Desenvolvimento de um Núcleo de Comunicação Multithread para um Ambiente Heterogêneo

Iuri Vieria de Moraes, Jaqueline Nascimento da Cunha, Evandro Preuss

e-mail : {iurim, jaque, epreuss}@andros.inf.pucrs.br
Pontifícia Universidade Católica do Rio Grande do Sul
Instituto de Informática
Av. Ipiranga, 6681 prédio 16 térreo
CEP 90619-900 - Porto Alegre - RS - BRASIL
Telefone: (051) 339-1511 ramal 3211

Resumo

Esse trabalho apresenta um núcleo de comunicação desenvolvido que possibilita a execução de programas paralelos dentro de um ambiente heterogêneo e que permite a comunicação entre processos locais (que executam em um mesmo nodo) e distantes (que executam em nodos diferentes). Essas comunicações são realizadas utilizando-se um protocolo RPC (Remote Procedure Call) síncrono. A comunicação entre os núcleos distantes, componentes do ambiente de execução, é suportada por sockets. Uma característica importante do núcleo é o paralelismo obtido com a utilização de threads especializadas para a execução de suas diferentes funções. A linguagem de implementação utilizada foi o C (Borland 4.5) para Windows 95 e GCC para Linux, e o estágio atual envolve a elaboração de testes e depuração.

Abstract

In this work describe the communication kernel that has been implemented to permit the execution of parallel and distributed programms on heterogeneous environment. This communication can be done between local and remote process. This communication (developed with sockets) use RPC model protocol. An important characteristic of the kernel is the parallelism obtained with the usefulness of threads for execution of its diferent functions.

1. INTRODUÇÃO

Um modelo de programa paralelo pode ser materializado sob a forma de uma nova linguagem, por extensões acrescentadas a uma linguagem já existente ou por uma biblioteca paralela. Um ambiente de programação paralela inclui também ferramentas de auxílio à edição, depuração e vizualização de programas. O suporte básico do modelo de programa paralelo constitui-se no ambiente de execução, que pode ser formado por uma camada de software (núcleo) em cada processador (nodo) das máquinas paralelas. Os núcleos dos diferentes nodos cooperam para a realização dos serviços. Um componente fundamental do núcleo é o seu subsistema de comunicação (aqui denominado núcleo de comunicação), que permite a comunicação entre os diferentes nodos do sistema. O trabalho desenvolvido consta a implementação de um núcleo de comunicação para a plataforma PC, em Windows 95 e Linux, tendo em vista a construção de um ambiente de execução paralela para uma rede heterogênea de estações de trabalho.

2. O AMBIENTE DE EXECUÇÃO

O núcleo de comunicação é um componente fundamental num modelo de execução paralela e distribuída. O núcleo implementado é composto por duas camadas: protocolo e encaminhamento de mensagens. A camada protocolo implementa um protocolo cliente-servidor de base. A camada encaminhamento de mensagens é responsável pela recepção e envio de mensagens.

O núcleo de comunicação tem a função de receber todas as mensagens de requisição de serviços dos processos clientes, identificar o servidor a que se destina a mensagem, descobrir a localização do mesmo e enviar a requisição do serviço. Uma thread é então criada para enviar essa requisição ao servidor. Essa thread fica bloqueada esperando a resposta e quando a recebe, envia para o processo cliente que a solicitou.

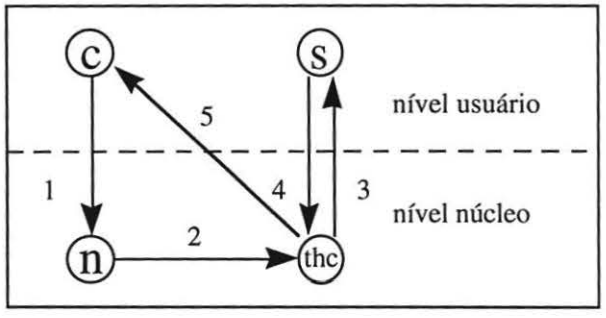
Além de atender todos os processos clientes locais e distantes, o núcleo também tem a função de registrar os processos servidores locais no servidor de nomes do sistema. Para isso, utiliza um serviço server_register, sendo este o único serviço atendido pelo núcleo. O processo servidor, quando inicializado, envia a requisição de serviço ao núcleo de comunicação local, que registra o servidor na tabela local e faz uma requisição ao servidor de nomes para o registro no sistema.

As threads do núcleo de comunicação trocam mensagens locais entre si e com os processos clientes e servidores. Se as mensagens forem distantes, as threads do núcleo local as enviam para um outro núcleo, que está num outro nodo da rede. Este núcleo distante faz as comunicações locais com os processos do nodo.

A comunicação entre os processos é feito através das primitivas send e receive implementados com o uso de sockets. As trocas de mensagens entre as threads do núcleo

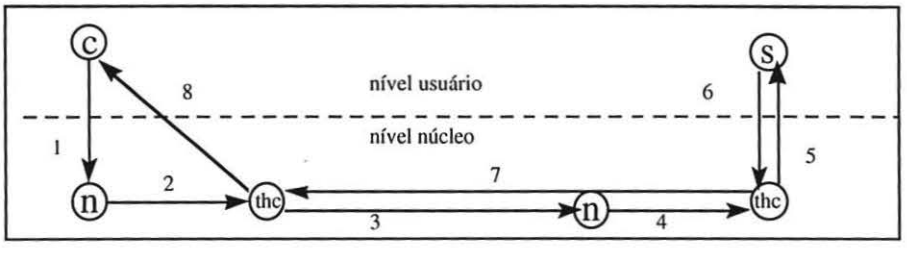
e entre os processos clientes e servidores, é feito através de fluxos de dados no domínio Internet com o uso do protocolo TCP. As figuras 2.1 e 2.2 ilustram, respectivamente, uma comunicação local e uma comunicação distante.

2.1 Comunicação Local



1. cliente envia uma requisição ao núcleo;
2. núcleo cria uma thread para tratar a requisição;
3. thread encaminha requisição ao servidor ;
4. servidor envia a resposta à thread do núcleo;
5. thread envia o resultado ao cliente.

2.2 Comunicação Distante



1. cliente envia uma requisição ao núcleo;
2. núcleo cria thread para tratar a requisição;
3. thread encaminha requisição ao núcleo distante;
4. núcleo cria uma nova thread para tratar a requisição;
5. a thread encaminha requisição ao servidor e aguarda o resultado;
6. servidor envia resultado à thread do núcleo;
7. a thread do núcleo encaminha resultado para a thread que fez a requisição;
8. a thread do núcleo encaminha resultado ao cliente que é acordado.

3. CONCLUSÃO

A concepção e a implementação do núcleo, bem como a integração com o núcleo para o ambiente Unix, estão concluídos. Estão sendo construídos programas servidores para suportar a programação e execução de programas que explorem o paralelismo em um ambiente heterogêneo.

Como trabalho futuro pretende-se:

- Implementar em outras plataformas(OS/2, Windows NT);
- Implementar mecanismos de tolerância à falhas.

4. BIBLIOGRAFIA

1. BIRREL, A. D. and NELSON, B. J. Implementing Remote Procedure Call. *In ACM Trans.Computer System*, Vol2, N° 1, Feb 1984.
2. COSTA, Celso Maciel da; FAVRE, Michel and BRIAT, Jacques. Implementação de um Mecanismo de RPC em uma Máquina Paralela sem memória Comum. *Laboratoire de Génie, Informatique*, Grupo PLoSys, Fr.1993.
3. COSTA, Celso Maciel da. Microkernel Paralelo: Concepção e Implementação em uma Máquina sem Memória Comum. *25 JAI, Buenos Aires, Argentina, set,* 1996.
4. DUMAS, Arthur. Programando Winsock. *Axcel Books*. 1995.
5. PETZOLD, Charles. Programming Windows 95. *Microsoft Press*. 1996.
6. PREUSS, Evandro. Trabalho Individual I. *Mestrado PUCRS*. 1996.
7. STEVENS, Richar W. Unix Network Programming. *Englewood Clifs,NJ: Pretice Hall*,1990.
8. TANENBAUM , Andrew S. Modern Operating Systems. *Pretince-Hall.* 1992.
9. TANENBAUM, Andrew S. Distributed Operating Systems. *Pretince-Hall.* 1985.