JRastro: Ferramenta de rastreamento de programas Java usando JVMTI*

Geovani Ricardo Wiedenhoft, Benhur Stein

Universidade Federal de Santa Maria - Laboratório de Sistemas de Computação
Faixa de Camobi, Km 9 -CEP 97105-900-Santa Maria-RS-Fone/Fax:(55) 220 8523
{grw,benhur}@inf.ufsm.br

Introdução

A depuração de programas paralelos e distribuídos não é uma tarefa simples. Entretanto, essa tarefa pode ser auxiliada quando se dispõe de ferramentas que permitem a visualização do comportamento do programa. Tais ferramentas necessitam que seja realizado o rastreamento dos principais eventos ocorridos durante a execução do programa, chamados rastros de execução. Para fazer esse rastreamento dos programas implementados na linguagem de programação Java está sendo desenvolvido a ferramenta JRastro no Laboratório de Sistemas da Computação (LSC). Este artigo descreve o progresso realizado nessa ferramenta desde a ERAD 2005 [WIE 2005]. Nas seções seguintes descreve-se o rastreamento de programas Java, bem como a ferramenta desenvolvida. Após isso, são apresentados os eventos para rastreamento, os conversores implementados que realizam o pré-processamento dos rastros para visualização e por fim os trabalhos futuros.

Rastreamento de programas Java

A depuração de programas a partir da visualização ocorre com o rastreamento dos principais eventos gerados durante as execuções desses programas. Em Java esse rastreamento pode ser realizado com o monitoramento e registro dos eventos gerados pela Máquina Virtual Java (JVM). Para isso, a linguagem oferece a JVMTI (*Java Virtual Machine Tool Interface*) [MIC 2004], que é uma interface de recuperação de eventos, disponibilizada desde a versão 1.5 do ambiente de desenvolvimento Java *JDK*.

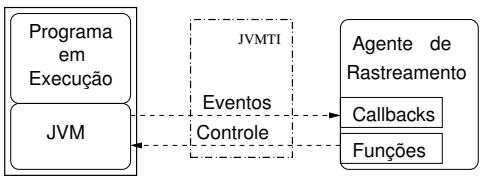

Figura 1: Arquitetura da JVMTI

Conforme a figura 1, a JVMTI intercepta eventos gerados pela JVM durante a execução do programa, e notifica um cliente JVMTI, chamado agente de rastreamento,

*Trabalho parcialmente financiado pelo CNPq

que o evento ocorreu. Isso é realizado através de funções (*callbacks*) pré-estabelecidas. Além disso, JVMTI oferece funções para o agente obter informações da JVM e para fazer o controle do monitoramento, como por exemplo selecionar os eventos a serem monitorados e habilitar as funções que irão tratar esses eventos.

Além de proporcionar a notificação de eventos através de funções, a JVMTI também oferece a possibilidade de se alterar o código objeto da classe do programa sendo monitorado. Isso é implementado através da instrumentação do *bytecode* (BCI - *byte code instrumentation*) antes da carga da classe.

Eventos rastreados

Os eventos que se podem rastrear pela JVMTI são muito numerosos. Desse modo, foi desenvolvido um agente configurável, que permite o usuário selecionar quais eventos, classes e métodos são necessários para visualização, através de arquivos de configurações. As seções seguintes descrevem os eventos registráveis por JRastro.

Rastreamento de criação / destruição de *threads*

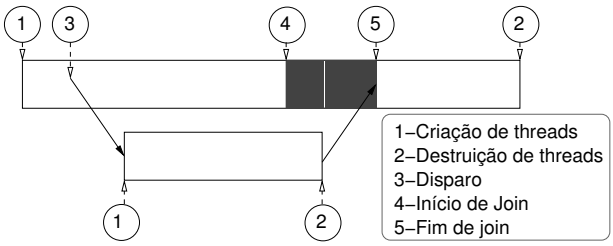

Figura 2: Criação / destruição de *threads*

A figura 2 mostra a visualização dos eventos de criação/destruição, disparo e *join* de *threads*. Além disso, apresenta uma seta que identifica qual *thread* criou e qual foi criada. Essa visualização auxilia a depuração do código pelo usuário, pois mostra a ordem de execução das *threads* no programa.

Essa visualização foi implementada através do uso de eventos de notificação de criação e destruição das *threads* oferecida pela JVMTI. No evento de criação a biblioteca de registro é inicializada, e é realizado o registro de algumas informações, como o identificador e o nome da *thread* criada. No evento de término, os registros correspondentes a *thread* são registrados em disco.

Rastreamento de chamadas / retornos de métodos

A figura 3 ilustra a chamada e retorno de métodos de uma determinada *thread*. Essa visualização permite o usuário saber o que uma *thread* está executando em um período, verificar concorrências e também identificar erros ou mesmo pontos para melhora de desempenho.

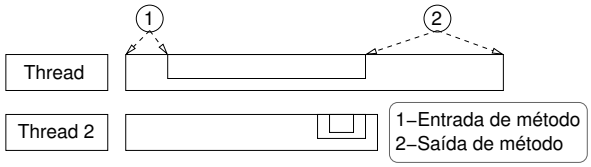

Figura 3: Chamadas / retornos de métodos

Primeiramente foi implementado com a utilização de funções de notificação pela JVMTI. Nessa implementação a JVMTI intercepta a chamada e retorno de métodos e notifica o agente, passando como parâmetros os identificadores da *thread* e do método que está sendo executado. Entretanto, foi constatado o baixo desempenho da biblioteca e a falta de informação, como o identificador do objeto.

A partir disso, se optou por implementar uma segunda versão através da inserção de *bytecodes*, BCI. Com isso, não é mais necessária a JVMTI monitorar os eventos de chamada e retorno de métodos, mas somente o momento que antecede a carga da classe para serem adicionados os *bytecodes* necessários. Porém, a JVMTI não oferece suporte direto a essa inserção, mas apenas possui em sua distribuição um código exemplo de alterações de classes, que foi modificado para suprir as necessidades do agente de rastreamento.

Através de BCI, cada método alterado é inserido código no início que realiza a chamada de um função que faz o tratamento do evento e o registro dos identificadores da *thread*, do método chamado e caso exista, do objeto que executa o método. Além dessa inserção, também é inserido código no fim do método, que faz o registro apenas do identificador da *thread*. Nesse caso não sendo necessários outros identificadores, pois são obtidos pela verificação de qual o último método que a *thread* chamou. Quando ocorrem exceções, essa forma de registro não é suficiente, porque os métodos têm sua execução interrompida, não executando o código de registro inserido em seu final. Para se registrar essa informação, os eventos que informam sobre exceções na JVM serão interceptados.

Rastreamento de monitores

A visualização de monitores e as *threads* que estão bloqueadas ajuda na busca de possíveis impasses, excesso/falta de sincronização, entre outras. Para esses elementos serão utilizados funções de notificação pela JVMTI de *threads* bloqueadas, ou seja, tentam entrar em uma região sincronizada e não conseguem. Além desse, também a notificação de *threads* que após ficarem bloqueadas conseguem entrar na região bloqueante. Nesses eventos são registrados os identificadores da *thread* e o monitor em questão.

Rastreamento de alocação e liberação de memória

A visualização da alocação e liberação de memória é interessante, pois mostra o estado de memória utilizado pelo programa em um determinado tempo e a execução do coletor de lixo. Para esses eventos pretende-se estudar formas de utilizar funções de notificação de eventos pela JVMTI, como a notificação de execução do coletor de lixo, com a união do método de inserção de *bytecodes*.

Rastreamento de comunicações RMI

Os eventos RMI mostram as comunicações entre os métodos de diferentes *threads*, através de setas no visualizador identificando qual método chamou e qual foi chamado. Para esses eventos, a JVMTI não possui funções de notificação diretas. Uma alternativa para a monitoração desses eventos é a identificação de métodos que são chamados para a comunicação e também a identificação de qual *thread* executará o método remoto.

Conversores

Após a geração dos rastros de execução pelo agente de rastreamento é necessária a conversão/pré-processamento desses rastros para o formato do visualizador Pajé, ferramenta escolhida para a visualização. Para isso, se pode fazer conversões diferentes para os mesmos rastros com objetivos de diferentes visualizações. Neste trabalho foram implementados dois conversores.

O primeiro conversor possibilita a visualização por fluxos de execução. Nessa visualização são mostrados os eventos associados ao fluxo de execução que os executou. O segundo conversor permite a visualização por objeto, mostrando quais métodos um objeto está executando em um determinado tempo, possibilitando a verificação do nível de execução concorrente no interior de cada objeto.

Conclusão

Este texto descreve o desenvolvimento e progresso da ferramenta JRastro, que tem como objetivo o rastreamento de programas Java para posterior visualização. Esse rastreamento é realizado com o uso da JVMTI e BCI para os principais eventos ocorridos na execução. Com isso, a JRastro permite o usuário fazer a seleção dos eventos, classes e métodos que deseja visualizar sem a necessidade de alteração do código do programa.

Como trabalhos futuros pretende-se identificar os métodos que fazem a criação de novas *threads* e também os métodos que fazem *join*. Além disso, deseja-se concluir os eventos de exceções e de monitores através da notificação de eventos pela JVMTI e reconhecimento de alguns métodos. Para os eventos de alocação/liberação de memória é necessário estudar formas para a união da interceptação dos eventos pela JVMTI e a alteração de determinados métodos. Finalizando, deseja-se pesquisar soluções para o rastreamento dos eventos RMI, como a identificação dos métodos que fazem a comunicação, porque a JVMTI não disponibiliza funções diretas para o reconhecimento desses eventos.

Referências

[MIC 2004] MICROSYSTEMS, S. **Java Virtual Machine Tool Interface**. http://java.sun.com/j2se/1.5.0/docs/guide/jvmti/jvmti.html.

[WIE 2005] WIEDENHOFT, G. R.; STEIN, B. Rastreamento e visualização de programas Java usando JVMTI. **Anais da Quinta Escola Regional de Alto Desempenho, Canoas, RS, Brasil**, 2005.