BTrace: Java™ Platform Observability by Bytecode Instrumentation

A. Sundararajan
http://blogs.sun.com/sundararajann
Kannan Balasubramanian
http://blogs.sun.com/kannan
Learn how to dynamically trace/observe running Java applications (JDK6+) using BTrace.
Agenda

- What is BTrace?
- Probes, Actions
- Trace Class
- “Hello World”
- Demo
- DTrace Integration
- Summary
- Related Sessions
- Q & A
What is BTrace?

- Dynamic tracing tool for Java applications
- Safe tracing tool
  - Observe, don't “disturb”
  - Read, but not “write”
  - Bounded tracing actions (no loops etc.)
- Platform independent
  - Works by bytecode instrumentation using ObjectWeb ASM
- Integration with DTrace on OpenSolaris
  - “whole-stack-tracing” possible
- Open source project - http://btrace.dev.java.net
- GPLv2 + CLASSPATH Exception
- VisualVM plugin and development-time plugin
What is BTrace? (contd.)

Target JVM

BTrace Agent

VisualVM + BTrace plugin

BTrace command line
Probes and Actions

> Probe – location or event in the target application's code
  - Method entry/exit
  - Exception return from a method
  - Line number
  - Field set/get
  - Method call/return (within specified method(s))
  - Exception throw (before)
  - Synchronization entry/exit
  - Timer

> Statements executed on probe “fire”
  - Static methods in trace class
  - Only a “safe” subset of Java allowed inside tracing methods
  - Mostly method calls into BTraceUtils class.
Trace Class

▶ Annotations in “com.sun.btrace.annotations”
▶ @BTrace – flags btrace class
▶ Probe point annotations
  ● @OnMethod
  ● @OnTimer
  ● @OnEvent – event sent by btrace client
  ● @OnExit – exiting btrace client
  ● @OnError – exception from trace method
  ● @OnLowMemory – low memory threshold on specific mem pool
▶ Probe action methods
  ● Public static methods
  ● Static fields for trace state
  ● Field annotations @TLS, @Export
“Hello World”

```java
import com.sun.btrace.annotations.*;
import static com.sun.btrace.BTraceUtils.*;
import java.io.File;

// every time new File(String) is called, print the file name
@BTrace
public class HelloBTrace {
   @OnMethod(
      clazz="java.io.File", method="<init>"
   )
   public static void onNewFile(File self, String name) {
      print("new file ");
      println(name);
   }
}
```
“Hello World” (contd.)

➢ Start target application

    java -jar java2demo.jar

➢ Find the process id using “jps”

    jps

➢ Start btrace client tool

    btrace <pid> HelloBTrace.java
Demo
DTrace Integration

➢ @DTrace and @DTraceRef annotations
  • Run inline/file D-script whenever a specific BTrace script is run

➢ Raise a DTrace USDT probe from BTrace action method
  • DTraceProbe method in BTraceUtils class.
  • Raises “btrace<pid>:::event” probe in D-script.

➢ More integration expected with JDK 7.
Summary

- BTrace – safe, dynamic tracing tool for Java
- [http://btrace.dev.java.net](http://btrace.dev.java.net) - please join and contribute
- More integration with DTrace (JDK 7+) - better JUSDT integration
- VisualVM ([http://visualvm.dev.java.net](http://visualvm.dev.java.net)) plugin
Related Sessions

- **TS-5716**

- **TS-6000**
  - Improving Application Performance with Monitoring and Profiling Tools

- **LAB-9400**
  - Exposing the Depth of Your JDK™ Release 7.0 Applications with Dynamic Tracing (DTrace)

- **TS-6145**
  - Using DTrace with Java™ Technology-Based Applications: Bridging the Observability Gap

- **BOF-4994**
  - End-to-End Tracing of Ajax/Java™ Technology-Based Applications, Using Dynamic Tracing (DTrace)

- **BOF-5223**
  - VisualVM: Integrated and Extensible Troubleshooting Tool for the Java™ Platform
THANK YOU

A. Sundararajan,
http://blogs.sun.com/sundararajans
Kannan Balasubramanian
http://blogs.sun.com/kannan