Caracterização do Desempenho de Aplicações Pipeline em Instâncias KVM e LXC de uma Nuvem CloudStack

Willian Baum[1], Carlos A. F. Maron[1,2], Dalvan Griebler[1,2], Claudio Schepke[3]

[1] Laboratório de Pesquisas Avançadas para Computação em Nuvem (LARCC)
Faculdade Três de Maio (SETREM) – Três de Maio – RS – Brasil

[2]Programa de Pós-Graduação em Ciência da Computação (PPGCC),
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS),
Grupo de Modelagem de Aplicações Paralelas (GMAP), Porto Alegre – RS – Brasil

[3]Universidade Federal do Pampa (UNIPAMPA),
Laboratório de Estudos Avançados (LEA), Alegrete – RS – Brasil

willianbaum@gmail.com,francocaam@gmail.com
dalvan.griebler@acad.pucrs.br,claudioschepke@unipampa.edu.br

Resumo. *Nuvens computacionais são uma alternativa para a computação de alto desempenho. Este artigo avalia o desempenho de aplicações estruturadas com o padrão pipeline em uma implantação de nuvem CloudStack com instâncias do tipo LXC e KVM. Foram testadas as aplicações Ferret e Dedup da suíte PARSEC, bem como constatado uma diferença significativa no Dedup. Na média geral, para esta aplicação a instância LXC é 40,19% melhor que a KVM.*

1. Introdução

Nuvens do tipo IaaS fornecem a infraestrutura necessária para gerenciar redes e máquinas virtuais [Buyya et al. 2013]. À medida que as tecnologias envolvidas sofrem constantes melhorias, as nuvens têm trazido maior consistência para os seus usuários. Essa evolução também tem garantido a sua característica principal, que é a flexibilidade de provisionamento de recursos computacionais sob demanda. O desafio mais recente é portar aplicações de alto desempenho para ambientes de nuvem sem comprometer o desempenho. Algumas pesquisas já demonstraram que tais aplicações sofrem pouca degradação quando portadas para uma nuvem privada, enquanto herdam os benefícios da computação em nuvem [Vogel et al. 2016a, Vogel et al. 2016b].

Neste trabalho, o objetivo foi implantar uma nuvem CloudStack com suporte a dois tipos de instâncias (KVM e LXC) para avaliar o desempenho de aplicações pipeline. A característica do paralelismo deste tipo de aplicação se assemelha com uma linha de produção, onde uma sequência de operações sobre um determinado fluxo contínuo de dados é aplicado. Além disso, algumas destas aplicações requerem uma baixa latência, enquanto outras necessitam de alta vazão. Assim, o estudo contribui para identificar a melhor forma de implantar uma nuvem ciente do desempenho para as aplicações pipeline Ferret e Dedup da suíte de *benchmarks* PARSEC.

Este trabalho está dividido em 5 seções. A Seção 2 discute os trabalhos relacionados. Na Seção 3 são descritas as ferramentas Dedup e Ferret. Na Seção 4 são mostrados os resultados obtidos. Por fim, na Seção 5, apresentamos a conclusão e os trabalhos futuros.

2. Trabalhos Relacionados

Em [Vogel et al. 2016b, Vogel et al. 2016a] foi avaliado o desempenho de implantações IaaS em ambientes de nuvem privada OpenNebula, OpenStack e CloudStack, usando

apenas um tipo de instância (KVM). Flexibilidade e resiliência das ferramentas foram as características consideradas. Para avaliar o desempenho, foram executadas as aplicações cientificas disponíveis na suíte NAS-NPB 3.3. Os resultados evidenciam pouca diferença no desempenho entre as instâncias na nuvem, apresentando baixa sobrecarga da virtualização e das ferramentas de nuvem. Em nosso trabalho, testamos aplicações pipeline em diferentes tipos de instâncias em uma nuvem CloudStack, uma vez que as cargas de trabalho influenciam apenas nas instâncias de acordo com os trabalhos relacionados.

Por outro lado, os trabalhos de [Iosup et al. 2011] e [Jayasinghe et al. 2014] realizaram uma análise do desempenho de cargas de trabalhos científicas em ambiente de nuvem IaaS utilizando diferentes *hypervisors*. Os trabalhos mostraram uma significativa variação de desempenho entre as aplicações, revelando um baixo desempenho na nuvem em comparação com infraestruturas GRID e PPI. Este trabalho, contrasta com os relacionados sobretudo ao utilizarmos as aplicações pipeline em instâncias KVM e LXC na nuvem CloudStack privada, o que leva à diferente caracterização do ambiente.

3. Aplicações Pipeline

Pipeline é uma estrutura algorítmica que se assemelha com uma linha de produção. Esta estrutura é composta por uma sequência de estágios que computam/realizam operações sobre cada um dos elementos que percorrem todos os estágios em um fluxo contínuo. O paralelismo é então obtido pela execução independente de diferentes operações, sendo que cada estágio computa sobre o elemento produzido pelo estágio anterior. As aplicações Dedup e Ferret do PARSEC[1] representam esse comportamento.

Dedup, ou *Data Deduplication*, é uma aplicação de compressão de dados que elimina dados redundantes em um certo conjunto de dados. É uma aplicação utilizada em sistemas de *backups*, pois reduz consideravelmente o tamanho dos dados processados, otimizando a utilização do armazenamento e também da largura de banda da rede.

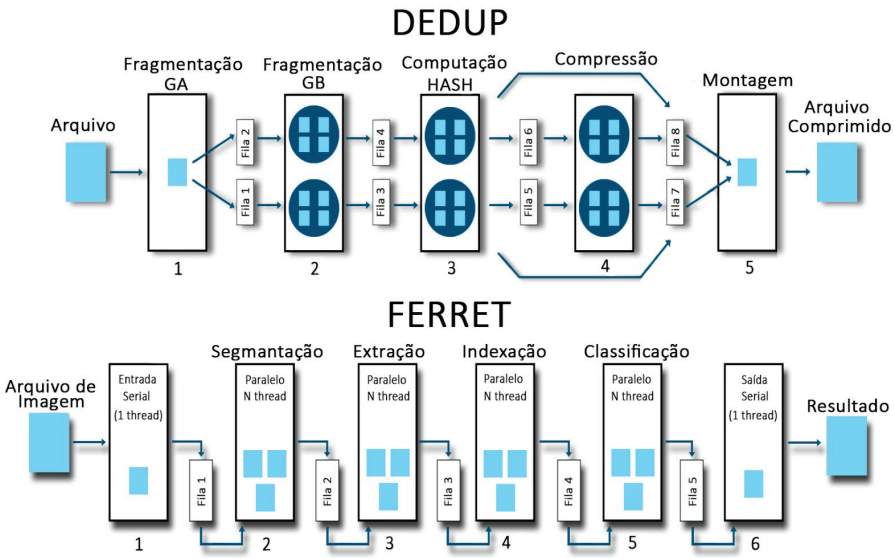


Figura 1. Dedup e Ferret.

Dedup é uma aplicação composta por 5 estágios (Figura 1). **Fragmentação com granularidade alta:** estágio sequencial que faz a leitura de *stream* de entrada quebrando em *chunks* de granularidade alta. **Fragmentação com granularidade baixa:** estágio paralelo que utiliza o algoritmo Rabin *fingerprint* para quebrar os elementos em *chunks*

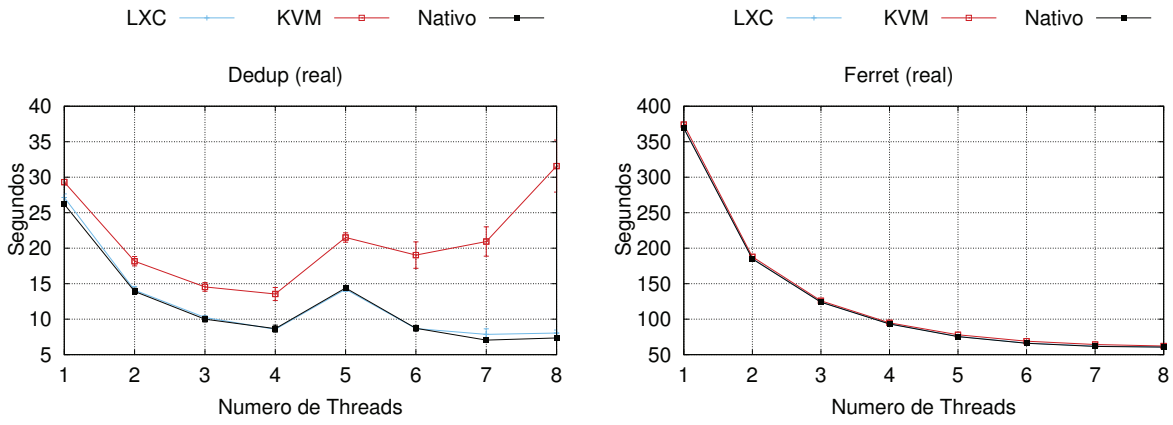
[1] http://parsec.cs.princeton.edu/

de baixa granularidade. **Computação Hash:** identifica os *chunks* através de uma chave HASH. **Compressão:** comprime os blocos de dados em paralelo utilizando o algoritmo Lempel-Ziv que constrói uma tabela *Hash* global que mapeia os valores dos dados. **Montagem do *stream* de saída:** reordena os blocos de dados e produz a saída já comprimida.

Ferret é uma aplicação que realiza a busca por similaridade de dados em áudio, imagens, vídeos e formas 3D. Ele é paralelizado utilizando um modelo de seis estágios (Figura 1). O primeiro e o último são sequenciais pois envolvem a entrada e saída dos dados. Nesta aplicação, é utilizado um processo de decomposição da imagem em segmentos. Cada área separada mostra diferentes objetos e permite atribuir um peso maior às partes das imagens que são consideradas relevantes para a busca (semelhantes) (estágio 2). Um vetor multi-dimensional armazena a descrição matemática dos conteúdos dos segmentos, propriedades de cor, forma e área da imagem (estágio 3). Assim que os vetores são completados, o estágio de indexação pesquisa a imagem no banco de dados para obter as imagens similares (estágio 4). Ao indexar as imagens similares, o estágio seguinte computa e ordena de acordo com um *ranking* calculado (estágio 5).

4. Resultados

Nesta seção são discutidos os resultados obtidos com a execução das aplicações Ferret e Dedup do PARSEC em uma implantação CloudStack (v.4.8.0.1) com uma instância KVM (v.2.0.0) e uma LXC (v.1.0.8). A média dos tempos de execução (10x) estão plotados nas Figuras 2(a) e 2(b), assim como o desvio padrão, que foi baixo (menos de 1%). O ambiente implantado utilizou Ubuntu Server 14.04 64bits em todos os testes, armazenamento distribuído com NFS (v.1.2.8-6) em uma rede 10/1000 Mb/s e instâncias KVM utilizando discos qcow2. As instâncias utilizaram a capacidade total dos nodos (24GB memória RAM e processador Intel Xeon X5560 2.80 GHz).



(a) Dedup (Armazenamento corporativo). (b) Ferret (Pesquisa de similaridades).

Figura 2. Tempo de Execução.

Os resultados das aplicações pipeline do PARSEC revelaram que Dedup exige uma alta demanda de E/S e uma grande sincronização entre os estágios. Com isso, o desempenho piora em uma instância KVM à medida que o número de *threads* aumenta, enquanto que na instância LXC o desempenho se assemelha ao apresentado na máquina nativa. O fluxo e o comportamento desta aplicação pode ser visto na Figura 1, onde operações em disco também podem ser adiantadas no terceiro estágio, ocasionando uma concorrência com o último. Assim, além de realizar a escrita em disco, é preciso fazer a reordenação dos dados processados. O sistema operacional implementa mecanismos de

releitura e reescrita utilizando memórias caches. Isto é, ao refazer estas operações em uma mesma região de disco feita anteriormente, o sistema operacional utiliza essas memórias para agilizar as operações. Utilizando o KVM, este desempenho não é totalmente portado ao sistema operacional em execução na nuvem, já que o KVM usa um formato específico de disco (qcow2) [Regola and Ducom 2010], enquanto LXC e nativo realizam essas operações ao nível do sistema operacional e de arquivos.

Na aplicação Ferret pouca diferença é percebida, mostrando que a característica do virtualizador não impacta na aplicação. A maior parte da carga de trabalho é realizada na CPU e memória, o que demonstra que ambos os tipos de instâncias se comportam bem em relação a esta característica de aplicação pipeline, atingindo desempenho semelhante ao ambiente nativo. Portanto, podemos concluir através dos experimentos que existe uma diferença de desempenho somente em aplicações pipeline E/S *bound*.

5. Conclusões

Este artigo apresentou uma avaliação de desempenho de aplicações pipeline utilizando dois tipos de instâncias implantadas em uma nuvem CloudStack privada. Com os resultados, conseguimos caracterizar que existem diferenças significativas em aplicações pipelines que exercem operações intensivas em disco. Enquanto que cargas intensivas em CPU não sofrem tal diferença e se assemelham com o ambiente nativo. Como trabalhos futuros, planeja-se: (I) Caracterizar o desempenho de outros tipos de instâncias na nuvem *CloudStack* com aplicações pipeline. (II) Ampliar o cenários de testes de aplicações pipeline, que possuem cargas sensíveis a rede, disco e memória. (III) Avaliar aplicações pipeline que executem em múltiplas instâncias na nuvem.

Referências

[Buyya et al. 2013] Buyya, R., Vecchiola, C., and Selvi, S. T. (2013). *Mastering cloud computing: foundations and applications programming*. Newnes.

[Iosup et al. 2011] Iosup, A., Ostermann, S., Yigitbasi, M. N., Prodan, R., Fahringer, T., and Epema, D. (2011). Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(6):931–945.

[Jayasinghe et al. 2014] Jayasinghe, D., Malkowski, S., Li, J., Wang, Q., Wang, Z., and Pu, C. (2014). Variations in performance and scalability: An experimental study in iaas clouds using multi-tier workloads. *IEEE Transactions on Services Computing*, 7(2):293–306.

[Regola and Ducom 2010] Regola, N. and Ducom, J. C. (2010). Recommendations for virtualization technologies in high performance computing. In *2010 IEEE 2º CloudCom*, pages 409–416.

[Vogel et al. 2016a] Vogel, A., Griebler, D., Maron, C. A. F., Schepke, C., and Fernandes, L. G. (2016a). Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack. In *24rd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 672–679, Heraklion Crete, Greece. IEEE.

[Vogel et al. 2016b] Vogel, A., Maron, C. A. F., Griebler, D., and Schepke, C. (2016b). Medindo o Desempenho de Implantações de OpenStack, CloudStack e OpenNebula em Aplicações Científicas. In *16th ERAD/RS*, pages 279–282, São Leopoldo, RS, Brazil. Sociedade Brasileira de Computação.