Model-based Run-Time Adaptation in Dynamic Virtualized Resource Landscapes

Open Excellence Workshop, Karlsruhe, December 4th, 2012

Nikolaus Huber (nikolaus.huber@kit.edu)
Overview

**WHY**
do we need models?

**HOW**
should the models look like?

**WHAT**
do we do with the models?
Motivation

Data Center 1

Data Center 2

Data Center n

Internet
Motivation

Internet

Data Center n

Data Center 1

Data Center 2

@ 🛒 📩

⚠️ SLA Violation

⚠️ Resource Inefficiency

Nikolaus Huber – Model-based Run-time Adaptation of Dynamic Virtualized Resource Landscapes
Motivation

- Nikolaus Huber – Model-based Run-time Adaptation of Dynamic Virtualized Resource Landscapes

- Data Center 1
  - Virtual Machines
  - Server 1 .. Server n
  - SLA Violation

- Data Center 2
  - Resource Inefficiency
Motivation

- Data Center 1
  - Virtual Machines
  - Server 1 .. Server n
  - SLA Violation

- Data Center 2
  - Resource Inefficiency

- Data Center n
  - Server Rack

- Internet

Nikolaus Huber – Model-based Run-time Adaptation of Dynamic Virtualized Resource Landscapes
Motivation

Internet

Data Center 1
- Server 1 .. Server n
- Virtual Machines
- SLA Violation

Data Center 2
- Resource Inefficiency

Data Center n
- Server Rack

Nikolaus Huber – Model-based Run-time Adaptation of Dynamic Virtualized Resource Landscapes
Motivation

Internet

Data Center 1

Virtual Machines

Server 1 .. Server n

⚠️ SLA Violation

Data Center 2

+1

Resource Inefficiency

Data Center n

Server Rack

Nikolaus Huber – Model-based Run-time Adaptation of Dynamic Virtualized Resource Landscapes
Motivation

Internet

Data Center n
Server Rack

Data Center 1
Virtual Machines
Server 1 .. Server n
⚠️ SLA Violation

Data Center 2
+1
Resource Inefficiency

Motivation

Nikolaus Huber – Model-based Run-time Adaptation of Dynamic Virtualized Resource Landscapes
Meta-Model that

- Captures structural information
- Reflects dynamic aspects
- Describes adaptation process
DMM: Resource Landscape

Adaptation Process

Adaptation Points

Application Architecture

Resource Landscape

Further details in:

Resource Landscape: Example
Resource Layers & Containers

- Common concepts in data centers
  - Containers/Layering (physical and logical)
  - Abstraction of resources
  - Resource sharing

- Advantage:
  - Flexibility
  - Structural information
Resource Layers & Containers

DistributedDataCenter
  \[\text{belongsTo} \rightarrow 0..1\]
  \[\text{consistsOf} \rightarrow 1..*\]

DataCenter
  \[\text{partOf} \rightarrow 1\]

AbstractHardwareInfrastructure
  \[\text{contains} \rightarrow 1..*\]

ConfigurationSpecification
  \[\text{configSpec} \rightarrow *\]

ContainerTemplate
  \[\text{template} \rightarrow 0..1\]

Container
  \[\text{contains} \rightarrow 0..1\]
  \[\text{partOf} \rightarrow 1\]

CompositeHardwareInfrastructure
  \[\text{contains} \rightarrow 1..*\]

StorageInfrastructure

NetworkInfrastructure

ComputingInfrastructure

RuntimeEnvironment
  \[\text{ofClass} : \text{RuntimeEnvironmentClasses}\]

ofContainer

Nikolaus Huber – Model-based Run-time Adaptation of Dynamic Virtualized Resource Landscapes
Resource Landscape: Container Templates and Runtime Environment Classes

ContainerRepository
  1. templates
     0..*

ContainerTemplate
  0..1. templateConfig
     0..1

ConfigurationSpecification

«enumeration»
RuntimeEnvironmentClasses

- HYPervisor
- OS
- OS_VM
- PROCESS_VM
- MIDDLEWARE
- OTHER
Resource Landscape: Configuration Specification

- PassiveResourceType
  - capacity : EBigInteger

- ConfigurationSpecification
  - passiveResourceSpecification
  - activeResourceSpec
  - parentResourceSpecification
  - processingResources
  - schedulingPolicy : SchedulingPolicy
  - processingRate : EDouble

- ActiveResourceSpecification
  - parentResourceSpecification

- CustomConfigurationSpecification
  - parentResourceSpecification
  - linkingResources

- PassiveResourceCapacity
  - capacity : EBigInteger

- SchedulingPolicy
  - DELAY
  - FCFS
  - PROCESSOR_SHARING
  - RANDOM
  - N/A

- ProcessingResourceSpecification
  - bandwidth : EDouble
  - communicationLinkResourceType
  - nrOfParProcUnits

- LinkingResourceSpecification
  - parentResourceSpecification

- EObject
  - 0..1 non-functionalProperties
Modeling Virtualization Platforms

Further details in:

Automated Experimental Analysis

Further details in:

Experiment Setup

- Virtualization Platforms
  - Citrix XenServer 5.5
  - VMware ESX 4.0

- Experimental environment
  - SunFire X4440 Server, AMD Opteron 24*2.4 GHz, 128 GB DDR2 RAM

- Different benchmark types
  - Passmark PerformanceTest v7 (CPU, Memory, HDD)
  - SPECcpu (CPU + Memory)
  - Iperf (Network)
Virtualization Overhead

XenServer 5.5
Throughput metric: higher values are better

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>native</th>
<th>virtualized</th>
<th>Delta (abs.)</th>
<th>Delta (rel.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passmark CPU, 1 core</td>
<td>639.3</td>
<td>634.0</td>
<td>5.3</td>
<td>0.83%</td>
</tr>
<tr>
<td>Passmark CPU, 2 cores</td>
<td>1232.0</td>
<td>1223.0</td>
<td>9.0</td>
<td>0.97%</td>
</tr>
<tr>
<td>SPECint(R)_base2006</td>
<td></td>
<td></td>
<td></td>
<td>3.61%</td>
</tr>
<tr>
<td>SPECfb(R)_base2006</td>
<td></td>
<td></td>
<td></td>
<td>3.15%</td>
</tr>
<tr>
<td>Passmark Memory, 1 core</td>
<td>492.9</td>
<td>297.0</td>
<td>195.9</td>
<td>39.74%</td>
</tr>
<tr>
<td>Passmark Memory, 2 cores</td>
<td>501.7</td>
<td>317.5</td>
<td>184.2</td>
<td>36.72%</td>
</tr>
<tr>
<td>Iperf (send)</td>
<td>527.0</td>
<td>393.0</td>
<td>134.0</td>
<td>25.43%</td>
</tr>
<tr>
<td>Iperf (receive)</td>
<td>528.0</td>
<td>370.0</td>
<td>158.0</td>
<td>29.92%</td>
</tr>
</tbody>
</table>
Scalability

- Scaling CPU resource
- Performance impact of affinity

Affinity OFF
Scalability

- Scaling CPU resource
- Performance impact of affinity

Affinity ON
Modeling Virtualization Platforms - II

Virtualization Platform

Virtualization Type
- Full Virtualization
- Para-Virtualization

VMM Architecture
- Monolithic
- Dom0

Resource Management Configuration
- CPU Scheduling
- CPU Allocation
- Core Affinity
- CPU Priority
- Resource Overcommitment
- Memory Allocation

Number of VMs

Workload Profile
- CPU
- I/O
- Memory
- Disk
- Network

Legend
- exclusive OR
- inclusive OR

Overhead reduction:
- ~ 5% (CPU)

Overhead reduction:
- ~ 12% (CPU)
- ~ 40% (MEM)
Resource Landscape - Summary

- Specification of containers
- Specification of resources
- Performance influence(s) of layers
  - Virtualization
  - Middleware
  - ...

Adaptation Points
Application Architecture
Resource Landscape
Adaptation Process
Further details in:

Modeling the Application Level

- Service Behavior Abstractions for Different Levels of Granularity

- Parameter and Context Dependencies

---

**Required service:**
- purchase(List demands)

**Provided service:**
- newOrder(String assemblyId, int ... service:
- scheduleManufacturing(String workOrderId)

---

**Diagram:**
- **AsynchronousForkAction>>**
- **AquireAction<<**
- **ResourceDemandingSEFF>>**
- **ResourceDemandingBehavior>>**
- **ResourceDemand>>**
- **ReleaseAction>>**

---

**System Components:**
- **Dealer**
- **Manufacturing**

- **Required/Provided service:**
  - scheduleManufacturing(String workOrderId)
  - newOrder(String assemblyId, int quantity)

- **Required service:**
  - purchase(List demands)
Deployment

Component Repository

System

Deployment

Resource Landscape

Diagram:

- Component Repository
- System
- Deployment
- Resource Landscape

- <<Assembly Context>>
- <<Resource Demand>>
- <<Deployment Context>>
- Application Server 1
- Application Server 2

Diagram labels:

- CompB
- CompB Allocation
- <<Basic Component>>
- <<Resource Demanding Element>>

Diagram relationships:

- Component Repository to System
- System to Deployment
- Deployment to Resource Landscape

Diagram notes:

- Model-based Run-time Adaptation of Dynamic Virtualized Resource Landscapes

Nikolaus Huber
DMM: Adaptation Points

Further details in:

Adaptation Points: Examples

■ Scaling Resources

■ Replicating VMs, Migrating VMs
Adaptation Points

- Specification of valid system configurations
- “Decorator” model of static view

AdaptationPointDescriptions

AdaptationPoint

AdaptableEntity

VariationType

ModelEntityConfigurationRange

ModelVariableConfigurationRange

PropertyRange

SetOfConfigurations

EObject

AdaptationPointDescriptions

0..* AdaptationPoint

AdaptableEntity

variationPossibility

adaptationPoints

1

ModelEntityConfigurationRange

minValue : EDouble
maxValue : EDouble

ModelVariableConfigurationRange

minValueConstraint : OclConstraint
maxValueConstraint : OclConstraint

possibleValues : OclConstraint

SetOfConfigurations

variants

0..*

eObject

1

entity

1

adaptableEntity

1

adaptationPoints
DMM: Adaptation Process

Details in:

Motivation

- Rapid growth of autonomic computing and self-adaptive systems engineering

- Open challenges
  - Hard-coded or system-specific reconfiguration techniques
  - How to separate software design and implementation from system reconfiguration logic?

- Main issues
  - How to abstract from system-specific details?
  - How to enable the reuse of adaptation strategies?

- Vision: Holistic model-based Approach
Model-based System Adaptation

Managed System

Adaptation Language

- Strategies
- Tactics
- Actions

Logical View

Technical View

Adaptation Points Model

System Architecture QoS Model

Managed System

1 GBit

4 GBit Switch

Gbit Switch

Database Server

Application Servers

...
Adaptation Language - Idea

- Describe system adaptation processes at the system architecture level
- Distinguish high-level reconfiguration objectives from low-level implementation details
- Explicitly separate technical from logical aspects
- Capture reconfiguration logic in a generic, human-understandable, machine processable and reusable way
- Provide intuitive modeling concepts that can be employed by system architects and software developers
- Facilitate maintenance and reuse
S/T/A Adaptation Language

Separate
- Logical view, high-level process
- Technical view, low-level operations
Separation of Concerns

Strategies
- High-level
- Independent of system specific details
- Describe process view
- Indeterminism

Tactics & Actions
- Low-level
- System specific
- Reconfiguration operations
- Deterministic
S/T/A Meta-Model

- Actions refer to adaptation points / DoF Model
- Tactics execute actions in adaptation plans
- Strategies use weighted tactics
Example Strategies

<<OverallGoal>>
Maintain SLAs using as little resources as possible

<<Objective>>
MaintainSLAs

<<Strategy>>
PUSH (Increase Resources)
<<WeightedTactic>>
addResources

<<Event>>
SlaViolated

<<Objective>>
OptimizeResourceEfficiency

<<Strategy>>
PULL (Decrease Resources)
<<WeightedTactic>>
removeResources

<<Event>>
ScheduledOptimization
<<WeightedTactic>>
undoPreviousAction
Example Tactics

<<Tactic>>
addResources

<<Loop>>
counter=#iterations

<<ActionReference>>
addVCPU

<<ActionReference>>
addAppServer

<<Tactic>>
removeResources

<<ActionReference>>
removeAppServer

<<ActionReference>>
removeVCPU

<<Tactic>>
undoPreviousAction

<<ActionReference>>
addAppServer

<<ActionReference>>
addVCPU

<<ActionReference>>
PrevActionAddServer

TRUE
FALSE
Evaluation

- S/T/A implemented in PerOpteryx
Self-Adaptive Resource Management

Online reconfiguration impact prediction for trade-off analysis

Service A
VM replication/cloning
Scaling up/Improving dependability
Online prediction

Service A
VM replication/cloning

Service B
LiveVM migration
Dynamic server consolidation
Online prediction

Service C
LiveVM migration
Dynamic server consolidation
Online prediction

Reconfiguration results

SLA OK

Further details in:

Self-Adaptive Resource Allocation

- Customer Specification(s)
- Workload Forecasting

Collect

- Monitor System and Workload

Act

- Reconfigure System
- Problem resolved

Analyze

- Anticipate/Detect Problem

Decide

- Predict Reconfiguration Effect(s)
- Generate Reconfiguration Scenario

Problem persists
Reconfiguration Algorithm

PUSH Phase
- Add resources
  - vCPUs (if available)
  - Application server nodes
- until

\[
\text{cap}(c, t) = \left[ \sum_{c^* \in C} c^*[\lambda] \cdot D(c^*[s]) \right] \cdot \text{cap}(c, t)
\]

PULL Phase
- Remove underutilized resources as long as no SLAs are violated
Experimental Setup

- Six blade servers
  - 2 Xeon E5430 4-core CPUs
  - 32 GB of main memory
- Citrix XenServer 5.5
- Oracle WebLogic 10.3.3
- Oracle Database 11g
What If: New Service Added?

![Graph showing mean response time for different services under different configurations]

- **EJB**
- **WS**
- **Purchase**
- **Manage**
- **Browse**

Legend:
- default configuration ($c_0$)
- new service ($c_0$)
- after reallocation ($c_1$)
What If: Workload Changes?

a) increasing workload from 2x to 4x

b) increasing workload from 4x to 6x
Benefits in Cost Savings

Assigned Capacity (VCPUs)

Workload

Assigned Servers

capacity
servers

static assignment

day1
1x
day2
4x
day3
6x
day4
4x
day5
6x
day6
2x
day7
1x
Self-Adaptive Resource Allocation

Customer Specification(s)

Workload Forecasting

Collect

Monitor System and Workload

Act

Reconfigure System

Problem resolved

Analyze

Anticipate/Detect Problem

Problem persists

Decide

Predict Reconfiguration Effect(s)

Generate Reconfiguration Scenario
Nikolas Herbst (Diploma thesis)

WORKLOAD CLASSIFICATION AND FORECASTING
(I) Select strategies

- **INITIAL**
  - Naïve, Smoothing, Random Walk

- **FAST**
  - Trend Interpolation

- **COMPLEX**
  - Decomposition & Seasonal Patterns

(II) Evaluate accuracy

Forecast Strategy

- Overhead Groups

- Forecast Objectives
  - Forecast Frequency
  - Overhead Limitation

- Confidence Level

- Forecast Horizon

- Workload Intensity Trace

- Accuracy Feedback
Experiment: Example for Forecast Accuracy Improvement

- Real-world workload intensity trace (IBM CICS transactions on System z)
- Comparison of **Workload Classification & Forecasting (WCF)** approach to **Extended Exponential Smoothing (ETS)** and **Naïve** forecast
Experiment

Cumulative Percentage Error Distribution
Comparison of WCF to Naive and ETS strategy
CICS transactions (5 days, 48 frequency, 240 forecast values)
Case Study: Example for Using Forecast Results

- **Scenario:** Additional server instances at certain thresholds, 3 weeks
- Real-world workload intensity trace (Wikipedia DE page requests per hour)

![Graph showing workload intensity and forecast results](image-url)

- **Classification:**
  - Upper confidence
  - Lower confidence
  - Forecasted value
  - Observation
  - Upper threshold
  - Lower threshold
  - Classification by WCF
  - # forecast
Case Study

Resource provisioning:
(I) Without forecasting (solely reactive):
    Resource provisioning actions triggered by
    76 SLA violations
(II) Interpreting WCF forecast results (add. proactive):
    Reduction to 34 or less SLA violations

→ No significant change in resource usage observed
   (server instances per hour)
Summary

- Meta-Model for
  - Resource Landscape
  - Performance Influences of Virtualization Layers
  - Adaptation Points
  - Adaptation Process
- Example: Model-based resource allocation
- Workload Forecasting
Thank you!
Any Questions?

http://www.descartes-research.net/