Niba Spaces: uma Solução Baseada em JavaSpaces para Exploração de Recursos Computacionais

Guilherme Luiz Lanius

Universidade do Vale do Rio dos Sinos – Unisinos
speka@uol.com.br

Introdução

Nos últimos anos pode-se observar uma crescente demanda de processamento em diversos setores, mas o custo para um hardware específico para estes processamentos maiores é bem elevado. Como solução para estes problemas, vimos surgir arquiteturas como as COW (Cluster of Workstations) e NOW (Network of Workstations), onde se aproveitam várias máquinas de "prateleira" e consegue-se montar uma arquitetura com alto poder de processamento. A NOW tem uma certa vantagem extra para muitas empresas, pois ela utiliza a infra-estrutura já existente, gerando um custo praticamente zero.

Várias tecnologias foram desenvolvidas nos últimos anos para explorar o poder computacional de redes de trabalho. Uma delas é o JavaSpaces. Neste artigo é apresentado um kit para se construir essa arquitetura e que facilitará a construção de aplicações para mesma.

JavaSpaces

A tecnologia de JavaSpaces [FREE01] é uma ferramenta simples, expressiva e poderosa para juntar processos em aplicações distribuídas. JavaSpaces fornece um modelo de programação fundamentalmente diferente, que vê a aplicação como uma coleção de processos cooperando através do fluxo de objetos para dentro e fora de um ou mais espaços. Este modelo de espaços baseado em computação distribuída tem suas raízes na linguagem Linda, desenvolvia pelo Dr. David Gelernter na Yale University [FREE01].

Um espaço é um repositório de objetos compartilhado acessível via rede. Processos podem coordenar trocando objetos através deste espaço, ao invés de se comunicarem diretamente. Este método de coordenação distribuída leva a sistemas que são flexíveis, escaláveis e confiáveis [FREE01].

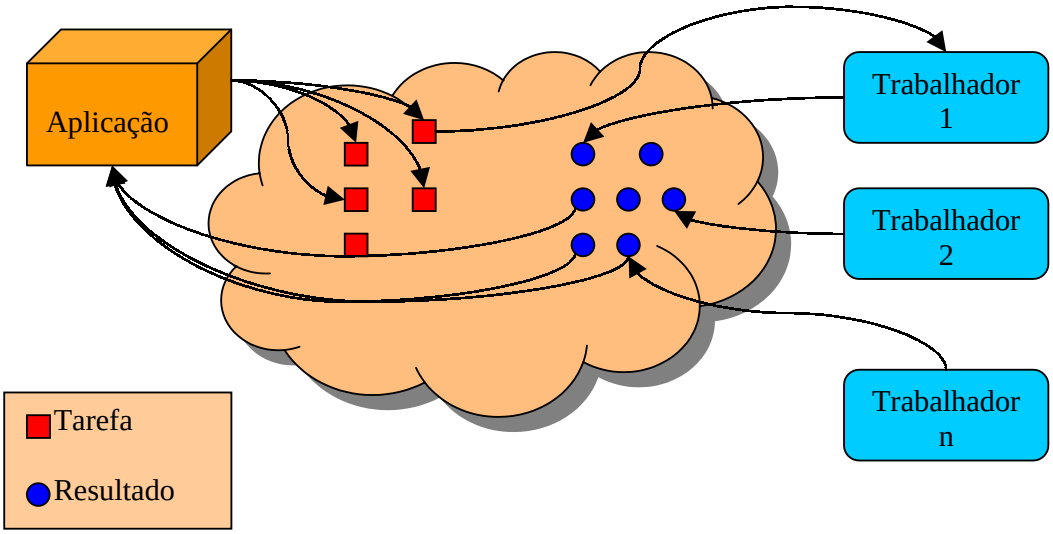
JavaSpaces é baseado na Tecnologia Jini [JSS00], levando vantagem de algumas de suas funcionalidades, como leasing, eventos distribuídos e transações.

Niba Spaces Toolkit(NST)

O NST é uma ferramenta desenvolvida para auxiliar na construção de aplicações distribuídas, que podem ser processadas em maquinas heterogêneas, sem a necessidade de alterar os trabalhadores. As aplicações e trabalhadores se comunicam através de um espaço de objetos(JavaSpaces).

O NST contêm um conjunto de classes que visa facilitar a vida do programador que vai construir uma aplicação para rodar numa arquitetura NOW. O usuário só tem o trabalho de instalar os trabalhadores(uma única vez, para qualquer aplicação) nas estações de trabalho, mas principalmente se preocupar em modelar seu problema através de tarefas, não precisando se preocupar com detalhes de implementação de comunicação e falhas.

Pode ser identificado 4 papéis básicos dentro do NST: aplicação, trabalhador, tarefas e resultados. A aplicação, responsável em gerar tarefas e colher resultados, e o trabalhador, responsável em colher tarefas e gerar resultados. Na figura abaixo temos uma ilustração dos 4 papéis e de uma aplicação em funcionamento.



Aplicação

Uma Aplicação feita com o Niba Spaces(NS) pode rodar em qualquer Trabalhador, sem que a aplicação ou o trabalhador precisam ser alterados. Do ponto de vista lógico, as aplicações do NS são divididas em tarefas, que produzem resultados. A Aplicação é responsável em gerar as tarefas, pegar e então pós-processar os resultados (que podem gerar novas tarefas) gerando um resultado final da aplicação.

Trabalhador

O Trabalhador é o componente do kit disponibilizado, sem que seja necessária qualquer alteração no mesmo para que seja instalado nos computadores da NOW, e que rode qualquer aplicação do NST. O trabalhador consiste na classe NSClient, e como já falado anteriormente é responsável em pegar tarefas, processá-las e gerar resultados.

Funcionamento

Para inicializar o trabalhador, apenas é necessário o endereço do espaço. Durante sua execução, o trabalhador fica monitorando o espaço a procura de tarefas, quando encontra uma disponível, pega, executa e após pronto coloca o resultado de volta no espaço.

Durante o processo de pegar, e executar uma tarefa, podem ocorrer falhas no trabalhador, que será comentando na seção a seguir.

Falhas

É possível que existam falhas durante o processamento de uma tarefa, como o trabalhador ser fechado de forma anormal, ou falhas de hardware da máquina. Para evitar que tarefas sejam perdidas, a estratégia adotada prevê redundância na execução das tarefas.

Ao pegar uma tarefa, o trabalhador, ao invés de removê-la do espaço, deixa uma cópia da mesma no espaço, mas com um flag setando que a tarefa já foi pega. Ao terminar de executar a tarefa, o trabalhador verifica se a tarefa ainda esta lá disponível no espaço, caso ainda estiver(ou seja, nenhum outro trabalhador gerou uma resposta), então remove a tarefa e grava o resultado. Caso não encontrar a tarefa, simplesmente descarta o resultado.

Ao procurar as tarefas, os trabalhadores darão preferência para pegar tarefas que não tenham sido pegues ainda, assim evitando a redundância desnecessária.

Estudo de Caso

Para validar, testar e mostrar em funcionamento, o NST será implementada uma aplicação que faz o cálculo do Ray Trace. Como o objetivo desta pesquisa não é estudar o algoritmo e funcionamento do Ray Trace, implementamos um algoritmo sintético, apenas com a finalidade de testar e avaliar a escalabilidade e performance da aplicação.

Conclusão

Durante o trabalho foi desenvolvido uma ferramenta que facilita bastante o desenvolvimento de aplicações distribuídas para utilizar recursos computacionais ociosos.

Através do estudo de caso, foi possível fazer testes e validar a ferramenta, verificando a viabilidade da proposta.

Referências

[FREE01] Eric Freeman, Susanne Hupfer and Ken Arnold. Java Spaces Principles, Patterns, and Practice. Addison Wesley Longman, Inc.

[JSS00] Sun Microsystems. JavaSpaces Service Specification. Sun Microsystem