Enabling Java-based VoIP backend platforms through JVM performance tuning

(Bruno Van Den Bossche, Filip De Turck, April 3rd 2006)

Outline

- ✓ Introduction
- ✓ Java 4 Telecom
- ✓ Evaluation Setup
  - ➢ Hardware
  - ➢ Software
- ✓ Java Virtual Machine
  - ➢ Default behavior
  - ➢ Possible Optimizations
  - ➢ Optimized Behavior
- ✓ Results
- ✓ Conclusions

Introduction

- ✓ Software backend platforms are increasingly popular in VoIP offerings.
- ✓ Java is currently one of the most popular programming languages for implementing business logic.
- ✓ Is Java suitable for implementing VoIP and telecom related applications?
- ✓ Java Application Servers for Telecom related applications are emerging:
  - ➢ SIP Servlet
  - ➢ JAIN SLEE

Telecom Applications: VoIP

- ✓ Telecom related applications have very specific requirements
  - ➢ Low Latency
  - ➢ High Throughput
- ✓ Example: Softswitch
  - ➢ How fast can it set up a call?
  - ➢ How many calls can in set up per second?
- ✓ Java 4 Telecom?

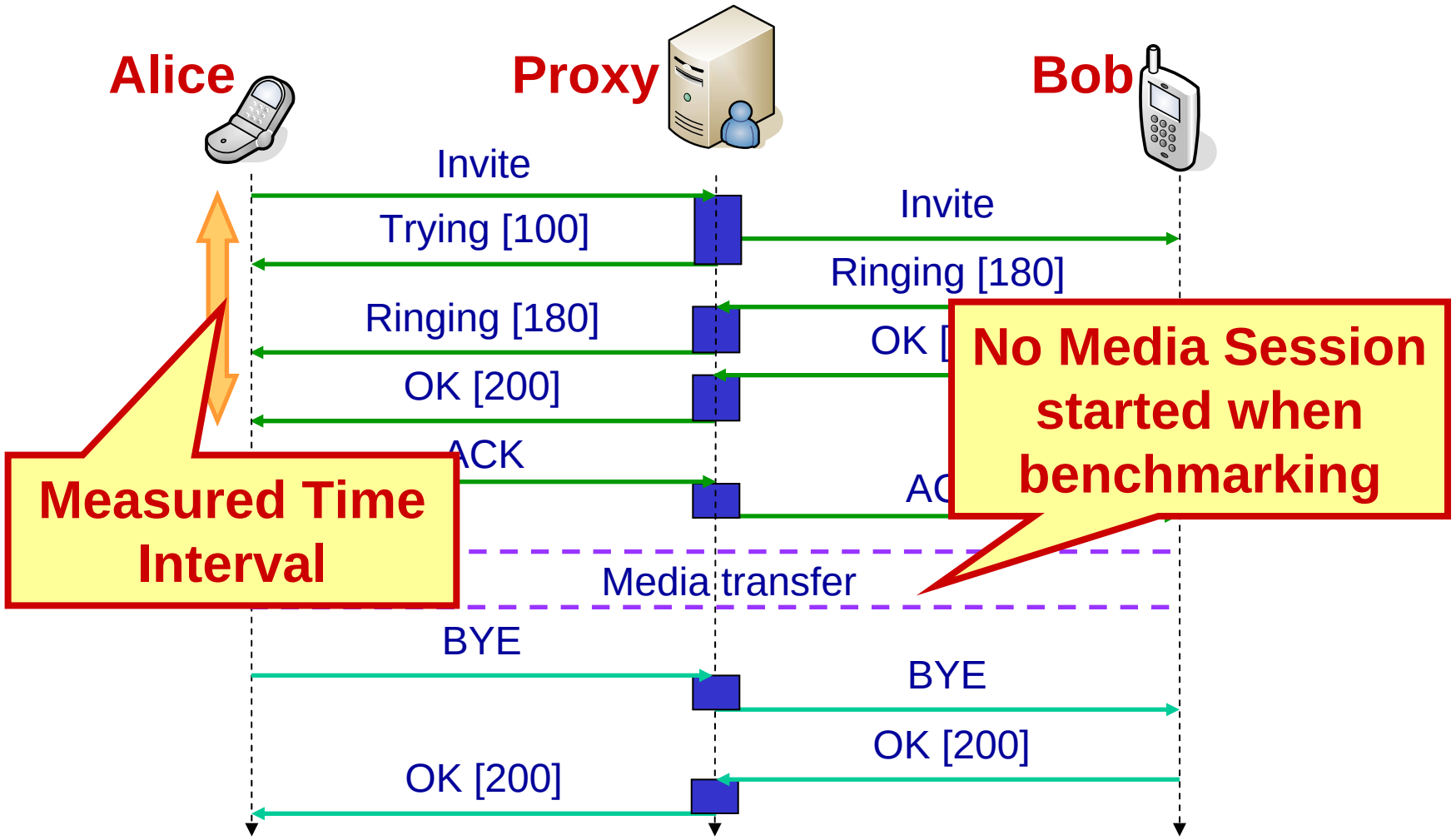
Java Virtual Machine: Features

- ✓ Java Byte Code is executed by a virtual Machine
  - ➢ This makes it platform independent
- ✓ Java features automatic Memory Management and a Garbage Collector
  - ➢ This simplifies the task of the developer

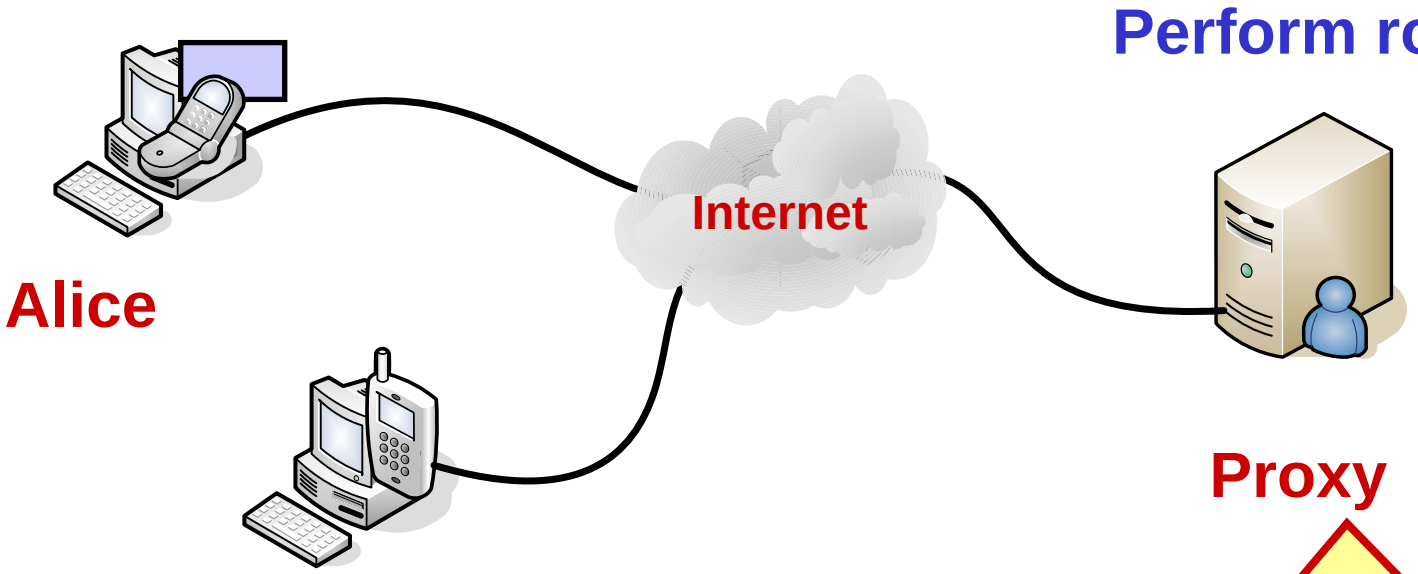
Java Virtual Machine: Problems

- ✓ Java Byte Code is executed by a virtual Machine
  - ➢ This can be a performance penalty
- ✓ Java features automatic Memory Management and a Garbage Collector
  - ➢ This can cause a performance penalty due to unpredictable behavior of the system.

Evaluation: Proxy 200 benchmark



Evaluation Setup: Hardware



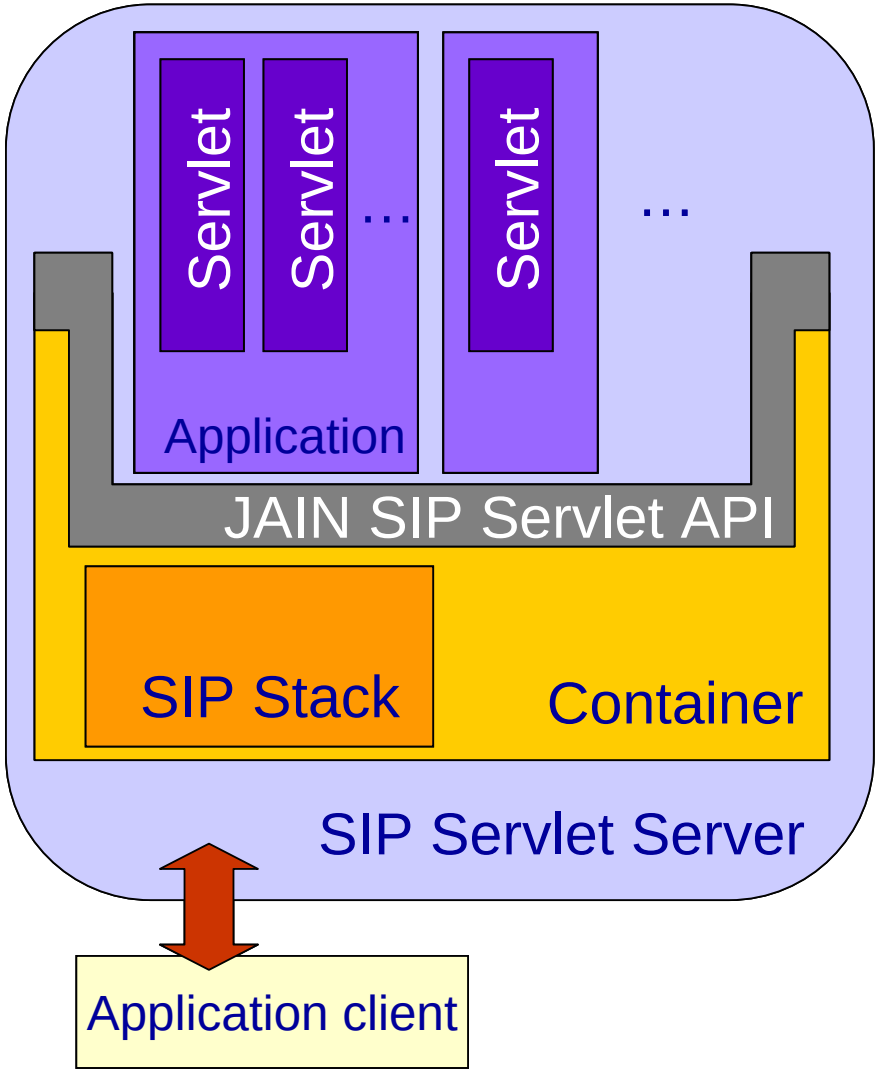
Evaluation Setup: Software

- ✓ SIP Servlet
  - ➢ BEA Weblogic SIP Server
- ✓ JAIN SLEE
  - ➢ Open Cloud Rhino

- ✓ Benchmark software
  - ➢ SIPp (test tool / traffic generator for the SIP protocol)

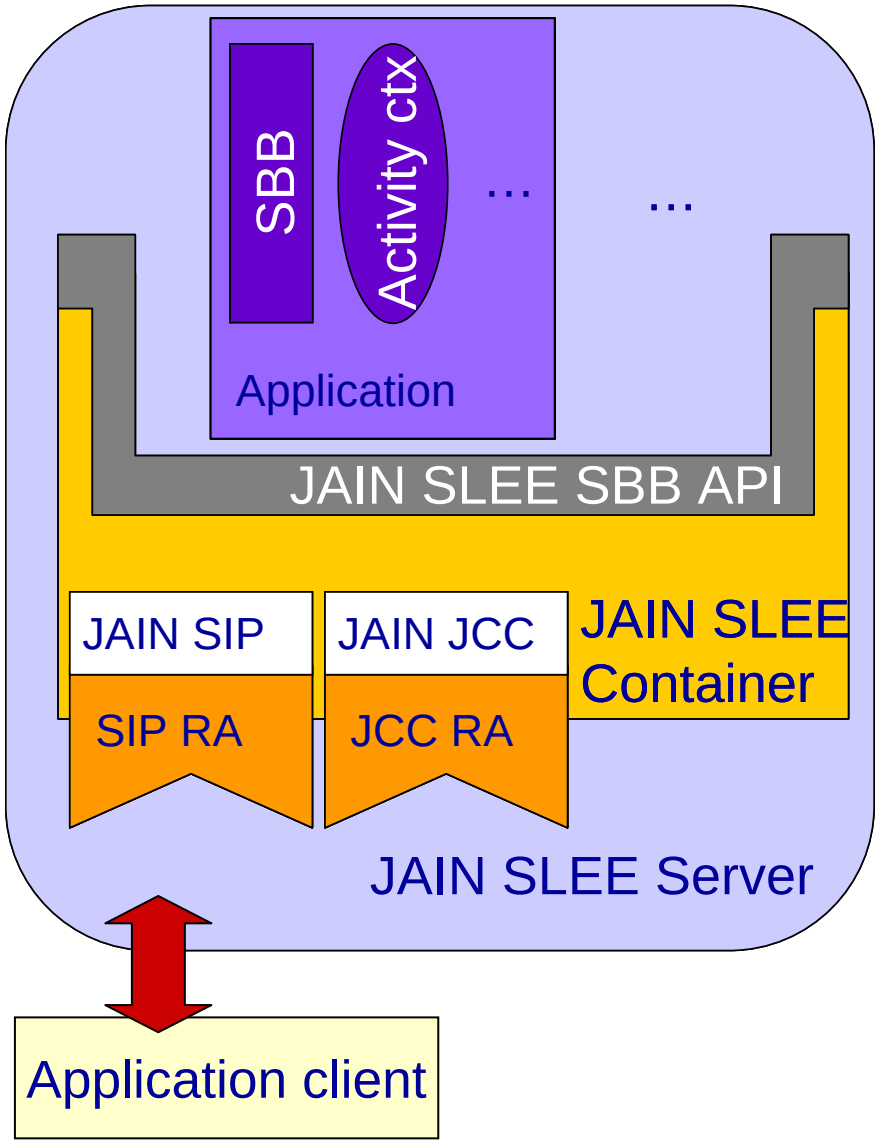
SIP Servlet: Architecture

- ✓Container based
  - ➢Life cycle management
  - ➢Manage network listen points
- ✓Very similar to HTTP-Servlet
- ✓SIP Protocol Specific
- ✓Request Response Model



JAIN SLEE: Architecture

- ✓Container based
  - ➢Life cycle management
- ✓Protocol Agnostic
- ✓Resource Adaptors allow protocols to be "plugged in"
  - ➢Manage network Listen Points
- ✓Event driven and asynchronous



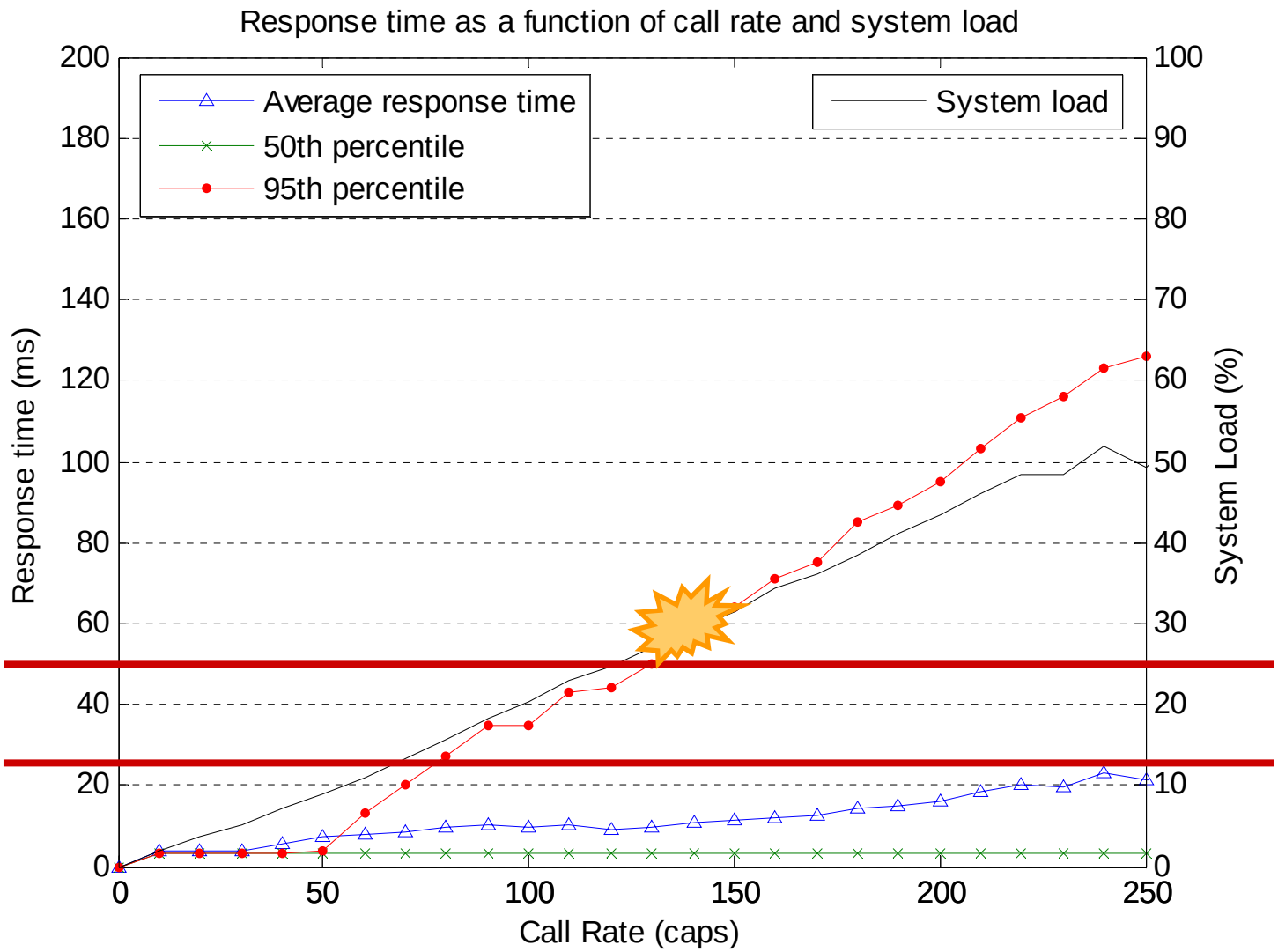
Message Handling

SIP Request	SIP SERVLET	JAIN SLEE
INVITE	doInvite()	onInviteEvent()
ACK	doAck()	onAckEvent()
OPTIONS	doOptions()	onOptionsEvent()
BYE	doBye()	onByeEvent()
CANCEL	doCancel()	onCancelEvent()

Methods defined in specification javax.servlet.sip.SipServlet

Methods defined by event names specified by the Resource Adapter

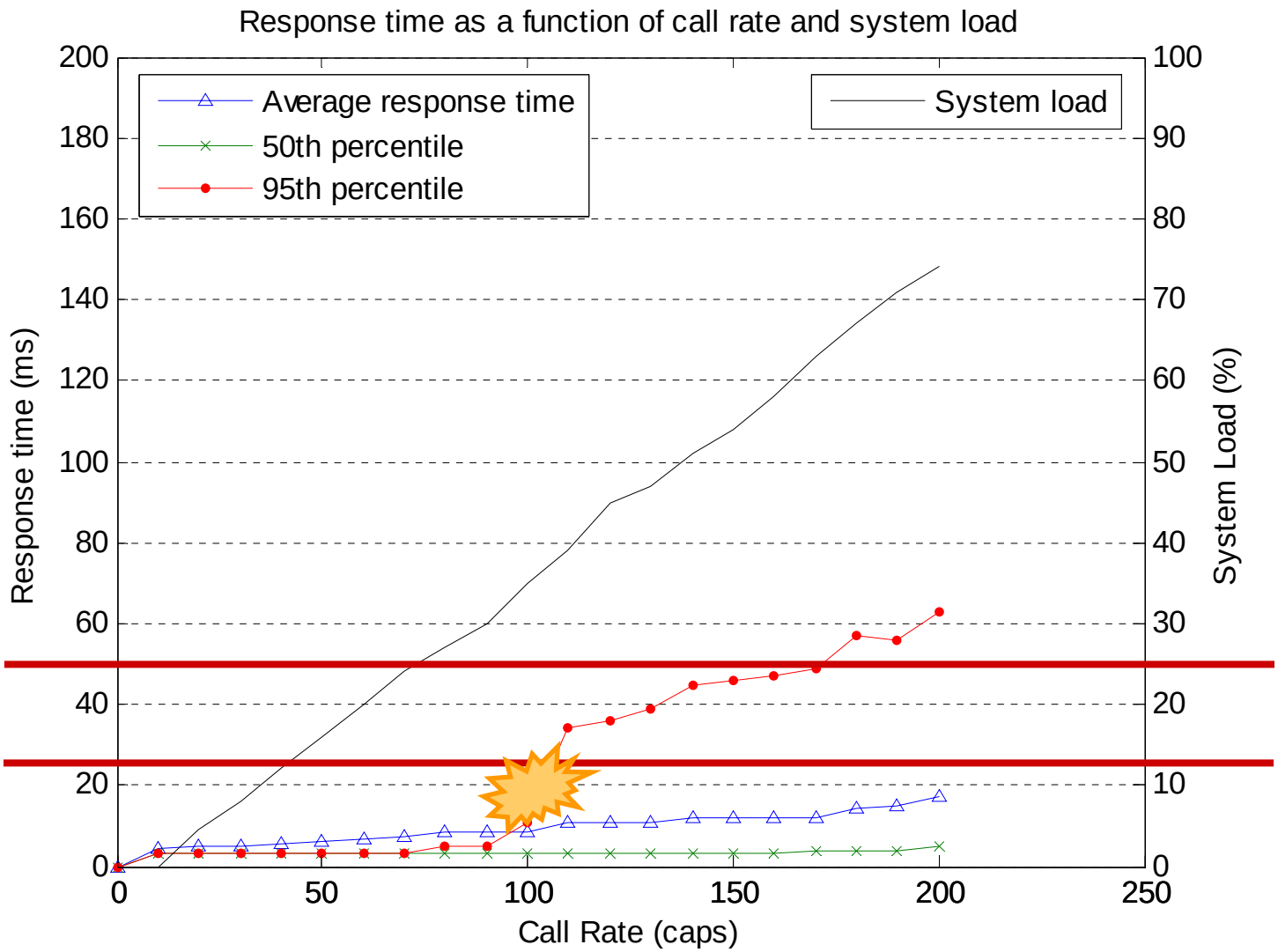
Results: SIP Servlet (default)



50 ms

25 ms

Results: JAIN SLEE (untuned)

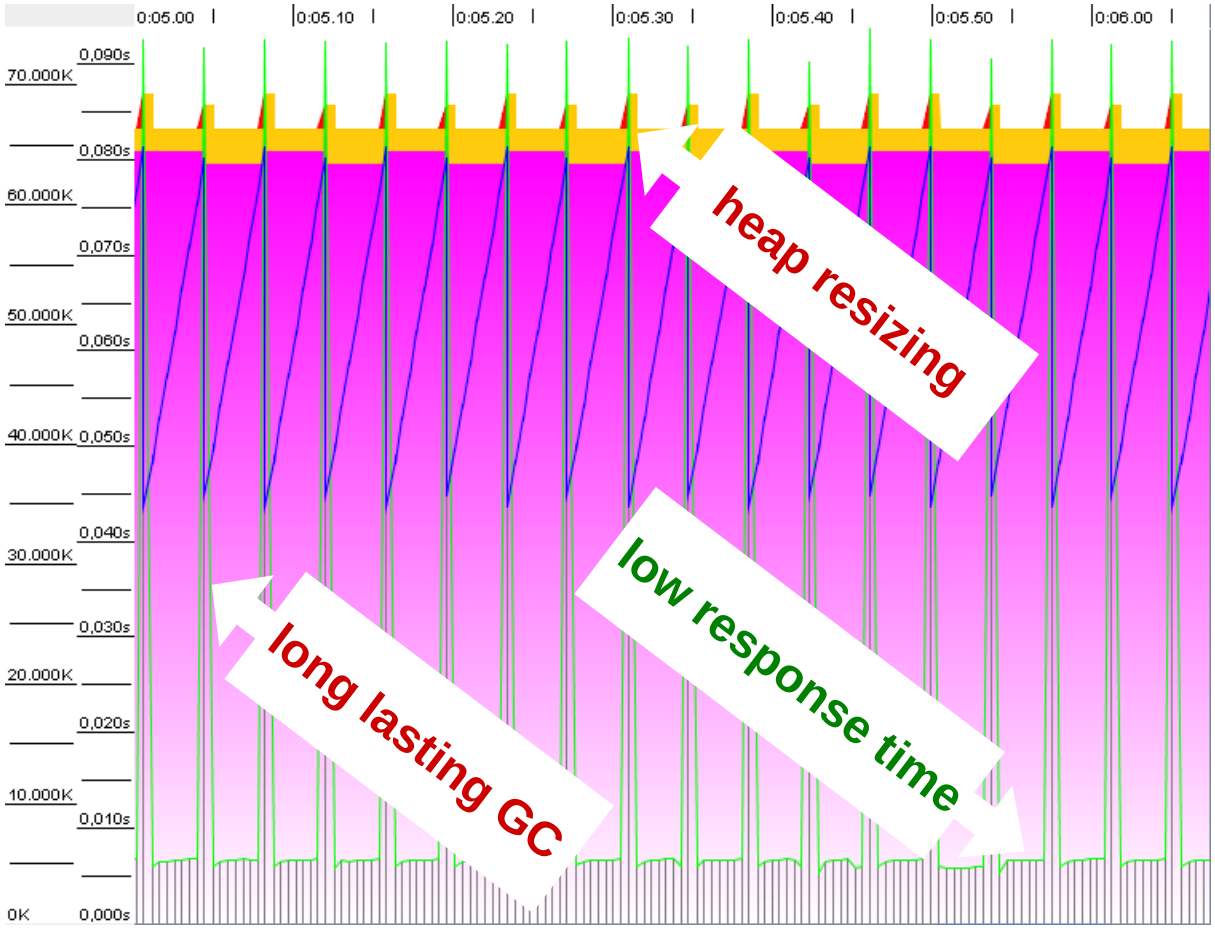


50 ms

25 ms

Cause of the problems?

- ✓ Memory Management and Garbage Collector



Legend

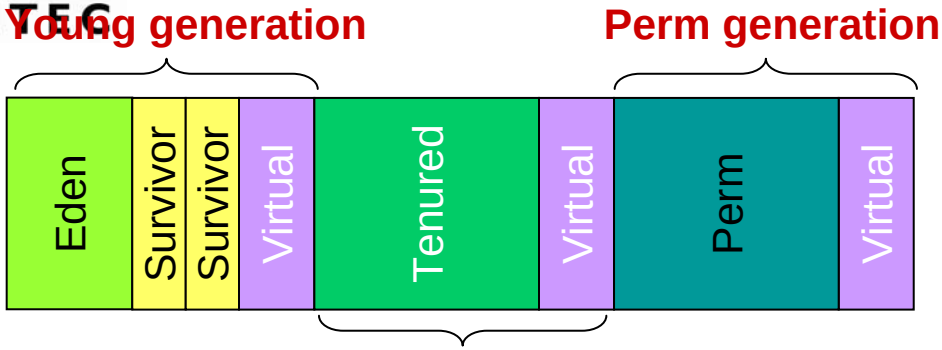
- Heap Usage
- GC-Time
- GC-occurrence
- Tenured Gen.
- Young Gen.

Image created with gcviewer (http://www.tagtraum.com/gcviewer.html)

Solution: Virtual Machine Tuning

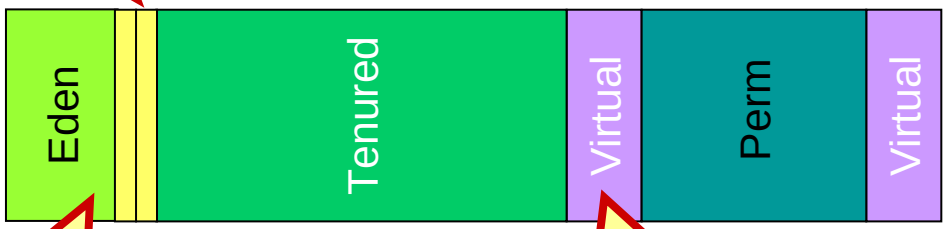
- ✓ Memory Organization
- ✓ Garbage Collection

Memory Tuning



Tiny Survivor Spaces

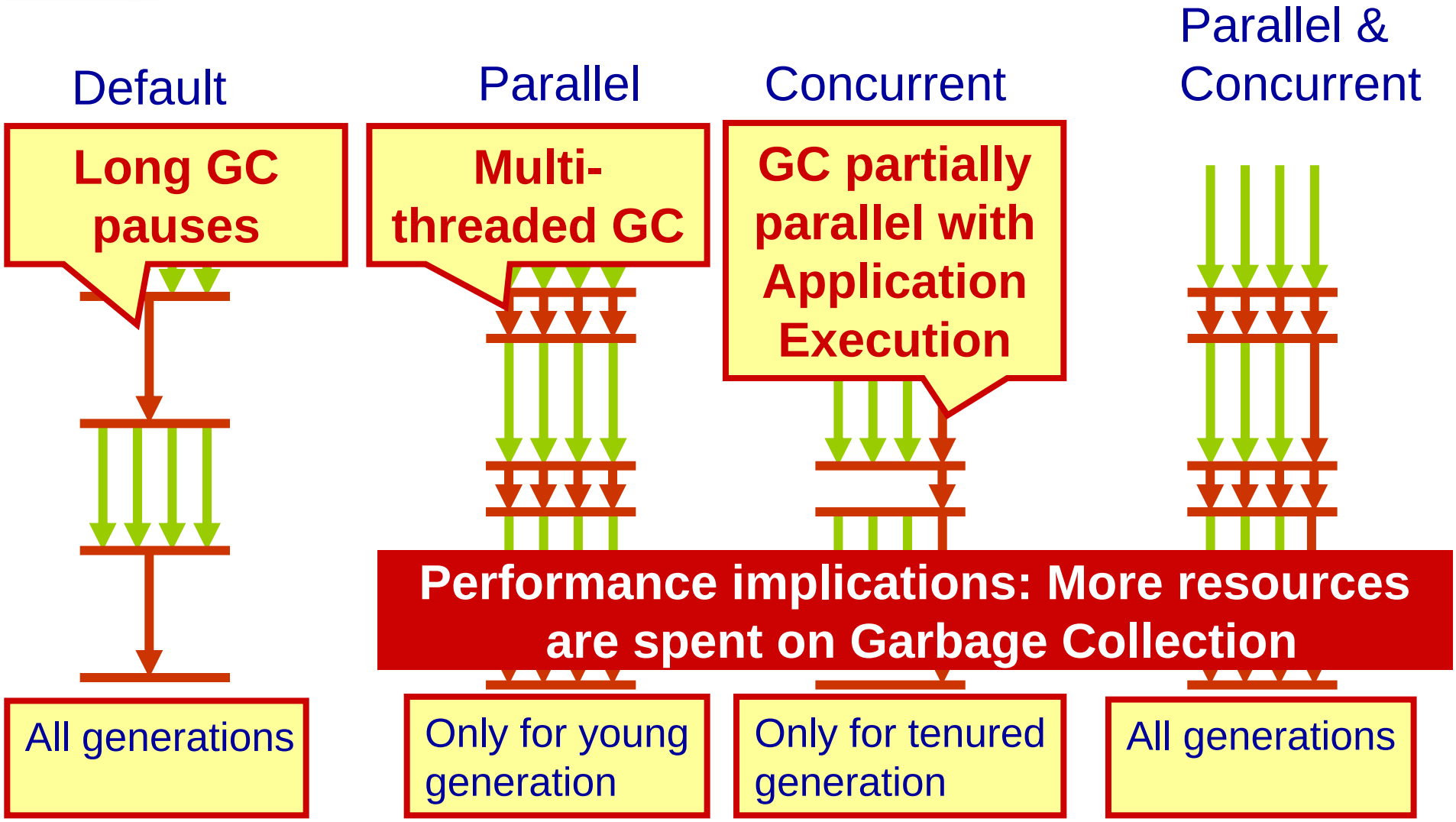
Total Heap Size large enough



Fixed Young Generation Size

Other Generation Sizes NOT Fixed

Garbage Collector Tuning



Tuning Options

- ✓ -Xmx512m
- ✓ -XX:MaxNewSize=32m
- ✓ -XX:NewSize=32m

Sizing of the memory generations

- ✓ -XX:+UseParNewGC
- ✓ -XX:+UseConcMarkSweepGC
- ✓ -XX:+CMSIncrementalMode
- ✓ -XX:+CMSIncrementalPacing
- ✓ -XX:CMSIncrementalDutyCycleMin=0
- ✓ -XX:CMSIncrementalDutyCycle=10

Selection of the Garbage Collector

- ✓ -XX:+UseTLAB

Allow multi-threaded memory allocation

- ✓ -XX:MaxTenuringThreshold=0
- ✓ -XX:SurvivorRatio=128

Move long living objects to Tenured Generation

Problems solved?

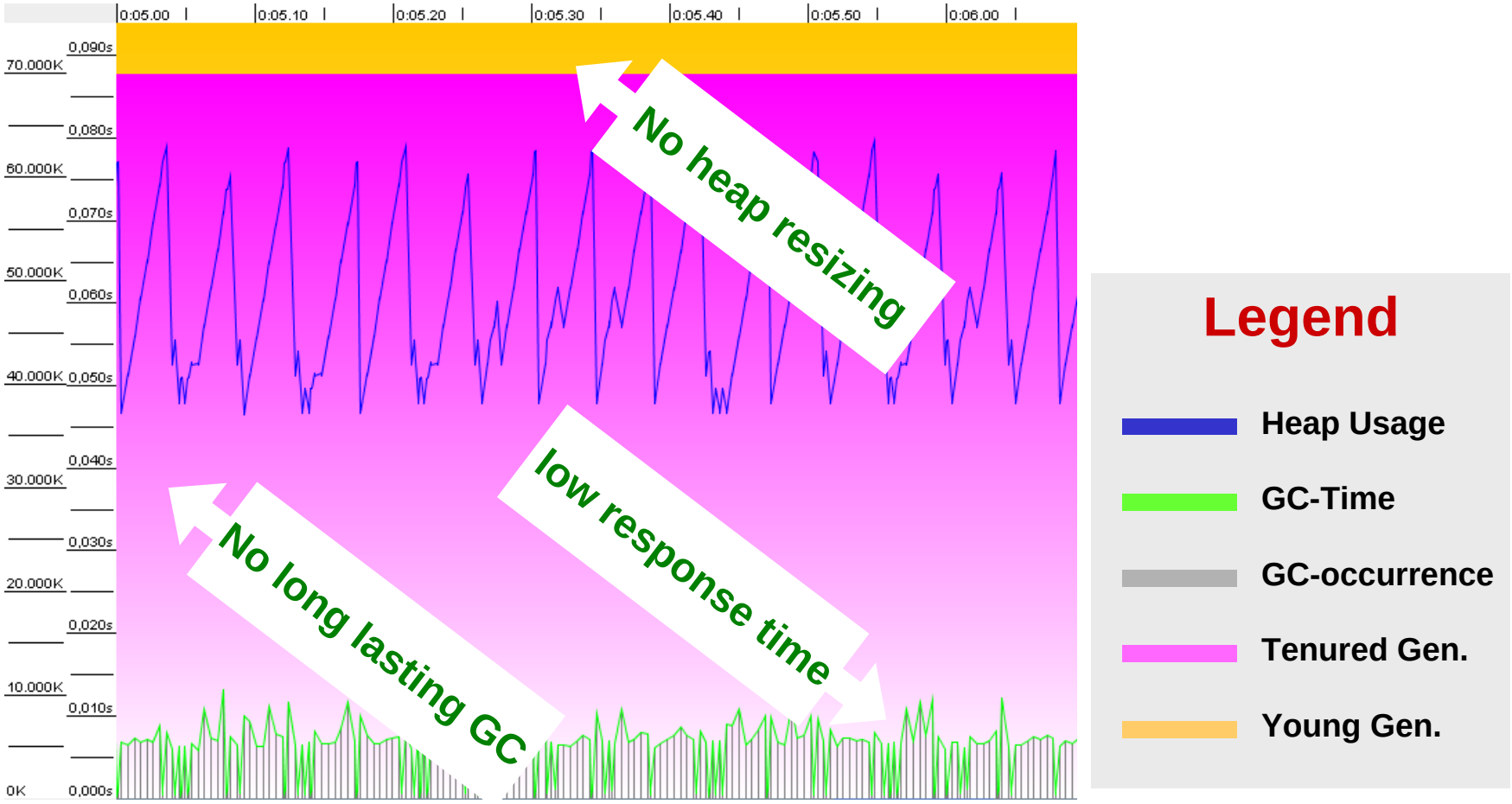
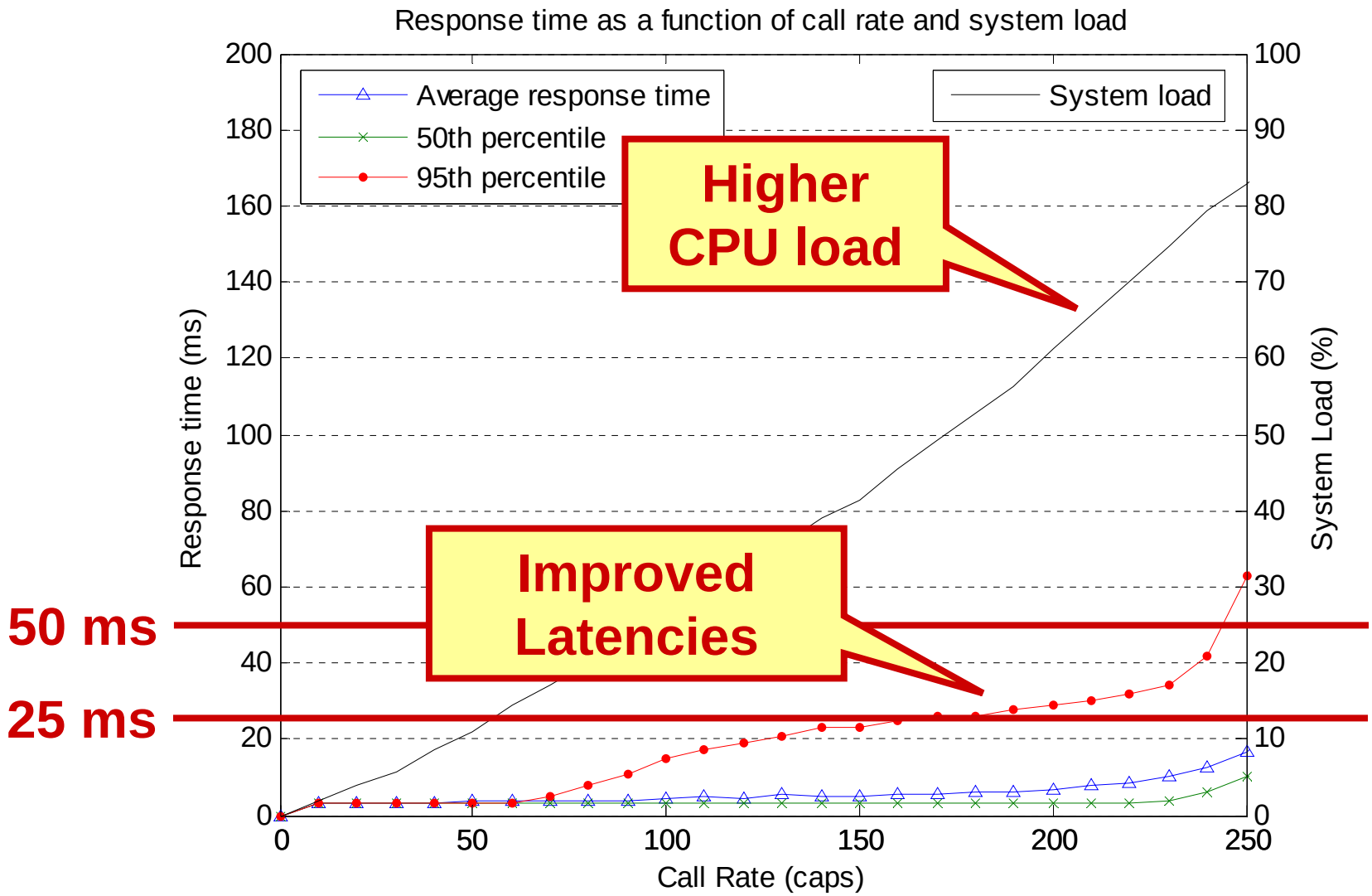
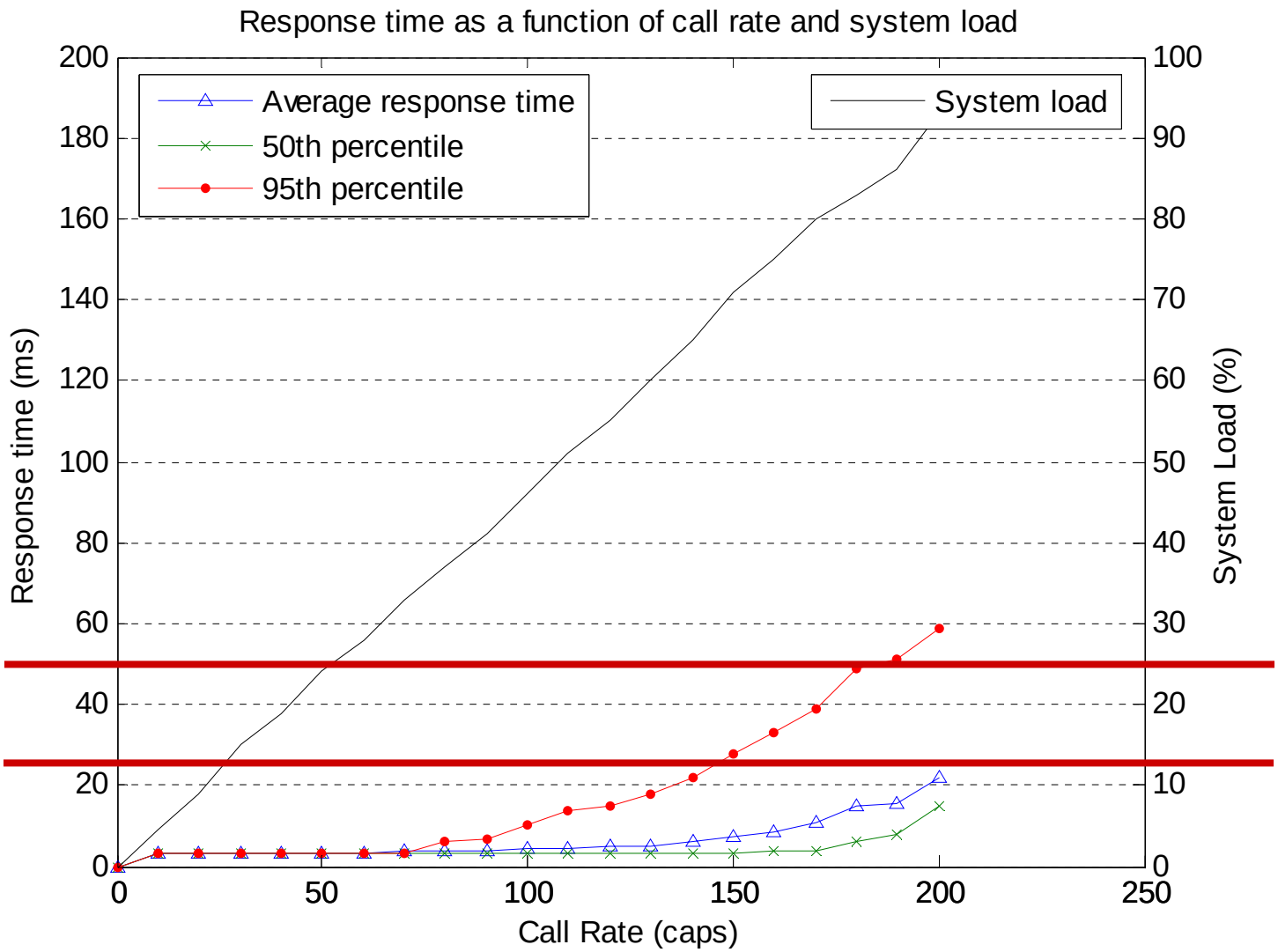


Image created with gcviewer (http://www.tagtraum.com/gcviewer.html)

Results: SIP Servlet



Results: JAIN SLEE



50 ms

25 ms

Conclusions

- ✓ Java Technologies can simplify the Design and Management of Telecom and VoIP related applications
- ✓ Java Virtual Machine Tuning can improve the Java Garbage Collection significantly
- ✓ Java Application Servers combined with appropriate tuning can meet strict Low Latency Requirements