agilis
statistics & informatics

Profiling Java Applications

Kostis Kapelonis - Agilis SA

The need for speed

agilis
statistics & informatics

Topics

- Software Quality with FindBugs
- Using Jconsole
- Monitoring with Netbeans 6
- Profiling CPU with Netbeans 6
- Profiling Memory with Netbeans 6
- Conclusion

agilis
statistics & informatics

Software quality tools

- FindBugs
- PMD
- CheckStyle
- Run from command line
- Use GUI (even webstart)
- Integrate into Netbeans/Eclipse
- Detect problematic situations

agilis
statistics & informatics

Problematic situations

- Ignoring return values from methods.
- Impossible casts.
- Unclosed Streams
- Security Problems
- Fields that should be static
- Performance problems
- Possible null pointers
- More ….

FindBugs Demo

QuickTime and a
decompressor
are needed to see this picture.

agilis
statistics & informatics

Topics

- Software Quality with FindBugs
- Using Jconsole
- Monitoring with Netbeans 6
- Profiling CPU with Netbeans 6
- Profiling Memory with Netbeans 6
- Conclusion

agilis
statistics & informatics

Jconsole

- Introduced in Java 5 (experimental)
- Enhanced in Java 6
- Shows basic information for JVM
- Memory (e.g. heap size)
- Threads (active/total)
- Classes Loaded
- JVM environment properties

agilis
statistics & informatics

Jconsole usage

1. Start application with -Dcom.sun.management.jmxremote
2. Launch jconsole
3. Attach it to the application

In Java6 step 1 is not needed

Jconsole Demo

QuickTime and a
decompressor
are needed to see this picture.

agilis
statistics & informatics

Topics

- Software Quality with FindBugs
- Using Jconsole
- Monitoring with Netbeans 6
- Profiling CPU with Netbeans 6
- Profiling Memory with Netbeans 6
- Conclusion

agilis
statistics & informatics

Netbeans 6 profiler suite

- Integrated into Netbeans by default
- Basic information (similar to jconsole)
- CPU analysis (hotspots)
- Memory analysis (memory leaks)
- Used to optimize a correct program
- Imposes *overhead* on the application

agilis
statistics & informatics

Basic profiling

- Offers information similar to jconsole
- Heap size
- Threads
- Classes loaded
- Minimal overhead for the application
- Used for an overview

Basic profile Demo

QuickTime and a
decompressor
are needed to see this picture.

agilis
statistics & informatics

Topics

- Software Quality with FindBugs
- Using Jconsole
- Monitoring with Netbeans 6
- Profiling CPU with Netbeans 6
- Profiling Memory with Netbeans 6
- Conclusion

agilis
statistics & informatics

CPU profiling

- Optimize always the frequent code
- Use CPU profile to find this code
- 20% of the code runs 80% of time
- Optimize this 20%
- Medium overhead for the application
- Also see time spent for GC/IO/Gui e.t.c.

CPU profile Demo

IntegratedPPP – NetBeans IDE 6.0

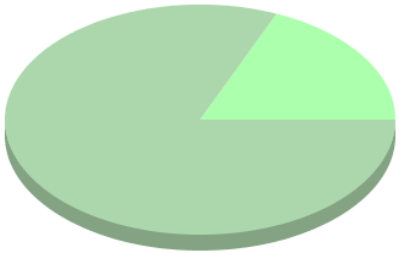
Threads × Live Profiling Results ×

Hot Spots – Method	Self time [%]	Self time	Invocations
com.agilis.ppp.det.readers.XMLItemListReader.**readSurveyTree** (String)		322 ms (16.4%)	1
com.agilis.ppp.det.ui.MainFrame.**main** (String[])		260 ms (13.2%)	1
com.agilis.ppp.det.ui.ItemPane.**getItemNotesPane** ()		187 ms (9.5%)	1
com.agilis.ppp.det.readers.SettingsReader.**read** (java.io.InputStream)		187 ms (9.5%)	1
com.agilis.ppp.det.readers.SDMXPPPFamilyReader.**parseFile** (String)		168 ms (8.5%)	1
com.agilis.ppp.det.ui.MainFrame.**decideOnStartup** ()		87.5 ms (4.4%)	1
com.agilis.ppp.det.ui.ColorRenderer.**getTreeCellRendererComponent** (javax.swin...		84.1 ms (4.3%)	18
com.agilis.ppp.det.ui.MainFrame.**showCurrentSurvey** (javax.swing.JTree)		79.6 ms (4%)	1
com.agilis.ppp.det.ui.MainFrame.**initialize** ()		75.8 ms (3.8%)	1
com.agilis.ppp.det.ui.DetJavaHelp.**<init>** ()		71.6 ms (3.6%)	1
com.agilis.ppp.det.ui.MainFrame.**enableSurveyMenus** (boolean)		64.5 ms (3.3%)	1
com.agilis.ppp.det.ui.ItemPane.**getVatField** ()		55.2 ms (2.8%)	1
com.agilis.ppp.det.fs.SurveyDetails.**load** (String)		39.0 ms (2%)	4
com.agilis.ppp.det.ui.MainFrame.**getAMenuBar** ()		36.8 ms (1.9%)	1
com.agilis.ppp.det.ui.MainFrame.**getHorizontalSplitPane** ()		32.2 ms (1.6%)	2
com.agilis.ppp.det.ui.filters.FilterTable.**populateMenu** (javax.swing.JPopupMenu, S...		21.0 ms (1.1%)	14
com.agilis.ppp.det.ui.MainFrame.**<init>** ()		19.1 ms (1%)	1
com.agilis.ppp.det.ui.ItemPane.**getAttributesScrollPane** ()		15.6 ms (0.8%)	1
com.agilis.ppp.det.ui.ItemPane.**getNameCodeLabel** ()		14.6 ms (0.7%)	1
com.agilis.ppp.det.ui.tables.TableMaker.**createRootSummaryTable** (Object[][])		14.5 ms (0.7%)	1
com.agilis.ppp.det.ui.SearchPane.**createGUI** ()		13.5 ms (0.7%)	2
com.agilis.ppp.det.ui.MainFrame.**getTreePanel** ()		11.7 ms (0.6%)	2
com.agilis.ppp.det.ui.ItemPane.**<init>** (com.agilis.ppp.det.ui.MainFrame, com.agili...		10.8 ms (0.6%)	1
com.agilis.ppp.det.ui.ItemPane.**getConceptPanel** ()		9.65 ms (0.5%)	1
com.agilis.ppp.det.fs.SurveyManager.**rememberSurveyDetails** (com.agilis.ppp.det...		8.67 ms (0.4%)	1
com.agilis.ppp.det.control.Controller.**loadTreeOnly** (com.agilis.ppp.det.fs.SurveyD...		8.55 ms (0.4%)	1
com.agilis.ppp.det.ui.filters.FilterTable.**fillTable** ()		7.31 ms (0.4%)	1
com.agilis.ppp.det.ui.tables.IntegerTableCellRenderer.**setValue** (Object)		6.37 ms (0.3%)	54
com.agilis.ppp.det.ui.filters.FilterTable.**<init>** (com.agilis.ppp.det.control.Controll...		6.21 ms (0.3%)	1
com.agilis.ppp.det.ui.tables.ItemStatsTable.**createGUI** ()		3.81 ms (0.2%)	1
com.agilis.ppp.det.ui.MainFrame.**showRootWindow** ()		3.64 ms (0.2%)	1
com.agilis.ppp.det.ui.DetJavaHelp.**bindToHelp** (javax.swing.JMenuItem, String)		3.29 ms (0.2%)	5
com.agilis.ppp.det.control.Controller.**updateWorkspacePath** (String)		3.24 ms (0.2%)	1

[Method Name Filter]

DrillDown

Scope: Project/Generic UI



AWT & Swing Painters: 18.4%

AWT & Swing Listeners: 81.6%

Method categories

AWT & Swing Listeners (100.0%)

HTTP Request tracker

No data available

agilis
statistics & informatics

Topics

- Software Quality with FindBugs
- Using Jconsole
- Monitoring with Netbeans 6
- Profiling CPU with Netbeans 6
- Profiling Memory with Netbeans 6
- Conclusion

agilis
statistics & informatics

Memory profiling

- Java code suffers from memory leaks
- Minor leaks (objects allocated once)
- Major leaks (object allocated multiple times)
- Garbage collector is your friend
- Difficult to distinguish memory leaks from long-lived objects
- Maximum overhead for the application

agilis
statistics & informatics

Introducing Generations

- Each object has an age.
- Age is the number of times it has survived garbage collection.
- Average age is sum of ages / objects
- Generations are different ages
- A high generation number might be a memory leak
- A high age might or might not be a memory leak

Generations example 1

QuickTime and a
decompressor
are needed to see this picture.

- Ages are 5,10,15
- Average age is 5 + 10 +15 / 3= 10
- Generations are 3

Generations example 2

QuickTime and a
decompressor
are needed to see this picture.

- Ages are 5,10,10,50,50
- Average age is sum 125 /5 = 25
- Generations are AGAIN 3

agilis
statistics & informatics

Profile goal

- Run the application normally
- Collect data for a period of time
- Pinpoint objects with extreme values
- High age might be a minor leak
- High generations might be a major memory leak
- Remember that profiling imposes overhead

agilis
statistics & informatics

Memory profile Demo

Live Bytes	Live Bytes	Live Objects	Allocated Objects	Avg. Age	Generations
	124,20...(21.9%)	2,936 (18.6%)	11,296	24.9	53
	68,160 B (12%)	2,840 (18%)	8,372	25.4	54
	27,680 B (4.9%)	718 (4.5%)	900	24.9	10
	25,888 B (4.6%)	809 (5.1%)	809	25.0	9
	23,232 B (4.1%)	726 (4.6%)	726	25.1	10
	22,800 B (4%)	1,425 (9%)	1,425	25.0	10
	21,512 B (3.8%)	33 (0.2%)	217	39.0	20
	17,712 B (3.1%)	738 (4.7%)	738	25.0	10

agilis
statistics & informatics

Conclusions

- The tools are great but
- The application must be correct
- Profiling imposes overhead
- Profiling is time consuming

agilis
statistics & informatics

Quotes

- *"Premature optimization is the root of all evil"* by Donald Knuth
- *"The First Rule of Program Optimization: Don't do it."*
- *"The Second Rule of Program Optimization (for experts only!): Don't do it yet."* by M A Jackson