HProfVis: um visualizador para o HProf

Mauricio C. Moraes, Cláudio F. R. Geyer

Instituto de Informática – Universidade Federal do Rio Grande do Sul
mmoraes@myway.com.br, geyer@inf.ufrgs.br

Introdução

A análise de desempenho considerada neste trabalho diz respeito ao comportamento dos métodos e ao uso da memória RAM. O primeiro pode ser estudado pela maneira como a CPU é utilizada pelos métodos durante a execução da aplicação. O segundo pode ser estudado pela alocação de objetos em memória RAM durante a execução do aplicativo.

O conceito adotado aqui é o mesmo de [ADV 02]: análise de desempenho significa observar a execução de um programa para saber onde gargalos ou outros problemas como desperdícios de memória podem ocorrer. Uma vez que se descobre onde esses potenciais problemas estão localizados, pode-se modificar o código da aplicação para remover ou reduzir seu impacto.

Este trabalho está inserido no contexto do projeto DiretoGNU, um projeto de um software de agenda, catálogo e correio eletrônicos desenvolvido no espírito do software livre, sobre a plataforma Java. O projeto direto está sendo totalmente desenvolvido no Brasil, pela PROCERGS em parceira com a UFRGS. Dentro desse projeto, o HProfVis tem a função de permitir a análise de desempenho gratuita de código Java.

Este texto descreve inicialmente a motivação deste trabalho (seção 2), seguida da modelagem da ferramenta (seção 3). Após a modelagem da ferramenta, vêm alguns detalhes sobre o seu protótipo (seção 4). O texto é finalizado com as considerações finais (seção 5).

Motivação

Freqüentemente, os desenvolvedores de software deparam-se com problemas de baixa velocidade de execução em seus programas, com o excesso de consumo de recursos por estes e com o desconhecimento de seu comportamento. Outras vezes, os desenvolvedores gostariam de saber qual é realmente o comportamento dos seus programas para saber se eles estão de acordo com o esperado ou se existem potenciais erros que podem fazer parte do código, mas que não são detectados por não se manifestarem em uma determinada bateria de testes a que foram submetidos. A busca por essas informações pode ser difícil e cansativa se não for auxiliada por uma ferramenta especializada.

Existem, hoje em dia, ferramentas de análise de desempenho (*profilers*) de código Java comerciais e gratuitas. As ferramentas comerciais são, via de regra, superiores às gratuitas, pois abrangem um conjunto mais amplo de informações monitoradas.

Entre as ferramentas de análise de desempenho de código Java gratuitas, existe uma, cujo nome é HProf, que se destaca pelo conteúdo de suas informações resultantes e pela sua distribuição universal, pois é parte integrante do JDK da SUN Microsystems. Esta

ferramenta, no entanto, possui como saída um arquivo texto de difícil compreensão, pois seus dados não ficam organizados convenientemente para a análise tanto pela sua distribuição no texto quanto pela natureza estática do arquivo texto.

A maior parte das outras ferramentas gratuitas é, na verdade, baseada no HProf, pois trabalham a partir da saída desta. Elas representam, assim, uma melhora de uso do HProf. Essa melhora, entretanto, não é em todos os aspectos, pois o que elas fazem é processar informações relativas a um subconjunto do total de informações que o HProf monitora. Dessa forma, não foi encontrada uma ferramenta gratuita que processe todas as informações que o HProf monitora e que as disponibilize para os usuários de uma forma gráfica, bem organizada e interativa.

Preencher esse espaço é a primeira motivação deste trabalho, que pode ser resumida da seguinte forma: tornar mais fácil, rápido e eficiente o uso de todas as funcionalidades da ferramenta de análise de desempenho gratuita HProf. A visualização com o HProfVis, entretanto, pode ser estendida a outros *profilers*, não estando limitada unicamente à visualização dos dados do HProf.

Modelo

A estrutura de monitoração pelo HProf e visualização do código Java pelo HProfVis pode ser vista na figura 1. A definição dos eventos de interesse pode ser feita na interface gráfica do HProfVis. A detecção e o registro dos eventos de interesse é responsabilidade do HProf através da JVMPI.

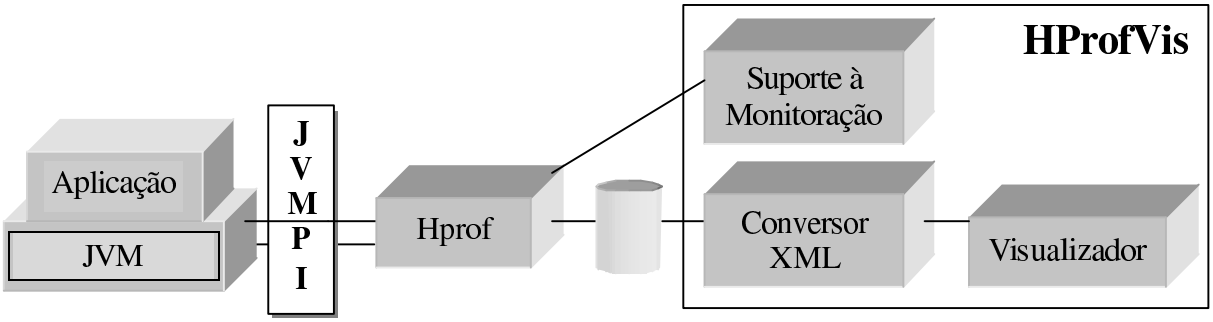

Figura 1: arquitetura de monitoração e visualização

A detecção dos eventos é realizada através de um agente de monitoração construído usando a JVMPI [SUN 02].

No HProfVis, a análise é feita por dois módulos: **o conversor XML** [UND 02] e o **visualizador**. O **conversor XML** transforma a saída do HProf, escrita em disco, em um arquivo intermediário em formato XML para que o módulo **visualizador**, a partir deste arquivo, mostre seus gráficos e tabelas. A existência desse formato intermediário é que torna o HProfVis extensível a outros *profilers*.

O módulo **visualizador** permite a visualização de múltiplas monitorações concorrentemente.

O módulo de **suporte à monitoração** permite a execução do HProf a partir do ambiente do HProfVis, com possibilidade de definição dos diversos parâmetros necessários para a monitoração: indicação da classe principal do aplicativo Java a ser monitorado,

indicação dos arquivos de armazenamento de resultados e tipos de monitoração (uso de memória RAM, uso de CPU e pilhas de traços de chamadas de métodos).

No HProfVis, os dados da saída do HProf podem ser rapidamente acessados através de uma interface gráfica subdividida em painéis. Existe um painel para cada grupo de informações. Os grupos de informações disponíveis são: **GENERAL, THREADS, HEAP SITES, OBJECT COUNT, CPU SAMPLES, CPU TIMES** e **TRACES.**

a) painel **GENERAL:** contém as informações gerais do arquivo visualizado (nome, caminho absoluto no sistema de arquivos, tamanho em bytes, nome da classe principal da aplicação monitorada);

b) painel **THREADS:** contém as informações sobre as *threads* da aplicação (ordem de criação, confirmação de término, nome e grupo);

c) painel **HEAP SITES:** contém as informações sobre o uso da memória RAM pelos objetos do aplicativo (porcentagem da quantidade de RAM utilizada por cada objeto, quantidade de bytes utilizada pelo objeto no momento da monitoração e quantidade de bytes utilizada pelo objeto até o momento da monitoração, desde o início da execução).

d) painel **OBJECT COUNT:** também contém informações sobre o uso da memória RAM pelos objetos do aplicativo mas segundo uma óptica diferente. Neste painel, os objetos estão agrupados por suas respectivas classes, oferecendo uma contagem do número de objetos por cada classe.

e) painel **CPU SAMPLES:** contém as informações sobre o uso da CPU pelos métodos do aplicativo segundo uma amostragem de uso de CPU feita durante a execução (porcentagem do número de vezes em que determinado método foi encontrado utilizando a CPU e número de vezes em que determinado método foi encontrado utilizando a CPU).

f) painel **CPU TIMES:** painel similar ao anterior que também contém informações sobre o uso da CPU pelos métodos mas segundo o tempo de uso de CPU por cada método.

g) painel **TRACES:** permite a busca de todas as pilhas de chamadas de métodos que contenham um determinado método, fornecido pelo usuário.

Protótipo

Atualmente este trabalho encontra-se desenvolvido e testado na plataforma JDK 1.3 para o *profiler* HProf. A ferramenta possui as funcionalidades descritas até aqui e o seu protótipo alcançou os objetivos esperados. A figura 2 mostra uma fotografia de exemplo do painel OBJECT COUNT.

O próximo passo deverá ser a integração com o DOMonitor, um ambiente de monitoração de aplicações distribuídas Java [ARA 02] que, assim como o HProf, disponibiliza o resultado da sua monitoração em um arquivo texto de difícil uso direto.

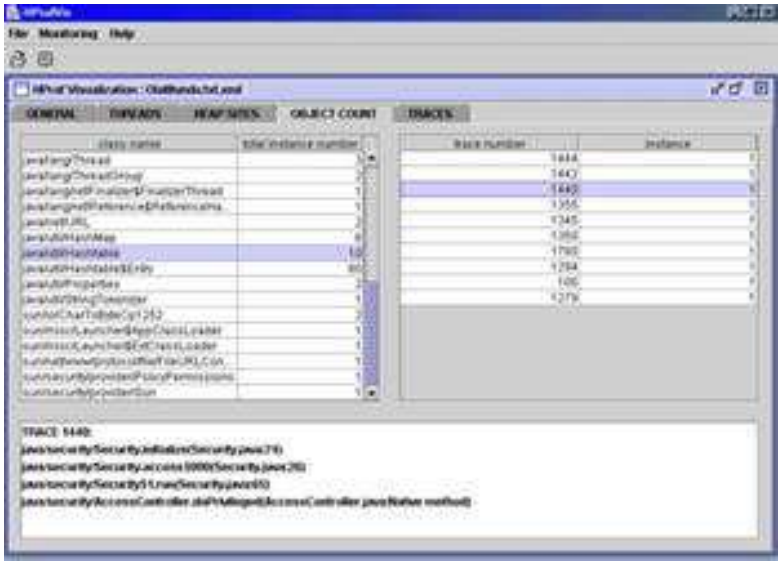

Figura 2: painel OBJECT COUNT do HProfVis

Conclusão

Através do uso do HProfVis, as informações estáticas e pouco acessíveis do arquivo textual de saída do HProf e de potencialmente outros *profilers* são reorganizadas e tornadas acessíveis ao usuário através do clique do *mouse*, pela sua disposição em painéis e tabelas gráficos.

O HProfVis também oferece uma suporte gráfico à monitoração de aplicativos, que dispensa o uso da linha de comando do HProf. Nesse suporte, uma interface gráfica *es* permite a seleção das opções de monitoramento e o controle de execução da aplicação de forma mais fácil para o usuário.

Fica claro, portanto, como havia sido proposto inicialmente, que o HProf-Visual torna mais fácil, rápido e eficiente o uso das funcionalidades da ferramenta de análise de desempenho gratuita HProf.

Referências

[SUN 02] SUN MICROSYSTEMS. **Documentação da linguagem Java**. Disponível em http:// java.sun.com/products/jdk/1.3/download-docs.html. Último acesso em fevereiro de 2002.

[ADV 02] SUN MICROSYSTEMS. **Advanced Programming for the Java 2 Plataform. Chapter 8 Continued: Performance Analysis**. Disponível em http://developer.java.sun.com/developer/onlineTraining/Programming/JDCBook/perf3.html

[ARA 02] ARAÚJO, E. B. **DOMonitor: Um Ambiente de Monitoração de Aplicações Distribuídas Java**. Porto Alegre: CPGCC-UFRGS, 2002.

[UND 02] SUN MICROSYSTEMS. **Understanding XML**. Disponível em http://java.sun.com/webservices/docs/ea1/tutorial/doc/IntroXML.html. Último acesso em novembro de 2002.