




**>
accenture**

High performance. Delivered.

System Performance Bottleneck

Jothi Gouthaman
Accenture Delivery Center for Technology in India

Copyright © 2008 Accenture All Rights Reserved. Accenture, its logo, and High Performance Delivered are trademarks of Accenture.



Introduction

and

Setting the Context

Copyright © 2008 Accenture All Rights Reserved.

2

Agenda



- Performance Testing
- Performance Engineering
- Bottlenecks
- Profiler and Profiling applications
- Bottleneck Identification Approach
- Inferring Performance Counter
- Performance Laws

Copyright © 2008 Accenture All Rights Reserved.

3

Performance Testing



Performance Testing is the discipline concerned with determining and reporting the current performance of a software application under various parameters

Performance Testing Types <ul style="list-style-type: none"> • Load & Stress 	Performance Test approaches <ul style="list-style-type: none"> • Scalability & Stability
Performance Testing terminologies <ul style="list-style-type: none"> • Scenario /Business Flow • Operational Profile • Work load 	Performance Metrics <ul style="list-style-type: none"> • Client side & Server side

Copyright © 2008 Accenture All Rights Reserved.

4

Performance Engineering



Performance engineering is the process by which software is tested and tuned with the intent of realizing the required performance.

Primarily used for...

- pro-active bottleneck identification as part of SDLC
- "fixing" systems that are not meeting requirements/SLAs
- extending the capacity of old systems

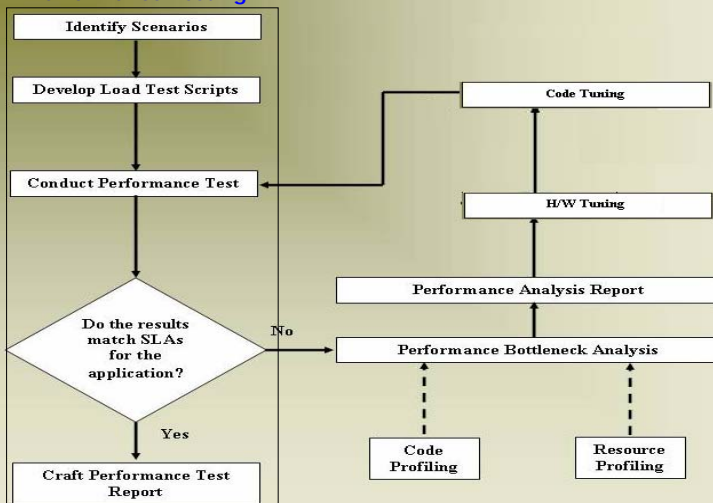
Copyright © 2008 Accenture All Rights Reserved.

5

Performance Engineering Flow



Performance Testing




6



Bottleneck

Copyright © 2008 Accenture All Rights Reserved.

7



Bottleneck

“ In a given time instance only one bottleneck cause the system to under perform”

“ Strength of the chain is equal to the weakest link”

Copyright © 2008 Accenture All Rights Reserved.

8

Bottleneck



- A Bottleneck can be any factor that prevents the system in meeting the performance target
- It is a resistance to the flow of data
- Bottleneck can be within any layer of the technology stack or the infrastructure



Copyright © 2008 Accenture All Rights Reserved.

9

Bottleneck Classification



Software

Application Layer	Database Layer
<ul style="list-style-type: none"> • Objects • Methods 	<ul style="list-style-type: none"> • Query • Locks

Hardware

Processor	Memory
I/O Layer	Network Layer

Copyright © 2008 Accenture All Rights Reserved.

10

Application Bottleneck



- **Single-threaded**

- Even Forking application are not efficient
 - Consume a single CPU even in availability of more CPUs.

- **Orphan and Loitering objects**

- Degrade performance by retaining memory from the operating system and increasing the activity of the memory management subsystem within the JVM

Database Bottleneck



- **Costly Queries**

- Out-of-date statistics
- Lack of useful indexes
- Lack of useful data extracting.

- **Table indexing**

- Occupies more space
- Increase SELECT but slows down INSERT and UPDATE

Database Bottleneck



- **Scans**
 - Large-table full-table scans increase the load on the disk I/O sub-system
 - Small table full table scans results in CPU consumption
- **Parent and Child Calls**
 - Caching of Call results

Copyright © 2008 Accenture All Rights Reserved.

13


Tuning Level



Tuning Level	Goals	Focus Areas	Improvement
High: System-level	Improve application interaction	<ul style="list-style-type: none"> • Network Problems • Disk Performance • Memory Usage 	3X
Medium: Application-level	Improve Application Algorithm	<ul style="list-style-type: none"> • Locks • Heap contention • Threading Algorithm • APIs Usage 	2X
Low: Micro Architecture-level	Improving how the application runs on the processor	<ul style="list-style-type: none"> • Architecture Coding Pitfalls • Data/Code Locality (Cache) • Data Alignment 	1.1-1.5X

Copyright © 2008 Accenture All Rights Reserved.


14



Profilers and Profiling

Copyright © 2008 Accenture All Rights Reserved.

15



Profiling and Profilers

- Profiling is a process of analyzing the application from performance perspective by identifying the hotspot
- Profiler are tools to identify the hotspots by monitoring the critical functions and process
- Profiler helps to analyze the application flow and the code level interaction across functions
- Line-level timing
 - Variable declaration
 - Iterations

Copyright © 2008 Accenture All Rights Reserved.

16

Application Profiling



Objective: Bottleneck Identification

Approach :

- Sampling
- Call Graph

Focus Area:

- Class and Object Details
- Threads & Process
- Variable types
- Reference details

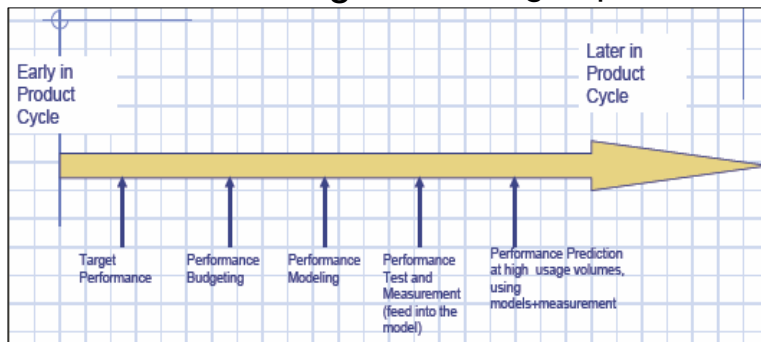
Bottleneck Identification Approach



SDLC



Performance testing as an integral part of SDLC



Copyright © 2008 Accenture All Rights Reserved.

19

Bottleneck Identification



Pro-active

- Historical analysis tools
- Capacity planning tools

Reactive

- Event Viewer – **Tell me before it is too late!**
- Performance Monitors – **More In-depth analysis**

Copyright © 2008 Accenture All Rights Reserved.

Source: www.sun.com

20

Test Approach



Single user Test

- Client side Vs Resource
- Components Vs Resource
- Data Volume Vs Client side Vs Resource

Scalability

- Load Vs Resource
- Load Vs Client side
- Load vs SLA

Test Approach



Stability Test


- Load Vs Duration
- Duration Vs Resource
- Load Vs Resource Load
- Load Vs Client side
- Load Vs SLA



Inferring Bottleneck

Copyright © 2008 Accenture All Rights Reserved.

23



Client Side

HTTP Status Code

- 1xx Informational
- 2xx Success
- 3xx Redirection
- 4xx Client Error
- 5xx Server Error

Copyright © 2008 Accenture All Rights Reserved.

24

Client Side Performance Metrics



A Counter is a unit that measures and gathers **performance**-relevant events of the system

Client Side Metrics

- Response time
- Throughput
- Hits Per Second
- Pass/Fail Statistics

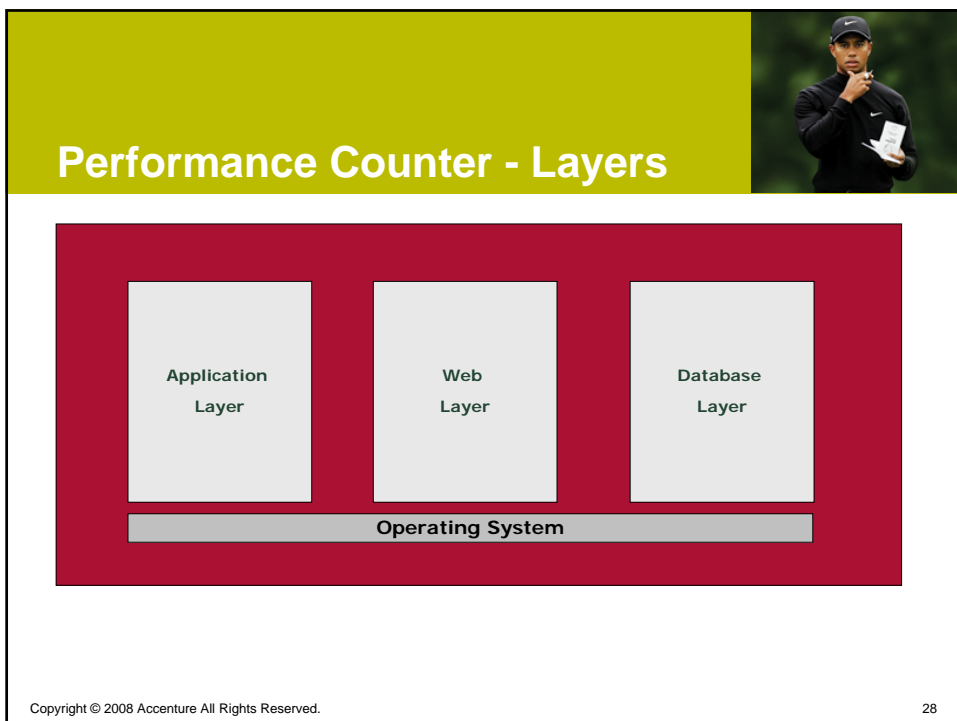
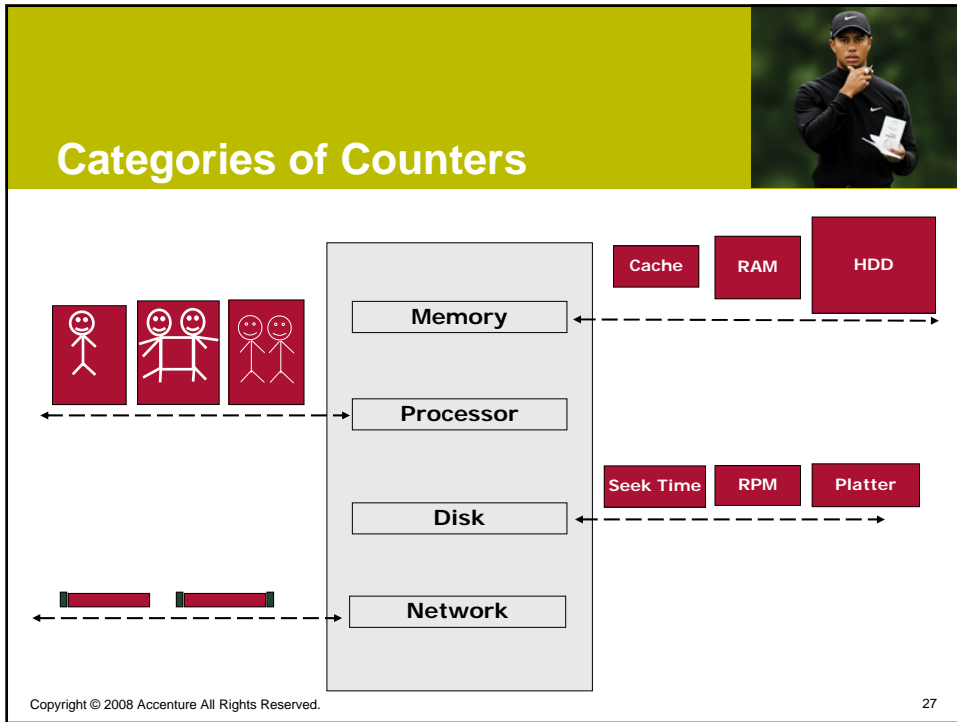
Server side Performance



A Counter is a unit that measures and gathers **performance**-relevant events of the system

Server Side Counters

- Memory
- Processor
- Disk
- Network



Operating System - Memory



Windows

- Memory\Available KBytes
- Memory\Committed Bytes
- Memory Page Read\Sec
- Memory Page Writes\Sec
- Memory Pages\Sec

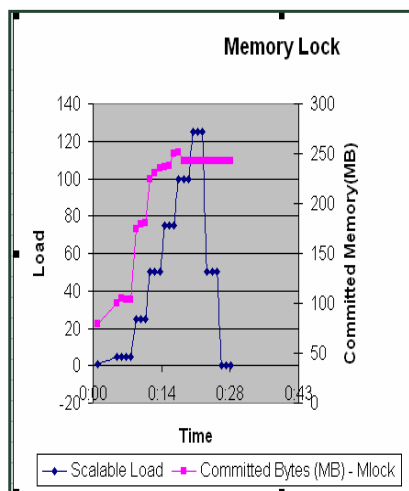
Linux

- Free Memory
- Page Ins
- Page Outs
- Page Faults/sec
- Major Page Faults/sec

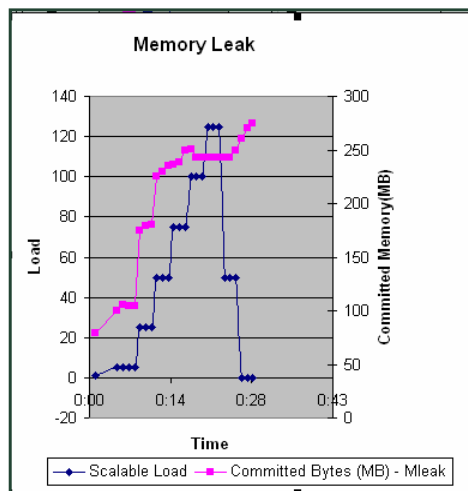
Copyright © 2008 Accenture All Rights Reserved.

29

Memory lock/leak - Inference



Copyright © 2008 Accenture All Rights Reserved.



30

Operating System - Disk



Windows

- Disk Queue Length
- Disk Read Bytes/sec
- Disk Reads/sec
- Disk Write Bytes/sec
- Disk Writes/sec

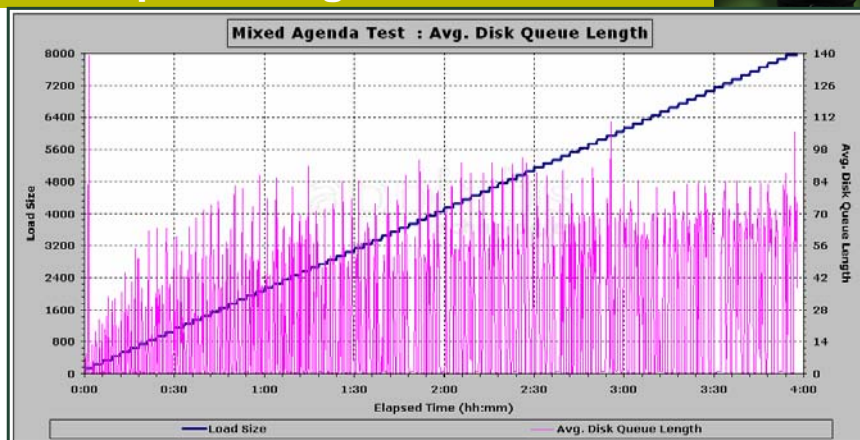
Linux

- Average Queue length
- Average Queue Time
- Average Request Service time
- %CPU when IO request was issued
- Reads/sec and Writes/sec

Copyright © 2008 Accenture All Rights Reserved.

31

Disk queue length - Inference



- Disc Queue increased with load
- Attributed towards increase in disk write queue length

Copyright © 2008 Accenture All Rights Reserved.

32

Operating System - Processor



Windows

- % Idle Time
- % Interrupt Time
- % Processor Time
- % User Time
- % Privileged Time

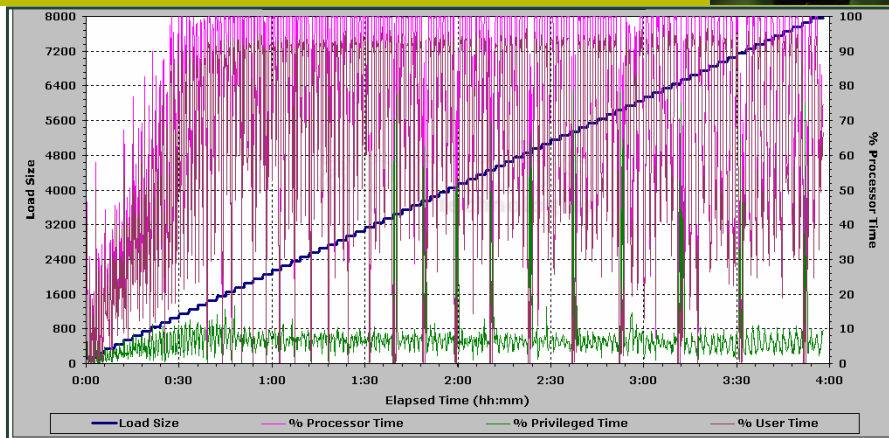
Linux

- System process%
- User process%
- IOWait process%
- Context Switches/sec
- Load Average
- number of processes waiting for run time
- number of processes in uninterruptible sleep.
- RunQueue size
- Top process

Copyright © 2008 Accenture All Rights Reserved.

33

Processor - Inference



Copyright © 2008 Accenture All Rights Reserved.

34

Operating System - Network



Windows

- Bytes Received/Sec
- Bytes Sent/sec
- Bytes Total/sec
- Packets Received/sec
- Packets Sent/sec

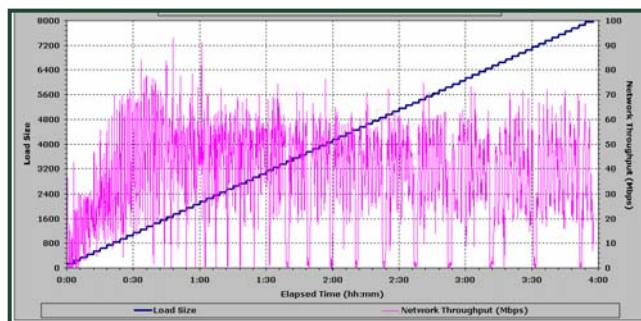
Linux

- Active connections
- Passive Connections
- Connections Established
- Connection resets
- Sent bytes
- Received Bytes

Copyright © 2008 Accenture All Rights Reserved.

35

Network - Inference



- Network Throughput increased with the increase in load and reached around 70 Mbps at the load of around 1350 users
- It then fluctuated around 40 Mbps for the rest of the run duration
- Maximum throughput was around 92 Mbps during the test run
- The network Throughput reached the maximum of 92 Mbps once at the load of 1650 users
- Average throughput was around 38 Mbps during the test run

Copyright © 2008 Accenture All Rights Reserved.


36



Performance Law

Copyright © 2008 Accenture All Rights Reserved.

37



Performance Laws

- **Little's Law**
 - The length of Queue is equal to the arrival rate (A) multiplied by the Residence time $L = A * W$
- **Utilization Law**
 - $U = (B/T) \rightarrow (B/C) * C/T \rightarrow \text{Throughput (T)} * \text{Service Time(S)}$
- **Forced Flow Law**
 - Queue Throughput (Xq) = Average Number Visits * Total System Throughput (X)
- **Service Demand Law**
 - Ratio of Utilization of the Queue/Total System throughput = (U/X)

Copyright © 2008 Accenture All Rights Reserved.

38

Performance Laws



Amdahl's law

- Amdahl's law is used to find the maximum expected improvement to an overall system when only part of the system is improved. It is often used in parallel computing to predict the theoretical maximum speedup using multiple processors
- Formula to calculate the improvements with Amdahl's law is:

$$1 / (S + (1-S)/N)$$
 where
 S - is how much of the process cannot be done in parallel to all other processes of the same application
 N - the number of processors
- As the amount of sequential code increases, the maximum advantage can be achieved with fewer and fewer processors - in other words, highly serial code is improved less and less for each processor that is added to it.
- This law shows that it is indeed the algorithm and *not* the number of processors which limits the speedup.

Copyright © 2008 Accenture All Rights Reserved.

39

Performance Laws




Amdahl's Law (Few numbers..)

S (Percent of sequential code)	N (Number of Processors)	Improvement (as times)
10 % (0.1)	5	3.57
10 % (0.1)	10	5.26
10 % (0.1)	20	6.90
10 % (0.1)	100	9.17
10 % (0.1)	100,000	9.99 (~10)
25 % (0.25)	5	2.50
25 % (0.25)	10	3.08
25 % (0.25)	20	3.48
25 % (0.25)	100	3.88
25 % (0.25)	100,000	3.99 (~4)
40 % (0.40)	5	1.92
40 % (0.40)	10	2.17
40 % (0.40)	20	2.33
40 % (0.40)	100	2.46
40 % (0.40)	100,000	2.5

Copyright © 2008 Accenture All Rights Reserved.


40



To Check....

Copyright © 2008 Accenture All Rights Reserved.

41



Front end Bottlenecks

- Make Fewer HTTP Requests
- Use a Content Delivery Network
- Add an Expires Header
- Gzip Components
- Put Style sheets at the Top
- Put Scripts at the Bottom
- Avoid CSS Expressions
- Make JavaScript and CSS External

Copyright © 2008 Accenture All Rights Reserved.

Source: Yahoo Developer Network

42

Front end Bottlenecks



- Reduce DNS Lookups
- Minify JavaScript
- Avoid Redirects
- Remove Duplicate Scripts
- Configure ETags
- Make Ajax Cacheable

Copyright © 2008 Accenture All Rights Reserved.

Source: Yahoo Developer Network

43



**>
accenture**

High performance. Delivered.

Thank You!

jothi.gouthaman@accenture.com

Copyright © 2008 Accenture All Rights Reserved. Accenture, its logo, and High Performance Delivered are trademarks of Accenture.