McGill University
School of Computer Science
Sable Research Group

*J Installation

Bruno Dufour

July 5, 2004

*J is a toolkit which allows to dynamically create event traces from Java programs and perform analyses on them. It consists of two main components:

1. The *J JVMPI agent (written in C)
2. The *J analyzer tool (written in Java)

1 Installation Requirements

1. Agent
   - zlib is required for reading `.jar` files as well as producing compressed traces.
   - `gcc` is the recommended compiler for compiling the agent.
   - GNU `make` is the recommended make tool for building the agent.
   - The agent has not been tested under non-UN*X platforms. Contributions relating to portability to other, non-UN*X platforms are most welcome.
2. Analyzer
   - A Java compiler which support Java 1.4 or above is required to compile the analyzer (not required for full distributions).
   - Ant is the recommended build tool for compiling the analyzer (not required for full distribution).
   - A Java Virtual Machine (JVM) which supports the Java 1.4 API or above is required to run the analyzer.

2 About the example commands

1. All example commands assume that an standard, `sh`-compatible UN*X shell is used to type the commands. Such shells include, but are in no way limited to, `sh`, `bash`, `ksh` and `zsh`.
2. All example commands follow the same prompt format. The leftmost part of the command will consist of a directory name inside square brackets. This indicates the current directory while typing (and executing) the command. Immediately following the directory name will be a '`%`' sign, acting as a delimiter between the shell prompt and the actual command to be typed (note that the '`%`' sign is padded on both sides with a space character). Only the part of the line at the right of the '`%`' is to be entered.
3. The output of the command, if relevant to the description, will be shown below the command line. The string '<`snip`>'will be used to denote irrelevant output or parts of the output from a command.
4. An empty prompt will be show after the execution of a command, in order to show the state of the shell at this point.

5. All example commands are provided as a convience to the user only. Users will want to read the description of the step first and ensure that the provided command will perform the described action on their own system. Differences in installations may prevent some commands from working correctly on particular systems; please adjust them as you see fit.

3 Extracting the files

1. If you have not already done so, get the latest copy of the *J distribution from:

   http://www.sable.mcgill.ca/starj/

   and save it somewhere on your system (these instructions will assume that the file is saved under the '`/tmp/`' directory). Change the directory in which you want to install the *J distribution (these instructions will assume that *J is to be installed in the directory named '`~/software/starj-0.1-beta8`'):

```
[~] % cd ~/software/
[software] %
```

2. Next, extract the contents of the file you downloaded:

```
[software] % tar xvzf /tmp/starj-0.1-beta8.tar.gz
<snip>
[software] %
```

3. Change to the newly created directory:

```
[software] % cd starj-0.1-beta8
[starj-0.1-beta8] %
```

4 Building the profiling agent

1. Change to the '`agent`' directory under the root *J installation directory:

```
[starj-0.1-beta8] % cd agent
[agent] %
```

2. Set your `JAVA_HOME` environment variable to point to the root directory of your local Java installation. This directory must include a directory named '`include`', which will itself contain the file '`jvmpi.h`':

```
[agent] % export JAVA_HOME=/opt/sun-jdk-1.4.1.03
[agent] % ls $JAVA_HOME/
bin/  include/  jre/  lib/  share/
[agent] % ls $JAVA_HOME/include/
jawt.h  jni.h  jvmdi.h  jvmpi.h  linux/
```

3. Type '`make`' to compile the agent:

```
[agent] % make
<snip>
[agent] %
```

4. Set your `LD_LIBRARY_PATH` environment variable to include the '`agent/lib`' directory. For example:

```
[agent] % export LD_LIBRARY_PATH=$PWD/lib:$LD_LIBRARY_PATH
[agent] %
```

5. Change back to the root *J directory and test your installation:

```
[agent] % cd ..
[starj-0.1-beta8] % java -Xrunsj:help
StarJ Agent Options
===================
file=<string>               Specifies the name of the output file
specfile=<string>           Specifies the name of the event specifica
events=<event_char>+        Specifies a list of events to record
counters=<event_char>+      Specifies a list of events to count
split=<size>                Specifies the file split threshold
pipe=<bool>                 Instructs the agent to output to a named
verbose=<verbosity_level>   Instructions the agent to produce additio
opt=<bool>                  Instructs the agent to attempt to reduce
cp=<classpath>              Specifies the agent's class path
cp+=<classpath>             Specifies a class path to append to the v
+cp=<classpath>             Specifies a class path to prepend to the
colour=<bool>               Specifies whether or not colour should be
bctags=<bool>               Instructs the agent to output bytecode ta
gzip=<bool>                 Instructs the agent to output a gzip'ped
unique_ids=<bool>           Instructs the agent to output unique iden
help=<string>               Displays usage information

[starj-0.1-beta8] %
```

5 Building the trace analyzer

NOTE: if you downloaded a full or a binary distribution of *J, or if you already have the file '`analyzer/lib/starj.jar`' and don't want to recreate it, change to the '`analyzer`' directory and skip to step 3.

1. Change to the '`analyzer`' directory under the root *J installation directory:

```
[starj-0.1-beta8] % cd analyzer
[analyzer] %
```

2. Type '`ant`' to build the analyzer:

```
[analyzer] % ant
[analyzer] %
```

3. Add the '`analyzer/classes`' directory to your `CLASSPATH`. For example:

```
[analyzer] % export CLASSPATH=$PWD/lib/starj.jar:$CLASSPATH
[analyzer] %
```

4. Change back to the root *J directory and test your installation:

```
[analyzer] % cd ..
[starj-0.1-beta8] % java starj.Main --help
Usage: java starj.Main [options] [trace file]
*J: A Tool for Dynamic Analysis of Java Programs

IO options:
  --dep-graph <filename>                  Outputs dependency graph
  -b <size>, --buffer-size <size>         Sets trace buffer
  -m <stream> <patterns>, --text          Defines a set of metric ele
    <stream> <patterns>                     collect text metric output
  -x <stream> <patterns>, --xml           Defines a set of metric ele
    <stream> <patterns>                     collect XML output from

General options:
  -H, --show-hierarchy                    Displays pack/operation hie
  -p <patterns> <configurations>,         Sets pack/operation options
    --config <patterns>
    <configurations>
  -cp <classpath>, -classpath             Sets the class path
    <classpath>, --classpath
    <classpath>
  -v, --version                           Prints version and exits
  -h, --help                              Print helps and exits

[starj-0.1-beta8] %
```

6 Post-installaction configurations

1. All environment variable changes can be made permanent by including them in you shell initialization scripts ('`.bashrc`', '`.cshrc`', '`.zshrc`' and the like – refer to your shell's documentation)