Modelando Não-Determinismo e Iterações no VirD-GM

Felipe Munhoz, Renata Reiser, Adenauer Yamin

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

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

Resumo. *A principal contribuição deste trabalho é a modelagem de extensões para o VirD-GM (*Virtual Distributed Geometric Machine Model*), centrada em duas principais metas: (i) disponibilizar a execução de computações não-determinísticas; e (ii) prover a execução de estruturas de controle para iteração, as quais introduzem o construtor de* `Iteração Seqüencial`*, caracterizado pela inserção de dependências entre os processos, e o construtor de* `Iteração Paralela`*, baseado na independência de dados entre os processos.*

1. Introdução

Este trabalho está inserido no contexto do projeto D-GM, que têm por objetivo contribuir com um ambiente para o desenvolvimento e execução distribuída de aplicações para a computação científica, considerando o ambiente de desenvolvimento visual denominado VPE-GM [Prestes et al. 2005] *(Visual Programming Environment for the Geometric Machine Model)*, no qual são modeladas as aplicações, e o ambiente de execução distribuída caracterizado pelo VirD-GM *(Virtual Distributed Geometric Machine Model)* [Fonseca 2008]. Ambos componentes de software fundamentam-se nas abstrações no modelo GM *(Geometric Machine)* [Reiser 2002] quando de sua construção. Assim, aplicações desenvolvidas no VPE-GM são exportadas através de um arquivo descritor da aplicação, em XML, possibilitando que o VirD-GM execute a aplicação de forma distribuída. O suporte à execução distribuída é implementado sobre o *middleware* EXEHDA *(Execution Environment for High Distributed Applications)* [Yamin et al. 2005], o qual provê abstrações inerentes ao processamento paralelo e/ou distribuído. A especificação do paralelismo nas aplicações desenvolvidas é feita de forma explícita pelo programador [Dongarra et al. 2003] no ambiente VPE-GM. Por sua vez o ambiente de execução VirD-GM analisa a aplicação e gerencia a disponibilidade de recursos computacionais para a execução. Em trabalhos anteriores [Fonseca 2008], a especificação arquitetural do ambiente de execução VirD-GM teve por objetivo prover suporte à execução de `processos elementares` e a dois tipos de construtores da GM: `processos seqüenciais` e `processos paralelos`, já validados quando do desenvolvimento de aplicações [Munhoz et al. 2008]. Considerando este cenário, como motivação para este trabalho, tem-se a extensão do VirD-GM para o suporte às computações não-determinísticas e estruturas de repetição de acordo com a modelagem do construtor `Soma Não-Determinística` e do construtor `Iterativo`, respectivamente.

2. Construtor Soma Não-Determinística

A estrutura dual ao construtor produto paralelo que modela o conflito de acesso à memória GM corresponde ao construtor `Soma Não-Determinística` (SND), tendo como

principal característica a aleatoriedade da escolha dentre dois ou mais processos GM a serem executados.

Pela aplicação do construtor `Soma Não-Determinística` a dois processos elementares d^k e e^k, tem-se a construção de processos mutuamente exclusivos, os quais estão definidos pelo mesmo índice k, representando a posição conflitante de escrita na memória, e pelas operações distintas d e e. Quando de sua execução, tem-se que a troca de valor da variável associada à *k-ésima* posição de memória ocorre de forma aleatória. A Figura 1 representa o processo resultante da aplicação do construtor `Soma Não-Determinística` sobre os processos elementares $d^{(k)}$ e $e^{(k)}$.

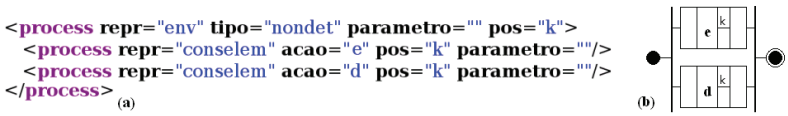

Figure 1. Construtor Soma Não-Determinística

No diagrama de atividades apresentado na Figura 2, resume-se o comportamento do construtor SND. Neste caso, tem-se:

- *Cria Lista de Processos:* Esta etapa é responsável pela interpretação do arquivo XML, referente aos parâmetros que descrevem o construtor SND. Os parâmetros que definem a lista de processos referentes ao construtor podem ser processos elementares ou composições envolvendo outros construtores, neste caso correspondendo a estruturas de dados do tipo `Envelope` [Prestes et al. 2005]. A Figura 2B apresenta o arquivo XML gerado no ambiente VPE-GM do construtor SND, o qual contém três processos;
- *Define Intervalo de Seleção:* Nesta etapa, determina-se o intervalo no qual será realizada a seleção aleatória de processo elementar ou composição de construtores de processos (Envelope) a ser enviado para lista de execução (Veja Figura 2);
- *Seleciona Processo:* Ocorre a escolha do processo a ser executado na posição de conflito, com base no método aleatório e no intervalo de seleção;
- *Adiciona Processo Selecionado:* Finalizando, ocorre o envio do processo selecionado para a lista de processos a serem executados, incluindo a verificação de suas dependências.

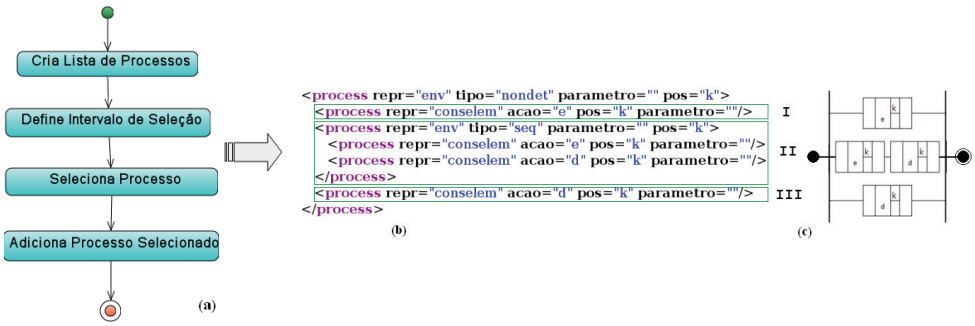

Figure 2. Diagrama de atividades do construtor Soma Não-Determinística.

3. Construtor Iterativo

De acordo com as abstrações do modelo D-GM, e correspondente modelagem na VPE-GM, a aplicação do construtor `Iterativo` ocorre a partir da estrutura do tipo envelope, caracterizando a repetição de operações, com número de repetições variável e definido por parâmetro de controle (tipo inteiro n) no momento de sua instanciação.

Originalmente, o construtor `Iterativo` já modelado no VPE-GM considerava somente a construção seqüencial de processos. Neste trabalho propõe-se duas extensões deste construtor: (i) a iteração paralela de processos, os quais não possuem dependência de dados entre os processos; e (ii) a iteração referente à posição da memória. Neste caso, na variável de controle indica-se não o número de iterações do construtor mas um intervalo de valores. Assim, a execução ocorre não apenas na primeira posição de memória indicada no processo elementar, e sim nas posições definidas no intervalo indicado pela variável de controle do construtor.

A Figura 3(a) mostra um diagrama de atividades que descreve o funcionamento do construtor de `Iteração Seqüencial`, cuja principal característica é a inserção de dependências entre cada uma das iterações, ou seja, a n-ésima iteração depende da correspondente execução da iteração $n - 1$. Mais especificamente, tem-se:

- *Verifica Dependências Externas:* com verificação de parâmetros para execução do processo referente à corrente iteração.
- *Testa Fim da Iteração:* realiza controle do número de iterações considerando duas etapas, descritas na seqüencia, caso o número de iterações não seja satisfeito:
  - verificação de dependências internas, diferenciando a estrutura de controle nos dois tipos de construtores `Iterativos`.
  - inserção do processo correspondente à iteração atual em uma lista de armazenamento temporário, já com as dependências de dados resolvidas.
- *Retorna Processos Iterados:* quando finaliza o processo de iteração, os processos gerados na iteração são enviados para a lista de processos de execução.

Salienta-se que a principal diferença no funcionamento do construtor de `Iteração Sequencial` para o construtor de `Iteração Paralela` é a inexistência de dependências entre cada uma das iterações, ou seja, as iterações podem ser executadas concorrentemente. Na Figura 3(b), verifica-se que o diagrama de atividades modelando o funcionamento da `Iteração Seqüencial` é análogo ao construtor de `Iteração Paralela`, exceto pela etapa de verificação de dependências internas, garantindo neste caso, a não ocorrência de conflito no acesso às posições de memória dos correspondentes processos [Rose and Navaux 2003].

4. Considerações Finais

O trabalho introduziu estruturas de controle de iteração ao VirD-GM, incluindo uma análise do não-determinismo. Obteve-se a modelagem do construtor Soma Não-Determinística e do construtor `Iterativo`, que nortearam a implementação dos mecanismos de gerência da execução do VirD-GM para atender as demandas introduzidas por estes novos construtores.

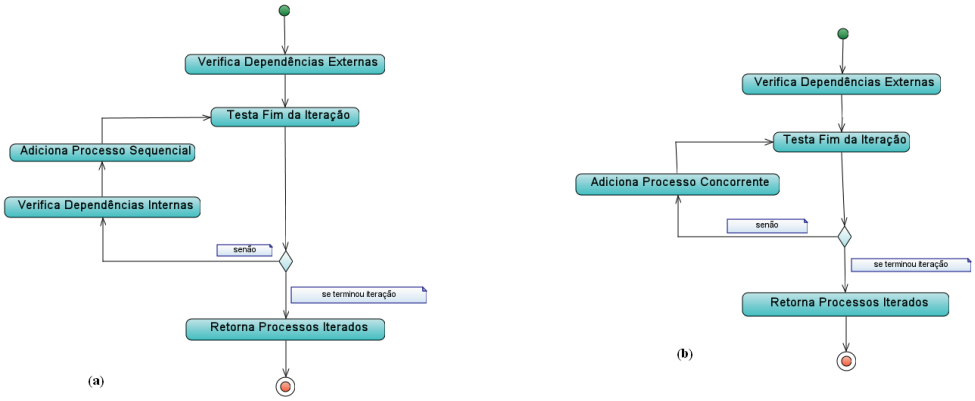

Figure 3. Diagrama de atividades da Iteração Seqüencial e Paralela.

Como continuidade da proposta de colaborar no Projeto D-GM, tem-se a modelagem e implementação do construtor Soma Determinística e da estrutura para suporte aos tipos de teste (universal, existencial e determinístico) disponibilizados nas interfaces do VPE-GM. Assim também, somam-se novos construtores relacionados com a geração de processos e alocação de memória de natureza dinâmica. Consideram-se ainda a construção dos módulos de suporte aos novos construtores no VirD-GM e indicação de adaptação nas interfaces do VPE-GM. Visando a validação dos construtores, são relevantes em etapas futuras as aplicações da computação científica no VirD-GM, independente do núcleo de gerência e, as quais possibilitem uma abstração nas etapas de desenvolvimento do Projeto D-GM.

References

Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L., and White, A. (2003). *Sourcebook of parallel computing*. Morgan Kaufmann Publishers Inc., San Francisco.

Fonseca, V. S. (2008). Vird-gm: Uma contribuição para o modelo de distribuição e paralelismo do projeto d-gm. Dissertação de mestrado, Mestrado em Ciência da Computação/UCPel, Pelotas/RS.

Munhoz, F. N., Fonseca, V. S., Reiser, R. H. S., Pilla, M. L., and Yamin, A. C. (2008). Modelo d-gm: Explorando o paralelismo na computação científica. In *ERMAC*, pages 1–10.

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 Conferência Latinoamericana de Informática*, pages 1–10, Cali. Universidad Javeriana. meio digital.

Reiser, R. (2002). *A máquina geométrica - um modelo computacional para concorrência e não-determinismo usando como estrutura os Espaços Coerentes*. PhD thesis, CPGCC/UFRGS - POA.

Rose, C. A. F. D. and Navaux, P. O. A. (2003). *Arquiteturas paralelas*. Editora Sagra-Luzzato, Porto Alegre.

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.