Avaliação do Impacto da Implementação de uma Infra-estrutura de Suporte a Contexto na Integração de Dispositivos Móveis a Grades Computacionais

Alaor José da Silva Junior[1], Fábio Moreira Costa[1], Marcio N. Miranda[2]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74001-970 – Goiânia, GO – Brasil

[2]Instituto de Computação – Universidade Federal de Alagoas (UFAL)
BR 104-Norte Km 14 BL 12, 57072-970 – Maceió, AL - Brasil

{alaor,fmc}@inf.ufg.br, marcio@tci.ufal.br

Abstract. *The integration of mobile devices into the realm of grid computing represents a new scenario for mobile computing. This scenario introduces entirely new concepts and challenges that need to be investigated, aiming at a transparent integration for the user and, at the same time, broadening the applications of grid computing. In this work we describe and evaluate ContextGrid, an infrastructure for context-aware computing that provides support for the integration of mobile devices into grid environments. This infrastructure enables the dynamic adaptation of the middleware according to the context variations under which both the grid and the mobile devices are subject. Measurements of resource consumption are also presented in order to demonstrate the feasibility of the architecture for mobile devices.*

Resumo. *A integração de dispositivos móveis em grades computacionais representa um novo cenário para a computação móvel. Esse cenário introduz novos conceitos e desafios que precisam ser investigados, com o objetivo de realizar uma integração transparente ao usuário e, ao mesmo tempo, aumentar o número de aplicações que utilizam grades computacionais. Neste trabalho é descrito e avaliado o ContextGrid, uma infra-estrutura para computação sensível ao contexto que provê suporte para a integração de dispositivos móveis em ambientes de grades computacionais. Essa infra-estrutura possibilita a adaptação dinâmica do middleware de acordo com as variações de contexto às quais estão sujeitos a grade e os dispositivos móveis. Medidas de consumo de recursos são realizadas para demonstrar a viabilidade da arquitetura proposta para implementar o ContextGrid.*

1. Introdução

A integração de dispositivos móveis e portáteis (*laptops*, *smartphones*, *palmtops* e *pocketPCs*) a grades computacionais possibilita explorar novas formas de recursos, onde as grades não são apenas fornecedoras de ciclos de CPU ou espaço de armazenamento, mas também fornecedoras de recursos como, por exemplo, sensores sem fio [16]. A exploração deste novo cenário depara-se com a diversidade dos dispositivos móveis e suas restrições, assim como com a natureza altamente dinâmica do ambiente computacional provido por eles [8]. Entre essas restrições, pode-se citar a

duração limitada da bateria, a conectividade intermitente e de qualidade muito instável e o limitado poder de processamento e armazenamento desses dispositivos. Todas estas características sugerem um ambiente ainda mais variável que os atuais sistemas em grade, com muitas restrições e com cenários diversos de disponibilidade de recursos e conectividade, surgindo como pré-requisito a re-configuração dinâmica dos componentes do ambiente. As aplicações devem se adaptar aos diferentes tipos de dispositivos, à largura de banda e às várias condições do ambiente computacional em que estão inseridas. Essa adaptabilidade permite que as aplicações executem eficientemente em uma grande variedade de condições.

Através de adaptação, uma aplicação pode mudar o comportamento de sua execução ao invés de mantê-lo constante para todas as situações. Para tanto, é preciso monitorar o contexto da aplicação, em termos dos recursos e do ambiente computacional, além de notificar as aplicações sobre suas mudanças. Ao aliar adaptação e monitoramento do contexto, introduz-se um novo conceito, a computação sensível ao contexto [2], onde são utilizadas as informações monitoradas e processadas de modo a prover adaptações do comportamento das aplicações e serviços.

Esse trabalho, ao aliar grades computacionais, dispositivos móveis e computação sensível ao contexto, tem como principal objetivo a definição de uma arquitetura de suporte à computação sensível ao contexto para adaptação dinâmica integrada ao middleware de grade. Essa arquitetura provê a base para a integração de dispositivos móveis em ambientes de computação em grade, permitindo a adaptação dinâmica do suporte de *middleware* de acordo com as variações de contexto destes dispositivos.

O restante deste trabalho está organizado como se segue. Na Seção 2 são apresentados alguns trabalhos relacionados. Na Seção 3 são discutidos alguns conceitos básicos de contexto e adaptação dinâmica e na Seção 4 é realizada uma breve descrição do ambiente MAG/InteGrade, no qual se baseia a infra-estrutura de integração descrita neste trabalho. Na Seção 5 é apresentada a arquitetura do ContextGrid, seu modelo de contexto e seus protocolos, enquanto que na Seção 6 é analisado o seu desempenho a partir dos resultados obtidos com os experimentos realizados. Finalmente, na Seção 7, são apresentados os comentários finais.

2. Trabalhos Relacionados

Considerável esforço de pesquisa tem sido dedicado à solução de problemas relacionados às redes sem fio e dispositivos móveis, às grades computacionais e às formas de integração dos primeiros a estas grades. Dentre os principais trabalhos relacionados, pode-se destacar o de Bruneo [1], que apresenta o uso de agentes móveis para permitir que usuários móveis utilizem recursos distribuídos em grades de forma transparente. Vários outros aspectos relacionados a grades móveis têm sido abordados, como requisitos e modelos de infra-estrutura [14] e a influência e benefícios desse novo modelo nas grades tradicionais [16]. Em relação aos trabalhos citados acima o ContextGrid propõe uma solução unificada para alguns temas que neles são abordados separadamente, tais como: agentes móveis, transparência para o usuários em relação ao uso da grade e mobilidade, especificação de uma infra-estrutura e a definição de um modelo de contexto.

Outro importante trabalhos nesta linha é o ISAM (Infra-Estrutura de Suporte às Aplicações Móveis Distribuídas) [18], que é uma proposta para suporte à mobilidade física e lógica, ciência de contexto, adaptação dinâmica e execução de aplicações altamente distribuídas, através de um *middleware* denominado EXEDA (*Execution Environment for Highly Distributed Applications*). Contudo, a infra-estrutura aqui apresentada tem como principal objetivo desenvolver um ambiente de grade computacional sensível ao contexto, onde o foco é a adaptação dinâmica dos componentes do *middleware* de grade. Destaca-se também o *middleware* MoCA [7, 17], que, de forma semelhante ao nosso trabalho, oferece uma infra-estrutura de suporte a contexto e à sua disseminação, provendo mecanismos e serviços para a construção de componentes adaptáveis com base em mudanças de contexto. Na MoCA, o foco é a adaptação de conteúdo no nível da aplicação, enquanto que neste trabalho o foco está, principalmente, na adaptabilidade do *middleware*.

Merecem destaque também os projetos MoGrid [19] e Akogrimo [20]. O MoGrid visa o desenvolvimento de um *middleware* para grades computacionais compostas por dispositivos sem fio onde se usa redes sem fio *ad hoc* como o sistema de comunicação para grades computacionais móveis. No projeto Akogrimo um dos principais objetivos é permitir que os usuários móveis não somente usem a grade, mas também que os dispositivos e recursos dos mesmos sejam parte da grade. Assim com no ContextGrid, ele faz uso de contexto e as aplicações devem se inscrever em um gerenciador de contexto. O Akogrimo tem seu modelo de contexto focado no usuário.

3. Contexto e Adaptação

Os humanos estão aptos a usar informações implícitas ao contexto para complementar e favorecer uma correta interpretação de um diálogo. No entanto, para que o contexto possa ser usado na interação homem-máquina, primeiramente é preciso compreender o que é contexto. Sendo um conceito relativamente recente, ainda não existe um consenso na sua definição. Atualmente, entre as várias definições existentes, a definição proposta por Dey [6] é a mais referenciada e aceita: "Contexto é qualquer informação que possa ser usada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, um lugar ou um objeto considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a própria aplicação". Essa definição torna um pouco mais fácil para o desenvolvedor definir o contexto para o cenário da sua aplicação. Assim, se uma informação puder ser usada para caracterizar a situação de um participante ou dos componentes envolvidos em uma interação usuário-aplicação, esta informação é considerada *contexto*.

À medida que são utilizadas essas informações de contexto, de modo a auxiliar na adaptação de aplicações e serviços, tem-se um *sistema sensível ao contexto* ou *ciente de contexto* [6]. A Figura 1 ilustra um modelo geral desse tipo de sistema. Nele temos uma série de sensores de *software* ou de *hardware* que monitoram o ambiente e os dispositivos, além de qualquer outro aspecto de interesse. As informações colhidas por esses sensores são passadas a um conjunto de serviços de contexto, onde são processadas e, se for o caso, modificadas para que possam ser entregues às aplicações e dispositivos que farão uso do contexto para auxiliar nas decisões de adaptação.

A computação sensível ao contexto apresenta inúmeras vantagens para as aplicações e serviços fornecidos por um sistema, principalmente em ambientes

extremamente dinâmicos e diversificados, como aqueles que possuem dispositivos móveis. Uma arquitetura sensível ao contexto, capaz de reagir e adaptar-se, possibilita: um melhor aproveitamento dos recursos existentes, uma melhor interação homem-máquina e uma melhor adequação às peculiaridades de cada dispositivo.

Figura 1 – Sistema Sensível ao Contexto

4. MAG/InteGrade

O InteGrade [10] é uma plataforma de *middleware* de computação em grade para o desenvolvimento de aplicações que podem fazer uso da capacidade de processamento ociosa de computadores compartilhados, sendo considerado um sistema de grade oportunista, a exemplo do sistema Condor [3]. O InteGrade é organizado em uma estrutura hierárquica, onde as unidades estruturais são os aglomerados (*clusters*), conjuntos de máquinas (nós) agrupadas de acordo com algum critério, como localização ou domínio administrativo. Dentre os componentes do InteGrade, destacam-se:

LRM (Local Resource Manager): executa em todos os nós que fornecem recursos, é responsável pela coleta de informações referentes à disponibilidade de recursos e pela execução de uma tarefa escalonada para o nó.

GRM (Global Resource Manager): concentra as informações de todos os LRMs do aglomerado e atua como escalonador de tarefas na grade.

ASCT (Application Submission and Control Tool): ferramenta que permite ao usuário submeter aplicações para serem executadas na grade, assim como controlá-las e coletar os resultados.

O projeto MAG (*Mobile Agents Technology for Grid Computing Environments*) [13] tem como base o InteGrade e pode ser considerado uma extensão do mesmo, provendo um mecanismo de execução de aplicações baseado em agentes móveis. O MAG permite a execução de aplicações paramétricas (*bag-of-task*s) escritas em Java, denominadas aplicações MAG, que executam sobre a camada MAG com o suporte de agentes móveis. Ele incorpora novos serviços ao InteGrade como: tolerância a falhas, liberação de nós e migração de tarefas. O uso de agentes móveis tem especial interesse devido à própria dinâmica do ambiente de grade e visa resolver problemas de alocação de recursos, balanceamento de carga e mobilidade de código [12]. Para tanto, o MAG incorpora alguns novos componentes ao InteGrade, entre os quais pode-se destacar:

***MagAgent*:** é um agente móvel responsável por instanciar e executar uma aplicação MAG, por monitorar sua execução, migrar a aplicação para outro nó caso seja solicitado e salvar o estado de sua execução periodicamente (*checkpointing*). Cada aplicação tem um *MagAgent*, sendo por ele encapsulada e executada como uma *thread* do mesmo.

***AgentHandler*:** é um componente que executa em cada nó da grade. Ele mantém um contêiner de *MagAgents* que executam no nó, sendo responsável por: criar um novo

MagAgent em resposta a uma requisição de execução de uma aplicação MAG, notificar os *MagAgents* para que migrem para outro nó, monitorar os agentes por ele criados e coletar os resultados de um agente que finalizou sua tarefa. O *AgentHandler* interage com o componente LRM do InteGrade de forma a tornar o escalonamento de aplicações MAG transparente ao InteGrade.

***mobileProxyAgent*:** é um agente responsável por permitir a conexão de dispositivos móveis ao MAG, convertendo as requisições dos dispositivos que se conectam a ele por *sockets* em requisições válidas na grade. Atualmente, o *mobileProxyAgent* é visto como um ASCT para uma grade fixa.

Neste trabalho, o termo MAG/InteGrade se refere a um *middleware* de grade computacional completo, formado pelo InteGrade e pelas extensões do MAG.

5. ContextGrid

ContextGrid [4,5] é uma infra-estrutura que fornece suporte para computação sensível ao contexto e explora o uso do contexto no auxílio a adaptações dinâmicas dos componentes do *middleware* de grade. É uma extensão do MAG/InteGrade para coletar, manipular e disponibilizar informações de contexto, possibilitando o uso dessas informações na adaptação dinâmica dos componentes do *middleware.* Seu objetivo é integrar dispositivos móveis à grade, otimizando o uso de seus recursos através de componentes configurados e re-configurados conforme o contexto atual da grade.

5.1 Arquitetura

A arquitetura do ContextGrid foi desenvolvida a partir da arquitetura MAG/InteGrade. A Figura 2 ilustra a arquitetura em camadas do MAG/InteGrade com a extensão do ContextGrid. A camada do ContextGrid acrescenta novas possibilidades para o escalonamento de aplicações no MAG/InteGrade, como a execução ciente de contexto onde passa-se a considerar o contexto do usuário, o contexto dos nós, o contexto geral da grade e o contexto da aplicação no momento da escolha dos nós para execução de uma aplicação. Como exemplo de informações de contexto relevantes podemos citar: quais aplicações estão em execução na grade, a distribuição atual dos recursos disponíveis, requisitos básicos e desejáveis para uma aplicação, usuário que requisitou a sua execução e as plataformas para as quais a aplicação possui código executável. Além da execução de aplicações, o ContextGrid atua também na adaptação dos componentes do *middleware*.

No ContextGrid os componentes desenvolvidos são notificados do contexto e de suas mudanças, ficando sob responsabilidade do desenvolvedor decidir qual(ais) informações de contexto(s) é(são) interessante(s) e qual(ais) ação(ões) deve(m) ser tomada(s) diante de uma determinada mudança de contexto. Isso torna a arquitetura independente das políticas de adaptação adotadas, tornando-a mais flexível.

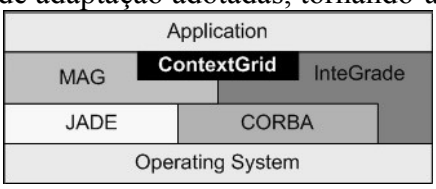

Figura 2 – MAG/InteGrade/ContextGrid

5.2 Componentes

Para compor a infra-estrutura do ContextGrid são introduzidos três componentes principais: o LCM (*Local Context Manager*), o GCM (*Global Context Manager*) e o *Wrapper*. Outros elementos importantes na arquitetura são os *Context Users* e o *Proxy*. A Figura 3 apresenta os componentes do ContextGrid, que são descritos a seguir:

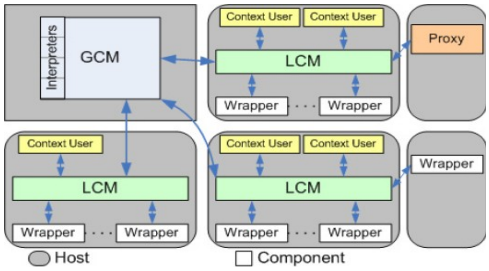

Figura 3 – Componentes do ContextGrid

***Local Context Manager* (LCM):** é o responsável por coletar as informações de contexto das mais diversas fontes através dos *wrappers*. Ele realiza um pré-processamento dessas informações, a fim de convertê-las em um formato padronizado para, posteriormente, enviá-las ao GCM. Essa conversão visa tornar a obtenção e a utilização das informações de contexto independentes das fontes (por ex., rede de sensores, PDAs e telefones celulares), tipo ou do formato em que são fornecidas. Para isso, basta apenas a criar um novo *wrapper* e realizar seu registro em um LCM.

***Global Context Manager* (GCM):** é o responsável por armazenar, processar e interpretar o contexto, transformando-o em tipos mais elaborados, seja pela junção de várias informações de contexto ou pela inferência de novas informações a partir daquelas obtidas dos LCMs como. Por exemplo, dado o contexto de cada dispositivo na grade, pode-se compor uma nova informação de contexto que traduz a taxa de uso dos recursos da grade como um todo, ou ainda, a partir de informações geradas por um receptor GPS (coordenadas), inferir o nome do local correspondente. Também é responsabilidade do GCM a classificação dos contextos em categorias, adotando critérios como similaridade, proximidade ou domínio administrativo. O GCM é responsável, ainda, por realizar notificações de mudança de contexto aos interessados.

Wrapper: é o responsável pela comunicação direta com as fontes de informação de contexto. Ele traduz os dados do formato nativo de cada tipo de sensor ou dispositivo conectado à grade para um dos formatos aceitos pelo LCM. Novos *wrappers* podem ser adicionados dinamicamente para expandir as fontes de informação de contexto.

Context Users: podem ser quaisquer dos componentes do *middleware* de grade, desde que tenham sido implementados ou alterados para fazer uso do contexto e adaptar-se ao mesmo. Contêm as políticas de adaptação e as ações a serem tomadas de acordo com as mudanças no contexto.

***Proxy*:** é o responsável por intermediar todas as interações dos dispositivos móveis com a grade, sendo o representante do dispositivo móvel na rede fixa. O *proxy* apresenta-se para a grade com se fosse o próprio dispositivo e é responsável por aplicar adaptações, principalmente de conteúdo e comunicação, às interações da grade com os dispositivos. Como representante do dispositivo na rede fixa, o *proxy* também é responsável por assumir as funcionalidades de dispositivos nele registrados cujo contexto de comunicação não seja favorável ou caso ocorra uma queda da conexão. Nesse cenário,

pode haver uma avaliação do contexto do usuário que utilizava o dispositivo para decidir a melhor maneira de enviar os dados recebidos durante o período da queda da conexão.

5.3 Modelo e Interpretação do Contexto

Novas técnicas vêm sendo apresentadas por diversos autores [9,15] para suprir as deficiências dos atuais modelos para desenvolvimento de *software*, que não oferecem suporte para a modelagem de aplicações sensíveis ao contexto. Entretanto há uma série de características sobre o contexto e sobre o poder de expressão que estas técnicas ou modelos devem apresentar. A fim de definir uma técnica de modelagem para o ContextGrid, observou-se as seguintes características e nível de expressividade que a técnica deve prover [4]: a informação de contexto possui vários níveis de temporalidade, a informação de contexto nem sempre é confiável, a informação de contexto pode ter diferentes níveis de abstração ou complexidade, as informações de contexto são inter-relacionadas umas com as outras. Para o ContextGrid foi adotado o modelo proposto por Henricksen [11]. Esta escolha deve-se ao fato de que uma diagramação do contexto, de forma visual, traz inúmeras vantagens para a sua compreensão e modelagem, além deste tipo de modelagem apresentar as características mencionadas para as informações de contexto. Outro aspecto que determinou a escolha desse modelo em detrimento de outros mais complexos foram as características limitadas dos dispositivos envolvidos, o que torna importante considerar o poder de processamento exigido pela efetiva manipulação da modelagem em si. Seguindo o meta-modelo proposto por Henricksen [11], a Figura 4 apresenta o modelo de contexto elaborado para o ContextGrid. Esse modelo representa uma visão do ambiente de computação em grade adotada neste trabalho. A técnica adotada é fundamentada em uma abordagem objeto-relacional, na qual o contexto é modelado em termos de um conjunto de entidades, que são elementos abstratos e representam as fontes de informação, a saber: usuário, aplicação, dispositivo, conectividade e ambiente computacional. Para cada entidade são definidos atributos que correspondem às informações de contexto sobre as mesmas, como preferências do usuário, modelo do dispositivo e largura de banda disponível. Os atributos coletados juntamente com seus inter-relacionamentos caracterizam o contexto do ambiente em um dado momento. Graficamente, as entidades são representadas por retângulos com bordas duplas; os atributos por retângulos com bordas simples e as associações ou inter-relacionamentos por um segmento de reta dirigido.

Em aplicações sensíveis ao contexto a interpretação do contexto é essencial. Dey [6] trata a interpretação do contexto como um processo onde se eleva o nível de abstração das informações de contexto, gerando-se informações mais elaboradas, tendo como ponto de partida informações primitivas. No ContextGrid este aspecto é tratado no LCM e no GCM [4]. O primeiro passo da interpretação de contexto ocorre no LCM, onde as informações de contexto são transformadas em um padrão uniforme baseado em XML. Essa informação, agora chamada contexto, é então passada ao GCM, onde, com o uso de módulos interpretadores, o nível de abstração do contexto é elevado ainda mais. O GCM ao receber um contexto do LCM, verifica quais atributos esse contexto contém e então verifica se existe um interpretador em particular para esse(s) atributo(s). Caso exista, o GCM faz a carga do interpretador e repassa para ele o contexto recebido. Ao final do processamento, o interpretador retorna um novo contexto ao GCM. O

contexto é mantido ao longo do processo, permitindo atender diferentes aplicações que necessitem de contexto em diferentes níveis. Com exemplo de um interpretador de contexto pode-se citar um que ao receber do LCM o contexto do ambiente computacional de um dado nó, utiliza-se das informações dos serviços que executam no nó para classificá-lo em nó de submissão de tarefas (nó de usuário), nó gerente ou nó provedor de recursos. Essa informação pode ser usada pelo GRM para adaptar o algoritmo de escalonamento.

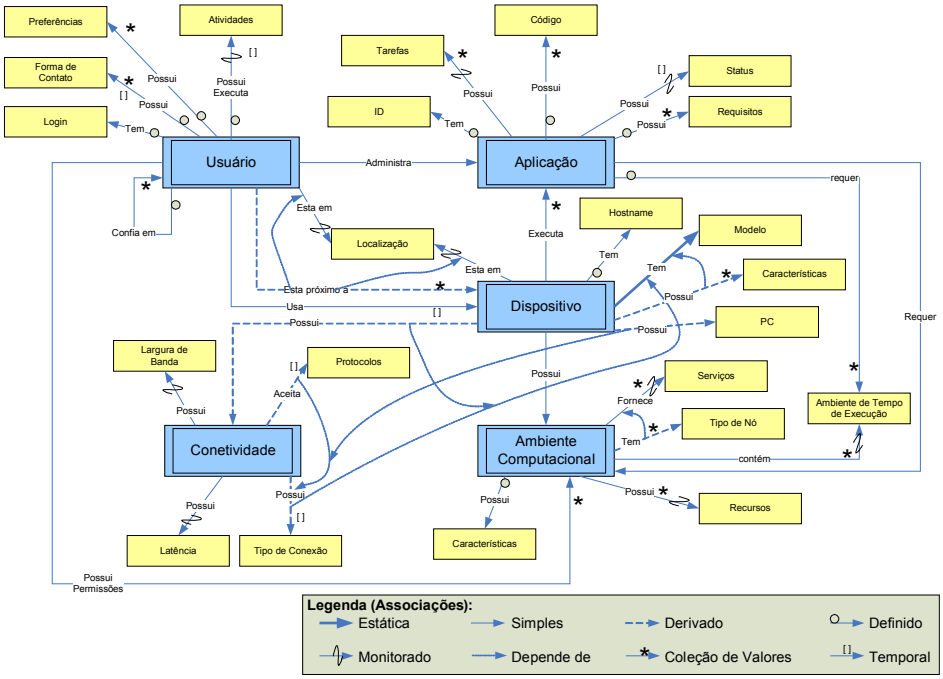

Figura 4 – Modelo de Contexto do ContextGrid

5.4 Protocolo de Atualização de Contexto

Este protocolo permite que o GCM mantenha o contexto global do MAG/InteGrade. A definição da periodicidade com a qual as informações de contexto são atualizadas é fundamental. Caso essas atualizações sejam freqüentes demais, os dispositivos móveis terão suas baterias drenadas e a rede pode ser tomada por este tipo de mensagem. Por outro lado, se o intervalo for grande demais, o GCM não terá um contexto que reflete o real estado do sistema. Dessa forma, o LCM utiliza tempos distintos e métricas específicas para decidir quando uma informação proveniente de um sensor dever ser repassada ao GCM. O LCM associa um intervalo para cada tipo de informação de contexto monitorada pelos *wrappers*, assim como prioridades e métricas que irão indicar quando a informação será passada ao GCM. Uma métrica pode ser a percentagem de variação acima da qual o contexto deve ser atualizado como, por exemplo, quando a fração de uso da CPU for mais de 10% diferente da medida anterior.

5.5 Protocolo de Notificação de Contexto

Este protocolo permite ao ContextGrid disseminar as informações de contexto pelo *middleware* de grade, possibilitando que elas cheguem aos *Context Users*, os quais dependem das mesmas para aplicar suas políticas de adaptação. Esse protocolo é dividido em duas partes: a inscrição em um canal de eventos e a notificação de

mudanças de contexto. A inscrição em um canal de eventos trata da inscrição de um *Context User* em um ou mais canais de eventos disponíveis, onde são escolhidos os canais que melhor refletem os contextos de interesse e as condições que devem ser satisfeitas para que o contexto seja efetivamente enviado a ele. A notificação de mudanças de contexto trata de como é feita a notificação dos *Context Users* quando ocorre uma mudança no contexto. Maiores detalhes sobre a implementação destes protocolos podem ser obtidos em [4].

6. Resultados

Foram realizados diversos testes com o objetivo de avaliar o impacto da implementação do ContextGrid sobre dispositivos móveis conectados à grade. Os testes foram realizados em um ambiente MAG/InteGrade formado por duas máquinas fixas e por dois dispositivos móveis: um celular Nokia 3650 com SynbianOS 6.1 e um PocketPC Dell Axim x50v com Windows Mobile 2003, ambos sistemas operacionais multitarefa de 32 bits. As máquinas fixas foram interligadas através de uma rede Fast-Ethernet, a 100 Mbps. O PocketPC foi conectado a seu *proxy* via rede sem fio 802.11b, com taxa de transmissão nominal de até 11 Mbps. O celular utiliza uma conexão *bluetooth*, com taxa de transmissão de até 1 Mbps. Foram realizados três experimentos quantitativos, um para cada tipo de dispositivo utilizado (fixo, pocketPC e celular), com o objetivo de avaliar o impacto do componente LCM. O componente LCM foi avaliado quanto ao protocolo de atualização de contexto. Por ser executado em todos os nós, inclusive nos nós móveis e por tempo indefinido, ele não deve consumir muitos recursos, principalmente dos dispositivos que possuem recursos limitados e que dependem de bateria. Ele também não deve interferir nas atividades do usuário. Os experimentos foram realizados com o LCM configurado para avaliar as informações de contexto a cada 10 segundos e, caso elas fossem 10% diferentes das anteriores, deveriam ser enviadas ao GCM. Foram realizados 3 experimentos de 5 minutos cada, para efeito do cálculo da média. Durante cada experimento foram coletadas 300 amostras.

6.1 Celular

A Figura 5 apresenta a média de consumo de CPU para o experimento realizado no celular. Pode-se notar que o consumo se mantém baixo, por volta de 3%, ficando muito próximo de zero na maior parte do tempo.

Neste experimento o LCM foi desenvolvido em Python, com o objetivo de facilitar a portabilidade. Observa-se que o consumo de memória, absoluto e o relativo ao disponível no dispositivo, com o LCM e com o interpretador Python é, em média, de 1692 KB (51,64%), sendo 1324 KB (40,41%) ocupados pelo interpretador Python e 368 KB (11,23%) ocupados pelo LCM. Apesar do consumo um pouco elevado, a memória restante é suficiente para que todos os aplicativos e funções do dispositivo operem normalmente. A implementação do LCM apresentou um tamanho elevado em memória. Isto se deve, principalmente, à carga de bibliotecas necessárias para o acesso às informações do dispositivo. Uma futura otimização no código do LCM pode reduzir significativamente este tamanho e diminuir o consumo de CPU. Isto pode ser feito reescrevendo-se o código do próprio componente ou realizando alterações em bibliotecas de modo a deixar somente as funcionalidades utilizadas pelo LCM.

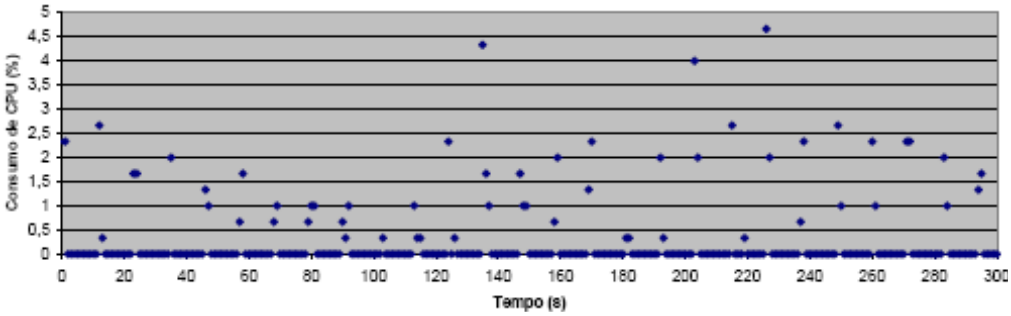

Figura 5 – Consumo de CPU do LCM – Celular

6.2 PocketPC

Neste caso, a falta de documentação e algumas restrições do Windows Mobile 2003 não permitiram a coleta direta de informações sobre o uso do processador. Na implementação do LCM os dados coletados para o atributo "recursos" ficaram restritos à memória primária e secundária. Foi possível avaliar o impacto do LCM apenas com o uso de um *software* de código fonte fechado, denominado RhinoStats, que apresenta os resultados em um gráfico na tela. Segundo observações dos dados apresentados por esse *software*, o consumo de CPU do LCM foi, em média, de 3%, mas nunca superior a 5%. O LCM também foi implementado em Python. O total de memória, absoluto e a relativo ao disponível no dispositivo, utilizada foi, em média, de 1960 KB (2,99%), sendo 1730 KB (2,64%) ocupados pelo interpretador da linguagem e 230 KB (0,35%) ocupados pelo LCM. A memória ocupada pelo LCM nesse dispositivo foi menor do que a do celular. Mas, por outro lado, a quantidade ocupada pelo interpretador foi maior. Isso se deve às características de cada interpretador de Python. O menor consumo de recursos do LCM nesse dispositivo deve-se à necessidade de importar um número menor de bibliotecas do que no celular.

6.3 Nó fixo

Na Figura 6 são apresentados os resultados do experimento para o nó fixo. Em média, o LCM apresentou um consumo de CPU de 1%. Nota-se que a grande maioria dos valores coletados do uso do processador se aproxima de 0% e nunca é superior a 3,5%, o que confirma o baixo consumo de processador por parte do componente em questão.

Neste tipo de dispositivo o LCM foi implementado em Java. O total de memória utilizada, absoluto e a relativo ao disponível no dispositivo, foi, em média, de 9 MB (1,76%). Esse consumo mais elevado se deve ao uso da linguagem Java que, em geral apresenta um consumo maior de memória. Contudo, o consumo de 9 MB, para um nó fixo, é bastante aceitável, um vez que, neste caso, dispõe-se uma quantidade de recursos bem superior aos dois primeiros tipos de dispositivos apresentados. Uma otimização dessa implementação pode ser atingida da mesma forma proposta para o caso do celular, podendo contribuir para a redução significativa na memória em uso pelo LCM.

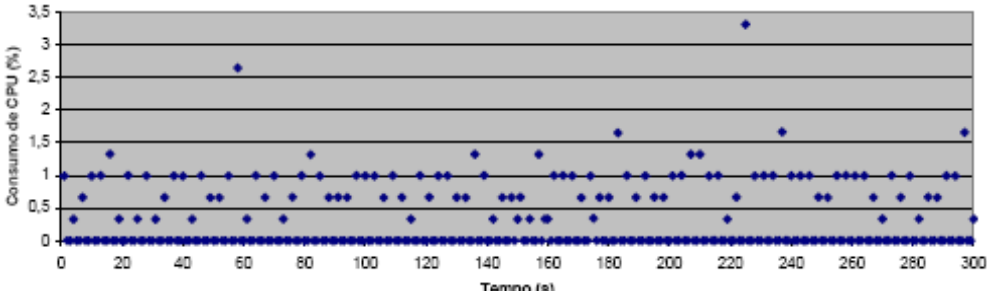

Figura 6 – Consumo de CPU do LCM – Nó Fixo

7. Comentários Finais

Considerando os novos paradigmas de computação ubíqua e a presença cada vez maior de dispositivos móveis, a abordagem seguida neste trabalho visa simplificar a integração de dispositivos móveis às grades computacionais, amenizando problemas como a diversidade de dispositivos e o comportamento dinâmico do ambiente. Os resultados obtidos demonstram que o uso de contexto é viável, mesmo em dispositivos com poucos recursos. O LCM apresentou um consumo de CPU bastante aceitável. Uma questão não muito satisfatória foi o consumo de memória nos dispositivos móveis, principalmente para o celular, cujos recursos são extremamente restritos. Testes para mensurar o impacto das mensagens trocadas pelo ContextGrid, o impacto causado nos componentes do *middleware* pela avaliação do contexto e pela aplicação de políticas de adaptação são etapas importantes para uma completa análise do ContextGrid e são temas para um próximo trabalho. Entre outros trabalhos futuros pode-se destacar: a investigação do uso do contexto para adaptação das aplicações de usuário, a utilização de reflexão computacional ciente de contexto no *middleware* de grade e a exploração dos conceitos de computação ubíqua em grades.

Agradecimentos

Este trabalho é suportado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico(CNPq), através do Edital 031/2004, Processo 506689/2004-2, pela Coordenação de Aperfeiçoamento de Pessoal de Ensino Superior (CAPES) e pela Fundação de Apoio à Pesquisa do Estado de Alagoas (FAPEAL)

Referências

[1] D. Bruneo, M. Scarpa, A. Zaia, and A. Puliafito. Communication paradigms for mobile grid users. *International Symposium on ClusterComputing and the Grid (CCGRID)*, 00:669, 2003.

[2] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical report, Dartmouth College, Computer Science, Hanover, NH, USA, 2000.

[3] Condor project homepage. http://www.cs.wisc.edu/condor/, mai 2006.

[4] A. J. da Silva Junior. ContextGrid: Uma infra-estrutura de suporte a contexto para integração de dispositivos móveis à computação em grade. Master's thesis, Instituto de Informática - Universidade Federal de Goiás, Goânia, GO, Brasil, Jul 2006.

[5] A. J. da Silva Junior, M. N. de Miranda, and F. Costa. Contextgrid - an infrastructure for context support for integration of mobile devices into the computational grid. In *Anais do IV Workshop on Grid Computing and Applications*, Curitiba, PR, Julho 2006.

[6] A. K. Dey. *Providing architectural support for building context-aware applications*. PhD thesis, College of Computing, Georgia Institute of Technology, Nov 2000. Director-Gregory D. Abowd.
[7] J. V. Filho, V. Sacramento, R. da Rocha, and M. Endler. Moca: Uma arquitetura para o desenvolvimento de aplicações sensíveis ao contexto para dispositivos móveis. In *Proceedings of the XXIV Simpósio Brasileiro de Redes de Computadores (SBRC)*,ToolSession, volume II, Curitiba, Brazil, May 2006.

[8] G. H. Forman and J. Zahorjan. The challenges of mobile computing. Computer, 27(4):38–47, Apr 1994.

[9] C. Ghidini and F. Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. Artif. Intell., 127(2):221–259, 2001.

[10] A. Goldchleger. Integrade: Um sistema de middleware para computação em grade oportunista. Master's thesis, IME - Universidade de São Paulo, dez 2004.
[11] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. In *Proceedings of the First International Conference on Pervasive Computing (Pervasive '02)*, pages 167–180, Zurich, Switzerland, June 2002.
[12] D. B. Lange and M. Oshima. Seven good reasons for mobile agents. *Commun. ACM*, 42(3):88–89, 1999.

[13] R. F. Lopes. Mag: uma grade computacional baseada em agentes móveis. Master's thesis, Universidade Federal do Maranhão, São Luís, MA, Brasil, Jan 2006.
[14] L.W. McKnight, J. Howison, and S. Bradner. Guest editors' introduction: Wireless grids–distributed resource sharingby mobile, nomadic, and fixed devices. *IEEE Internet Computing*, 8(4):24–31, 2004.
[15] P. Oztürk and A. Aamodt. Towards a model of context for case-based diagnostic problem solving. In *Context-97: Proceedings of the interdisciplinary conference on modeling and using context*, pages 198–208, Rio de Janeiro, RJ, Brazil, Feb 1997.
[16] T. Phan, L. Huang, and C. Dulan. Challenge: integrating mobile wireless devices into the computational grid. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 271–278, Atlanta, Georgia, Sep 2002.
[17] H. Rubinsztejn, M. Endler, and N. Rodrigues. A framework for building customized adaptation proxies. In *Proceedings of the IFIP conference on Intelligence in Communication Systems (INTELLCOMM2005)*, Montreal, Canadá, Oct 2005.
[18] A. Yamin, J. Barbosa, I. Augustin, L. Silva, and C. G. and G.Cavaleiro. Towards merging context-aware, mobile and grid computing. *International Journal Of High Performance Applications*, 17(2):191–203, 2003.
[19] L. Lima, A. Gomes, A. Ziviani, M. Endler, L. Gomes, and B. Schulze. Peer-to-Peer Resource Discovery in Mobile Grids. In *Proceedings. of the 3rd International Workshop on Middleware for Grid Computing -- MGC 2005*, Grenoble, France, November, 2005.
[20] Akogrimo Project, http://www.akogrimo.org, Feb 2007