Planning Configuration Relocation on the BonFIRE Infrastructure

H. Herry and P. Anderson
School of Informatics, University of Edinburgh

UNICO CloudCom Workshop
5 December 2013, Bristol, UK
Outline

• Motivation
• Problem
• Solution
• Experiment
• Conclusion
Motivation

Replacing current version with the latest version without disturbing the clients.
Problem

- Live VM migration is not normally possible between different cloud sites
- Reconfiguration of every stack components
- Relocation configuration:
  - Move a declarative configuration of a system between cloud sites
- Global state constraint
Related Works

- Recontextualization with dynamic virtual devices [Amstrong et.al., 2012]
- Different technology, reconfigure application and service layer
- Declarative configurations tools such as Puppet
- Reuse the same specification on different site
- Cannot maintain global state constraints during reconfiguration
• BonFIRE is an ideal testbed
  • Federated clouds
  • Heterogeneous platforms
  • No capability of transferring VM between sites
  • Common broker to manage sites
Solution

• Nuri configuration tool
  • Automatically planning and orchestrating the relocation of multi-tier applications
  • Maintain the global state constraint

• Steps
  1. Modelling desired state
  2. Planning (automated)
  3. Orchestration (automated)
include "schemata.sfp"

proxy isa Machine {
    sfpAddress is "localhost"
    epcc isa Cloud { location = "uk-epcc"; }
    inria isa Cloud { location = "fr-inria"; }
}

vm1 isa VM {
    created = true; in_cloud = proxy.epcc
    s1 isa Service { installed = true
        running = true; version = 1; }
}

vm2 isa VM {
    created = true; in_cloud = proxy.inria
    s2 isa Service { installed = true
        running = true; version = 2; }
}

pc isa Client {
    sfpAddress is "172.18.240.21"
    refer = vm1.s1; }

include "schemata.sfp"

proxy isa Machine {
    sfpAddress is "localhost"
    epcc isa Cloud { location = "uk-epcc"; }
    inria isa Cloud { location = "fr-inria"; }
}

vm1 isa VM { created = false
    s1 isa Service;
}

vm2 isa VM {
    created = true
    in_cloud = proxy.epcc
    s2 isa Service { installed = true
        running = true
        version = 2;
    }
}

pc isa Client {
    sfpAddress is "172.18.240.21"
    refer = vm2.s2;
    global { pc.refer.running = true;
}
Model of Resource Component

```plaintext
schema Machine

schema VM extends Machine {
    created = false
    in_cloud isref Cloud
}

schema Cloud {
    location = ""
    sub create_vm (vm : VM) {
        condition {
            vm.created = false;
        }
        effect {
            vm.created = true
            vm.in_cloud = this;
        }
    }
    sub delete_vm (vm : VM) {
        ...
    }
}
```

```plaintext
schema Service {
    installed = false
    running = false
    sub install {
        ...
    }
    sub uninstall {
        ...
    }
    sub start {
        ...
    }
    sub stop {
        ...
    }
}
```

```plaintext
schema Client {
    refer isref Service
    sub redirect (s : Service) {
        ...
    }
}
```
proxy.inria.delete(vm=vm2)

proxy.epcc.create(vm=vm2)

vm2.s2.install

vm2.s2.start

pc.redirect(s=vm2.s2)

proxy.epcc.delete(vm=vm1)
Architecture

Master Node
1. SFP model (goal and global constraint)
2. Module Database
3. Compiler
4. Monitoring Manager
5. Planner
6. Plan
7. Execution Manager

Client Node
8. Agent Daemon
9. Component
10. SFP Object
11. Ruby Object
12. Shell Commands
13. Resource Ruby API

Client Node
- Agent Daemon
- Component
- SFP Object
- Ruby Object
- Shell Commands
- Resource Ruby API
Planning

• Compile the reconfiguration problem to a classical planning problem

• Planning time
  • Total variables = sum(component’s attributes)
  • Complexity of goal constraint and pre/post-conditions of procedures

• 2-stage search
  • greedy search + inadmissible heuristic
  • A* search + admissible heuristic

• Partial-order plan
Experiment

Current

INRIA
- WordpressDB (v2)
  - MySQL
  - VM
- WordpressWeb (v2)
  - Apache
  - VM
- ApacheLB
  - VM

Clients

EPCC
- WordpressDB (v1)
  - MySQL
  - VM
- WordpressWeb (v1)
  - Apache
  - VM
- ApacheLB
  - VM

Desired

INRIA

EPCC
- WordpressDB (v2)
  - MySQL
  - VM
- WordpressWeb (v2)
  - Apache
  - VM
- ApacheLB
  - VM

Clients
Experiment

- Nuri master
  - VM in EPCC: 2 CPUs + 2GB RAM
- BonFIRE proxy component
  - Connect to BonFIRE broker with restfully
- Managed system
  - VM: 1 CPU + 1 GB RAM
  - Debian Squeeze 10G v5
  - Apache Web Server, MySQL, Wordpress
  - BonFIRE WAN network
Experiment

Average after 5 times running
Experiment

Creating VMs

EPCC: OpenNebula  HPLabs: Cells
Conclusion

- Declarative specifications and Automated Planner are a viable approach to practical configuration relocation
- Nuri can automatically generate the workflow within a reasonable time
- Global constraint is maintained during relocation
Thank you to…