Uma Abordagem Adaptada para Execução Coordenada de Aplicações em uma Configuração de Grade Móvel

V.C.M. Borges, A.G.M. Rossetto e M.A.R. Dantas
Universidade Federal de Santa Catarina (UFSC),
Departamento de Informática e Estatística (INE),
Laboratório de Pesquisa em Sistemas Distribuídos (LaPeSD),
88040-900 Florianópolis - SC, Brasil.

E-mail: {vcunha, anubis, mario}@inf.ufsc.br

Resumo

Alguns trabalhos relacionados na presente pesquisa abordam a interação entre dispositivos móveis e grades não tratam alguns problemas ou restrições destes aparelhos, como por exemplo, tempo de vida da bateria limitada e desconexões freqüentes por causa da natureza intermitente do sinal da rede wireless. Este artigo apresenta uma abordagem para fornecer uma forma adaptada para execução de aplicações em configurações de grades móveis que trate estes problemas. Através desta abordagem foi alcançada uma adequação do fluxo de execução, em caso de desconexão, considerando os requisitos da aplicação submetida a partir dos dispositivos móveis e as opções definidas pelo usuário. Em adição, os resultados experimentais evidenciam também que a abordagem consumiu menos energia da bateria de um PDA para submeter e monitorar aplicações em comparação com alguns trabalhos relacionados. Para atingir tais fins, foram desenvolvidas interfaces com características otimizadas e adaptadas conforme alguns problemas característicos e limitações encontradas nos dispositivos móveis.

1. Introdução

A resolução de problemas complexos vem se tornando possível devido a grande evolução observada em termos de hardware e software. Desta forma, tem-se utilizado essa evolução para o processamento paralelo envolvendo recursos heterogêneos e de alto poder de processamento, tais como os recursos encontrados em configurações de grade computacional. Entretanto, limitações geralmente encontradas em dispositivos móveis impõem dificuldades para fornecer aos usuários uma opção para resolver problemas complexos quando usando estes dispositivos. Por esta e outras razões, a infra-estrutura de grade móvel foi proposta [18, 4, 17, 23, 13], na qual integra dispositivos móveis e o ambiente de grade computacional, possibilitando fornecer aos dispositivos móveis os recursos compartilhados pela grade. Segundo [4, 17, 13, 9], o ambiente de grade móvel pode ter dois aspectos de interação: dispositivos são considerados como usuários dos recursos de grades (interfaces) ou como próprios recursos da grade (por exemplo, processando alguma tarefa, microfones, câmeras, receptores GPS e sensores). O poder computacional desses dispositivos tem apresentado melhorias nos últimos anos. Entretanto, a atual capacidade de processamento e armazenamento ainda não os habilita a resolução de problemas complexos. Desta forma, neste trabalho de pesquisa, os dispositivos móveis são considerados como usuários de recursos da grade.

Os seguintes trabalhos relacionados [17, 23, 21, 6] somente permite características de submissão e monitoração de uma tarefa por vez em um dispositivo específico. Na abordagem desses trabalhos, os usuários podem submeter tarefas de uma aplicação em ordem errada para resolver um problema. É possível, também, ocorrer atrasos entre submissão de tarefas devido à retransmissão causadas pela alta taxa de erros em uma rede sem fio. Desta forma, torna-se interessante ter uma abordagem onde estes dispositivos, dentro do seu contexto, poderiam mostrar uma facilidade coordenada, automatizada e transparente relacionada ao fluxo de execução de tarefas em grade para resolver um único problema.

Outro aspecto importante considerado neste estudo é a capacidade de energia da bateria dos dispositivos móveis, que vem tendo poucos avanços em minimizar o seu consumo, causando restrições na utilização destes dispositivos.

Por exemplo, a capacidade de processamento destes dispositivos para executar alguma aplicação não poderá ser mais obtida por aumentos crescentes da capacidade de seus processadores, em função do impacto disto sobre a energia consumida da bateria. Desta forma, seria interessante se os dispositivos móveis pudessem usufruir alternativas que forneçam soluções para reduzir o consumo de bateria desses dispositivos, como por exemplo, processamento remoto para execução de suas aplicações (isto é, utilizar o poder de processamento fornecido pela grade) e/ou interfaces (abordagens) que ajudem a prolongar o tempo de vida da bateria.

Em adição, considerando sua natureza móvel e de recursos limitados, estes dispositivos tornam-se menos confiáveis por serem mais suscetíveis a desconexões durante a execução de uma aplicação. Por exemplo, desconexões podem ser causadas pelo tempo de vida da bateria reduzida ou interferências na rede *wireless*. A ocorrência de uma desconexão voluntária (ou involuntária) do dispositivo móvel é uma falha. Esta falha pode provocar um erro, e este erro pode gerar um defeito. Por exemplo, a falha é a desconexão do dispositivo, o erro é a impossibilidade de receber informações dos usuários móveis (isto é, parâmetros de entrada em uma tarefa específica conforme os resultados obtidos em tarefas anteriores, ou ainda referenciar dados armazenados no dispositivo), e este erro pode gerar um estado de defeito, isto é, um resultado incorreto do fluxo de execução. Por esta razão, torna-se interessante o desenvolvimento de abordagens que apóiem este fluxo de execução e forneçam uma modificação adaptativa em tempo real, para que tal execução leve em conta mudanças ou problemas ocorridos no ambiente móvel enquanto a aplicação é executada.

Este artigo apresenta um trabalho de pesquisa que permite que dispositivos móveis submetam aplicações ao ambiente de grade, baseado no conceito de *workflow*. Neste contexto, uma abordagem foi concebida para executar a submissão e monitoração de aplicações em uma forma coordenada e automatizada. O ambiente tem como contribuição principal possibilitar algumas vantagens para usuários móveis, isto é, ele possui formas adaptativas e otimizadas para fornecer um melhor ajuste para alguns problemas e limitações encontrados nos dispositivos móveis (por exemplo, intermitência da rede sem fio, tempo de vida da bateria reduzido e por fim, pequeno tamanho de tela).

O artigo é organizado como segue. Na seção 2, são apresentadas características verificadas nos trabalhos relacionados. Aspectos interessantes do projeto e implementação da arquitetura são mostradas na seção 3. Na seção 4, é apresentada uma comparação entre esta abordagem e a abordagem apresentada na maioria dos trabalhos relacionados. Finalmente, na seção 5 algumas conclusões e trabalhos futuros são apontados.

2. Trabalhos Relacionados

Em [23] é proposta uma arquitetura que fornece um método uniforme para gerenciar interações de QoS entre dispositivos móveis e grade. Em [21] é apresentado o AutoMAGI, um *middleware* autonômico que pode tratar a complexidade de estender o potencial da grade aos dispositivos móveis. O AutoMAGI incorpora características de *context-awareness*, auto-configuração, auto-otimização, auto-cura e auto-proteção. A contribuição de pesquisa apresentada em [6] trata o problema de possibilitar acesso a serviços interoperáveis baseado em ontologias. Entretanto, estes três exemplos representativos da literatura não consideram o desafio de submissão e monitoração de várias tarefas de uma aplicação em recursos da grade para resolver um único problema de forma automatizada e coordenada. Estas abordagens permitem submeter e monitorar uma tarefa por vez na interface do dispositivo móvel, com a sua ordem de submissão controlada pelo usuário móvel.

Há vários trabalhos em grade que utilizam *workflows* para controlar a execução do fluxo de tarefas (Condor DAGMan [24], [22, 26, 10]). No entanto, nenhum destes trabalhos abrangem os dispositivos móveis como recurso ou interface de acesso a grade. Em adição, as abordagens apresentadas em [3, 8], sugerem que clientes PDAs possam submeter várias tarefas que trabalham juntas para resolver o mesmo problema empregando o conceito de *workflow*. Por outro lado, estas abordagens mostram claramente que clientes PDAs implementam um pequeno conjunto de funcionalidades *workflow*. As contribuições também não mostram as funcionalidades que foram implementadas. Outro aspecto que pode ser observado nestes trabalhos de pesquisa é: não mencionam qualquer adaptação de funcionalidade da interface de submissão e monitoração da aplicação para o contexto dos dispositivos móveis, com intenção de melhor ajustar estas abordagens ao conjunto de problemas e limitações encontradas nestes dispositivos.

A abordagem de [2] também possibilita uma submissão e monitoração de aplicações de forma coordenada e automatizada através do conceito de *workflow*. Em adição, fornece interfaces adaptadas ao pequeno tamanho da tela de um PDA. No entanto, similar a [3, 8], a abordagem em [2] não trata o problema da desconexão dos dispositivos móveis durante a execução de uma aplicação.

Em [17], os autores, particularmente, discutem o problema de desconexão para tratá-lo do ponto de vista que o dispositivo móvel trabalha como recurso da grade, isto é, para executar tarefas no próprio dispositivo. Neste sentido, a abordagem propõe um novo algoritmo de escalonamento baseado na instabilidade do ambiente móvel para prever uma provável desconexão e assim, migrar a aplicação executada para outro dispositivo antes de ocorrer a desconexão. Entretanto, esta forma de interação de grade móvel não per-

mite que os usuários móveis possam utilizar a grande quantidade de recursos compartilhados na grade para resolver problemas complexos que demandem grande capacidade de processamento e armazenamento (por exemplo, seqüenciamento de DNA em bioinformática ou análise de dados de uma galáxia em astro-física). Na Tabela 1 sumariza as características dos trabalhos citados..

Trabalhos Relacionados	Consumo de Energia	Desconexão	Workflow
Park et al. [17]	Não	Sim	Não
Shi et al. [23]	Não	Não	Não
Sajjad et al. [21]	Não	Não	Não
Grabowski et al. [6]	Não	Não	Não
Brooke et al. [3]	Não	Não	Sim
Hummel et al. [8]	Não	Não	Sim
Borges et al. [2]	Não	Não	Sim
Andronico et al. [1]	Não	Não	Não
Novotny et al. [16]	Não	Não	Não

Tabela 1. Características dos Trabalhos Relacionados

Como pode ser visto, a maior parte dos trabalhos relacionados mostrados em [17, 23, 21, 6, 3, 8, 2, 1, 16] não argumentam ou mostram soluções para algumas limitações (ou problemas sérios) destes dispositivos, como por exemplo: pequeno tamanho de tela, tempo de vida da bateria reduzido e, a adaptação do fluxo de execução da aplicação nos recursos da grade em casos de desconexão dos dispositivos. Estes problemas são considerados na abordagem proposta e são tratados em um modo diferencial como característica da presente proposta.

3. Abordagem Proposta

Nesta seção é descrita a arquitetura da abordagem e suas características de projeto e desenvolvimento. A arquitetura é mostrada na Figura 1. Ela foi concebida com três componentes principais: **Gerenciador Workflow**, **Agente** e **GUI Móvel**.

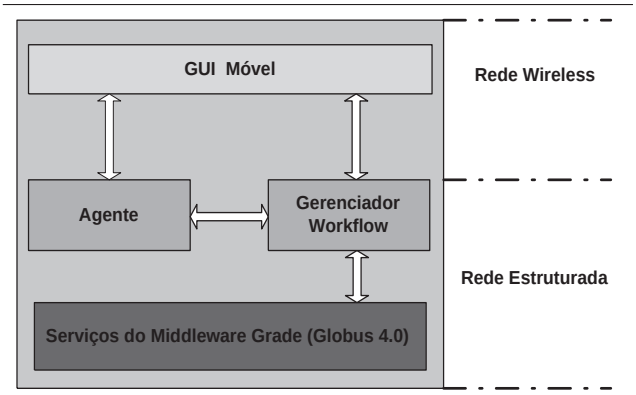

Figura 1. Arquitetura Proposta

Os componentes da arquitetura são como segue:

- **GUI Móvel** apresenta interfaces de acesso único à grade e adaptadas para PDAs;
- **Gerenciador Workflow** processa e gerencia os pedidos (por exemplo, submissão, monitoração de aplicação) vindos dos dispositivos móveis para executar em um ambiente de grade. Este componente possibilita a execução de aplicações na qual ele pode coordenar recursos distribuídos em múltiplas organizações virtuais, obtendo capacidades específicas de processamento através da integração de organizações envolvidas em partes diferentes da aplicação e assim, promover colaborações entre as mesmas;
- **Agente** adapta o fluxo de execução para garantir a consistência da aplicação em uma eventual desconexão.

3.1. Gerenciador Workflow

Devido ao advento de grade computacional, fornecendo vários serviços, recursos e a capacidade de resolver problemas complexos, está se tornando cada vez mais comum a execução de aplicações com várias tarefas para a resolução de um único problema. Em geral, este agregado de tarefas tem interações e dependências, necessitando do uso de algumas ferramentas computacionais (por exemplo, software e banco de dados) que estão compartilhados pelas organizações virtuais da grade.

Estas tarefas representam um fluxo de trabalho no qual dados são enviados/recebidos entre as tarefas obedecendo a certas regras. Desta forma, torna-se necessária a utilização de um mecanismo para coordenar, organizar e automatizar este fluxo de tarefas. Para este propósito, o conceito de *workflow* é empregado nesta abordagem, similar a alguns projetos de pesquisa em grade [22, 26, 10]. A proposta apresenta uma solução genérica e de granulosidade fina para definição de recursos da grade usados em cada estágio da execução de aplicação em um modo coordenado e

automatizado. O conceito clássico de *workflow* é apresentado em [7].

Além de processar e gerenciar os pedidos vindos dos dispositivos móveis para executar no ambiente de grade, este componente também coleta as informações relacionadas à execução de tarefas, realizando todas estas funcionalidades de modo transparente ao usuário móvel. Por fim, este componente também fornece automatização para usuário móvel, despachando todas as tarefas para o escalonador de tarefas da grade sem a necessidade de interação do usuário. Desta forma, ele permite mais agilidade na execução de tarefas que trabalham juntas para resolver o mesmo problema. Este componente possui três módulos: **Controlador**, **Motor** e **Coletor**. Uma descrição mais detalhada deste componente pode ser encontrada em [2].

Cada aplicação submetida pelos usuários móveis tem sua própria instância **Motor**, própria instância **Coletor** e um identificador único. A interface entre este componente e o *middleware* Globus [11] é realizada usando *Java CoG Kit*, este pacote de software contém APIs, como por exemplo, uma linguagem e API *workflow* Karajan [10] (usada para criar os scripts *workflows* das aplicações). Além disso, possibilita a criação rápida e fácil de ferramentas que interagem diretamente com serviços do Globus. O módulo **Controlador** é responsável por receber os pedidos que chegam dos dispositivos móveis e ordenar estes pedidos, comandando e controlando sua ordem através do Identificador. Quando este módulo recebe um pedido de submissão, ele cria uma instância do módulo **Motor** e direciona o pedido para esta instância, assim como mostrado na Figura 2.

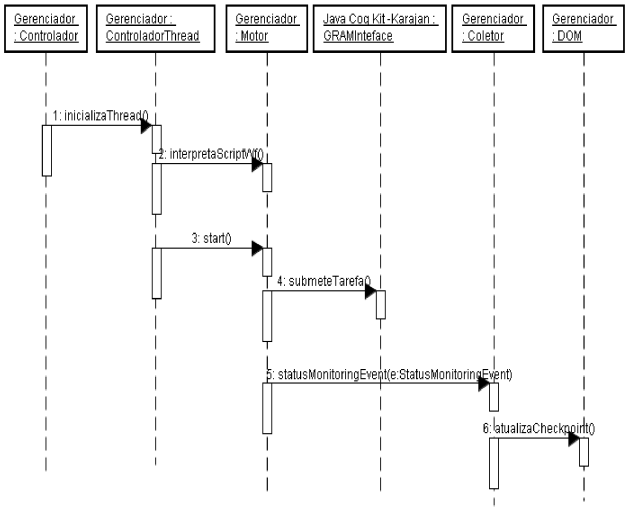

Figura 2. Diagrama de Seqüência do Gerenciador Workflow

O **Motor** é o principal módulo deste componente. Este módulo pode interpretar diferentes arquivos de definição *workflow* (scripts *workflows*) de diferentes aplicações submetidas para saber como deve submeter e controlar a execução de todas as tarefas da aplicação. Sendo assim, estes scripts definem a forma de controle do fluxo e dos dados e as ferramentas computacionais invocadas em cada tarefa. Cada aplicação tem seu próprio script *workflow* especificando seu fluxo de execução. Desta forma, este componente possibilita a execução de diferentes aplicações.

Na fase posterior, a submissão é efetuada através do serviço de grade *GRAM* [5], assim como mostrado na Figura 2. A utilização deste serviço padrão do Globus 4.0 possibilita lidar com mais facilidade com a submissão de tarefas nos recursos da grade. O escalonador Condor [24] é responsável por escalonar as tarefas nos recursos da grade. À medida que as tarefas são submetidas, seus status (estados de execução) são informados através de eventos gerados durante a execução. Desta forma, o **Motor** cria uma instância chamada **Coletor** para capturar estas informações e atualizá-las nos arquivos *checkpoint* (isto é, arquivo XML) que reportam o status que cada tarefa está atualmente.

3.2. Agente

Este componente busca adaptar o fluxo de execução para garantir a consistência das aplicações de maneira personalizada para o usuário em casos de desconexão. Para alcançar este propósito, o componente **Agente** observa o ambiente e especifica as adaptações a serem realizadas no fluxo de execução da aplicação submetida pelo dispositivo. Assim, a intenção é detectar a desconexão e ajustar a execução, evitando um estado de defeito.

Na Figura 3 são mostradas as interações entre os componentes. Inicialmente, o usuário submete a aplicação ao **Gerenciador Workflow** (passo 1) que coordena as tarefas da aplicação (por exemplo, uma aplicação com quatro tarefas) e controla o fluxo de execução considerando as definições do script *workflow* (passo 2a). Após, remete as tarefas para o Condor, usando-o como escalonador do *middleware* Globus (passo 2b). No próximo passo, o **Gerenciador** cria uma instância do **Agente** (passo 3), no qual verifica em um determinado intervalo de tempo (com base em vários testes foi definido em 20s) se o dispositivo que fez o pedido de submissão está conectado (passo 4 e 5). O **Agente** continua verificando a conexão até a finalização da execução da aplicação. Se o **Agente** não receber uma resposta do dispositivo neste intervalo de tempo, é caracterizada uma desconexão e o **Agente** adapta o fluxo de execução (passo 6). O módulo **Agente** possui três operações básicas: **Observador**, **Analisador** e **Adaptador**. A figura 4 apresenta o diagrama de atividades do **Agente** com as suas operações.

O **Observador** tem a função de verificar o status da co-

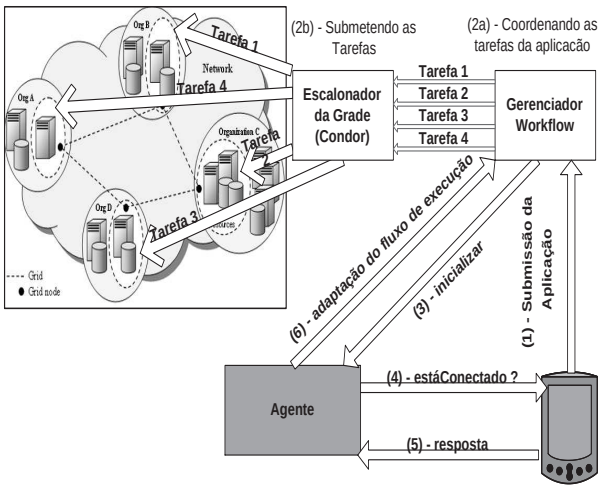

Figura 3. Interações dos Componentes da Arquitetura

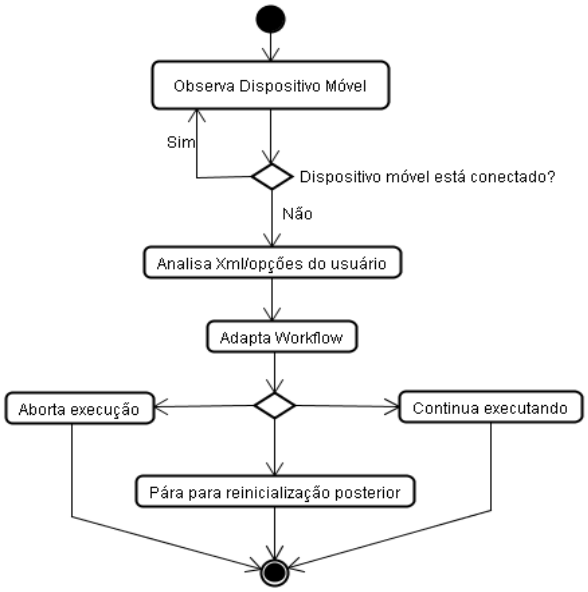

Figura 4. Diagrama de Atividades do Agente

nexão com o dispositivo. Tal tarefa é realizada por meio de uma thread que é criada para cada dispositivo conectado ao componente. A thread aguarda sucessivas mensagens de retorno do dispositivo. Quando a thread não recebe retorno dentro do tempo estipulado, considera-se que o dispositivo está desconectado e então é ativado o **Analisador**. Por exemplo, se alguma tarefa da aplicação necessita da entrada de dados do usuário móvel, o **Gerenciador Workflow** pausa a execução e aguarda os dados no intervalo de tempo definido, se os dados não chegam durante este intervalo, uma exceção é gerada e o **Analisador** é ativado para examinar os requisitos da aplicação (dependências) e decidir como proceder nesta situação.

Outro interessante aspecto da presente proposta refere-se ao fato que o usuário pode definir opções antes da submissão da aplicação. As opções definidas podem ser: parar a execução da aplicação em determinada tarefa até que o dispositivo reconecte, continuar ou abortar a execução da aplicação, em caso de desconexão. A opção de continuar a execução somente é considerada quando uma desconexão ocorre e a aplicação submetida não tem qualquer dependência do usuário móvel. Ou seja, um usuário pode decidir parar a aplicação até a conexão retornar, possibilitando a monitoração dos estados da aplicação. No entanto, se a aplicação tem dependências, não é permitido ao usuário definir que a aplicação continue executando. Sendo assim, o **Adaptador** obrigatoriamente pára ou aborta a execução conforme a escolha do usuário. Esta característica fornece facilidades para a abordagem tratar a desconexão e, se necessário, adaptar o fluxo de execução considerando a natureza da aplicação e também a especificação do usuário.

Após a definição do **Analisador**, o **Adaptador** é responsável pelas modificações, quando necessárias. Há três possíveis situações para o **Adaptador** intervir no fluxo de execução: abortar a execução, parar a execução para reinício posterior ou permitir que a aplicação continue executando. Nestes casos, o **Adaptador** comunica com o **Motor** para proceder às adequações. Se o **Analisador** identificar que a aplicação pode continuar a execução, apenas são armazenados os resultados do processamento para futura entrega ao usuário. Quando é detectada uma desconexão e o usuário não tiver definido nenhuma das opções listadas acima, por *default* (ou padrão) foi estabelecido que a execução da aplicação será paralisada na tarefa que está executando naquele momento e será iniciada novamente a partir deste ponto quando o dispositivo voltar a conectar.

Quando o **Adaptador** definir que a aplicação deve ser parada para possibilitar a reinicialização posterior, o **Gerenciador Workflow** armazenará os estados da execução em um arquivo de *checkpoint*. Assim, quando um dispositivo voltar a conectar ao **Gerenciador**, será verificada se alguma tarefa da aplicação submetida não foi completada. Neste caso, o usuário será notificado da existência de uma aplicação submetida que não completou a execução. Caso o usuário confirme a reinicialização da aplicação, será recuperado o estado das tarefas e reiniciada a execução no ponto que havia parado.

3.3. GUI Movel

Atualmente, é possível observar que a tecnologia de grade móvel tem introduzido mudanças na forma de intera-

ção entre as pessoas e os recursos distribuídos em redes de computadores. Tais mudanças ocorrem em função da maior disponibilidade das redes de comunicação, com banda larga e latência reduzida, juntamente com a tecnologia de computação móvel. Assim, há maior visibilidade para comunicação e compartilhamento de recursos para os mais variados usuários. Desta forma, torna-se necessário o desenvolvimento de interfaces de acesso único e facilitado para usuários móveis que permitam utilizar esta tecnologia da melhor forma possível, levando em consideração as restrições destes aparelhos. Por esta razão, a **GUI Móvel** foi projetada e desenvolvida, especificadamente para PDAs, fornecendo algumas novas funcionalidades que permitem interfaces de acesso único à grade. Importante destacar que algumas destas interfaces são adaptadas conforme as restrições desses dispositivos. Desafios de adaptações são argumentados nesta seção.

Na fase de implementação, foi decidido empregar ferramentas e protocolos de comunicação para fornecer portabilidade a um diversificado número de dispositivos. Desta forma, a GUI foi implementada usando J2ME (*Java 2 Micro Edition*) *Wireless Toolkit* [25] que oferece suporte em grande parte dos dispositivos móveis. A submissão, monitoração e visualização de resultados finais da execução da aplicação são as principais funcionalidades deste componente. Além disso, ele foi projetado para fornecer uma interface com mais funcionalidades e mais informações detalhadas para os usuários dos dispositivos móveis do que os trabalhos mostrados na seção anterior. As novas funcionalidades são:

- Acessar a descrição do *workflow* de cada aplicação, detalhando os recursos (por exemplo, banco de dados e softwares) que são invocadas pelas tarefas e o papel de cada tarefa durante o fluxo de execução. Essa funcionalidade da GUI auxilia o usuário a escolher a aplicação que pode resolver seu problema dentre alguns já existentes no GUI. As descrições dos *workflows* das aplicações estão estruturadas em arquivos XML;

- Monitorar as aplicações, acompanhando o progresso da execução através do fornecimento de informações de status de cada tarefa da aplicação (por exemplo, executando, falhou e completou);

- Visualizar: **(1)** Os resultados parciais e finais em um modo otimizado, ou seja, somente partes dos arquivos de resultado que são considerados relevantes para o usuário são carregadas na interface do dispositivo. **(2)** Os problemas ocorridos durante a execução da aplicação. Alguns exemplos de problemas mostrados são: erro no formato de arquivos de entrada e programas não devidamente configurados. **(3)** O tempo de execução de cada tarefa da aplicação.

A monitoração da aplicação pode ser uma funcionalidade interessante para usuários móveis de grades computacionais e aglomerados de computadores (*cluster*), assim como observado em [19]. Uma vez que estes dispositivos permitem a apresentação de como está avançando a execução da aplicação em recursos grade de qualquer lugar e em qualquer hora. Vale a pena ressaltar também que algumas execuções de aplicações podem levar horas (ou mesmo dias) para concluir [22]. Sendo assim, torna-se necessário fornecer alternativas para usuários terem conhecimento do progresso de sua execução de diferentes localidades.

Esta abordagem fornece uma interface de monitoração conforme as restrições de tela destes dispositivos. Sendo assim, a monitoração é iniciada sem necessidade de qualquer interação (ou intervenção) do usuário com a interface da **GUI Móvel**. A monitoração foi desenvolvida para fornecer uma capacidade melhorada para acompanhar a execução de cada tarefa da aplicação. Este aspecto pode ser possível usando informações atualizadas dos status e interfaces adaptadas aos PDAs. A Figura 5 ilustra a interface de monitoração.

Figura 5. Interface de Monitoração

A interface do PDA foi implementada para representar o *workflow* da aplicação graficamente em uma estrutura única, fornecendo uma visualização de modo organizado dos status das tarefas na interface. Além disso, permite acompanhar simultaneamente a execução das várias tarefas da aplicação. Entre as opções mostradas em [26], o formalismo Não-DAG de representação *workflow* foi o mais ideal para esta abordagem, por ser flexível, permitindo representar todas as estruturas de definição (tais como, seqüencial, laços e paralelismo) também contidas em outros formalismos. Desta forma, a interface do PDA pode representar *workflows* com estruturas complexas e diferentes formas de relacionamentos. Em adição, ele também apresenta uma representação gráfica que possibilitou economizar espaços na tela do dispositivo. Em outras palavras, utilizando o mínimo de elementos gráficos nos quais círculos (ou nodos) indicam tarefas e arcos indicam transições ou dependências entre estas tarefas.

4. Resultados Experimentais

A configuração escolhida para os testes experimentais foi um cenário real de grade móvel com uma WLAN (*Wireless LAN* com ponto de acesso) e recursos de grade em uma rede estruturada. Este ambiente é projetado para alcançar uma avaliação real da execução da aplicação e dos casos relacionados. As características do PDA são: Palm Tungsten C executando o sistema operacional Palm 5.2.1 com processador 400 MHz, 64 MB RAM, Built-in Wi-Fi (802.11b).

A abordagem presente permite a submissão e monitoração de aplicações através de um único pedido de submissão. Em outras palavras, o **Gerenciador Workflow** controla o fluxo de execução e invoca as ferramentas necessárias sem necessitar de interação dos usuários para executar todos os passos. Este é o aspecto diferencial quando comparado à abordagem **Outro** apresentada em [17, 23, 6, 21, 1, 16], na qual cada tarefa da aplicação é submetida uma de cada vez através da interface do usuário. Desta forma, nestes outros trabalhos de pesquisa, o usuário controla a ordem de submissão de cada tarefa que trabalha junto cooperativamente para resolver o mesmo problema.

É bem conhecido que a redução no acesso à rede sem fio pode fornecer uma menor dissipação de energia de bateria destes dispositivos [14], isto é, aumentando seu tempo de vida. Estender o tempo de vida da bateria é um dos problemas mais críticos e desafiadores nestes dispositivos [20]. Desta forma, uma abordagem que fornece maior automatização, como acontece nesta arquitetura, envia menos pedidos de submissão pode reduzir o consumo de energia destes dispositivos.

No gráfico da Figura 6 é apresentada uma comparação da média de consumo de bateria de um PDA para submeter e monitorar tarefas para resolução de um problema. A comparação considera o uso da **Arquitetura** em contraste com a abordagem **Outro** implementada em [17, 23, 6, 21, 1, 16]. A aplicação considerada é um seqüenciamento de DNA que requer a execução de 9 tarefas que devem ser executadas em uma forma organizada e controlada para alcançar sua resolução. A implementação foi obtida de [12]. O resultado mostra que a **Arquitetura** alcançou uma economia média de 28% quando comparada à abordagem **Outro**. Ela permite um número maior de execuções, isto é, em média mais de 12 execuções de aplicações do que **Outro**, como pode ser visto no gráfico seguinte.

Foram executados 24 experimentos em cada abordagem, cada experimento representa a porcentagem do consumo de bateria de 5 execuções de aplicações do problema de seqüenciamento (ou seja, um total de 120 submissões da aplicação em cada abordagem). Em cada execução de aplicação, foi acumulada a porcentagem de consumo para submeter e monitorar cada tarefa da aplicação na abordagem **Outro** e foi verificada a porcentagem de consumo para submeter e monitorar uma aplicação na **Arquitetura**. Além disso, em cada experimento, os recursos da grade foram exclusivamente dedicados para os testes e o intervalo de tempo para envio de notificações foram iguais em ambas as abordagens. Desta forma, estas variáveis não influenciaram o resultado.

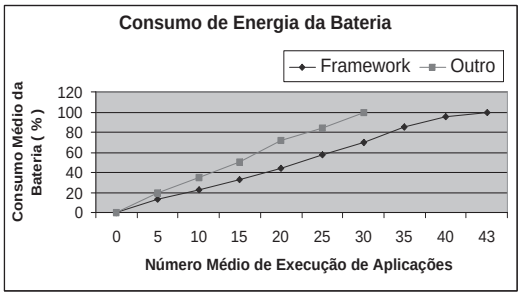

Figura 6. Consumo Médio da Bateria

A média e o intervalo de confiança de consumo de bateria para cada experimento foram respectivamente 11% e ***(+/-)*** 1,6603% na **Arquitetura**, enquanto na abordagem **Outro** foi 16,17% e ***(+/-)*** 1,6323%. Um ***teste T de 2 amostras*** [15] foi usado para comparar se existe uma diferença significativa entre a **Arquitetura** da presente proposta e a abordagem **Outro**. Em um nível de significância de 95%, o **teste T** evidenciou uma diferença significativa entre as duas amostras, devido a variável ***P-value*** ter como resultado o valor 0,0002, como [15] observa, quando ***P-value*** $\leq$ 0,05, ele indica uma diferença significativa entre as duas amostras. Em outras palavras, a **Arquitetura** economiza significativamente mais energia de bateria do que a abordagem **Outro** para submeter e monitorar aplicação no PDA.

5. Conclusões e Trabalhos Futuros

Neste artigo, foi apresentada uma abordagem que permite submissão e monitoração de aplicações em ambientes de grades móveis. A proposta foi projetada e implementada considerando três componentes, o **Gerenciador Workflow**, o **Agente** e a **GUI Móvel**, que levam em consideração também as características não-funcionais, como consumo de energia e confiabilidade. Estes componentes permitiram a resolução de problemas complexos usando o conceito de *workflow*.

O conceito adotado forneceu ao usuário móvel uma forma mais automatizada e coordenada para executar aplicações na grade a partir dos dispositivos móveis. Além disso, a proposta possibilitou que as tarefas da aplicação trabalhassem em um estilo colaborativo para resolver um único problema e combinando redução de consumo de energia com características de alto desempenho e processa-

mento em sistemas paralelos e heterogêneos, característicos dos ambientes de grade.

A abordagem desenvolvida forneceu uma facilidade personalizada com característica adaptativa conforme o problema de desconexão que existe nos dispositivos móveis. Ou seja, ele fornece uma facilidade de ajuste para a execução do fluxo das tarefas, considerando os requisitos da aplicação submetida e as opções definidas pelo usuário. Desta forma, esta funcionalidade garante a consistência do fluxo de uma maneira personalizada para o usuário móvel. Em adição, os resultados experimentais indicaram que a abordagem proposta permitiu consumir menos energia da bateria de um PDA para submeter e monitorar aplicações.

Como próximos passos da pesquisa apresentada neste artigo será elaborado um algoritmo que forneça um intervalo de tempo para verificação do estado de desconexão e para monitorar a aplicação em um estilo mais dinâmico. Este mecanismo considerará parâmetros de entrada, por exemplo, tempo de vida da bateria e/ou volume de tráfego na rede sem fio.

Referências

[1] G. Andronico, R. Barbera, A. Falzone, G. L. Re, A. Pulvirenti, and A. Rodolico. The genius web portal: grid computing made easy. *(ITCC'03) - IEEE Computer Society*, pages 425–431, April 2003.

[2] V. C. M. Borges and M. A. R. Dantas. Uma abordagem de submissão e monitoração de múltiplas tarefas para ambientes de grade computacional utilizando dispositivos móveis. *XXXIII SEMISH*, pages 403–418, 2006.

[3] J. Brooke and M. Parkin. A pda client for the computational grid. In *WETICE '05*, pages 325–330, Washington, DC, USA, 2005. IEEE Computer Society.

[4] D. Bruneo, M. Scarpa, A. Zaia, and A. Puliafito. Communication paradigms for mobile grid users. *3rd (CCGrid'03)*, pages 669–676, May 2003.

[5] I. Foster. Globus toolkit version 4: Software for service-oriented systems. *IFIP NPC'05*, pages 2–13, 2005.

[6] P. Grabowski, K. Kurowski, J. Nabrzyski, and M. Russell. Context sensitive mobile access to grid environments and vo workspaces. *MDM - IEEE Computer Society*, 0:87, 2006.

[7] D. Hollingsworth. Workflow management coalition. reference model and api specification. *WfMC-TC00-1003*, 1996.

[8] K. A. Hummel, G. Bohs, P. Brezany, and I. Janciak. Mobility extensions for knowledge discovery workflows in data mining grids. *DEXA - IEEE Comp. Society*, 0:246–250, 2006.

[9] J. Hwang and P. Aravamudham. Middleware services for p2p computing in wireless grid networks. *IEEE Internet Computing*, 8(4):40–46, 2004.

[10] G. Laszewski and M. Hategan. Workflow concepts of the java cog kit. *Journal of Grid Computing*, 3(3-4):239–259, 2005.

[11] G. V. Laszewski, I. Foster, J. Gawor, and P. Lane. A java commodity grid kit. *Concurrency and Computation: Practice and Experience*, 13(8-9):643–662, 2001.

[12] M. Lemos. Workflow para bioinformática. Master's thesis, Pontifícia Universidade Católica do Rio de Janeiro - Brazil (PUC-Rio), 2004.

[13] L. W. McKnight, J. Howison, and S. Bradner. Guest editors' introduction: Wireless grids–distributed resource sharing by mobile, nomadic, and fixed devices. *IEEE Internet Computing*, 8(4):24–31, 2004.

[14] S. Mohapatra, R. Cornea, H. Oh, K. Lee, M. Kim, N. D. Dutt, R. Gupta, A. Nicolau, S. K. Shukla, and N. Venkatasubramanian. A cross-layer approach for power-performance optimization in distributed mobile systems. In *IPDPS*, 2005.

[15] D. Montgomery. *Design and Analysis of Experiments*. John Wiley and Sons Ltd, 2005.

[16] J. Novotny, M. Russell, and O. Wehrens. Gridsphere: An advanced portal framework. *euromicro - IEEE Computer Society*, 00:412–419, 2004.

[17] S. M. Park, Y.-B. Ko, and J.-H. Kim. Disconnected operation service in mobile grid computing. *(ICSOC'03). LNCS 2910 Springer*, pages 499–513, December 2003.

[18] T. Pham, L. Huang, and C. Dulan. Challenge: Integrating mobile wireless devices into the computational grid. *8th (MOBICOM'02)*, pages 271–278, Setembro 2002.

[19] C. Rista and M. A. R. Dantas. A wireless monitoring approach for a ha-oscar cluster environment. *HPCS - IEEE Computer Society*, pages 302–306, 2005.

[20] P. Rong and M. Pedram. Extending the lifetime of a network of battery-powered mobile devices by remote processing: a markovian decision-based approach. In *proceedings of DAC '03*, pages 906–911. ACM Press, 2003.

[21] A. Sajjad, H. Jameel, U. Kalim, S. M. Han, Y.-K. Lee, and S. Lee. Automagi - an autonomic middleware for enabling mobile access to grid infrastructure. *ICAS-ICNS - IEEE Computer Society*, 0:73, 2005.

[22] J. Schneider, B. Linnert, and L.-O. Burchard. Distributed workflow management for large-scale grid environments. *SAINT - IEEE Computer Society*, 0:229–235, 2006.

[23] W. Shi, S. Li, and X. Lin. Towards merging pervasive computing into grid - lightweight portal, dynamic collaborating and semantic supporting. *IMSCCS - IEEE Computer Society*, 1:560–563, 2006.

[24] D. Thain, T. Tannenbaum, and M. Livny. *Condor and the Grid*. John Wiley and Sons, NJ, USA, 2003.

[25] S. J. W. Toolkit. *Java 2 Platform, Micro Edition (J2ME) Wireless Toolkit*. Web Page [Online]. Disponível em: http://java.sun.com/products/sjwtoolkit/, 2007.

[26] J. Yu and R. Buyya. A taxonomy of workflow management systems for grid computing. *(SIGMOD'05)*, 34(3):44–49, July 2005.