Impacto da Migração de Máquinas Virtuais de Xen na Execução de Programas MPI*

Marcelo Veiga Neves, Rodrigo da Rosa Righi, Nicolas Maillard e Philippe O. A. navaux
Programa de Pós-Graduação em Computação – Instituto de Informática – UFRGS
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil
{marcelo.veiga, rodrigo.righi, nicolas, navaux}@inf.ufrgs.br

Resumo

A virtualização pode ser vista como a manutenção de máquinas virtuais, cada qual com seu sistema operacional, sobre um conjunto de recursos. Para área de alto desempenho, ela possibilita criar várias máquinas virtuais em cada recurso, permitindo testes de desempenho de aplicações científicas com diversas configurações. Além disso, o conceito de virtualização abrange a idéia de migração, onde é possível transportar uma máquina virtual para um diferente recurso. Nesse contexto, a ferramenta Xen suporta as características apresentadas e foi usada para avaliar o custo da migração de máquinas virtuais que executam programas MPI. O propósito final é avaliar a viabilidade de Xen para o desenvolvimento de uma biblioteca de migração de processos MPI, tornando possível um re-escalonamento eficiente deles em tempo de execução. O presente artigo apresenta os resultados sobre o impacto da migração de máquinas virtuais Xen na execução de programas MPI. Ele também aborda alguns testes de computação e comunicação com o Xen e trabalhos relacionados sobre migração com virtualização.

1. Introdução

A virtualização de recursos computacionais é um tema aplicado em várias áreas da ciência da computação, como tolerância a falhas[14], alto desempenho[4] e armazenamento[7]. Nesse sentido, a utilização de máquinas virtuais permite a execução de diversas instâncias de sistemas operacionais sobre uma mesma arquitetura de *hardware* de forma transparente e isolada. No entanto, as técnicas de virtualização historicamente impõem um sobrecusto de desempenho devido as abstrações de *software* realizadas[10]. Com o intuito de reverter esse panorama, pesquisas recentes conseguem reduzí-lo, como é o caso da **paravirtualização**[3]. 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. Um dos sistemas que possibilita a paravirtualização é o Xen[3].

Em ambientes de alto desempenho, a virtualização possibilita que um nó de um agregado (*cluster*) possa manter vários sistemas operacionais em uso. Por conseqüência, um agregado com virtualização pode oferecer para a aplicação uma quantidade maior de pontos finais para a sua execução e diferentes variações de configuração. Além disso, o isolamento entre as máquinas virtuais permite o uso compartilhado de nós de um agregado. Por exemplo, é possível criar uma máquina virtual para cada processador de um nó SMP (*Symmetric Multi-Processor*) do agregado, onde cada máquina dessas teria uma finalidade específica e poderia ser disponibilizada para um conjunto específico de usuários.

Uma outra vantagem da utilização de virtualização na área de alto desempenho é a capacidade de **migração de máquinas virtuais**[6, 14]. 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[13]. Além da facilidade de uso, a migração de máquinas virtuais torna possível o controle dinâmico de recursos em nível administrativo, balanceamento de carga entre recursos e o re-escalonamento de processos em aplicações paralelas. Nessa última questão, processos acoplados a uma máquina virtual que executa sobre um recurso computacional sobrecarregado podem ser migrados com ela para outro com menor sobrecarga em tempo de execução. Dessa forma, o re-escalonamento de processos transferindo máquinas virtuais pode contribuir para diminuir o tempo de duração de uma aplicação ou para melhorar o balanceamento de cargas entre os recursos.

Nesse contexto, esse artigo aborda o tema da migração de máquinas virtuais que executam aplicações científicas escritas com a interface MPI (*Message Passing Interface*),

*Pesquisa parcialmente financiada pelos Órgãos Capes e CNPq

por ser um padrão para programação com memória distribuída na área de alto desempenho. 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 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[19] e outra para o dinâmico[5]. 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. Essas bibliotecas reforçam o interesse na interface MPI e em escalonamento de processos.

Analisando as técnicas da paravirtualização (virtualização com melhor desempenho) e migração de máquinas virtuais, pretende-se considerar a migração de processos 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 presente 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. Nessa análise foi testada a redistribuição de processos MPI entre os nós do agregado local nas aplicações paralelas HPL (*High Performance Linpack*) [16] e NPB (*NAS Parallel Benchmark*) [2]. Além disso, foram usados *micro-benchmarks* para avaliar a sobrecarga imposta por Xen tanto em aplicações que realizam bastante processamento (*CPU-bound*) quanto comunicação (*IO-bound*). No caso de comunicação, foram avaliados diferentes cenários para analisar a viabilidade de agrupar máquinas virtuais que comunicam mais em um mesmo nó.

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 apresenta alguns trabalhos que usam o sistema Xen. A Seção 4 descreve a metodologia de testes usada e a Seção 5 mostra os resultados obtidos com os testes. A Seção 6 descreve a conclusão, enfatizando as contribuições do artigo e relatando trabalhos futuros.

2. Xen e a Migração de Máquinas Virtuais

Em ambientes virtualizados o gerenciamento dos recursos é realizado por um monitor de máquinas virtuais, ou MMV. Dentre os monitores MMV existentes[1], o sistema Xen[3, 4] se destaca por permitir a migração de máquinas virtuais como está ilustrado na Figura 1. Essa figura mostra o caso de um agregado no qual os nós executam máquinas virtuais, onde a máquina B é migrada do nó n1 para o nó n2 do agregado. No procedimento de migração, todos os processos residentes na máquina virtual alvo são levados até o rescurso destino e seguem sua execução normalmente (sem uma reinicialização). No contexto deste trabalho, essa migração permite o re-escalonamento de processos que compõem um programa do tipo MPI. Por exemplo, a Figura 1 pode representar uma aplicação MPI com 6 processos, sendo que um deles é migrado com a máquina virtual B para um recurso (nó n2) com menor carga no momento.

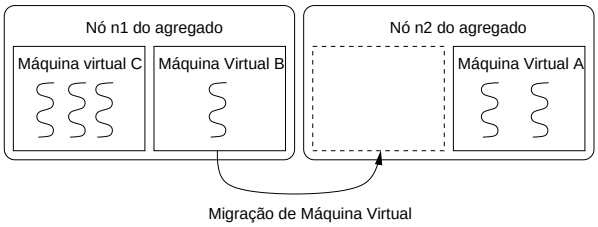

Figura 1. Migração de máquina virtual entre nós de um agregado.

O sistema Xen é um monitor 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*[6] que permite a transferência de uma máquina virtual para um novo computador sem a interrupção da sua execução. Neste mecanismo, primeiramente é feita 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.

A abordagem de migração usada por Xen é chamada de pré-cópia (*pre-copy*). Essa abordagem combina as estratégias de uma fase iterativa de cópia sob demanda (*iterative push*) com uma fase curta de parada-e-cópia (*stop and copy*)[6]. O termo iterativo significa que a pré-cópia ocorre em turnos, sendo que no primeiro deles todas as páginas

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

de memória são transferidas. Logo após, as páginas transferidas no turno n são aquelas que foram modificadas no turno $(n-1)$. Normalmente, as máquinas virtuais possuem um conjunto de páginas que são modificadas freqüentemente e que podem fazer com que o número de turnos seja muito grande. Assim, é limitado o número de turnos de pré-cópia, baseado numa análise do comportamento do conjunto de páginas frequentemente modificadas (WWS - *Writable Working Set*) da máquina virtual. No caso de aplicações de alto desempenho, é de se esperar que o WWS sejam grandes devido sua natureza de computação intensiva, o que pode aumentar o tempo para a migração[6].

Diferentemente da migração em nível de processos, que é normalmente baseada em *checkpoint/restart*[13, 18], a migração de máquinas virutais em Xen tende a apresentar tempo constante como será mostrado na Seção 5. Isto facilita a realização de previsões do custo de migrações para aplicações paralelas. É importante ressaltar também que o Xen não possui mecanismo para migração de sistema de arquivos. Portanto, quando uma máquina virtual é migrada é necessário que o sistema de arquivo esteja disponível no computador destino. Neste caso, normalmente utiliza-se um sistema de arquivos distribuído como é o caso do NFS. Já a migração de conexões é realizada através do envío de respostas ARP não solicitadas (*ARP reply*) [17]. 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. Trabalhos Relacionados

Huang et al.[11] apresentam um arcabouço para trabalhar com máquinas virtuais de Xen e aplicações de alto desempenho. Eles atacam a questão da comunicação entre máquinas virtuais através da exploração do sobrepasso do sistema operacional (*OS-bypass*) e do uso de redes de alta velocidade Infiniband. Pan et al.[15] trabalham com programas MPI sobre máquinas virtuais distribuídas pela Internet. Assim como o trabalho anterior, o seu foco é na comunicação. Eles propuseram uma técnica de virtualização de soquete para reduzir a latência de rede na interação entre máquinas virtuais. Com essa técnica, o tempo de comunicação no ambiente virtualizado ficou próximo daquele obtido na rede física.

Boufleur et al.[4] apresentaram resultados experimentais sobre o custo de utilização de Xen em ambientes de alto desempenho. Neste artigo, foram realizadas migrações de processos de uma aplicação MPI residentes em diferentes máquinas virtuais. Entretanto, os autores desse artigo focam sua análise de migrações no comportamento das transferências de dados segundo o protocolo de migração baseado em iterações de Xen[6]. Explorando a união de Xen e MPI, também é possível citar o trabalho desenvolvido por Youseff et al. [23]. Esses autores realizam uma análise completa do custo de execução de Xen sobre programas MPI. Contudo, o mecanismo de migração não é explorado no seu artigo.

Nagarajan et al.[14] apresentam uma solução de tolerância a falhas proativa utilizando a migração de máquinas virutais disponibilizadas por Xen. Esta solução também inclui um sistema de balanceamento de carga que intergra Xen com o sitema de monitoramento de agregados Ganglia[12]. Neste caso, os dados de monitoramento de Ganglia são usados para selecionar um nó destino que esteja com a menor carga de processamento. O artigo também inclui uma avaliação do custo de migração de programas MPI. No entanto, o seu foco permenace na área de tolerância a falhas.

Analisando o estado da arte, nota-se a carência por uma avaliação que quantifique o custo de migração de máquinas virutuais com o objetivo de inferir sobre a viabilidade de usar essa técnica para re-escalonamento de processos MPI. Assim, o diferencial do presente artigo é mostrar o impacto da migração de máquinas virtuais de Xen sobre diferentes aplicações MPI. Em adição, esse artigo apresenta alguns testes experimentais de programas que realizam bastante computação e comunicação em sistemas nativo e virtualizado. Esses testes são relevantes para mensurar o desempenho de Xen em tarefas tradicionais de aplicações paralelas que visam alto desempenho.

Na parte de comunicação, o presente artigo faz medições da largura de banda obtida na comunicação entre processos MPI com diferentes mapeamento deles em máquinas virtuais. Estas medições são essencias para analisar a viabilidade de aproximar processos que trocam muitas informações, através da migração das máquinas virtuais, e reduzir o volume de dados trafegados pela rede.

4. Metodologia de Avaliação

Nesta seção é apresentada a metodologia para a avaliação de desempenho de programas MPI que executam em máquinas virtuais de Xen. Para a realização dos testes, foi utilizado um agregado homogêneo (Corisco do Instituto de Informática da UFRGS) 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).

Os primeiros testes realizados foram de processa-

mento (computação) e de comunicação utilizando *micro-benchmarks* específicos para este fim. O próximo passo foi a avaliação do sobrecusto da migração de máquinas virtuais de Xen sobre aplicações paralelas do tipo MPI. Com o intuito de avaliar o desempenho de computação de Xen e compará-lo com um *kernel* não virtualizado, utilizou-se o *benchmark* Linpack[8]. Linpack realiza uma decomposição LU em duas fases, fatoração e *back-solve*, e reporta o tempo gasto em cada uma delas. Para cada um dos testes com Linpack, foi realizada uma média aritmética de 50 execuções. Já a avaliação da troca de mensagem em programas MPI sobre máquinas virtuais Xen foi realizada com o auxílio do *benchmark* NetPIPE[20]. NetPIPE é um programa que realiza testes de ping-pong entre dois processos MPI e mede o desempenho das comunicações. As medições são realizadas variando o tamanho da mensagem de 0 até 8 MBytes. Foram realizadas 300 medições para cada teste de comunicação.

Foram escolhidas duas aplicações para avaliar a sobrecarga da migração de máquinas virtuais de Xen em programas MPI. A primeira delas é o *benchmark* HPL (*High Performance Linpack*) [16]. HPL é uma aplicação que resolve sistemas lineares densos em dupla precisão e é comumente utilizado para avaliação de desempenho de computadores paralelos como um agregado[2]. O tamanho da matriz de entrada utilizada foi de N=8000 (tamanho calculado para utilizar toda a memória disponível nas máquinas virtuais). A outra aplicação utilizada foi o *benchmark* NPB (*NAS Parallel Benchmark*)[2]. NPB é conjunto de 7 sub-programas que realizam operações críticas em ambientes paralelos. Dentre eles optou-se pelo SP (Pentadiagonal solver) por apresentar um longo tempo de execução e, para o tamanho de problema, optou-se pela classe B por utilizar toda a memória das máquinas virtuais.

A metodologia usada para realizar a migração de máquinas virtuais foi a seguinte. O mecanismo de migração foi agendado para iniciar exatamente 60 segundos após o lançamento da aplicação MPI. Ou seja, a migração não parte da própria aplicação paralela. A migração é disparada de linha de comando que executa: "xm migrate –live". Tal comando ainda recebe como parâmetros a máquina virtual alvo e o seu nó destino no agregado. Foram realizados 30 testes na parte da migração e os tempos apresentados representam uma média aritmética.

5. Avaliação e Resultados

Esta seção apresenta os resultados obtidos com a avaliação de Xen. Primeiramente é observado o impacto de Xen nas partes de computação e comunicação. No caso de comunicação, é feita uma avaliação da largura de banda para diferentes esquemas de mapeamento de processos MPI em máquinas virtuais. Na seqüência, é apresentada uma análise do custo de migração de máquinas virtuais de Xen na execução de aplicações do tipo MPI.

5.1. Desempenho de Computação

Como mencionado na seção anterior, utilizou-se o *benchmark* Linpack para avaliar o sobrecusto imposto por Xen na execução de um programa de computação intensiva. O gráfico da Figura 2 contém os tempos de execução de Linpack com uma matriz de tamanho 3000x3000 com valores de entrada em dupla precisão. Além dos tempos informados pelo próprio Linpack, também utilizou-se os tempos coletados pelo comando time do sistema Linux.

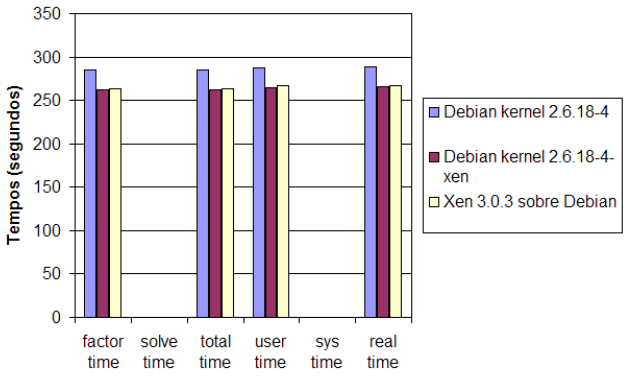

Figura 2. Desempenho na execução do *benchmark* Linpack com matriz 3000x3000.

Os resultados do gráfico da Figura 2 mostram que, apesar das modificações para prover a virtualização, o *kernel* Xen apresenta um desempenho superior ao *kernel* nativo que acompanha a distribuição Debian. Este resultado deve-se ao escalonador de processos do kernel virtualizado, que implementa de forma eficiente o escalonador BTV (*Borrowed Virtual Time*) [9]. Continuando a análise no gráfico, pode-se notar que o Domain0 (representado no gráfico por Debian *kernel* 2.6.18-4-xen) apresenta um desempenho melhor que o das máquinas virtuais Xen (Xen 3.0.3 sobre Debian). Acredita-se que a causa disto seja o fato de Domain0 ter maiores privilégios que os outros domínios Xen.

5.2. Desempenho de Comunicação

O gráfico da Figura 3 contém uma comparação entre a largura de banda obtida na comunicação de dois processos MPI que executam NetPIPE em dois ambientes. O primeiro ambiente é um sistema nativo e o outro um sistema virtualizado com Xen (2 máquinas virtuais, uma em cada nó do

[2]HPL também é usado no sítio que avalia as 500 máquinas mais poderosas do mundo: www.top500.org

agregado). Os resultados obtidos no gráfico da Figura 3 mostram que o sistema virtualizado possui um desempenho de rede bastante próximo ao do sistema nativo. Em termos de percentagem, os tempos do sistema virtualizado são em média de 1 a 2% menores, principalmente para mensagens com tamanho maior.

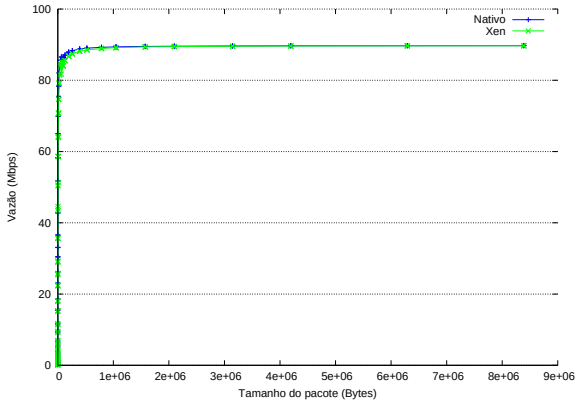

Figura 3. Largura de banda do NetPIPE usando 2 nós do agregado.

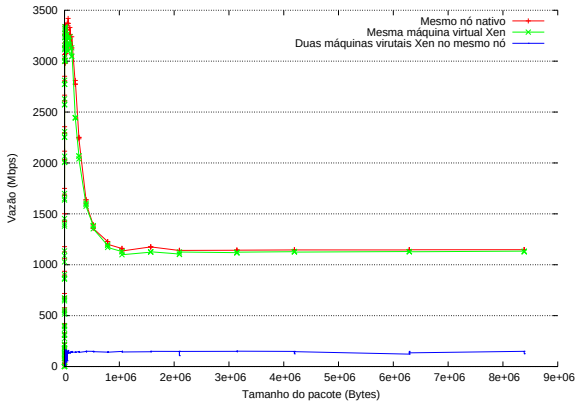

Figura 4. Largura de banda do NetPIPE usando um nó do agregado.

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. O gráfico da Figura 4 apresenta os resultados obtidos nesta comparação. Analisando este gráfico, 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 atingem 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 do sistema virtualizado deve-se ao fato de todas 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.

Outra linha mostrada no gráfico da Figura 4 é aquela onde são colocados 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 como pode ser visto na Figura 4. 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 de 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 (conforme mostrou o gráfico da Figura 3).

5.3. Desempenho com a Aplicação HPL

O gráfico da Figura 5 apresenta os resultados da execução de HPL em 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 5.2.

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. Os resultados são apresentados no gráfico da Figura 6. 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 to-

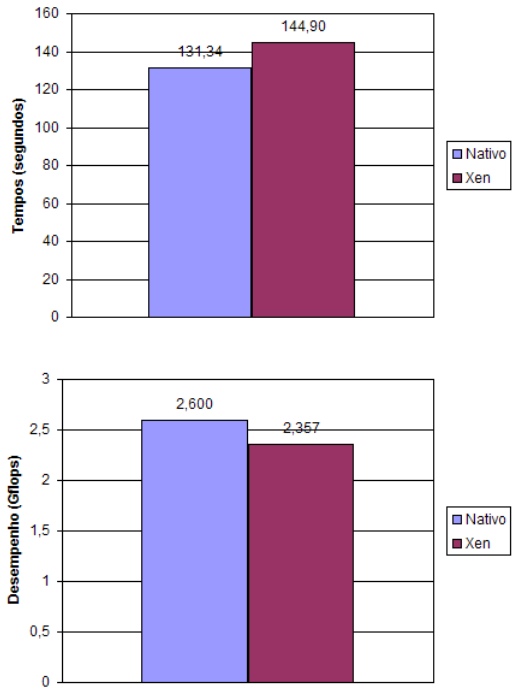

Figura 5. Comparação de desempenho de HPL com 6 máquinas virtuais e 1 processo por máquina.

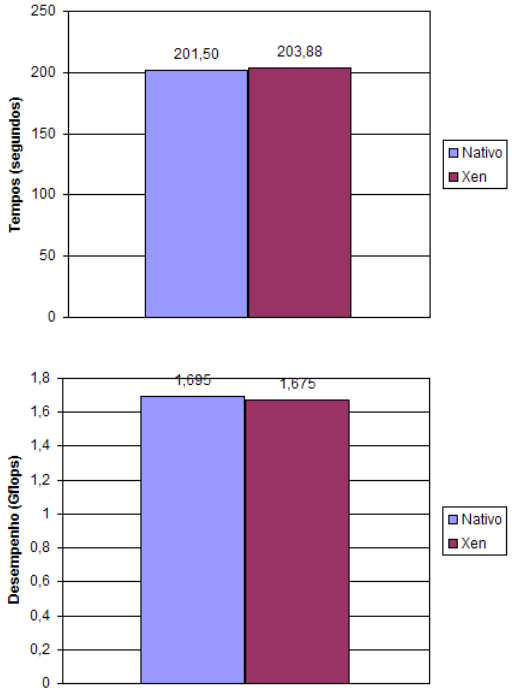

Figura 6. Comparação de desempenho de HPL com 3 máquinas virtuais, 1 processo por máquina.

dos os recursos disponíveis, já que o nós utilizados são bi-processados.

5.4. Migração de Máquinas Virtuais

Antes de realizar os testes 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 (ver funcionamento da migração Xen na Seção 2).

O próximo passo foi executar a aplicação HPL em 3 máquinas virtuais, uma em cada nó do agregado como no experimento do gráfico da Figura 6, e migrar uma delas para um quarto nó utilizando o comando de *live migration* de Xen. O resultado desse experimento pode ser observado no gráfico da Figura 7.

Com a realização de uma migração, houve um acréscimo de 28,4% no tempo de execução de HPL (conforme o gráfico da Figura 7). Verificou-se também que o tempo de migração da máquina virtual durante a execução de HPL

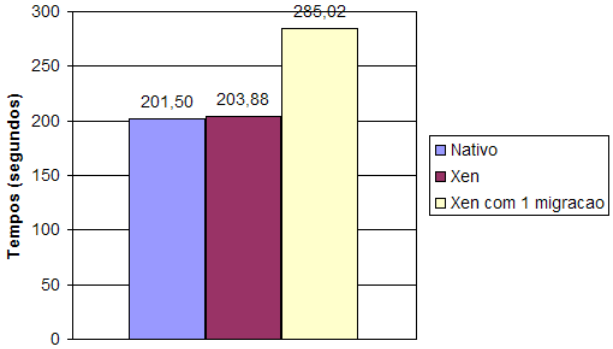

Figura 7. Custo de uma migração na aplicação HPL.

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 turnos 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 mostrado no experimento da Figura 7, optou-se por utilizar a

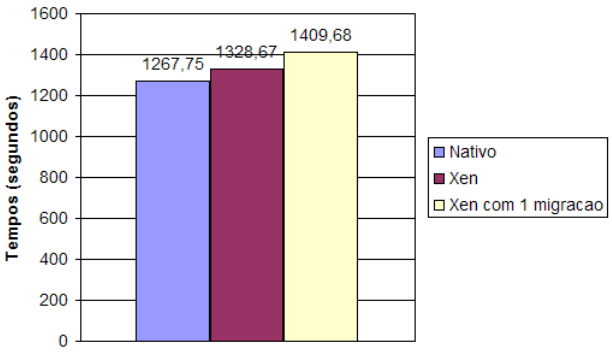

Figura 8. Custo de uma migração na aplicação SP do programa NPB.

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 8 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 8 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[21]), o custo de migração passe a se tornar ínfimo.

6. Conclusão

A utilização de máquinas virtuais é relevante para criar mais nós lógicos que físicos em um agregado, aumentar a utilização dos seus recursos e proporcionar mais eficiência na execução de aplicações paralelas. Especificamente nessa última questão, tem-se a opurtunidade de realizar a migração de uma máquina virtual para um novo recurso. No caso do sistema Xen, na tranferência de uma máquina virtual ocorre a migração de todos os processos que estão executando no seu sistema operacional e todas as conexões de rede deles são refeitas de forma transparente usando o mecanismo de *ARP Reply*. A migração de máquinas virtuais é importante para tentar diminuir o tempo de duração de uma aplicação paralela, pois uma máquina virtual sobrecarregada pode ter seus processos executados sobre um outro recurso com melhores condições de desempenho no momento.

O presente artigo apresentou 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 5. Uma migração na aplicação HPL aumentou a sua execução em 28%, enquanto que na NPB essa sobrecarga foi de aproximadamente 10%.

Além de migração, esse artigo mostrou testes de Xen nas partes de computação e comunicação. A idéia desses testes foi avaliar Xen em tarefas comuns de aplicações paralelas com troca de mensagens, como é o caso de MPI. Os resultados nesse sentido mostraram que Xen é competitivo e apresenta tempos próximos, ou melhores, ao sistema nativo (não virtualizado). Nos testes de comunicação 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.

Além do caráter numérico em relação ao tempo de migração, outro aspecto observado foi a facilidade para realizar a migração de processos através do sistema Xen. Ela é dada simplesmente numa linha de comando onde são informados a máquina virtual alvo (que executam um ou mais processos MPI) e o nó destino. Métodos tradicionais realizam a migração de processos MPI usando *checkpoint* de tempos em tempos. Entretanto, essa estratégia é onerosa pois engloba algumas dificuldades como a obtenção de um estado global consistente e trabalham com soluções específicas[18, 22].

Nesse artigo foram mostrados resultados do uso de Xen para a migração de processos de aplicações MPI. 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. 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[21] será estudada.

Referências

[1] K. Adams and O. Agesen. A comparison of software and hardware techniques for x86 virtualization. In *ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 2–13, New York, NY, USA, 2006. ACM Press.

[2] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The nas parallel benchmarks - summary and preliminary results. In *Supercomputing '91: Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, pages 158–165, New York, NY, USA, 1991. ACM Press.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] T. Clark. *Storage Virtualization: Technologies for Simplifying Data Storage and Management*. Addison-Wesley Professional, 2005.

[8] Dongarra, Moler, Bunch, and Stewart. *Linpack*. Philadelphia, 1986.

[9] K. J. Duda and D. R. Cheriton. Borrowed-virtual-time (BVT) scheduling: supporting latency-sensitive threads in a general-purpose scheduler. *Operating Systems Review*, 33(5):261–276, Dec. 1999.

[10] G. Goth. Virtualization: Old technology offers huge new potential. *IEEE Distributed Systems Online*, 8(2):3, 2007.

[11] W. Huang, J. Liu, B. Abali, and D. K. Panda. A case for high performance computing with virtual machines. In *ICS '06: Proceedings of the 20th annual international conference on Supercomputing*, pages 125–134, New York, NY, USA, 2006. ACM Press.

[12] M. L. Massie, B. N. Chun, and D. E. Culler. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817–840, July 2004.

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

[14] A. Nagarajan, F. Mueller, C. Engelmann, and S. L. Scott. Proactive fault tolerance for HPC with Xen virtualization. In *Proceedings of the* 21^{st} *ACM International Conference on Supercomputing (ICS) 2007*, Seattle, WA, USA, June 16-20, 2007. To appear.

[15] Z. Pan, X. Ren, R. Eigenmann, and D. Xu. Executing mpi programs on virtual machines in an internet sharing system. In *20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*. IEEE, 2006.

[16] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary. Hpl - a portable implementation of the high-performance linpack benchmark for distributed-memory computers. Disponivel em http://www.netlib.org/benchmark/hpl/. Acessado em junho de 2007.

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

[18] 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.

[19] 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.

[20] Q. Snell, A. Mikler, and J. Gustafson. NetPIPE: A Network Protocol Independent Performace Evaluator, 1996.

[21] 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.

[22] G. Stellner. CoCheck: Checkpointing and Process Migration for MPI. In *Proceedings of the International Parallel Processing Symposium*, pages 526–531, Honolulu, HI, Apr 1996. IEEE Computer Society Press.

[23] 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 of *Lecture Notes in Computer Science*, pages 474–486. Springer, 2006.