Tópicos em Ciência da Computação
Relatório

Aluno: Jeferson Roberto Marques
(jmarques@ime.usp.br)

Orientador: Fábio Kon
(kon@ime.usp.br)

A implementação encontra-se disponível em:
http://lcpd.ime.usp.br/~jmarques/GrmExt.tgz

1 Introdução

O desenvolvimento e implantação recentes de tecnologias de redes de alta velocidade, aliados à proliferação de máquinas de baixo custo e alto desepenho, deu oportunidade ao surgimento de ambientes de metacomputação(*networked virtual computing*). Estes ambientes podem combinar centenas de máquinas e domínios administrativos conectados por uma rede local ou até mesmo global, como por exemplo a Internet.

A tecnologia em que se baseiam tais ambientes nos permite utilizar estas máquinas interligadas como sendo um recurso unificado, que chamamos de agrupamento. Estes podem aparecer em várias formas: agrupamentos de alto desempenho, agrupamentos de alta disponibilidade, agrupamentos dedicados, etc.

O propósito desse ambiente é oferecer compartilhamento de recursos entre conjuntos dinâmicos de indivíduos e instituições. Além disso, este ambiente deve permitir acesso consistente e de baixo custo aos recursos independentemente de sua posição física.

Um dos problemas mais importantes neste ambiente é coordenar o compartilhamento de recursos altamente dinâmicos. Gerenciar os recursos é uma tarefa complexa, sobretudo sob estas circunstâncias, porém fundamental para executar programas de usuários de maneira eficiente e econômica.

O trabalho desenvolvido foi uma extensão do gerenciamento de recursos no 2K, descrito abaixo e detalhado em [4]. O 2K é um sistema operacional distribuído adaptável.

2 Gerenciamento de Recursos no 2K

O serviço de gerenciamento de recursos no 2K[1] é estruturado como uma coleção de LRMs(*Local Resource Managers*), GRMs(*Global Resource Managers*) e CRs(*Component Repositories*).

Normalmente, um conjunto de máquinas é agrupado formando um agrupamento onde o 2K é instalado. Cada máquina do agrupamento possui um LRM, que informa periodicamente para o GRM mudanças relativas aos recursos disponíveis da máquina, como memória e carga. Além disso, ele oferece um serviço para execução de processos.

O GRM é essencialmente um *trader* estendido para proporcionar algumas funcionalidades adicionais. Sua função é manter a disponibilidade de recursos dos LRMs. É também responsável por encaminhar pedidos de recursos para outros LRMs a fim de garantir a qualidade de serviço pedida por um componente(objeto) qualquer.

A figura 1 ilustra o relacionamento entre esses objetos.

Para que uma consulta seja feita por um programa, os LRMs registram-se no GRM e passam os dados referentes à máquina, que são estruturados como um conjunto de propriedades. Assim, um programa pode verificar a disponibilidade de recursos consultando as entradas de cada LRM no GRM.

O *trader* estendido faz parte do GRM e agrupa a disponibilidade de recursos de uma máquina particular e os serviços hospedados na mesma máquina. Esta característica não faz parte das funcionalidades de um *trader* padrão, onde cada serviço oferecido registrado nele é independente de outros.

2.1 Descoberta de serviços

Basicamente, as referências para os objetos são encontradas percorrendo os seguintes passos(figura 2):

---

[1] http://choices.uiuc.edu/2k

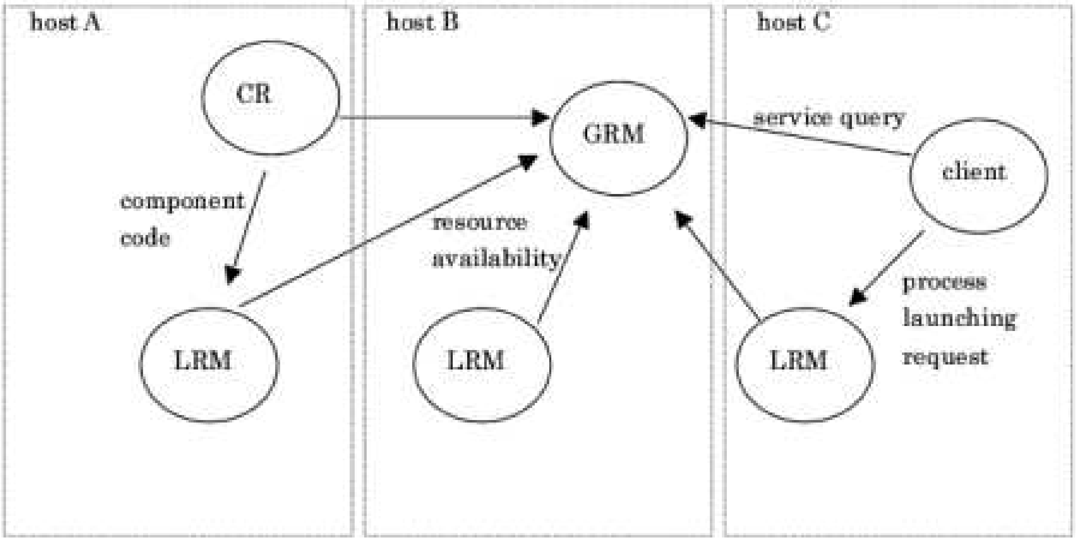

Figura 1: Gerenciamento de Recursos

1. programa-cliente localiza o LRM da sua máquina;
2. LRM encontra LRM local – LRM duplicado;
3. LRM localiza GRM;
4. LRM localiza CR;

3 Extensão do modelo

O gerenciamento de recursos acima descrito atende bem objetos que estejam num determinado ou agrupamento(*cluster*). No entanto, eles ficam isolados de recursos disponíveis existentes além de suas fronteiras. É possível que um objeto necessite de uma determinada quantidade de memória para ser executado e as máquinas do agrupamento onde está localizado não possam atendê-lo adequadamente no momento da requisição.

Dessa forma, seria interessante haver um mecanismo que "expandisse" seus domínios e agregasse mais recursos disponíveis para uma localidade, aumentando, portanto, a probabilidade de sucesso ao atender uma requisição

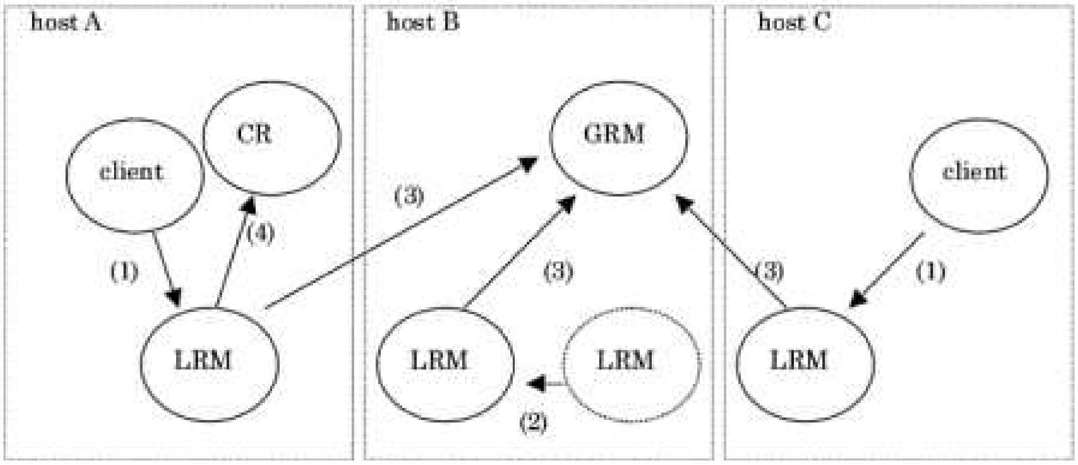

Figura 2: Localização de objetos

de algum recurso por um objeto. Além deste, outros benefícios seriam adicionados direta e indiretamente:

- Replicação de recursos, diminuindo a tolerância a falhas;
- Maior diversidade dos serviços e recursos oferecidos.

Como descrito anteriormente, um agrupamento possui um número arbitrário de máquinas, cada uma com um LRM operando, e um GRM para estes LRMs. Para que dois ou mais agrupamentos utilizem recursos e componentes mutuamente, é necessário desenvolver uma estrutura que permita que cada integrante possa ter acesso a informações sobre onde estão localizados os recursos desejados. Assim, o propósito deste trabalho é apresentar e criar um mecanismo para interligar diferentes agrupamentos.

A preocupação primordial no *design* deste mecanismo é garantir a escalabilidade do sistema. É preciso também introduzir e retirar agrupamentos com o mínimo de impacto sobre o resto do sistema.

Para construir um ambiente como o citado acima, foi escolhida a estrutura de árvore para interligar informações sobre os agrupamentos. Dois ou mais domínios que pretendem disponibilizar os recursos são “conectados” por meio das informações de um outro GRM, que possui dados aproximados sobre as condições dos recursos de cada um de seus descendentes diretos.

A escolha também foi influenciada por outras fontes, como os trabalhos do grupo de desenvolvimento do 2K, particularmente na parte de gerenciamento de recursos, do trabalho desenvolvido em [2] e de outras literaturas em geral.

É importante deixar claro que, para garantir a escalabilidade também é necessário evitar manter dados exatos sobre as condições dos descendentes, pois isso traz uma sobrecarga – na rede, por exemplo – muito grande cada vez que é necessária uma atualização dos dados, que propagaria por todo o caminho de um nó até a raiz.

3.1 *Design*

Os LRMs de cada agrupamento mantêm-se inalterados, eles não possuem qualquer relacionamento com outro GRM, senão o responsável pelo próprio agrupamento. Os GRMs, no entanto, passam a colaborar com outros GRMs, obedecendo uma estrutura de árvore. Assim, a organização ilustrada na figura 1 permanece inalterada. Entretanto, o GRM teve de ser modificado para comunicar-se com outro GRM fora do seu agrupamento. Sua funcionalidade teve de ser estendida para que ao invés de retornar uma resposta negativa se não houver recursos para atender uma determinada requisição, ele a envia para um nível superior da árvore para que a requisição seja analisada por outros agrupamentos e assim por diante.

Este novo GRM para o qual os GRMs repassam consultas é chamado de **GrmExt**. Basicamente, eles mantêm dados referentes à média de cada recurso oferecido em cada uma de suas subárvores. Dessa forma, as informações representam apenas uma "pista" de onde os recursos devem ser procurados. Com isto resolve-se em parte o problema de sobrecarga de informações em cada componente do sistema. Existe, contudo, uma margem de erro implícita que deve ser aceita.

No processo de descoberta de recursos são incluídos alguns passos adicionais, resultando no seguinte protocolo:

1. Um cliente contacta o LRM para criar um processo.
2. O LRM verifica se há recursos suficientes disponíveis. Se houver, ele inicia o processo – passo 7;
3. Se não houver, encaminha a requisição para o GRM que procura candidatos no domínio do agrupamento para atender a requisição.

4. Caso ainda não tenha sucesso, a requisição é encaminhada para um nível superior, para um GrmExt, que fará um processo semelhante de procura. Este passo é feito sucessivamente até que a árvore seja percorrida. Pretende-se, assim, atingir primeiramente recursos que estão mais "próximos" do objeto que fez a solicitação, ou seja, recursos que não aumentem muito a latência gerada por uma possível deterioração da qualidade da comunicação pela rede, por exemplo.
5. Encontrando uma subárvore que pode oferecer o recurso pedido, a requisição é novamente encaminhada, mas desta vez para a subárvore abaixo.
6. Se nenhum nó puder oferecer o especificado na requisição, o objeto recebe um NACK, informando o ocorrido.
7. Se a requisição puder ser satisfeita, o LRM onde encontra-se o objeto inicia o processo.
8. O processo aloca os recursos especificados fazendo uma reserva de recursos.
9. O processo carrega o componente especificado do repositório de componentes.

Cada GRM possui as informações sobre todos os seus filhos e envia a média dos recursos oferecidos periodicamente para seu pai.

4 Obstáculos

Alguns dos obstáculos encontrados no desenvolvimento já foram apresentados implicitamente. Os obstáculos mais desafiadores que surgiram durante o processo de desenvolvimento foram:

- Manter cada agrupamento como uma unidade. A idéia é manter um único subsistema responsável pelo acesso a outros agrupamentos. O GRM foi escolhido para ser esta porta de acesso, embora as referências necessárias já estivessem com cada LRM. O GRM é basicamente um *trader*, que não possui referência para nenhum outro objeto, principalmente para a interface que dá acesso ao método responsável pelas consultas.

- Estrutura a ser utilizada para garantir escalabilidade. Como dito anteriormente foi escolhida uma árvore para representar o conjunto dos componentes do sistema.
- Identificar as informações relevantes sobre as subárvores que devem estar em cada nó.
- Garantir tolerância a falhas. O desligamento de um dos nós da árvore particionaria o sistema tornando as duas partes incomunicáveis. No momento este ponto não está solucionado.
- Inclusão de novos membros num grupo já existentes, isto é, união de duas ou mais árvores numa única árvore. Este ponto também não possui uma solução.

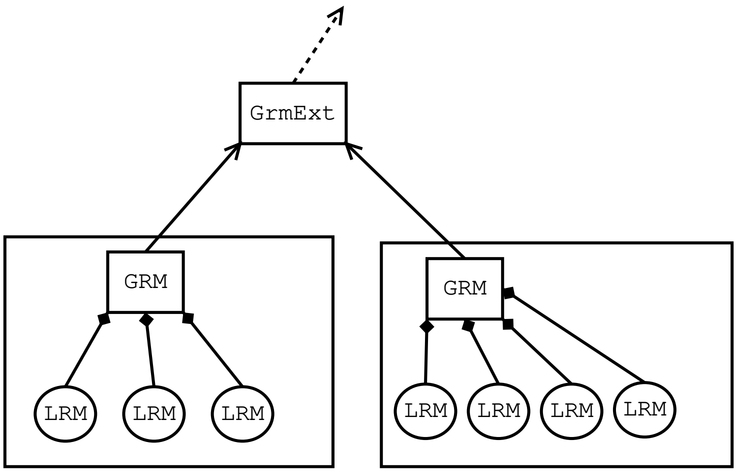

Figura 3: Modelo estendido

5 Relacionamento Entre os Sistemas de Gerenciamento de Recursos

O relacionamento descrito a seguir mostra a correspondência entre sistemas de diferentes infraestruturas [1].

	Globus	Legion
Escalonadores	Globus Resource Brokers	Legion Schedulers
Informações para descoberta de recursos	Globus Information Services	Legion Collections
Alocação e reserva de recursos	Globus Co-Allocators	Legion Enactors
Alocação de recurso local e criação de *jobs*	Globus GRAMs	Legion Host Objects

Referências

[1] Steve J. Chapin, Dimitrios Katramatos, John Karpovich, and Andrew S. Grimshaw. Resource Management in Legion. Technical Report CS-98-09, 11, 1998.

[2] P. Ezhilchelvan, M. Khayyambashi, G. Morgan, and D. Palmer. Measuring the Cost of Scalability and Reliability for Internet-Based, Server-Centered Applications. Sixth International Workshop on Object-oriented Real-time Dependable Systems (WORDS01), 2001.

[3] Fabio Kon. *Automatic Configuration of Component-Based Distributed Systems.* PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, May 2000.

[4] Tomonori Yamane. The Design and Implementation of the 2K Resource Management Service. Master's thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, February 2000.