MPI sobre Message-Oriented Middleware

Caciano Machado, Alexandre Carissimi, Philippe Navaux
Universidade Federal do Rio Grande do Sul
Grupo de Processamento Paralelo e Distribuído
{caciano,asc,navaux}@inf.ufrgs.br

Resumo

The present work proposes an implementation of a MPI (Message Passing Interface) communication channel which is going to work as a backend for MOMs (Message Oriented Middlewares) of the AMQP (Advanced Message Queueing Protocol) Standard. The goal is to allow MPI processes to communicate transparently in a decoupled way through Point-to-Point and Publish-Subscribe mechanisms. The decoupled message passing between MPI processes is an interesting property which has many potential benefits. One of the possible advantages with this kind of communication is easy process migration mechanisms because it simplifies the re-establishment of communications after migrations. Other benefits are related with grid computing environments where MOM communication would enable fault-tolerant features.

1. Introdução

Comunicação orientada a mensagens é um mecanismo de comunicação onde a mensagem é unidade básica de dados transmitida pelos processos. Entre os sistemas que se baseiam no conceito de mensagens estão o padrão MPI [12] (*Message Passing Interface*) e o MOM [4] (Middleware orientado a mensagens – *Message-Oriented Middleware*).

Apesar de ambos os sistemas terem como base a troca de mensagens, existem algumas diferenças fundamentais relacionadas ao acoplamento dos processos transmissores e receptores. No MPI os processos da aplicação precisam estar disponíveis simultaneamente para que haja a troca de mensagens. Além disso, quando um processo MPI envia uma mensagem, ele precisa do endereço do processo destinatário. No caso dos MOMs estas restrições não existem. Estas são as diferença básicas entre os sistemas de mensagem acoplados (MPI) e desacoplados (MOM).

O MPI é um padrão de troca de mensagens criado para o desenvolvimento de aplicações paralelas de alto desempenho. O MOM, por sua vez, surgiu da necessidade da indústria de integrar aplicações. As diferentes necessidades de comunicação modelaram estes dois tipos de sistemas.

O presente trabalho propõe a implementação de um módulo para o MPI que funcionará como um *backend* para comunicação através de MOMs. O objetivo é analisar os benefícios que este tipo de comunicação pode trazer para a computação paralela utilizando MPI.

Entre as possíveis vantagens está a implementação simplificada de um mecanismo de migração de processos. Com o desacoplamento da comunicação entre os processos MPI, o reestabelecimento das comunicações se torna mais simples. Outros possíveis benefícios deste modelo de comunicação estão relacionados à computação em grade. Acredita-se que a comunicação desacoplada permita alcançar algumas propriedades de tolerância a falhas.

Na seção 2 explicaremos com mais detalhes alguns conceitos importantes da comunicação orientada a mensagens com MPI e MOM. Na seção 3 apresentaremos os objetivos do trabalho e algumas decisões de implementação já tomadas.

2. Comunicação orientada a mensagens

Comunicação orientada a mensagens [15] (*Message-Oriented Communication*) é um mecanismo de comunicação entre processos. Neste tipo de comunicação as mensagens são a unidade básica de dados entregue aos processos.

Estes sistemas são classificados em transientes ou persistentes, de acordo com o comportamento do sistema quando uma das pontas da comunicação fica indisponível. A Figura 1 apresenta as possíveis combinações de disponibilidade das pontas nas comunicações ponto a ponto.

Persistentes As mensagens são armazenadas pelo sistema de comunicação (em buffers ou disco) mesmo quando algum transmissor ou receptor estiver indisponível; Ex: Email, Middleware Orientado a Mensagens (MOM)

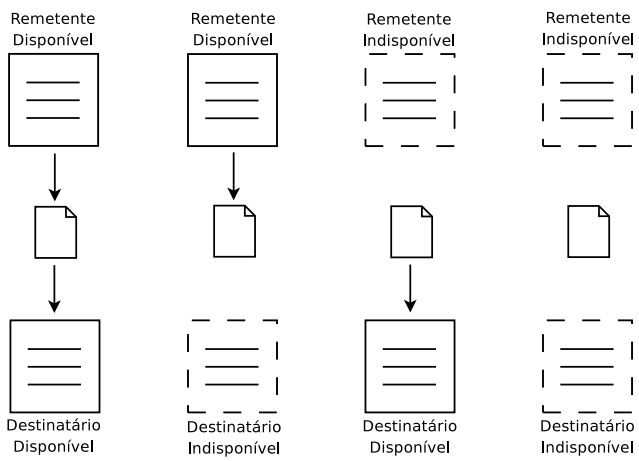

Figura 1. Quatro combinações de comunicação desacoplada

Transientes As mensagens só são armazenadas pelo sistema de comunicação enquanto os transmissores e os receptores das mensagens estão disponíveis; Ex: Sockets, MPI

Outra taxonomia [1][1] classifica estes mesmos conceitos como acoplamento temporal (transiente) e desacoplamento temporal (persistente). Além disso, acrescenta o conceito de acoplamento e desacoplamento espacial que indica se cada ponta da comunicação têm conhecimento do endereço real da outra ponta ou se utiliza um endereço virtual para o envio das mensagens.

Acoplamento Temporal

Acoplado Comunicação transiente

Desacoplado Comunicação persistente

Acoplamento Espacial

Acoplado Endereços reais; Ex: Sockets, RPC, MPI

Desacoplado Canais de comunicação; Ex: Email, RMI, MOM

As taxonomias consultadas também apresentam classificações de acordo com a sincronização dos processos com entidades externas, no envio e recebimento das mensagens. Omitiremos estas classificações pois elas estão fora do escopo do trabalho proposto. A seguir apresentamos o MPI e o MOM, como cada um surgiu e como o MPI pode se beneficiar das características dos MOMs.

1 Nesta classificação não é considerada apenas a Comunicação Orientada a Mensagens. Outros meios de comunicação como RPC e RMI também são levados em conta.

2.1. Message Passing interface

O padrão MPI surgiu de uma necessidade de desenvolvedores de aplicações paralelas de alto desempenho. A necessidade era de um padrão de primitivas de troca de mensagens com um nível de abstração adequado para facilitar o desenvolvimento de aplicações paralelas. As primitivas do padrão deveriam permitir também implementações com overhead de comunicação mínimo em diferentes tecnologias de interconexão.

A necessidade de um overhead de comunicação mínimo foi responsável pelo acoplamento temporal e espacial do MPI. As implementações do MPI suportam tecnologias de interconexão de alta velocidade que teriam o desempenho comprometido com múltiplas cópias de mensagem necessárias para o desacoplamento temporal ou a utilização de canais de comunicação para o desacoplamento espacial.

2.2. Middleware Orientado a Mensagens

Os MOMs surgiram de uma necessidade da indústria de integrar aplicações. Um dos problemas da integração é o das aplicações que executam em domínios administrativos diferentes. Estas aplicações necessitam de um mecanismo desacoplado de comunicação pois um domínio administrativo não tem controle sobre o outro e pode estar indisponível em determinados momentos.

Várias outras tecnologias surgiram paralelamente para resolver este problema de integração de aplicações corporativas: *Web Services* [2], *Message Brokers*, *Enterprise Application Integration* (EAI), *Service Oriented Architeture* (SOA) e *Enterprise Service Bus* (ESB).

O ESB [8] é considerado pelo mercado como a principal solução para o problema. O ESB combina as funcionalidades de MOM, *Message Brokers* e *Web Services* para interconectar diferentes aplicações corporativas através de um barramento de serviços com segurança e integridade transacional. Os *Web Services* e *Message Brokers* oferecem a entrada das mensagens ao sistema de MOM.

O MOM é responsável por garantir a troca de mensagens desacoplada entre as aplicações. Desacopladas espacialmente, no sentido que não é necessário que o produtor e o consumidor das mensagens tenham conhecimento um do outro, e desacopladas temporalmente, no sentido de que o produtor e o consumidor não necessitam estar disponíveis simultaneamente.

MOMs podem ser classificados em diferentes modelos de comunicação que determinam que cliente recebe as mensagens. Os modelos mais comuns são:

PTP *Point-to-Point*

Pub/Sub *Publish-Subscriber*

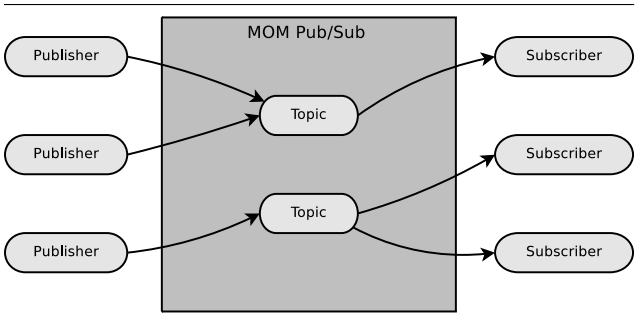

Figura 2. Pub/Sub - Comunicação 1 para N

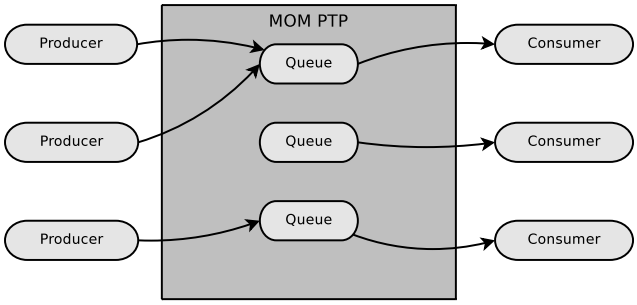

Figura 3. PTP - Comunicação 1 para 1

Os canais de cada um destes modelos de comunicação são denominados, respectivamente, filas e tópicos. As aplicações consumidoras se registram nas filas ou tópicos para receber mensagens. As aplicações produtoras, por sua vez, enviam mensagens para as filas ou para os tópicos. As figuras 2 e 3 ilustram o funcionamento básico da comunicação Pub/Sub e PTP. No caso da comunicação Pub/Sub, as mensagens enviadas podem ser recebidas por qualquer número de assinantes (*subscribers*) dos tópicos. Para a comunicação PTP existe a restrição de que somente um consumidor pode receber cada mensagem.

Os sistemas de MOM podem ser formados por clusters de servidores para atender propriedades de alta disponibilidade e escalabilidade. Servidores distribuídos geograficamente também são utilizados para oferecer alta disponibilidade. Neste último caso as aplicações se conectam ao servidor local que se encarrega do roteamento das mensagens pelos demais servidores do sistema de MOM.

O JMS [14] (*Java Message Service*) é o padrão *de facto* para MOMs. Entretanto, o JMS oferece uma solução parcial para padronização de MOMs pois só possui interface para a linguagem Java.

AMQP [3] (*Advanced Message Queuing Protocol*) *Specification* é uma proposta de um padrão aberto para MOMs. O padrão está em fase de desenvolvimento por um consórcio de empresas e está na versão 0-9[2]. O AMQP especifica uma solução que pode ser aplicada em qualquer plataforma e em qualquer linguagem.

O padrão AMQP prevê o funcionamento do *Message Broker* e o conjunto de componentes do MOM que roteiam e armazenam as mensagens através dos servidores. Em cada servidor existem três tipos básicos de componente:

Exchange Recebe as mensagens dos produtores e redireciona para as filas baseado em critérios como o conteúdo ou propriedades da mensagem

Message Queue Armazena mensagens até que sejam consumidas pelas aplicações clientes

Bind Define uma relação entre uma message queue e um exchange, e especifica o critério de roteamento das mensagens

Tanto o modelo Pub/Sub quanto o PTP são previstos pelo AMQP. Atualmente existem algumas implementações do AMQP. Estamos analisando o OpenAMQ [11] que é a implementação utilizada como referência pelo consórcio do AMQP.

3. MPI sobre MOM

A proposta do trabalho é a implementação de um *backend* que permita que primitivas do MPI utilizem um mecanismo desacoplado de comunicação como mostra a Figura 4. Uma das motivações para a criação deste módulo é a possibilidade de realizar a migração de processos MPI com mais facilidade.

A abordagem escolhida para realizar a migração de processos é a *Checkpoint/Restart* utilizando a biblioteca BLCR [10]. No *Checkpoint/Restart* é realizado um *checkpoint* do processo no nodo origem e um *restart* no nodo destino. O problema que permanece é o reestabelecimento das comunicações entre os processos. No caso do protocolo TCP/IP, utilizado em algumas implementações de MPI, os outros processos continuarão fazendo referência ao endereço IP do nodo origem da migração.

O acoplamento espacial da comunicação do MPI faz com que os processos utilizem endereços reais. A utilização de canais de comunicação (no caso as filas de mensagens e tópicos) facilita o reestabelecimento das comunicações com o processo emigrante. Além disso, os MOMs possuem a capacidade de refazer o roteamento das mensagens após mudanças na topologia dos servidores que o compõem. [9].

Decidimos utilizar o AMQP como plataforma de MOM para o módulo MPI por se tratar de um padrão aberto e por

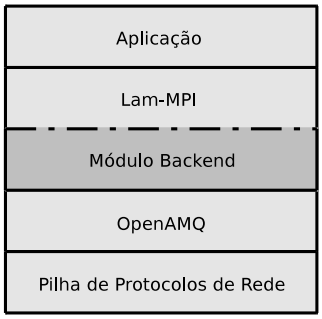

Figura 4. Backend MPI-AMQP

possuir implementações de código aberto. Após o estudo de algumas implementações do padrão decidiu-se utilizar o OpenAMQ [11] que é a implementação de referência do padrão. Escolhemos o OpenAMQ por ele ser escrito em C e C++, o que facilita o desenvolvimento do módulo MPI cujas versões são predominantemente nestas linguagens.

Quanto à versão do MPI, a princípio, optou-se por implementar o módulo no LAM-MPI [5] pela sua estabilidade, e pela sua interface modular bem definida e documentada [13]. Outro motivo que levou a escolher o LAM-MPI foi a possibilidade de integração com outros trabalhos, realizados pelo grupo de pesquisa, sobre balanceamento de carga dinâmico de aplicações paralelas [6].

O nosso objetivo principal com a migração de processos é fornecer o suporte para balanceamento de carga dinâmico de aplicações paralelas. Atualmente existem algumas implementações do MPI que oferecem o recurso de migração de processos e balanceamento de carga dinâmico. Uma das mais notáveis é o AMPI [7]. No entanto, deseja-se avaliar o comportamento de um mecanismo de migração de processos MPI que utilize comunicação desacoplada.

4. Conclusão e Andamento

A comunicação orientada a mensagens pode ser classificada de acordo com o acoplamento da comunicação. O desacoplamento da comunicação oferecido por sistemas de MOM pode apresentar vantagens em aplicações MPI que possuem tradicionalmente comunicação acoplada. Das possíveis vantagens podemos citar a simplificação da migração de processos e mecanismos de tolerância a falhas.

Prosseguiremos com o trabalho realizando testes na implementação do AMQP para decidir qual configuração é mais adequada para utilizá-lo como infra-estrutura de MOM para o MPI. Posteriormente partiremos para a implementação do módulo *backend* entre o LAM-MPI e OpenAMQ, e seus respectivos testes.

Referências

[1] L. J. Aldred, W. M. van der Aalst, M. Dumas, and A. H. ter Hofstede. On the Notion of Coupling in Communication Middleware. In *Proceedings On the Move to Meaningful Internet Systems - 7th International Symposium on Distributed Objects and Applications (DOA) 3761*, pages 1015–1033, Agia Napa, Cyprus, 2005. Springer-Verlag.

[2] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services Concepts, Architectures and Applications*. Springer-Verlag, 2004.

[3] AMQP Working Group. Advanced Message Queueing Protocol 0-9. Disponível em: http://www.amqp.org/. Acesso em: julho de 2007.

[4] 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.

[5] G. Burns, R. Daoud, and J. Vaigl. LAM: An Open Cluster Environment for MPI. In *Proceedings of Supercomputing Symposium*, pages 379–386, 1994.

[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 *Proceedings of EuroPVM/ MPI'2006. European PVM/MPI Users' Group Meetings, LNCS 4192*, pages 247–255, Bonn, Germany, September 2006. Springer Berlin/Heidelberg.

[7] 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.

[8] D. A. Chappell. *Enterprise Service Bus*. O'Reilly, 2004.

[9] P. Giotta, G. Scott, M. Kovacs, S. Maffeis, K. S. Morrison, G. S. Raj, and M. M. Kunnumpurath. *Professional JMS Programming*. Wrox Press, Birmingham, UK, 2000.

[10] P. H. Hargrove and J. C. Duell. Berkeley Lab Checkpoint/Restart (BLCR) for Linux Clusters. In *Proceedings of SciDAC 2006, LBNL-60520*, Denver, US, 2006.

[11] iMatix Corporation. Open Advanced Message Queue. Disponível em: http://www.openamq.org/. Acesso em: agosto de 2007.

[12] MPI FORUM. The MPI message passing interface standard. Technical report, University of Tennessee, Knoxville, 1994.

[13] J. M. Squyres and A. Lumsdaine. A Component Architecture for LAM/MPI. In *Proceedings, 10th European PVM/MPI Users' Group Meeting*, number 2840 in Lecture Notes in Computer Science, pages 379–387, Venice, Italy, September / October 2003. Springer-Verlag.

[14] Sun Microsystems, Inc. Java Message Service Specification 1.1. Disponível em: http://java.sun.com/products/jms/docs.html. Acesso em: julho de 2007.

[15] A. S. Tanenbaum. *Distributed Systems Principles and Paradigms*. Prentice Hall, Upper Saddle River, US, 2002.