Um Suporte à Computação Pervasiva para o Holoparadigma[1]

Daniel Torres Bonatto Jorge Luis Victória Barbosa
Gerson Geraldo H. Cavalheiro
Universidade do Vale do Rio dos Sinos
Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Av. Unisinos, 950 - São Leopoldo, Rio Grande do Sul, Brasil
{bonatto,barbosa,gersonc}@exatas.unisinos.br

Jose Dirceu G. Ramos
Hewlett-Packard Computadores Ltda.
Laboratório de Computação
Av. Ipiranga 6681, Prédio 91A, Porto Alegre, Rio Grande do Sul, Brasil
jose.dirceu@hp.com

Resumo

Nos últimos anos, tem-se observado a crescente evolução dos dispositivos portáteis, bem como de diversas novas tecnologias de comunicação sem fio. Esse avanço tecnológico propicia o surgimento de um cenário ideal para o desenvolvimento de ambientes que suportam a criação de aplicações pervasivas. Porém, um ambiente altamente dinâmico como este demanda a utilização de abstrações mais poderosas do que as existentes. O Holoparadigma propõe uma nova abstração, que foi criada pensando em aplicações distribuídas executando em ambientes móveis. Neste artigo é apresentada a proposta para uma arquitetura de suporte a aplicações pervasivas para o Holoparadigma. Esta proposta estende as funcionalidades da HoloVM e cria novos serviços para atender às demandas da computação pervasiva. Para as primeiras experimentações foi criado o suporte à execução distribuída. São mostrados resultados de experimentos realizados com este novo suporte.

1. Introdução

Na última década, os dispositivos computacionais vêm diminuindo consideravelmente em tamanho e custo. Um exemplo disso são equipamentos como PDAs, *notebooks* e *tablet* PCs, cada vez mais populares em número de usuários e em opções de mercado. A primeira menção a um mundo como o que começamos a ver hoje foi feita por Mark Weiser em 1991 [27]. De forma visionária, Weiser descreveu um mundo onde os ambientes cheios de dispositivos computacionais e comunicação interagiam naturalmente com as pessoas, de tal forma que passavam a fazer parte deste ambiente. Dentro desta visão surgiu o termo *computação ubíqua*, hoje também chamada de *computação pervasiva* [24, 23]. Contudo, em sua época, Weiser e seus colegas não tinham a tecnologia para implementar esta visão.

Para que a computação pervasiva se torne uma realidade, ainda existem alguns desafios a serem vencidos. Desde que Weiser concebeu sua visão de pervasividade, evoluções importantes no hardware podem ser constatadas. Esta evolução permitiu a criação de dispositivos menores e mais portáteis, bem como sensores e dispositivos de controle com crescente poder de processamento. É importante citar ainda as tecnologias para comunicação sem fio, como o Bluetooth [3] e o IEEE 802.11 [18], que criam a possibilidade de acesso a informação a qualquer hora em qualquer lugar, o que não era possível com as redes cabeadas.

Um ambiente pervasivo demanda que as aplicações sejam adaptativas, por natureza. Neste sentido, torna-se interessante adquirir novos elementos lógicos (mobilidade de código) em tempo de execução, ganhando assim funcionalidade para atender às demandas do ambiente. Elas devem também ser sensíveis ao contexto, ou seja, estar conscientes de modificações e adaptar o seu comportamento. A questão da adaptação é importante também quando se fala da necessidade de suportar

1 Este trabalho foi parcialmente desenvolvido em colaboração com a HP Brasil P&D.

a heterogeneidade de um ambiente pervasivo. Neste ambiente pode existir uma grande variedade de dispositivos e redes de comunicação, de forma que a aplicação deve utilizar os recursos disponíveis da melhor forma possível. Outra questão importante é a disponibilidade de dados e serviços, contornando desconexões e falhas em partes da aplicação.

A forma de programação e composição da aplicação é uma questão importante. A maioria das soluções existentes utilizam-se de paradigmas já consagrados para implementar suas soluções. Contudo estes paradigmas foram concebidos para criar aplicações que executam em apenas uma máquina. Desta forma, tais paradigmas não são suficientemente expressivos para modelar aplicações para um ambiente tão dinâmico quanto o que é apresentado na computação pervasiva [14]. Esse fator motiva o desenvolvimento de novas abstrações para programação.

O Holoparadigma [8] (de forma abreviada, Holo) oferece uma abstração intuitiva para a modelagem de ambientes móveis, permitindo criar representações mais fiéis ao mundo real. Utilizando as diretivas de mobilidade do paradigma, é possível manter o modelo coerente com o sistema real representado, mesmo havendo alterações dinâmicas em seus elementos. Esta modelagem aplica-se tanto a elementos fixos (prédios, salas, computadores, etc.) quanto a elementos móveis (*laptops*, PDAs, celulares, etc.). Este artigo apresenta a especificação inicial de execução de uma arquitetura para o Holoparadigma, com o foco voltado para o suporte à pervasividade, sendo esta chamada de PHolo. O PHolo tem por objetivo englobar uma série de serviços considerados essenciais para aplicações pervasivas. Como parte desta proposta é abordada a implementação dos serviços que dão suporte à execução distribuída e à mobilidade na arquitetura. O principal diferencial deste suporte é a abstração criada pela linguagem e pelo ambiente em relação ao usuário, que se preocupa apenas com a programação em alto nível enquanto o ambiente se encarrega de dar o suporte a execução distribuída.

O presente artigo está organizado da seguinte maneira: Na seção 2 é apresentado o Holoparadigma. A proposta do PHolo é descrita na seção 3. Em seguida, na seção 4, são apresentados resultados iniciais obtidos comparando a nova solução com as existentes. Nas seções 5 e 6 são apresentados respectivamente os trabalhos relacionados e as conclusões deste artigo.

2. Holoparadigma

O Holoparadigma é um modelo multiparadigma que possui uma semântica simples e distribuída. Seu principal objetivo é a exploração automática da distribuição. A semântica da linguagem busca diminuir o gap semântico da linguagem em relação ao mundo real.

2.1. Modelo

O Holoparadigma propõe um modelo multiparadigma orientado ao desenvolvimento de software distribuído. Os conceitos apresentados pelo Holoparadigma são implementados na Hololinguagem [9]. Holo explora um mecanismo de coordenação baseado em *blackboards* [10], que possui como unidade de modelagem o *ente* e como unidade de informação o *símbolo*. Um ente elementar (Figura 1a) é organizado em três partes: *interface*, *comportamento* e *história*. Um ente composto (Figura 1b) possui a mesma organização, no entanto, suporta a existência de outros entes na sua composição (entes componentes). Cada ente possui uma história, a qual fica encapsulada no ente e, no caso dos entes compostos, é compartilhada pelos entes componentes. Sendo assim, podem existir vários níveis de encapsulamento da história, porém os entes acessam somente a história em um nível. A composição varia de acordo com a mobilidade dos entes em tempo de execução. A Figura 1c mostra um ente composto de três níveis e exemplifica a história encapsulada.

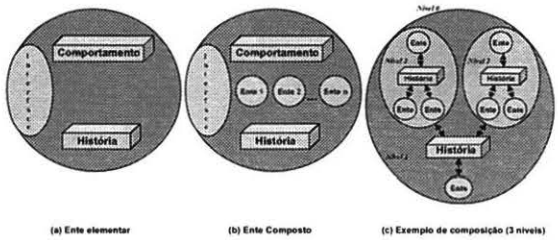


Figura 1. Entes Elementar, Composto e Composição em 3 níveis

No escopo de sistemas distribuídos, um ente pode assumir dois estados de distribuição: centralizado ou distribuído. No estado centralizado todos os entes do sistema modelado estão representados na mesma plataforma. Já no estado distribuído as representações dos entes do sistema modelado estão em mais de uma plataforma. A Figura 2b mostra um possível estado de distribuição para um ente. Neste caso, a história do ente composto atua como uma memória compartilhada distribuída (DSM) entre seus entes componentes.

No Holoparadigma, a mobilidade é a propriedade que um ente possui e permite que este se mova. São considerados dois tipos de mobilidade (lógica e física). A mobilidade lógica relaciona-se com o deslocamento em

nível de modelagem, ou seja, sem considerações sobre a plataforma de execução. Neste contexto, um ente se move quando cruza uma ou mais fronteiras de entes. A mobilidade física relaciona-se com o deslocamento entre nodos de uma arquitetura distribuída. Desta forma, um ente se move quando se desloca de um nodo para outro. A Figura 2a exemplifica uma possível mobilidade lógica no ente apresentado na Figura 1c. Merece atenção o fato de que a mobilidade física não implica obrigatoriamente em mobilidade lógica, ou seja, a ocorrência de um tipo de deslocamento não implica a ocorrência do outro. A Figura 2b mostra um exemplo de mobilidade física sem mobilidade lógica.

2.2. Linguagem

A Hololinguagem é uma linguagem de programação que surgiu para implementar os conceitos propostos pelo modelo. Esta linguagem suporta concorrência, mobilidade e *blackboards* hierárquicos.

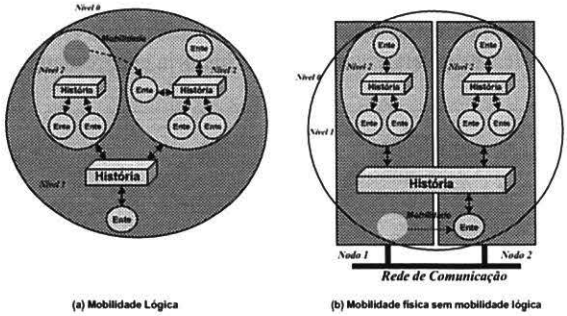


Figura 2. Mobilidade no Holoparadigma

O modelo de coordenação dos entes compostos (Figura 3b) assemelha-se a um *blackboard* [17] (Figura 3a). Neste caso, os Knowledge Sources (KSs) são entes e o blackboard é a história. Existem várias limitações estabelecidas pelo uso da invocação implícita usada nos *blackboards*. O uso da invocação explícita conjuntamente com a invocação implícita é salientado como uma solução para essas limitações. Em Holo, ambos os estilos de invocação são utilizados. Os entes influenciam uns aos outros através da história (invocação implícita), mas também podem trocar informações diretamente através de mensagens (Figura 3b).

2.3. Suporte de execução

O suporte à execução de programas desenvolvidos sobre o Holoparadigma é realizado por uma máquina virtual especialmente desenvolvida, a HoloVM [13]. Como é natural de uma máquina virtual, ela cria uma camada de

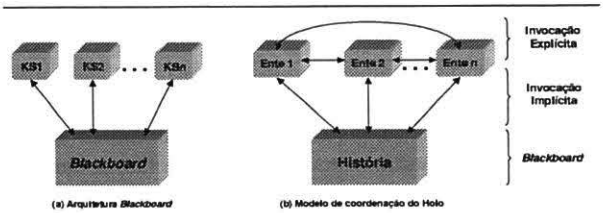


Figura 3. Modelo de coordenação no Holo

abstração entre o programa e o hardware sobre o qual este será executado. Isto permite que programas Holo sejam executados em qualquer plataforma, desde que exista uma versão da HoloVM disponível para a mesma, facilitando a execução distribuída no Holo. A HoloVM oferece um conjunto de instruções específicas para dar suporte às funcionalidades propostas no Holoparadigma.

Atualmente é suportada a execução de um programa utilizando apenas uma HoloVM. Este fato demanda a criação de uma solução que suporte a execução distribuída de programas em Holo. Com o objetivo de prover um suporte inicial foi implementado o History Server (HS), que é um servidor de histórias, permitindo que várias HoloVMs centralizem seu dados em um único lugar. De certa forma, isto resolve o problema, mas não é uma solução robusta, criando um suporte limitado que tem sua principal fraqueza na centralização de dados do programa.

3. Pervasive Holo (PHolo)

O PHolo consiste em uma proposta de arquitetura pervasiva para o ambiente de execução do Holoparadigma. A revisão feita sobre ambientes de execução que possuem propostas semelhantes (Seção 5) permitiu o levantamento dos requisitos para este tipo de arquitetura. O intuito é propor uma arquitetura que adapte o atual suporte executivo (HoloVM), a um ambiente pervasivo, trazendo assim uma série de conceitos desta nova classe de aplicações. Para que fossem feitas experimentações com a execução distribuída de programas Holo, foi implementado um protótipo para um dos elementos que compõem esta arquitetura. Este elemento é chamado *Holo Naming System* (HNS) [11]. Os demais elementos desta arquitetura

3.1. Arquitetura Proposta

O objetivo desta arquitetura é estender o suporte de execução existente no Holoparadigma, agregando novos elementos e serviços. Com isto é criado um ambiente onde aplicações executam de forma distribuída sem que o programador tenha que se preocupar com questões tais como: localização de uma entidade, forma de comunicação

entre HoloVMs, encontrar um HNS ou outra HoloVM, ou como irá distribuir os elementos a serem processados.

Desta forma, a arquitetura do PHolo (Figura 4) é composta por duas camadas: a primeira camada é responsável pela execução de programas. Nela estão a HoloVM e o HNS. A HoloVM executa as aplicações, de forma distribuída, oferecendo todo o suporte à comunicação entre entes sem que isso acarrete em uma complexidade maior no desenvolvimento. Já o HNS tem o controle de todos os entes em execução na aplicação. Este controle é feito através de uma estrutura chamada *DHolotree*, na qual estão contidas as árvores de execução de cada HoloVM. Com esta estrutura e os endereços físicos dos entes, o HNS pode informar para as HoloVMs sobre a localização de um ente, permitindo assim a modificação da organização lógica e/ou física da aplicação em tempo de execução.

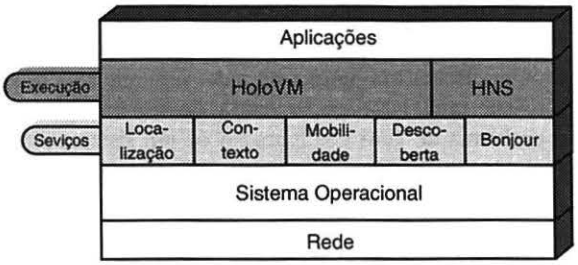


Figura 4. Arquitetura do PHolo.

Na segunda camada da arquitetura estão os serviços do ambiente. Como pode ser observado na Figura 4, os serviços foram colocados abaixo da camada de execução. Isto se deve principalmente ao fato de que todos os serviços pensados para o PHolo se agregam na camada de execução como módulos, que fazem parte da HoloVM e/ou do HNS. Os serviços serão abordados nas seções a seguir.

3.2. Localização Física

Uma questão de bastante interesse em uma aplicação pervasiva são os sistemas de localização. Agregado a este projeto, existe uma iniciativa que objetiva pesquisar o uso de sistemas de localização no Holoparadigma [19]. Uma das principais características do Holoparadigma é sua habilidade para representar a movimentação de entes. Estes por sua vez podem modelar elementos físicos ou lógicos. Dada a estrutura hierárquica de uma aplicação Holo, é possível que, em tempo de execução, um determinado ente passe a compor outro ente, também saindo da composição de um terceiro ente neste processo. É dito por um desenvolvedor que um ente "entra" ou "sai" de outro ente, e esta ação é implementada pelo comando *move* na Hololinguagem.

Como foi comentado anteriormente, esta abstração cria uma ferramenta poderosa para um desenvolvedor, especialmente quando se quer modelar elementos do mundo real. Estas abstrações facilitam a modelagem de ambientes reais, como salas e prédios, de dispositivos fixos dentro da sala, como computadores ou impressoras, bem como de dispositivos móveis, como *notebooks* e PDAs.

Para garantir a modelagem fiel do mundo real, é necessário que as alterações reflitam diretamente na aplicação. Por exemplo, quando um PDA entra em uma sala, o ente da aplicação deve ser movido para dentro do ente que representa a sala na aplicação. Desta forma a aplicação que está executando no PDA terá acesso a informações de contexto bem como aos serviços providos pelo ente da sala.

Apesar de existirem comandos na Hololinguagem através dos quais o programador pode especificar ações de mobilidade, atualmente o programa não tem como saber a sua localização em um mundo real. A idéia é agregar uma extensão na camada de execução que crie o suporte à mobilidade automática de entes, refletindo o comportamento dos elementos que eles simbolizam.

3.3. Contexto

É essencial para aplicações pervasivas que estas tenham mecanismos para aquisição de informações sobre o mundo real. Desta forma é possível utilizar estas informações para adaptar a aplicação às necessidades do usuário. O sistema de localização citado na seção anterior é um exemplo de adaptação guiada por informações de contexto. A modelagem de aplicações refletindo a organização de espaços físicos, juntamente com características do Holoparadigma, como o encapsulamento de contextos e dados, cria um ambiente onde a disponibilidade de informações organizadas de acordo com os contextos aos quais elas pertencem parece bastante natural.

3.4. Tipos de Mobilidade

Quando se está falando de aplicações pervasivas, o termo mobilidade se torna bastante genérico. Da forma como está definido no Holoparadigma, o conceito de mobilidade física pode ficar dúbio. Para facilitar o entendimento deste tópico, serão redefinidas aqui as nomenclaturas para os três tipos de mobilidade considerados:

- **Mobilidade Física**, que consiste na movimentação de um dispositivo, dentro de um espaço físico;
- **Mobilidade Lógica**, ocorre quando um ente se move, neste processo modificando a HoloTree, porém permanecendo na mesma HoloVM;

- **Mobilidade de Código**, que ocorre quando um ente é migrado, durante sua execução, de uma HoloVM para outra. Não necessariamente esta ação dispara uma mobilidade lógica.

O PHolo tem potencial para explorar estes três tipos de mobilidade. Com a solução baseada em sensibilidade à localização, mudanças no ambiente físico podem facilmente refletir em modificações no comportamento da aplicação. Na Figura 5 pode ser visto um exemplo de uma aplicação que utiliza mobilidade lógica. Neste exemplo, duas HoloVMs dividem a execução de uma aplicação que tem por objetivo gerenciar um evento científico.

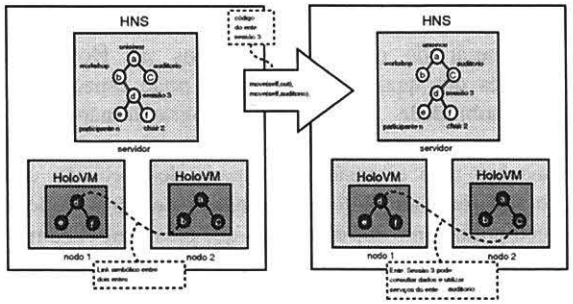


Figura 5. Exemplo de mobilidade lógica.

O **nodo 1** tem a aplicação para o gerenciamento de uma sessão executando em sua HoloVM. O **nodo 2** é responsável pela execução dos entes que mapeiam o espaço físico no qual o evento está acontecendo. No lado esquerdo da figura pode ser vista a aplicação, já em execução, onde o **ente d** compõe o **ente b**. Como estes não se encontram na mesma máquina, é criada uma ligação simbólica entre os dois entes. Desta forma o ambiente de execução mantém o controle desta situação. Caso os dois entes queiram executar uma ação, ou fazer acesso à história um do outro, a HoloVM consulta a localização do ente remoto no HNS e envia uma mensagem com a requisição do serviço para a outra HoloVM. No passo seguinte de execução, o **ente d** executa duas instruções *move* que resultam em uma mobilidade lógica deste, do **ente b** para o **ente c**. Com esta ação, o **ente d** passa a compor o ente responsável pelo auditório, tendo portanto acesso aos dados de contexto existentes nele, bem como aos serviços. Desta forma, pode-se dizer que o PHolo implementa o que é chamado de TDS (Sigla em Inglês para Sistema Verdadeiramente Distribuído). Um sistema TDS é aquele onde um indivíduo pode utilizar os serviços do ambiente sem perceber de que nem todos os recursos estão disponíveis localmente.

A mobilidade de código [12], não é um tópico recente de pesquisa, e já existe uma boa classificação das soluções possíveis para a questão. Atualmente o Holoparadigma não possui suporte a este tipo de mobilidade, sendo este um dos objetivos do trabalho. Dentro das soluções existentes, escolheu-se seguir a linha chamada de mobilidade forte. Esta se divide em duas possibilidades: uma delas é a *migração de código*, que consiste em ser capaz de parar um fluxo de execução, encapsular todos os dados do fluxo, enviar para outra máquina e colocá-lo novamente em execução. Outra possibilidade é a *clonagem remota*, onde uma instância do código é clonada e enviada para a máquina cliente pronta para executar.

3.5. Descoberta de Serviços

A descoberta de serviços é uma característica importante para aplicações pervasivas, pois permite a cooperação entre dispositivos e diminui o custo com configuração [20]. No PHolo a descoberta de serviços engloba duas funcionalidades importantes. A primeira é permitir que HoloVMs descubram servidores HNS e mesmo outras HoloVMs na mesma rede, sem que seja necessário existir uma intervenção direta do usuário para isso. O problema de auto configuração de um sistema já recebeu algumas soluções, como: SLP [16], Jini [26], UPnp [7] e Bonjour [1]. Para o PHolo, chegou-se à conclusão de que a abordagem mais interessante é o Bonjour.

O Bonjour, também conhecido como Rendesvouz ou zeroconf, é uma solução para configuração automática de redes e descoberta de serviços, desenvolvida pela Apple. O Bonjour utiliza padrões abertos, como *DNS Multicast* e *DNS Service Discovery*. Outro fato importante é que o próprio Bonjour é de código livre, tendo seu código fonte liberado pela Apple. Ele também já se encontra portado para algumas arquiteturas.

A segunda funcionalidade é o anúncio e a descoberta de serviços desempenhados por entes de uma aplicação. A idéia é que, utilizando-se de premissas novas na linguagem, seja possível anunciar e consultar serviços prestados por um ente. Neste caso o HNS desempenha o papel de centralizador das informações, sendo responsável por conhecer os serviços anunciados, fornecendo mecanismos eficientes de consulta.

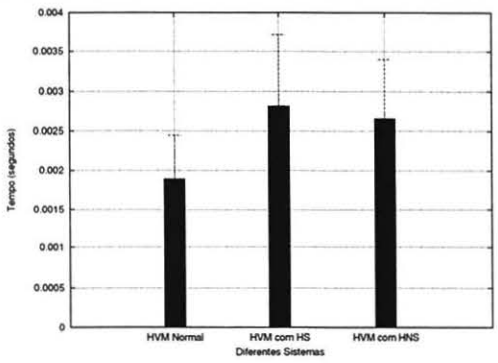
4. Desenvolvimento do HNS

O objetivo do HNS é prover um serviço que possibilite a execução de programas Holo distribuídos entre várias HoloVMs. Para isso, é necessário levar em consideração dois problemas principais. O primeiro é a localização de entes em um ambiente de execução distribuído. O segundo problema é, partindo da informação sobre a localização de um ente, prover uma camada de execução para que duas HoloVMs possam dar suporte ao uso de todos os recursos propostos pelo Holoparadigma, que envolvam

a interação entre entes (acessos à história e execução de ações). Um protótipo desta solução foi implementado [11], e os resultados de alguns experimentos realizados são apresentados na próxima seção.

Para que fosse possível ter uma primeira avaliação da solução proposta, foram feitos alguns testes comparativos. As aplicações foram executadas em três ambientes diferentes: uma HoloVM sem nenhum suporte à distribuição, uma com suporte a HS e outra com suporte ao HNS. A plataforma utilizada foi um AMD Athlon(tm) XP 2200+.

O primeiro teste apresentado teve por objetivo analisar o impacto que os diferentes suportes à distribuição têm na inicialização da HoloVM (Figura 6). Isso foi feito utilizando um programa em Holo que executa apenas a criação de um ente, nenhuma outra operação. Esta análise se torna pertinente, pois considera o tempo de processamento gasto a cada nova conexão de usuário com o ambiente. Considerando que a versão distribuída da HoloVM conta com rotinas de inicialização, um sobrecusto de execução não existente na versão seqüencial é acrescentado.



HoloVM	Média	Desvio Padrão
Normal	0.001890	0.000559
Com HS	0.002818	0.000896
Com HNS	0.002657	0.000746

Figura 6. Tempos de inicialização da HoloVM

Como pode ser visto, os dois suportes (HS e HNS) têm impactos semelhantes nos tempos de inicialização da HoloVM. Isso tem explicação principalmente no fato de que todo ente criado na HoloVM gera uma mensagem, que é enviada para o servidor informando sua existência. É importante ressaltar o desempenho obtido pela HoloVM com HNS, já que ela em particular possui um servidor *multithread* interno. Este servidor deve ser inicializado junto com a HoloVM. Mesmo assim o seu tempo médio de inicialização foi bastante próximo à HoloVM com HS.

Resultado semelhante foi obtido no teste com a escrita de tuplas na história (Figura 7). Este experimento reflete uma aplicação típica do Holoparadigma, onde entes acessam informações (em leitura e/ou escrita) na história de entes remotos. Para este teste foi utilizado um programa que escreve 100 tuplas na história de um ente em execução em outra HoloVM. Foi feita uma variação na aridade das tuplas para verificar se a queda no desempenho seria significativa.

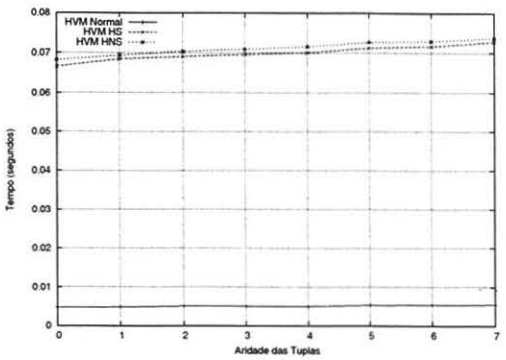


Figura 7. Escrita de tuplas na história

Este teste é particularmente importante pois o gerenciamento da história é uma especialidade do HS. No entanto, pode ser notado que ambas soluções obtiveram tempos bastante semelhantes. Outro fato que torna este resultado relevante vem da observação de que o HNS utiliza uma arquitetura mais complexa e que envolve um maior número de variáveis no seu gerenciamento.

O último teste a ser mostrado aqui avaliou o desempenho das diferentes soluções ao executar criações massivas de entes (Figura 8). Para isso foi utilizado um programa em Holo que cria um determinado número de entes. Sobre este número foi feita a variação mostrada no gráfico.

Neste teste é avaliado o desempenho das diferentes estratégias de comunicação adotadas pelas duas soluções. Aqui a solução do HS levou vantagem em todas as configurações. Tal resultado era esperado, dada a maior complexidade do serviço implementado pelo HNS em comparação com o HS.

Em geral, o desempenho do HNS foi satisfatório, se levando em consideração o suporte necessário para o seu funcionamento, bem como o aumento na complexidade do gerenciamento das aplicações, em relação à considerável melhora nos serviços oferecidos pelo HNS em relação ao HS. Contudo, esforços estão sendo feitos para otimizar os serviços oferecidos pelo HNS.

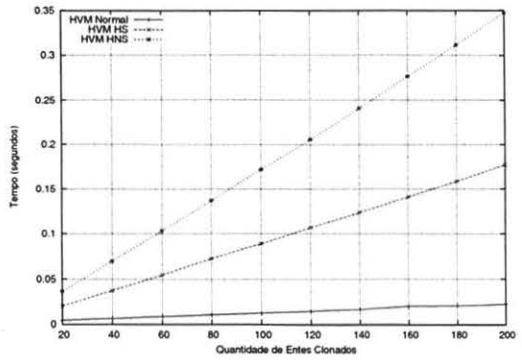


Figura 8. Tempos de clonagem de entes

5. Trabalhos Relacionados

Para a formulação desta proposta foram estudados alguns sistemas que oferecem suporte à criação de aplicações pervasivas. Este estudo abrangente foi essencial para o entendimento da área, que é recente e ainda não possui um consenso sobre seus requisitos básicos.

Um problema importante nesta área está na forma de modelagem do mundo real. Segundo Robert Grin, idealizador do one.world [14, 15, 4], os sistemas que utilizam objetos distribuídos encapsulam dados e funcionalidades em uma única abstração, que é chamada de objeto. Desta forma, utiliza-se em aplicações distribuídas uma abstração criada para aplicações que executam em um único nodo. Esta abstração é limitada na forma como os dados podem ser utilizados, complicando o compartilhamento, busca e filtro dos dados. Este fato fez com que fosse criada uma nova forma de modelagem para o one.world, que na verdade é bastante semelhante à abstração criada pelo Holoparadigma. No one.world os *ambientes* dão estrutura à aplicação e são organizados hierarquicamente, assim como os entes. É usada a idéia de *componente*, onde são implementadas as funcionalidades de cada ambiente, assim como o comportamento para o Holo. Por último, o one.world possui um *espaço de tuplas* para cada ambiente armazenar seus dados, de forma semelhante a história no Holo. O projeto ISAM [28, 6] também faz uso do Holoparadima para a modelagem de suas aplicações.

Uma característica importante nestes sistemas é a sensibilidade ao contexto. A aplicação deve estar sempre atualizada sobre o que acontece no ambiente em que está inserida, para que possa se adaptar a qualquer modificação que venha a ocorrer. Nos sistemas estudados é comum criar um serviço para tratar o gerenciamento de contextos. Contudo, foi a solução adotada no Gaia [21, 22, 2] que chamou a atenção para a possibilidade de utilizar recursos já existentes no Holo para esta tarefa. No Gaia, o seu serviço de contexto armazena os eventos do ambiente em tuplas de aridade quatro que são compostas por: tipo do contexto, assunto, relator e objeto. Uma abordagem como esta pode ser vantajosa para o PHolo, pois as tuplas são um formato familiar no Holo, podendo ficar armazenadas na história de cada ente.

A adaptabilidade é tida como um fator importante em uma aplicação pervasiva, que permite modificações de comportamento e mesmo de lógica em tempo de execução. Esta é uma questão que tem ligação direta com o suporte à mobilidade de código na aplicação e com a sensibilidade ao contexto. A mobilidade de código permite que novas funcionalidades sejam agregadas na aplicação sem que ela tenha sido compilada previamente com aquele código. Isto também tem uma aplicação direta, por exemplo, no processamento da aplicação, que pode ser direcionado para um nodo com maior poder computacional e poupar os recursos (CPU e bateria) de um PDA. A solução mais comum na implementação dos sistemas estudados é a utilização da linguagem Java e de sua máquina virtual, que é o caso dos sistemas one.world, Aura [25, 5], ISAM e Gaia. Esta abordagem dificulta a implementação da mobilidade do código, já que não se possui o controle da máquina virtual. No PHolo a utilização da HoloVM oferece o total controle sobre a aplicação em execução, o que facilita a implementação da mobilidade de código.

6. Conclusão

Neste trabalho foi apresentada uma arquitetura para criar o suporte à execução de aplicações pervasivas utilizando o Holoparadigma. Esta arquitetura de software recebeu o nome de PHolo. Este trabalho é proposto com base nos princípios do Holoparadigma. Desta forma é possível tirar proveito da sua forma de abstração para modelar as aplicações do ambiente. Com isto obtém-se uma plataforma onde a criação de aplicações distribuídas é feita abstraindo boa parte da complexidade envolvida.

O principal módulo desta arquitetura é o Holo Naming System (HNS). Para este módulo foi desenvolvido um protótipo contendo todas as funcionalidades necessárias para executar aplicações Holo de forma distribuída. Foi feita uma avaliação experimental deste protótipo, através da qual foi possível concluir que o HNS possui um desempenho satisfatório, dada a complexidade inserida na aplicação, em comparação a solução existente (HS).

A revisão feita sobre outros sistemas permitiu que fossem levantados os requisitos para um sistema que suporte a execução de aplicações pervasivas. A partir destes requisitos foram definidos os serviços que compõem a arquitetura proposta.

Referências

[1] Bonjour - networking, simplified. Disponível em: `http://www.apple.com/macosx/features/bonjour/`. Acesso em Junho de 2005.

[2] Gaia - active spaces for ubiquitous computing. Disponível em: `http://gaia.cs.uiuc.edu/`. Acesso em Junho de 2005.

[3] The official bluetooth website. how it works. Disponível em: `http://www.bluetooth.com/`. Acesso em Abril de 2005.

[4] one.world. Disponível em: `http://www.cs.nyu.edu/rgrimm/one.world/`. Acesso em Junho de 2005.

[5] Project aura - distraction-free ubiquitous computing. Disponível em: `http://www-2.cs.cmu.edu/~aura/`. Acesso em Junho de 2005.

[6] Project ISAM - infra-estrutura de suporte às aplicações móveis. Disponível em: `http://www.inf.ufrgs.br/~isam/`. Acesso em Junho de 2005.

[7] Upnp device architecture 1.0. Disponível em: `http://www.upnp.org/resources/documents.asp`. Acesso em Junho de 2005.

[8] J. L. V. Barbosa. *Holoparadigma: Um Modelo Multiparadigma Orientado ao Desenvolvimento de Software Distribuído*. Tese (doutorado em ciência da computação), Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002. 213p.

[9] J. L. V. Barbosa and C. F. R. Geyer. Uma linguagem multiparadigma orientada ao desenvolvimento de software distribuído. *In Proceedings of the V Simpósio Brasileiro de Linguagens de Programação (SBLP)*, maio 2001.

[10] J. L. V. Barbosa, A. C. Yamin, P. K. Vargas, D. N. Ferrari, A. E. Schaeffer, and C. F. R. Geyer. Using mobility and blackboards to support a multiparadigm model oriented to distributed processing. In *13th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2001)*, pages 187–194, Pirenópolis, September 2001.

[11] D. T. Bonatto, F. Kellermann, J. L. V. Barbosa, J. D. G. Ramos, and G. G. H. Cavalheiro. Estratégias para localização em um ambiente de computação móvel. In *XXXII Seminário Integrado de Software e Hardware (SEMISH)*, São Leopoldo, 2005. Unisinos. Aceito para publicação.

[12] A. Fuggetta, G. P. Picco, and G. Vigna. Understanding code mobility. *IEEE Transactions on Software Engineering*, 24(5):342–361, 1998.

[13] A. S. Garzão and J. L. V. Barbosa. Uma máquina virtual com suporte à concorrência, mobilidade e blackboards. In *XXIX Conferência Latinoamericana de Informática (CLEI)*, volume 24, La Paz, 2003.

[14] R. Grimm, J. Davis, B. Hendrickson, E. Lemar, A. MacBeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble, and D. Wetherall. Programming for pervasive computing environments. *In Proceedings of the 18th ACM Symposium on Operating Systems Principle*, 2001.

[15] R. Grimm, J. Davis, B. Hendrickson, E. Lemar, A. MacBeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble, and D. Wetherall. System support for pervasive applications. *In Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, Elmau, Germany.:147–151, 2001.

[16] E. Guttman. Service location protocol: Automatic discovery of ip network services. *IEEE Internet Computing*, 3(4):71–80, 1999.

[17] H. P. Nii. Blackboard systems: the blackboard model of problem solving and the evolution of blackboard architectures. pages 447–474, 1995.

[18] W.-. Planet. Maximizing wireless lan performance. Disponível em: `http://www.80211-planet.com/`. Acesso em Abril de 2005.

[19] J. D. G. Ramos. Integração de um sistema de localização no MHolo. Comunicação pessoal feita em Junho de 2005.

[20] I. Richard, G.G. Service advertisement and discovery: Enabling universal device cooperation. *IEEE Internet Computing*, 4(5):18–26, Sep./Oct. 2000.

[21] M. Roman and R. H. Campbell. Gaia: enabling active spaces. In *EW 9: Proceedings of the 9th workshop on ACM SIGOPS European workshop*, pages 229–234, New York, 2000. ACM Press.

[22] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. Campbell, and K. Nahrstedt. A middleware infrastructure for active spaces. *IEEE Pervasive Computing*, 1(4):74–83, 2002.

[23] D. Saha and A. Mukherjee. Pervasive computing: A paradigm for the 21st century. *IEEE Computer*, 36(3):25–31, 2003.

[24] M. Satyanarayanan. Pervasive computing: vision and challenges. *Personal Communications, IEEE [see also IEEE Wireless Communications]*, 8(4):10–17, 2001.

[25] J. P. Sousa and D. Garlan. Aura: an architectural framework for user mobility in ubiquitous computing environments. In *WICAS3: Proceedings of the IFIP 17th World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture*, pages 29–43, Deventer, The Netherlands, 2002. Kluwer, B.V.

[26] Sun Microsystems Inc. Jini technology architectural overview. Technical report, Sun Microsystems, Inc., 1999. Disponível em: `http://wwws.sun.com/software/jini/whitepapers/architecture.html`.

[27] M. Weiser. The computer for the 21st century. *Scientific American*, Setembro 1991.

[28] A. C. Yamin, J. V. Barbosa, I. Augustin, L. C. da Silva, R. Real, C. Geyer, and G. Cavalheiro. Towards Merging Context-Aware, Mobile and Grid Computing. *International Journal of High Performance Computing Applications*, 17(2):191–203, 2003.