GraphProgramming: Ferramenta de Auxílio a Programação para Ambientes Paralelos e Distribuídos

Felipe L. Severino[1], Marcelo T. Rebonatto[1]

[1]Instituto de Ciências Exatas e Geociências – Universidade de Passo Fundo (UPF)
Passo Fundo – RS – Brasil

felipelseverino@gmail.com, rebonatto@upf.br

1. Introdução

A utilização de recursos computacionais de alto desempenho, em muitos casos, é realizada por pesquisadores de outras áreas do conhecimento. A dificuldade encontrada por este tipo de usuário pode ser equiparada à dificuldade encontrada por estudantes da disciplina de Processamento Paralelo e Distribuído. Desta forma, pode-se recomendar o uso de ferramentas para auxiliar o desenvolvimento de programas paralelos ou distribuídos.

De forma semelhante, a utilização de recursos visuais para representação da estrutura do programa pode facilitar a compreensão, desenvolvimento e depuração de programas paralelos e distribuídos [MALACARNE 1999]. Uma das principais formas de representação gráfica de programas paralelos e distribuídos é a utilização de grafos, onde os nodos representam os elementos de processamento e as arestas representam os elementos de comunicação.

Este artigo apresenta a ferramenta *GraphProgramming*, uma ferramenta *opensource* para auxílio ao desenvolvimento de programas para ambientes paralelos e distribuídos.

2. Projeto de desenvolvimento

Buscando facilitar o processo de desenvolvimento de aplicações para ambientes paralelos e distribuídos foi desenvolvido um projeto para uma ferramenta de programação visual utilizando, em um primeiro momento, o padrão *Message Passing Interface* (MPI) para o desenvolvimento. Apesar de inicialmente suportar apenas o padrão MPI, a ferramenta deve ser capaz de permitir a inclusão futura de outras tecnologias para programação para ambientes paralelos e distribuídos.

O principal objetivo da ferramenta é a geração do código-fonte da aplicação desenvolvida. Este código deverá ser gerado através de uma estrutura criada pelo usuário, que deverá utilizar recursos visuais para a criação desta estrutura. Estes recursos gráficos são representados analogamente a grafos, onde os elementos de processamento são representados pelos nodos e os elementos de comunicação são representados pelas arestas.

2.1. Elementos de processamento

Cada nodo da estrutura deverá conter dados referentes ao elemento de processamento que representa, como os códigos relativos à computação ou o número de

processo. Os nodos podem ser de três tipos: nodos que representam um único processo (P), nodos que representam todos os processos (N) e nodos que representam todos os processos exceto o processo *root* (N-1).

Dois nodos, delimitando o início e o término da estrutura da aplicação desenvolvida, devem ser inseridos automaticamente pela ferramenta. Estes nodos devem possuir as rotinas de inicialização e término do ambiente.

2.1. Elementos de comunicação

Cada elemento de comunicação deverá conter dados acerca da comunicação que representa, como nome da variável de envio e recebimento, tipo de dado comunicado, entre outros. As comunicações são divididas em dois grupos: comunicação ponto-a-ponto e coletiva, e os tipos de dados enviados podem ser escalares ou vetoriais.

As comunicações ponto-a-ponto representam comunicações diretamente entre dois processos. Normalmente este tipo de comunicação envolve dois nodos do tipo P (comunicação P para P) ou o retorno de um dado por todos os processos exceto o processo *root* (comunicação N-1 para P), mas pode ser utilizado para enviar um grupo de dados distintos de um processo para todos exceto o *root* (comunicação P para N-1). Este tipo de comunicação é análoga às funções MPI_Send e MPI_Recv no padrão MPI.

As comunicações coletivas têm como característica o envolvimento de mais de um processo por vez. As comunicações coletivas são classificadas como: comunicação de um único dado ou de um mesmo grupo de dados do processo *root* para todos os demais (comunicação de P para N-1), sendo este análogo a função MPI_Bcast no padrão MPI; comunicação de um dado, ou de um grupo distinto de dados, de todos os processos para o processo *root* (comunicação N para P) podendo, ou não, haver processamento destes dados, sendo análogos às funções MPI_Gather (sem processamento dos dados) e MPI_Reduce (com processamento dos dados); e a comunicação de um grupo distinto de dados do processo *root* para todos os processos (comunicação P para N), que é análogo à função MPI_Scatter. Outro tipo de comunicação tratada é a sincronização explícita dos processos, análogo à função MPI_Barrier no padrão MPI.

3. GraphProgramming

GraphProgramming é uma ferramenta *opensource* (protegida pela licença GNU *General Public License* (GPL)) para programação visual para ambientes paralelos e distribuídos. A ferramenta foi desenvolvida na linguagem Java, utilizando o *Java Development Kit* (JDK) 1.6.0_07, escolhida devido aos benefícios para programação gráfica e portabilidade. Além das bibliotecas padrões do JDK, foi utilizada a biblioteca *NetBeans Visual Library* para desenvolvimento da interface gráfica.

A interface de interação com o usuário é composta por quatro partes básicas, conforme ilustrado na Figura 1. A interface possui uma área de menus (A), botões de acesso aos recursos da área gráfica (B), diferentes interfaces de interação (C) e legenda de cores para distinção dos diferentes tipos de comunicação (D).

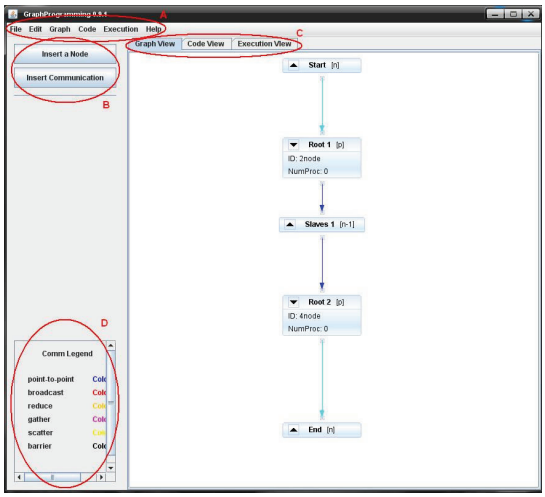

Figura 1. Interface de interação com o usuário.

A ferramenta possui três tipos de interface de interação com o usuário. A interface gráfica (*Graph View*) possui elementos de representação gráfica da estrutura do programa em desenvolvimento pelo usuário. É nesta interface que serão inseridos novos nodos, comunicações e onde existe acesso às funções de inserção de código na estrutura.

A segunda interface é a interface de edição de código (*Code View*), onde o código gerado pela ferramenta será exibido para, caso seja necessário, alterações manuais posteriores. A última interface é a interface de execução (*Execution View*), onde trabalhos futuros poderão inserir recursos de auxílio a execução e monitoramento aos programas gerados.

O projeto gerado pela ferramenta pode ser armazenado em um arquivo para utilização futura. Para o armazenamento do projeto foi criado o formato 'gp1', que segue o padrão *eXtensive Markup Language* (XML), que facilmente se adapta ao formato de grafo na qual o projeto é desenvolvido.

4. Validação

Para realizar a validação dos códigos-fontes gerados pela ferramenta foram utilizados programas com pouca necessidade computacional e diferentes tipos de comunicação entre os processos.

Um dos programas utilizados para teste consiste no envio de uma mensagem do processo *root* para todos os demais processos, que deverão realizar algum processamento com os dados recebidos e retornar o resultado deste processamento ao processo *root* que, por sua vez, deverá realizar um processamento durante o recebimento dos valores e exibir o resultado obtido. A Figura 1 ilustra o mapeamento deste programa para a ferramenta *GraphProgramming*. Esse programa teve seu código-fonte gerado pela ferramenta e foi compilado e executado com sucesso, obtendo os resultados esperados.

5. Comparação com outras ferramentas

Entre outras ferramentas que possuem similaridade de propósitos com a *GraphProgramming*, um comparativo foi realizado com as ferramentas HeNCE [2008], P-GRADE [2008] e ASKALON [2008].

Entre os quesitos utilizados para comparação entre a ferramenta estão [Severino 2008]: disponibilidade para uso, sendo irrestrita na *GraphProgramming* e HeNCE porém restrita nas demais; Plataforma de execução, sendo a *GraphProgramming* multiplataforma (utilizando Java), a HeNCE utilizando a plataforma *X-Window* e as demais não possuindo informações sobre plataforma de execução; O padrão de programação da ferramenta *GraphProgramming* é o MPI, enquanto a HeNCE utiliza o PVM e as demais suportam diversos padrões de programação; por fim, a representação dos elementos, que na *GraphProgramming* utiliza uma representação através de grafos, semelhante a HeNCE e P-GRADE, enquanto a ASKALON representa seus elementos através de uma estrutura *Unified Modeling Language* (UML).

6. Considerações finais e trabalhos futuros

Os resultados positivos na execução dos códigos gerados pela ferramenta, assim como a simplicidade da estrutura resultante dos mapeamentos dos problemas apresentados, indicam que a ferramenta pode ser utilizada para facilitar o desenvolvimento de aplicações mais complexas. Dessa forma, os usuários da *GraphProgramming* podem se estender de alunos de disciplinas de processamento paralelo e distribuído a pesquisadores de outras áreas do conhecimento, que possuem grande demanda computacional e limitações em programação para ambientes que supram esta demanda.

O desenvolvimento da ferramenta foi feito de forma modular, permitindo o desenvolvimento de melhorias nos padrões de programação já desenvolvidos ou a adição de novos padrões de programação. Um aprimoramento já em andamento trata do auxílio à execução do programa gerado para ambientes de *grids* computacionais que utilizem o *middleware Globus Toolkit*, através do auxílio à criação do arquivo *Resource Specification Language* (RSL), que pode ser utilizado para execução de programas MPI.

Referências

MALACARNE, Juliano. (1999) "Ambiente Visual para Programação Distribuída em Java". Dissertação (Mestrado em Ciência da Computação). Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 1999.

HENCE. (2008) "HeNCE: a user's guide", http://www.netlib.org/hence/hence-2.0-doc-html/hence-2.0-doc.html, Novembro.

ASKALON. (2008) "ASKALON", http://www.askalon.org, Novembro.

P-GRADE. (2008) "P-GRADE", http://www.p-grade.hu, Novembro.

SEVERINO, Felipe L. (2008) "GraphProgramming: ferramenta para desenvolvimento de aplicações para ambientes paralelos e de grids computacionais". Trabalho de conclusão de curso (Graduação em Ciência da Computação). Instituto de Ciências Exatas e Geociências, Universidade de Passo Fundo, Passo Fundo, 2008.