
July 2005

# Overview of Profiling Java Applications



- Richard Sharpe
- Technical Specialist

# Presentation Goal

**Making the most of your Java applications through profiling techniques.**

# Performance Results

- Throughput for a bank – a bank charges 10c a transaction and can handle 10000 transactions a day.
- By  Performance Profiling they increased throughput by 5% a day.
- The bank can make an extra €1000 x 5% = €50 a day.
- Number of working days a year = 255 days x €50 = €12750 extra revenue

# Numerous Performance Restrictions

- Hardware Constraints
  - CPU
  - Memory
- Network Capabilities
  - Bandwidth
  - Mis-configured load balancers
- Design Restrictions
  - Device oriented
  - Customer Standards
- Code Development
  - Slow I/O – network/disk/DB access
  - Inexperienced developers

# Approach To Profiling

- Don’t leave it too late
- Define Benchmarks
  - How much faster?
- Time vs. Performance Increase
  - Concentrate on functionality first
- Design of the application is important
  - Plan and budget for profiling as part of testing or contingency (threading deadlocks)
- Recursive process
  - Repeat function testing if code is changed
  - More problems may filter in after initial bottlenecks are fixed

# Tweaking Your System

- Using Java will always result in some overhead
- CPU overloading and Memory paging
- Set –Xms to the same value as –Xmx
- Use up to date JRE (if possible)

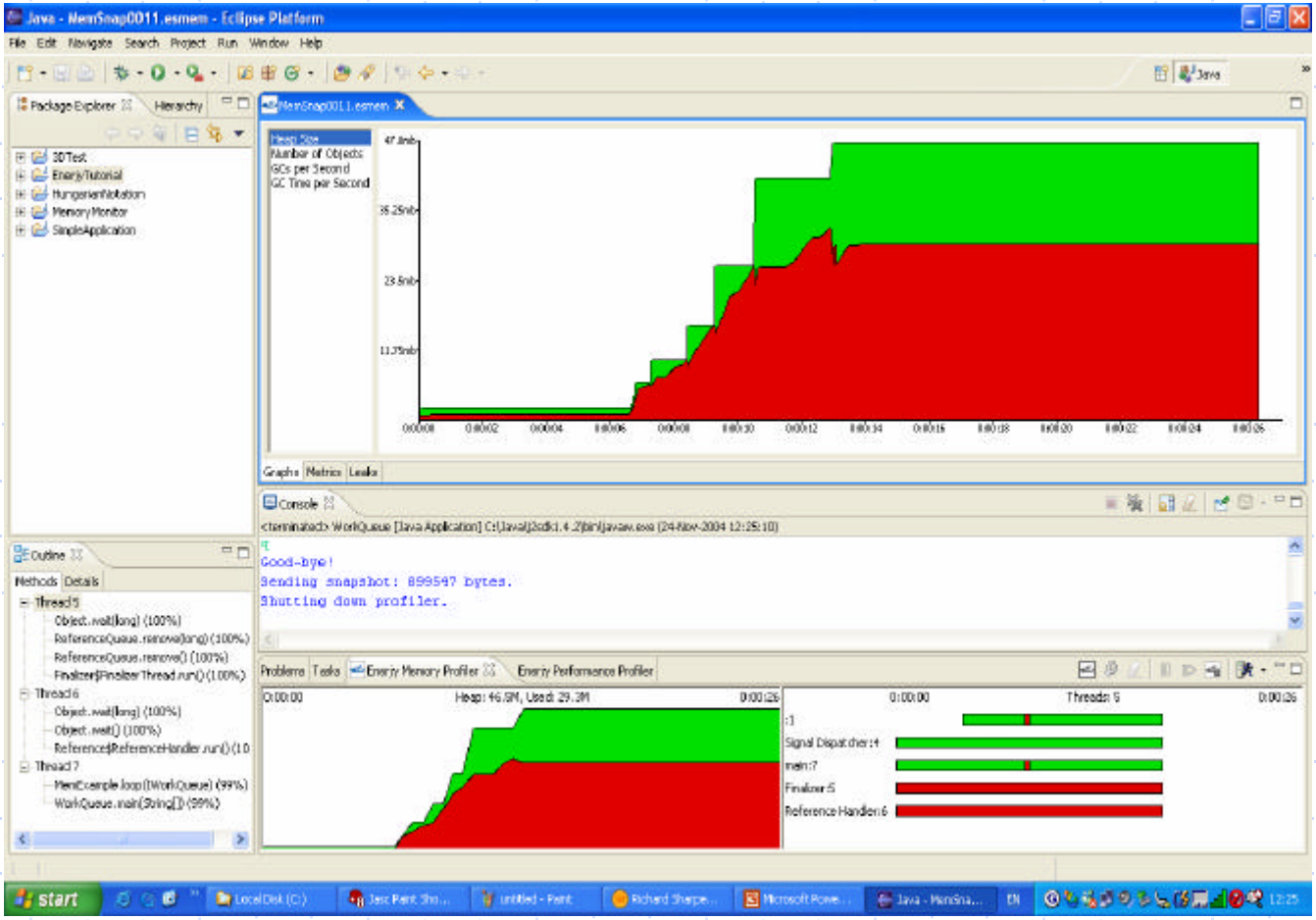
# Profiling Your Code

- Configuring the JVM does not address the issue of Memory Leakage.
  - Neither does simply adding more memory!
- Certain bottlenecks can simply be avoided
  - Logging
  - Disk access
- Use short lived objects and more efficient data structures.

# Arm Yourself With the Right Tools

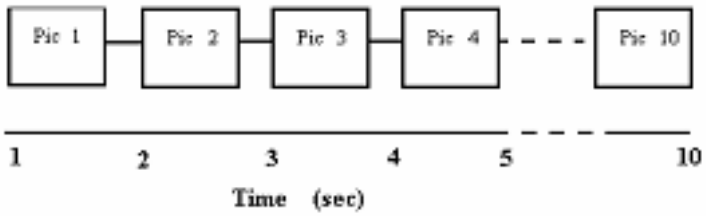
- Use the tools available to profile your applications
- Wide range available:
  - Memory, CPU, Thread, Load Testing, Network optimisation, Load balancing, Complete end to end
- Inherent problems with these tools:
  - How do we know an application is taking longer than it should?
  - How do we fix these problems?
- Save Time & Money!

# Profiling Memory Issues

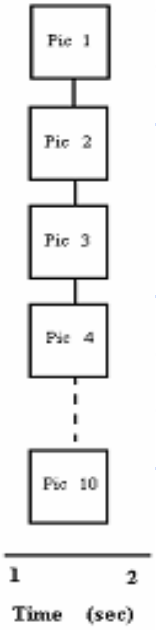


# Does Threading Solve These Issues?

Loading pictures in a web page.




- Pictures loading in a single threaded app.




- Pictures loading in a multi-threaded app.

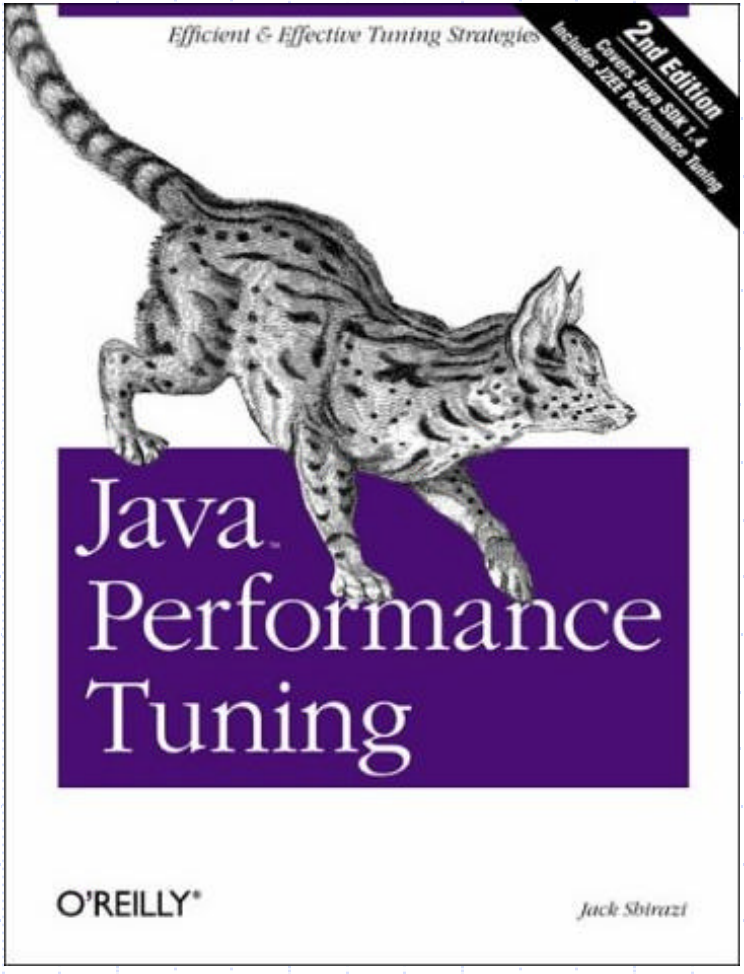
# Does Threading Solve These Issues?

- Used correctly threading should increase performance
- Profile the threads
  - For issues such as race conditions and deadlocks
- Use look ahead algorithms for performance increase
- Use realistic loads

# J2EE Applications

- Try different Application Servers
- More difficult to profile as applications can be running over multiple VM's
- Memory leaks are one of the primary causes of J2EE performance problems
- Load test, load test, load test!
- Use best technology for the job

# Next Steps – Performance Tuning




# Final Thoughts

- Unlikely that the first version will be the optimal version
- Budget for performance issues
- Define the benchmark for success
- Always get your test system as close as possible to the production system
- Use the tools available