Software livre na construção de grades computacionais para processamento de alto desempenho

André Leon S. Gradvohl[1,2]

[1]Universidade São Francisco – São Paulo – Brasil

[2]Centro Nacional de Processamento de Alto Desempenho em São Paulo – Brasil

andre.gradvohl@saofrancisco.edu.br

Resumo. *Um dos grandes desafios que se impõem à Computação é a possibilidade de realizar cálculos de forma cada vez mais rápida e o menos custosa possível. Para tanto, uma das estratégias para aumentar a velocidade de obtenção desses resultados é o chamado processamento de alto desempenho (PAD), combinado com o conceito de grades computacionais. Desta forma, esse artigo apresenta uma proposta de infra-estrutura para uma grade computacional para PAD, baseada em softwares livres indicados ao longo do texto. A proposta foi implementada para testes no Centro Nacional de Processamento de Alto Desempenho em São Paulo (CENAPAD-SP).*

Abstract. *One of the great challenges imposed to Computer Science is the ability to perform calculations in a fastest and cheapest way possible. Thus, one of the strategies available to increase the speed to obtain those results is the so-called high performance computing (HPC), combined with the grid computing concept. Therefore, this paper presents a proposal of a HPC grid infrastructure based on free software indicated through the text. The proposal was implemented on the National Center for High Performance Computing in São Paulo (CENAPAD-SP).*

1. Introdução

Discutir o uso de software livre para a construção de grades computacionais para processamento de alto desempenho (PAD) parece redundante, afinal, por definição, uma grade computacional utiliza padrões abertos [Foster and Kesselman 2003]. No entanto, embora na academia muito se estude e pesquise sobre o uso das grades computacionais, ainda não há consenso sobre a infra-estrutura mínima, em termos de software, para montar uma grade.

Assim, neste artigo propõe-se uma arquitetura básica para a construção de uma grade computacional, baseada em softwares livres e padrões abertos. O intuito não é fechar a questão sobre os softwares necessários, mas indicar uma combinação de programas que implemente uma grade computacional para PAD. É evidente que existem outras combinações possíveis, mas a que se propõe nesse artigo é uma das que tem melhor documentação e facilidade de implementação. Por essas razões, serve de um bom início para a implementação de grades computacionais para PAD.

Portanto, uma vez definido o conceito de grade computacional para PAD, o restante deste texto adota a seguinte estrutura:

- na Seção 2 define-se uma grade computacional para PAD;
- na Seção 3 é apresentada uma estrutura mínima de grade computacional para PAD, com sugestões de softwares para implementá-la;
- a Seção 4 propõe uma arquitetura de grade computacional para PAD;
- a Seção 5 apresenta as conclusões do trabalho.

2. Definição de grade computacional para PAD

Antes de se propor qualquer arquitetura de software, é conveniente descrever uma grade computacional para PAD. De acordo com [Jacob et al. 2005], uma grade computacional é um sistema pelo qual o usuário obtém acesso a recursos computacionais com pouco ou nenhum conhecimento sobre a localização ou a tecnologia que esses recursos utilizam.

Note-se que o conceito de grade em si não é recente. A independência de localização e de tecnologias são conceitos essenciais em sistemas distribuídos. A diferença chave das grades computacionais em relação aos sistemas distribuídos tradicionais é a integração de padrões abertos. As idéias embutidas nas grades computacionais conduzem a um outro conceito chamado virtualização [Plaszczak and Wellner 2005].

Virtualização é uma técnica que esconde características de hardware e software dos componentes de um sistema, de forma que o usuário tenha a impressão de um sistema único e de uma capacidade superior àquele que utiliza usualmente. Em outras palavras, implementar o conceito de virtualização significa aumentar o poder de utilização do sistema sem que o usuário perceba onde estão os recursos computacionais e que tecnologias usam.

Pode-se perceber, então, que o conceito de grade computacional pode ser utilizado por uma série de aplicações e não somente aquelas que necessitam de alto desempenho. Contudo, como o foco deste artigo é em processamento de alto desempenho é importante definir o que significa PAD. Assim, computação ou processamento de alto desempenho é a utilização de supercomputadores ou agrupamentos (*clusters*) de computadores para realizar cálculos que, em uma única unidade de processamento, demorariam muito tempo. Na maioria das aplicações, esses cálculos envolvem operações com vetores e matrizes.

Atente-se para a convergência entre os dois conceitos. De um lado as grades computacionais provêem a virtualização de recursos; de outro, PAD demanda várias unidades de processamento. Portanto, uma grade computacional para PAD é um sistema que agrega vários recursos computacionais, eventualmente distante geograficamente, para realizar cálculos de forma distribuída e apresentar seus resultados.

3. Componentes da grade para PAD

Pode-se dividir a estrutura de uma grade computacional para PAD em três níveis, a saber:

- nível básico: engloba o gerenciamento dos recursos computacionais (e. g. unidades de processamento, memória principal e discos), disponíveis em uma única organização ou sítio;
- nível de integração: interliga as organizações ou sítios formando a chamada "organização virtual".

- nível de apresentação: ponto de interação ou interface do usuário com a grade computacional.

Em cada um dos níveis mencionados, há um conjunto de softwares, geralmente abertos, que provêem as funcionalidades necessárias. A seguir, cada um desses níveis é detalhado.

3.1. Nível básico

Para o nível básico, o software essencial é o sistema operacional (SO). É possível utilizar, nesse nível, SOs proprietários. Contudo, as dificuldades de compilação do núcleo e a personalização que esses sistemas impõem, não os recomenda para compor a estrutura de grade computacional. Há ainda a questão dos custos envolvidos que, nos sistemas proprietários, tendem a ser maiores.

Por outro lado, SOs livres como Scientific Linux e CentOS são boas alternativas, pois são mais leves, uma vez que possuem as funcionalidades mínimas requeridas de um SO. Entre essas funcionalidades estão os sistemas de gerenciamento de memória principal, de gerenciamento de processos e conectividade.

Outros detalhes típicos de sistemas monousuários como gerenciamento de dispositivos de entrada e saída (e. g. portas USB, placas de vídeo, entre outros) não são tão necessários em SOs para grades. Enfim, o que importa em SOs para grades computacionais é a redução no consumo de recursos gastos pelo próprio SO em detrimento dos recursos disponíveis para os demais processos em execução.

Deve-se considerar também a facilidade para incorporar novas bibliotecas e interfaces de programação de aplicações (*Application Programming Interfaces* - API), em especial aquelas para PAD, por exemplo, MPI (*Message Passing Interface*), LAPACK (*Linear Algebra Package*) e BLAS (*Basic Linear Algebra Subprograms*). Muitas dessas bibliotecas e APIs são também de domínio público e, geralmente, não são compatíveis com SOs proprietários.

3.2. Nível de integração

O nível anterior, apesar de necessário, não é suficiente para afirmar que se tem uma grade computacional para PAD. Embora seja possível ter ganhos consideráveis na velocidade de processamento ao usar um *cluster* com os sistemas operacionais e bibliotecas citados no parágrafo anterior, ao se integrar diversos *clusters* em uma grade, ainda é possível ter aumentos de desempenho, em função do aumento da quantidade de nós de processamento.

Para integrar os diversos *clusters* em uma grade é preciso de uma camada intermediária que interligue os SOs das diferentes unidades de processamento, oferecendo ao usuário um ambiente de execução integrado. Esse ambiente deve prover as seguintes funcionalidades: informações para gerenciamento e monitoramento de recursos; possibilidade de movimentação de dados entre as diversas unidades de processamento; segurança e confidencialidade dos dados.

Com os requisitos descritos no parágrafo anterior, foi proposta a *Open Grid Services Architecture* (OGSA) que descreve uma arquitetura para uma grade computacional

baseada em serviços [Berry et al. 2006]. Uma implementação dessa arquitetura de referência é o Globus Toolkit (GT), um software aberto que permite a integração de diversos ambientes computacionais. O GT inclui módulos para segurança, infraestrutura de informações para gerenciamento de recursos e dados, comunicação, detecção de falhas e portabilidade.

Desta forma, o GT, atualmente na versão 4.0.x, pode prover a interligação necessária entre os vários ambientes (*clusters*), mesmo que eles possuam um conjunto diferente de softwares, em especial o SO, no nível básico.

Outro software importante no nível de integração em uma grade computacional para PAD é o sistema de filas. Geralmente os sistemas de PAD possuem mais do que um usuário para utilizar os recursos. Por outro lado, havendo muitos processos em execução "ao mesmo tempo", isso pode degradar o desempenho do sistema como um todo.

Assim, um sistema de filas é um software útil para gerenciar as tarefas (*jobs*) que serão submetidas para processamento. Essas tarefas podem envolver a execução de um ou mais programas (ou *scripts*) e, nesse caso, o sistema de filas disciplina o uso e o acesso aos recursos de maneira que não sobrecarregue o sistema computacional. Existem vários sistemas de filas, proprietários ou não. Entre os softwares livres que se destacam para essa função estão o Torque (baseado no *Portable Batch System*), LSF (*Load Sharing Facility*), o SGE (*Sun Grid Engine*) e Condor.

3.3. Nível de apresentação

O nível de apresentação abrange o portal e o meta-escalonador. O principal objetivo do portal, também chamado de *front-end*, é servir como um ponto único na *world wide web* (WWW) onde os usuários podem submeter suas tarefas para processamento nos diversos sítios e coletar os resultados. Com isso, os usuários vêem uma imagem única do sistema, sem perceber sua complexidade.

O meta-escalonador também é útil nessa tarefa, pois ele interage diretamente com os diversos sistemas de filas administrado-os e recolhendo várias informações úteis para a definição e implementação de políticas de uso.

Para o portal, os softwares de destaque para essa tarefa são o GridSphere e o GridPortlets. O primeiro é um *framework* para a construção de um portal; o segundo inclui no portal algumas funcionalidades típicas de uma grade computacional como, por exemplo, transferências de arquivos, integração com o sistema de filas para submissão de *jobs* e visualização de resultados, entre outras.

No caso do portal é importante que alguns conceitos sejam considerados na escolha do software para implementá-lo, e.g. *web services*, *servlets* e *portlets* [Shankar 2006]. Os *web services* são aplicações para a WWW modulares, distribuídas, dinâmicas, baseadas em padrões abertos (XML, SOAP, HTTP) e que são descritas, publicadas, localizadas e invocadas pela Internet. Por ser baseados nos padrões abertos, os *web services* fornecem grande interoperabilidade entre várias aplicações.

Os *servlets* são programas modulares em Java para desempenhar uma função específica e criar uma saída em HTML (*HyperText Markup Language*). Por fim, os *portlets* são *servlets* especiais que produzem fragmentos de HTML que podem ser agregados dinamicamente em uma página do portal. Há dois padrões de *portlets*: *Java Specification*

Request (JSR) 168, lançado em outubro de 2003 e o JSR 286, sem previsão para a versão final. A aderência aos padrões é necessária para garantir a portabilidade com aplicações para a grade computacional. A combinação desses três conceitos facilita a criação de portais dinâmicos, mais interativos e mais personalizados, tornando a interface com o sistema mais amigável.

Em relação ao meta-escalonador, esse é um software responsável por definir quando e onde os *jobs* serão executados. Para tanto, são usados três conjuntos de informações: informação de *job*, informação de recursos e políticas de escalonamento. O meta-escalonador é fundamental para a otimização dos recursos computacionais e balanceamento de carga entre as unidades de processamento.

É importante ressaltar a diferença entre um sistema de filas e um metaescalonador. O primeiro gerencia a fila de *jobs* e os nós de processamento. O segundo informa o que o sistema de filas deve fazer, quando e onde deve executar os *jobs*. Por causa desse controle, o meta-escalonador pode fornecer mais informações para gerenciamento e, se bem configurado, pode até tomar decisões mais "inteligentes" facilitando o balanceamento de carga entre as unidades de processamento. O Maui é uma meta-escalonador que serve como uma boa alternativa de código fonte aberto para desempenhar as funções citadas.

4. Arquitetura do sistema

Nesta seção é ilustrada uma arquitetura geral para uma grade computacional para PAD. Conforme se observa na Figura 1, o sistema possui três níveis: básico, na parte inferior; integração, na parte central; e apresentação na parte superior da figura, descritos a seguir.

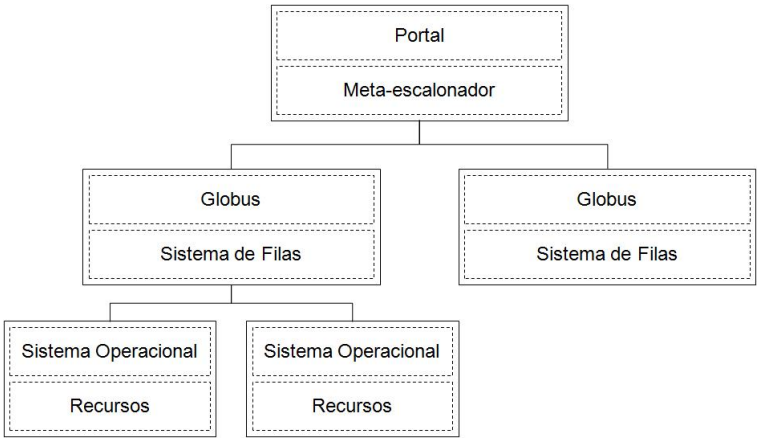

Figura 1. Proposta de arquitetura para grade computacional de alto desempenho.

O nível básico é composto pelo sistema operacional e recursos computacionais, instalados em uma única unidade (e. g. uma máquina ou um *cluster*). Essas unidades podem, juntas, compor um *cluster* ou conjunto de *clusters*. Sugere-se, nesse nível, um sistema operacional leve mas, ao mesmo tempo, robusto.

O nível de integração é composto pelo sistema de filas, que despacha tarefas de processamento para as unidades do nível inferior, e um conjunto de serviços e bibliotecas (encapsulados no Globus) para segurança, monitoramento de recursos, comunicação, gerenciamento de dados e detecção de falhas, entre outros. A sugestão é o sistema de filas

Torque, em razão da sua robustez, facilidade de configuração, código aberto e manutenção freqüente pela comunidade.

O nível superior, também chamado de *front-end*, contém o portal, que atua como uma interface para o usuário e o meta-escalonador responsável pelo despacho de tarefas aos recursos distribuídos. A tarefa do *front-end* é gerenciar e fornecer um ponto único de acesso às unidades dos níveis inferiores. Nesse nível, indica-se no portal a combinação GridSphere e GridPortlets, em razão da compatibilidade com o padrão JSR 286 e a facilidade de implementação de novas funcionalidades; e o Maui, como meta-escalonador, por causa da sua compatibilidade com o Torque.

5. Conclusão

Este artigo apresentou uma proposta de arquitetura para a construção de grades computacionais para processamento de alto desempenho totalmente baseada em softwares de código aberto. Pode-se perceber, portanto, que em razão das características desse tipo de aplicação, i. e. PAD, a utilização de software proprietário pode inviabilizar o projeto. A razão para tal afirmação é que a maioria dos softwares proprietários não permite a personalização de determinados elementos centrais da implementação, e. g. alteração dos algoritmos de escalonamento utilizados no núcleo do SO.

Por outro lado, a utilização dos softwares livres, mencionados ao longo deste texto em alguns projetos de grades computacionais para PAD em andamento, mostrou-se bem escalável, robusta e bastante adaptável às diversas situações e configurações. Ressalte-se que a arquitetura proposta neste artigo foi implementada no Centro Nacional de Processamento de Alto Desempenho em São Paulo (CENAPAD-SP), no âmbito do projeto GradePAD, usando os softwares Fedora, Globus, Torque, Gridsphere e Gridportlets. Portanto, é fácil comprovar que na área de PAD e em grades computacionais a escolha por softwares livres é, sem dúvida, a melhor escolha.

Agradecimentos

O apoio financeiro, suporte e os equipamentos gentilmente cedidos pelo Centro Nacional de Processamento de Alto Desempenho em São Paulo foram essenciais para a aplicação e teste dos conceitos reportados neste artigo. Agradece-se também o apoio financeiro complementar da Universidade São Francisco.

Referências

Berry, D., Djaoui, A., Grimshaw, A., Horn, B., Maciel, F., Siebenlist, F., Subramaniam, R., and J. Treadwell, H. (2006). *The Open Grid Services Architecture, Version 1.5*. The Open Grid Forum.

Foster, I. and Kesselman, C. (2003). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufman Publishers, 2 edition.

Jacob, B., Brown, M., Fukui, K., and Trivedi, N. (2005). *Introduction to Grid Computing*. IBM Redbooks.

Plaszczak, P. and Wellner, R. (2005). *Grid Computing: The Savvy Manager's Guide*. Morgan Kaufman Publisher.

Shankar, A. (2006). A general (gentle?) introduction to (grid) portals/gateways. Technical report, TeraGrid Gateways Team, Indiana University.