Integração AppMan e PBS

Trabalho Final de Disciplina

Rômulo Bandeira Rosinha

CMP157 - Programação Distribuída e Paralela - 05/2 - Prof. Cláudio Geyer

Dezembro 2005

1 Introdução

Este trabalho apresenta a modelagem e a implementação de um *wrapper* Java para o gerenciador de recursos PBS, tendo como princípio básico seguir a especificação DRMAA.

O trabalho apresenta inicialmente o contexto em que está inserido, introduzindo o GRAND e o AppMan e apresentando a motivação. Seguem uma introdução sobre o PBS e outra sobre a especificação DRMAA. A modelagem e a implementação são apresentadas em seguida, formando o corpo principal do trabalho. Para encerrar, considerações finais são feitas a respeito do trabalho realizado e dos resultados obtidos.

O objetivo principal é realizar a integração entre o AppMan e o PBS, possibilitando que o AppMan delegue a tarefa de executar trabalhos ao PBS através de uma interface Java bem definida.

1.1 GRAND

O modelo GRAND (Grid Robust Application Deployment) [1] oferece gerenciamento e monitoração de aplicações que envolvem um número muito grande de tarefas em ambientes de grade. Ele fornece um modelo de arquitetura para ser implementado a nível de middleware, conforme a figura 1. Como entrada recebe uma linguagem simples de descrição da aplicação. As dependências entre tarefas são definidas através de arquivos de dados. O usuário descreve apenas as características de cada tarefa, as dependências entre elas são inferidas automaticamente através de de uma análise de dependência do fluxo de dados.

O modelo GRAND trata três questões importantes no contexto de aplicações que envolvem um grande número de tarefas:

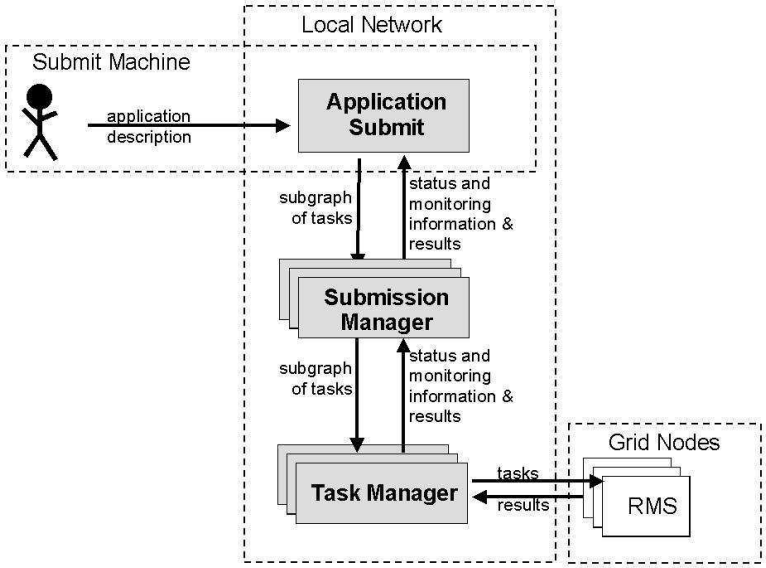

Figura 1: Arquitetura GRAND

1. particionar aplicações de modo que tarefas dependentes sejam alocadas no mesmo nó de uma grade para evitar migração desnecessária de arquivos de dados intermediários

2. particionar aplicações de modo que tarefas sejam alocadas próximas aos seus arquivos de dados de entrada

3. distribuir o processo de submissão de modo que a máquina de submissão não fique sobrecarregada

1.2 AppMan

O AppMan (Application Manager) [2] é uma implementação simplificada do modelo GRAND. O AppMan é implementado em Java para possibilitar portabilidade. Ele usa a ferramenta JavaCC para analisar gramaticalmente e interpretar a linguagem GRID-ADL. O ambiente de execução do AppMan usa os serviços fornecidos pelo middleware Exehda que possibilitam execução remota e monitoração. O AppMan implementa as principais funções do modelo GRAND, incluindo a submissão distribuída de tarefas e a monitoração da aplicação, fornecendo retorno para o usuário.

Uma instância do AppMan e do Exehda precisa estar presente em cada nó participante da grade. Todos os nós podem submeter e executar tarefas. Os arquivos de dados de entrada devem estar disponíveis em um servidor web. O

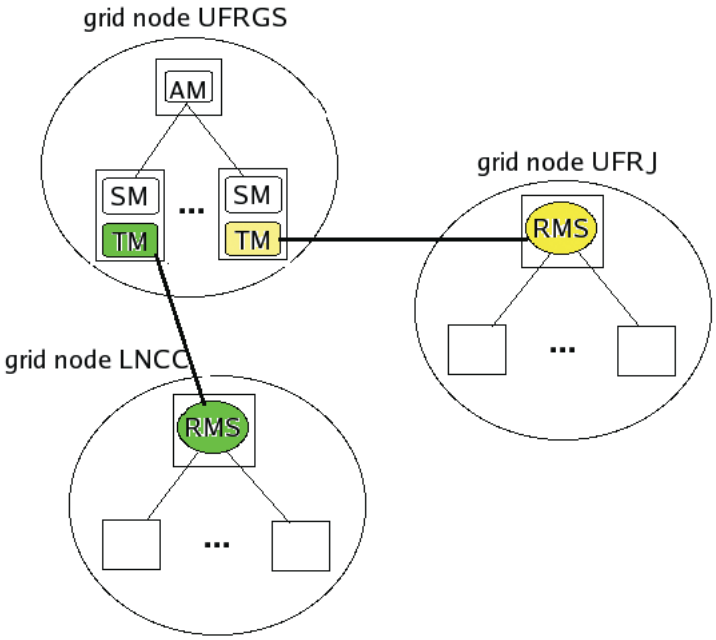

Figura 2: Cenário com o AppMan utilizando vários gerenciadores de recursos

grafo da aplicação é construído em memória e o Application Manager (AM) é iniciado. O AM particiona o grafo usando um algoritmo de agregação. Então, o AM instancia um ou mais Submission Managers (SM) e distribui sub-grafos para os SMs. Os arquivos de entrada e os executáveis são obtidos do servidor web. Os SMs atualizam o AM com o progresso da execução das tarefas. Cada SM, independentemente, verifica a lista de nós disponíveis e aleatoriamente escolhe um para executar a tarefa. Imediatamente um Task Manager (TM) é instanciado, ficando responsável por criar a tarefa remota e monitorá-la até que a sua execução termine.

Esses são os principais passos na implementação atual. No entanto, o AppMan deveria possibilitar o escalonamento através de gerenciadores de recursos (RMS). Nesse cenário mais desejável apresentado na figura 2, o TM em um nó da grade contata diretamente o gerenciador de recursos em um nó remoto da grade para executar as tarefas.

1.3 Motivação

Como apresentado na seção anterior, a implementação atual do AppMan faz um escalonamento aleatório das tarefas nos nós disponíveis de uma grade. Um cenário mais desejável também foi mostrado, onde o AppMan seria inte-

grado com gerenciadores de recursos, possibilitando a execução em ambientes maiores com menos complexidade de instalação do Exehda.

Resumindo, essa integração do AppMan com um gerenciador de recursos possibilitaria:

- utilizar um número maior de nós de processamento
- integrar grades dispersas geograficamente para execução de uma aplicação com baixa complexidade de software
- facilitar a realização de testes experimentais

Não foi encontrado nenhuma implementação em Java que possibilite uma interface simples e robusta entre o AppMan e o PBS, gerenciador de recursos mais popular e disponível em vários nós das grades acessíveis pelos participantes do projeto. Essa é a motivação para este trabalho: implementar uma interface em Java para possibilitar que o AppMan contate o PBS para a submissão de tarefas.

2 PBS

O PBS (Portable Batch System) [3] é um sistema que extende um sistema operacional POSIX ou Unix e que permite a submissão e controle de jobs em um ambiente de batch, independente de um ambiente interativo.

O propósito do PBS é de oferecer controles adicionais sobre a inicialização e o escalonamento da execução de trabalhos tipo batch, além de permitir que esses jobs sejam direcionados para diferentes nós de uma rede. O PBS permite que um site defina e implemente políticas sobre que tipos de recursos e quando de cada recurso pode ser usado por diferentes jobs. O PBS também fornece um mecanismo com o qual o usuário pode garantir que o job terá acesso aos recursos necessários para que ele seja completado.

Para interface com o PBS, são oferecidas duas maneiras: um conjunto de scripts para linha de comando e uma biblioteca de programação. Ambas permitem acesso a todas as funcionalidades que o PBS oferece. A biblioteca é escrita usando a linguagem C e é chamada Batch Interface Library (IFL), possibilita a construção de novos clientes de acordo com a necessidade de cada usuário.

3 DRMAA

A Distributed Resource Management Application API (DRMAA) [4] é a especificação de uma API genérica para gerenciadores de recursos distribuídos

que tem como objetivo facilitar a integração de aplicações. O escopo da DRMAA é limitado à submissão de jobs, monitoração e controle de jobs e obtenção do estado final da execução de um job. A DRMAA oferece aos programadores de aplicações e as desenvolvedores de gerenciadores de recursos um modelo de programação que possibilita a construção de aplicações distribuídas fortemente acopladas com os gerenciadores de recursos usados. A API também preserva a flexibilidade no momento da implantação do sistema, permitindo que se utilizem os gerenciadores de recursos disponíveis.

A especificação DRMAA foi desenvolvida por um grupo de trabalho do Global Grid Forum e a sua primeira versão foi publicada em Junho de 2004. Inicialmente desenvolvida em cima da linguagem C, apresentou mais tarde uma especificação para a linguagem Java. Muito deste trabalho com Java foi feito pela equipe de desenvolvimento do gerenciador de recursos da Sun, o Sun Grid Engine. Existem versões disponíveis publicamente da API tanto na linguagem C quanto em Java, mas poucas implementações foram encontradas, nenhuma para o PBS.

4 Implementação

A maneira escolhida para realizar a integração entre o AppMan e o PBS foi através da utilização da tecnologia Java Native Interface (JNI), que permite acesso a código em outras linguagens, C por exemplo, a partir de código Java. Com essa escolha, foi feita a implementação de um wrapper que expõe as funções da biblioteca IFL do PBS através de métodos Java simples.

4.1 Programação JNI

A programação usando a tecnologia JNI [5] envolve uma série de passos que precisam ser executados para se obter o resultado desejado. Esses passos são enumerados a seguir.

4.1.1 Definir a classe Java com métodos nativos

O primeiro passo consiste em definir quais os métodos de uma classe Java serão implementados usando uma linguagem nativa. Isso é feito através da utilização do modificador *native* na declaração de um método. No caso deste trabalho, seguindo a especificação DRMAA, a classe Java `pbswrapper.drmaa.SessionImpl` contém todos os métodos nativos necessários. Por exemplo, abaixo está a declaração do método nativo de controle de jobs encontrado nessa classe:

```
private native void nativeControl (String jobId, int action)
  throws DrmaaException;
```

4.1.2 Gerar um arquivo de cabeçalho em C

Esse passo envolve executar a ferramenta javah, encontrada de maneira padrão no JDK, especificando a classe Java que contém os métodos nativos. Nesta implementação o comando é o seguinte:

```
$JAVA_HOME/bin/javah \
  -classpath ./:../lib/jdrmaa.jar pbswrapper.drmaa.SessionImpl
```

Nesse comando são especificados o classpath, onde são encontradas todas as classes dependentes, e o nome completo da classe que contém os métodos nativos. Como saída desse comando, é gerado um arquivo de cabeçalho em C com o nome `pbswrapper_drmaa_SessionImpl.h` contendo a definição das funções que deverão ser implementadas.

4.1.3 Implementar em C a interface definida

O próximo passo é o de escrever o código em C para implementar a interface definida pelo arquivo gerado no pasos anterior. Para esta implementação são incluídos, além do arquivo de cabeçalho gerado anteriormente, os cabeçalhos necessários para a utilização da biblioteca IFL, no caso `pbs_error.h` e `pbs_ifl.h`. Esse último arquivo contém a definição de todos as funções disponíveis para interação com o PBS, funções essas que serão chamadas para implementar a funcionalidade de cada função definida originalmente pela classe Java contendo os métodos com o marcador *native*.

4.1.4 Gerar uma biblioteca de código compartilhada

O passo de geração da biblioteca compartilhada consiste apenas do seguinte comando:

```
gcc --shared --Wl,--soname,libpbsdrmaa \
  pbswrapper\_drmaa\_SessionImpl.c --o libpbsdrmaa.so --lpbs
```

O comando acima gera uma biblioteca compartilhada com o nome `libpbsdrmaa.so` a partir da compilação do arquivo `pbswrapper_drmaa_SessionImpl.c`, utilizando a biblioteca `pbs` (no sistema de arquivos em /usr/local/lib/libpbs.a). Essa é a mesma biblioteca IFL referida anteriormente e é gerada através da compilação do cliente PBS.

4.1.5 Carregar a biblioteca a partir do código Java

Tendo a biblioteca compilada, o último passo é carregar a biblioteca a partir do código Java para que ela possa ser utilizada. O seguinte trecho de código presente na classe `pbswrapper.drmaa.SessionImpl` é responsável por realizar a operação de carga.

```
static {
  AccessController.doPrivileged (new PrivilegedAction () {
   public Object run () {
     System.loadLibrary ("pbsdrmaa");
     return null;
   }
  });
}
```

4.2 Exemplo

A implementação realizada neste trabalho permite a submissão para o PBS de um job. Para exemplificar, foi criado um script, chamado `testpbs.sh`, que exibe algumas informações sobre o nó onde o script for executado:

```
#!/bin/sh
#testpbs
echo This is a test
echo Today is ‘date‘
echo This is ‘hostname‘
echo The current working directory is ‘pwd‘
ls -alF /home
uptime
```

O código fonte do programa Java é bastante simples: apenas instancia um objeto `org.ggf.drmaa.Session`, conecta ao servidor PBS, instancia um objeto `org.ggf.drmaa.JobTemplate`, popula algumas propriedades, envia o job para o PBS, exibe o identificador gerado do job e desconecta do servidor. O método main é o seguinte:

```
public static void main(String[] args) {

  // instancia um objeto Session
  SessionFactory factory = SessionFactory.getFactory();
  Session session = factory.getSession();
```

```
    try {
      // conecta ao servidor
      session.init("moe");

      // instancia um objeto JobTemplate
      JobTemplate jt = session.createJobTemplate();

      // popula propriedades, nome do script no caso
      jt.setRemoteCommand("testpbs.sh");

      // envia o job para o PBS
      String id = session.runJob(jt);

      // exibe o identificador do job
      System.out.println("Job submetido com id " + id);

      // desconecta do servidor PBS
      session.exit();

    } catch (DrmaaException e) {
      System.out.println("Error: " + e.getMessage());
    }
  }
```

Para executar o programa é usado o seguinte comando:

```
java -Djava.library.path=/usr/local/lib/
  -Dorg.ggf.drmaa.SessionFactory=pbswrapper.drmaa.SessionFactoryImpl
  -cp bin/:lib/jdrmaa.jar pbswrapper.PBSWrapperTester
```

Duas propriedades são passadas para a JVM: (i) `java.library.path`: indica onde as bibliotecas nativas devem ser procuradas; (ii) `org.ggf.drmaa.SessionFactory`: indica o nome da classe que implementa a classe abstrata de mesmo nome e que será usada para instanciar objetos tipo `org.ggf.drmaa.Session`.

A execução desse script gera um arquivo de saída no nó onde ele foi executado, no diretório home do usuário que foi usado para a execução. Em uma execução de exemplo foi gerado o arquivo `none.o32` com o seguinte conteúdo:

```
This is a test
Today is Sex Jan 13 11:38:27 BRST 2006
This is moe
The current working directory is /home/rbrosinha/tf-pdp/pbs-wrapper-drmaa/
total 30
```

```
drwxrwsr-x   21 root      staff      504 2005-11-07 15:52 ./
drwxr-xr-x   25 root      root       712 2005-12-15 11:24 ../
drwxr-xr-x   14 clairmont clairmont  760 2003-12-04 09:06 clairmont/
drwxr-xr-x   28 egon      egon      1240 2005-09-28 16:35 egon/
drwxr-xr-x   19 frainer   frainer    904 2005-08-04 13:12 frainer/
dr-xr-xr-x    8 root      root       200 2004-03-26 10:40 ftp/
drwxr-xr-x   10 fwf       fwf        480 2004-03-09 12:03 fwf/
drwxr-xr-x    5 guilherme guilherme  240 2004-11-16 14:05 guilherme/
drwxr-sr-x    2 root      staff      192 2004-02-20 17:27 images/
drwxr-xr-x   14 isam      isam      1592 2003-09-12 08:31 isam/
drwxr-xr-x   26 jquadro   jquadro   1440 2005-11-14 18:04 jquadro/
drwxr-xr-x   12 laurino   laurino    600 2004-03-08 11:18 laurino/
drwxr-sr-x   17 root      staff      592 2005-06-15 15:07 local/
drwxr-xr-x   92 lucasa    lucasa    4224 2005-12-08 15:31 lucasa/
drwxr-xr-x   12 lucc      lucc       568 2005-04-20 18:20 lucc/
drwxr-xr-x    4 mcmoraes  mcmoraes   280 2005-02-16 13:08 mcmoraes/
drwxrwxrwx    2 root      staff       48 2004-05-26 18:15 movies/
drwxr-xr-x   41 rbrosinha rbrosinha 2600 2006-01-13 11:36 rbrosinha/
drwxr-xr-x  115 rreal     rreal     7056 2004-11-26 18:02 rreal/
drwxr-xr-x    3 wine      wine       160 2003-10-13 15:22 wine/
 11:38:27 up 1 day, 20:45,  3 users,  load average: 1.65, 0.72, 0.37
```

5 Considerações Finais

Através da pesquisa realizada pode se chegar a avaliação que a especificação DRMAA apresenta baixa adesão. Apesar de já ter sido publicada há mais de um ano, são poucas as implementações de sua API encontradas publicamente. Um dos mais populares gerenciadores de recursos, o PBS, ainda não apresenta nenhuma implementação, seja na linguagem C ou na linguagem Java. As principais implementações encontradas são para o gerenciador de recursos Sun Grid Engine, da Sun, empresa que participou ativamente da definição da especificação DRMAA.

Este trabalhou apresentou uma implementação parcial de um wrapper Java que segue a especifição DRMAA para o gerenciador de recursos PBS. A principal contribuição é fornecer uma solução para possibilitar a integração entre o AppMan e o PBS, seguindo padrões especificados pelo Global Grid Forum e não dependendo de outros componentes de software para o seu funcionamento.

Com este trabalho espera-se que a experimentação com o AppMan se torne mais simples e robusta, permitindo que um conjunto maior de resultados possa ser obtido e que o modelo GRAND tenha melhores avaliações práticas para as hipóteses que levanta.

Referências

[1] GRAND. 2005. Disponivel em: <http://www.cos.ufrj.br/~kayser/grand/>.

Acesso em: dezembro 2005.

[2] APPMAN. 2005.
Disponível em: <http://www.cos.ufrj.br/∼kayser/grand/appman/>.
Acesso em: dezembro 2005.

[3] PBS. 2005. Disponível em: <http://www-unix.mcs.anl.gov/openpbs/>.
Acesso em: dezembro 2005.

[4] DRMAA. 2005. Disponivel em: <http://www.drmaa.org>. Acesso em:
dezembro 2005.

[5] JNI. 2005. Disponível em: <http://java.sun.com/j2se/1.4.2/docs/guide/jni/>.
Acesso em: dezembro 2005.