Arquitetura de computadores - SBC/SI
Cluster
Computação em grade
Programação distribuída
393259
CN.PP 1.03.0300-6

Um Modelo para a Concepção de Grades Computacionais baseadas em Clusters

Marcos Ennes Barreto[1]
Philippe Olivier Alexandre Navaux[2]

Resumo

Este artigo apresenta o modelo MultiCluster, voltado para a concepção de grades computacionais baseadas em clusters possivelmente heterogêneos.

1 Introdução

O uso de clusters, a partir da metade dos anos 90, mudou drasticamente o panorama da programação paralela, uma vez que as bibliotecas de multiprogramação e comunicação até então utilizadas tiveram que ser adaptadas para protocolos específicos para essas redes de alto desempenho (BUYYA, 1999). Até o presente momento, muita pesquisa já foi realizada com esse tipo de plataforma, resultando em ambientes de programação e gerenciamento de agregados até mesmo com centenas de nós processadores.

Atualmente, o foco das pesquisa nessa área evoluiu para o estabelecimento de grandes plataformas computacionais, onde cada nó corresponde não mais a um PC mono ou multiprocessado, mas sim a um recurso muito mais especializado, tal como um cluster, uma base de dados ou um instrumento científico. Esse novo panorama de estudo e desenvolvimento é denominado de computação em grade (*grid computing*) (FOSTER et al., 1999) e seu objetivo principal é a integração de recursos geograficamente distribuídos com o intuito de formar uma única plataforma computacional, que possa ser compartilhada por diferentes usuários considerando-se aspectos relacionados à interfaces de programação, compartilhamento de dados e arquivos, segurança, autenticação, gerenciamento de recursos, balanceamento de carga, tolerância a falhas, entre outros.

[1]barreto@inf.ufrgs.br
[2]navaux@inf.ufrgs.br

2 Computação baseada em grades computacionais

A utilização de grades computacionais como plataforma de desenvolvimento de aplicações é referenciada como *grid computing* ou *metacomputing*, termos que surgiram no início dos anos 90 para representar a idéia de um metacomputador, que correspondia a um ambiente dinâmico composto por nós processadores que podiam ser adicionados e removidos livremente e que poderiam ser vistos como um único recurso computacional. Essa "idéia" foi estendida posteriormente para comtemplar não só a integração de processadores, mas também de outros recursos, tais como fontes de informações e de dados, instrumentos científicos, componentes de software, pessoas, etc. (FOSTER et al., 1999).

Um dos principais aspectos relativos às plataformas baseadas em grades diz respeito ao compartilhamento de recursos em larga escala. Entretanto, diversos outros fatores devem ser considerados para a definição de um modelo estrutural para tal plataforma.

Estruturalmente, uma arquitetura de grade combina aspectos advindos de outras arquiteturas, exigindo, portanto, uma estrutura de software que forneça funcionalidades que explorem eficientemente e de forma segura os recursos computacionais da grade.

Ian Foster e outros (FOSTER et al., 1999) abordam a questão estrutural de uma forma bastante coerente ao dizer que uma vez que uma arquitetura de grade poderá ser compartilhada por diferentes "organizações virtuais", cada qual com características e necessidades específicas, é incorreto pensar em uma arquitetura padrão para grades, mas sim na definição de um conjunto básico de serviços que devem ser providos, possibilitando que cada organização virtual implemente e estenda essas funcionalidades da forma mais adequada.

Basicamente, uma grade computacional pode ser organizada em quatro camadas: i) **dispositivos de fábrica**, que correspondem a todos os recursos (geograficamente distribuídos) que podem ser acessados dentro da grade (PCs, clusters, dispositivos de armazenamento, instrumentos científicos, bancos de dados, etc.); ii) **camada de serviços**, que corresponde às funcionalidades básicas para a utilização de grades, tais como serviços de localização, alocação de recursos, escalonadores, protocolos de comunicação, autenticação, informação e monitoramento; iii) **camada de ferramentas**, que engloba ambientes de programação e sistemas que permitem o desenvolvimento de aplicações orientadas a grades; e iv) **aplicações** que podem ser desenvolvidas ou adaptadas para plataformas de grades, variando em relação ao seu propósito e recursos necessários.

O estabelecimento de plataformas baseadas em grades certamente requer o desenvolvimento de uma série de serviços, entre eles suporte a diversidade de aplicações, modelos de programação e ferramentas, gerência de recursos e segurança, baseados em modelos e protocolos aceitos por todas as organizações dispostas a compartilhar seus recursos computacionais.

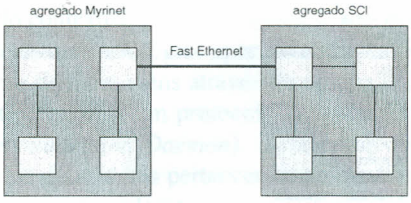

Figura 1: Possível configuração física do modelo MultiCluster.

3 O modelo MultiCluster

O projeto MultiCluster (BARRETO; ÁVILA; NAVAUX, 2000) tem por objetivo a definição de um modelo de integração de clusters independentes e heterogêneos que pode ser empregado para a "construção" de uma arquitetura paralela integrada e acessível através de uma única interface de programação. Este modelo pode ser abordado sob dois aspectos: hardware e organização do software. Em ambos os aspectos, o que se pretende é estudar um caso específico de grade computacional onde os recursos são exclusivamente clusters de alto desempenho.

Em termos de hardware, o modelo assume, inicialmente, a existência de pelo menos dois clusters geograficamente próximos, conforme o exemplo ilustradro na figura 1.

Neste exemplo, um cluster está internamento conectado pela rede Myrinet e outro pela rede SCI. Independente da rede interna, o modelo pressupõe a existência de uma conexão física entre os clusters que se pretende integrar, a qual pode ser feita através da ligação direta de um dos nós de um cluster com um nó de outro cluster, através da ligação entre as máquinas servidoras (*front-ends*) de cada um dos clusters, ou pela configuração de uma máquina como pertencente a mais de um cluster (com suas respectivas interfaces de rede para cada cluster).

No primeiro e terceiro caso, seria possível que todos os nós, independentemente do cluster ao qual pertencem, se comunicassem por intermedio dos nós que interligam os clusters. No segundo caso, a comunicação entre nós de diferentes clusters poderia ser feita através das máquinas servidoras. De qualquer forma, para o modelo MultiCluster, é importante que em cada cluster exista um nó *gateway* que implemente as funções relacionadas com a comunicação com outros clusters.

Em termos de software, o primeiro ponto a considerar diz respeito a como combinar diferentes modelos de comunicação. Para tanto, o modelo MultiCluster segue algumas premissas que determinam como os clusters podem ser integrados e posteriormente utilizados.

A primeira premissa diz respeito às diferentes visões da arquitetura integrada, definidas em termos de nós físicos e lógicos. Um **nó físico** corresponde a um nó pertecente a um determinado cluster e somente interessa para questões de organização física da arquitetura. Um **nó lógico** corresponde a um recurso computacional do ponto de vista da aplicação; sendo que

```
// máquina virtual
@virtual_machine:
verissimo , quintana , euclides , dionelio , scliar , ostermann , meyer , luft
// sub- redes
@comm _contexts:
myrinet 0
sci 1
// nós lógicos
@logical_nodes:
node 0:0 machines    verissimo
node 1:0 machines    quintana
node 2:0 machines    euclides
node 3:0 machines    dionelio
node 4:1 machines    scliar , luft
node 5:1 machines    ostermann , meyer
// nós gateway
@gateways:
quintana , scliar
```

Figura 2: Arquivo de configuração do modelo MultiCluster.

para clusters baseados em comunicação por troca de mensagens, cada nó lógico corresponde a um nó físico, e para clusters baseados em comunicação por memória compartilhada, cada nó lógico pode corresponder a mais de um nó físico.

A determinação dos nós lógicos dentro da arquitetura fica a cargo do programador, através da especificação de um arquivo descritor. Nesse arquivo, ilustrado na figura 2, o usuário deve determinar quais as máquinas que compõem a arquitetura integrada, quais os contextos ou sub-redes de comunicação que serão utilizados e quais máquinas (nós lógicos) pertencem a cada sub-rede.

A distinção entre os nós lógicos é feita pelo identificador de cada nó e pelo contexto de comunicação ao qual o nó pertence. Um **contexto de comunicação** corresponde ao conjunto de nós capazes de se comunicar através de uma determinada rede. No exemplo da figura 2, "node 1:0" significa o segundo nó da sub-rede 0, enquanto que "node 4:1" significa o primeiro nó da sub-rede 1. A numeração dos nós é seqüencial, independente da sub-rede a qual o mesmo pertence. No exemplo, existem 6 nós lógicos, numerados de 0 a 5.

Dentro de um nó lógico, a comunicação é realizada através de memória compartilhada. Se o nó lógico corresponde a uma máquina pertencente a um cluster conectado pela rede Myrinet, é possível estabelecer comunicação entre processos através de troca de mensagens (ou por memória compartilhada, de maneira mais otimizada). Se o nó lógico corresponde a um conjunto de máquinas pertencentes a um cluster conectado pela rede SCI, é possível a criação de um espaço de endereçamento global através do mapeamento de porções individuais de memória de cada nó.

A comunicação entre diferentes nós lógicos é obrigatoriamente realizada por troca de mensagens, uma vez que esses nós lógicos podem pertencer a clusters diferentes. Quando a comunicação se dá entre nós lógicos dentro de um mesmo contexto de comunicação, é possível otimizar essa comunicação através da utilização de um único protocolo de comunicação

para a rede representada pelo contexto.

Quando a comunicação envolve nós lógicos pertencentes a diferentes contextos, então é preciso realizar o roteamento de mensagens através do nó que se comporta como *gateway*. Para tanto, cada nó desse tipo executa um protocolo de comunicação entre clusters, denominado RCD (*Remote Communication Daemon*). Este protocolo possui um endereço de comunicação conhecido pelos demais nós pertencentes ao cluster, permitindo que qualquer nó possa enviar mensagens para este endereço.

O RCD atua quando um nome definido remotamente (em outro cluster) precisa ser encontrado e quando uma mensagem precisa ser transmitida para caixas postais remotas. Quando uma pritimiva de comunicação não consegue descobrir um nome consultando o servidor de nomes local, ela interage com o RCD, a fim de que o mesmo repasse a consulta para outros RCDs. No segundo caso, quando uma primitiva de comunicação precisa enviar uma mensagem para um endereço remoto, ela interage com o RCD local para que o mesmo repasse a mensagem para o RCD remoto do cluster onde a caixa postal destinatária se encontra.

É importante salientar que a comunicação entre threads em diferentes nós lógicos, bem como em diferentes clusters, é sempre realizada via troca de mensagens. Mesmo no caso de clusters SCI, deve existir pelo menos uma caixa postal para permitir a comunicação com o RCD e, eventualmente, receber mensagens.

A fim de validar o modelo em relação a organização de hardware suportada e as premissas definidas, um protótipo de integração de clusters foi desenvolvido. Este protótipo considerou a implementação do ambiente DECK sobre o protocolo BIP para redes Myrinet e teve como objetivo validar o esquema de nomeação de nós (físicos e lógicos) e medir os tempos de comunicação entre nós pertencentes a diferentes clusters. Neste esquema, dois clusters Myrinet, cada qual contendo dois nós processadores, foram integrados. Em cada cluster, um dos nós desempenha o papel de *gateway* do cluster, ou seja, executa o protocolo RCD em um *thread* específico. Para esta organização de hardware, uma avaliação de desempenho da comunicação inter-cluster foi realizada, através da execução de um algoritmo de *ping-pong* com diferentes tamanhos de mensagens.

4 Trabalhos relacionados

Atualmente, existem dezenas de ambientes de desenvolvimento de aplicações para plataformas baseadas em agregados e grades. Alguns desses ambientes correspondem a evoluções de ambientes já existentes, quando a computação paralela e distribuída era explorada em supercomputadores e em redes de computadores ou estações, respectivamente.

Globus (FOSTER et al., 1998) é um projeto americano cujo objetivo é possibilitar a construção de grades computacionais, vistas como uma infraestrutura de hardware e software que provê acesso a recursos computacionais, independente da localização geográfica dos mesmos. O principal componente dessa infraestrutura é o GMT — *Globus Metacomputing Toolkit*, que

corresponde a uma camada de software que implementa serviços e funcionalidades requeridos para o estabelecimento de grades computacionais, tais como gerenciamento de processos, alocação de recursos, comunicação ponto-a-ponto e coletiva, autenticação, segurança, acesso remoto a dados e a informações de estado, monitoramento dos recursos do sistema e gerência de executáveis.

JavaPorts (MANOLAKOS et al., 2001) é um ambiente de programação voltado para arquiteturas de agregados e que provê um conjunto de ferramentas destinadas a facilitar o desenvolvimento de aplicações.

O ambiente JavaPorts é definido a partir de quatro elementos: i) *grafo de tarefas*: usado para especificar o mapeamento de "partes" da aplicação para os recursos disponíveis, através de um arquivo de configuração, que ao ser compilado, gera informação ao ambiente para a criação ou atualização de modelos (*templates*) de código e *scripts* necessários ao disparo da aplicação; ii) *modelos de código*: para cada tarefa definida no arquivo de configuração, um modelo *template* de código é gerado, já incorporando os principais métodos necessários para que a tarefa possa ser executada dentro do ambiente; iii) *interface de comunicação*: as primitivas de comunicação entre tarefas são implementadas em uma interface Java denominada *Ports*, composta por atributos estáticos e métodos para a troca de mensagens; e iv) *protocolo de comunicação*: um protocolo desenvolvido sobre a interface RMI do Java que executa todas as operações relacionadas com o registro de portas locais em um nó e a procura de portas remotas.

Madeleine (BOUGÉ et al., 1999) é uma biblioteca que provê suporte para múltiplas interfaces e protocolos de comunicação, permitindo que uma aplicação possa utilizar diferentes interfaces/protocolos de forma transparente, já que a biblioteca se encarrega de determinar qual a interface e o protocolo mais adequados para cada operação de comunicação, baseada em critérios de tráfego e desempenho.

A biblioteca define duas abstrações principais: *canal*, que representa um domínio fechado de comunicação e que está associado a um protocolo de comunicação, a uma interface de comunicação e a um conjunto de objetos de conexão; e *conexão*, que representa uma ligação ponto-a-ponto confiável entre dois processos comunicantes.

A versão atual da biblioteca, Madeleine III, está voltada para a integração de clusters. Para tanto, utiliza *canais físicos*, que correspondem às redes "reais" de comunicação; e *canais virtuais*, que são criados sobre os canais físicos para permitir a comunicação entre nós de diferentes clusters. Uma vez que um canal virtual é criado, a aplicação não consegue mais "enxergar" os canais físicos usados por esse canal virtual.

Toda e qualquer comunicação entre processos pertencentes a clusters distintos é realizada através de um protocolo TCP sobre uma rede Ethernet. Dentro de cada cluster, um protocolo de comunicação específico é utilizado, dependendo do tipo de rede existente.

A configuração dos canais físicos e virtuais fica a cargo do programador, que utiliza arquivos de configuração para tal fim. Esses arquivos são processados por uma ferramenta de gerência de sessão e de conexões, denominada *Leonie*, que se encarrega do disparo de pro-

cessos e do estabelecimento de rotas (e respectivas tabelas de roteamento) entre todos os nós dos clusters integrados.

Além desses, existem outros ambientes voltados à exploração de clusters e grades como plataformas computacionais para a execução de aplicações paralelas e distribuídas, entre os quais Legion (GRIMSHAW et al., 1999), NetSolve (CASANOVA et al., 1995), Stardust (CABILLIC et al., 1997), VMI (VMI..., 2001), PLUS e PACX-MPI.

Referências Bibliográficas

BARRETO, M.; ÁVILA, R.; NAVAUX, P. The MultiCluster model to the integrated use of multiple workstation clusters. In: ROLIM, J. et al. (Ed.). *Proc. of the 3rd Workshop on Personal Computer based Networks of Workstations*. Cancun: Berlin, Springer-Verlag, 2000. (Lecture Notes in Computer Science, v. 1800), p. 71–80.

BOUGÉ, L. et al. *Madeleine: An Efficient and Portable Communication Interface for RPC-based Multithreaded Environments*. Lyon, 1999.

BUYYA, R. (Ed.). *High Performance Cluster Computing: Architectures and Systems*. Upper Saddle River: Prentice Hall PTR, 1999. 849 p.

CABILLIC, G. et al. Stardust: An environment for parallel programming on networks of heterogeneous workstations. *Journal of Parallel and Distributed Computing*, v. 40, n. 2, p. 65–80, 1997.

CASANOVA, H. et al. *NetSolve: A network server for solving computational science problems*. [S.l.], 1995.

FOSTER, I. T. et al. *Globus: A Metacomputing Infrastucture Toolkit*. 1998. Disponível por WWW em http://www.globus.org (dez. 1998).

FOSTER, I. T. et al. (Ed.). *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco: Morgan Kaufmann, 1999. 677 p.

GRIMSHAW, A. S. et al. Legion: an operating system for wide-area computing. *IEEE Micro*, v. 32, n. 5, p. 29–37, maio 1999.

MANOLAKOS, E. S. et al. *JavaPorts: An environment to facilitate parallel computing on a heterogeneous cluster of workstations*. 2001. Disponível por WWW em http://www.acm.org/crossroads/xdrs5-3/elias.html (jan. 2001).

VMI: An Efficient Messasing Lybrary for Heterogeneous Cluster Communication. 2001. Available at http://archive.ncsa.uiuc.edu/People/apant.