Toward Broad-Spectrum Autonomic Management

Edmund Smith <edmund.smith@stir.ac.uk>
Paul Anderson <dcspaul@inf.ed.ac.uk>

ICN 2007, April 2007
Overview

• Background
  – “system configuration” & “autonomics”
  – a comparison & a practical illustration

• The problem
  – most practical “fabrics” require elements of both
  – this means that there is usually no clear declarative specification for the fabric configuration

• A proposed approach
  – a integrated multi-resolution framework
  – further research
System Configuration

Convergence

Hardware

Configuration

Software

“Fabric” performing according to specification

Specifications & Policies
System Configuration

- Starting with:
  - several hundred new machines with empty disks
  - a repository of all the necessary software packages
  - a specification of the required service
- Load the software and configure the machines to provide the service
- This involves many internal services:
  - DNS, LDAP, DHCP, NFS, NIS, SMTP, Web …
  - the relationships are most important
- Maintain the specification when things change
  - either the requirements, or the system (failures)
Autonomics

• Autonomic computer systems are ones which:
  “Maintain and adjust their operation in the face of changing workloads, demands, and external conditions, and in the face of hardware or software failures of innocent, or malicious origin.”

  J Kephart
Autonomics

• Autonomic operations can occur at any level in the configuration “stack” -
  - recreate a corrupt configuration file on one host
  - restart a failed daemon on one host
  - redeploy a service to another host when a host fails
  - configure extra hosts into a cluster when the overall load increases

• Autonomics can be thought of as the automation of the entire system configuration process
A comparison

- The fields of *Autonomics* and *System Configuration* have evolved in separate communities with a different emphasis -
  - *System configuration* provides ways of specifying and managing the configuration of an infrastructure, and “low-level” autonomics
  - *Autonomics* provide solutions to “high-level” automation, such as service migration
- Current practical installations use elements of both
An example

- A high-level autonomic process may determine which nodes are to be used to provide the elements of a web-service - web front-end, database etc.
  - this will handle service migration and reconfiguration in response to load or failures

- Lower-level system configuration tools support (re-)configuration of the underlying infrastructure
  - DNS, Kerberos, firewall holes, backups, etc.
The problem

• Modern system configuration tools allow the fabric configuration to be maintained from an explicit declarative specification.

• This is important -
  - we can reason about the configuration - eg. for security verification
  - we can compose multiple aspects

• If the configuration is manipulated by a separate autonomic layer, then the overall declarative specification is usually lost
What do we need?

• A model which supports a uniform configuration process at all levels

• Manual and autonomic elements of the specification must be integrated so that it is possible to reason across the whole system
  - for example, about authorisation

• Autonomic components must generate configurations which are governed by declarative constraints
  - “loose” specifications
A multi-resolution framework

- Installation
- Service classes
- Services
- Aspects
- Nodes

Bindings:
- Autonomics
- Team 1
- Team 2

Specifications:
Further research

• Specification languages -
  – constraints (mcfg)
  – aspect composition (mcfg)
  – authorisation (cfgas)
  – multi-resolution specifications

• Deployment
  – distributed evaluation
  – deployment sequencing
Some Links

- Slides:

- Paper:

- Paul Anderson
  http://homepages.inf.ed.ac.uk/dcspaul
  <dcspaul@inf.ed.ac.uk>

- LISA, Large installation System Administration Conference
  November 11th-16th 2007, Dallas
  http://www.usenix.org/events/lisa07/