Relatório Final da Disciplina MAC 5701
Implementação do Protocolo de Atualização de Informações Intra-Aglomerado do InteGrade

Código-Fonte Disponível em:

`http://www.ime.usp.br/~andgold/research/proto_atl.tgz`

Aluno - Andrei Goldchleger

Orientador - Prof. Fabio Kon

Professora Responsável - Yoshiko Wakabayashi

9 de dezembro de 2002

1 Introdução

Este documento descreve as atividades de pesquisa realizadas como parte da disciplina MAC 5701 - Tópicos em Ciência da Computação. O trabalho foi realizado na área de sistemas distribuídos e posteriormente será integrado à minha dissertação de mestrado, que versará sobre a arquitetura de um sistema de computação em grade em início de desenvolvimento no IME-USP, *InteGrade* [GKL+02].

1.1 Motivação

A computação atingiu um patamar surpreendente em termos da qualidade de recursos. Processadores que hoje equipam computadores pessoais possuem capacidade de processamento encontrada em supercomputadores há menos de uma década; as redes de computadores estão cada vez mais rápidas e disseminadas e o software evolui juntamente tornando-se cada vez mais dinâmico, configurável e modularizável.

Mesmo com os avanços em hardware, certos problemas ainda persistem. Alguns tipos específicos de usuários necessitam de grande poder computacional e não o possuem, ao passo que, em um instante qualquer, milhões de computadores pessoais estão ociosos ao redor do mundo. A Internet trouxe uma maneira inédita de conexão entre sistemas, mas certos recursos ainda não podem ser compartilhados e acabam sub-utilizados. Paradoxos que pareciam não possuir solução.

Visando resolver essas e outras questões, surge a idéia de Grades de Computação (*Computational Grids*). O objetivo é prover maneiras de interconectar computadores e recursos computacionais diversos para estender a fronteira do que é possível realizar utilizando-se sistemas de computação. O nome deriva da rede de energia elétrica (*Power Grid*), e simboliza a esperança de que, quando bem estabelecidas, as grades permitirão acesso a recursos computacionais de maneira tão simples quanto utilizar a rede elétrica. Da mesma forma que podemos entrar em qualquer sala do mundo e ligar um aparelho na tomada, esperamos desenvolver uma infra-estrutura que nos permitirá obter um enorme poder computacional através de uma simples ligação à Grade.

2 InteGrade

InteGrade é um projeto de computação em grade em início de desenvolvimento no IME-USP. Dentre os outros projetos similares em desenvolvimento, acreditamos que InteGrade possui ao menos os diferenciais listados abaixo.

- **Reutilização da base instalada**: é um princípio chave do projeto InteGrade. Tal característica também pode ser observada em poucos dos projetos existentes, porém o diferencial é que InteGrade considerará essa característica no desenvolvimento de sua arquitetura.
- **Utilização de tecnologias modernas de objetos distribuídos**: característica única do InteGrade, que utilizará CORBA na sua construção. Com isso, obtemos duas vantagens: (1) poderemos re-utilizar os inúmeros serviços já disponíveis na arquitetura CORBA e (2) será mais fácil a integração de outros serviços e aplicações à grade dado que esta utilizará uma tecnologia padrão de interconexão orientada a objetos.
- **Baixa sobrecarga**: o middleware do InteGrade vai priorizar o proprietário do recurso. Assim, é necessário que o sistema não consuma muitos recursos de maneira a não se tornar um problema para o usuário. Em particular, pretendemos utilizar ORBs CORBA que consumam poucos recursos.

2.1 Proposta Inicial de Arquitetura

A arquitetura e os requisitos de InteGrade ainda não estão completamente definidos, mas algumas características necessárias já estão evidentes. A figura 1 apresenta a nossa proposta atual de arquitetura para um aglomerado do InteGrade. Futuramente, diversos aglomerados serão interligados, dando origem a uma poderosa Grade de computação com potencial para abrigar milhões de máquinas, utilizando uma arquitetura hierárquica.

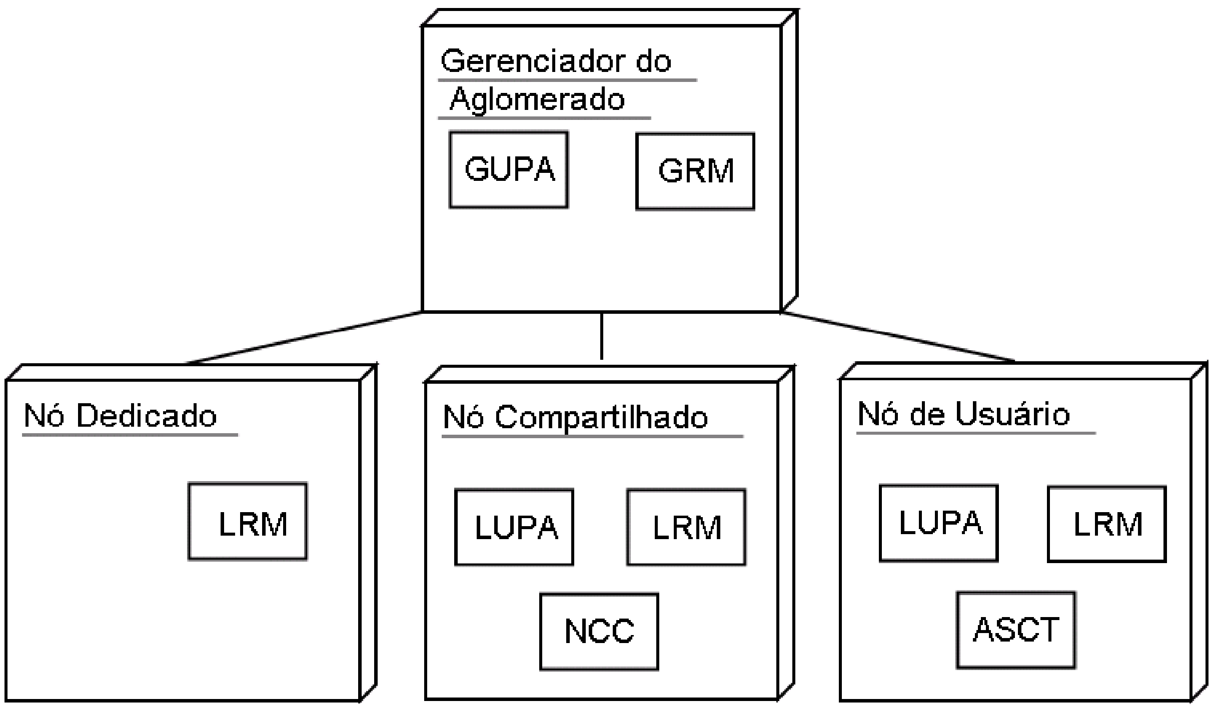

Figura 1: Arquitetura Intra-Aglomerado do InteGrade.

2.1.1 Descrição de cada nó

- **Nó Dedicado**: representa um recurso computacional dedicado, isto é, que não é utilizado como estação de trabalho.
- **Nó Compartilhado**: nó da Grade não dedicado, isto é, que é utilizado diretamente por um usuário, e que provê recursos computacionais para uso da Grade.
- **Nó de Usuário**: nó da Grade capaz de submeter aplicações para execução.
- **Gerenciador do Aglomerado**: nó da Grade responsável pelo monitoramento dos demais nós, escalonamento de aplicações, análise de uso dos nós do aglomerado e coleta de informações.

2.1.2 Descrição dos Módulos

- **LRM - *Local Resource Manager***: responsável por gerenciar recursos de uma máquina.
- **GRM - *Global Resource Manager***: responsável pelo gerenciamento de um aglomerado do InteGrade.
- **LUPA - *Local Usage Pattern Analyzer***: módulo que é executado em máquinas utilizadas como estações de trabalho. Seu objetivo é compilar um conjunto de informações que representem com certo grau de confiança o padrão de uso da máquina, com o objetivo de possibilitar melhorias de desempenho no sistema como um todo.
- **GUPA - *Global Usage Pattern Analyzer***: módulo presente apenas no nó gerenciador de um aglomerado, objetiva utilizar as informações fornecidas pelos diversos **LUPA** de maneira a consolidar padrões que reflitam as condições típicas de uso do aglomerado.
- **NCC - *Node Control Center***: este módulo permite que o proprietário de uma estação de trabalho configure as condições de utilização de seus recursos pela Grade.
- **ASCT - *Application Submission and Control Tool***: módulo que permitirá que usuários da Grade submetam aplicações para execução, monitorem o estado das execuções e recebam resultados.

3 Objetivo do Trabalho

A proposta de trabalho desenvolvida nessa disciplina foi a implementação de parte dos módulos responsáveis pelo gerenciamento de recursos de um aglomerado, a saber: LRM e GRM. A funcionalidade implementada é o protocolo de atualização dinâmica das quantidades dos recursos disponíveis em cada nó de um aglomerado InteGrade. Como já apresentado na proposta inicial, tais módulos já estavam implementados em outros trabalhos de nosso grupo de pesquisa; porém, tal implementação era inadequada às necessidades de InteGrade, pois encontra-se atrelada a outro sistema e apresenta consumo de memória superior ao aceitável pelo LRM. Tal restrição é especialmente importante para os nós compartilhados, uma vez que os usuários das estações de trabalho não devem perceber perda de desempenho de suas máquinas relacionada à execução do LRM.

4 Tarefas Desenvolvidas

4.1 Estudo da Implementação Existente

Devido às restrições apresentadas na seção anterior, decidimos reescrever a aplicação do início. Inicialmente foi necessário estudar a implementação já existente, de maneira a descobrir quais aspectos estariam presentes na nova implementação e quais deveriam ser removidos. Nessa fase dois aspectos relevantes se destacaram, e são descritos a seguir.

- Mudança no envio das informações: como já descrito, de tempos em tempos o LRM envia para o GRM uma atualização de seus recursos disponíveis. Existe uma divisão clara desses recursos em dois grupos: estáticos e dinâmicos. Recursos estáticos são aqueles que raramente sofrem variações, tais como o tipo de processador instalado ou a quantidade total de memória física presente no sistema. Já recursos dinâmicos são os que alteram-se constantemente, tais como memória disponível e porcentagem de uso do processador. No sistema pré-existente havia tal divisão das informações, mas, curiosamente, na hora da atualização todas as informações eram enviadas, inclusive as estáticas. Decidimos mudar esse aspecto da seguinte maneira: quando um LRM se registra em um GRM ele envia as suas informações estáticas. Posteriormente, a cada atualização, são enviadas apenas informações dinâmicas.

- Melhoria no encapsulamento: Os autores do sistema pré-existente decidiram implementar o GRM como um *Trader* CORBA estendido. A escolha do *Trader* é adequada em nosso ponto de vista, pois permite armazenar e atualizar facilmente as informações de cada LRM. Entretanto, tal implementação apresenta o defeito de atrelar o GRM a um serviço específico. Assim, decidimos unir as facilidades providas pelo *Trader* à possibilidade de trocar o mecanismo de armazenamento de informações caso seja necessário. Tal mecanismo poderia ser um Banco de Dados SQL ou até um arquivo em disco, por exemplo. Assim, a decisão tomada foi a de oferecer aos LRMs uma interface que omitisse qualquer referência a um determinado serviço. Assim, os LRMs enviam suas informações ao GRM utilizando uma interface simples e independente de quaisquer serviços específicos. Internamente ao GRM, utilizamos o *Trader* para armazenamento de informações.

5 A Implementação

Para a implementação, assim como na original, utilizamos CORBA, que facilita a construção de sistemas distribuídos, permitindo a produção de código de qualidade e isolando o programador de detalhes de rede, tais como soquetes (*sockets*).

O LRM está implementado em C++ sobre o ORB UIC-CORBA [KRC01]. Tal ORB foi escolhida por ser extremamente compacta, ocupando apenas algumas dezenas de kilobytes de memória. Como já mencionado, os LRMs devem ocupar o mínimo de memória possível, daí a importância na escolha do ORB.

O GRM foi implementado em JAVA sobre o ORB JacORB [Jac02]. Ressaltamos que não seria possível implementar o GRM apenas com o UIC-CORBA, uma vez que este não possui *Trader* integrado. Seria possível utilizar uma implementação do *Trader* à parte e integrá-lo ao UIC-CORBA, porem as limitações de uso de memória referentes ao GRM são bem menos rígidas em relação ao LRM. Assim, optamos por JAVA/JacORB por acreditarmos que JAVA facilita a implementação, ao mesmo tempo que permite a criação de código de qualidade. JacORB é também um ORB de excelente reputação.

Abaixo segue a interface IDL do GRM que permite que os LRM se registrem e enviem atualizações.

```
interface Grm{

    struct StaticInfo{
        string HostName;
        string OsName;
        string OsVersion;
        string ProcessorName;
        long TotalRAM;
        long TotalSwap;
    };

    struct DynamicInfo{
        long FreeRAM;
        long FreeSwap;
    };

    oneway void register_Lrm (in string lrmId, in StaticInfo staticInfo);
    oneway void send_update (in string lrmId,in DynamicInfo dynamicInfo);
};
```

A implementação se encontra disponível em:

http://www.ime.usp.br/~andgold/research/proto_atl.tgz

6 Obstáculos Encontrados

A grande maioria dos obstáculos encontrados durante a implementação estão relacionados ao LRM, mais especificamente ao UIC-CORBA. Já ressaltamos previamente as virtudes desse ORB. Entretanto, o mesmo se encontra em desenvolvimento, portanto apresenta problemas inerentes a software em tais condições. Os principais problemas são discutidos a seguir.

- Erros de funcionamento no ORB: O desenvolvimento foi marcado por um erro de funcionamento do UIC-CORBA que quase inviabilizou o seu uso. No início do desenvolvimento, tanto o LRM como o GRM foram escritos em C++, sendo o GRM apenas um protótipo simplificado que não armazenava informações no *Trader*. Após a conclusão do desenvolvimento do LRM, iniciei o desenvolvimento do GRM utilizando JAVA e JacORB. Entretanto, na hora de realizar testes com o LRM já implementado em C++, a surpresa foi que os ORBs não inter-operavam, o que é inadmissível em CORBA.

  A solução de tal problema incluiu a depuração do código-fonte da própria ORB, algo que certamente não estava nos planos iniciais. De forma resumida, a origem do problema deve-se ao fato de que o UIC-CORBA tem dificuldades de entender as referências CORBA geradas pelo JacORB e utilizadas para a descoberta e comunicação com o GRM. Tal problema foi comunicado ao autor do ORB.

- Deficiências no compilador IDL: O compilador IDL é a ferramenta que traduz as interfaces IDL de alto nível para uma linguagem de programação determinada, incluindo código necessário para o uso do ORB. Assim, qualquer ORB bem desenvolvida deve contar com um compilador IDL robusto. Infelizmente esse ainda não é o caso do UIC-CORBA.

  O compilador IDL de UIC-CORBA ainda é experimental. Nas palavras do próprio autor, não se deve confiar na saída de tal compilador, o que requer alguma edição manual dos arquivos de saída. Estes também não estão livres de erros, apesar de tais erros serem facilmente corrigidos.

  Outro problema é que o compilador aceita apenas um subconjunto de tipos IDL. Não é possível, por exemplo, utilizar números do tipo *float* nas interfaces. Além disso, não é possível especificar mais de uma interface IDL por arquivo, nem utilizar alguns designadores de métodos, como *oneway*, que indica uma chamada de método assíncrona.

  A limitação do compilador me obrigou a escrever duas interfaces IDL semelhantes para o GRM, uma para uso do JacORB e outra para uso do UIC-CORBA. A interface aqui previamente apresentada é a utilizada com JacORB. Note que no caso de ORBs plenamente desenvolvidas isso não seria necessário, sendo suficiente apenas uma interface. Tal fator pode se tornar um problema com o prosseguimento do desenvolvimento, uma vez que teremos interfaces IDL duplicadas, o que dificulta a manutenção das mesmas.

As virtudes de UIC-CORBA são consideráveis e não podem ser desprezadas. Entretanto, a medida que o desenvolvimento do LRM progrida e mais funcionalidades sejam adicionadas, é possível que as limitações do ORB impeçam ou limitem o desenvolvimento. Tendo em vista tais fatos, iniciaremos o estudo das qualidades de outros dois ORBs compactos, a saber: Orbix/E [ION02] e LOrb (trabalho ainda sem página descritiva ou artigos publicados). Infelizmente a disponibilidade de ORBs compactas ainda é limitada.

7 Desenvolvimento Futuro

Em um futuro próximo, iniciarei o desenvolvimento do protocolo de execução de componentes, que permitirá que usuários de estações de trabalho utilizem seus LRMs para executar aplicações em outros nós do aglomerado. Daí a importância do protocolo implementado nesta disciplina: permitir que o GRM tome decisões sobre o emparelhamento entre requisições de execução e nós do aglomerado. Baseado nas informações recolhidas pelo protocolo de atualização e utilizando informações sobre os requisitos das aplicações, o GRM pode alocar aplicações a nós mais apropriados a executá-las.

Referências

[GKL+02] Andrei Goldchleger, Fabio Kon, Alfredo Goldman vel Lejbman, Marcelo Finger, and Siang Wun Song. InteGrade: Rumo a um Sistema de Computação em Grade para Aproveitamento de Recursos Ociosos em Máquinas Compartilhadas, Outubro 2002. Relatório Técnico RT-MAC-2002-08.

[ION02] IONA. Sítio do ORB Orbix/E, 2002. `http://www.iona.com/products/orbix-e.htm`.

[Jac02] JacORB. Sítio do ORB Jacorb, 2002. `http://www.jacorb.org`.

[KRC01] Fabio Kon, Manuel Román, and Roy H. Campbell. Reflective middleware: From your desk to your hand. *Distributed Systems Online*, Vol. 2(No. 5), 2001.