Mecanismos de Suporte ao Escalonamento em Sistemas com Objetos Distribuídos Java

Luciano da Silva[1], Adenauer Yamin[1,2], Iara Augustin[1,3], Edvar Bergmann Araujo[1], Cláudio Geyer[1]

[1] Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS)
Av. Bento Gonçalves, 9500, Porto Alegre, RS, Brasil
{lucc, adenauer, august, edvar, geyer}@inf.ufrgs.br

[2] Escola de Informática, Universidade Católica de Pelotas (UCPel)
R. Félix da Cunha, 412, Pelotas, RS, Brasil
{adenauer}@atlas.ucpel.tche.br

[3] Departamento de Eletrônica e Computação, Universidade Federal de Santa Maria (UFSM)
Campus Universitário, Santa Maria, RS, Brasil
august@inf.ufsm.br

Resumo —
Este artigo propõe recursos para o escalonamento de objetos distribuídos. Para tanto, são definidos dois mecanismos tidos como centrais para tal: um primeiro que irá realizar a instanciação remota de objetos Java, e um segundo que terá a cargo a coleta de informações dinâmicas sobre a execução da aplicação distribuída, tais como: uso de processador, uso de memória e perfil de comunicação entre os objetos. Os mecanismos serão integrados com a API RMI de Java, buscando preservar a natureza da OO – Orientação a Objetos – na construção de aplicações distribuídas, e conseqüentemente a compatibilidade com a semântica nativa da linguagem Java. A pesquisa está inserida no contexto da proposta EXEHDA – *Execution Environment for High Distributed Applications* – em desenvolvimento na Universidade Federal do Rio Grande do Sul, e tem por objetivo dar suporte a execução de aplicações móveis distribuídas desenvolvidas utilizando o modelo Holoparadigma.

Palavras-Chave — Escalonamento na Programação Orientada à Objetos, Escalonamento em Sistemas Distribuídos e Paralelos e Programação Distribuída e Paralela com Java.

Abstract —
This article proposes support resources for distributed object scheduling. Two mechanisms, understood as essential for such task, are defined: a first one that implements remote instantiation of Java objects, and a second one that collects dynamic information about the execution of the distributed application. Additionally, the support for optimized communication and the construction of inter-objects communication profiles complement the proposal. The mechanisms will be integrated with Java RMI API, aiming to preserve the nature of the OO – Object Oriented model - in the construction of distributed applications and, consequently, the compatibility with the native semantics of the Java language. The research is inserted in the context of the EXEHDA - Execution Environment for High Distributed Applications - proposal in development in the Universidade Federal do the Rio Grande do Sul, and has as objective to provide means of supporting the execution of distributed mobile applications developed using the Holoparadigma model.

Keywords — Distributed Objects Scheduling, Distributed and Parallel Processing, Java.

I. INTRODUÇÃO

O preço pago pela portabilidade da linguagem Java é o desempenho da execução dos seus programas. Numa primeira análise pode parecer paradoxal a escolha de Java como linguagem para uso em processamento onde o desempenho é um aspecto a ser considerado.

No entanto, a utilização de Java em processamento de alto desempenho não constitui uma iniciativa isolada da proposta EXEHDA (*Execution Environment for High Distributed Applications*) [YAM 99], mas reflete uma tendência de grupos de pesquisa internacionais. Rapidez de aprendizado, somada ao fato do modelo de objetos ser adequado ao tratamento de problemas distribuídos e paralelos e a ausência de ponteiros, o que facilita a otimização de código pelos compiladores, são as justificativas mais comuns para esta tendência [FOS 99]. Diversos estudos no sentido de tornar a execução de programas Java mais competitiva em relação àquelas decorrentes de linguagens tradicionais vêm sendo desenvolvidos. Tais estudos variam desde a utilização de compiladores Java nativos [GET 98, PHI 97, NIE 00] e [YEL 98], otimização dos *bytecodes* gerados [SUN 01b] e [CIE 97], até a otimização do mecanismo de serialização usado pela API RMI de Java [PHI 97] e [NIE 00]. Os resultados atingidos por tais esforços são satisfatórios e bastante animadores [GRA 01].

Tendo em vista o uso de arquiteturas de memória distribuída, onde os custos de comunicação são heterogêneos, considerar os aspectos de localidade é crucial para o desempenho de uma aplicação distribuída. Particularmente, para um ambiente de programação distribuída baseado em uma linguagem orientada a objetos, objetos que compartilham dados ou que apresentam um perfil de intensa troca de mensagens precisam ficar o mais "próximo possível" (ou no mesmo nodo processador, ou em nodos conectados através de um canal rápido de comunicação).

É esperado que os futuros ambientes de execução para aplicações distribuídas contemplem obrigatoriamente suporte tanto para a mobilidade lógica como física, de processos e/ou recursos [AUG 00]. Estes ambientes vão ser caracterizados por nodos de processamento bastante heterogêneos, os quais serão interconectados por redes sujeitas a freqüentes flutuações nos serviços fornecidos. Como conseqüência deste cenário, o projeto de software para ambientes móveis é complexo, seus componentes são variáveis no tempo e no espaço em termos de conectividade, portabilidade e mobilidade. Existem, portanto, requerimentos emergindo para uma nova classe de aplicações projetadas especificamente para este ambiente dinâmico. Esta nova classe de aplicações tem sido referenciada na literatura de muitas formas: *environment-aware, network-aware, resource-aware, context-aware*. A característica comum entre elas é a capacidade das aplicações adaptarem sua funcionalidade às condições dos recursos envolvidos nos diferentes momentos da execução. Desta forma, as aplicações devem obter informações sobre o estado corrente de seu contexto de execução a fim de se adaptar.

A linguagem Java [SUN 01a] oferece uma solução para o problema de portabilidade de código, ao mesmo tempo em que apresenta um modelo de programação de Objetos Distribuídos, chamado RMI [FAR 98] – *Remote Method Invocation* , bastante conhecido. Java, entretanto, não fornece mecanismos para obtenção de informações contextuais, como estatísticas sobre os recursos utilizados em cada nodo do sistema distribuído e nem para alocação de objetos no sistema distribuído, não provendo, deste modo, um suporte ao balanceamento de carga nativo à linguagem.

Para eliminar parte dessas restrições e fornecer suporte ao escalonador de tarefas – parte integrante da proposta EXEHDA, foram projetados dois mecanismos: (*i*) instanciação remota de objetos e (*ii*) coleta de informações dinâmicas sobre a execução. A proposta EXEHDA integra o projeto ISAM (Infra-estrutura de Suporte às Aplicações Móveis), o qual propõe uma arquitetura de software que simplifica a tarefa de implementação de aplicações móveis distribuídas. O objetivo é conceber um ambiente de desenvolvimento e execução no qual todos os componentes, mesmo os básicos, estão comprometidos com a premissa de elevada adaptabilidade. O escalonamento é

projetado como uma estratégia central de adaptação, e por conseqüência de aumento de desempenho.

O texto está organizado da maneira como segue. Na seção II, caracteriza-se a arquitetura para qual os mecanismos propostos são voltados. A seção III introduz a proposta ISAM para computação móvel. Na seção IV, apresentam-se os pressupostos do escalonamento projetado para a ISAM, e por sua vez os mecanismos de suporte a este escalonamento são detalhados na seção V. Faz-se na seção VI uma análise dos trabalhos relacionados, e na seção VI apresentam-se as conclusões.

II. A ARQUITETURA DESTINO

A computação móvel abrange uma faixa de cenários, os quais têm diferentes requerimentos no sistema de suporte à execução. A princípio, uma categorização distingue entre dois cenários básicos:

- **infra-estruturado** - cenário composto pela presença de uma rede fixa onde alguns *hosts*, referenciados como estações-base, constituem os pontos de acesso para os *hosts* móveis;
- **ad-hoc** - cenário dinâmico composto somente por *hosts* móveis (sem o suporte dado por uma rede fixa). A topologia resultante é altamente variável, constituída a partir das intersecções das áreas (células) de abrangência dos *hosts* móveis.

Considera-se que, para o desenvolvimento de aplicações distribuídas mais avançadas, é necessário que os *hosts* móveis usufruam a infra-estrutura da rede fixa existente, e possam se beneficiar de ambientes como o oferecido pela Internet. Desta forma, o modelo de rede adotado é o de uma rede móvel infra-estruturada. Este modelo é refletido nos elementos básicos do ambiente de execução do sistema apresentado na figura 1, tais como: *hosts* móveis, estação-base (servidor com suporte para comunicação sem fio), e demais componentes de um sistema distribuído heterogêneo.

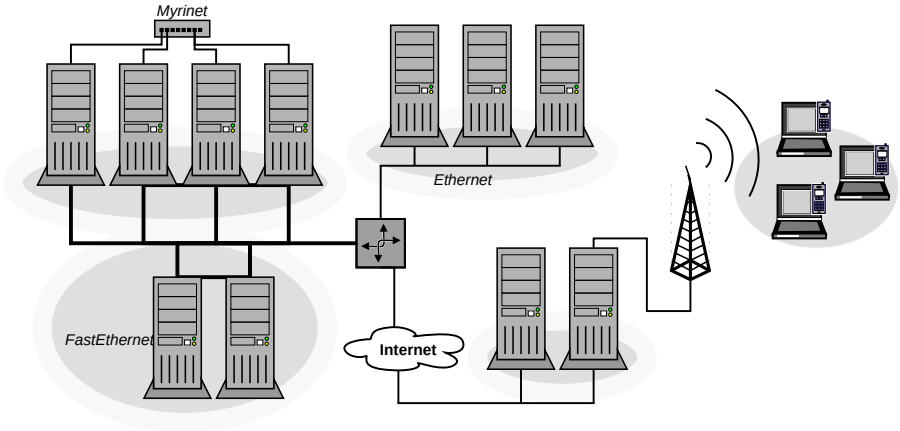

FIGURA 1 – Arquitetura Distribuída Destino (ADD)

A crescente disponibilidade de facilidades de comunicação tem deslocado as aplicações da computação móvel de uma perspectiva de uso pessoal para outras mais avançadas de uso corporativo. Exatamente este domínio de aplicações constitui o escopo de interesse da arquitetura ISAM, ilustrada na figura 2. A arquitetura proposta é organizada em camadas com níveis diferenciados de abstração.

A camada superior (SUP) constitui a aplicação do usuário desenvolvida com a HoloLinguagem [BAR 01a], uma linguagem de programação que integra os paradigmas em lógica, imperativo, orientado a objetos e distribuído. O modelo de coordenação é baseado no conceito de memória logicamente compartilhada e suporta invocações implícitas (*blackboard*) e explícitas (mensagens) [BAR 01b]. Este modelo é apropriado a o ISAM porque contempla o desacoplamento

espacial e temporal da comunicação e sincronização, propriedades importantes para a computação móvel [PIC 99]. A Hololinguagem é executada a partir de um mapeamento feito para Java [BAR 01c]. A camada inferior (INF) é composta dos sistemas de infra-estrutura distribuída pré-existentes, tais como sistemas de rede móvel, sistemas operacionais nativos e a Máquina Virtual Java.

A camada intermediária (INTERM) é o núcleo funcional da arquitetura ISAM, e é fornecida em dois níveis de abstração. O primeiro nível é composto por três módulos de suporte à aplicação: Ambiente Virtual do Usuário, Gerenciamento da Mobilidade Física, Gerenciamento de Recursos. No segundo nível da camada intermediária estão os serviços básicos ao suporte da execução no ISAM. E é neste nível, que se inserem os mecanismos de suporte ao escalonamento detalhados neste artigo.

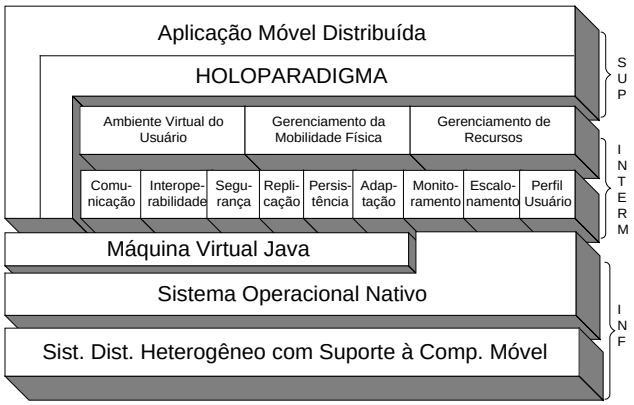

FIGURA 2 - A Arquitetura ISAM

III. O ESCALONAMENTO PROPOSTO PARA O ISAM

Uma aplicação móvel deve oferecer resultados o mais rapidamente possível ao seu usuário. Isto é uma exigência implícita à situação de mobilidade do usuário: equipamento com pouca autonomia (operado a bateria), custo de uso da rede móvel, inserção no tempo/espaço do contexto da tomada de decisão (e.g. reuniões com clientes), etc.

O *framework* EXEHDA que fornece o suporte de escalonamento para a arquitetura ISAM tem como meta de projeto ser flexível e extensível [YAM 01, SIL 01]. O ambiente móvel se caracteriza com aplicações com elevada dinamicidade de execução. As propostas de balanceamento de carga difusas [COR 99] se mostram adequadas para este tipo de aplicação. Inspirado nestas, as características mais significativas do escalonamento ISAM são:

- sua operação ocorre sem exigir a alteração do sistema operacional. Isto potencializa a portabilidade;
- pode suportar tanto execuções paralelas como distribuídas. Para tal, interfaces de programação para comunicação interprocessos, tanto síncronas quanto assíncronas, são disponibilizadas;
- não está comprometido com uma heurística de escalonamento em particular. Ao contrário, disponibiliza facilidades para que novas heurísticas sejam implementadas;
- a heurística a ser utilizada é selecionada e/ou contextualizada por usuário e aplicação;
- os componentes que tomam decisão são replicados, e são capazes de atividades autônomas e assíncronas;
- as metas de escalonamento são perseguidas em escopos. Cada componente que toma decisão escalona serviços no seu domínio;
- uso intensivo de registro histórico como auxiliar na tomada de decisão.

O modelo de escalonamento adotado utiliza uma organização fisicamente distribuída e cooperativa [CAS 88]. No que diz respeito às estratégias para maximização do desempenho, o escalonamento no ISAM utiliza as seguintes:

- balanceamento de carga nos nodos responsáveis pelo processamento;
- localização dos recursos (software e hardware) mais próximos (reduzir custo de comunicação);
- emprego de replicação de serviços e de dados;
- disponibilização antecipada, por usuário, da demanda de componentes das aplicações e dos dados;
- otimização no volume de comunicações, utilizando transferências de contextos e componentes de aplicação personalizadas por usuário;
- monitoração da comunicação praticada pelos componentes das aplicações em execução, com intuito de otimizar aspectos de mapeamento.

O emprego destes procedimentos fica potencializado pela alternância do ponto de conexão dos equipamentos móveis no contexto da rede. Comportamento este inerente à computação móvel. Pelas suas atribuições, além da consideração de custos de comunicação e balanceamento de carga, o escalonamento no ISAM atua de forma intensiva sobre aspectos de replicação e migração. Uma discussão neste sentido pode ser encontrada nos trabalhos [FER 00, AHM 98].

À medida que o usuário interage com o sistema, seu comportamento é monitorado e o seu perfil é definido. Esta informação é utilizada pelo ambiente de escalonamento para tomadas de decisão em diferentes situações (vide figura 3).

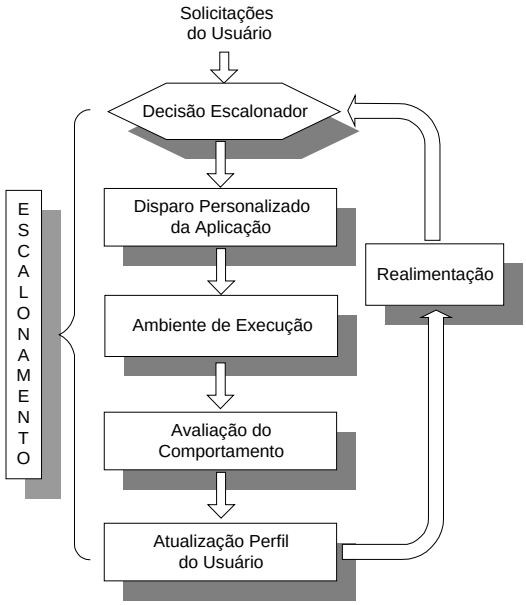

FIGURA 3 – Aprendizado no Escalonamento ISAM

Nesta proposta, o escalonador emprega uma abordagem estocástica com aprendizado por reforço, na qual são construídas correlações estatísticas entre o usuário, o comportamento das suas aplicações e o ambiente de execução. Trabalho neste mesmo sentido, voltado para o escalonamento de aplicações paralelas em multiprocessadores, pode ser encontrado em [ZOM 98]. Um dos principais objetivos de dotar o escalonamento com uma estratégia de aprendizado por reforço é viabilizar uma instanciação otimizada e antecipada de recursos nos *hosts* do sistema.

A mobilidade física do usuário trás outra questão: a variabilidade no contexto de execução da aplicação (recursos, serviços, dados, etc.) devido a dinamicidade da rede móvel e da

alteração dos pontos de contato que determinam contextos diferentes de acesso. Para tratar dos problemas introduzidos pela variabilidade da localização do usuário, os autores sugerem a adoção do conceito de adaptação [KAT 94, DAV 97]. Porém, diferente dos sistemas analisados, o modelo ISAM propõe usar o escalonador como estratégia de adaptação implícita. Para se adaptar o sistema necessita obter informações de estado dos recursos que utiliza. Com es te objetivo, foram projetados alguns mecanismos que atendam às exigências da arquitetura, descritos na seqüência.

IV. OS MECANISMOS DE SUPORTE AO ESCALONAMENTO

O suporte ao escalonamento de objetos distribuídos no EXEHDA (vide figura 4) está fundamentado em quatro mecanismos: (*i*) primitivas otimizadas de comunicação, (*ii*) métricas para caracterização da *workload* dos hosts, (*iii*) primitivas de instanciação remota e migração de objetos e a (*iv*) construção de perfis de comunicaçã o entre objetos. Os principais aspectos destes mecanismos serão descritos a seguir.

A. Primitivas Otimizadas de Comunicação

O *deamon EXEd* está projetado para comportar a operação simultânea com a camada de transporte das diversas tecnologias de rede que forma m o sistema distribuído (com o objetivo de otimizar as comunicações). Assim, na comunicação entre sistemas de alta latência, sujeitos a perdas de mensagens freqüentes, automaticamente será empregado um protocolo com controle de fluxo (como TCP/IP) ao custo de um maior *overhead*, enquanto que na comunicação entre nodos de uma rede local, ou no caso ótimo de redes de alto velocidade (e.g. *SCI* ou *Myrinet*), pode-se empregar um protocolo nativo com uma política mais relaxada de controle de erros e assim ser dimi nuído o *overhead*, tornando a comunicação mais eficiente.

O emprego de uma arquitetura em camadas, enquanto recomendável do ponto de vista de engenharia de software por simplificar o emprego de técnicas de estruturação, pode, devido a necessidade de repeti das copias dos *buffers* de comunicação entre as camadas, comprometer o desempenho do sistema.

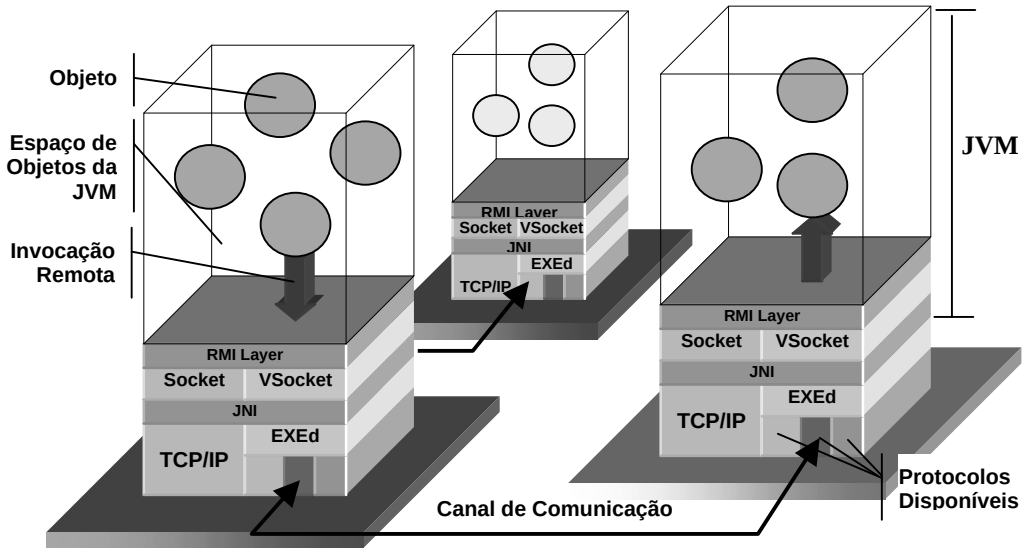

FIGURA 4 – Arquitetura de Suporte ao Escalonamento no EXEHDA

Técnicas de otimização como algoritmos *zero-copy* [VAH 96] são, portanto, altamente desejáveis. O *EXEd* implementa no contexto da camada de comunicação o comportamento *zero-copy* pelo emprego de um segmento de memória compartilhada entre os processos [XXX 99], de forma a minimizar os efeitos da modelagem em camadas sem, no entanto,

sacrificar a estruturação do software. Outras soluções no nível do Sistema Operacional (eg. remapeamento de páginas de memória no *kernel* do S.O.) [XXX 99] foram descartadas por aspectos de segurança e portabilidade.

B. Caracterização da Workload nos Hosts do Sistema

O segundo mecanismo proposto enfoca o fornecimento de subsídios para decisões de alocação (ou migração) de objetos dentro do sistema distribuído a partir da caracterização de índices que descrevem a carga em cada *host* e a qualidade dos canais que os interconectam.

A construção de tais índices de avaliação coloca -se como uma ferramenta de explicitação da heterogeneidade do ambiente distribuído. Esta heterogeneidade pode decorrer diretamente de diferenciações nos recursos de hardware e software disponib ilizados ou ser um resultado da utilização não homogênea dos mesmos. Portanto, a avaliação do poder de processamento disponível um sistema distribuído envolve a consideração de parâmetros tanto estáticos como dinâmicos.

B.1 Parâmetros estáticos

Os parâmetros e státicos descrevem, de certa forma, o comportamento ótimo ou o potencial computacional máximo, ao mesmo tempo que caracterizam, num primeiro nível, a heterogeneidade do sistema distribuído. Os parâmetros utilizados na proposta EXEHDA são:

- **Poder de Processamento**: a avaliação do poder de processamento dos nodos será feito através de *benchmarks* [XXX 99] no momento que os mesmos forem cadastrados para comporem a Arquitetura Distribuída Destino (figura 1);
- **Quantidade de Memória** : valor diretamente decorrente da quantidade de memória RAM disponível no nodo;
- **Conectividade entre os Nodos** : esse parâmetro procura descrever a qualidade de conexão de um nodo para os demais nodos do sistema. Será utilizada a velocidade nominal da infraestrutura de conexão (rede) como base.

Os três primeiros itens são associados aos nodos. Já a conectividade é melhor caracterizada empregando-se uma estrutura hierárquica, de forma a caracterizar nodos pertencentes a um mesmo segmento de rede (conexão direta), bem como as conexões entre os diversos segmentos que compõem o sistema distribuído. Pela utilização de tal estrutura, minimiza -se o armazenamento de informações redundantes sobre a conectividade dos nodos. Na proposta EXEHDA, o escalonamento privilegia aspectos de localidade (menor custo de comunicação) no mapeamento dos objetos distribuídos no sistema..

B.2 Parâmetros dinâmicos

A proposta EXEHDA considera que os recursos do sistema distribuído são compartilhados por diversos usuários/aplicações. Dessa form a, avaliações dinâmicas do estado dos recursos que formam a ADD (figura 1) se mostram indispensáveis e complementam a informação dada pelos parâmetros estáticos.

- **Disponibilidade do Poder de Processamento:** na EXEHDA é util izado o tamanho da fila de processos em execução. Experimentos conduzidos por [ZHOE 87] induzem que o tamanho da fila de execução agrega uma informação mais significativa do que as outras alternativas (eg. percentual de CPU utilizado, tempo de CPU idle, etc.);
- **Ocupação dos canais de interconexão** : quando objetos distribuídos no sistema são fortemente acoplados (apresentam elevada comunicação),

informações sobre a ocupação dos canais de comunicação entre os nodos podem trazer otimizações significativas para o escalonamento. O índice utilizado na caracterização dinâmica dos canais de comunicação é a latência, a qual é avaliada por procedimentos do tipo *ping-pong* [XXX 99]. Uma desvantagem desta aboradagem é a inserção de trafego unicamente com o propósito de in strumentação. Uma otimização prevista no sistema, obtida com a camada de comunicação proposta, é a inserção de um campo para tal, nos pacotes normalmente utilizados durante a comunicação entre objetos. Com isto ficam reduzidos os efeitos colaterais desse tipo de instrumentação.

C. Primitivas de Instanciação Remota e Migração de Objetos

O objetivo destas primitivas é permitir a instanciação remota de objetos. Esta funcionalidade não é nativa à linguagem Java, porém é indispensável à implementação de políticas de alocação de objetos distribuídos pois, independentemente da heurística de escalonamento a ser empregada, é ela que suportará a efetiva criação dos objetos remotos. No contexto do EXEHDA esse efeito é alcançado pela utilização de uma classe denominada *EXEStarter*, a qual é pré - instanciada em todas as JVM criadas pelo *EXEd*. Nessa arquitetura o *EXEd* funciona como elemento de transporte para as mensagens de instanciação trocadas entre os *EXEStarter* de cada JVM. A fase de inicialização das JVMs também é respon sável pela instalação de políticas de segurança específicas à aplicação, como controle de acesso aos recursos locais do *host*.

D. Construção de Perfis de Comunicação entre Objetos

Para a construção de perfis de comunicação dinâmicos entre os objetos da aplicaç ão distribuída torna-se necessária a adoção de uma política de atribuição de identificadores únicos aos objetos da aplicação. Os perfis de comunicação são utilizados em duas situações: (*i*) numa análise pós-morten e assim otimizar o mecanismo de alocação do s objetos distribuídos nas execuções subseqüentes; (*ii*) durante a execução propriamente dita, alimentando decisões de reposicionamentos dinâmicos de objetos. A forma de uso das informações disponibilizadas será responsabilidade da heurística de mapeamento escolhida para a aplicação em questão.

Os identificadores de objetos distribuídos são baseados em um esquema de IDs de 64 bits denominados *EXEid*s. O identificador *EXEid* pode ser decomposto em três campos: um campo de 32 bits que contém o endereço IP da hos t no qual o objeto está sendo executado; um campo de 16 bits que representa a máquina virtual e os últimos 16 bits que identificam o objeto dentro da máquina virtual. Essa escolha visa minimizar o esforço de manipulação desse identificador, tendo em vista que várias arquiteturas de hardware hoje já manipulam de forma eficiente inteiros de 16, 32 e 64 bits. Essa escolha também permite que esse identificador possa ser manipulado dentro das JVM – Máquina Virtual Java – diretamente com uma variável do tipo *long*.

Os *EXEid* são utilizados como unidades básicas de endereçamento pelo *EXEd*, apresentando uma semântica semelhante a de portas em sistemas de troca de mensagens baseados em datagramas. Assim, comunicações baseadas em troca de mensagens podem ser construíd as sobre o núcleo provido.

A integração com a API RMI se dá pela utilização de objetos da classe proposta *VSocket*, a qual oferece, para o nível da aplicação, a abstração de uma conexão baseada em *stream* como esperado pelo nível de RMI – cuja implementação padrão esta baseada em *sockets* TCP/IP. Cada objeto *VSocket* possui um *EXEid*, requisitado junto ao *EXEd*, estando portanto habilitado a utilizar as primitivas de comunicação disponibilizadas pelo mesmo. Desse modo, é possível aos objetos *VSocket* exercerem o papel de redirecionadores da comunicação executada no nível RMI para o *EXEd*. A partir do interfaceamento às primitivas de troca de mensagens, o *VSocket* fornece a

semântica de uma conexão orientada a bytes, permitindo a instrumentação da quantidade de dado s trocada entre os objetos.

Os identificadores alocados só fazem sentido no escopo do *EXEd*. Para efetiva construção de perfis de comunicação faz -se necessária a utilização de um serviço de informações distribuída no qual o mapeamento *EXEid<->objeto remoto* seja estabelecido. As abordagens possíveis para essa questão variam desde uma distribuição do serviço entre todos os nodos envolvidos até uma solução completamente centralizada. A escolha adotada foi a de um serviço replicado, no modelo *primário-backups* [FER 00]. Esta escolha foi feita considerando os seguintes aspectos: (*i*) implementação mais simples de gerenciar que a implementada por uma solução completamente distribuída, (*ii*) reduz os custos do processo de atualização e obtenção de informações, pois nã o e necessária um sincronização entre os nodos do sistema distribuído a cada vez que o *snapshot* do sistema for requisitado; (*iii*) possui boa escalabilidade visto que pode ser facilmente estendido para o emprego de uma estrutura hierárquica. Além disso, o a cesso em leitura pode ser liberado para os *backups* de forma a melhorar o desempenho do sistema. Por sua vez, considerando também uma possível utilização futura da proposta em concepções tolerantes à falhas, a estratégia *primário-backups* se mostra oportuna.

V. TRABALHOS RELACIONADOS

Uma quantidade substancial de pesquisa no campo da computação móvel é devotada a tornar as aplicações adaptativas e cônscias dos recursos [AND 00, BAG 98, NOB 00]. O foco das soluções é variável, tendo uma concentração em técnicas de adaptação dos tipos de dados à variação nos recursos da rede (largura de banda, em especial) [ANG 98, BAG 98, NOB 00, RAN 97, WEL 98]. Em geral, esses sistemas usam processos intermediários entre o cliente móvel e o servidor, na forma de *proxy* ou agente, os quais executam algum tipo de filtro que modifica a estrutura/quantidade de dados antes de serem transmitidos na rede sem fio. Outra estratégia de adaptação muito usada é a migração, de *thread* [RAN 97], de *proxy* [AHM 95] ou de agente [GRA 97], onde a de cisão de migrar para um ponto específico é da aplicação. Diferentemente destes sistemas, ISAM utiliza o conceito de escalonamento como uma estratégia central de adaptação. No ISAM o escalonador pode ser visto como um gerente geral distribuído que negocia c om a aplicação as decisões de adaptação. O escalonador, com base nas informações de alteração de contexto e nas políticas adotadas para a aplicação, pode deliberar ações de adaptação. Por sua vez, a aplicação pode requisitar a intervenção do escalonador.

Na perspectiva de utilizar a Internet como infra -estrutura para aplicações altamente distribuídas, têm surgido diversos trabalhos. Sistemas como o Condor [LIT 88] são voltados para aplicações de alto desempenho em *clusters* de estações de trabalho. Diferent emente do ISAM, utilizam um mecanismo central para disparar processos. O projeto Globus [FOS 98] disponibiliza uma “grade de recursos computacionais” [FOS 99] integrando equipamentos heterogêneos em um único sistema. De forma similar à proposta ISAM ele co ntempla uma estrutura escalável e distribuída para o gerenciamento de recursos. Apesar de conter módulo específico para o controle de aplicações (GEM – *Globus Executable Management Service*), a atual versão trata as aplicações como um único executável, ao i nvés de uma coleção de componentes que podem ser parcial e dinamicamente instanciados como na arquitetura ISAM.

Por sua vez, sistemas como Globe [STE 99], Legion [GRI 97], e WebOs [VAH 98], apesar de suportarem diferentes níveis de configuração, não consi deram a adaptabilidade e a configuração automática do ambiente de execução como uma questão central.

VI. CONCLUSÃO

A adaptação é o mais importante requisito para as aplicações móveis atingirem o grau de desempenho que atenda as expectativas dos usuários. A ada ptação pode ser realizada em

diversos níveis: rede, sistema e aplicação. Muitas técnicas, tais como *prefetching* e *caching*, filtragem, compressão e migração, são empregadas com este propósito. Diferentemente, a proposta da arquitetura ISAM é usar o escalona mento como uma estratégia global de adaptação implícita e explícita para as aplicações móveis distribuídas. Em sistemas altamente distribuídos com suporte à computação móvel, o conceito de escalonamento como estratégia de adaptação não foi abordado pela comunidade científica, no que é de nosso conhecimento. A simplicidade desta idéia contrasta com a complexidade de sua implementação em um ambiente altamente dinâmico. Esta circunstância exige um alto grau de adaptação tanto da aplicação móvel quanto da próp ria plataforma de suporte à execução.

O conceito de adaptação é desenvolvido como uma negociação entre a aplicação e a arquitetura ISAM. O escalonamento toma decisões baseadas em informações sobre o ambiente de execução, coletadas e mantidas pela arquitetu ra e pela política de adaptação adotada pela aplicação. Este artigo apresentou os mecanismos de suporte para o escalonamento em sistemas que utilizem objetos distribuídos Java. O escalonamento de objetos distribuídos permite sanar uma deficiência no modelo de objetos Java, ao mesmo tempo que contribui para um aumento de desempenho das aplicações já existentes, através do emprego de técnicas adaptativas na alocação dos objetos em sistemas distribuídos/paralelos.

REFERÊNCIAS BIBLIOGRÁFICAS

[AHM 95] AHMAD, Ta hir; et al. The DIANA Approach to Mobile Computing. In *Mobile Computing*: Kluwer Academic Press, 1995.

[AHM 98] AHMAD, I; KWOK, Y. On Exploiting Task Duplication in Parallel Program Scheduling. *IEEE Transactions on Parallel and Distributed Systems*. New York, v.9, n.9. 1998.

[AND 00] ANDRÈ, Françoise; SEGARRA, Maria -Teresa. A Generic Approach to Satisfy Adaptability Needs in Mobile Environments. In: 33rd ANNUAL HAWAI INTERNATIONAL CONFERENCE ON SYSTEM SCIENCE. *Proceedings*... Maui, Hawaii, USA. 2000.

[ANG 98] ANGIN, Oguz; et al. The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking. *IEEE Personal Communications Magazine*. Special Issue on Adapting to Network and Client Variability. Aug. 1998.

[AUG 00] AUGUSTIN, Iara. Acesso aos Dados no Contexto da Computação Móvel. PPGC/UFRGS. Porto Alegre. Dez. 2000 (Exame de Qualificação).

[BAG 98] BAGGIO, Aline. System Support for Transparency and Network -aware Adaptation in Mobile Environments. In: ACM SYMPOSIUM ON APPLIED COMPUTING - Special Track o n Mobile Computing Systems and Applications. *Proceedings*... Atlanta, USA. Feb. 1998.

[BAR 01a] BARBOSA, Jorge L. V.; Geyer, Cláudio F.R. Uma Linguagem Multiparadigma Orientada ao Desenvolvimento de Software Distribuído. V SIMPÓSIO BRASILEIRO DE LINGUA GENS DE PROGRAMAÇÃO (SBLP). *Anais.* Maio. 2001.

[BAR 01b] BARBOSA, Jorge L. V.; Geyer, Cláudio F.R. Integrating Logic Blackboards and Multiple Paradigm for Distributed Software Development. Proceedings of International Conference on Parallel and Distributed P rocessing Techniques and Applications (PDPTA). June. 2001.

[BAR 01c] BARBOSA, Jorge et al. HoloJava: Translating a Distributed Multiparadigm Language into Java. CLEI'2001 (approved to appear in CLEI 2001).

[CAS 88] CASAVANT, Thomas L.; KUHL, Jon G. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on Software Engineering*, New York, v. 14, n. 2. 1988.

[CIE 97] CIERNIAK, Michal; LI, Wei. Just-In-Time Optimizations for High-Performance Java Programs. Concurrency: Practice and Experience, 1997.

[COR 99] CORRADI, Antonio; LETIZIA, Leonardi; ZAMBONELLI, Franco. Diffuse Load-Balancing Policies for Dynamic Applications. *IEEE Concurrency*. New York, v7, n.1. 1999.

[DAV 97] DAVIES, N.; FRIDAY, A .; WADE, S.; BLAIR, G. Limbo: a Tuple Space Based Platform for Adaptive Mobile Applications. INTERNATIONAL CONFERENCE ON OPEN DISTRIBUTED PROCESSING / DISTRIBUTED PLATFORMS (ICODP/ICDP´97). *Proceedings...* Toronto, Canada. May. 1997.

[FAR 98] FARLEY, Jim; LOUKIDES, Mike. Java Distributed Computing. O'Reilly, 384p. 1998.

[FER 00] FERRARI, D. Nice. Um modelo de Replicação em ambientes que suportam mobilidade. PPGC/UFRGS. 2000. (Dissertação de Mestrado).

[FOS 98] FOSTER, I; KESSELMAN, C. The Globus Project: A Status Report. In Proceedings of the IPPS/SPDP – Heterogeneous Computing Workshop. 1988.

[FOS 99] FOSTER, I; KESSELMAN; C, Editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers. San Francisco. 1999.

[GET 98] GETOV, Vladmir; et al. High-Performance Parallel Programming in Java: Exploiting Native Libraries. In Proc. of 1998 ACM Workshop on Java for High-Performance Network Computing, 1998

[GRA 01] Java Grande Fórum. Disponível em http://www.javagrande.org. Acessado em 6 de maio de 2001.

[GRA 97] GRAY, Robert; KOTZ, David; NOG, Saurab; RUS, Daniela; CYBENKO, George. Mobile Agents for Mobile Computing. Proceeding of 2 nd International Symposium on Parallel Algorithms/Architectures Synthesis. Japan. Mar. 1997.

[GRI 97] GRIMSHAW, A; et al. The Legion Vision of a World-Wide Virtual Computer. *Communications of the ACM*. New York, v.40, n.1. 1997.

[KAT 94] KATZ, R.H. Adaptation and Mobility in Wireless Information Systems. *IEEE Personal Communications*. vol.1, n.1, p.6-17. 1994.

[KUN 99] KUNZ, Thomas; BLACK, J.P. An Architecture for Adaptive Mobile Applications. Proceedings 11th International Conference on Wireless Communications. Alberta, Canada. Jul. 1999.

[LIT 88] LITZKOW, M. et al. Condor – A Hunter of Idle Workstations. In Proceedings of th 8th Internacional Conference of Distributed Computing Systems. 1988.

[NIE 00] van NIEWPOORT, Rob; et al. Wide Area Parallel Programming using the Remote Method Invocation Model. Concurrency: Practice and Experience, Vol. 12, No. 8, pp. 643-666, 2000.

[NOB 00] NOBLE, Brian. System Support for Mobile, Adaptive Applications. *IEEE Personal Computing Systems*. v.7,n.1,p. 44-9, Feb. 2000.

[PHI 97] PHILIPPSEN, Michael; ZENGER, Matthias. JavaParty - Transparent Remote Objects in Java. Concurrency: Pratice & Experience, Vol. 9. No.11, pp. 1225-1242, November 1997.

[PIC 99] PICCO, Gian Pietro; MURPHY, Amy L.; ROMAN, Gruia -Catalin. LIME: Linda Meets Mobility. Proceedings of 21 International Conference on Software Engineering (ICSE´99). Los Angeles, USA. May. 1999.

[RAN 97] RANGANATHAN, M.; ACHARYA, A.; SALTZ, J. Sumatra: a Language for Resource-aware Mobile Programs. In *Mobile Objects Systems: Towards the Programable Internet*: Springer-Verlag Publisher, Serie Lecture Notes on Computer Science. v.1222. Apr. 1997.

[SIL 01] SILVA, Luciano. Uma Contribuição para a Alocação de Objetos Distribuídos em Java. Semana Acadêmica do PPGC. 2001.

[STE 99] STEEN, M. van; et al. Globe: A Wide-Area Distributed System. *IEEE Concurrency*. New York, v.7, n.1. 1999.

[SUN 01a] SUN Microsystems. Documentação da linguagem Java. Disponível na web em http://java.sun.com. Ùltimo acesso em maio de 2001.

[SUN 01b] SUN Microsystems. The Java Hotspot Performance Engine Architecture. Disponível em: http://java.sun.com/products/hotspot/whitepaper.html. Acessado em 6 de maio de 2001.

[VAH 96] VAHALIA, Uresh, Unix Internals: the new Frontiers. Prentice Hall, New Jersey, 1996. 605p.

[VAH 98] VAHDAT, T. et al. WebOS: Operating System Services for Wide Area Applications. In Proceedings of the Seventh Symposium on High Performance Distributed Computing. 1998.

[WEL 98] WELLING, Girish; BADRINATH, B.R. Na Architecture for Exporting Environment Awareness to Mobile Computing Applications. *IEEE Transactions on Software Engineering*. v. 24, n.5. 1998.

[YAM 01] YAMIN, Adenauer. Escalonamento em Sistemas Paralelos e Distribuídos. ERAD 2001, Gramado, RS. SBC/UFRGS/PUCRS. Jan. 2001.

[YAM 99] YAMIN, A. C. *An Execution Environment for Multiparadim Models*. PPGC/UFRGS, 1999. (PHD Proposal)

[YEL 98] YELICK, Kathy; et al. Titanium: A High -Performance Java Dialect. In Proc. of ACM 1998 Workshop on Java for High -Performance Network Computing, Stanford, California, Feb/ 1998.

[ZOM 98] ZOMAYA, Albert; CLEMENTS, Matthew; OLARIU, Stephan. A Framework for Reinforcement-Based Scheduling Parallel Processor System. *IEEE Transactions on Parallel and Distributed Systems*. New York, v.9, n.3. 1998.