



Root Cause Analysis of CPU Performance Bottlenecks

*Presenter:
Sreekanth Makam S
Sr. Performance Engineer
Symphony Services Corp(India) Pvt Ltd.*

Agenda

- Background
- Problem Statement
- Misconceptions on CPU performance bottlenecks
- Methodology used
- Solution – Step by Step

- **Background**
- Problem Statement
- Misconceptions on CPU performance bottlenecks
- Methodology used
- Solution – Step by Step

Background

- The customer business is changing rapidly day to day.
- The business is growing to very high-unexpected loads.
- The technology is also changing rapidly and becoming more and more complex.

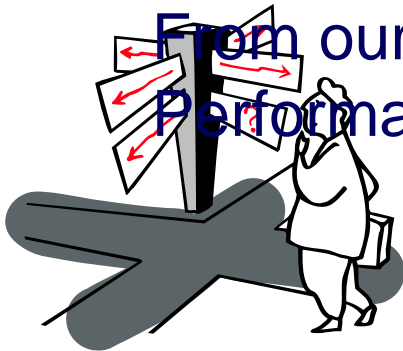
All these are challenging the IT industry with respect to **PERFORMANCE.**

Background – Cont'd

The problem can be anywhere !!!!



From our experience we found that most of the performance bottlenecks faced are related to CPU only.



- Background
- **Problem Statement**
- Misconceptions on CPU performance bottlenecks
- Methodology used
- Solution – Step by Step

Problem Statement

Hot to find the root cause of all the CPU performance bottlenecks ?

In fast, efficient and cost effective manner

- Manually troubleshooting J2EE application to find CPU performance bottlenecks and isolate the performance problems involves a lot of guesswork making it a very cumbersome, slow and tedious process.
- Getting to the root of CPU performance problems in the complex, distributed, and dynamic J2EE environment is truly a challenge.

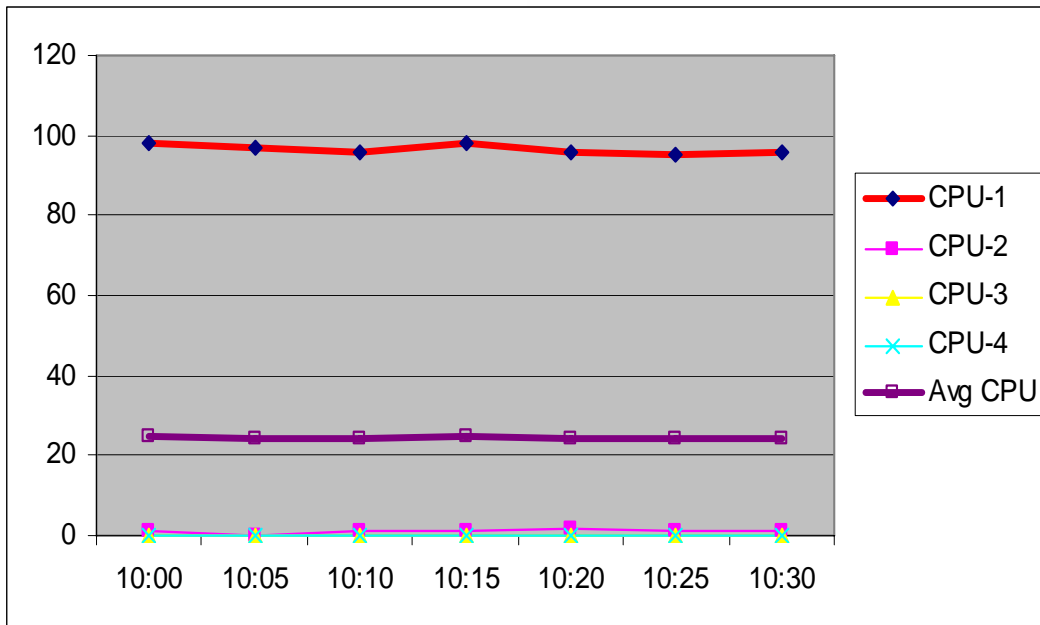
- Background
- Problem Statement
- **Misconceptions on CPU performance bottlenecks**
- Methodology used
- Solution – Step by Step

Misconceptions

1. CPU Util \approx 100%, then CPU bottleneck exists.
2. If the CPU utilization is not increasing with the load then application is working well.

Misconceptions – Cont'd

- If Average CPU utilization is less, then the chance of CPU bottleneck is 0%.



Case Study:

Even Avg CPU Util < 25%,

This is an best example for CPU performance bottleneck as CPU-1 alone is over loaded.

Misconceptions – Cont'd

Try all your magic bullets to solve the bottleneck like...

- Increase Application server's threads
- Increase Web server's threads or server processes.
- Increase Database Pool size.
- Increase the JVM heap size to maximum possible and use parallel garbage collection policy.
- Increase all DB instance parameters such as buffer size, sort area etc.,
- **Resize the Hardware.**
- **Upgrade the software.**

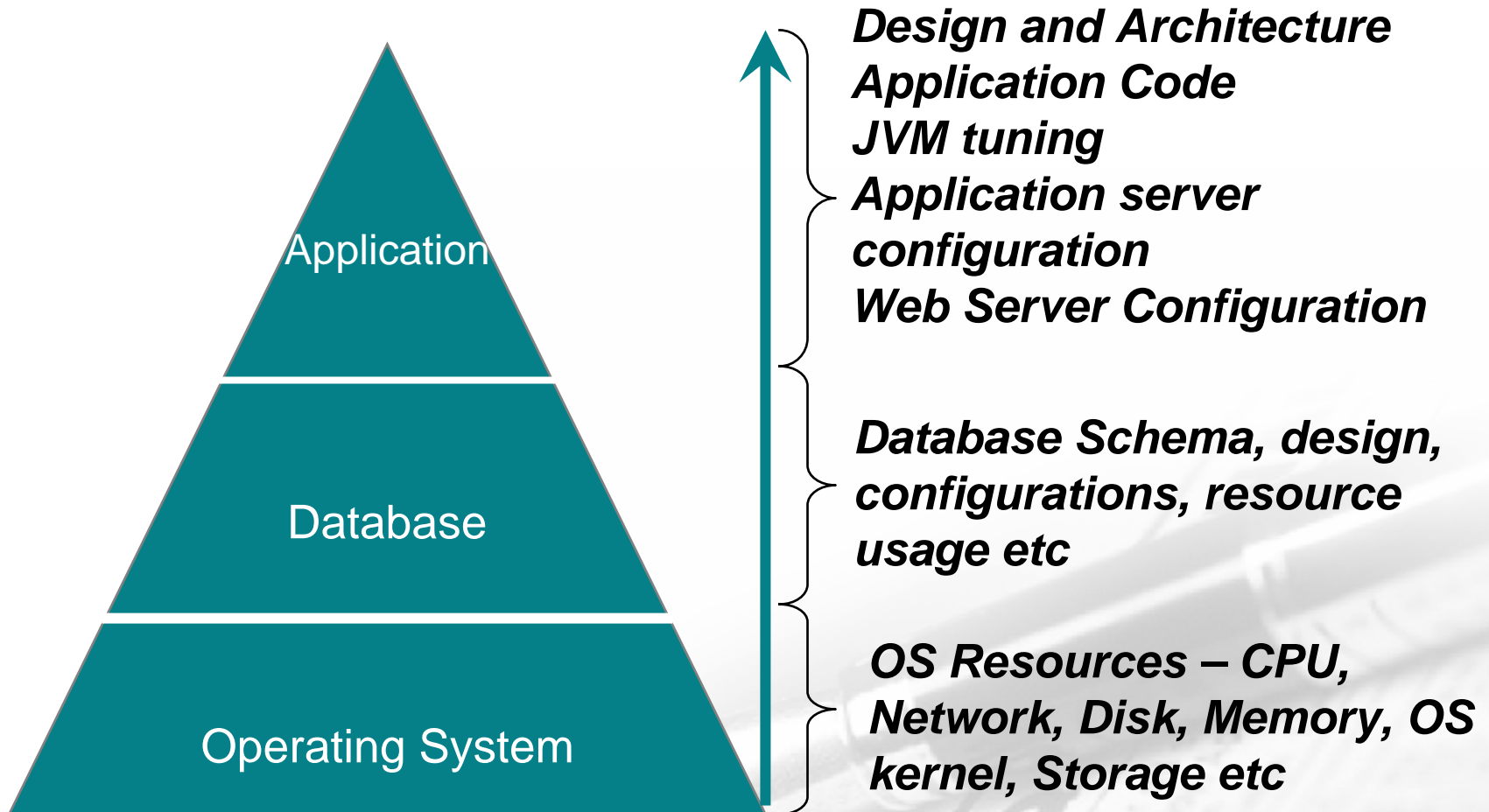
Misconceptions – Cont'd

- **Case Study:** To show the effect of tuning with out the knowledge of the problem.

	4 CPU / Windows 2000 / Weblogic appserver / Jrocket 1.4 JVM / 30 threads	4 CPU / Windows 2000 / Weblogic appserver / Jrocket 1.4 JVM / 100 threads
App Server CPU Total Util	63%	64%
App server Memory Usage	40%	55%
Avg Response time(sec)	1.21	1.20
HITS per Sec	120	119
Users	500	500

- Background
- Problem Statement
- Misconceptions on CPU performance bottlenecks
- **Methodology used**
- Solution – Step by Step

Methodology



- Background
- Problem Statement
- Misconceptions on CPU performance bottlenecks
- Methodology used
- **Solution – Step by Step**

Solution

20:80 RULE

→ Tuning 20%, 80% performance can be archived

Finding that 20% is the goal of “**ROOT CAUSE ANALYSIS**”

Does CPU bottleneck exist?

CPU bottlenecks can be classified in two categories...

CPU Over utilization

If CPU Util > 95% and Run Queue > Number of CPUs

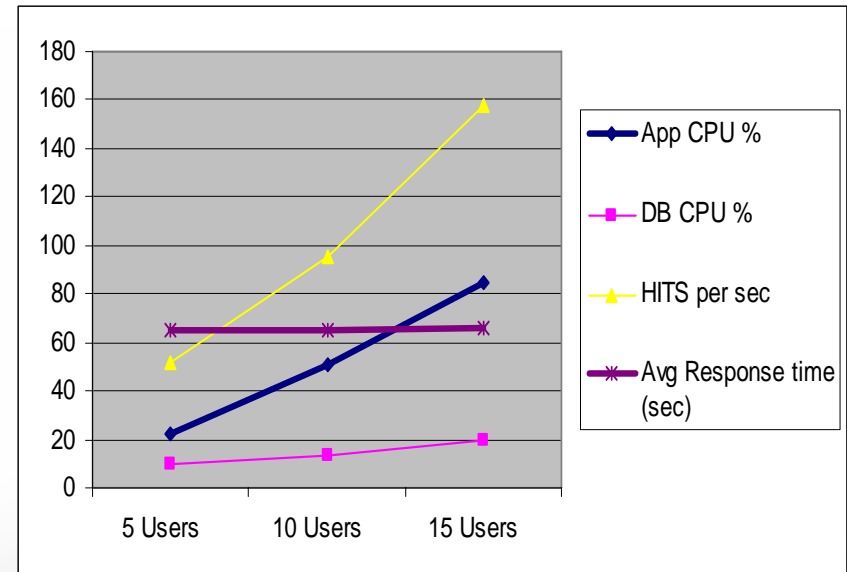
CPU Under utilization

If CPU Util is not increasing linearly with the load and response time is degrading

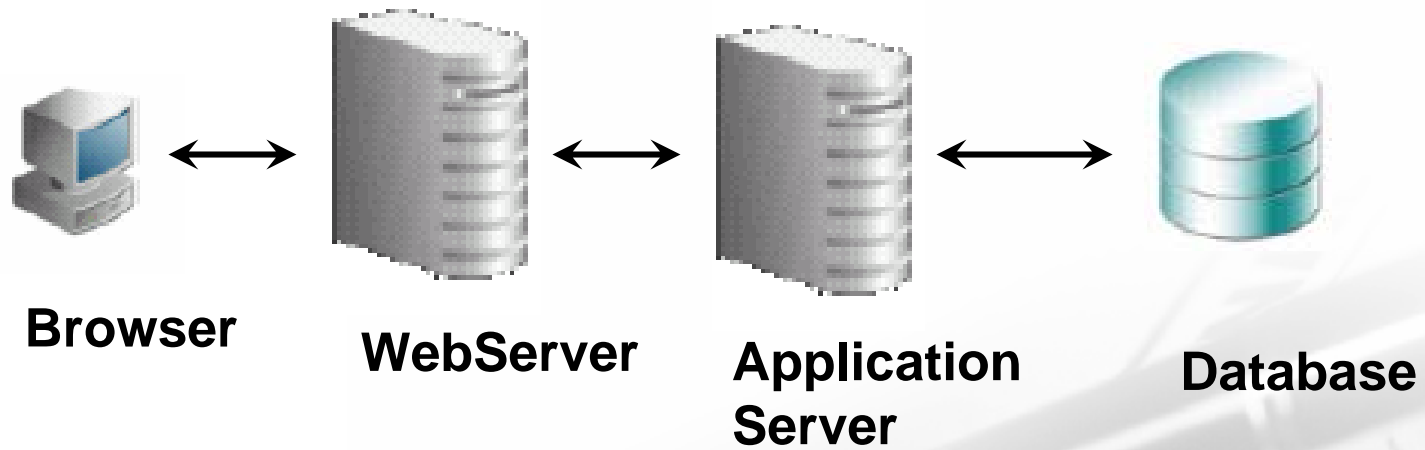
Does CPU bottleneck exist?

Case Study 3: The application with no CPU bottleneck

	5 Users	10 Users	15 Users
App CPU	22.50%	51%	85%
DB CPU	10%	13%	20%
HITS per sec	52	95	158
Response time (sec)	0.65	0.65	0.66
Throughput	1MBPS	1.8 MBPS	3 MPBS



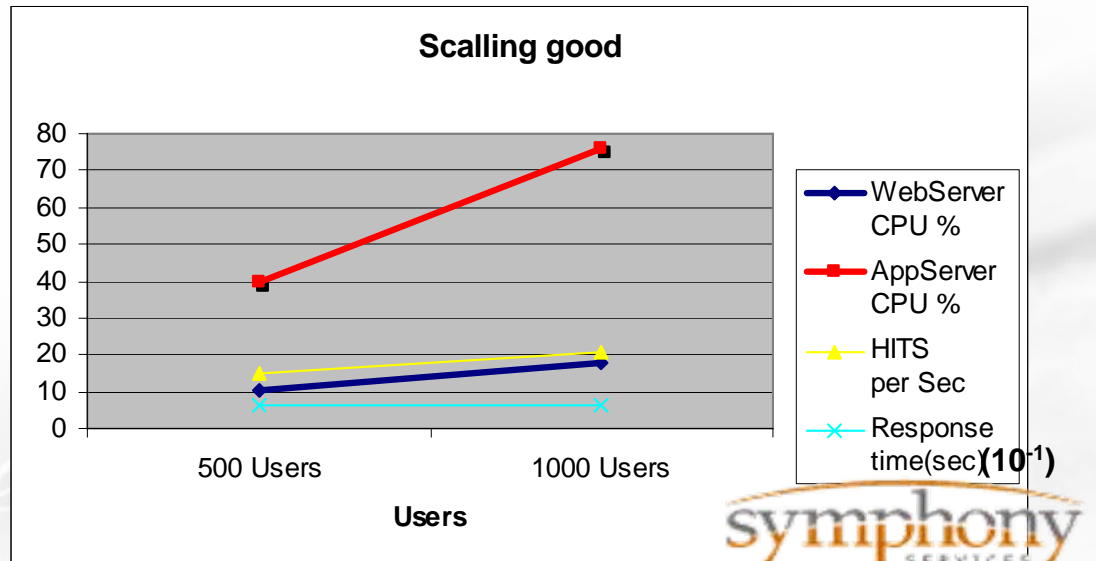
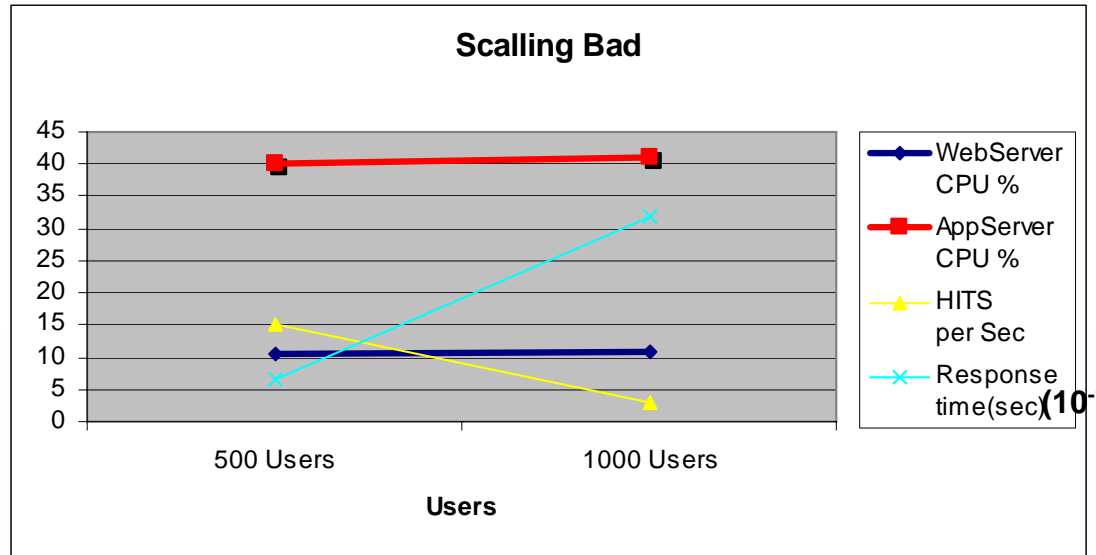
Which layer has the problem?



WebServer tuning

Most common problem that is observed in Webserver is that CPU utilization will not increase with load.

Case Study: The effect of ThreadsPerchild directive in httpd.conf



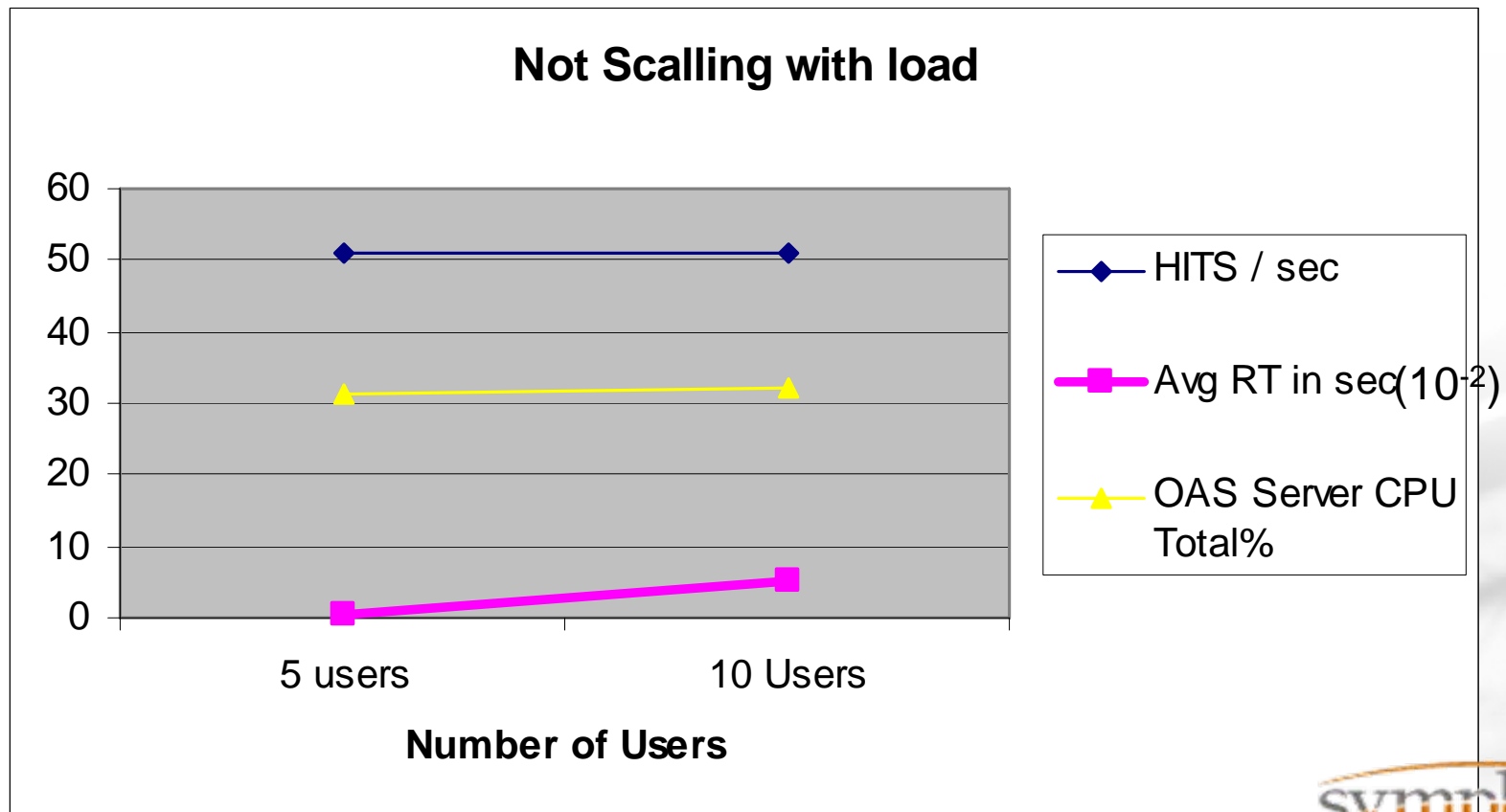
Application server Tuning

Thread dump analysis

- If CPU util is under utilized and the response time is increasing then the best way analyse the problem is thread dump analysis.
- Thread dump analysis is best way to find the lock contentions and deadlocks in threads.

Thread dump analysis

Case Study: The CPU utilization is not increasing with the load.



Thread dump Analysis

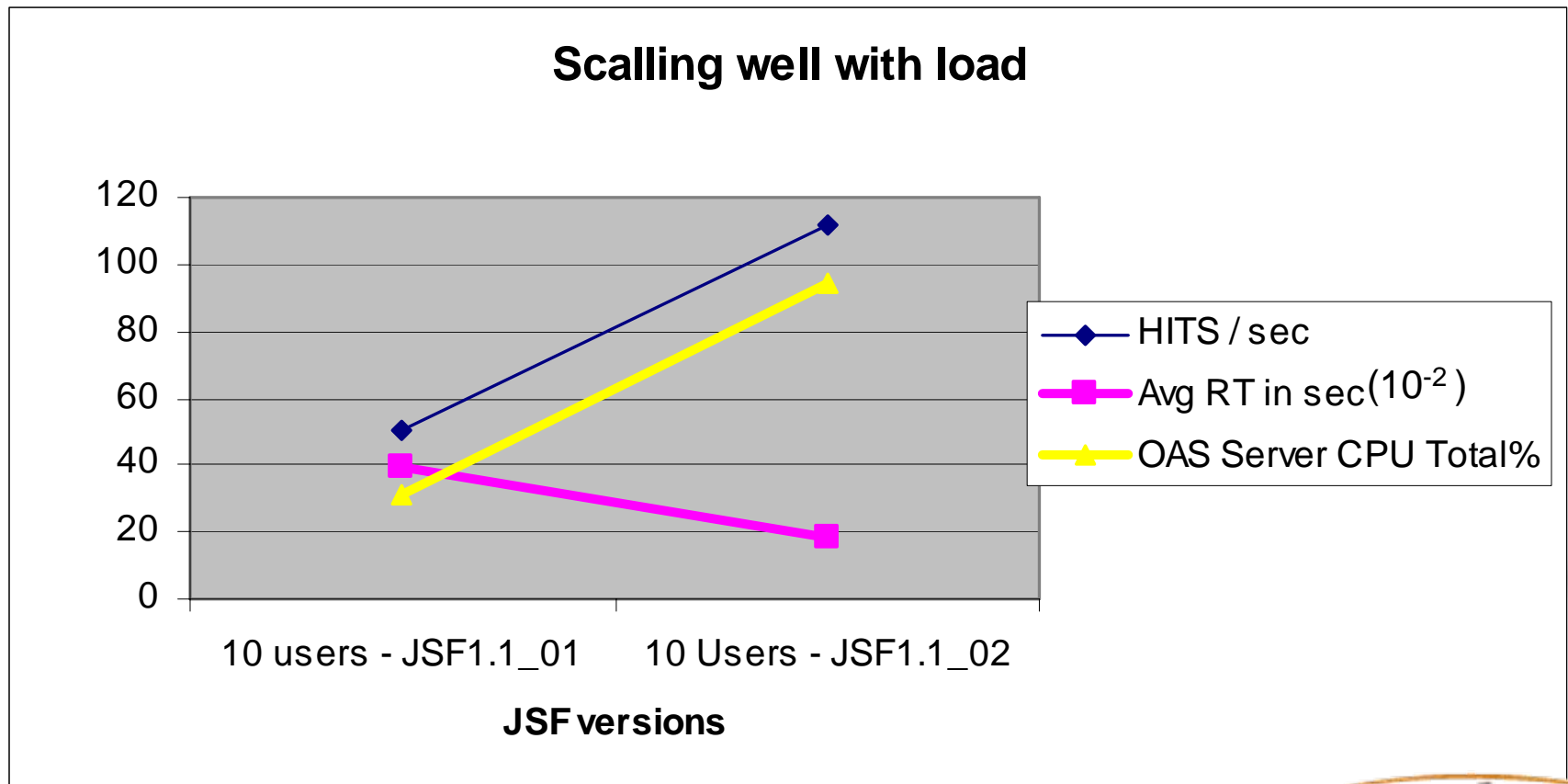
Thread dump was generated using JRocket management console.

Observations from thread dump

1. For 10 users run there are 9 blocked chains and for 20 users there are 18 blocked chains and for 30 users there are 29 blocked.
2. All the blocked threads are blocked for a fat lock in `com/sun/faces/lifecycle/LifecycleImpl.phase()` method.
3. From the `LifecycleImpl.java` it is clear that there was lot of synchronizations on arraylist object which was creating these fat locks.

Thread dump Analysis

Results after solving synchronization problem.



Thread pool tuning

Increase thread pool size until there is no further improvement in CPU utilization.

JVM tuning

- **Goal:** Total cycles spent on Garbage collection < 5% of total cycles used.
- JVM tuning includes both heap and stack tuning.
- Monitor the GC activity using `-Xverbosegc`.

DB pool tuning

Increase DB pool size until there is no further improvement in database machine CPU utilization.

EJB pool tuning

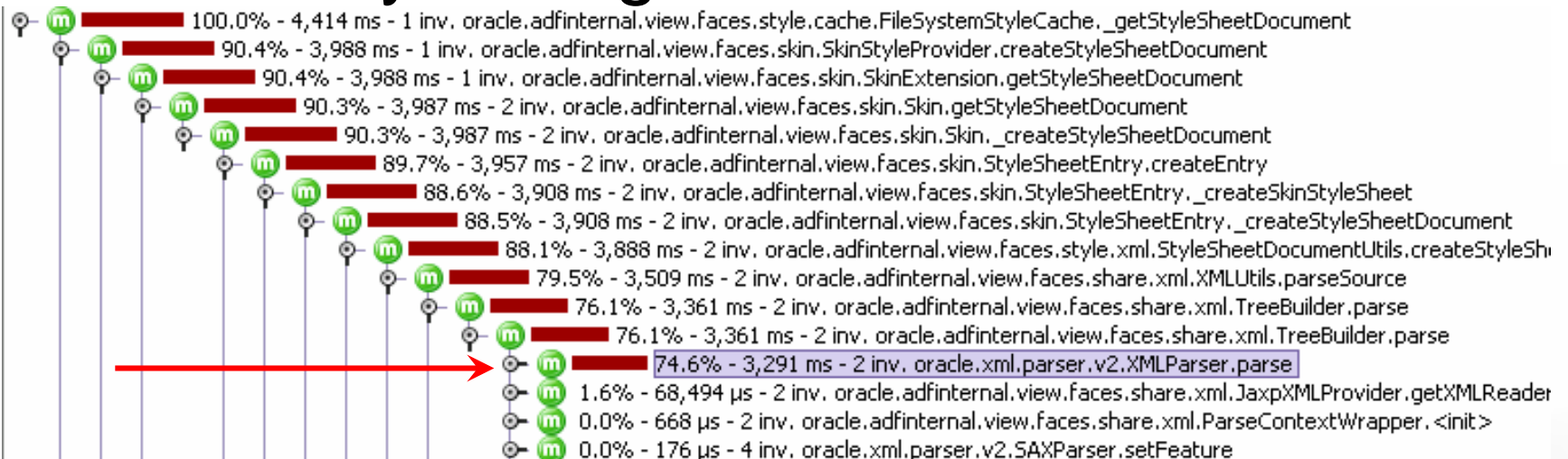
Maintaining stateless beans pool properly will always helps in getting good performance.

Application code tuning

- Profile the application with profiling tools such as JProfile, HPJMeter, etc
- Tune all most frequently executed methods
- Tune hot methods where most of the time is spent.

Application code tuning

Case Study: Using JProfile to find the



Analysis:

- 75% time is spent on parsing.
- The bottleneck can be oracle v2 parser or the xml file is big.

Solution:

- Try other parsers such as Xerces, Crimson etc.,
- Find why the XML document is big.

Database CPU bottlenecks tuning

Two main parts of Database machine CPU bottlenecks are...

- Tune SQLs
- Tune Database Instance

SQL Tuning

Case Study: CPU utilization is not increasing linearly with load

ADFBC App	300 Users	400 Users	500 Users
HITS / sec	39.183	51.5	56.44
Avg RT in sec	0.266	0.507	2.842
Avg Throughput Bytes/sec	745497	980381	1074248
Appserver CPU Total%	30.73%	38.14%	45.62%
Database Server CPU Total %	55.83%	76.00%	88.20%
DB Run Queue	0.71	4.95	33.66

SQL Tuning – Cont'd

After creating missing index in database,

ADFBC App	500 Users With out Index	500 Users With Index
HITS / sec	56.44	120
Avg RT in sec	2.842	0.3
Appserver CPU Total%	45.62%	60.30%
Database Server CPU Total %	88.20%	20.10%
DB Run Queue	33.66	0
Execution time of top SQL (sec)	1.6	0.08