

Mecanismos de Suporte ao Escalonamento em Sistemas com Objetos Distribuídos Java

Luciano da Silva¹, Adenauer Yamin^{1,2}, Jara Augustin^{1,3}, Edvar Bergmann Araujo¹, Cláudio Geyer¹

¹Instituto 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

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

³Departamento 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 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 como a API RMI de Java, buscando preservar a natureza do OO – Orientação a Objetos – na construção de aplicações distribuídas, e consequentemente 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 à execução de aplicações móveis distribuídas desenvolvidas utilizando o modelo Holo paradigma.

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

Abstract —

This article proposes support resources for distributed objects 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 Rio Grande do Sul, and has as objective to provide means of supporting the execution of distributed mobile applications developed using the Holo paradigm 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. Num 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*) [YAM99], mas reflete uma tendência de grupos de pesquisas internacionais. Rapidez de aprendizado, somada a o fato do modelo de objetos ser adequado ao tratamento de problemas distribuídos e paralelos e a ausência de pontos fortes, o que facilita a otimização de código pelos compiladores, são as justificativas mais comuns para a tendência [FOS99]. 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 [GET98, PHI97, NIE00] e [YEL98], otimização dos *bytecodes* gerados [SUN01b] e [CIE97], até a otimização do mecanismo de serialização usado pela API RMI de Java [PHI97] e [NIE00]. Os resultados atingidos por tais esforços são satisfatórios e bastante animadores [GRA01].

Tendo em vista o uso de arquitetura 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 nó do processador, ou em nós 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/ou recursos [AUG00]. Estes ambientes vão ser caracterizados por nós de processamento bastante heterogêneos, os quais serão interconectados por redes sujeitas a frequentes flutuações nos serviços fornecidos. Como consequência 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 [SUN01a] oferece uma solução para o problema de portabilidade de código, a o mesmo tempo em que apresenta um modelo de programação de Objetos Distribuídos, chamado RMI [FAR98] – *Remote Method Invocation*, bastante conhecido. Java, entretanto, não fornece mecanismos para obtenção de informações contextuais, como o estatístico sobre os recursos utilizados em cada nó do sistema distribuído e nem para a 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 escalonamento de tarefas – parte integrante da proposta EXEHDA, foram projetados dois mecanismos: (i) instância remota de objeto 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 consequência de aumento de desempenho.

O texto está organizado da maneira como segue. Na seção II, caracteriza-se a arquitetura para a 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 VII apresentam-se as conclusões.

II. A ARQUITETURA DESTINO

A computação móvel abrange uma faixa de cenários, os quais são diferentes em seus 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 pontos de acesso para os *hosts* móveis;
- **ad-hoc** - cenário dinâmico composto somente por *hosts* móveis (sem suporte a 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 uma infraestrutura-estruturada de 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, e estação-base (servidor com suporte para comunicação sem fio), e demais componentes de um sistema distribuído heterogêneo.

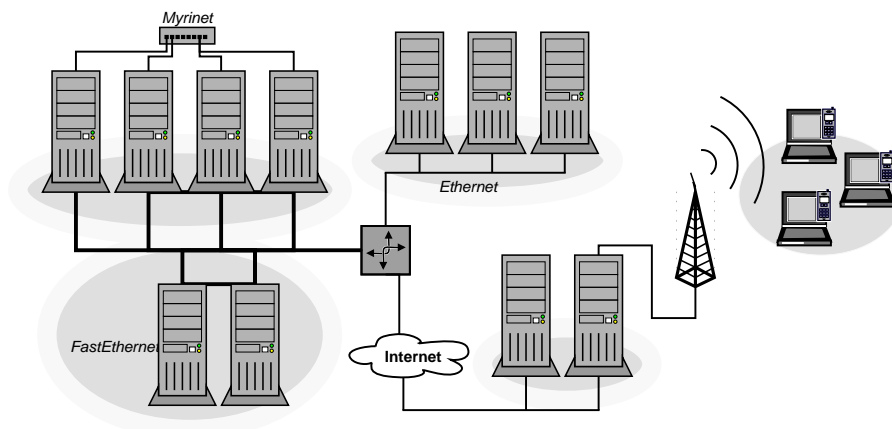


FIGURA 1 – Arquitetura Distribuída Destino (ADD)

A crescente disponibilidade e facilidade de comunicação tem deslocado as aplicações de 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 como HoloLinguagem [BAR01a], uma linguagem de programação que integra o paradigma lógico, imperativo, orientado a objetos e distribuído. O modelo de coordenação é baseado no conceito de memória logicamente compartilhada e suporte a invocação simplificada (*blackboard*) e explícitas (mensagens) [BAR01b]. Este modelo é apropriado para a ISAM porque contempla o desacoplamento

espacialetemporal dacomunicaçãoesincronização,propriedadesimportantesparaacomputação móvel[PIC99].AHololingüageméexecutadaapartirdeummapeamentofeitoparaJava[BAR 01c]. Acamadainferior(INF) écompostadossistemasdeinfra -estruturadistribuídapré -existentes, taiscomosistemasderedemóvel,sistemasoperacionaisnativoeMáquinaVirtualJava.

Acamadaintermediária(INTERM)éonúcleofuncionaldaarquitecturaISAM,é fornecidaemdoisníveisdeabstração.Oprimeironívelécompostoportrêsmódulosdesuporteà aplicação:AmbienteVirtualdoUsuário,GerenciamentodaMobilidadeFísica,Gerenciamentode Recursos.Nosegundoníveldacamadaintermediáriaestãoosserviçobásicosaosu porteda execuçãoonoISAM.Eénestenível,queseinseremosmecanismosdesuporteaoescalamento detalhadostenestartigo.

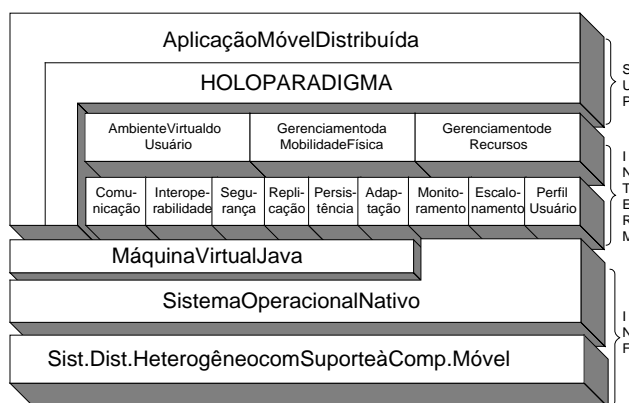


FIGURA 2 -AArquiteturaISAM

III. OE SCALONAMENTO PROPOSTOPARAO ISAM

Umaaplicação móvel deveoferecerresultadosomaisrapidamentepossívelaoseu usuário.Istoéumaexigênciainplícitaàsituaçãodemobilidadedousuário:equipamentocom poucaautonomia(operadoabateria),custodeusodaredemóvel,inserçãonotempo/espaçodo contextodatomadadedecisão(e.g.reuniõescomclientes),etc.

O *framework* EXEHDA que fornece o suporte de escalonamento para a arquitetura ISAM tem como meta de projetar ser flexível e extensível[YAM01,SIL01].O ambiente móvel se caracteriza com aplicações com elevada dinamicidade de execução.As propostas de balanceamento de carga difusas[COR99]se mostram adequadas para este tipo de aplicação.Inspiradonestas,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.Paratãl, 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 decisões são replicados, são capazes de atividades autônomas e assíncronas;
- as metas de escalonamento são perseguidas em escopos.Cada componente que tomadecisãoescalonaos serviços no seu domínio;
- uso intensivo de registro histórico como auxílio na tomada de decisão.

O modelo de escalonamento adotado utiliza uma organização fisicamente distribuída e cooperativa [CAS88]. No que diz respeito às estratégias para maximização do desempenho, o escalonamento no ISAM utiliza as seguintes:

- balanceamento de carga nos nós responsáveis pelo processamento;
- localização dos recursos (software e hardware) mais próximos (reduzindo o custo de comunicação);
- emprego de replicação de serviços e de dados;
- disponibilização antecipada, por usuário, da demanda de componentes das aplicações e de dados;
- otimização do volume de comunicações, utilizando transferências de contexto e componentes de aplicação personalizadas por usuário;
- monitoração da comunicação praticada pelos componentes das aplicações em execução, com intuito de otimizar aspectos de mapeamento.

em

O emprego destes procedimentos fica potencializado pela alternância do ponto de conexão do equipamento móvel 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 [FER00, AHM98].

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 tomada de decisão em diferentes situações (vide figura 3).

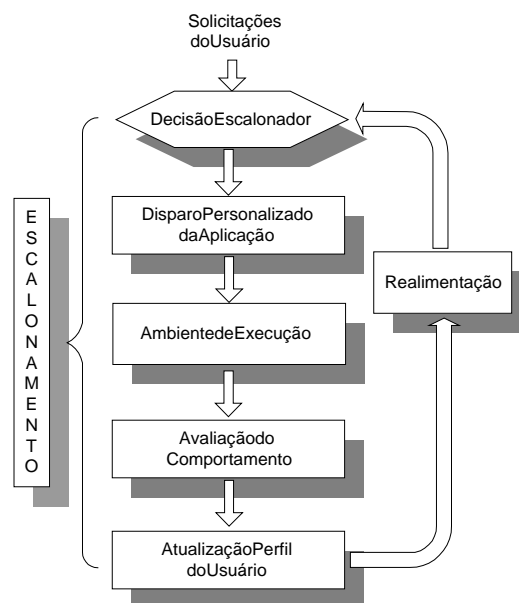


FIGURA 3 – Aprendizagem no Escalonamento ISAM

Nesta proposta, o escalonador emprega uma abordagem estocástica com aprendizado por reforço, na qual são construídas correlações estatísticas entre usuário, o comportamento das suas aplicações e o ambiente de execução. Trabalhando neste mesmo sentido, voltado para o escalonamento de aplicações paralelas em múltiplos processadores, pode ser encontrado em [ZOM98].

Um dos principais objetivos de dotar o escalonamento com uma estratégia de aprendizado por reforço é viabilizar uma instância 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 à 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, o autor sugere a adoção do conceito de adaptação [KAT94, DAV97]. Porém, diferentes dos sistemas analisados, o modelo ISAM propõe usar o escalonador como estratégia de adaptação implícita. Para se adaptar ao sistema necessita obter informações de estado dos recursos que utiliza. Com esse objetivo, foram projetados alguns mecanismos que atendam às exigências da arquitetura, descritos na sequência.

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 (iv) construção de perfis de comunicação entre objetos. Os principais aspectos destes mecanismos serão descritos a seguir.

A. Primitivas Otimizadas de Comunicação

O *daemon EXE* está projetado para comportar a operação simultânea em uma camada de transporte das diversas tecnologias de rede que formam o sistema distribuído (como objetivo de otimizar as comunicações). Assim, na comunicação entre sistemas de alta latência, sujeitos a perdas de mensagens frequentes, automaticamente será empregado um protocolo com controle de fluxo (como TCP/IP) a custo de um maior *overhead*, enquanto que na comunicação entre nós de uma rede local, ou no caso de redes de alta velocidade (e.g. *SCIO* ou *Myrinet*), pode-se empregar um protocolo nativo com uma política mais relaxada de controle de erro e assim ser diminuído 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 repetição das cópias dos *buffers* de comunicação entre as camadas, comprometer o desempenho do sistema.

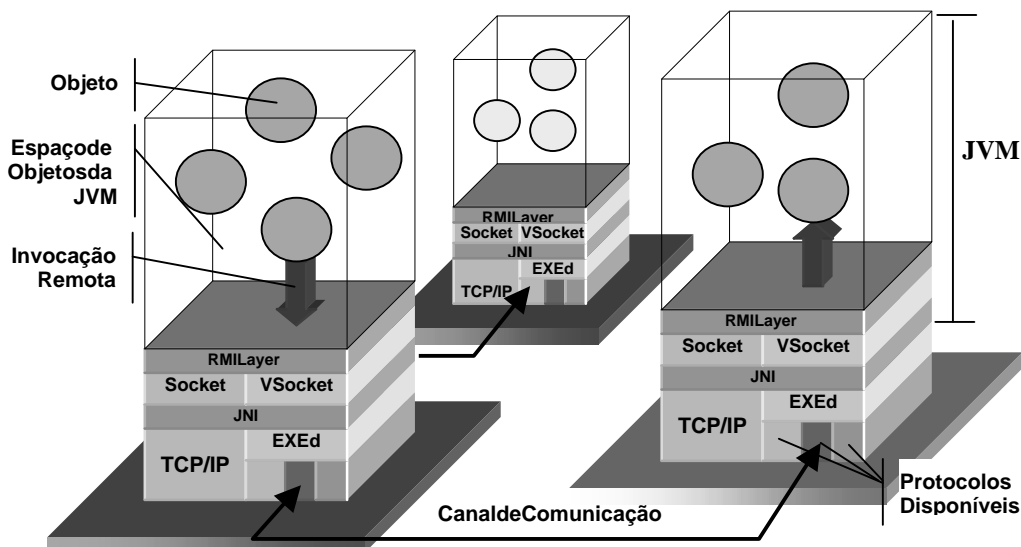


FIGURA 4 – Arquitetura de Suporte ao Escalonamento no EXEHDA

Técnicas de otimização como o algoritmo *zero-copy* [VAH96] são, portanto, altamente desejáveis. O *EXE* implementa no contexto da camada de comunicação o comportamento *zero-copy* pelo emprego de um segmento de memória compartilhada entre os processos [XXX99], de forma a minimizar os efeitos da modelagem em camadas sem, no entanto,

sacrificara estruturação do software. Outras soluções no nível do Sistema Operacional (eg. remapeamento de páginas de memória no *kernel* do S.O.) [XXX99] 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 da diferenciação dos recursos de hardware e software disponíveis ou serem resultado da utilização não homogênea dos mesmos. Portanto, a avaliação pode ser de processamento disponível em um sistema distribuído envolvendo a consideração de parâmetros tanto estáticos como dinâmicos.

B.1 Parâmetros estáticos

Os parâmetros estáticos descrevem, de certa forma, o comportamento ótimo ou o potencial computacional máximo, aomesmo 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 nós será feita através de *benchmarks* [XXX99] no momento que os mesmos forem cadastrados para compor a Arquitetura Distribuída Destino (figura 1);
- **Quantidade de Memória :** valor diretamente decorrente da quantidade de memória RAM disponível no nó;
- **Conectividade entre os Nós :** esse parâmetro procura descrever a qualidade de conexão de um nó para os demais nós do sistema. Será utilizada a velocidade nominal da infraestrutura de conexão (rede) como base.

Os três primeiros itens são associados aos nós. Já a conectividade é melhor caracterizada empregando -se uma estrutura hierárquica, de forma a caracterizar nós pertencentes a um mesmo segmento de rede (conexão direta), bem como as conexões entre os diversos segmentos que compõem o sistema distribuído. Pela utilização desta estrutura, minimiza -se o armazenamento de informações redundantes sobre a conectividade dos nós. 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 forma, a avaliação dinâmica do estado dos recursos que formam a ADD (figura 1) sem ostrar indispensáveis e complementam a informação dada pelos parâmetros estáticos.

- **Disponibilidade do Poder de Processamento:** na EXEHDA é utilizado o tamanho da fila de processos em execução. Experimentos conduzidos por [ZHOE87] 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 (a presente 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* [XXX99]. Uma desvantagem desta abordagem é a inserção de tráfego unicamente como propósito de instrumentação. Uma otimização prevista no sistema, obtida com a camada de comunicação proposta, é a inserção de um campo paratá, 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, por isso é indispensável a implementação de políticas de alocação de objetos distribuídos, independentemente da heurística de escalonamento a ser empregada, é a 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 JVMs 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 responsável 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 aplicação distribuída torna-se necessária a adoção de umapolítica de atribuição de identificadores únicos aos objetos da aplicação. Os perfis de comunicação são utilizados em duas situações: (i) numa análise pós-mortem para otimizar o mecanismo de alocação dos objetos distribuídos nas execuções subsequentes; (ii) durante a execução propriamente dita, alimentando decisões de reposicionamento dinâmico de objetos. A forma de uso das informações disponibilizadas será a 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 *EXEids*. O identificador *EXEid* pode ser decomposto em três campos: um campo de 32 bits que contém o endereço IP da host 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 *EXEids* são utilizados como unidades básicas de endereçamento pelo *EXEd*, apresentando uma semântica semelhante de portas em sistemas de troca de mensagens baseados em datagramas. Assim, comunicações baseadas em troca de mensagens podem ser construídas 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 se esperada pelo nível de RMI – cuja implementação padrão está 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 dados trocados 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 distribuído no qual o mapeamento *EXEId* <-> *objeto remoto* seja estabelecido. As abordagens possíveis para essa questão variam desde uma distribuição do serviço entre todos os nós envolvidos até uma solução completamente centralizada. A escolha adotada foi a de um serviço replicado, no modelo *primário-backups* [FER00]. 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 não é necessária uma sincronização entre os nós 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 acesso em leitura pode ser liberado para os *backups* de forma a melhorar o desempenho do sistema. Por suavez, considerando também uma possível utilização futura da proposta em concepções tolerantes a falhas, a estratégia *primário-backups* se mostra oportuna.

V. TRABALHOS RELACIONADOS

Uma quantidade substancial de pesquisas no campo da computação móvel é dedicada a tornar as aplicações adaptativas e conscientes dos recursos [AND00, BAG98, NOB00]. Ofocodas soluções é variável, tendendo a uma concentração em técnicas de adaptação dos tipos de dados à variação nos recursos da rede (largura de banda, em especial) [ANG98, BAG98, NOB00, RAN97, WEL98]. Em geral, esses sistemas usam processos intermediários entre o cliente móvel e o servidor, na forma de *proxy* 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* [RAN97], de *proxy* [AHM95] ou de agente [GRA97], onde a decisão 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 servir como um gerente geral distribuído que negocia uma aplicação as decisões de adaptação. O escalonador, com base nas informações de alteração de contexto e nas políticas adotadas para a aplicação, pode liberar ações de adaptação. Por suavez, a aplicação pode requisitar a intervenção do escalonador.

Na perspectiva de utilizar a Internet como infraestrutura para aplicações altamente distribuídas, têm surgido diversos trabalhos. Sistemas como o Condor [LIT88] são voltados para aplicações de alto desempenho *clusters* de estações de trabalho. Diferentemente do ISAM, utilizam um mecanismo central para disparar processos. O projeto Globus [FOS98] disponibiliza uma "grade de recursos computacionais" [FOS99] integrando equipamentos heterogêneos em um único sistema. De forma similar à proposta ISAM ele contempla uma estrutura escalável distribuída para o gerenciamento de recursos. Apesar de conter módulos específicos para o controle de aplicações (GEM – *Globus Executable Management Service*), a atual versão trata as aplicações como um único executável, ao invés de uma coleção de componentes que podem ser parciais e dinamicamente instanciados como na arquitetura ISAM.

Por suavez, sistemas como o Globe [STE99], Legion [GRI97], e WebOs [VAH98], apesar de suportarem diferentes níveis de configuração, não consideram 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 adaptação 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 escalonamento 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 ideia 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 própria 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 em tempo real pela arquitetura 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, aomesmo 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

- [AHM95] AHMAD, Tahira; et al. The DIANA Approach to Mobile Computing. In *Mobile Computing*: Kluwer Academic Press, 1995.
- [AHM98] 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.
- [AND00] ANDRÈ, Françoise; SEGARRA, Maria Teresa. A Generic Approach to Satisfy Adaptability Needs in Mobile Environments. In: 33rd ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEMS SCIENCE. *Proceedings...* Maui, Hawaii, USA. 2000.
- [ANG98] 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.
- [AUG00] AUGUSTIN, Iara. Acesso aos Dados no Contexto da Computação Móvel. PPGC/UFRGS. Porto Alegre. Dez. 2000 (Exame de Qualificação).
- [BAG98] BAGGIO, Aline. System Support for Transparency and Network-aware Adaptation in Mobile Environments. In: ACMSYMPOSIUM ON APPLIED COMPUTING - Special Track on Mobile Computing Systems and Applications. *Proceedings...* Atlanta, USA. Feb. 1998.
- [BAR01a] BARBOSA, Jorge L. V.; Geyer, Cláudio F. R. Uma Linguagem Multiparadigma Orientada ao Desenvolvimento de Software Distribuído. V SIMPÓSIO BRASILEIRO DE LINGUAGEM DE PROGRAMAÇÃO (SBLP). *Anais*. Maio. 2001.
- [BAR01b] 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 Processing Techniques and Applications (PDPTA)*. June. 2001.
- [BAR01c] BARBOSA, Jorge et al. HoloJava: Translating a Distributed Multiparadigm Language into Java. CLEI'2001 (approved to appear in CLEI 2001).

- [CAS88] CASAVANT, Thomas L.; KUHL, Jon G. A taxonomy of scheduling in general -
 purposed distributed computing systems. *IEEE Transactions on Software
 Engineering*, New York, v. 14, n. 2. 1988.
- [CIE97] CIERNIAK, Michal; LI, Wei. Just-In-Time Optimizations for High -Performance
 Java Programs. *Concurrency: Practice and Experience*, 1997.
- [COR99] CORRADI, Antonio; LETIZIA, Leonardi; ZAMBONELLI, Franco. Diffuse Load -
 Balancing Policies for Dynamic Applications. *IEEE Concurrency*. New York, v 7,
 n. 1. 1999.
- [DAV97] 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.
- [FAR98] FARLEY, Jim; LOUKIDES, Mike. *Java Distributed Computing*. O'Reilly, 384p.
 1998.
- [FER00] FERRARI, D. Nice. Um modelo de Replicação em ambientes que suportam
 mobilidade. PPGC/UFRGS. 2000. (Dissertação de Mestrado).
- [FOS98] FOSTER, I; KESSELMAN, C. The Globus Project: A Status Report. In Proceedings
 of the IPPS/SPDP - Heterogeneous Computing Workshop. 1988.
- [FOS99] FOSTER, I; KESSELMAN, C, Editors. *The Grid: Blueprint for a New Computing
 Infrastructure*. Morgan Kaufmann Publishers. San Francisco. 1999.
- [GET98] GETOV, Vladimir; 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
- [GRA01] Java Grande Fórum. Disponível em <http://www.javagrande.org>. Acessado em 6 de
 maio de 2001.
- [GRA97] GRAY, Robert; KOTZ, David; NOG, Saurab; RUS, Daniela; CYBENKO, George.
 Mobile Agents for Mobile Computing. Proceeding of 2nd International Symposium
 on Parallel Algorithms/Architectures Synthesis. Japan. Mar. 1997.
- [GRI97] GRIMSHAW, A; et al. The Legion Vision of a World -Wide Virtual Computer.
Communications of the ACM. New York, v. 40, n. 1. 1997.
- [KAT94] KATZ, R.H. Adaptation and Mobility in Wireless Information Systems. *IEEE
 Personal Communications*. vol. 1, n. 1, p. 6 -17. 1994.
- [KUN99] KUNZ, Thomas; BLACK, J.P. An Architecture for Adaptive Mobile Applications.
 Proceedings 11th International Conference on Wireless Communications. Alberta,
 Canada. Jul. 1999.
- [LIT88] LITZKOW, M. et al. Condor - A Hunter of Idle Workstations. In Proceedings of the
 8th International Conference of Distributed Computing Systems. 1988.
- [NIE00] 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.
- [NOB00] NOBLE, Brian. System Support for Mobile, Adaptive Applications. *IEEE Personal
 Computing Systems*. v. 7, n. 1, p. 44 -9, Feb. 2000.
- [PHI97] PHILIPPSEN, Michael; ZENGER, Matthias. Java Party - Transparent Remote
 Objects in Java. *Concurrency: Practice & Experience*, Vol. 9. No. 11, pp. 1225 -1242,
 November 1997.

- [PIC99] PICCO, GianPietro; MURPHY, AmyL.; ROMAN, Gruia -Catalin. LIME: Linda Meets Mobility. Proceedings of 21 International Conference on Software Engineering (ICSE '99). Los Angeles, USA. May. 1999.
- [RAN97] RANGANATHAN, M.; ACHARYA, A.; SALTZ, J. Sumatra: a Language for Resource-aware Mobile Programs. In *Mobile Objects Systems: Towards the Programmable Internet* : Springer -Verlag Publisher, Serie Lecture Notes on Computer Science. v. 1222. Apr. 1997.
- [SIL01] SILVA, Luciano. Uma Contribuição para a Alocação de Objetos Distribuídos em Java. Semana Acadêmica do PPGC. 2001.
- [STE99] STEEN, M. van; et al. Globe: A Wide Area Distributed System. *IEEE Concurrency* . New York, v. 7, n. 1. 1999.
- [SUN01a] SUN Microsystems. Documentação da linguagem Java. Disponível na web em <http://java.sun.com>. Último acesso em maio de 2001.
- [SUN01b] 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.
- [VAH96] VAHALIA, Uresh, Unix Internals: the new Frontiers. Prentice Hall, New Jersey, 1996. 605p.
- [VAH98] VAHDAT, T. et al. WebOS: Operating System Services for Wide Area Applications. In Proceedings of the Seventh Symposium on High Performance Distributed Computing. 1998.
- [WEL98] 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.
- [YAM01] YAMIN, Adenauer. Escalonamento em Sistemas Paralelos e Distribuídos. ERAD 2001, Gramado, RS. SBC/UFRGS/PUCRS. Jan. 2001.
- [YAM99] YAMIN, A. C. *An Execution Environment for Multiparadigm Models* . PPGC/UFRGS, 1999. (PHD Proposal)
- [YEL98] 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.
- [ZOM98] 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.