Introduction to Software Testing

Lecture 12

Performance Testing

Instructor: Morteza Zakeri

Slides by: **Morteza Zakeri**

March 2024

Performance Testing Tips and Techniques

TOC

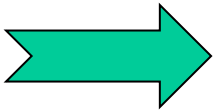
- What is performance test?
- Fundamentals of effective application performance testing
- Process of performance test.
- Interpreting results: Effective root-cause analysis
- Application Technology and Its Impact on Performance Testing

Why Performance Test

- What is it?
- Why is so important?
- Why carry out performance testing in the first place

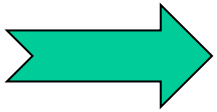
What is performance

- performance requirements
  - Service-oriented
    - Availability
    - Response time



they measure how well (or not) an application is providing a service to the end users.

  - efficiency-oriented
    - Throughput
    - Utilization



they measure how well (or not) an application makes use of the application landscape

Levels of performance testing maturity

Resolving Performance Defects (2006)

Approach	***% Resolved in Production***
Firefighting	100%
Performance Validation	30%
Performance Driven	5%

Source: Forrester Research

Some questions related to application performance

- How many end users will the application need to support at release? After 6 months, 12 months, 2 years?
- Where will these users be located, and how will they connect to the application?
- How many of these users will be concurrent at release? After 6 months, 12 months, 2 years?
- How many and what specification of servers will I need for each application tier?
- What sort of network infrastructure do I need to provide?

The Fundamentals of Effective Application Performance Testing

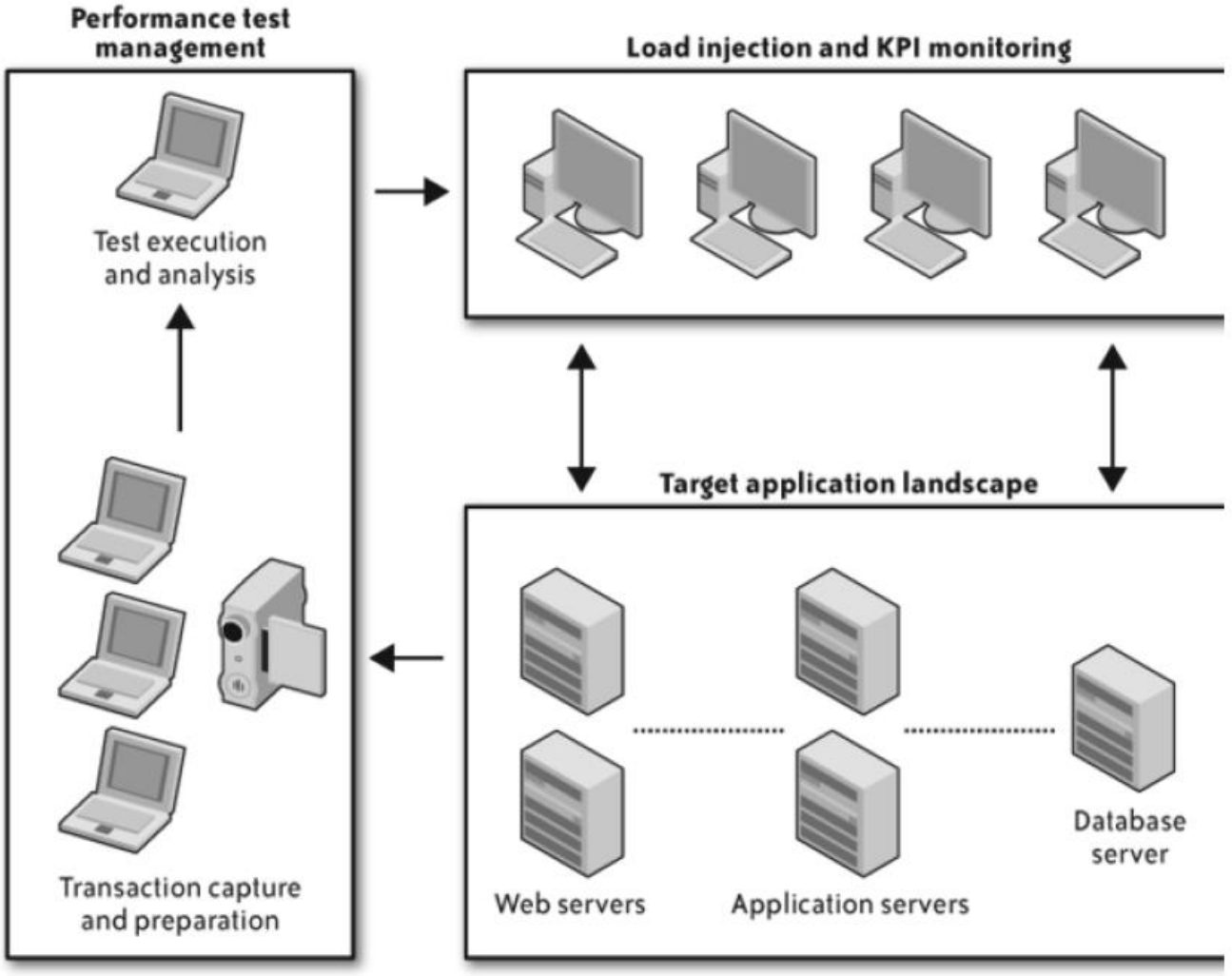
- Choosing an appropriate performance testing tool
- Designing an appropriate performance test environment
- Setting realistic and appropriate performance targets
- Making sure your application is stable enough for performance testing
- Obtaining a code freeze
- Identifying and scripting the business-critical transactions
- Providing sufficient test data of high quality
- Ensuring accurate performance test design
- Identifying the server and network monitoring key Performance Indicators (KPIs)
- Allocating enough time to performance test effectively

Choosing an Appropriate Performance Testing Tool

- Automated performance test tools typically have the following components.
  - Scripting module
  - Test management module
  - Load injector(s)
  - Analysis module

Typical Performance Tool Deployment

Figure 2-1. Typical performance tool deployment



Evaluation of a performance testing tool (continue)

- evaluation of a performance testing tool should include the following steps:
  - Tool vendor support
  - Licensing model
  - Proof of Concept (POC)
  - Scripting effort
  - Solution versus load testing tool
    - automated requirements management
    - automated data creation and management
    - pre-performance test application tuning and optimization
    - response-time prediction and capacity modeling
    - application server analysis to component level
    - integration with end-user experience (EUE) monitoring after deployment.
  - In-house or outsource?

Designing an Appropriate Performance Test Environment

- Differences between performance test environment and operational environment
  - ✓The number and specification of servers (probably the most common reason)
  - ✓Bandwidth and connectivity of network infrastructure
  - ✓Number of application tiers
  - ✓Sizing of application databases

3 Levels of preference when it comes to designing a test environment

- An exact or very close copy of the live environment
- A subset of live environment with fewer servers but specifications and tire development matches to that of the live environment
- A subset of the live environment with fewer servers or lower specification

Some tips for test environment

- Use virtualization
  - Bus Versus LAN-WAN
  - Physical versus virtual NIC
- Capability of load generators
- How application reacts to IP address of each incoming virtual user.
- Situations that affect how many injectors you will need:
  - The application technology may not be recordable at the middleware level
  - Measure performance from presentation layer
- From where user access the application:
  - Available bandwidth
  - Network latency

Check list for test environment

- Number of server
- Load balancing strategy
- Hardware inventory
- Software inventory
- Application component inventory
- External links

A number of ways to simulate WAN-based users

- Modify transaction replay: slowing down the rate of execution
- Load injection from a WAN location: load injector machines at the end of real WAN links
- Network simulation: simulate WAN conditions from a network perspective, including bandwidth reduction

The groups involved in setting performance targets

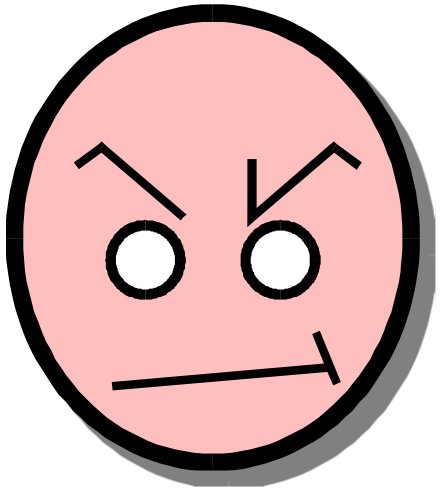
- Chief information officer (CIO)
- Chief technology officer (CTO)
- Chief financial officer (CFO)
- Departmental heads
- The developers
- The testers (internal or outsourced):
- The testers are the people at the sharp end, so among other considerations they need to know the correct transactions to script, the type of test design required, and the performance targets they are aiming for.
- The infrastructure team(s)
- The end users

Setting realistic and appropriate performance targets

- Availability
- Concurrency, scalability, and throughput
- Response time
- Network utilization
  - Data volume
  - Data throughput
  - Data error rate
- Server utilization
  - CPU utilization
  - Memory utilization
  - Disk input/output
  - Disk space

Making sure your application is stable enough for performance testing

- Poorly performing SQL
- Large number of application network round trips
- Undetected application errors



Obtaining a code freeze

- It is absolutely vital to carry out performance testing against a consistent release of code.
- automated performance testing relies on scripted transactions. These scripts are version dependent;
- An unanticipated new release of code may partially or completely invalidate these scripts
- It's often the result of a communication breakdown between test teams and management

Identifying and scripting the business-critical transactions

- For each transaction you should complete the following actions:
  - Define and document each execution step so that there is no ambiguity in the navigation path
  - Identify all input data requirements and expected responses
  - Determine the type of user that the transaction involves
  - What is the connectivity path for this transaction?
  - Will this be an active or passive transaction?

Some Considerations

- Verify single user replay
- Verify multi-user replay
- decide which parts of the transaction to measure in terms of response time
- Decide which parts of a transaction are repeated during test execution
- consider is whether your application will exist in isolation or have to share resource with other applications.

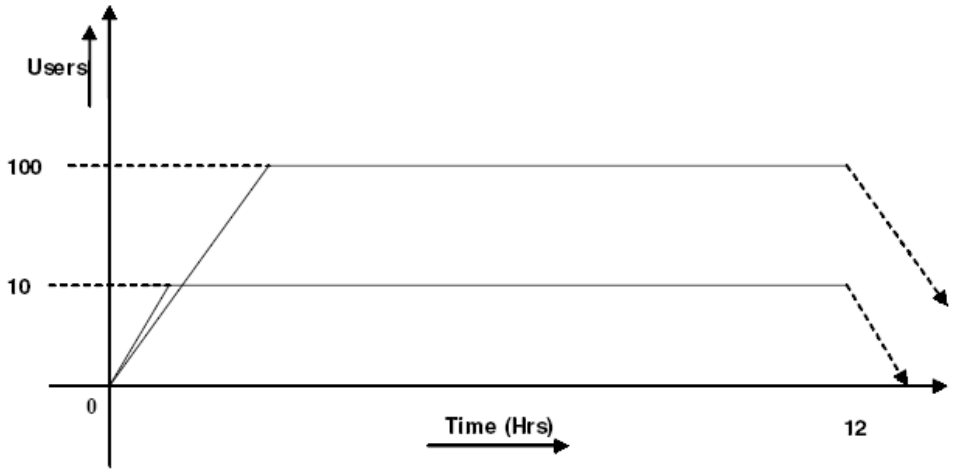
Providing sufficient test data of high quality

- Three types of test data
  - input data
    - User credentials
    - Search criteria
    - *Associated documents*
  - target database
  - runtime data
- Challenges of creation test database
  - Sizing
  - Data rollback
- Data management tools

Ensuring accurate performance test design

- Baseline test
- Load test
  - up to the target concurrency but usually no further

Figure 1.2. Load vs. time during load testing



  - performance targets
    - Availability
    - concurrency or throughput
    - response time
  - closest approximation of real applicatio

Ensuring accurate performance test design

- Stress test
  - determine the upper limits or sizing of the infrastructure.
  - a stress test continues until something breaks
    - no more users can log in
    - response time exceeds the value you defined as Acceptable
    - the application becomes unavailable
  - to know your upper limits

Ensuring accurate performance test design

- Stress test

Figure 1.4. Stress testing during a stable state

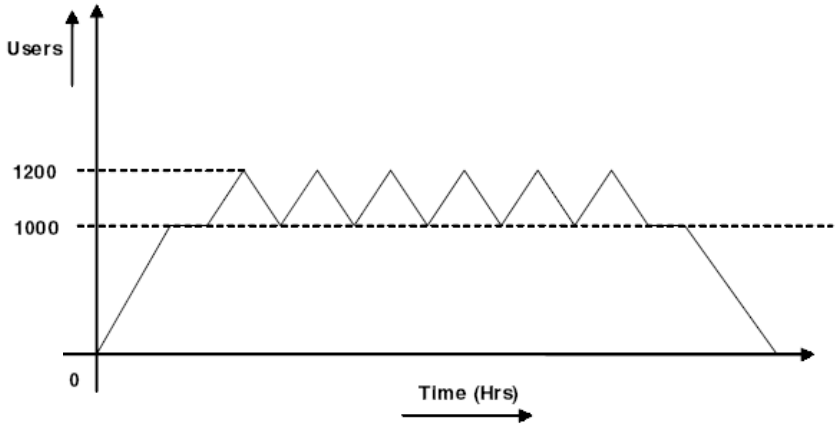


Figure 1.5. System degradation during stress testing

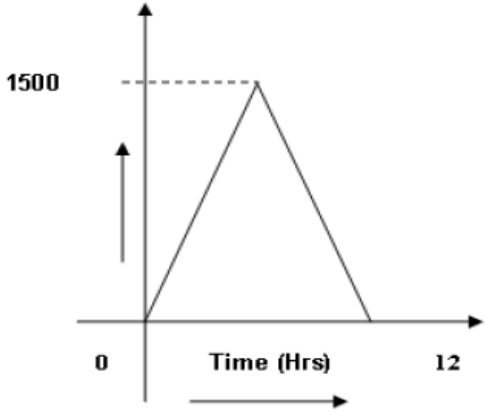


Figure 1.5a. Graceful degradation

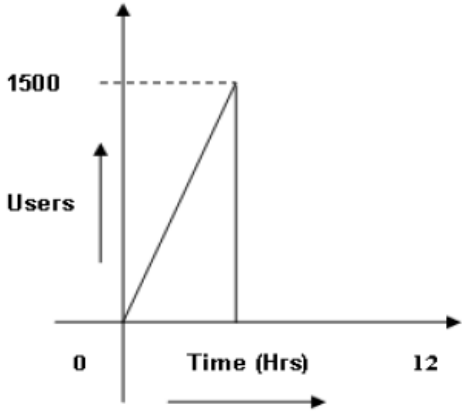
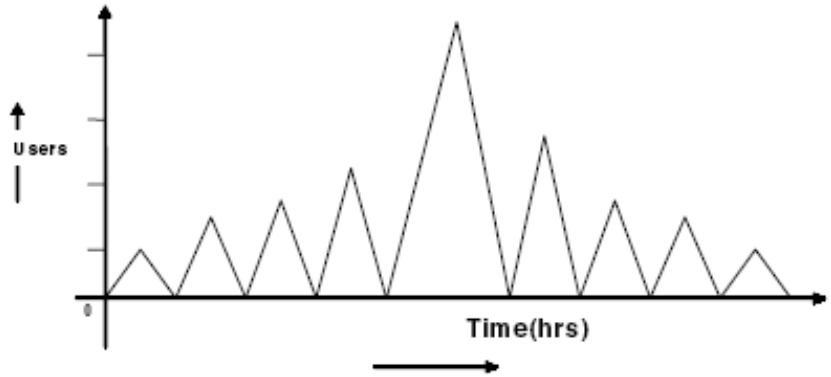


Figure 1.5b. Instant degradation

Ensuring accurate performance test design

- Stress (Spike) test

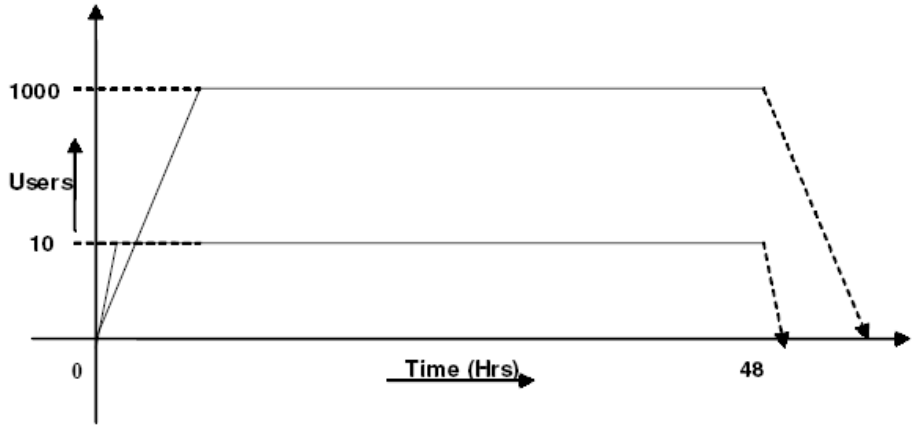
Figure 1.6. Load vs. spike testing



Ensuring accurate performance test design

- Soak or stability or endurance test
  - This sort of test cannot be carried out effectively unless appropriate server monitoring is in place.

Figure 1.3. Load vs. time during endurance testing



Ensuring accurate performance test design

- Smoke test
  - to focus only on what has changed
- Isolation test
  - used to home in on an identified problem

Ensuring accurate performance test design

Types of testing	Number of concurrent users and ramping up	Duration
Load Testing	1 User → 50 Users →100 Users →250 Users →500 Users…….. →1000Users	12 Hours
Stress Testing	1 User → 50 Users →100 Users →250 Users →500Users…….. →1000Users →Beyond 1000Users…….. →Maximum Users	12 Hours
Spike Testing	1 User→ 50Users→ Beyond 1000 Users	12 Hours → 10 Hours → 8 Hours… Hour….Minutes
Endurance Testing	Maximum Users	12 Hours -> …..Longer duration(days)

Some consideration

- Think Time
  - it introduces an element of realism into the test execution.
  - With think time removed, as is often the case in stress testing, execution speed and throughput can increase tenfold, rapidly bringing an application infrastructure that can comfortably deal with a thousand real users to its knees.
  - always include think time in a load test.
  - think time influences the rate of transaction execution

- Pacing
  - another way of affecting the execution of a performance test
  - affects transaction throughput

Type of injection profile

- Big Bang
  - Stress test
  - This provides a better simulation of real user behavior.
  - For Big Bang load injection, caution is advised because large numbers of virtual users starting together can create a fearsome load, particularly on web servers. This may lead to system failure before the performance test has even had a chance to start properly.

- Ramp-up
  - This mode begins with a set number of virtual users (typically 0 or 1) and then adds more users at specified time intervals until a target number is reached.
  - This is the usual way of testing to see if an application can support a certain number of concurrent users.

Type of injection profile

- Ramp-up (with step)
  - In this variation of straight ramp-up, there is a final target concurrency or throughput, but the intention is to pause injection at set points during test execution.
  - For example, the target concurrency may be 1,000 users, but there is a need to observe the steady-state response time at 250, 500, and 750 concurrent users;

- Ramp up (with step), Ramp down (with step)

Sample Structure of performance test

- **Baseline test each transaction**
  - This provides an indication of the best performance you're likely to get from whatever activity the transaction represents.
  - You should run this sort of test for a set period of time or a number of iterations.
  - There should be no other application activity going on at the same time
- **Load test each transaction**
  - Decide on the maximum number of concurrent users or throughput you want for each transaction;
  - I tend to use the ramp-up with step approach, since this allows fine-grain analysis of problems that may occur at a particular level of concurrency or throughput.
- **Isolation test individual transactions**

Sample Structure of performance test

- Load test transaction groups
  - database lock contention
- Isolation test transaction groups
- Soak test transaction groups
  - Repeat the individual transaction and/or transaction group performance tests for an extended duration. This should reveal any problems that manifest themselves only after a certain amount of application activity has occurred.
- Stress test transaction groups
  - Repeat the individual transaction and/or transaction group tests but reduce the amount of sleep time and pacing to create a higher throughput than was achieved during the load tests. This allows you to ascertain the upper capacity limits of the application landscape and to determine how the application responds to sudden peaks of activity.
- Non-performance tests

Identifying the server and network monitoring key Performance Indicators (KPIs)

- It is important to approach server KPI monitoring in a logical fashion—ideally, using a number of layers.

- The ideal approach is to build separate templates of performance metrics for each layer of monitoring. Once created, these templates can form part of a reusable resource for future performance tests

key Performance Indicators (KPIs)

- depending on the application architecture, any or all of the following models or templates may be required:
  - ✓Generic templates
    - Processor utilization percentage
    - Top 10 processes
    - Available memory in bytes
    - Memory pages/second
    - Processor queue length
    - Context switches per second
    - Physical disk: average disk queue length
    - Physical disk: % Disk Time
    - Network interface: Packets Received Errors
    - Network interface: Packets Outbound Errors
  - ✓Web and application server tier
  - ✓Database server tier
  - ✓Mainframe tier

Key Performance Indicators (KPIs)

- network monitoring focuses
  - Packet round-trip time
  - Detection of any errors that may occur as a result of high data volumes
  - For the Windows and Unix/Linux operating systems there are performance counters that monitor the amount of data being handled by each NIC card as well as the number of errors (both incoming and outgoing) detected during a performance test execution.
  - To help better differentiate between server and network problems, some automated performance test tools separate server and network time for each element within a web page

Example

Figure 2-5. Example server/network response time breakdown

Name	Response Code	Total Requests	Average response time	Average server	Average network	Server / Networ
https://OnlineBanking.uk/online/img/arrow.gif	200	2	2.0816	99.9881 %	0.0119 %	
https://OnlineBanking.uk/online/css/advIE.css	200	2	2.9855	31.8739 %	68.1261 %	
http://OnlineBanking.uk/js/loan.js	200	6	0.4469	99.9912 %	0.0088 %	
http://OnlineBanking.uk/img/warning.jpg	200	6	0.0219	99.863 %	0.137 %	
http://OnlineBanking.uk/img/logo.gif	200	6	0.0519	99.9554 %	0.0446 %	
http://OnlineBanking.uk/img/award.gif	200	6	0.1396	99.9756 %	0.0244 %	
http://OnlineBanking.uk/img/small.jpg	200	6	0.4033	99.9864 %	0.0136 %	

The Process of performance testing

Allocating enough time to performance test effectively

- Lead time to prepare test environment
- Lead time to prepare the injection environment
- Time to identify and script business transactions
- Time to identify and create enough test data
- Time to prepare and execute performance test runs
- Time to deal with any problems identified

Sample of time scale

- Scripting performance test transactions: Allow half a day per transaction.
- Creating and validating performance test sessions or scenarios: Typically one to two days' work.
- Performance test execution: Allow a minimum of five days.
- Data collection (and software uninstall): Allow one day.

Necessary information for project plan creation

- Deadlines available to complete performance testing, including the scheduled deployment date.
- Whether to use internal or external resources to perform the tests.
- Test environment design agreed. (An appropriate test environment will require longer to create than you estimate.)
- Ensure that a code freeze applies to the test environment within each testing cycle.
- Ensure that the test environment will not be affected by other user activity.
- All performance targets identified and agreed to by appropriate business stakeholders.
- The key application transactions identified, documented, and ready to script.
- Which parts of transactions should be monitored separately (i.e.,checkpoints).
- Identify the input, target, and runtime data requirements for the transactions that you select.

Necessary information for project plan creation

- Make sure that you can create enough test data of the correct type within the time frames of your testing project. Don't forget about data security and confidentiality.
- Performance tests identified in terms of number, type, transaction content, and virtual user deployment. You should also have decided on the think time, pacing, and injection profile for each transaction deployment.
- Identify and document server, application server, and network KPIs.
- Identify the deliverables from the performance test in terms of a report on the test's outcome versus the agreed performance targets.
- A procedure is defined for submission of performance defects discovered during testing cycles to development or the application vendor.

performance testing in-house

- Do you have a dedicated performance testing team? At a minimum you will need a project manager and enough testing personnel (rarely are more than two needed) to handle the scale of the project.
- Make sure the team has the tools and resources it needs to performance test effectively.
- Ensure all team members are adequately trained in the testing concepts and tools to be used.

Using previous information for the following

- Develop a high-level plan that includes resources, time lines, and milestones based on these requirements.
- Develop a detailed performance test plan that includes all dependencies and associated time lines, detailed scenarios and test cases, workloads, and environment information.
- Include contingency for additional testing cycles and defect resolution if problems are found with the application during performance test execution.
- Include a risk assessment of not meeting schedule or performance targets.

4.Interpreting Results

- Response-time data
- Transaction throughput
- State of load generator
- Server, application server and network KPI
- Performance threshold
- Error occurrence

Types of Output from a Performance Test

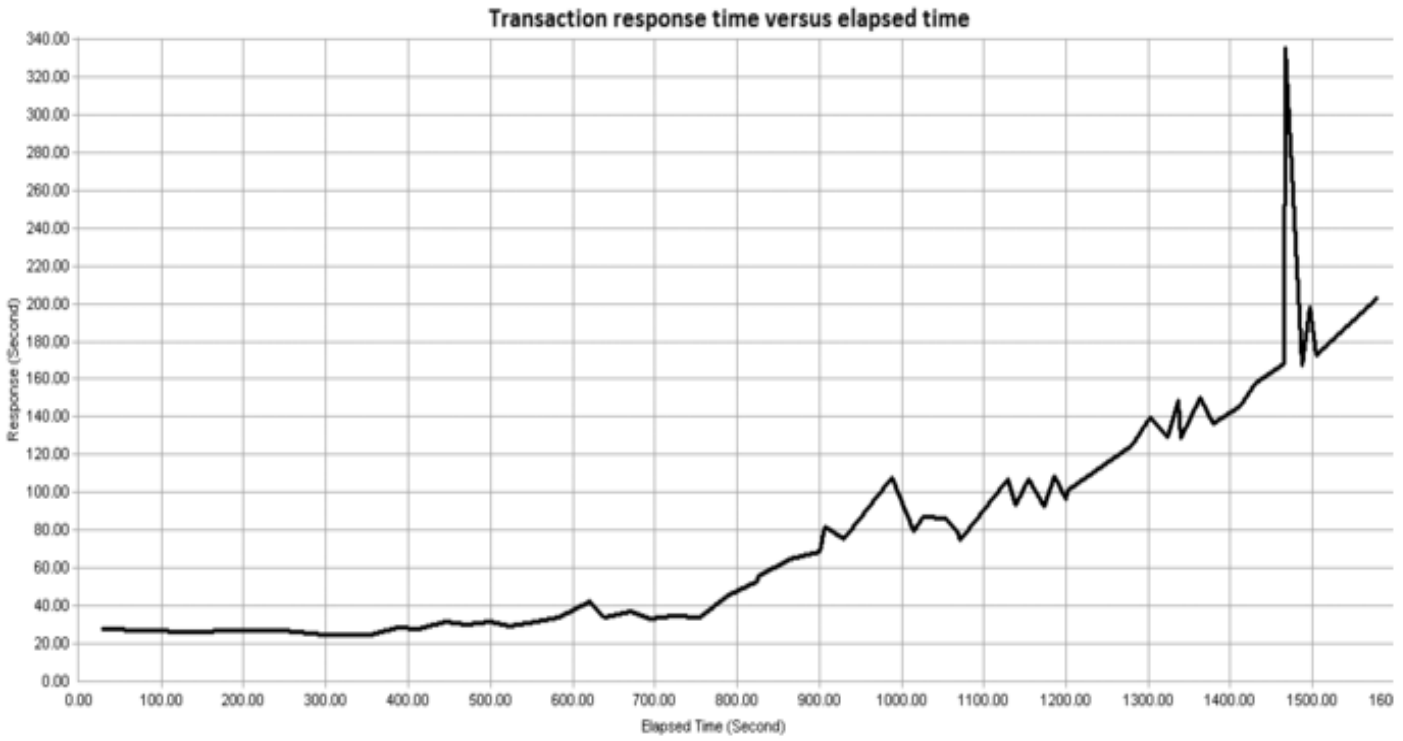
- *Mean and median*
  - It is commonly used
  - average response times
  - 1, 2, 2, 2, 3, 9
- *Standard deviation and normal distribution*
- Response-time distribution

Important Tip

- If the application fails to respond in the required time, the performance tool will record some form of time-out error. If this situation occurs then it is quite likely that an overload condition has occurred somewhere in the application landscape. We then need to check the server and network KPIs to help us determine where the overload occurred.
- An overload doesn't always represent a problem with the application. It may simply mean that you need to increase one or more time-out values in the transaction script or the performance test configuration.
- The think-time delays are not normally included in response time measurement, since your focus is on how long it took for the server to send back a complete response after a request is submitted. Some tools may break this down further by identifying at what point the server started to respond and how long it took to complete sending the response.

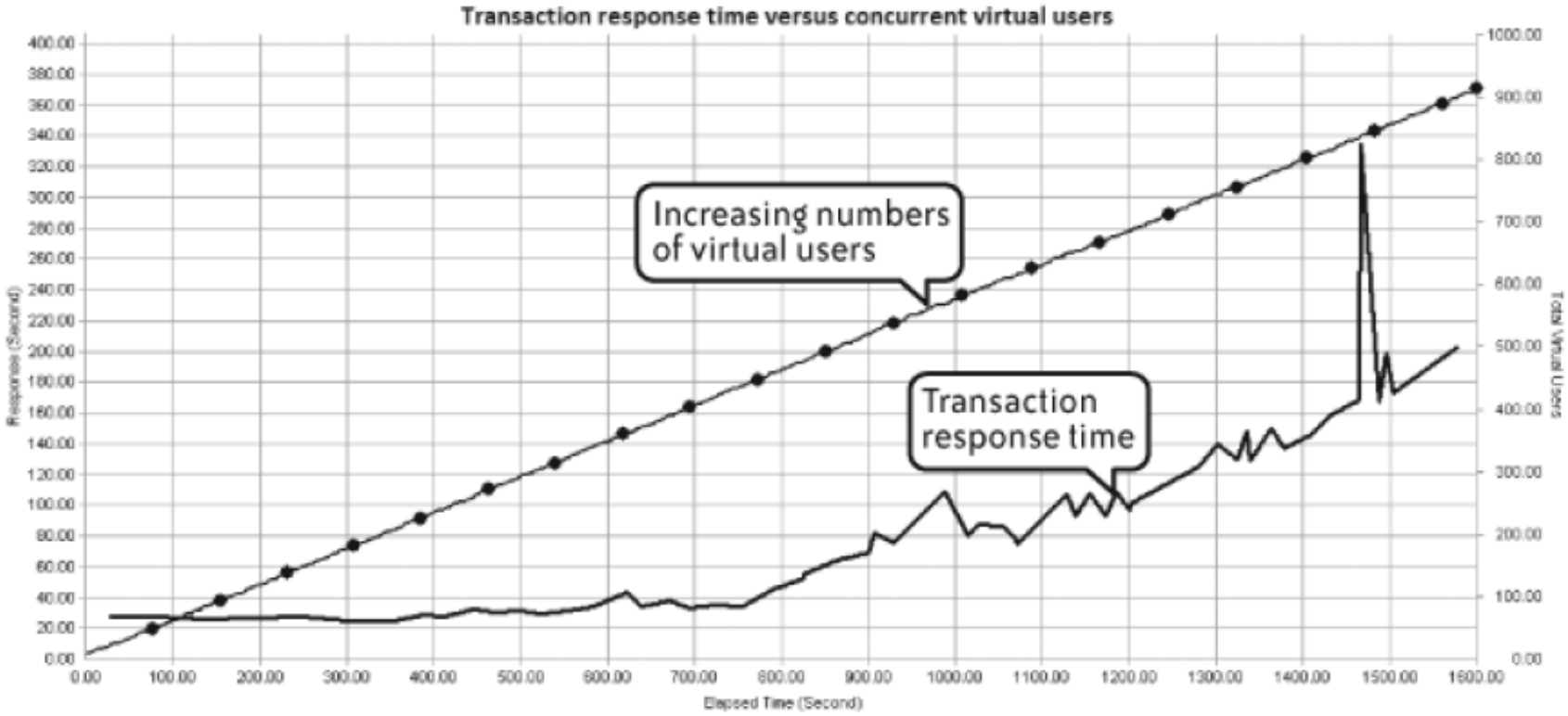
Response time

Figure 4-3. Transaction response time for the duration of a performance test



Response time versus concurrent virtual users

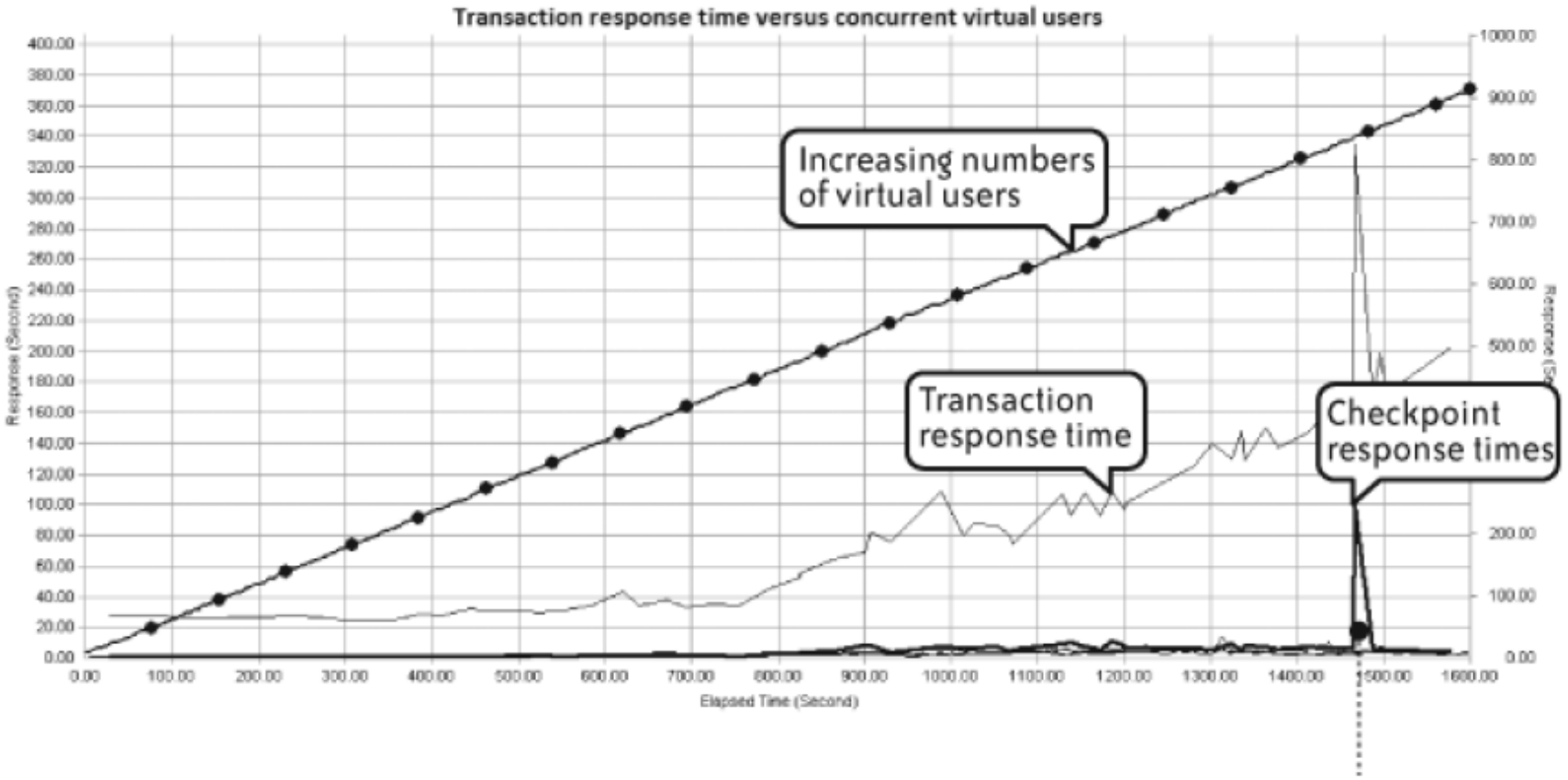
Figure 4-4. Transaction response time correlated with concurrent users for the duration of a test



You would normally expect an increase in response time as more virtual users become active, but this should not vary in lockstep with increasing load.

Checkpoint

Figure 4-5. Transaction and checkpoint response time correlated with concurrent users

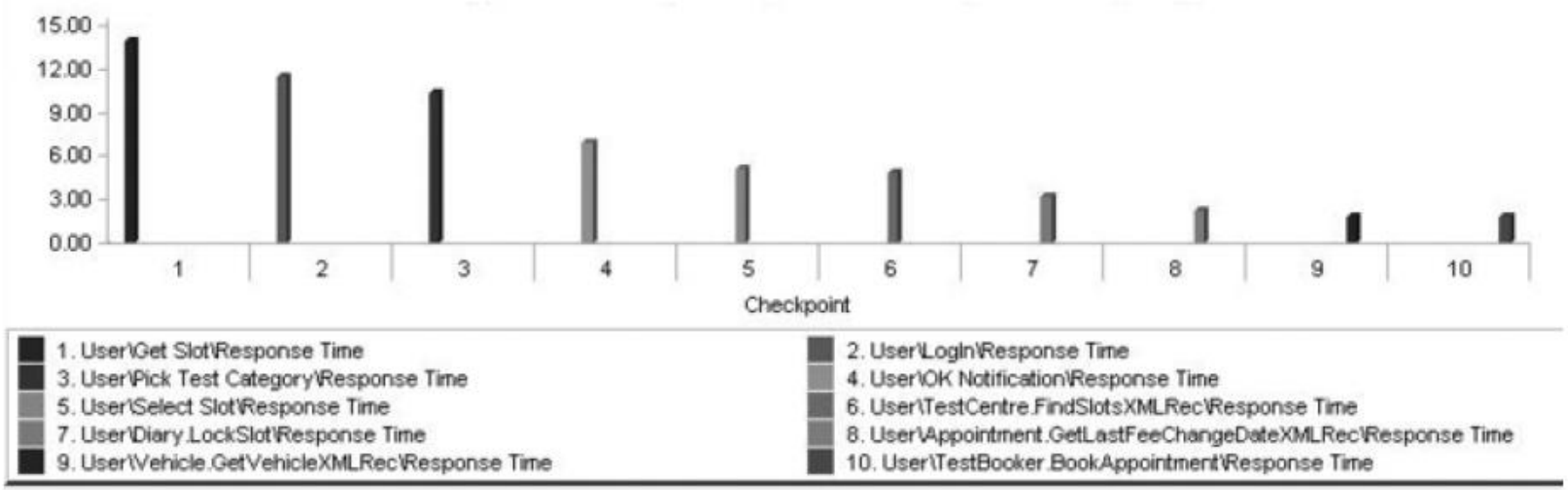


Checkpoint

Figure 4-6. Table of response-time data

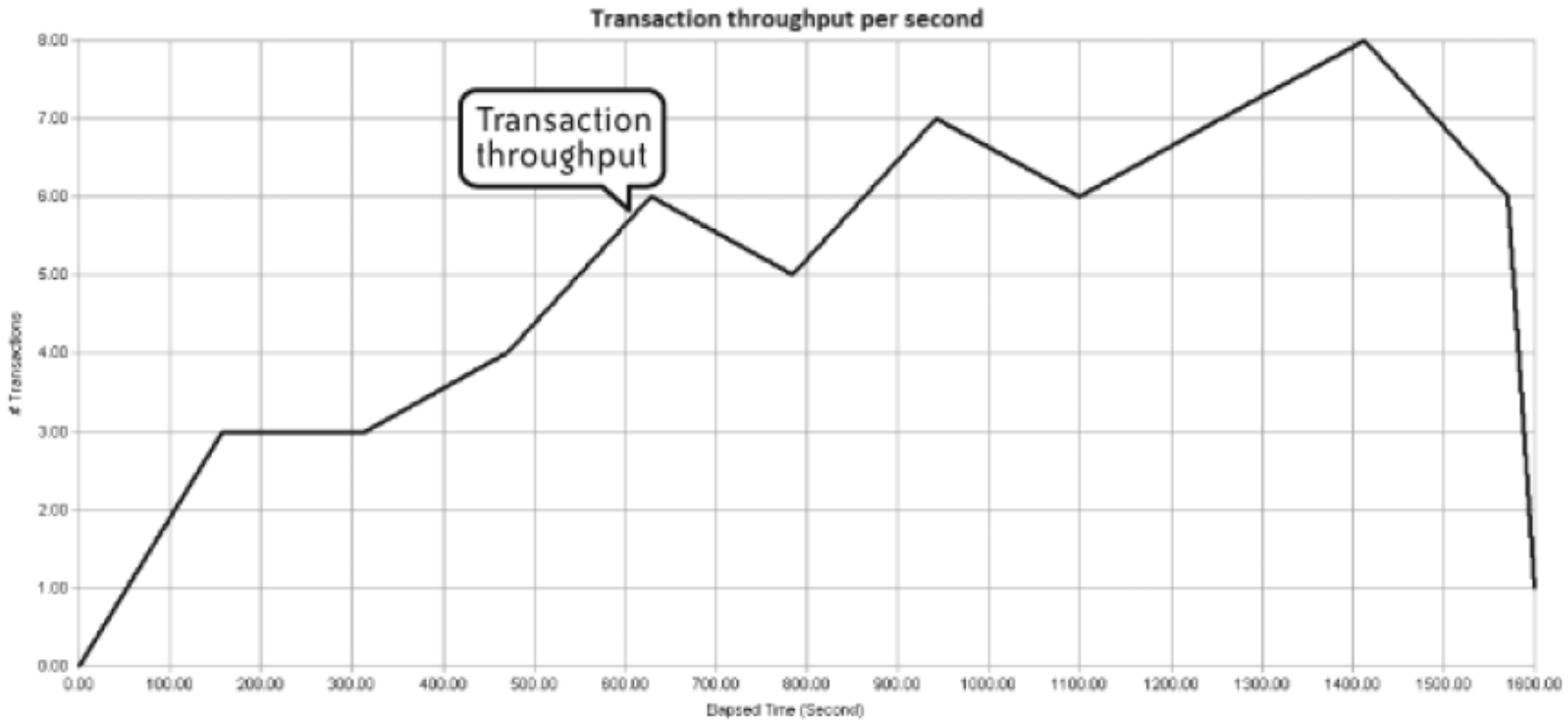
Name	Min	Mean	Max	Last	Std.Dev.
Transaction Response Time	24.9380	86.0258	335.1410	202.7350	61.5771
Total Running Virtual Users	10.0000	458.7383	913.0000	913.0000	261.8142
Login Response Time	0.7660	7.3413	24.2350	7.1870	5.3617
Security Question Response Time	0.5930	6.7524	20.6250	6.9690	5.0116
Click Custom Statements Response Time	0.8120	6.0654	35.0470	4.7970	5.2828
Show Statement Response Time	0.7970	6.0582	15.9690	8.4840	4.0250
Logoff Response Time	1.2030	15.1568	246.0940	10.7820	32.2105

Response time in each transaction



Transaction throughput

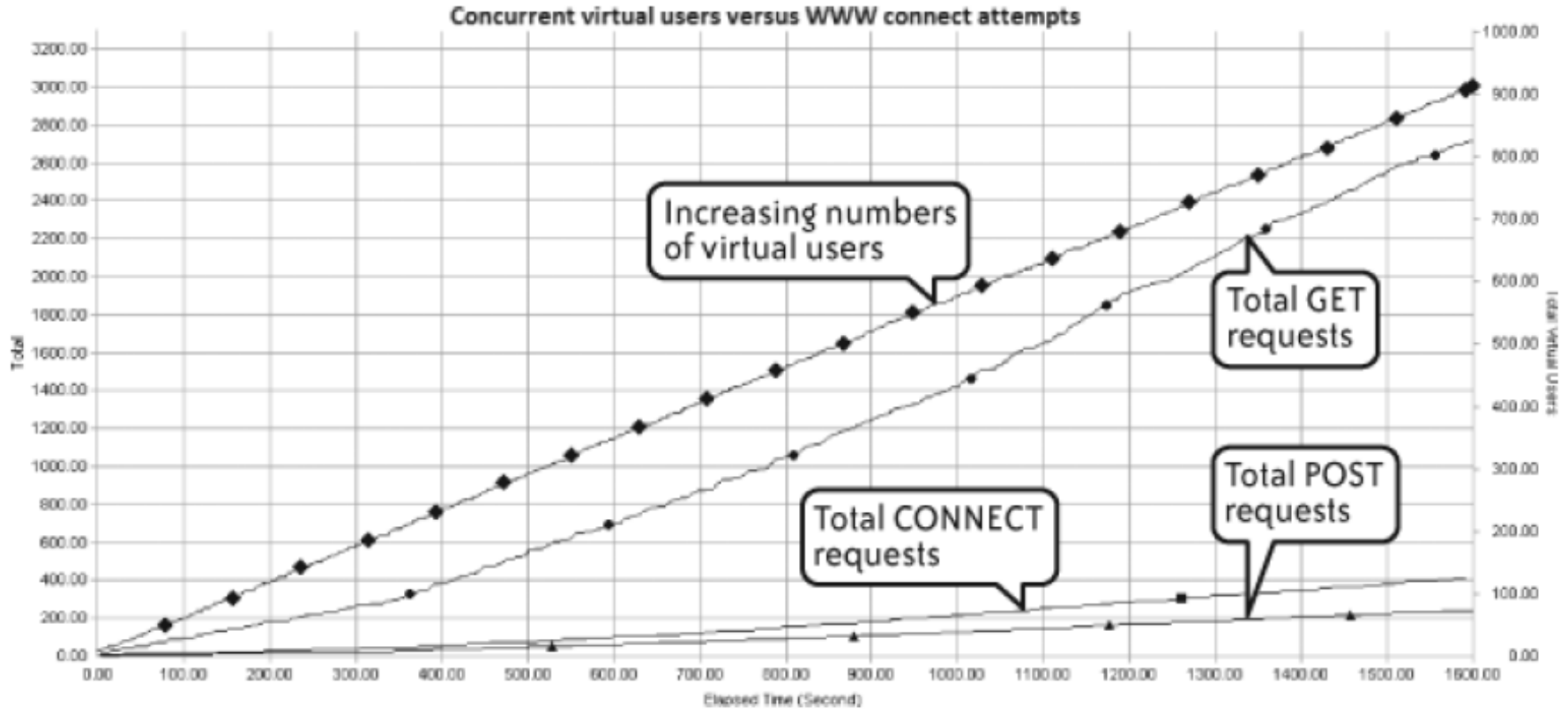
Figure 4-8. Transaction throughput



- This view shows when peak throughput was achieved and whether any significant variation in transaction throughput occurred at any point.
- reduced throughput is a useful indicator of the capacity limitations in the web or application server tier

Concurrent virtual users correlated with web request attempts

Figure 4-9. Concurrent virtual users correlated with web request attempts



Mechanisms to monitor server and network performance

- application technology
- capabilities of your performance testing solution
  - **Remote monitoring**
    - monitor many servers from a single location
    - usually, no need to install any software onto the servers
  - **Installed agent**

Remote monitoring technologies

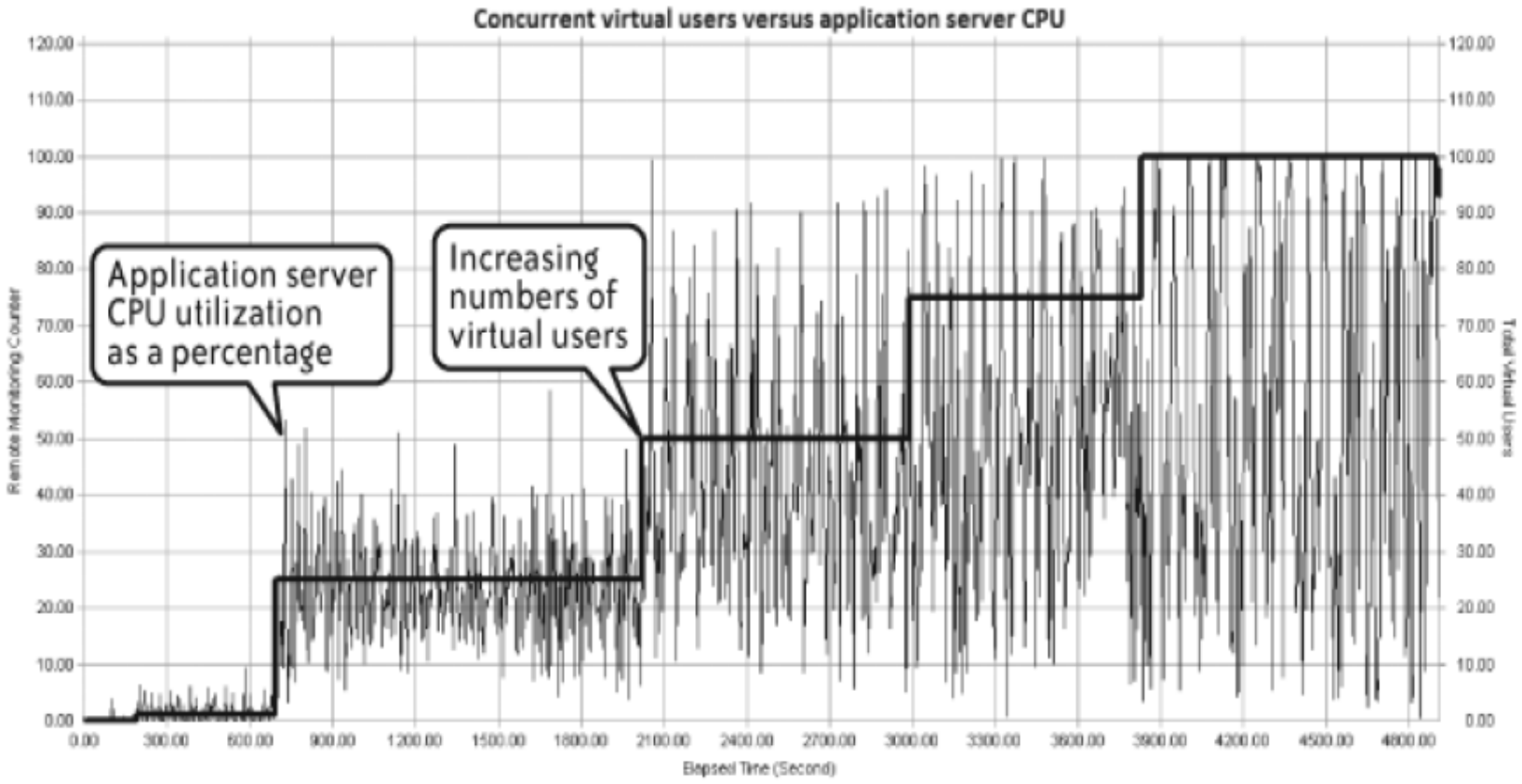
- Windows Registry
  - Microsoft's Performance Monitor (Perfmon) application.
  - Windows operating systems
- Simple Network Monitoring Protocol (SNMP)
- Java Monitoring Interface (JMX)
  - JMX is useful mainly when monitoring Java application servers such as IBM WebSphere, ORACLE WebLogic, and JBOSS
- Rstatd
  - It provides basic kernel-level performance information

Installed agent

- When it isn't possible to use remote monitoring—perhaps because of network firewall constraints or security policies—your performance testing solution may provide an agent component that can be installed directly onto the servers you wish to monitor

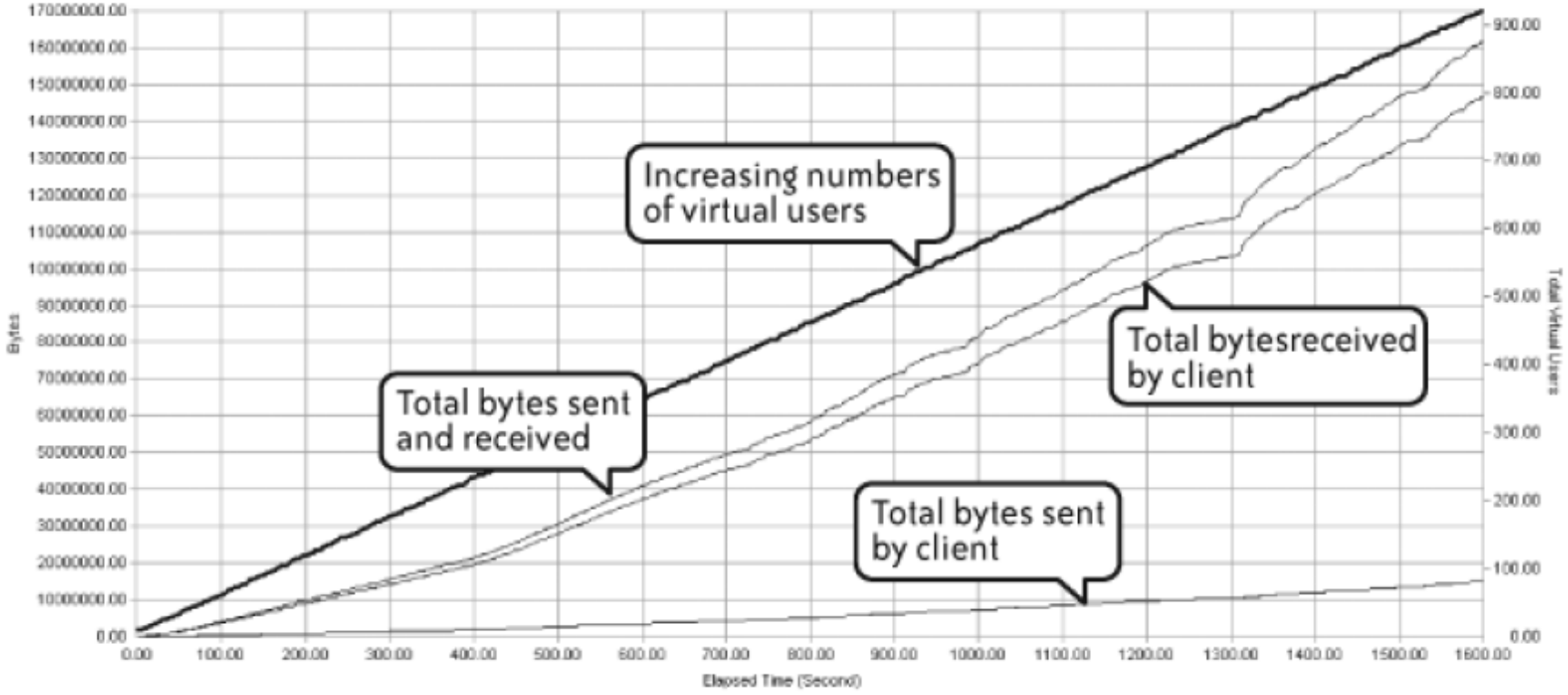
the number of concurrent virtual users against how busy the server CPU is

Figure 4-10. Concurrent virtual users correlated with database CPU utilization



categories of data presented to the network

Figure 4-11. Network byte transfers correlated with concurrent virtual users

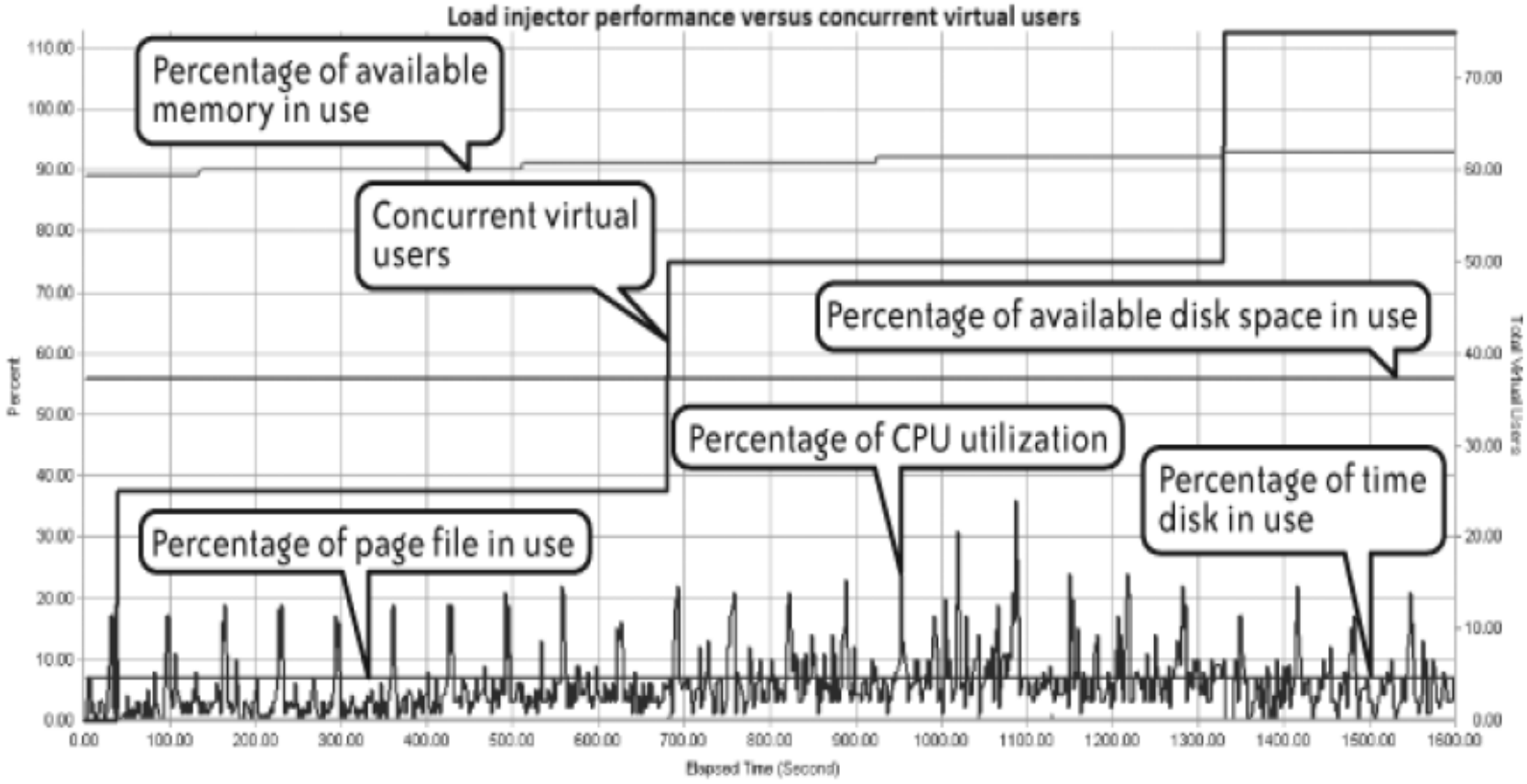


In this example it's pretty obvious that a lot more data is being received than sent by the client, suggesting that whatever caching mechanism is in place may not be optimally configured.

Monitor load injectors

- Typical metrics you need to monitor include:
  - Percent of CPU utilization
  - Amount of free memory
  - Page file utilization
  - Disk time
  - Amount of free disk space

Monitor load injectors



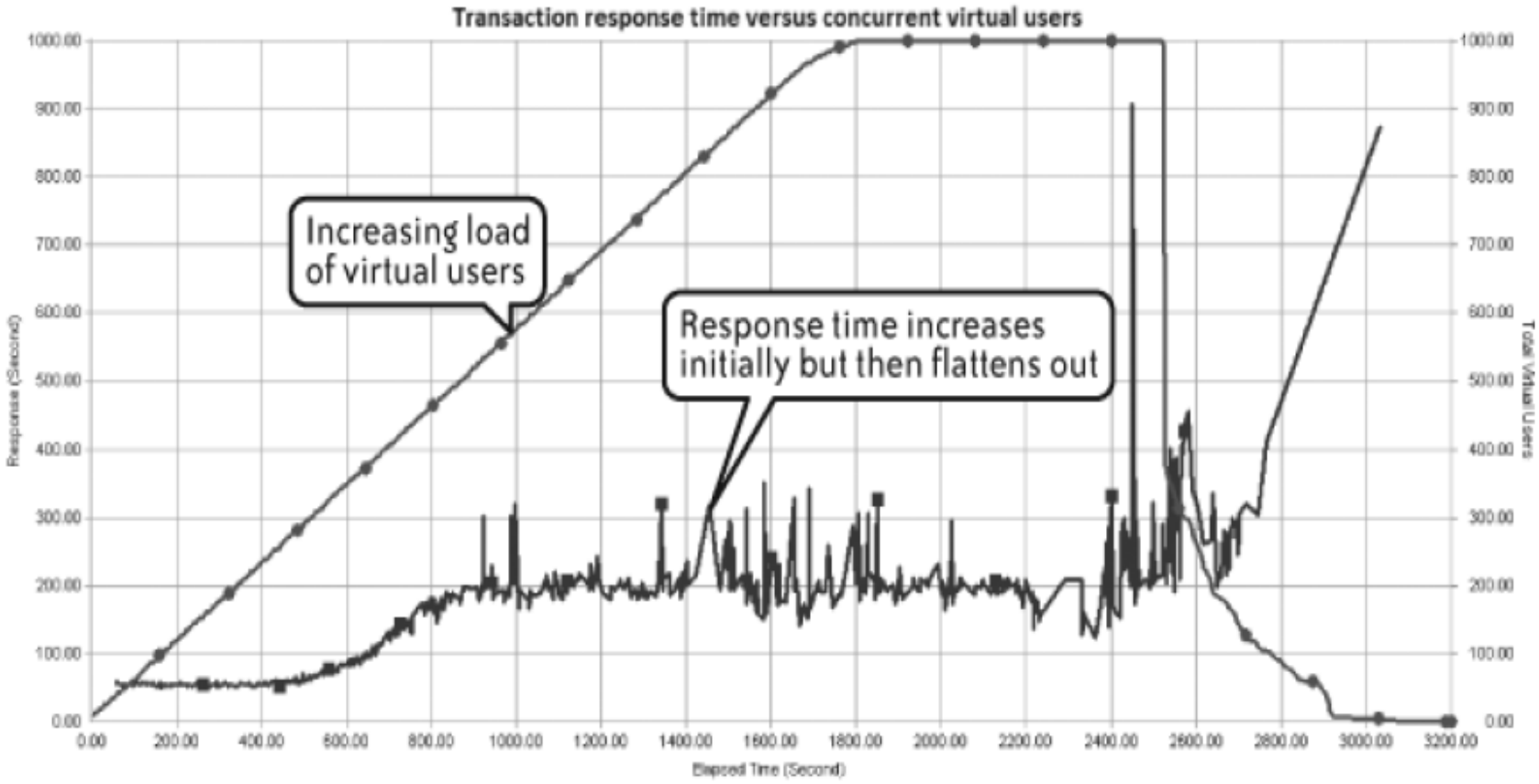
Root-Cause Analysis

- A good model of scalability and response time demonstrates a moderate but acceptable increase in mean response time as virtual user load and transaction throughput increase.

- A poor model exhibits quite different behavior: as virtual user load increases, response time increases in lockstep and either does not flatten out or starts to become erratic, exhibiting high standard deviations from the mean.

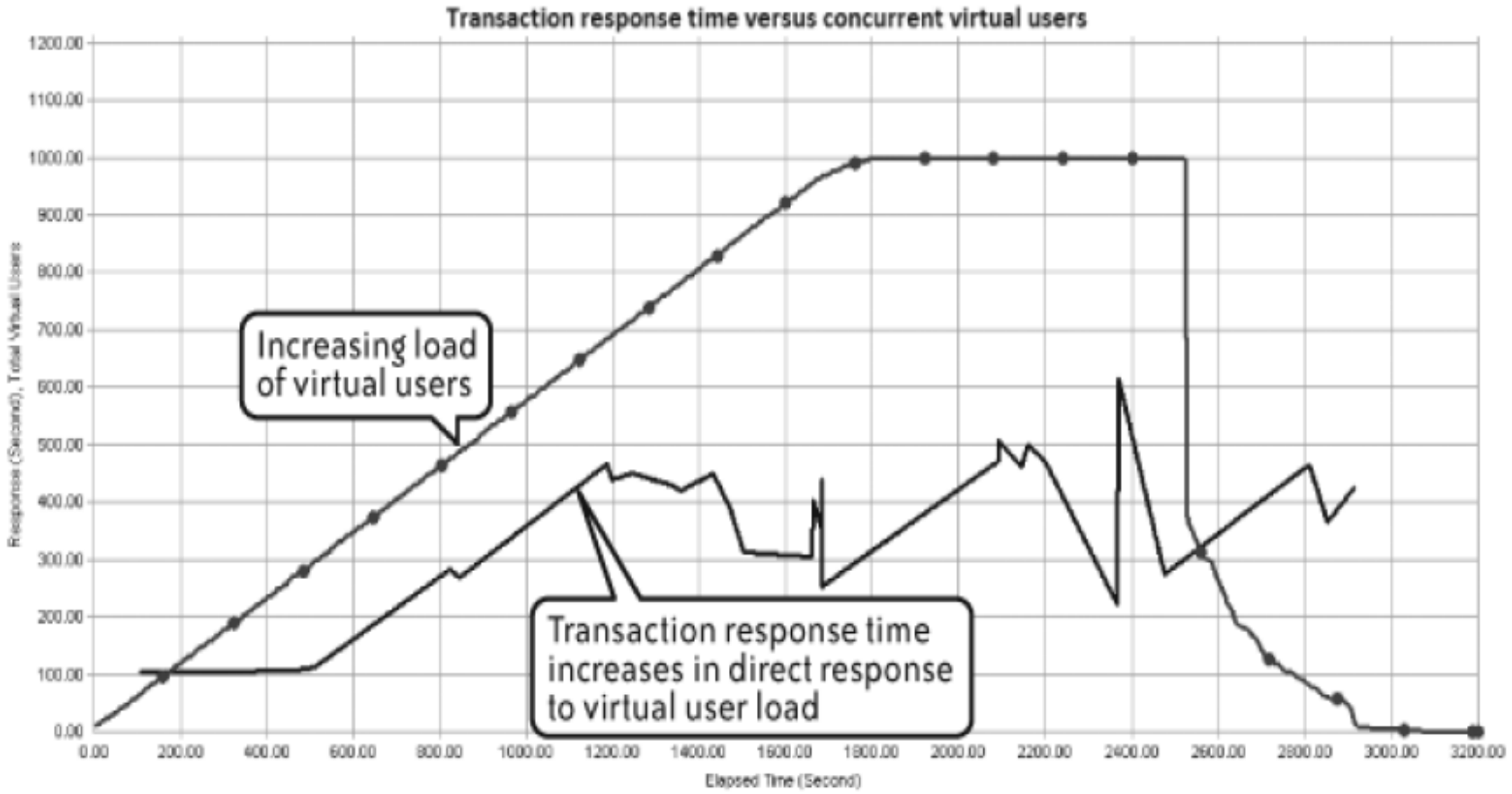
Sample of good scalability

Figure 4-13. Good scalability/response time model

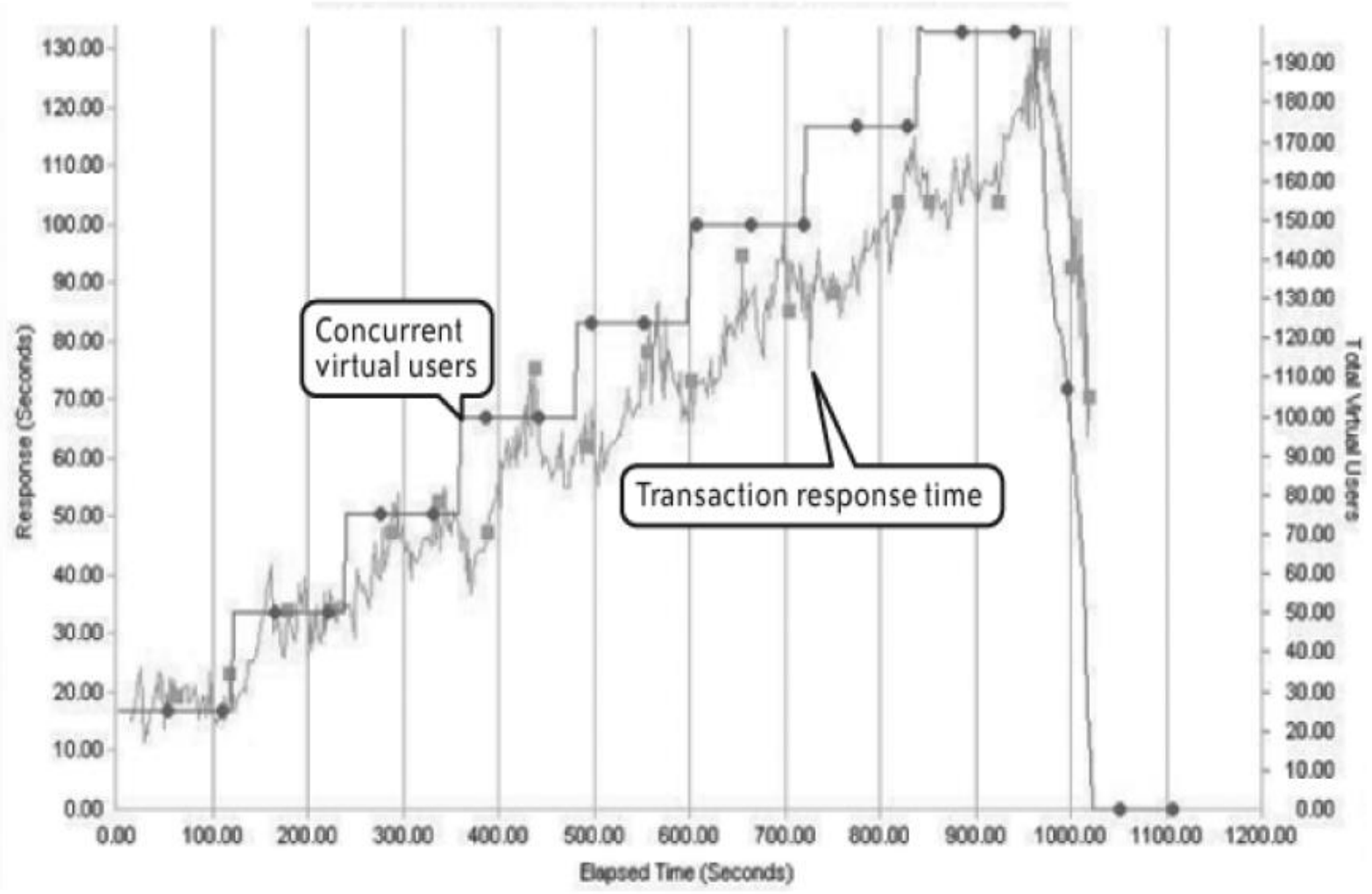


Sample of poor scalability

Figure 4-14. Poor scalability/response time model

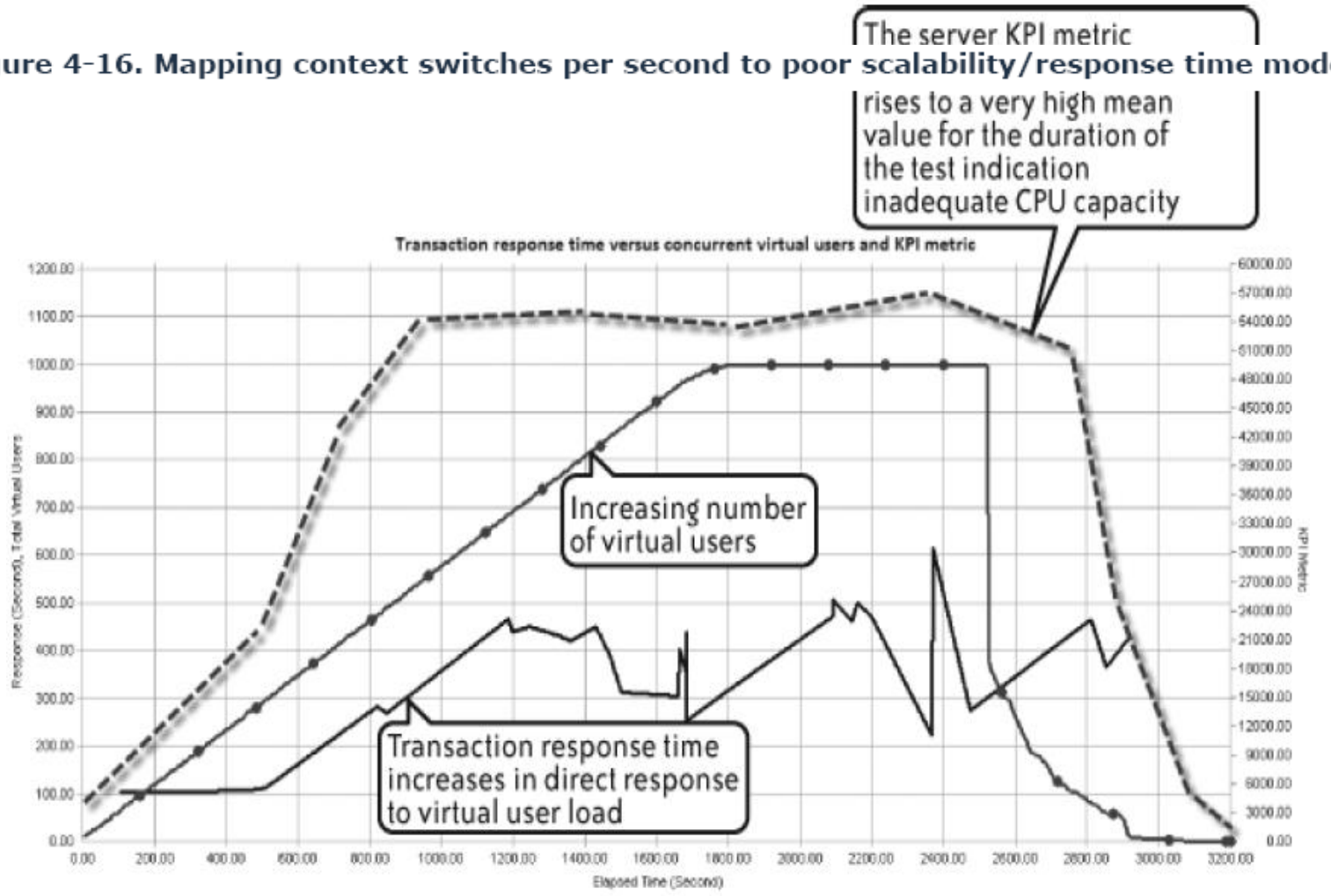


Sample of poor scalability



Poor Scalability (Continue)

Figure 4-16. Mapping context switches per second to poor scalability/response time model



Application server monitoring

- Simple generic monitoring of application servers won't tell you much if there are problems in the application.
- In the case of a Java-based application server in a Windows environment, all you'll see is one or more java.exe processes consuming a lot of memory or CPU.
- You may also run into the phenomenon of the stalled thread, where an application server component is waiting for a response from another internal component or from another server such as the database host. When this occurs, there is usually no indication of excessive CPU or memory utilization, just slow response time. The cascading nature of these problems makes them difficult to diagnose without detailed application server analysis

- Java application server's worst-performing SQL calls
- Java application server's worst-performing methods

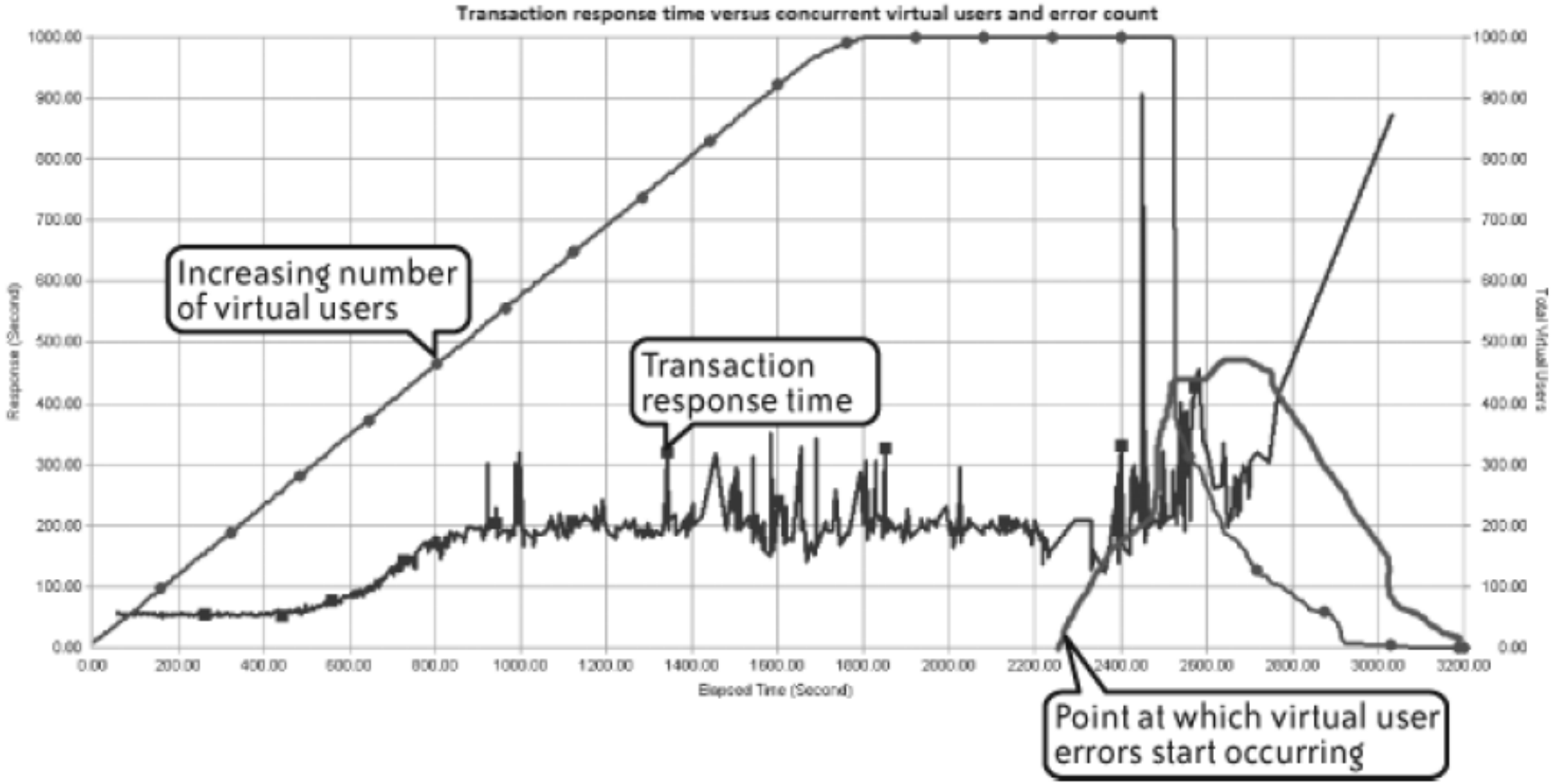
Application server's worst-performing methods

Figure 4-18. Java application server's worst-performing methods

Class Name	Method Name	Type	Last 90 Sec Avg Resp Time (sec)	Overall Avg Resp Time (sec)	Longest Resp Time (sec)	Total Invocation Cou
.*BNTNET	itemlist_aspx(*)	ASP	0.085	0.085	6.874	133
BNTNETWebApp.itemlist	Page_Load(object,EventArgs)	.NET	0.084	0.084	6.843	127
System.Data.OleDb.OleDbCommand	ExecuteReader(CommandBehavior)	SQL Query	0.046	0.046	6.796	217
.*BNTNET	account_aspx(*)	ASP	0.108	0.108	6.069	132
BNTNETWebApp.account	Page_Load(object,EventArgs)	.NET	0.105	0.105	5.610	129
.*BNTNET	default_aspx(*)	ASP	0.039	0.039	1.516	81
System.Web.UI.Page	ProcessRequest(HttpContext)	.NET	0.024	0.024	0.468	90
ASP.default_aspx	ProcessRequest(HttpContext)	.NET	0.031	0.031	0.468	40
.*BNTNET	myorders_aspx(*)	ASP	0.019	0.019	0.453	50
BNTNETWebApp.myorders	Page_Load(object,EventArgs)	.NET	0.019	0.019	0.421	47
.*BNTNET	WebResource_axd(*)	ASP	0.025	0.025	0.343	24
.*BNTNET	itemlink_aspx(*)	ASP	0.023	0.023	0.343	30
.*BNTNET	login_aspx(*)	ASP	0.018	0.018	0.314	33
ASP.itemlink_aspx	ProcessRequest(HttpContext)	.NET	0.021	0.021	0.311	25
.*BNTNET	checklogin_aspx(*)	ASP	0.024	0.024	0.299	30
BNTNETWebApp.checklogin	Page_Load(object,EventArgs)	.NET	0.023	0.023	0.283	28
ASP.itemlist_aspx	BeginProcessRequest(HttpContext,AsyncCallback,object)	.NET	0.060	0.060	0.267	5
System.Data.Common.DbDataAdapter	Fill(DataSet,string)	SQL Query	0.022	0.022	0.266	44
ASP.login_aspx	ProcessRequest(HttpContext)	.NET	0.017	0.017	0.204	25
BNTNETWebApp.Global	session_start()	.NET	0.024	0.024	0.187	22

Dealing with Errors

Figure 4-20. Example of errors occurring during a performance test



Monitoring & Analysis Checklist

- Make sure that you have configured the appropriate server, application server, and network KPIs. If you are planning to use installed agents instead of remote monitoring, make sure there will be no obstacles to installing and configuring the agent software on the servers.

- If your performance testing tool provides the capability, set any automatic thresholds for performance targets as part of your performance test configuration. This capability may simply count the number of times a threshold is breached during the test, and it may also be able to control the behavior of the performance test as a function of the number of threshold breaches that occur—for example, more than ten breaches of a given threshold could terminate the test.

Analysis Checklist (continue)

- If your performance testing tool provides the capability, configure autocorrelation between transaction response time, concurrent virtual users, and server or network KPI metrics.

- If you are using third-party tools to provide some or all of your KPI monitoring, make sure that they are correctly configured before running any tests.

Things to look out for during execution

- The sudden appearance of errors
  - some limit has been reached within the application landscape
  - reduce the number of active users for a troublesome transaction
  - Sudden errors can also indicate a problem with the operating system's default settings
- A sudden drop in transaction throughput
- An ongoing reduction in available server memory
- Panic phone calls from infrastructure staff

Application Technology and Its Impact on Performance Testing

- Asynchronous Java and XML (AJAX)
- HTTP Protocol
- Java
- Oracle

Asynchronous Java script and XML (AJAX)

- Most automated performance tools find this sort of technology difficult to handle.
- automated test tools are by default designed to work in synchronous fashion.
- It becomes a challenge to match requests with the correct responses.
- From a programming standpoint, the performance tool must spawn a thread that waits for the correct response while the rest of the script continues to execute

HTTP Protocol

- Http protocol
- Caching behavior
- SSL
  - Certificates
  - Increased overhead

Java

- Because of the component-based nature of Java applications, it is vital to have access to the internal workings of the application server under load and stress.
- The following are some common problems that cannot easily be detected without detailed monitoring at the component and method level:
  - Memory leaks
  - Excessive object creation
  - Stalled threads
  - Slow SQL

Oracle

- ODBC
- OCI
- UPI
- Oracle Forms Server (OFS)
- ORACLE provides a browser plug-in called JInitiator that manages the connection to the Forms Server, which is a free download from the Oracle web site.