Padrões de Projeto para Mobilidade de Código no Desenvolvimento de Aplicações de Alto Desempenho no *Middleware* GRADEp

Giulian G. Vivan[1], Vinícius P. Ferreira[2], Ana M. Pernas[1], André R. Du Bois[2], Adenauer C. Yamin[2,3]

[1]Instituto de Física e Matemática – Universidade Federal de Pelotas (UFPEL)

[2]Escola de Informática – Universidade Católica de Pelotas (UCPEL)

[3]Centro de Informática – Universidade Federal de Pelotas (UFPEL)

{gvivan.ifm,marilza}@ufpel.edu.br, {vpf,adenauer,dubois}@ucpel.tche.br

Resumo. *O middleware GRADEp oferece suporte para modelagem, implementação e execução de aplicações da Computação Pervasiva. Para testar a estrutura fornecida por esse middleware sob a coordenação de um framework para mobilidade de código, este trabalho faz uma análise do comportamento de uma aplicação de alto desempenho sobre o GRADEp. A modelagem e a implementação da aplicação fazem uso de um padrão de projeto para mobilidade de código. Esta aplicação faz uma estimativa do valor de PI através do método Monte Carlo.*

1. Introdução

A computação pervasiva segundo Satyanarayanan (2001) é um passo evolucionário cujos precedentes foram a computação distribuída e a computação móvel, onde algumas das funcionalidades bem como alguns dos problemas técnicos dos passos anteriores são herdados no novo paradigma.

Grimm et al. (2001) argumentam que é necessária uma arquitetura de software específica que forneça às aplicações pervasivas os subsídios necessários para a exploração da infra-estrutura física disponível. Dentre as plataformas pervasivas disponíveis encontra-se a arquitetura definida pelo *middleware* GRADEp.

O foco do GRADEp é oferecer suporte à aplicações distribuídas, móveis e conscientes de contexto [Yamin 2004]. O GRADEp fornece subsídios para o desenvolvimento destas aplicações. Embora o *middleware* ofereça as primitivas necessárias, permitindo flexibilidade, as características inerentes a estas aplicações introduzem esforços extras de programação, os quais costumam ser recorrentes.

O trabalho de [Vivan, Pernas and Yamin 2007] apresenta um *framework* que facilita a exploração dos recursos de mobilidade de código [Fuggetta, Picco and Vigna 1998] do GRADEp. Esse *framework* fornece padrões de projeto [Gamma, Helm, Johnson and Vlissides 1995] que encapsulam padrões recorrentes de mobilidade de código. Desta maneira, o programador pode reusar soluções já testadas.

O objetivo deste trabalho é testar o suporte a execução distribuída juntamente com os mecanismos de mobilidade de código fornecidos pelo *middleware* para o

desenvolvimento de aplicações de alto desempenho, sob a coordenação de um *framework* para mobilidade de código. Este trabalho fez uma análise da execução de uma aplicação de alto desempenho sintética sobre o GRADEp. A aplicação implementada consiste na estimativa do valor de PI através do Método Monte Carlo. Com o uso do *framework* foi possível reduzir esforços tanto de modelagem como de implementação da aplicação desenvolvida neste trabalho.

2. O *middleware* GRADEp

O GRADEp define um ambiente de execução para aplicações pervasivas, implementado na forma de um *middleware*. Este *middleware* é responsável por prover funcionalidades para gerenciamento de uma grade pervasiva em tempo de execução, e fornecer subsídios para desenvolvimento de aplicações sobre a mesma.

Os recursos utilizados do *middleware* neste trabalho foram abstraídos através da utilização de um *framework* para mobilidade de código. Toda sincronização, comunicação e distribuição entre os nodos envolvidos no processamento é gerenciada pelo padrão de projeto disponibilizado pelo *framework*.

3. O Método Monte Carlo

O método de Monte Carlo [Mathews 2007] é um método estatístico de resolução de problemas. É baseado em números aleatórios e probabilidades estatísticas sendo frequentemente utilizado em problemas que não podem ser resolvidos analiticamente. O método consiste em calcular a probabilidade de um evento acontecer em determinadas condições. O computador é usado para gerar essas condições repetidamente e aleatoriamente. O número de vezes em que o evento acontece dividido pela quantidade de condições geradas será aproximadamente o valor da probabilidade calculada.

A aplicação do método neste trabalho acontece da seguinte maneira: supõe-se um círculo de raio R inscrito em um quadrado de lado 2R (Figura 1). A área do círculo é determinada por PI*R*R e a área do quadrado por 2R*2R. Então a razão da área do círculo para a área do quadrado é de PI/4. Sob uma visão focada no método, pode-se deduzir que essa razão é a probabilidade de um ponto qualquer do quadrado estar dentro da área do círculo.

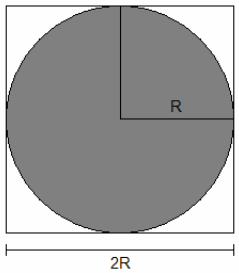

Figura 1. Círculo circunscrito em quadrado

A escolha do método Monte Carlo aqui não visa o cálculo mais rápido ou preciso do valor de PI, e sim avaliar a estabilidade e a eficiência da estrutura de suporte para as aplicações pervasivas, o GRADEp, bem como do *framework* que implementa o

padrão de projeto utilizado, avaliando suas funcionalidades em um contexto de alto desempenho.

3.1 A Aplicação

Na aplicação desenvolvida através do método Monte Carlo, o usuário entra com a quantidade de pontos que deseja gerar (N) e os nodos que serão incluídos no processamento, e é retornado como saída o valor de PI estimado e o tempo gasto no processamento. A aplicação pega aleatoriamente pontos de dentro do quadrado e testa cada um desses pontos se estão dentro do círculo. Ela sabe quando um ponto está dentro do círculo se X*X + Y*Y < R*R, onde X e Y são as coordenadas do ponto e R é o raio do círculo. A aplicação guarda a quantidade de pontos que foram gerados (N) e a quantidade de pontos que incidiram na área do círculo (C). Assim, o valor de PI pode ser aproximado por: PI/4 = C/N → PI = 4*C/N.

4. Padrão de Projeto para Mobilidade Utilizado

Este trabalho faz uso dos recursos providos pelo *framework* desenvolvido em [Vivan, Pernas and Yamin 2007], mais especificamente, o padrão de projeto utilizado foi o Mmap.

O padrão Mmap define um comportamento de *broadcast* (difusão). Esse padrão estrutura uma aplicação que envia uma tarefa a vários nodos de uma rede, contidos em uma lista. O resultado retornado pela execução do padrão também deve ser uma lista, do mesmo tamanho da lista de nodos, contendo o resultado da execução de cada tarefa (Figura 2).

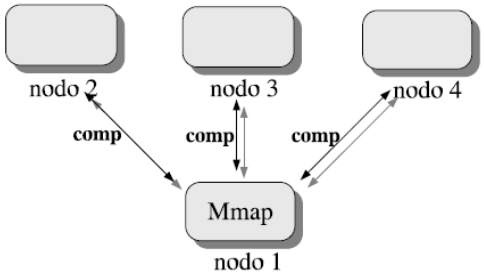

Figura 2. Comportamento do Mmap

Uma vez que o processamento em cada nodo e o conjunto de dados entrada é o mesmo para cada nodo, o comportamento de distribuição de processamento na forma de *multicast* foi preenchido adequadamente pelo Mmap, onde cada nodo recebe um número de pontos a ser gerado, e retorna quantos deles incidiram no círculo.

5. Testes

A aplicação foi executada em um ambiente formado por quatro nodos: uma base, que também participa da computação, e três nodos de processamento. Cada um com configuração: processador celeron 300MHz, RAM 64MB, HD 20GB, com sistema operacional Linux (distribuição DSL) e ligados em rede ethernet 10Mbps. Os tempos de execução e os *speed-ups* obtidos estão descritos na Tabela 1. Durante os testes, os nodos foram alocados exclusivamente para as execuções.

Tabela 1. Tempos de execução e *speed-ups* para 100.000.000 pontos

	1 nodo	2 nodos	3 nodos	4 nodos
Tempo de execução	289,203s	144,806s	97,152s	73,953s
Speed-up	1	~2	~2,98	~3,91

6. Conclusões

Os resultados preliminares obtidos sugerem que o GRADEp com a utilização do *framework* para mobilidade de código pode gerenciar aplicações de alto desempenho de forma satisfatória para um pequeno número de nodos.

O GRADEp mostrou-se funcional durante todas as execuções, bem como o *framework* utilizado no desenvolvimento da aplicação.

O *middleware*, sob a coordenação do *framework*, teve um desempenho satisfatório no gerenciamento da distribuição de processamento, onde os *speed-ups* obtidos com dois, três e quatro nodos foram quase ideais.

Para trabalhos futuros podem-se aprofundar os testes, levando em consideração outras variáveis como a estrutura de rede, assim como, estender o número de nodos, usar computadores com maior poder de processamento ou um *cluster*.

7. Referências

Fuggetta A., Picco G. and Vigna G. (1998) Understanding Code Mobility. In *IEEE Transactions on Software Engineering*, pages 342–361.

Gamma E., Helm R., Johnson R. and Vlissides J. (1995) Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley.

Grimm, R. et al. (2001) Systems Directions for Pervasive Computing. In *Workshop on Hot Topics in Operating Systems*, pages 147–151. IEEE Computer Society.

Mathews, J. (2007) "Numerical Analysis - Numerical Methods Project: Module for Monte Carlo PI", http://math.fullerton.edu/mathews/n2003/MonteCarloPiMod.html.

Satyanarayanan, M. (2001) Pervasive Computing: Vision and Challenges. In *IEEE Personal Communications*, pages 10-17.

Vivan G., Pernas A. and Yamin A. (2007) "Desenvolvimento de Aplicações Baseadas em Padrões de Projeto para Mobilidade de Código na Computação Pervasiva". Workshop em Sistemas Computacionais de Alto Desempenho – Concurso de Trabalhos de Iniciação Científica, Gramado, Brasil.

Yamin, A. (2004) "Arquitetura para um Ambiente de Grade Computacional Direcionado as Aplicações Distribuídas Móveis e Conscientes do Contexto da Computação Pervasiva". Tese de Doutorado, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil.