Biblioteca de Comunicação com Mensagens Ativas*

Evandro Clivatti Dall'Agnol†, Gerson G. H. Cavalheiro

Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Centro de Ciências Exatas e Tecnológicas
Universidade do Vale do Rio dos Sinos
Avenida Unisinos, 950. Fone/Fax: 590-8161
{ecd,gersonc}@exatas.unisinos.br

Introdução

Aglomerados de computadores têm se tornado um alternativa viável para o processamento de alto desempenho (PAD). Contudo, este tipo de arquitetura tem algumas características importantes que devem ser levadas em consideração na programação de aplicações eficientes, características como memória distribuída, quantidade de processadores por máquina e comunicação entre máquinas.

Neste contexto, este trabalho descreve uma implementação de uma biblioteca de comunicação para uso na programação de alto desempenho (PAD) utilizando arquiteturas multiprocessadas com memória distribuída, também chamados de aglomerados ou *clusters*. Esta biblioteca utiliza em sua concepção o princípio de *mensagens ativas* [CAR 99], onde a chegada de uma mensagem causa a execução de algum procedimento.

A estrutura e a implementação de um protótipo contextualizando seu uso no ambiente de PAD Anahy [CAV 2003] são abordadas. Características como os modelos de memória e de execução são levadas em consideração na implementação da biblioteca. Por fim, resultados preliminares de tempos de comunicação são apresentados e analisados.

Mensagens Ativas

O conceito de mensagens ativas [EIC 92, CAR 99] é de que a chegada de uma mensagem provoca a execução de um procedimento utilizando dados contidos nessa mensagem. As mensagens ativas têm por característica serem compostas por uma identificação do remetente, por uma identificação de uma função a ser executada e por uma certa quantidade de parâmetros utilizados pela função especificada. Segundo [CAR 99], apesar da semelhança com RPC (*Remote Procedure Call*) [BIR 84], as mensagens ativas têm um objetivo diferente: a regra principal é extrair suas informações e executar o mais breve possível o cálculo requisitado pela mensagem.

Existem basicamente dois métodos [CAR 99] de tratamento de mensagens ativas: `i)` por interrupção: ao chegar uma mensagem, o fluxo em execução é interrompido e a mensagem é tratada. Este tratamento consiste em identificar a função associada à mensagem, extrair todas as informações ou dados necessários, transferir essas informações e dados a alguma área de memória destinada para tanto e chamar a função identificada. `ii)` por *polling*: neste método, a aplicação é responsável por examinar a sua interface com a rede periodicamente de forma explícita. Neste exame, a existência de alguma mensagem

*Projeto Anahy - CNPq (55.2196/02-9)
†ITI/CNPq

é verificada e, caso exista alguma, ela é então tratada. Como no caso por interrupção, o tratamento consiste em extrair toda e qualquer informação necessária a alguma área de memória e chamar a função identificada.

Nos dois métodos, por interrupção e por *polling*, o tratamento é executado dentro do fluxo de execução ativo no momento, evitando custos adicionais de criação de um novo fluxo. Esta característica, segundo [CAR 99], é a grande vantagem trazida por este método de comunicação. O método utilizado para a aquisição de mensagens implica que o mecanismo de interrupção seja eficaz. Esse tipo de tratamento geralmente é utilizado em sistemas nativos ou até mesmo sistemas de tempo real. A eficácia das mensagens ativas está intimamente ligada à capacidade do sistema reagir à rede e às interrupções, ou seja, ligada a aspectos de programação que, por sua vez, são influenciados pelo nível técnico do programador e da qualidade das camadas de rede de um sistema operacional.

Biblioteca de Mensagens Ativas

Analisados os conceitos de mensagens ativas, é proposta uma implementação de uma biblioteca de comunicação baseada nestes conhecimentos. Esta biblioteca foi projetada tendo em vista atender inicialmente as necessidades de comunicação de Anahy [CAV 2003], um ambiente de desenvolvimento e execução de aplicações paralelas em aglomerados, através do método de tratamento de mensagens por *polling*.

A implementação, em sua versão atual, provê um mecanismo de envio, recepção e tratamento de mensagens ativas utilizando o serviço de *sockets* do sistema operacional GNU/Linux. A utilização de *sockets* objetiva menor carga de processamento necessária às funções da biblioteca e assim maior eficiência de comunicação, através da implementação somente dos serviços necessários ao seu objetivo de, inicialmente, como já mencionado, suprir as necessidades de Anahy. A interface com o usuário programador (API) provê primitivas de envio e recepção de mensagens no formato *act_msg_send()* e *act_msg_recv()*. Outras primitivas de criação e manipulação de mensagens ativas como *act_msg_create()*, *act_msg_init()*, *act_msg_pack()*, *act_msg_unpack()* e *act_msg_destroy()* também estão à disposição do programador. Para a manipulação direta dos *sockets* foi criada outra API para ser usada pela própria biblioteca de mensagens ativas, modularizando o código e facilitando a leitura, entendimento e possível modificação por pessoas não envolvidas diretamente em seu desenvolvimento, tendo primitivas do tipo *am_socket_create_connect()* e *am_socket_destroy()*. Esta modularização possibilita a separação do código específico para o conceito de mensagens ativas do código dependente do sistema operacional ou da sua camada de rede, facilitando assim sua manutenção e aprimoramento.

Resultados

Com o objetivo de analisar o comportamento da biblioteca de mensagens ativas implementada foram realizados testes de tempo de comunicação no formato cliente-servidor, em que o cliente requisita a realização de algum serviço por parte do servidor através do uso de mensagens ativas.

Os resultados de tempo vistos na Tabela 1, foram obtidos através da troca de mensagens entre duas máquinas de configuração Pentium 4 1.8 Ghz, 256Mb RAM, rede 10/100 Mbits Ethernet rodando GNU/Linux 2.4.20, interligadas por um *switch* 10/100 Mbits e conectadas a uma sub-rede da Universidade do Vale do Rio dos Sinos, estando portanto sujeitas a eventos da sub-rede, como acesso remoto por outros usuários ou algum

Tabela 1: Tempos de comunicação em milisegundos.

Mensagem em bytes	Lado da comunicação	Fibonacci(10)		Fibonacci(35)		Fibonacci(40)	
		Média	DesvPad	Média	DesvPad	Média	DesvPad
12	Cliente	0,91963	21,05442	235,01551	19,45725	2650,84875	87,64139
	Servidor	0,02679	0,228120	234,00528	7,946890	2649,02125	72,27328
	Diferença	0,89284	-	1,01023	-	1,8275	-
100	Cliente	0,39785	0,13942	236,21396	5,04173	2475,914	110,27892
	Servidor	0,04791	0,11930	235,84254	5,04254	2475,531	110,27823
	Diferença	0,34994	-	0,37142	-	0,383	-
Iterações	-	100.000		44.900		3.000	

tipo de intervenção do servidor de arquivos e NIS (*Network Information Server*) à qual também estavam conectadas. Durante a realização dos testes, não houve nenhuma espécie de acesso físico às máquinas.

O algoritmo recursivo de cáculo dos números de Fibonacci foi utilizado para simular carga computacional referente às funções associadas às mensagens ativas. Para tanto escolheu-se valores de Fibonacci que refletissem cargas pequenas, médias e grandes. Os resultados dos testes podem ser vistos na Tabela 1. Utilizou-se mensagens de 12 e 100 bytes, valores também escolhidos para refletir mensagens que carregam pouca ou bastante informação. Os tempos mostrados nas linhas *Cliente* foram tomados no intervalo que compreende o envio de uma mensagem e a espera de uma resposta relativa à mensagem tratada por parte do servidor. Esta espera pode ser uma confirmação de que o servidor tratou a mensagem ou mesmo alguma informação requisitada pelo cliente. A semântica da resposta está associada à função identificada na mensagem. Os tempos mostrados nas linhas *Servidor* foram tomados no intervalo que compreende a recepção da mensagem, seu tratamento de acordo com a função identificada e o envio de uma resposta, também de acordo com a função identificada. Os valores mostrados nas linhas *Diferença* correspondem à diferença média entre o tempos dos clientes e dos servidores, podendo ser associados ao tempo necessário para uma mensagem sair da fonte, trafegar pela rede e começar a chegar ao seu destino. Para a obtenção de cada média foram executadas pelo menos 3.000 iterações, como visto na linha *Iterações*.

Diferentemente do que se poderia esperar, os valores da média para as mensagens de 100 bytes são menores que os valores para as mensagens de 12 bytes. Este fato pode refletir a influência que o estado da máquina utilizada (sob possível interferência por parte da sub-rede mencionada) pode ter sobre o desempenho da comunicação. As médias para as colunas Fibonacci(10) e Fibonacci(40) também se apresentaram maiores na utilização de mensagens com 12 bytes do que com 100 bytes. Novamente, este fato pode ter sido influenciado pelo estado da máquina. Somente no caso de Fibonacci(35) a utilização de mensagens de 100 bytes obteve médias maiores. Outra particularidade é encontrada quando analisados os desvios padrões. Na coluna Fibonacci(10) notam-se valores consideravelmente maiores que as respectivas médias. A causa deste comportamento pode ser visualizada na Figura 1 B, onde é notado um intervalo de tempo praticamente constante por quase todas as 100.000 iterações, exceto em alguns poucos casos em que os intervalos medidos chegam a ser por volta de 6.500 vezes maior. Esta figura caracteriza o comportamento esperado ao longo do tempo para uma máquina em um estado não dedicado. Na Figura 1 A é visto o caso das mensagens de 100 bytes com Fibonacci(35) e nota-se uma

distância menor entre os picos de tempo e a média, fazendo com que o valor de seu desvio padrão fique por volta de 12 vezes menor que a média.

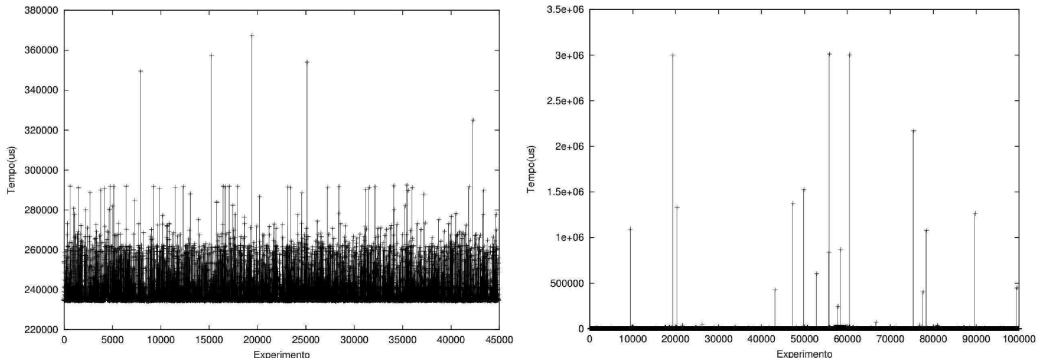

Figura 1: Tempos do cliente em microsegundos. A: utilizando mensagem de 100 bytes para Fibonacci(35). B: utilizando mensagem de 12 bytes para Fibonacci(10).

Conclusão

Uma biblioteca de comunicação aplicando o conceito de mensagens ativas foi implementada e é funcional. Sua execução em determinadas situações foi avaliada, tendo como resultados uma imagem de seu comportamento em ambientes compartilhados como a sub-rede utilizada para os testes. Esta avaliação levanta pontos específicos para trabalhos futuros em testes de tempo de comunicação entre máquinas exclusivamente dedicadas ao PAD, pontos, como o caso avaliado neste trabalho, do desvio padrão muitas vezes maior que a média correspondente. Novos testes possibilitarão comparações de comportamento em diferentes configurações de máquina e rede, gerando informações para novas versões aprimoradas. Em um futuro imediato, a biblioteca proposta neste trabalho será utilizada para implementar a versão distribuída de Anahy, provendo serviços como migração de tarefas e dados, possibilitando características como balanceamento de carga em um ambiente de memória distribuída como um aglomerado de computadores.

Referências

[BIR 84] BIRRELL, A. D.; NELSON, B. J. Implementing remote procedure calls. **ACM Transactions on Computer Systems**, v.2, n.1, p.39–59, Feb. 1984.

[CAR 99] CARISSIMI, A. S. **Le noyau exécutif athapascan-0 et l'exploitation de la multiprogrammation légère sur les grappes de stations multiprocesseurs**. 1999. Thèse de doctorat — Institut National Polytechnique de Grenoble, Grenoble, France.

[CAV 2003] CAVALHEIRO, G. G. H.; REAL, L. C. V.; DALL'AGNOL, E. C. Uma biblioteca de processos leves para a implementação de aplicações altamente paralelas. In: IV WORKSHOP EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO, 2003, São Paulo, SP. **Anais...** [S.l.: s.n.], 2003.

[EIC 92] EICKEN, T. von et al. **Active messages**: a mechanism for integrated communication and computation. University of California, Berkeley, CA 94720.