Implementação de uma Arquitetura Paralela usando Computadores Pessoais e Linux

Alexandre Munaretto *amunaretto@hotmail.com*
Rômulo Rieder *romulorieder@via-rs.net*
Giovani Lazzari *giovani@morlass.com.br*
Conrado Ruch Junior *conrado@uricer.edu.br*
Alexandro M. S. Adário *adario@uricer.edu.br*
Universidade Regional Integrada do Alto Uruguai e das Missões – Campus de Erechim
Av. 7 de Setembro, 1621 – Erechim/RS, Fone:(54)520-9000, Fax: (54)520-9090

Introdução

Este trabalho descreve a construção de um cluster Beowulf utilizando o sistema operacional GNU/Linux realizada por alunos do Curso de Ciência da Computação da URI – Campus de Erechim. O objetivo principal foi a implementação de uma arquitetura paralela para a divulgação e mostra no meio acadêmico das vantagens da utilização de um cluster para o processamento de alto desempenho. Todo o processo de configuração do cluster Beowulf foi detalhadamente documentado, objetivando a geração de material didático para auxílio na construção destas arquiteturas. Como testes de funcionalidade, foi realizada a programação e a execução de programas paralelos explorando os recursos da arquitetura. Para isso, foi utilizada a implementação gratuita MPICH da biblioteca MPI. Os processos de configuração dos computadores, instalação da biblioteca e os testes realizados são apresentados nesse trabalho.

O Projeto Beowulf

Em 1994, Thomas Sterling e Donald Becker, do CESDIS (*Center of Excellence in Space Data and Information Sciences*) da NASA, interligaram 16 computadores 486 DX-4 100 Mhz, usando o sistema operacional GNU/Linux e uma rede Ethernet. O cluster, batizado de Beowulf [BEO03], foi sucesso instantâneo na comunidade mundial, por satisfazer a necessidade de poder computacional demandada por universidades e comunidades de pesquisa, baseando sua construção em computadores de baixo custo, muitas vezes considerados obsoletos para uso individual. O esforço de desenvolvimento do primeiro Beowulf cresceu rapidamente e originou o Projeto Beowulf.

Uma característica chave do Beowulf é o sistema operacional usado, de desempenho elevado e gratuito, sobre o qual podem ser facilmente instaladas ferramentas que viabilizam o processamento paralelo, como as bibliotecas MPI e PVM para troca de mensagens.

Em resumo, um Beowulf é constituído por nós escravos controlados por um computador principal (ao qual se tem acesso direto), ligados por uma LAN. O mestre (*front-end* ou controlador) controla o cluster, monitora e distribui as tarefas, atua como servidor de arquivos e faz o elo entre usuários e o cluster. Os escravos (clientes ou *back-ends*) são dedicados ao processamento das tarefas enviadas pelo controlador. Eles não necessitam de teclados, monitores ou discos rígidos, pois podem realizar boot e login remotos.

Construção do Cluster Beowulf

Para a construção do cluster foi utilizada a distribuição Slackware, versão 9, do sistema operacional GNU/Linux, com kernel versão 2.4.2.. O cluster montado utilizou 30 computadores pessoais, sendo 1 controlador e 29 nós escravos, interligados em rede Ethernet, cuja estrutura está descrita na Tabela 1.

Tabela 1 - Configuração/Estrutura dos Componentes do Cluster

Item	Configuração
Computador Mestre	Pentium IV 2.4 GHz, 256 MB de RAM, HD Maxtor 40 GB, 7200 rpm, cabo IDE ATA 133 Mbps, Placa de Rede SIS 900 10/100 Mbps
Computador Escravo	Pentium IV 2.4 GHz, 256 MB de RAM, HD Maxtor 40 GB, 7200 rpm, cabo IDE ATA 133 Mbps, Placa de Rede SIS 900 10/100 Mbps
Rede Local	Rede Ethernet 10 Mbps, Switch 3Com 10/100 Mbps 24 portas, Hub Encore 10 Mbps 8 portas, Cabos Multi-LAN com conectores RJ-45

Após a instalação do sistema operacional foram definidos os parâmetros de rede apresentados na Tabela 2.

Tabela 2 - Configurações de rede dos nós do cluster

Computador	**Mestre**	**EscravoX***
IP	192.168.1.1	192.168.1.X*
Máscara de Rede	255.255.255.0	
Nome	mestre.cdi.uri.com.br	escravoX.cdi.uri.com.br*

* cada nó escravo foi identificado por um número, onde X representa um valor de 2 a 30

Além da configuração de rede necessária ao funcionamento do cluster, é preciso alterar alguns arquivos de sistema do Linux, conforme apresenta a Tabela 3.

Tabela 3 - Inclusões de parâmetros nos arquivos de sistema

Arquivos	Linhas adicionadas

`/etc/hosts`	`192.168.1.1 mestre mestre.cdi.uri.com.br`
	`192.168.1.2 escravo1 escravo1.cdi.uri.com.br`
	`192.168.1.3 escravo2 escravo2.cdi.uri.com.br`
	`...`
	`192.168.1.30 escravo29 escravo29.cdi.uri.com.br`

`/etc/hosts.equiv`
`/home/.rhosts`
`/home/.rhosts`
	`mestre`
	`escravo1`
	`escravo2`
	`...`
	`escravo29`

`/etc/exports`	`/home escravo (rw, no_root_squash)` `/usr escravo (rw, no_root_squash)`
`/etc/fstab`	`192.168.1.2:/home /home nfs exec,dev,suid,rw 1 1` `192.168.1.2:/usr /usr nfs exec,dev,suid,rw 1 1`
`/etc/securetty`	`rsh` `rlogin`
`/etc/inetd.conf`	Obs.: Neste arquivo devem ser habilitados os serviços `shell`, `login`, `exec`, `klogin`, `eklogin`, `kshell`
`/etc/ssh/sshd_config`	`PermitEmptyPassword yes`

O arquivo `/etc/hosts` serve para resolver o nome da máquina no endereço lógico de rede. O arquivo `/etc/hosts.equiv` define a relação de confiança entre os hosts através de equivalência, sem a necessidade de autenticação por senha, o que significa um risco

de segurança, mas é requerido pelo protocolo RSH. Os arquivos `/home/.rhosts` e `/root/.rhosts` também são usados pelo RSH para execução de comandos remotos e aplicações de monitoramento. No nó mestre foi configurado o sistema de arquivos NFS (arquivo `/etc/exports`), permitindo o acesso transparente a discos remotos e centralizando a administração e o compartilhamento de diretórios na rede. A configuração dos clientes NFS é feita pelo arquivo `/etc/fstab` Em todas as máquinas foram ainda habilitados os serviços **shell**, **login**, **exec**, **klogin**, **eklogin**, **kshell**, editando o arquivo `/etc/inetd.conf`. O arquivo `/etc/ssh/sshd_config` permite habilitar o acesso aos nós remotos sem senha.

Biblioteca MPI

Para a realização deste trabalho foi utilizada uma implementação do padrão MPI-2, chamada MPICH. Essa implementação surgiu em 1993 e sua última atualização data de 2002, na versão 1.2.4, e é considerada uma das únicas implementações existentes que combinam portabilidade e alto desempenho. A configuração é relativamente simples e necessita apenas dos seguintes passos:

- Configuração e instalação:

```
./configure –prefix=/usr/local/mpich
make
make install
```

- Edição de `/usr/local/mpich/util/machines/machines.LINUX`, incluindo os nomes dos nós, no mesmo padrão do arquivo `/etc/hosts.equiv` (vide Tabela 3)

Implementações realizadas

Para testes no cluster foram utilizados três algoritmos escritos em C ANSI utilizando a biblioteca MPICH para a troca de mensagens. No primeiro, o mais simples, cada nó escravo realizava o cálculo do valor de PI e comparava com o valor 3.14159265358979. No segundo, um processo mestre recebia um valor para cálculo do seu fatorial, distribuía a tarefa entre os nós escravos, obtinha as respostas do processamento e apresentava o resultado final da operação. Nestes dois algoritmos pretendeu-se comprovar a funcionalidade do cluster montado e demonstrar a utilização da programação paralela.

A terceira implementação objetivou a análise do processamento de um programa de cálculo de quantidade de números primos existentes entre 1 e um valor informado, explorando os modelos de comunicação síncrona e assíncrona. Para isso, foram produzidos dois programas para testes, um para cada modelo.

O programa destinado a testar a comunicação assíncrona realizou o envio de mensagens para os nós escravos, a partir do mestre, sem a necessidade de confirmação instantânea do recebimento destas, permitindo a continuidade de execução do programa. A tarefa foi distribuída de maneira que o nó controlador pudesse gravar valores do intervalo (entre um e o valor informado) no buffer de recebimento das máquinas escravas. Cada cliente, a partir da leitura dos valores realizou o processamento para identificar se os mesmos eram números primos, retornando ao buffer do nó controlador o resultado. O mestre, verifica em seu buffer estes resultados calculados e imprime.

Na comunicação síncrona, o mecanismo de controle baseia-se no envio e confirmação do recebimento das mensagens. A máquina mestre realizou o envio de um ou mais valores do intervalo aos nós escravos e aguardou a confirmação do recebimento. Quando a confirmação retorna, cada nó escravo pode iniciar a execução do cálculo para identificação. Após o processamento, os clientes retornam os resultados para o controlador, que confirma o recebimento para cada escravo, e então exibe os resultados.

Conclusão

Com a construção do cluster conseguiu-se demonstrar que é possível, fácil e eficiente desenvolver computadores poderosos a partir de outros mais simples. O sistema operacional e as ferramentas utilizadas, de livre distribuição, foram fundamentais para a concretização da implementação.

A execução de algoritmos paralelos, dividindo o processamento na rede, permitiu a validação da arquitetura. A análise de processamento entre os modelos de comunicação permitiu verificar que o tempo de processamento utilizando comunicação assíncrona é menor, por não haver espera pela confirmação de recebimento de mensagens.

A documentação apresentada, do processo de configuração dos nós, visa auxiliar a construção de outros clusters Beowulf, haja vista as diferenças entre distribuições do sistema GNU/Linux, o que dificulta uma padronização de instalação e configuração do sistema, e a escassez de bons tutorias em nosso idioma.

O Beowulf mostra-se um sistema que pode ser aperfeiçoado constantemente graças à evolução da tecnologia dos computadores pessoais, à facilidade de aquisição dos componentes que o compõem, bem como a redução de seus custos. Outro detalhe importante é que mesmo mudando processadores, tecnologias de rede, e custos relativos de equipamento, o modelo de programação não se altera, o que, na prática, significa compatibilidade futura para os programas.

Referências

[POG00] POGORZELSKI, S.; PARABONI, R. **Beowulf – Implementação de um Arquitetura Paralela usando Computadores Pessoais e Linux**. Erechim: Universidade Regional Integrada do Alto Uruguai e das Missões, 2000.

[TEI00] TEIXEIRA, H. E. G.; BARROS, M. J. A. S.; COELHO, P. J. M. **Clusters Beowulf**. Faculdade de Engenharia da Universidade do Porto, Portugal, 2000.

[COS02] COSTA, C. M.; STRINGHINI, D.; CAVALHEIRO, G. G. H. **Programação Concorrente: Threads, MPI e PVM**. In: Escola Regional de Alto Desempenho, 2002, São Leopoldo, Rio Grande do Sul, Brasil. p 31-65.

[PIT02] PITANGA, M. **Construindo Supercomputadores com Linux**. Rio de Janeiro: Brasport, 2002.

[LIN03] LINUX CLUSTER. **Linux Cluster**, Engenharia de Redes de Comunicação, LabRedes - Universidade de Brasília, 2003. Disponível via WWW: http://cluster.redes.unb.br/cluster/ (acessado em Jun/2003)

[BEO03] BEOWULF.ORG. **The Beowulf Cluster Site**, 2003. Disponível via WWW: http://www.beowulf.org/ (acessado em Jun/2003)

[MPI03] MPI. **The Message Passing Interface Standard**, 2003. Disponível via WWW: http://www-unix.mcs.anl.gov/mpi/ (acessado em Jun/2003)