Projeto e Implementação de um Cluster Beowulf

Cândice Corradi Kindel, Jorge Alberto Messa Menezes Júnior, Marcus Kindel, Robinson Pizzio

Pontifícia Universidade Católica do Rio Grande do Sul – Câmpus Uruguaiana - RS
BR 472 Km 7 – Cx. Postal 249 CEP 97500-970 Uruguaiana-RS, Telefone – (0**55) 4131515/, Fax – (0**55) 4131280
candy@pucrs.campus2.br, menezesjr@pucrs.campus2.br, kindel@pucrs.campus2.br, rpizzio@pucrs.campus2.br

Introdução

A demanda por grande capacidade de processamento é cada vez maior, tornando-se imprescindível à utilização de máquinas mais velozes.
Essa capacidade de processamento, até meados da década de 90, só era alcançada com o uso de supercomputadores, que por sua vez eram máquinas caras, com softwares proprietários e de uma manutenção complexa e cara.
Foi então que surgiu uma alternativa aos supercomputadores da época, que consistia na utilização de um aglomerado ou cluster de computadores rodando software de código aberto e funcionando como uma arquitetura paralela.

Tipos de Clusters

Basicamente existem três tipos de clusters:
Pra computação de alto desempenho (HPC-High *Performance Computing*): têm como principal objetivo atingir um alto desempenho para um determinado tipo de aplicação especialmente desenvolvida para processamento paralelo.
Para alta disponibilidade (HA - *High Availability*): têm como função manter um determinado sistema rodando permanentemente.
Para balanceamento de carga (HS – *Horizontal Scaling*): não há processamento paralelo, mas sim distribuição de processos individuais entre os componentes do sistema. Logo se trata de um sistema onde a disponibilidade de um determinado serviço é grande.

O cluster Beowulf

O primeiro *Cluster Beowulf* [BEO 03] foi desenvolvido no início da década de 90, pelos pesquisadores Donald Becker e Thomas Sterling do *NASA Goddard Space Flight Center* [DUN 03]. Eles precisavam aumentar sua capacidade de processamento, mas os recursos para financiar os seus projetos estavam diminuindo.
Desenvolveram então o seu próprio supercomputador e o denominaram de *Cluster Beowulf*, que consiste de um sistema de computação paralelo composto de vários computadores autônomos (nodos), cada um executando seu próprio sistema operacional e que se comunica com os demais através de uma rede de interconexão. Dentre os nodos, um deles é considerado mestre, sua função é gerenciar os outros nodos (escravos) e distribuir o processamento entre eles, no projeto em questão o nodo mestre funciona também como *gateway* para a rede da Universidade.
A distribuição de processamento consiste da divisão de uma tarefa em várias partes e o despacho destas para os nodos escravos do cluster executarem. Tornando assim, em condições ideais, o tempo de resposta muito menor do que o de uma máquina apenas.Este aglomerado de máquinas comporta-se, do ponto de vista do usuário, como se fosse uma única máquina.

A filosofia de um *Cluster* do tipo *Beowulf* é de ser um projeto com baixa relação custo/desempenho, pois são utilizados softwares de código aberto, que permitem uma melhor adaptação dos sistemas a serem utilizados e não há uma dependência de hardware, pois é possível implementar um *Cluster Beouwlf* com praticamente qualquer equipamento existente hoje no mercado, logo, nenhum componente precisa ser desenvolvido especificamente para o projeto.

O Projeto

O projeto em questão tem como objetivo ser usado de forma acadêmica, no ensino de disciplinas relacionadas com arquiteturas paralelas e processamento distribuído.
Considerando as limitações do hardware e principalmente a largura de banda, se espera um desempenho modesto do *Cluster*. Entretanto do ponto de vista acadêmico, este é um aspecto secundário, pois as configurações e análises não dependem da capacidade de processamento absoluto do sistema.
O projeto encontra-se hoje em fase de construção e execução de aplicações paralelas, onde será verificado o desempenho inicial do C*luster*, ainda sem qualquer tipo de ajuste ou otimização deste.

Recursos de hardware e software

Hardware

O protótipo em desenvolvimento usa equipamentos que estavam sendo descartados pelo CPD da Universidade e foram doados ao projeto. O nodo mestre é um Pentium MMX 166 MHz, com 80 MBytes de RAM e disco rígido de 30 GBytes. Os demais nodos escravos seguem um mesmo padrão, sendo Pentium 100 MHz, com 32 MBytes de RAM e os discos rígidos variam entre 500 MBytes e 8 GBytes.
A comunicação entre as máquinas do *Cluster* é feita através de um *hub* TRENDnet de 24 portas e 10Mbps.
Cada máquina com exceção do nodo mestre que possui duas placas de rede (uma para conectar a rede do C*luster* e outra para conectar a rede da universidade) possuem apenas uma placa de rede de 10Mbps.

Software

Durante a especificação do projeto, decidiu-se usar o sistema operacional GNU/LINUX [GNU 96], apesar de outros sistemas de código aberto baseados em UNIX [UNI 95] estarem disponíveis, principalmente pela maior disponibilidade de documentação. Depois de alguns testes, foi escolhida a distribuição Slackware [SLA 03], que se adaptou melhor aos escassos recursos de hardware disponíveis. Além disso, o Slackware permite maior flexibilidade na instalação de seus componentes.
Os serviços básicos necessários no nodo mestre são:

- Programas de avaliação de desempenho (*benchmarks*);
- Bibliotecas de programação paralela (PVM [GEI 03] ou MPI [GRO 03]);
- Serviços de rede (TCP/IP, RPC, etc);
- Compiladores e suas bibliotecas (C, C++, Fortran).
- Protocolos para compartilhamento de arquivos (NIS [KUK 03], NFS [NFS 03]);

Os nodos escravos são apenas clientes Linux, com uma instalação do sistema operacional mínima, protocolos de rede e módulos necessários para o processamento distribuído.

Rede de comunicação

Neste projeto será utilizada a topologia em estrela. Uma topologia estrela consiste de vários computadores interligados através de um concentrador (*switch* ou *hub*) a uma máquina servidora.

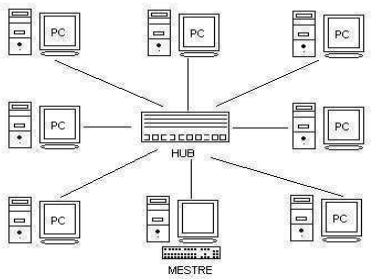

Fig. 1. Topologia estrela

Programação distribuída

A divisão, comunicação e sincronização entre os processos no *Cluster*, que residem em memórias distintas e que são executados em processadores de máquinas diferentes ficam a cargo dos protocolos de passagens de mensagens.
No projeto serão testados dois ambientes muito utilizados para aplicações paralelas, o MPI (*Message Passing Interface*) e o PVM (*Parallel Virtual Machine*).
PVM e MPI são bibliotecas de subrotinas de comunicação que permitem escrever programas que utilizem passagem de mensagens para rodar em sistemas distribuídos.
PVM surgido em 1989 nos laboratórios da *Emory University e Oak Ridge National Laboratory* [FRE 99], e seu objetivo inicial era criar e executar aplicações paralelas.
MPI teve sua primeira versão publicada em 1992 e foi desenvolvido para ser o padrão na comunicação entre processos em ambientes de memória distribuída, define um conjunto de rotinas para facilitar a comunicação (troca de dados e sincronização) entre processos em memória, sendo portável para praticamente qualquer arquitetura, tem aproximadamente 125 funções para programação e ferramentas para análise de desempenho.

Paralelização de programas

Neste projeto a paralelização dos programas será feita explicitamente. Ou seja, o trabalho de paralelizar o software é de responsabilidade do programador, que deve se preocupar com a sincronização dos processos, garantindo assim a correta ordem de execução.

Análise do desempenho

Nessa fase do projeto serão feitas as análises do desempenho do C*luster*. Será avaliado o desempenho da rede, da comunicação entre os nodos e do C*luster* como um todo. Esses parâmetros serão obtidos e analisados utilizando-se conjuntos de aplicações conhecidos como *benchmarks*.

Os objetivos serão determinar o limite superior de desempenho do sistema, a fim de estabelecer uma comparação com outros resultados publicados, e investigar o desempenho de programas e algoritmos na arquitetura implementada. Em relação ao último, serão realizados ensaios a fim de determinar o *speedup* e a eficiência para um número de variável de nodos.

Esse procedimento será usado também para otimizar o desempenho do sistema desenvolvido, bem como para identificar os componentes que limitam o desempenho global.

Benchmarks a serem utilizados:

- NetPerf : mede o desempenho na transferência de dados para os protocolos TCP e UDP.
- NetPipe : fornece informações como largura da banda, vazão e tempo de transmissão.
- Linpak: testa o desempenho do sistema resolvendo um problema de sistemas lineares.
- LMBench : executa testes que avaliam a latência do sistema, como carga de memória, escritas e leituras no disco e *overhead* em trocas de contexto.
- NAS: mede o desempenho de um conjunto de aplicações paralelas.

Fase de ajustes

Baseado nos resultados obtidos na fase de avaliação do desempenho será realizada uma série de testes e mudanças de configuração a fim de maximizar o desempenho do sistema.
Entre estes ajustes está a compilação do *Kernel* usando apenas os módulos que são realmente essenciais para a operação do sistema, a fim de minimizar o uso de memória pelo sistema operacional.

Conclusão

A utilização de *Clusters* de computadores na área de computação de alto desempenho é cada vez maior. Hoje grande parte dos supercomputadores é baseada em diferentes tipos de C*lusters*. O uso de C*lusters Beowulf* se dá em função da facilidade de implementação associada a um custo relativamente baixo.
Para um C*luster* alcançar um bom desempenho, vários fatores devem ser levados em conta, tais como: a escolha de um sistema operacional que possa ser adaptado às reais necessidades do sistema; uma estrutura de rede que proporcione um bom desempenho na comunicação entre as máquinas; e configuração de todo o conjunto a fim de obter o melhor desempenho possível.
A principal contribuição deste trabalho é disponibilizar um sistema de processamento distribuído para uso acadêmico, estimulando o desenvolvimento de aplicativos para esse ambiente no ambiente da Universidade.

Referências

[BEO 03] BEOWULF.ORG. **The Beowulf Cluster Site, Scyld Computing Corporation, 2000 - 2003**. Acessado em 10 de julho de 2003. Disponível on-line: http://www.beowulf.org/.

[DUN 03] DUNBAR, B. **NASA, 2003**. Acessado em 20 de julho de 2003. Disponível on-line: http://www.nasa.gov/.

[GEI 03] GEIST, A. et al. **PVM**: parallel virtual machine, Oak Ridge National Labs, 2001. Acessado em 20 de julho de 2003. Disponível on-line: http://www.csm.ornl.gov/pvm/.

[GRO 03] GROP, W.; LUSK, E. **MPI**: The Message Passing Interface (MPI) Standard, Argonne National Laboratories, 2003. Acessado em 28 de julho de 2003. Disponível on-line: http://www-unix.mcs.anl.gov/mpi/.

[GNU 96] GNU. **Free Software Foundation, 1996.** Acessado em 28 de julho de 2003. Disponível on-line: http://www.gnu.org/.

[UNI 95] UNIX. **The Open Group, 1995.** Acessado em 28 de julho de 2003. Disponível on-line: http://www.unix.org/.

[SLA 03] SLACKWARE. **The Slackware Linux Project.** Acessado em 10 de julho de 2003. Disponível on-line: http://slackware.com/.

[FRE 99] FREITAS, E. L. **Uma comparação entre modelos de Message Passing MPI e PVM, UFRGS, 1999.** Acessado em 27 de julho de 2003. Disonível on-line: http://www.inf.ufrgs.br/procpar/disc/cmp134/trabs/T2/981/mpi.html/.

[KUK 03] KUKUK, T. **Homepage of Linux NIS/NIS+ Projects, 2003.** Acessado em 29 de julho de 2003. Disponível on-line: http://www.linux-nis.org/.

[NFS 03] NFS-HOWTO. Acessado em 28 de julho de 2003. Disponível on-line: http://nfs.sourceforge.net/.