An Architecture for Handling Disconnections in Workflow Applications in Mobile Grid Environments

V.C.M. Borges, A.G.M. Rossetto e M.A.R. Dantas

***Abstract*— This article presents an approach for providing automated and coordinated features for applications execution in mobile grid configurations. Therefore, in this article is proposed and implemented an architecture that provides an execution flow adjustment, in disconnection cases, matching requirements of submitted application and options defined by the user of the mobile device. Experimental results show also that the approach consumed less battery energy of a PDA for submitting and monitoring applications in comparison with some related works (about 43%). In addition, it provided interfaces with adapted and optimized characteristics according to some limitations and problems found in different mobile devices.**

***Keywords*— workflow application, disconnection, PDA, cellular, mobile grid and battery energy consumption.**

I. Introdução

A resolução de problemas complexos vem se tornando[1] possível devido a grande evolução observada em termos de hardware e software. Para tal fim, tem-se utilizado 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 [2, 12], esta infra-estrutura integra dispositivos móveis e o ambiente de grade computacional, possibilitando fornecer aos dispositivos móveis os recursos compartilhados pela grade.

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, processamento, microfones, câmeras, receptores GPS e sensores) [2, 6, 12]. O poder computacional desses dispositivos tem apresentado melhorias nos últimos anos. Porém, a atual capacidade de processamento e armazenamento ainda não possibilita a resolução de problemas complexos nestes dispositivos. Neste trabalho de pesquisa os dispositivos móveis são considerados como usuários de recursos da grade.

A maioria das pesquisas em grade móvel somente permite características de submissão e monitoração de uma tarefa por vez em um dispositivo específico. Nessa abordagem 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 da rede *wireless*. Desta forma, torna-se interessante ter uma abordagem onde os dispositivos móveis, dentro do seu contexto, poderiam proporcionar uma facilidade coordenada, automatizada e transparente (ou seja, alto nível de abstração) relacionada ao fluxo de execução de tarefas em configurações de grade para resolver um único problema.

Outro aspecto importante considerado neste estudo é a capacidade de energia da bateria dos dispositivos móveis, que vêm tendo poucos avanços no sentido de minimizar o seu consumo, causando restrições no emprego e utilização destes dispositivos. Por exemplo, a capacidade de processamento destes dispositivos para executar alguma aplicação não poderá mais ser obtida por aumentos ilimitados da capacidade de seus processadores, em função do impacto deste aumento sobre a energia consumida da bateria. Diante destes aspectos, tornam-se necessárias alternativas 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.

Considerando sua natureza móvel e de recursos limitados, os dispositivos móveis 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 durante a execução.

Consciente que estes aspectos mencionados anteriormente não são abordados de forma completa pelos trabalhos relacionados, este artigo apresenta um trabalho de pesquisa que proporciona uma abordagem mais completa, onde dispositivos móveis submetem aplicações ao ambiente de grade, baseado no conceito de *workflow*. Neste contexto, uma arquitetura real foi

Manuscrito recebido em 28 de fevereiro de 2009.

V.C.M. Borges, A.G.M. Rossetto e M.A.R. Dantas são membros do Laboratório de Pesquisa em Sistemas Distribuídos (LaPeSD) no Departamento de Informática e Estatística (INE) da Universidade Federal de Santa Catarina (UFSC). E-mail: {vcunha, anubis, mario @inf.ufsc.br}.

desenvolvida para executar a submissão e monitoração de aplicações em uma forma coordenada e automatizada. O ambiente tem como contribuição adicional 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 e diversificado tamanho de tela). O artigo é organizado como segue: na seção II, são apresentadas as características verificadas nos trabalhos relacionados; os aspectos interessantes do projeto e da implementação da arquitetura são mostradas na seção III; na seção IV é apresentada uma comparação entre esta abordagem e a abordagem apresentada na maioria dos trabalhos relacionados; finalmente, na seção V algumas conclusões e trabalhos futuros são apontados.

II. Trabalhos Relacionados

As abordagens apresentadas em [1, 5], 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. Outros aspectos que podem ser observados nestes trabalhos de pesquisa são: a) não mencionam qualquer adaptação de funcionalidade da interface de submissão e monitoração da aplicação para o contexto dos dispositivos móveis, e b) as abordagens foram desenvolvidas especificadamente em PDAs, não levando em consideração os dispositivos celulares como terminais de acesso à grade. O principal foco em [1] é a implementação de segurança no cliente PDA, visto que a submissão de tarefas no *middleware* UNICORE requer obrigatoriamente uma comunicação segura entre clientes e este *middleware* grade. Em [5], o principal foco é a descoberta de *workflows* de *data mining* para submissão a partir de PDAs.

Em [15], os autores, particularmente, focam no problema de desconexão para tratá-lo do ponto de vista que o dispositivo móvel trabalha como recurso da grade, por exemplo, para executar tarefas no próprio dispositivo. Tal abordagem propõe um novo algoritmo de escalonamento baseado na instabilidade do ambiente móvel para prever uma provável desconexão e consequentemente, migrar a aplicação executada para outro dispositivo antes de ocorrer a desconexão. Entretanto, esta forma de interação de grade móvel não permite 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, sequenciamento de DNA em bioinformática [15] e análise de dados de uma galáxia em astro-física [20]).

Em [8,9], os autores apresentam abordagens que lidam com desconexão em clientes móveis de grade. O principal foco de [8,9] é *checkpointing* para tolerância a falhas eficiente em sistemas de grade móvel. Entretanto, estas abordagens não possibilitam submissão de aplicações baseada em *workflow* e, também, não fornecem interação para diferentes dispositivos.

Como pode ser visto na Tabela I, a maior parte dos trabalhos relacionados [1, 5, 8, 9, 15] não argumentam ou mostram soluções para as limitações destes dispositivos de forma mais completa, como por exemplo: pequeno e diversificado tamanho de tela (isto é, necessidade de interfaces adaptadas a diferentes dispositivos), 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 tratados na presente proposta.

Tabela I – Comparação entre os Trabalhos Relacionados

Trabalhos Relacionados	Consumo de Energia	Desconexão	Workflow	Interfaces Adaptadas
Park et al. 2003 [15]	Não	Sim	Não	Sim
Brooke and Parkim 2005 [1]	Não	Não	Sim	Não
Hummel et al. 2006 [5]	Não	Não	Sim	Não
Paul et al. 2007 [8]	Não	Sim	Não	Não
Ihram et al. 2007 [9]	Não	Sim	Não	Não

III. Abordagem Proposta

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

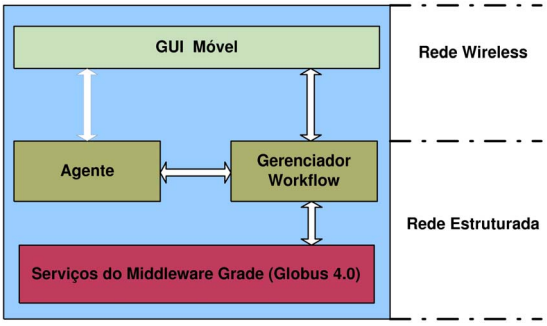

Figura 1. Arquitetura Proposta

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

A. 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, onde dados são enviados/recebidos entre as tarefas obedecendo a certas regras para resolver o mesmo problema. Desta forma, torna-se necessário a utilização de um mecanismo para coordenar, organizar e automatizar este fluxo de tarefas. Para este propósito, o conceito de *workflow* é empregado nesta arquitetura, similar a alguns projetos de pesquisa em grade [7, 17, 19]. Este conceito apresenta uma solução genérica e de granulosidade fina para definição de recursos da grade usados em cada estágio da execução da aplicação em um modo coordenado e automatizado. O conceito clássico de *workflow* é apresentado em [4].

Além de processar e gerenciar os pedidos vindos dos dispositivos móveis para executar no ambiente de grade, este componente também coleta as informações relacionados a execução de tarefas, realizando todas estas funcionalidades de modo transparente ao usuário móvel. Por fim, este componente também fornece automatização para o usuário móvel, despachando todas as tarefas para o escalonador de tarefas da grade sem a necessidade de interação do usuário. Como pode ser visto na Fig. 2, este componente possui três módulos: **Controlador**, **Motor** e **Coletor**

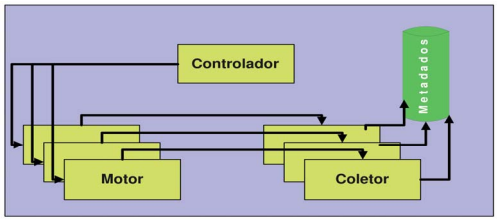

Figura. 2. Módulos do Gerenciador Workflow

Cada aplicação submetida pelos usuários móveis tem suas próprias instâncias **Motor** e **Coletor, além de** um identificador único. O pacote *Java CoG Kit* [10] foi utilizado para interação deste componente com o *middleware* grade Globus 4.0 [3]. 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.

O **Motor** é o principal módulo deste componente. Este módulo pode interpretar diferentes arquivos de definição *workflow* (scripts *workflows*) para saber como deve submeter e controlar a execução de todas as tarefas da aplicação. 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. Estes arquivos de definição ou *scripts* são descritos na linguagem *workflow Karajan* [7] do pacote *Java CoG Kit* e estão armazenados no repositório de metadados.

Na fase posterior, a submissão é efetuada através do serviço de grade *GRAM* [3]. 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 [18] é responsável por escalonar as tarefas nos recursos da grade. À medida que as tarefas são submetidas, seus status são informados através de eventos gerados durante a execução. Assim, o **Motor** cria uma instância chamada **Coletor** para capturar estas informações e atualiza-las nos arquivos *checkpoint* (arquivo XML) que reportam o status de cada tarefa.

B. 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. A intenção é detectar a desconexão e ajustar a execução, evitando um estado de defeito.

O Observador tem a função de detectar a desconexão e notificar o Gerenciador Workflow. O observador é instanciado pelo Gerenciador Workflow, quando este recebe um pedido de submissão. A partir disso, o observador envia uma mensagem inicial para o dispositivo solicitando que este envie mensagens de notificação dentro de um determinado intervalo de tempo a fim de notificar que está conectado. A partir deste momento o observador fica monitorando a chegada das mensagens no intervalo de tempo. Quando o observador não receber retorno dentro do tempo estipulado, considera-se que o dispositivo está desconectado e então é ativado o Analisador.

O Analisador examina (a) os requisitos da aplicação (dependências) e (b) as opções pré-definida pelo usuário para decidir como proceder na situação de falha.

a) Requisitos da aplicação (Dependências): é considerada dependência qualquer interação que dependa do dispositivo móvel, por exemplo, quando um workflow possuir definições de entrada de dados do usuário móvel, referências a arquivos armazenados no dispositivo, referência para gravar dados no dispositivo.

b) Opções pré-definidas: o usuário do dispositivo móvel pode configurar opções para situações de falha. Estas opções são enviadas juntamente com o pedido de submissão. Na GUI Móvel estas opções estão armazenadas em arquivo xml e o usuário pode alterá-las pela interface da GUI Móvel. O usuário pode optar por uma das seguintes opções para casos de falha: parar a execução, continuar a execução ou parar para reinício posterior. Assim, mesmo que a aplicação possua características que permitam continuar a execução, o usuário poderá determinar que esta seja interrompida. Este mecanismo permite que os usuários tomem decisões a respeito das suas aplicações considerando suas necessidades, por exemplo, usuários que desejam acompanhar

todo o processo de execução, monitorando cada tarefa.

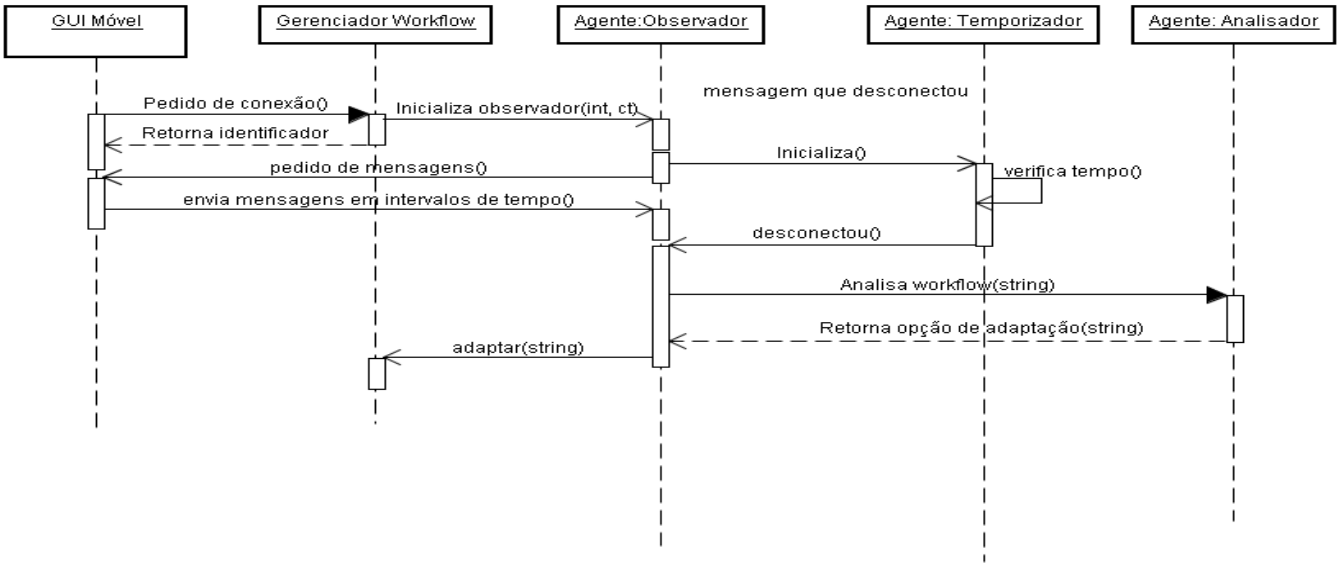

Figura. 3. Diagrama de Seqüência do módulo Agente

O Adaptador é responsável pelas modificações, quando necessárias. Há duas possíveis situações para o Adaptador intervir no fluxo de execução: a) abortar a execução, e b) parar a execução para reinício posterior. Nestes casos, o Adaptador comunica com o módulo Motor do Gerenciador Workflow para proceder as adequações. Vale ressaltar que o Gerenciador Workflow armazena os estados da execução de cada tarefa em um arquivo de checkpoint. Os estados são armazenados a cada alteração gerada pelos eventos do Karajan.

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. Desta forma, 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 sua execução. Caso o usuário confirme a reinicialização da aplicação será recuperado o estado das tarefas e reiniciada a execução no ponto que havia parado.

O diagrama de seqüência da Fig. 3 apresenta as interações do Agente com o Gerenciador Workflow e a GUI Móvel para um pedido de submissão. Inicialmente, o usuário móvel faz um pedido de conexão ao Gerenciador Workflow que responde com o identificador gerado. Após o usuário pode realizar o pedido de submissão de uma aplicação ao Gerenciador que coordena as tarefas da aplicação e controla o fluxo de execução e remete as tarefas para o Condor, usado como escalonador do middleware Globus. No próximo passo, o Gerenciador cria uma instância do Observador, este envia uma mensagem à GUI Móvel para que esta comece a enviar mensagens dentro de um intervalo de tempo, indicando que está conectado. O intervalo de tempo foi definido em 20s tomando como base o tempo médio de execução das tarefas do workflow (isto é, média de 50 amostras) utilizado no presente de trabalho de pesquisa, visto que o intervalo de tempo dependerá do tempo de execução do workflow e das condições da rede sem fio.

Para controlar o tempo de recebimento das mensagens o Observador instancia um Temporizador que tem um contador de tempo. Quando o Observador recebe uma mensagem, o contador é reiniciado. O observador continua verificando a conexão até a finalização da execução da aplicação. Se o Observador não receber uma resposta do dispositivo no intervalo de tempo estipulado, é caracterizada uma desconexão. Neste caso, o Observador faz um pedido de análise do workflow para o Analisador que retorna a opção de adaptação. Por fim, o observador envia uma mensagem com a opção ao método adaptar (Adaptador) do Gerenciador que realiza as adequações necessárias.

C. GUI Móvel

A **GUI Móvel** foi projetada e desenvolvida especificadamente para PDAs e telefones celulares, fornecendo algumas funcionalidades que permitem interfaces flexíveis e ajustáveis para facilitar o acesso único à grade. Importante destacar que algumas destas interfaces são adaptadas conforme as restrições de diferentes dispositivos. Desafios de adaptações são argumentados nesta seção.

Por esta razão, na fase de implementação, foi decidido empregar ferramentas e protocolos de comunicação para fornecer portabilidade a um diversificado número de dispositivos. A GUI foi implementada usando J2ME (*Java 2 Micro Edition*) *Wireless Toolkit* e protocolo de comunicação HTTP, ambos possuem suporte em grande parte dos dispositivos móveis. Exemplos de interfaces são: submissão de aplicações, visualização de resultados finais e parciais de uma aplicação 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 e, por fim, monitorar a execução da aplicação,acompanhando o progresso da execução através do status de cada tarefa.

A abordagem fornece interfaces com as mesmas funcionalidades para PDAs e celulares. Sendo assim, ela permite que o usuário tenha flexibilidade para escolher qual

dispositivo usar, dependendo das diferentes necessidades de cada usuário. Por exemplo, usuários podem utilizar vantagens oferecidas por ambos PDA (menos limitação de recursos do que celular e não possui custos financeiros para transmissão de dados) ou celular (baixo custo de aquisição e um maior raio de cobertura).

A monitoração da aplicação pode ser uma funcionalidade interessante para usuários móveis de grades computacionais e aglomerados de computadores (*cluster*). Uma vez que estes dispositivos permitem a visualização de como está o avanço da 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 [17]. Por esta razão, torna-se necessário fornecer alternativas para usuários ter conhecimento do progresso de sua execução de diferentes localidades.

Este componente fornece uma interface de monitoração conforme as restrições de tela destes dispositivos. A monitoração é iniciada sem necessidade de qualquer interação (ou intervenção) do usuário com a interface da **GUI Móvel**. A monitoração foi desenvolvida a fim de 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. O lado esquerdo da Fig. 4 ilustra a interface desta monitoração.

A interface do PDA foi implementada para representar o *workflow* da aplicação graficamente em uma estrutura única e fornecendo uma visualização de modo organizado do status das tarefas na interface. Além disso, permite acompanhar simultaneamente a execução de várias tarefas da aplicação. Entre as opções mostradas em [19], o formalismo Não-DAG, (em inglês, *Non-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.

A interface do PDA pode representar *workflows* com estruturas complexas e diferentes formas de relacionamentos. Ele também apresenta uma representação gráfica que possibilitou economizar espaços na tela do dispositivo. Devido ao fato de utilizar um mínimo conjunto de elementos gráficos, onde círculos (ou nodos) indicam tarefas e arcos indicam transições ou dependências entre estas tarefas.

Figura. 4. Interface de Monitoração da Aplicação.

A interface do celular tem as mesmas funcionalidades contidas no PDA. Entretanto, a interface de monitoramento é apresentada de forma diferente no telefone celular. Isso se deve ao fato que a KVM (*Kilobyte Virtual Machine*) de alguns telefones celulares não terem fornecido suporte a determinadas classes e métodos do J2ME necessários para desenhar a estrutura do *workflow* na interface do PDA (incluindo o celular usado nesta abordagem). Desta forma, foi necessário apresentar o status das tarefas em um outro modo. Por esta razão, a interface de monitoramento do *workflow* apresenta o status das tarefas em um formulário com campos texto para cada tarefa no celular. Como mostrado no lado direito da Fig. 4.

Embora a interface de monitoração da aplicação permita a apresentação de uma estrutura única e adaptada em seu contexto para monitorar as tarefas da aplicação simultaneamente, ela não possui um padrão de cores para indicar o status. Ao invés do padrão de cores, notificações de mensagens na tela e emissões de sons foram usadas para reportar os status das tarefas no celular.

Com o objetivo de testar e validar a arquitetura proposta foi utilizada uma aplicação que trata o problema de sequenciamento de DNA de uma bactéria [11]. Esta aplicação produz arquivos textos com tamanho variando de poucos bytes até a 10 kilobytes. Entretanto, foi detectado que o tempo de transferência de arquivos no telefone celular utilizou uma quantidade de tempo consideravelmente maior do que o tempo verificado no PDA.

Uma vez que a taxa de transmissão do celular empregado é bem menor do que nos PDAs, isto é, no celular usado tem uma taxa de 32-48 kpbs (o celular possui a seguinte configuração: Modelo Motorola C385 com 3 MB de memória RAM) e o PDA tem uma taxa de 11Mbps. Além disso, o celular utiliza a tecnologia GPRS (*General Packet Radio Service*) para transmitir dados na rede; esta tecnologia estabelece um custo financeiro por volume de dados transmitidos.

Desta forma, foi decidido empregar um método de compressão de arquivos texto GZIP em 3 tipos de tamanho de arquivo:Maior (6 KB a 10 KB), Médio (3 KB a 6 KB) e Menor (512 bytes a 3 Kylobytes). Com isso, foi possível reduzir o custo financeiro e o tempo de transferência de dados para o celular. Foram realizados testes em arquivos com tamanho de 512 bytes, 5 KB e 10 Kb para representar respectivamente os 3 tipos de tamanho (Grande, Médio e Pequeno).

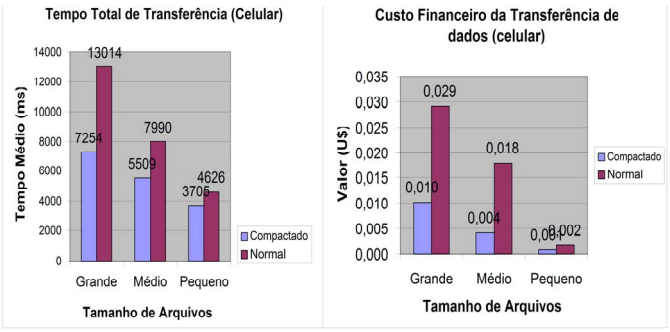

Figura. 5. Compressão de Dados no Celular.

O tempo total de transferências e o custo dos arquivos recebidos no celular em um total de 10 experimentos (como mostrado nos gráficos da Fig. 5) apresentaram em média um menor valor aplicando a compressão, isto é, 40% (Grande),

31% (Médio) e 20% (Pequeno) em tempo de transferência. Já em custos financeiros obteve-se 65% para Grande, 77% para Médio e 34% para Pequeno. Importante destacar que o tempo total de transferência dos arquivos compactados, mostrado no gráfico da Fig. 5, contabiliza o tempo para comprimir, transferir em redes sem fio e estruturada e descomprimir no telefone celular.

IV. 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. Uma vez que o componente **Gerenciador Workflow** controla o fluxo de execução e invocação das ferramentas necessárias sem necessitar de interação de usuários para executar todos os passos. Este é o aspecto diferencial quando comparado acom as abordagens apresentadas em [8, 9, 15]. Nestes trabalhos relacionados, cada tarefa da aplicação é submetida, uma de cada vez, através da interface do usuário. Desta forma, o usuário fica responsável por controlar a ordem de submissão de cada tarefa que trabalha cooperativamente para resolver o mesmo problema. Nesta presente seção estas abordagens são referenciadas como **Outro**.

É bem conhecido que a redução no acesso à rede sem fio pode fornecer uma menor dissipação de energia de bateria destes dispositivos [13], isto é, aumentando seu tempo de vida. Estender o tempo de vida da bateria é um dos problemas mais críticos e desafiadores nestes dispositivos [16]. Desta forma, uma abordagem que fornece mais automatização, como acontece na presente abordagem, enviando menos pedidos de submissão, possibilita a redução da dissipação de energia da bateria dos dispositivos.

No gráfico da Fig. 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 proposta** em contraste com a abordagem **Outro** implementada em [8, 9, 15]. A aplicação considerada é um sequenciamento de DNA que requer a execução de 9 tarefas em uma forma organizada e controlada para alcançar sua resolução. A implementação foi obtida de [11]. O resultado mostra que a **Arquitetura** alcançou uma economia média de 43% quando comparada à abordagem **Outro**. Uma vez que ela permite um número maior de execuções, isto é, em média mais de 12 execuções de aplicação, como pode ser visto no gráfico seguinte.

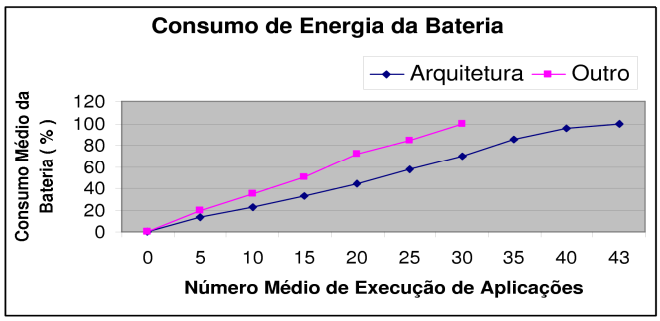

Figura. 6. Consumo Médio da Bateria.

Foram executados 24 experimentos em cada abordagem, cada experimento representa a porcentagem do consumo de bateria de 5 execuções de aplicações do problema de sequenciamento (ou seja, um total de 120 submissões da aplicação em cada abordagem). Em cada execução da aplicação foi acumulada a porcentagem de consumo para submeter e monitorar cada tarefa da aplicação na abordagem **Outro** e foi checada a porcentagem de consumo para submeter e monitorar uma aplicação na **Arquitetura proposta**. 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.

A média e o intervalo de confiança de consumo de bateria para cada experimento foi respectivamente 11% e ***(+/-)*** 1,6603% na **Arquitetura proposta**, enquanto na abordagem **Outro** foi 16,17% e ***(+/-)*** 1,6323%. Um ***teste T de 2 amostras*** [14] foi usado para comparar se existe uma diferença significante 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 [14] observa, quando ***P-value*** $\leq$ 0,05, indica-se uma diferença significativa entre as duas amostras. Isto demonstra que a **Arquitetura** economiza significativamente mais energia de bateria do que a abordagem **Outro** para submeter e monitorar a aplicação no PDA.

V. Conclusões

Neste artigo foi apresentada uma arquitetura que permite submissão e monitoração de aplicações workflow 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 de workflow 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 também, combinando redução de consumo de energia com características

de alto desempenho e processamento 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. Assim, fornece uma facilidade de ajuste para a execução do fluxo das tarefas, considerando os requisitos da aplicação submetida e as opções definidas pelo usuário. Esta funcionalidade garante a consistência do fluxo de uma maneira personalizada para o usuário móvel. Os resultados experimentais indicaram que a abordagem proposta permitiu consumir menos energia da bateria de um PDA para submeter e monitorar aplicações (aproximadamente 43%).

Como direção futura de pesquisa, será proposto 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] J. Brooke and M. Parkin. A pda client for the computational grid. In *WETICE'05*, pp. 325–330, Washington, DC, USA. IEEE Computer Society,2005.

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

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

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

[5] A. K. 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.

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

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

[8] J. Paul, I. Darby, N. F. Tzeng, Peer-to-peer checkpointing arrangement for mobile grid computing systems, in: HPDC '07: Proc of the 16th international symposium on High performance distributed computing, ACM, New York, NY, USA, 2007, pp. 211–212.

[9] N. Imran, I. Rao, Y.-K.. Lee and S Lee. A proxy-based uncoordinated checkpointing scheme with pessimistic message logging for mobile grid systems, in: HPDC '07: Proceedings of the 16th international symposium on High performance distributed computing, ACM, New York, NY, USA, 2007, pp. 237–238.

[10] 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.

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

[12] 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.

[13] S. Mohapatra, R. Cornea, H. Oh, K. Lee, M. Kim, N. Dutt, N, 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.

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

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

[16] 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*, pp. 906–911. ACM Press, 2003.

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

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

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

[20] A. Ikram, A. Ali, A. Anjum, C. Steenberg, H.B. Newman, J.J. Bunn, M. Thomas, T. Azim , "Grid enabled data analysis on handheld devices," *Networking and Communication, 2004. INCC 204. International Conference on*, vol., no., pp. 158-164, 11-13 June 2004.

Vinicius da Cunha Martins Borges: graduou-se em Ciência da Computação pela Universidade de Rio Verde em 2003, obteve o grau de mestre em Ciência da Computação pela Universidade Federal de Santa Catarina (2006). Atualmente, está cursando o doutorado em Ciência e Tecnologia da Informação pela Universidade de Coimbra (Portugal). Seus principais interesses de pesquisa são: Redes de Computadores e Sistemas Distribuídos, mais especificadamente grade móvel, computação móvel e redes sem fio.

Anubis Graciela Moraes Rossetto: graduou-se pela Universidade de Passo Fundo (1998), obteve o grau de mestre pela Universidade Federal de Santa Catarina (2007). Atualmente, é Professora Assistente II da Universidade de Passo Fundo. Tem experiência em Ciência da Computação, com ênfase em Sistemas Distribuídos. Seus principais interesses de pesquisa são grade e computação móvel.

Mario Antonio Ribeiro Dantas: graduou-se em Engenharia Elétrica em 1984 na Universidade Gama Filho, e também se formou em Analise de Sistemas em 1986 pela Universidade Católica, ambas no Rio de Janeiro, Brasil e obteve o título de PhD pela Universidade de Southampton (Inglaterra). Trabalhou no Departamento E&P de Petrobrás por 20 anos, como Analista de Sistemas especializado em aplicações distribuídas e de engenharia. Em 1998 tornou-se professor pela UnB (Universidade de Brasília) onde pesquisou e ensinou, como Professor Associado, assuntos relacionados a Sistemas Distribuídos e Redes de Computadores em cursos de bacharelado e mestrado. Em 2002, mudou-se para Universidade Federal de Santa Catarina, onde continua sua atividade de pesquisa e ensino atualmente.