An Adaptation Framework Enabling Resource-Efficient Operation of Software Systems

André van Hoorn¹, M. Rohr¹, A. Gul¹, and W. Hasselbring²

¹ Graduate School TrustSoft®, CS Dept., Univ. Oldenburg (GER)
² Software Engineering Group, CS Dept., Univ. Kiel (GER)

VJP 2009
April 2, 2009 (Cape Town)

This work is supported by the German Research Foundation (DFG), grant GRK 1076/1

Motivation & Goal

SLAs in Application Service Provision

Service Level Agreements (SLAs)

Contractual specification between the provider and the client of a service regarding the Quality of Service (QoS) that must be satisfied by the service provider under well-defined conditions.

Capacity Management

System Capacity

Maximum workload level not violating the [performance] SLOs

Capacity management activities required by ASP:
- Based on the anticipated workload conditions,
  - Provision of appropriate computing (and storage) infrastructure &
  - Deployment of software components to this infrastructure.

Capacity management strategy over the last years:

- S/W components deployed to fixed infrastructure which satisfies the needs for the anticipated worst-case workload conditions.
- Future demands satisfied in the spirit of "kill-it-with-iron" (KIWI)

Goal & Approach

Goal

Reducing operating costs of S/W systems while satisfying the SLAs

Approach

Architecture-based runtime adaptation for resource-efficient operation
SLAStic (s’laStic) Adaptation Framework

- **Middleware**
  - Provides instrumentation infrastructure (e.g., [RvHM’08]) and
  - Executes adaptation operations (e.g., [Bun08])
- **Controller**
  - Executes the self-adaptation cycle
  - Maintains the runtime models
  - Triggers adaptation operations

**Self-adaptation Cycle**

1. **Observation**
   - Extract and pre-process measurement data of elapsed period
   (S/W system is continuously being monitored)
2. **Analysis**
   - Performance Analysis
   - Workload Analysis
   - Prediction
   - Adaptation Analysis
3. **Adaptation**
   - Trigger middleware to execute selected adaptation operations
   - Reflect changes in runtime models (after adaptation committed)

**Architectural Modeling**

- Approach requires explicit modeling of relevant aspects of the software system architecture at design time
- Architecture-level aspects to be modeled
  - Components (interfaces, behavior) and assembly
  - Deployment environment (available nodes and resources)
  - Component deployment (assignment of components to deployment environments)
  - (Performance) SLAs/SLOs
  - Adaptation constraints & policies
  - Adaptation costs (monetary and/or time)
  - During runtime, (parts of) these models are refined, kept synchronized and used for the analyses

**Summary & Related Work**

**Summary:**
- **Problem:** Overprovisioning capacity management is cost/resource-inefficient
- **Goal:** QoS-aware reduction of operating costs (e.g., power consumption)
- **Approach:** Self-adaptive, architecture-based runtime capacity management

**Related Work:**
- Capacity Planning (e.g., [MA00, MA02, MAD04])
- Software performance prediction (e.g., [SG08, SW02, BK09])
- Autonomic QoS management (e.g., [MBR05, NKJT08])
- Self-* software architectures (e.g., [OMT98, KM07, OMT08])

**Current & Future Work**

1. **Confirmation of assumptions**
   - Varying workload and resource utilization
   - Analysis of potential cost savings
2. **Specification of adaptation operations**
3. **Specification of modeling formalisms and runtime models**
4. **Adaptation framework (instantiation)**
   - **Focus:** Development of adaptation analysis activity
   - Runtime performance prediction using performance models
   - Selection of adaptation operations (adaptation plans)
   - Updates to runtime models
   - Proof-of-concept implementation
5. **Evaluation (simulation + lab study + field study)**
   - Does the approach improve resource efficiency?
   - Is it applicable to realistic scenarios?
References I

Martin F. Arbib, Horia Vosniakos, and Jerry Rola.
Characterizing the variability of a single web-based shopping system.

Stef die Benson, Volker Siegel, and Rolf Resseguie.
The Palermo Container Model for model-driven performance prediction.

Swen Giesecke, Matthias Klahmet, Volker Siegel, and Rolf Resseguie.
International on Software Performance Measurement and Analysis.

Nikos E. Doudis and Karl Fata.
Self-monitoring in World Wide Web traffic: evidence and possible causes.

Jeff Kramer and Jeffrajan.
Self-adaptive systems: an architectural challenge.

Daniel A. Menascé, Jorgo V. A. Petrović, Jörgen Torngren, and Mathias Klahmet.
Capacity Planning for Web Services: Metrics, Models, and Methods.

References II

Daniel A. Menascé, Viggo A. F. Arnaud, and Lawrence W. Dowdy.
Performance by Design: Computer Capacity Planning By Example.

Daniel A. Menascé, Mohamed S. Benatallah, and Hongxi Fan.
On the rise of next generation performance models, in self-managing and self-organizing computing systems.

References III

Bridger Sittler and David Sarnoff.
Architecture-driven performance analysis.