Desenvolvimento de uma biblioteca de comunicação em software fazendo uso do modelo de Mensagens Ativas

César Henrique Vortmann[1], Daniela Saccol Peranconi[2], Gerson G. H. Cavalheiro[1]

[1]Instituto de Informática – Universidade Federal de Pelotas (UFPEL)
Caixa Postal 354 – 96.010-900 – Pelotas – RS – Brasil
[2]Departamento de Informática – Universidade de Santa Cruz do Sul (UNISC)
Santa Cruz do Sul – RS – Brasil

cvortmann@gmail.com,danielap@unisc.br,gersonc@anahy.org

Resumo. *Este trabalho apresenta uma biblioteca de comunicação desenvolvida em software, a qual implementa o modelo de Mensagens Ativas. O modelo original de Mensagens Ativas foi desenvolvido para ser usado com um processador de comunicação em hardware, entretanto o poder de processamento oferecido pelas atuais arquiteturas multiprocessadas permite sua simulação em software através do uso de multithreading. Utilizou-se a biblioteca MPI para realização da troca de mensagens entre os diferentes nós da arquitetura.*

1.Introdução

Os aglomerados de computadores tornaram-se arquiteturas populares no desenvolvimento do Processamento de Alto Desempenho (PAD) devido, principalmente, a sua facilidade de aquisição e o poder de processamento proporcionado em relação aos supercomputadores.

Ferramentas clássicas utilizadas para programação de aglomerados empregam troca de mensagens, tanto síncronas quanto assíncronas. Para obtenção do PAD deve-se, portanto, minimizar os custos com a comunicação entre os nós da arquitetura. O modelo de Mensagens Ativas [Eiken et al 1992] propõe a realização da comunicação com pequeno custo. Isto é feito através da sobreposição de computação e comunicação com o uso de um processador em *hardware* dedicado à comunicação. Esta sobreposição pode ser alcançada em *software* através do uso de *multithreading* [Valiant 1990] na simulação do processador de comunicação.

Neste contexto, este trabalho apresenta uma biblioteca de comunicação desenvolvida em *software*, a qual implementa o modelo de Mensagens Ativas e simula o processador dedicado à comunicação através do uso de *multithreading*.

Na seqüência, a Seção 2 apresenta o modelo de Mensagens Ativas. Já a Seção 3 apresenta a biblioteca desenvolvida e sua interface de programação aplicativa. Na Seção 4 são discutidos resultados de desempenho da biblioteca e a Seção 5 conclui o artigo.

2.Mensagens Ativas

Uma aplicação implementada para executar em arquiteturas como aglomerados e que vise alto desempenho, deve buscar minimizar o tempo entre a produção de um dado e seu consumo [Eicken et al 1992]. Nesse tipo de arquitetura, produção e consumo podem ocorrer em nós diferentes, o que exige troca de mensagens entre as partes envolvidas.

Uma solução eficiente para implementação de troca de mensagens entre os nós de um aglomerado consiste na utilização do modelo assíncrono denominado Mensagens Ativas [Eicken et al 1992] que, normalmente, está entre os mecanismos de comunicação mais rápidos disponíveis [Lumetta et al 1997]. Nesse modelo, a mensagem contém em seu cabeçalho a identificação de um serviço a ser executado pelo nó receptor e em seu

corpo os dados que compõem os argumentos [Lumetta et al 1997]. A idéia é que o transmissor envie uma mensagem para a rede e continue seu processamento. A recepção de uma mensagem dispara um serviço a ser executado pelo receptor.

O modelo original de Mensagens Ativas foi desenvolvido utilizando um processador, em *hardware*, dedicado à comunicação e funcionava de duas maneiras: com uma política de interrupção, onde o processo em execução era interrompido com a chegada de uma nova mensagem, e com uma política de *polling*, onde o processador de comunicação verificava constantemente a interface de rede.

Como este modelo apresentava problemas em relação a operações de sincronização, desenvolveram-se implementações alternativas: *UpCall*, *PopUp* e Fila de Execução. O modelo *UpCall* caracteriza-se por possuir um *thread* independente para recebimento e tratamento das mensagens. Já o modelo *PopUp* caracteriza-se pela criação de um novo *thread* a cada recebimento de mensagem. Ambos os modelos podem funcionar tanto com a política de interrupção quanto com a política de *polling.*

O modelo de Fila de Execução é um modelo híbrido entre *UpCall* e *PopUp,* com o objetivo de minimizar os problemas encontrados nestes. Caracteriza-se por um *daemon* de comunicação que é responsável por verificar a interface de rede e no recebimento de uma nova mensagem, inseri-la em uma lista de tarefas. Esta lista é constantemente verificada pelos *threads* executando no nó da arquitetura, com o objetivo de executar rapidamente as tarefas descritas pela aplicação [Peranconi 2005].

3.Biblioteca Implementada

A biblioteca foi desenvolvida [Vortmann, 2008] na linguagem de programação C e fez uso do padrão de comunicação MPI [Gropp et al 1994] para troca de mensagens. Optou-se pelo uso de MPI, pois este oferece um nível mais alto de abstração em relação a outras ferramentas como o padrão *sockets*.

O formato das primitivas da interface seguiu o seguinte padrão: os acrônimos *act* e *msg* separados pelo caractere '_' e a identificação, em inglês, do serviço realizado pela primitiva. Na Tab.1 podem ser conferidas as principais primitivas desenvolvidas. Além destas ainda existem: *av_init*, *av_finalize*, *act_msg_realloc* e *act_msg_getserv.*

Tabela 1. Primitivas da Biblioteca

Primitiva	Funcionalidade
act_msg_create	Criar mensagem
act_msg_reset	Apagar mensagem
act_msg_init	Inicializar mensagem com serviço
act_msg_regserv	Registrar serviço
act_msg_send	Enviar mensagem
act_msg_recv	Receber mensagem
act_msg_pack	Empacotar dado na mensagem
act_msg_unpack	Desenpacotar dado da mensagem

Uma aplicação que fará uso da biblioteca deverá definir serviços. Sua implementação se dá na forma de funções sem retorno que recebem como parâmetro um ponteiro para uma área de memória. Estes serviços não devem realizar operações de sincronização ou serem muito extensos, pois executam no corpo do processador de comunicação. Além disso, devem ser registrados em cada nó da arquitetura através da primitiva específica da biblioteca para que os endereços dos serviços na memória de cada nó sejam abstraídos em índices de seus endereços em um vetor de serviços.

O processador de comunicação é executado, no nó zero (0) da arquitetura virtual MPI, em um *thread* criado no processo de inicialização da biblioteca. Nos demais nós, o processador de comunicação é executado sobre o fluxo de execução principal. *UpCall*

com a política de *polling* foi o modelo seguido para implementação deste processador.

4.Resultados de Desempenho

A obtenção dos resultados de desempenho foi realizada, localmente, utilizando um computador com processador *Intel Core Duo* com freqüência de *clock* 1,66GHz, memória *cache* de 2MB e memória RAM de 2,5GB. Para realização dos testes em um ambiente de aglomerado foi conectado, através de uma interface de rede *Ethernet* de 100Mbit/s, um computador com processador *Intel Pentium 4*, com freqüência de clock 1,70Ghz, memória *cache* de 256KB e memória RAM de 512MB. O sistema operacional utilizado foi o *GNU/Linux*, versão do *kernel* 2.6.24-21 e a versão da biblioteca Open MPI utilizada foi a de número 1.2.6.

A metodologia de avaliação de desempenho consistiu em contabilizar o tempo de execução das operações básicas realizadas pela biblioteca. Entre as operações avaliadas estão as de empacotar e desempacotar um dado do tipo *int* com o uso das primitivas *pack/unpack*; a de criação (*create*) de mensagens de tamanho *1024* e *2048* bytes e a de envio (*send*) de mensagem com um dado do tipo *int* empacotado na área de dados e fazendo uso de um serviço vazio. Além destas, utilizou-se um serviço exemplo tipo *ping-pong*, onde um nó virtual desempacota um dado, adiciona um valor ao dado, empacota-o e reenvia para o nó transmissor.

Foram obtidas as médias de 100 operações na operação de *pack/unpack*, *create* e *send.* Para o serviço *ping-pong* foi obtida a média de 20 pares de operações *ping-pong*. As operações de *pack/unpack* e *create* foram avaliadas somente localmente, com a utilização de somente um nó virtual MPI, enquanto que a operação de *send* e o serviço exemplo *ping-pong* foram avaliados tanto localmente quanto em um ambiente de aglomerado com o uso de dois nós virtuais MPI.

Na Tab. 2 pode-se conferir o tempo de execução médio das operações de Pack e Unpack. A análise quantitativa destes resultados permitiu verificar que são satisfatoriamente relevantes aos objetivos procurados.

Tabela 2. Tempo de Execução Médio das Operações Pack/Unpack

Operação	Tempo de Execução Médio (s)
Pack	0,0005
Unpack	0,0004

Já na Tab. 3 pode-se conferir o tempo de execução médio da operação de criação de mensagens tanto com *1024* bytes quanto com *2048* bytes. Os resultados mostraram que o tempo de criação de uma nova mensagem varia linearmente de acordo com o tamanho da mensagem criada.

Tabela 3. Tempo de Execução Médio da operação de criação de mensagens

Tam. da mensagem	Tempo de Execução Médio (s)
1024	0,00012
2048	0,00025

Na Tab. 4 pode-se conferir o tempo de execução médio da operação de envio de mensagens tanto localmente quanto remotamente. Novamente, os resultados mostraram-se satisfatórios em relação aos resultados esperados, apesar de mostrar que o desempenho em rede foi menor que o desempenho local.

Tabela 4. Tempo de Execução Médio da operação de envio de mensagens

Operação Send	Tempo de Execução Médio (s)
Local	0,004
Remoto	0,008

Na Tab. 5 pode-se conferir o tempo de execução médio do serviço exemplo *ping-pong*. Este serviço permitiu verificar que o desempenho das diversas operações necessárias para o envio de uma mensagem estava, assim como o desempenho da biblioteca desenvolvida, dentro das expectativas.

Tabela 5. Tempo de Execução Médio do serviço exemplo *ping-pong*

Serviço *Ping-Pong*	Tempo de Execução Médio (s)
Local	0,01
Remoto	0,02

5.Conclusões e Trabalhos Futuros

Com o uso da biblioteca, o programador pode desenvolver aplicações voltadas ao processamento de alto desempenho utilizando o modelo de Mensagens Ativas sem depender de *hardware* específico e, ainda assim, obter desempenho semelhante.

Como trabalhos futuros, tem-se a inclusão da biblioteca no ambiente de programação paralela Anahy, permitindo estender este ambiente para arquiteturas com memória distribuída e permitir que estas façam uso do mecanismo de escalonamento disponível em Anahy.

Referências

Eicken, T. von; Culler, D. E.; Goldstein, S. C.; Schauser, K. E. (1992) “Active messages: a mechanism for integrated communication and computation”. In ISCA’92, New York, NY, USA. ACM. p.256–266.

Gropp, W.; Lusk, E.; Skjellum, A. (1994) “Using MPI: portable parallel programming with the message-passing interface”. Cambridge, MA, USA: MIT Press.

Lumetta, S. S.; Mainwaring, A. M.; Culler, D. E. (1997) “Multi-protocol active messages on a cluster of SMP’s”. In SUPERCOMPUTING’97 New York, USA. ACM. p.1–22.

Peranconi, D. S. (2005) “Alinhamento de Sequências Biológicas em Arquiteturas com Memória Distribuída”. Dissertação (Mestrado) - UNISINOS, São Leopoldo.

Valiant, L. G. (1990) “A bridging model for parallel computation”. Commun. ACM, New York, NY, USA, v.33, n.8, p.103–111.

Vortmann, C. (2008) "Desenvolvimento de uma biblioteca de serviços de comunicação para um aglomerado de computadores". Monografia (Graduação). Pelotas: UFPel.