THE UNIVERSITY of EDINBURGH
informatics

cisa | Centre for Intelligent Systems and their Applications

LABS hp

Automated Planning for Configuration Changes

Herry

h.herry@sms.ed.ac.uk or herry13@gmail.com
http://homepages.inf.ed.ac.uk/s0978621

FLOSS UK Spring 2012
Edinburgh

About Me

- PhD student in School of Informatics, University of Edinburgh
- Supervisor
  - Paul Anderson
  - Michael Rovatsos
- Automation on system configuration
- Previous jobs
  - System engineer, system analyst
  - Lecturer

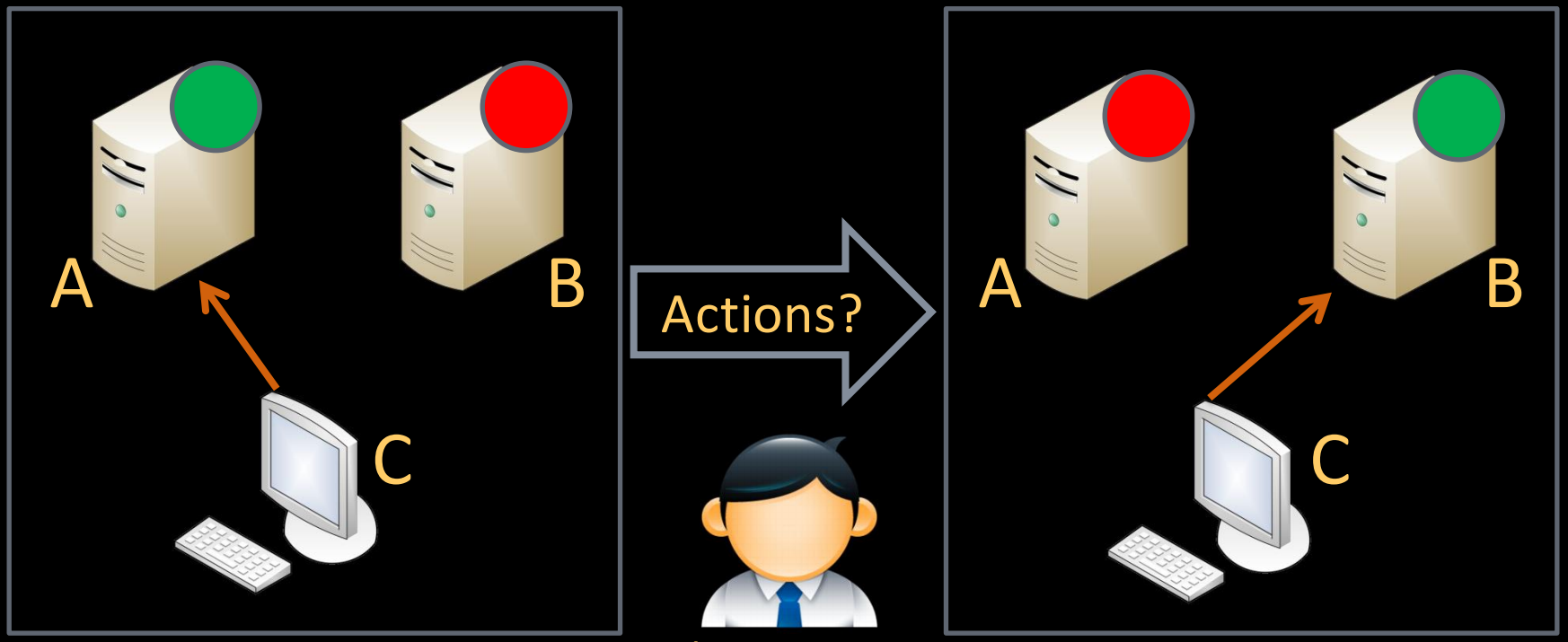
Outline

- System Configuration: Declarative Approach
- Example: Configuration Problem
  - Solution: Declarative Tool
  - Solution: Our Prototype
- New planner: SFp planner
  - SFp language
- Demo
  - Service Reference problem
  - Cloud-Burst Problem

Configuration Tools: Declarative Approach

- Most commonly used today
- Popular tools: Puppet, Chef, LCFG
- Critical shortcomings
  - Indeterminate order executions of actions
  - Could violates the system's constraints

Example: Configuration Problem



Constraint:
C must always refer to a running server!

Running Stopping

Solution: Declarative Tools

Administrator

Desired State
- A.running = false
- B.running = true
- C.service = B

Submit

Puppet
Chef
LCFG

Implement

- Possible sequences of states

1)	A.running = false	C.service = B	B.running = true	X
2)	C.service = B	A.running = false	B.running = true	X
3)	B.running = true	A.running = false	C.service = B	X
4)	A.running = false	B.running = true	C.service = B	X
5)	C.service = B	B.running = true	A.running = false	X
6)	B.running = true	C.service = B	A.running = false	√

- Highly likely producing the wrong sequence!

Solution: Our Prototype

- All actions must be orchestrated as a workflow to
  - achieve the desired state
  - satisfy the constraints
- Method – using Automated Planning technique

Declarative approach:

Our Prototype:

pre: preconditions
eff: effects

Solution: Our Prototype (2)

Administrator

Define

Desired State
- A.running = false
- B.running = true
- C.service = B

Global Constraint
- C.service.running = true

Monitoring Agent

Retrieve

Current State
- A.running = true
- B.running = false
- C.service = A

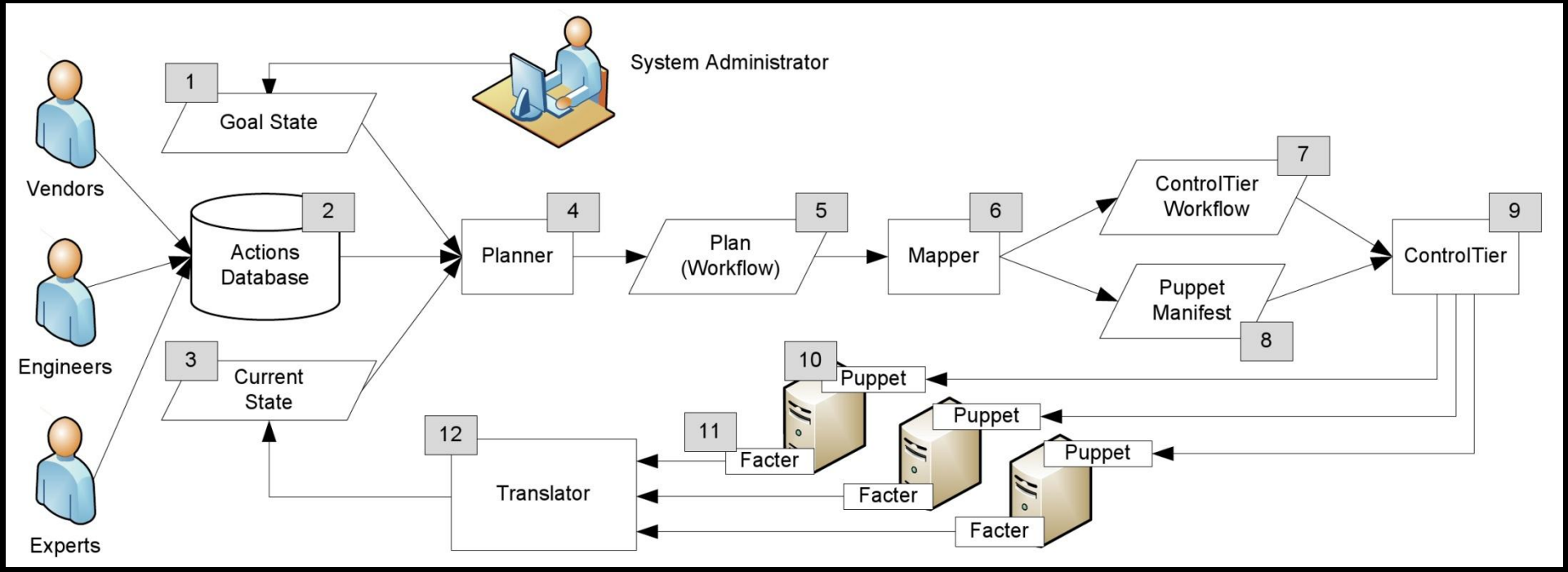
Experts, Engineers

Define

Actions

pre	start (server)	*eff*
pre	stop (server)	*eff*
pre	change (s1, s2, c)	*eff*

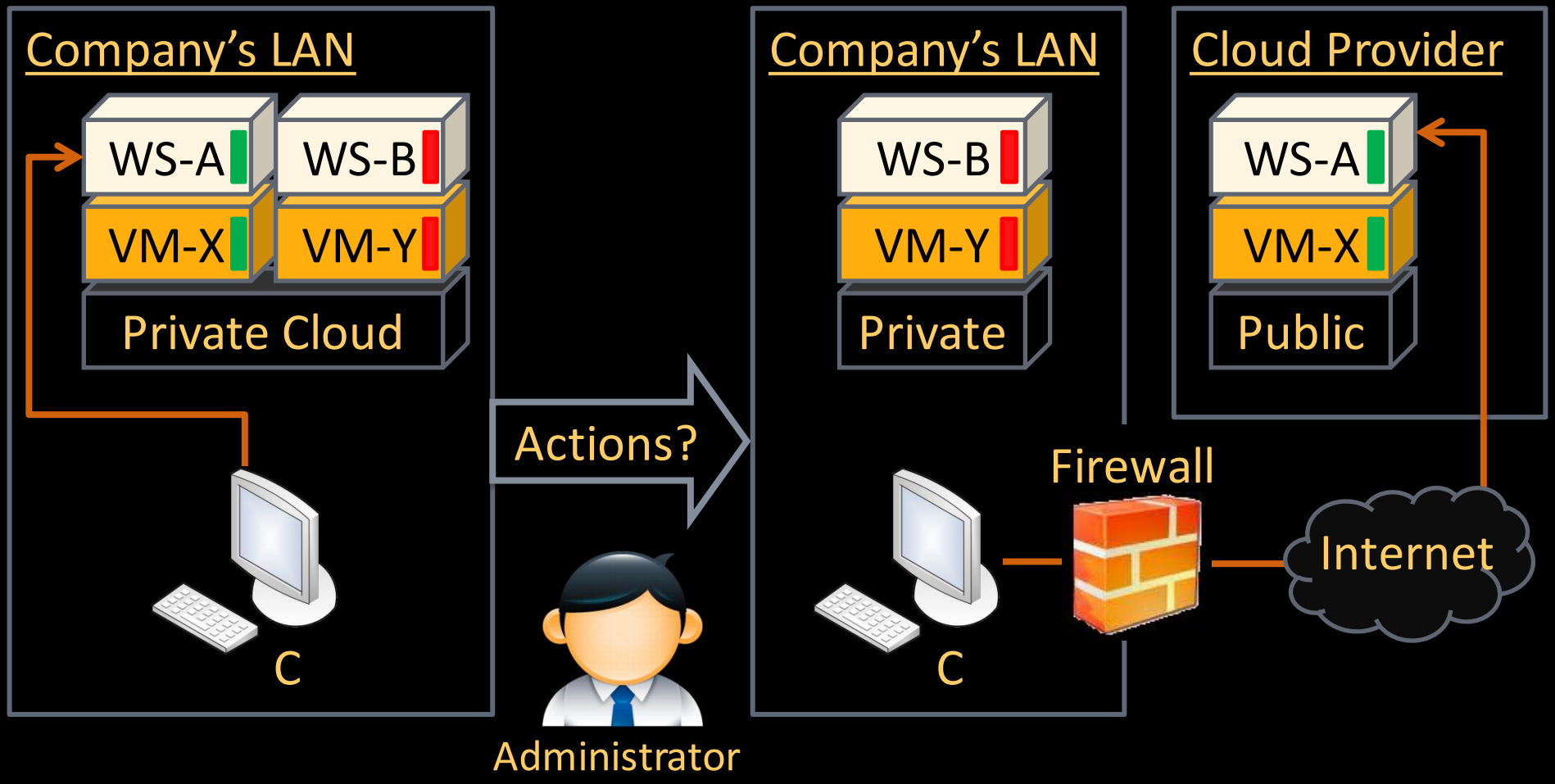
Solution: Our Prototype (3)



Experiment: Cloud-Burst Problem

- Cloud-Burst
  - Migrate application from private to public cloud
  - Address spikes in demand
- Constraints
  - No down-time
  - Reconfigure the firewall
  - Full migration but not duplication

Experiment: Cloud-Burst Problem



Demo

- http://goo.gl/Qph7F

Mapper

Generated Workflow

```
......
start-vm(vm-y,priv-cloud)
start-service(ws-b,vm-y)
change-ref(ws-a,ws-b,c)
......
```

ControlTier Workflow

```
<command name="config_changes"
 command-type="WorkflowCommand"
 description="" is-static="true"
 error-handler-type="FAIL">
......
  <command
   name="start-vm_y_priv-cloud"/>
  <command
   name="start-service_ws-b_vm-y"/>
  <command
   name="change-ref_ws-a_ws-b_c"/>
......
</command>
```

ControlTier + Puppet

ControlTier Command (start-service_ws-b_vm-y.xml)

```
<command name="start-service_ws-b_vm-y" description=""
 command-type="Command" is-static="true">
 <execution-string>exec.rb</execution-string>
 <argument-string>start-service.pp ws-b vm-y</argument-string>
</command>
```

Puppet Manifest template (start-service.pp)

```
#!/usr/bin/puppet
$service_name = "<service_name>“
$status = “running" # stopped / running
service { $service_name:
        ensure => $status
}
```

```
$ ctl-exec -I tags=vm-y -s /tmp/start-service_ws-b_vm-y.pp
```

It’s only a prototype ☹

New planner: SFp Planner

- SFp language – an object-oriented language
- Support Global Constraint
- Can be run from
  - Console
  - Web-service (HTTP Post– cURL)
- Workflow in JSON

To get SFp Planner

- Open source project
  - https://github.com/herry13/SFp-planner
  - https://github.com/herry13/SFp-planner/wiki
  - http://homepages.inf.ed.ac.uk/s0978621/sfp.html
- Web-service Planner
  - http://hpvm2.diy.inf.ed.ac.uk/sfp/planner

SFp language: class

```
class Machine {
      name          “”
      running       false
}
class PM extends Machine
class VM extends Machine {
      on            as *PM
      hasService    as *Service
}
class Service {
      name          “”
      running       false
}
class Client {
      refer         as *Service
}
```

SFp language: action

```
action startService (s as *Service) {
	precondition { }
	postcondition {
		$s.running true
	}
}
action stopService (s as *Service) {
	precondition { }
	postcondition {
		$s.running false
	}
}
action changeReference (c as *Client, s as *Service) {
	precondition {
		$s.running true
	}
	postcondition {
		$c.refer $s
	}
}
```

SFp language: action of a class

```
class Service {
      name     “”
      running  false

      action start {
            precondition { }
            postcondition {
                  $this.running  true
            }
      }

      action stop {
            precondition { }
            postcondition {
                  $this.running  false
            }
      }
}
```

SFp language: planning problem

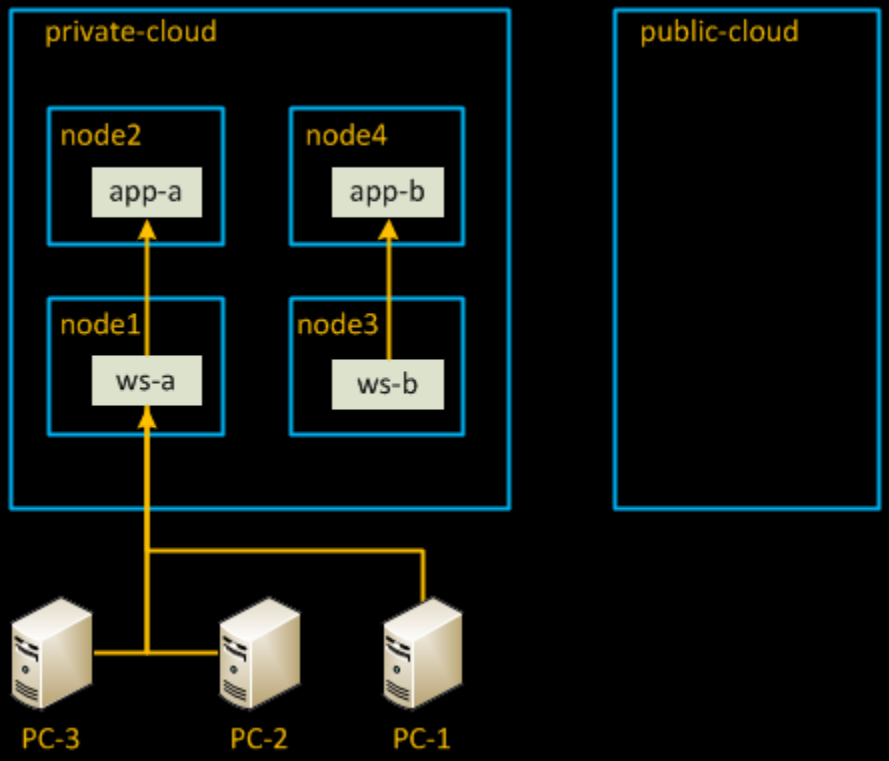
```
ws_a as Service {
      name    "HTTP Server A"
      running true
}
ws_b as Service {
      name    "HTTP Server B"
      running false
}
pc as Client {
      refer   $ws_a
}

constraint goal {
      $ws_a.running false
}
constraint global {
      $pc.refer.running true
}
```

Generated Workflow

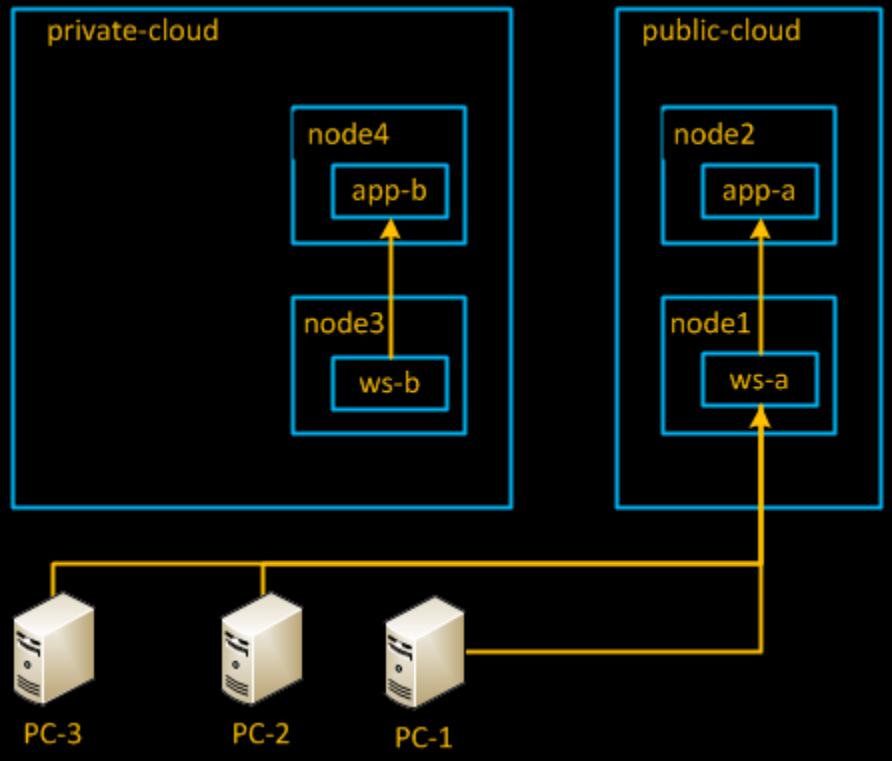
```
$ curl --data “action=sequential” --data-urlencode \
  sfp@problem.sfp http://hpvm2.diy.inf.ed.ac.uk/sfp/planner
```

```
{  "type": "sequential",
   "workflow": [
      {  "name": "startService",
         "parameters": {"s": "$ws_b"}
      },
      {  "name": “changeReference",
         "parameters": {
            "s": "$ws_b",
            "c": "$pc“
         }
      },
      {  "name": "stopService",
         "parameters": {"s": "$ws_a"}
      }
   ],
   "version": "1“,
   "total_actions", “3"
}
```

Use case: Cloud Burst



Initial State



Desired State

Demo

- http://hpvm2.diy.inf.ed.ac.uk/sfp/planning

Incoming Features

- Parallel Workflow
- Numerical operation
- Cost function
- Array/Set data structure
- Enumeration

Thank you!