Self-Management for Large Scale Distributed Systems

Ahmad Al-Shishtawy (ahmadas@kth.se)

Advisors:
Vladimir Vlassov (vladv@kth.se)
Seif Haridi (seif@sics.se)

KTH Royal Institute of Technology
Stockholm, Sweden

The 5th EuroSys Doctoral Workshop (EuroDW 2011)
April 10, 2011
Outline

1. Introduction
2. Niche Platform
3. Robust Management Elements
4. Future Work
Introduction

Niche Platform

Robust Management Elements

Future Work
Dealing with Complexity

Problem

All computing systems need to be managed
Dealing with Complexity

Problem

All computing systems need to be managed
Problem

Computing systems are getting more and more complex.
Dealing with Complexity

Problem

Complexity means higher administration overheads
Problem

Complexity poses a barrier on further development
Dealing with Complexity

Solution

The Autonomic Computing initiative by IBM
Solution

Self-Management: Systems capable of managing themselves
Dealing with Complexity

Solution

Use **Autonomic Managers**

Autonomic Manager

Monitor

Analyze

Plan

Execute

Autonomic Manager

Monitor

Analyze

Plan

Execute

Autonomic Manager

Monitor

Analyze

Plan

Execute

Autonomic Manager

Monitor

Analyze

Plan

Execute

Autonomic Manager

Monitor

Analyze

Plan

Execute

Autonomic Manager

Monitor

Analyze

Plan

Execute

Autonomic Manager

Monitor

Analyze

Plan

Execute

Autonomic Manager

Monitor

Analyze

Plan

Execute

Autonomic Manager

Monitor

Analyze

Plan

Execute

Autonomic Manager

Monitor

Analyze

Plan

Execute
Dealing with Complexity

Open Question
How to achieve Self-Management?
The Autonomic Computing Architecture

- Managed Resource
- Touchpoint (Sensors & Actuators)
- Autonomic Manager
  - Monitor
  - Analyze
  - Plan
  - Execute
- Knowledge Source
- Communication
- Manager Interface
The Goal

Large-scale distributed systems
- Complex and require self-management
- May run on unreliable resources
- Major sources of complexity:
  - Scale (resources, events, users, ...) 
  - Dynamism (resource churn, load changes, ...)

Goal
- A platform (concepts, abstractions, algorithms...) that facilitates development of self-managing applications in large-scale and/or dynamic distributed environment.
- A methodology that help us to achieve self-management.
The Goal

Large-scale distributed systems
- Complex and require self-management
- May run on unreliable resources
- Major sources of complexity:
  - Scale (resources, events, users, ...)
  - Dynamism (resource churn, load changes, ...)

Goal
- A platform (concepts, abstractions, algorithms...) that facilitates development of self-managing applications in large-scale and/or dynamic distributed environment.
- A methodology that help us to achieve self-management.
Research Plan

Self-Management in large-scale distributed systems. Consists of four main parts:

- Part 1: Touchpoints and feedback loops in distributed systems
- Part 2: Robust Management
- Part 3: Improve management logic
- Part 4: Integrate previous parts in a self-managing system.
**Niche**

- **Niche** is a Distributed Component Management System
- **Niche implements** the Autonomic Computing Architecture for large-scale distributed environment
- **Niche leverages** Structured Overlay Networks for communication and for provisioning of basic services (DHT, Publish/Subscribe, Groups, etc.)
Management Part

- Management Elements
  - Watchers
  - Aggregators
  - Managers
  - Executors
- Communicate through events
- Publish/Subscribe
- Autonomic Managers (control loops) built as network of MEs
- Sensors and Actuators for components and groups
- Actuation API
Management Part

- Management Elements
  - Watchers
  - Aggregators
  - Managers
  - Executors
- Communicate through events
- Publish/Subscribe
- Autonomic Managers (control loops) built as network of MEs
- Sensors and Actuators for components and groups
- Actuation API
Management Part

- Management Elements
  - Watchers
  - Aggregators
  - Managers
  - Executors

- Communicate through events
- Publish/Subscribe
- Autonomic Managers (control loops) built as network of MEs
- Sensors and Actuators for components and groups
- Actuation API
Management Part

- **Management Elements**
  - Watchers
  - Aggregators
  - Managers
  - Executors

- Communicate through events
- Publish/Subscribe
- Autonomic Managers (control loops) built as network of MEs
- Sensors and Actuators for components and groups
- Actuation API
Management Part

- Management Elements
  - Watchers
  - Aggregators
  - Managers
  - Executors
- Communicate through events
- Publish/Subscribe
- Autonomic Managers (control loops) built as network of MEs
- Sensors and Actuators for components and groups
- Actuation API
Management Part

- Management Elements
  - Watchers
  - Aggregators
  - Managers
  - Executors

- Communicate through events
- Publish/Subscribe
- Autonomic Managers (control loops) built as network of MEs
- Sensors and Actuators for components and groups
- Actuation API
Runtime Environment

- **Containers** that host components and MEs
- Use a Structured Overlay Network for communication
- Provide overlay services
### Runtime Environment

- Containers that **host** components and MEs
- Use a Structured Overlay Network for communication
- Provide overlay services
Containers that host components and MEs

Use a Structured Overlay Network for communication

Provide overlay services
Dealing with Resource Churn

How to deal with failures?
- MEs heal the functional part
- How to heal failed MEs?
  - Programmatically in the management logic
  - Transparently by the platform
Dealing with Resource Churn

How to deal with failures?
- MEs heal the functional part
- How to heal failed MEs?
  - Programmatically in the management logic
  - Transparently by the platform
Dealing with Resource Churn

How to deal with failures?

- MEs heal the functional part

How to heal failed MEs?

- Programmatically in the management logic
- Transparently by the platform
Robust Management Elements

A Robust Management Element (RME):

- is replicated to ensure fault-tolerance
- tolerates continuous churn by automatically restoring failed replicas on other nodes
- maintains its state consistent among replicas
- provides its service with minimal disruption in spite of resource churn (high availability)
- is location transparent, i.e., RME clients communicate with it regardless of current location of its replicas
Robust Management Elements

A Robust Management Element (RME):

- is **replicated** to ensure fault-tolerance
- tolerates **continuous churn** by automatically restoring failed replicas on other nodes
- maintains its **state consistent** among replicas
- provides its service with **minimal disruption** in spite of resource churn (high availability)
- is **location transparent**, i.e., RME clients communicate with it regardless of current location of its replicas
A Robust Management Element (RME):

- is replicated to ensure fault-tolerance
- tolerates continuous churn by automatically restoring failed replicas on other nodes
- maintains its state consistent among replicas
- provides its service with minimal disruption in spite of resource churn (high availability)
- is location transparent, i.e., RME clients communicate with it regardless of current location of its replicas
Robust Management Elements

A Robust Management Element (RME):

- is replicated to ensure fault-tolerance
- tolerates continuous churn by automatically restoring failed replicas on other nodes
- maintains its state consistent among replicas
- provides its service with minimal disruption in spite of resource churn (high availability)
- is location transparent, i.e., RME clients communicate with it regardless of current location of its replicas
Robust Management Elements

A Robust Management Element (RME):

- is replicated to ensure fault-tolerance
- tolerates continuous churn by automatically restoring failed replicas on other nodes
- maintains its state consistent among replicas
- provides its service with minimal disruption in spite of resource churn (high availability)
- is location transparent, i.e., RME clients communicate with it regardless of current location of its replicas
Robust Management Elements

A Robust Management Element (RME):

- is **replicated** to ensure fault-tolerance
- tolerates **continuous churn** by automatically restoring failed replicas on other nodes
- maintains its **state consistent** among replicas
- provides its service with **minimal disruption** in spite of resource churn (high availability)
- is **location transparent**, i.e., RME clients communicate with it regardless of current location of its replicas
Solution Outline

- **Replicated state machine**
  - An algorithm to **reconfigure** the replicated state machine. (We used the SMART algorithm)
  - Our decentralized algorithm to **automate** reconfiguration
Solution Outline

- Replicated state machine
- An algorithm to reconfigure the replicated state machine. (We used the SMART algorithm)
- Our decentralized algorithm to automate reconfiguration
Solution Outline

- Replicated state machine
- An algorithm to reconfigure the replicated state machine. (We used the SMART algorithm)
- Our decentralized algorithm to automate reconfiguration
SMART

The SMART Reconfiguration Algorithm

Automatic Reconfiguration

Evaluation
SMART

A B C D E

Admin

Solution Outline
The SMART Reconfiguration Algorithm
Automatic Reconfiguration
Evaluation
SMART

Configuration Repository

Admin

A B C D E

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
Configuration Repository
{A,B,C,D,E}

Admin

Configuration {A,B,C,D,E}
SMART

Configuration Repository
\{A,B,C,D,E\}

Configuration \{A,B,C,D,E\}

Admin
SMART

Configuration Repository
\{A,B,C,D,E\}

Admin

Configuration \{A,B,C,D,E\}

A B C D E

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
SMART

Configuration Repository
{A,B,C,D,E}

Admin

Configuration {A,B,C,D,E}

A B C D E X Y

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
SMART Reconfiguration Algorithm

Automatic Reconfiguration

Evaluation

Configuration Repository
{A,B,C,D,E}

Admin

Configuration {A,B,C,D,E}

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
SMART

Configuration Repository
{A,B,C,D,E}

Admin

Configuration {A,B,C,D,E}

{A,X,C,D,E}
**SMART**

Configuration Repository: \{A, X, C, D, E\}

Admin:

Configuration: \{A, X, C, D, E\}

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
Creating a Replicated State Machine (RSM)

Any node can create a RSM. Select ID and replication degree
Creating a Replicated State Machine (RSM)

Any node can create a RSM. Select ID and replication degree

RSM ID = 10, f=4, N=32
Creating a Replicated State Machine (RSM)

The node uses **symmetric replication scheme** to calculate replica IDs.
Creating a Replicated State Machine (RSM)

The node uses **lookups** to find responsible nodes . . .
Creating a Replicated State Machine (RSM)

...and gets direct references to them

RSM ID = 10, f=4, N=32
Replica IDs = 10, 18, 26, 2
Responsible Node IDs = 14, 20, 29, 7
Configuration = D, F, I, B
Creating a Replicated State Machine (RSM)

The set of direct references forms the **configuration**
Creating a Replicated State Machine (RSM)

The node sends a *Create* message to the configuration.

RSM ID = 10, f=4, N=32
Replica IDs = 10, 18, 26, 2
Responsible Node IDs = 14, 20, 29, 7
Configuration = D, F, I, B
Creating a Replicated State Machine (RSM)

Now replicas communicate directly using the configuration
SMART with Multiple Admins

Configuration Repository
{A,B,C,D,E}

Admin

Configuration {A,B,C,D,E}

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
SMART with Multiple Admins

Configuration Repository
{A,B,C,D,E}

A B C D E
Admin
Admin2

Configuration {A,B,C,D,E}

A B C D E X Y

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
SMART with Multiple Admins

Configuration Repository
{A,B,C,D,E}

Admin

Admin2

Configuration {A,B,C,D,E}

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
SMART with Multiple Admins

Configuration Repository
{A,B,C,D,E}

{A,X,C,D,E}

Admin

Admin2

Configuration {A,B,C,D,E}

A B C D E X Y

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
SMART with Multiple Admins

Configuration Repository
{A,B,C,D,E}

{A,X,C,D,E}

Admin

Admin2

Configuration {A,B,C,D,E}

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
SMART with Multiple Admins

Configuration Repository
{A,B,C,D,E}

Admin
{A,X,C,D,E}

Admin2
{A,B,C,Y,E}

Configuration {A,B,C,D,E}
SMART with Multiple Admins

Configuration Repository
{?,?,?,?,?}

Configuration {?,?,?,?,?}

Configuration {?,?,?,?,?}

Admin

{A,X,C,D,E}

Admin2

{A,B,C,Y,E}

A B C D E X Y

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
SMART with Multiple Admins

Configuration Repository
\{A,B,C,Y,E\}

{A,X,C,D,E}  \[\rightarrow\]  Admin

{A,B,C,Y,E}  \[\rightarrow\]  Admin2

A B C D E X Y

Configuration \{A,B,C,Y,E\}

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
SMART with Multiple Admins

Configuration Repository
{A,B,C,D,E}

Admin

Admin2

Configuration {A,B,C,D,E}

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
SMART with Multiple Admins

Configuration Repository
\{A, B, C, D, E\}

Configuration \{A, B, C, D, E\}
SMART with Multiple Admins

Configuration Repository
{A,B,C,D,E}

{X, , , }
Admin

Admin2

Configuration {A,B,C,D,E}
Proposed Changes { ,X, , , }
{ ,X, , , }

A B C D E X Y

Admins

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
SMART with Multiple Admins

Configuration Repository
{A,B,C,D,E}

Admin
{X, , ,}

Admin2
{ , , Y,}

Configuration {A,B,C,D,E}
Proposed Changes { ,X, , , }
SMART with Multiple Admins

Configuration Repository
{A,B,C,D,E}

Admin
{X,Y}

Admin2
{X,Y}

Configuration {A,B,C,D,E}

Proposed Changes {X,Y}

A B C D E X Y
SMART with Multiple Admins

Configuration Repository
{A, B, C, D, E}

Admin

Admin2

Configuration {A, B, C, D, E}
Proposed Changes {X, Y, }

New Configuration {A, X, C, Y, E}

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy) 18/25
SMART with Multiple Admins

Configuration Repository
{A,X,C,Y,E}

Admin

Admin2

Configuration {A,X,C,Y,E}

Self-Management for Large Scale Distributed Systems (A. Al-Shishtawy)
Handling Churn

Configuration_1

SM10 r1
SM10 r2
SM10 r3
SM10 r4
Handling Churn
Handling Churn
Handling Churn

[Diagram showing a circular configuration with nodes labeled SM10 r1, SM10 r2, SM10 r3, and SM10 r4, and a configuration labeled Configuration_1 with nodes D, F, I, and B.]

SM10 r2 = G
SM10 r3
SM10 r4
Handling Churn
Handling Churn
Handling Churn
Evaluation

- Built a **prototype** implementation of RME
- **Simulation**-based performance evaluation
- Focused on the effect of the **churn rate and replication degree** on request **critical path and failure recovery**
- Used the **King latency dataset**
Request latency for a single client

![Graph showing request latency vs request number]

- X-axis: Request Number
- Y-axis: Request Latency (ms)

The graph illustrates the request latency for a single client over a range of request numbers. The latency values vary, indicating fluctuations in response times.
Outline

1. Introduction
2. Niche Platform
3. Robust Management Elements
4. Future Work
Improve Management Logic

- Apply control theory to distributed systems
- Distributed optimization
- Reinforcement Learning
Self-Management in Cloud Applications

- Study elastic services in the Cloud
- Develop self-management techniques for Cloud applications
- Integrate all pieces into an elastic storage system
Questions?