Uma análise do uso de *containers* para portabilidade de código para GPU na nuvem computacional

Jeferson Rech Brunetta[1], **Charles Boulhosa Rodamilans**[1,2], **Caian Benedicto**[1], **Edson Borin**[1]

[1]Centro de Estudos de Petróleo (CEPETRO)
Universidade Estadual de Campinas (UNICAMP)
Campinas – SP – Brasil

[2]Faculdade de Computação e Informática (FCI)
Universidade Presbiteriana Mackenzie
São Paulo – SP – Brasil

jeferdjex@gmail.com, charles.rodamilans@mackenzie.br,
caian@ggaunicamp.com, edson@ic.unicamp.br

Abstract. *Recent advances in virtualization technologies are enabling the execution of GPU code on cloud computing services. Moreover, container technologies are facilitating the development and migration of high-performance application to the cloud. In this work, we leverage the Singularity container to facilitate the migration of seismic processing GPU code and analyze its impact on performance. Our results indicate that the Singularity container does not add performance overhead to applications that were parallelized with CUDA, OpenCL and OpenMP.*

Resumo. *Avanços recentes em tecnologias de virtualização têm permitido a execução de aplicações de alto desempenho que fazem uso de GPUs em serviços de nuvem computacional. Além disso, avanços em tecnologias de* containers *têm facilitado o desenvolvimento e migração de aplicações de alto desempenho para execução na nuvem computacional. Neste trabalho, nós analizamos o uso do* container *Singularity para facilitar a migração de código de processamento geofísico que faz uso de GPUs para a nuvem computacional. Os resultados indicam que o uso do* container *Singularity não adiciona sobrecarga no desempenho de aplicações paralelizadas com OpenCL, CUDA e OpenMP.*

1. Introdução

O modelo de computação na nuvem permite que usuários façam uso de sistemas computacionais grandes e pague apenas pelo uso, sem terem que se preocupar com a aquisição, instalação e manutenção do sistema. Além disso, tecnologias de alta capacidade computacional (HPC) (e.g. GPUs e redes de interconexão de alta velocidade) estão sendo sendo entregues como serviços na nuvem computacional e popularizando o acesso a HPC.

Desenvolvedores de aplicações implementam o seu programa em um ambiente onde todas as dependências estão instaladas e, em muitos casos, enfrentam dificuldades para portar o seu programa para execução em outras máquinas. Uma possível solução para contornar este problema é a utilização da virtualização de CPUs. A virtualização

de CPUs com *hypervisors* (e.g. KVM e Xen) permite que toda a pilha de *software* do sistema, incluindo o sistema operacional, bibliotecas e aplicações, seja encapsulada em uma imagem de máquina virtual e executada em outros sistemas sem que haja problemas de compatibilidade de *software*. Dessa forma, os desenvolvedores podem criar a aplicação em uma máquina virtual local e depois migrar a máquina virtual para uma infraestrutura de nuvem, se beneficiando das características da nuvem anteriormente apresentada.

A inclusão do sistema operacional faz com que a imagem da máquina virtual fique muito grande, o que torna a transferência e replicação da imagem na nuvem um desafio à parte. Na virtualização com *containers*, por outro lado, o sistema operacional não é encapsulado junto com as bibliotecas e a aplicação, fazendo com que a pilha de *software* contida na imagem do *container* seja muito menor. Por não incluir o sistema operacional na pilha de *software*, a imagem do *container* tem que ser executada sobre um sistema operacional equivalente ao que foi utilizado para o desenvolvimento do *software*. Além disso, o *software* tem que ter sido compilado para a mesma arquitetura do conjunto de instruções da máquina alvo. Apesar destas restrições, a diversidade de sistemas operacionais e arquitetura do conjunto de instruções é pequena e os ganhos obtidos com a redução da imagem do *container* oferecem um bom custo benefício [Pahl 2015].

O laboratório High-Performance Geophysics (HPG), da Unicamp, têm utilizado o *container* Singularity para facilitar a instalação e migração de aplicações em diferentes sistemas computacionais. Neste trabalho, nós avaliamos o impacto do *container* Singularity sobre o desempenho de aplicações de processamento sísmico desenvolvidas para fazer uso de GPUs. Como principal resultado, verificou-se não haver diferença significativa de desempenho das aplicações de alto desempenho programadas com CUDA e OpenCL quando executadas sobre o Singularity.

2. Materiais e métodos

Os experimentos foram realizados em máquinas virtuais na nuvem computacional Azure, com as instâncias NC6 e NC6s. As instâncias NC6 possuem 6 vCPUs, 56 GB de memória, 380 GB de armazenamento SSD, 1 GPU da aceleradora NVIDIA Tesla K80 e custam U$ 1,084 por hora de uso. As instâncias NC6v2 possuem 6 vCPUs, 112 GB de memória, 336 GB de armazenamento SSD, 1 GPU da aceleradora NVIDIA Tesla P100 e custam U$ 2,493 por hora de uso. Utilizamos o sistema operacional Ubuntu Server 16.04 LTS 64-bit em todas as instâncias avaliadas.

Utilizamos as aplicações CRS e CMP, que implementam os métodos sísmicos Common Reflection Surface (CRS) e Common Midpoint (CMP), respectivamente [Mann et al. 1999], e ambas aplicações são CPU *bound* . Utilizamos três versões destas aplicações: a primeira paralelizada com CUDA, a segunda com OpenCL e a terceira com OpenMP (apenas CPU). As aplicações foram compiladas utilizando-se os drivers e compiladores CUDA 7.5, OpenCL 1.8 e gcc 5.2.1.

A métrica adotada para os experimentos foi o tempo de execução para cada versão paralela das aplicações. As execuções foram repetidas 3 vezes para cada versão e o resultado resumido com a aplicação da mediana.

Os algoritmos foram executados dentro do ambiente nativo da máquina virtual na nuvem e dentro do *container* Singularity [Kurtzer et al. 2017] versão 2.4, sendo este escolhido por possuir suporte para execução de código CUDA e OpenCL em GPUs.

3. Resultados e Discussões

A Tabela 1 mostra o tempo de execução das aplicações quando executadas diretamente na máquina virtual da nuvem computacional (Nativo) e sobre o *container* Singularity. Os aplicativos CMP e CRS acelerados com CUDA obtiveram um desempenho melhor em todos os experimentos quando comparados com a implementação utilizando OpenCL. Em nossa análise dos resultados observamos que, para o CMP, como o tempo de execução é muito pequeno, código de configuração do OpenCL (busca de plataforma, compilação do *kernel*, etc.) se torna significativo. De fato, se levarmos em consideração apenas o tempo de execução do *kernel* da aplicação, a diferença de desempenho entre CUDA e OpenCL se torna muito menor. Como o foco to trabalho está na análise do impacto que o Singularity tem sobre o desempenho das aplicações, nós deixamos esta análise e discussão para um trabalho futuro.

Tabela 1. Tempo de execução dos algoritmos na nuvem computacional (em segundos).

Instância (GPU)		NC6 (K80)		NC6v2 (P100)	
Ambiente		Nativo	Singularity	Nativo	Singularity
CMP	CUDA	0,1215	0,1299	0,0728	0,0720
	OPEN_CL	1,8583	1,8712	1,7601	1,7564
	OPEN_MP	0,4505	0,4524	0,4208	0,4128
CRS	CUDA	2,9602	2,9825	0,9481	0,9497
	OPEN_CL	8,3715	7,9770	3,1993	3,2324
	OPEN_MP	86,5156	87,1994	74,2885	72,8096

A Tabela 2 apresenta a relação do desempenho entre a execução das aplicações no *container* Singularity em nuvem e no ambiente nativo (sem *container*) da nuvem. A relação é dada pela divisão do tempo de execução com *container* pelo tempo de execução sem o *container*. Dessa forma, valores próximos de 1 indicam que o desempenho é semelhante enquanto que valores maiores que 1 indicam que o uso do *container* adicionou sobrecarga de desempenho na execução. Os resultados sugerem que há uma pequena perda de desempenho ($\sim$7%) quando a aplicação CMP paralelizada com CUDA é executada sobre o *container*, no entanto, uma análise dos tempos de execução (0,1345, 0,1215 e 0,1119 vs 0,1336, 0,1299 e 0,1254) indica que esta diferença é causada por variações na execução. Dessa forma, concluímos que o *container* não afetou de forma significativa o desempenho das aplicações aceleradas com CUDA, OpenCL ou OpenMP.

Tabela 2. Comparação ambiente Nativo e com *container* (Singularity/Nativo).

Instância (GPU)		NC6 (K80)	NC6v2 (P100)
CMP	CUDA	1,07	0,99
	OPEN_CL	1,01	1,00
	OPEN_MP	1,00	0,98
CRS	CUDA	1,01	1,00
	OPEN_CL	0,95	1,01
	OPEN_MP	1,01	0,98

A Tabela 3 mostra o desempenho de cada versão das aplicações normalizada pelo menor tempo. Como esperado, a instância (NC6v2) que possui a placa aceleradora NVIDIA Tesla P100 obteve os melhores resultados de desempenho, sendo de 1,06 até 3,15 vezes mais rápida do que a instância que possui a placa K80 (NC6). Observa-se, no entanto, que o custo financeiro do uso da instância NC6v2 (U$ 2,493 por hora) é aproximadamente 2,5 vezes maior do que o custo da instância NC6 (U$ 1,084 por hora). Neste contexto, a instância NC6 apresenta um custo benefício melhor para a execução da aplicação CMP enquanto que a instância NC6v2 tem um custo benefício melhor para a execução da aplicação CRS paralelizada com CUDA e OpenCL.

Tabela 3. Desempenho relativo entre as diferentes instâncias.

Instância (GPU)		NC6 (K80)		NC6v2 (P100)	
Ambiente		Nativo	Singularity	Nativo	Singularity
CMP	CUDA	1,69	1,80	1,01	1,00
	OPEN_CL	1,06	1,07	1,00	1,00
	OPEN_MP	1,09	1,10	1,02	1,00
CRS	CUDA	3,12	3,15	1,00	1,00
	OPEN_CL	2,62	2,49	1,00	1,01
	OPEN_MP	1,19	1,20	1,02	1,00

4. Conclusão

Neste trabalho realizou-se um estudo comparativo do desempenho de diferentes versões paralelas de aplicações de geofísica com e sem o uso do *container* Singularity na nuvem computacional. A principal contribuição foi demonstrar não haver diferença significativa de desempenho ao utilizar Singularity na nuvem, viabilizando a portabilidade de aplicações de alto desempenho. Como complemento, ressaltou-se que o desempenho obtido utilizando aceleradoras com maior capacidade de processamento pode não compensar o seu custo financeiro em algumas situações. Como trabalhos futuros, pretendemos investigar as diferenças de desempenho observadas em código OpenCL e CUDA, avaliar o desempenho do Singularity em ambientes não virtualizados e avaliar novas aplicações.

Agradecimentos

Este trabalho foi possível graças ao apoio da Petrobras, da Fapesp, da CAPES e da Microsoft. Os autores também agradecem às equipes do laboratórios *High Performance Geophysics* (HPG) e LMCAD pelo suporte técnico.

Referências

Kurtzer, G. M., Sochat, V., and Bauer, M. W. (2017). Singularity: Scientific containers for mobility of compute. *PloS one*, 12(5):e0177459.

Mann, J., Jäger, R., Müller, T., Höcht, G., and Hubral, P. (1999). Common-reflection-surface stack—a real data example. *Journal of applied geophysics*, 42(3-4):301–318.

Pahl, C. (2015). Containerization and the paas cloud. *IEEE Cloud Computing*, 2(3):24–31.