Arquitetura de Software da VirD-GM: Modelagem e Funcionalidades

Felipe Munhoz, Vanessa Fonseca, Guilian Vivan, Renata Reiser, Adenauer Yamin

[1]Escola de Informática – Universidade Católica de Pelotas (UCPel)
Rua Felix da Cunha, 412 – Pelotas – RS – Brazil

{vandag,reiser,adenauer}@ucpel.tche.br

Resumo. *A proposta central do Projeto D-GM é a construção da extensão distribuída do modelo de Máquina Geométrica. Obtém-se a execução distribuída das computações pelas especificações de processos e memória compartilhada, visualmente geradas pelo emprego de editores gráficos do ambiente de programação visual VPE-GM. Este trabalho introduz a implementação da arquitetura de software denominada* Virtual Geometric Machine Model.

1. Introdução

O Projeto *Distributed Geometric Machine Model* (D-GM) [Reiser et al. 2004] tem como principal meta a integração da ferramenta *Visual Programming Environment for the Geometric Machine Model* (VPE-GM) [Prestes 2005] com o ambiente de execução para computação paralela e distribuída *Execution Environment for High Distributed Applications*(EXEHDA) [Yamin et al. 2005]. A perspectiva é prover um suporte à exploração da concorrência quando da execução de processos representados no modelo GM. Para alcançar esta meta implementa-se a arquitetura de software, denominada *Virtual Geometric Machine Model* (VirD-GM), de acordo com [Fonseca et al. 2007]. Assim, a utilização de recursos gráficos construídos por expressões visuais no ambiente VPE-GM viabiliza a construção dos processos para o modelo D-GM, permitindo também a validação dos parâmetros que serão entregues a VirD-GM para controle e execução das computações.

Quanto à organização do texto, para alcançar uma visão de abstração de mais alto nível, na Seção 2, apresentam-se os diagramas dos principais componentes de software da VirD-GM e seus relacionamentos. A descrição das classes de implementação e suas principais interações são apresentados na Seção 3, caracterizando-se assim, uma visão direcionada da implementação já efetivada. Finalmente, na Seção 4, encerra-se o texto com os trabalhos em andamento e as considerações finais.

2. Principais Componentes de Software da VirD-GM

Nesta seção, apresenta-se a visão dos componentes funcionais e lógicos da VirD-GM, viabilizando uma visão do processo de implementação de maior abstração. As etapas mais significativas desenvolvidas para alcançar a execução dos processos no modelo D-GM são brevemente descritas. Observa-se na Figura 1 o diagrama dos principais componentes da VirD-GM, os quais estão descritos logo a seguir.

2.1. Loader VirD-GM

O módulo *Loader VirD-GM* recebe os arquivos em XML exportados pelo ambiente VPE-GM, descrevendo além das estruturas de processos e de memória, os parâmetros para

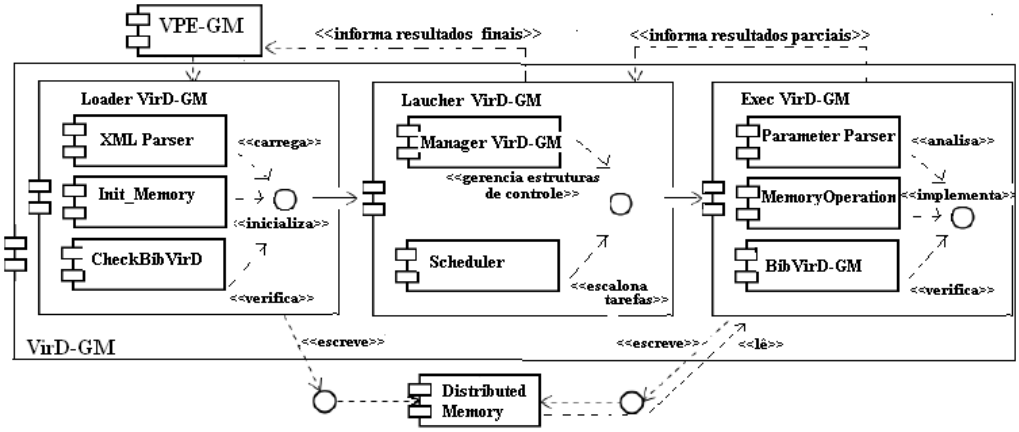

Figura 1. Diagrama de Componentes de Software para a VirD-GM

execução. Integram este módulo os seguintes componentes de software: (i) *XML Parser* que realiza o mapeamento do arquivo XML para estruturas internas da arquitetura; (ii) *InitMemory* responsável por instanciar e inicializar a memória compartilhada, sendo então atualizada com os valores obtidos após o mapeamento do XML; (iii) *CheckBib VirD-GM* que verifica a disponibilidade de bibliotecas auxiliares nos repositórios de funções, as quais são requeridas pelos processos descritos no arquivo XML. Consideram-se também inclusas as atividades de preparação da arquitetura assim como a criação da matriz de adjacência modelando o fluxo de execução.

2.2. Launcher VirD-GM

O módulo *Launcher VirD-GM* tem como principal funcionalidade o gerenciamento do controle e do disparo da execução, após mapeamento do arquivo XML para as estruturas internas da arquitetura que ocorre no módulo *Loader VirD-GM*. Dentre seus componentes, destacam-se: (i) o componente *Manager VirD-GM* que gerencia o controle da execução da computação baseado nos dados obtidos quando do acesso à matriz de adjacências; em seqüência, após esta verificação o *Manager VirD-GM* insere os processos na lista de execução; (ii) o componente *Scheduler*, responsável pelo escalonamento dos processos já inclusos na já referida lista de execução. Este escalonamento está baseado em políticas, previamente definidas quando da especificação da arquitetura do modelo D-GM.

2.3. Exec VirD-GM

A funcionalidade do módulo *Exec VirD-GM* consiste na execução das computações paralelas nos nodos remotos da célula, pela utilização de serviços do middleware EXEHDA. Esta execução tem suporte em bibliotecas auxiliares, previamente definidas pela aplicação em desenvolvimento. Para tal, faz-se necessário o uso de operações de acesso à memória compartilhada. Consideram-se os componentes: (i) *Parameter Parser* que é um analisador dos parâmetros de entrada e saída dos processos de acordo com a sintaxe associada a cada tipo de aplicação; e (ii) o componente *Memory Operation*, tendo como funcionalidade o acesso à memória distribuída e a realização de operações de leitura e escrita, de acordo com as posições definidas nos parâmetros de entrada e saída; (iii) e o componente

Bib VirD-GM, a qual permite acesso ao repositório de funções associadas às operações dos processos do modelo D-GM.

3. Diagramas de Classes de Implementação da VirD-GM

A definição das classes de implementação está baseada na visão dos componentes apresentados na seção anterior. Consideram-se as classes de implementação e a modelagem das interações no correspondente diagrama de classe da linguagem UML, conforme mostra a Figura 2. Utiliza-se uma versão simplificada das classes geradas pela Ferramenta [IDE-NetBeans 2007], a partir do código já implementado da VirD-GM. As funcionalidades das correspondentes classes de implementação são brevemente descritas.

- Na classe *VirDProc-Loader*, que implementa o componente *Loader VirD-GM*, é construído o grafo dirigido que representa o fluxo de execução de processos, viabilizando o mapeamento do arquivo de entrada XML. Após a construção deste grafo, ocorre a estruturação de sua representação como matriz de adjacência, consistindo na estrutura de ordenação do fluxo de execução. O atributo *inputFile* representa o arquivo a ser carregado, o qual está associado ao construtor da classe *VirDProc-Loader*.
- *VirDMemLoader* consiste na classe da VirD-GM que estrutura a memória, de acordo com o arquivo descritor de memória.
- A classe *VirDGraph* representa uma estrutura de grafo, em concordância com os requisitos da arquitetura VirD-GM. A ação inicial do construtor da classe é a geração de um grafo base, cujos nodos são, dinamicamente, gerados pelo arquivo de entrada.
- A classe *VirDNode* define um nodo do grafo da VirD-GM, representando um processo elementar no modelo D-GM. Caracteriza-se como uma agregação da classe *VirDGraph*, ou seja, os objetos da primeira completam as informações referentes os objetos da segunda.
- A classe de implementação *VirDLauncher*, que implementa o componente *Launcher VirD-GM*, recebe como parâmetro a estrutura da matriz de adjacência e, assim, ocorre a verificação de quais processos podem ser executados, respeitando-se as dependências e a exclusão mútua no acesso à memória. Os processos liberados são inseridos em uma lista de processos para execução, a qual será lida pelo escalonador. O escalonamento é definido através de políticas pré-definidas. Esta classe utiliza os serviços do middleware EXEHDA, para criação de objetos remotos (*Executor*) e para chamada destes métodos (*Worb*).
- A classe abstrata *VirDExec*, associada ao componente *Exec VirD-GM*, define métodos que devem ser, obrigatoriamente, implementados por suas classes filhas.
- A classe *VirDExecImpl*, implementando a classe abstrata *VirDExec*, é responsável pela análise dos parâmetros de entrada e saída dos processos e pelo posterior acionamento das funções presentes na biblioteca da VirD-GM. Além destes, para garantir execução das computações, são também consideradas as operações de leitura e escrita na memória compartilhada.
- A classe *VirDProc-Elem* provê representação para um processo elementar, cujos atributos estão associados a correspondente definição no modelo D-GM, como a ação (operação aritmética) e uma posição de escrita na memória (modelada por um ponto no espaço geométrico).

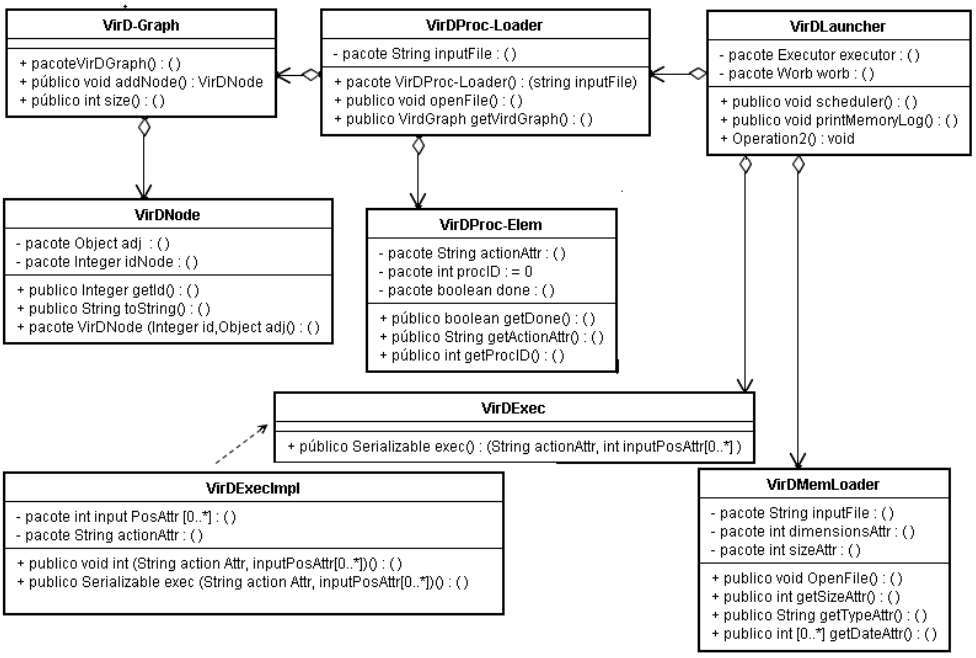

Figura 2. Diagrama de Classes para a VirD-GM

4. Considerações Finais

A implementação da arquitetura VirD-GM deverá facultar a execução distribuída e concorrente dos programas concebidas segundo o modelo D-GM, usando como suporte o middleware EXEHDA. A definição da visão funcional dos componentes da VirD-GM, previamente modelados em [Fonseca et al. 2007], orientou a implementação das classes e suas respectivas interações. Como trabalhos futuros, destacam-se a otimização do módulo de controle de execução e ampliação da biblioteca de funções da VirD-GM e a complementação da integração dos módulos de controle ao *middleware* EXEHDA.

Referências

Fonseca, V. S., Reiser, R. H. S., Yamin, A. C., and Pilla, M. L. (2007). Vird-gm: Towards a grid computing environment. In *Proceedings of CCGRID 2007*, pages 1–6.

IDE-NetBeans (2007). Netbeans modular, standard-based integrated development environment. Disponível via WWW em http://www.netbeans.org. Acessado em 18/01/08.

Prestes, D. G. (2005). Ambiente de programação visual para o modelo de máquina geométrica. Monografia de graduação, ESIN/UCPEL, Pelotas/RS.

Reiser, R., Costa, A. C. R., and Dimuro, G. (2004). Distributed approach for the geometric machine model. In *PARA 2004 - Workshop on State-of-the-art in Scientific Computing Valideted*, number 1, pages 106–114, Ligby.

Yamin, A. C., Augustin, I., Barbosa, J., Silva, L., Real, R., Schaffer, A., and Geyer, C. (2005). Exehda: adaptive middleware for building a pervasive grid environment. In Czap, H., Unland, R., Branki, C., and Tianfield, H., editors, *Frontiers in Artificial Intelligence and Applications - Self-Organization and Autonomic Informatics*, volume 135, pages 203–219. IOS Press.