Um Estudo da Performance de Sistemas Distribuídos para o Processamento de Streams

Otávio M. de Carvalho, Philippe O. A. Navaux
Universidade Federal do Rio Grande do Sul
Grupo de Processamento Paralelo e Distribuído
{omcarvalho,navaux}@inf.ufrgs.br

Introdução

O processamento de tarefas intensivas em dados envolve, historicamente, o processamento em lotes de grandes conjuntos de dados estáticos, utilizando conjuntos de máquinas interconectadas via rede. Visando processar esse tipo de dados, sistemas de gerenciamento de banco de dados *(DBMSs)* e *frameworks* de processamento distribuído, como o por exemplo *MapReduce*, se tornaram populares. Ao longo do tempo, foi demonstrado que estes modelos, focados no processamento de grandes lotes de dados, não são capazes de atender à demandas por baixas latências de processamento. Uma nova gama de aplicações vem sendo desenvolvidas, focadas no processamento de dados em tempo-real ou quase tempo-real, e novas ferramentas são necessárias para elas possam se tornar escaláveis.

Visando atender à demanda de processamento de grandes volumes de dados, em baixas latências, novas ferramentas foram desenvolvidas. Estas ferramentas são o resultado da combinação das aplicações clássicas de *Stream Processing* e das novas idéias oriundas do modelo *MapReduce*. Dentre essas ferramentas de *Distributed Stream Processing* (Processamento Distribuído de Streams), destacam-se o Apache Storm e o Spark Streaming, que analisaremos à seguir.

Storm

O Apache Storm é um *framework* para desenvolvimento de aplicações de processamento de fluxos de dados, de forma distribuída, com aplicações descritas por *DAGs* (Grafos Direcionais Acíclicos).

Suas aplicações são compostas por *Spolts*, que fornecem os eventos para a aplicação; E por *Bolts*, que implementam modificações sobre as tuplas a elas passadas.

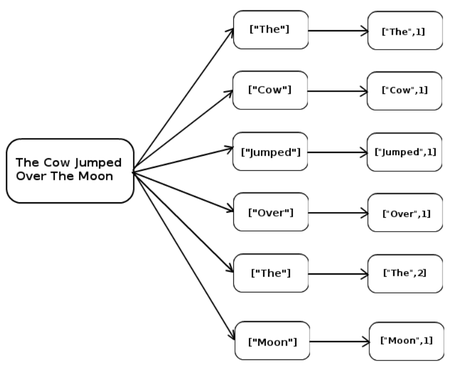


Figura 1 – Diagrama Ilustrativo do Storm

Spark Streaming

O Spark Streaming é um *framework* para desenvolvimento de aplicações de processamento de fluxos de dados distribuídos, que faz parte do conjunto de aplicações do*Apache Spark.* O seu conceito principal são os RDDs (Conjuntos de dados distribuídos e resilientes), que são uma especilização dos batches, em uma granularidade mais fina. Isso faz com que seja possível processar fluxos de dados compondo a execução paralelamente através comunicação entre múltiplos batches de curta duração, e reutilizando a arquitetura do *Apache Spark* como plataforma de execução.

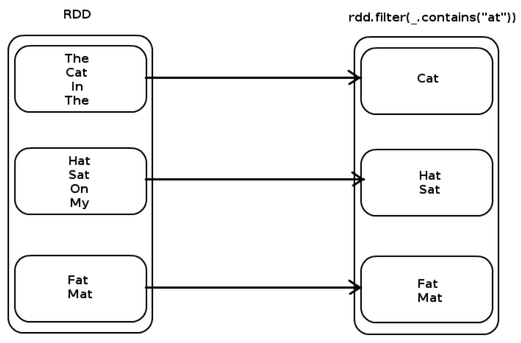


Figura 2 – Diagrama Ilustrativo do Spark Streaming

Ambiente

O ambiente utilizado consistiu na utilização da plataforma Microsoft Azure, onde foram utilizadas instâncias de máquinas virtuais, 8 nodos ao total, utilizando a seguinte configuração:

Nodos Azure A3:	Aplicações:
- Ubuntu 13.10 - 4 cores - 7GB RAM	- Storm 0.9.2 - Storm-YARN 1.0-alpha - Zookeeper 3.4.3 - Apache Spark 1.0.2 - Apache Flume 1.4.0 - Java 1.7.0

Análise

A análise das ferramentas foi realizada através da aplicação de fluxos constantes de dados, utilizando a ferramenta Apache Flume. Após definida a entrada de dados, foram desenvolvidas aplicações de micro-benchmark, preparadas para analisar a performance máxima alcançável pelas plataformas. O que foi posteriormente avaliado, através do ambiente preparado para a realização dos testes.

Através do surgimento de questionamentos sobre o impacto da utilização da ferramenta de coordenação de tarefas, denominada Apache YARN, sobre a performance geral do Apache Storm, e através da percepção da sua rápida adoção, foram realizados testes comparativos. Os testes não apontaram diferenças significativas de impacto na performance, mas demonstraram o grande potencial da ferramenta para gerenciar grandes conjuntos de máquinas.

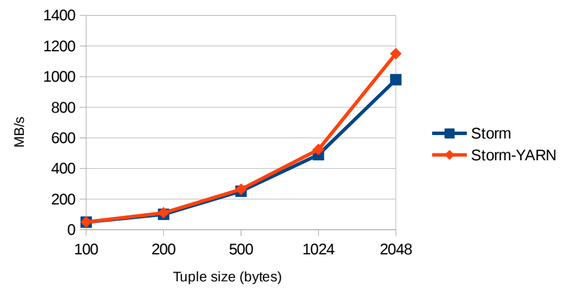


Figura 3 – Storm x Storm-YARN

O Apache Storm foi analisado mais detalhadamente, devido às inúmeras possibilidades de criação de topologias e diferentes conjuntos de configurações. No gráfico à seguir, demonstramos a análise do impacto do tamanho do buffer de mensagens na performance geral da aplicação.

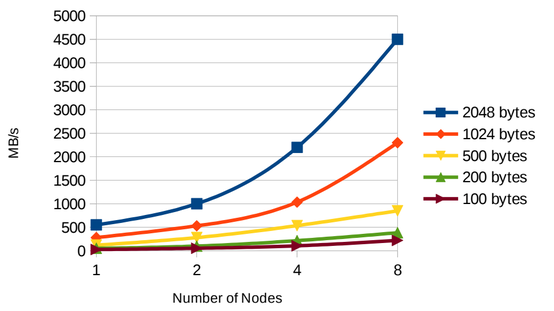
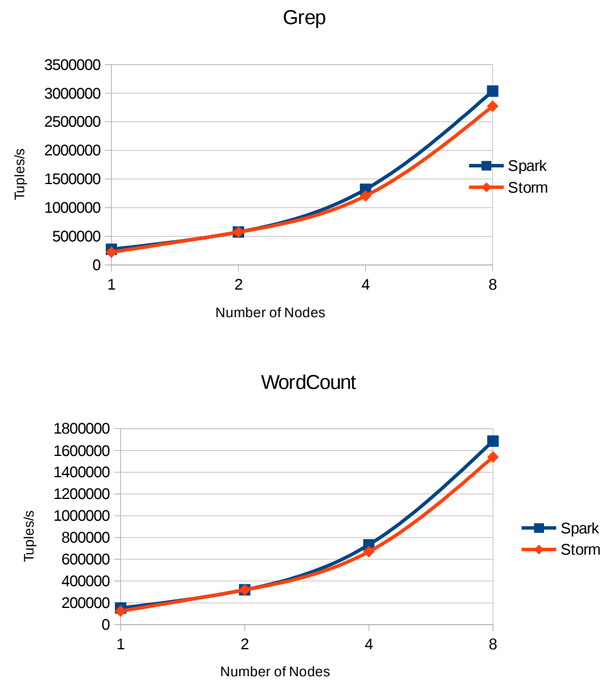


Figura 4 – Impacto do tamanho do buffer no Apache Storm

Após analisadas as possibilidades de configuração de ambas as ferramentas, foi realizado o desenvolvimento de aplicações para avaliar a performance, visando uma forma equivalente nas diferentes APIs de programação, para comparar as duas frameworks. Abaixo, seguem os resultados , que demonstram a proximidade dos resultados em ambas as plataformas.



Figuras 5 e 6 – Performance dos Micro-benchmarks no Storm e no Spark Streaming

Conclusões

Neste trabalho, analisamos a performance de aplicações de processamento distribuído de streams, através de micro-benchmarks, e pudemos perceber a proximidade da performance de ambas as ferramentas analisadas.

Através dessa análise, obtivemos uma melhor compreensão dos limites alcançáveis pelas plataformas atuais. Tais como, por exemplo, o throughput máximo de dados, o grau de escalabilidade e as características específicas de configuração de cada uma delas.

Através dessas análises, iremos guiar nossos trabalhos futuros, utilizando essas ferramentas para a implementação de novas aplicações distribuídas, bem como a análise mais profunda dos fatores que impactam na sua performance, tais como a latência e a tolerância à falhas.

GPPD Grupo de Processamento Paralelo e Distribuído
.inf INSTITUTO DE INFORMÁTICA UFRGS
UFRGS UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico