MPI sobre MOM: Migração de processos para o Open MPI

Caciano dos Santos Machado, Alexandre da Silva Carissimi, Philippe Olivier Alexandre Navaux
Grupo de Processamento Paralelo e Distribuído

E-mail: {caciano,asc,navaux}@inf.ufrgs.br

Resumo

Process migration mechanisms in parallel applications need a way to manage messages in transit. Current migration solutions for MPI don't allow sending messages to the migrating process until it's finished. The proposed work is an analisys of feasibility of using Message-Oriented Middleware as a transport layer for Message Passing Interface during the process migration. MOM enables decoupled communication which allows to send messages to process even when they are migrating.

1. Introdução

O padrão de troca de mensagens MPI não prevê nem balanceamento de carga nem migração de processos. Apesar disso, existem algumas implementações do MPI que oferecem esses recursos. Cada implementação possui uma forma de gerenciar as comunicações entre os processos, durante a migração. O presente trabalho visa analisar a viabilidade e vantagens da comunicação desacoplada em aplicações MPI durante a migração. A comunicação desacoplada permite que um processo envie mensagens para o outro sem que o destinatário esteja disponível e sem precisar conhecer seu endereço real. Essas propriedades são vantajosas nas migrações pois os remetentes de mensagens podem continuar o processamento sem bloquear nas operações de envio se seus destinatários estiverem migrando.

O artigo prossegue apresentando os trabalhos relacionados, uma descrição do Open MPI e do OpenAMQ. Posteriormente mostra os testes preliminares de desempenho realizados com o OpenAMQ como transporte de mensagens no Open MPI. Por útimo são feitas considerações finais e é mostrado como será o andamento do trabalho.

2. Trabalhos Relacionados

A migração de processos em arquiteturas de memória distribuída requer o tratamento das mensagens que são encaminhadas ao processo que está migrando. O tratamento pode ser feito simplesmente bloqueando os processos que enviam mensagens ou através de logs de mensagens. Na segunda opção as mensagens guardadas devem ser encaminhadas posteriormente para o processo que migrou. A seguir serão apresentadas algumas implementações de MPI que realizam migração de processos e a maneira com que as mensagens são tratadas durante a migração.

O projeto MPICH-V [3] visa oferecer suporte a tolerância a falhas para arquiteturas dinâmicas e algumas das implementações do projeto também permitem migração de processos. As implementações MPICH-V1 e MPICH-V2 utilizam um protocolo não coordenado de *checkpoint* que armazena os estados dos processos em um servidor. As mensagens recebidas desde o último *checkpoint* do processo são armazenadas em uma fila para que a aplicação possa ser restaurada para o estado atual. Nas outras duas implementações do MPICH-V, o MPICH-VCL e o MPICH-PCL, e também no CoCheck [8] são realizados checkpoints coordenados. Só é possível migrar processos durante um estado global consistente e nenhum outro processo pode se comunicar com o processo que está migrando.

O AMPI [4] (*Adaptative MPI*) utiliza o *framework* orientado a objetos Charm++ para oferecer balanceamento de carga dinâmico. Esse *framework* utiliza o conceito de processadores virtuais e permite a comunicação dos objetos através de chamadas remotas de métodos. O suporte a migração do AMPI é realizado transferindo os objetos de um processador virtual para outro.

O MPI-Mitten [5] propõe uma solução de migração que também suporta o gerenciamento da comunicação em grupo. Para que um processo migre é necessária o consenso dos processos membro do grupo. Quando um processo migra de um grupo para outro o middleware reestabelece os comunicadores MPI dos processos nos grupos resultantes. O MPI-Mitten é um middleware que opera entre a aplicação e as implementações do MPI. Segundo os criadores é possível utilizar qualquer implementação do MPI.

3. MPI sobre MOM

A motivação do presente trabalho partiu da possibilidade de realizar as comunicações durante a migração de processos utilizando Midlewares Orientados a Mensagem (*Message-Oriented Middleware – MOM*) [2]. O funcionamento da proposta é similar ao implementado no MPICH-V no qual existem servidores que guardam as mensagens enviadas até o processo destinatário consumí-las. A principal diferença é a forma como as filas são utilizadas. No MPICH-V, além de servir de canal de comunicação, as filas servem como logs de mensagens para restauração do estado atual da aplicação em caso de falha. Na proposta implementada, as mensagens consumidas da fila não são guardadas pois o objetivo é apenas permitir a troca de mensagens entre os processos e não a tolerância a falhas.

O MOM possui como propriedade a comunicação desacoplada, formalizada por Eugster [6] em seu trabalho sobre modelos Publish/Subscriber. No trabalho são caracterizadas três propriedades de acoplamento dos *middlewares*: acoplamento temporal, acoplamento espacial e acoplamento de sincronização. Para cada uma dessas propriedades um determinado *middleware* pode ser classificado como acoplado ou desacoplado. Considerando as duas primeiras propriedades, o MOM é classificado com desacoplamento temporal e desacoplamento espacial. O MPI, por sua vez, possui comunicação acoplada temporalmente e espacialmente. O desacoplamento temporal indica que, quando uma aplicação realiza uma operação de envio ou de recebimento de mensagem, a aplicação comunicante não precisa estar envolvida na interação. O desacoplamento espacial significa que uma aplicação que está enviando uma mensagem não necessita conhecer o endereço real da aplicação destinatária. Um endereço simbólico (canal) é utilizado para esse fim. Essas duas propriedades são as peças chave para a implementação de migração de processos proposta, pois permitem que um processo indisponível durante a migração possa continuar recebendo mensagens em seus canais de comunicação.

Para o presente trabalho foi utilizada a implementação OpenAMQ da especificação AMQP [1] *Advanced Message Queuing Protocol*. No entanto, qualquer outro MOM que pudesse trabalhar com o modelo PTP poderia ser utilizado sem perda de generalidade da proposta. O MPI (*Message Passing Inteface*) é um padrão para troca de mensagens e outras funcionalidades utilizadas em aplicações paralelas. O Open MPI [7] é uma implementação de código aberto do MPI que combina os esforços de quatro outras implementações. A semântica de comunicação ponto-a-ponto do Open MPI é definida pelo *framework* PML (Point-to-Point Message Layer). Os três componentes PML principais são o OB1, o DR e o CM. O OB1 e o DR implementam reconhecimento de mensagens e transferência de dados, e utilizam o *framework* BTL (*Byte Transfer Layer*) para realizar a interface com as tecnologias de interconexão. O componente CM gerencia as requisições de comunicação ponto-a-ponto utilizando o *framework* MTL (*Message Transfer Layer*), que fica responsável pelo reconhecimento de mensagens e transferência dos dados. O MTL foi projetado especificamente para redes como a Portals e a Myrinet MX que são capazes de implementar reconhecimento de mensagens na sua própria biblioteca de comunicação.

O Open MPI possui uma implementação de *checkpoint/restart* para tolerância a falhas. Essa implementação é similar ao serviço oferecido pelo LAM/MPI. O serviço utiliza coordenação global centralizada para estabelecer estados globais consistentes de *checkpoint*. O componente CRS dos sistema *checkpoint/restart* de de é de especial interesse para a presente proposta. O CRS permite realizar o *checkpoint/restart* de processos individuais e pretende-se utilizá-la no mecanismo de migração proposto.

4. Implementação do Componente MTL para Open MPI

O trabalho realizado foi o desenvolvimento de um componente do *framework* MTL do Open MPI. O componente em questão atua como um *backend* para a troca de mensagens ponto-a-ponto através do MOM OpenAMQ. Essa implementação permite que seja realizada troca de mensagens com as características de desacoplamento temporal e espacial descritas anteriormente.

A implementação desenvolvida suporta atualmente comunicações ponto-a-ponto bloqueantes (MPI_Send e MPI_Recv) para mensagens de até 100KB. Para isso, cada processo da aplicação paralela cria uma fila para recebimento de mensagens em um servidor OpenAMQ. Nas Subseções seguintes serão ilustrados os passos para inicialização do componente e o funcionamento da troca de mensagens.

A seleção do *framework* e respectivos componentes de comunicação que serão carregados no Open MPI é realizada em tempo de execução na chamada do comando *mpirun*. Após o carregamento do componente as rotinas de inicialização são ativadas.

A Figura 1 mostra cada processo da aplicação paralela registrando as informações para conexão no seu servidor OpenAMQ [1] no serviço de nomes do Open MPI (1). As informações de conexão consistem em IP, porta, usuário, senha e chave de roteamento (*routing key*) para a fila de

1 Cada processo pode estar vinculado a um servidor diferente. O OpenAMQ permite uma configuração de clusters de servidores para distribuir a carga e minimizar gargalos.

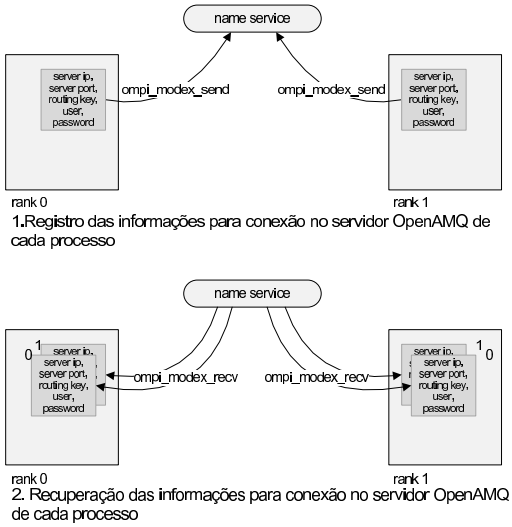

Figura 1. Informações de conexão.

mensagens. Em seguida, cada um dos processo busca as informações de conexão dos demais processos (2).

A Figura 2 mostra cada processo se conectando ao servidor OpenAMQ (1). Cada um deles cria uma fila para si que será usada para recebimento de mensagens (2.1). A operação de *bind* realiza a associação de uma chave de roteamento a uma fila de mensagens (2.2). Assim, cada vez que um processo enviar uma mensagem para a *exchange* do servidor, que contenha uma determinada chave de roteamento, essa mensagem será encaminhada para a fila associada à chave.

O envio de mensagens é realizado publicando uma mensagem na *exchange* utilizando a chave de roteamento que foi associada ao processo destinatário, como mostra a Figura 3 na etapa 1. A *exchange* fica encarregada de encaminhar a mensagem para a fila correspondente (2). O recebimento ocorre consumindo a mensagem publicada na fila (3).

5. Avaliação do Desempenho

Foram realizados testes de desempenho preliminares na implementação. Os resultados foram comparados com o desempenho do componente OB1 para comunicação via TCP/IP. Para avaliar o impacto na latência e na largura de banda da comunicação foi utilizada uma aplicação ping-pong. Os resultados apresentados são a média de 1000 execuções da aplicação.

Cada nó utilizado nos testes possui 16GB de RAM, 2 Quad Core Xeon E5310 e interconexão Gigabit Ethernet. Os processos MPI se comunicam diretamente quando utilizam o componente OB1/TCP e indiretamente, através do servidor OpenAMQ, quando utilizam o componente CM/MOM. Em um dos nós foi executado o servidor OpenAMQ e em cada um dos outros dois nós foi executado um processo da aplicação ping-pong.

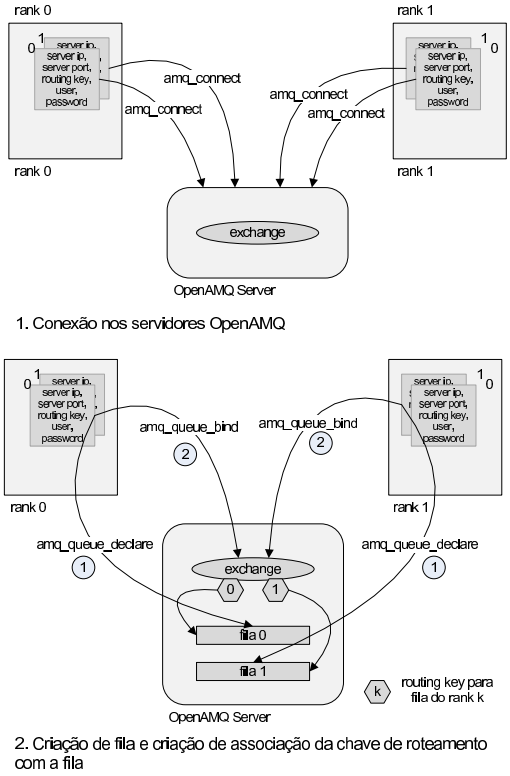

Figura 2. Conexão nos servidores OpenAMQ.

Os resultados da Figura 4 demonstram que a utilização do mecanismo de filas acarreta em uma degradação no desempenho. A latência média das mensagens sofre um grande impacto nas mensagens pequenas e passa para cerca de 5,5 vezes o valor obtido com a implementação OB1/TCP. A razão diminui progressivamente conforme as mensagens aumentam de tamanho, até aproximadamente 32K, valor no qual a razão se estabiliza em cerca de 2 vezes a latência do OB1/TCP. A largura de banda, por sua vez, é a metade da obtida com o OB1. O desempenho mais baixo no CM/MOM era esperado e se deve à dupla transferência de mensagens que é realizada através da rede.

6. Conclusão

Os mecanismos de migração de processos são úteis em diversos cenários. No contexto de aplicações paralelas esses mecanismos são particularmente importantes para o balanceamento de carga e para o gerenciamento das arquiteturas. Existem algumas implementações do padrão MPI que oferecem suporte a migração de processos. O presente artigo mostrou as vantagens que a comunicação de-

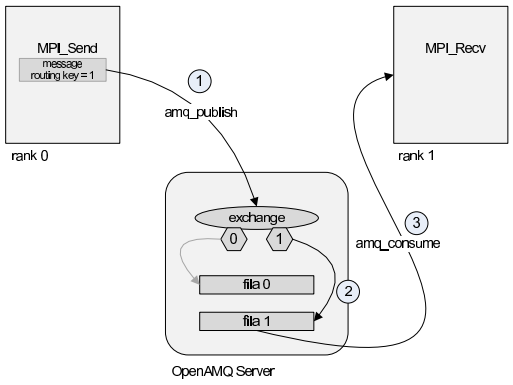

Figura 3. Envio e recebimento.

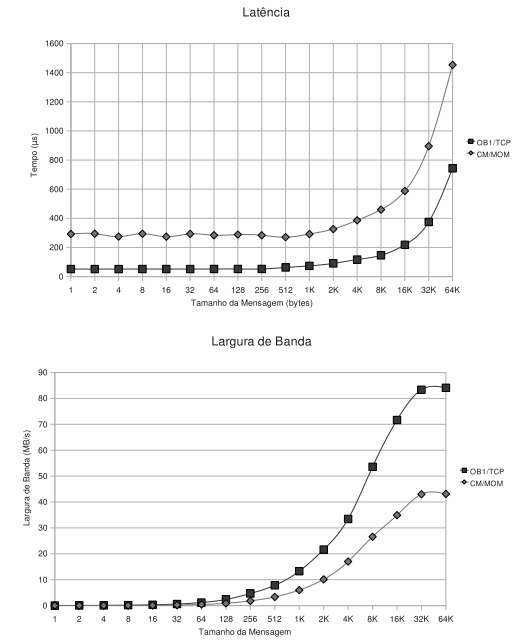

Figura 4. Latência e Largura de Banda

sacoplada pode trazer para a migração, uma implementação de componente para o Open MPI que permite esse tipo de comunicação e seus respectivos testes.

A utilização de soluções de MOM como transporte para mensagens de aplicações MPI se mostrou viável através dos testes realizados na implementação do componente do Open MPI. O desempenho obtido com o componente CM/MOM é mais baixo que o obtido com o OB1/TCP devido à dupla transferência de mensagens realizada através da rede. No entanto, deve-se levar em consideração a vantagem que o desacoplamento na comunicação pode trazer para o desempenho das aplicações nos casos de migração. Além disso, pode-se estudar a possibilidade de utilizar a comunicação desacoplada somente quando houver necessidade de migração de processos.

O componente desenvolvido é um elemento importante para a continuidade do trabalho. Como próximo passos será objeto de estudo a implementação de *checkpoint/restart* de processos individuais do Open MPI para realizar a migração de processos. Se não houver viabilidade de utilizar a implementação existente então será criada uma nova para que se possa dar andamento à pesquisa proposta. Pretende-se posteriormente utilizar *benchmarks* para verificação da latência, largura de banda, disponibilidade do processador e aplicações sintéticas. Os testes serão comparados com outras implementações de migração que forem julgadas relevantes.

Referências

[1] AMQP Working Group. Advanced Message Queuing Protocol 0-10. Disponível em: http://amqp.org/. Acesso em: junho de 2008.

[2] G. Banavar, T. Chandra, R. Strom, and D. Sturman. A Case for Message-Oriented Middleware. In *Proceedings of the 13th International Symposium on Distributed Systems, LNCS 1693*, pages 1–18, Bratislava, Slovak Republic, 1999. Springer Berlin/Heidelberg.

[3] G. Bosilca, A. Bouteiller, F. Cappello, S. Djilali, G. Fedak, C. Germain, T. Herault, P. Lemarinier, O. Lodygensky, F. Magniette, V. Neri, and A. Selikhov. MPICH-V: toward a scalable fault tolerant MPI for volatile nodes. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–18, Los Alamitos, CA, USA, November 2002. IEEE Computer Society Press.

[4] L. V. K. Chao Huang, Orion Lawlor. Adaptive MPI. In *Proceedings of the 16th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2003), LNCS 2958*, pages 306–322, Bratislava, Slovak Republic, 2003.

[5] C. Du and X.-H. S. Sun. MPI-Mitten: Enabling Migration Technology in MPI. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID '06)*, pages 11–18, Washington, DC, USA, May 2006. IEEE Computer Society.

[6] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[7] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings of the 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.

[8] G. Stellner. CoCheck: Checkpointing and Process Migration for MPI. In *Proceedings of the 10th International Parallel Processing Symposium (IPPS '96)*, Honolulu, Hawaii, USA, 1996.