Utilizing PCM for Online Capacity Management of Component-Based Software Systems

André van Hoorn

Software Engineering Group, University of Kiel
http://se.informatik.uni-kiel.de/

Nov. 18, 2011 @ Palladio Days, Karlsruhe
Context of this Work
Online Capacity Management for Increased Resource Efficiency

Introduction

Adaptive Capacity Management

- Business-critical software systems
• Business-critical software systems

• Quality of service (performance, availability, . . . )
Context of this Work

Online Capacity Management for Increased Resource Efficiency

Introduction ▸ Adaptive Capacity Management

- Business-critical software systems
- Quality of service (performance, availability, . . .)
- Varying workloads + static capacity management
Context of this Work
Online Capacity Management for Increased Resource Efficiency

Introduction

- Adaptive Capacity Management

- Business-critical software systems
- Quality of service (performance, availability, ...)
- Varying workloads + static capacity management

Problem: Overprovisioning — unnecessarily high operating costs

Underutilized resources during medium/low workload periods

Goal: Increase resource efficiency while meeting SLAs

- **SLAStic** [van Hoorn et al. 2009a, van Hoorn 2011]:
  - Online capacity management employing architecture-based runtime reconfiguration
1. Introduction — Adaptive Capacity Management
2. SLAStic Approach
3. Extracting SLAStic Models via Dynamic Analysis
4. Utilizing PCM for SLAStic
5. Conclusions
SLAStic Runtime Reconfiguration Operations

1. (De-)Replication of Software Components

2. Migration of Software Components

3. (De-)Allocation of Execution Containers

André van Hoorn (CAU Kiel)

PCM for Online Capacity Management of C-B Systems

Nov. 18, 2011 @ Palladio Days
SLAStic Runtime Reconfiguration Operations

SLAStic Approach ▶ Overview

1. (De-)Replication of Software Components
   - replicate (component: AssemblyComponent, to: ExecutionContainer)
   - dereplicate (component: DeploymentComponent)

2. Migration of Software Components
   - migrate (component: DeploymentComponent, to: ExecutionContainer)

3. (De-)Allocation of Execution Containers
   - deallocate (container: ExecutionContainer)
   - allocate (containerType: ExecutionContainerType)
SLAStic Approach ▶ Overview

Online Capacity Management Framework

instrumented, runtime reconfigurable s/w system

SLAStic. MONITORING

SLAStic. RECONFIGURATION

SLAStic. CONTROL

André van Hoorn (CAU Kiel)

PCM for Online Capacity Management of C-B Systems

Nov. 18, 2011 @ Palladio Days 5 / 19
SLAStic.MONITORING utilizes Kieker [van Hoorn et al. 2009b]

Online Capacity Management Framework (cont’d)

SLAStic Approach ▶ Overview

- Instrumented, runtime reconfigurable s/w system
- Raw monitoring records
- Reconfiguration actions
- Reconfiguration plans

Monitoring Log/Stream
- E.g., file system, database, message-oriented middleware, Monitoring Log/Stream

Monitoring Record
- E.g., AOP-based method call interception
- E.g., trace information, workload, response times, resource utilization, loop counts

Analysis Plug-In
- E.g., architecture reconstruction, performance evaluation, online adaptation control, failure diagnosis

Kieker.Monitoring
- Kieker.Analysis

André van Hoorn (CAU Kiel)
PCM for Online Capacity Management of C-B Systems
Nov. 18, 2011 @ Palladio Days
Usage Scenarios of the SLAStic Framework

Online Capacity Management Framework (cont’d)

SLAStic Approach ➤ Overview

1 Online analysis (production/lab system)

2 Offline analysis (log replay)

3 Simulation-based analysis
System Partition (also used as runtime model)

1. Type repository (e.g., component types, interfaces, connector types, execution container types)

Example type repository
**System Partition** (also used as runtime model)

1. **Type repository** (e.g., component types, interfaces, connector types, execution container types)
2. **Component assembly** (e.g., assembly of components via connectors)

Example component assembly
System Partition (also used as runtime model)

1. **Type repository** (e.g., component types, interfaces, connector types, execution container types)
2. **Component assembly** (e.g., assembly of components via connectors)
3. **Execution environment** (e.g., execution containers and interconnection via links)

Example execution environment
Component deployment
System Partition (cont’d)
SLAStic Approach ▶ Meta Model

**System Partition** (also used as runtime model)

1. **Type repository** (e.g., component types, interfaces, connector types, execution container types)
2. **Component assembly** (e.g., assembly of components via connectors)
3. **Execution environment** (e.g., execution containers and interconnection via links)
4. **Component deployment** (mapping: assembly components → containers)

Example component deployment

[Diagram showing assembly and deployment of components]
**System Partition** (also used as runtime model)

1. **Type repository** (e.g., component types, interfaces, connector types, execution container types)
2. **Component assembly** (e.g., assembly of components via connectors)
3. **Execution environment** (e.g., execution containers and interconnection via links)
4. **Component deployment** (mapping: assembly components → containers)

---

**SLAstic2PCM** transforms SLAstic model to PCM [Günther 2011]

- **SLAstic-System-Modell**: TypeRepository, ExecutionEnvironment, ComponentAssembly, ComponentDeployment
- **SLAstic2PCM**: Ausführung durch: Eclipse-Launcher, Kommandozeile, Java-API
- **PCM-Modell**: ResourceType, Repository, ResourceEnvironment, Allocation, System, UsageModel

---

Adaptation/Reconfiguration
Completions/Decorations
SLAStic Approach ▶ Meta Model

Completions/Decorations

- **Adaptation / Reconfiguration** (e.g., plans, operations, capabilities, properties)

---

PCM-specific implementation of operations part of **SLAStic.SIM** [von Massow et al. 2011]


(As presented at Palladio Days 2010)
**Completions/Decorations**

- **Adaptation / Reconfiguration** (e.g., plans, operations, capabilities, properties)
- **Measurement** (e.g., workload, timing, utilization)

---

**OperationExecution**
- traceId : long
- eoi : int
- ess : int
- tin : long
- tout : long
- sessionId : String

**DeploymentComponentOperationExecution**
- deploymentComponent : DeploymentComponent
- operation : Operation

**CPUUtilization**
- user : double
- system : double
- wait : double
- nice : double
- irq : double
- combined : double
- idle : double

**ResourceMeasurement**
- timestamp : long
- utilization
  - resource

**MemSwapUsage**
- memUsedBytes : long
- memFreeBytes : long
- swapUsedBytes : long
- swapFreeBytes : long

---

*André van Hoorn (CAU Kiel)*

*PCM for Online Capacity Management of C-B Systems*

*Nov. 18, 2011 @ Palladio Days*
Completions/Decorations

- **Adaptation / Reconfiguration** (e.g., plans, operations, capabilities, properties)
- **Measurement** (e.g., workload, timing, utilization)
- **Usage** (e.g., operation call frequencies, calling relationships)
Completions/Decorations

- Adaptation / Reconfiguration (e.g., plans, operations, capabilities, properties)
- Measurement (e.g., workload, timing, utilization)
- Usage (e.g., operation call frequencies, calling relationships)
- ... — e.g., “cost” profiles (not yet implemented)
1. Introduction — Adaptive Capacity Management

2. SLAStic Approach

3. Extracting SLAStic Models via Dynamic Analysis

4. Utilizing PCM for SLAStic

5. Conclusions
1. **Raw monitoring records** delivered by Kieker, e.g.
   - OperationExecutionRecord (incl. timing & tracing information)
   - ResourceUtilizationRecord and MemSwapUsageRecord

2. **Processing of records** by SLAStic.MONITORING
   1. Possible abstraction of component/operation names etc.
   2. Lookup/creation of architectural entities using ModelManager
   3. Transformation of raw monitoring records into monitoring events
   4. Send monitoring events to complex-event processing (CEP) engine

3. **Trace reconstruction**
   1. TraceReconstructor registers match/recognize CEP statement to collect executions grouped by trace ID (online)
   2. Send each reconstructed message trace to CEP engine

4. **Update of system and usage model**
   1. UsageAndSystemModelUpdater registers CEP statement to collect valid message traces
   2. Process each trace and update models using ModelManager
Utilizing PCM in SLAStic

1. Simulation-based analysis
   - SLAStic.SIM as substitute for real system
   - Evaluation of approach, adaptation plans etc.

2. PCM @ Runtime
   - PCM instance decorated by SLAStic model

3. Online performance prediction (e.g., SLAStic.SIM)

Semi-automated extraction of PCM instances

1. Extraction of SLAStic instance by dyn. analysis
2. SLAStic2PCM transformation [Günther 2011]
3. Assumption: Manual refinement/calibration (offline)
SLAstic.SIM
Simulation of Reconfigurable PCM instances
Utilizing PCM for SLAstic

Simulation driven by recorded (varying) workload
Simulation of PCM models with SLAstic’s reconfiguration capabilities
(limitations in terms of supported PCM features)

(As presented at Palladio Days 2010)
SLAStic.SIM
Simulation of Reconfigurable PCM instances
Utilizing PCM for SLAStic

- Simulation driven by recorded (varying) workload
- Simulation of PCM models with SLAStic’s reconfiguration capabilities
- (limitations in terms of supported PCM features)

(As presented at Palladio Days 2010)
SLAStic2PCM Transformation

- SLAStic2PCM implemented as an ATL-based M2M transformation

1. Transformation of SLAStic System Model to PCM counter parts pretty straight-forward

<table>
<thead>
<tr>
<th>SLAStic</th>
<th>PCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TypeRepository</td>
<td>Repository, ResourceRepository</td>
</tr>
<tr>
<td>ComponentAssembly</td>
<td>System</td>
</tr>
<tr>
<td>ExecutionEnvironment</td>
<td>ResourceEnvironment</td>
</tr>
<tr>
<td>ComponentDeployment</td>
<td>Allocation</td>
</tr>
</tbody>
</table>

2. Generation of RDSEFFs (detailed on next slide) and Usage Models
   - RDSEFFs: Generated from CallingRelationships and OperationCallFrequencyS
   - Usage Scenario (open workload) based on SystemProvidedInterfaceDelegationConnectorFrequencies and observation period
Extraction of RDSEFFs (Pattern)
SLAStic2PCM Transformation (cont’d)
Utilizing PCM for SLAStic ▶ SLAStic2PCM M2M Transformation

For each Operation

1. Create StartAction
2. Create InternalAction for initialization
3. Create LoopAction for each Interface Signature called by Operation
   1. Create StartAction + InternalAction for initialization
   2. Create ExternalCallAction
   3. Create InternalAction for termination + StopAction
4. Create InternalAction for termination
5. Create StopAction

 André van Hoorn (CAU Kiel)  
 PCM for Online Capacity Management of C-B Systems  
 Nov. 18, 2011 @ Palladio Days
Extraction of RDSEFFs (Example)
SLAstic2PCM Transformation (cont’d)
Utilizing PCM for SLAstic ▶ SLAstic2PCM M2M Transformation

André van Hoorn (CAU Kiel)

PCM for Online Capacity Management of C-B Systems
Nov. 18, 2011 @ Palladio Days
Scalability of the Transformation

Utilizing PCM for SLA\textit{stic} → SLA\textit{stic}2PCM M2M Transformation

- Kieker file system log

<table>
<thead>
<tr>
<th>System</th>
<th>Size of Monitoring Log</th>
<th># Operation Executions</th>
<th># Traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bookstore</td>
<td>892 KB</td>
<td>6,540</td>
<td>1,635</td>
</tr>
<tr>
<td>JPetStore</td>
<td>63 MB</td>
<td>259,852</td>
<td>48,720</td>
</tr>
<tr>
<td>Customer Portal</td>
<td>164 MB</td>
<td>847,758</td>
<td>316,980</td>
</tr>
<tr>
<td>SPECjEnterprise</td>
<td>380 MB</td>
<td>189,4830</td>
<td>75,018</td>
</tr>
</tbody>
</table>

- SLA\textit{stic} model metrics

<table>
<thead>
<tr>
<th>System</th>
<th># Components</th>
<th># Signatures</th>
<th># Containers</th>
<th># CallingRelationships</th>
<th># value pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bookstore</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>JPetStore</td>
<td>9</td>
<td>42</td>
<td>1</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Customer Portal</td>
<td>11</td>
<td>212</td>
<td>4</td>
<td>97</td>
<td>174</td>
</tr>
<tr>
<td>SPECjEnterprise</td>
<td>39</td>
<td>272</td>
<td>1</td>
<td>198</td>
<td>463</td>
</tr>
</tbody>
</table>

- Durations of SLA\textit{stic}2PCM transformations

<table>
<thead>
<tr>
<th>System</th>
<th>Mean duration (seconds)</th>
<th>Std dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bookstore</td>
<td>0.9985</td>
<td>0.049</td>
</tr>
<tr>
<td>JPetStore</td>
<td>1.177</td>
<td>0.043</td>
</tr>
<tr>
<td>Customer Portal</td>
<td>2.2608</td>
<td>0.032</td>
</tr>
<tr>
<td>SPECjEnterprise</td>
<td>4.9192</td>
<td>0.101</td>
</tr>
</tbody>
</table>
Conclusion

Conclusions

Summary

• Outlined SLAStic approach: Online capacity management of CB systems
• Architectural models used @ Runtime (PCM by decoration)
• Extraction of SLAStic (→ SLAStic2PCM → PCM) models via dynamic analysis
• Extraction step evaluated for larger systems already
• Supported languages (thanks to Kieker’s language support):
  • Java,
  • .NET,
  • Visual Basic 6
  • (in progress: COBOL, . . . )

Future Work

• Extend SLAStic2PCM to produce SLAStic/PCM decorator model
• Basic extraction of resource demands for PCM RDSEFFs
• Evaluate quality of extracted models
• Resolve some of SLAStic.SIM’s limitations (e.g., PMF support in loops)


“The fundamental organization of a system embodied in
its components,
their relationships to each other and to the environment,
and the principles guiding its design and evolution.”

Software Architecture (IEEE Def. [IEEE 2000])

- Views & Viewpoints
  [Clements et al. 2002]
  - Structural
  - Behavioral
- ADLs
- Architectural styles
  [Taylor et al. 2009]
  - Component-based
  - Service-oriented, etc.
- Runtime reconfiguration/Change management
  [Kramer and Magee 1990, Hofmeister 1993, Matevska 2009]
Software Performance Engineering (cont’d)

Bonus  ▶  Foundations and Related Work

**Performance**

- Concerned with
  - Timing
  - Resource usage
- Metrics [Jain 1991]
  - Response time,
  - Throughput,
  - Utilization, etc.
- Model- vs. Measurement-based
- Capacity Planning [Menascé and Almeida 2002]
  - SLAs/SLOs
  - Workload Characterization
  - Workload Forecasting
  - Performance Prediction

---

Software Performance Engineering [Woodside et al. 2007]

“Represents the entire collection of software engineering activities and related analyses used throughout the software development cycle which are directed to meeting performance requirements.”
IBM’s autonomic computing initiative [Kephart and Chess 2003, IBM 2010]

“Systems manage themselves according to an administrator’s goals.”
Related Work

**Performance Modeling of Component-Based Software Systems**


**Architecture-Based Runtime Reconfiguration**


**Online Performance & Resource Management**