Biblioteca CE: Comunicação Eficiente

Márcia Cristina Cera, Daniela Saccol Peranconi,
Marcelo Pasin

Universidade Federal de Santa Maria - Laboratório de Sistemas de Computação
Faixa de Camobi, Km 9 - CEP 97105-900 - Santa Maria - RS - Fone/Fax: (55) 220 8523
{cera, dany, pasin}@inf.ufsm.br

Introdução

Bibliotecas de comunicação são ferramentas utilizadas para facilitar a implementação de programas paralelos ou distribuídos. Elas possuem rotinas e operações que tornam viável a comunicação entre os processos que compõem o programa paralelo. Cabe, ao programador, invocar tais rotinas e operações sem ater-se a implementá-las, muito menos a como se estabelecerá a comunicação.

Dentre as bibliotecas de comunicação, as mais difundidas são PVM (*Parallel Virtual Machine*) [SUN 90] e MPI (*Message Passing Interface*) [GRO 94]. Em ambas, a troca de mensagens é implementada através de primitivas de envio e recebimento (*send/receive*) de dados, possibilitando a cooperação entre os vários processos que compõem uma aplicação paralela.

A nossa proposta visa elaborar e conceber uma biblioteca de comunicação leve e eficiente. Leve por apresentar um conjunto reduzido de funções para estabelecer a comunicação entre processos e assim facilitar sua compreensão e uso. E eficiente para que, ao utilizá-la, atinjam-se bons níveis de desempenho.

CE

A CE é uma biblioteca de comunicação que foi desenvolvida objetivando proporcionar vantagens não apresentadas pelas demais bibliotecas. Ela foi idealizada sobre o conceito de RPC[1] (*Remote Procedure Call*) [BLO 92], ou seja, suas tarefas serão executadas remotamente. Este conceito foi empregado pois visa-se que a CE seja a biblioteca utilizada para a concepção de um mecanismo de execução remota de aplicações elementares de programas paralelos, fazendo uso de computadores pessoais ociosos.

Os canais de comunicação e os fluxos de execuções da CE são estabelecidos utilizando diretamente *sockets*[2] e *pthreads*[3] do padrão POSIX, oferecidos pelos sistemas operacionais atuais. Tal fato garante à biblioteca uma interação direta com o sistema operacional.

[1] Uma RPC nada mais é do que um conjunto de entidades (clientes) que precisam de um serviço e outro conjunto que executa serviços (servidores), havendo comunicação entre eles.

[2] canais de comunicação

[3] fluxos de execuções ou processos leves

Principais Funcionalidades

A CE oferece onze funções ao todo. No entanto, a lógica de funcionamento da biblioteca está representada em apenas duas delas, as quais são diretamente relacionadas ao processo de RPC. Uma delas, `ce_call()`, faz o lançamento remoto dos procedimentos e a outra, `ce_wait()`, é responsável pela certificação do final de uma execução remota. A título de facilidade, quando houver o lançamento de múltiplos procedimentos, pode ser utilizada a `ce_wait_any()`, que espera pelo retorno dos resultados de qualquer um dos procedimentos. Os procedimentos disponíveis à execução remota devem ser registrados previamente para que todos os processos tomem conhecimento de sua existência. Tal registro é feito através da função `ce_register()`.

A função que marca o início do uso da CE é a `ce_start()`. Esta função é responsável pelo estabelecimento da conexão entre os nós do sistema, pela criação de um controlador de porta para conexão e pelo lançamento do programa em todos os nós. Ao término da `ce_start()` o sistema está pronto para realizar execuções paralelas. Finalizando o uso da CE tem-se a função `ce_stop()`. Esta função garante que o ambiente de execução paralela só é desfeito quando todos os nós já executaram suas tarefas, impedindo assim uma finalização incompleta de um programa paralelo.

A fim de propiciar a sincronização das execuções dos vários processos que compõem um programa paralelo, a CE utiliza o mecanismo de barreiras de sincronização [TAN 02]. Tais barreiras são estabelecidas pela chamada da função `ce_barrier()`.

Implementação

A preparação do ambiente para realizar execuções paralelas ocorre através da chamada da função `ce_start()`. Um dos argumentos desta função é o número de nós total disponíveis, uma vez que a CE apresenta alocação estática [WIL 99]. Durante a chamada desta função, no caso do primeiro nó, ocorre o lançamento da aplicação nas demais máquinas. Quando estas executam a chamada `ce_start()` elas conectam-se ao primeiro nó e umas com as outras, de forma que ao final temos uma conexão todos com todos (figura 1).

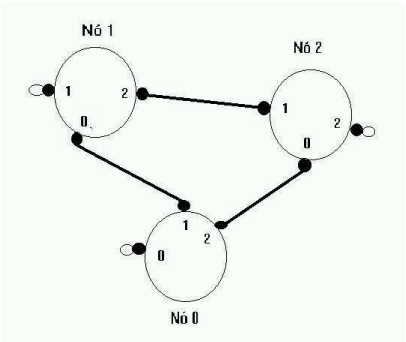

Figura 1: Exemplo de ambiente para execuções paralelas da CE com 3 nós.

Na CE, as RPC's são implementadas com duas mensagens ativas. Uma chamada `ce_call()` dispara uma mensagem ativa. Esta é composta por duas mensagens, uma com argumentos e outra que faz o lançamento de um determinado procedimento. Após executar, o procedimento dispara outra mensagem ativa, enviando, numa mensagem, os argumentos ou os resultados e, noutra, um sinal para a liberação de uma variável de

condição. O processo chamador de `ce_call()`, quando necessitar dos resultados do procedimento remoto, permanecerá bloqueado numa variável de condição, através da chamada da função `ce_wait()`, até que receba o sinal de liberação enviado pelo procedimento.

Análise de Desempenho

Os testes de desempenho da CE foram executados em máquinas que possuem, cada uma, dois processadores de 1 GHz Pentium III, 786 Mbytes de memória principal, 20 GB de disco rígido, 512 kbytes de memória cache. O sistema operacional utilizado foi o Red Hat Linux 7.2 com versão de núcleo de 2.4.18 smp. As máquinas comunicam-se através de um adaptador Gigabit Ethernet, modelo 3Com 3C996-T.

Foram realizadas análises sobre os tempos de comunicação, obtidos em execuções de *ping-pongs* entre duas máquinas, e tempo de processamento provenientes da execução de uma aplicação paralela. Num *ping-pong* mede-se o tempo que uma mensagem leva para sair de uma máquina e chegar a outra. E com a CE ele consiste de uma RPC onde ocorre a chamada de um procedimento que retorna imediatamente ao processo chamador. A aplicação paralela, utilizada aqui, encontra soluções para sistemas de equações integrais, e a mesma apresenta uma demanda de tempo de processamento relativamente grande. O problema utilizado nos testes visa encontrar as transições de fases das propriedades dos supercondutores em condições ambientes[PAS 01].

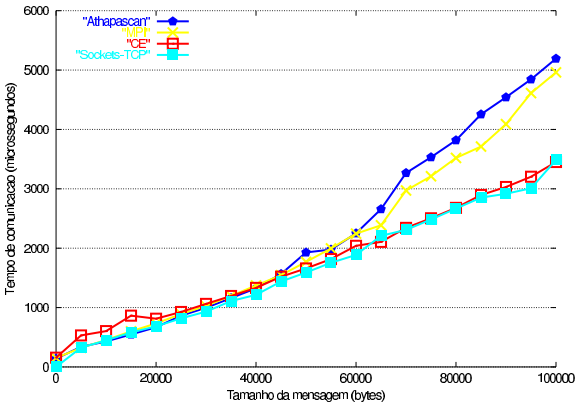

Figura 2: Gráfico do tempo de comunicação dos *ping-pongs*.

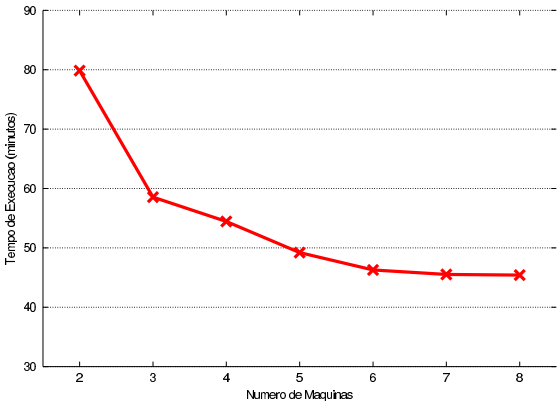

Figura 3: Gráfico do tempo de processamento da aplicação paralela

O gráfico da figura 2 mostra os tempos de comunição utilizando CE, *sockets*(TCP) [STE 90], MPI e Athapascan [BRI 97]. Nele percebemos que a CE apresenta um tempo de comunicação maior que os demais quando se trata de mensagens com poucos bytes (em torno de 30 Kbytes). Isso ocorre porque o tempo de lançamento de uma RPC é maior que o tempo necessário para simplesmente enviar uma mensagem como ocorre nos demais casos. Conforme aumenta-se o tamanho da mensagem, o comportamento da CE acompanha o de *sockets*. Estes, no entanto, apresentam melhores resultados por se tratarem de uma solução em mais baixo nível e porque a CE faz, adicionalmente, o lançamento de *threads*.

O gráfico da figura 3 mostra os tempos de processamento encontrados para um conjunto de dados iniciais, conforme a variação do número de máquinas empregadas na

paralelização. A execução seqüencial da aplicação tem um tempo de processamento de 118 minutos. Já com a utilização de duas máquinas o tempo de execução reduz para 80 minutos. Como pode ser visto no gráfico da figura 3, houve uma redução dos tempos de comunicação com a paralelização da aplicação.

Conclusão e Trabalhos Futuros

Neste trabalho almejou-se a construção de uma biblioteca de comunicação leve e de fácil utilização. Pelo fato dela estar fundamentada no conceito de RPC, sua lógica de programação baseia-se no uso de apenas duas funções. Uma, responsável pela chamada dos procedimentos remotos e outra responsável pela certificação do término de uma execução remota. O conjunto reduzido de funções oferecido pela CE, quando comparada as demais bibliotecas, facilita seu manuseio, além de proporcionar a implementação de códigos claros e sintéticos.

Uma análise comparativa mostrou que a CE possui tempo de comunicação menor que MPI e Athapascan para mensagens superiores a 30 kbytes. Para as mensagens inferiores a este valor o tempo total de comunicação recebe um acréscimo gerado pelo estabelecimeto das RPC's. O comportamento da vazão da CE acompanha o de *sockets* logo, sua vazão é maior que a das outras duas bibliotecas.

A partir da versão atual da CE pretende-se implementar um modelo de comunicação dinâmico e um sistema de chamada de procedimentos totalmente assíncrono. Tais altera-ções possibilitarão o uso da CE na implementação de um mecanismo para controle e alocação de computadores pessoais ociosos para a execução remota de aplicações elementares de programas paralelos.

Referências

[BLO 92] BLOOMER, J. **Power Programming with RPC**. 1.ed. O'Reilly & Associates, Inc. 1992.

[BRI 97] BRIAT, J.; GINZBURG, I.; PASIN, M.; PLATEAU, B. **Athapascan Runtime: Efficiency for Irregular Problems**. Euro-Par'97 Parallel Processing, 1997.

[GRO 94] GROPP, W.; LUSK, E.; SKJELLUM, A. **Using MPI: Portable Parallel Programming with the Message-Passing Interface**. 1.ed. MIT Press, 1994.

[PAS 01] PASIN, M.; PADOIN, E.L. **A parallel solution for systems of integral equations.** Anais SBAC-PAD, 2001.

[SUN 90] SUNDERAM, V. S. **PVM: a framework for parallel distributed computing.** *Concorrency, practice and experience*, 2(4):315-339, December 1990.

[STE 90] STEVENS, W. R. **Unix Network Programming.** 1.ed. Prentice Hall, 1990.

[TAN 02] TANENBAUM, A. S.; STEEN, M. van. **Distributed Systems - Principles and Paradigms**. 1.ed. Prentice Hall, 2002.

[WIL 99] WILKINSON, B.; ALLEN, M. **Parallel Programming - Techniques and Applications Using Networked Workstations and Parallel Computers**. 1.ed. Prentice Hall, 1999.