MAG, um *middleware* de grade baseado em agentes: estado atual e perspectivas futuras

Francisco José da Silva e Silva, Rafael Fernandes Lopes, Bysmarck Barros de Sousa, Antônio Eduardo Viana, Stanley Araújo de Sousa

[1]Universidade Federal do Maranhão (UFMA)
Av. dos Portugueses s/n, Campus Universitário do Bacanga
São Luís – MA – Brasil

{fssilva,rafaelf}@deinf.ufma.br, {bysmarck,eduardo,stanley}@lsd.ufma.br

Abstract. *MAG (*Mobile Agents for Grid Computing Environments*) explores the mobile agent paradigm as a way to overcome the construction challenges of computational grids. MAG executes Grid applications by dynamically loading the application code into mobile agents. The MAG agent can be dynamically re-allocated to Grid nodes through a transparent migration mechanism, as a way to provide load balancing and support for non-dedicated nodes. MAG includes mechanisms for fault tolerance, pervasive grid, and data grid. We make extensive use of the agent paradigm to design and implement MAG components, forming a multi agent infrastructure for computational Grids.*

Resumo. *O MAG (*Mobile Agents for Grid Computing Environments*) explora a tecnologia de agentes móveis como uma forma de superar os desafios de construção de grades de computadores. O MAG executa aplicações carregando dinamicamente seus códigos nos agentes móveis. O agente do MAG pode ser realocado dinamicamente entre nós da grade através de um mecanismo de migração transparente chamado MAG/Brakes, como uma forma de prover suporte a nós não dedicados. O MAG inclui mecanismos de tolerância a falhas de aplicações, de grade pervasiva e grade de dados. O paradigma de agentes foi extensivamente utilizado para projetar e implementar os componentes do MAG, formando uma infraestrutura multiagente para grades computacionais.*

1. Introdução

A computação em grade permite a integração e o compartilhamento de computadores e recursos computacionais, como software, dados e periféricos, em redes corporativas e entre estas redes, estimulando a cooperação entre usuários e organizações, criando ambientes dinâmicos e multi-institucionais, fornecendo e utilizando recursos de forma a alcançar objetivos comuns e individuais [Foster et al. 2001]. Através da infraestrutura da grade é possível executar um grande número de aplicações como: aplicações de supercomputação distribuída, alto rendimento, aplicações sob-demanda, aplicações que fazem uso intensivo de dados e aplicações colaborativas

Entretanto, a construção de um *middleware* de grade é uma tarefa complexa. Os desenvolvedores deste tipo de *middleware* devem superar muitos desafios de projeto e implementação, tais como: gerenciamento e alocação eficiente de recursos distribuídos;

escalonamento dinâmico de tarefas de acordo com a disponibilidade de recursos; mobilidade transparente de código, como uma forma de promover balanceamento de carga e suporte a nós não dedicados; mecanismos de tolerância a falhas, uma vez que os nós da grade não constituem um ambiente controlado; suporte a alta escalabilidade e grande heterogeneidade de componentes de hardware e software; e mecanismos eficientes para comunicação colaborativa entre os nós da grade.

A tecnologia de agentes móveis exibe grande adequação para o desenvolvimento de infraestruturas de grade, por causa de algumas de suas características, como:

- *Cooperação:* agentes têm a habilidade de interagir e cooperar com outros agentes. Isto pode ser explorado para o desenvolvimento de mecanismos de comunicação complexos entre os nós da grade;
- *Autonomia:* agentes são entidades autônomas, significando que suas execuções ocorrem sem nenhuma, ou com muito pouca intervenção dos clientes que os iniciaram. Este é um modelo adequado para a submissão e a execução de aplicações na grade;
- *Heterogeneidade:* muitas plataformas de agentes móveis podem ser executadas em ambientes heterogêneos, uma característica importante para um melhor uso dos recursos computacionais entre ambientes multi-institucionais;
- *Reatividade:* agentes podem reagir a eventos externos, como variações na disponibilidade de recursos;
- *Mobilidade:* agentes móveis podem migrar de um nó para outro, movendo parte da computação sendo executada de forma a prover balanceamento de carga entre os nós da grade e suporte a nós não dedicados.

O MAG[1] (*Mobile Agents Technology for Grid Computing Environments*) [Lopes et al. 2005, Lopes 2006] é um projeto de *middleware* de grade desenvolvido por alunos e professores do Departamento de Informática da Universidade Federal do Maranhão (DEINF/UFMA), cujo principal objetivo é disponibilizar uma infraestrutura de software livre baseada na tecnologia de agentes de software que permita a resolução de problemas computacionalmente intensivos em grades de computadores.

O MAG utiliza o *middleware* Integrade [Goldchleger et al. 2004] como fundação para sua implementação, evitando assim, a duplicação de esforços no desenvolvimento de uma série de componentes. O MAG adiciona ao Integrade um novo mecanismo de execução de aplicações, através do uso da tecnologia de agentes móveis, para permitir a execução de aplicações Java, não suportadas nativamente pelo Integrade. Além desta funcionalidade, o MAG possui, em seu estado atual de implementação, mecanismos para prover tolerância a falhas de suas aplicações, migração transparente (de forma a prover suporte a nós não dedicados), além de permitir o acesso de clientes móveis e nômades à grade (através de dispositivos de computação portátil e da web).

Este artigo está organizado como a seguir. A seção 2 descreve a arquitetura do MAG. A seção 3 descreve a infra-estrutura para o acesso de clientes móveis e nômades à grade. A seção 4 apresenta a arquitetura dos serviços de dados e metadados desenvolvidos no contexto do MAG. Por fim, na seção 5 são apresentadas as conclusões obtidas a partir do trabalho realizado e descritos seus próximos passos.

[1]Disponí vel em: http://www.lsd.ufma.br/mag

2. Arquitetura Geral do MAG

A arquitetura do MAG está organizada na forma de uma pilha de camadas, conforme pode ser visto na Figura 1. Atualmente no MAG é possível executar duas classes distintas de aplicações: regulares e paramétricas. As aplicações regulares são aplicações seqüenciais compostas por um único binário que executa em uma só máquina. Já as aplicações paramétricas são aquelas que executam múltiplas cópias do mesmo binário em máquinas distintas e com entradas diferentes. Nesta classe de aplicações (também chamada de BoT ou *Bag-of-Tasks*) as tarefas são divididas em sub-tarefas menores que executam de forma independente, sem haver comunicação entre elas. Várias aplicações enquadram-se nesta categoria, como as de processamento de imagens, simulação e mineração de dados. Aplicações escritas na linguagem Java são executadas pelo MAG, enquanto que aplicações nativas do sistema operacional são executadas pelo Integrade.

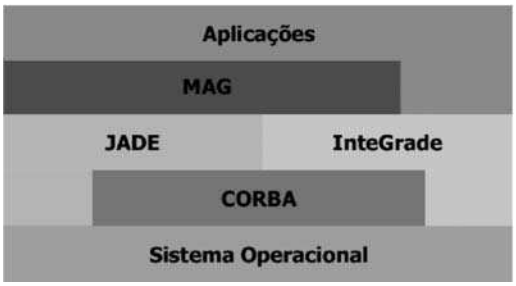

Figura 1. Arquitetura em camadas do *middleware* MAG

A camada JADE (*Java Agent Development Framework*) representa o arcabouço utilizado para a construção de sistemas multiagentes adotado no MAG. Este arcabouço provê ao MAG várias funcionalidades como facilidades de comunicação, controle do ciclo de vida e monitoração da execução dos agentes móveis. O JADE é uma plataforma portável (pois foi desenvolvida com tecnologia Java) e aderente à especificação FIPA[2].

As camadas superiores utilizam a tecnologia de objetos distribuídos CORBA para promover a comunicação entre seus componentes. Além disso, esta tecnologia fornece diversos serviços que podem ser utilizados pela infraestrutura do MAG.

2.1. Integrade

Integrade é um *middleware* de grade cujo desenvolvimento envolve um conjunto de universidades brasileiras (USP, Puc-Rio, UFMS, UFG e UFMA), coordenadas pelo Instituto de Matemática e Estatística da Universidade de São Paulo (IME/USP). O Integrade é estruturado em aglomerados organizados de uma forma hierárquica. Um aglomerado é definido como uma unidade autônoma dentro da grade, uma vez que o mesmo contém todos os componentes necessários para funcionar independentemente. Os componentes centrais da arquitetura do Integrade são:

- *Application Submission and Control Tool* (ASCT): interface gráfica que permite aos usuários da grade, submeter aplicações para serem executadas na grade;

[2] *Foundation for Intelligent Physical Agents* – organização sem fi ns lucrativos que objetiva a produção de padrões de interoperabilidade entre agentes de software heterogêneos. Disponí vel em http://www.fi pa.org/

- *Application Repository* (AR): armazena as aplicações a serem executadas na grade;
- *Local Resource Manager* (LRM): componente que executa em cada nó do aglomerado, coletando informações sobre o estado dos recursos dos nós da grade como memória CPU, disco e utilização da rede. Ele é também responsável por instanciar aplicações escalonadas ao nó;
- *Global Resource Manager* (GRM): gerencia os recursos do aglomerado, recebendo notificações de utilização de recursos a partir dos LRMs e executando o escalonador que aloca tarefas aos nós, de acordo com a disponibilidade de seus recursos.

Para permitir a execução de aplicações Java através da tecnologia de agentes de software, o MAG adiciona dois novos componentes à arquitetura original do Integrade: o *MagAgent* e o *AgentHandler*.

O *MagAgent* é um agente móvel responsável por executar aplicações na grade. Em resposta a cada solicitação de execução de aplicação, um novo *MagAgent* é criado. Ele atua como um *wrapper* da aplicação, sendo responsável por: (a) baixar o *bytecode* das aplicações a partir do repositório de aplicações; (b) dinamicamente instanciar e executar a aplicação utilizando mecanismos de reflexão computacional fornecidos pela plataforma Java; (c) capturar qualquer exceção não capturada pela aplicação; e (d) capturar e salvar periodicamente o estado de execução da aplicação (*checkpoint*), de forma a permitir sua posterior recuperação caso uma falha ocorra.

Um *AgentHandler* executa em cada nó da grade que fornece recursos para a execução de aplicações. Ele mantém o *container* de agentes executando no nó e também é responsável por criar *MagAgents* ao receber requisições de execução. Este componente torna o paradigma de agentes de software transparente ao Integrade.

Além de permitir a execução de aplicações Java através da tecnologia de agentes de software, o MAG disponibiliza um mecanismo para migração transparente de aplicações e um mecanismo de tolerância a falhas de aplicações executadas pelo MAG.

O impacto dos componentes de gerenciamento de recursos local e global (LRM e GRM) e do *AgentHandler*, utilizando como métrica o uso relativo de CPU, foi medido através de experimentos realizados sobre estes componentes. Os resultados obtidos comprovam a baixa sobrecarga de CPU causada pela presença dos componentes do MAG em nós da grade. Como resultado deste experimento, foram obtidas as médias de 0.086%, 0.040% e 0.012% de consumo de CPU causado pelos componentes *AgentHandler*, *LRM* e *GRM*, respectivamente. Maiores detalhes a respeito deste e de outros experimentos realizados na plataforma podem ser encontrados em [Lopes et al. 2005] e [Lopes 2006].

2.2. Mecanismo de migração transparente de aplicações

O MAG utiliza o poder computacional ocioso de máquinas existentes em redes computacionais para executar aplicações. Isto implica dizer que as máquinas que compõem a grade não são, necessariamente, dedicadas à grade, podendo pertencer a outros usuários. Dessa forma, é fundamental que o MAG disponibilize um mecanismo através do qual os usuários possam solicitar o uso exclusivo de seus recursos. Assim, quando um usuário solicita o uso exclusivo de sua máquina, todas as computações que

nela executam devem ser migradas para um outro nó da grade de forma transparente, sem perda de nenhuma computação já realizada.

Uma vez que o MAG executa aplicações Java utilizando agentes móveis, passa a ser necessário que exista um mecanismo que permita a captura e a recuperação do estado de execução de *threads* Java. Entretanto, a plataforma Java não provê mecanismos suficientes para capturar o estado de execução de computações. Seu mecanismo de serialização somente permite a salva do código e do valor dos atributos de objetos. Além disso, as classes Java não podem acessar informações nativas e internas da Máquina Virtual Java (por exemplo, o contador de instrução e a pilha de chamadas), necessárias para a captura do estado completo de execução de *threads* Java.

Durante os últimos anos, algumas técnicas foram desenvolvidas no intuito de permitir a captura do estado de execução de *threads* Java. Estas técnicas podem ser classificadas em quatro abordagens básicas [Illmann et al. 2000]: modificação da máquina virtual, instrumentação do código-fonte das aplicações, instrumentação do *bytecode* das aplicações e a modificação da *Java Platform Debugger Architecture*[3]. Cada uma destas técnicas traz vantagens e desvantagens, sendo que a abordagem de instrumentação de *bytecode* mostrou-se a mais adequada para uso em ambientes de grades computacionais.

O mecanismo de captura e recuperação do estado de execução de *threads* Java do MAG é baseado em uma versão modificada do arcabouço Brakes [Truyen et al. 2000], desenvolvido no Katholieke Universiteit Leuven, Bélgica, pelo grupo de pesquisa em sistemas distribuídos e redes de computadores (DistriNet). Esta versão modificada do Brakes foi chamada MAG/Brakes [Lopes and Silva 2005a, Lopes and Silva 2005b], e fornece à versão original do Brakes algumas melhorias, como a redução da sobrecarga imposta pela inserção de *bytecode* do código da aplicação, a migração iniciada por uma entidade externa (i.e. o agente pode solicitar a captura do estado de execução da aplicação), além de tornar mais flexível o uso do MAG/Brakes por seus usuários. Um comparativo entre o arcabouço MAG/Brakes e outros arcabouços de migração transparente de código de *threads* Java, bem como testes de avaliações de desempenho realizadas sobre o mesmo podem ser encontradas em [Lopes and Silva 2005a], [Lopes and Silva 2005b] e [Lopes 2006].

2.3. Mecanismo de tolerância a falhas de aplicações

Tolerância a falhas é uma característica essencial em ambientes de grade. Uma vez que a grade atua como um sistema massivamente paralelo, a perda de tempo computacional deve ser evitada. De fato, a probabilidade de ocorrerem erros é grande, uma vez que, na grade, muitas aplicações executarão longas tarefas que podem requerer vários dias de computação.

A infraestrutura de tolerância a falhas do MAG [Lopes and Silva 2006] contempla atualmente falhas do tipo *fail-stop* (i.e. colapso de nós e processos). Para tanto, uma sociedade de agentes foi desenvolvida para prover tolerância a falhas às aplicações que executam na grade, sendo responsáveis por detectar e tratar estas falhas de maneira autônoma. Futuramente pretende-se dar suporte a outras categorias de falhas, como por exemplo, falhas na rede.

[3]A *Java Platform Debugger Architecture* (JPDA) é parte da especificação padrão da máquina virtual. Através da JPDA as informações de execução das aplicações podem ser acessadas em modo de depuração. Isto pode ser explorado para prover migração transparente.

A recuperação de aplicações no MAG é baseada na abordagem de recuperação por retorno (i.e. *checkpointing*), onde o estado de execução das aplicações é salvo periodicamente em um armazém estável de dados. Assim, se uma falha vier a ocorrer, as computações podem ser recuperadas a partir de seu último salvo no armazém estável. Além disso, está sendo atualmente implementado no MAG o suporte a uma outra importante técnica de tratamento de falhas de aplicações: a replicação de processos. Nesta técnica, várias cópias da mesma aplicação (e com o mesmo conjunto de dados de entrada) são submetidas simultaneamente para execução na grade. Assim, a falha de uma réplica da aplicação não acarreta perda de tempo computacional já realizado pela aplicação, uma vez que as outras réplicas da mesma aplicação continuarão a executar sem problemas.

Com a implementação da técnica de replicação de processos será possível fornecer ao usuário da grade um esquema flexível para o tratamento de falhas de aplicações. Esta flexibilidade deriva da combinação das duas técnicas fornecidas pelo MAG, resultando em quatro diferentes abordagens para o tratamento de falhas: reinício da aplicação (sem *checkpointing* ou replicação), *checkpointing* sem replicação, replicação sem *checkpointing* e replicação com *checkpointing*. Em um primeiro momento, o uso de cada uma destas técnicas deverá ser configurado pelo usuário através da interface de submissão de aplicações, enquanto que posteriormente pretende-se tornar a grade adaptável, fazendo com que cada *MagAgent* decida qual a melhor estratégia a utilizar para sua aplicação. Esta decisão pode ser amparada por fatores relativos à aplicação e ao ambiente da grade.

O mecanismo de tolerância a falhas do MAG é composto por três agentes de software: o *StableStorage*, o *AgentRecover* e o *ExecutionManagementAgent* (EMA). O *StableStorage* representa o armazém estável que armazena o *checkpoint* de todas as aplicações em execução na grade. Sua implementação atual é centralizada e é executada no nó gerenciador do aglomerado. O *AgentRecover* é responsável por iniciar o processo de recuperação de aplicações, caso uma falha de colapso de nó venha a ocorrer. Este componente é criado sob demanda pelo GRM (*Global Resource Manager*, veja a Seção 2.1), que é responsável por detectar falhas de colapso de nós. O EMA fornece três principais funcionalidades para o mecanismo de tolerância a falhas do MAG: (1) ele fornece informações a respeito das execuções de aplicação (como os argumentos de linha de comando passados à aplicação e o seu estado atual de execução), utilizadas pelo processo de recuperação para restaurar a aplicação como os mesmos parâmetros que tinha antes da falha; (2) ele mapeia cada execução de aplicação com o nó em que executa, podendo assim descobrir que aplicações executavam em um dado nó que falhou; e (3) os dados que o mesmo armazena são utilizados na geração de estatísticas que amparam decisões relativas às técnicas de tratamento de falhas empregadas.

O custo computacional introduzido pelo mecanismo de tolerância a falhas no ambiente de execução do MAG foi avaliado por meio de testes realizados com uma aplicação denominada GLCM (*Gray Level Co-occurrence Matrix*) [Haralick et al. 1973], um conhecido método de análise de texturas. Os resultados obtidos através dos experimentos mostram que houve um aumento em torno de 8.93% no tempo total de execução da aplicação GLCM em relação à sua execução normal. Este aumento deve-se à captura e salva periódica do *checkpoint* da mesma, dos quais 2.37% devem-se ao mecanismo de captura do estado de execução (o arcabouço MAG/Brakes foi utilizado para este fim), enquanto que os outros 6.56% devem-se à transferência de dados entre o *MagAgent* e

o *StableStorage*. Este custo pode ser considerado aceitável para várias aplicações, dados os benefícios providos pelo mecanismo de tolerância a falhas de aplicações. Uma análise aprofundada dos experimentos e seus resultados, bem como comparações com outros importantes trabalhos relacionados a tolerância a falhas de aplicações em grades de computadores, podem ser encontrados em [Lopes and Silva 2006] e [Lopes 2006].

3. Grade Pervasiva

Uma das promessas da computação em grade é fornecer acesso a recursos computacionais de alto desempenho de forma simples e transparente, analogamente ao que ocorre com o acesso à rede elétrica (*power grid*). Isto significa que usuários de grades de computadores devem poder utilizar os serviços providos pela mesma a partir de casa, do trabalho, e até mesmo durante sua locomoção. Entretanto, para que esta promessa se torne realidade, é necessário que a infraestrutura da grade forneça um mecanismo através do qual os usuários possam ter acesso aos serviços da mesma, independentemente de localização física. Será utilizado o termo *grade pervasiva* para denotar este mecanismo.

O desenvolvimento de uma infraestrutura de grade pervasiva envolve uma série de desafios introduzidos pela mobilidade de seus usuários. Esta infraestrutura pode, no entanto, ser dividida em dois diferentes mecanismos, de acordo com o tipo de mobilidade que deverá ser suportada pela infraestrutura da grade: (1) usuários nômades e (2) usuários móveis. A mobilidade nômade é caracterizada pelo deslocamento de usuários / dispositivos através de limites institucionais [Kleinrock 1997, McKnight et al. 2004], permanecendo desconectados enquanto se locomovem, e.g. um usuário que esteja utilizando um acesso *dial-up* em uma localidade deve desconectar-se antes de mudar para outra localidade, reconectando-se novamente no destino. Ao contrário do que ocorre com usuários nômades, usuários móveis não devem ter suas conexões interrompidas durante sua locomoção [McKnight et al. 2004], sendo estas conexões mantidas através de tecnologias de comunicação sem fio, e.g. um usuário acessando a Internet através de um *Smart-Phone* utilizando tecnologia GPRS, representa um usuário móvel.

O mecanismo de grade pervasiva do MAG (chamado *PervMAG*) propõe a extensão da arquitetura básica do *middleware* MAG de forma a permitir a utilização dos serviços da grade por parte de usuários móveis e nômades. Para dar suporte a estes usuários foram criados dois mecanismos que representam as duas frentes deste projeto:

- *Mecanismo de suporte a usuários móveis*: através deste mecanismo, usuários de dispositivos de computação móvel podem interagir com o MAG para utilizar seus serviços, podendo a grade funcionar como uma extensão do poder computacional de seus dispositivos. Na versão atual, a comunicação entre os dispositivos e a infraestrutura da grade se dá através da tecnologia de comunicação sem fio IEEE 802.11x e utiliza-se dispositivos PalmOS;
- *Mecanismo de suporte a usuários nômades*: este mecanismo prevê que os usuários possam interagir com a infraestrutura da grade através da Internet, utilizando para isto uma interface web. Assim, sempre que o usuário desejar se conectar à grade, poderá fazê-lo através de qualquer computador que tenha acesso à web.

Através destes mecanismos é possível interagir com o MAG para solicitar serviços. Entre estes serviços encontram-se os de submissão, acompanhamento e coleta dos resultados de execuções de aplicações. As arquiteturas dos mecanismos de acesso à grade por usuários móveis e nômades serão detalhadas nas seções a seguir.

3.1. Suporte a usuários móveis

Dispositivos de computação móvel normalmente apresentam restrições de hardware que impedem seus usuários de realizar tarefas que exijam um alto poder de processamento ou uma grande capacidade de armazenamento de dados, restringindo as classes de aplicações que podem ser executadas pelos mesmos. Estas limitações tornam os usuários de dispositivos móveis um emergente segmento de clientes de grades, uma vez que estas podem funcionar como uma extensão dos recursos computacionais destes dispositivos, adicionando-lhes novas capacidades e funcionalidades [González-Castaño et al. 2002].

O integração desses dois novos paradigmas apresenta desafios inerentes à comunicação sem fio e às características dos dispositivos de computação móvel, como baixa largura de banda, conectividade intermitente, heterogeneidade de dispositivos e carência de recursos computacionais [Phan et al. 2002]. Para tratar estes desafios e torná-los transparentes à infraestrutura da grade foi adotado o modelo *cliente / agente / servidor* proposto por Pitoura et al. [Pitoura and Samaras 1998]. Nesta arquitetura, o *agente* exerce o papel de um *proxy* responsável por realizar a interação entre os *clientes móveis* e o *servidor* (a grade), tornando os desafios da computação móvel transparentes ao *servidor*.

A arquitetura do mecanismo de suporte à usuários móveis do MAG é composta pelos seguintes componentes:

- **Mobile Application Submission and Control Tool (MASCT)**: interface gráfica que permite aos usuários de dispositivos móveis requisitar os seguintes serviços da grade: submeter aplicações, acompanhar o andamento da execução de aplicações, receber notificações e visualizar os resultados das computações finalizadas. Durante períodos de desconexão ou baixa conectividade, o MASCT armazena em *log* as requisições de execução efetuadas por seus usuários, sendo enviadas à grade assim que a conexão é reestabelecida;
- **Mobile Proxy Agent**: representa o *agente* da arquitetura *cliente / agente / servidor*. O *MobileProxyAgent* é responsável por tornar os dispositivos móveis e os problemas relacionados à comunicação sem fio transparentes aos componentes da grade. O *MobileProxyAgent* recebe as requisições de cada MASCT executando nos dispositivos móveis e as encaminha para execução na grade. Notificações de aplicações finalizadas são encaminhadas ao *MobileProxyAgent* para serem, então, enviadas aos dispositivos móveis que as requisitaram. Caso o dispositivo esteja desconectado esta notificação será armazenada em um banco de dados (mantido pelo componente *DataBaseAgent*) até que a conexão de rede do dispositivo seja novamente estabelecida;
- **Mobile Local Resource Manager (MLRM)**: monitora os recursos do dispositivo móvel (e.g. energia da bateria e memória disponível) e informa ao *MobileProxyAgent* o estado atual dos mesmos. Isto tem como objetivo tornar o *MobileProxyAgent* ciente do estado de conectividade e de disponibilidade de recursos de cada dispositivo para permitir a utilização de adaptação dinâmica de conteúdo nos arquivos de saída das aplicações executadas pela grade. Além disso, o *MobileProxyAgent* é responsável por informar ao MASCT a respeito das condições de rede;
- **Mobile DataBase Agent (MDBA)**: mantém uma base de dados que associa cada execução de aplicação com o dispositivo que a requisitou. O DBA é também res-

ponsável por armazenar características dos dispositivos portáteis. São armazenadas características estáticas, como: o identificador do dispositivo, a profundidade de cores, a dimensão da tela e o poder de processamento; bem como as seguintes características dinâmicas: o endereço de rede do dispositivo, o estado atual da conexão (conectado ou desconectado) e a quantidade de energia da bateria.

3.2. Suporte a usuários nômades

Muitas vezes o acesso à redes sem fio não está disponível, o que torna inviável a utilização da arquitetura de suporte a usuários móveis do MAG. Torna-se então necessário que existam outras maneiras de interagir com a infraestrutura da grade, de forma a garantir acesso ubíquo à mesma. Segundo Kindberg et al. [Kindberg and Barton 2001], a web é a infraestrutura básica para fornecer suporte à usuários nômades, uma vez que pode ser acessada a partir de um enorme (e crescente) número de lugares. A arquitetura de suporte a usuários nômades do MAG é centrada na utilização da tecnologia de serviços web, e é composta pelos seguintes componentes:

- **ASCTWeb ou *Application Submission and Control Tool for Web***: ferramenta visual executada em navegadores que possibilita aos usuários registrarem as aplicações para a execução na grade, acompanhar o estado da execução das aplicações e obter os resultados das computações finalizadas. É uma interface web desenvolvida com a tecnologia JSP (*Java Server Pages*), semelhante à interface do ASCT disponibilizado pelo *middleware* Integrade. É através desta interface que usuários nômades interagem com a grade;
- **MagWS ou *MAG Web Service***: exerce o papel de um *proxy* da grade, cuja interface pública disponibiliza os serviços oferecidos pelo *middleware* MAG. Fornece a interface de comunicação entre a web e a grade, recebendo requisições do AsctWeb e repassando-as à grade (e vice-versa).

4. Grade de Dados

Atualmente um grande volume de informações é gerado em vários domínios de aplicação, originando grandes coleções de dados. Estas coleções encontram-se geograficamente dispersas, assim como os seus usuários. A combinação dessas coleções, a distribuição geográfica de recursos e usuários e também a computação intensiva sobre dados, demandam mecanismos sofisticados de acesso e compartilhamento. Estes mecanismos devem ser capazes de tratar questões como: falta de interoperabilidade entre tecnologias de sistemas de banco de dados (XML, relacional, arquivos), segurança, replicação, transferência rápida e confiável, publicação e descoberta dos dados.

Para discutir essas questões, a computação em grade especializou-se, objetivando fornecer uma infraestrutura para manipular grandes volumes de dados [Foster et al. 2001]. Esta especialização é chamada de *grade de dados* (*datagrid*). Chervernak et. al descreve uma arquitetura geral, enfatizando dois serviços básicos [Chervenak et al. 2001]: (a) *serviço de metadados* e (b) *serviço de dados*. O serviço de metadados fornece os mecanismos necessários para a publicação de dados, de forma que usuários possam descobrí-los através de seus metadados; já o serviço de dados engloba mecanismos de acesso, gerenciamento e transferência de dados, objetivando fornecer transparência quanto às características distribuídas do ambiente.

A arquitetura do serviço de metadados do MAG, chamado MagCat [Sousa et al. 2006], é composta pelos seguintes agentes:

- *CatalogManager*: realiza operações sobre o repositório de metadados (adição, remoção, atualização e consulta);
- *SchemaManager*: responsável pela extensibilidade do esquema de metadados para diferentes domínios de aplicação;
- *SearchAgent*: executa consultas em repositórios distribuídos de metadados;
- *RequestMonitor*: monitora as operações executadas por usuários do MagCat;
- *ReplicationManager*: responsável pela replicação e sincronização de réplicas dos metadados, utilizando as informações coletadas pelo *RequestMonitor* para decidir quando e onde criar réplicas;
- *SecurityAgent*: é responsável pela autenticação e autorização usando mecanismos como o *Community Authorization Service* (CAS) [Pearlman et al. 2002].

A implementação atual do serviço de metadados consiste nos agentes *CatalogManager* e *SchemaManager*, os quais provêem as funcionalidades de publicação e descoberta de metadados e extensão de esquema de metadados, respectivamente. Estes agentes foram projetados de forma a permitir um alto grau de independência do sistema de armazenamento de metadados utilizado. Atualmente é fornecido suporte ao serviço de diretórios LDAP e ao sistema de banco de dados relacional MySQL. Outras importantes características como distribuição, replicação, sincronização de réplicas e segurança são alcançadas através do sistema de armazenamento de metadados.

O serviço de dados do MAG encontra-se em uma versão preliminar. O agente *DataManager* é responsável pelo fornecimento deste serviço, possibilitando o armazenamento e a recuperação dos dados em um repositório de arquivos. Este agente executa no nó gerenciador de cada um dos aglomerados da grade, permitindo o acesso a bases distribuídas. Uma nova versão deste serviço está atualmente sendo projetada de forma a permitir o uso de técnicas de replicação dos dados armazenados nos aglomerados.

O modelo de dados definido para o MagCat enfatiza a extensibilidade e agrupamento de objetos relacionados. A unidade conceitual para representação da informação é um *arquivo*, que pode ser agrupado em *coleções*. Um especialista de um dado domínio pode agrupar os arquivos em *visões* de acordo com suas necessidades. Cada arquivo, coleção e visão tem um identificador único. Cada conceito no modelo de dados tem um conjunto de atributos básicos, mas o especialista do domínio pode adicionar atributos específicos do domínio de sua aplicação.

5. Conclusões e Trabalhos Futuros

Questões como gerenciamento e alocação eficiente de recursos distribuídos, escalonamento dinâmico de tarefas de acordo com a disponibilidade de recursos, tolerância a falhas de nós e aplicações individuais, mobilidade de código, gerenciamento eficiente da execução de aplicações, comunicação entre computações cooperantes executando em nós distintos e diversos aspectos relacionados à segurança, constituem importantes desafios para a construção de um *middleware* de grade.

Considerando algumas características relativas à tecnologia de agentes móveis, como cooperação, autonomia, reatividade e mobilidade, pode-se concluir que este paradigma simplifica potencialmente o desenvolvimento da infraestrutura da grade. Como

prova de conceito, este artigo descreveu o MAG, um *middleware* de grade que explora o paradigma de agentes como uma forma de superar diversos desafios relativos ao desenvolvimento de um *middleware* de grade.

O MAG executa aplicações na grade carregando dinamicamente o código da aplicação em um agente móvel. O agente do MAG pode ser dinamicamente realocado entre nós da grade através de um mecanismo de migração transparente de aplicações denominado MAG/Brakes, permitindo que nós não dedicados sejam utilizados como parte da infraestrutura da grade. O MAG também fornece um mecanismo de tolerância a falhas de aplicações. Este mecanismo é essencial em ambientes de grades, dado que evita a perda de tempo computacional realizado durante longos períodos. Atualmente, o mecanismo de tolerância a falhas do MAG trata somente falhas de colapso de nós, e é baseado na abordagem de *checkpoint*, em que o estado de execução da aplicação é periodicamente salvo em um armazém estável. Assim, se uma falha ocorrer, as computações podem ser recuperadas a partir de seu último *checkpoint*. Finalmente, foi descrita a infraestrutura de grade pervasiva implementada no contexto do *middleware* MAG. Esta infraestrutura permite que os usuários móveis e nômades possam ter acesso à grade através de uma rede sem fio e através da web, respectivamente. Estes mecanismos são um primeiro passo em rumo à promessa de acesso ubíquo da grade, isto é, poder utilizar seu poder computacional a qualquer hora e em qualquer lugar.

Algumas direções futuras do projeto MAG são: investigar abordagens adaptativas para o mecanismo de tolerância a falhas, de forma a prover mecanismos dinâmicos para a detecção e a recuperação de falhas; o desenvolvimento de mecanismos de proteção contra aplicações hostis e defeituosas; o desenvolvimento de mecanismos de balanceamento de carga para melhor aproveitar o mecanismo de migração forte do MAG; prover suporte à aplicações paralelas do tipo MPI, bem como migração e tolerância a falhas de processos MPI; personalização da plataforma JADE de forma a remover serviços que não sejam utilizados pelos agentes do MAG, economizando recursos de memória e processamento.

6. Agradecimentos

O MAG faz parte dos projetos FlexiGrid e Integrade2, financiados pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), processos 50.6689/2004-2 e 55.0094/2005-9, respectivamente. Agradecemos ainda a Estevão Freitas e Gilberto Cunha pelo fundamental apoio empregado no desenvolvimento deste projeto.

Referências

Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., and Tuecke, S. (2001). The Data Grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23:187–200.

Foster, I., Kesselman, C., and Tueke, S. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputing Applications*.

Goldchleger, A., Kon, F., Goldman, A., Finger, M., and Bezerra, G. C. (2004). Integrade: Object-oriented grid middleware leveraging idle computing power of desktop machines. *Concurrency and Computation: Practice & Experience. Vol. 16, pp. 449-459.*

González-Castaño, F. J., Vales-Alonso, J., Livny, M., Costa-Montenegro, E., and Anido-Rifón, L. (2002). Condor grid computing from mobile handheld devices. *SIGMOBILE Mobile Computing Communications*.

Haralick, R. M., Shanmugam, K., , and Dinstein, I. (1973). Textural features for image classification. *IEEE Trans. Syst., Man, Cybern., SMC-3*, pages 610–621.

Illmann, T., Kargl, F., Krueger, T., and Weber, M. (2000). Migration in Java: problems, classifications and solutions. In *MAMA'2000*, Wollongong, Australia.

Kindberg, T. and Barton, J. (2001). A Web-based nomadic computing system. *Computer Networks, Elsevier*, Vol. 35(No. 4):443–456.

Kleinrock, L. (1997). Nomadicity: anytime, anywhere in a disconnected world. *Mobile networks and applications*, 1(4):351–357.

Lopes, R. F. (2006). MAG: uma grade computacional baseada em agentes móveis. Master's thesis, Universidade Federal do Maranhão, Sã Luís, MA, Brasil. In Portuguese.

Lopes, R. F. and Silva, F. J. d. S. (2005a). Migration Transparency in a Mobile Agent Based Computational Grid. In *Proceedings of the 1st WSEAS International Symposium on GRID COMPUTING*, pages 31–36, Corfu, Greece.

Lopes, R. F. and Silva, F. J. d. S. (2005b). Strong Migration in a Grid based on Mobile Agents. *WSEAS Transactions On Systems*, Volume 4(Issue 10):1687–1694.

Lopes, R. F. and Silva, F. J. d. S. (2006). Fault Tolerance in a Mobile Agent Based Computational Grid. In *4th International Workshop on Agent Based Grid Computing. IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'06)*, Singapore. IEEE Computer Society Press.

Lopes, R. F., Silva, F. J. d. S., and Sousa, B. B. d. (2005). MAG: A Mobile Agent based Computational Grid Platform. In *In Proceedings of the 4th International Conference on Grid and Cooperative Computing (GCC 2005)*, Lecture Notes in Computer Science, Beijing. Springer-Verlag.

McKnight, L. W., Howison, J., and Bradner, S. (2004). Wireless grids: Distributed resource sharing by mobile, nomadic, and fixed devices. *IEEE Internet Computing*, pages 24–31.

Pearlman, L., Welch, V., Foster, I., Kesselman, C., and Tuecke, S. (2002). A community authorization service for group collaboration.

Phan, T., Huang, L., and Dulan, C. (2002). Challenge: Integrating mobile wireless devices into the computational grid.

Pitoura, E. and Samaras, G. (1998). *Data Management for Mobile Computing*. Kluwer Academic Publisher.

Sousa, B. B. d., Silva, F. J. d. S., Teixeira, M. M., and Filho, G. C. (2006). MagCat: An Agent-based Metadata Service for Data Grids. In *4th International Workshop on Agent Based Grid Computing. IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'06)*, Singapore. IEEE Computer Society Press.

Truyen, E., Robben, B., Vanhaute, B., Coninx, T., Joosen, W., and Verbaeten, P. (2000). Portable support for transparent thread migration in java. In *ASA/MA*.