Stuttgart, Germany, 2014-11-27

Evaluating the Prediction Accuracy of Generated Performance Models in Up- and Downscaling Scenarios

Symposium on Software Performance (SOSP) 2014

Andreas Brunnert[1], Stefan Neubig[1], Helmut Krcmar[2]

[1]fortiss GmbH, [2]Technische Universität München

fortiss GmbH
An-Institut Technische Universität München

Agenda

- Motivation and Vision
- Performance Model Generation
  - Data Collection
  - Data Aggregation
  - Model Generation
- Evaluation
  - SPECjEnterprise2010
  - Overhead Evaluation
  - Experiment Setup
  - Scenario Description
  - Scenario Results
    - Upscaling
    - Downscaling
- Future Work

Agenda

- Motivation and Vision
- Performance Model Generation
  - Data Collection
  - Data Aggregation
  - Model Generation
- Evaluation
  - SPECjEnterprise2010
  - Overhead Evaluation
  - Experiment Setup
  - Scenario Description
  - Scenario Results
    - Upscaling
    - Downscaling
- Future Work

Motivation & Vision

- Numerous performance modeling approaches are available to evaluate the performance (i.e., response time, throughput, resource utilization) of enterprise applications
- Performance models are especially useful for scenarios that cannot be tested on real systems, e.g.:
  - Scaling a system up or down in terms of the available hardware resources (e.g., number of CPU cores) during the capacity planning process.
- Creating a performance model requires considerable manual effort
  - → low adoption rates of performance models in practice

Motivation & Vision

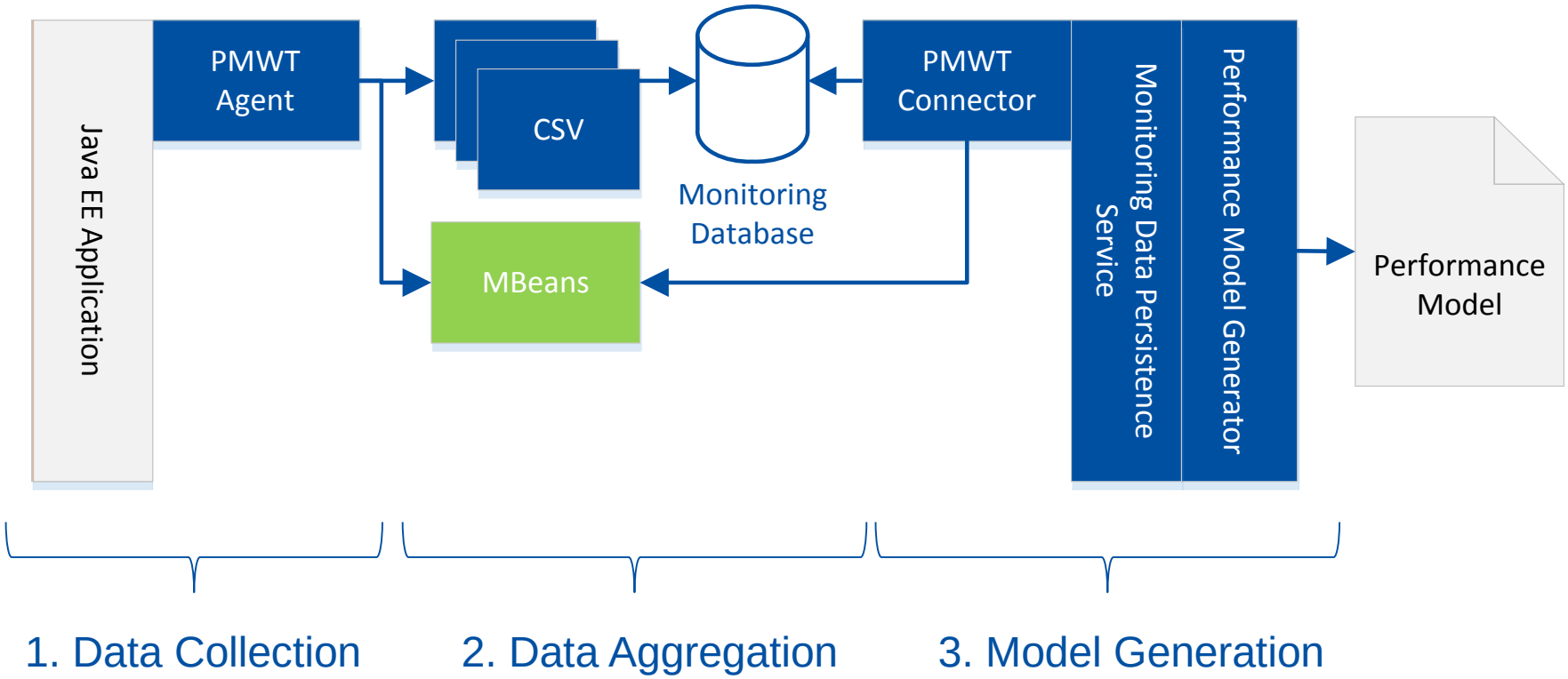
- To increase the adoption rates of performance models in practice!
  - …For that purpose, we have proposed an automatic performance model generation approach for Java Enterprise Edition (EE) applications.
  - …This work improves the existing approach by further reducing the effort and time for the model generation.
  - ... This work evaluates the prediction accuracy of these generated performance models in up- and downscaling scenarios, i.e.:
    - Increased and reduced number of CPU cores
    - Increased and reduced number of concurrent users

Agenda

- Motivation and Vision
- Performance Model Generation
  - Data Collection
  - Data Aggregation
  - Model Generation
- Evaluation
  - SPECjEnterprise2010
  - Overhead Evaluation
  - Experiment Setup
  - Scenario Description
  - Scenario Results
    - Upscaling
    - Downscaling
- Future Work

Performance Model Generation

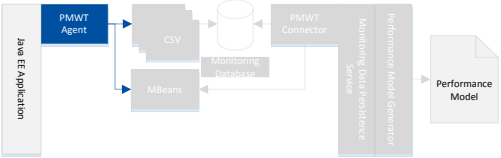
Overview

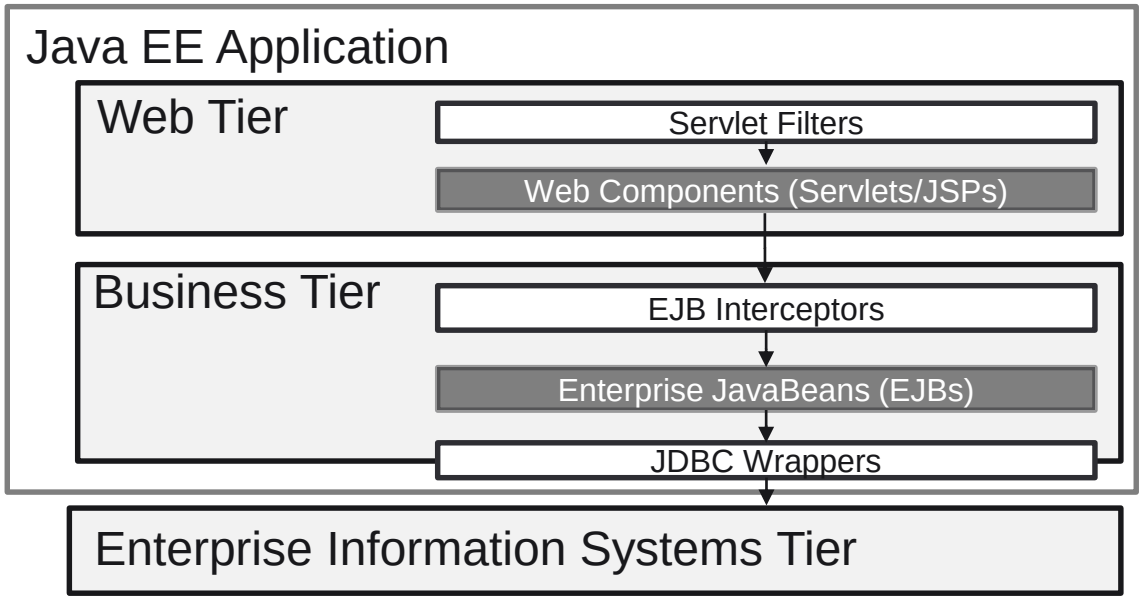


(adapted from Willnecker et al. (2014))

Performance Model Generation

Data Collection



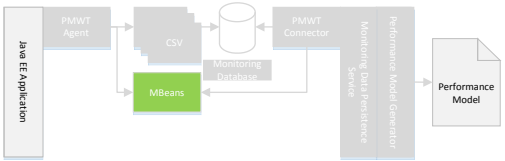


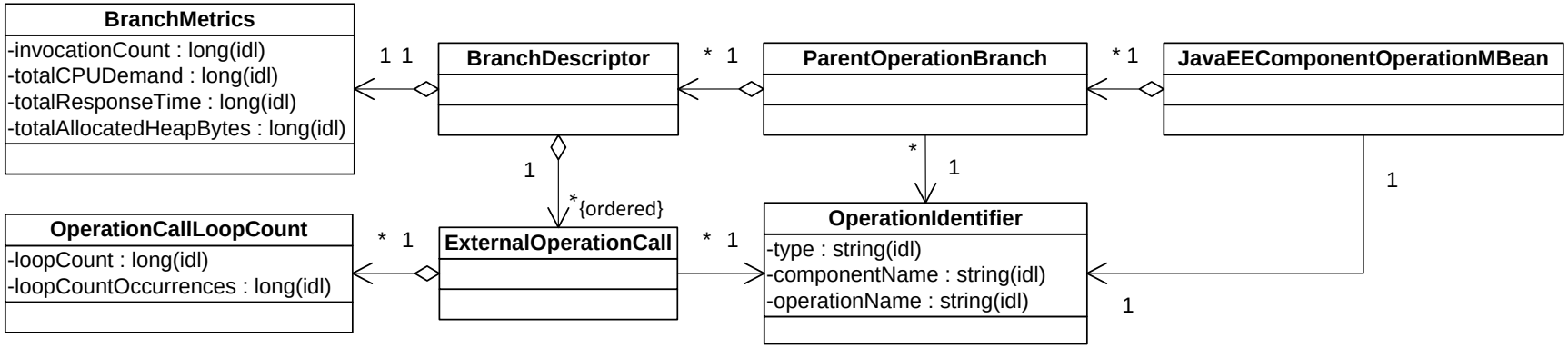
Data collected:

- EJB and Web components
- EJB and Web component operations
- EJB and Web component relationships on the level of single component operations
- Resource demands for single component operations (CPU, Memory)

Performance Model Generation

Data Aggregation

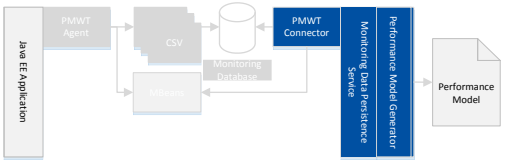




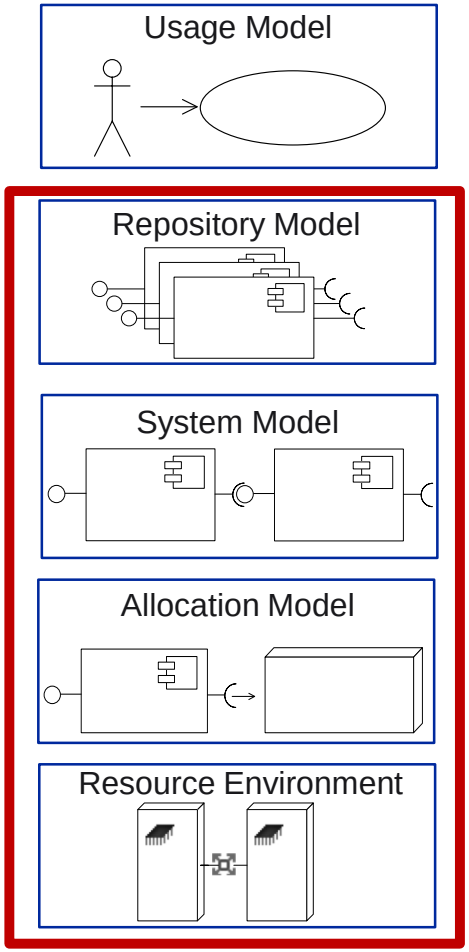
(Brunnert et al. 2014_1)

Performance Model Generation

Model Generation



- Except for the usage model, default models for all model layers of the Palladio Component Model (PCM) are generated automatically:
  - Repository model containing the components of an EA, their relationships and resource demands
  - System model containing the deployment units detected during the data collection (no single components)
  - A simple resource environment with one server and an allocation model that maps all deployment units to this server



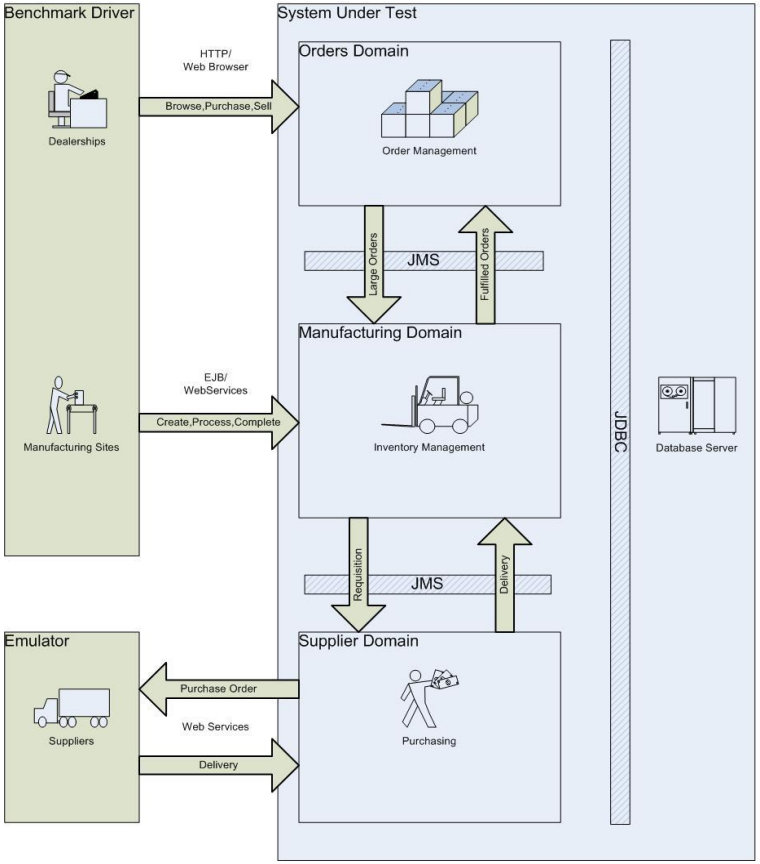
(adapted from a presentation for Brunnert et al. (2014_2)

Agenda

- Motivation and Vision
- Performance Model Generation
  - Data Collection
  - Data Aggregation
  - Model Generation
- Evaluation
  - SPECjEnterprise2010
  - Overhead Evaluation
  - Experiment Setup
  - Scenario Description
  - Scenario Results
    - Upscaling
    - Downscaling
- Future Work

Evaluation

SPECjEnterprise2010



SPECjEnterprise 2010 Architecture (SPEC, 2009)

- Benchmark Driver / Emulator
  - Simulates interactions with the SUT
  - Defines workload

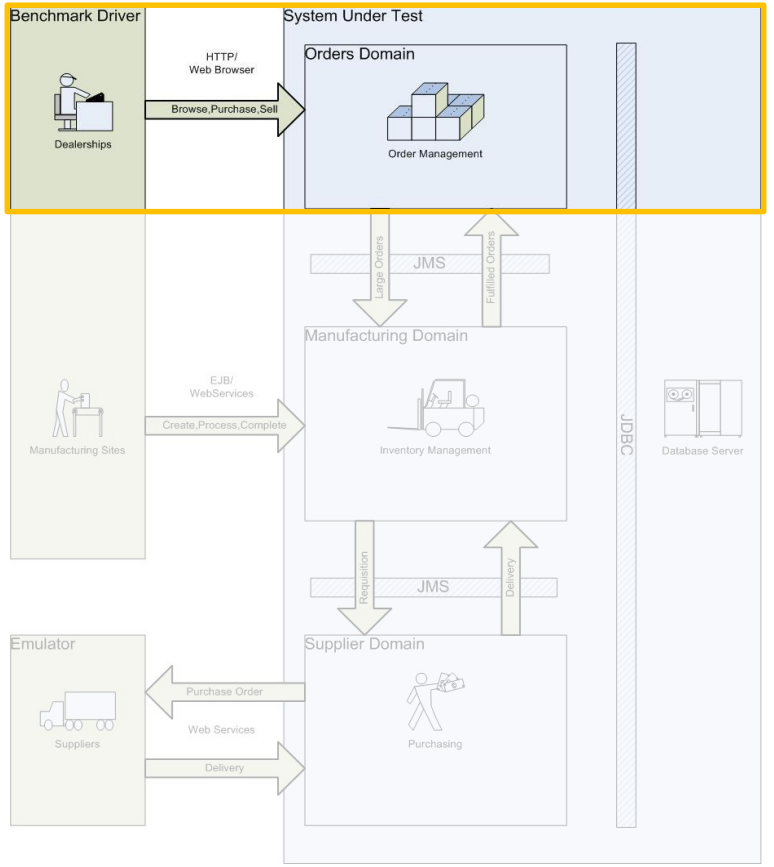
- Automobile Manufacturer
  - Orders Domain (CRM)
  - Manufacturing Domain
  - Supplier Domain (SCM)

- Database Server

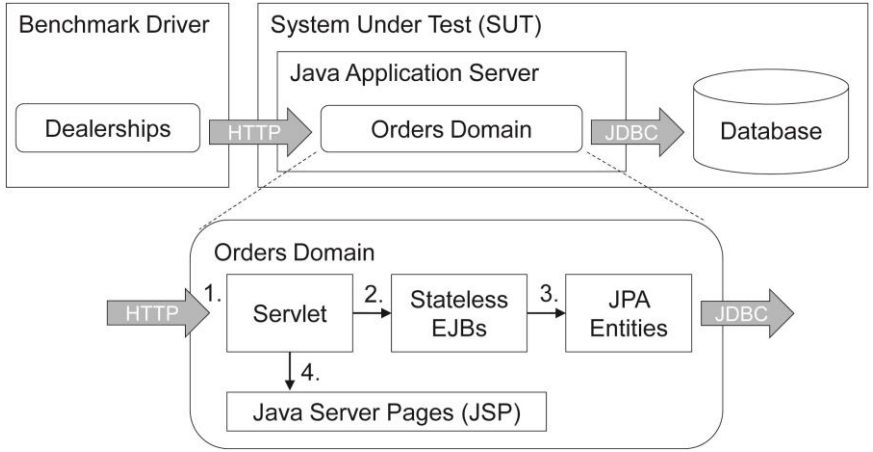
Evaluation

SPECjEnterprise2010

(Standard Performance Evaluation Corporation 2009)



SPECjEnterprise 2010 Architecture
(SPEC, 2009)



- Workload defined by driver
- Business Transactions
  - Browse, Manage, Purchase
  - Predefined sequence of HTTP requests including probabilities

Orders Domain Architecture
(Brunnert et al. 2013)

Evaluation

Overhead Evaluation 1/2

- Monitoring code is called before **and** after each monitored invocation
  → Considerable instrumentation overhead!

- Overhead Evaluation: CPU & Heap vs. CPU only
  - 4 CPU cores
  - 20 GB RAM
  - 600 Users
  - Only steady state data is collected

Evaluation

Overhead Evaluation 2/2

Component Operation		CPU & Heap				CPU only	
		All levels		Top level		All levels	Top level
		CPU	Heap	CPU	Heap	CPU	CPU
1	app.sellinventory	1.023 ms	33,650 B	3.001 ms	225,390 B	0.756 ms	3.003 ms
2	CustomerSession.sellInventory	0.785 ms	60,450 B	-	-	0.731 ms	-
3	CustomerSession.getInventories	0.594 ms	49,540 B	-	-	0.548 ms	-
4	OrderSession.getOpenOrders	0.954 ms	70,600 B	-	-	0.878 ms	-
5	dealerinventory.jsp.sellinventory	0.108 ms	16,660 B	-	-	0.103 ms	-
Total Resource Demand		3.464 ms	230,900 B	3.001 ms	225,390 B	3.015 ms	3.003 ms
Mean Data Collection Overhead		0.116 ms		1378 B		0.003 ms	

(Brunnert et al. 2014_1)

Evaluation

Overhead Evaluation 2/2

Component Operation		CPU & Heap				CPU only	
		All levels		Top level		All levels	Top level
		CPU	Heap	CPU	Heap	CPU	CPU
1	app.sellinventory	1.023 ms	33,650 B	3.001 ms	225,390 B	0.756 ms	3.003 ms
2	CustomerSession.sellInventory	0.785 ms	60,450 B	-	-	0.731 ms	-
3	CustomerSession.getInventories	0.594 ms	49,540 B	-	-	0.548 ms	-
4	OrderSession.getOpenOrders	0.954 ms	70,600 B	-	-	0.878 ms	-
5	dealerinventory.jsp.sellinventory	0.108 ms	16,660 B	-	-	0.103 ms	-
Total Resource Demand		3.464 ms	230,900 B	3.001 ms	225,390 B	3.015 ms	3.003 ms
Mean Data Collection Overhead			0.116 ms	1378 B		0.003 ms	

(Brunnert et al. 2014_1)

Evaluation

Overhead Evaluation 2/2

Component Operation		CPU & Heap				CPU only	
		All levels		Top level		All levels	Top level
		CPU	Heap	CPU	Heap	CPU	CPU
1	app.sellinventory	1.023 ms	33,650 B	3.001 ms	225,390 B	0.756 ms	3.003 ms
2	CustomerSession.sellInventory	0.785 ms	60,450 B	-	-	0.731 ms	-
3	CustomerSession.getInventories	0.594 ms	49,540 B	-	-	0.548 ms	-
4	OrderSession.getOpenOrders	0.954 ms	70,600 B	-	-	0.878 ms	-
5	dealerinventory.jsp.sellinventory	0.108 ms	16,660 B	-	-	0.103 ms	-
Total Resource Demand		3.464 ms	230,900 B	3.001 ms	225,390 B	3.015 ms	3.003 ms
Mean Data Collection Overhead		0.116 ms		1378 B		0.003 ms	

(Brunnert et al. 2014_1)

Without heap monitoring, the Mean Data Collection Overhead drops dramatically!

→ We focus on collecting CPU demand only!

Evaluation

Experiment Setup

- Model generation
  - Collect **only** CPU demands using Servlet Filters and EJB Interceptors
  - Only steady state data is collected
  - SUT under moderate load (~ 40-60 %)

- Model simulation
  - Generated PCM models are taken as input for SimuCom
  - Results compared with benchmark runs

- Benchmark runs
  - Servlet Filter at front controller to measure response times and throughput
  - JMX connection to measure CPU utilization of JVM
  - Only steady state data is collected
  - Performed three times (same weight)

Evaluation

Scenario Description

- Upscaling: *How many resources do I need for an increasing workload?*
  - Workload: 600 Users → 900 Users → 1200 Users
  - CPU: 4 Cores → 6 Cores → 8 Cores
  - Model generation: 4 cores (~52 % load)

- Downscaling: *How many resources do I need for the given workload?*
  - Workload: 800 Users
  - CPU: 8 Cores → 6 Cores → 4 Cores
  - Model generation: 8 cores (~39 % load)

Evaluation

Scenario Results: Upscaling 1/2

CPU Cores	Number of Users	Business Transaction	Measured RT	Simulated RT	**Prediction Error**	Measured Throughput	Simulated Throughput	**Prediction Error**	Measured CPU	Simulated CPU	**Prediction Error**
4	600	Browse	63.23 ms	65.06 ms	**2.91 %**	1820.6	1813.1	**0.41 %**	48.76 %	46.87 %	**3.88 %**
		Manage	11.58 ms	13.28 ms	**14.71 %**	906.8	917.3	**1.16 %**			
		Purchase	8.27 ms	9.73 ms	**17.67 %**	904.9	900.3	**0.50 %**			
6	900	Browse	69.25 ms	57.56 ms	**16.89 %**	2708.3	2721.5	**0.49 %**	51.72 %	46.85 %	**9.42 %**
		Manage	12.54 ms	11.95 ms	**4.69 %**	1354.3	1354.4	**0.01 %**			
		Purchase	8.95 ms	8.72 ms	**2.60 %**	1352.4	1368.1	**1.16 %**			
8	1200	Browse	88.82 ms	56.25 ms	**36.66 %**	3617.8	3641.9	**0.67 %**	57.34 %	46.97 %	**18.09 %**
		Manage	14.13 ms	11.64 ms	**17.67 %**	1806.4	1795.0	**0.63 %**			
		Purchase	9.31 ms	8.46 ms	**9.15 %**	1811.6	1819.2	**0.42 %**			

(Brunnert et al. 2014_1)

Evaluation

Scenario Results: Upscaling 1/2

CPU Cores	Number of Users	Business Transaction	Measured RT	Simulated RT	**Prediction Error**	Measured Throughput	Simulated Throughput	**Prediction Error**	Measured CPU	Simulated CPU	**Prediction Error**
4	600	Browse	63.23 ms	65.06 ms	**2.91 %**	1820.6	1813.1	**0.41 %**	48.76 %	46.87 %	**3.88 %**
		Manage	11.58 ms	13.28 ms	**14.71 %**	906.8	917.3	**1.16 %**			
		Purchase	8.27 ms	9.73 ms	**17.67 %**	904.9	900.3	**0.50 %**			
6	900	Browse	69.25 ms	57.56 ms	**16.89 %**	2708.3	2721.5	**0.49 %**	51.72 %	46.85 %	**9.42 %**
		Manage	12.54 ms	11.95 ms	**4.69 %**	1354.3	1354.4	**0.01 %**			
		Purchase	8.95 ms	8.72 ms	**2.60 %**	1352.4	1368.1	**1.16 %**			
8	1200	Browse	88.82 ms	56.25 ms	**36.66 %**	3617.8	3641.9	**0.67 %**	57.34 %	46.97 %	**18.09 %**
		Manage	14.13 ms	11.64 ms	**17.67 %**	1806.4	1795.0	**0.63 %**			
		Purchase	9.31 ms	8.46 ms	**9.15 %**	1811.6	1819.2	**0.42 %**			

(Brunnert et al. 2014_1)

Evaluation

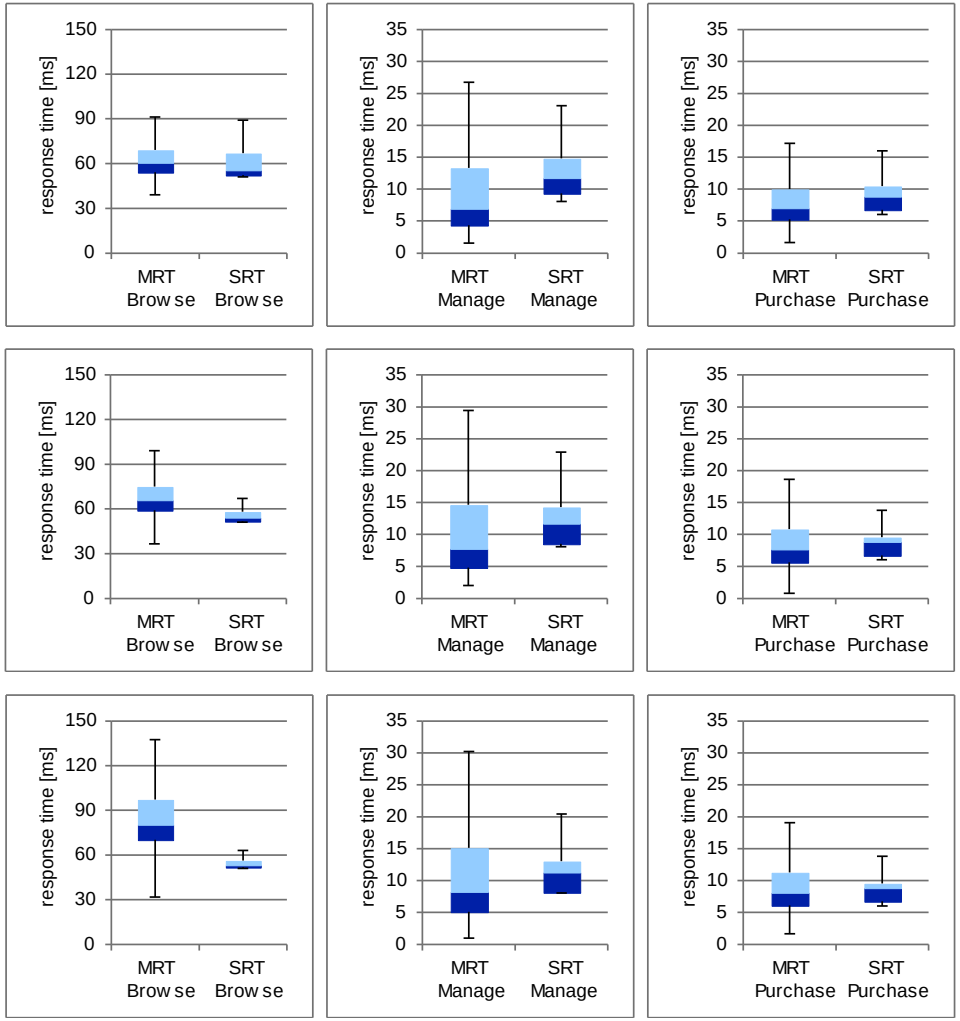
Scenario Results: Upscaling 2/2

(Brunnert et al. 2014_1)

4 CPU cores, 600 users

6 CPU cores, 900 users

8 CPU cores, 1200 users



Performance Management Group

fortiss

Evaluation

Scenario Results: Downscaling 1/2

CPU Cores	Number of Users	Business Transaction	Measured RT	Simulated RT	**Prediction Error**	Measured Throughput	Simulated Throughput	**Prediction Error**	Measured CPU	Simulated CPU	**Prediction Error**
8	800	Browse	71.54 ms	64.03 ms	**10.50 %**	2413.9	2415.8	**0.08 %**	37.41 %	35.17 %	**5.99 %**
		Manage	12.96 ms	12.64 ms	**2.49 %**	1203.5	1209.2	**0.48 %**			
		Purchase	9.36 ms	9.33 ms	**0.25 %**	1215.9	1228.7	**1.05 %**			
6	800	Browse	67.62 ms	66.03 ms	**2.35 %**	2413.9	2425.4	**0.48 %**	46.38 %	46.94 %	**1.21 %**
		Manage	12.52 ms	13.08 ms	**4.45 %**	1202.0	1196.6	**0.45 %**			
		Purchase	9.05 ms	9.64 ms	**6.57 %**	1208.2	1215.0	**0.56 %**			
4	800	Browse	71.15 ms	87.46 ms	**22.92 %**	2437.0	2420.8	**0.66 %**	65.60 %	70.27 %	**7.12 %**
		Manage	12.98 ms	17.04 ms	**31.29 %**	1199.7	1193.5	**0.51 %**			
		Purchase	8.93 ms	12.88 ms	**44.33 %**	1211.6	1212.1	**0.04 %**			

(Brunnert et al. 2014_1)

Evaluation

Scenario Results: Downscaling 1/2

CPU Cores	Number of Users	Business Transaction	Measured RT	Simulated RT	**Prediction Error**	Measured Throughput	Simulated Throughput	**Prediction Error**	Measured CPU	Simulated CPU	**Prediction Error**
8	800	Browse	71.54 ms	64.03 ms	**10.50 %**	2413.9	2415.8	**0.08 %**	37.41 %	35.17 %	**5.99 %**
		Manage	12.96 ms	12.64 ms	**2.49 %**	1203.5	1209.2	**0.48 %**			
		Purchase	9.36 ms	9.33 ms	**0.25 %**	1215.9	1228.7	**1.05 %**			
6	800	Browse	67.62 ms	66.03 ms	**2.35 %**	2413.9	2425.4	**0.48 %**	46.38 %	46.94 %	**1.21 %**
		Manage	12.52 ms	13.08 ms	**4.45 %**	1202.0	1196.6	**0.45 %**			
		Purchase	9.05 ms	9.64 ms	**6.57 %**	1208.2	1215.0	**0.56 %**			
4	800	Browse	71.15 ms	87.46 ms	**22.92 %**	2437.0	2420.8	**0.66 %**	65.60 %	70.27 %	**7.12 %**
		Manage	12.98 ms	17.04 ms	**31.29 %**	1199.7	1193.5	**0.51 %**			
		Purchase	8.93 ms	12.88 ms	**44.33 %**	1211.6	1212.1	**0.04 %**			

(Brunnert et al. 2014_1)

Evaluation

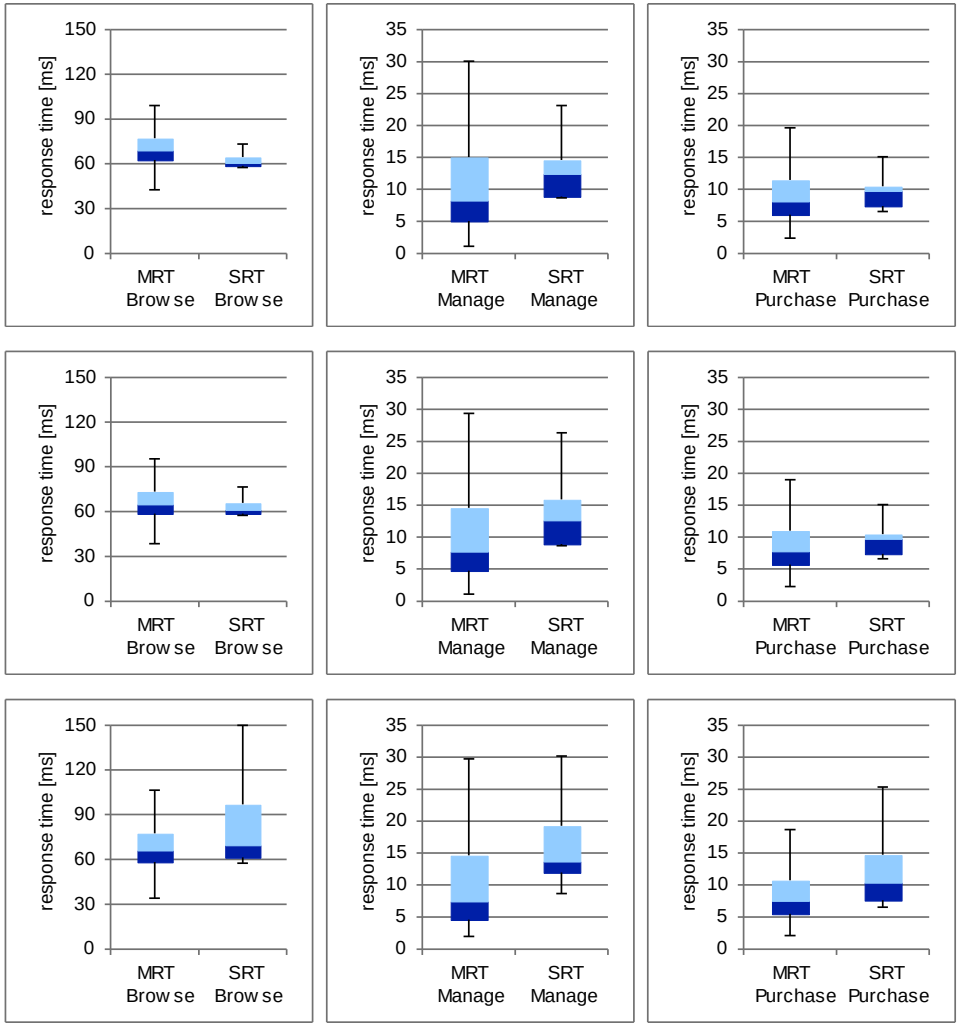
Scenario Results: Downscaling 2/2

(Brunnert et al. 2014_1)

8 CPU cores, 800 users

6 CPU cores, 800 users

4 CPU cores, 800 users



Agenda

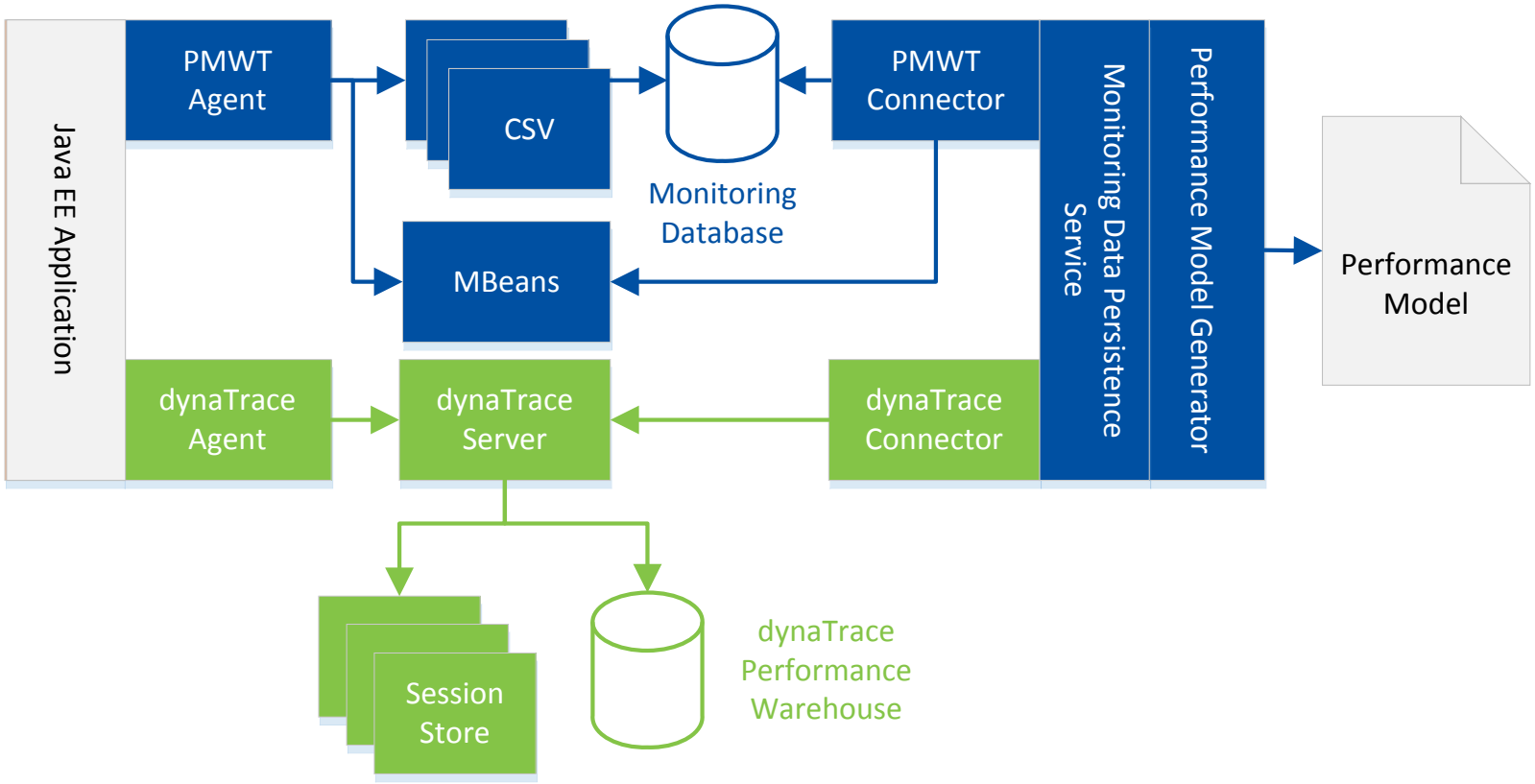
- Motivation and Vision
- Performance Model Generation
  - Data Collection
  - Data Aggregation
  - Model Generation
- Evaluation
  - SPECjEnterprise2010
  - Overhead Evaluation
  - Experiment Setup
  - Scenario Description
  - Scenario Results
    - Upscaling
    - Downscaling
- Future Work

Future Work

- Usage model generation
  - Master Thesis recently finished based on session traces
  - Christian Vögele currently extends this approach
- Representing distributed systems
  - Distributing the transaction ID within distributed Java EE environments using the Application Response Measurement (ARM) standard or similar means
  - Aggregating data from multiple Java EE instances
    - Recently finished a prototype for distributed deployments communicating over SOAP
  - Setting thread limits for each tier (Web, EJB, JDBC Conn. Pools)
- Supporting new component types:
  - JSF, web-services, Message driven beans, Other Java EE 7.0 enhancements, e.g. Batch Jobs, ..

Future Work

- Using APM data for the model generation (collaboration with Dynatrace)



(Willnecker et al. 2014)

Bibliography

Brunnert, A.; Vögele, C.; Krcmar, H. (2013): Automatic performance model generation for java enterprise edition (ee) applications, in: M. S. Balsamo, W. J. Knottenbelt, A. Marin (Eds.), Computer Performance Engineering, Vol. 8168 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 74-88.

Brunnert, A.; Neubig, S.; Krcmar, H. (2014): Evaluating the prediction accuracy of generated performance models in up- and downscaling scenarios, in: Proceedings of the Symposium on Software Performance, SOSP 14, 2014, pp. 113-130.

Brunnert, A.; Wischer, K.; Krcmar, H. (2014): Using architecture-level performance models as resource profiles for enterprise applications, in: Proceedings of the 10th International ACM Sigsoft Conference on Quality of Software Architectures, QoSA '14, ACM, New York, NY, USA, 2014, pp. 53-62.

Standard Performance Evaluation Corporation (SPEC) (2009): SPECjEnterprise2010 Design Document. http://www.spec.org/jEnterprise2010/docs/DesignDocumentation.html, accessed at 2014-11-26

Willnecker, F.; Brunnert, A.; Gottesheim, W.; Krcmar, H. (2014): Using dynatrace monitoring data for generating performance models of Java EE applications. Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering, Austin, TX, USA, (accepted for publication, to appear).

CONTACT US

Andreas Brunnert,
Stefan Neubig
performancegroup@fortiss.org
pmw.fortiss.org