JGridTS: Um framework Java para Execução de Tarefas em Ambientes de Grades Computacionais Baseadas em Espaços de Tuplas

Tiago Lazarotto[1], Robison Cris Brito[2], Fábio Favarim[3]
[1,2,3]Universidade Tecnológica Federal do Paraná

Grades Computacionais (*Grid Computing*) consistem da agregação de recursos heterogêneos conectados em rede, formando um sistema distribuído de larga escala, a fim de coletivamente resolver problemas de computação intensiva. Nesse tipo de ambiente, a execução de aplicações consiste em quebrar um problema científico computacional grande em pequenas tarefas e atribui-las aos recursos que fazem parte desse ambiente. A essa atribuição dá-se o nome de escalonamento.

Um dos problemas desses ambientes é fazer o uso efetivo dos recursos, isto é, fazer um bom escalonamento. O escalonador é o componente responsável por essa tarefa, e para tal, necessita saber quais recursos estão disponíveis e suas características.

A agregação das informações sobre os recursos que compõem uma grade é feita normalmente por um serviço de informação. O processo de agregar informações sobre recursos de uma grade é conhecido na literatura de sistemas distribuídos, como estado global.Isto é como obter a "fotografia" da grade em um determinado instante.

Já foi provado na literatura que a obtenção de estados globais em sistemas distribuídos assíncronos (como por exemplo as Grades Computacionais), não possuem solução (CHANDY; LAMPORT, 1985). Portanto, o escalonamento baseado em informações, fornecidas por um serviço de informação, esta sujeito a definir atribuições que não se efetivam, devido a desatualização das informações usadas pelo escalonador, reduzindo assim a sua a eficácia.

Favarim, Fraga e Lung (2012), propuseram uma infraestrutura (GridTS) que provê uma nova solução para o escalonamento, no qual não se tem um escalonador global, mas cada recurso faz o escalonamento de acordo com suas disponibilidades e poder computacional.

No GridTS, são os recursos que buscam as tarefas mais apropriadas para suas condições de execução, ou seja, é invertida a ordem tradicional na qual os escalonadores é que buscavam os recursos disponíveis para a execuçãodas tarefas. A ideia por trás dessa concepção é distribuir a tarefa do escalonamento. A solução proposta não faz uso de um serviço de informação e mesmo assim, permite decisões do escalonamento serem feitas com informações atualizadas. O GridTS é baseado fortemente no modelo de comunicação generativa conhecido na literatura por Espaço de Tuplas (GELERNTER, 1985). O modelo de comunicação fornecido pelos Espaços de tuplas é desacoplado no tempo e no espaço, ou seja, os processos que se comunicam não precisam estar executando ao mesmo tempo e assim como não sabem a localização um dos outros.

De forma resumida, no GridTS, o usuário que possui tarefas a serem executadas, as insere juntamente com suas especificações, no Espaço de Tuplas, ou ainda, pode encaminhar a um *broker* que fica então responsável por dividir a aplicação em tarefas menores e inserir no espaço de tuplas.

Os recursos disponíveis na grade, buscam nesse Espaço de Tuplas por tarefas que são capazes de executar segundo as especificações da tarefa. Ao concluir a execução, o recurso disponibiliza no Espaço de Tuplas o resultado da execução da tarefa e, por fim, o usuário obtém esse resultado do Espaço de Tuplas, ou no caso do *broker*, este obtém os resultados das tarefas e encaminha ao usuário.

Outro ponto importante é que a infraestrutura do GridTS, assim como a maioria das infraestruturas de grade existentes, ainda não cobrem todas as características de um sistema distribuído real, pois, somente suportam aplicações do tipo *bag-of-tasks* (SMITH, 1996). As aplicações *bag-of-tasks* são formadas por tarefas independentes, cuja execução não depende das demais, ou seja, não há nenhuma comunicação ou sincronização entre os recursos que as processam.

Desta forma, este trabalho visa também estender a infraestrutura do GridTS de modo a suportar outros tipos de aplicações, como aquelas cujas tarefas precisam se comunicar entre si. Tais aplicações são conhecidas como aplicações acopladas.

Atualmente, não existem implementações de grades computacionais que usam Espaço de Tuplas. O modelo de escalonamento do GridTS somente foi testado em um ambiente simulado, que foi construído somente para validação do modelo, não possuindo uma implementação para uso em ambientes reais. Assim, há uma grande lacuna entra a infraestrutura proposta e a implementação da mesma, que é inexistente. Deste modo, neste trabalho é proposto um *framework* para implementação da infraestrutura, a qual é denominadoJGridTS. O JGridTSestá em desenvolvimento e sendo escrito na linguagem de programação Java. O JGridTS está disponibilizado com a licença GPL v3, no endereço http://code.google.com/p/jgridts. Todos os componentes da infraestrutura do GridTS foram mantidos e representados por classes, descritas a seguir:

Task: consiste de uma interface que representa a tarefa a ser executada. Essa classe possui o método execute() o qual deve ser implementado de acordo com o que a tarefa deve executar. Deste modo, cada aplicação deve implementar a classe Task de acordo com a sua especificação. O Algoritmo 1 representa a interface Task.

Algoritmo 1: Task

```
1:inteface Task(){
2:   public Object[] execute(Tuple data);
3:   public Object[] getDependency(Object[] req);
4:   public void setDependency(
     ArrayList<Object[]> dep);
5:   public Object[] reqForDep();
6:   public boolean isDepNeeded();
7:}
```

Broker: fornece o método processJob() que deve ser implementado por cada broker específico de aplicação, isto é, nesse método é implementado como a aplicação deve ser dividida em tarefas.

Algoritmo 2: Broker

```
1: public void generateTasks(Object...
   application) {
2:   for(int i = 1; i < n; i++) {
3:     setDependecy(ID_tasks);
4:     buildTuples(application);
5:   }
6: }
7: public Object[] joinResults(ArrayList<Tuple>
   results){
8:   for(int i = 1; i < n; i++) {
9:     concatenateResults(results[i],endResult);
10:  }
11:  return(endResult);
12:}
```

Brokers já desenvolvidos podem ficar disponíveis para outros usuários que pretendam usar a grade para a execução da mesma aplicação. Um exemplo de implementação está demonstrado no Algoritmo 2.

Espaço de Tuplas: qualquer implementação de Espaço de Tuplas existente pode ser utilizada, como o DepSpace (CORREIA, 2008), JavaSpaces (Sun, 2003) e IBM Tspaces (LEHMAN, 2001). Isto é alcançado através da utilização do padrão de projeto Adapter, isto é, para qualquer implementação do Espaço de Tuplas, bastar implementar a classe TSAdapter. Essa classe provê a assinatura dos métodos propostos pela linguagem Linda (GELERNTER, 1985). A Figura 1 demonstra a arquitetura do *frame-work* desenvolvido.

Figura 1. Utilização do Espaço de Tuplas pelo TSAdapter

Recurso: essa classe basicamente tem a função de buscar no espaço de tuplas por uma tarefa e invocar o seu método execute(). Após a execução da tarefa, insere no Espaço de Tuplas o resultado da tarefa. Caso a tarefa dependa do resultado parcial ou total de outra tarefa, ou seja, caso a mesma possua uma dependência, então uma comunicação entre os dois recursos dependentes será estabelecida e uma troca de informações se iniciará, fazendo com que os recursos tenham todas as informações necessárias para processar suas tarefas. O Algoritmo 3 demonstra este processo.

Algoritmo 3: Recurso

```
1: ticket = ts.rd("TICKET");//ts-espaço detuplas
2: Tuple job = ts.rd("JOB", ticket);
3: Tuple task = ts.in("TASK", "*",job[APPNAME]);
4: if (hasRequirements(task)) {
5:   if (hasDependencies()) {
6:     Sender();
7:     Retreiver();
8:   }
9:   result = execute(task);
10:  ts.out(result);
11:}
```

Usando o JGridTS, para fazer uso da grade um usuário precisa somente implementar a classe *broker* e a classe *Task*. Para colocar o ambiente de grade em funcionamento, basta instanciar o Espaço de Tuplas e em cada recurso da grade instanciar a classe Recurso.

A partir disso, um usuário pode fazer uso da grade inserindo no Espaço de Tuplas as tarefas e posteriormente obtendo os resultados. O usuário também pode fazer uso de *brokers* de aplicação que já estejam implementados, instanciando-os em seu computador ou usando instâncias que já estejam executando em outros computadores.

Caso ainda não exista o *broker* específico para sua aplicação implementado, o mesmo pode ser implementado pelo usuário de modo

que em execuções futuras o mesmo possa ser reutilizado. A Figura 2 ao lado demonstra como o sistema funciona.

Figura 2. Infraestrutura do JGridTS

A implementação em Java proporciona a execução do *framework* na maioria dos sistemas operacionais existentes, assim permitindo a sua execução em uma grande quantidade de máquina que possam agir como recursos e também um grande número de potenciais usuários.

Com o JGridTS é possível se utilizar de Grades Computacionais que satisfaçam as necessidades de cientistas, pesquisadores e empresas que necessitam de liberdade de sistema e alto poder computacional, sem a necessidade de supercomputadores, apenas se utilizando de todos os computadores que já estão disponíveis aos mesmos.

Agradecimentos

Agradecemos a Fundação Araucária por ter financiado parcialmente este trabalho.

Referências

FAVARIM, F.; FRAGA, J. S; LUNG, L.C. Towards an opportunistic grid scheduling infrastructure based on tuple spaces.Journal of Internet Services and Applications, v.3, 2012.

CHANDY, K. M.; LAMPORT, L. "Distributed snapshots: determining global states of distributed systems". ACM Transactions Computer Systems, 3(1):63–75, 1985.

GELERNTER, D. "Generative Communication in Linda". ACM Transactions on Programming Languages and Systems, 7(1):80–112, janeiro de 1985.

T. J. LEHMAN, A. COZZI, Y. XIONG, J. GOTTSCHALK, V. VASUDEVAN, S. LANDIS, P. DAVIS, B. KHAVAR, and P. BOWMAN. Hitting the distributed computing sweet spot with TSpaces. Computer Networks, 35(4):457–472, Mar. 2001.

Sun Microsystems. JavaSpaces Service specification. 2003.

BESSANI, A. N., ALCHIERI, E., CORREIA, M., E FRAGA, J. S. "DepSpace: A Byzantine Fault-Tolerant Coordination Service". In Proceedings of the 3rd ACM/SIGOPS/EuroSys

European Conference on Computer Systems (EuroSys 2008), April 2008.