiency has motivated more research in executing applications on the cloud. Cloud computing presents an attractive model that solves some of the problems that arise when you purchase a system for HPC.

By studying related work, we observe that previous research refer mostly to porting applications to existing cloud environments and performance evaluation of cloud systems. Some migration work focus on the economic aspect of moving to the cloud, but most of them only analyze web applications and their requirements, databases, storage, etc. Other works use weather models only to benchmark HPC performance versus cloud, but none of them evaluates the deployment of the application or how user the constraints can accept the final results as valid for their experiments.

3 BRAZILIAN REGIONAL ATMOSPHERIC MODELING SYSTEM (BRAMS)

One of the most widely employed numerical models in regional weather centers in Brazil is the Brazilian Regional Atmospheric Modeling System (BRAMS) (CPTEC-INPE, 2014), a model based on the Regional Atmospheric Modeling System (RAMS) (PIELKE et al., 1992). RAMS, was developed by the Atmospheric Science Department at the Colorado State University, it is a numerical weather prediction model used to simulate atmospheric conditions. The developments integrated into BRAMS include computational modules modified to simulate in a better way the tropical atmosphere. The primary objective of BRAMS is to provide a single model to brazilian regional weather centers.

The execution of BRAMS requires downloading the input data to create the initial conditions and the satellite data pre-processed by the Centro de Previsão de Tempo

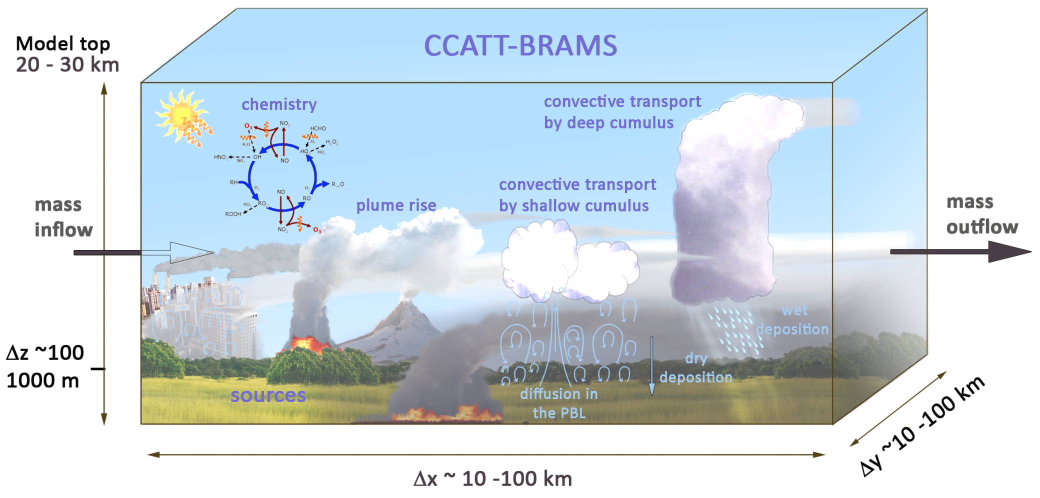


Figure 3.1 – Illustration of BRAMS regional model depicting the interactions between atmospheric variables (CCATT-BRAMS) (LONGO et al., 2013).

e Estudos Climáticos (CPTEC) using their global and regional model. The output of CPTEC contains the complete South American territory. Also, CPTEC publishes other input data that BRAMS need to execute, like fire location, environmental chemistry among others. A generic overview of the BRAMS model and its submodels is depicted on Figure 3.1.

Multiple weather research centers in South America use the consolidated input datasets produced by CPTEC for their Numerical Weather Prediction (NWP) simulations. NWP is the usage of mathematical models to predict accurately atmospheric behavior for a future period. The results are denominated weather or climate forecasts, a few days for the weather, months for the climate. (SOUTO et al., 2007)

Some of these research centers also use BRAMS in grid environments for their regional forecasts. Because input datasets concentrates the data for the unified continent, each weather prediction center have to download that dataset for the entire South America region, even if they are going to use just a part of the dataset for their regional forecasting.

3.1 BRAMS workflow

As depicted in Figure 3.2, one execution of BRAMS consists of a pre-processing stage, three main stages to generate the forecasts files and a post-processing stage to produce the forecast visualization. These stages need to be executed in order, a more detailed description of each one of them and how their characteristics were ported to the cloud are described in the following subsections.

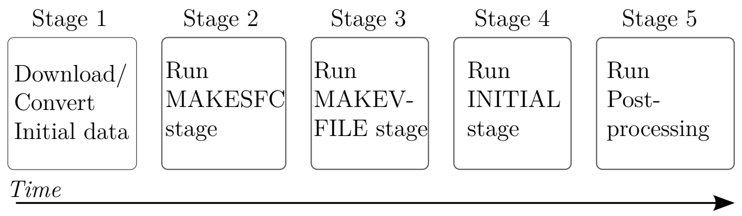


Figure 3.2 – Workflow of BRAMS running in a local environment.

We used BRAMS version 5.0 Beta, available online directly on the website of CPTEC [1]. Test-case data is also available at the same website.

[1]<http://brams.cptec.inpe.br>

3.1.1 Stage 1: Download and Convert Initial Data

The preprocessing stage of executing BRAMS depends on various factors. If it is the first time the model is going to be executed to be forecasted, data must be already available in the storage system. For this task, some scripts download the up-to-date input data needed for the next stages. Some files are available in a format not compatible with the one used by BRAMS, because of this, they need to be converted to BRAMS format.

3.1.2 Stage 2: MAKESFC

The second stage is the first processing stage, it is called MAKESFC. In this stage the global data files of soil type, sea surface temperature and topography are converted to files that cover only the area that is going to be forecasted. In Figure 3.3 is depicted the input data in the upper part. Some data is downloaded only once as the topography data, another data is not available from recent dates as the vegetation index. Asterisks are used to clarify that the output data is a modified version of the input data, which is the reason that the name it is the same as in the input.

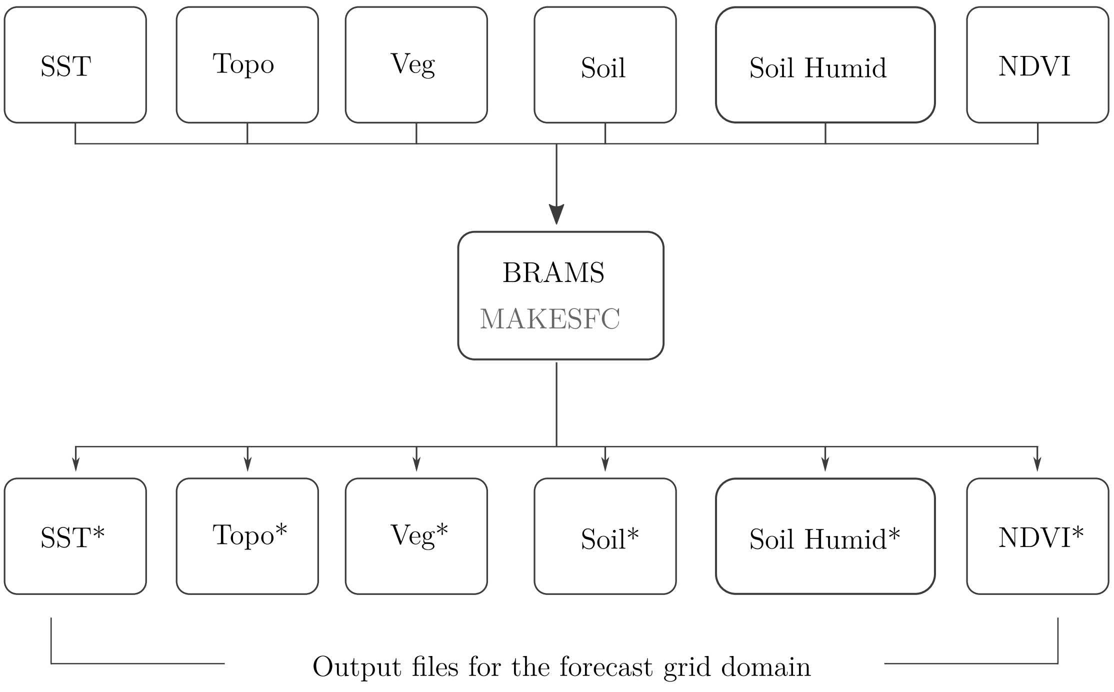
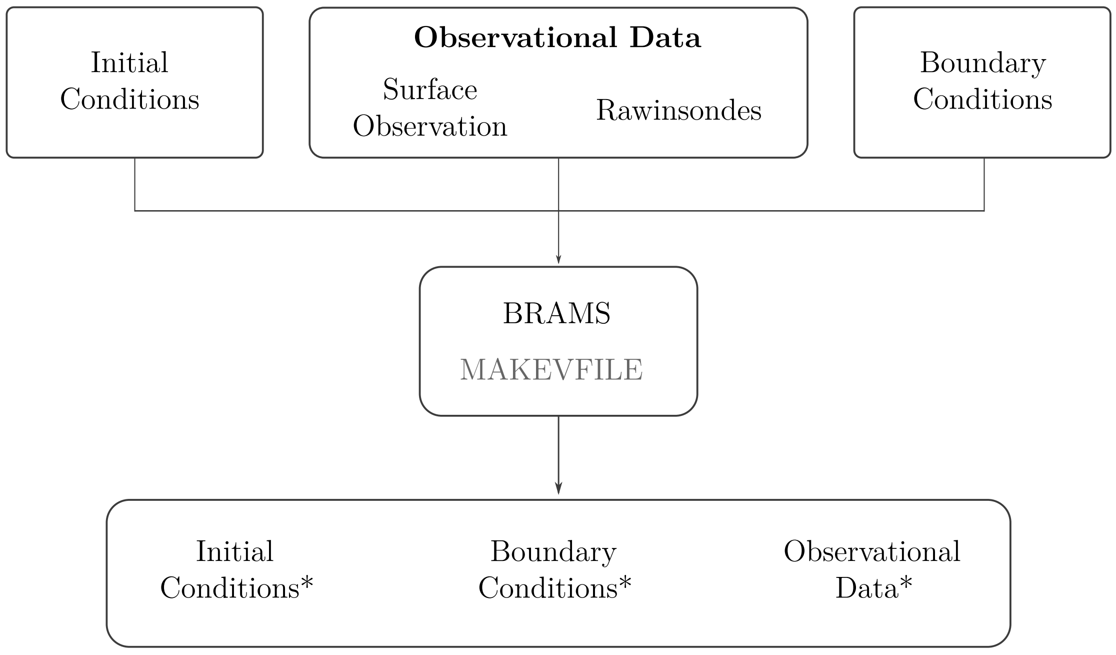


Figure 3.3 – MAKESFC stage. Depiction of the input and output data flow.

BRAMS performs this stage in a sequential fashion, so it is performed by only



Output files for the forecast grid domain

Figure 3.4 – MAKEVFILE stage. The depiction of the input and output data of this stage.

one process in the workflow of the application. The output of this stage produces data restricted to the grid domain configured on the model. After this stage finishes BRAMS does not continue automatically, it is necessary to reconfigure the model settings file to perform the next stages.

3.1.3 Stage 3: MAKEVFILE

The second processing stage, MAKEVFILE, generates files with the initial and boundary conditions of each grid in the area to be analyzed for the integration (forecasting) time, as shown in Figure 3.4. This stage performs data analysis over one or more observational datasets and produces variable initialization files, called *varfiles*. These *varfiles* contain the atmospheric fields defined on the model grids for the specified grid domain.

In this stage, also only one core performs the processing. The output files contain data restricted only to the grid domain configured on the model. Again, after this stage finishes BRAMS does not continue automatically to the next stage.

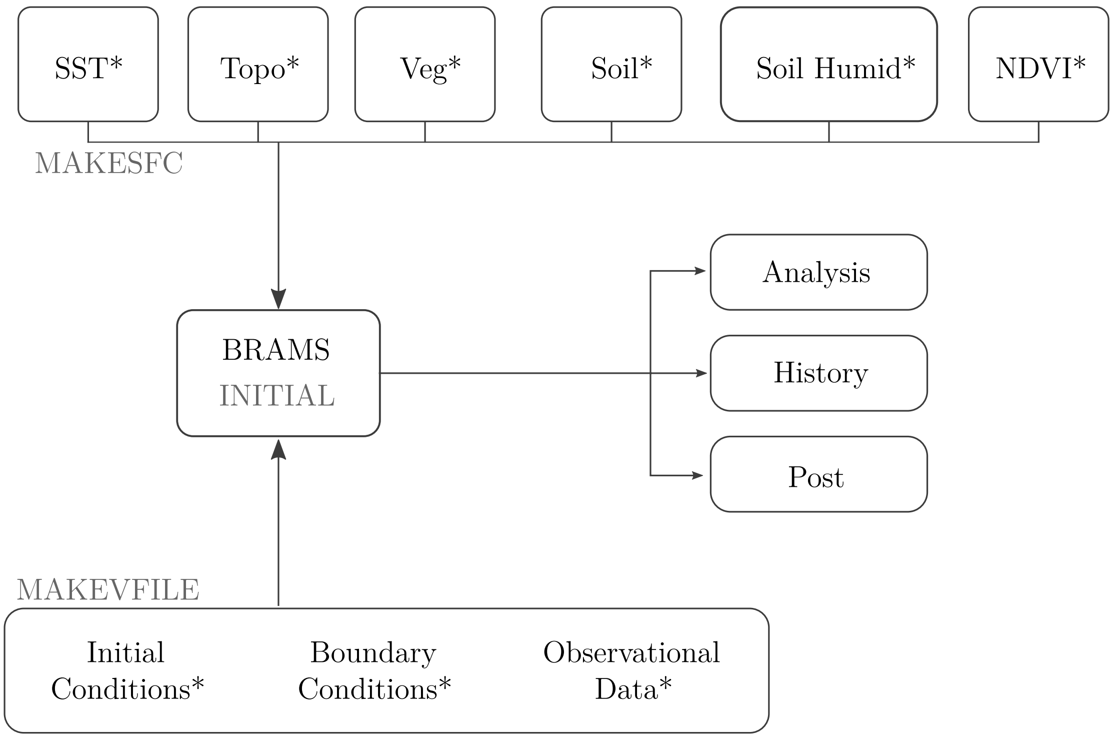


Figure 3.5 – INITIAL stage. The depiction of the input and output data of this stage.

3.1.4 Stage 4: INITIAL

The third of the processing stages is the actual simulation and forecast. As depicted on Figure 3.5 the output data from the MAKESFC and MAKEVFILE are used as input for the execution of this stage. This stage, called INITIAL, uses the number of cores available in the environment, and it is executed in parallel using MPI. The output of this stage contains the analysis files with the forecast data.

Also, depending on the configuration of the variables it could also save history files. These files allow check-pointing during the forecast process to continue in case something fails. If output variables are defined on the post-processing sections of the BRAMS configuration file, the files ready for post-processing are also saved. This stage is the most compute intensive of the three stages and also the one that generates the largest amount of data.

3.1.5 Stage 5: Post-Processing

After the forecast simulation has finished, a post-processing stage is performed on the output data to visualize the forecasts. For this step, a different application is used, not BRAMS. Multiple iterations over the output data are required to produce the visualization for each one of the weather variables defined in the model forecast configuration. This stage is also performed by only one CPU process.

Table 3.1 – Size of the input dataset files used for the experiments.

Content	**Size (GB)**
CLIMATOLOGY	0.52
DP	86.00
EMISSION-DATA	3.60
FIRES-DATA/DSA	0.05
FIRES-DATA/MODIS	0.18
SOIL-MOISTURE	23.00
SURFACE-DATA	8.80
Total	122.15

3.2 BRAMS datasets

BRAMS data dependencies are organized based on the frequency of their availability. Most of the data are generated every day and preprocessed by the INPE before being published. Another data are generated every week. Those datasets are used on the different stages of the workflow as depicted on Figures 3.3, 3.4 and 3.5.

For the experiments of this thesis the INPE help us by providing us the complete input dataset of the year 2009. The size of the entire dataset and the name of its input folders are shown on Table 3.1. DP files were already converted from GRIB format to the format used by BRAMS. Another set of input data was used for different experiments and will be described in Section refresults.

3.3 Summary

This chapter described the workflow of the scientific application we ported to the cloud environment. The primary objective of this chapter was to help to understand the workflow and the terminology used in the next chapters of this work. Explaining each one of the stages and presenting the amount of data used by the application allows getting insights on the requirements for the cloud. In the next chapter, we introduce the migration procedure and discuss the solutions proposed for this process.

4 MIGRATING BRAMS TO THE CLOUD

Migrating a scientific application to a cloud environment is a challenging task. The change in the amount of control of the environments and resources compared to executing the application over on-premises machines represents a limitation for some migration efforts. But the benefits of using the cloud should not be underestimated.

One of the concerns from migrating HPC applications is performance. As presented in Section 2, related work on moving an application into the cloud conclude that focusing on performance would yield acceptable results at most. But some applications presents other constraints besides raw performance. There are other aspects that should be analyzed using the cloud for scientific applications.

In numerical weather prediction (NWP), the results are valid only for a limited timeframe, that is the forecasted period. While porting the application, we focused not only on performance but also on how to run the forecast inside this timeframe while preserving the usability of the application and automating most of the procedures. Because we are using the Infrastructure as a Service cloud model (IaaS), the service provider would charge the usage of three main aspects: storage, networking and deployed time of every instance. We analyzed those meaningful aspects that could be charged while executing the application.

In this section, we discuss our approach to migrating the NWP application to the cloud. We present an analysis of the challenges and solutions we provide based on our findings while trying to migrate the application. After that, we explain how we created the BRAMS service in the cloud and how we managed this service for our experiments. Finally, we show how we modified the workflow of BRAMS to reduce the costs of porting and executing BRAMS in the cloud while maintaining the time to completion of the forecast into an acceptable timeframe.

4.1 Migrating BRAMS: Challenges and Solutions

In IaaS cloud environments, the users have the responsibility to set up the entire software stack starting from the OS to the compilers and required libraries before they can compile and execute their applications. Besides, most of the time the application source code requires several modifications to run in those environments. Cloud services, such as Microsoft Azure, present a distinctly different environment from that encountered by a user on a traditional HPC system. Thus migrating the complete application code becomes a challenging, expensive and time-consuming task as BRAMS is composed of more than 350,000 lines of code in Fortran.

Setting up, configuring and testing a base VM image for deployment represented another challenge for porting. The source code provided for HPC applications often lack good documentation, code comments and most of the time has been optimized for a very specific type of machine or architecture. We have to experiment with several versions of system libraries, compilers, frameworks and combinations of them to get BRAMS running correctly in a virtual machine. Recompiling and modifying BRAMS was not an easy task. Using the same environment (VMs) in which the application was going to be executed in the cloud, allowed us to generate the necessary tools to test the BRAMS execution and automate all the process. Given that one of the essential characteristics of the cloud is the virtualization, we initially used local virtual machines in this step, reducing costs. At the end of this process, each one of the VM instances must have a working BRAMS setup with all the necessary configuration to execute the forecasting jobs sent by the frontend. Configuration includes the access to the shared storage subsystem.

One of the challenges we found while migrating BRAMS to the cloud was the access to the legacy file-system BRAMS used. Storing such large amount of data in one VM would create a bottleneck for the application. On the other hand, copying the datasets on every machine would be expensive because of transfer time and replication.

At grid environments, it is common to use a dedicated network storage device with a distributed file system. Using on-premises computers, the network usage is limited only by the hardware available. In contrast, in the Cloud the network transfer is charged, because of this, the amount of bandwidth required by HPC applications constitutes another challenge to use the cloud as a viable alternative. Thus minimizing the costs to run the experiments and the network usage is a requirement in the cloud.

An additional challenge was setting up a reproducible process for compiling, installing all the software needed, setting up the cloud service and controlling the lifecycle of BRAMS. This process needs to be completely automated to allow a complete pay-as-you-go usage of BRAMS in the cloud. As indicated before, the Azure cluster was configured to replicate an HPC environment with a frontend node, which the users could use as a login node to launch BRAMS. We used the Microsoft

Azure cross platform command line tool (azure-CLI[1]) to control the workflow of the cloud service creation of BRAMS.

The process to setup the environment and creating the initial scripts to launch the VMs is complicated. However, once set up, adding more nodes and scaling the cluster size is straightforward. In a cloud service, the time and resources spent configuring, compiling the applications and getting the initial instance ready is charged. The process to experiment until the base image is ready has to be taken into account. To minimize unnecessary costs using the cloud environment, is important to automate most of the tasks, from the creation of an initial cloud instance to the deployment steps before begin processing.

In order to perform forecasts using BRAMS in the cloud each one of those issues, namely the storage, networking and deployment aspects should be addressed. The next subsections show how we solved these challenges and how these solutions modified the BRAMS workflow.

4.1.1 Storage

Our solution in order to port the storage aspect of the HPC application was testing two different mechanisms in the cloud. In the cloud, the storage access depends on the features available by the cloud provider. Our initial approach was to implement a distributed file system. We looked into the manageability of the file system, and possible scalability taking into account the way a numerical weather application behaves. In order to replicate the behavior of a file system with those characteristics, we selected two options based on the possibilities in the cloud provider.

The first one, Fraunhofer Parallel File System, FhGFS[2], was selected by looking into the options available for distributed storage in the cloud. We tried Ceph[3] file system and FhGFS but as we found some issues related to the configuration in the cloud environment, we decided to use only FhGFS. FhGFS is an scalable file system used in several HPC projects. After experimenting and running some benchmarks, we decided that FhGFS fitted better with the requirements of BRAMS. FhGFS provided high throughput and scalability and was suitable to the cloud infrastructure. The scalable metadata replication on FhGFS provided the best option based on the requirements of BRAMS. The architecture of the FhGFS file system on BRAMS is depicted in Figure 4.1.

As shown in Figure 4.1, a computing infrastructure of a weather research center was replicated inside an Azure datacenter. On this *virtual* research center, we have

[1]https://www.npmjs.com/package/azure-cli
[2]http://fhgfs.com
[3]http://ceph.com

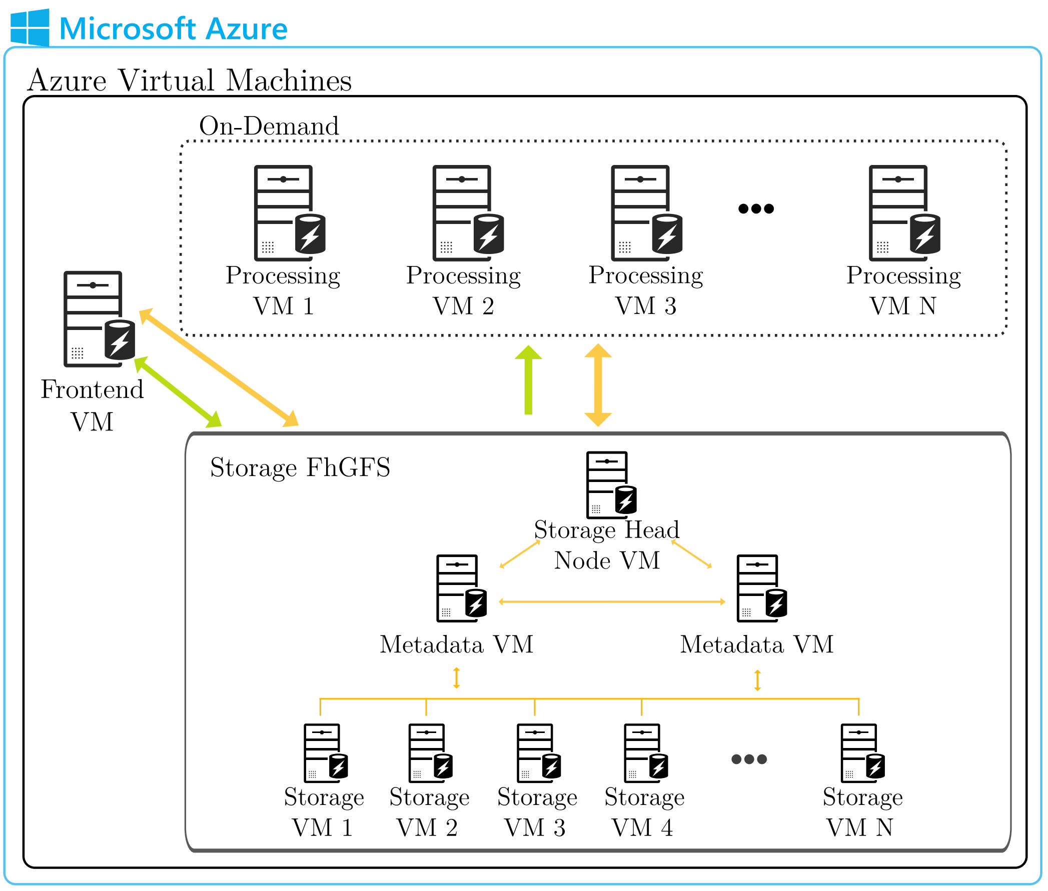


Figure 4.1 – Architecture of the implemented storage system using FhGFS. All the nodes are virtual machines. Each storage node with a capacity of 1TB.

the computing and storage. The storage was provided by 8 A3 instances each one with 1TB disk using the Fraunhofer Parallel File System. The computing nodes were used on demand. One of the main issues we found implementing the data storage architecture is that from a cost perspective, is expensive to have some VMs only for storage. Scaling would depend mostly on the paying capacity of the user and not on the amount of data stored. The metadata VM were very delicate parts of the architecture, and they are not as scalable as the storage VMs, managing those machines was a complex part in the implementation. Only preliminary experiments were performed using this storage system. The Azure subscription have a CPU-core quota limit, CPU cores used on storage VM also limited the number of VMs available to processing. Implementing other distributed filesystems, for instance, GlusterFS[4] or Lustre[5], would present the same issues, managing the storage infrastructure and wasting resources on idle VMs. Those VM were used during initial testing, but we found that a better solution have to be implemented to make an efficient usage of the cloud resources.

[4]http://gluster.org
[5]http://lustre.org

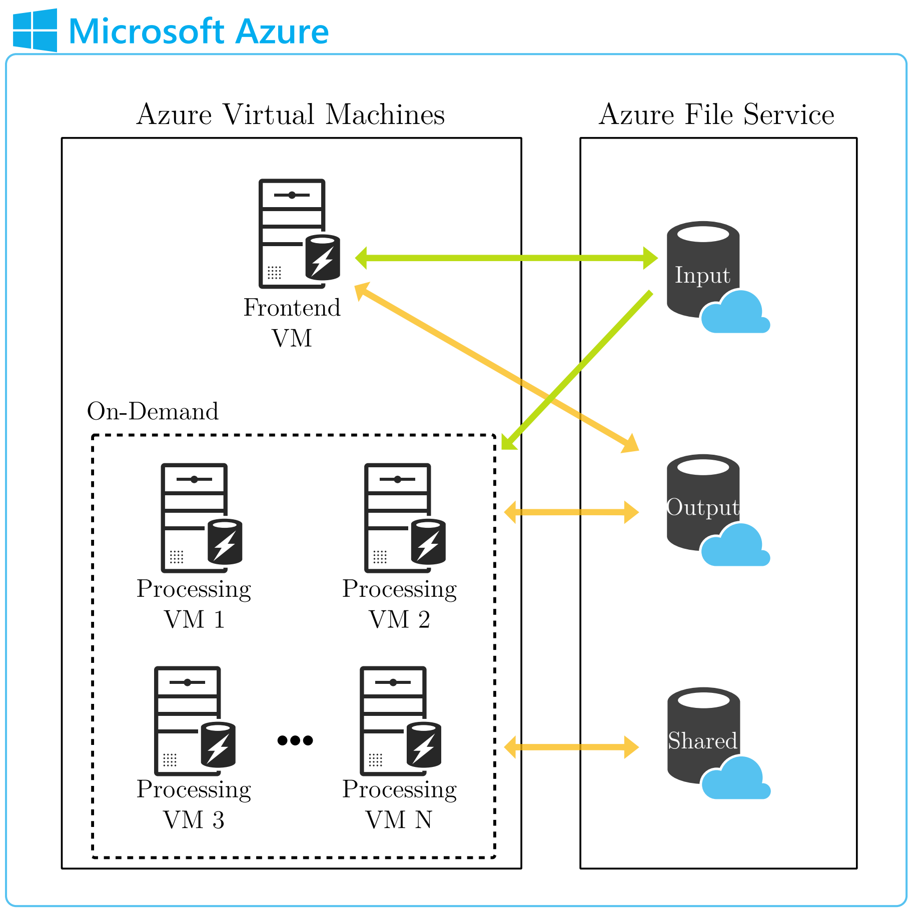


Figure 4.2 – BRAMS in the cloud using Azure File Service as storage. No more compute resources are spent on storage deployments.

The second option used the cloud storage services provided in Microsoft Azure. Azure allows storing files using two different mechanisms; one called blobs to store using a proprietary API and a second one called Azure File Service (AFS) that uses Server Message Block protocol(SMB).

We opted for using the AFS as the usage of blobs required an intrusive modification of optimized input/output mechanisms of BRAMS. As we mentioned before, we focused in porting the application performing the minimal amount of modifications to the source code. Because of the high volume of data required to perform NWP we store the input dataset for the simulations and all the output files also in AFS. Figure 4.2 depicts the changes related to the FhGFS implementation. We do not need to spend resources on VMs to store the data, this is managed directly by the service provider.

AFS allow sharing data between virtual machine (VMs) instances and between cloud services within the same Azure region. Sharing input and output datasets creates an interesting opportunity to integrate multiple cloud services of meteorological centers executing BRAMS within Azure to share their resulting datasets.

By using the AFS approach, we used SMB to access a shared directory with all the files that BRAMS requires to execute. Using AFS allowed us to use BRAMS with minimal modifications and focus on the rest of the issues. The main advantage is that AFS does not require using VMs to keep the data available, minimizing the costs to pay only for the size of the stored data.

4.1.2 Networking

The networking aspect of the cloud represented a challenge that is not presented in a grid or cluster infrastructure, paying by the amount of transferred data. Cloud service providers charges by the amount of bandwidth used. In the case of Microsoft Azure transferring data outside the datacenter in which the VM is located is charged, however, transfer in is free. Transferring outside the Azure infrastructure is charged, but also data ingress to their infrastructure is free.

Considering this scenario we could upload and store the complete input dataset provide by INPE to the AFS system without paying for the transferring. Communication between the computing nodes would be very expensive while processing data in the forecast execution stage. The solution to this issue is the usage of affinity groups and using the same datacenter for the location of storage account and the compute nodes. Azure does not charge anything for traffic kept inside the same datacenter. By taking advantage of this feature, we could reduce the cost of performing the weather forecast significantly. The only transfer data charged would be related to getting data out of the datacenter. Keeping data in the cloud service, is precisely one of the main purposes of performing processing in the cloud, for this reason moving large amounts of data out of the cloud service is not expected.

4.2 BRAMS Lifecycle in Azure

Addressing the deployment challenge, including the setup of the initial environment and launching the VMs, we automated all these steps to execute BRAMS in the cloud. We developed complete lifecycle fo BRAMS, from the creation of the cloud service, passing by the instantiation of VM to perform the forecast going until deleting everything that was used in Azure. These features are necessary also to reproduce the experiments and evaluate the impact of changing configuration parameters in Azure. Figure 4.3 shows the complete lifecycle of BRAMS service on the cloud. This section describes each one of the lifecycle stages and steps.

In order to set up, configure and manage the environment in Microsoft Azure using IaaS, we used the Azure command line interface tool provided by Microsoft. This Azure-CLI is the core of the developed lifecycle control. Azure offers several

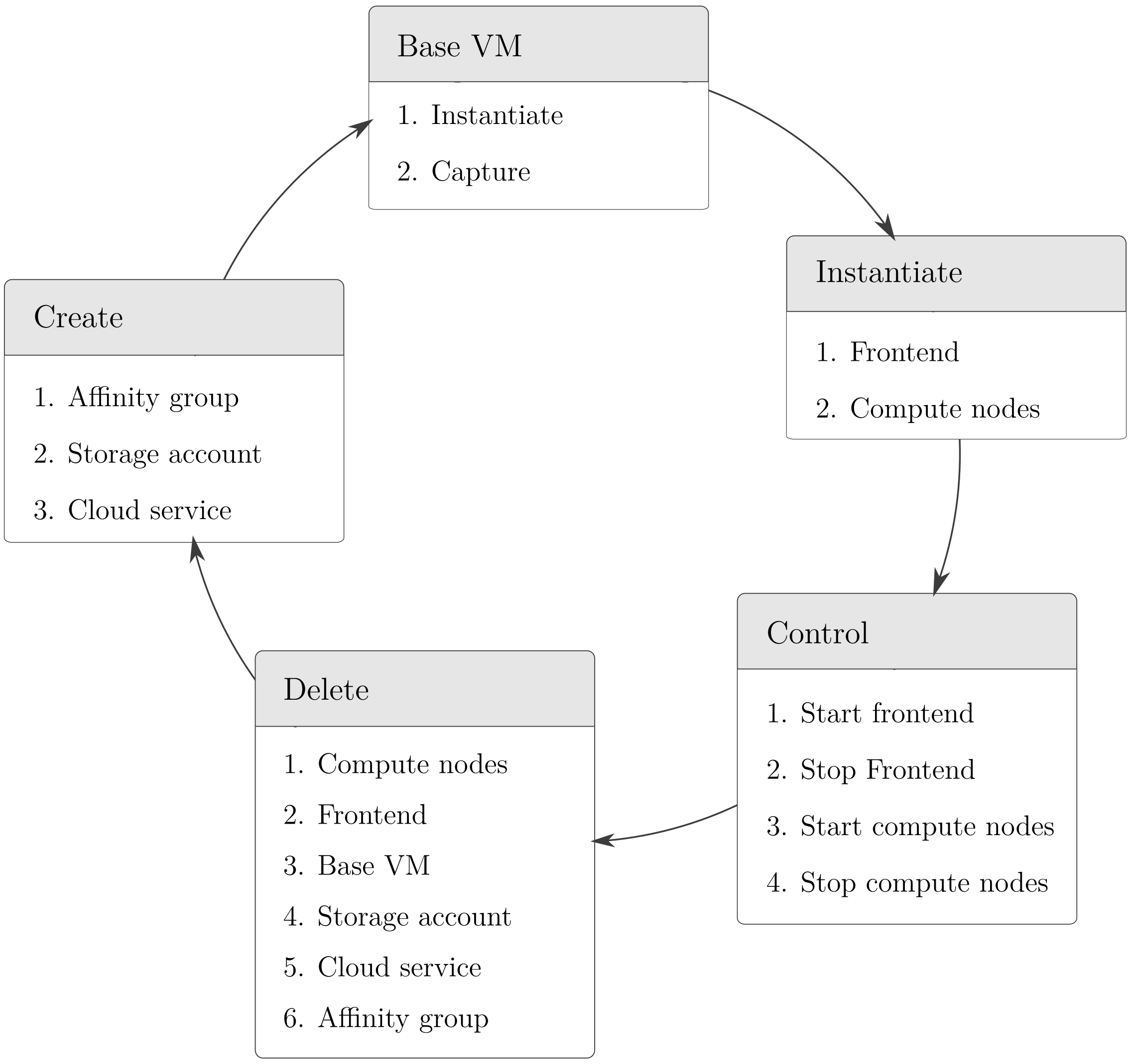


Figure 4.3 – Lifecycle of BRAMS service on Azure.

options regarding the OS based on GNU/Linux: Debian, Ubuntu, Suse, and CentOS. Each one of the distributions is also available in more than one release. For the operating system, we chose Ubuntu 14.04. Ubuntu is a popular distribution, because of that, the learning curve for users of the BRAMS service would be easier. We selected the 14.04 release, the more recent LTS version.

4.2.1 Create

Using the available options of the Azure CLI we can start the lifecycle of BRAMS in the cloud. For the create stage of the cloud service, the user needs to follow three initial steps.

- Create an Affinity Group: This is a crucial step for the service. If the affinity group is defined, all the VM instances are going to be placed as close as physically possible by the resource scheduler of Azure. The parameters configurable for this step are the affinity group name and the datacenter name, basically its location. This setting impacts directly on the networking aspect of the deployment. If it is not configured, the machines could be allocated anywhere on the datacenter, affecting latency between VMs.
- Create storage account: The storage account would act as a container for the OS virtual hard drives (VHD) of the VMs, the attached VHDs for the storage model and to save the files using AFS. We could modify which kind of replication we want to use, base on the characteristics of the data to be stored.
- Create cloud service: The cloud service is the access point to the service, is a web address that allows resolving the public IP address of our service. The cloud service could be seen as a container for the VMs of the deployment. Using this address and endpoints we could access any of the machines inside the deployment.

Those initial steps allow us to create a container for our virtual machines and a way to locate the cloud service on the internet. After this *create* stage, we go to the deployment of the Base VM.

4.2.2 Base VM

The Base VM is an instantiable object of our deployment, it could be replicated as much as is needed to. In this stage, this VM is created. This VM must contain all the software, tuning and configuration that allow the machine to act as a part of a cluster without further modifications. This Base VM is the stepping stone of the usage of the cloud to replicate experiments and to use the same environment.

- Instantiate: In this step we instantiate a VM specifying the instance size we require to install and configure the application we are going to use. This VM size does not need to be the exactly the same used when running the application, it can be changed on posterior deployments. We must include in the instantiation parameters the specific OS that we are going to use. Also, we use authentication using certificates, by using this method we can allow passwordless access to the virtual machines, which is an important setting to use MPI. We installed and configured the BRAMS version modified to be used on this VM environment. This process is fully automated to allow creating a new version of BRAMS with only changes in the install scripts. After the machine is completely configured, we could go to the next step.

- Capture VM: The process to capture a VM allow us to save the machine in an state called *deprovisioned*. This state basically removes the parameters that make the VM unique and allow further utilization of the same image. The only parameter required for this step is the name that the Base VM will have. After the machine is captured, it is stored as an operating system image and could be instantiated by passing the image name as a parameter in the place of the OS.

4.2.3 Instantiate

After instantiate, the cloud service of the application is usable. The frontend of the cloud service should be able to access all the compute nodes that the cloud deployment will have. We notice that to make the environment homogeneous, all the virtual instances, including the frontend should be instantiated using the Base VM captured. In this way, we need to add the software needed by the frontend and configure some parts of it. The nodes need minimal configuration to work as a part of the cloud cluster.

- Frontend: Frontend instantiation requires some individual settingsThe parameters to instantiate includes name of the frontend, the name of the cloud service, the base VM image name and the service endpoints. The VM is configured to allow monitoring the compute nodes using Ganglia[6], an open source monitoring utility. Also, the configuration allows post processing of BRAMS data and implementing a notification system to the end user. The frontend can passwordless access all the compute nodes in order to execute BRAMS forecasts.
- Compute nodes: The compute nodes are almost entirely configured at instantiation time. The only parameters different from the frontend instantiation are the name of each node. Compute nodes do not require endpoints as they are used on the internal Azure network. They need information about the frontend in order to send monitoring information. The customization of each VM includes setting the variables or information for the forecasting process.

4.2.4 Control

To take advantage of the on-demand characteristics of the cloud, we must control when the VMs are running and when they need to stop to reduce the billing by the service provider. The control stage of the lifecycle was developed to manage those

[6]http://ganglia.sourceforge.net

steps based on what is required at the time by BRAMS.

- Start Frontend: In case the frontend is stopped, like after using the stop frontend step, we need to start the frontend again. Stopping the frontend depends on how the service is going to be used. If using the service in full on-demand mode, the frontend is stopped and only started again using this step when there is data to process.
- Stop frontend: As mentioned in the previous step when using entirely on-demand these steps are used. Both control steps are used from a computer outside Azure infrastructure.
- Start compute nodes: When a new processing task is ready to start as depicted in Figure 4.4 this step is executed. As the VMs are already configured this step only require the names of the VM instances to start again.
- Stop compute nodes: After the INITIAL stage of BRAMS the compute VM instances are not needed anymore, this step takes care of releasing those resources to stop being charged for them. When a node is stopped, we are charged only for the VHD size of each node stopped.

4.2.5 Delete

The final step in the lifecycle is the elimination of resources used on the cloud architecture for the application. Most of these steps are performed as final procedures before finishing the experiments. The step *delete compute nodes* was used when we performed initial experiments deleting the VMs and re-instantiating them to measure the time spent on deployment operations.

- Compute nodes: Deletes the stopped computed nodes and each one of their VHDs.
- Frontend: Deletes the frontend and its associated VHD.
- Base VM: Deletes the Base VM image and removes the entry from the available images list.
- Storage account: By removing the storage account, ALL the data stored in the cloud service is deleted. It is not possible to recover the data after this step is performed.
- Cloud service: Deletes the cloud service, releasing the DNS name in Azure.
- Affinity group: Deletes the affinity group.

The last two steps delete parts of the service that are not charged by Azure. The affinity group and the cloud service could be left undeleted to save some steps in the future if we need to re-create the service.

4.3 Adapted workflow in the cloud

Executing BRAMS in the cloud required modifying the workflow presented in Section 3.1. The modified workflow is described in this section, Figure 4.4 shows the changes introduced. In this adapted workflow is assumed that the frontend instance is left deployed. The frontend VM is in charge of executing the tasks related to controlling the workflow execution.

The workflow starts as soon as a new processing task is received by the frontend. The processing task could start with the availability of new data to process or with a forecast experiment.

The frontend checks the availability of the distributed storage because all the data in the following steps are stored in it. Contrary to a physical grid or cluster in which all the machines are powered even if they are not processing in the pay-as-you-go model the virtual machines are not available until they are provisioned. It would be a waste of resources to create instances if it is not possible to save anything. If everything is in order, the allocation of the processing nodes starts using the lifecycle stage *control*, and the *start* step. At the same time the download of the new data, if required begins. The frontend checks when a processing VM is ready to start processing to start running the next stages.

The MAKESFC and MAKEVFILE stages could start as soon as the first instance is available. Because these stages only use one CPU core they could start the processing with only one VM. This process could also be performed by the frontend, but as we used a frontend VM with lower specifications than the processing nodes, we decided not doing it.

After those initial stages of the workflow, the frontend needs to wait until all the compute instances are deployed and ready. The barrier stage in Figure 4.4 represents that.

When all the instances are deployed and ready to compute the INITIAL stage starts. The frontend waits until the end of the forecast execution, as soon as

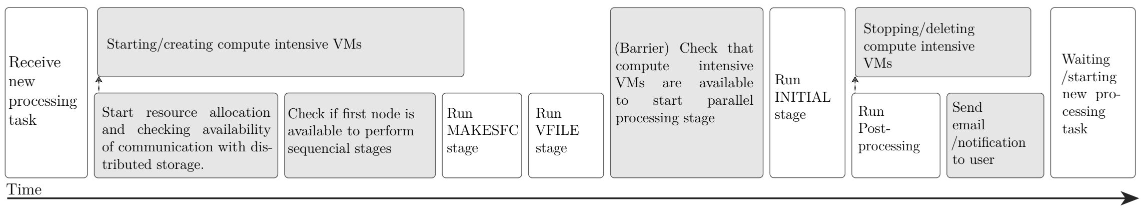

Receive new processing task
Starting/creating compute intensive VMs
Start resource allocation and checking availability of communication with distributed storage.
Check if first node is available to perform sequencial stages
Run MAKESFC stage
Run VFILE stage
(Barrier) Check that compute intensive VMs are available to start parallel processing stage
Run INITIAL stage
Stopping/deleting compute intensive VMs
Run Post-processing
Send email /notification to user
Waiting /starting new processing task
Time

Figure 4.4 – BRAMS execution workflow, the white boxes are the steps to perform a simulation in an environment with physical machines already provisioned like a grid or cluster, gray boxes indicates the steps introduced to run BRAMS on-demand in the cloud.

the forecast finishes the post-processing stage is started. The post-processing was performed by the frontend as it is not a compute intensive operation. Simultaneously to the post-processing the compute VMs are *stopped* or *deleted* depending on the selected configuration in the lifecycle stage. An analysis of the differences between stopping or deleting the compute VM is performed on Chapter 6.

At the end of the post-processing, the user is notified about the status of the forecast and the frontend returns to its awaiting state.

4.4 Concluding Remarks

In this Chapter, we showed how BRAMS was migrated from a traditional HPC environment to a cloud infrastructure. As described along this section identifying the significant aspects of running HPC in the cloud was a challenging task. To evaluate the migration impact, storage, networking and time spent on computing and non-computing operations should be analyzed.

The lifecycle technique proposed in this chapter could be applied to others applications in order to make the managing process easier while reducing cost when moving to the cloud. In the next Chapter, the methodology to perform the evaluation of aspects is discussed. Then, in Chapter 6 the experimental results of the evaluation are presented.

5 EVALUATION METHODOLOGY

In this chapter, we present the methodology used to conduct the experiments. Two different scenarios are proposed to evaluate the main aspects related to migrating BRAMS to the cloud. The first one is using a single deployment of BRAMS in the cloud, performing forecast simulations.

The second scenario is composed of the creation of multiple deployments of BRAMS in the cloud, each one acting as an independent service. We refer to this scenario as *MBRAMS*, meaning multiple BRAMS. With MBRAMS, we intended to simulate the deployment of BRAMS environment for multiple weather research centers. Those scenarios and the aspects analyzed on each one will be discussed in this chapter.

5.1 Evaluation of aspects: *BRAMS*

The methodology to evaluate the aspects and its purpose is defined in each section. Most aspects could be evaluated from an economic point of view. Those aspects are based on the analysis performed on Chapter 4. The architecture of BRAMS ported and deployed to the cloud is depicted in the Figure 5.1. In the figure the frontend is the only instance with external communication, all other instances are confined to Azure infrastructure.

5.1.1 Performance and Scalability

In order to analyze and compare the performance of BRAMS in a cloud environment, we ran a series of experiments in a local cluster and compared it with the version running on the Azure platform. The tests were performed by scaling the number of nodes available. The experiment consisted of performing a simulation five times using computing nodes with 8 CPU cores. At every iteration, we added a new node up to 8 nodes (64 CPU cores). With this experiment, we intended to compare the performance of different Azure instances sizes against the real hardware in a local cluster. This experiment also allowed us to check the scalability of BRAMS in those

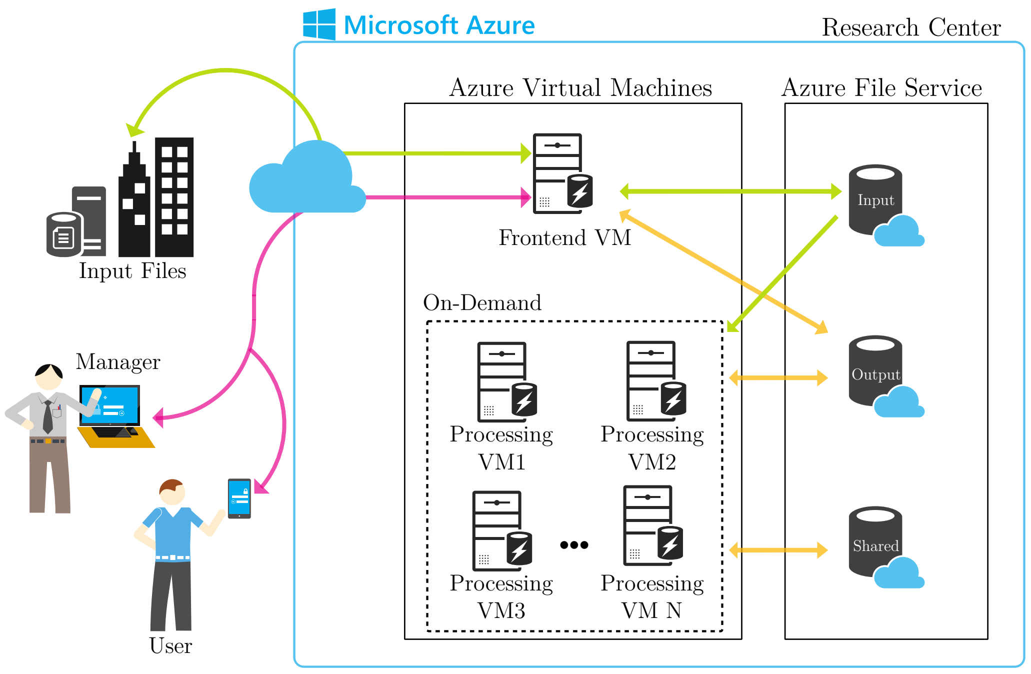


Figure 5.1 – Single deployment of BRAMS in the Cloud. The Figure illustrates the deployed architecture.

two environments and help us to identify possible issues related to the migration to the cloud.

Scalability in the Cloud is used to establish the possibility of an application to use more resources in case it need to use them. Scalability could be seen on two different axes, horizontal or vertical scaling. Because we are comparing a cloud version of an application used in **on-premises** computers we need to clarify what scalability means for us in both environments.

In the cloud, horizontal scaling is referred to the possibility of adding more virtual machines to the workflow. These new machines could reach a limit in which adding more machines only causes growing the prices of the experiment but does not add more performance benefits. We tested the limits on the horizontal scalability of the application in order to find that limit.

Vertical scaling in the cloud is used commonly to upgrade the characteristics of the instance running a service. Usually, this process occurs on-demand and over a running instance. Is not realistic to run an application that balances resources usage and uses homogeneous characteristics between the machines while upgrades the instance characteristics in the middle of an experiment. Because of this, our vertical scaling experiments were based on using cloud instances with different sizes, but maintaining the same NWP configuration for all the machines during each experiment. Using this approach we could measure in which point was not possible to get more performance gains using BRAMS in the cloud.

5.1.2 Storage

The methodology was to measure the amount of data required by the modified workflow and analyze the different options regarding replication in Azure. Because AFS was in preview stage, performance evaluation was not a priority for our deployments. The pricing scheme and how to optimize the usage of the storage system was the focus of these experiments. To evaluate two options we analyze the stability and the pricing characteristics of each one.

5.1.3 Networking

The networking aspect is one of the most important factors in the cloud. As the entire concept of the cloud is based on the availability of remote resources, the network is expected to impact strongly on the results of the experiments. In order to approximate the network hops between virtual machines, we performed a series of experiments using Nmap[1]. Using this tool we could calculate the number of hops, and determine if using some characteristics of the cloud service allow us to optimize the networking performance of the application in the cloud. In the local environment, we also measured this metric.

Another aspect measured was the latency between the virtual machines and the frontend. HPC applications require low network latency values. We also used Nmap to retrieve the values. This metric in some cases influence more the performance than the bandwidth. We obtained this metric by calculating the latency between the virtual machines and the computing nodes. The large amount of communication between the processing instances and also to the frontend instance during the INITIAL step of the workflow could be impacted by the latency.

Considering that cloud network infrastructure varies on each VM allocation, we design an experiment to collect networking data using eight computing nodes.We obtained their latencies and the number of hops between these VMs.The experiment consisted of 2 scenarios with two types of network analysis. The first one started by creating 8 Azure instances of BRAMS and analyzed the hops between each node. After obtaining data, the VMs were deleted and created again. The same process was repeated five times from each one of the 8 VMs. In the second scenario, the VMs were not deleted but stopped. The Azure VM managing mechanism could reallocate in different physical machines the instances that are stopped when started again. The second one obtained the hops from the frontend to the eight instances. We wanted to analyze the latencies in those cases too.

Network stability was measured by taking into account failures during the experi-

[1]http://nmap.org

ments and connection issues.

5.1.4 Provisioning

In a final experiment, we analyzed the time spent in the provisioning and controlling operations in the cloud service, namely start, stop, delete and create instances. The time spent in this operations affect the costs of running a cluster service on demand.We measured the time spent when machines are deallocated or deleted at the end of each experiment, and when started or reallocated at the beginning of the next one. We notice that those control operations in Azure blocked the subscription from performing another task. The time it takes to perform these operations affect the expected run time of the experiments as we perceived that those operations are performed as a sequential procedure. In the cloud, we are charged by the number of virtual machines allocated even if they are not performing useful computing we wanted to know how much time was spent on each one of those operations.

5.1.5 Pricing

Cloud computing is at its core a commercial model, monetization of services is the motivation of its growth and the optimal usage of the resources provided by this paradigm allow us to evaluate how to use them.

We calculate the prices of each experiment taking into account the variables more important to execute a scientific application as BRAMS in a cloud environment. The experiments evaluate the price on several datacenters of the service provider and calculate the approximate price of running the workflow of BRAMS during realistic periods of time. These results were compared afterward with the price of buying on-premises server to provide the same service, including the associated cost that are eliminated by cloud as power bill, refrigeration, placing, security aspects, etc.

Table 5.1 – Virtual machine instance sizes selected for the experiments. Prices in US dollars. The Azure datacenter used was Europe North.

	Standard tier		Compute + SSD		Network	
Instance Size	A4	A7	D13	D14	A8	A9
CPU cores	8	8	8	16	8	16
Memory [GB]	14	56	56	112	56	112
Disk Size [GB]	605	605	400	800	382	382
Price/hour	0.720	1.200	1.318	2.372	2.450	4.900

5.1.6 Environments

The environments used in our experiments were a set of cloud clusters and a local cluster. The characteristics of each environment are described in this section. A summary of the hardware details of the machines used and characteristics is depicted in Table 5.2.

Each local machine consists of two Intel Xeon E5310 Processors, each one with four cores running at 1.60GHz and 8 GB of memory, Interconnected by a Gigabit switch.

For the local cluster machines, we used eight server machines of the same model with the same characteristics. The storage was provided by four nodes with the same characteristics but not involved in processing.

In Microsoft Azure using IaaS, we avoided using the smaller Linux compute instances available for our subscription to minimize the communication overhead of the HPC application. We use the compute instances that were closer to the characteristics of the local machines. As can be seen in Table 5.2 the D13, A4, A7 and A8 instance types include 8 CPU cores. Only D14 and A9 include 16 CPU cores. Those instances with 16 CPU cores were used with eight cores during some of the scaling tests, to obtain comparable figures. The cloud service provider defined the name of the instances types.

Each one of those virtual machines is advertised in categories for specific scenarios like fast networking, performance computing and intensive I/O usage. We use two machines of each category to compare them. The bandwidth available for the network interconnection in each instance is different and is a hard limit imposed by the service provider. For the storage, we used SMB shares in AFS to save the input and output datasets of the experiments.

Table 5.2 – Specifications of the local machines and the virtual instances used for performance experiments. Networking bandwidth limits on the VMs are imposed by the service provider.

Characteristic	Local Cluster	Cloud Instances					
		A4	A7	D13	D14	A8	A9
Processor Model	Xeon E5310	Xeon E5-2660				Xeon E5-2670	
Processor Speed (GHz)	1.6	2.2				2.6	
Number of CPU Cores	8	8	8	8	16	8	16
Memory (GB)	8	14	56	56	112	56	112
Networking (Mbps)	1000	800	2000	1000	2000	5000	10000

The experiments on this scenario consisted of executing BRAMS to perform a forecast of 72 hours with an spatial resolution of 50 km covering a square perimeter

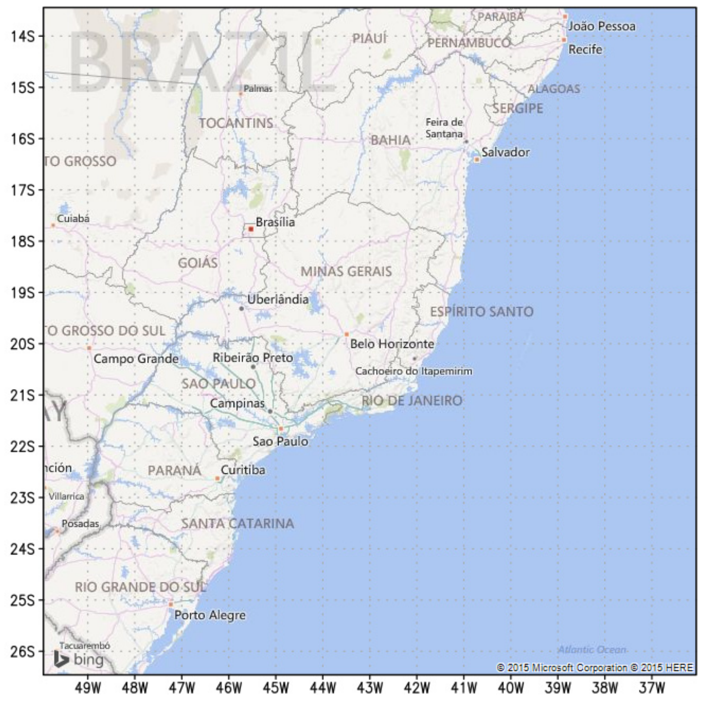


Figure 5.2 – Geographic area used for forecasting during the initial experiments using the local cluster and the cloud instances. BRAMS with a model resolution of 50KM

of 6000 km, an area of 1500x1500 km. The forecasted geographic area was always the same for all of the experiments in this scenario. Figure 5.2 shows the area. All of them were executed in the same Azure datacenter location using affinity groups in order to achieve a better performance as the VM would be as physically close to each other as possible. The datacenter we selected was Europe North. This datacenter provided the lowest cost and included access to the A8 and A9 instances, such instances sizes were not available on every datacenter when the experiments were performed.

5.2 Evaluation of aspects: *MBRAMS*

As a final evaluation of the viability of cloud for HPC, we run multiple BRAMS deployments in parallel sharing the input datasets in order to simulate an integration of multiple users.

The multiple deployment characteristic of *MBRAMS* is depicted in Figure 5.3. Each one of the BRAMS deployments has access to a shared AFS storage account in charge of obtaining the input datasets from the sources. The main difference with the single deployment is that a shared storage account is used between them. In one scenario data produced by each deployment will be saved in its own AFS storage account, only input data will be shared. In a second scenario, the shared account will be used to store input and output data.

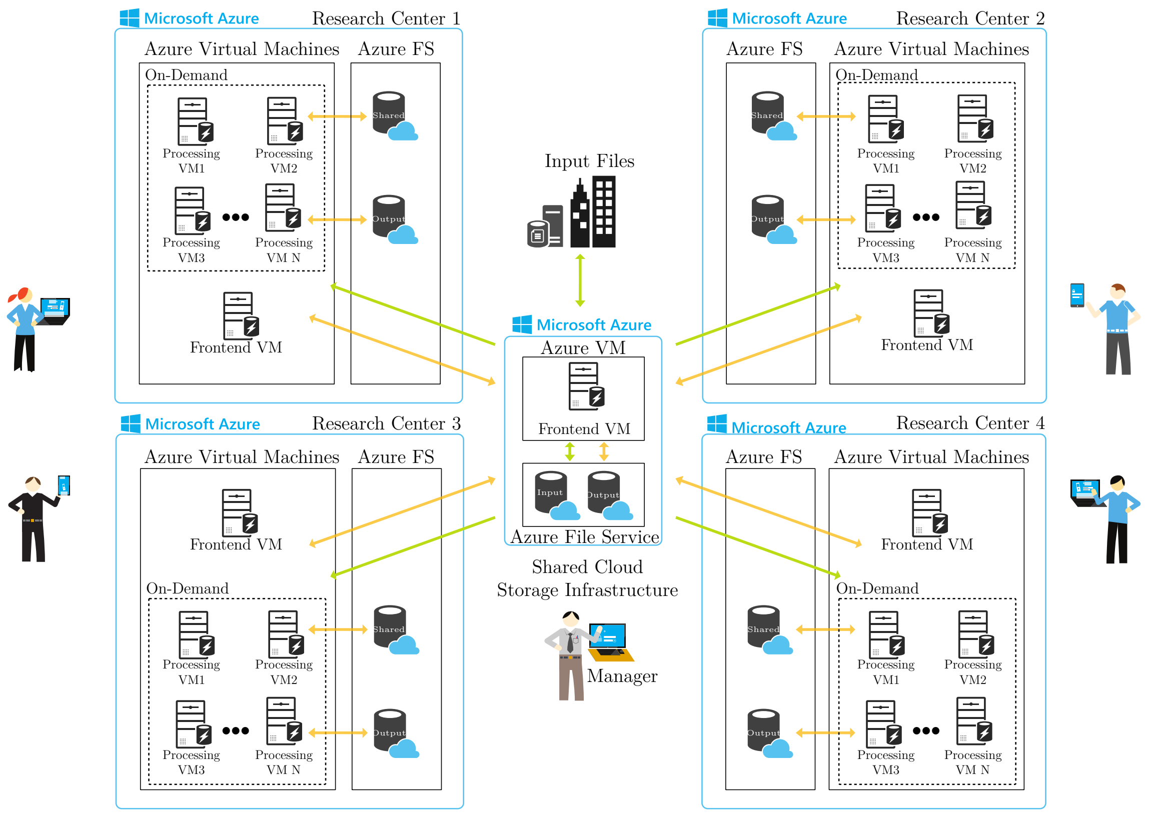


Figure 5.3 – Multiple deployments of BRAMS in the Cloud. The deployments share input files, and they could share output files if they need to, by using the storage system.

To achieve this, we run forecast with several resolutions over different areas of South America. Figure 5.4 shows the areas used for forecasting using a 50 km resolution. Most of the aspects are evaluated in the same way as with BRAMS, the differences in the methodology are discussed next.

5.2.1 Aspects

We did not apply the methodology to evaluate pricing, networking and provisioning in this scenario. The storage evaluation differs because we wanted to measure how much was the performance impacted mostly when the input datasets were shared between the multiple deployments.

For this aspect, we wanted to analyze the impact on the execution time, based on shared storage between multiple instances of BRAMS running in parallel. We evaluated whether using the same storage account to save the input and output datasets created any difference.

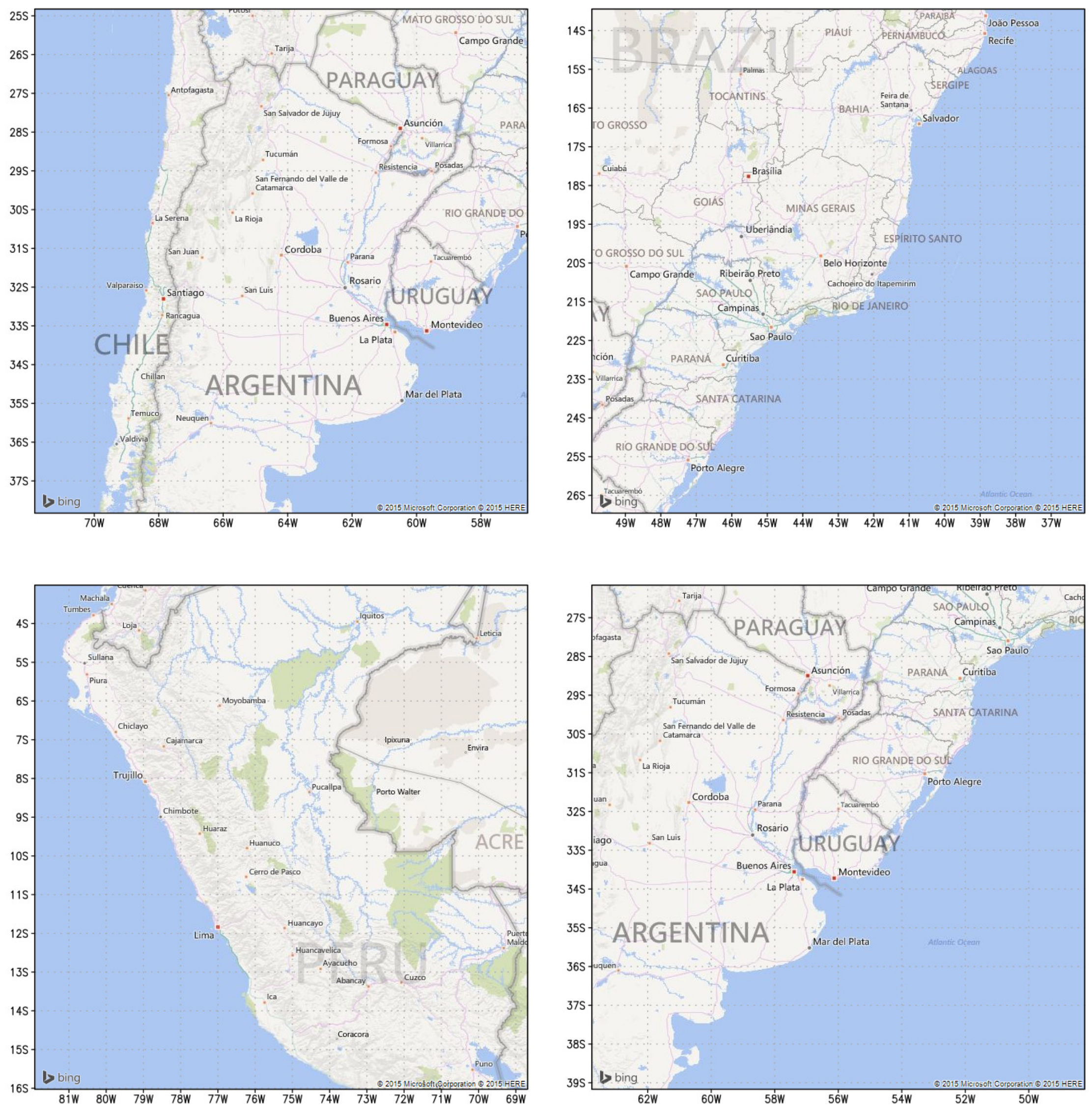


Figure 5.4 – Regions over South America used for the simultaneous 50KM resolution forecasting.

5.2.2 Environments

For these experiments, we used the same VM instance size for all of the deployments. We selected the A8 size because it have the fastest network interface (10Gb). The forecasted geographic area changed in all the deployments for the experiments in this scenario. As we wanted to simulate real weather research centers, the areas were established on locations of interest for the GPPD. The GPPD is part of some research initiatives, as the Latin American Grid for Climate, LAGClima. The Areas were centered spatially over Argentina (Córdoba), Brazil (São Paulo), Peru and Uruguay.

The first environment includes running BRAMS on each deployment over a

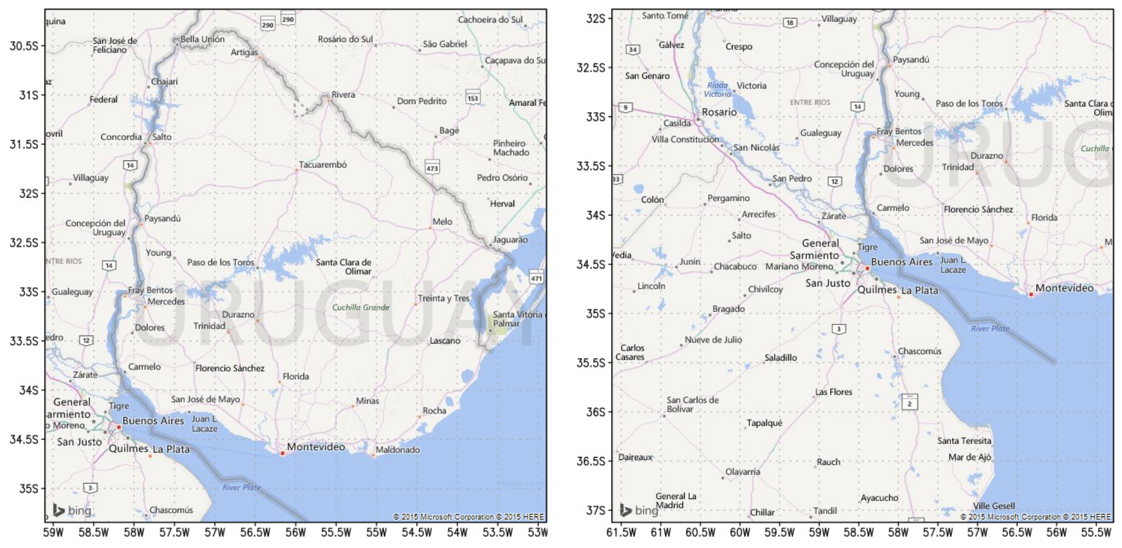


Figure 5.5 – Regions over South America used for the simultaneous 20KM resolution forecasting experiments.

different geographic area. Figure 5.4 depicts the four regions over which the model was run. In the Figure 5.4 is shown that the model resolution is more than enough to cover the area of a small-sized country, as Uruguay. In the other cases, some parts of the country are not included in the forecast. We used 50 km as an initial resolution because on the single deployment of BRAMS we used the same model resolution.

Each one of these BRAMS deployments was created using the same lifecycle procedures. The only difference was the customization of each instance. This process

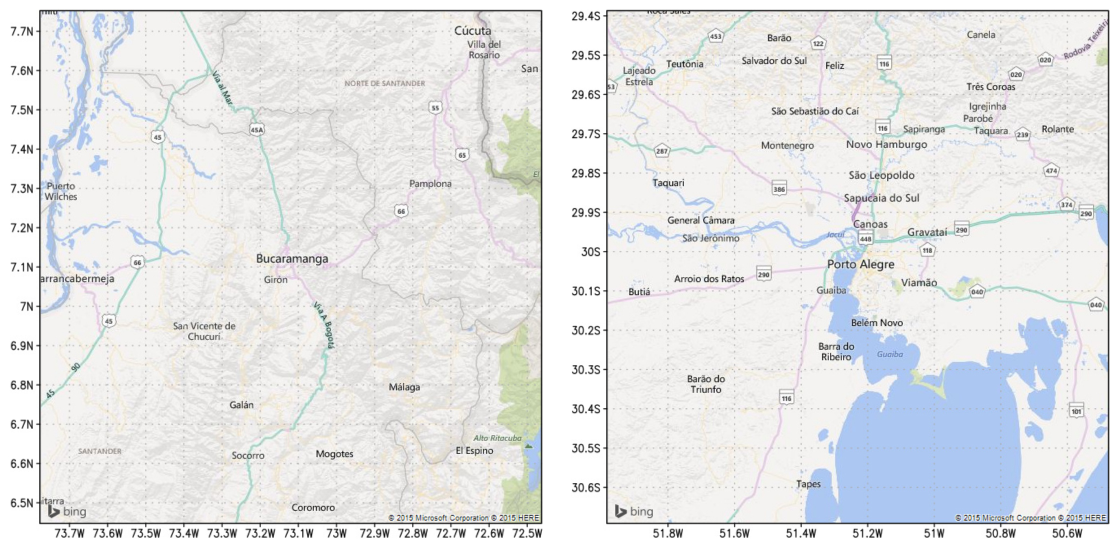


Figure 5.6 – Two city areas in South America used for the simultaneous 5KM resolution forecasting.

included the configuration of the shared storage account to be used by all of the deployments.

Because it would be expected to run the BRAMS model with a higher resolution over some regions we performed experiments using 20 km as a model resolution. In this case, we evaluated two parallel deployments of BRAMS. Figure 5.5 depicts the two regions in which the model was run. Those two areas are centered spatially over Uruguay (center) and Argentina (the Rio de la Plata Basin). This resolution was enough to cover Uruguay completely, but only a region of Argentina. This environment was used to evaluate a more detailed BRAMS forecast.

Finally, we performed experiments using 5 km model resolutions over two cities in South America. The forecasted areas are shown in Figure 5.6 We wanted insight on the performance of this high-resolution forecasting as is the most resource demanding resolution for the BRAMS model. This resolution cover the geographic area of a city. Those two regions are centered spatially over Colombia (Bucaramanga) and Brasil (Porto Alegre).

5.3 Final Remarks

Defining the experiments to evaluate a migration represented another challenge because there is not a clear procedure to follow. The evaluation must include the relevant aspects of the application in the cloud but also must allow getting an insight into possible issues generated by the migration.

The next section presents the experimental results of the numerical weather application migration to the cloud.

6 EXPERIMENTAL RESULTS

In the first part of this chapter, we show and discuss the results of BRAMS in the cloud experiments and their significance. These results include our findings regarding the network latencies and how they could impact the execution of NWP, and also the overhead created by the provisioning operations.Then we analyze the scaling of BRAMS in the cloud and the pricing and cost efficiency. Finally, we show how BRAMS execution in the cloud using multiple parallel running of the model using several cloud services.

6.1 Processing

The experiment consisted of running a complete weather forecast scaling horizontally from one node up to 8 nodes both locally and in Azure. We performed five runs of the INITIAL stage in the forecast simulation for each number of nodes and instances and use the average. BRAMS stages MAKESFC and MAKEVFILE were not taken into account because they were developed to use only one CPU core, that means they receive no performance gains by changing the number of nodes.

For some of the executions, the local machine was faster than some of the cloud instances as depicted in Table 6.1. However, the results differ depending notably between instance sizes.

Table 6.1 – Performance of BRAMS in the cloud using AFS. For each instance we have the execution time and the standard deviation at its right, all values in seconds. Experiments shown relatively low variability as none of the standard deviation values is higher than 0.04% of the execution time.

Cores	Cluster		Cloud Instances											
	Local		A4		A7		D13		D14		A8		A9	
8	6828.54	6.59	7046.46	23.89	7042.57	22.17	4362.22	32.90	2230.84	55.30	3305.54	14.59	3301.70	22.97
16	3148.36	0.95	3458.10	3.08	3505.88	23.31	2260.46	19.58	1797.90	36.85	1730.78	27.01	1733.76	30.25
24	2347.30	0.68	2671.32	11.66	2675.86	32.07	1804.64	29.60	1797.90	36.85	1387.94	22.13	1376.64	7.32
32	1976.72	0.70	2313.34	10.33	2446.62	16.80	1649.56	73.83	1602.82	22.38	1288.04	25.85	1260.60	30.60
40	1371.04	0.79	1756.42	8.82	1865.88	23.44	1299.42	28.92	1268.52	35.55	974.08	6.95	997.86	11.81
48	1363.54	0.31	1683.10	5.88	1840.02	41.45	1267.86	17.95	1212.74	13.68	940.32	8.70	989.56	6.73
56	1125.08	0.34	1448.72	3.62	1593.16	18.17	1138.48	8.90	1095.18	18.47	861.08	16.38	931.20	15.80
64	1126.72	1.06	1438.96	7.41	1606.02	12.64	1127.38	19.87	1128.96	30.35	860.58	13.74	958.98	24.59

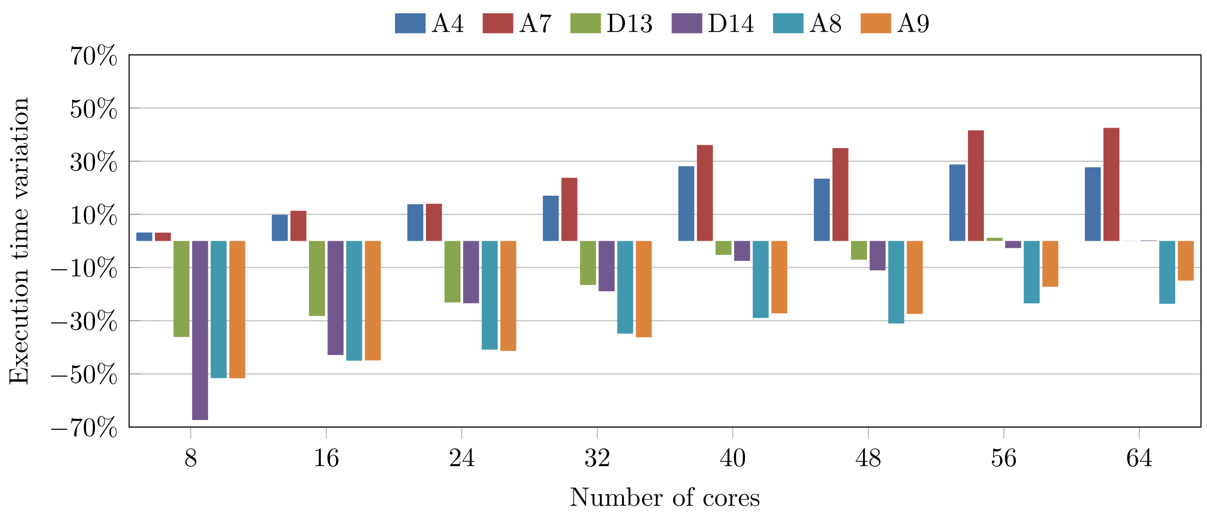


Figure 6.1 – Variation in execution time of different cloud instance sizes for a 72 hours forecast. Horizontal axis shows the number of cores. Values were normalized to the execution time of the local cluster. Lower values mean faster execution times in the instances compared with the local cluster.

As shown in Table 6.1, A8 and A9 instances got faster execution times for most of the experiments, except in the case of 8 cores in which D14 was faster. D14 finished in 33 percent of the time compared to local, A8 and A9 finished in 48 percent less time. D13 and D14 started with good performance, but the performance got worse as more nodes were added to the experiments, with 64 cores the execution time was almost equal to local. The standard size A4 and A7 instances perform increasingly worse in all the cases. A7 spent 43 percent more time than the local machines with 64 cores. Interestingly the A7 has better specifications than A4, nevertheless the performance of A7 was worse than A4 in all the experiments but the first.

The CPU frequency difference between the local machines with 1.60 GHz against the 2.20 GHz of the A4 and A7 instances is 37.5 percent. As shown with the results this fact did not reflect an equivalent advantage for those Azure instances. We found that the amount of performance degradation due to the virtualization was higher than the expected in the A4 and A7 instances.

Generally speaking the performance is worse as more nodes were added to the experiment for all the instance sizes. This issue can be attributed to the fact that BRAMS is a CPU bound application with a high amount of communication between cores and nodes. For this reason, adding more nodes impacts network performance, preventing a better scaling of the cloud cluster.

The best results overall were obtained by the A8 instance, reaching 76 percent with 64 cores. The A8 instance size is not the best instance according its specifications but performed a little better than A9, a similar instance size with twice the number of cores and price than A8. In some cases, the A8 instance was 5 percent faster than

A9. This behavior did not appear between D13 and D14 sizes, two similar instances also. From 32 to 64 cores, the difference was kept under 5% with faster results for D14. With eight cores, the difference against the local cluster was 31 percent. With 16 was 25 percent, on both cases D14 got better results. It is interesting that instance sizes with better specifications in some of the categories did not represent a remarkable benefit in performance.

In Figure 6.1 are depicted normalized results of the BRAMS execution experiments. Execution times of A4 and A7 in all cases were worse than local. The performance behavior of all instances is similar, suggesting scaling issues.

The variability of these experiments was low, around 2.92 seconds (0.16%) on average on Azure and 1.43 seconds (0.05%) on the local cluster. Results obtained in this section, and their consistency shows that BRAMS behavior was uniform between experiments. Also, that is possible to scale BRAMS horizontally using the Azure infrastructure with an acceptable performance even with the penalty imposed by the virtualization overhead.

6.2 Networking

For HPC applications like NWP, the network latency impacts the scalability and performance. As defined in Section 5.1.3 we measure the network latency, and the possible impact on it caused by reallocation or redeployment. The frontend was instantiated only once; the nodes were the ones redeployed or reallocated.

The first part of the results of this experiment is shown in the Figure 6.2. By using the cloud deployment of A4 instances we obtain an average of 70 µs with a 24 percent standard deviation. The high variability suggests high networking traffic in the datacenter. However, the latency value did not change a lot in the experiments, on average latency was 70.25 µs, and the standard deviation was less than 5 percent.

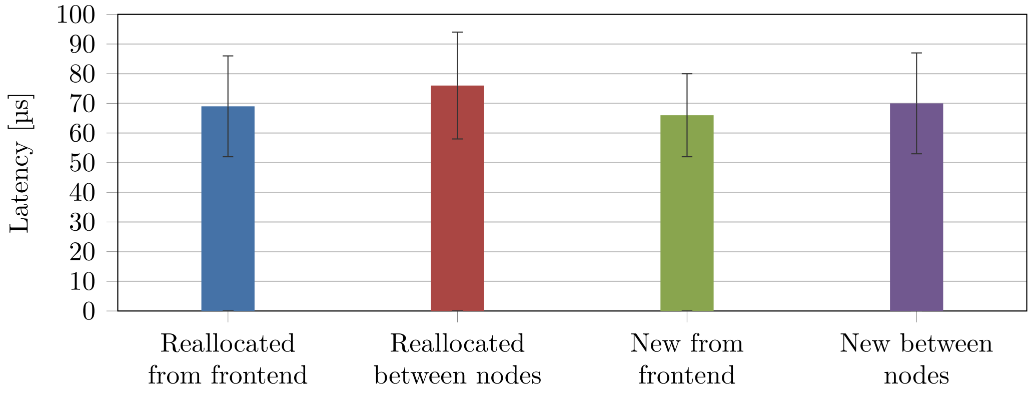


Figure 6.2 – Shows network latency between the nodes and frontend.

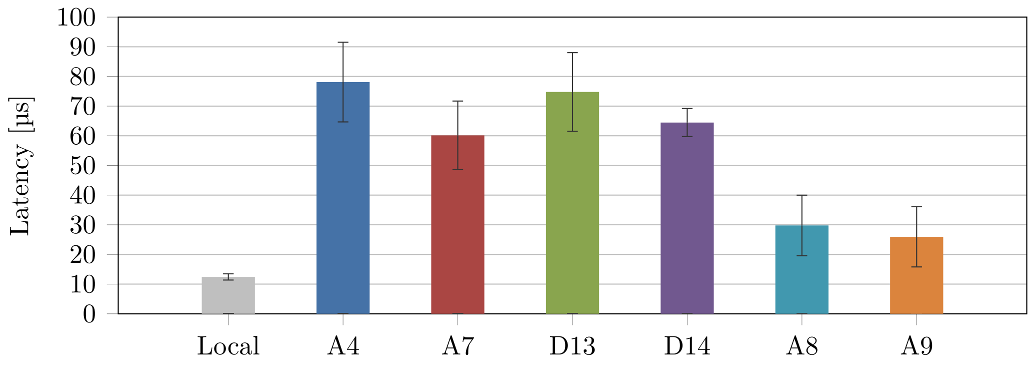


Figure 6.3 – Network latency between the nodes on varying instance sizes.

We could conclude that reallocating or creating nodes do not impact on latency.

The number of hops between nodes was consistent, four hops in 99 percent of the cases. The hops are the number of networking devices between the source and destination of a packet. The lower the number of hops, the better for communication. The number of hops suggested that the processing nodes were located close in the datacenter as expected by the usage of affinity groups in Azure. Contrasting the latencies obtained in Azure, the local cluster with only one hop had an average latency of 12µs. For HPC applications like BRAMS, the latency in Azure network impacts the scalability and performance.

We also evaluate if latency values were altered using different instance sizes. In order to do that we used the six types of instances selected and instantiated 32 VM of each one, we measured latency between all the nodes, five times from each node.

Results of this evaluation are show in Figure 6.3. Some instances sizes, namely A8 and A9 had low latency compared with the other instance sizes. A8 and A9 are instances advertised for its networking capabilities; they have Infiniband connections only usable from Windows-based deployments. Based on the results obtained we could infer that the network infrastructure for this type of machines is better even when not using Infiniband. Latency in average on the A8 was 29.78 µs, on A9 was 25.93 µs. Instance type A7 presented 60.14 µs latency with a high variability. On D13 and D14 instances results were closer to the previous ones, D13 74.77 µs and D14 64.45 µs. Variability on D14 was the lowest of the 155 tests performed with each instance type with 4.72 percent.

The relatively high latencies could impact the deployment of larger BRAMS simulations in Azure. The networking latencies in this tightly coupled application could be limiting factor for running bigger experiments due to the performance loss compared with the gains in raw processing power.

6.3 Provisioning and Controlling Operations

The life cycle of a VM instance includes the steps to create, start, reboot, stop and delete the instance. Those operations are an essential part of the pay-as-you-go model since each deployed VM has to pass for some of them. The cloud provider begins charging from the moment of the VM creation to the release of its resources, not only for the actual computing usage. We measured the time spent in each one of those operations.

Results are presented in Figure 6.4. We noticed that in some cases, the time spent in node creation is more than twice the time of the other three operations. We found a significant variability, up to 16 percent in the case of stopping nodes. The behavior observed could be due to a situation generated by the command line tools of Azure. We found that it was not possible to create and add a VM to a cloud service without assigning it a public SSH endpoint, even if the VM was not going to use it. Regarding security concerns, we proceed to delete the SSH endpoint after the provisioning of the VM. This operation halts the creation of new nodes until it finishes. Besides this, the Azure subscription is restricted to only one deployment operation at the same time, preventing a faster generation of experiments that include instantiating a lot of VMs. The time it would take to start or create a greater number of VMs could be larger than the time the experiment would take to finish.

Time spent in operations not related to an actual processing usage could impose an adoption barrier to analyze further HPC applications in Azure. The processing stage in applications like BRAMS require all the processing nodes ready before starting. In this case, all the instances already allocated are in the waiting for the last instance in the deployment queue to start. An approach to minimize this delay is not recreating the instances but redeploying. However, batch operations from the

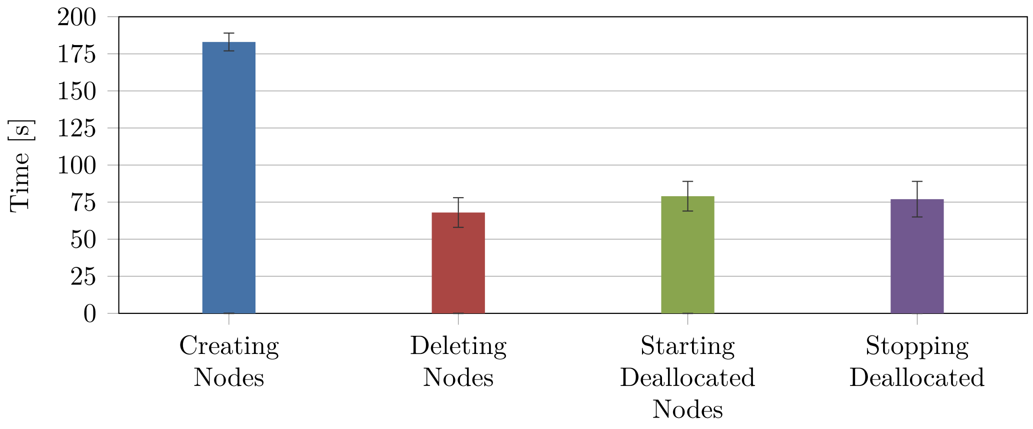


Figure 6.4 – Time spent on the deployment operations in the cloud service, Provisioning/Controlling. Time per operation.

service provider are needed to simplify this kind of situations.

The delay creates a non-processing state in which the VMs already started are waiting to start processing but waiting noticeable amounts of time before performing a useful computation. In this waiting state, all the idle VMs generate costs that are expected to be reduced by the usage of a cloud infrastructure.

6.4 Scalability

The results of the processing section, shown in Table 6.1 showed some peculiarities. The execution time in some cases was faster with 56 cores than with 64. In order to analyze if the execution time of BRAMS is reduced more after 64 cores, we measured the processing scalability of the application. In this case, we were not focused on running the experiments on different instances but in a larger number of them. BRAMS was scaled until reaching 168 cores.

The first experiment was performed with standard A4 and A7 instances, VMs with eight cores each but different memory sizes. As depicted in Figure 6.5 the application was not able to scale well beyond 88 cores. Even using that amount of cores is not worth it, passing 56 cores the time reduction is less than two percent. Basically, the application is not scaling horizontally as much as expected.

We evaluated if the reason why BRAMS did not scale was because the instances used were not the ones with the best performance in the previous experiments. We performed another pair of experiments using the instance sizes with better networking capabilities.

The results are shown on Figure 6.6. This experiment was performed using

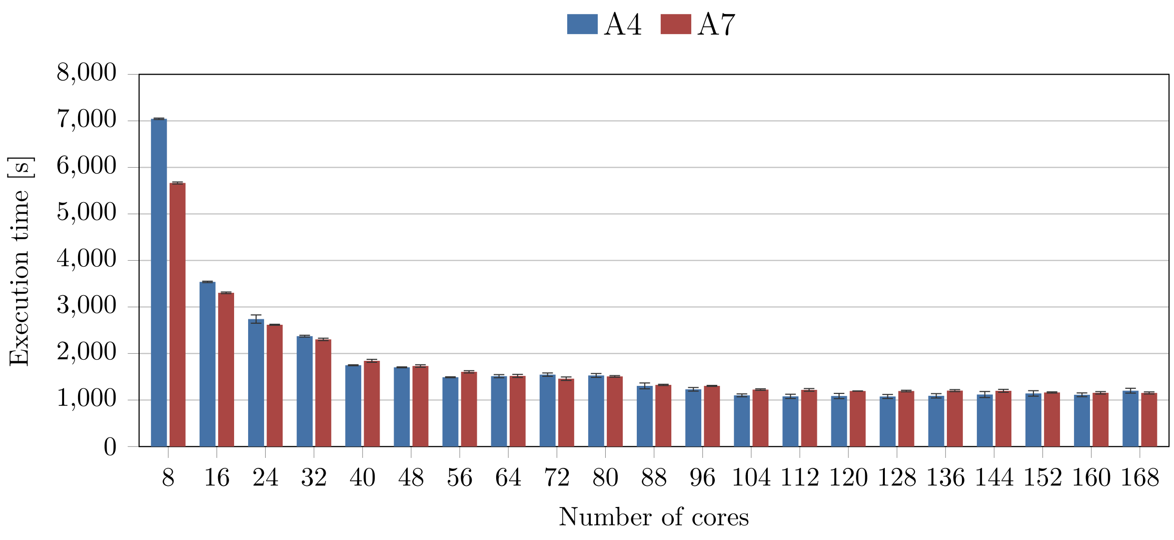


Figure 6.5 – Scaling experiments with 168 cores using a 20KM model resolution with A4 and A7 VM instances.

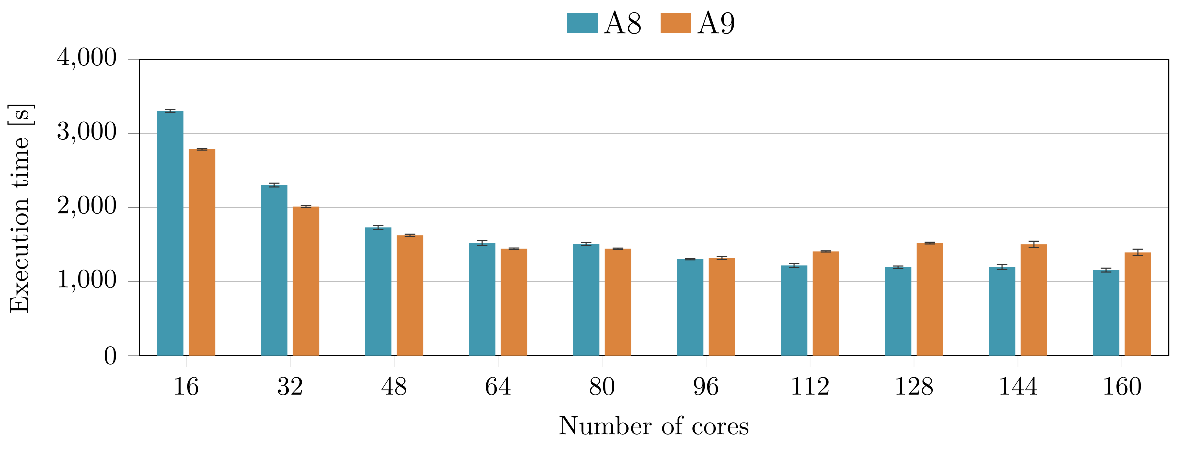


Figure 6.6 – Experiments with 160 cores and a 5KM model resolution using A8 and A9 VM instances.

A8 and A9 instances, because of the difference in the number of cores, we run the forecasts only using 16 cores at a time for the A9 instance or pairs of A8 instances.

In this case, BRAMS behaves a little different but also confirming that scalability was limited. The application scaled correctly up to near 80 cores. After that point, performance was worse with each node added to the A9 instances, for the A8 after 88 cores performance gains were low.

Regarding the experiment results of this section we conclude that there is a very specific point in which is better to use a few virtual machines. That point would depend on the characteristics of the application. In some cases, running the application on instances with low-performance characteristics would be a better option than scaling, especially when the performance benefits are so minimal that running the application would mean wasting the available budget.

6.5 Pricing

The price of using BRAMS to execute weather forecast are subject to the fundamental aspects of the cloud model, storage, networking and utilization time of those resources. To simplify the calculations, we are going to use a period of one month, running BRAMS every 6 hours, using a growing amount of data and an established bandwidth usage.

We use a period of one month to calculate deployment prices because it is the minimum billing period for our subscription for the services in Azure. The price of an experiment depends on the amount of storage used, the network transfer caused by communication outside the datacenter and the deployed and running time of the instances. Other aspects of pricing as the deploying operations were are not taken into account.

In order to calculate the price of an experiment we need to evaluate the following equation:

$$P_E = S + N + P_t \tag{6.1}$$

Where P_E is the final price of an experiment, S is the price of amount of storage used, N is the cost of networking bandwidth and P_t is the cost of running the selected VM during the processing time.

The next subsections describe how each variable on Equation 6.1 was calculated and then the final price of running the experiments is discussed.

6.5.1 Storage

The storage used is calculated as follows In Azure storage is charged as the average of time of the month that a gigabyte was used. Pricing of using Local Redundant Storage (LRS) or Geographically Redundant Storage (GRS) varies. The fundamental difference between both of them is the amount of copies performed on saved data and the location of the copy. Saving more copies and using different datacenters are more expensive. For some kinds of applications with critical data, data replication has benefits. The drawback on replication is that data throughput is reduced, as when using GRS. The prices are shown in Table 6.2, prices are for the first stored terabyte per month.

Table 6.2 – Price per gigabyte on Azure Files in USD dollars.

Datacenter	Locally Redundant Storage (LRS)	Geo-redundant storage (GRS)
Europe North	0.040	0.050

To calculate the price we use the information from Table 3.1, it includes the amount of yearly input data required by BRAMS. We are going to assume for these two scenarios that in a month, in average, a twelve part of that data is going to be stored. That means 10.17 GB data stored per month, only for input data.

For the output data, the calculation is trickier, from what we could conclude from our experiments and the forecasts results, output data size depends on the grid size configured on BRAMS, not the model resolution. Table 6.3 shows the results on data usage on each experiment. If output data is not deleted after a forecast, the cost associated with this data increases nearly the same amount every 6 hours.

At the end of the month, we could have 152 GB of output data in total, but by using the average of each experiment, the service provider will charge for 18.05 GB of the output data. This value could be calculated as a progression of the value

Table 6.3 – Data size of output files in one BRAMS experiment.

Output data	Size [GB]
MAKESFC & MAKEVFILE	0.012
Analysis	1.100
History	0.147
Post-process	0.008
Total	1.267

charged by the fraction of the month that the output files have been stored. So the price paid each month for storage will be the value of the input and output data sizes multiplied by the datacenter storage price:

$$\begin{aligned} S &= (S_i + S_o) \times DC_s \\ &= (18.05 + 10.17) \times 0.40 \\ &= 11.29 \text{ USD/Month} \end{aligned} \tag{6.2}$$

Where S_i is the amount of stored input data, S_o represents the amount of stored output data and DC_s is the price for storage in the selected datacenter. This monthly cost is calculated only for the first month, as storage usage grows with time, usage has to be optimized to prevent uncontrolled growing, reducing the costs associated.

6.5.2 Networking

As discussed in Section 4.1.2 we used a feature of Azure to reduce the costs related to bandwidth usage. To compare the cost reduction of using this feature for BRAMS, we calculate the price of having the data stored in a different datacenter, generating traffic in the same zone. Transfer prices are shown in Table 6.4

Table 6.4 – Outbound data transfer prices, price per gigabyte on USD.

Transfer Size	Zone 1	Zone 2	Zone 3
5 GB–10.0 TB	0.087	0.138	0.181
10–50 TB	0.083	0.135	0.175
50–150 TB	0.070	0.130	0.170
150–500 TB	0.050	0.120	0.160

Zone 1: US West, US East, US North Central, US South Central, US East 2, US Central, Europe West, Europe North
Zone 2: Asia Pacific East, Asia Pacific Southeast, Japan East, Japan West, Australia East, Australia Southeast
Zone 3: Brazil South

In the first networking scenario, we use the Zone 3, that includes the Azure datacenter located in Brazil. In this case, we would need to pay for the networking transfers outside Zone 3.

As calculated in the previous subsection the output data from the experiments is 152 GB. Transferring this amount of between nodes would be charged using the first limit shown in Table 6.4.

$$\begin{aligned} N &= T_o \times DC_n \\ &= 152 \times 0.181 \\ &= 27.51 \text{ USD/Month} \end{aligned} \tag{6.3}$$

Where T_o is the amount of outbound transferred data and DC_n is the price of transferring outside the selected datacenter.

In the second scenario, all the BRAMS deployment is inside the same datacenter. Because the price is zero no matter which datacenter is used in this scenario, we could use Zone 3 for calculation again.

$$\begin{aligned} N &= T_o \times DC_n \\ &= 152 \times 0.00 \\ &= 0.00 \text{ USD/Month} \end{aligned}$$

In Section 4.1.2 we discussed the fact that using this feature represented a big benefit while porting BRAMS to the cloud. Having a 100% price reduction in some of the core aspects of the cloud represents significant savings in a cloud deployment.

These two scenarios are analyzed because storing the data near the source, in this case the INPE in Brasil, could lead to better transfer speeds. On the other hand, price difference on the computing instances should compensate . Most of the instance sizes in Brazil datacenter are more expensive when compared to the instances prices in the other Azure datacenters.

6.5.3 Processing

Using the performance results obtained in Section 6.1 we calculate the price of running each experiment depending on the number of instances deployed. The prices were calculated using the following expression:

$$P = P_t \times VM_n \times DC_p \tag{6.4}$$

Where P_t is the execution time of the experiment, VM_n is the number of VMs deployed for the experiment and DC_p is the Datacenter price of the deployed instance size.

Table 6.5 – Price per experiment based on instance size, number of instances and duration of the experiment. Include only include compute costs. Prices in USD.

Cores	A4	A7	D13	D14	A8	A9
8	1.41	2.35	1.60	1.47	2.25	4.49
16	1.38	2.34	1.66	1.18	2.36	2.36
24	1.60	2.68	1.98	2.37	2.83	3.75
32	1.85	3.26	2.42	2.11	3.51	3.43
40	1.76	3.11	2.38	2.51	3.31	4.07
48	2.02	3.68	2.79	2.40	3.84	4.04
56	2.03	3.72	2.92	2.89	4.10	5.07
64	2.30	4.28	3.30	2.98	4.69	5.22

The price of the instance is charged by the minute even if it is commonly advertised as by the hour. Results of are depicted in Table 6.5.

All the prices were calculated using Europe North datacenter. The prices were obtained in January of 2015.

6.5.4 Final Cost per Experiment

With all the variables calculated we could finally arrive at the final costs of running forecast experiments using BRAMS in the cloud. Using Table 6.5 data and the results of Equation 6.2 and Equation 6.3. we could use the Equation 6.1.

Results of computed final price are depicted in Table 6.6. The price reduction by using networking in same datacenter is about 5.88 percent.

We can conclude that taking advantages of the features of a cloud provider, and creating an automated environment we could perform high performance computing in the cloud with low budget. Regarding the reductions in networking costs, in practice the deployment will have some amount of outbound transfer, even using

Table 6.6 – Price per experiment based on instance size, number of instances, duration of the experiment, storage and network transfer. Prices in USD.

Cores	A4	A7	D13	D14	A8	A9
8	1.50	2.44	1.69	1.56	2.34	4.58
16	1.47	2.43	1.74	1.27	2.45	2.45
24	1.69	2.77	2.07	2.46	2.92	3.84
32	1.94	3.35	2.51	2.20	3.60	3.52
40	1.85	3.20	2.47	2.60	3.40	4.16
48	2.11	3.77	2.87	2.49	3.93	4.13
56	2.12	3.81	3.01	2.98	4.19	5.16
64	2.39	4.37	3.39	3.07	4.77	5.31

an SSH connection would be charged because is accessed from outside the cloud infrastructure. However, keeping most of the data in the cloud and transferring only the data that critically needs to get out of the cloud, as the forecast visualization files, the transfer amount will still be low compared to moving complete datasets.

6.5.5 Cost Efficiency

The final decision about moving entirely to the cloud based on pricing depends completely on the user constraints. For example, for a user it would be acceptable having results in 1438.96 seconds paying $2.39, as in the results we obtained using A4 instances. However, another user could consider that time unacceptable and set its requirements on 860.58 seconds. In that case, the user would pay for A8 instances, $4.77

A question that could arise from those case examples is if it is worth paying almost two times more for a 40 percent reduction in execution time. Or as the issue related to A9 instances, without insight comparing different results, the end user could end up paying for a more expensive machine and receiving worse performance.

Analyzing the user constraints is complicated because of the amount of evaluations involved. In order to simplify this process, we could use a cost efficiency model to help the user in the decision to move to the cloud. In Figure 6.7 the results of execution time are plotted against the execution cost of an experiment. Two execution time values were not plotted to allow easier scaling the number of symbols in the figure. The values are over 7,000 seconds in the A4 and A7 instances.

An example of a user constraint could be the following: we would execute forecasting using BRAMS in the cloud only if execution time is less than 1,500 seconds and if price per experiment is less than $3 USD.

In that case, the selection of the instances inside the area of those two constraints is a reduced subset of all the possible options. With those constraints, the user would probably select D14 with 48 cores because is cheaper than other options and execution time difference is low enough to be accepted.

6.6 MBRAMS: BRAMS as Multiple Cloud Services

We performed experiments over several geographic areas as proposed in Section 5.2. This group of experiments consisted of running a complete weather forecast scaling horizontally from one node up to 8 nodes in Azure. The Figures show the average execution time of 5 runs of the INITIAL stage in the forecast simulation for each number of nodes using A8 instances. BRAMS stages MAKESFC and MAKEVFILE were not taken into account because they were developed to use only one CPU core,

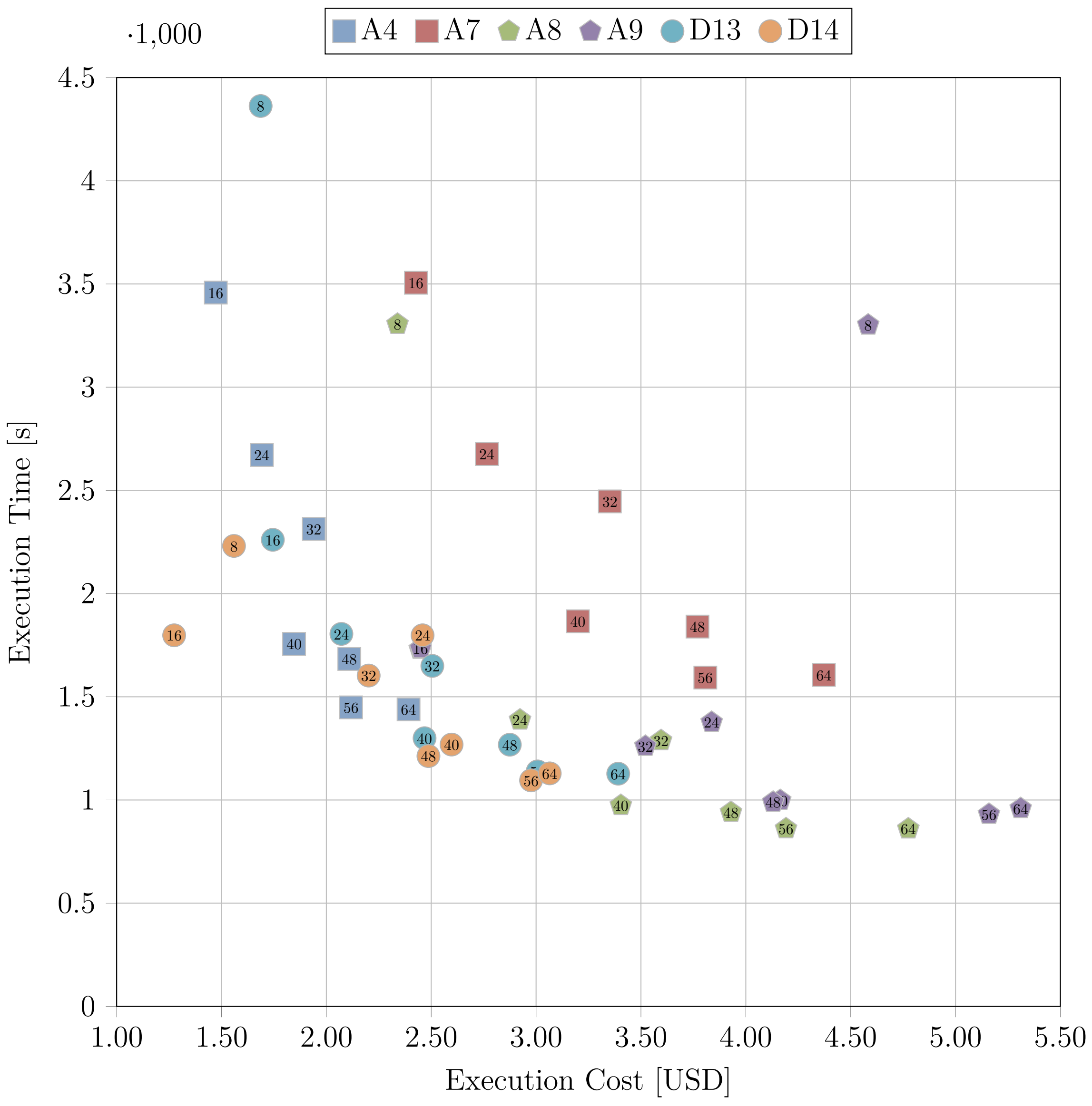


Figure 6.7 – Cost-efficiency of BRAMS instances deployed on Azure. The number inside the symbol indicates the number of cores. Instances closer to the origin have a higher cost efficiency ratio.

that means they receive no performance gains by changing the number of nodes. Results are detailed taking into account the scenarios sharing storage accounts in *MBRAMS*.

6.6.1 Processing and Storage

The first experiments evaluate the usage of a sharing storage account. Each one of the deployments of BRAMS has access to two storage accounts. One account shared containing input data and another for each deployment output data.

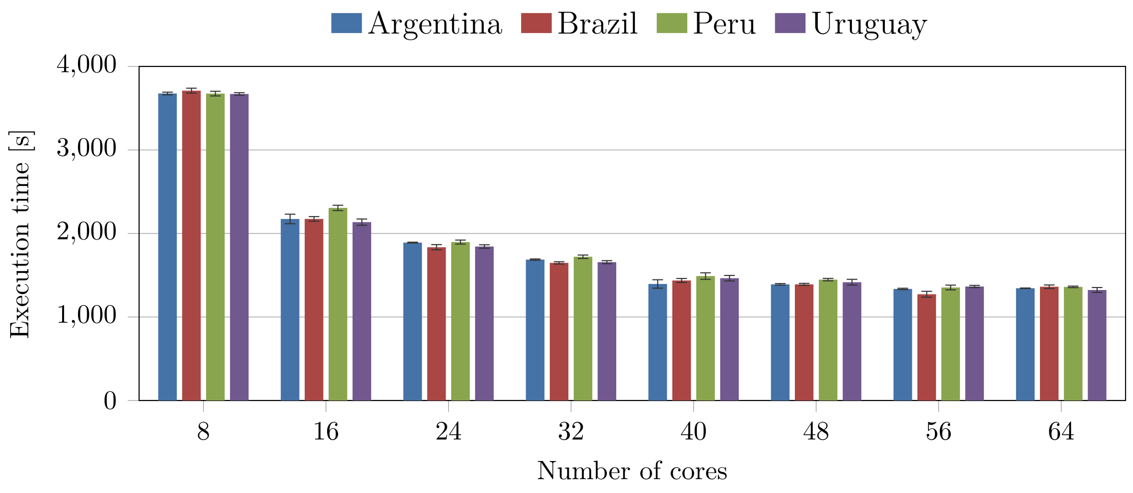


Figure 6.8 – Average execution time reduction while using a shared storage account for input and output data. All tests using A8 VM instances.

The results of running forecast over the four geographic areas using 50KM resolution are depicted in Figure 6.8. As seen in the Figure, results were homogeneous between the four deployments. Scaling was consistent between deployments all the way to 64 cores. Standard deviation was low in the experiments, under three percents in all cases. These results show the usage of a shared storage account for input and output data.

Executing the same set of experiments but setting the storage of output files into another storage account yield interesting results. These results are depicted on Figure 6.9. As seen in the Figure, there was a significant reduction in the execution

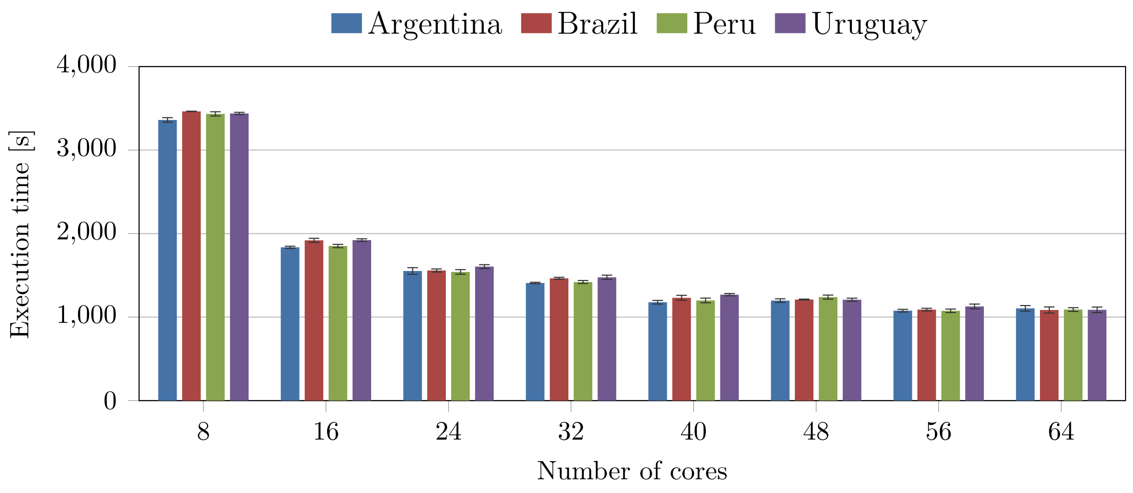


Figure 6.9 – Average execution time reduction while using a shared storage account only for input data. All tests using A8 VM instances.

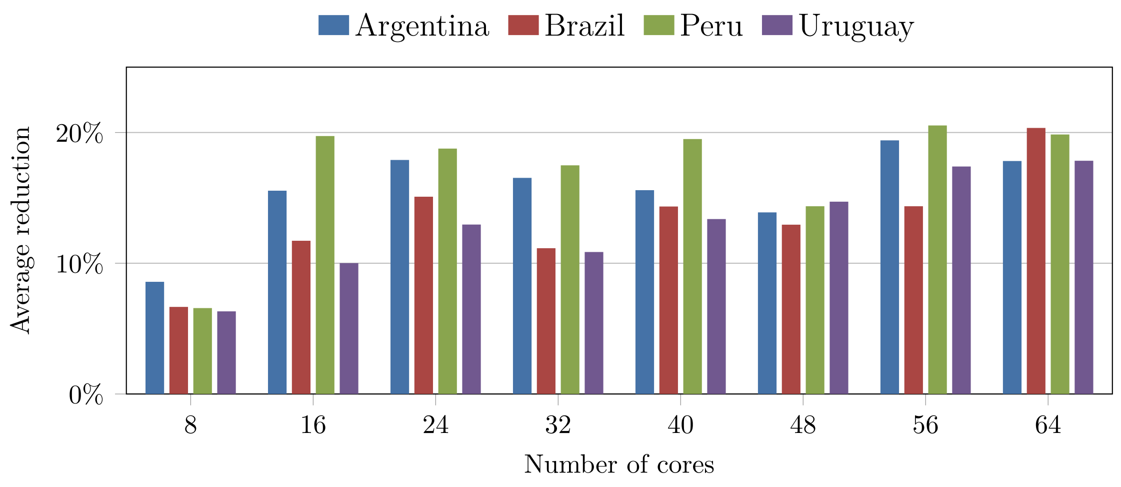


Figure 6.10 – Average reduction in execution time while using a shared storage account only for input data. All tests using A8 VM instances.

time for all of the experiments. The execution time reduction is significant using this storage approach.

A comparison of the results is shown in Figure 6.10. On average time reduction was 14.75 percent, in the cloud as *time is money*, these results reduce the cost of the experiments. Those results confirm our ideas about how sharing cloud storage impacts performance for *MBRAMS*. It is important to say that using different storage accounts does not increase the cost of the deployment. In Azure, the number of storage accounts is not billed. However, the user subscription has a limit on the number of accounts that can create. We conclude that is possible to optimize the impact of storage on performance, but an in-depth analysis of how storage should be used while keeping the system manageable is required.

6.6.2 Scaling Resolution

In order to evaluate this *MBRAMS* scenario in a more detailed forecast, we performed more experiments scaling the BRAMS model resolution. These experiments used the storage system sharing input data but storing in its own account. The first group of experiments was conducted using a 20KM spatial resolution. Results are depicted in Figure 6.11.

As shown in Figure 6.11 the results were similar to the previous ones, execution time was acceptable, and the application managed to scale. *MBRAMS* behavior on the reduction in the percentage of performance gains as more instances were added was similar to that found in the previous experiments when scaling up to 64 cores.

Interestingly, execution time did not increase that much with the change in model resolution. We expected at least twice the execution time, but on average the increase

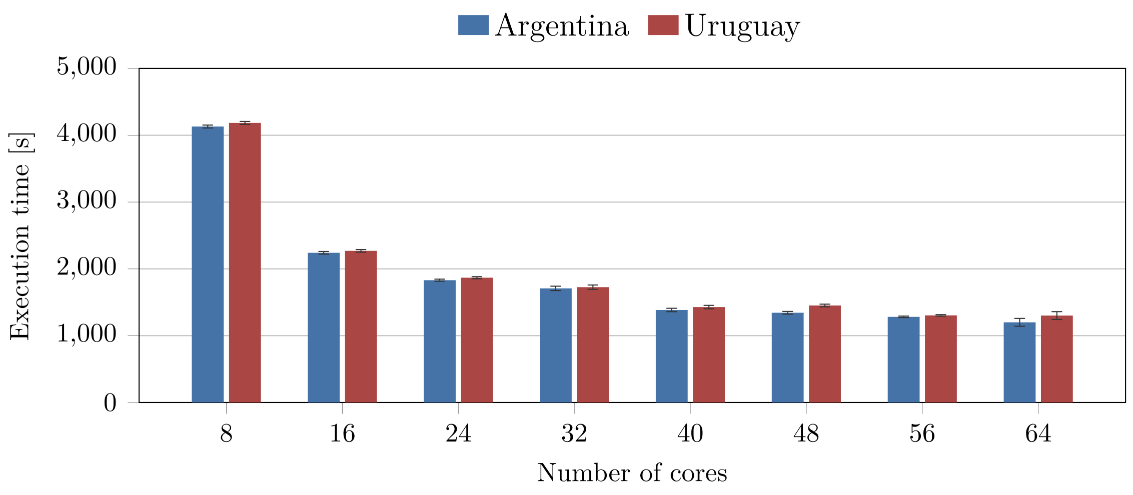


Figure 6.11 – Average execution time reduction while using a shared storage account only for input data. All tests using A8 VM instances.

was nearly 18% in both cases, with 4.99% standard deviation in the case of Argentina and 2.9% for Uruguay.

The next scaling experiment was performed scaling to a 5KM model resolution. Figure 6.12 show the results. *MBRAMS* deployments scaled correctly, time to execution increases more than in 20KM experiments. Again execution time was not heavily impacted; this time resolution change was greater than in the previous experiment and execution time increased 26.75% in average, with 4.22% standard deviation.

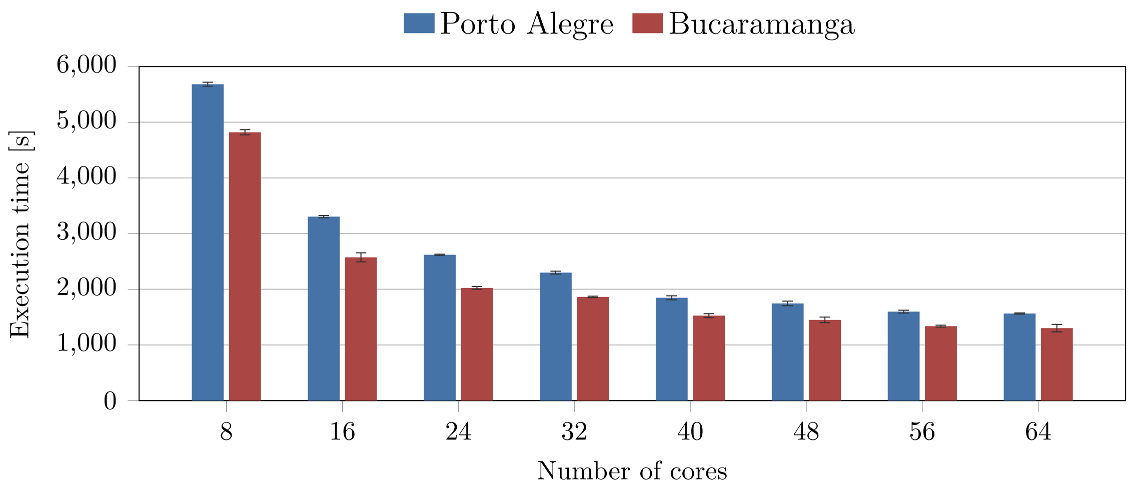


Figure 6.12 – 5KM model resolution, Average execution time reduction while using a shared storage account only for input data. All tests using A8 VM instances.

6.7 BRAMS Post-processing

The post processing stage requires the sequential execution of GrADS[1]. GrADS, Grid Analysis, and Display System, allows access, manipulation, and visualization of BRAMS and other scientific applications. The tool reads the output data of BRAMS and extracts the data for each forecasted variable and print the output to a graphic file.

We used the post-processing output files to visualize if output data was consistent between experiments. Those graphics are also automatically generated by using the developed scripts that control BRAMS executions. We used BING maps[2], a mapping technology from Microsoft to include location information in the background of the forecast resulting image. The purpose of this section is to show some examples of this output files and how was the integration with BING maps. Visualization examples

[1]http://www.iges.org/grads
[2]http://www.bingmapsportal.com

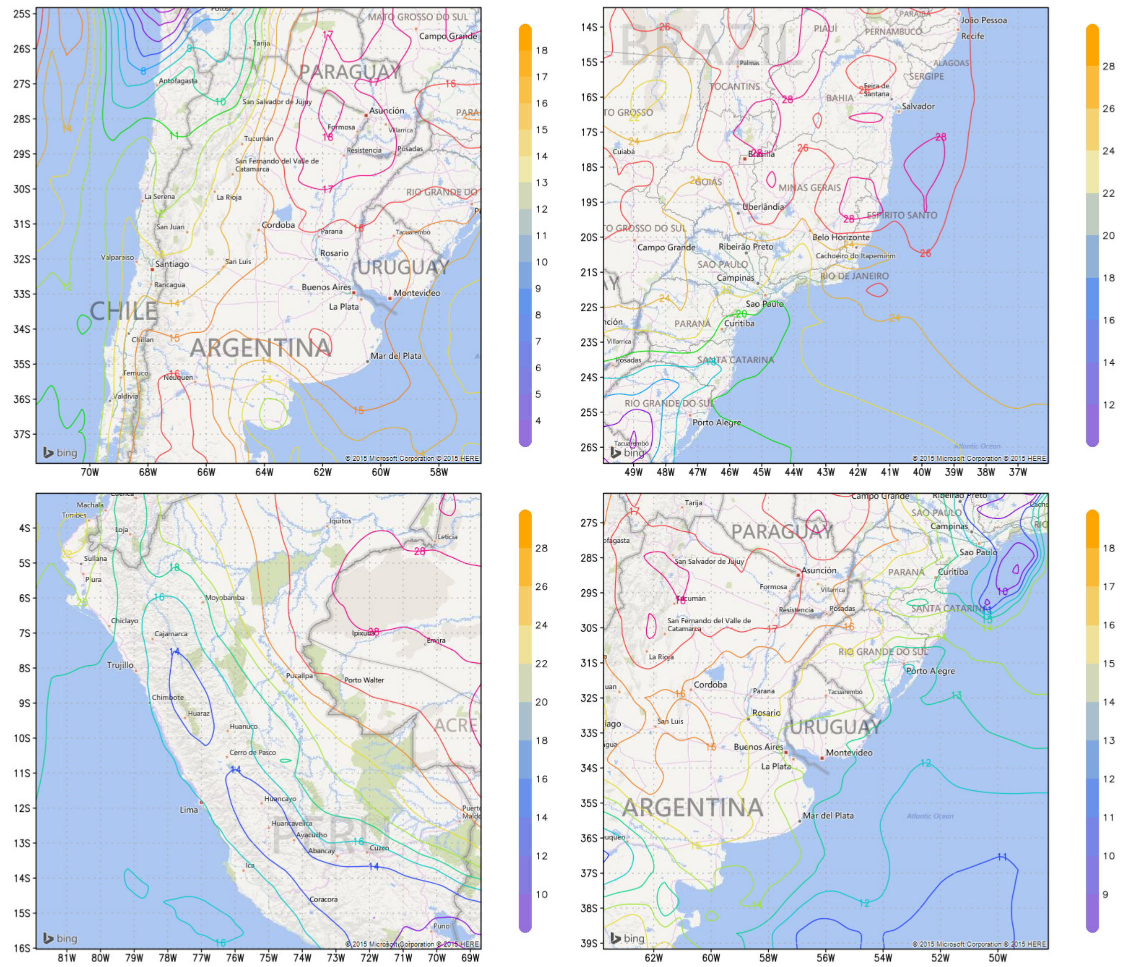


Figure 6.13 – 2 meter height air temperature in celsius degrees, resolution 50KM

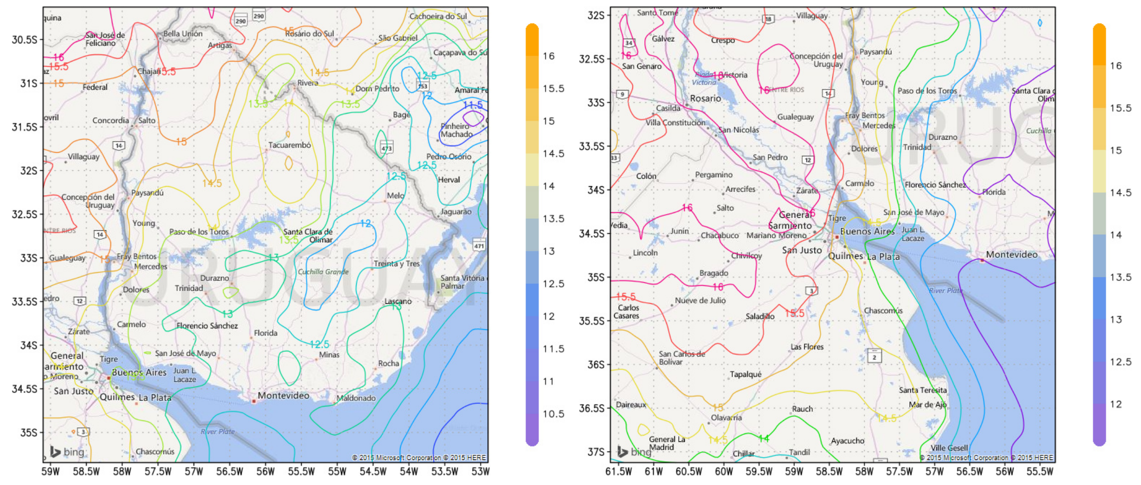


Figure 6.14 – Post processing on the 20 KM forecast results

of the output obtained while running *MBRAMS* are depicted as follows. All the figures were generated using the actual forecast produced by BRAMS as part of the adapted workflow in the cloud. The figures from a forecast of the same date and hour in all cases.

Figure 6.13 shows the output of the forecasted area, country level, for each one of the experiments using the 50 KM resolutions of BRAMS. The four deployments of BRAMS were executed simultaneously. Figure 6.14 shows the output of the forecasted area, state level, for each one of the experiments using the 20 KM resolutions of BRAMS. Both deployments of BRAMS were executed simultaneously. Figure 6.15 shows the output of the forecasted area, in this case, at the city level, for each one of the experiments using the 5 KM resolutions of BRAMS.

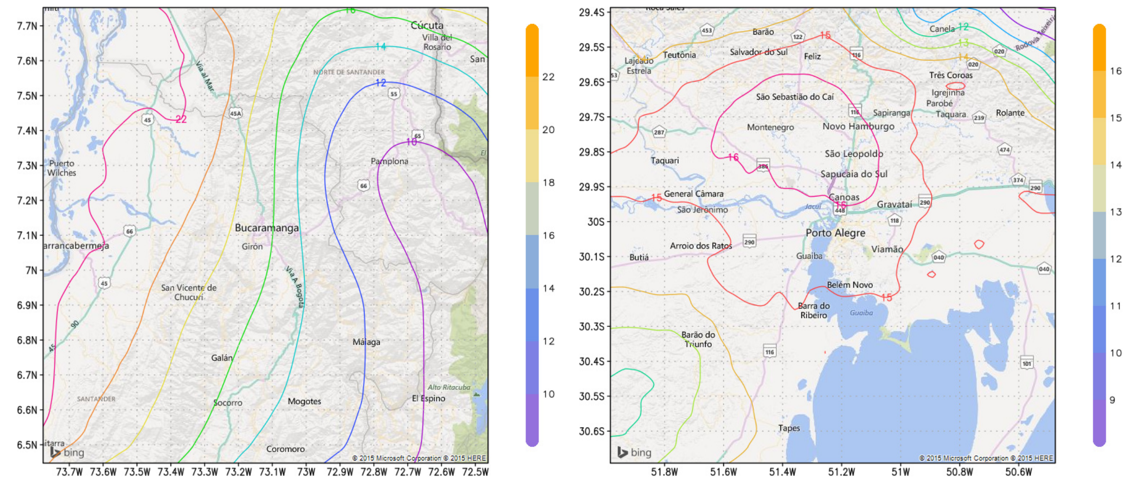


Figure 6.15 – Post processing on the 5 KM forecast results

7 CONCLUSION

This work presented the challenges in running an HPC application in a cloud computing infrastructure using IaaS. We present our solutions for the process of porting BRAMS to the Microsoft Azure Cloud platform. Performance results show that cloud infrastructure offers possibilities to move applications from legacy code and that it is possible to provide an environment for the application while reducing the setup complexity when running HPC applications. Variability between experiments was low in Azure, which is an important characteristic for HPC as it leads to a higher predictability of experiments, both in the time it takes and how much it will cost.

The scalability experiments show that is possible to scale BRAMS up to 168 CPU cores with certain mistrust. Our results showed that reduction in execution time beyond 88 CPU cores is less than two percent at every scaling step. Also, we performed scaling test with BRAMS model resolution from 50km to 5km in the cloud deployment.

In the networking aspect of the migration, by using the same datacenter for the deployment we reduced the transfer bandwidth cost to almost zero percent during the processing stage. This decision reduced in nearly 6% the total costs executing BRAMS in the cloud. We obtain network latency results of 70 µs in average, with a 24% standard deviation for some instance sizes. The instances with better network capabilities showed better results in this aspect with 25.93 µs, near the 12 µs latency of the local cluster. The relatively high latencies create an issue for the deployment of larger BRAMS simulations in Azure and in general for any kind of HPC application ported to this infrastructure.

Regarding storage, by defining an storage condition for output files it was possible a reduction of 14.75% in execution time when multiple BRAMS deployments are using the shared storage accounts for input files.

The overhead caused by latencies and the time spent in operations not related to an actual usage could impose an adoption barrier to analyze further HPC applications in Azure. This overhead creates a non-processing state in which the VMs already started are waiting to start processing but waiting noticeable amounts of time before performing a useful computation. In this waiting state, all the idle VMs generate costs that are expected to be reduced by the usage of a cloud infrastructure.

The lifecycle technique developed for this thesis could be applied to others applications in order to make the managing process easier, reducing costs when moving to the cloud. The cost efficiency plot allows a quick and precise way to check if user constraints allows the execution of an application in the cloud. One of the expected primary contributions of our work is to create the possibility of small research centers in South America to perform weather prediction of their regions.Without depending on supercomputers, and with limited, or absent, upfront costs. Using shared storage accounts to allow the creation of research networks using the same datasets.

We expect that performing this kind of experiments in other cloud providers as Amazon EC2 and Google Compute shows similar results, as most of the commodity hardware in their datacenters still uses high-latency network links. This network-intensive usage limitation is not unique of BRAMS, as it is present in more HPC applications using the distributed memory paradigm. We hope that the experience obtained can be applied to port other HPC applications. Cloud computing is an evolving, promising alternative, as long as providers upgrade their hardware and keep lowering the prices of their services, driven by the competitive market.

For the future, we intend to perform a full analysis of the cost of running BRAMS simulations in the cloud, as some aspects of the deployment cost require a more detailed study. We also plan to capture more metrics to cover all the aspects of the execution and try to improve the performance of this HPC application in a cloud environment.

REFERENCES

ANDRIKOPOULOS, V. et al. How to adapt applications for the Cloud environment: Challenges and solutions in migrating applications to the Cloud. **Computing**, v. 95, n. 6, p. 493–535, 2013. ISSN 0010485X. One citation in page 31.

BENEDICT, S. Performance issues and performance analysis tools for hpc cloud applications: a survey. **Computing**, Springer Vienna, v. 95, n. 2, p. 89–108, 2013. ISSN 0010-485X. One citation in page 22.

BESERRA, P. V. et al. Cloudstep: A step-by-step decision process to support legacy application migration to the cloud. In: 2012 IEEE 6TH INTERNATIONAL WORKSHOP ON THE MAINTENANCE AND EVOLUTION OF SERVICE-ORIENTED AND CLOUD-BASED SYSTEMS, MESOCA 2012. **Proceedings...** Trento, Italy, 2012. p. 7–16. One citation in page 31.

BROWN, D. et al. Scientific grand challenges: Crosscutting technologies for computing at the exascale. **Pacific Northwest National Laboratory, PNNL-20168, February**, 2010. One citation in page 29.

CARREÑO, E.; ROLOFF, E.; NAVAUX, P. O. A. Challenges and Solutions in Executing Numerical Weather Prediction in a Cloud Infrastructure. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE. **Proceedings...** Reykjavík, Iceland, 2015. (ISCC 2015). One citation in page 26.

CPTEC-INPE. **Brazilian Regional Atmospheric Modelling System (BRAMS)**. 2014. <http://www.cptec.inpe.br/brams>. Accessed August 10, 2014. One citation in page 33.

DONGARRA, J. et al. The international exascale software project roadmap. **IJHPCA**, v. 25, n. 1, p. 3–60, 2011. One citation in page 29.

EVANGELINOS, C.; HILL, C. N. Cloud Computing for parallel Scientific HPC Applications: Feasibility of Running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2. In: IN THE 1ST WORKSHOP ON CLOUD COMPUTING AND ITS APPLICATIONS (CCA). **Proceedings...** [S.l.], 2008. 2 citations in pages 21 and 29.

EVOY, G. M.; SCHULZE, B. Understanding scheduling implications for scientific applications in clouds. In: PROCEEDINGS OF THE 9TH INTERNATIONAL WORKSHOP ON MIDDLEWARE FOR GRIDS, CLOUDS AND E-SCIENCE.

Proceedings... New York, NY, USA: ACM, 2011. (MGC '11), p. 3:1–3:6. ISBN 978-1-4503-1068-0. One citation in page 30.

FOSTER, I. et al. Cloud computing and grid computing 360-degree compared. In: GRID COMPUTING ENVIRONMENTS WORKSHOP 2008 (GCE '08). **Proceedings...** [S.l.], 2008. p. 1–10. ISBN 978-1-4244-2860-1. One citation in page 25.

FOX, A. et al. Above the clouds: A Berkeley view of cloud computing. **Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS**, v. 28, p. 13, 2009. One citation in page 25.

HAJJAT, M. et al. Cloudward Bound : Planning for Beneficial Migration of Enterprise Applications to the Cloud. **Configurations**, ACM, v. 40, n. 4, p. 243–254, 2010. One citation in page 31.

HERNÁNDEZ, S. et al. Cost evaluation of migrating a computation intensive problem from clusters to cloud. In: ALTMANN, J.; VANMECHELEN, K.; RANA, O. (Ed.). **Economics of Grids, Clouds, Systems, and Services**. [S.l.]: Springer International Publishing, 2013, (Lecture Notes in Computer Science, v. 8193). p. 90–105. ISBN 978-3-319-02413-4. 2 citations in pages 30 and 31.

HOHENSTEIN, U. et al. Towards Cost Aspects in Cloud Architectures. In: IVANOV, I. et al. (Ed.). **Cloud Computing and Services Science**. [S.l.]: Springer International Publishing, 2013, (Communications in Computer and Information Science, v. 367). p. 117–134. ISBN 978-3-319-04518-4. One citation in page 31.

HUBER, N. et al. Evaluating and modeling virtualization performance overhead for cloud environments. In: CLOSER. **Proceedings...** [S.l.], 2011. p. 563–573. One citation in page 25.

HUMPHREY, M. et al. Assessing the value of cloudbursting: A case study of satellite image processing on windows azure. In: E-SCIENCE (E-SCIENCE), 2011 IEEE 7TH INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.], 2011. p. 126–133. One citation in page 30.

IOSUP, A. et al. Performance analysis of cloud computing services for many-tasks scientific computing. **IEEE Trans. Parallel Distrib. Syst.**, IEEE Press, Piscataway, NJ, USA, v. 22, n. 6, p. 931–945, jun. 2011. ISSN 1045-9219. One citation in page 21.

ISO/IEC. **Information technology – Cloud computing – Overview and vocabulary**. [S.l.]: ISO, 2014. v. 2014. One citation in page 25.

JOHNSTON, S.; COX, S.; TAKEDA, K. Scientific computation and data management using microsoft windows azure. In: FIORE, S.; ALOISIO, G. (Ed.). **Grid and Cloud Database Management**. [S.l.]: Springer Berlin Heidelberg, 2011. p. 169–192. ISBN 978-3-642-20044-1. One citation in page 30.

LANGMEAD, B. et al. Searching for snps with cloud computing. **Genome Biology**, BioMed Central, v. 10, n. 11, 2009. One citation in page 29.

LONGO, K. M. et al. The Chemistry CATT-BRAMS model (CCATT-BRAMS 4.5): A regional atmospheric model system for integrated air quality and weather forecasting and research. **Geoscientific Model Development**, v. 6, n. 5, p. 1389–1405, 2013. ISSN 1991959X. One citation in page 33.

LU, W.; JACKSON, J.; BARGA, R. AzureBlast: A Case Study of Developing Science Applications on the Cloud. In: 19TH ACM INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING. **Proceedings...** New York, NY, USA: ACM, 2010. (HPDC '10), p. 413–420. ISBN 978-1-60558-942-8. One citation in page 30.

LV, H.; LIU, J. RCMS: Rapid Cloud Migration Solution. In: COMPUTATIONAL INTELLIGENCE AND DESIGN (ISCID), 2013 SIXTH INTERNATIONAL SYMPOSIUM ON. **Proceedings...** [S.l.], 2013. v. 1, p. 426–429. One citation in page 31.

MELL, P.; GRANCE, T. **The NIST Definition of Cloud Computing**. 26. ed. [S.l.], 2011. Available from Internet: <http://www.csrc.nist.gov/groups/SNS/cloud-computing/>. One citation in page 25.

MENYCHTAS, A. et al. Artist methodology and framework: A novel approach for the migration of legacy software on the cloud. In: SYMBOLIC AND NUMERIC ALGORITHMS FOR SCIENTIFIC COMPUTING (SYNASC), 2013 15TH INTERNATIONAL SYMPOSIUM ON. **Proceedings...** [S.l.], 2013. p. 424–431. One citation in page 31.

PIELKE, R. et al. A comprehensive meteorological modeling system - RAMS. **Meteorology and Atmospheric Physics**, Springer-Verlag, v. 49, n. 1-4, p. 69–91, 1992. ISSN 0177-7971. One citation in page 33.

RAMAKRISHNAN, L. et al. Evaluating interconnect and virtualization performance for high performance computing. In: PROCEEDINGS OF THE SECOND INTERNATIONAL WORKSHOP ON PERFORMANCE MODELING, BENCHMARKING AND SIMULATION OF HIGH PERFORMANCE COMPUTING SYSTEMS. **Proceedings...** New York, NY, USA: ACM, 2011. (PMBS '11), p. 1–2. ISBN 978-1-4503-1102-1. One citation in page 25.

REED, D. A. et al. **Computational science: ensuring America's competitiveness**. [S.l.], 2005. One citation in page 29.

ROLOFF, E. et al. Evaluating high performance computing on the windows azure platform. In: CLOUD COMPUTING (CLOUD), 2012 IEEE 5TH INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.], 2012. p. 803–810. ISSN 2159-6182. 2 citations in pages 26 and 30.

ROLOFF, E. et al. High performance computing in the cloud: Deployment, performance and cost efficiency. In: CLOUD COMPUTING TECHNOLOGY AND SCIENCE (CLOUDCOM), 2012 IEEE 4TH INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.], 2012. p. 371–378. One citation in page 30.

SIMALANGO, M.; OH, S. Feasibility study and experience on using cloud infrastructure and platform for scientific computing. In: FURHT, B.; ESCALANTE, A. (Ed.). **Handbook of Cloud Computing**. [S.l.]: Springer US, 2010. p. 535–551. ISBN 978-1-4419-6523-3. 2 citations in pages 30 and 31.

SIMON, H. Exascale challenges for the computational science community. **Lawrence Berkeley National Laboratory and UC Berkeley, Tech. Rep**, 2010. One citation in page 29.

SOUTO, R. et al. Processing mesoscale climatology in a grid environment. In: CLUSTER COMPUTING AND THE GRID, 2007. CCGRID 2007. SEVENTH IEEE INTERNATIONAL SYMPOSIUM ON. **Proceedings...** [S.l.], 2007. p. 363–370. 2 citations in pages 21 and 34.

TAK, B. C.; TANG, C. AppCloak: Rapid Migration of Legacy Applications into Cloud. In: CLOUD COMPUTING (CLOUD), 2014 IEEE 7TH INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.], 2014. p. 810–817. One citation in page 31.

TOP500. **http://www.top500.org**. 2014. Available from Internet: <http://www.top500.org>. One citation in page 29.

TRAN, V. T. et al. Size estimation of cloud migration projects with cloud migration point (cmp). In: 2013 ACM / IEEE INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT. **Proceedings...** Los Alamitos, CA, USA: IEEE Computer Society, 2011. v. 0, p. 265–274. ISBN 978-0-7695-4604-9. One citation in page 31.

VECCHIOLA, C.; PANDEY, S.; BUYYA, R. High-performance cloud computing: A view of scientific applications. In: PERVASIVE SYSTEMS, ALGORITHMS, AND NETWORKS (ISPAN), 2009 10TH INTERNATIONAL SYMPOSIUM ON. **Proceedings...** [S.l.], 2009. p. 4–16. 2 citations in pages 21 and 25.

WARD, C. et al. Workload migration into clouds - Challenges, experiences, opportunities. In: 2010 IEEE 3RD INTERNATIONAL CONFERENCE ON CLOUD COMPUTING, CLOUD 2010. **Proceedings...** [S.l.], 2010. p. 164–171. ISBN 9780769541303. One citation in page 31.

WU, J. et al. Migrating a digital library to a private cloud. In: CLOUD ENGINEERING (IC2E), 2014 IEEE INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.], 2014. p. 97–106. One citation in page 31.

YANG, X. et al. Cloud computing in e-science: research challenges and opportunities. **The Journal of Supercomputing**, Springer US, v. 70, n. 1, p. 408–464, 2014. ISSN 0920-8542. One citation in page 22.