Road Test Results of the “Big Three” Web Application Frameworks

Sujoe Bose,
Architect, Sabre Holdings.

TS-6517

Java

Sun microsystems

Learn about the Performance Characteristics of the Web Application Frameworks – Struts, JavaServer™ Faces (JSF) and Spring MVC

GOAL

Agenda

- Benchmark Motivation
- Challenges and overall strategy
- Web Frameworks Overview
- Benchmark Application
- Road Test Configuration
- Road Test Results
- Conclusion Summary
- Future Work

Motivation

- Many Web Frameworks available to choose from
- Each have some distinguishing feature or functionality
- Frameworks vary in Architecture and Design
  - Moving from one framework to another may not be easy!
  - Most cases an entire paradigm shift occurs
- Choice of which frameworks to use based on different indicators
  - Time-tested and proven
  - Better design
  - Adoption rate
  - Industry Standard
  - Tools availability

Motivation Continued…

- Static properties of Frameworks well understood and contrasted
- Dynamic Properties of Web Frameworks not available on a level-playing field
- This Benchmark is aimed at identifying the Dynamic Properties of frameworks in a level-playing field
  - Apples-to-apples comparison
- Positive viewpoint to be maintained when comparing frameworks
  - Find current standing so improvements can be done
  - Make educated decisions on framework usage

Motivation: System Sizing

- Based on benchmark results apply Queuing Theory to estimate number of servers needed
- Define and control QoS parameters such as response time etc.
- Apply appropriate Load balancing schemes
  - Simple vs. Adaptive
- Based on resource consumption, estimate hardware resources

Agenda

- Benchmark Motivation
- Challenges and overall strategy
- Web Frameworks Overview
- Benchmark Application
- Road Test Configuration
- Road Test Results
- Conclusion Summary
- Future Work

Challenges

- Each framework is different in the design and organization
  - Page-centric vs. Component-centric
- Each framework employs different adaptor contracts resulting in different execution paradigms
- Apples-to-apples comparison difficult to attain
  - External Dependencies such as database, networks etc
  - Code variations to fit into the framework
- Variations within the frameworks in usage patterns
- Various versions of frameworks available to choose from

Strategy to overcome/minimize challenges

- Each framework is different in the design and organization
  - Common abstraction and design to isolate differences
- Apples-to-apples comparison
  - External Dependencies such as database, networks etc
    - **Use in-memory data structures**
    - **Dedicated network with load and app machines separated out**
  - Code variations to fit into the framework
    - **Design with glue and application code isolated and abstracted**
- Variations within the frameworks in the usage
  - Compare a few of those variations
- Various versions of frameworks available
  - Compare some prominent versions

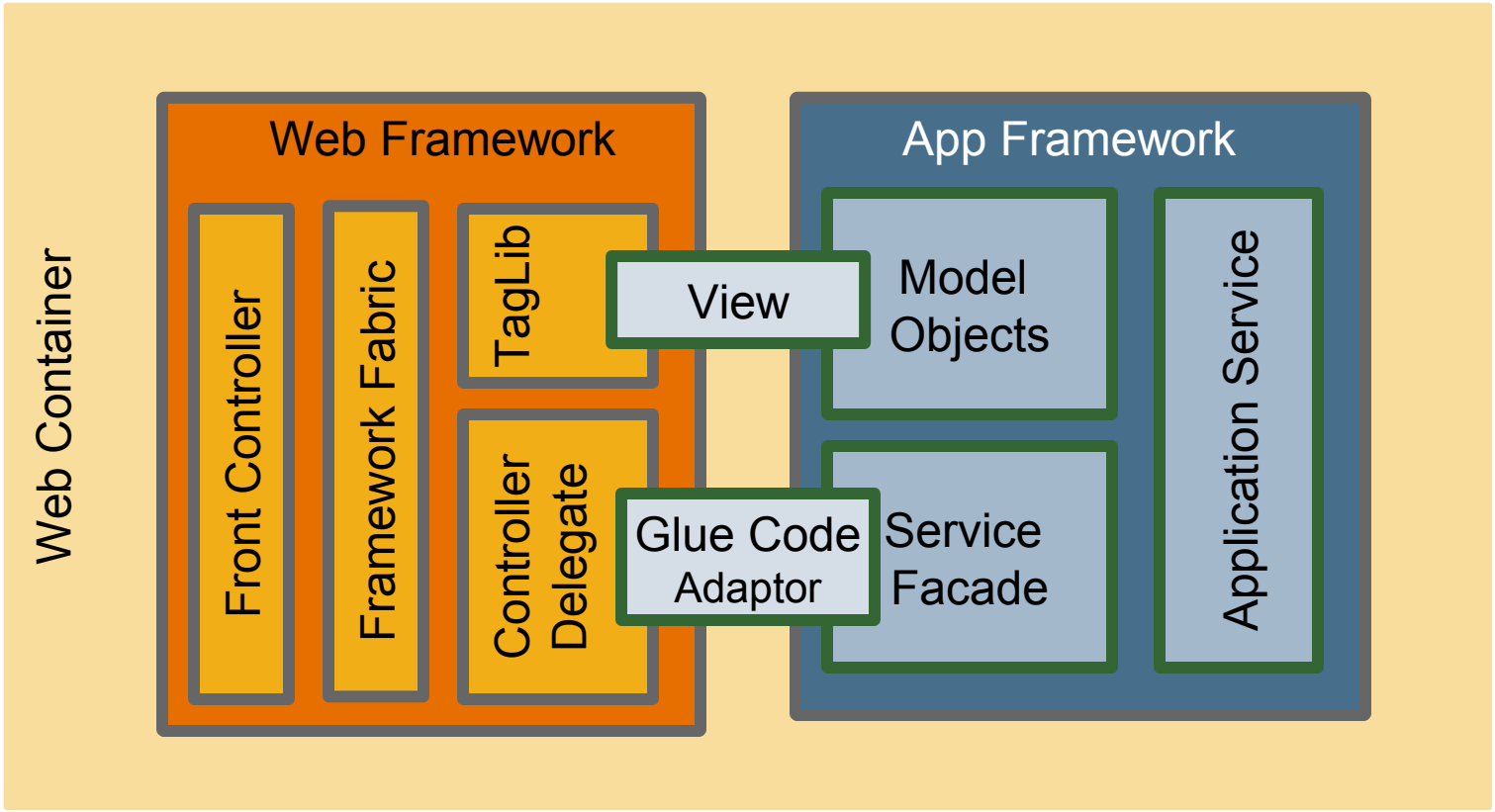
Agenda

- Benchmark Motivation
- Challenges and overall strategy
- Web Frameworks Overview
- Benchmark Application
- Road Test Configuration
- Road Test Results
- Conclusion Summary
- Future Work

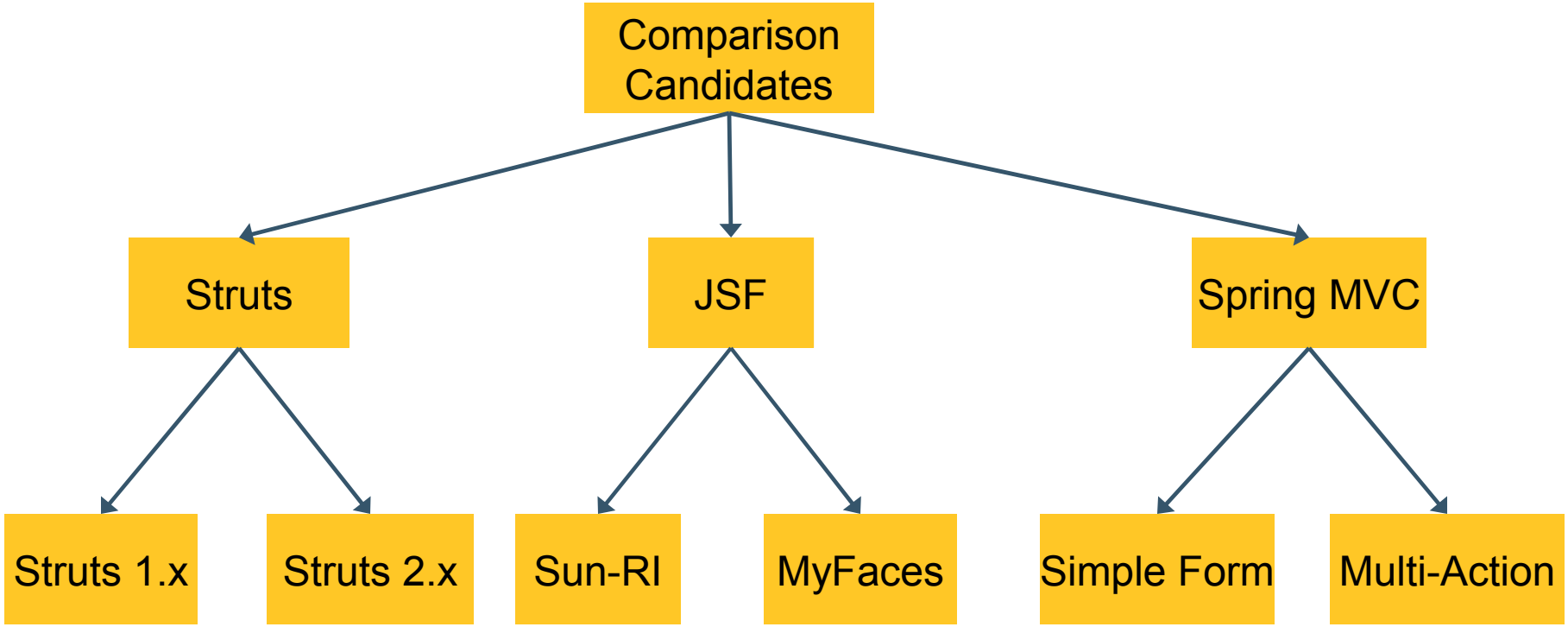
Common Extensible Design

- Separation of Glue Code from Application Code
  - Glue code satisfies framework contract
  - Application code provides business services
- Application Code is common to all frameworks
- Glue Code is specific to a framework and is kept to a minimum
  - Coupling with Framework is minimized
  - Performance differences due to glue are minimized
- Facilitates apples-to-apples comparison
- Views are dependent on the framework due to usage of tag libraries, however model classes are same

Benchmark Design Template

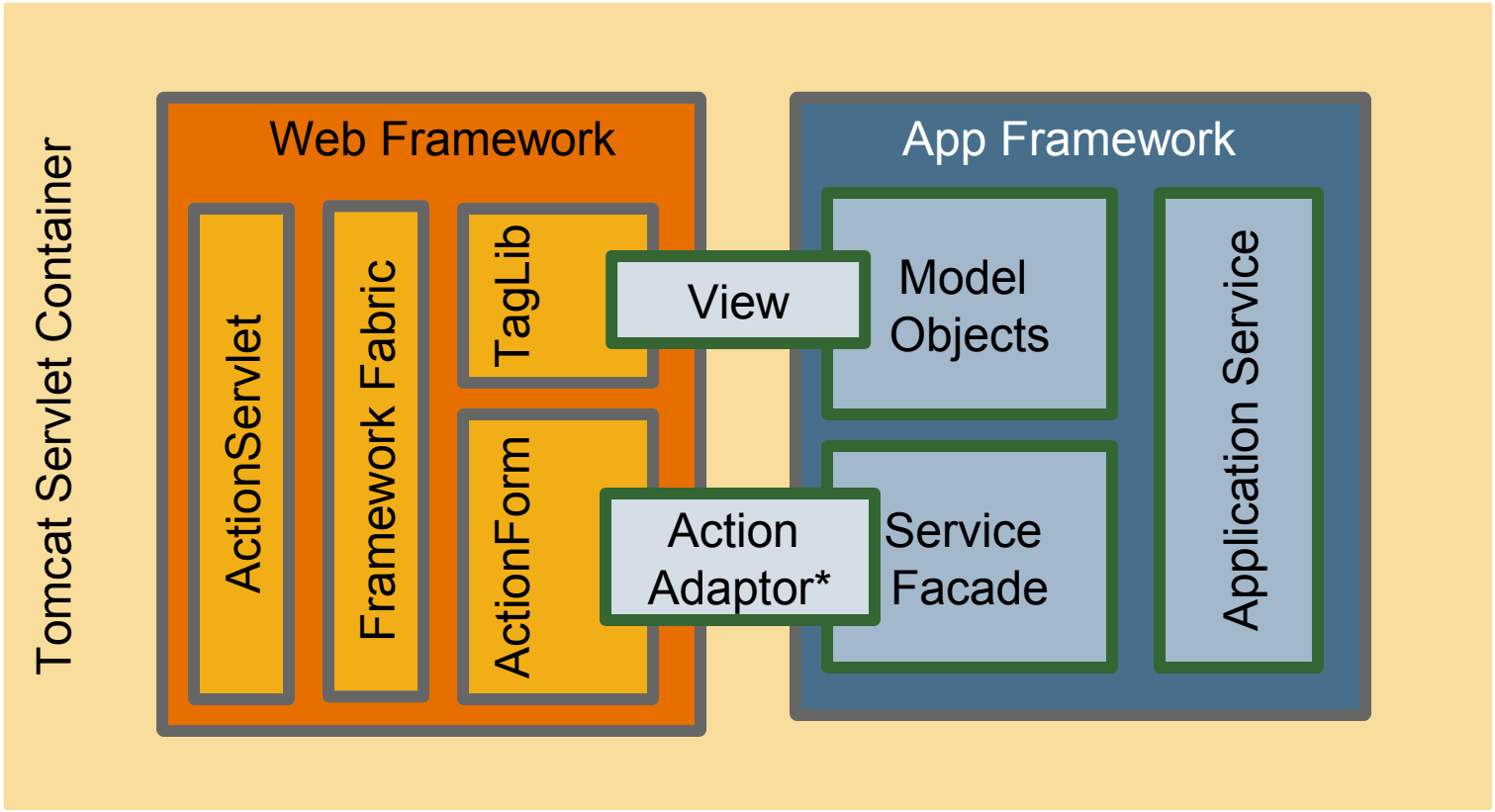


Benchmark Constituents



Struts

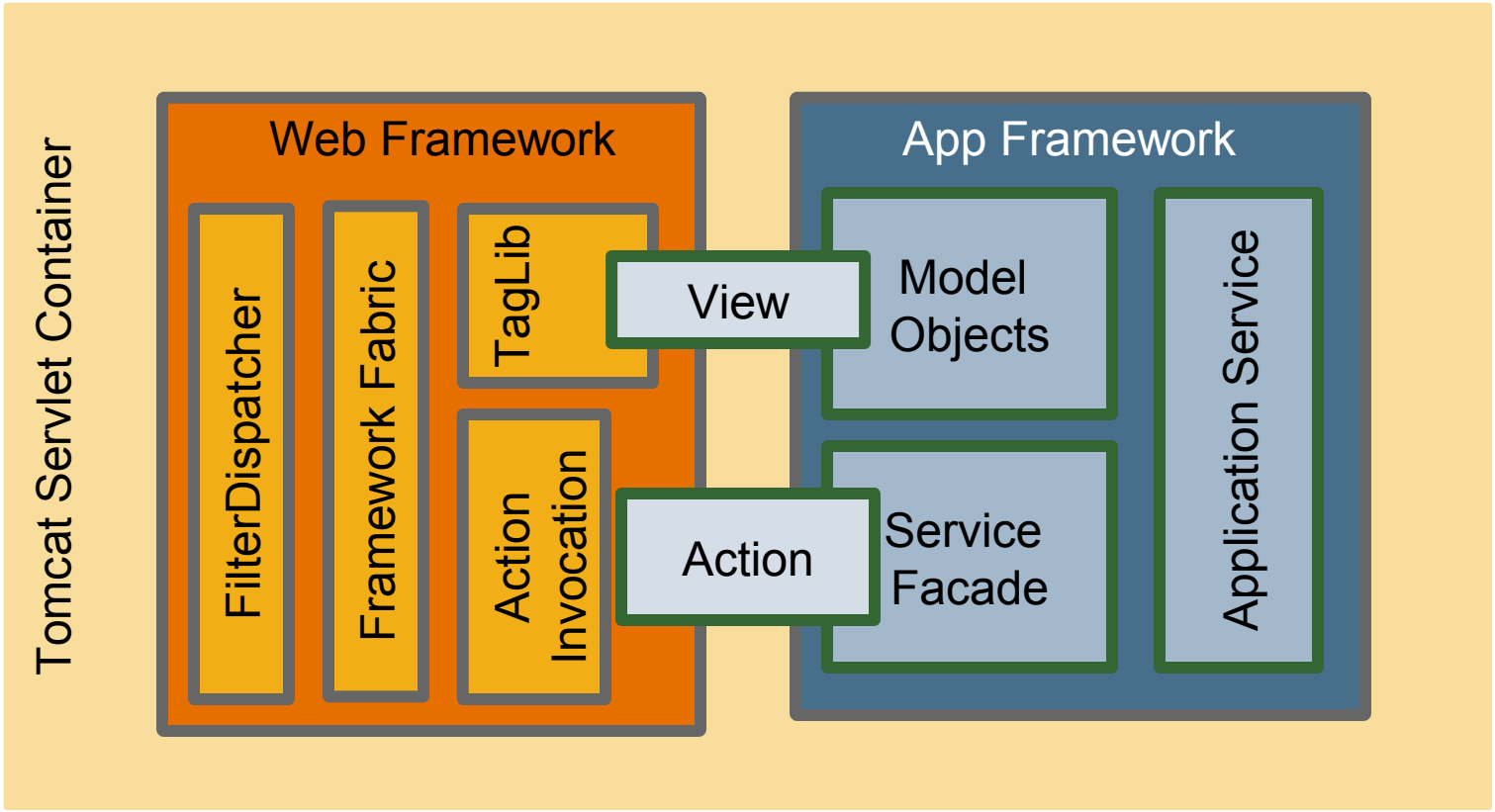
1.x



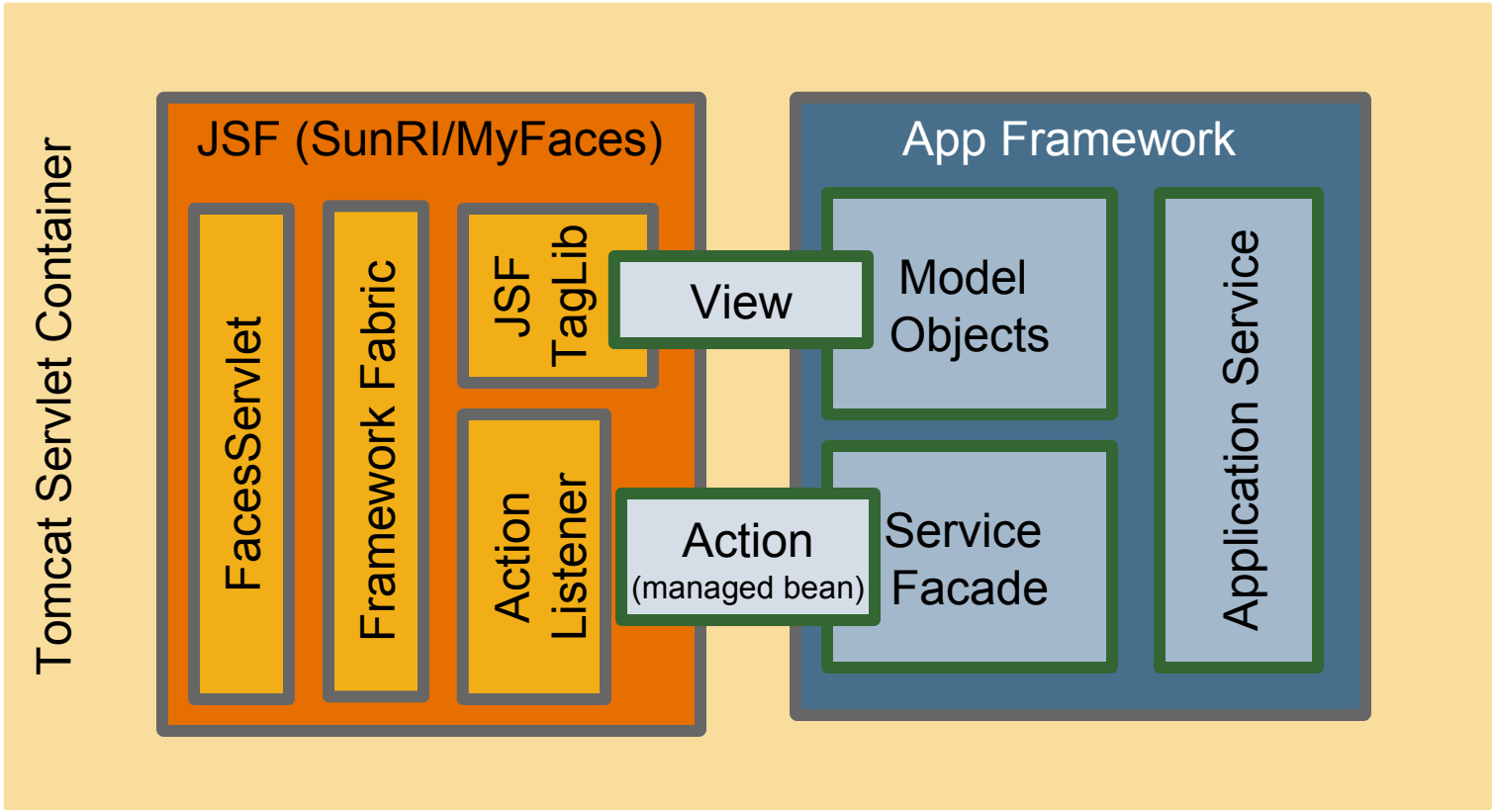
* One Action Adaptor for each Form

Struts

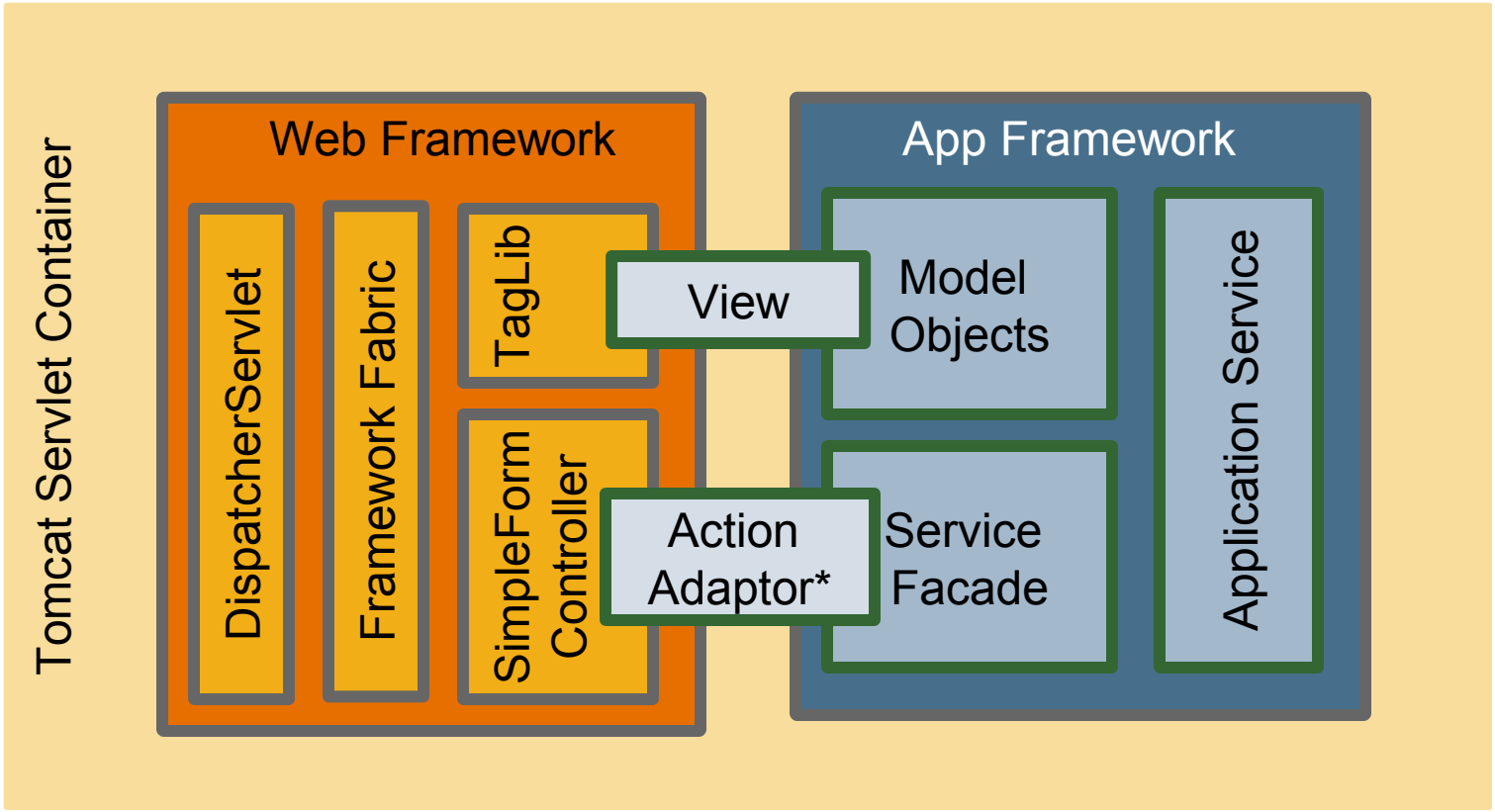
2.x


Tomcat Servlet Container
Web Framework
FilterDispatcher
Framework Fabric
TagLib
Action Invocation
View
Action
App Framework
Model Objects
Service Facade
Application Service

JSF
Sun-RI and MyFaces

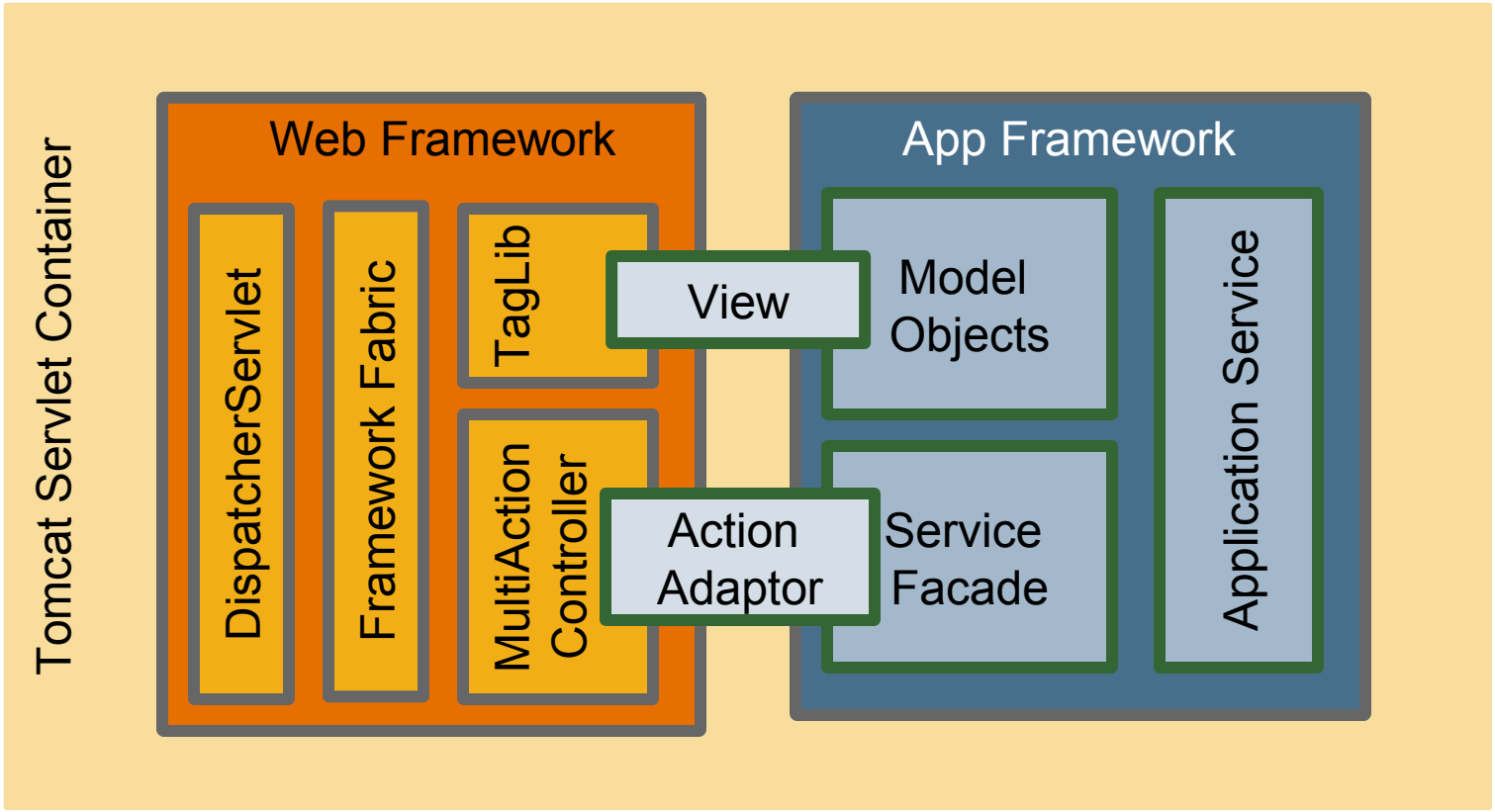


Spring MVC
Simple Form Controller



* One Action Adaptor for each Form

Spring MVC
Multi-Action Form Controller



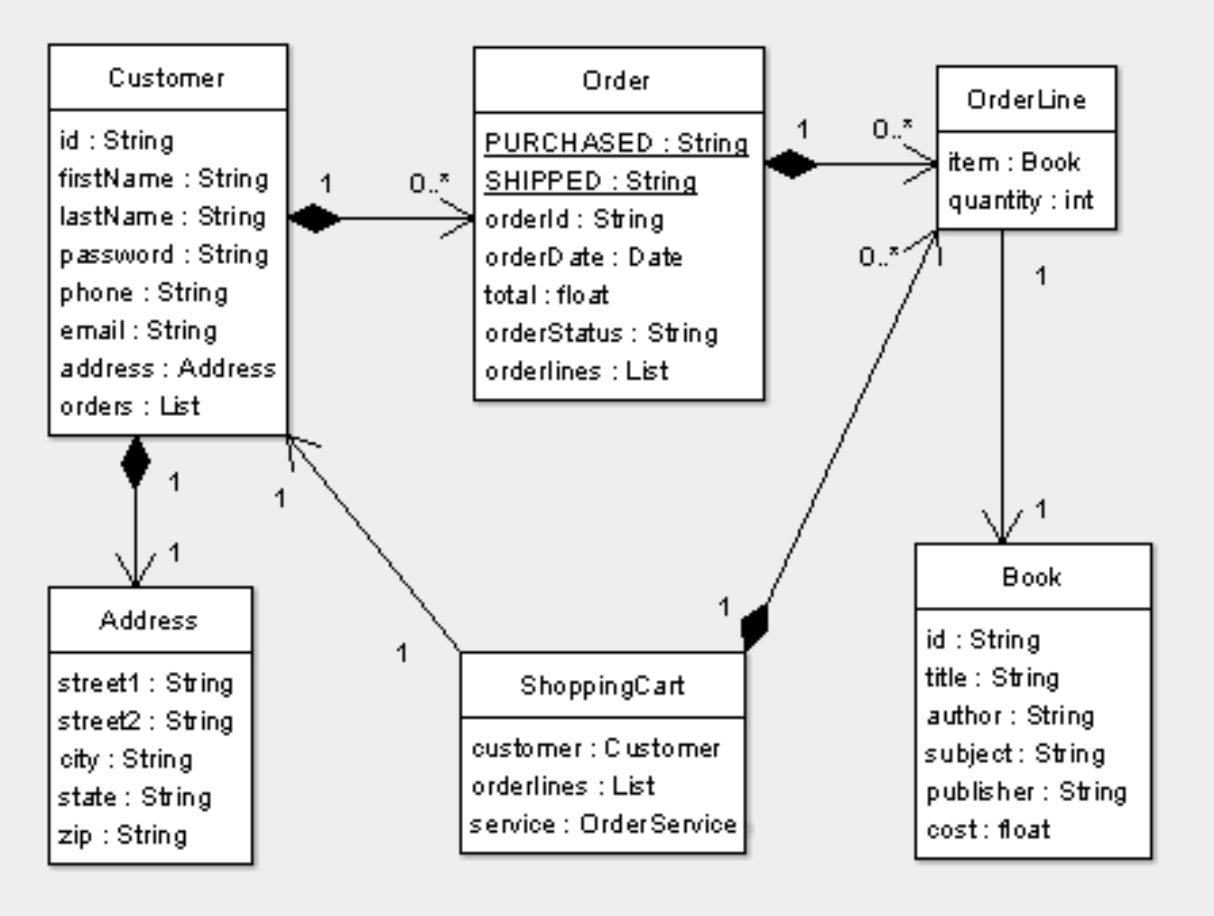
Agenda

- Benchmark Motivation
- Challenges and overall strategy
- Web Frameworks Overview
- Benchmark Application
- Road Test Configuration
- Road Test Results
- Conclusion Summary
- Future Work

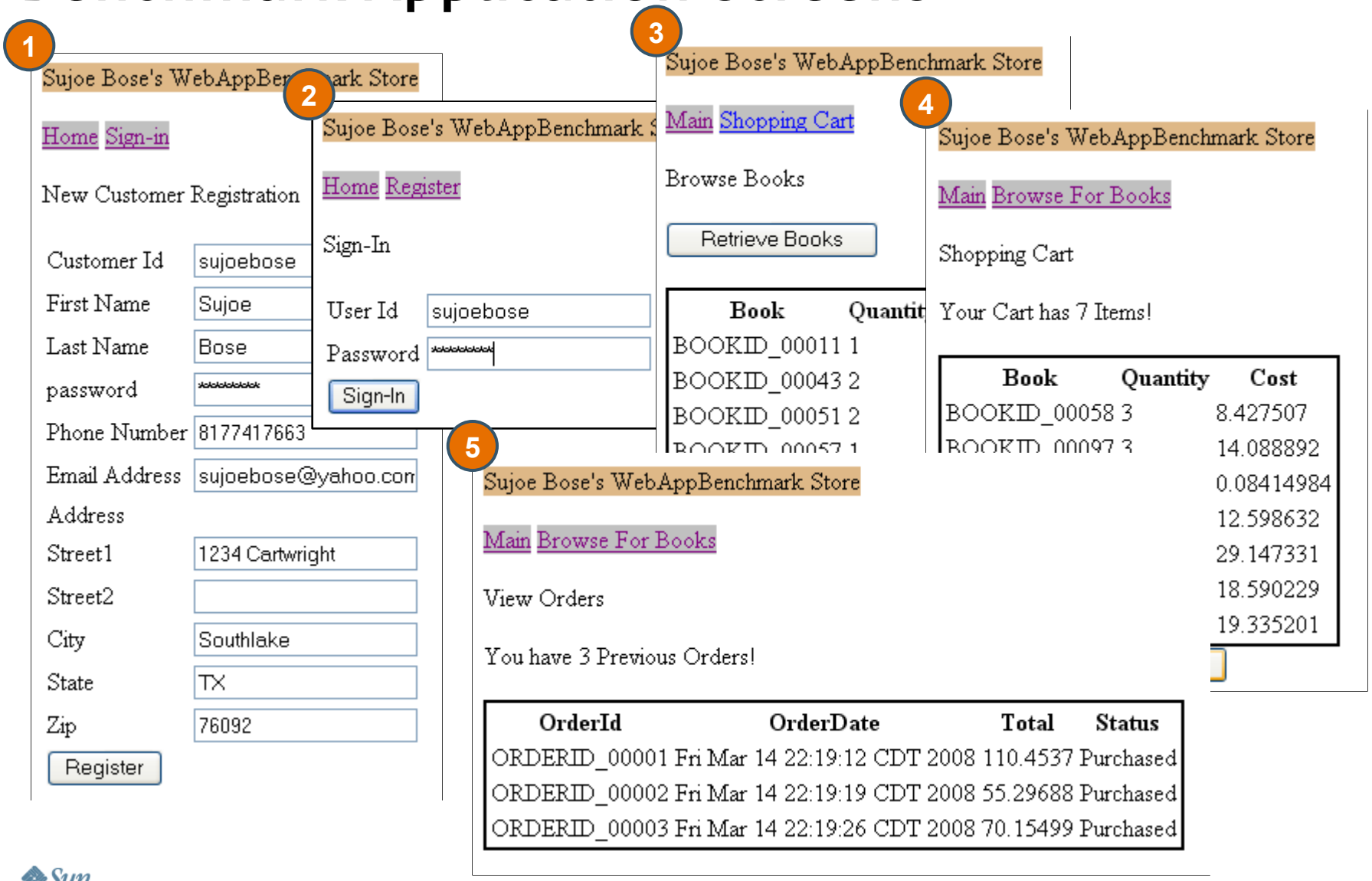
Benchmark Application

- Simple application based on TPC-W web store application
  - Scaled-down version with most of the main class attributes
- No databases involved
  - Eliminate external dependencies on performance
  - Using in-memory data structures to simulate data store
- Network dependencies eliminated with dedicated router configuration
- Web layout and components are made to be very similar across all applications
  - No validations or localizations performed

Benchmark Application Model



Benchmark Application Screens



Agenda

- Benchmark Motivation
- Challenges and overall strategy
- Web Frameworks Overview
- Benchmark Application
- Road Test Configuration
- Road Test Results
- Conclusion Summary
- Future Work

Benchmark Disclaimers

- This benchmark setup is in a way an ideal one and hence…
  - The results may not be reflective of real world implementations
- The versions of software used may not be the optimal combination for better performance
- The workload selected in the Max Load Test case
  - Does not contain user think time to simulate real-world traffic
  - Does not drive traffic simulating real-world distributions
- The frameworks themselves may not be used in their most efficient configuration and consumption mechanism

Benchmark Parameters

- Sustained Performance, under:
  - Steady Traffic
  - Bursty Traffic
- Performance Measures
  - Throughput
  - Response Time
- Resource Consumption
  - CPU Utilization
  - Memory Utilization
- Aim is to determine maximum sustainable throughput

Benchmark Load Configuration

- Load configuration:
  - Users
  - Iterations
  - Ramp-up delay
  - Think time
- With Think Time – simulating real-world scenario
  - Users:10, Ramp-up: 2s, Iterations:10, Think time: 3s (**10-2-10-3**)
  - Users:100, Ramp-up: 2s, Iterations:10, Think time: 3s (**100-2-10-3**)
- Without Think Time – simulating max throughput scenario
  - Users:10, Ramp-up: 0s, Iterations:10, Think time: 0s (**10-0-10-0**)
  - Users:100, Ramp-up: 0s, Iterations:10, Think time: 0s (**10-0-10-0**)

Software Versions Used

- Struts1 – 1.3.8
- Struts2 – 2.0.11
- JSF Sun-RI – 1.2.05
- JSF MyFaces – 1.2.2
- SpringMVC – 2.0.6
- Tomcat - 5.5.23
- jMeter – 2.3.1
- JDK™ – version 1.6.0

Benchmark Environment

- Load Driver and Application in separate machines
- Both machines are Intel® Pentium® running Windows® XP
  - 1.6Ghz
  - 1GB RAM
- Dedicated 100Mbps Ethernet router between the load and application machines
- jMeter on the Load Machine to simulate transaction load and capture metrics
- Windows Performance Counters used for recording resource utilization

Agenda

- Benchmark Motivation
- Challenges and overall strategy
- Web Frameworks Overview
- Benchmark Application
- Road Test Configuration
- Road Test Results
- Conclusion Summary
- Future Work

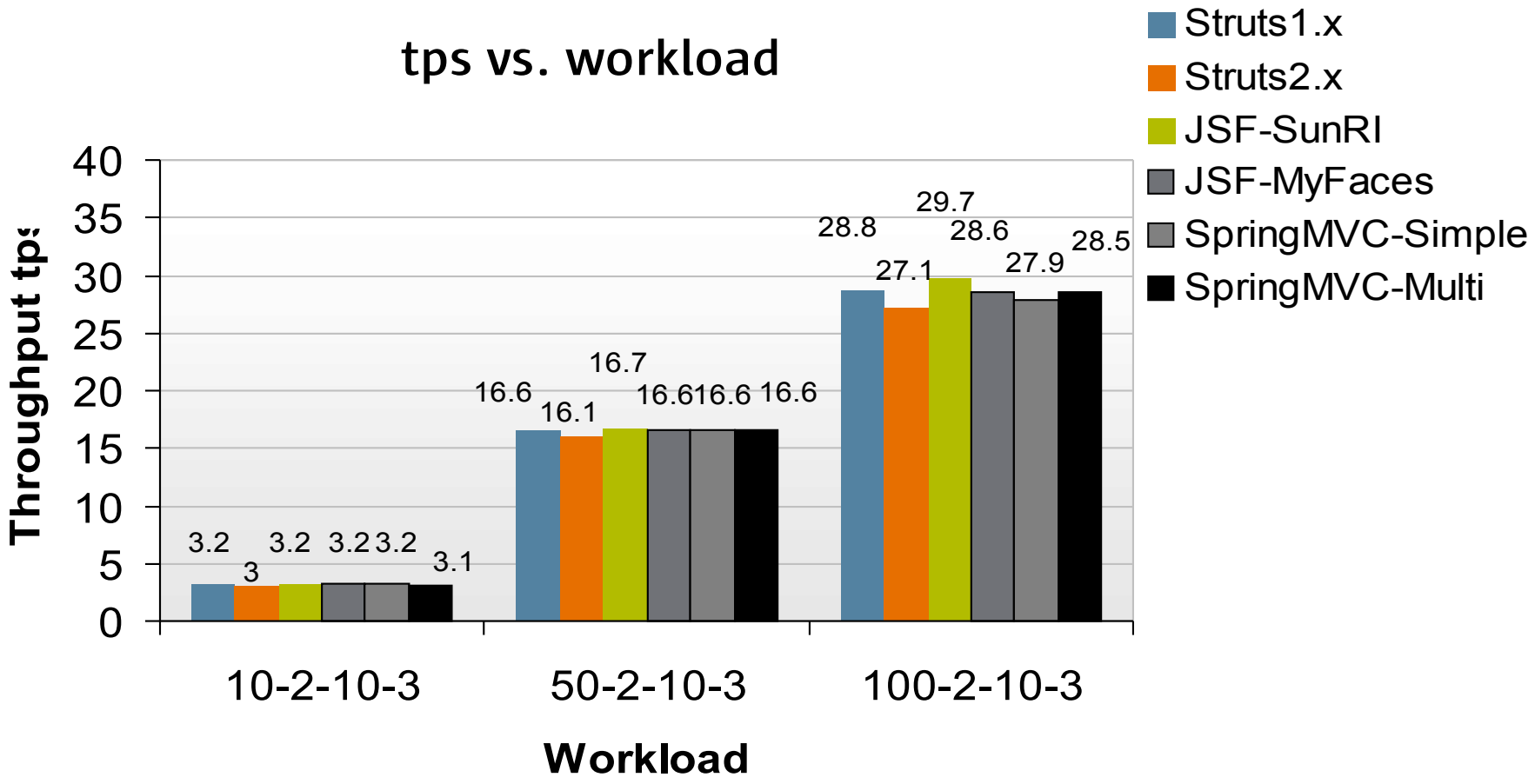
Metrics

- Throughput with Think time
- Max Throughput
- Overall (Average) Throughput
- Response Times for some transactions
- Max CPU Utilization
- Min Memory Requirement
- Errors in Load Test
- Ease of Use

Test Results under Sustained Load with Think time

Overall Throughput

Think time = 3s



Response Time (ms)

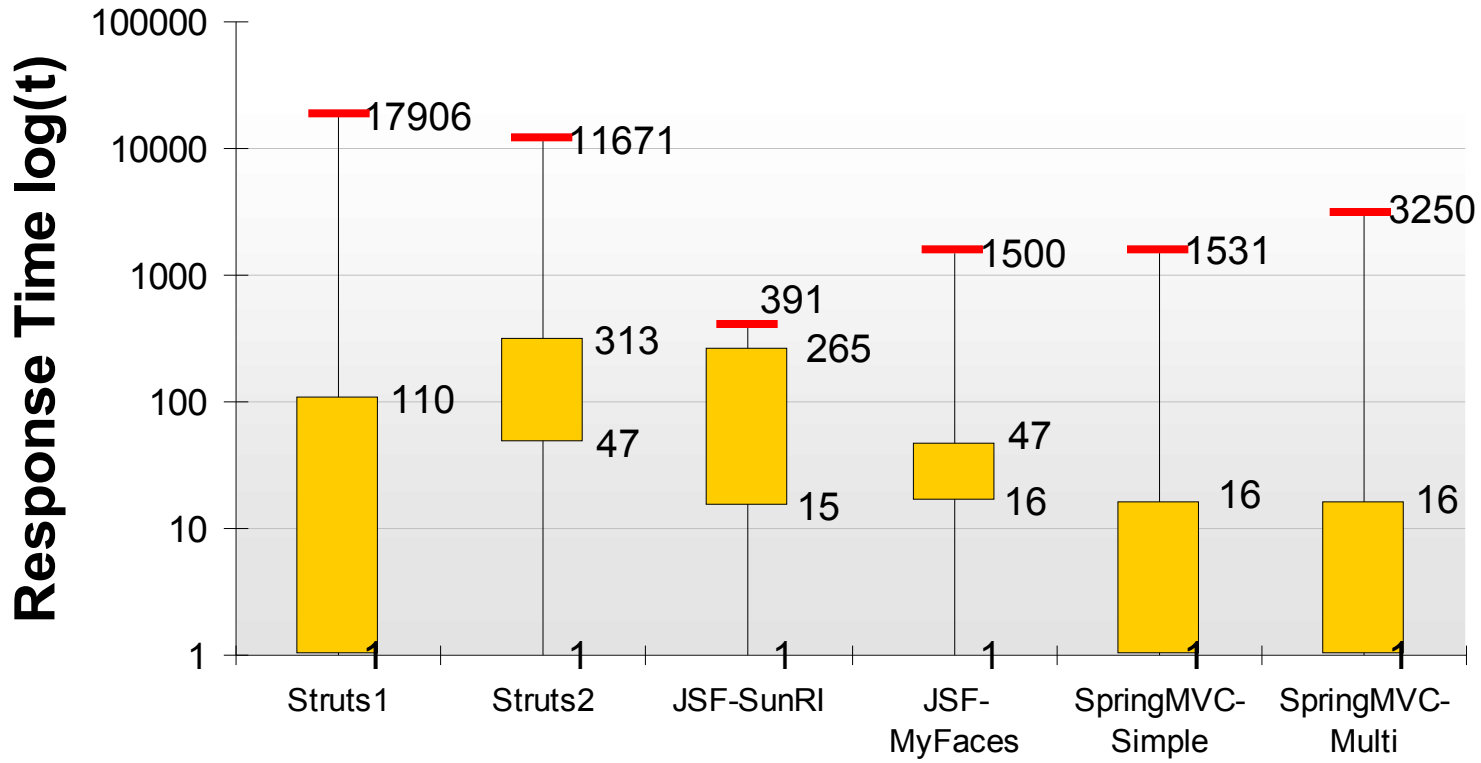
Registration Page with think time of 3s

Workload: 10-2-10-3

Framework	Median	Average	90% Line	Max	σ
Struts1	0	214	110	17906	1370.88
Struts2	47	167	313	11671	832
JSF-SunRI	15	41	265	391	100.27
JSF-MyFaces	16	65	47	1500	233.61
SpringMVC-Simple	0	51	16	1531	236.28
SpringMVC-Multi	0	50	16	3250	317.73

Response Time stat plot*

Registration page for Workload 10-2-10-3



* Box span: Median - 90% Line

Response Time (ms)

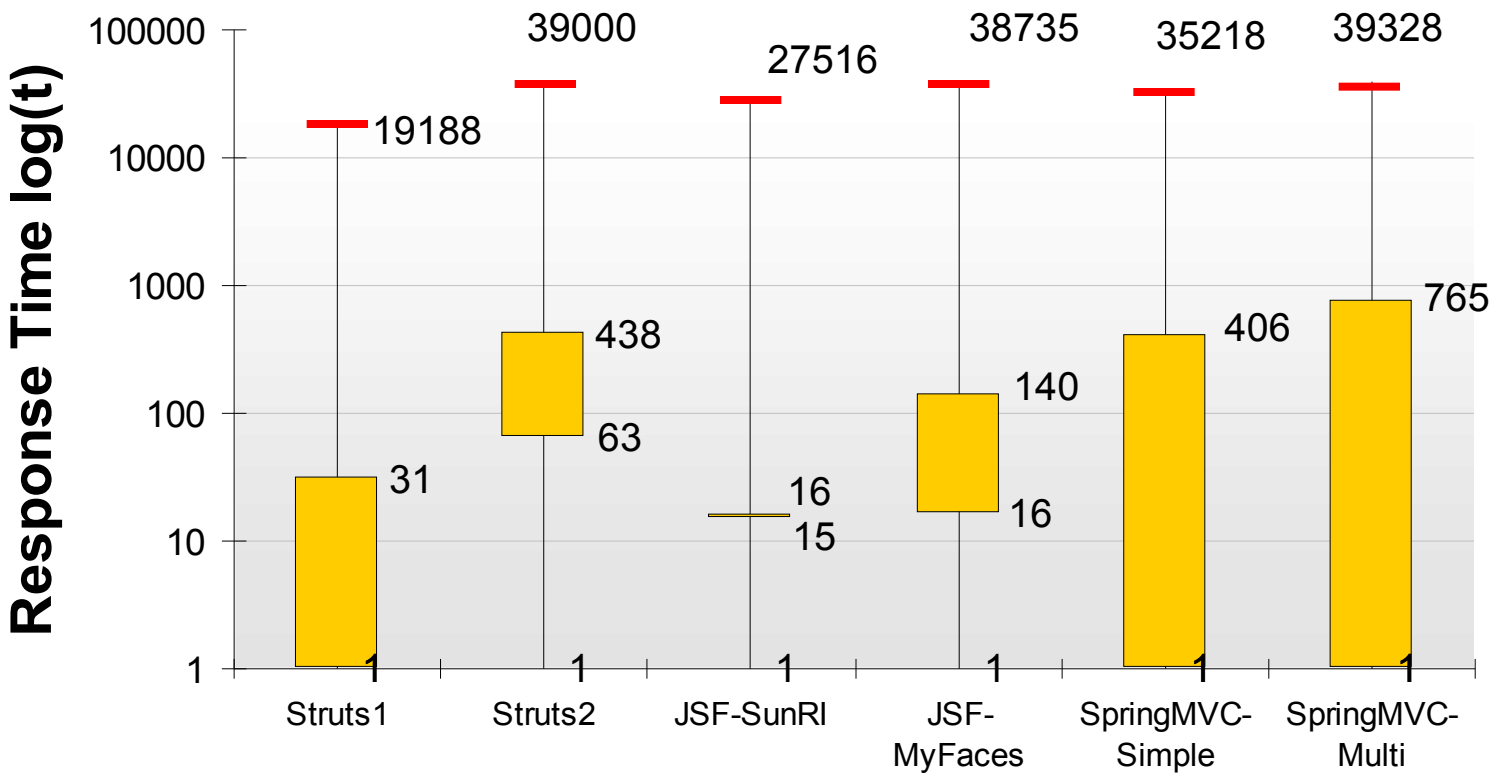
Registration Page with think time of 3s

Workload: 100-2-10-3

Framework	Median	Average	90% Line	Max	σ
Struts1	0	60	31	19188	755.91
Struts2	63	673	438	39000	3602.5
JSF-SunRI	15	176	16	27516	1801.05
JSF-MyFaces	16	753	140	38735	4152.54
SpringMVC-Simple	0	549	406	35218	3329.74
SpringMVC-Multi	0	287	765	39328	1352.28

Response Time stat plot*

Registration page for Workload 100-2-10-3



* Box span: Median - 90% Line

Max CPU Utilization*

	10-2-10-3	100-2-10-3
Struts1.x	13%	94%
Struts2.x	81%	98%
JSF-SunRI	78%	57%
JSF-MyFaces	100%	100%
SpringMVC-Simple	40%	42%
SpringMVC-Multi	25%	52%

* Windows Performance Counter – 1sec refresh interval

Memory Consumption*

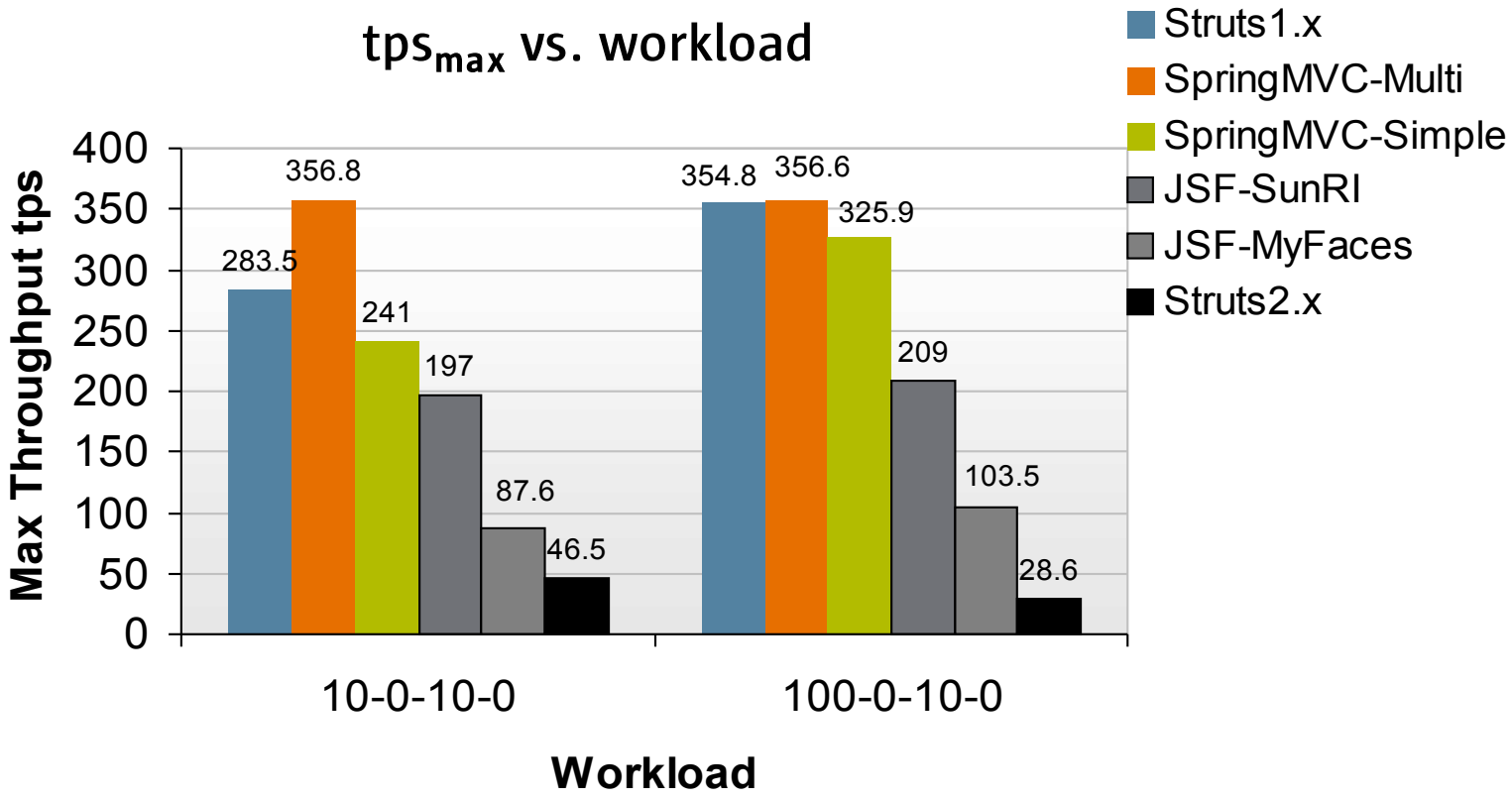
	10-2-10-3	100-2-10-3
Struts1.x	8.3MB	32.4MB
Struts2.x	16.6MB	44.9MB
JSF-SunRI	21.5MB	24.5MB
JSF-MyFaces	24.1MB	127.8MB
SpringMVC-Simple	13.5MB	35.2MB
SpringMVC-Multi	7.8MB	34.5MB

* Max memory recorded minus memory consumed at startup
* Windows Performance Counter – 1sec refresh interval

Test Results under Sustained Load Flood to evaluate Peak Performance

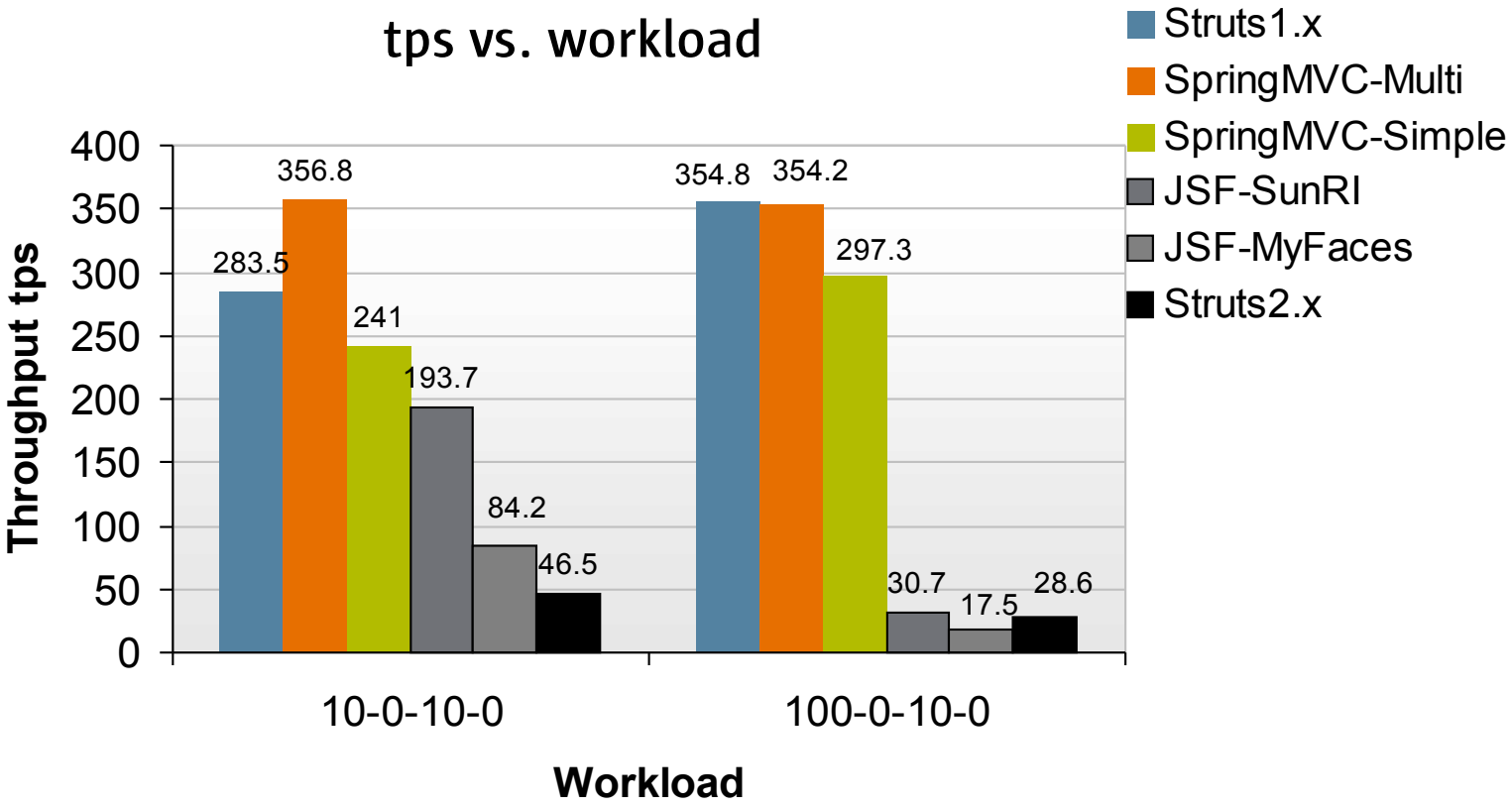
Maximum Throughput

No Think time



Overall Throughput

No Think time



Response Time (ms)

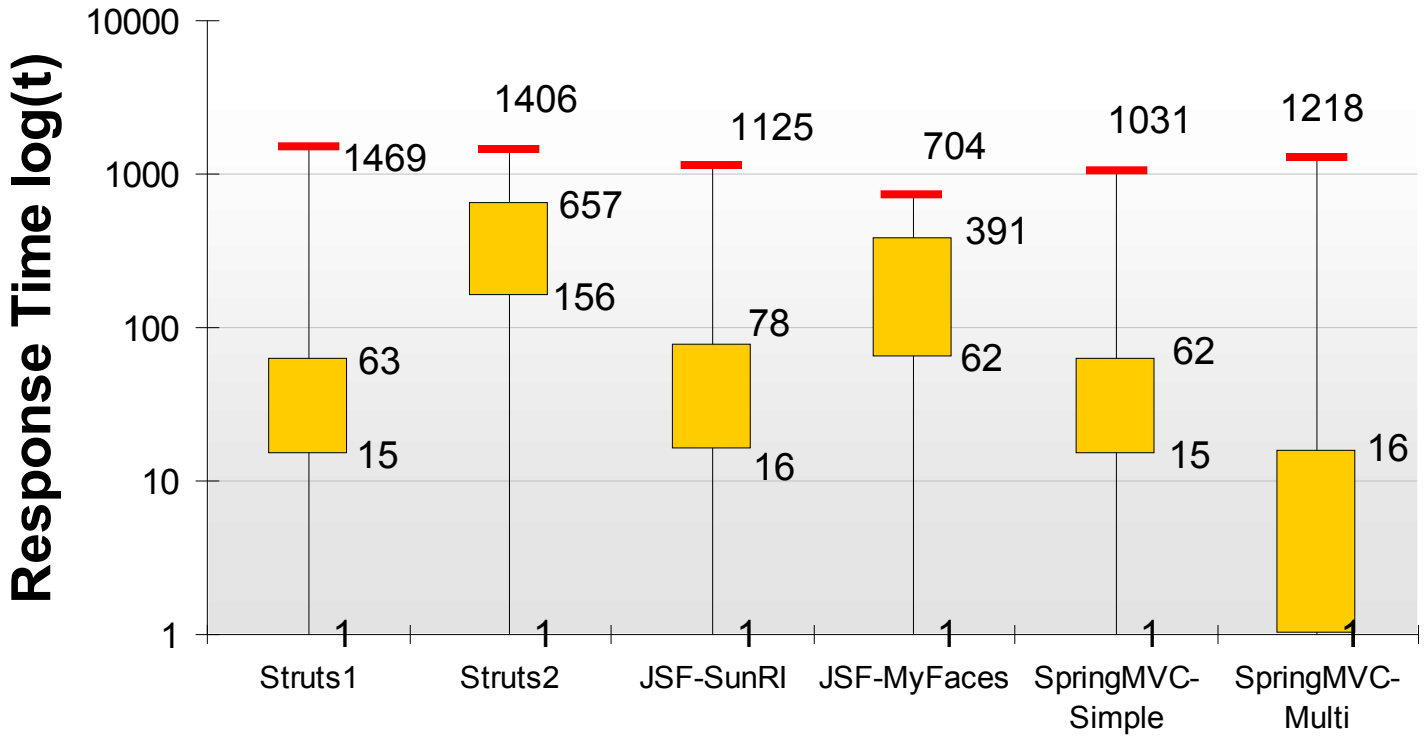
Registration Page

Workload: 10-0-10-0

Framework	Median	Average	90% Line	Max	σ
Struts1	15	40	63	1469	136.32
Struts2	156	253	657	1406	262.10
JSF-SunRI	16	40	78	1125	116.35
JSF-MyFaces	62	125	391	704	155.83
SpringMVC-Simple	15	33	62	1031	109.66
SpringMVC-Multi	0	16	16	1218	87.95

Response Time stat plot*

Registration page for Workload 10-0-10-0



* Box span: Median - 90% Line

Response Time (ms)

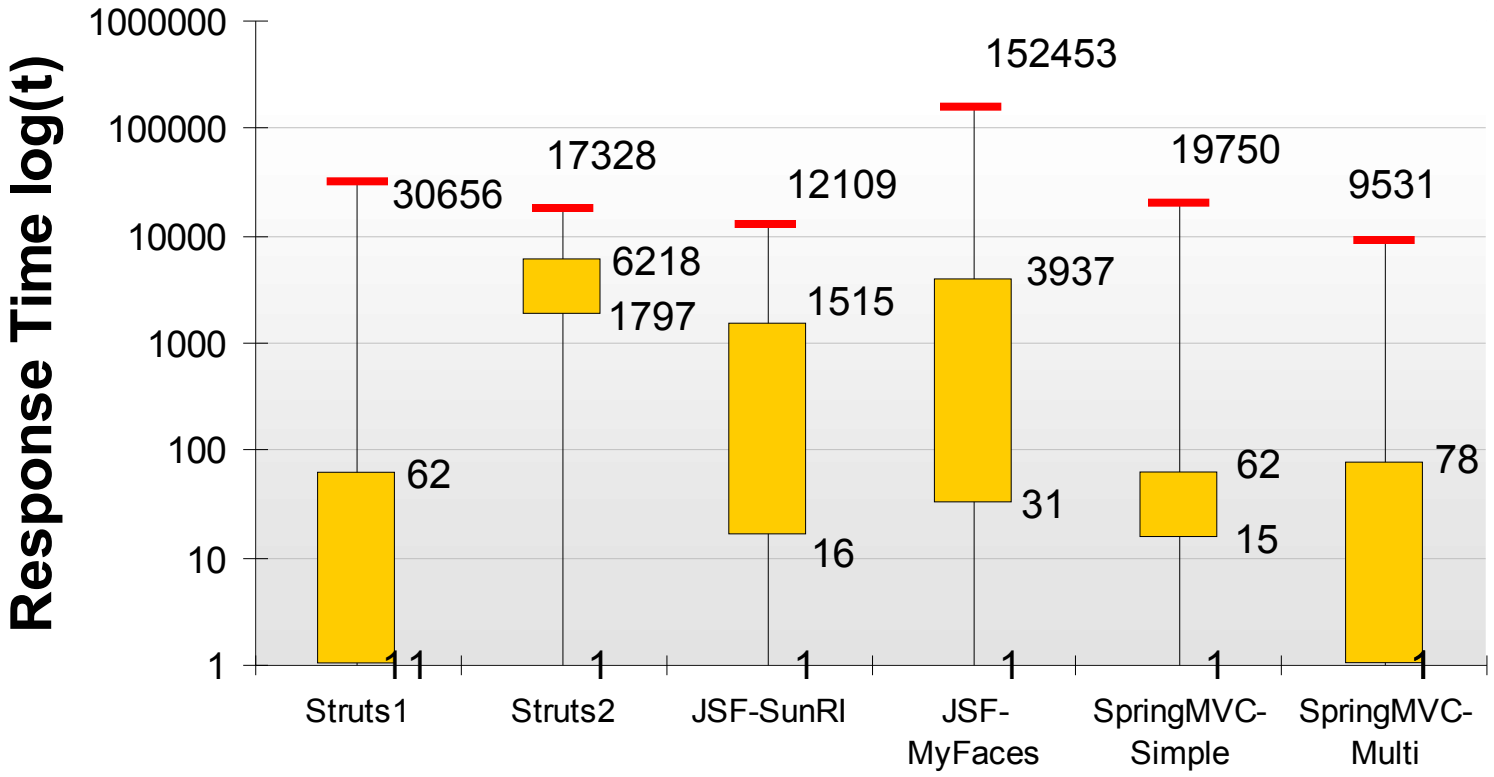
Registration Page

Workload: 100-0-10-0

Framework	Median	Average	90% Line	Max	σ
Struts1	0	445	62	30656	2870.29
Struts2	1797	2447	6218	17328	2614.29
JSF-SunRI	16	409	1515	12109	1100.85
JSF-MyFaces	31	2198	3937	152453	12752.54
SpringMVC-Simple	15	231	62	19750	1430.13
SpringMVC-Multi	0	215	78	9531	924.47

Response Time stat plot*

Registration page for Workload 100-0-10-0



* Box span: Median - 90% Line

Max CPU Utilization*

	10-0-10-0	100-0-10-0
Struts1.x	34%	46%
Spring MVC-Multi	27%	26%
Spring MVC-Simple	26%	50%
JSF-SunRI	94%	99%
JSF-MyFaces	99%	99%
Struts2.x	98%	100%

* Windows Performance Counter – 1sec refresh interval

Memory Configuration Requirement

Heap Space

	10-0-10-0	100-0-10-0
Struts1.x	64MB	64MB
Spring MVC-Multi	64MB	64MB
Spring MVC-Simple	64MB	64MB
JSF-SunRI	64MB	512MB*
JSF-MyFaces	64MB	512MB*
Struts2.x	128MB*	256MB*

* JVM Settings: OutOfMemory errors or excessive memory management when configured in lesser configuration

Response Errors

HTTP 500 Code

	10-0-10-0	100-0-10-0
Struts1.x	1	15
Spring MVC-Multi	0	1
Spring MVC-Simple	0	1
JSF-SunRI	0	0
JSF-MyFaces	0	0
Struts2.x	0	0

* Tag library Errors

Ease of Use

Rank

	Rank*
JSF	1
Struts2.x	2
Spring MVC-Simple	3
Spring MVC-Multi	4
Struts1.x	4

* 1 Very Easy, 2 – Easy, 3 – Medium, 4 – Not so easy

Agenda

- Benchmark Motivation
- Challenges and overall strategy
- Web Frameworks Overview
- Benchmark Application
- Road Test Configuration
- Road Test Results
- Conclusion Summary
- Future Work

Summary

- What is the message from these benchmarks?
- JSF SunRI performed well under loads with sufficient think time
- SpringMVC Multi-Action consistently performed well at max load
- JSF SunRI, MyFaces and Struts2 took no HTTP Errors
- Struts2 has the most predictable response time* under max load

* Registration Page Response Time

Summary Continued

- SpringMVC exhibits high variation in response times* under spread-out load
- Memory and CPU Consumption high in Struts2, SunRI and MyFaces
- Struts1 and MyFaces show isolated swings in response times*
- SunRI, MyFaces and Struts2 not able to sustain throughput at high loads [in that order]

* Registration Page Response Time

Agenda

- Benchmark Motivation
- Challenges and overall strategy
- Web Frameworks Overview
- Benchmark Application
- Road Test Configuration
- Road Test Results
- Conclusion Summary
- Future Work

Future Work

- Optimize integration with candidate frameworks
- Compare with other JSF technology implementations
- Compare other Servlet-based frameworks
- Compare with other non-Servlet based frameworks
- Compare features within the frameworks such as Validation, Localization etc.
- Impact of views via tag libraries specific to frameworks
- Impact of AJAX on response times and resource utilization

Questions?

For detailed information on this benchmark and related research, visit:

http://sujoebose.com

THANK YOU

Sujoe Bose,
Architect, Sabre Holdings.

TS-6517