New Concepts for Online Architecture-Level Performance Models

Fabian Brosig (fabian.brosig@kit.edu)

Open Excellence Workshop, December 4, 2012, Karlsruhe
Motivation:
Online Performance Prediction

• Performance query (Performance question, Constraints)

• Question: "How does Change X affect the performance of Y in terms of metric Z?"

• Constraints: Accuracy, timing constraints, overhead
Motivation: Online Performance Prediction

Performance Query

• Question:
  How does Change X affect the performance of Y in terms of metric Z?
Motivation: Online Performance Prediction

- Performance query (Performance question, Constraints)
- Question: "How does Change X affect the performance of Y in terms of metric Z?"
- Constraints: Accuracy, timing constraints, overhead

Online Performance Prediction
Motivation: Online Performance Prediction

- Performance query (Performance question, Constraints)
- Question: "How does Change X affect the performance of Y in terms of metric Z?"
- Constraints: Accuracy, timing constraints, overhead

Online Performance Prediction

- Responsiveness
- SLA (Service Level Agreement)
- Online prediction
  - SLA violation
  - SLA OK
- Workload change
- Cost/Energy efficiency metric
- Inefficient resource usage
- Optimal resource usage

Fabian Brosig - Online Architecture-Level Performance Models
Motivation:
Online Performance Prediction

Online Performance Prediction
• Problem anticipation
• Predictions for different configurations

Precondition for
• Avoiding Service Level Agreement (SLA) violations
• Efficient resource management
State-of-the-Art: Models for performance prediction at run-time

- Based on, e.g., (layered) queueing networks or queueing petri nets (*predictive performance models*)
- Simple models that abstract the system at a very high level
- Services modeled as black boxes
- Software architecture and configuration not modeled
- Many restrictive assumptions imposed

[G. Pacifici et al], [A. D’Ambrogio et al], [G. Tesauro et al], [D. Menasce et al], [C. Adam et al], [I. Foster er al], [A. Othman et al], [P. Shivam et al], …
State-of-the-Art: Models for design&deployment time

- Software architecture models
  - annotated with descriptions of system’s performance-relevant behavior

- Used at design time
  - Evaluate alternative system designs

- Used at deployment time
  - Predict the system performance for capacity planning purposes

[R. Reussner et al], [M. Woodside et al], [D. Petriu et al], [C. Smith et al], [R. Mirandola et al], [K. Trivedi et al], [V. Cortellessa et al], [I. Gorton et al], [E. Eskenazi et al], …
Approach

Use architecture-level performance models for online performance prediction

- Differences in the type of data available for model parameterization
- Current approaches to model component execution context not suitable for use at run-time
- Overhead in analyzing models
Outline

- Online performance prediction scenario

- Application Architecture Model
  - Abstractions to model service behavior
  - New modeling concepts for expressing parameter and context dependencies
ONLINE PREDICTION SCENARIO
Online Performance Prediction Scenario

Customer A -> Workload A

Customer B -> Workload B

Enterprise Application deployed in a clustered environment

Application Server Cluster

Database Server

Fabian Brosig - Online Architecture-Level Performance Models
SPECjEnterprise2010

Customer Relationship Management (CRM)

Manufacturing

Supply Chain Management (SCM)
Online Performance Prediction Scenario

Customer A → Workload A → Cluster A

Customer B → Workload B → Cluster B

SPECjEnterprise2010 deployed in a clustered environment

Cluster A

Application Server Cluster

Cluster B

Oracle WebLogic Server® 11g

Oracle Database

Dell PowerEdge R904
4 x 6-core AMD CPUs, 128 GB main memory

20 Compute Nodes
each node has
2 x 4-core Intel CPUs, 32GB main memory

GBit LAN
APPLICATION ARCHITECTURE MODEL
Descartes Meta-Model

Adaptation Process

Adaptation Points

Application Architecture

Resource Environment

Degrees-of-Freedom
Further details in:

Resource Landscape: Example
DMM: Application Architecture

Further details in, e.g.:

Application Architecture Model

- Service Behavior Abstractions for Different Levels of Granularity
- Probabilistic Parameter Dependencies
- Deployment-Specific Resource Demands / Response Time
„Recap“ – (Composite) Components and Interfaces

**Diagram:.we酮Wei**

- **CompA**
  - **CompB**
    - **CompC**
    - **CompD**
  - **InterfaceX**
    - **InterfaceY**
    - **InterfaceZ**

**Links:**
- **<<Provides>>** from CompA to CompB, CompB to CompC, and CompB to CompD.
- **<<Requires>>** from CompB to InterfaceY and from InterfaceY to CompC.
- **<<Provides>>** from InterfaceY to CompC and from InterfaceZ to CompD.
Application Architecture Model

- Service Behavior Abstractions for Different Levels of Granularity
- Probabilistic Parameter Dependencies
- Deployment-Specific Resource Demands / Response Time
Service Behavior Abstractions

- What do we want to predict?
- What performance-relevant data is available?

Provided service: purchase(..)

Outsourced Service? 3rd Party Component?

<<Component>>
SupplyChain Management

<<Assembly Context>>
SupplyChain Management
Service Behavior Abstractions

Provided service:
sendPurchaseOrder(..)

Required services:
processPurchaseOrderAttachment(..)
processPurchaseOrderInline(..)

Response Time Statistics

Response Time [ms]

Frequency

0 100 200 300 400 500 600 700

0 20 40 60 80

<<CompositeComponent>>
SupplyChainManagement

<<CompositeComponent>>
Purchase

<<Assemble Component>>
Purchased

Fabian Brosig - Online Architecture-Level Performance Models
Service Behavior Abstractions

- BlackBoxBehavior
  - No information about resources, resource demands, control flow, call frequencies,…

- CoarseGrainedBehavior
  - Information at component boundary level (external services, resource consumption,…)

- FineGrainedBehavior
  - Information about component-internals (control flow, resource demands, parametric dependencies,…)

Fabian Brosig - Online Architecture-Level Performance Models
Service Behavior Abstractions - Metamodel

- **ServiceBehaviorAbstraction**
  - **Signature**: 1

- **BlackBoxBehavior**
  - **CoarseGrainedBehavior**: *
  - **FineGrainedBehavior**: *

- **InterfaceProvidingEntity**
  - **InterfaceRequiringEntity**
  - **InterfaceProvidingRequiringEntity**: 1

- **RepositoryComponent**
  - **System**

- **BasicComponent**
  - **CompositeComponent**
  - **Subsystem**
Service Behavior Abstractions - Metamodel

- **BlackBoxBehavior**
  - ResponseTime: 0..1

- **CoarseGrainedBehavior**
  - ExternalCallFrequency: 1
  - CallFrequency: 1

- **ResourceDemandSpecification**
  - ProcessingResourceType: 1
  - ResourceDemand: 1
Service Behavior Abstractions - Metamodel

Control flow abstraction

- FineGrainedBehavior
  - AcquireAction
  - ReleaseAction
  - PassiveResource
  - StartAction
  - StopAction

- ComponentInternalBehavior
  - AbstractAction
    - pred: 0..1
    - succ: 0..1

- BranchTransition
  - forks: \{ordered\}

- ExternalCallAction
- InternalAction
- LoopAction
  - body
- ForkAction
- BranchAction
- BranchProbabilities
Service Behavior Abstraction - Example

sendPurchaseOrder: FineGrainedBehavior

FineGrainedBehavior

<<BranchAction>>

<<BranchTransition>>
Probability: 0.5

<<ExternalCallAction>>
processPurchaseOrderAttachment

<<BranchTransition>>
Probability: 0.5

<<ExternalCallAction>>
processPurchaseOrderInline
Service Behavior Abstraction - Example

sendPurchaseOrder: CoarseGrainedBehavior

CoarseGrainedBehavior

<<ExternalCallFrequency>>
frequency: IntPMF[(0;0.5)(1;0.5)]
processPurchaseOrderAttachment

<<ExternalCallFrequency>>
frequency: IntPMF[(0;0.5)(1;0.5)]
processPurchaseOrderInline
Service Behavior Abstraction - Example

Predicted Response Time Statistics

FineGrained

Predicted Response Time Statistics

CoarseGrained

Measured Response Time Statistics
Parameter and Context Dependencies

- Characterize dependencies probabilistically

- Influencing parameters
  - Service input parameters

- Influenced variables
  - Loop iteration numbers (FineGrainedBehavior)
  - Branch probabilities (FineGrainedBehavior)
  - Call frequencies (CoarseGrainedBehavior)
  - Resource demands (FineGrainedBehavior, CoarseGrainedBehavior)
  - Response times (BlackBoxBehavior)
Parameter Dependencies

Required/Provided service: scheduleManufacturing(String workOrderId)

Required service: purchase(List demands)
Parameter Dependencies

the higher the number of assemblies to manufacture, the higher the probability of a purchase request.

Required/Provided service:
scheduleManufacturing(String workOrderld)

Provided service:
newOrder(String assemblyld, int quantity)

Required service:
purchase(List demands)
Parameter Dependencies

b) Summarized Statistics

Monitor Throttling:
Trace only a subset of the system requests, e.g., every 1000th request
## Parameter Dependencies

<table>
<thead>
<tr>
<th>Workload:</th>
<th>Measurements:</th>
<th>Predictions:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U_WLS</td>
<td>R_avg [ms]</td>
</tr>
<tr>
<td>scheduleManufacturing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>quantity in [0..10]</td>
<td>0.166</td>
<td>17.2</td>
</tr>
<tr>
<td>quantity in [0..200]</td>
<td>0.627</td>
<td>32.9</td>
</tr>
</tbody>
</table>

With $P(\text{"more parts are needed"}) = 0.10$: $U_{\text{WLS}} = 0.196$, $R_{\text{avg}} = 16.8$ ms
Parameter Dependencies - Approach

SPECjEnterprise2010

Provided service:
newOrder(String assemblyId, int quantity)

Required/Provided service:
purchase(List demands)

<<AssemblyContext>>
Manufacturing

<<AssemblyContext>>
Dealer

Required/Provided service:
scheduleManufacturing(String workOrderId)

<<AssemblyContext>>
Influenced Variable1

<<AssemblyContext>>
SupplyChain Management
Parameter Dependencies - Approach

<<CompositeComponent>>  SPECjEnterprise2010

- Provided service: `newOrder(String assemblyId, int quantity)`
- Required/Provided service: `purchase(List demands)`

<<AssemblyContext>>

- Manufacturing
- Influencing Parameter 1
- Influenced Variable 1

<<AssemblyContext>>

- Dealer

<<AssemblyContext>>

- SupplyChain Management

- Required/Provided service: `scheduleManufacturing(String workOrderId)`
Parameter Dependencies - Approach

Provided service: 
newOrder(String assemblyId, int quantity)

Required/Provided service: 
purchase(List demands)

Path from the influenced variable to the system boundary.
Model Variables - Metamodel

ModelVariable

InfluencingParameter
- name : string
- description : string

InfluencedVariable

ServiceBehaviorAbstraction

ControlFlowVariable
- desc : RandomVariable

ResourceDemand
- desc : RandomVariable

ResponseTime
- desc : RandomVariable

BranchProbabilities

CallFrequency

LoopIterationCount

CallParameter
Call Parameter Hierarchy - Metamodell

CallParameter

ServiceInputParameter
  *
  1
  ProvidedRole

ExternalCallParameter
  *
  1
  ExternalCall

ExternalCallReturnParameter
  *
  1
  ExternalCall
Validity of Measurements – „Scope“

Customer-specific application server cluster

Scenario: Replicate Server Instance

Diagram:

<<Subsystem>> CustomerA

- <AssemblyContext>
  - SPECjEnterprise2010

- <AssemblyContext>
  - SPECjEnterprise2010

- ...
Validity of Measurements – „Scope“

- Customer-specific application server cluster
- Scenario: Replicate Server Server Instance

Scenario:
- Replicate Server Instance

CustomerA
- SPECjEnterprise2010

CustomerB
- SPECjEnterprise2010

≠
Modeling Scopes - Example

System A

\[ \text{scopeC(inst}_{C(p)}_{1}) = \text{scopeC(inst}_{C(p)}_{2}) \]

\[ \text{scopeC(inst}_{C(p)}_{3}) = \text{scopeC(inst}_{C(p)}_{4}) \]
Modeling Scopes - Example

<<System>>
System A

<<Composite Component>>
X

<<Composite Component>>
Y

<<Composite Component>>
Z

Scope $S(p)$

Influencing parameter $p$

contains
encapsulates
contains
encapsulates
contains
encapsulates

<<Assembly Context>>
X_AC1

<<Assembly Context>>
X_AC2

<<Assembly Context>>
Y_AC1

<<Assembly Context>>
Y_AC2

<<Assembly Context>>
Z_AC1
Deployment-Specific Resource Demands

Component Repository

ResourceDemand: 15 ms CPU

System

ResourceDemand: 12 ms CPU

Deployment

Deployment:

Resource Landscape

Fabian Brosig - Online Architecture-Level Performance Models

Descartes Research Group
http://www.descartes-research.net
Summary

- Use architecture-level performance models for online performance prediction

- Application Architecture Model
  - Different abstractions to model service behavior
  - New modeling concepts for expressing parameter and context dependencies

- Descartes Meta-Model (DMM)