Monitoração na Ferramenta VisualProg

Juliano Malacarne, Cláudio Fernando Resin Geyer

{malacarn, geyer}@inf.ufrgs.br

Instituto de Informática - UFRGS
Caixa Postal 15064
91501-970 Porto Alegre - RS

Resumo

VisualProg [SCH96][SCH97] é uma ferramenta de programação de aplicações paralelas e distribuídas que permite ao programador especificar visualmente a sua aplicação. O objetivo deste artigo é apresentar os mecanismos de monitoração usados para visualizar os programas gerados por esta ferramenta. Através da análise das mensagens e utilização dos processadores, a visualização ajuda o programador a melhorar o desempenho da sua aplicação.

Palavras-Chave: Ambiente de Programação, Programação Paralela Distribuída, Programação Visual, Visualização de Programas, Monitoração.

Abstract

VisualProg [SCH96][SCH97] is a parallel programming tool that allows the programmer to visually specify the application. This paper intends to present the monitoring mechanisms implemented to visualize the programs generated by this tool. Through the message statistics and the utilization of the processors, the visualization helps the programmer to improve the performance of the application.

Keywords: Programming Environment, Parallel and Distributed Programming, Visual Programming, Program Visualization, Monitoring.

1 Introdução

VisualProg é um ambiente de programação visual de aplicações paralelas e distribuídas. Seu objetivo é liberar o programador do tratamento de processos e mensagens das aplicações, ao mesmo tempo em que apresenta uma visão geral do sistema que está sendo criado. Isto é importante em se tratando de aplicações paralelas, pois a sua complexidade é bem maior em relação aos programas seqüenciais.

O propósito deste artigo é apresentar os mecanismos de monitoração implementados para visualizar a execução dos programas gerados com esta ferramenta. A monitoração visa principalmente a avaliar o desempenho na busca de soluções para melhorá-lo. Para isto são analisados principalmente a quantidade de mensagens trocadas no sistema e o comportamento dos processos em função do tempo (bloqueado, executando). Todos esses parâmetros são visualizados graficamente para facilitar a análise do programador.

Inicialmente, na seção 2 será dada uma breve descrição da ferramenta VisualProg e do seu modelo de programação. A seção 3 tratará da monitoração e a seção 4 realizará uma comparação com os trabalhos já existentes. Por fim, uma conclusão.

Maiores detalhes sobre a implementação da monitoração estão em [MAL97].

2 Ferramenta VisualProg

Uma aplicação nesta ferramenta é descrita como um grafo dirigido, onde os nodos e os arcos representam, respectivamente, os processos e as mensagens. Durante a edição do grafo, os nodos são anotados com a definição do comportamento que terão na execução. Para isso, possuem regras de ativação que informam o que deverão fazer quando recebem mensagens e

regras de propagação de dados que indicam o instante em que deverão liberar dados. Guardam também as declarações locais de variáveis e o código do usuário. Os arcos definem a estrutura das mensagens, indicando os tipos e os identificadores dos dados correspondentes e também o tipo de interação entre o processo que envia e o que recebe a mensagem.

Durante a execução da aplicação, após serem inicializados, os processos possuem um fluxo de execução constante. Enquanto a aplicação não encerrar, recebem mensagens, analisam as expressões de ativação com execução do código do usuário e propagam dados. Antes de a aplicação ser encerrada, executam um código de finalização.

Com relação ao seu uso, a ferramenta é composta por um editor de grafos e por um gerador de código. O primeiro é utilizado pelo programador para a entrada da estrutura do grafo e da anotação de seus elementos. O gerador de código toma essas informações e cria os arquivos fontes da aplicação. Os arquivos podem ser gerados para rodar nos ambientes HetNOS [BAR94] e PVM [BEG93]. A ferramenta foi construída em C++ nos sistemas operacionais Solaris e Linux e usa as rotinas gráficas da biblioteca de interface XView.

3 Monitoração

Monitoração é a extração de informação a respeito de um processo computacional à medida que ele executa [SNO88]. O objetivo principal na monitoração de programas paralelos é avaliar o desempenho e comportamento geral de tais programas. Analisando parâmetros do sistema é possível encontrar pontos onde a eficiência pode ser incrementada. Observando o comportamento geral do programa, o usuário entende melhor o funcionamento da aplicação, relacionamento entre processos e concorrência de eventos, agilizando a tarefa de depuração.

3.1 Monitoração na ferramenta VisualProg

O processo de monitoração se compõe basicamente de duas fases: coleta de dados e análise dos dados. Na fase de coleta, o sistema de monitoração retira da aplicação em execução as informações de interesse do usuário. Com os dados em mãos, as informações são processadas e apresentadas ao usuário de uma maneira que lhe facilite a análise.

O processamento dos dados pode ocorrer de duas maneiras, classificando a monitoração em *on-line* e *post-mortem*. No primeiro caso, a coleta, o processamento e a exibição dos dados são feitos durante a execução da aplicação. Esta alternativa possui o inconveniente de produzir um alto nível de intromissão na aplicação, podendo distorcer a análise. Para amenizar este problema existe a monitoração *post-mortem*, onde os eventos são armazenados em memória não volátil para processamento posterior à execução da aplicação.

3.2 Exibição dos resultados

A monitoração implementada suporta visualização textual e gráfica. A apresentação textual é importante para análises mais detalhadas, mas prejudica o programador na percepção imediata de relações entre os parâmetros envolvidos na análise. A visualização gráfica, por ter a capacidade de representar mais dimensões, é muito mais adequada à monitoração de programas paralelos. Ela consegue tratar melhor a grande quantidade de dados extraídos de vários processos rodando concorrentemente, produzindo análises mais elaboradas.

A monitoração na ferramenta VisualProg suporta os seguintes tipos de análises gráficas:

- **Diagrama de Tempos**: exibe os eventos de trocas de mensagens, criação e destruição de processos, ordenados segundo a relação de ordem causal;
- **Diagrama de Utilização de Gantt**: mostra o comportamento dos processos em função do tempo (executando, bloqueado esperando mensagem, encerrado);
- **Diagrama de Kiviat**: compara um determinado parâmetro (número de mensagens enviadas, por exemplo), entre todas máquinas onde executam processos da aplicação;

- **Diagrama de Passagem de Mensagens**: exibe as mensagens enviadas e ainda não recebidas num determinado momento;
- **Diagrama de Informações sobre as Máquinas**: informa dados sobre as máquinas, como o número de mensagens e *bytes* trocados e a utilização do processador;
- **Diagrama de Estatísticas de Mensagens**: mostra o número de mensagens e *bytes* enviados e recebidos por cada um dos processos;
- **Diagrama de Mapeamento de Processos**: exibe as trocas de mensagens ocorridas na aplicação, organizando os processos por máquinas;
- **Diagrama de Detalhes de Eventos**: exibe o último evento gerado em cada processo.

Além desses diagramas, a visualização dos estados dos processos e trocas de mensagens é feita no próprio editor de grafos da ferramenta. Há ainda o Diagrama de Fluxo de Execução que permite visualizar a estrutura do programa e é utilizado principalmente para depuração.

A exibição dos resultados pode também ser feita textualmente, através de relatórios. Eles apresentam os eventos gerados pelos processos, ordenados pelos *timestamps* lógicos. Para cada evento é exibido qual processo o originou, além de seus parâmetros. Após a seção de eventos, segue uma estatística geral sobre os processos e as máquinas.

3.3 Implementação do monitor

A coleta de dados é feita pela instrumentação do código fonte. Como a ferramenta gera o código das aplicações, ela tem a capacidade de incluir o código adicional para as funções de monitoração. Na monitoração *on-line*, os dados coletados são enviados através de mensagens para o módulo que irá processar os dados e na *post-mortem* eles são gravados em um arquivo de *trace*. Para diminuir o nível de interferência na aplicação, é permitida a seleção de eventos, onde somente uma determinada classe de eventos pode ser tratada (por exemplo, somente as trocas de mensagens).

Para solucionar o problema da ordenação de eventos, o método implementado é semelhante ao usado na ferramenta PVaniM [STA96]. Cada evento recebe um *timestamp* lógico e um físico. O *timestamp* lógico carrega a informação a respeito da ordenação causal dos eventos e é tratado quando o monitor recebe as mensagens e lê o arquivo de *trace*. O outro é empregado nas medidas de tempo.

4 Trabalhos Relacionados

Esta ferramenta se situa principalmente no conjunto daquelas que monitoram programas distribuídos com comunicação por mensagens. Assim como a ferramenta VisualProg, as ferramentas de seu grupo suportam em geral programas feitos para o ambiente PVM. Com relação às ferramentas de programação visual, é difícil se encontrar suporte à visualização dos programas. Dentre as mais conhecidas (HeNCE [BEG93] e CODE [NEW93]), apenas a ferramenta HeNCE apresenta algum suporte à monitoração. Ainda assim, esse suporte não é completo. É possível realizar monitoração tanto durante quanto após a execução, mas o nível de interferência é o mesmo, pois o arquivo de *trace* é criado durante a monitoração *on-line*. Além disso, a análise leva em conta apenas a utilização dos processadores e o estado dos processos através do grafo do programa.

Existem ferramentas mais especializadas para visualizar programas PVM. Dentre as mais simples, o XPVM [GEI96] visualiza os programas tanto *on-line* como *post-mortem*. Entretanto, apresenta o mesmo problema do HeNCE, gerando o arquivo de *trace* durante a monitoração *on-line*. A ferramenta PVaniM 2.0 [STA96] apresenta o Diagrama de Gantt, o Diagrama de Causalidade (semelhante ao Diagrama de Tempos) e o Diagrama de Passagem de Mensagens. Entretanto, por enquanto há o problema de não haver separação entre as máquinas e de todos os processos serem nomeados apenas como números inteiros, o que pode

dificultar a sua identificação por parte do usuário. Na monitoração do VisualProg, os processos também são representados por número inteiros, mas esta representação é válida pois é do mesmo tipo de representação utilizada na programação da aplicação.

A principal vantagem desta ferramenta em relação às demais é a integração que ela proporciona entre programação e monitoração. Nas duas situações o usuário visualiza o mesmo tipo de representação e de estruturas. Além disso, estas estruturas estão mais próximas da aplicação, permitindo, por exemplo, que se analise qual procedimento está sendo executado e não somente que algum procedimento está sendo executado.

5 Conclusões

Este trabalho apresentou a implementação de um sistema de monitoração para a ferramenta VisualProg. Foram descritos os mecanismos empregados para obter os dados da execução e os tipos de análise oferecidos. A principal característica deste sistema é que é possível realizar tanto monitoração *on-line* como *post-mortem* de forma integrada com a programação. É importante se ter durante a visualização dos programas uma análise mais abstrata da aplicação. Com a integração dessas partes, o sistema tem condições de informar o comportamento de estruturas de mais alto nível, o que não é possível em ferramentas de visualização que trabalham sobre códigos fontes gerais.

Bibliografia

[BAR94] BARCELLOS, A. M. P. et al. **Um ambiente para Programação de Aplicações Distribuídas em Redes de Workstations**. In: Congresso Nacional de Redes de Computadores, 12., 1994, Curitiba. **Anais...** Rio de Janeiro: SBC, 1994.

[BEG91] BEGUELIN, A. et al. **A User's Guide to PVM Parallel Virtual Machine**. Oak Ridge National Laboratory, July 1991. (Technical Report ORNL/TM-11826).

[BEG93] BEGUELIN, A. et al. **HeNCE: A Heterogeneous Network Computing Environment**. University of Tennessee, 1993. (Technical Report CS-93-205).

[GEI96] GEIST, G. A.; KOHL, J.; PAPADOPOULOS, P. **Visualization, Debugging and Performance in PVM**. Oak Ridge National Laboratory, 11 pp., Oak Ridge, Technical Report disponível por FTP em ftp://netlib2.cs.utk.edu/pvm3/xpvm (2 Dez. 1997).

[MAL97] MALACARNE, J. **Implementação de um Ambiente Gráfico para o Desenvolvimento de Aplicações Distribuídas**. Trabalho de Diplomação, Instituto de Informática, UFRGS, pp. 91, Dez. 1997.

[NEW93] NEWTON, P. **A Graphical Retargetable Parallel Programming Environment and Its Eficient Implementation**. University of Texas at Austin, 1993. (Technical Report TR93-28).

[SCH96] SCHRAMM, J. F. L. **Ambiente Gráfico para o Desenvolvimento de Aplicações Distribuídas**. CPGCC-UFRGS, pp. 78, 1996.

[SCH97] SCHRAMM, J. F. L.; MALACARNE, J.; GEYER, C. F. R. **Ambiente Gráfico para o Desenvolvimento de Aplicações Distribuídas**. XXIII Conferência Latinoamericana de Informática. Universidad Tecnica Federico Santa Maria, Valparaiso, Chile, p.505-514. Nov. 1997.

[SNO88] SNODGRASS, R. **A Relational Approach to Monitoring Complex Systems** ACM Transactions on Computer Systems, vol.6, n. 2, p. 157-196, Mai. 1988.

[STA96] STASKO, J. **PVaniM 2.0-Online and Postmortem Visualization Support for PVM**. Disponível por WWW em http://www.gatech.edu/gvu/softivz/parviz/pvanim (2 Dez. 1997).