



Open Source

in the Corporate World

Monitoring & Profiling

Monitoring with open source tools



Agenda

- Monitoring
 - Applications
 - Infrastructure
 - Protocols & Tools
- Profiling
 - Application profiling
 - Service profiling
- Stress Testing



Monitoring

- Application
 - JMX
 - Logging
- Infrastructure
 - SNMP



JMX

- Java Management Extensions – an API and framework for manageable services
 - Defines manageable objects called MBeans
 - Defines attributes, operations and notifications
 - Part of J2EE 1.4 specification
 - J2EE 1.4 also defines JSR 77 and 88 for application monitoring and deployment respectively
 - JSR 160 created a standard for remote JMX that does not require client stubs or the direct use of RMI



Monitoring Tools

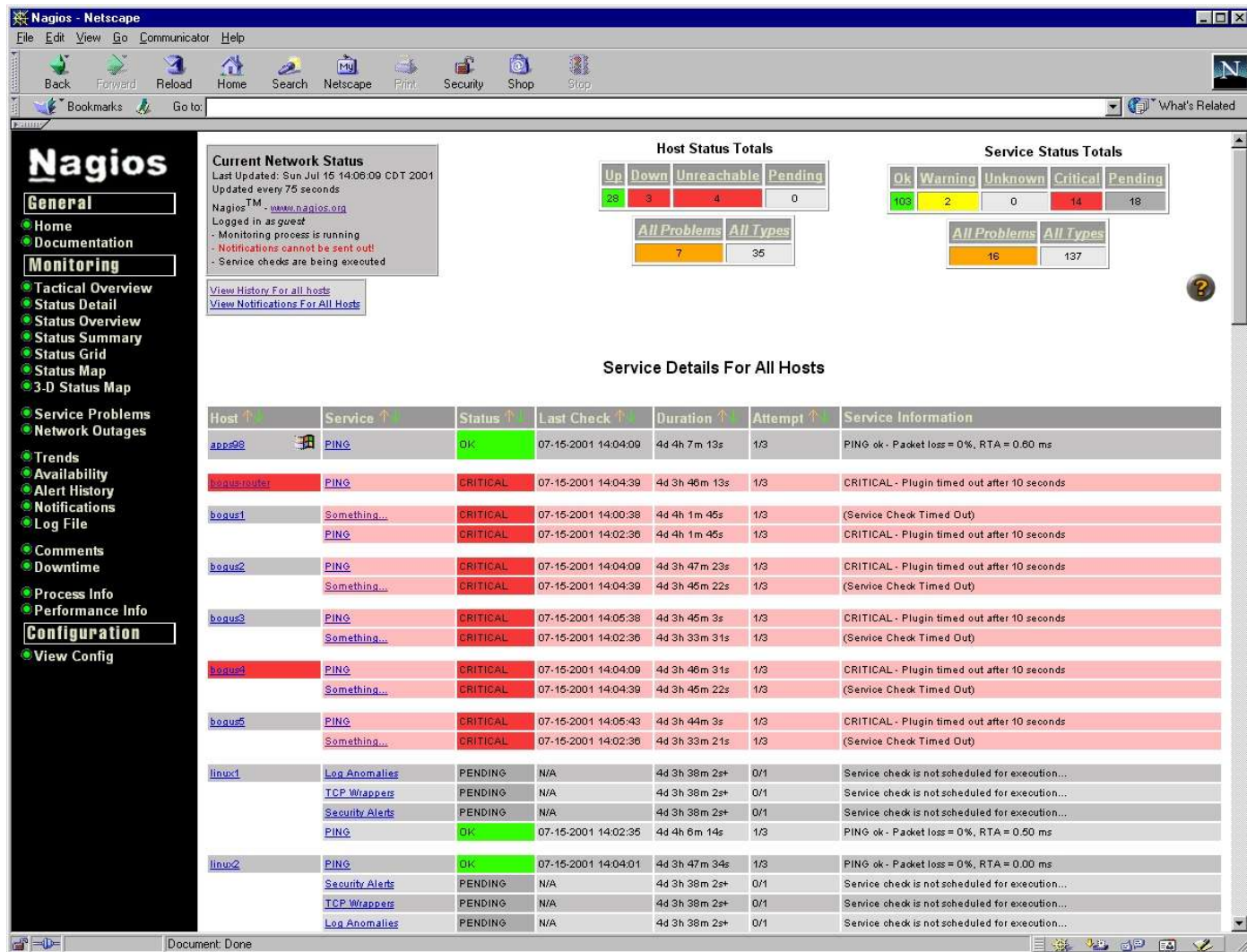
- Nagios
- JMeter
- MC4J
- Commons Modeler
- Chainsaw
- Chires



Nagios

- Network and application monitoring
 - Web-based interface
 - Service monitoring: SMTP, POP3, HTTP, Ping, NNTP
 - Resource monitoring: disk, CPU, memory
 - Contact notifications (pager, email)
 - Problem history reporting
 - Log access

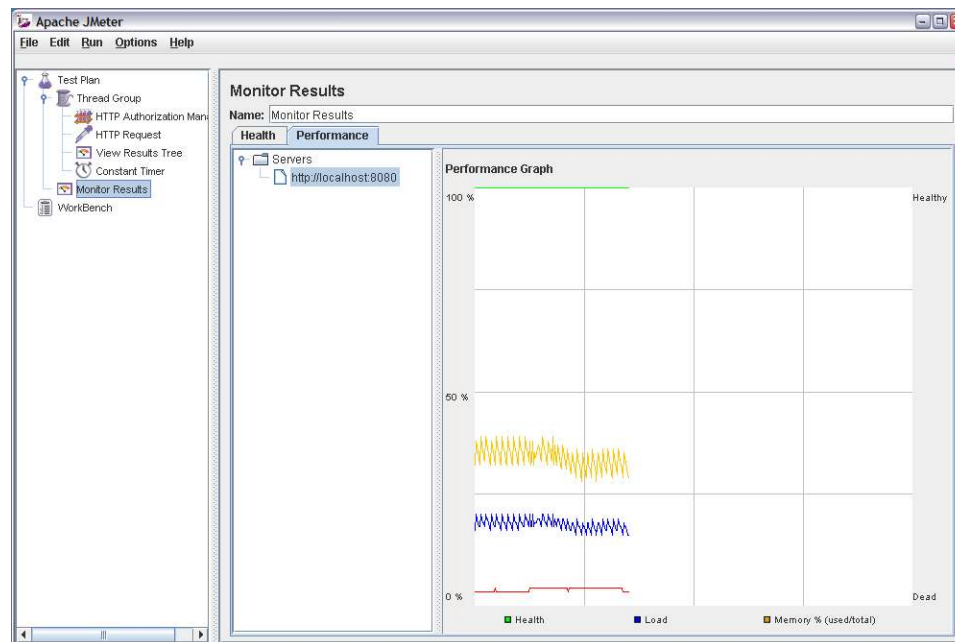
Nagios





JMeter

- Very basic monitoring for Tomcat servers
- Monitors threads and memory usage
- More useful for stress testing than monitoring



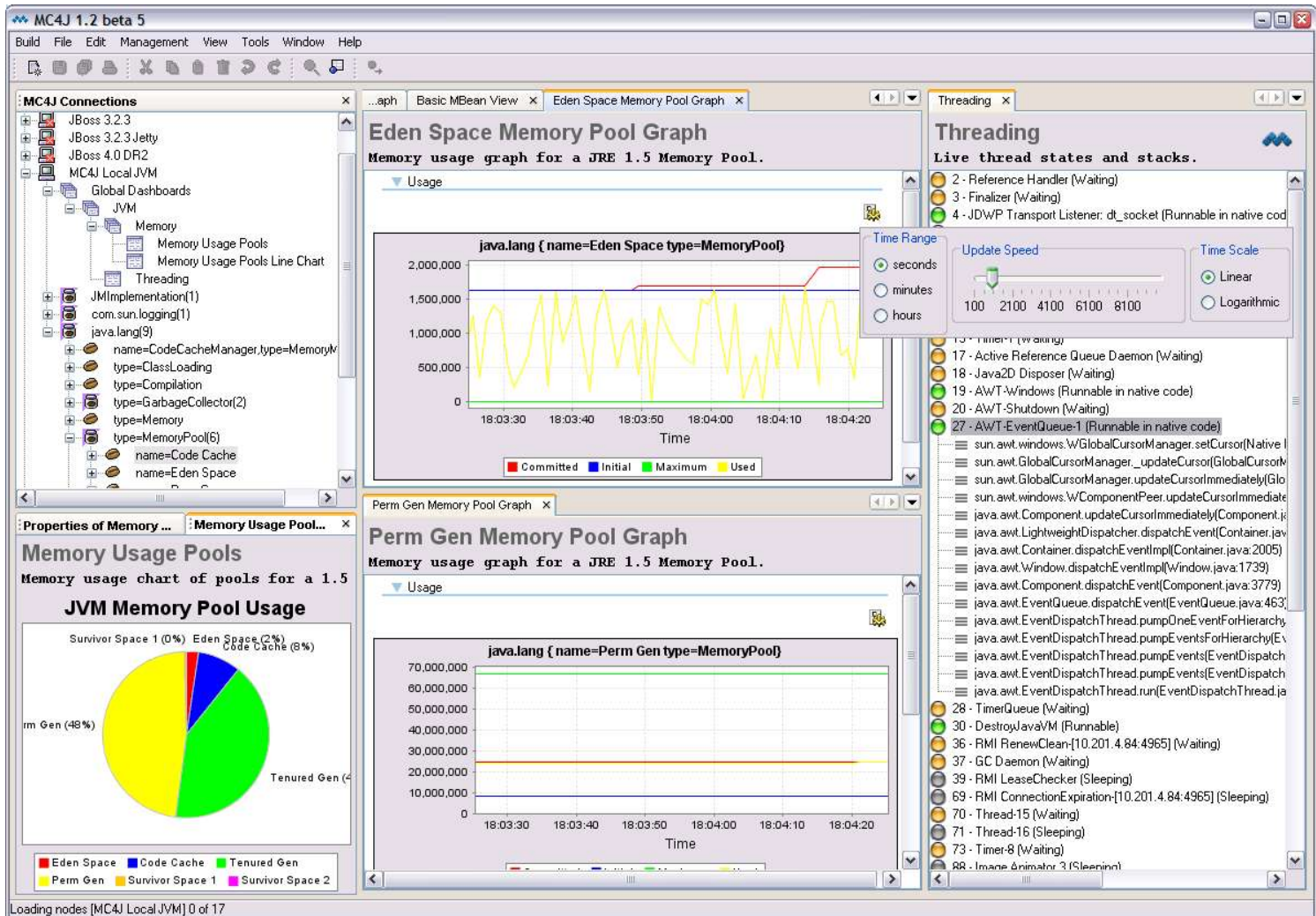


MC4J

- Disclaimer – presenter is author
- JMX Management console
 - Supports all major J2EE application servers
 - Supports parts of JSR 77 for contextual application monitoring and statistics
 - Can directly monitor any Java 5 JVM and supports standard JVM monitoring such as threads, memory pools and GC
 - Can graph numerical and statistical values
 - Can perform supported operations and configurations on a server
 - Supports customized dashboard views for service tailored interfaces



MC4J





Commons Modeler

- Simplifies the exposure of objects to JMX
- Externalizes MBean metadata into XML
- Declaratively choose which attributes and operations are exposed

```
URL url= this.getClass().getResource
        ("/org/chiress/model/history/generic/mbeans-descriptors.xml");
Registry registry = Registry.getRegistry(this,null);
registry.loadMetadata(url);
registry.registerComponent(
    this,
    "Chiores:type=GenericModel,name=" + key,
    "org.chiores.model.history.generic.GenericModel");
```



Chainsaw

- Utility for browsing and searching log files
- Quickly filter by logger and highlight important messages through expressions
- Can read XML files or receive messages via socket connections from server processes



Chainsaw

Chainsaw v2 - Log Viewer

File View Current tab Help

Refine focus on:

ID	Timestamp	Level	Logger	Thread	Message
142	2004-05-12 15:43:02,311		com.mycompany....	Thread-1	infomsg 141
143	2004-05-12 15:43:02,311		com.mycompany....	Thread-1	warnmsg 142
144	2004-05-12 15:43:02,311		com.someotherco...	Thread-1	errmsg 143
145	2004-05-12 15:43:03,313		com.mycompany....	Thread-1	debugmsg 144 g dg so
146	2004-05-12 15:43:03,313		com.mycompany....	Thread-1	infomsg 145
147	2004-05-12 15:43:03,313		com.someotherco...	Thread-1	warnmsg 146
148	2004-05-12 15:43:03,313		com.mycompany....	Thread-1	errmsg 147
149	2004-05-12 15:43:03,313		com.mycompany....	Thread-1	debugmsg 148
150	2004-05-12 15:43:03,313		com.someotherco...	Thread-1	infomsg 149
151	2004-05-12 15:43:03,313		com.mycompany....	Thread-1	warnmsg 150
152	2004-05-12 15:43:03,313		com.mycompany....	Thread-1	errmsg 151
153	2004-05-12 15:43:03,313		com.someotherco...	Thread-1	debugmsg 152
154	2004-05-12 15:43:03,313		com.mycompany....	Thread-1	infomsg 153
155	2004-05-12 15:43:03,313		com.mycompany....	Thread-1	warnmsg 154
156	2004-05-12 15:43:03,313		com.someotherco...	Thread-1	errmsg 155

Level ERROR

Logger com.someothercompany.corecomponent

Time 2004-05-12 15:43:03,313

Thread Thread-1

Message errmsg 155

NDC null

Class

Method

Line

File

Properties {{hostname,localhost}}{some string,some valueGenerator 3}{log4jid,156}{application,Generator 3}}

Throwable org.apache.log4j.chainsaw.Generator.run(Unknown Source) at java.lang.Thread.run(Thread.java:534)

localhost-Generator 3 localhost-Generator 2 localhost-Generator 1 ChainsawCentral Welcome

Receiver's panel: false

Chainsaw Tutorial

Start Tutorial Stop Tutorial

Welcome to the Chainsaw v2 Tutorial. Here you will learn how to effectively utilise the many features of Chainsaw.

[Expressions](#)

[Color filters](#)

[Display filters](#)

Conventions

To assist you, the following documentation conventions will be used

- Interesting items will be shown like this
- Things you should try during the tutorial will be shown like this

Outline

The built-in tutorial installs several "pretend" Receiver plugins that generate some example LoggingEvents and post them into Log4j just like a real Receiver.

- If you would like to read more about Receivers first, then click here. (TODO)

When you are ready to begin the tutorial, [click here](#), or click the "Start Tutorial" button in this dialog's toolbar.

Receivers

After you have said yes to the confirmation dialog, you should see 3 new tabs appear in the main GUI. This is because the tutorial has installed 3 'Generator' Receivers into the Log4j engine.

Confirm this by choosing the Receivers



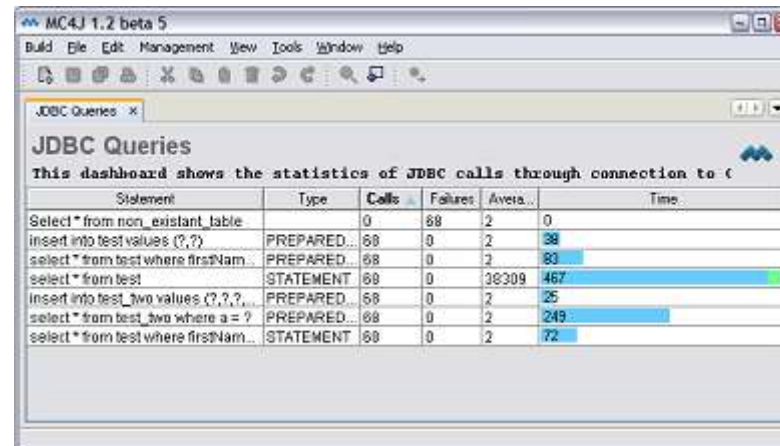
Chires

- Utilize AOP to introduce JMX MBean support into plain old Java objects
- Automatically registers objects on creation or lifecycle events
- Configured via annotations or XML
- May introduce value monitoring and call execution

```
@MxBean(  
    description = "I am an auto-managed bean. Hello.",  
    objectName = "name=ManageMe")  
public class AutoManageMe {  
  
    @MxAttribute(  
        description = "This is an attribute that is only backed by a simple field",  
        isReadable = true, isWritable = false)  
    private String infoFieldAttribute = "Some other default text";  
  
    @MxOperation(description = "This is a basic operation", impact = "info")  
    public String getMeAString() {  
        return nonAttributeField;  
    }  
}
```



Chires





Profiling Tools

- Application Profiling
 - NetBeans Profiler
 - Eclipse Profiler
- Service Profiling
 - P6Spy
 - Chires
 - Helios
 - JRat

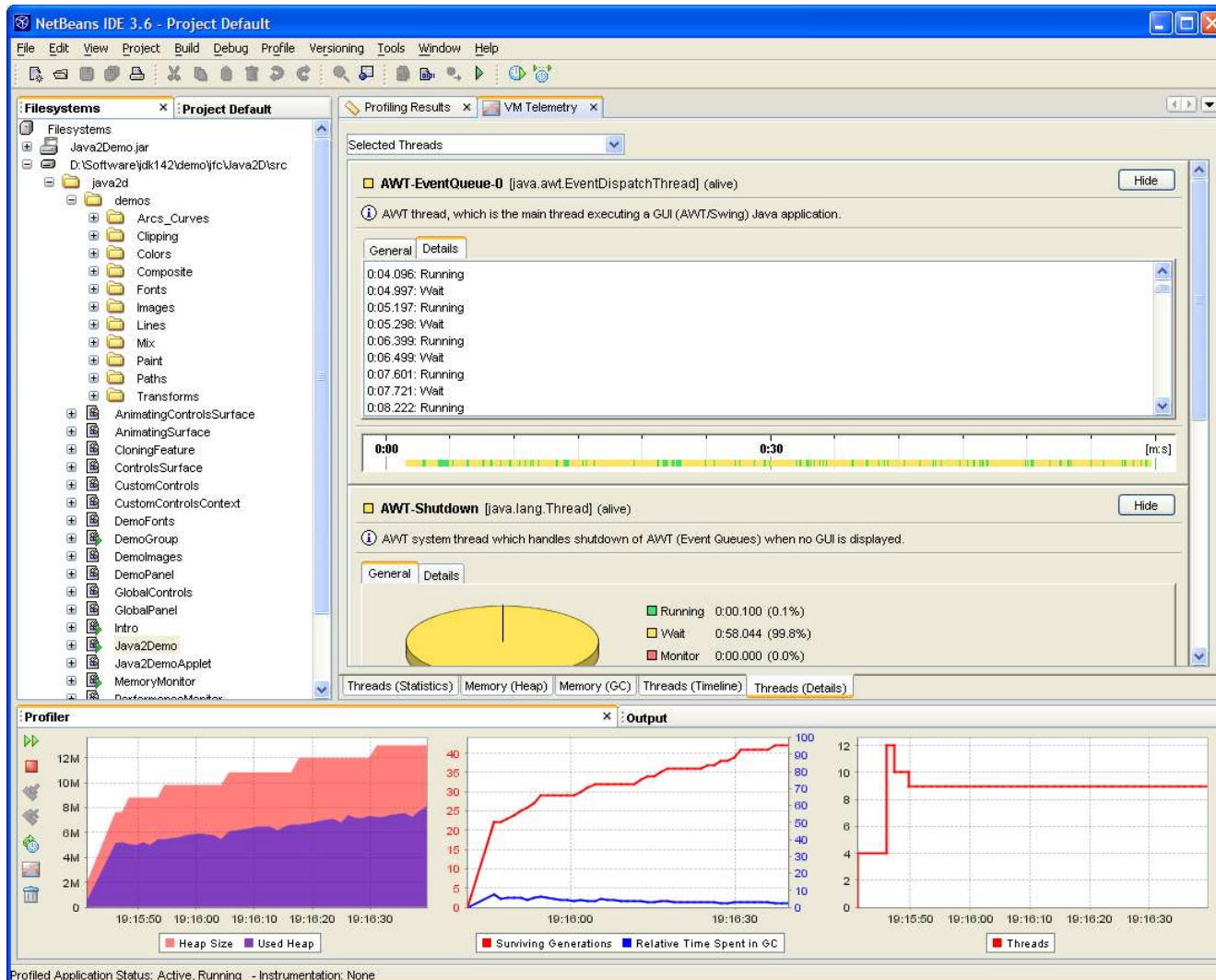


NetBeans Profiler

- Low overhead profiling
- Utilizes JFluid JVM technology for fast byte-code enhancement
- Requires use of special JVM
- Performance profiling – CPU time
- Memory profiling
- Thread profiling, concurrency



NetBeans Profiler



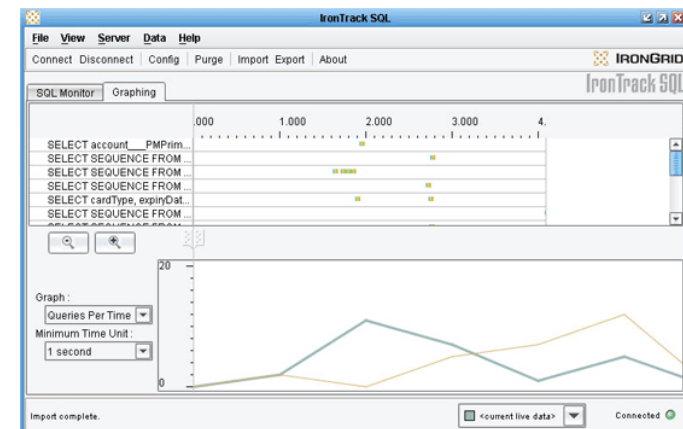
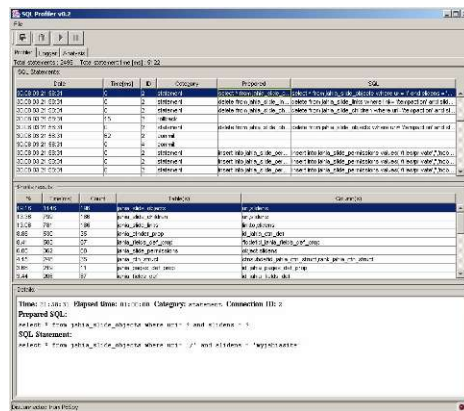


-
- The call graph illustrates the execution flow of the de.dreger.logazer.parser.log.TFFileLogZip.GetLine method. The graph shows a series of method calls starting from the 'main' node, through 'next', 'Eat', and several instances of 'getValue' and 'calcValue', eventually leading to 'readCor'. A yellow box highlights performance statistics for the 'GetLine' method.
- | Method | % of time | Time (ms) | Invocations |
|---|-----------|-----------|-------------|
| de.dreger.logazer.parser.log.TFFileLogZip.GetLine | 1.26% | (1269 ms) | |
| used directly | 0.63% | | (629 ms) |
| invoked | 0.22% | | (100902) |
| used in invoked methods | 0.64% | | (640 ms) |



P6Spy

- P6Spy is a framework for intercepting JDBC calls
- It installs as a proxy to your real JDBC driver
- P6Log logs SQL statements as they are sent to the driver and can intercept anything using JDBC (e.g. EJBs)
- Two visual tools for tracking and monitoring SQL executions from P6Spy (Irongrid IronTrack and Sql Profiler)





Chires

- An AOP and JMX monitoring framework
- Instruments any Java class through byte code alteration
- Provides JMX interface to statistics, state and operations without code changes

MC4J 1.2 beta 7
Management Tools Window Help
67 8.9/31.8MB

Chires Profiler x
The Chires Profiler view.

Name	Calls	Time	Throws
FlowEntry: org.chires.test.flow.TargetClass::thirdAlternateMethod	30	2 secs 411 ms	5
FlowEntry: org.chires.test.flow.TargetClass::secondMethod	25	41 secs 693 ms	0
FlowEntry: org.chires.test.flow.TargetClass::thirdMethod	25	37 secs 939 ms	0
FlowEntry: org.chires.test.flow.TargetClass::fourthMethod	5	37 secs 587 ms	0
FlowEntry: org.chires.test.flow.TargetClass::fifthMethod	100	37 secs 167 ms	0
FlowEntry: org.chires.test.flow.TargetClass::sixthMethod	100	34 secs 899 ms	0
FlowEntry: org.chires.test.flow.TargetClass::sixthMethod	100	29 secs 355 ms	0
FlowEntry: org.chires.test.flow.TargetClass::sixthMethod	100	23 secs 721 ms	0
FlowEntry: org.chires.test.flow.TargetClass::sixthMethod	100	16 secs 977 ms	0
FlowEntry: org.chires.test.flow.TargetClass::sixthMethod	100	11 secs 327 ms	0
FlowEntry: org.chires.test.flow.TargetClass::sixthMethod	100	5 secs 783 ms	0
FlowEntry: org.chires.test.flow.TargetClass::thirdAlternateMethod	4	260 ms	3



Helios

- New project for JMX-based resource monitoring
- Supports real-time resource monitoring, aggregation and history reporting
- Will support Nagios reporting when completed
- Utilizes JRat for application monitoring



JRat

- Java Runtime Analysis Tool
- Accumulate timing statistics
- Create trace logging
- Track response times of methods over time
- Can integrate via offline bytecode instrumentation (with an ant task)
- Supports JBoss 4 AOP
- Can also instrument via dynamic proxies



Stress testing

- Purpose
 - Scalability
 - Finding failure point
 - Testing degradation
 - Failure
 - Failover behavior



JMeter

- Pure Java Swing application
- Provides testing of HTTP, web services, JDBC, FTP and LDAP
- Offers configuration of test scripts via a proxy, allowing recording of browsing
- Supports form parameters from files
- Manages cookies
- Threaded load generation with remote engine control from a single GUI



OpenSTA

- Windows only stress testing
- Supports distributed load generation
- Custom scripting language
- Weak proxy script generation to script



Questions?