Integrando o Sistema *Terracotta* ao Ambiente *D-GM*[*]

Anderson Pinheiro[†], Gustavo Zechlinski, Adriano Maron [†], Renata Reiser[‡], Adenauer Yamin[‡]

[1]Centro Politécnico – Universidade Católica de Pelotas (UCPEL)

{a.boettge,gustavo.zechlinski,maron.adriano}@gmail.tche.br
{renata.reiser,adenauer}@gmail.tche.br

Resumo. *Este trabalho tem por objetivo realizar a integração do sistema DSM Terracotta ao Ambiente D-GM, disponibilizando uma estrutura de memória compartilhada distribuída implementada através do módulo ShareD-GM.*

1. Introdução

Este trabalho colaborou para a integração do sistema *DSM* (*Distributed Shared Memory*) *Terracotta*[Terracotta 2008] ao Ambiente *D-GM* (*Distributed Geometric Machine*). O Ambiente *D-GM* visa a simulação paralela de aplicações com ênfase na computação científica. As aplicações são modeladas no módulo de programação visual *VPE-GM* (*Visual Programming Environment for the Geometric Machine*)[Prestes et al. 2005], através da utilização de construtores de processos e da inicialização de estados indexados por posições em uma memória global. As aplicações são executadas no módulo de execução *VirD-GM* (*Virtual Distributed Geometric Machine*)[Fonseca 2008], o qual recebe as informações exportadas pelo módulo de programação e através de uma lista de nodos disponíveis realiza a distribuição das tarefas.

O resultado deste trabalho é a implementação do módulo *ShareD-GM* (*Shared Distributed Geometric Machine*) [Zechlinski 2010], disponibilizando a representação do espaço de memória compartilhada do Ambiente *D-GM*. A utilização do módulo *ShareD-GM* propicia um aumento de desempenho na execução de aplicações que exijam um alto poder computacional. Para este fim, o módulo *ShareD-GM* satisfaz a integração entre um sistema DSM chamado *Terracotta* e o Ambiente *D-GM*. Neste contexto, foram necessários estudos com relação aos sistemas *DSM* (*Distributed Shared Memory*) e a infraestrutura do módulo de execução *VirD-GM*, viabilizando a identificação de todos os componentes que realizam algum tipo de acesso a memória. As Seções 2 e 3 apresentam as características e informações básicas dos módulos *VPE-GM* e *VirD-GM*, enquanto que a Seção 4 refere-se ao módulo ShareD-GM resultante deste trabalho. Nas Seções 5 e 6 são detalhadas as avaliações realizadas e as considerações finais respectivamente.

2. Ambiente de Desenvolvimento *VPE-GM*

O *VPE-GM* é o módulo de programação visual das aplicações no Ambiente *D-GM*. Implementado na linguagem *Python*, gera arquivos no formato *XML* para comunicação externa. A menor unidade computacional consiste em um processo elementar (PE), caracterizado

[*]Projeto parcialmente financiado: Processo 476933/2007-2 Edital MCT/CNPq 15/2007 Universal Faixa A e Processo 502999/2008-0 Apoio Técnico

[†]Bolsista BIC/UCPEL

[‡]Centro de Desenvolvimento Tecnológico - Universidade Federal de Pelotas (UFPEL)

por alterar somente uma posição de memória. Seus principais componentes são: (i) Editor de Processos, onde as aplicações são modeladas e (ii) Editor de Memória, possibilitando a configuração do estado inicial da aplicação.

3. Ambiente *VirD-GM*

O *Vird-GM* é o módulo de execução distribuída para o Ambiente *D-GM*. Este módulo é constituído por três principais componentes, responsáveis pela execução dos processos que fazem parte das aplicações: (i) **Loader**, faz a interpretação dos arquivos descritores; (ii)**Launcher** - realiza o controle do fluxo de execução e o escalonamento das tarefas; e (iii) **Exec** - promove o envio e recebimento das informações sobre cada processo para os VirD-nodos (nodos do *cluster* com uma instância do *VirD-GM*). O controle do fluxo de execução ocorre através do emprego de uma matriz de adjacências, impedindo que dois processos concorrentes executem ao mesmo tempo. A matriz de adjacências é utilizada conjuntamente com as listas de controle, um mecanismo usado para distribuição de processos bloqueados ou livres para execução.

Na implementação anterior ao módulo *ShareD-GM*, utilizava-se o componente $Exec$ para distribuir o objeto $virdMemory$, cuja função é representar a memória do módulo de execução do Ambiente D-GM. Neste contexto, sempre que um processo é selecionado para execução em algum nodo do *cluster*, o objeto $virdMemory$ é adicionado aos dados do processo e enviado via *sockets* para o referido nodo. Está técnica agrega um alto custo de comunicação para memórias extensas. Visando reduzir este custo, buscam-se melhorias referentes ao compartilhamento do objeto empregando o sistema *Terracotta*.

4. Módulo *ShareD-GM*

O módulo *ShareD-GM* tem como principal função fornecer uma estrutura de memória compartilhada e distribuída para o Ambiente *D-GM*, através da integração do sistema *Terracotta* ao módulo de execução *VirD-GM*. O sistema *Terracotta* caracteriza-se por não exigir grandes modificações no código das aplicações as quais esta sendo integrado. Esta integração é realizada através de um arquivo no formato XML, no qual são configurados os parâmetros das aplicações necessários para que o sistema *Terracotta* possa gerenciar a memória compartilhada.

As principais modificações realizadas estão relacionadas ao envio e o recebimento do objeto $virdMemory$, ambas funções desempenhadas pela classe $ExecImpl$ do componente $Exec$ através dos métodos $send$ e $exec$. O método $send$ é acionado durante o escalonamento das tarefas no $Launcher$ sendo o responsável por transmitir os parâmetros de um determinado processo para o $VirD-nodo$ disponível. Por sua vez, o método $exec$ no lado cliente realiza uma chamada para a biblioteca com os parâmetros recebidos. Ambos métodos deixaram de receber o objeto $virdMemory$ como parâmetro. Na Figura 1 têm-se a arquitetura do ambiente *VirD-GM* contendo o módulo *ShareD-GM*.

5. Estudo de Caso

O objetivo deste estudo de caso é exclusivamente validar a implementação da integração do módulo *VirD-GM* ao sistema *Terracotta*. Os ganhos de desempenho obtidos serviram como incentivo para a realização de outros estudos de caso com o intuito de consolidar esta tendência.Esta Seção apresenta a implementação do Método de *Smith-Watermam*

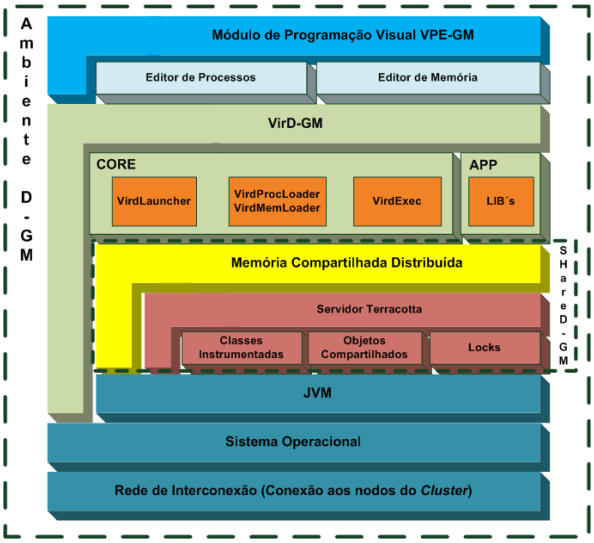

Figura 1. Arquitetura do Ambiente *VirD-GM* [Zechlinski 2010]

[Waterman and Smith 1981], cuja principal função é calcular a similaridade entre duas sequências de caracteres (ex. DNA).

Para este algoritmo adota-se o método da programação dinâmica para realizar a comparação entre uma sequência desconhecida e as presentes em um banco de dados. Dessa forma, uma matriz é criada dinamicamente, atribuindo-se uma nota a cada comparação. Considere duas sequências $S = s_1 s_2 s_3 ... s_m$ e $T = t_1 t_2 t_3 ... t_n$ de tamanhos m e n, respectivamente. A solução é encontrada através da matriz $D_{(m+1)(n+1)}$ onde cada entrada indexada pelas variáveis i e j é dada através de relações concorrentes [Waterman and Smith 1981], com base nos caracteres das sequências. O passo seguinte consiste na localização do maior valor presente na matriz e realizar o "caminho de volta" a partir deste valor máximo, seguindo os ponteiros armazenados para cada valor na matriz (*Backtracking Pointers*). Esses ponteiros tem a função de indicar a posição do próximo valor máximo, considerando as posições anteriores adjacentes a posição atual.

A modelagem do algoritmo foi desenvolvida no módulo *VPE-GM*, seguindo as normas do modelo *GM*. Para a realização dos testes, adotou-se um banco de dados com mil sequências, ou seja, a aplicação deverá realizar mil comparações e armazenar o resultado de cada uma em uma posição de memória. Em resumo a aplicação utiliza duas funções "SW" e "RES", responsáveis pelos cálculos referentes as comparações e a apresentação do resultado final, respectivamente. Ambas funções foram implementadas e adicionadas a biblioteca de funções do módulo *VirD-GM*. Ressalta-se que a função "SW" foi implementada de acordo com [Waterman and Smith 1981].

5.1. Resultados Obtidos

Para esta aplicação, utilizou-se um *cluster* de no máximo quatro nodos cada um com dois *VirD-nodos* instanciados, com uma configuração homogênea: *Intel Dual Core 1600 MHz; 2048 MBytes de RAM; Ubuntu 9.04; Fast Ethernet.* Foram realizadas vinte iterações para cada configuração. Os tempos médios alcançados para simulação são apresentados nas Tabelas 1 e 2, representando a versão sem o módulo *ShareD-GM* e a versão com o módulo, respectivamente. Em cada experimentação são executados mil e um processos.

Tabela 1. Tempos de Simulação *Smith-Watermam* (*VirD-GM*)

Nº VirD-Clients	*Alocação de Nodos*	*T.M.de Execução (s)*	*Desvio Padrão*	*Speedup*
1	N1	148,341	1,284	-
4	N1, N2	76,761	1,115	1,93
8	N1, N2, N3, N4	74,623	0,598	1,99

Tabela 2. Tempos de Simulação *Smith-Watermam* (*ShareD-GM*)

Nº VirD-Clients	*Alocação de Nodos*	*T.M.de Execução (s)*	*Desvio Padrão*	*Speedup*
1	N1	58,721	0,252	-
4	N1, N2	24,805	1,017	2,36
8	N1, N2, N3, N4	18,345	1,592	3,2

As tabelas demonstram claramente o menor tempo de execução da aplicação utilizando o módulo *ShareD-GM*. É possível observar também que o tempo de simulação de quatro para oito nodos não possui uma diferença significativa, esse comportamento pode acontecer por diferentes fatores, como custo de comunicação maior que o custo de computação de cada tarefa e a forma de escalonamento atual do módulo *VirD-GM*. Neste caso, o custo de cada tarefa poderia ser aumentado, cada processador passaria a calcular mais de uma sequência, reduzindo a comunicação entre os nodos.

6. Considerações Finais

Os resultados obtidos até o momento são satisfatórios demonstrando o poder de comunicação inter-processos disponibilizado pelo sistema *Terracotta*. Como trabalhos futuros consideram-se otimizações no escalonador do módulo *VirD-GM*, e também em relação a integração com o sistema *Terracotta*. Atualmente está sendo realizada a integração do módulo *VirD-GM* ao módulo de desenvolvimento *VPE-qGM* (*Visual Programming Environment for the Quantum Geometric Machine*).

Referências

Fonseca, V. S. d. (2008). VirD-GM: Uma contribuição para o modelo de distribuição e paralelismo do Projeto D-GM. Dissertação de mestrado em ciência da computação, UCPel, Pelotas,RS.

Prestes, D., Cardoso, M., Reiser, R., and Costa, A. (2005). Extending the geometric machine model to a visual programming environment. *CLEI05 XXXI*, pages 1–10.

Terracotta (2008). *The Definitive Guide to Terracotta: Cluster the JVM for Spring, Hibernate and POJO Scalability.* Apress.

Waterman, M. S. and Smith, T. F. (1981). Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197.

Zechlinski, G. M. (2010). ShareD-GM: Uma proposta de aplicação de sistemas DSM no ambiente D-GM. Dissertação de mestrado em ciência da computação, UCPEL - Universidade Católica de Pelotas.