© Copyright 2003, Quest Software

Measuring J2EE Application Performance in Production

Brad Micklea

Principal Systems Consultant

Quest Software

brad.micklea@quest.com

© Copyright 2003, Quest Software

Presentation Goal

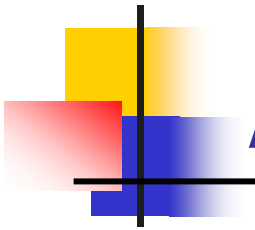
Understand the impact of J2EE performance measurement techniques on the system under test, and how the resulting metrics should be used.

© Copyright 2003, Quest Software

About the Speaker

- Brad Micklea is a Principal System Consultant at Quest Software
- Three years experience in Java and J2EE performance profiling and tuning
- Presented talks at Colorado Software Summit 2002 and JavaOne 2003
- Worked with many Fortune 500 companies to improve their J2EE performance processes

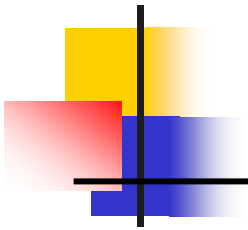
© Copyright 2003, Quest Software



Agenda

- J2EE Performance Measurements
- Measurement Variables
- Measurement Techniques
- Instrumentation Techniques
- Tools

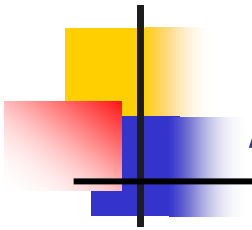
© Copyright 2003, Quest Software



Understand the Cost

All measurements have a cost. Unless you understand the cost, you don't understand the measurement.

© Copyright 2003, Quest Software



Agenda

- J2EE Performance Measurements
- Measurement Variables
- Measurement Techniques
- Instrumentation Techniques
- Tools

© Copyright 2003, Quest Software

Performance Measurements

- Too many people measure without thinking
  - Turn it all on, see what I get
  - Production != pre-production
- Nothing is free
  - You should know what everything costs
- What does the measurement mean?
  - Interpretation is difficult
    - Average *vs*. min/max
    - Servlet response time *vs*. request response time

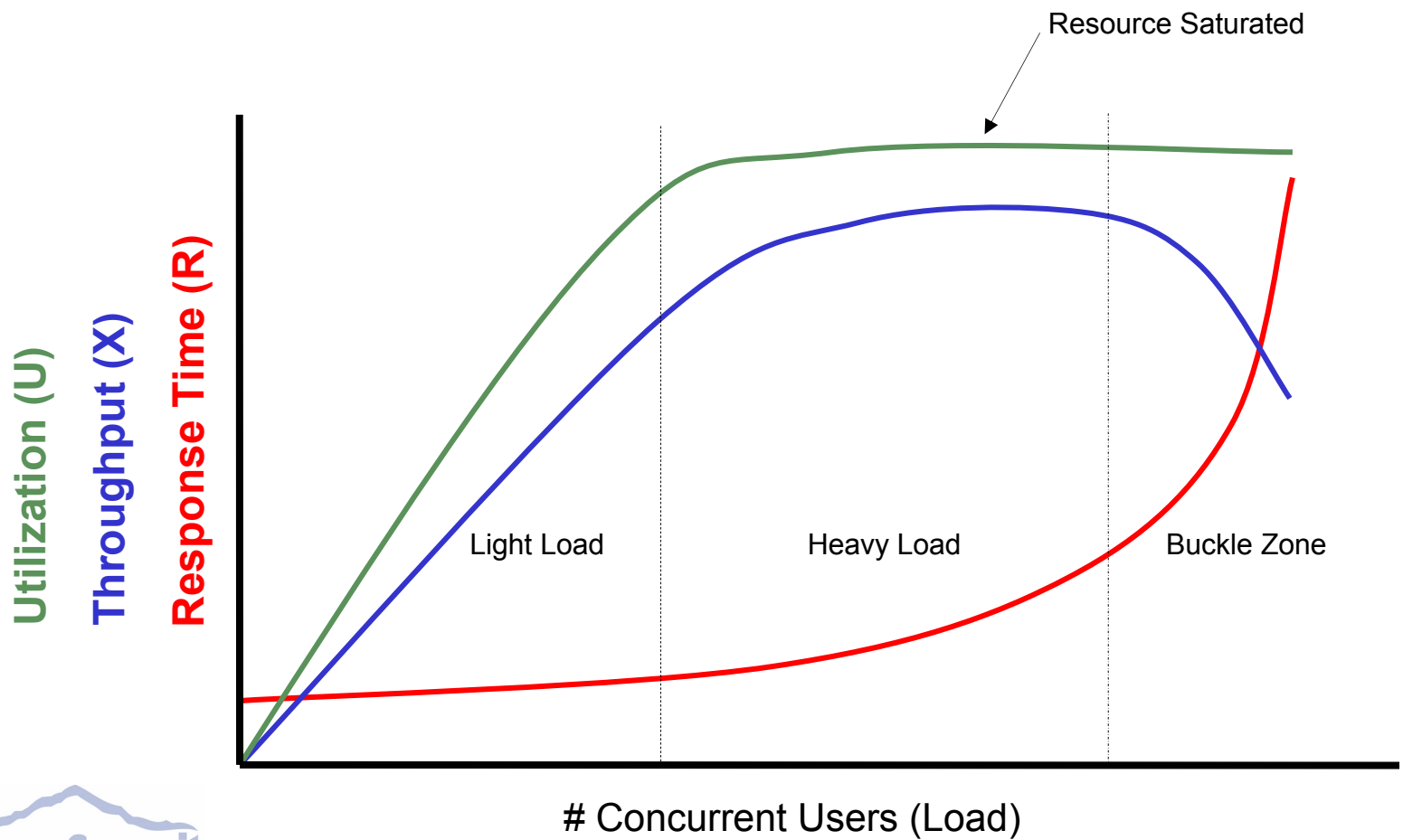
© Copyright 2003, Quest Software

Basic Metrics

- Response Time (R)
- Throughput (X)
- Resource Utilization (U)
- Related to one another
  - R tends to increase with load
  - X and U increase linearly until U is maxed out
  - Once U is maximum, X plateaus or decreases, R climbs exponentially

© Copyright 2003, Quest Software

Basic Metric Behaviour



© Copyright 2003, Quest Software

Response Time

- Response Time (R)
  - Measures the time spent executing response to a request
  - Good for understanding end user experience
  - Can vary significantly
    - Locking, resource contention, container activity
- Measure a distribution of response times
  - Standard deviation
  - Histogram (buckets)
  - Most users get 2s, but 20% get 10s
    - Who's going to call? ☺

© Copyright 2003, Quest Software

Throughput

- Throughput (X)
  - Measures the number of transactions that are executed by the system over a period of time
    - *e.g.* 1200 TPS
  - A measure of the system's capacity for load
    - Not typically a user measurement
    - Useful in non-interactive systems
- X and R can be at odds
  - In the lab, want highest X
  - In production, want lowest R
  - Common goal:
    - Maximize X with 95% of requests <= some value of R

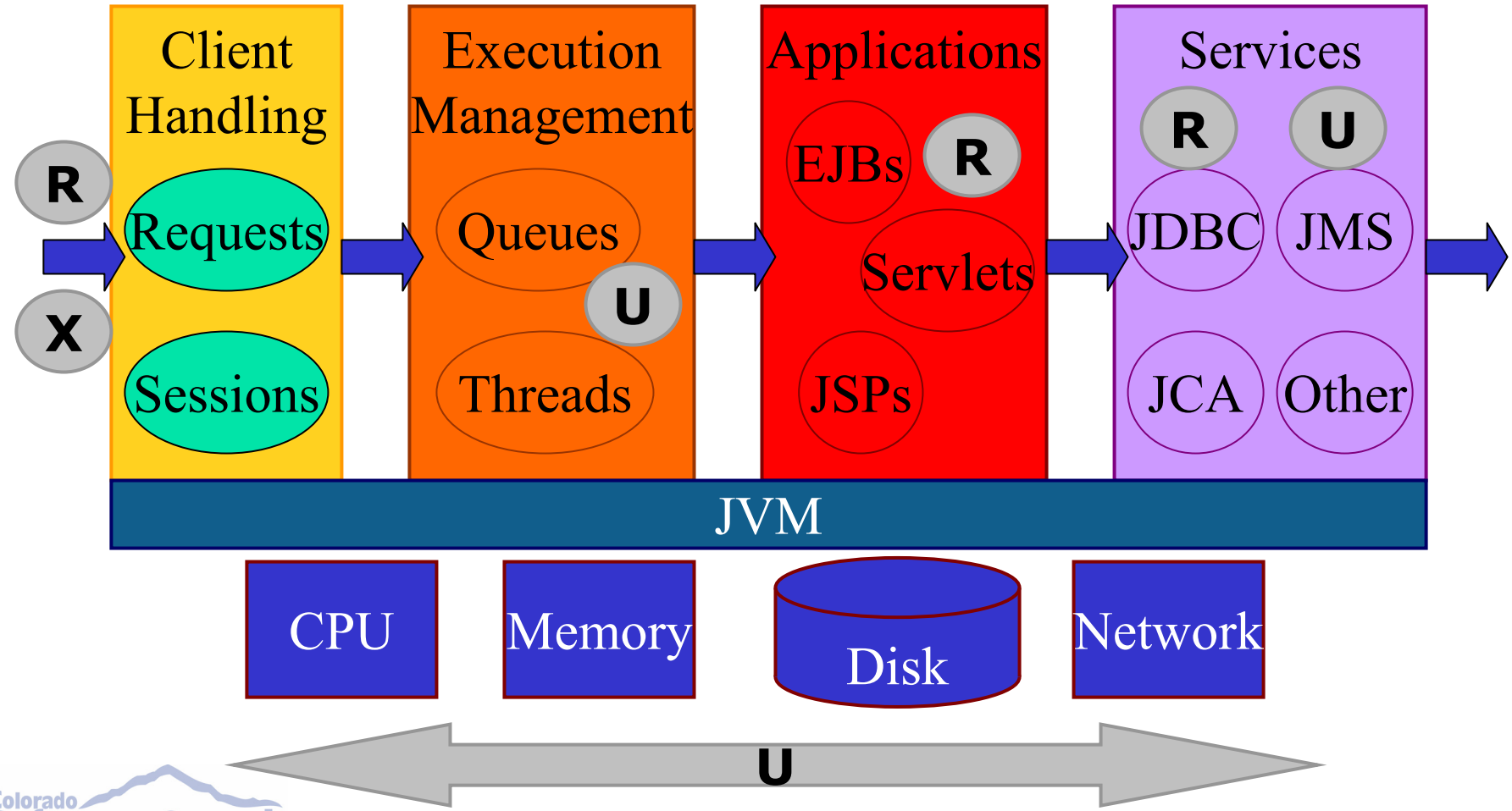
© Copyright 2003, Quest Software

Resource Utilization

- Resource Utilization (U)
  - Measures use of a resource
    - Memory, disk, network, CPU
  - Translates application performance to the lowest level
  - Helpful for sizing current system and planning future capacity requirements
- U is the easiest measurement to understand
  - System requires 56% CPU and 256Mb memory

© Copyright 2003, Quest Software

A Model for J2EE Systems



© Copyright 2003, Quest Software

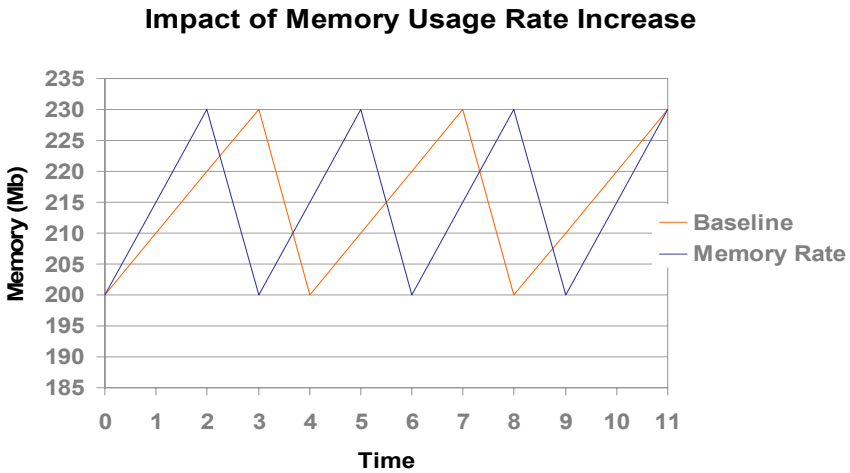
Overhead of Measurements

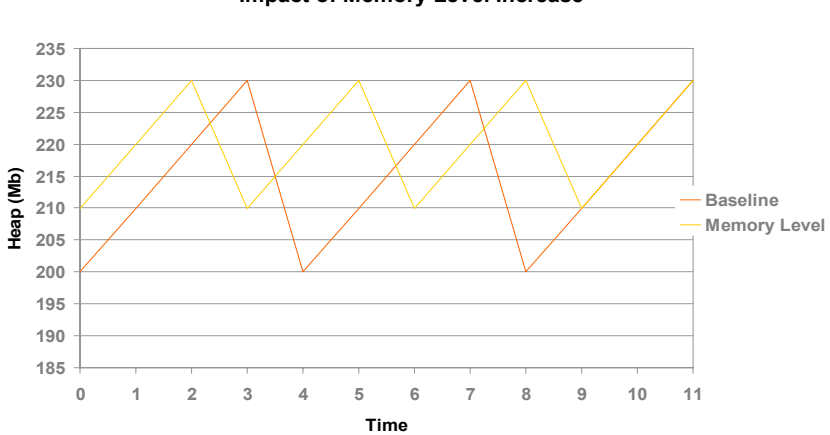
- Measurements aren't free
  - Overhead is the added cost imposed by a measurement
  - Can be in terms of R, X or U
- Service Demand
  - D = U / X
    - Utilization normalized for throughput change
- Best overhead measurement is D
  - Second best is R with standard deviation

© Copyright 2003, Quest Software

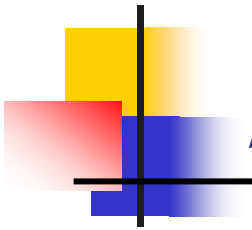
Overhead of Measurements

- Memory impact
  - Memory rate
    - More frequent garbage collection
  - Memory level
    - More frequent garbage collection
    - Can be worked around by adjusting heap size
- GC Impact
  - Frequency
    - Chews up system resources
  - Size
    - Disruptive to response times





© Copyright 2003, Quest Software



Agenda

- J2EE Performance Measurements
- Measurement Variables
- Measurement Techniques
- Instrumentation Techniques
- Tools

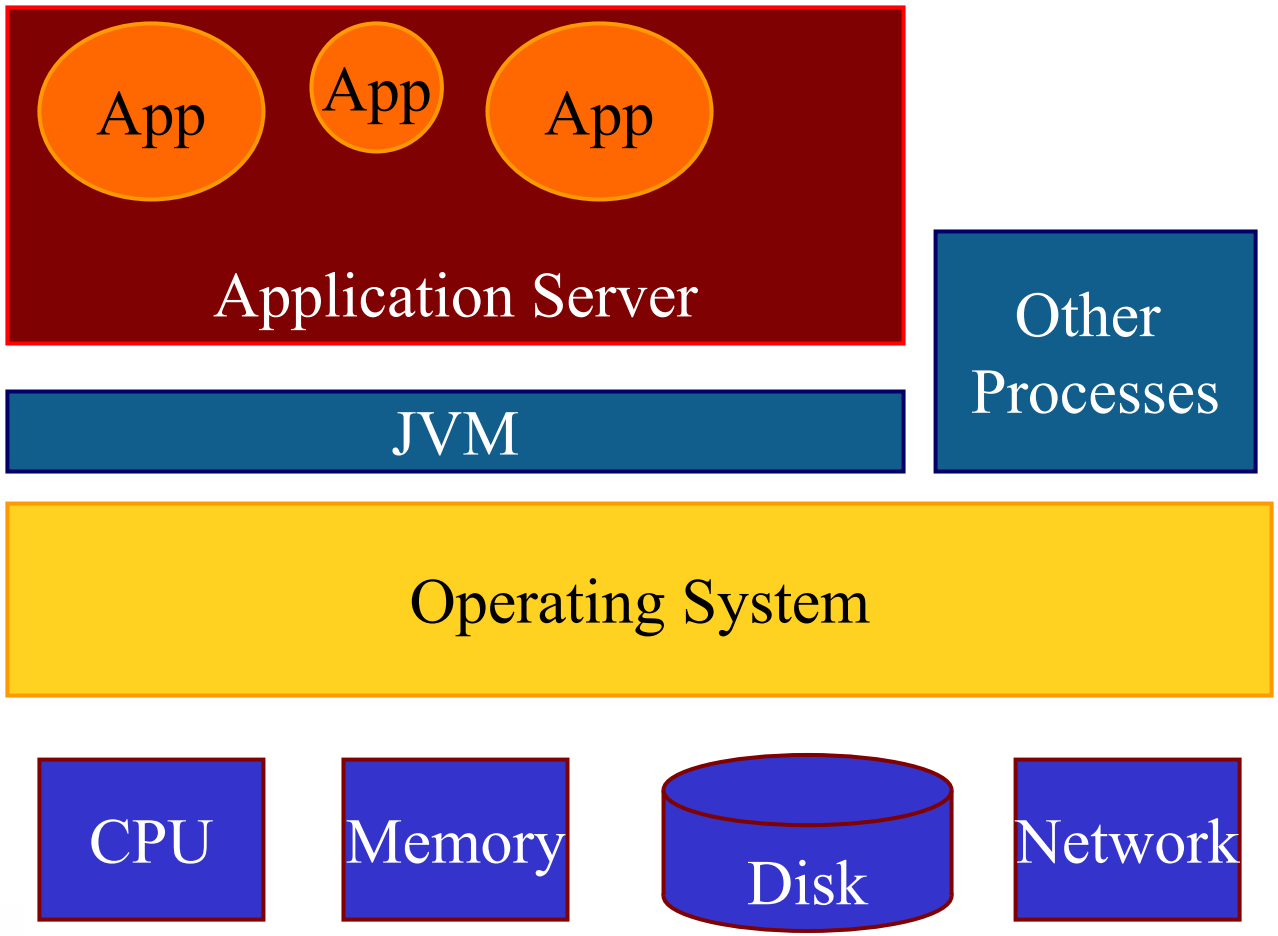
© Copyright 2003, Quest Software

Performance Measurement Variables

- Important to understand the variables that affect performance measurement
  - Mapping from pre-production to production
  - Mapping between production systems
- Many variables affect performance
- Difficult to map between systems and setups
  - You think you know, but you don't
  - Measure, measure, measure

© Copyright 2003, Quest Software

What Are the Variables?



© Copyright 2003, Quest Software

Handling Variables

- Lock down what you can
  - Try to change only one thing between measurements
  - Redo baseline measurements
- Create a measurement landscape
  - Like shining a flashlight in a dark room
    - Can't turn on the light
    - Shine flashlight in many directions and you'll get the picture

© Copyright 2003, Quest Software

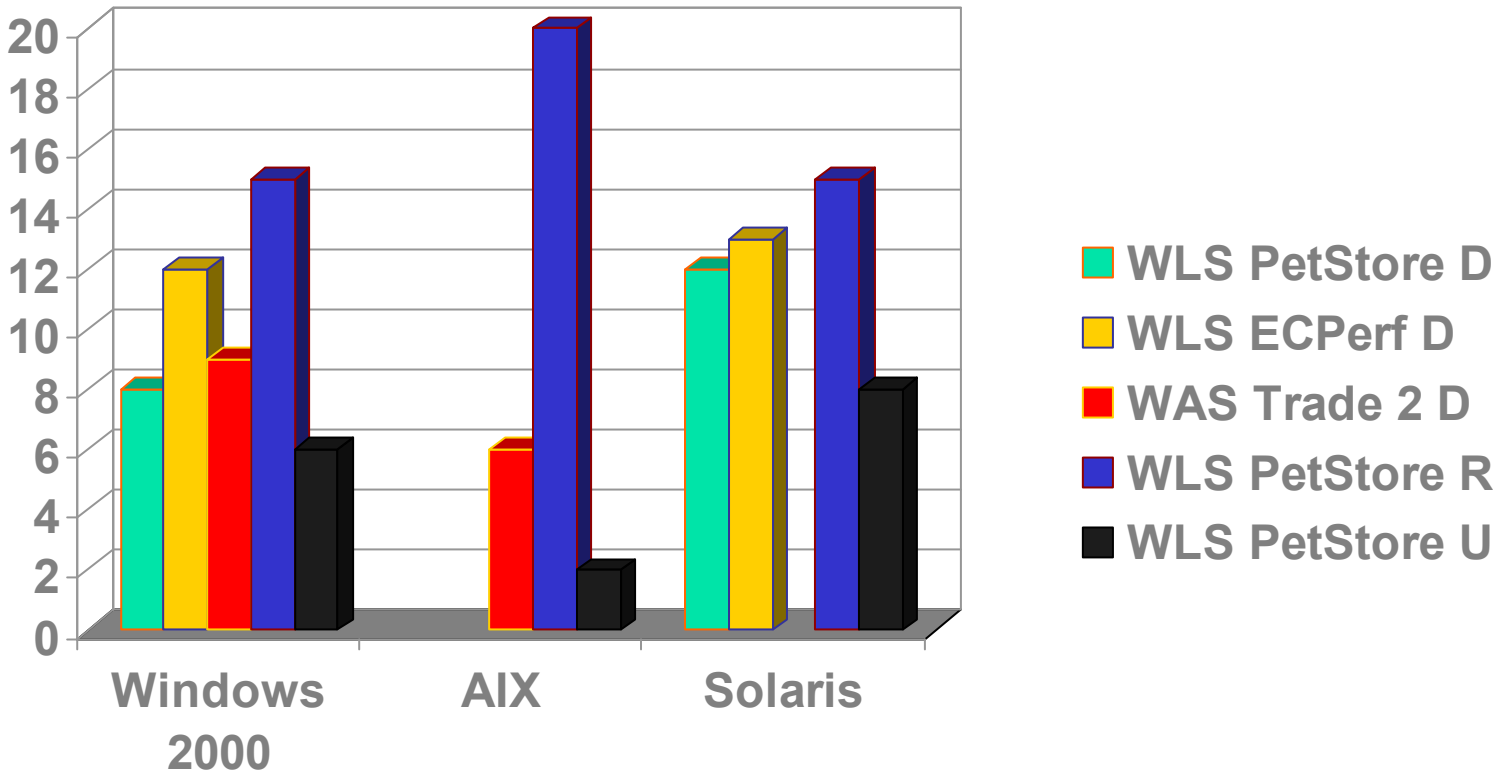
Example: Overhead Measurements

- Project: measure the overhead of a J2EE performance measuring tool
  - Trying to map to production result using pre-production systems
  - PetStore *vs.* ECPerf *vs.* Trade 2
  - WebLogic *vs.* WebSphere
  - AIX 4.3.3 *vs.* Solaris 2.8 vs. Win2K
  - R *vs.* U *vs.* D

© Copyright 2003, Quest Software

Example: Overhead Measurements

Data "Landscape"



© Copyright 2003, Quest Software

Agenda

- J2EE Performance Measurements
- Measurement Variables
- Measurement Techniques
- Instrumentation Techniques
- Tools

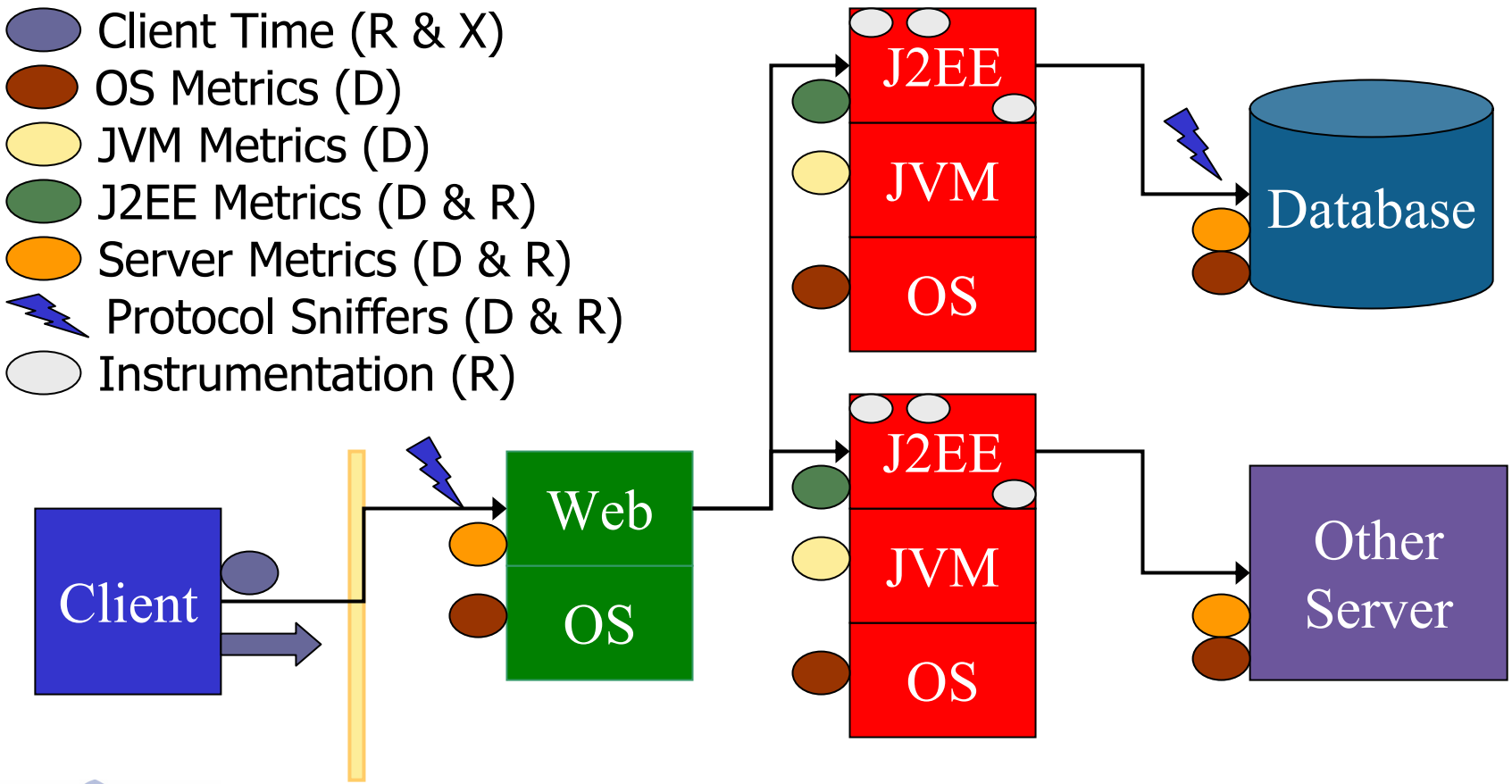
© Copyright 2003, Quest Software

Measurement Techniques

- What measurement techniques are available?
- J2EE systems are transparent
  - Can get deep measurements in particular context
    - Utilization on an application's connection pool
  - Many require no changes to System Under Test (SUT)
- Outline techniques
  - Value, drawbacks, impact, bottom line
  - Understand which techniques will help you

© Copyright 2003, Quest Software

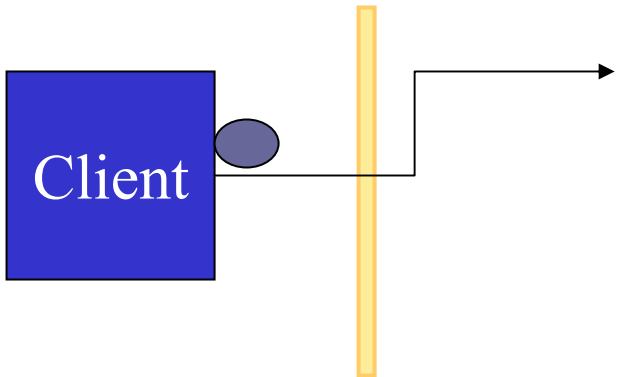
Measurement Points



© Copyright 2003, Quest Software

Client Response Time

- Client Response Time
  - Measures true client response
  - Requires modified browser



© Copyright 2003, Quest Software

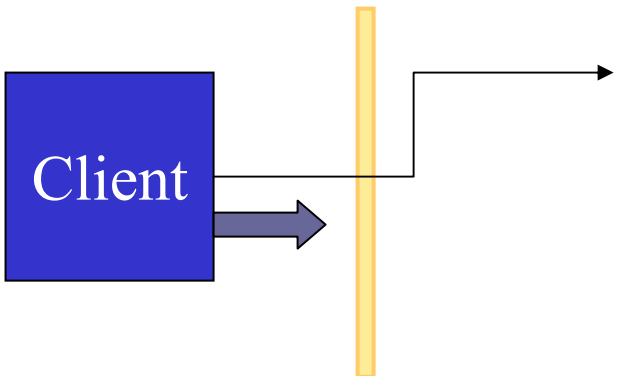
Client Response Time

- Value:
  - Best measure of user experience
- Drawbacks:
  - Difficult to sample all clients
- Impact:
  - Slight impact on some clients
  - Deployment, maintenance

© Copyright 2003, Quest Software

Synthetic Transactions

- Synthetic Transactions
  - Inject transactions for measurement
  - Injection point measures response
  - Could be used to trigger other measurement activities
- Value:
  - Measure user experience without inconveniencing user
  - Control when measurement occurs
  - Can represent compound transactions



© Copyright 2003, Quest Software

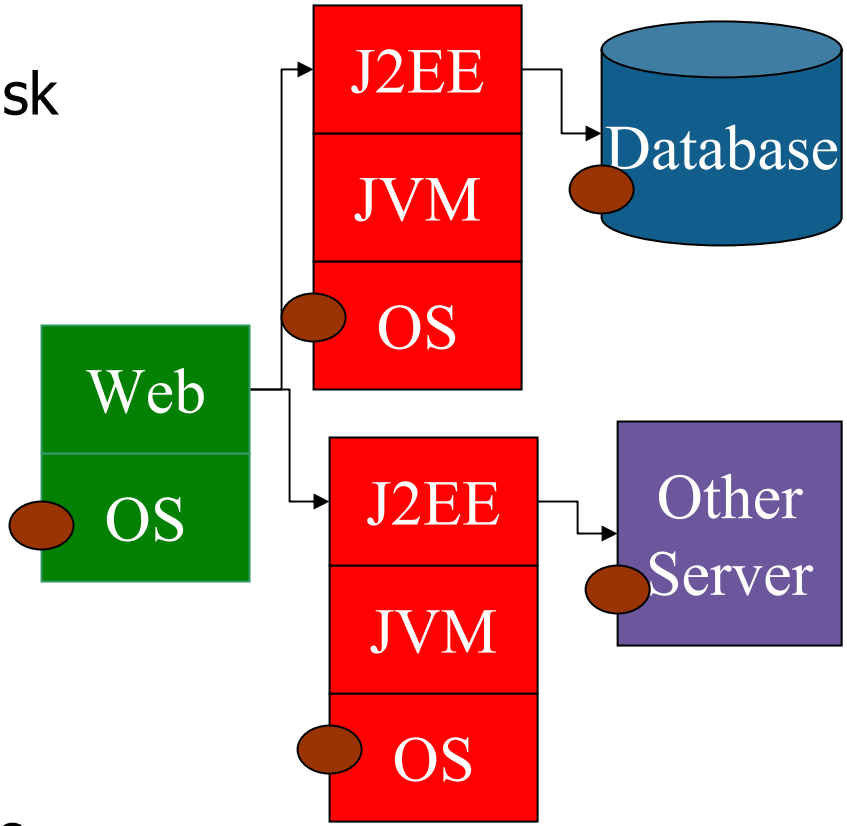
Synthetic Transactions

- Drawbacks:
  - Synthetic - not actual user experience
  - May need to rollback
  - Difficult to include all of the network
- Impact:
  - Extra load on system
- Bottom line:
  - Better version of client response time for a small number of measurements
  - Make sure service request ratios match real-world activity

© Copyright 2003, Quest Software

Operating System Metrics

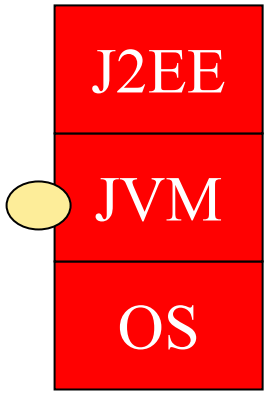
- Operating System Metrics
  - CPU utilization, memory use, disk activity
- Value:
  - General indicators of load
  - Data on all systems
- Drawbacks:
  - Not application specific
- Impact:
  - Typically almost for free
- Bottom line:
  - You'll have OS metrics
  - Correlate with application-specific metrics



© Copyright 2003, Quest Software

JVM Metrics

- JVM Metrics
  - Heap usage
  - Locks, thread call stacks
  - JVMPI
    - Method exclusive time
    - Memory usage
- Available for application servers and some web servers
- Available for stand-alone servers written in Java



© Copyright 2003, Quest Software

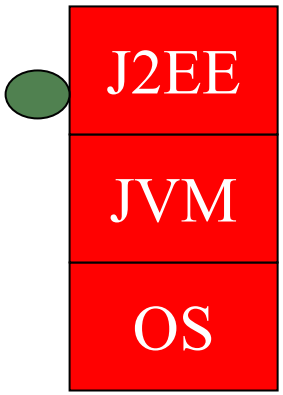
JVM Metrics

- Value:
  - Peek inside JVM operations
  - High-level general indicators for Java code
  - With JVMPI: very detailed information
    - Methods, memory
- Drawbacks:
  - Nothing application specific
  - Requires restart to enable/disable
- Impact:
  - High overhead with JVMPI (depending on what is measured)
  - Cannot turn JVMPI off
    - May change in the future
- Bottom Line:
  - Low-impact JVM metrics often come free with app server metrics

© Copyright 2003, Quest Software

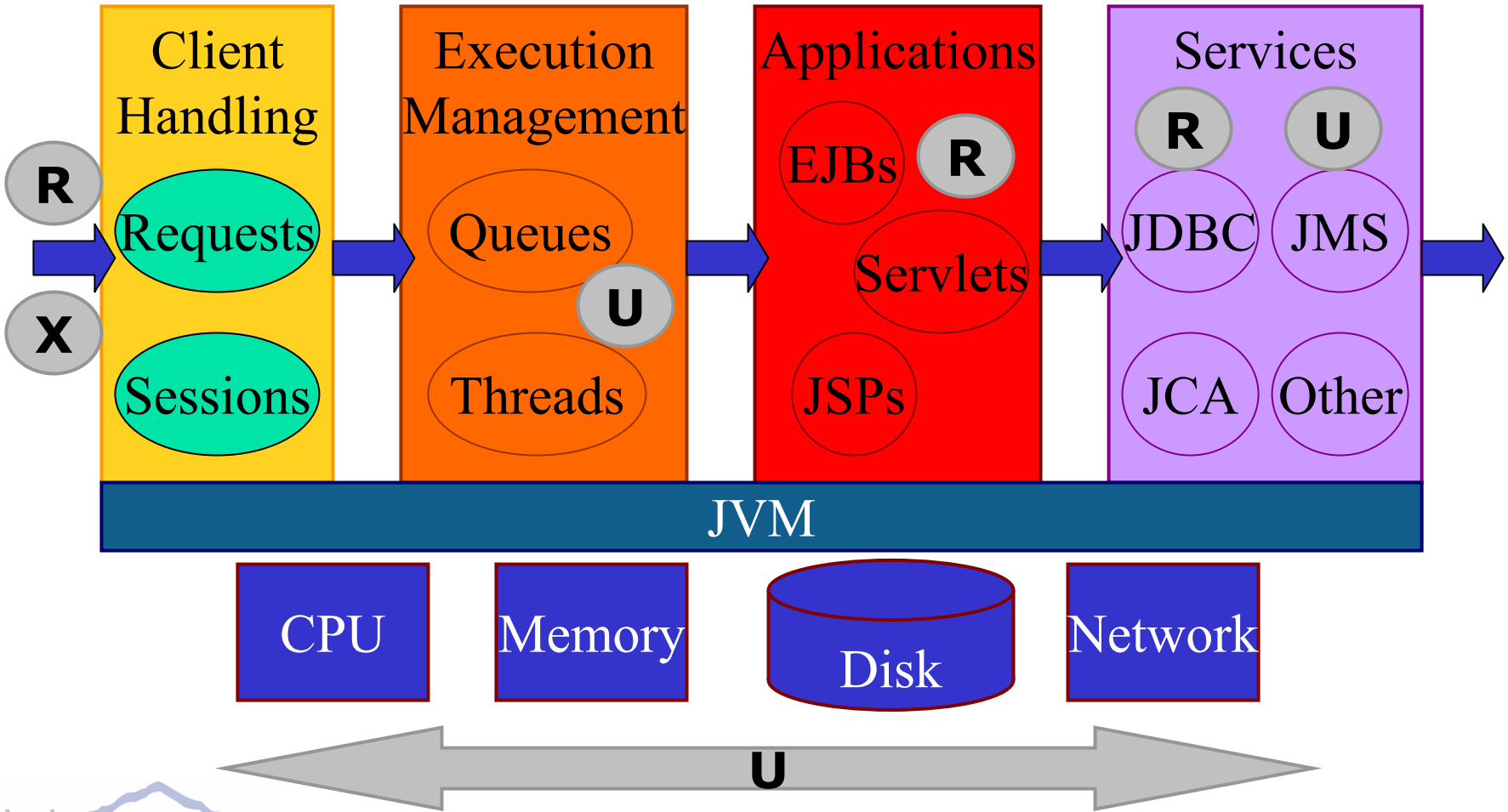
Application Server Metrics

- App Server Metrics
  - Metrics provided by application server
  - Varies by application server
  - Metric Taxonomy
    - J2EE Components
      - ✓ Servlets, JSPs, EJB Utilization
      - ✓ Response times
    - Services
      - ✓ JDBC, JMS, JCA, JNDI Utilization
    - Transactions
    - Threading/queueing
    - General
      - ✓ Configuration
      - ✓ JVM
      - ✓ Web
  - Metric access varies
    - API, utility program, database table, Web console, thick client



© Copyright 2003, Quest Software

Application Server Metrics + Model



© Copyright 2003, Quest Software

Weblogic Metrics

- WebLogic
  - 5.1 exposed a MIB
  - 6.0+ expose metrics as MBeans
  - Always-on
  - Configuration and Runtime MBeans
    - Configuration: deployed apps, settings
      - ✓ If you can do it in the console, you can get it from an MBean
    - Runtime: EJB cache activity, response times
      - ✓ Almost as good as full WebSphere coverage
      - ✓ Better than Oracle (breadth and depth)

© Copyright 2003, Quest Software

Useful Weblogic Metrics

- J2EE Components
  - EJB Caching (per EJB)
    - Accesses, hits, beans, passivations
  - EJB Locking and Pooling
    - Locks, Beans in Use, Idle Beans
  - EJB Home
    - Cached beans for entity, stateful session
    - Cached beans, beans in use, idle beans for stateless session
  - Servlets
    - Average execution time

© Copyright 2003, Quest Software

Useful Weblogic Metrics

- Services
  - JDBC Connection Pools
    - # of connections, prepared statement cache hits and misses, waiters
  - JMS
    - # of connections
- Transactions
    - # of commits, # of rollbacks
- Threading/Queuing
  - Execute Queues
    - # of idle threads, queue length
- General
  - Servers
    - # of open sockets
  - Web Applications
    - # of sessions

© Copyright 2003, Quest Software

Weblogic Metrics Example

```
import weblogic.management.runtime.ServletRuntimeMBean;

public static void retrieveMBeanData() {
Environment env = new Environment();
env.setProviderUrl("t3://rd-geoff:7001");
env.setSecurityPrincipal("system");
env.setSecurityCredentials("********");

try {
    Context ctx = env.getInitialContext();
    MBeanHome       home=(MBeanHome)ctx.lookup(MBeanHome.ADMIN_JNDI_NAME);
    ServletRuntimeMBean poolR =(ServletRuntimeMBean)home.getMBean(new
            WebLogicObjectName(responseTime));
    int eTimeTotal = poolR.getExecutionTimeTotal();
    int countTotal = poolR.getInvocationTotalCount();
  }
  catch (InstanceNotFoundException infe) {
    Logger.getInstance().WARNING("Instance not found");
  }
  catch (MalformedObjectNameException mone) {
    Logger.getInstance().ERROR("Bad??", mone);
  }
}
```

© Copyright 2003, Quest Software

Weblogic Metrics Issues

- Some MBean names get assigned at runtime…

```
private static String responseTime =
"petstore:Location=petstoreServer,Name=petstoreServer_petstoreServer_petstore_com.sun.j2ee.blu
eprints.petstore.control.web.MainServlet_471,ServerRuntime=petstoreServer,Type=ServletRuntime";

[d:/] java -cp d:/bea/wlserver6.1/lib/weblogic.jar weblogic.Admin -username system -password ********
-url t3://rd-geoff:7001 GET -pretty -type ServletRuntime
---
MBeanName:
"petstore:Location=petstoreServer,Name=petstoreServer_petstoreServer_petstore_com.sun.j2ee.blu
eprints.petstore.control.web.MainServlet_471,ServerRuntime=petstoreServer,Type=ServletRuntime"
        CachingDisabled: true
        ContextPath: /estore
        ExecutionTimeAverage: 452
        ExecutionTimeHigh: 3164
        ExecutionTimeLow: 10
        ExecutionTimeTotal: 14020
        InvocationTotalCount: 31
```

© Copyright 2003, Quest Software

WebSphere Metrics

- 3.* had nothing
- 4.* expose metrics using PMI
- 5.* exposes metrics using PMI or JMX (MBeans)
- 5 Levels of instrumentation: None, Low, Medium, High, Maximum
- Can turn instrumentation on/off
  - EJB metrics are very strong
    - ✓ Metrics on all parts of life cycle
    - ✓ Response times
  - Thread utilization, but no queuing
  - Detailed connection pool data

© Copyright 2003, Quest Software

Useful WebSphere Metrics

- J2EE Components
  - Life cycle:
    - Activates, Passivates, Creates, Destroys, Instantiates, Loads, Removes, Stores
    - Concurrent Actives, Concurrent Lives
    - Drains from Pool, Pool Size, Returns Discarded, Returns from Pool, Gets Found, Gets from Pool
    - Average Create Time, Average Drain Size, Average Remove Time
  - Response Times
    - Active methods (per EJB)
    - Method Calls
    - Total Method Calls (per EJB)
    - Method Response Time, Average Method Response Time
  - Servlets
    - Response time per servlet/JSP
    - Concurrent, total requests per servlet/JSP
    - Errors, # of servlets

© Copyright 2003, Quest Software

Useful WebSphere Metrics

- Services
  - Database
    - Allocates, Average Wait Time, Concurrent Waiters, Creates, Destroys, Faults, Percent Maxed, Percent Used, Pool Size,Prepared Statement Cache Discards, Returns
- Transactions
  - Active, begun, committed, rolled back, timed out
- Threading
  - Thread Pools (per pool)
    - Active threads, pool size, created/destroyed
- General
  - Sessions
    - Active, Created, Invalidated, Live Sessions in system
    - Session Life Time
  - JVM
    - Free, Total and Used Memory

© Copyright 2003, Quest Software

JBoss Metrics

- JBoss
  - JMX/MBean backbone
  - Before 3.2, MBeans focused on deployment and configuration information
  - As of 3.2, more runtime performance data
    - Implementing JSR-77
    - Behind other app servers, but catching up

© Copyright 2003, Quest Software

Useful JBoss 3.2 Metrics

- J2EE Components
  - EJBs
    - Cache sizes, passivations
    - Count, average, min, max method timing for all EJB methods
  - Servlets/JSPs
    - Configuration data
    - If running Jetty web server, can get service request times
- Services
  - JMS
    - Message count, subscription count
  - JDBC
    - Connection counts
- Transactions
  - # of commits, # of rollbacks

© Copyright 2003, Quest Software

Oracle Metrics

- Oracle 9iAS
  - Dynamic Monitoring Service
    - Drops data into Oracle database tables
      - ✓ Exportable
  - Metrics on HTTP Server, JVM, JDBC, JSP, EJB
    - Strong on JDBC metrics
    - Basic response times for servlets, JSPs and EJBs
      - ✓ No EJB caching???
    - Nothing on JMS, JNDI

© Copyright 2003, Quest Software

Useful Oracle Metrics

- J2EE Components
  - Servlet (oc4j_servlet) (per servlet)
    - Service time (min, max, average, # active)
  - JSP (oc4j_jsp)
    - Service time (min, max, average, # active)
  - EJB method metrics (oc4j_ejb_method)
    - Service time (min, max, average, # active)
    - One entry per method
    - Wrapper and instance time
- Services
  - JDBC (lots of data, not all shown here)
    - Connection cache (JDBC_DataSource)
      - ✓ Size, hits, misses, free slots
    - Statements (JDBC_Statement)
      - ✓ Execute time for specific SQL queries

© Copyright 2003, Quest Software

Useful Oracle Metrics

- General
  - Web
    - HTTP Server Metrics (ohs_server)
    - HTTP request times (avg, min, max)
    - Web Module (oc4j_web_module) (per module)
      - ✓ Processing time for requests in a module
    - Web Context (oc4j_context)
      - ✓ # of active sessions, average session lifetime
  - JVM (JVM)
    - Free and total memory (value, min, max)
    - # of threads

© Copyright 2003, Quest Software

Application Server Metrics

- Value:
  - Under-the-covers look at J2EE
  - Covers R, U, X in all parts of the model
    - J2EE components, Services, Transactions, Threading, Web, JVM
  - Component-specific measurements
  - Different collection techniques
    - Utility program, console, API
- Drawbacks:
  - Lots of data, hard to navigate
  - Data not tied to application or to individual requests
  - Some app servers provide little
  - No standards (yet)
- Impact:
  - Cost can vary from free to significant

© Copyright 2003, Quest Software

Cost of App Server Metrics

- Weblogic
  - Can't turn off MBean data collection
  - Application server memory use for remote MBean queries is higher
- WebSphere
  - All PMI collection levels are roughly equivalent
  - Memory is the main overhead
    - Low CPU overhead with sufficient memory
    - High CPU overhead with insufficient memory

© Copyright 2003, Quest Software

Application Server Metrics

- Bottom Line:
  - Great value in the data
    - Reasonable coverage on latest app servers
    - Best way to get JVM data
    - Trend is for this data to get better
  - Multiple access points
    - Standards would be nice
  - Must understand the cost of these metrics
    - Don't necessarily come for free

© Copyright 2003, Quest Software

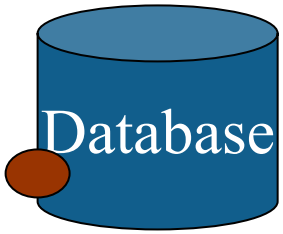
What Is JSR-77?

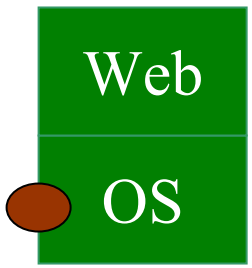
- JSR-77 Management Specification
  - Defines a model for management
    - J2EE Management Model
      - ✓ Applies to configuration and performance
  - Standard APIs to the model
    - JMX/MBean, CIM, SNMP MIB
- This will standardize the metric data in J2EE systems
- http://www.javaworld.com/javaworld/jw-06-2002/jw-0614-mgmt.html

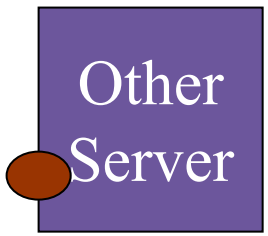
© Copyright 2003, Quest Software

Other Server Metrics

- Other servers have metrics
  - Database
  - Web Server
  - Messaging Server (MQSeries)
- SNMP/MIB
- Treat like OS metrics
- Correlate to J2EE metrics
  - *e.g.* JDBC execute time with SQL query time from database



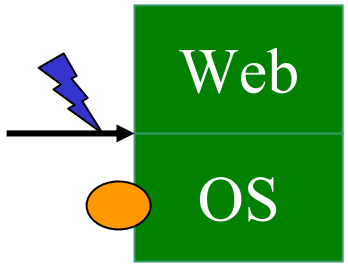


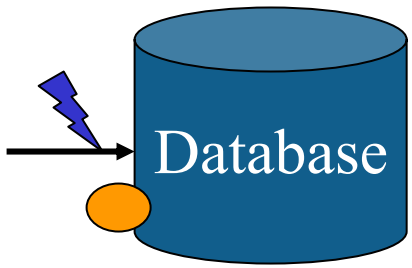


© Copyright 2003, Quest Software

Protocol Sniffers

- Protocol Sniffers
  - Understand protocols by analyzing traffic over the wire
  - Can derive metrics by reassembling transactions
  - Web (TCP/IP, HTTP)
  - SQL





© Copyright 2003, Quest Software

Protocol Sniffers

- Value:
  - Touchless installation with little overhead
- Drawbacks:
  - Deep dive – hard to derive application context
  - Hard problem to solve, edge cases require tuning
  - You're going to have to spend money
- Impact:
  - Almost none
- Bottom Line:
  - Valuable if you can afford them

© Copyright 2003, Quest Software

Log Analysis

- Log Analysis
  - Search log file for particular events
    - Exceptions, user-generated messages
    - Count occurrences
  - Derive measurements from events
    - Method enter/exit in Servlet.doGet()
    - Time from start of transaction to commit
  - Real-time or post-processing
  - Correlate logs from different JVMs

© Copyright 2003, Quest Software

Log Analysis

- Value:
  - Customized results
- Drawbacks:
  - Requires coordinated effort between dev and measurement
  - Need a regexp-based log file processing tool
- Impact:
  - Log messages add overhead to code
  - Uses CPU and memory, creates contention for log file
- Bottom line:
  - Handy post-processing technique
  - Good way to measure method timings for a small number of critical methods
  - For a large number of methods, use an instrumentation-based solution
    - Overhead, scalability

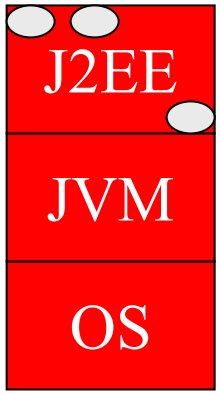
© Copyright 2003, Quest Software

Agenda

- J2EE Performance Measurements
- Measurement Variables
- Measurement Techniques
- Instrumentation Techniques
- Tools

© Copyright 2003, Quest Software

Inserted Instrumentation



- Extra code inserted into application to measure performance
  - Custom: inserted by developer
  - Automatic: inserted by a tool
    - Nature of J2EE allows inserted instrumentation
    - Byte code insertion
    - Adds to core app server measurements
      - ✓ For older app servers, BCI used to gather app server metrics
- Resident for lifetime of class

© Copyright 2003, Quest Software

Custom Instrumentation

- Custom instrumentation
  - Developer inserts custom measurement code
    - ARM, manufacturer
  - Send results to a central recording mechanism
    - Which you also have to write…

```
void myMethod() {
    long start = System.currentTimeMillis();
    for (int i = 0; i < 10; i ++) {
        System.out.println("Wasting time " + i);
    }
    long end = System.currentTimeMillis();
    Recorder.getRecorder().addMethodTime("test.myMethod", end-start);
}
```

© Copyright 2003, Quest Software

Custom Instrumentation

- Value:
  - Customized results
- Drawbacks:
  - How is the data gathered?
    - Log file, expose as MBean
- Impact:
  - Adds overhead to code
    - More than automatic instrumentation
- Bottom Line:
  - Similar to log analysis
    - Except log analysis doesn't require central infrastructure
    - Good way to measure method timings for a small number of critical methods
    - Bad for large number of methods
  - Great if API + infrastructure provided by tool vendor
    - Less work for you

© Copyright 2003, Quest Software

Automatic Instrumentation

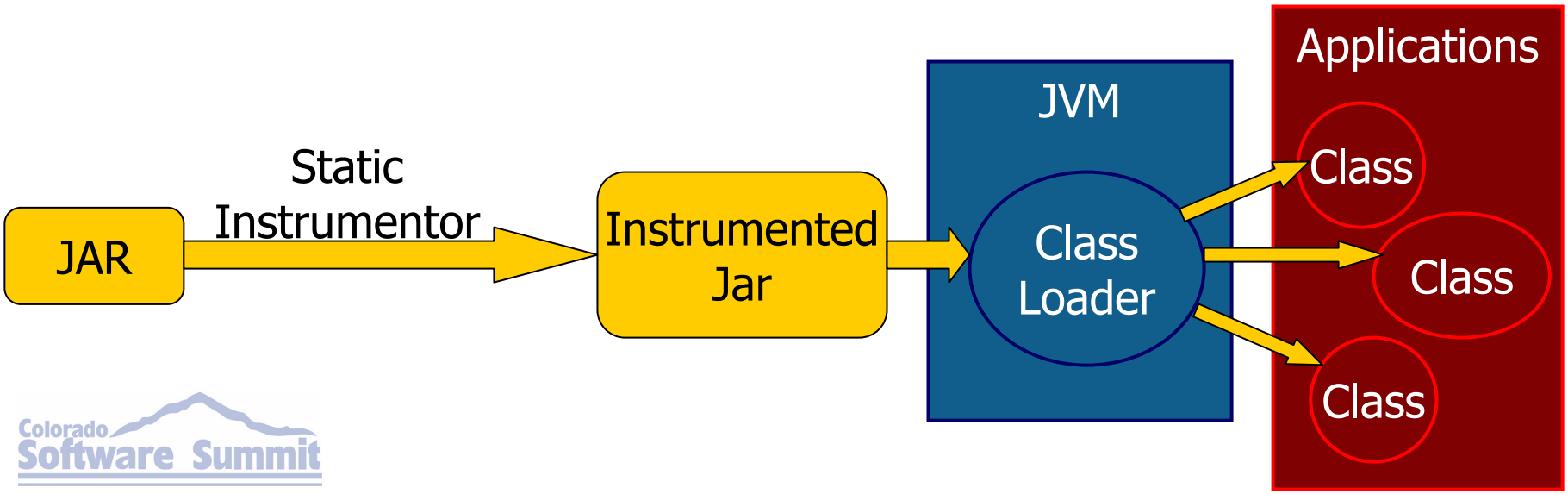
- Automatic instrumentation
  - Measurement code is inserted automatically
  - Sends results to a central recording mechanism

```
void myMethod() {
    if (recording) {
        Recorder.getRecorder().enterCallback(this);
    }
    for (int i = 0; i < 10; i ++) {
        System.out.println("Wasting time " + i);
    }
    Recorder.getRecorder().exitCallback(this);
}
```

© Copyright 2003, Quest Software

Pre-Instrumentation

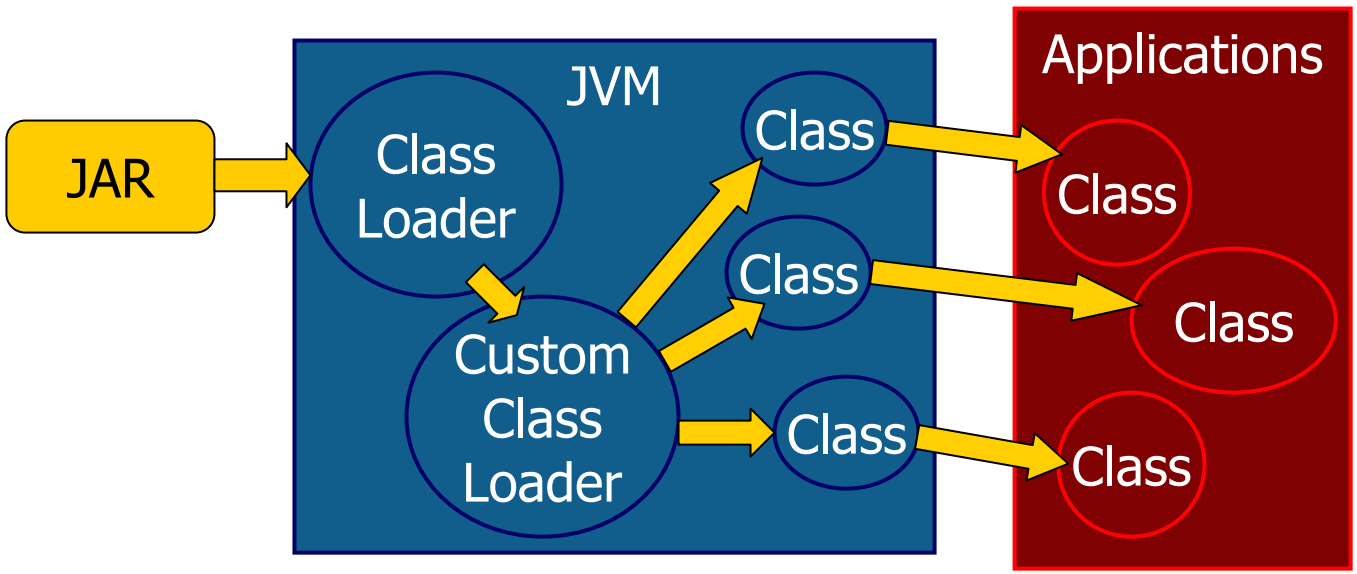
- Pre-instrumentation of JAR files
  - Instrumentation added statically
  - No runtime impact to instrument
  - Must know in advance what to instrument



© Copyright 2003, Quest Software

Class Loader Instrumentation

- Class loader hooks
  - Use custom class loader to modify code
    - Class loader hooks



© Copyright 2003, Quest Software

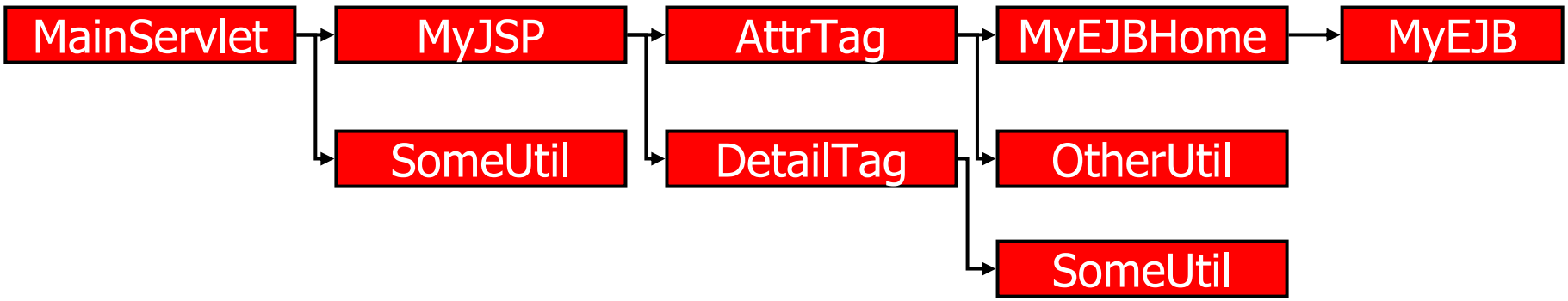
Instrumentation Data

- Variety of data can be gathered
  - Call counts
  - Method exclusive time
    - Time spent in the method
  - Method cumulative time
    - Time spent in the method and all methods it calls
    - Good for SLAs, back-end response times
  - Exceptions thrown
    - Exceptions are caught by instrumentation, then rethrown
  - Bytes transferred/serialized in RMI
  - Stack information
    - Generate call trees after collection (post-processing)
- Averages, min/max, standard deviation

© Copyright 2003, Quest Software

Instrumentation Data: Full Detail

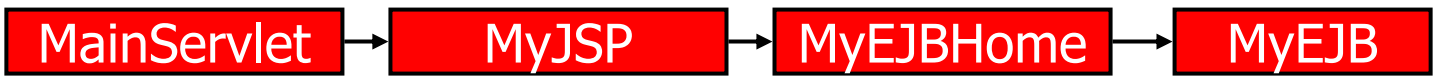
- Full Instrumentation
  - Instrument all methods
  - Filter out application server methods
  - Allow custom filters to reduce method set



© Copyright 2003, Quest Software

Instrumentation Data: Component Detail

- J2EE perimeter
  - Instrument only classes that are J2EE objects
    - Services (JMS, JNDI, JDBC)
    - EJB home and EJB methods
    - Servlet and JSP processing
  - Better for diagnosis



© Copyright 2003, Quest Software

Full Versus Component Detail

Call Tree Height

Service Request Type	Full	Component
Checkout	14	8
Cart	14	7
Commit Order	16	8
Category	9	5
Product	9	5
Product Details	13	7
Averages	**10.64**	**5.64**

© Copyright 2003, Quest Software

Full Versus Component Detail

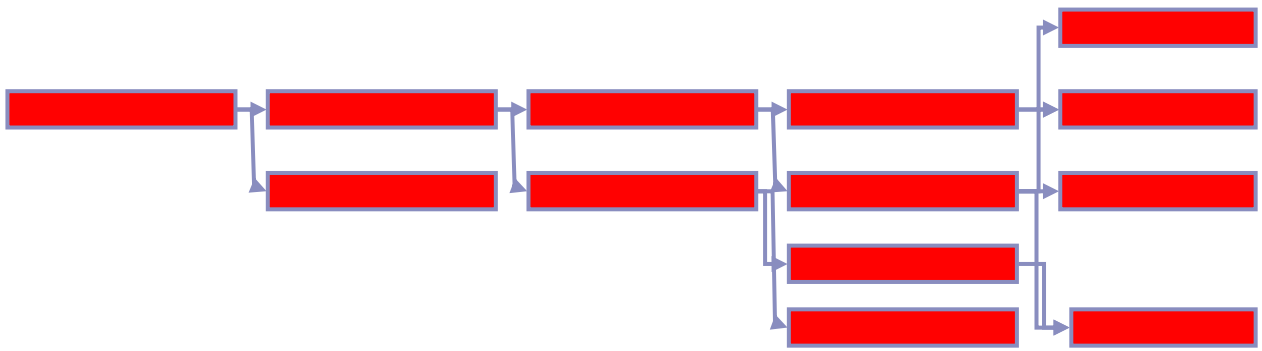
Call Tree Methods

Service Request Type	Full	Component
Checkout	99	30
Cart	165	45
Commit Order	332	114
Category	52	12
Product	69	13
Product Details	95	33
Averages	**110.27**	**33.36**

© Copyright 2003, Quest Software

Service Requests

- Data can be gathered in the context of a service request
  - Method calls reassembled into a call graph
  - Can even track request across system boundaries
- Benefit: can measure time of end user request
  - Or even just groups of calls



© Copyright 2003, Quest Software

Automatic Instrumentation

- Value:
  - Good data for deep diagnosis
    - Method times, exceptions
  - Only mechanism that can measure component response times for a service request
  - Enables application visualization
- Drawbacks:
  - Requires centralized data collection
  - Cannot be removed at runtime
- Impact:
  - Overhead for every instrumented method
  - Infrastructure overhead

© Copyright 2003, Quest Software

Automatic Instrumentation Overhead

- Overhead contributors
  - Deciding whether to record
    - On/off, full/component, sampling
  - Measuring start/stop times
  - Calculating and storing data
    - Object creation is a factor
  - Communicating with collection infrastructure
    - Object locking
    - Disk/network

© Copyright 2003, Quest Software

Automatic Instrumentation: Bottom Line

- Bottom Line:
  - Look for ability to turn collection on/off
    - Flexibility on collection level is a good thing
  - Look for sampling mechanisms to avoid data collection for all calls (minimize overhead)
  - Use for collecting data you don't get from application server metrics
    - Otherwise not worth the overhead
- Ask micro-level and macro-level questions
  - How does it do what it does?
  - What is the impact on a system like mine?

© Copyright 2003, Quest Software

Future Instrumentation Features

- What's next for instrumentation?
  - Trigger data collection based on slow transactions
  - Trigger data collection for a tagged synthetic transaction
  - Many notches on the dial
    - Not collecting, collecting
    - Servlets, JSPs, EJBs, JNDI, JMS, JDBC
  - Smart sampling
    - Dial collection up/down based on load
    - Allow user to choose overhead
  - Instrumentation removal without reboot
  - Data on memory use
    - Collection instrumentation
    - Transaction/memory attribution

© Copyright 2003, Quest Software

Summary of Measurement Techniques

- Looked at a variety of measurement techniques
  - Client Response Times, Synthetic Transactions
  - OS, JVM, App Server Metrics
  - Log Analysis, Protocol Sniffers
  - Inserted Instrumentation
    - ✓ Custom, automatic
- Recommendations?
  - ➢ Choice depends on impact *vs*. quality
    - Low impact: OS metrics, log analysis, sniffers
    - High quality: app server metrics, inserted instrumentation

© Copyright 2003, Quest Software

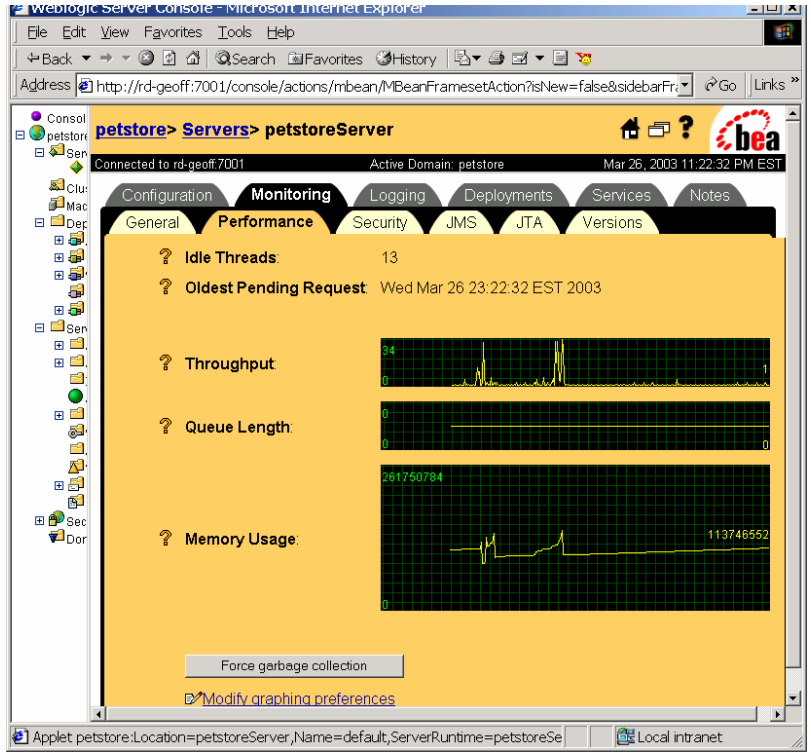
Agenda

- J2EE Performance Measurements
- Measurement Variables
- Measurement Techniques
- Instrumentation Techniques
- Tools

© Copyright 2003, Quest Software

Measurement Tools

- What are the tools you can use for measurement?
  - Application server
    - MBeans/PMI and other metrics
    - Administration console
  - Free
  - Commercial
    - Profilers
      - ✓ Developer-centric
    - Monitoring tools
      - ✓ Ops-centric
    - More vendors than you would think possible



© Copyright 2003, Quest Software

Measurement Tools

- Factors that impact tool decisions
  - Budget
  - Users
    - Production support (DBAs, app server admins)
    - Developers
      - ✓ Forgotten by production support, but who do they call? ☺
  - Application
    - Some apps may not tolerate any overhead
    - Back end systems in use

© Copyright 2003, Quest Software

Measurement Tools: Correlation

- Navigating the metric maze is tough
  - Correlation: Automatically map metrics that may be related
    - *e.g.* EJB response times and EJB passivation curve
  - Find features that may be related
    - *e.g.* rise in execute thread usage corresponds with connection pool usage
- Industry trend: intelligence and advice
  - Data is a commodity

© Copyright 2003, Quest Software

Summary

- All measurements have a cost
  - Including application server metrics
  - Measure cost on your system
- Many measurement techniques available
  - Client Response Times, Synthetic Transactions
  - OS, JVM, App Server Metrics
  - Log Analysis, Protocol Sniffers
  - Inserted Instrumentation
- There is usually a tradeoff between the quality of the measurement and its cost
  - Balancing this depends on your situation
  - Look at free solutions to see if they are sufficient

© Copyright 2003, Quest Software

Related Talks

- Simon Roberts: Performance

© Copyright 2003, Quest Software

Acknowledgements

- Geoff Vona
- Kant Hung
- Pat Stephenson
- Steve Haines
- Dmitri Bourlatchkov
- Ethan Henry
- Ed Lycklama