Escalonamento Dinâmico de Programas MPI Utilizando Migração de Máquinas Virtuais

Marcelo Veiga Neves e Nicolas Maillard
Programa de Pós-Graduação em Computação – Instituto de Informática – UFRGS
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil
{marcelo.veiga, nicolas}@inf.ufrgs.br

Resumo

MPI (Message Passing Interface) is a standard for inter-process communication on distributed-memory multiprocessor, widely used and accepted. However, the MPI standard does not define a way to schedule the processes. In this paper, we present an initial effort to allow dynamic reschedule of MPI processes via automatic decision-making and process migration. We also investigate the efficacy of using a virtualization tool for perform migration of MPI processes.

1. Introdução

A biblioteca MPI (*Message Passing Interface*) tornou-se, nos últimos anos, um padrão para programação com memória distribuída na área de alto desempenho. No entanto, a norma MPI por si só não especifica escalonamento de processos, ficando a cargo do programador realizá-lo preenchendo a ordem do arquivo de nós da aplicação[1]. Da mesma forma, MPI também não trata nativamente com o **re-escalonamento de processos** em tempo de execução para diferentes recursos. Para lidar com essas questões, foram desenvolvidas duas bibliotecas no GPPD (Grupo de Processamento Paralelo e Distribuído), uma para tratar o escalonamento estático de programas MPI[11] e outra para o dinâmico[6]. Essa última atua sobre a norma 2 de MPI e escolhe o destino de processos que são lançados de forma dinâmica através de novas diretivas dessa interface.

Uma das principais dificuldades de realizar re-escalonamento de programas MPI em tempo de execução, é a migração de processos que precisa ser realizada para trocar a localização física dos processos. Neste contexto, a **migração de máquinas virtuais**[7] surge como uma solução. A migração do sistema operacional inteiro, com toda a imagem da memória, evita os problemas clássicos que envolvem a migração em nível de processo, tais como a dependência residual e o re-estabelecimento de conexões[8]. Este tipo de migração pode ser usado, por exemplo, para permitir que processos acoplados a uma máquina virtual que executa sobre um recurso computacional sobrecarregado podem ser transferidos com ela para outro com menor sobrecarga em tempo de execução.

Neste contexto, pretende-se considerar a migração de máquinas virtuais para escrever uma biblioteca, como continuação do trabalho do grupo, que possibilite o re-mapeamento de processos em tempo de execução. A idéia é migrar processos demorados ou aproximar aqueles que possuam um padrão de comunicação elevado com o intuito de minimizar o tempo da aplicação. A melhoria de desempenho depende da complexidade da própria aplicação e do custo de realizar a migração de máquinas virtuais. Sendo assim, o este artigo apresenta uma análise do custo de migração de máquinas virtuais, e por conseqüência de processos, para diferentes recursos. Para tal, a ferramenta escolhida foi Xen [4]. Nessa análise foi testada a redistribuição de processos MPI entre os nós do agregado local nas aplicações paralelas HPL (*High Performance Linpack*) [1] e NPB (*NAS Parallel Benchmark*) [2]. Além disso, foram avaliados diferentes cenários de mapeamento para analisar a viabilidade de agrupar, em um mesmo nó, máquinas virtuais que comunicam mais.

O presente artigo é organizado em seções. Na Seção 2, apresenta-se Xen e como ele trata a migração de máquinas virtuais. A Seção 3 descreve a metodologia de testes usada e mostra os resultados obtidos com os testes. A Seção 4 descreve a conclusão, enfatizando as contribuições do artigo e relatando trabalhos futuros.

2. Xen e a Migração de Máquinas Virtuais

As técnicas de virtualização historicamente impõem um sobrecusto de desempenho devido as abstrações de *software* realizadas. Com o intuito de reverter esse panorama,

1 Normalmente, implementações MPI fazem escalonamento Round-Robin sobre a lista de nós do arquivo de máquinas da aplicação

pesquisas recentes conseguem reduzí-lo, como é o caso da **paravirtualização**[4]. A paravirtualização simplifica o processo de virtualização através da eliminação de funcionalidades específicas de *hardware* e instruções que são difíceis de serem virtualizadas com eficiência. Neste caso, é oferecida uma abstração de máquina virtual que é similar ao *hardware* da camada inferior, mas não idêntica. Em ambientes virtualizados o gerenciamento dos recursos é realizado por um monitor de máquinas virtuais, ou MMV. Dentre os monitores MMV existentes, o sistema Xen[4, 5] se destaca por permitir a migração de máquinas virtuais.

O sistema Xen é um MMV com código fonte aberto baseado na tecnologia de paravirtualização. A arquitetura de Xen é composta por dois elementos principais: (i) um monitor de máquinas virtuais propriamente dito (chamado de *hypervisor*); (ii) as máquinas virtuais que são controladas pelo monitor (chamadas de domínios Xen). O MMV abstrai a camada de *hardware* e provê acesso para os diferentes domínios. Um domínio especial é aquele chamado Domain0, o qual é capaz de acessar diretamente a interface de controle do MMV. Através desse acesso, é possível criar e gerenciar outros domínios Xen.

As máquinas virtuais Xen são independentes do *hardware* existente e, assim, podem ser encapsuladas e posteriormente migradas para outro computador. Nesse sentido, Xen implementa um mecanismo de migração chamado *live migration*[7] que permite a transferência de uma máquina virtual para um novo computador sem a interrupção da sua execução. Neste mecanismo, primeiramente é feito uma reserva de recursos no computador destino. Caso sejam atendidos alguns requisitos mínimos de compatibilidade, inicia-se a transferência da máquina virtual para um novo destino, de forma iterativa e sob demanda.

Diferentemente da migração em nível de processos, que é normalmente baseada em *checkpoint/restart*[8, 10], a migração de máquinas virtuais em Xen tende a apresentar tempo constante como será mostrado na Seção 3. Isto facilita a realização de previsões do custo de migrações para aplicações paralelas. Além disso, a migração de conexões é realizada através da técnica de *ARP reply* [9]. Nesta técnica, todos os computadores da rede Ethernet são informados que o IP da máquina virtual migrada foi modificado para informar um novo local. Conseqüentemente, todos os processos com conexões remotas podem seguir executando normalmente e todo o mecanismo de gerenciamento de conexões é tarefa do Xen.

3. Avaliação e Resultados

Esta seção apresenta os resultados obtidos com a avaliação de Xen para migração de processos MPI. Primeiramente, é feita uma avaliação da comunicação em diferentes cenários de mapeamento de processos MPI em máquinas virtuais, com o objetivo de analisar a viabilidade de agrupar processos que trocam muitas informações. Na seqüência, é apresentada uma análise do custo de migração de máquinas virtuais de Xen na execução de aplicações do tipo MPI. Há que se ressaltar que não é o objetivo deste trabalho fazer uma análise completa de desempenho de Xen, mas sim a viabilidade de sua utilização para escalonar processos MPI. Além disso, outros autores já se dedicam à avaliação do desempenho deste sistema, inclusive em programas MPI [5, 13].

Para a realização dos testes, foi utilizado um agregado homogêneo composto por máquinas bi-processadas Intel Pentium III a 1.1 GHz, cada qual com 512 Mbytes de memória principal e 256 Kbytes de memória *cache*. Os nós são interligados por uma rede com equipamentos Fast Ethernet. Todas as máquinas possuem o sistema operacional GNU/Linux na distribuição Debian Sarge 3.1 (*kernel* versão 2.6.18-4) com Xen 3.0.3 (*kernel* versão 2.6.18-4-xen).

3.1. Desempenho de Comunicação

A fim de avaliar a viabilidade de utilizar migração de máquinas virtuais para migrar processos MPI, foram realizados alguns testes de desempenho de rede com o *benchmark* NetPIPE[3] configurado para transmitir sobre MPI. O primeiro teste realizado foi uma comparação entre um sistema nativo e um sistema virtualizado com Xen (2 máquinas virtuais, uma em cada nó do agregado). Os resultados obtidos mostram que o sistema virtualizado possui um desempenho de rede bastante próximo ao do sistema nativo. Em termos de percentagem, a largura de banda alcançada no sistema virtualizado é em média de 1 a 2% menor, principalmente para mensagens com tamanho maior.

Outro teste realizado foi colocar duas máquinas virtuais em um mesmo nó do agregado e mapear um processo MPI em cada uma delas. Objetivo deste teste foi comparar o desempenho de dois processos MPI em um mesmo nó, tanto em um sistema nativo quanto virtualizado. Este cenário é comum em aplicações reais, já que o ambiente de testes é formado por nós bi-processados. Além disso, este teste permite avaliar se é viável utilizar migração de máquinas virtuais para agrupar, em um mesmo nó físico, processos que trocam um volume grande de dados. Analisando os resultado obtidos, é possível observar uma degradação no desempenho de comunicação quando coloca-se duas máquinas virtuais no mesmo nó do agregado. Por exemplo, quando transfere-se 6 Mbytes de dados, processos MPI no mesmo nó usando o sistema nativo atingiu uma largura de banda de aproximadamente 1200 Mbps, enquanto o sistema com duas máquinas virtuais (1 processo MPI em cada) no mesmo nó consegue 180 Mbps.

Acredita-se que a degradação de desempenho de comunicação do sistema virtualizado deve-se ao fato de to-

das as operações de rede das máquinas virtuais exigirem algum processamento. Assim, ocorre concorrência pelo uso da rede e o desempenho fica degradado. Uma prova dessa colocação foi que, ao desabilitar a verificação de somatório (*checksumming*) para controle de erro nas duas interfaces virtuais, o desempenho melhorou consideravelmente.

Outro cenário avaliado foi colocar 2 processos MPI na mesma máquina virtual de um nó. Nesse esquema, a comunicação entre os processos não sai de dentro da máquina virtual e o desempenho fica muito próximo ao do sistema nativo. Observando os resultados dessa subseção, é possível inferir que agrupar processos em um mesmo nó através do uso de máquinas virtuais pode não ser viável. No entanto, em se tratando de redes com diferentes velocidades, o uso migração para agrupar os processos em um mesmo nível de rede (mesma rede local) pode se tornar atrativo. Isso porque o tempo de comunicação de Xen em nós diferentes é bastante próximo ao do sistema nativo.

3.2. Desempenho da Migração de Máquinas Virtuais

Para avaliar o desempenho de execução e migração de máquinas virtuais em aplicações MPI, utilizou-se o *benchmark* com N=8000 (valor calculado para preencher a memória disponível nas máquinas virtuais). O primeiro teste com HPL utilizou 3 nós, cada qual com 2 máquinas virtuais. Assim, no total tem-se 6 processos MPI executando (um em cada máquina virtual). Os resultados mostram um tempo de execução, em média, 9% maior para o ambiente virtualizado quando comparado com a execução do sistema nativo. Em termos de desempenho, o sistema nativo atingiu 2.6 GFlops, enquanto o virtualizado ficou em 2.35 Gflops. Acredita-se que isto seja causado pelo problema de desempenho de rede descrito anteriormente na Seção 3.1.

Outro teste realizado foi colocar apenas uma máquina virtual por nó do agregado e cada uma delas executa somente um processo, totalizando uma aplicação com 3 processos MPI. A idéia é comparar o desempenho com um sistema nativo com essa mesma distribuição de processos. Desta vez, o desempenho de Xen ficou bastante próximo ao do ambiente nativo, apresentando um acréscimo de apenas 1,1% no tempo de execução da aplicação. No entanto, é importante salientar que esta configuração não explora todos os recursos disponíveis, já que o nós utilizados são biprocessados.

Antes de realizar os testes de migração com a aplicação HPL, mediu-se o tempo de migração de uma máquina virtual ociosa (que executa somente os processos do sistema operacional). O tempo total de sua migração foi em torno de 30 segundos. Este tempo mostrou-se constante uma vez que todas as máquinas virtuais possuem uma memória virtual de mesmo tamanho. Em adição, pelo fato que as máquinas estarem com pouca atividade, a migração ocorre em poucos turnos (iterações).

O próximo passo foi executar a aplicação HPL em 3 máquinas virtuais, uma em cada nó do agregado, e migrar uma delas para um quarto nó utilizando o comando de *live migration* de Xen. Com a realização de uma migração, houve um acréscimo de 28,4% no tempo de execução de HPL. Verificou-se também que o tempo de migração da máquina virtual durante a execução de HPL passou de 30 segundos (no caso da máquina ociosa) para 104 segundos. A causa deste aumento é que, por HPL realizar muitas modificações na memória, Xen atingiu o número limite de iterações para realizar a migração.

Para verificar se o impacto de uma migração de máquina virtual de Xen em uma aplicação mais demorada é menor que aquele obtido com HPL, optou-se por utilizar a aplicação SP do pacote NPB. Foi utilizado um problema de classe B para o programa NPB e usados 4 nós do agregado, cada um com uma máquina virtual. Observando o gráfico da Figura 1 percebe-se que o aumento no tempo de execução para o programa SP utilizando Xen foi de aproximadamente 4%. Já a migração de uma máquina virtual para um quinto nó representa um aumento de aproximadamente 10%. O gráfico da Figura 1 mostra que o tempo de execução da aplicação com uma migração no ambiente virtualizado foi de 1409 segundos. O tempo no ambiente nativo sem migração foi de 1267 segundos, 142 segundos a menos que o anterior. Com base nestes resultados, espera-se que em aplicações que executam por um longo período de tempo (por exemplo, algumas horas como é o caso da previsão de tempo [12]), o custo de migração passa a se tornar ínfimo.

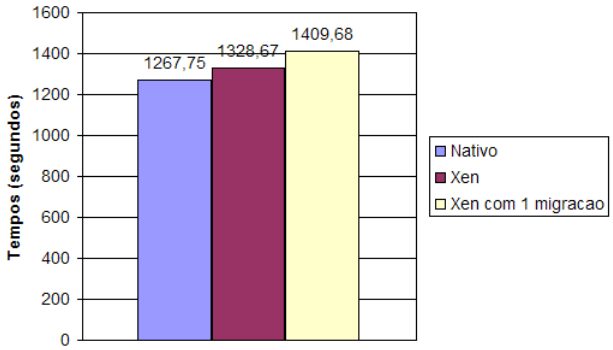

Figura 1. Custo de uma migração na aplicação SP do programa NPB

4. Conclusão

O presente artigo apresentou o andamento de um trabalho que visa permitir escalonamento dinâmico de processos MPI. Na etapa atual do trabalho, realizou-se uma análise do impacto da migração de máquinas virtuais de Xen sobre a execução de aplicações do tipo MPI. Os resultados mostraram que o sobrecusto da migração de uma máquina virtual de Xen numa aplicação MPI é praticamente constante, pois a cada migração é transferida toda a imagem da memória entre dois nós do agregado. Portanto, o custo de uma migração é menos perceptível numa aplicação que tenha um longo tempo de execução. Em contra-partida, em aplicações que executam rapidamente, a migração pode se tornar inviável, como foi visto na Seção 3. Uma migração na aplicação HPL aumentou a sua execução em 28%, enquanto que na NPB essa sobrecarga foi de aproximadamente 10%. De maneira geral, conclui-se que o sucesso do emprego de migração para o re-escalonamento de processos depende do tempo de execução da aplicação MPI.

Além de migração, esse artigo mostrou testes de comunicação através dos quais foi possível observar uma degradação de desempenho de rede obtida quando colaca-se duas máquinas virtuais no mesmo nó físico. Essa degradação pode tornar inviável a estratégia de agrupar, em um mesmo nó, os processos que comunicam um grande volume de dados.

A próxima etapa da pesquisa é estender a biblioteca de escalonamento dinâmico desenvolvida no grupo GPPD para suportar o re-escalonamento de processos MPI usando migração de máquinas virtuais de Xen. Para isso, o Xen apresenta uma interface de programação baseada em XML-RPC para expressar a migração que será usada na confecção da biblioteca de escalonamento. Dessa forma, a migração deixaria de ser em linha de comando para partir da própria execução da aplicação (ligada com a nova biblioteca de escalonamento). Além disso, outro trabalho futuro inclui a procura de aplicações MPI intensivas que demandam horas para a sua execução, Nesse sentido, a aplicação MPI de previsão do tempo BRAMS [12] será estudada.

Referências

[1] High-Performance Linpack. http://www.netlib.org/ benchmark/hpl/. Acessado em julho de 2007.

[2] NAS Parallel Benchmarks. http://www.nas.nasa.gov/ Software/NPB/. Acessado em julho de 2007.

[3] NetPIPE: A Network Protocol Independent Performace Evaluator. http://www.scl.ameslab.gov/ netpipe/. Acessado em julho de 2007.

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, volume 37, 5 of *Operating Systems Review*, pages 164–177, New York, 2003. ACM Press.

[5] M. P. Boufleur, G. P. Koslovski, and A. S. Charão. Avaliação do uso de xen em ambientes de alto desempenho. In *Workshop em Sistemas Computacionais de Alto Desempenho - WSCAD 2006*, pages 141–147, Ouro Preto - MG, 2006.

[6] M. C. Cera, G. P. Pezzi, E. N. Mathias, N. Maillard, and P. O. A. Navaux. Improving the dynamic creation of processes in mpi-2. In *Lecture Notes in Computer Science - 13th European PVMMPI Users Group Meeting*, volume 4192/2006, pages 247–255, Bonn, Germany, 2006. Springer Berlin / Heidelberg.

[7] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation*, pages 20–34, Boston, MA, USA, May 2005.

[8] D. S. Milojicic, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou. Process migration. *ACM Computing Surveys*, Sept. 2000.

[9] D. Plummer. An ethernet address resolution protocol. *RFC 826*, Nov. 1982.

[10] S. Sankaran, J. M. Squyres, B. Barrett, A. Lumsdaine, J. Duell, P. Hargrove, and E. Roman. The LAM/MPI checkpoint/restart framework: System-initiated checkpointing. *International Journal of High Performance Computing Applications*, 19(4):479, Winter 2005.

[11] R. E. Silva, G. Pezzi, N. Maillard, and T. Diverio. Automatic data-flow graph generation of mpi programs. In *SBAC-PAD '05: Proceedings of the 17th International Symposium on Computer Architecture on High Performance Computing*, pages 93–100, Washington, DC, USA, 2005. IEEE Computer Society.

[12] R. P. Souto, R. B. Avila, P. O. A. Navaux, M. X. Py, T. A. Diverio, H. F. C. Velho, S. Stephany, A. J. Preto, J. Panetta, E. R. Rodrigues, E. S. Almeida, P. L. S. Dias, and A. W. Gandu. Processing mesoscale climatology in a grid environment. In *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, pages 363–370, Washington, DC, USA, 2007. IEEE Computer Society.

[13] L. Youseff, R. Wolski, B. C. Gorda, and C. Krintz. Paravirtualization for HPC systems. In G. Min, B. D. Martino, L. T. Yang, M. Guo, and G. Rünger, editors, *ISPA Workshops*, volume 4331. Springer, 2006.