Aplicação de Ferramentas de *Big Data* para Alocação de Infraestruturas Virtuais

Guilherme Xavier[1], **Anderson Raugust**[2], **Guilherme Koslovski**[1 2]

[1]Ciência da Computação – UDESC

[2]Programa de Pós-graduação em Computação Aplicada – UDESC

{gxc.xavier,anderson.raug}@gmail.com, guilherme.koslovski@udesc.br

Resumo. *A alocação de recursos para hospedar infraestruturas virtuais é um desafio para provedores de nuvens computacionais, dada a natureza NP-difícil do problema. O presente trabalho propõe uma alternativa à essa problemática utilizando a ferramenta Spark GraphX, que trabalha de forma paralela por meio da heurística MapReduce.*

1. Introdução

São incontáveis os vídeos, imagens, músicas e textos produzidos e difundidos pelas redes a cada segundo. Tal quantidade massiva de dados possui uma variedade de mídias tão grande quanto seu próprio volume. O tamanho dos conjuntos de dados está além da capacidade de ferramentas típicas de banco de dados [Manyika et al. 2011]. Destarte, a grande questão estabelecida está em armazenar tal volume de dados, de modo que sua manipulação seja ágil e eficaz.

Considerando tais aspectos, a utilização de nuvens computacionais como dispositivo de armazenamento de grande volume de dados tornou-se uma solução promissora. As nuvens computacionais são definidas pelo NIST (*National Institute of Standards and Technology*) como um modelo para permitir acesso sob demanda a um conjunto compartilhado de recursos computacionais configuráveis (redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados com mínimo esforço gerencial ou interação com o provedor de serviços [Mell and Grance 2011]. Em suma, a computação em nuvem surgiu como paradigma para hospedar e entregar serviços pela Internet [Zhang et al. 2010]. Dentre os serviços entregues por provedores de nuvem, o presente trabalho foca na alocação de Infraestruturas Virtuais (IVs) [Oliveira and Koslovski 2016].

Devido ao elevado número de equipamentos que compõem um *data center*, aliado com os requisitos de provisionamento informados por clientes (qualidade de serviço, configuração de máquinas e redes virtuais, entre outros), a identificação de um subconjunto de recursos para hospedar uma IV é uma tarefa complexa [Fischer et al. 2013]. Sobretudo, a alocação de recursos físicos para hospedar IVs é um problema NP-difícil, o que torna a sua resolução computacionalmente custosa. Por essa razão, ela necessita de ferramentas que possam lhe proporcionar um funcionamento coeso e eficiente.

Sob esta perspectiva, o *framework Apache Spark* surgiu como uma ferramenta candidata. *Spark* é capaz de executar tarefas *MapReduce* diversas vezes, sequencial e paralelamente. O conceito fundamental utilizado pelo *MapReduce* é o de mapear uma base de dados, em uma coleção de pares *chave/valor*, para posteriormente reduzir os

pares de mesma chave na saída final do processamento. O *Spark* constrói sua efetividade no encadeamento de funções junto à conexão das respostas associadas, bem como na velocidade do carregamento dos arquivos a ser manipulados. Isso se deve ao *Spark* utilizar o máximo de memória possível para posteriormente recorrer ao disco.

O presente artigo tem como objetivo utilizar duas ferramentas manipuladoras de *Big Data*, *Spark GraphX* e consequentemente *MapReduce*, para resolver a alocação de IVs. O artigo está organizado da seguinte maneira: a Seção 2 descreve a alocação de IVs, enquanto a Seção 3 apresenta a solução em *Spark GraphX*. Uma análise dos dados é discutida na Seção 4. A Seção 5 é reservada às conclusões e perspectivas de continuidade.

2. Alocação de IVs

Provedores de nuvem IaaS (*Infrastructure as a Service*) podem hospedar IVs nos recursos ociosos de seus *data centers*. Cada cliente requisitante pode informar a configuração necessária para as máquinas virtuais que compõem a IV. Ainda, uma topologia de rede virtual pode ser especificada, informando requisitos de latência e largura de banda [Fischer et al. 2013].

A alocação de recursos físicos para hospedar IVs pode ser vista como um problema de mapeamento de grafos. O grafo representando a IV deve ser mapeado sobre o grafo representando o *data center*. Assim, vértices representam servidores e *switches*, enquanto arestas denotam os enlaces virtuais. A capacidade virtual solicitada e a configuração ociosa no *data center* são descritos por pesos em vértices e arestas [Oliveira and Koslovski 2016].

Durante a alocação, existem restrições que devem ser respeitadas. Ou seja, o grafo representando o *data center* deve ser capaz de suportar as capacidades solicitadas para os recursos virtuais. Por exemplo, os vértices do grafo físico devem suportar os requisitos mínimos de capacidade de processamento, memória e armazenamento exigidos pelos vértices do grafo virtual. De forma semelhante, os enlaces físicos de comunicação devem suportar os requisitos de qualidade de serviço exigidos pelo canal virtual. Devido as suas características, a alocação de IVs é um problema NP-difícil.

3. Solução Proposta

A proposta para alocar uma IV compreende uma incorporação fracionada dos elementos do grafo. Assim, em um grafo físico já determinado e modelado como uma base de dados *Spark*, uma requisição de IV é submetida e pesquisada. A requisição é decomposta e analisada em partes. Dentre as etapas do algoritmo (representação de dados, pesquisa, *merge* e seleção de resposta), a fase de pesquisa representa a busca pela alocação. Primeiramente avalia-se a existência de um vértice físico que suporte o virtual requerido. Quando encontrados mais de um vértice candidato, é realizado uma seleção baseada em *Best Fit* para escolher o vértice hospedeiro no grafo físico. A abordagem *Best Fit* foi previamente utilizada apresentando resultados satisfatórios [Ruck et al. 2014].

Posteriormente, uma verificação é realizada sobre os vértices associados ao vértice selecionado na etapa anterior. Neste ponto, a pesquisa verifica se as arestas que os unem, possuem capacidade suficiente para alocar as arestas associadas ao vértice virtual inicial. Por fim, é realizado um processo de *merge* com todos os subgrafos subsequentes do grafo físico inicial, dando forma ao grafo final que suporta requisição da IV.

Quanto à implementação, a solução foi implementada em *Scala* e as pesquisas por subgrafos foram detalhadas como requisições em uma base de dados. Assim, requisições individuais foram desenvolvidas para vértices e arestas.

4. Análise Experimental

A análise experimental foi realizada em uma nuvem computacional IaaS gerenciada pelo *framework* OpenStack (versão Juno). Foram criadas três máquinas virtuais (MVs) com 2GB de RAM e 1VCPU, 4GB de RAM e 2VCPUs e 8GB de RAM e 4VCPUs, todas executando o sistema operacional Ubuntu 16.04. As MVs executaram o *Apache Spark* versão 2.0.2 com a extensão *GraphX*.

Os grafos físicos foram compostos por 1000 vértices, interconectados aleatoriamente. Quanto as requisições de IVs, três cenários foram desenvolvidos, com 50, 100 e 200 vértices, aleatoriamente interconectados. É importante ressaltar que embora aleatórios, os grafos resultantes eram conexos. O peso das arestas e vértices seguiram uma distribuição uniforme entre 1 e 100. O tempo de processamento para alocar as IVs foi quantificado nas três MVs. A Figura 1 apresenta a média do tempo de execução com intervalo de confiança de 95%.

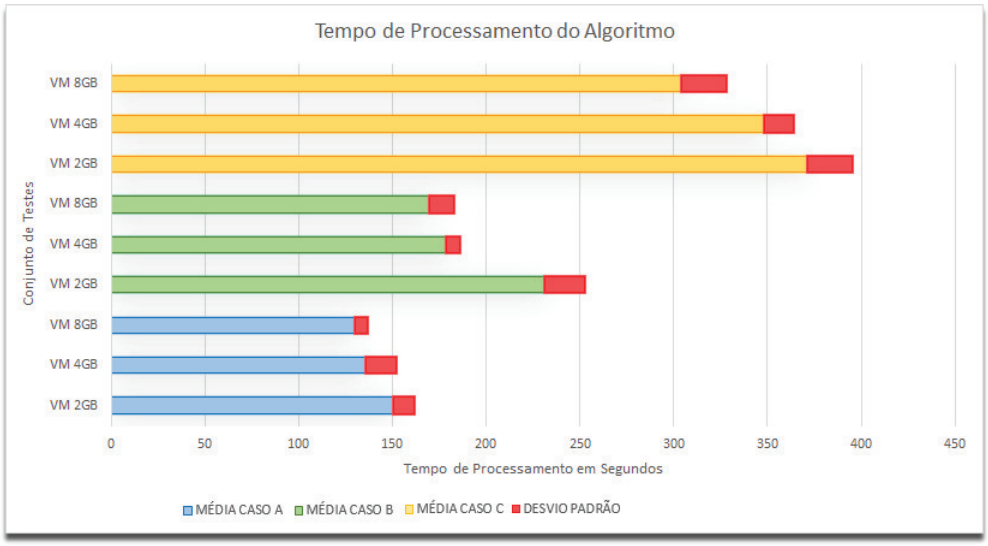

Figura 1. Tempo de Processamento Médio em Segundos

Na Figura 1 os grupos foram classificados em diferentes cores. O grupo A (Azul) é o conjunto de MVs que receberam um grafo virtual com 50 vértices. O grupo B (Verde) é o conjunto de MVs que analisaram uma requisição com 100 vértices. Por sua vez, o grupo C (Amarelo) processou um grafo virtual com 200 vértices. No gráfico também está apresentado em Vermelho o desvio padrão respectivo para àquela dada média gerada.

Uma análise revela que no primeiro caso de testes (grupo A) considerando a existência dos desvios, não há uma diferença significativa nas execuções das três diferentes máquinas. Ou seja, para uma requisição pequena, 50 vértices, não há uma modificação drástica no desempenho. É sensato afirmar isso tomando em conta que o processamento neste primeiro grupo não atingiu o máximo de capacidade de memória da menor máquina, assim o seu tempo foi categoricamente similar ao de outras MVs com mais recursos.

Aumentando-se a base de investigação, trabalhando com um grafo de 100 vértices, mudanças mais significativas ocorreram. Analisando novamente a MV de 2GB de RAM,

seu tempo teve um aumento considerável em relação ao outro caso de teste, o que faz, por exemplo, que sua marca não seja alcançada mesmo considerando os desvios existentes. Isso significa que um maior número de instruções já passou a afetar o processamento das MVs. Logo uma MV com pouca memória esforça-se mais que as outras para alcançar os mesmos resultados, corroborando assim os fundamentos do *Spark*, que usa primeiramente a memória em seu processamento até que seu máximo seja usufruído. É perceptível também que as MVs de maior memória, 4 e 8GB de RAM, continuaram a apresentar um processamento muito similar, o que revela a não influencia expressiva da quantidade de vértices sobre a taxa de memória.

Um exame no último conjunto amostral apresenta novamente os tempos de processamento mais próximos, nas três MVs. Contudo, pode-se constatar que as máquinas de 2 e 4GB de RAM tiveram, considerando os desvios, tempos muito próximos ao analisar uma base de 200 vértices, o que ratifica as afirmações apresentadas durante o embasamento teórico, de que o tempo de processamento do *Spark* está diretamente associado à memória disponível e à quantidade utilizada pelas instruções processadas. É factível averiguar também, que o tempo de processamento da MV de 8 GB na sua última análise, por mais que se aproxime no desvio do tempo da penúltima máquina do conjunto, foi influenciado diretamente pela maior porção de instruções a se analisar, mas ainda assim foi mais eficaz que as outras no alcance da meta final.

5. Considerações Finais

Os conjuntos de testes indicaram que ferramentas de *Big Data*, como o *Apache Spark*, podem ser aplicadas para resolver a alocação de Infraestruturas Virtuais. Quanto aos trabalhos futuros, a ferramenta proposta será analisada em um aglomerado de MVs executando *Apache Spark* para averiguar a escalabilidade de desempenho da solução proposta.

Agradecimentos O trabalho foi financiado pelos editais da UDESC e desenvolvido no LabP2D.

Referências

Fischer, A., Botero, J. F., Beck, M. T., de Meer, H., and Hesselbach, X. (2013). Virtual network embedding: A survey. *IEEE Communications Surveys Tutorials*, 15(4):1888–1906.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute*, pages 4–146.

Mell, P. M. and Grance, T. (2011). Sp 800-145. the nist definition of cloud computing. Technical report, NIST, Gaithersburg, MD, United States.

Oliveira, R. and Koslovski, G. P. (2016). A tree-based algorithm for virtual infrastructure allocation with joint virtual machine and network requirements. *International Journal of Network Management*, pages n/a–n/a. nem.1958.

Ruck, D. B., Oliveira, R., and Koslovski, G. P. (2014). Comparação de algoritmos para alocação de Infraestruturas Virtuais. *Revista Brasileira de Computação Aplicada*, 6(2):98–112.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *J Internet Serv Appl*, pages 7–18.