Modelando o Paralelismo na Arquitetura VirD-GM

Vanessa Souza da Fonseca[1], **Renata Hax Sander Reiser**[1], **Adenauer Correa Yamin**[1]

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

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

1. Introdução

O Projeto D-GM apresenta uma plataforma para desenvolvimento e execução distribuída de aplicações modeladas na Máquina Geométrica (GM) [Reiser et al. 2004]. A principal meta é permitir aos projetistas de software a explicitação do paralelismo sem ter de se preocupar com a especificação de soluções particulares, mediante os recursos disponíveis. Este trabalho implementa a arquitetura de software denominada *Virtual Geometric Machine Model* (VirD-GM) para o ambiente de execução da D-GM [Fonseca et al. 2007], integrando o ambiente de desenvolvimento VPE-GM (D-GM Visual Programming Environment) [Prestes et al. 2005] com o middleware EXEHDA (Execution Environment for Highly Distributed Applications) [Yamin et al. 2005], veja Figura 1(A).

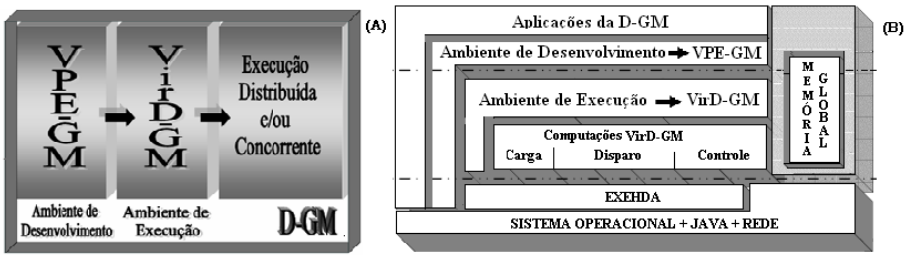

Figura 1. Visão Funcional e Arquitetural da D-GM

2. Modelo D-GM e Arquitetura VirD-GM

Dentre as propriedades do Modelo D-GM, salientam-se: (i) facilidade provida pela abordagem visual, com manipulação direta dos objetos gráficos e construtores; (ii) composição funcional dos construtores (produto seqüencial e paralelo, processos condicionais e somas não-determinística) quando do desenvolvimento de programas; (iii) metodologia indutiva, onde construtores são aplicados sobre processos definidos por ações indexadas por posições da memória global e compartilhada, viabilizando uma medida de custos computacionais; (iv) independência arquitetural em relação às aplicações do modelo D-GM; (v) semântica explícita e inerente ao ambiente de programação visual, apresentando aspectos da semântica do modelo D-GM embora não formalmente solicitados pelo programador. O nível de abstração do paralelismo no modelo D-GM é totalmente explícito, as construções como assinalamento e decomposição são especificadas quando da construção do programa, na interface gráfica dos editores de processos e memória da VPE-GM, considerando sua notação posicional. O mapeamento, a sincronização de tarefas e o gerenciamento do conflito de acesso à memória é controlado e interpretado

pela função-posição, recursivamente definida pelo conceito de envelopes, os quais implementam as construções sobre processos elementares (cuja execução altera apenas uma posição de memória). O espaço de endereçamento da memória compartilhada está caracterizado por uma estrutura matricial, multidimencional, e com topologia compatível com a estruturação do espaço geométrico associado ao modelo D-GM. A sincronização de processos é obtida de forma direta na interface gráfica, possibilitando melhor uso e flexibilidade de recursos, incluindo também reconfiguração em caso de falha, quando das descrições gráficas do processo. A estrutura paralela da memória pode ser compartilhada pelos processadores, permitindo a execução simultânea de dados com vários processos em execução, de forma CREW (*Concurrent Read Exclusive Write*).

Sob a óptica da arquitetura VirD-GM, apresentada na Figura 1(B), tem-se uma organização em camadas lógicas, com níveis diferenciados de abstração, destacando-se:
• a camada de aplicação, disponibilizando as abstrações para programação de aplicações distribuídas, na área da computação científica, cuja natureza poderá compreender computações estocásticos, intervalares e quânticas;
• a camada de suporte e o ambiente de execução, concentrando os serviços da VirD-GM, que estendem as funcionalidades do EXEHDA para atendimento da especificações do modelo D-GM. Os módulos de carga, disparo e controle de execução de computações paralelas e distribuídas estão representados neste nível;
• a camada de sistemas básicos, composta pelos sistemas e linguagens nativas que integram o meio físico de execução, consistindo na plataforma base de implementação e caracterizada pela Máquina Virtual Java em suas diferentes abordagens.

3. Considerações Finais

Diversos desafios foram vencidos, destacando-se: (i)reavaliação da expressividade dos mecanismos textuais e visuais providos pela VPE-GM; (ii)construção dos módulos de tradução e gerenciamento de código produzido no ambiente VPE-GM para a arquitetura de software do middleware EXEHDA; (iii)implementação do fluxo de controle e execução na VirD-GM, baseado na aplicação de grafos de dependência. A atual fase concentra-se na implementação da memória global e compartilhada. E, como trabalhos futuros, tem-se a implementação de políticas de escalonamento e mecanismos de heurísticas quando da execução de aplicações distribuídas na computação científica.

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.

Prestes, D., Reiser, R., Costa, A., and Cardoso, M. (2005). Estendendo o modelo de máquina geométrica a um ambiente de programação visual. In *CLEI2005 XXXI*, pages 1–10, Cali. Universidad Javeriana. meio digital.

Reiser, R., Costa, A. C. R., and Dimuro, G. (2004). Distributed approach for the geometric machine model. In *PARA 2004*, 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 *Frontiers in Artificial Intelligence and Applications - Self-Organization and Autonomic Informatics*, volume 135, pages 203–219. IOS Press.