Depuração de programas paralelos

Mairo Pedrini[1], Philippe Olivier Alexandre Navaux

Universidade Federal do Rio Grande do Sul - Instituto de Informática
Avenida Bento Gonçalves, 9500 – Bloco IV – Campus do Vale
Caixa Postal 15064 – 91501-970
Porto Alegre – RS – Brasil
{mpedrini, navaux}@inf.ufrgs.br

Introdução

Desde o início da era da computação, a necessidade de pesquisar e corrigir os erros na lógica de um programa tem ocupado grande parte do tempo de desenvolvimento do mesmo. Para programas simples, revisar o código-fonte ou imprimir o valor das variáveis em pontos estratégicos eram formas aceitáveis de depuração, mas, para programas mais complexos, tais métodos se tornavam complicados e, geralmente, pouco eficazes.

Ferramentas de depuração que permitiam a execução do programa passo a passo se tornaram comuns, tanto por permitir que o fluxo de execução do programa fosse visualizado quanto por permitir a visualização do valor de variáveis em qualquer ponto. Tais ferramentas simplificaram grandemente a depuração de programas seqüenciais.

Mas estes métodos, especialmente a execução passo a passo, dependiam muito do determinismo dos programas seqüenciais ou seja, o programa poderia ser repetido quantas vezes fossem necessárias até que o erro fosse encontrado, e se tornaram insuficientes para lidar com problemas de sincronização ou *deadlocks* em programas paralelos. Os programas paralelos dependem muito do ambiente onde são executados: a carga do processador, a política de escalonamento de tarefas, etc.

O advento da programação distribuída aumentou ainda mais as dificuldades, introduzindo fatores relacionados à heterogeneidade das máquinas e às características da comunicação.

Novas características eram desejadas nos depuradores, tais como mecanismos de seleção e visualização do estado dos processos, controle sobre o fluxo de múltiplos programas ao mesmo tempo, etc. Para tanto, algumas ferramentas foram criadas, tais como TotalView, Annai, p2d2, etc.

O objetivo deste trabalho é apresentar uma destas ferramentas, a PADI [STR 2002], e suas vantagens e desvantagens quando usada para depurar programas em ambientes de programação paralela e distribuída.

---

1 Bolsista do Projeto LabTeC – DELL – UFRGS (Laboratório de Tecnologia em Clusters)

Características e arquitetura interna

A PADI como interface de depuração

O objetivo principal da PADI, como descrito em [STR 2002], era muito mais de criar uma interface amigável e intuitiva para o usuário do que de criar um depurador que permitisse encontrar e corrigir todas as possíveis formas de erro dos programas. Antes de tudo, a PADI é uma interface para um depurador paralelo, oferecendo as características de seleção e visualização enquanto o depurador oferece a execução passo a passo, visualização de variáveis e controle individual do fluxo dos programas.

Como depurador, a PADI faz uso do Fiddle [CUN 98], que pode ser visto como um conjunto de servidores e bibliotecas especiais para depuração. O Fiddle é capaz de centralizar o controle de vários depuradores seqüenciais (atualmente, o depurador seqüencial usado é o GDB [STA 94]), centralizando a interface de diversos depuradores distribuídos pela rede.

O Fiddle foi desenvolvido para depuração on-line, ou seja, para executar e controlar o fluxo de execução do programa. Não é possível depurar programas de forma off-line, analisando e visualizado uma única execução com defeitos, contornando o não-determinismo de tais aplicações.

Características principais

Como citado anteriormente, a PADI é uma interface gráfica para uma ferramenta de depuração on-line. Sua interface se divide em dois níveis: um nível de coordenação e outro de processos. O nível de coordenação é responsável pela seleção e visualização dos processos como um todo, bem como o agrupamento dos mesmos. O nível de processos é semelhante a um depurador seqüencial para aquele processo.

O mecanismo de seleção de processos é um de seus pontos fortes, permitindo que o usuário escolha entre grupos de processos pré-definidos, ou crie seus próprios grupos. Tal mecanismo de seleção atua, ainda, como um filtro para os processos, ou seja, é possível ocultar os processos que não são importantes num dado momento. Em [STR 2002], a autora faz uma comparação entre as características dos principais depuradores paralelos, e apresenta um conjunto de características desejáveis, procurando implementa-las, em seguida, na PADI.

Depuração de programas paralelos

Tendo as características básicas para depuração funcionando, o enfoque do trabalho se tornou testar e, onde fosse possível, simplificar o uso da PADI na depuração de programas paralelos e distribuídos. Em um primeiro momento foram feitos testes com programas seqüenciais e, em seguida, programas utilizando *threads* (especificamente, programas utilizando a biblioteca *pthread*).

O passo seguinte foi testar a depuração de programas nos ambientes PVM [GEI 94] e MPI [MPI 94], iniciando os testes com programas distribuídos.

A depuração de programas seqüenciais e baseados transcorreu sem problemas, garantindo o funcionamento básico da ferramenta. A depuração de programas baseados

em *threads* mostrou que a ferramenta possui recursos muito rudimentares para tal modelo.

Foi na depuração de programas no ambiente PVM que as primeiras falhas começaram a aparecer, algumas causadas pelo modelo de programação paralela, alguns causados por pequenos problemas com a ferramenta e, ainda, uma parte causada por defeitos com o Fiddle, cuja versão utilizada era antiga e contendo alguns defeitos.

Os principais problemas foram dois, um causado pelo modelo: a carga de novos processos, e o outro causado pelo Fiddle, que parava de responder caso fosse enviado um comando para um processo bloqueado.

O problema da carga de novos processos consiste no seguinte: o disparo de novos processos não é interceptado pelo depurador, ou seja, assim que o novo processo é carregado, ele já é executado. Para depurá-lo, é preciso que este bloqueie a espera de uma mensagem ou entre em um laço infinito, depois se usa o comando *Attach* para carregá-lo no depurador. Se a forma escolhida para fazer o programa parar foi um laço, pode se usar o comando *Set Variable* para sair do laço, permitindo que o programa prossiga, sob controle do depurador.

O segundo problema era um problema do Fiddle, e consistia no seguinte: se fosse enviado um comando de *step*, *next* ou *continue* para um processo bloqueado, este comando não seria executado diretamente, mas o processo (na PADI) entraria no estado *debugging* (este estado, representado por uma elipse cinza, indica que um comando está sendo executado naquele processo e que, portanto, este não está aceitando novos comandos). O esperado é que este processo ficasse em tal estado até que fosse desbloqueado, quando então o comando seria executado e o Fiddle retornaria um resultado a PADI. Mas não é o que acontece: o servidor fica bloqueado, não respondendo a nenhum outro comando enviado, mesmo que para processos que não estivessem bloqueados.

Na depuração de programas MPI, o problema surge já na hora de carregá-los no depurador: por serem executados a partir de um script, não é possível usar o comando *Load* diretamente. Ao invés disto, o método usado é o mesmo que para processos filhos no PVM: um laço no início do programa, o comando *Attach* e, em seguida, o comando *Set Variable*. Com o programa carregado, surge novamente o problema dos processos bloqueados no envio de mensagens.

Conclusão

No atual estágio do trabalho, muito pouco pode ser dado como certo, visto que foram efetuados apenas alguns testes. Os objetivos principais desta primeira fase eram, além de explorar e conhecer os pontos fortes e fracos da ferramenta, conhecer sua estrutura interna e ganhar familiaridade com seu código-fonte.

Pudemos constatar a existência de diversos problemas e fraquezas, sendo que alguns podem até mesmo impedir o trabalho de depurar certos programas.

Continuação do trabalho

Dado o exposto acima, o prosseguimento do trabalho na PADI está se concentrando em tentar corrigir alguns destes problemas. Inicialmente atualizando a PADI para usar a versão mais recente do Fiddle. Espera-se, com isto, eliminar o problema de o servidor bloquear onde não deveria.

Em [STR 2002], a autora propõe também o uso do Fiddle_J, uma versão Java das bibliotecas do Fiddle. Tal modificação simplificaria grandemente a manutenção do código-fonte da PADI, eliminando a parte de comunicação com o servidor Fiddle e, assim, deixando o código-fonte mais claro e objetivo.

Feita esta atualização, que pode ser considerada como a principal no momento, os testes de depuração continuam com PVM, MPI e DECK [BAR 2000] e, conforme o andamento destes, com outras bibliotecas.

Referências

[BAR 2000] BARRETO, M. **DECK**: Um ambiente para programação paralela em agregados de multiprocessadores. 2000. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[CUN 98] CUNHA, J. C.; LOURENÇO, J. et al . A Framework to Support Parallel and Distributed Debugging. In: INTERNATIONAL CONFERENCE AND EXHIBITION ON HIGH-PERFORMACE COMPUTING AND NETWORKING, HPCN EUROPE, 1998, Amsterdam. **High-performance computing and networking:** proceedings. Berlin: Springer-Verlag, 1998. p.708-717. (Lecture Notes on Computer Science, v. 1401).

[GEI 94] GEIST, Al et al. **PVM:** Parallel and Virtual Machine – A User's Guide and Tutorial for Networked Parallel Computing. London: MIT, 1994.

[MPI 94] MPI FORUM. **The MPI message passing interface standard.** Knoxville: University of Tennessee, 1994.

[STA 94] STALLMAN, R. **Debugging with GDB**. Boston: Free Software Foundation, 1994.

[STR 2002] STRINGHINI, D.; NAVAUX, P. **Depuração de Programas Paralelos - Projeto de uma interface intuitiva.** 2002. Tese (Doutorado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.