MAC5715 — Tópicos Avançados em POO
Profº Fábio Kon

Andrei Goldchleger
Márcio Rodrigo de Freitas Carneiro

27 de outubro de 2002

Grade (Grid)

O padrão Grade permite que recursos computacionais distribuídos por diversos domínios administrativos sejam compartilhados de forma transparente e eficiente.

Exemplo

Suponha que desejamos executar uma aplicação paralela que deve ser executada em nós distribuídos por uma rede local. Em muitos casos, esse tipo de aplicação demanda recursos muitas vezes superiores aos disponíveis, o que pode inviabilizar sua execução. Entretanto, é provável que existam recursos suficientes à execução em lugares geograficamente separados, ou mesmo dentro da própria instituição, considerando recursos presentes em outros domínios administrativos.

Para o uso de nós distribuídos, é necessária uma infra-estrutura que viabilize o compartilhamento de recursos dispersos e heterogêneos, de maneira a viabilizar a execução de tal aplicação. Uma solução seria a configuração de cada nó separadamente, entretanto esta solução não é reutilizável, nem tão pouco escalável, além de ser praticamente inviável em redes maiores ou sob diversos domínios administrativos. Queremos, portanto, um sistema que permita a interconexão de recursos dispersos e ofereça uma visão uniforme às aplicações.

Contexto

Sistemas distribuídos e heterogêneos que devem compartilhar recursos de hardware e software, de maneira a viabilizar a execução de aplicações que fazem uso intensivo de computação.

Problema

Apesar do avanço das tecnologias de computação tais como processadores, redes, e software, ainda existem problemas difíceis a serem resolvidos nas mais diferentes áreas de atuação, tais como sistemas para previsão do tempo, simulações mercadológicas, análise de prospecção de petróleo e algoritmos de otimização. A característica comum desses problemas é a necessidade de grandes quantidades de computação, muitas vezes por médios e longos períodos de tempo. Tais aplicações demandam uma quantidade de processamento que só pode ser atendida através do uso de equipamentos de alto desempenho, geralmente muito caros, ou aglomerados de máquinas comuns, mas dedicadas a tais tarefas. A dedicação exclusiva de recursos é muitas vezes contra-producente, uma vez que é grande a possibilidade de tais recursos estarem ociosos durante boa parte do tempo.

Paralelamente a tal questão, podemos abstrair necessidades não só de recursos como processador e memória, mas também outros, como armazenamento persistente, software e hardware específico, tais como placas especiais que desempenham funções dedicadas, e que possuem custo alto. Assim, é necessária a criação de uma infra-estrutura que permita o compartilhamento de recursos de forma eficiente, segura e barata.

Tal infra-estrutura deve contemplar recursos dispersos por diversos domínios administrativos, e possivelmente em diferentes localizações geográficas.

Forças

A infra-estrutura para a resolução dos problemas aqui já mencionados deve considerar as forças a seguir.

- Deve executar em diversas plataformas sobre os diversos sistemas operacionais pré-existentes, não exigindo assim nenhum tipo de substituição.
- Deve permitir o compartilhamento eficiente de recursos computacionais, tais como processador, memória, disco, outros tipos de hardware e software.
- Deve oferecer qualidade de serviço tanto aos usuários das aplicações quanto aos proprietários dos recursos. É necessário encontrar um balanço em que as necessidades de ambos sejam atendidas. Quando impossível, pode-se priorizar umas das partes, em geral os proprietários dos recursos.
- Deve prover suporte às aplicações paralelas.
- Deve oferecer autonomia para usuários e administradores, evitando impor políticas rígidas de utilização e oferecimento dos recursos.
- Deve ser escalável, podendo englobar de poucas máquinas até possivelmente milhões.
- Deve possuir baixo custo de instalação.

- Não deve causar sobrecarga perceptível ao usuário.
- Deve ser fácil de usar, permitindo que aplicações pré-existentes sejam facilmente adaptadas ao novo contexto.

Solução

Para solucionar o problema descrito, o padrão Grade pode ser aplicado. A Grade consiste em um *middleware* que integra recursos distribuídos, de modo a criar um ambiente de execução unificado para as aplicações. Esse middleware deve ser executado nos diversos nós participantes, de modo a permitir que seus recursos sejam integrados à Grade.

O usuário da Grade pode a utilizar para executar aplicações, opcionalmente especificando seus requisitos, sem a necessidade de localizar e configurar os recursos distribuídos. Do outro lado, os proprietários podem disponibilizar os recursos na Grade, podendo especificar a maneira como esses serão compartilhados. A Grade se encarrega de realizar a negociação entre o oferecimento e demanda de recursos, bem como da execução das aplicações no ambiente distribuído, cuidando de serviços como migração, comunicação e tolerância a falhas.

A vantagem dessa solução é a transferência de todas as complicações envolvidas no ambiente distribuído para uma infra-estrutura comum e unificada, o que permite a reutilização da mesma por diversas aplicações.

Estrutura

A arquitetura do padrão Grade é composta por serviço de informação, serviço de escalonamento, cliente de acesso, provedor de recursos e serviço de contabilização. A Grade é composta por uma hierarquia de aglomerados, cada um contendo de algumas dezenas a poucas centenas de máquinas, interconectadas por uma rede local. Esses aglomerados se conectam em estruturas hierárquicas, permitindo a eficaz integração de recursos distribuídos por diversos departamentos e localizações geográficas.

- O *serviço de informação* é responsável pela consolidação de diversas informações produzidas por outros serviços da Grade. Tais informações podem ser sobre características de nós provedores de recursos, como memória disponível, percentual de utilização do processador ou quantidade de disco livre para aplicações remotas. O serviço de informação é fundamental para que outros serviços tomem decisões eficientes para o bom funcionamento da Grade, uma vez que é indispensável possuir conhecimento sobre seu estado.

  Em geral, cada aglomerado deve ter seu serviço de informação. Os serviços de informação se organizam hierarquicamente, para que se possa ter informação sobre toda a Grade, disponível em qualquer ponto, e de maneira escalável.

- O *serviço de escalonamento* cuida do gerenciamento dos recursos disponíveis na Grade. Esse serviço recebe as requisições de execução de aplicações. Utilizando as informações coletadas pelo serviço de informação, define quais recursos serão utilizados por quais aplicações. O serviço de escalonamento tenta emparelhar requisições às ofertas da Grade, de modo a maximizar o número de aplicações atendidas. No caso de não ser possível atender uma requisição, pode-se oferecer recursos abaixo das necessidades, caso a aplicação aceite ser executada nessas condições. Em resumo, o serviço de escalonamento é um negociador entre os usuários e provedores de recursos, para que a utilização destes seja de forma organizada e eficiente.

  Devido a grande abrangência da Grade, é necessário que cada aglomerado tenha ao menos um serviço de escalonamento. Os serviços de aglomerados diferentes se comunicam, formando uma hierarquia de escalonadores. Isso permite balanceamento de carga e alternativas de execução caso as máquinas de um aglomerado estejam totalmente utilizadas, além de permitir o uso de recursos disponíveis somente em outro aglomerado, como por exemplo, hardware específico de alto custo.

- O *cliente de acesso* é executado em cada nó da Grade cujo usuário deseje submeter aplicações. Esse serviço é uma ferramenta de interação com a Grade: com ele o usuário pode especificar as necessidades de recursos da aplicação, verificar o estado das aplicações em execução na Grade e receber os resultados destas.

- O *provedor de recursos* é um serviço executado localmente nos nós que desejam compartilhar recursos com a Grade. A partir desse serviço, o proprietário pode especificar que tipo de recursos deseja compartilhar, e em que condições. Pode-se determinar, por exemplo, o horário desejado para compartilhar, como somente à noite, ou ainda determinar condições adicionais, como inatividade do teclado e mouse, ou queda na intensidade do uso do processador. Além disso, o proprietário pode definir a quantidade de cada recurso que deseja compartilhar, como quantidade de disco e memória, ou porcentagem de processamento.

- O *serviço de contabilização* mantém informações sobre a utilização dos diversos recursos da Grade. Tais informações servem para verificação do estado da Grade, e como consolidação do uso dos recursos, o que permite que se determine eventuais problemas como picos de utilização, gargalos na Grade e degradação na rede.

  Esse serviço também é hierárquico, com pelo menos um serviço de contabilização por aglomerado. Cada serviço recebe atualizações periódicas dos serviços de informação do mesmo aglomerado, consolidando tais informações de maneira sucinta, para que os diversos serviços de contabilização possam estabelecer perfis de utilização da Grade.

A Figura 1 mostra a relação entre cada elemento de um aglomerado que pertence à Grade. A Figura 2 nos dá uma idéia do esquema de implantação em um aglomerado. Já a figura 3 mostra uma possível hierarquia de aglomerados que compõem uma Grade.

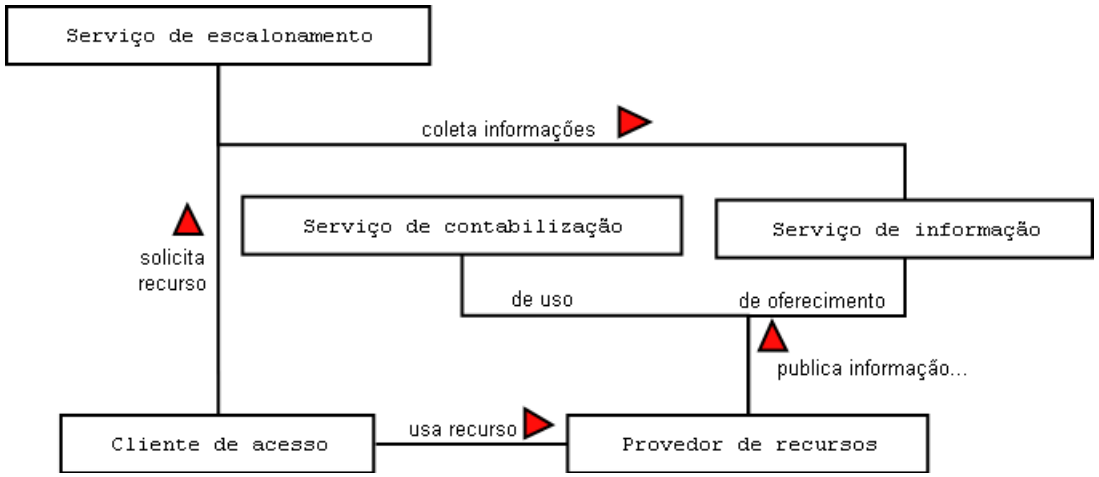

Figura 1: Relação entre os componentes do padrão Grade.

Dinâmica

Esta seção descreve alguns cenários relevantes do padrão Grade.

Cenário I Ilustra o comportamento quando um cliente de acesso solicita a execução de uma aplicação.

- O cliente de acesso envia uma requisição de execução para o serviço de escalonamento, possivelmente com definições de recursos necessários.
- O serviço de escalonamento solicita informações ao serviço de informação sobre os recursos disponíveis.
- O serviço de escalonamento determina se é possível a execução da aplicação com os recursos disponíveis.
- Caso seja possível, a aplicação é enviada para o nó (ou os nós, no caso de aplicações distribuídas), cujos recursos foram escalonados.
- Após o término da execução, os resultados são enviados para ao cliente de acesso.
- Caso não seja possível a execução, o cliente de acesso recebe uma mensagem de falha, ficando a critério do usuário quando e como tentar novamente sua submissão.

A figura 4 mostra o Cenário I aqui descrito.

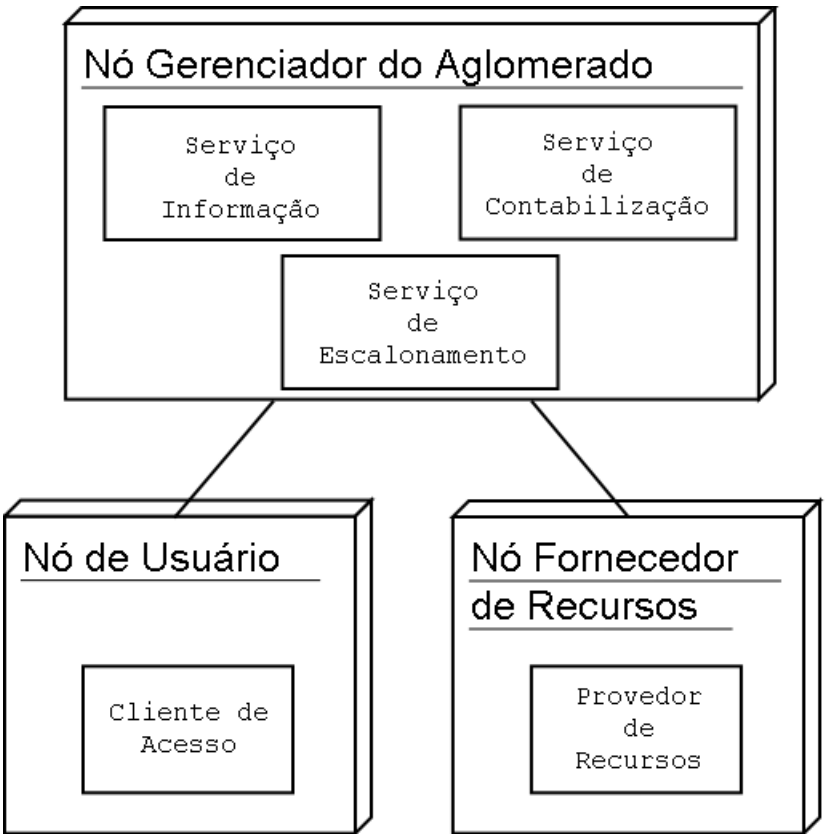

Figura 2: Diagrama de implantação de um aglomerado.

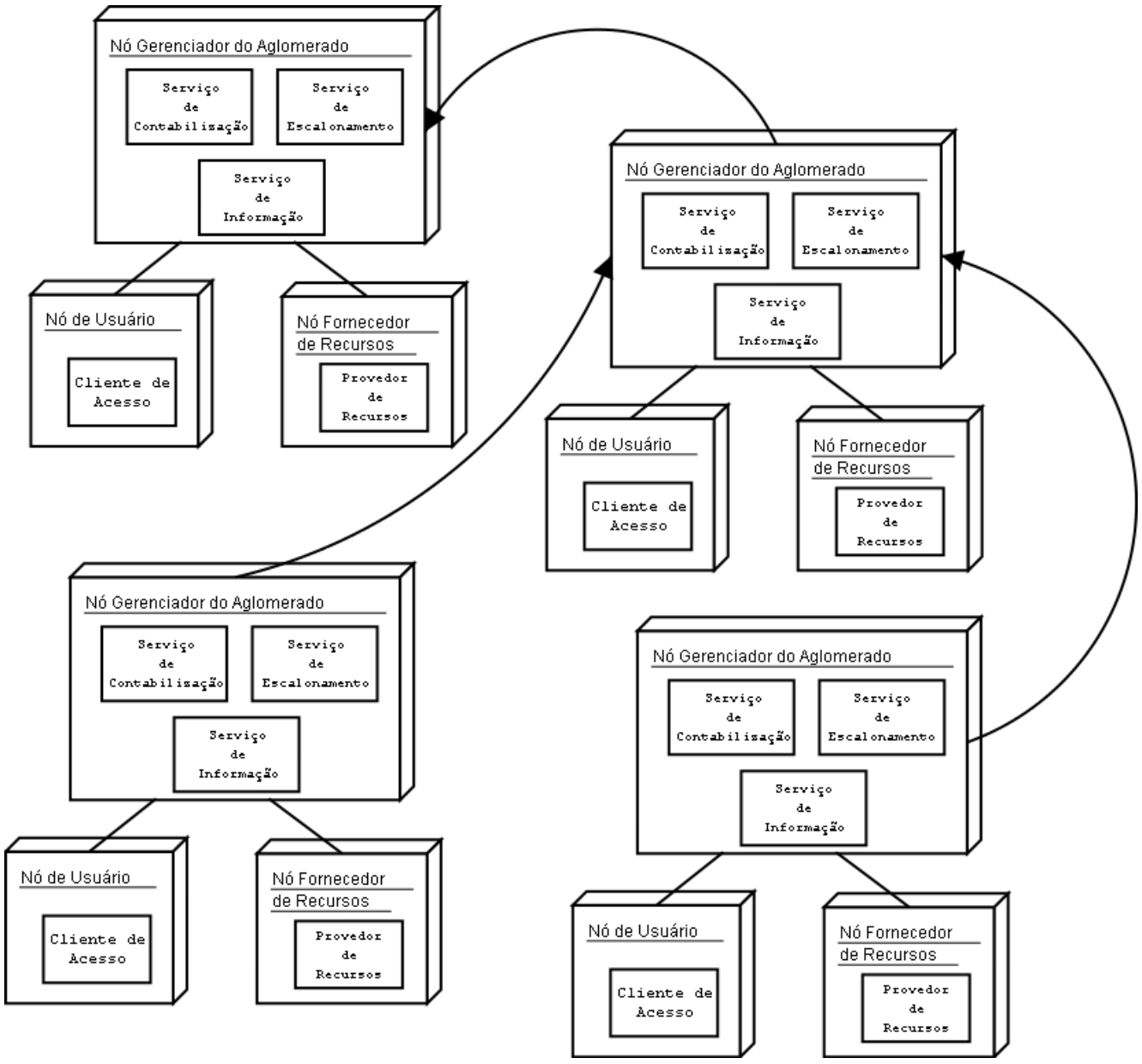

Figura 3: Hierarquia de aglomerados que compõem uma Grade.

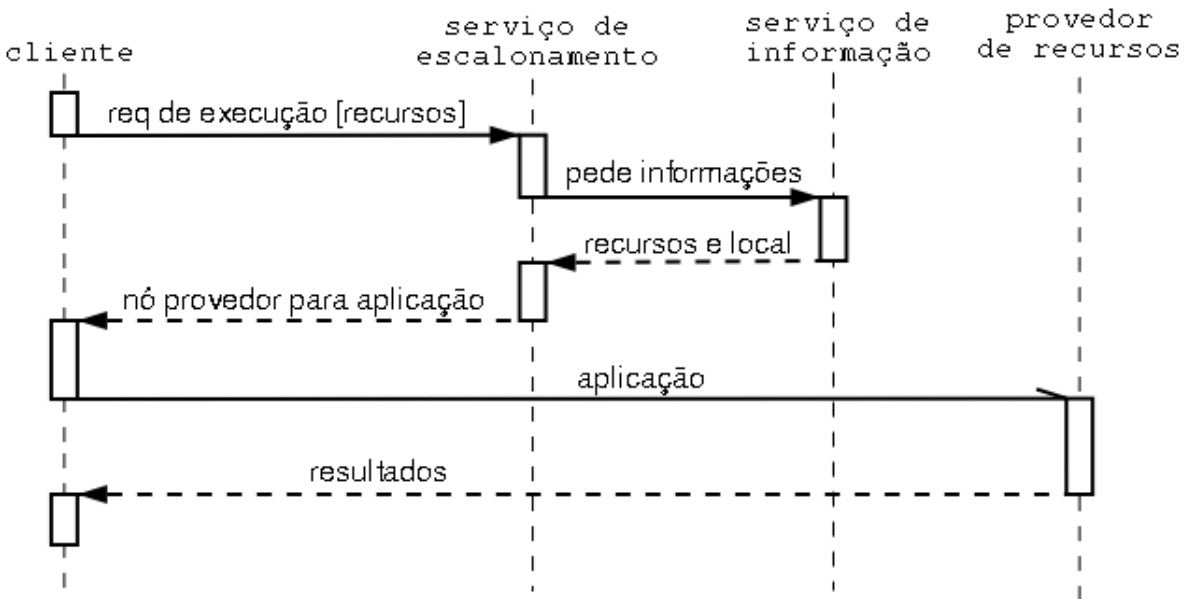

Figura 4: **Cenário I** Cliente de acesso enviando aplicações

Cenário II Ilustra o comportamento dos serviços de informação na troca de informações entre aglomerados.

- Os serviços de informação recebem atualizações periódicas dos provedores de recursos de seu aglomerado.
- Periodicamente, os serviços de informação enviam atualizações para os serviços em nível superior na hierarquia.

Implementação

Apresentamos algumas considerações sobre a implementação do padrão Grade.

- É necessário decidir sobre o formato das aplicações submetidas à Grade. Uma possibilidade é submeter código binário pronto para execução. A vantagem é que não é necessário compilar o código nos recursos remotos. Entretanto, seria necessário enviar mais de um formato, já que o sistema conta com a possibilidade de diversas plataformas e sistemas operacionais. Outra possibilidade é o envio de código fonte, para a compilação na máquina provedora do recurso. A vantagem nesse caso é a simplificação do transporte da aplicação, já que é necessário apenas um pacote para todas plataformas e sistemas operacionais. Contudo, a necessidade

de compilação na máquina provedora dos recursos apresenta como desvantagem o fato de se gastar mais tempo e exigir um item complicador na configuração do sistema. Esta questão não se aplica a código interpretado ou compilado para *bytecode* e executado sobre máquina virtual. Entretanto, grande parte das aplicações paralelas é escrita em linguagens que devem ser compiladas para código de máquina, daí a importância de tal questão.

- Além das atualizações periódicas de disponibilidade de recursos, os provedores de recursos podem conter mecanismos mais elaborados para a determinação de padrões de uso. Esses mecanismos coletariam informações de uso ao longo de períodos maiores de tempo, permitindo elaboração de heurísticas que se aproximariam do verdadeiro padrão de uso das máquinas provedoras de recursos. Tais informações poderiam auxiliar o serviço de escalonamento. Por exemplo, se o padrão de uso de uma máquina indica que ela fica ociosa por curtos períodos de tempo, não é recomendável que uma aplicação que necessite de execuções por longos períodos de tempo seja alocada para esta máquina.

- Os intervalos de tempo para as trocas de dados entre os diversos serviços de informação devem ser escolhidos com critério. Se forem muito curtos, é provável que haja degradação do desempenho da rede, por uma sobrecarga de informações de atualização. Se forem longos, o estado representado pelas informações pode ser muito diferente do estado real da Grade. Tal questão é fundamental para o bom desempenho da Grade, e deve ser determinada através de testes criteriosos.

- Outra decisão relevante é determinar em que situações repassamos uma requisição de execução para um escalonador de outro aglomerado. Quando temos um recurso requisitado cuja existência se restringe a um aglomerado remoto, esta decisão se torna óbvia. Também repassamos a requisição quando os recursos do aglomerado se esgotarem. Entretanto, pode-se decidir entre repassar ou não uma requisição para um escalonador remoto com o objetivo de implementar balanceamento de carga.

- Outra questão de implementação que oferece alternativas é a maneira como a aplicação é transferida para o provedor de recursos. Uma alternativa é o escalonador devolver uma referência para o provedor de recursos ao cliente de acesso, que nesse caso envia a aplicação direto ao provedor. Outra possibilidade é utilizar um repositório de aplicações, de onde o provedor pode baixar a aplicação a ser executada.

- O usuário necessitará de uma interface que permita o uso dos serviços da Grade, tais como comunicação entre diversos executáveis de uma aplicação paralela e reserva de recursos. Tal interface deverá estar disponível para todas as linguagens que virão a ser comportadas pela Grade.

- Como última observação, temos as questões de *checkpointing* e migração. Como a Grade é composta por recursos que podem tornar-se indisponíveis a qualquer momento, é necessário

algum mecanismo que permita a continuidade da execução mesmo em tal condição. *Checkpointing* consiste em salvar periodicamente o estado da aplicação, permitindo que a execução continue posteriormente no mesmo recurso, ou imediatamente, caso seja possível migrar a aplicação para outra máquina. Porém, *checkpointing* pode ser difícil de ser implementado, especialmente no caso de aplicações paralelas. Além disso, o custo de *checkpointing* pode inviabilizar o seu uso. Uma solução é tentar utilizar padrões de acesso (baseados nos dados fornecidos pelos provedores de recurso), que minimizem ou eliminem a necessidade de migrações.

Exemplo Resolvido

No nosso exemplo de uma aplicação paralela que deve ser executada em vários nós de uma rede, podemos utilizar um sistema que implementa o padrão Grade. A aplicação deve ser implementada numa linguagem disponível no sistema, e deve utilizar a interface disponibilizada para utilizar os serviços da Grade. Podemos também criar apenas um invólucro [GHJV95] para a aplicação pré-existente, através do uso da interface.

A aplicação se beneficia do escalonamento, comunicação, e migração (se disponível) transparentes. Não há necessidade de especificar recursos e máquinas para execução. Ao utilizar o cliente de acesso para a submissão da aplicação, todos os problemas de alocação e execução remotas são tratados pelos serviços da Grade.

Usos Conhecidos

Globus [FK97] é um sistema de computação em grade que provê um conjunto de serviços necessários para desenvolvedores escreverem aplicações aptas a executar em grades Globus. Uma característica importante do sistema é o paradigma de caixa de ferramentas (*toolkit*), que permite que aplicações usufruam de serviços da grade de maneira incremental. Assim, pode-se produzir uma série de versões da mesma aplicação ao longo do tempo, cada uma utilizando mais funcionalidades da grade que a anterior. No sistema encontramos exemplos dos serviços descritos nesse padrão: assim, o serviço de informação de Globus é o **MDS** (*Metacomputing Directory Service*) [CFFK01] e um dos serviços de escalonamento é o **GRAM** (*Globus Resource Alocation Manager*) [CFK$^+$98].

Legion [GWF$^+$94] é uma infra-estrutura de grade que, assim como Globus, permite que recursos computacionais distribuídos sejam utilizados de forma transparente, criando um meta-computador. Uma característica importante de Legion é sua arquitetura baseada em um modelo de *Objetos Núcleo* [LG96]. Esta arquitetura permite que serviços necessários às aplicações sejam construídos sobre um arcabouço comum, facilitando assim o desenvolvimento de novos serviços e sua integração às aplicações.

Condor [LLM88] possibilita a integração de vários recursos computacionais para compor um aglomerado, que permite a execução de aplicações. Condor enfatiza o uso de recursos que perma-

necem ociosos boa parte do tempo, como estações de trabalho, para realizar computação útil. O proprietário de um recurso pode especificar as condições em que permite o uso do mesmo. Aplicações seqüenciais podem beneficiar-se de *checkpointing* para permitir continuidade de execução mesmo quando a migração é necessária. Um exemplo de serviço aqui mencionado e que se encontra em desenvolvimento em Condor é o serviço de contabilização, denominado **Hawkeye** [Haw02].

InteGrade [GKL+02] é um sistema de computação em grade em início de desenvolvimento no IME-USP, que objetiva construir uma infra-estrutura de middleware que permita a construção de aplicações que se beneficiem do uso da grade. O diferencial de InteGrade em relação a outros projetos é a ênfase na re-utilização de equipamentos comuns compartilhados, sem exigir a compra de hardware dedicado e caro; a utilização de tecnologias modernas de objetos distribuídos, como CORBA; e a baixa sobrecarga que o sistema causará às máquinas que ofereçam recursos à grade.

Conseqüências

O padrão Grade traz algumas vantagens importantes:

- *Reusabilidade.* Uma vez implementado, o padrão Grade servirá para várias aplicações distribuídas. O sistema é basicamente um middleware, que pode fornecer uma camada de abstração para diversas aplicações diferentes em diversos ambientes e condições.

- *Encapsulamento da heterogeneidade.* A Grade oculta os detalhes específicos de cada plataforma e sistema operacional. Com isso, as aplicações podem ser desenvolvidas mais facilmente, sem a necessidade de se preocupar com detalhes de comunicação e reserva de recursos.

- *Facilidade de execução.* O cliente de acesso, junto com outros serviços da Grade, encapsula os detalhes de alocação da aplicação entre as diversas máquinas. Isso facilita a tarefa do usuário, que não precisa se preocupar em determinar as máquinas em que sua aplicação deve ser executada.

- *Transparência na transposição de domínios administrativos.* O uso da Grade elimina as dificuldades de integração de recursos dispersos em diferentes domínios. Cada administrador de domínio pode especificar com quais outros domínios e em quais circunstâncias deseja compartilhar recursos. Os serviços da Grade se encarregam da utilização de recursos de outros domínios quando necessário, respeitando as restrições indicadas pelos administradores.

- *Utilização eficiente de recursos.* O desperdício de recursos é consideravelmente eliminado ao se utilizar uma Grade. Quando necessário, o recurso disponível é utilizado diretamente pelo usuário, sem degradação, ao mesmo tempo que os recursos ociosos (máquinas sem usuários utilizando, ou capacidade excedente de processamento ou armazenamento) são aproveitados da melhor maneira possível pelas aplicações da Grade.

- *Compartilhamento de recursos dedicados.* Recursos caros e especialistas quase sempre são escassos e dispersos num país ou instituição. A Grade pode possibilitar o aproveitamento desses recursos por qualquer participante, de maneira racional e organizada.

Algumas desvantagens do padrão:

- *Complexidade.* A implementação do padrão Grade implica em vários problemas que não possuem solução consolidada. Assim, esta tarefa necessita de muito tempo e trabalho despendido, além de alta capacidade dos integrantes da equipe para tal.

- *Alto custo.* O custo da implementação da Grade é alto (principalmente pelo fato anterior), e exige tempo e uso efetivo em diversas aplicações e ambientes, para justificar o investimento em tal sistema.

- *Necessidade de alterar aplicações.* Para a utilização da Grade, normalmente é necessário reescrever partes da aplicação, o que só é possível quando se detém o código fonte das mesmas, fato que raramente ocorre com aplicações comerciais. Mesmo que se possua o código de uma aplicação, a re-escrita e adaptação para a Grade pode consumir consideráveis quantidades de tempo e dinheiro.

Veja Também

O padrão Broker [BMR+96] possui similaridade com o padrão Grade, pois assim como a Grade, objetiva encapsular diversos detalhes referentes a sistemas distribuídos e simplificar o desenvolvimento de aplicações. Porém, ao contrário da Grade, o Broker é recomendado para aplicações do cotidiano, como sistemas de negócios baseados em arquitetura cliente-servidor, por exemplo. Entretanto, não prevê a necessidade de diversas características presentes na Grade, como os serviços de Escalonamento e Informação. Como o Broker encapsula detalhes de comunicação com entidades remotas, ele pode ser utilizado como substrato para a implementação do padrão Grade. **InteGrade**, por exemplo, pretende utilizar CORBA, uma implementação do padrão Broker, como base de sua implementação.

O padrão Adapter [GHJV95], também conhecido como Wrapper, permite que aplicações pré-existentes sejam envolvidas em um invólucro de software, que permite que tais aplicações se beneficiem das funcionalidades da Grade. O Wrapper facilita a integração entre as aplicações pré-existentes e a Grade, pois minimiza os pontos de mudança e os concentra em uma interface bem definida.

Referências

[BMR+96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern - Oriented System Architecture: a System of Patterns*, chapter 2.3. John Wiley & Sons, 1996.

[CFFK01] Karl Czajkowski, Steven Fitzgerald, Ian Foster, and Carl Kesselman. Grid Information Services for Distributed Resource Sharing. In *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*. IEEE Press, August 2001.

[CFK+98] Karl Czajkowski, Ian Foster, Nick Karonis, Carl Kesselman, Stuart Martin, Warren Smith, and Steven Tuecke. A Resource Management Architecture for Metacomputing Systems. In *Proceedings of IPPS/SPDP98 Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82, 1998.

[FK97] Ian Foster and Carl Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputing Applications*, 2(11):115–128, 1997.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*, chapter 4, pages 139–150. Addison-Wesley, 1995.

[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. Relatório Técnico (Trabalho em Andamento), disponível em: `http://www.ime.usp.br/~andgold/research/rt_integrade.pdf`, 2002.

[GWF+94] Andrew S. Grimshaw, Willian A. Wulf, James C. French, Alfred C. Weaver, and Paul F. Reynolds Jr. A Synopsis of the Legion Project. Technical report, University of Virginia Computer Science Department, june 1994.

[Haw02] Hawkeye. Sítio do projeto Hawkeye, 2002. `http://www.cs.wisc.edu/condor/hawkeye`.

[LG96] Michael J. Lewis and Andrew Grimshaw. The Core Legion Object Model. In *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing*. IEEE Computer Society Press, August 1996.

[LLM88] Michael Litzkow, Miron Livny, and Matt Mutka. Condor - A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, pages 104–111, June 1988.