Relatório final da Disciplina MAC-5701
Implementação do Repositório de Aplicações do InteGrade
IME - Instituto de Matemática e Estatística
USP - Universidade de São Paulo

Alexandre Cesar Tavares Vidal - N.o USP: 4791925
Orientador - Prof. Fábio Kon
Professora Responsável - Yoshiko Wakabayashi

25 de Junho de 2004

1 Introdução

Este Relatório apresenta o trabalho de pesquisa realizado como parte da disciplina MAC5701 - Tópicos em Ciência da Computação. Esse trabalho compõe o Projeto InteGrade, sistema de computação em grade, em desenvolvimento no IME-USP, que abrange principalmente tópicos em computação paralela e distribuída. O cerne do trabalho é a implementação da versão estendida do Repositório de Aplicações do InteGrade.

Inicialmente, o repositório de aplicações atual foi avaliado, buscando identificar as reais necessidades de extensão para atender demandas funcionais do sistema. Concomitantemente, pesquisou-se soluções técnicas adotadas para problemas similares encontrados em alguns projetos reconhecidamente relevantes em computação distribuída, mais especificamente os Projetos Globus e Gaia. Essas atividades, possibilitaram a definição de uma nova interface para o repositório de aplicações do InteGrade. A versão definitiva desta interface, sua descrição e funcionalidade, os problemas encontrados e as soluções sugeridas foram discutidos pelo grupo de desenvolvedores do InteGrade, em um processo sinergético e produtivo.

Este trabalho está organizado como segue: a Seção 2 relaciona os principais fatores que motivaram a pesquisa. A Seção 3 contém uma visão geral do InteGrade e de seu atual estágio de implementação. A Seção 4 apresenta as soluções Globus e Gaia para a implementação de Serviço de Informação em sistemas distribuidos. Em particular, a interface `ComponentRepository` do Gaia influenciou a definição da interface adotada neste trabalho. A Seção 5 trata do módulo Repositório de Aplicações e descreve a semântica da nova interface IDL `ApplicationRepository`, já disponibilizada para implantação no InteGrade. Também é descrita a relacão dessa interface com outros módulos do sistema. A Seção 6 contém as conclusões obtidas.

2 Motivação

O projeto InteGrade [1] foi motivado principalmente pela necessidade de aproveitamento de recursos computacionais compartilhados (computadores pessoais, estações de trabalho), em geral subutilizados - com predomínio de períodos ociosos - para execução de aplicações com grande demanda de tais recursos. Em particular, as características do InteGrade incluem suporte para uma faixa abrangente de aplicações paralelas e mecanismos que tornem imperceptível aos proprietários dos recursos compartilhados, qualquer possível perda de qualidade de serviço [K+04b].

[1] `http://gsd.ime.usp.br/integrade/`

Mais especificamente, este trabalho contribui para a evolução da implementação do repositório de aplicações do Sistema InteGrade. Essa necessidade foi apontada na 6a. reunião de *Checkpoint* do InteGrade, realizada em setembro de 2003. Entendeu-se necessário refinar o repositório de aplicações, que oferecia funcionalidades extremamente limitadas, para permitir categorização de aplicações, entre outras características [K^{+}04a].

3 O InteGrade

A grade InteGrade se constitui de `aglomerados` (*clusters*) de computadores organizados de forma hierárquica e escalável. Nesta seção apresentamos uma visão resumida da arquitetura do InteGrade através da descrição dos componentes de um aglomerado e dos módulos implantados nestes componentes. A Figura 1, descreve graficamente esta arquitetura.

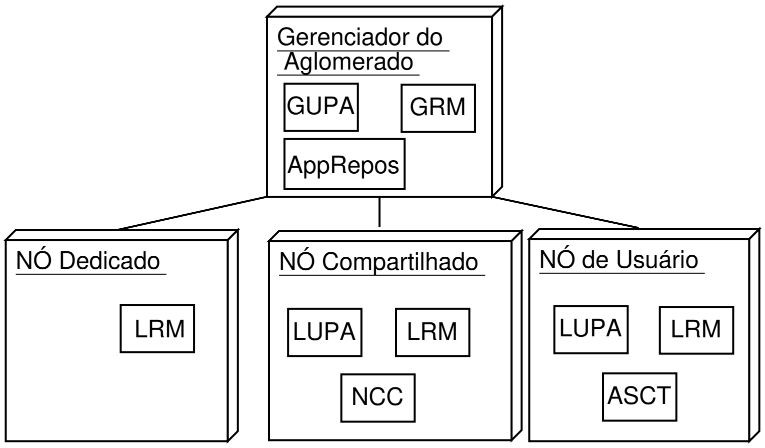

Figura 1: Arquitetura do InteGrade

3.1 Arquitetura do InteGrade

Os componentes de um aglomerados do InteGrade são classificados conforme suas responsabilidades:

Gerenciador do aglomerado: responsável por gerenciar o aglomerado e pela comunicação com gerenciadores de outros aglomerados.

Nó dedicado: reservado para computação na grade.

Nó compartilhado (fornecedor de recursos): tipicamente uma estação de trabalho, disponibiliza parte de seus recursos para execução de aplicações dos usuários da grade.

Nó de usuário: pertence ao usuário da grade que submete aplicações à grade.

O *Local Resource Manager* (LRM) e o *Global Resource Manager* (GRM) cooperam no gerenciamento intra-aglomerado, em um processo conhecido como *Information Update Protocol*. O LRM é executado em cada nó do aglomerado, coletando informações sobre o estado do nó (uso de CPU, disco, memória e recursos de rede) e as envia periodicamente ao GRM. Este último executa no nó gerenciador do aglomerado e usa essas informações enviadas pelos LRMs para escalonamento de máquinas para execução das aplicações submetidas à grade.

LRM e GRM também colaboram no *Resource Reservation and Execution Protocol*. O GRM seleciona nós candidatos para executar uma aplicação submetida pelo usuário da grade, com base nos requisitos da aplicação e na disponibilidade de recursos. Estas últimas informações são coletadas pelos LRMs, nos nós fornecedores de recursos.

De modo semelhante à cooperação LRM/GRM, os módulos *Local Usage Pattern Analyzer* (LUPA) e *Global Usage Pattern Analyzer* (GUPA) também trabalham cooperativamente na análise de padrões de uso das máquinas para contribuir para um escalonamento eficiente na grade [G.04].

O módulo *Node Control Center* (NCC) permite aos proprietários de recursos configurar as condições de compartilhamento, como o percentual de CPU e memória física disponibilizados para a grade, ou definir quando tais recursos podem ser considerados ociosos e, portanto, disponíveis.

O módulo *Application Submission and Control Tool* (ASCT) permite aos usuários do InteGrade submeter as aplicações para execução na grade. As aplicações registradas no ASCT são armazenadas em um repositório de aplicações e recuperadas pelos LRMs para execução.

Ao longo do primeiro semestre de 2004, dois novos módulos foram desenvolvidos para serem adicionados ao InteGrade: O módulo de segurança, proposto por outro membro do grupo de pesquisa do InteGrade, e o repositório de aplicações. A redefinição e implementação do repositório de aplicações é objeto do presente trabalho.

4 Globus e Gaia

Os projetos Globus e Gaia são exemplos conhecidos de sistemas distribuídos bem sucedidos. No entanto, a escolha desses sistemas como referências para abordar o problema do repositório de aplicações tem motivações diferentes.

Globus é um sistema em grade. O projeto Globus provê uma infraestrutura para integrar, em um sistema em grade único, máquinas geograficamente dispersas e ferramentas que permitem a construção de sistemas para grade em um modo incremental. Contudo, o foco do Globus é a construção de infraestrutura para uma extensa faixa de máquinas, enquanto o InteGrade tem seu foco na disponibilização do poder de processamento de estações de trabalho ociosas [G.04].

O serviço de informação de grade usado pelo *Globus Toolkit* é o MDS2 (*Monitoring and Discovery Service*). O MDS2 usa um arcabouço extensível para gerenciamento de informação estática e dinâmica sobre o status de uma grade computacional e todos os seus componentes. MDS2 é construido sobre o protocolo LDAP (*lightweight directory access protocol*) [XZS03].

GAIA é um sistema operacional com suporte para espaços ativos. Um espaço ativo é um modelo que mapeia a percepção abstrata de um espaço físico em como um sistema computacional. Os Serviços Gaia são implementados como "Componentes Unificados" (objetos CORBA), que executam no contexto do espaço ativo e são baseados no sistema operacional 2k [MR00]. O sistema operacional 2k é baseado em componentes, é reflexivo, adaptável e centrado no usuário.

O serviço de informação do Globus é voltado para prover recursos de informação para a grade e o serviço de informação do Gaia provê recursos de informação para o espaço ativo. Esses serviços têm funcionalidades muito mais abrangentes e complexas que as desejadas para o repositório de aplicações do Sistema InteGrade. O conhecimento desses serviços mostrou-se importante na definição de uma interface adequada para o repositório de aplicações do InteGrade.

Os protocolos de informação, execução e reserva do InteGrade são baseados no subsistema de gerenciamento de recursos do 2k [G.04]. A interface do repositório de aplicações do InteGrade é voltada apenas para tratar a persistência da aplicações executadas na grade e foi fortemente inspirada no repositório de componentes do Gaia.

5 O Repositório de Aplicações

O atual repositório de aplicações do InteGrade é muito simples e sua utilização é satisfatória apenas como prova de conceito. No entanto, o InteGrade pretende dar origem a uma vasta grade de computação com potencial para milhões de máquinas [G.04]. Nesse contexto, o repositório de aplicações deve atender a demandas mais sofisticadas.

Atualmente, o repositório de aplicações oferece apenas duas funcionalidades básicas: registrar uma aplicação e recuperar uma aplicação. Essas funcionalidades são usadas pelo módulo ASCT quando um usuário da grade deseja executar uma aplicação. O módulo LRM também participa desse caso de uso, intermediando a execução da aplicação. Além de limitadas, tais funcionalidades ainda apresentam problemas na identificação das aplicações registradas.

Contudo, o problema mais grave na atual implementação do repositório de aplicações é a ausência de persistência. Isso obriga ao usuário do InteGrade registrar uma aplicação sempre que desejar executá-la, posto que o repositório mantém referências para as aplicações registradas, apenas enquanto estiver ele próprio sendo executado. A qualidade e produtividade do sistema é, dessa forma, obviamente comprometida.

Esse contexto suscitou a necessidade de reimplementação do repositório de aplicações para prover, idealmente, as seguintes funcionalidades e características:

Persistência: armazenamento secundário das aplicações com facilidade de recuperação através de uma chave identificadora única; utilização de nomes compostos no registro das aplicações, facilitando a implementação de uma interface de usuário para navegação pelo espaço de nomes (diretórios); espaço de nomes contendo aplicações registradas e armazenadas provavelmente em um sistema de arquivos.

Versões: suporte para execução das aplicações em diferentes sistemas operacionais e controle de versões. Estruturas descrevendo aplicações e binários podem ser úteis para manter esse suporte.

Segurança: suporte à mecanismos que permitam colaborar com o módulo de segurança do InteGrade.

Replicação: mecanismos de replicação, para armazenar cópias de segurança do repositório de aplicações.

5.1 Semântica das funcionalidades do repositório de aplicações

A interface `ApplicationRepository` segue como Apêndice deste relatório. Essa interface está definida no arquivo `ApplicationRepository.idl`. As operações declaradas na interface permitem ao usuário manipular o espaço de nomes do repositório de aplicações do InteGrade de forma transparente, incluindo o lançamento de exceções específicas desse domínio, que deverão ser tratadas pelo usuário. Os nomes de diretório passados como parâmetros nas operações abaixo devem descrever o caminho absoluto com origem na raiz do espaço de nomes do repositório. As funcionalidades descritas abaixo correspondem às operações declaradas na interface `ApplicationRepository`.

Registrar uma aplicação no repositório - os nomes do diretório e da aplicação são os parâmetros passados para essa operação e servirão como chave única da aplicação.

Adicionar um código binário à uma aplicação registrada no repositório - essa operação associa o código binário passado como segundo parâmetro da operação, à aplicação registrada no repositório e descrita pela estrutura `AppDescription`, passada como primeiro parâmetro da operação.

Recuperar um código binário adicionado à uma aplicação no repositório - essa operação retorna um código binário de uma aplicação, identificado pelo nome do diretório, nome da aplicação e nome do binário, passados como parâmetros da operação.

Remover um código binário adicionado à uma aplicação no repositório - operação que exclui do repositório de aplicações um código binário de uma aplicação, identificado pelo nome do diretório, nome da aplicação e nome do binário, passados como parâmetros da operação.

Remover uma aplicação registrada no repositório - exclui uma aplicação registrada no repositório de aplicações usando os nomes do diretório e da aplicação, que são passados como parâmetros e servirão como chave única da aplicação.

Criar um diretório no repositório - o nome do diretório é passado como parâmetro da operação.

Remover um diretório criado no repositório - o nome do diretório é passado como parâmetro da operação.

Fornecer uma lista de arquivos e diretórios contidos em um diretório - retorna um arranjo de estruturas `ContentDescription` contendo o nome do arquivo ou diretório e um flag indicando sua natureza (*file* ou *directory*).

Na implementação foi utilizado o *plug-in* para Java do Eclipse, o ORB JacORB v1.4 e a ferramenta de compilação Ant. A utilização dessas ferramentas teve um impacto muito positivo para produtividade do trabalho de implementação. As principais classes implementadas são mostradas na Figura 2.

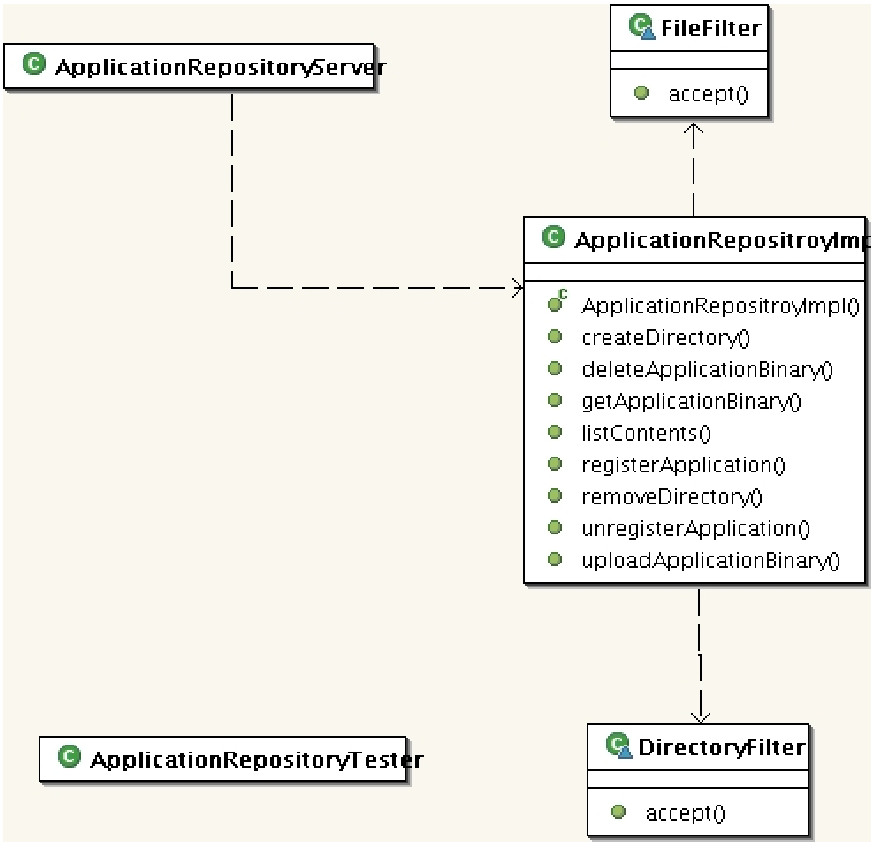

Figura 2: Diagrama de Classes - Repositório de Aplicações

6 Conclusão

Este relatório descreveu o problema de estender as funcionalidades do repositório de aplicações do InteGrade, a abordagem adotada para efetivação dessa tarefa, e os produtos resultantes de sua realização.

As atividades realizadas incluiram: estudo do código fonte do repositório de componentes do Gaia e do Globus, definição da interface IDL do repositório de aplicações, e a implementação do repositório de aplicações, do ambiente de testes e do *launcher* do servidor. O código-fonte na sua íntegra está disponível no seguinte endereço web: `http://www.lcpd.ime.usp.br/~vidal/apprepos`.

As funcionalidades associadas à persistência foram completamente atendidas. Utilizou-se, como previsto, o sistema de arquivos, possibilitando o uso de nomes compostos no registro e remoção das aplicações e de seus binários; a identificação unívoca das aplicações e seus binários e a navegação pelo espaço de nomes.

As interfaces `AppDescription` e `ContentDescription`, descrevem, respectivamente, uma aplicação e o conteúdo de um diretório. Essas interfaces contêm informações sobre as principais abstrações do repositório de aplicações, ou seja, as aplicações e suas localizações. O mecanismo de segurança do InteGrade, usará operações do repositório de aplicações para implementar esses mecanismos. Como proposta de trabalho futuro, o suporte para versões e os mecanismos de replicação, serão parte de uma segunda fase deste projeto, que por ser mais complexa vai exigir mais tempo para especificação e implementação.

Em nível pessoal, a realização dessa atividade de pesquisa se mostrou uma oportunidade apropriada para participar de forma ativa na construção do sistema InteGrade. Foi possível conhecer detalhadamente os atuais problemas relacionados ao desenvolvimento do sistema, participar das discussões voltadas para a solução desses problemas, e não menos importante, resultou em uma produtiva interação com os demais membros do grupo.

Referências

[G.04] Goldchleger A. Kon F. Goldman A. Finger M. Bezerra G. Integrade:object-oriented Grid middleware leveraging idle computing power of desktop machines. *Concurrency and Computation: Practice and Experience*, 4, 2004.

[K+04a] Fábio Kon et al. Integrade: checkpoints. *http://gsd.ime.usp.br/integrade/documentation/internal/checkpointings/*, 2004.

[K+04b] Fábio Kon et al. Integrade: Open Research Topics. *http://gsd.ime.usp.br/integrade/documentation/internal/openTopics.html*, 2004.

[MR00] Roy H. Campbell Manuel Roman. Gaia:enabling Active Spaces. 2000.

[XZS03] J. Freschl X. Zhang and J. Schopf. A Performance Study of Monitoring and Information Services for Distributed Systems. August 2003.

Apêndice

```
// new ApplicationRepository IDL interface
// @author Alexandre Cesar Tavares Vidal

module types{
  //*A serialized application


  struct AppDescription {
    string dirname;   /* the name of the direrctory (absolute pathname) */
    string appname;   /*  the name of the application file */
    string binaryname;/* the name of the application binary */
    string platform;  /* a binary platform especification HW+SW */

    /* um campo com uma URL apontando para a descricao da aplicacao ? */
    /* um campo string com uma descricao da aplicacao ? */
    /* ... */
  };

  struct ContentDescription {
        string filedirname; /* the name of a directory or file */
        string filedirflag; /* the flag dir or file */
  };

  typedef sequence<ContentDescription> Contents;

  typedef sequence<octet> Application;

  typedef sequence<string> StringSequence;

};


module interfaces{

  ///* ApplicationRepository defines a service that allows applications
  //   to be stored, retrieved and manipulated
  //*/
  interface ApplicationRepository{

    exception AppRegisterException {};
    exception DirectoryCreationException {};
    exception BinaryCreationException {};
    exception DirectoryNotFoundException {};
    exception AppNotFoundException {};
    exception BinaryNotFoundException {};
    exception DirectoryNotEmptyException {};

    ///*
    // Registers an application in the repository
    // @param dirname  the name of the direrctory (absolute pathname)
    // @param appname the name of the application
    // @raises DirectoryCreationException if it can’t create the directory
    // @raises AppRegisterException if it can’t to register the application file
    //*/
    void registerApplication(in string dirname, in string appame )
      raises (DirectoryCreationException, AppRegisterException);
```

```
///*
// Adds a binary code to an application in the repository
// @param appDesc application features needed to register an application
// @param app serialized application to be registered
// @raises DirectoryNotFoundException if it can’t create the directory
// @raises AppNotFoundException if the application does not exist
// @raises BinaryCreationException if it can’t create the binary code file
//*/
void uploadApplicationBinary(in types::AppDescription appDesc,
                             in types::Application app)
  raises (DirectoryNotFoundException, AppNotFoundException,
          BinaryCreationException);


///*
// Retrieves a binary file added to an application in the repository
// @param dir\_name  the name of the directory (absolute pathname)
// @param app\_name the name of the application
// @param binary\_name the name of the binary file
// @raises DirectoryNotFoundException if the dirname does not exist
// @raises AppNotFoundException if the application does not exist
// @raises BinaryNotFoundException if the bynary does not exist
// @returns the serialized application
//
//*/
types::Application getApplicationBinary(in string dir\_name,
                                        in string app\_name,
  raises (DirectoryNotFoundException, AppNotFoundException,
          BinaryNotFoundException);


///*
// Deletes an application binary stored on the ApplicationRepository
// @param dir\_name  the name of the directory (absolute pathname)
// @param app\_name the name of the application
// @param binary\_name the name of the binary file
// @raises DirectoryNotFoundException if the dirname does not exist
// @raises AppNotFoundException if the application does not exist
// @raises BinaryNotFoundException if the bynary does not exist
//
//*/
void deleteApplicationBinary(in string dir\_name, in string app\_name,
  raises (DirectoryNotFoundException, AppNotFoundException,
          BinaryNotFoundException);


///*
// Unregisters an application registered on the ApplicationRepository
// @param appname the name of the application
// @param dirname  the name of the direrctory (absolute pathname)
// @raises DirectoryNotFoundException if the dirname does not exist
// @raises AppNotFoundException if the filename does not exist
// @raises DirectoryNotEmptyException if the directory is not empty
//
//*/
void unregisterApplication (in string dirname, in string appname)
  raises (DirectoryNotFoundException, AppNotFoundException,
          DirectoryNotEmptyException);

//*
```

```
    // Creates the directory with the provided dirname
    // @param dirname the name of a directory
    // @raises DirectoryCreationException if it can't create the directory
    //*//
    void createDirectory(in string dirname)
      raises (DirectoryCreationException);


    ///*
    // Returns the directories and files names under the provided dirname
    // @param dirname the name of a directory
    // @return an array of ContentDescription structures
    // @raises DirectoryNotFoundException if the dirname does not exist
    //*/
    types::Contents listContents(in string dirname)
       raises (DirectoryNotFoundException);


    //*
    //  Removes the directory with the provided dirname
    //  @param dirname the name of a directory
    //  @raises DirectoryNotFoundException if the dirname does not exist
    //  @raises DirectoryNotEmptyException if the directory is not empty
    //*//
    void removeDirectory(in string dirname)
      raises (DirectoryNotFoundException, DirectoryNotEmptyException);
  };
};
```