Profiling

Jörg Kienzle

Profiling (1)

- A Profiler allows developers to analyze a program execution (an run-time or post run-time) in order to find bugs or optimize performance
- A Profiler reveals
  - Which methods called which other methods
  - How many times methods are called
  - How long methods take to execute
  - When objects are allocated
  - How many objects are allocated
  - How many threads are running
  - How long threads need to wait for resources

Profiling (2)

- The information gathered by a profiler can be used to
  - Measure and locate performance bottlenecks in your code
    - Methods that take a long time to execute
    - Methods that are called (too) many times
    - Unnecessary memory allocations
      - E.g. object allocations inside a loop that could be replaced by a single allocation outside of the loop
  - Find memory leaks
  - Detect deadlocks
    - Threads waiting for mutual resources

Profiling Problems

- Profiling produces overhead
- A Profiler only gives insight on code that is executed
- Interpretation of results can be difficult
  - Identifying the “crucial” parts of the software is not always straightforward
  - Recognizing potential for improvements is also non-trivial

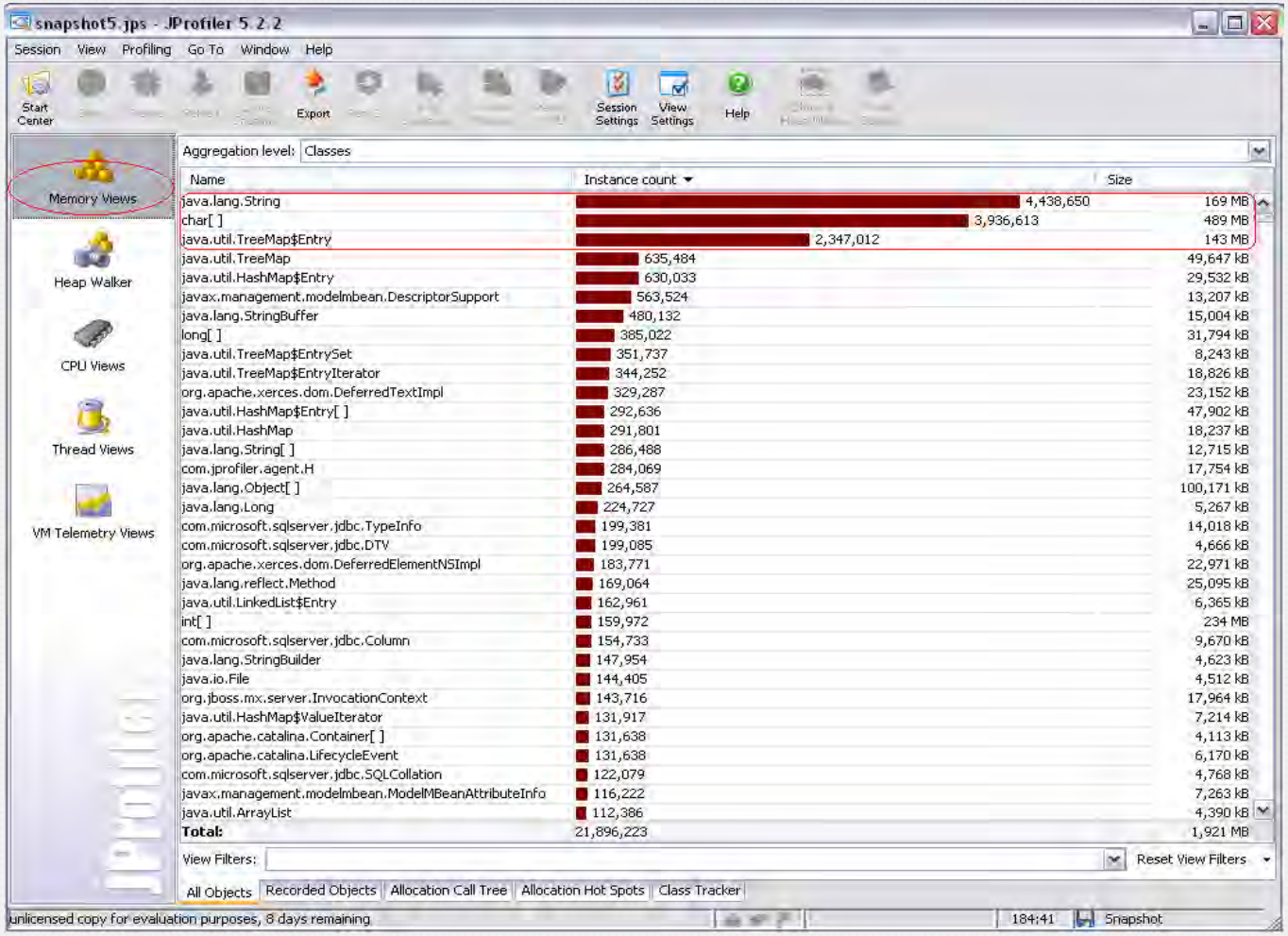
Profilers

- Profilers exist for most programming languages / development environments
- C / C++
  - gprof
    - Application needs to be compiled with option profiling (gcc -pg)
- Java
  - JProfiler (http://www.ej-technologies.com/)
    - The profiler attaches to the Java Virtual Machine
  - YourKit Java Profiler (http://www.yourkit.com/java/profiler/)
  - Open Source Profilers (http://java-source.net/open-source/profilers)

JProfiler

- Current version: 8.1
- 10 day free trial evaluation key
  - For each email address ☺
- Integrates with Eclipse
- Allows to selectively turn profiling on or off
- Allows to specify filters to focus on specific packages/ classes / methods only
- Help / Tutorials / Screencasts available at
  - Http://www.ejtechnologies.com/

Memory Analysis using the Memory View



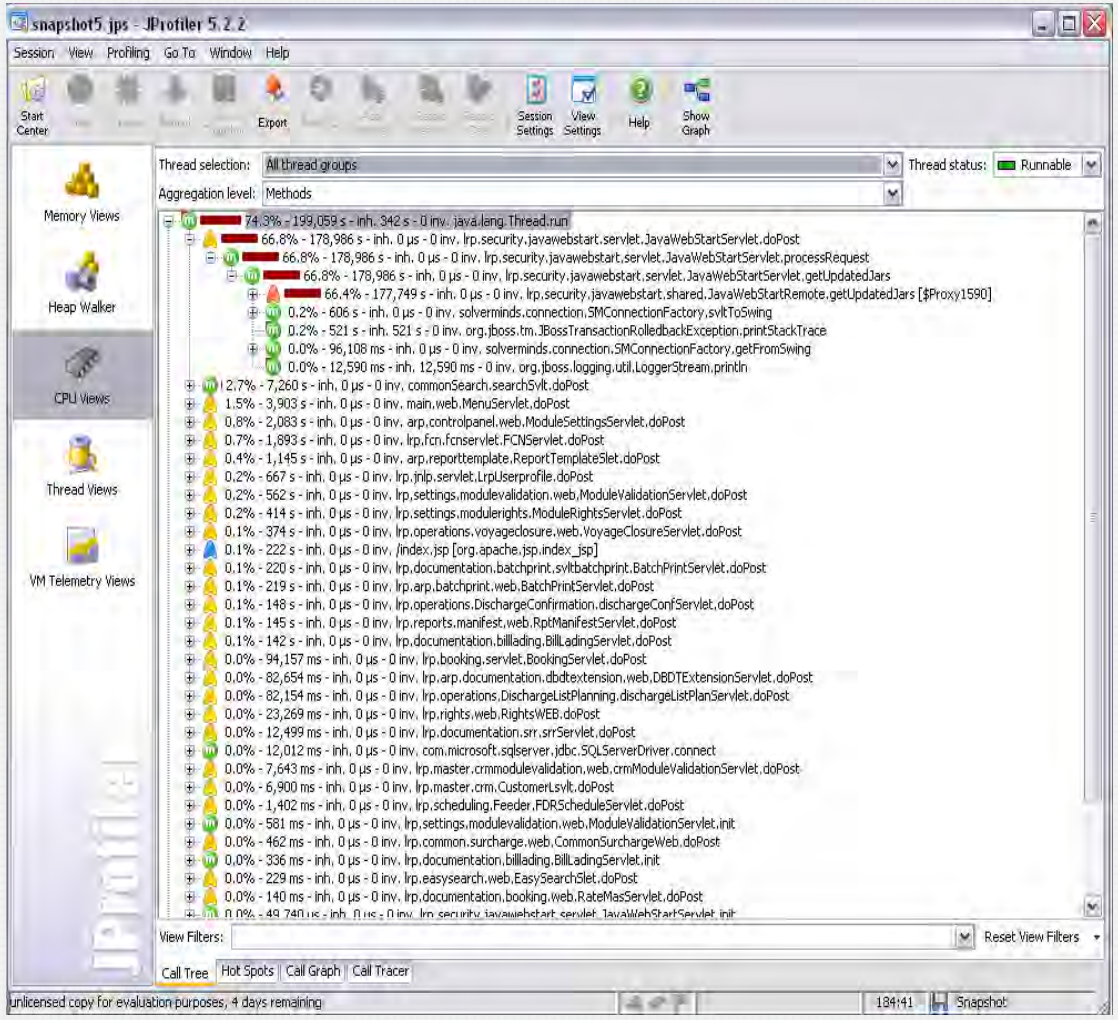
Memory View Details

- "All objects" view shows all classes and the number of live instances at the moment
- Recording of allocations can be turned on in the "Recorded Objects" view
  - It is possible to "mark" a current state to see comparatively how many new objects are allocated from then on
- Allocation Hotspot view shows where (the most) objects are allocated

Using the Heap Walker

- Take a Snapshot of the Heap
- Select Class of Interest
- Select Object of Interest
  - Display when it was allocated
  - Look at what other objects reference this object
  - Look at how it is connected to a GC root

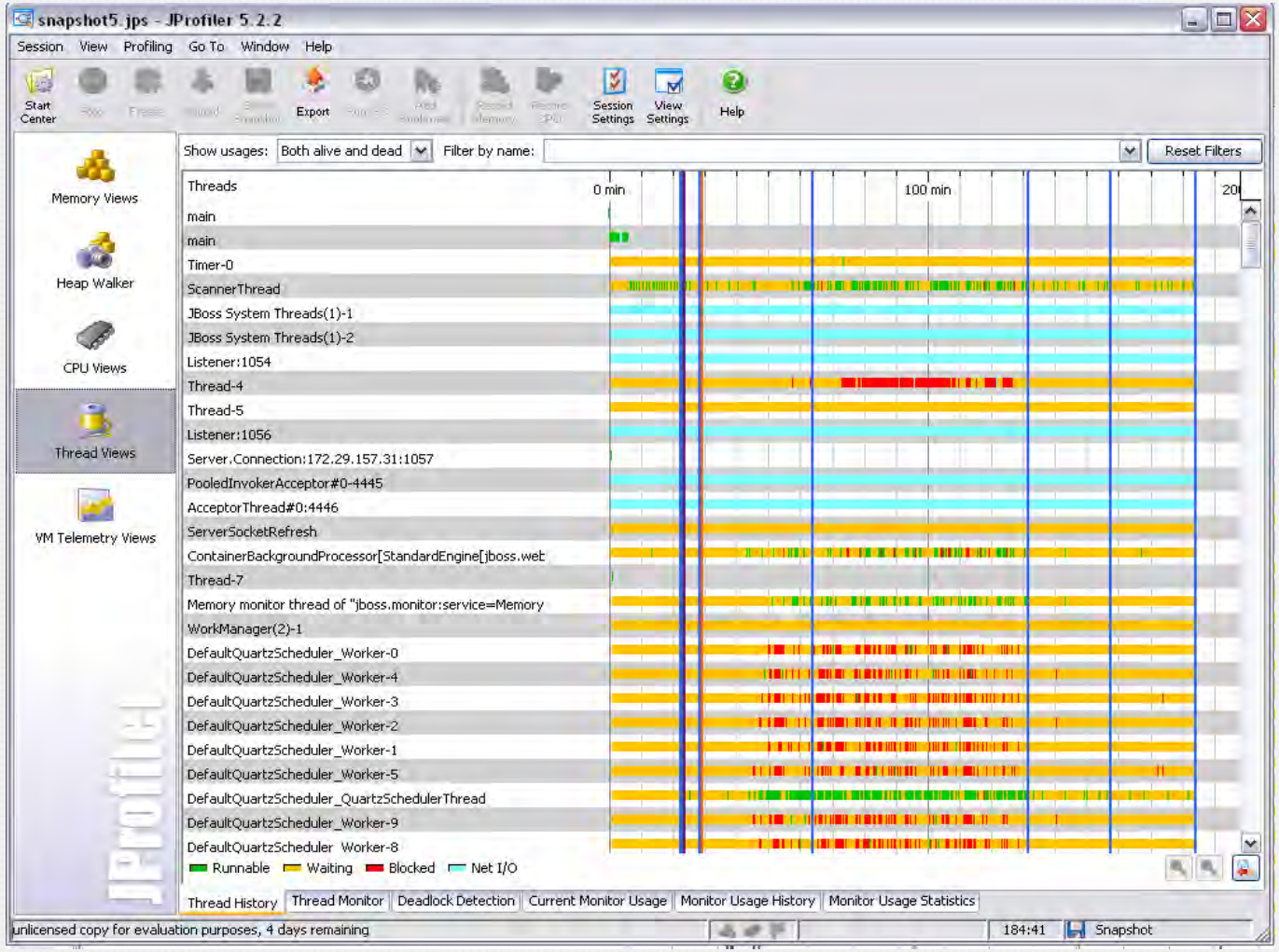
Using the CPU View



Using the CPU Views

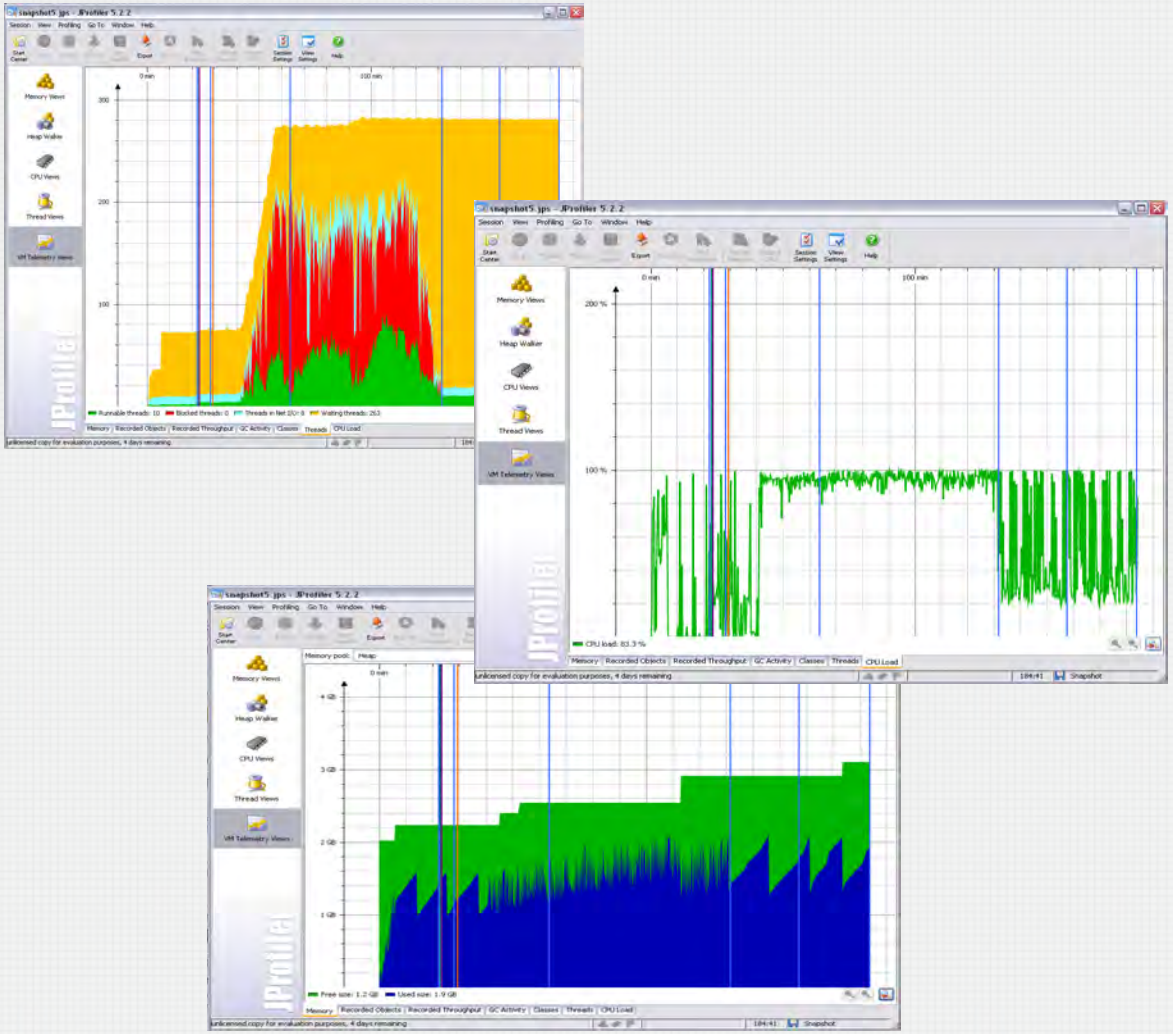
- Turn on CPU Recording
- Call Tree View
  - Displays cumulated method call information
    - Execution time of the method under inspection including called methods
      - Default is “runnable” time
    - What other methods the method under inspection calls
    - What other methods invoke the method under inspection
- Hot Spot View
  - Displays methods in decreasing order of total execution time spent in them
- Call Graph View to navigate through the call graph
- Method Statistics View to display minimal, average, maximum execution time
- Call Tracer View to record a detailed history of each call
  - Very high overhead!

Thread View


snapshot5.jps - JProfiler 5.2.2
Session View Profiling Go To Window Help
Start Center
Export
Session Settings
View Settings
Help
Memory Views
Heap Walker
CPU Views
Thread Views
VM Telemetry Views
Show usages: Both alive and dead
Filter by name:
Reset Filters
Threads
0 min
100 min
main
main
Timer-0
ScannerThread
JBoss System Threads(1)-1
JBoss System Threads(1)-2
Listener:1054
Thread-4
Thread-5
Listener:1056
Server.Connection:172.29.157.31:1057
PooledInvokerAcceptor#0-4445
AcceptorThread#0:4446
ServerSocketRefresh
ContainerBackgroundProcessor[StandardEngine[jboss.web
Thread-7
Memory monitor thread of "jboss.monitor:service=Memory
WorkManager(2)-1
DefaultQuartzScheduler_Worker-0
DefaultQuartzScheduler_Worker-4
DefaultQuartzScheduler_Worker-3
DefaultQuartzScheduler_Worker-2
DefaultQuartzScheduler_Worker-1
DefaultQuartzScheduler_Worker-5
DefaultQuartzScheduler_QuartzSchedulerThread
DefaultQuartzScheduler_Worker-9
DefaultQuartzScheduler_Worker-8
Runnable Waiting Blocked Net I/O
Thread History
Thread Monitor
Deadlock Detection
Current Monitor Usage
Monitor Usage History
Monitor Usage Statistics
unlicensed copy for evaluation purposes, 4 days remaining
184:41
Snapshot

VM Telemetry Views

- Display Info On the JVM
  - Memory Usage
  - CPU Usage
  - Thread States
  - GC Activity



Demo

Questions?