Manual de instalação e utilização da *JRastro*

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

A instalação e utilização da biblioteca *JRastro* é fácil e simples. Na seção seguinte apresenta-se o requisito principal à utilização da *JRastro*. Após, descrevem-se a instalação (seção 2) e a utilização (seção 3) dessa biblioteca. Finalizando este manual, a seção 4 contém os eventos e as funções disponibilizados pela JVMTI.

1 Requisito

O requisito principal para a utilização do agente é que esteja instalado no sistema a versão 1.5 ou superior da JDK. Pois, versões anteriores não possuem o módulo JVMTI, não sendo compatíveis com a biblioteca *JRastro*.

2 Instalação

A biblioteca *JRastro* pode ser encontrada em `http://www.inf.ufsm.br/lsc/libJRastro`. Após o *download* da *JRastro* é necessário alterar as variáveis `JDK_HOME`, `JRASTRO_DIR` e `JRASTRO_DIR_BIN` do arquivo *Makefile*, descritas a seguir:

- A variável `JDK_HOME` deve ser alterada para conter o caminho correto da instalação da JDK.
- A variável `JRASTRO_DIR` contém o local que serão instalados os arquivos auxiliares da biblioteca.
- A variável `JRASTRO_DIR_BIN` indica o local onde serão instalados os binários e programas auxiliares. O valor dessa variável deve estar ou ser colocado na variável de ambiente *PATH*.

Com as alterações no arquivo *Makefile*, os processos de compilação, instalação e desinstalação estão prontos para serem executados. Esses processos são descritos abaixo:

- A compilação da biblioteca *JRastro* ocorre com a execução do comando:
  - $ make
- Na instalação da biblioteca *JRastro* executa-se o comando como usuário root:
  - # make install
- Para desinstalar a biblioteca *JRastro*, execute como root:
  - # make uninstall

3 Utilização

A biblioteca *JRastro* está pronta para ser utilizada, após compilação e instalação. Para realizar o rastreamento de um programa Java (por exemplo *MeuPrograma*) através da biblioteca *JRastro* tem-se duas possibilidades:

- `java -agentlib:JRastro=sEventos,sMetodos MeuPrograma`

  A opção *-agentlib* na execução do programa *MeuPrograma*, faz com que a máquina virtual carregue a biblioteca *JRastro*. A biblioteca será carregada com os parâmetros *sEventos e sMetodos*, que são dois arquivos de configurações para o agente de rastreamento. O arquivo *sEventos* deve conter a seleção de quais eventos se deseja rastrear, e o arquivo *sMetodos* contém a seleção das classes e métodos que serão rastreados, conforme descritos na seção 4.2 do trabalho de graduação. Essa opção indica que a biblioteca *JRastro* será carregada do diretório padrão das bibliotecas (colocada no momento da instalação), ou de um dos diretórios da variável de ambiente `LD_LIBRARY_PATH`.

- `java -agentpath:/usr/libJRastro.so=sEventos,sMetodos MeuPrograma`

  A opção *-agentpath* nessa linha de execução carrega a biblioteca *JRastro* do caminho específico "`/usr/`" com os parâmetros *sEventos e sMetodos*.

Após a execução do programa, os arquivos de rastros gerados devem ser convertidos para o formato do visualizador Pajé. Essa conversão pode ser realizada por um dos três conversores fornecidos com a *JRastro*. Esses conversores recebem como parâmetros um arquivo contendo as informações dos ajustes dos relógios das máquinas[1], ou um arquivo vazio, caso o programa tenha sido executado em uma máquina, e os rastros gerados durante o rastreamento. Abaixo são apresentadas as três conversões possíveis:

- # JRastro_read arq rastro-*

  Essa conversão gera o arquivo no formato do Pajé, chamado "rastros.trace", contendo as informações para a visualização por *threads*.

- # JRastro_readObj arq rastro-*

  Essa conversão gera o arquivo "rastrosObj.trace", que contém as informações para a visualização por *objetos*.

- # JRastro_readObjClass arq rastro-*

  Essa conversão tem como saída o arquivo "rastrosObjClass.trace" contendo as informações para a visualização por *objetos classificados pelas classes*.

[1] Arquivo gerado com a utilização da biblioteca *libRastro* antes e depois da execução do programa.

4 Eventos e funções da JVMTI

A JVMTI dispõem de uma grande quantidade de eventos e funções para o monitoramento de programas Java. A tabela 1 apresenta os eventos disponibilizados pela JVMTI, incluindo os eventos usados neste trabalho (eventos em destaque na tabela).

Tabela 1: Eventos disponibilizados pela JVMTI.

Breakpoint	Method Entry
Class File Load Hook	Method Exit
Class Load	**Monitor Contended Enter**
Class Prepare	**Monitor Contended Entered**
Compiled Method Load	**Monitor Wait**
Compiled Method Unload	**Monitor Waited**
Data Dump Request	Native Method Bind
Dynamic Code Generated	**Object Free**
Exception	Single Step
Exception Catch	**Thread End**
Field Access	**Thread Start**
Field Modification	VM Death Event
Frame Pop	**VM Initialization Event**
Garbage Collection Finish	VM Object Allocation
Garbage Collection Start	VM Start Event

A tabela 2 contém a classificação das funções disponíveis pela JVMTI.

Tabela 2: Classificação das funções da JVMTI.

Memory Management	Field
Thread	Method
Thread Group	Raw Monitor
Stack Frame	JNI Function Interception
Heap	Event Management
Local Variable	Extension Mechanism
Breakpoint	Capability
Watched Field	Timers
Class	System Properties
Object	General