Utilizando Adaptação Consciente da Aplicação No Acesso a Arquivos em um Ambiente Pervasivo

Gustavo Frainer, Luciano Cavalheiro da Silva, Cláudio Fernando R. Geyer
PPGC-II-Univ. Federal do Rio Grande do Sul
Porto Alegre, RS, Brazil
frainer, lucc, geyer@inf.ufrgs.br

Iara Augustin
PPGI-INF-Univ. Federal de Santa Maria
Santa Maria, RS, Brazil
august@inf.ufsm.br

Adenauer Corrêa Yamin
PPGINF-ESIN-Univ. Católica de Pelotas
Pelotas, RS, Brazil
adenauer@ucpel.tche.br

Resumo

O Espaço Pervasivo de Arquivos (EPA) é um serviço do middleware EXEHDA que provê acesso a arquivos de forma a implementar o conceito de semântica 'siga-me' das aplicações pervasivas. Esse serviço introduz um novo modelo para adaptação ciente da aplicação que fornece métodos para que as aplicações provejam informações específicas para guiar a adaptação ou estendam o EPA com módulos de aplicação que podem se adaptar a novos elementos de contexto. A combinação dessas duas estratégias permite que o modelo do EPA ultrapasse muitas das limitações apresentadas em trabalhos sobre sistemas de arquivos para computação pervasiva.

1 Introdução

A visão de ambientes de computação pervasiva em escala global, caracterizada por permitir aos usuários mobilidade espacial irrestrita com manutenção do acesso a seu ambiente computacional, tem como requisito o suporte à semântica 'siga-me' das aplicações [2]. O projeto ISAM tem como objetivo estudar possíveis soluções para viabilizar esse tipo de ambiente de computação global, tendo como premissa a integração de conceitos advindos da computação em grade, da computação sensível ao contexto e da computação móvel [1].

No estágio atual, a arquitetura de software ISAM disponibiliza um ambiente de execução, implementado na forma de um *middleware* baseado em serviços, chamado EXEHDA [17]. Esse middleware possui duas funções principais: (i) gerenciar o ambiente pervasivo de execução das aplicações, chamado ISAMpe; e (ii) viabilizar a semântica 'siga-me', provendo mecanismos que permitam a mobilidade das aplicações para diferentes contextos, seguindo o usuário conforme este se desloca. Por questões de equilíbrio entre flexibilidade e desempenho para tratar com o ambiente dinâmico e heterogêneo que é intrínseco à visão de computação pervasiva, o EXEHDA utiliza uma abordagem colaborativa entre middleware e aplicação para implementar a tomada de decisão da adaptação baseada em informações de contexto [18].

O suporte a mobilidade das aplicações requer um tratamento adequado do acesso a dados. Assim, para permitir que a aplicação siga o usuário, um dos serviços que o EXEHDA deve oferecer é o de acesso pervasivo a arquivos. Entende-se por acesso pervasivo o acesso de qualquer lugar, a qualquer tempo e a partir de qualquer dispositivo.

O projeto desse serviço foi inspirado em trabalhos anteriores que verificaram a importância, na gerência de arquivos em um ambiente pervasivo, do uso de adaptação que seja consciente não apenas do contexto do ambiente de execução subjacente, mas que também o seja com relação à aplicação que nele executa [10, 13, 9].

A necessidade de adaptação ciente da aplicação é potencializada em propostas de *middleware* de propósito geral, como o EXEHDA, os quais não são projetados para trabalhar com um domínio específico de aplicação. Enquanto a necessidade de cada aplicação com relação ao acesso a arquivos exibe características diferenciadas, o *middleware* de propósito geral não tem conhecimento prévio de todos os tipos de arquivos que este virá a gerenciar e nem da melhor abordagem para tratar cada um desses arquivos ou tipos de arquivos. Devido a tais requisitos diferenciados de acesso a dados, na maior parte das vezes, é a própria aplicação a en-

tidade melhor situada para escolher qual adaptação deve ser realizada no contexto atual, de forma que suas necessidades sejam atendidas.

Os primeiros sistemas de arquivos que abordaram a adaptação consciente da aplicação no acesso a dados [11] apresentavam grandes limitações, sendo pouco flexíveis. Em tais sistemas, o conjunto de elementos de contexto passíveis de serem considerados nos processos de adaptação era fixo e bastante restrito (consideravam em geral alguma métrica de comunicação, como vazão de dados). Além disso, os mecanismos disponibilizados para a construção da adaptação de forma colaborativa com a aplicação também eram bastante limitados nos tipos de adaptação contempladas.

Sistemas mais recentes [5, 4] têm considerado um número maior de elementos de contexto, mas ainda são capazes de responder apenas a um pré-determinado subconjunto de elementos de contexto.

Entendemos que qualquer sistema que limite os tipos de elementos de contexto que podem ser considerados nas decisões de adaptação provavelmente não será adequado para aplicações de diferentes domínios. Uma expressão mais flexível de aplicações é importante principalmente nesse estágio inicial da computação pervasiva onde ainda não está claro quais tipos de aplicações vão existir nesse novo cenário.

Em termos de adaptação, a maioria dos sistemas conscientes da aplicação trabalha somente com adaptação de conteúdo e/ou comunicação. Apesar dos projetos mais recentes terem proposto o uso de sistemas de replicação mais complexos , que deixam de lado a arquitetura cliente/servidor e optam por organizar os recursos em redes P2P [6, 8], eles não propõem maneiras de permitir que as aplicações influenciem a maneira pela qual esses sistemas de replicação se adaptam para lidar com o ambiente dinâmico.

Motivado por essas limitações e pelo requisito do middleware EXEHDA de permitir a semântica 'siga-me' das aplicações foi projetado o Espaço Pervasivo de Arquivos (EPA), um serviço adaptativo de arquivos flexível e extensível que provê um modelo novo para adaptação ciente da aplicação.

O restante desse artigo apresenta o EPA. Na seção 2 os trabalhos relacionados e a motivação são discutidos com mais detalhes. Na seção 3 os aspectos relevantes do middleware EXEHDA são descritos. A seção 4 descreve a arquitetura e implementação do EPA. A seção 5 apresenta resultados obtidos. Conclusões são apresentadas na seção 6.

2 Trabalhos Relacionados

Para facilitar a discussão os trabalhos relacionados são divididos em dois grupos: adaptação ciente da aplicação e sistemas de arquivos pervasivos baseados em infra-estrutura.

2.1 Adaptação Ciente da Aplicação

Odyssey [10] é um trabalho pioneiro que introduziu e definiu muitos dos conceitos de adaptação ciente da aplicação. Ele permitia que aplicações modificassem a fidelidade dos dados transferidos entre servidor e o cliente , em resposta a mudanças em um conjunto de recursos que incluía banda da rede e CPU. Entretanto, não houve nenhuma discussão sobre como novos recursos, além daqueles já previstos, poderiam ser monitorados e como poderia haver adaptação a mudanças nesses recursos.

No Odyssey , adaptação é sempre a "troca de qualidade dos dados por consumo de recursos". Esse é um tipo de adaptação importante em sistemas de arquivos pervasivos, mas não a única.

Outros trabalhos expandiram o que foi proposto pelo Odyssey. Em [4] os pesquisadores propõem um processo para adaptação a um grande número de contextos e aplicam esse processo a uma aplicação específica. Contudo, tratam apenas com a decisão de qual dado pré-existente deve ser usado e não discutem como obter os elementos de contexto relevantes em outras aplicações. O trabalho apresentado em [5] estende muitos dos conceitos do Odyssey para lidar com vida da bateria e consumo de energia.

Também existem vários sistemas que não seguem estritamente a definição de adaptação ciente da aplicação do Odyssey, mas que permitem que componentes específicos da aplicação ou do tipo de dados sejam adicionados ao sistema e usados para guiar a adaptação [14, 7, 9]. A maior parte desses sistemas trata apenas de adaptação de conteúdo.

2.2 Sistemas de Arquivos Pervasivos Baseados em Infraestrutura

O termo baseado em infra-estrutura é usado aqui para definir aqueles sistemas de arquivos pervasivos que suportam o acesso a arquivos de dispositivos mais limitados através do uso de alguma infra-estrutura, geralmente composta de um ou mais servidores que são computadores mais poderosos e que possuem uma conexão estável.

O trabalho pioneiro nessa área é o sistema de arquivos Coda [13]. Entretanto, pesar dos servidores do Coda poderem ser replicados por questões de tolerância a falhas, a arquitetura Coda ainda é essencialmente Cliente/Servidor. Esse também é o caso de outros sistemas de arquivos pervasivos baseados em infra-estrutura [14].

Conforme o usuário se move para longe do servidor , os custos de contactar este através de uma rede de larga escala sobem e o desempenho do sistema baixa. Para lidar

com esse problema , alguns pesquisadores propuseram uma extensão do modelo cliente/servidor para incluir nodos auxiliares. Tais nodos podem estar mais próximos ao dispositivo do usuário e que agem como um intermediário entre usuário e servidor, fazendo *cache* de arquivos [6], propagando as modificações para o servidor [8], e assim por diante. Outros pesquisadores propuseram a substituição do servidor por esses nodos auxiliares, permitindo que eles se comunicassem entre si de forma par-a-par [15].

Os trabalhos citados acima que estendem o modelo cliente/servidor muitas vezes observam o padrão de acesso a arquivos do usuário e utlizam o que foi observado para informar as suas decisões sobre o posicionamento das cópias dos arquivos. Alguns também permitem que o usuário informe diretamente que arquivos ele considera mais importantes. O trabalho descrito em [6], por exemplo, utiliza ambas essas abordagens. Mas nenhum deles fornece uma interface para que as aplicações influenciem as decisões de posicionamento.

Esses sistemas muitas vezes também não consideram os mecanismos para encontrar, escolher e instalar o software necessário nos nodos auxiliares [7, 6, 8], ou impõem limites fortes no conjunto de nodos que podem ser usados [15].

Essas duas limitações, juntamente com as limitações discutidas acima presentes em sistemas de adaptação consciente da aplicação, motivaram o desenvolvimento do EPA.

3 O Middleware EXEHDA

O EPA foi implementado com o suporte do *middleware* EXEHDA, o qual fornece dois componentes essenciais: acesso a um ambiente pervasivo com escopo global e um *framework* para a definição e monitoramento do contexto de execução.

A forma como esses componentes são implementados pelo EXEHDA será discutida agora.

3.1 O ISAM Pervasive Environment

Na arquitetura de software ISAM, todos os recursos disponíveis são organizados dentro do ISAM Pervasive Environment (ISAMpe) [17]. O ISAMpe é composto por várias células, cada uma delas composta por vários dispositivos móveis e estacionários, e por uma infra-estrutura de rede interconectando-os. Fica a cargo desses recursos realizar várias tarefas de administração. Uma célula geralmente é composta pelos recursos de uma organização pré-existente. Um conjunto desses recursos é definido pelos administradores da célula para formarem a base da célula.

Não existe um serviço central que administre todas as células ou que controle a comunicação entre elas. Toda a comunicação intercelular ocorre de uma maneira par-a-par.

Para realizar as suas tarefas o EPA interage com outros serviços do EXEHDA. Esses serviços provêem ao EPA, acesso aos dispositivos do ambiente, incluindo seus sistemas de arquivos, como detalhado na seção 3.

3.2 Serviço de Reconhecimento de Contexto

A arquitetura ISAM define contexto como "todas as informações relevantes para a aplicação que podem ser obtidas do sistema". Cada aplicação pode definir e identificar os elementos que compõem o seu contexto.

Uma vez que a aplicação tenha definido quais são os seus elementos de contexto de interesse, ela precisa ser capaz de obter informações sobre estes. O ContextServer, um serviço do EXEHDA, registra o interesse da aplicação, monitora os elementos de contexto e notifica a aplicação quando alterações acontecem. Esse serviço monitora os elementos de contexto através da agregação de dados de sensores distribuídos através do ambiente pervasivo. Sensores podem obter uma grande variedade de informações, desde o estado de um elemento de hardware em particular até que usuário está utilizando atualmente um certo dispositivo.

4 O Espaço Pervasivo de Arquivos

O EPA foi modelado através da noção de arquivo pervasivo, metadados e adaptação ao contexto. Esse modelo permite que a aplicação participe nas decisões de adaptação do sistema de três maneiras: adição aos arquivos de metadados; desenvolvimento de módulos que são usados em vários pontos, substituindo o comportamento padrão do sistema; e através da primitiva de *pre-fetch*.

4.1 A Abstração de Arquivo Pervasivo

Um aspecto central do EPA é a noção de "arquivo pervasivo", entendida como um arquivo composto de várias réplicas, espalhadas pelo ISAMpe. Esse espalhamento procura atender as propriedades de disponibilidade e acessibilidade da informação ao usuário a qualquer tempo, em qualquer lugar, com qualquer dispositivo. Cada arquivo pervasivo pode ter mais de uma versão. Nesse caso, cada uma das versões será ela mesma um arquivo pervasivo.

Cada arquivo pervasivo é identificado por um id único. Esse id é formado pela identificação de uma célula do ISAMpe e por um número. A porção identificando uma célula do id refere-se à célula onde o arquivo foi originalmente criado.

A célula identificada no id de um arquivo será responsável por aquele arquivo pervasivo. Na prática isso significa que a célula irá manter uma cópia e a localização de cada outra cópia do arquivo. Outros dispositivos que sejam

parte do ISAMpe vão informar a célula responsável sempre que uma nova cópia for criada ou quando uma cópia antiga for descartada. Usando os resultados das pesquisas de Satyanarayanan [13], o EPA utiliza um sistema de replicação otimista [12], o qual admite que a informação de localização possa estar desatualizada em um momento específico. Contudo, as atualizações relevantes serão eventualmente realizadas, desde que as conexões de rede não tenham sido permanentemente comprometidas. A abordagem de replicação otimista permite que se alcance uma mais alta disponibilidade dos arquivos, e seu uso é muito comum em sistemas que precisam lidar com dispositivos móveis [13].

Para tornar o acesso mais rápido e confiável, cada célula também armazena a localização de todas as cópias de qualquer arquivo armazenadas em algum dispositivo que pertença à célula, além de armazenar a localização das réplicas de um arquivo pelo qual a célula é responsável.

Quando um dispositivo quer localizar uma cópia de um dado arquivo pervasivo, o primeiro passo é fazer uma consulta à célula local, para procurar por cópias hospedadas em algum computador pertencente a ela. Se não existe nenhuma cópia na célula local o serviço do EXEHDA para descoberta de recursos, PERDiS, é usado para procurar uma cópia em alguma célula próxima [17]. Se esta procura falhar, a célula responsável é contatada e a localização das cópias do arquivo é obtida. Esse processo mantém a maioria das procuras dentro da célula local, evitando a necessidade de contatar células mais distantes.

4.2 MetaDados

O EPA permite tanto que aplicações e quanto o *middleware* acessem e atualizem metadados associados a um arquivo, os quais são representados na forma de um conjunto de pares {chave,valor}. Esses metadados podem ser utilizados para associar informações semânticas ao arquivo, como o tipo do dado (imagem,texto, etc.), padrões de acesso ao arquivo é utilizado (somente lido,constantemente escrito, etc.), ou ainda informações específicas (e.g., versão) que a aplicação e o *middleware* considerem importante.

Esses metadados são uma parte essencial dos mecanismos de adaptação utilizados pelo EPA. A forma exata como os metadados são utilizados pelo EPA para guiar a adaptação são discutidos nas seções 4.4 e 4.5.

Os metadados associados a um arquivo são eles prórpios armazenados na forma de arquivos junto com todas as cópias dos dados do arquivo pervasivo ao qual eles estão associados. Essa abordagem garante que sempre que um arquivo estiver acessível, os metadados para aquele arquivo também o estarão. Ainda que estejam desatualizados em algumas situações, essa propriedade é importante para manter a habilidade de o serviço de operar mesmo sob conexão de rede limitada e indisponibilidade de certos recursos.

4.3 Adaptação no EPA

O EPA implementa um mecanismo de adaptação flexível e extensível baseado em módulos de aplicação que podem ser adicionados sem a necessidade de modificar todo o *middleware*. No protótipo atual, tais módulos correspondem classes Java que implementam interfaces específicas e que são carregadas durante a execução sob demanda.

Existem três tipos de módulos de aplicação que podem ser utilizados no EPA: módulos de prioridade de arquivos, módulos de escolha de versão e módulos de conversão de conteúdo. As funções e interfaces desses módulos são discutidas nas seções seguintes.

4.4 Posicionamento e Movimentação de Cópias de Arquivos

Do ponto de vista do EPA, cada dispositivo pode assumir um de três papéis: cliente, servidor de cópias e servidor base.

As máquinas que são clientes são capazes de acessar e modificar arquivos pervasivos, mas elas não armazenam cópias; entretanto, mantêm um *cache* para acesso local aos arquivos que foram recentemente acessados pelo dispositivo. Esse *cache* pode ser utilizado, por exemplo, durante os periodos em que o dispositivo se encontrar desconectado do resto da rede.

Por sua vez, servidores de cópias podem manter cópias de arquivos pervasivos. Essas cópias são registradas nas bases das células apropriadas e podem ser localizadas e acessadas por outros computadores. Já os servidores do tipo base são responsáveis por algumas tarefas de gerenciamento, como comunicar com a Base de Informações da Célula (CIB) [17] e com outras células e decidir em qual dispositivo cada cópia deve ser colocada.

Conforme modelado na administração do ISAMpe [17], a decisão de qual dispositivo vai assumir cada papel fica a cargo dos administradores de cada célula. Por exemplo, alguns computadores, como PDAs, devem sempre receber o papel de cliente, já que não possuem recursos para desempenhar outro papel, enquanto que servidores de cópia devem ser computadores mais robustos e confiáveis.

4.4.1 Mecanismo de Posicionamento Adaptativo

Sempre que um arquivo pervasivo é aberto em certo dispositivo, o EPA ativa o seu mecanismo de posicionamento adaptativo. O objetivo desse mecanismo é decidir se o arquivo que é objeto dessas chamadas deve receber uma nova cópia e, sendo esse o caso, escolher onde a cópia deve ser hospedada.

Quando um arquivo é aberto , o EPA primeiramente determina a prioridade daquele arquivo. A prioridade é um

valor numérico entre 1 e 5 que serve a dois propósitos: (i) definir se novas cópias do arquivo precisam ser criadas e (ii) guiar decisões de substituição no *cache*.

Para determinar a prioridade de um arquivo, o EPA começa atribuindo uma prioridade padrão 3 ao arquivo. O EPA então começa a inspecionar os metadados do arquivo. Alguns atributos presentes nos metadados fazem com que o EPA diminua a prioridade do arquivo. Por exemplo, um atributo indicando que o arquivo geralmente só é lido uma vez em cada execução da aplicação, ou que ele nunca é escrito. Outros atributos aumentam a prioridade, como atributos que o arquivo é popular e continuamente acessado, ou que ele é essencial para a operação da aplicação. Alguns atributos podem afetar a prioridade de forma diferente dependendo do contexto. Um atributo indicando quem é o "proprietário" de um arquivo pode reduzir a prioridade se o usuário que faz o acesso não é o proprietário ou aumentar a prioridade se ele o for.

Para poder interpretar os atributos , e o efeito que eles devem ter sobre a prioridade do arquivo , é usado um conjunto de módulos de prioridade de arquivos. Cada um desses módulos é associado a um atributo. Quando o atributo está presente em um arquivo o EPA delega a decisão ao modulo correspondente aquele atributo. O módulo, então, indica que mudanças devem ser feitas à prioridade. A aplicação pode adicionar novos módulos de prioridade ao *middleware* para controlar a prioridade dos arquivos que ela utiliza.

O nível de prioridade influencia tanto a politica de caching quanto de armazenamento de cópias. Um arquivo com uma prioridade mais baixa não pode substituir um arquivo com uma prioridade mais alta em um servidor de cópias. E quando existe uma necessidade de remover arquivos para criar espaço, seja em um *cache* local ou em um servidor de cópias, os arquivos com prioridades mais baixas são removidos primeiro. Entre os arquivos que tem a mesma prioridade é usada a heurística de menos recentemente utilizado. A prioridade de arquivos armazenados em um *cache* ou em um servidor de cópias diminue gradualmente com o tempo se eles não forem acessados.

Se a prioridade de um arquivo for 4 ou 5 e uma cópia do arquivo não existir naquele dispositivo, o EPA irá tentar criar uma nova cópia do arquivo no dispositivo. Se o dispositivo não tiver capacidade de manter cópias, ou se não for possível criar a cópia devido a limitações de espaço, uma nova cópia irá ser criada em outro dispositivo na mesma célula, desde que não exista outra cópia na célula. Essa estratégia vale-se do fato de que computadores na mesma célula tendem a pertencer a mesma organização e também tendem a ter entre si conexões mais rápidas e confiáveis.

A prioridade 3 tem uma abordagem similar à descrita acima, com a diferença que uma cópia não vai ser criada no dispositivo se para isso arquivos precisarem ser removidos. A prioridade 1 ou 2 não permite a criação de uma nova cópia.

Observe que mesmo nos casos de prioridade baixa o arquivo vai ser mantido no *cache* local, desde que haja espaço. O descrito acima se refere apenas a criação de novas cópias.

4.4.2 A Primitiva *pre-fetch*

Em ambientes móveis, as aplicações podem ter informações sobre arquivos que elas vão precisar acessar no futuro. Nesses casos, pode ser interessante tomar algum tipo de ação pró-ativa para preparar o acesso a esses arquivos. No EPA, tem-se modelado uma primitiva chamada *pre-fetch* que permite às aplicações indicar ao EPA que um arquivo vai ser acessado, com uma alta probabilidade, em um computador específico no futuro próximo. Baseado nessa informação, o EPA pode agir de forma pró-ativa para tornar o acesso futuro mais eficiente.

A chamada de um *pre-fetch* dispara um mecanismo de posicionamento que vai tentar colocar uma cópia do arquivo no dispositivo onde se espera que ele seja acessado. Se isso não for possível, devido a restrições no tamanho do *cache* por exemplo, o EPA vai tentar colocar uma cópia arquivo em um dispositivo próximo. Essa é uma tentativa de melhorar o acesso através da diminuição de latência e, possivelmente, através do aumento da banda de acesso.

4.5 Adaptação de Conteúdo e Escolha de Versão

Como já mencionado, uma arquivo pode ter mais de uma versão. Se esse for o caso o EPA vai decidir, no momento de abertura, qual a versão é mais adequada dependendo do contexto de execução.

Existem duas maneiras pelas quais versões alternativas podem ser criadas para um dado arquivo. A primeira é uma aplicação usar a interface do EPA para registrar um arquivo pervasivo já existente como sendo uma versão alternativa de outro arquivo pervasivo já existente. Essa informação vai ser armazenada nos metadados do arquivo e o EPA vai verificar ela quando estiver abrindo um arquivo.

A segunda maneira envolve a criação dinâmica de versões alternativas. Pode ser possível e útil ser capaz de converter um arquivo em outro em certos contextos. Um exemplo comum é a conversão de um grande arquivo BMP em um arquivo GIF menor antes de transmitir o arquivo através de uma conexão de rede lenta.

Para suportar esse tipo de conversão, o EPA permite o uso de módulos de conversão criados pelas aplicações. Eles também devem ser capazes de criar os metadados apropriados para o novo arquivo.

Quando um novo módulo conversor é registrado, o tipo de arquivo com o qual ele pode lidar é informado na forma de um conjunto de pares {chave,valor} que devem estar presentes nos metadados do arquivo. Quando um arquivo com

os metadados adequados é aberto o EPA cria um "arquivo virtual" para representar a versão alternativa que pode ser criada. O arquivo só é realmente criado se aquela versão for selecionada para uso.

Nos casos em que um dado arquivo tem mais de uma versão, ou quando uma versão alternativa pode ser criada, uma heurística é utilizada para decidir que versão deve ser utilizada quando o arquivo for aberto. Essa heurística pode levar em conta os metadados das várias versões e o atual contexto de execução na sua decisão. Seguindo a filosofia da arquitetura do EPA, as heurísticas são módulos, e novas heurísticas podem ser adicionadas pela aplicação e por outros serviços.

A escolha de qual heurística utilizar com certo arquivo é feita através de indicações presentes nos metadados do arquivo principal.

5 Resultados

Para demonstrar as capacidades do EPA e como elas afetam o desempenho de acesso a arquivos, vários experimentos foram realizados usando uma aplicação simples de visualização de imagens. Essa aplicação permite que o usuário visualize as imagens que ele seleciona , no tamanho apropriado para cada dispositivo que ele esteja utilizando .

Nos testes é simulado um usuário utilizando essa aplicação em um PDA para acessar imagens inicialmente localizadas em um computador em uma célula diferente daquela onde ele se encontra. Para facilitar os testes, um computador desktop é utilizado como dispositivo de acesso. Mas o funcionamento de um PDA é simulado limitando a *cache* do dispositivo a 64mb e através do uso da versão do EPA voltada a dispositivos com recursos limitados. Apesar de desktop ter mais poder computacional do que um PDA típico, isso não resulta em uma grande diferença nos testes já que a maior parte da computação é enviada a uma base EPA.

Todos os dispositivos estão conectados a mesma rede local. O NISTnet [3] é utilizado para adicionar um atraso de 30ms entre a célula local e a célula remota e para adicionar um limite de 1 Mb/s a largura de banda da conexão entre elas. Esses valores foram baseados em medidas realizadas sobre a conexão entre computadores nas cidades de Porto Alegre(RS) e Pelotas(RS), e estão na faixa mais otimista dos valores observados na realidade.

Cada imagem do conjunto pode ser acessada mais de uma vez. O número de vezes que cada imagem é acessada é determinado de acordo com uma distribuição Zipf. Já foi demonstrado que essa distribuição oferece um bom modelo para os padrões de acesso da internet, incluindo o acesso a coleções de imagens baseadas na *web* [16]. O total de dados acessados pela aplicação, contando acessos repetidos, foi de aproximadamente 390Mb.

Como este trabalho está interessado apenas nos aspectos relacionados a arquivos da aplicação, apenas o código de acesso aos arquivos é executado, e é medido o tempo para abrir e ler cada arquivo. Todos os acessos são executados de forma consecutiva.

Utilizando esse arranjo foram realizados 5 experimentos:

1. Leitura dos arquivos diretamente do servidor remoto;
2. Leitura através do EPA, sem o uso de metadados ou módulos específicos;
3. Leitura através do EPA com adição de metadados aos arquivos indicando quantas vezes os arquivos foram acessados anteriormente. Um módulo de aplicação é utilizado para dar aos arquivos uma prioridade 1 no primeiro acesso e para dar prioridades cada vez mais altas conforme o arquivo é acessado mais vezes;
4. Leitura através do EPA, mas convertendo os arquivos originais antes de transferi-los. As figuras são convertidas em versões JPG comprimidas que também são redimensionadas para tamanho compatível com a tela de PDA. Um módulo de conversão é utilizado para realizar a transformação. Uma nova heurística de escolha é utilizada pra escolher a versão adequada e metadados são adicionados aos arquivos identificando-os como imagens e indicando seu tipo e tamanho;
5. Leitura dos arquivos para mostrar um *slide show* de um conjunto pré-definido de imagens. Nesse experimento foi usado um conjunto menor de imagens e foi incluido um delay de 5 segundos entre a visualização de uma imagem e outra. A primitiva *pre-fetch* é utilizada para recuperar as imagens do servidor remoto antes de exibi-las.

Esses experimentos exploram algumas das maneiras pelas quais a aplicação pode influenciar o EPA, através do uso de metadados, módulos de aplicação e *pre-fetch*.

A figura 1 mostra os resultados dos primeiros quatro experimentos. O uso do EPA no segundo experimento diminui o tempo total em aproximadamente 75% , quando comparado com o acesso remoto a todos os arquivos. Como os arquivos no primeiro experimento não têm nenhum metadado associado, eles são abertos com a prioridade padrão (3). Baseado nessa prioridade e na inabilidade do dispositivo do usuário de manter cópias, o EPA cria uma cópia na base EPA após ela ser acessada pela primeira vez. Visto que a conexão entre base e dispositivo de acesso é muito mais rápida do que a entre as duas células , os acessos subseqüentes a esses arquivos apresentam um melhor desempenho.

Os resultados do terceiro experimento são um pouco piores dos que os do segundo. Existem duas grandes diferenças entre os dois. A primeira é que no terceiro experimento uma

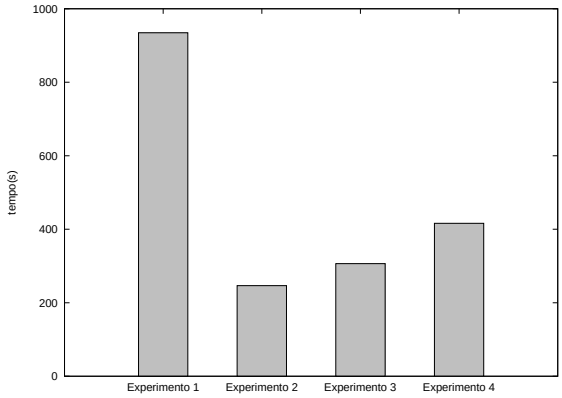

Figura 1. Resultados dos primeiros quatro experimentos. O tempo é a media de 10 execuções.

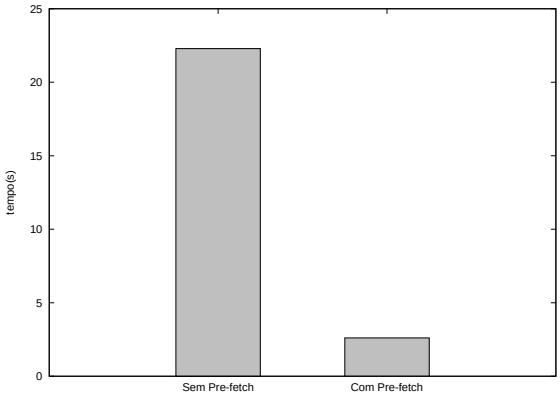

Figura 2. Tempo "perdido" durante a apresentação do *slide show*. Todos os resultados representam a média de dez execuções.

nova cópia não é feita após o primeiro acesso, no qual a prioridade do arquivo é 1. Isso acaba gerando uma perda de eficiência nesse caso. A segunda diferença é que no terceiro experimento o EPA consegue manter um número maior das imagens mais acessadas no *cache* do dispositivo que faz o acesso, já que elas gradualmente têm suas prioridades aumentadas e não são substituídas por arquivos acessados menos comumente.

A habilidade de manter arquivos mais populares no *cache* do dispositivo de acesso não causa muita diferença positiva nesse teste. Acessar os arquivos na base EPA usando a rede local é apenas levemente pior do que acessar os arquivos localmente. Se a conexão de rede com a base EPA fosse mais limitada, a expectativa é que o terceiro experimento teria apresentado resultados melhores dos que aqueles observados no segundo experimento. Manter os arquivos mais populares no *cache* local também seria vantajoso em caso de desconexão onde o dispositivo de acesso não pode acessar nenhum arquivo que já não esteja no *cache*.

No experimento 4, um tempo é gasto para converter e redimensionar cada imagem, mas o os ganhos obtidos pela transmissão mais rápida e a habilidade de manter mais arquivos no *cache* local compensam isso, gerando um resultado melhor do que simplesmente acessar os arquivos originais. Os resultados desse experimento são mais marcantes se for considerado que a aplicação gastaria algum tempo redimensionando as imagens para mostrá-las, se isso já não tivesse sido feito.

A figura 2 mostra os resultados do experimento 5, comparando o tempo "perdido" quando o *pre-fetch* é e não é utilizado. Tempo perdido é o tempo que a aplicação leva para ler um arquivo antes de mostrá-lo. No caso do uso de *pre-fetch* o tempo é mínimo, por que os arquivos são transferidos da célula remota durante o período em que outras imagens estão sendo mostradas.

Uma variação desse experimento seria a aplicação ter, ela própria, explicitamente gerenciado a transferência dos arquivos que ela viria a utilizar, sem o uso do *pre-fetch*. Nesse caso, entretanto, a aplicação teria que se preocupar com o espaço em *cache*, em como colocar o arquivo em um nodo auxiliar caso o *cache* local não fosse suficiente, e assim por diante. Ao usar o *pre-fetch* o *middleware* é o responsável por lidar com essas questões e pode se adaptar para levar em conta a melhor abordagem dependendo do contexto em que executa.

Os testes discutidos acima não são de nenhuma forma exaustivos. Mas eles fornecem uma boa indicação de que os mecanismos do EPA podem resultar em uma qualidade maior de acesso a arquivos em ambientes pervasivos.

6 Conclusões

6.1 Contribuições

O EPA usa um novo modelo para a adaptação ciente da aplicação. Pode-se destacar as seguintes contribuições principais desse modelo em comparação com outros trabalhos:

- Um número maior de comportamentos adaptativos que podem ser influenciados pela aplicação, incluindo não apenas decisões sobre o conteúdo mas também sobre a distribuição das cópias dos arquivos;
- Capacidade de se adaptar a qualquer elemento de contexto. Com o suporte dos serviços de reconheci-

mento de contexto do EXEHDA o EPA pode monitorar qualquer recurso ou outro elemento de contexto que a aplicação julgue importante, e os módulos das aplicações permitem que o EPA seja estendido para responder a esses novos elementos de contexto;

- Mecanismos para encontrar, escolher e instalar o software necessário nos nodos auxiliares. O EPA lida com essa questão através do uso do *middleware* de grade EXEHDA que permite o acesso e gerência de recursos compartilhados e através da utilização de uma infraestrutura de grade previamente construída;

- Uso de módulos de aplicação. Utilizar módulos de aplicação com interfaces bem definidas , ao invés de simplesmente contactar a aplicação , permite que os módulos sejam compartilhados entre as aplicações, facilitando o desenvolvimento de aplicações pervasivas.

6.2 Considerações Finais

Atualmente, existe um grande número de aplicações usando diferentes tipos de dados e com diferentes necessidades no que se refere ao acesso a eles. Não existe razão para acreditar que aplicações pervasivas vão ser diferentes. Devido à diversidade, dinamicidade e imprevisibilidade do ambiente pervasivo, um sistema de arquivos pervasivo precisa oferecer adaptação não apenas consciente do contexto, mas também ciente da aplicação. O EPA oferece um modelo que prove esse tipo de adaptação.

Referências

[1] I. Augustin. *Abstrações para uma Linguagem de Programação Visando Aplicações Móveis Conscientes do Contexto em um Ambiente de Pervasive Computing*. PhD thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil, 2003. 193p.

[2] I. Augustin, A. Yamin, and C. F. R. Geyer. Managing the follow-me semantics to build large-scale pervasive applications. In *MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, pages 1–8, New York, NY, USA, 2005. ACM Press.

[3] M. Carson and D. Santay. Nist net: a linux-based network emulation tool. *SIGCOMM Comput. Commun. Rev.*, 33(3):111–126, 2003.

[4] D. Chalmers, N. Dulay, and M. Sloman. A framework for contextual mediation in mobile and ubiquitous computing applied to the context-aware adaptation of maps. *Personal Ubiquitous Comput.*, 8(1):1–18, 2004.

[5] J. Flinn and M. Satyanarayanan. Managing battery lifetime with energy-aware adaptation. *ACM Trans. Comput. Syst.*, 22(2):137–179, 2004.

[6] J. Flinn, S. Sinnamohideen, N. Tolia, and M. Satyanarayanan. Data staging on untrusted surrogates. In *2nd USENIX Conference on File and Storage Technologies*, pages 15–28, San Francisco, CA, March 31 - April 2, 2003. USENIX.

[7] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti. Cans: Composable, adaptive network services infrastructure. In *USITS*, pages 135–146. USENIX, 2001.

[8] M. Kim, L. P. Cox, and B. D. Noble. Safety, visibility, and performance in a wide-area file system. In D. D. E. Long, editor, *FAST*, pages 131–144. USENIX, 2002.

[9] H. Lufei and W. Shi. Fractal: A mobile code based framework for dynamic application protocol adaptation in pervasive computing. In *IPDPS*. IEEE Computer Society, 2005.

[10] B. Noble. System support for mobile, adaptive applications. *IEEE Personal Communications*, 7(1):44–49, Feb. 2000.

[11] B. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker. Agile application-aware adaptation for mobility. In *SOSP*, pages 276–287, 1997.

[12] Y. Saito and M. Shapiro. Optimistic replication. *ACM Comput. Surv.*, 37(1):42–81, 2005.

[13] M. Satyanarayanan. The evolution of coda. *ACM Trans. Comput. Syst.*, 20(2):85–124, 2002.

[14] W. Shi, S. Santhosh, and H. Lufei. Cegor: An adaptive distributed file system for heterogeneous network environments. In *ICPADS*, pages 145–152. IEEE Computer Society, 2004.

[15] S. Sobti, N. Garg, F. Zheng, J. Lai, Y. Shao, C. Zhang, E. Ziskind, A. Krishnamurthy, and R. Y. Wang. Segank: A distributed mobile storage system. In *FAST '04 Conference on File and Storage Technologies, March 31 - April 2, 2004, Grand Hyatt Hotel, San Francisco, California, USA*, pages 239–252. USENIX, 2004.

[16] N. Talagala, S. Asami, and D. A. Patterson. Usage patterns of a web-based image collection. In *IEEE Symposium on Mass Storage Systems*, pages 203–214, 1999.

[17] A. Yamin, I. Augustin, L. da Silva, R. Real, and C. Geyer. Exehda middleware: Aspects to manage the isam pervasive environment. In *XXV International Conference of the Chilean Computer Science Society (SCCC 2005), 09-11 November 2005, Valdivia, Chile*, 2005.

[18] A. C. Yamin. *Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva*. PhD thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil, Julho 2004.