Pinpointing and Exploiting Specific Performance Bottlenecks

Revised for:

Software Test & Performance Conference

November 2005 New York, NY

First Presented for:

IBM Rational Users Conference, 2004

Scott Barber

Chief Technology Officer

PerfTestPlus, Inc.

Agenda

Introduction

Analyzing Results to Identify Bottlenecks

What the Development Team Needs to Know

Identify Tier of Detected Issue

Identify Component of Detect Issue

Develop Test To Exploit Issue

Available Tools

Examples (Time Permitting)

Want More Information?

Summary/Questions

Introduction

This presentation is adapted from *User Experience, not Metrics*: Parts 6, 8, 9 and10 and *Beyond Performance Testing*: Parts 6, 7, 8, 9, 10 located at http://www-106.ibm.com/developerworks/rational/library/ (RDW) and http://www.perftestplus.com/.

Introduction

One part of the system is always slowest (the bottleneck). Until it is remedied, no other tuning will actually improve the overall performance of the application along that path. Before that bottleneck can be tuned, it must first be conclusively identified.

Once a bottleneck is identified, resolution can be reached more quickly if your existing tests are modified to eliminate distraction from ancillary issues. Pinpointing exactly where the bottleneck is an art all its own.

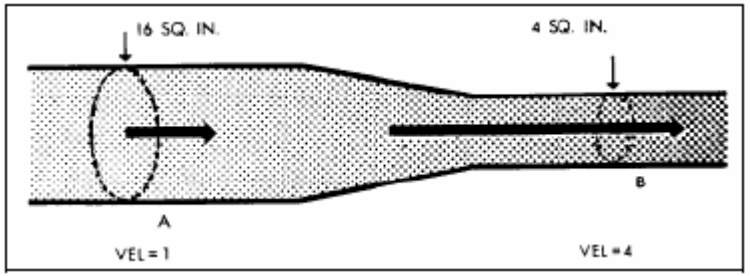
After determining where the bottleneck is architecturally, a new test will likely be needed to exploit it in order to help the development team with tuning. These bottleneck exploiting tests needn't bear any resemblance to real user activity but rather focus on the bottleneck alone. In fact, these tests often don't even interact with the system in ways users could and may include direct interaction with back-end tiers.

Intro – Scott's Rules of Bottlenecks

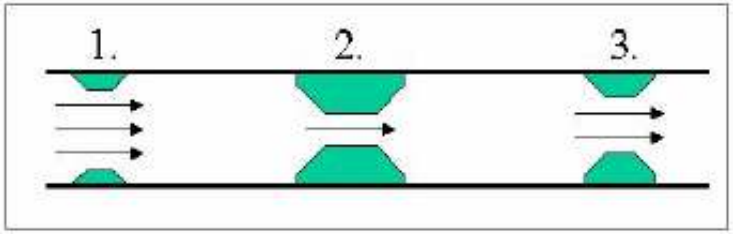
A bottleneck is a slowdown, not a stoppage. A stoppage is a failure.

Bottlenecks don't only exist under load.

The symptoms of the bottleneck are (virtually) never observed at the actual location of the bottleneck.



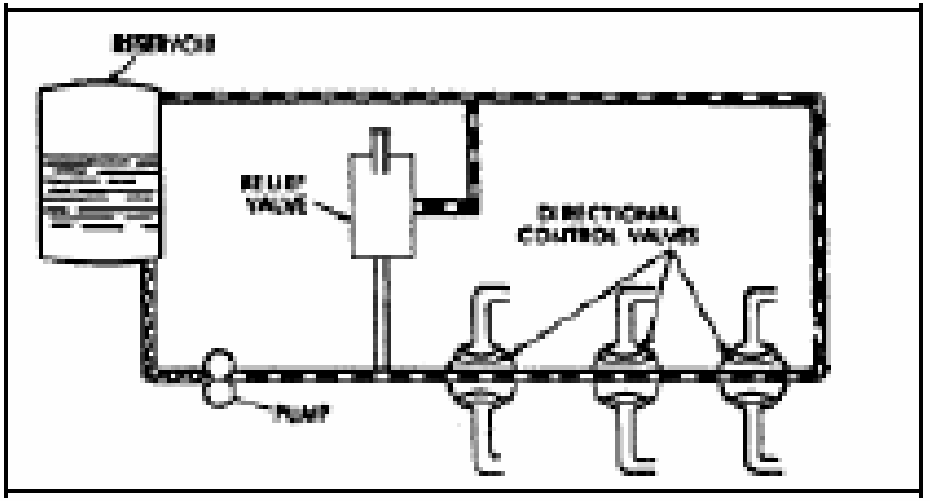
The critical bottleneck is the one bottleneck along a particular user path the removal of which will improve both performance and the ability to find other bottlenecks.



If you have multiple paths through a system and think there's a bottleneck, you should isolate each path and evaluate it separately.

Intro – Scott's Rules of Bottlenecks

The bottleneck is more likely to be found in the hardware than in the network, but the network is easier to check.



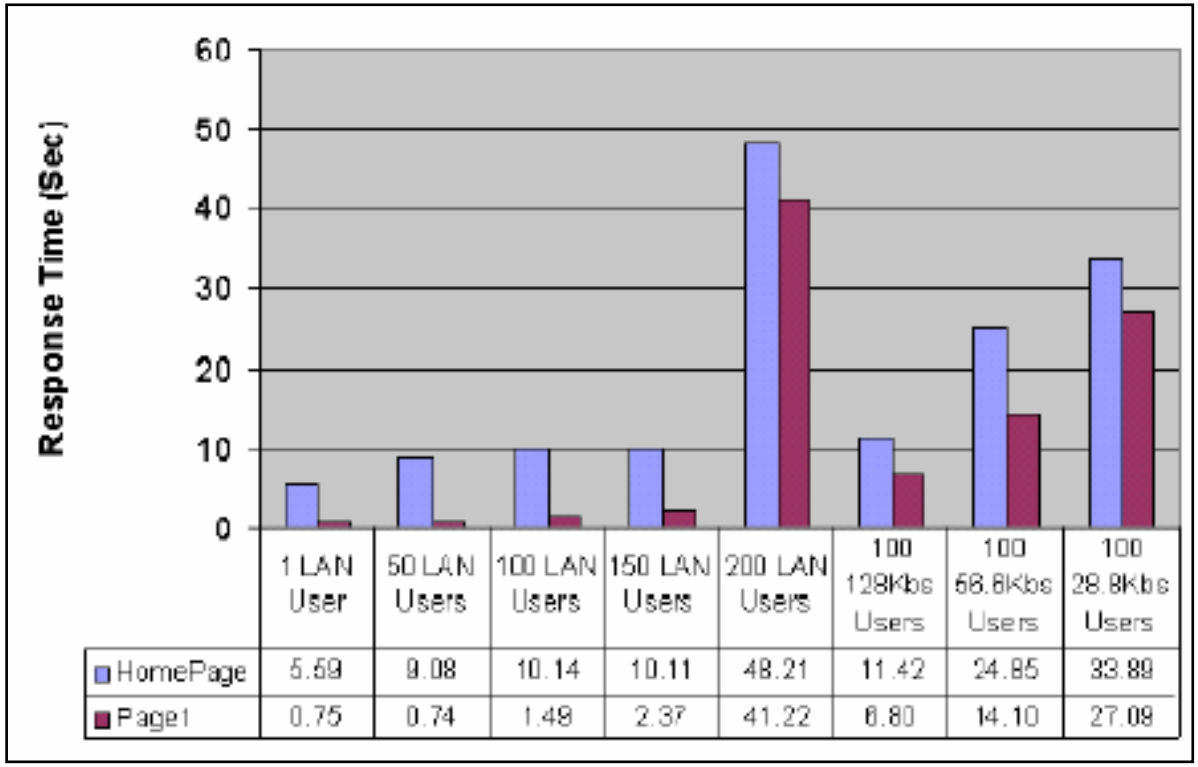
Unless other activities and/or users are affected by the observed slowness or its cause, it's not a bottleneck but a slow spot.

When reporting bottleneck suspects, don't assume you know the cause, just report the symptoms.

Analyzing Results to ID Bottlenecks

Examine Response vs. Time Charts/Tables

- Help identify bottleneck suspects
- Parts 6, 7, 8, and 9 of the “User Experience, Not Metrics” series

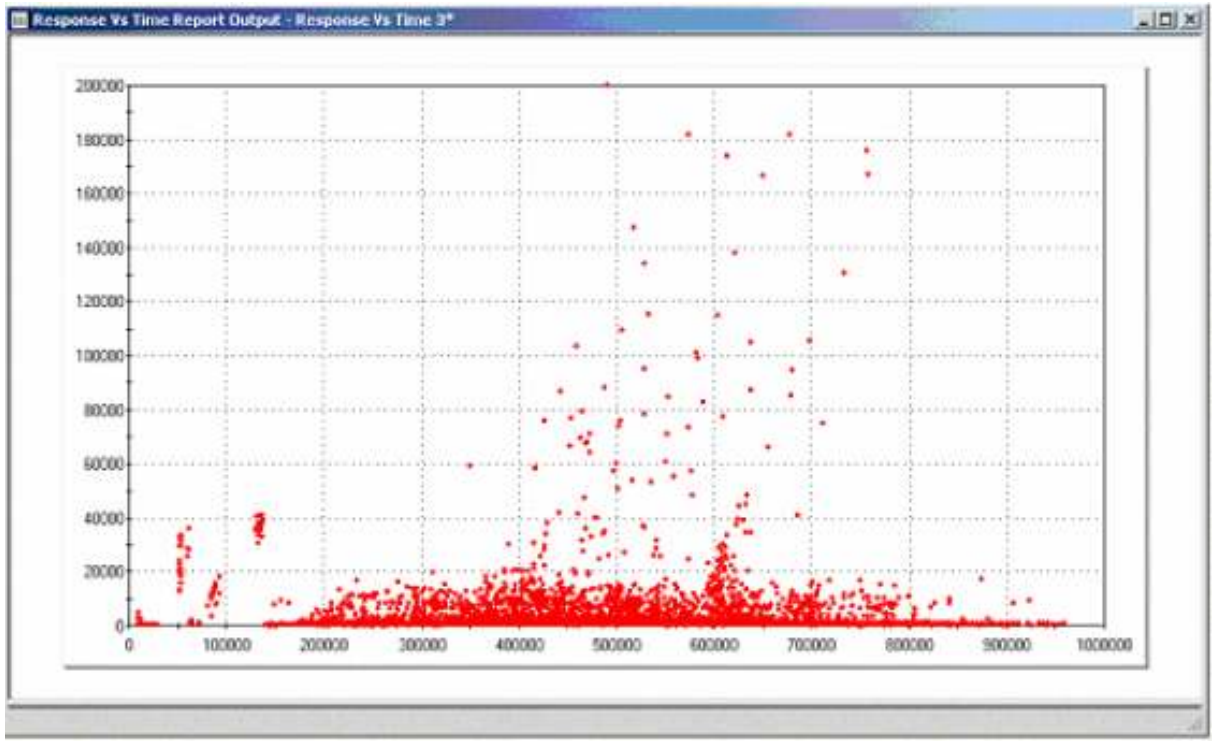


	1 LAN User	50 LAN Users	100 LAN Users	150 LAN Users	200 LAN Users	100 128Kbs Users	100 56.6Kbs Users	100 28.8Kbs Users
HomePage	5.59	9.08	10.14	10.11	48.21	11.42	24.85	33.89
Page1	0.75	0.74	1.48	2.37	41.22	6.80	14.10	27.09

Analyzing Results to ID Bottlenecks

Study Scatter Charts

- Any pattern that shows more than one dot outside of your predefined acceptable performance levels is a potential bottleneck
- Part 6 of the “Beyond Performance Testing” series



Analyzing Results to ID Bottlenecks

- Rely on Personal Observation
- Listen to Third Party Comments
- Confirm Suspects
- Reproduce Results
  - Exactly
  - Manually
  - With Similar Tests
  - With Minimalist Tests
  - With Not-So-Similar Tests
- Report Suspects
  - Verbally
  - Visually
  - Via Demonstration

What the Dev Team Needs to Know

Which related activities produce the same symptoms?

Which other activities are affected by the bottleneck?

What were the load characteristics of the test yielding the symptoms?

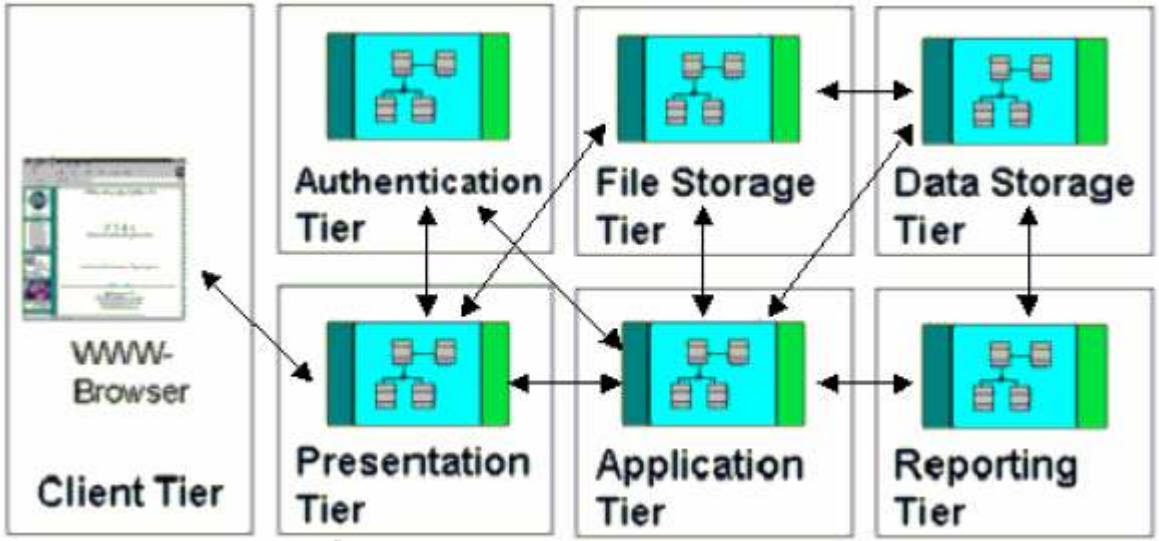
What data did you use to create the symptoms?

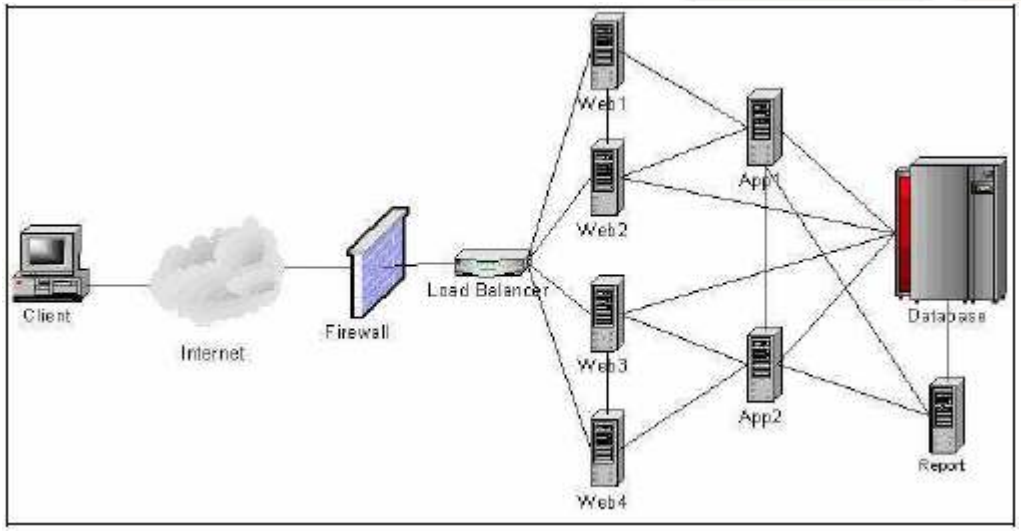
What's the configuration of the environment you're testing?

Other metrics the developers wanted you to collect.

Identify Tier of Detected Issue

Logical Architecture

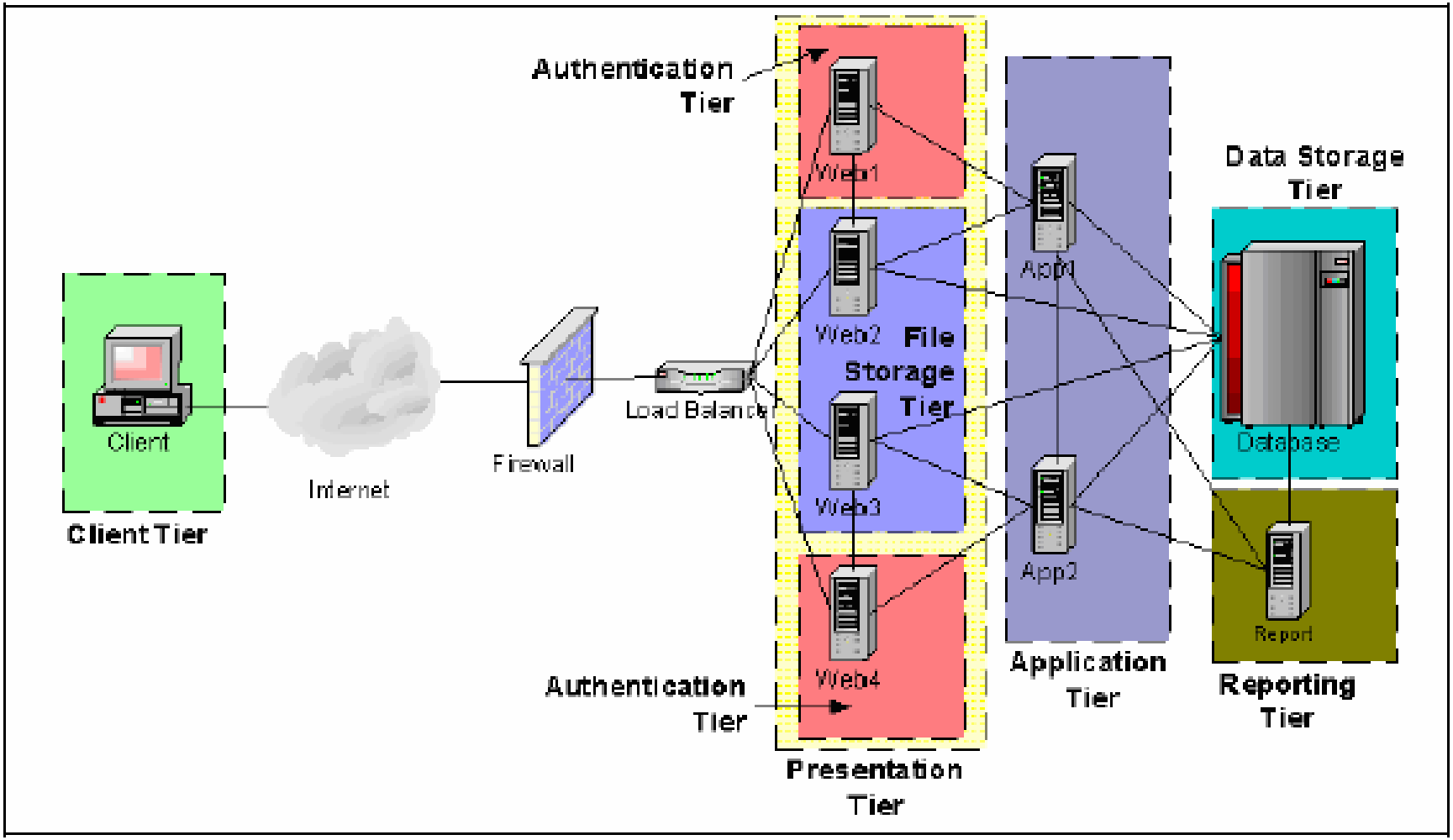




Physical Architecture

Identify Tier of Detected Issue

Physical Architecture with Logical Overlay



Identify Tier of Detected Issue

Design Tests to Determine Tier

- Ask “What if…? Questions.
- Ask Developers to Speculate
- Evaluate Commands with Slow Responses
- Think in Terms of Distinguishing Failures, Slow Spots and Bottlenecks
- Visualize and Prioritize

Modify Existing Tests

Create New Tests

- Use Same Tool
- Use Different Tool
- Use Test Harnesses

Identify Tier of Detected Issue

Speak Intelligently with the Development Team

Capture Metrics by Tier

- Resource Utilization
- Response Times
- Others Identified by Developers

Interpret Metrics

- Look for the Obvious
- Consult Development Team
- Change Tests to Prove (or Disprove) Theories

Identify Component of Detected Issue

Once Tier is Identified…

- Further Narrowing may be Required
- Same Principles as Identifying Tier

Speak Intelligently with the Development Team

Capture Metrics by Component

- Resource Utilization
- Response Times
- Others Identified by Developers

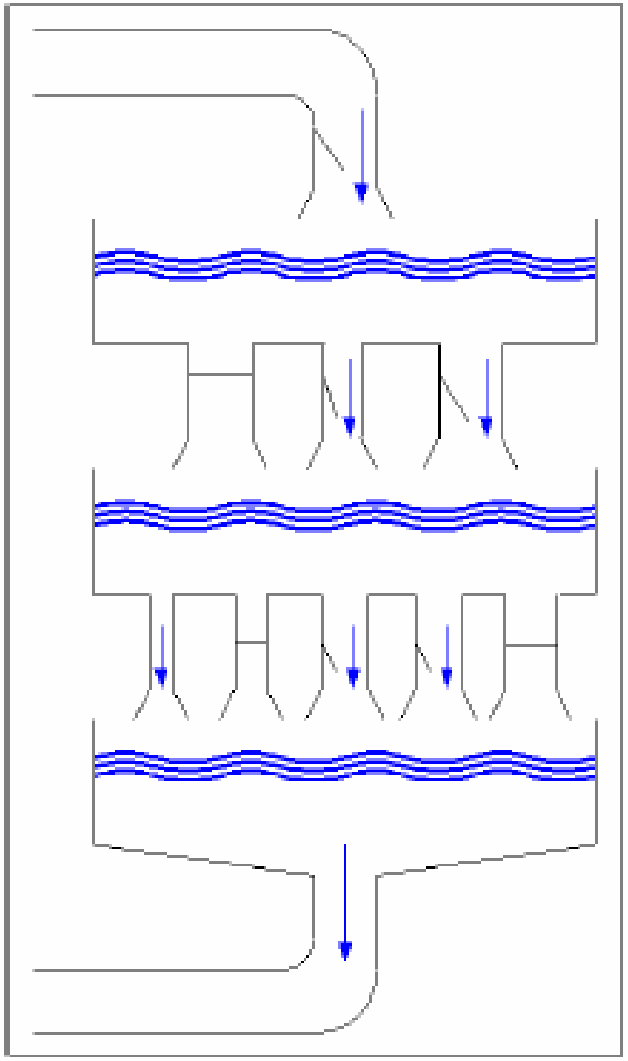
Interpret Metrics

- Look for the Obvious
- Consult Development Team
- Change Tests to Prove (or Disprove) Theories

Develop Test To Exploit Issue

Lessons from Hydrodynamics

- Pools/Queues
- Flows/Threads/Processes
- Pipe Size/Throughput



Develop Test To Exploit Issue

Exploitation Methods

- Bounds Conditions
- Breakpoints
- Resource Constraints

Hand off to Development Team

- Following Development Team's Lead (Case Studies, BPT 10)

Different Testing Styles

- Black-Box
- Grey-Box
- White-Box

Knowing When to put the Load Generation Tool Away

Available Tools

Common

- LoadRunner, Silk Performer, Performance Tester, OpenSTA, Visual Studio Test System, eLoad
- Purify, Quantify, Performassure
- J-Meter, Perfmon, Perfmeter, Top
- WebTrends, WebLogic, Tivoli

Other Performance Test/Monitoring Tools

- Load Generation
- Performance Monitoring
- Performance Analysis
- OS/System Specific Tools
- Application Specific Tools

Available Tools

Other Analysis Tools

- Spreadsheets
- Statistical Calculators
- Mathematical Graphing
- Graphical Presentation

Most Important Tools

- Your Brain
- Your Development Team

Examples

EXAMPLES

Want More Information?

Information adapted from User Experience, not Metrics: Parts 6, 8, 9 and10 and Beyond Performance Testing: Parts 6, 7, 8, 9, 10 located at http://www-106.ibm.com/developerworks/rational/library/ (RDW) and http://www.PerfTestPlus.com

Ask me directly on the Performance and Load Testing forums on RDW (http://www-106.ibm.com/developerworks/forums/dw_rforums.jsp) or http://QAForums.com (Huge QA Forum)

Good sources for additional information about Performance Testing:

- http://www.PerfTestPlus.com (Methodology, Templates, Articles, Presentations)
- http://www.loadtester.com (Good articles and links)
- http://www.keynote.com/resources/resource_library.html (Good articles and statistics)

Graphical Presentation of Information – Edward Tufte, PhD. http://www.edwardtufte.com (Books and seminars)

Summary

Report Symptoms, not Solutions/Reasons

Verify Observations

Analyze Results Collaboratively

Don’t Over-depend on your Initial Tests

Determine Related and Unrelated Activities

Analyze, Analyze, Analyze (Collaboratively)

Know When to Hand-off to the Development Team

Yours is a Support Role

Document Conclusions

Document Recommendations

Questions

Contact Info

Scott Barber

Chief Technology Officer

PerfTestPlus, Inc

E-mail:

sbarber@perftestplus.com

Web Site:

www.PerfTestPlus.com