MDX: Um Ambiente de Programação Paralela com Suporte a Troca de Mensagens e a Memória Compartilhada Distribuída[3]

Celso M. da Costa, Getúlio A dos Santos, Alessandro Copetti, Rafael Scopel
Faculdade de Informática da PUCRS
Av. Ipiranga, 6681 - 90619-900 - Porto Alegre, RS, Brasil
Fone: +55 51 320-3611
E-Mail:{celso,gas, scopel, copetti}@inf.pucrs.br

1. Introdução

Este trabalho apresenta o projeto de um ambiente de programação paralela chamado MDX, que oferece suporte à programação paralela com Troca de Mensagens e com Memória Compartilhada Distribuída. O sistema foi projetado para funcionar em um cluster de workstations, está sendo desenvolvido no sistema operacional Linux, suporta programas paralelos escritos em C e C++ e executa em redes IP e ATM nativo. Atualmente existe um protótipo operacional, resultado de vários trabalhos desenvolvidos no Programa de Pós-graduação em Ciência da Computação da PUCRS.

2. Objetivos do Projeto

Os objetivos do projeto MDX englobam:

- criar um ambiente para ensino e pesquisa de na área, propiciando melhoria da qualidade do ensino de graduação e pós-graduação;
- formar mão de obra qualificada para atuação na área, carente de profissionais capacitados;
- disponibilizar à comunidade científica uma ferramenta de programação paralela fácil de usar e com um conjunto reduzido de primitivas;
- fortalecimento e fomento da área de pesquisa na **PUCRS**.

3. Visão Geral do MDX

O sistema foi concebido como uma biblioteca de funções e suporta um conjunto de conceitos e operadores paralelos que permitem a criação dinâmica de tasks e threads, portas e barreiras de sincronismo e de dados compartilhados.

MDX segue a abordagem cliente/servidor. Os serviços do sistema são oferecidos por servidores especializados. Um programa cliente solicita um serviço a um servidor especializado enviando os parâmetros apropriados. O cliente é suspenso enquanto o servidor está executando a requisição. Ao final da execução, os resultados são retornados ao cliente, que é acordado e continua sua execução.

[3] Project partially supported by FAPERGS.

O núcleo de comunicação oferece primitivas para os clientes e para os servidores tais que os clientes podem enviar requisições transparentemente para servidores locais e remotos, e recuperar os resultados. As mensagens destinadas a um servidor são armazenadas na fila do servidor. Para processar uma requisição o servidor recupera a primeira mensagem da fila, e se bloqueia caso a fila esteja vazia.

Dois processos do núcleo são responsáveis pela recepção e o envio de mensagens. Um processo faz a recepção das mensagens dos clientes locais, localiza o servidor e envia a mensagem. O outro processo do núcleo recebe as mensagens enviadas remotamente, identifica o servidor destinatário e a coloca da fila de requisições do servidor.

4. Resultados do projeto

Dentre os resultados destacam-se:

- criação de um ambiente de programação paralela com suporte a troca de mensagens e a memória compartilhada distribuída, que deverá ajudar no desenvolvimento, depuração, testes e execução de outras ferramentas, além de suportar diretamente inúmeros trabalhos práticos propostos em disciplinas de graduação e pós-graduação;
- assimilação e desenvolvimento de novas tecnologias para suporte a programação distribuída e paralela. Os benefícios desse conhecimento são grandes, uma vez que esses temas são extremamente atuais e ainda não completamente resolvidos;
- desenvolvimento de dissertações de mestrado, trabalhos de graduação, publicação de artigos e relatórios técnicos.

5. Conclusão

O ambiente MDX está em desenvolvimento desde 1997, sendo que um protótipo está operacional. Este protótipo é o resultado de duas dissertações de mestrado e de um trabalho de conclusão de curso de graduação. Em uma das dissertações foi definido e implementado o ambiente de suporte de execução, permitindo a programação paralela com o uso de memória compartilhada distribuída. A outra dissertação de mestrado se dedicou a avaliação e ao porte do MDX para uma rede ATM, com o uso de um driver específico para permitir diretamente o uso dos recursos oferecidos por esta tecnologia de rede. No trabalho de conclusão foi desenvolvida a ferramenta de programação paralela com suporte a troca de mensagens (TCX).

Existem dois trabalhos em andamento: uma dissertação de mestrado e um trabalho de iniciação científica. Na dissertação de mestrado está sendo concebido e implementado um algoritmo de balanceamento de carga. No trabalho de iniciação científica está sendo feita a avaliação da ferramenta de programação paralela TCX.

Os trabalhos futuros incluem o porte do MDX para uma rede rápida (SCI ou Myrinet) e o desenvolvimento de mecanismos de monitoração e visualização de programas paralelos.

Referências

1. A. S. Tanenbaum, M. Franz Kaashoek and Henri Bal, (1992). E. Parallel Programming Using Shared Objects and Broadcasting. In IEEE Computer, August 1992.
2. Anne Dinning (1989). A Survey of Synchronization Methods for Parallel Computers. In IEEE Computer, July1989, pp. 66-77.
3. Celso Maciel da Costa (1993). Environnement D´éxécution Parallèle: Conception et Architecture. *In.* Laboratoire de Génie Informatique, Université Joseph Fourier, Grenoble France. Thése de Doctorat, 1993.
4. David Gelernter (1985). Generative Communication in Linda. In ACM Transactions on Programming Languages and Systems, Vol. 7, No. 1, January 1985.
5. Gregory R. Andrews (1991). Paradigms for Process Interaction in Distributed Programs. In ACM Computing Surveys, Vol. 23, No. 1, March 1991, pp. 49-90.
6. H. E. Bal., M.F. Kaashoek and A.S Tanembaum (1992). Orca: A Language For Parallel Programming of Distributed Systems. In IEEE Transactions on Software Engineering , Vol. 18, No.3, March 1992 pp. 190-205.
7. Henry E. Bal, Jennifer G. Steiner and A. S. Tanembaum (1989). A.S. Programming Languages for Distributed Computing Systems. In ACM Computing Surveys, Vol.21, No.3, September 1989, pp. 261-322.
8. Ian Foster (1995). Designing and Building Parallel Programs. *In* Addison-Wesley Publishing Company, Inc. 1995.
9. J. Briat, M. Favre et Celso Maciel da Costa (1992). Un noyau de système de exploitation pour une machine parallèle sans mémoire commune. In 4es Rencontres du Parallelisme, Université des Sciences et Technologies de Lille, Villeneuve d`Asc, 18-20 Mars 1992.
10. N. H. Gehani and W. D. Roome (1986). Concurrent C. In IEEE Software Practice and Experience , Vol. 16(9), Septembre 1986, pp. 821-844.
11. N. H. Gehani and W. D. Roome (1989). The Concurrent C Programming Language. In Prentice Hall, 1989.
12. N. H. Gehani and W. D. Roome (1992). The Implementing Concurrent C. In IEEE Software-Practice and Experience , Vol. 22(3), March 1992, pp. 265-285.
13. Nicholas Carriero and David Gelernter (1989). How to write parallel programs: A Guide to Perplexed. In ACM Computing Surveys, Vol.21, No. 3, September 1989.
14. Ralph Butler and Ewing Lusk (1992). User`s guide to the p4 Programming system. In ANl-92/17-Mathematics and Computer Science Division, Argonne National Laboratory, October 1992.
15. Raymond Namyst, (1996). PM^2: un environnement pour unce conception protable et une exécutin efficace des applications parallèles irregurlières. Laboratoire d'Informatique Fondamentale de Lille, Lille, Dez. 1996.
16. Robert F. Cmelik, N.H. Gehani, and E.D. Roome (1989). Experience with Multiple Processor Versions of Concurrent C. In IEEE Transactions on Software Engineering, Vol. 15, No. 3 March !989, pp. 335-344.
17. Willy Zwaenepoel et al (1996). TreadMarks: Shared Memory Computing on Network of Workstations. IEEE, Fev 1996. pp. 18-28.