THE UNIVERSITY *of* EDINBURGH informatics

cisa Centre for Intelligent Systems and their Applications

Planning with Global Constraints for Computing Infrastructure Reconfiguration

Herry and Paul Anderson

h.herry@sms.ed.ac.uk
http://goo.gl/c4wK3

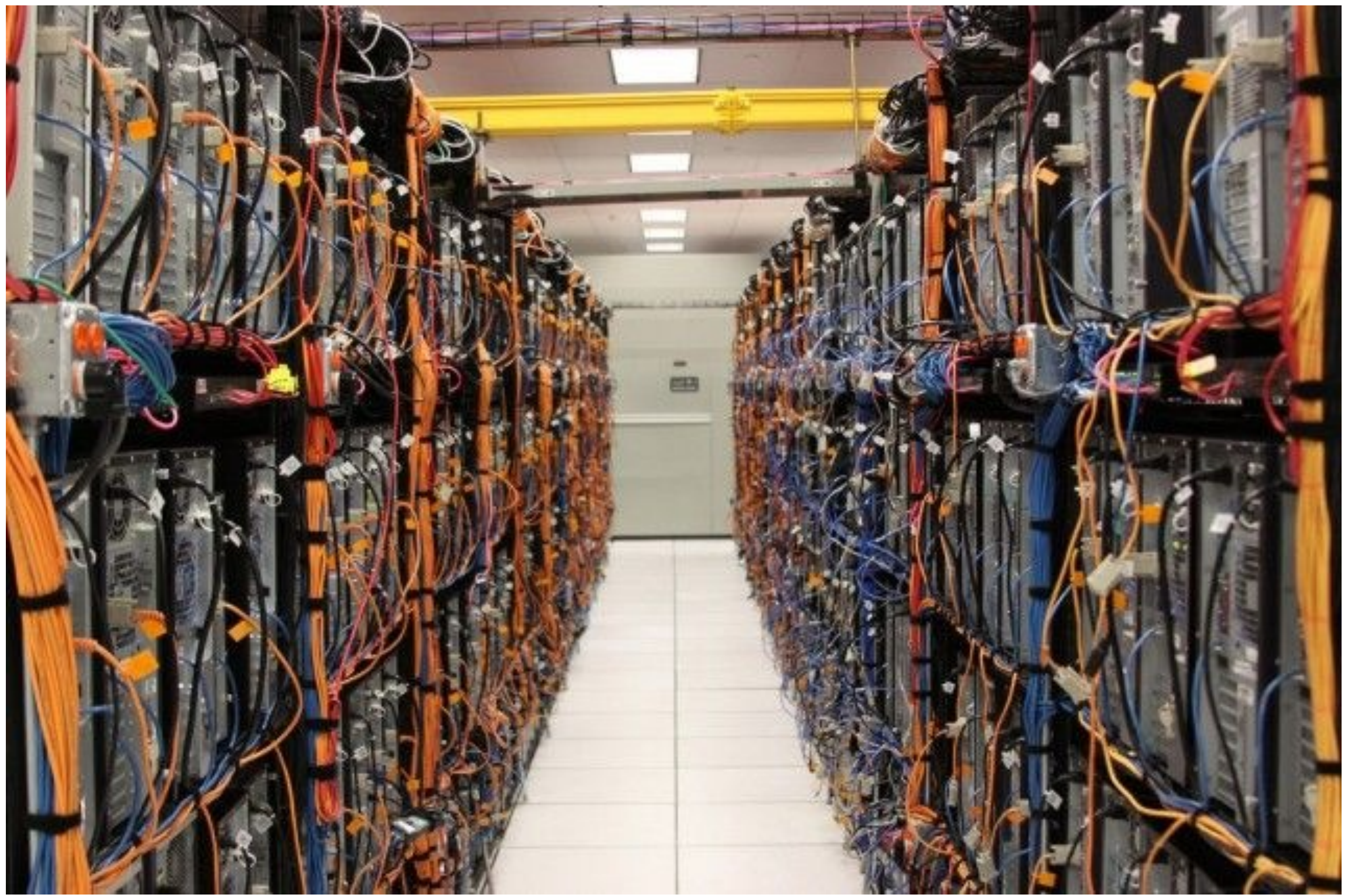
AAAI-12 Workshop on
Problem Solving using Classical Planners
Toronto, July 2012

Outline

- System Configuration
- Current Approaches
- Global constraints
- Proposed Approach
- Example Use-Case: Cloud-Burst
- Conclusion

System Configuration

- Transform computing resources into a functioning system according to a given specification
- Resources – operating system, software, user-account, services, network, etc
- Challenges
  - Size
  - Multi-platform
  - Dependencies
  - Multi-aspect
  - Multi-user



Current Approaches (Popular)

Method	Declarative	Fixed Workflow	Dynamic Workflow
User	Define goal state	Define workflow	Define goal state
Tools	Select & execute actions	Execute the given workflow	Generate & execute workflow
Ordering	No	Yes	Yes
Examples	Puppet LCFG BCFG SmartFrog Chef	IBM Tivoli Microsoft Opalis ControlTier	Keller et.al. ('04) El Maghraoui et.al.('06) Hagen et.al. ('10) Herry et.al. ('11)

Global Constraints

- Constraints that must be satisfied in the intermediate (during reconfiguration) and goal states
  - DNS service must be available all the time

- None supports global constraints
- User could modify particular actions
  - It's not feasible in practice

Why need Global Constraints?

- Modifying actions is as hard as planning
- Administrator usually does not have deep knowledge of the resources
- Lack of permission
  - Copyright, License
- Separation of concerns
  - Reusability of components

Proposed Approach

- SFplanner
  - Allowing user to define global constraints
  - Use planner to generate workflow
- Modelling the configuration problem
  - SFp language
- Planner
  - Fast-Downward

SFp language

- Extension of SmartFrog (SF) language
- Current
  - Object-oriented
  - Modules
  - Goal & Global constraints
  - Reference
- Future
  - Abstract Data-Types (Array, Set)
  - Create/delete objects
  - Composite

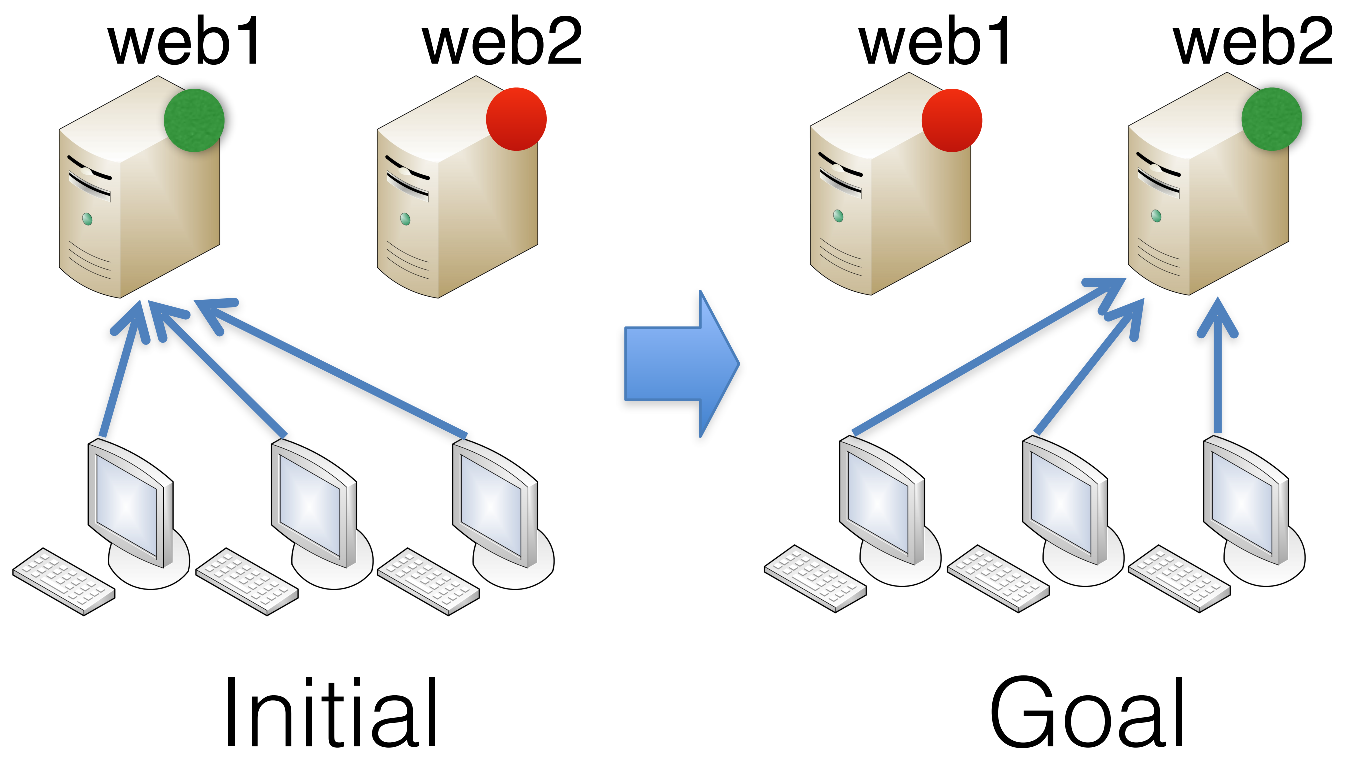
server.sfp

```
class Server {
   running false
   action start {
      precondition { }
      postcondition {
         $this.running true
      }
   }
   action stop {
      precondition { }
      postcondition {
         $this.running false
      }
   }
}
```

client.sfp

```
class Client {
   refer as *Server
   action changeReference(s as *Server) {
      precondition { }
      postcondition {
         $this.refer $s
      }
   }
}
```

SFp language



desktops.sfp

```
#include file("client.sfp”)

pc1 as Client {
   refer $web1
}
pc2 as Client {
   refer $web1
}
pc3 as Client {
   refer $web1
}
```

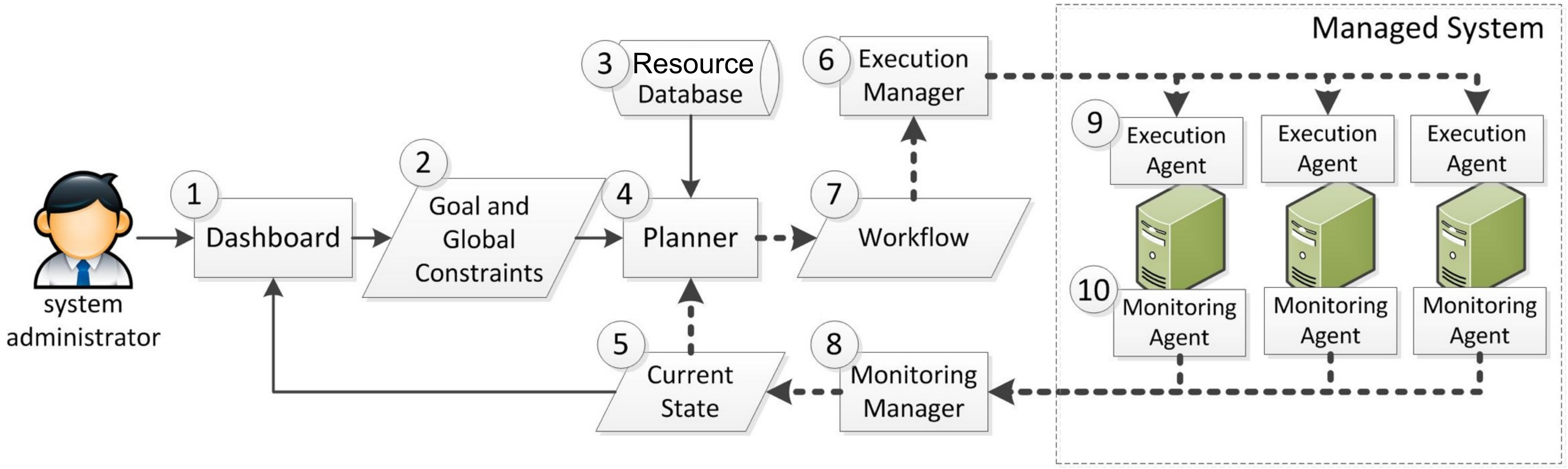
web-servers.sfp

```
#include file(”server.sfp”)

web1 as Server {
   running true
}
web2 as Server {
   running false
}
```

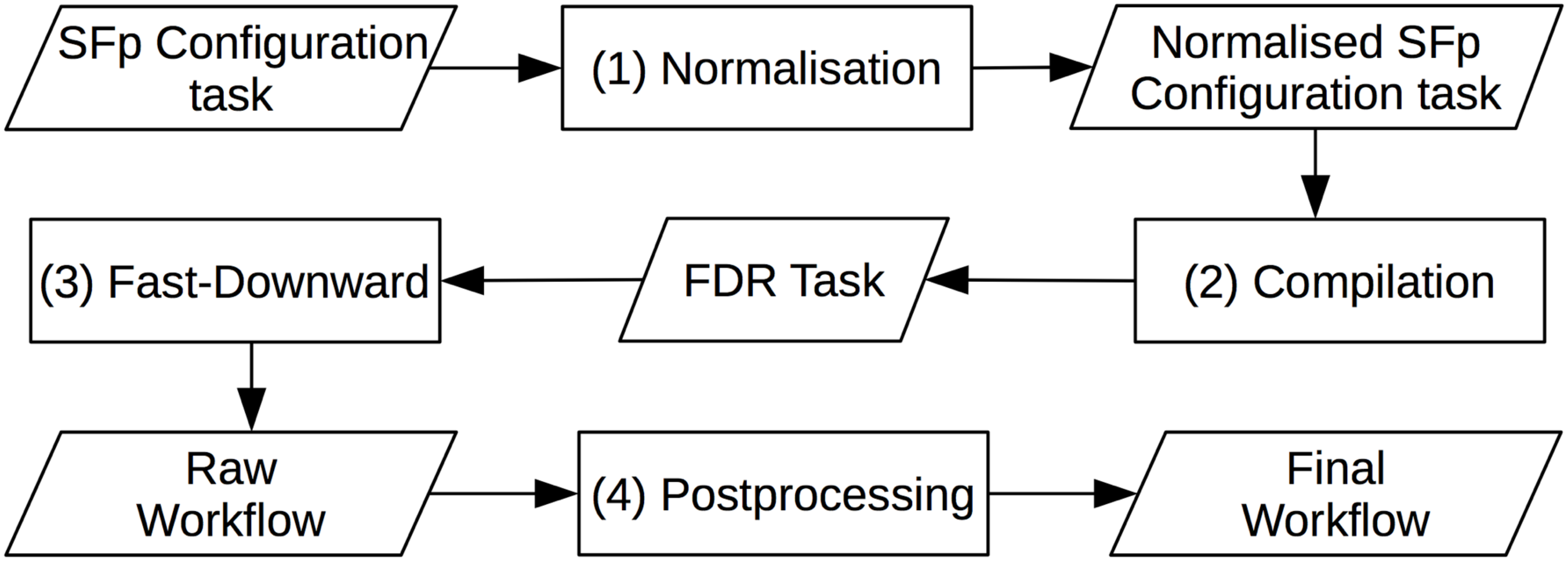
system1.sfp

```
#include file(”web-servers.sfp”)
#include file(“desktops.sfp”)

constraint goal {
   $web1.running false
}
constraint global {
   $pc1.refer.running true
   $pc2.refer.running true
   $pc3.refer.running true
}
```

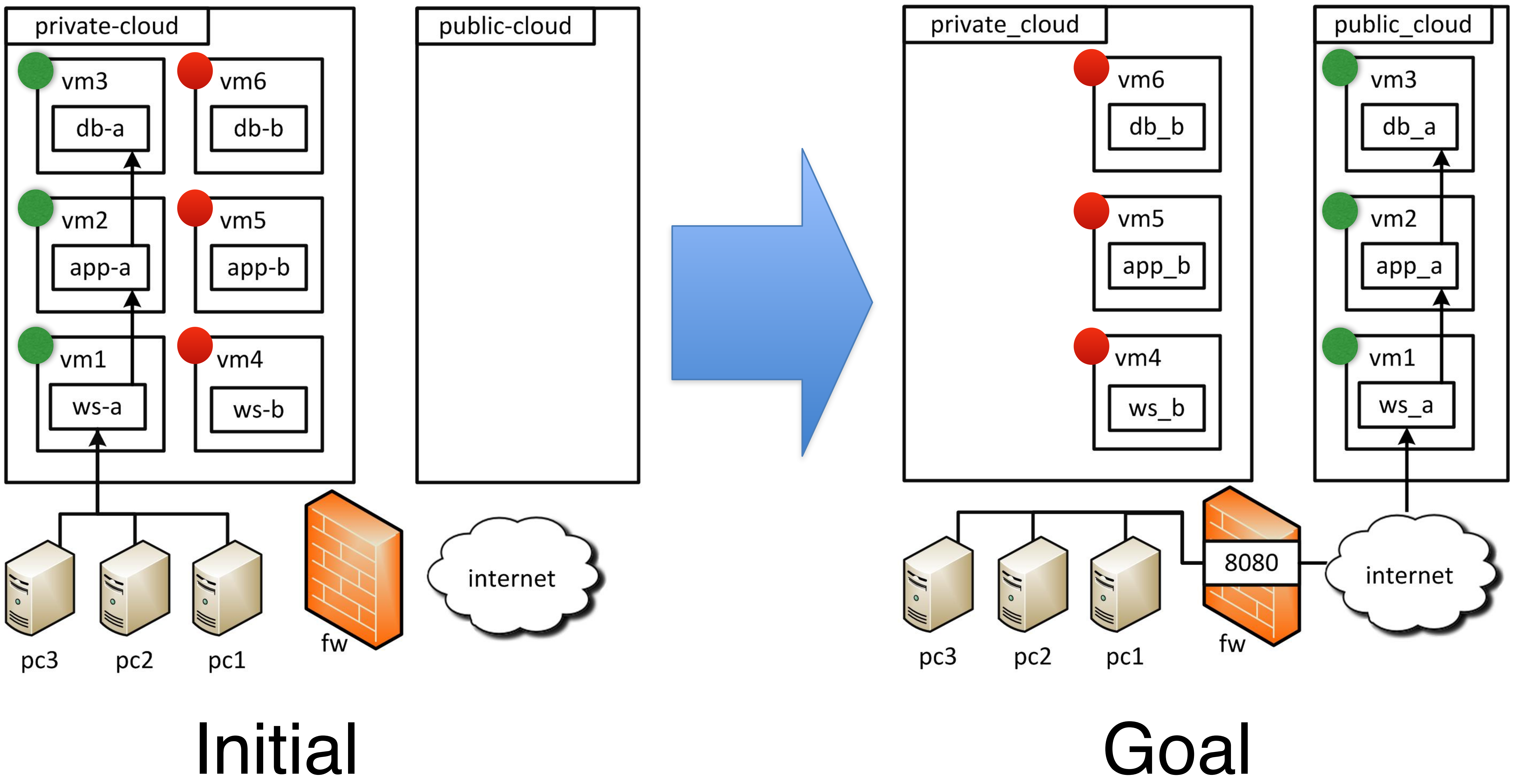
SFplanner


Managed System
3 Resource Database
6 Execution Manager
9 Execution Agent
Execution Agent
Execution Agent
1 Dashboard
2 Goal and Global Constraints
4 Planner
7 Workflow
system administrator
10 Monitoring Agent
Monitoring Agent
Monitoring Agent
5 Current State
8 Monitoring Manager

Planning Process

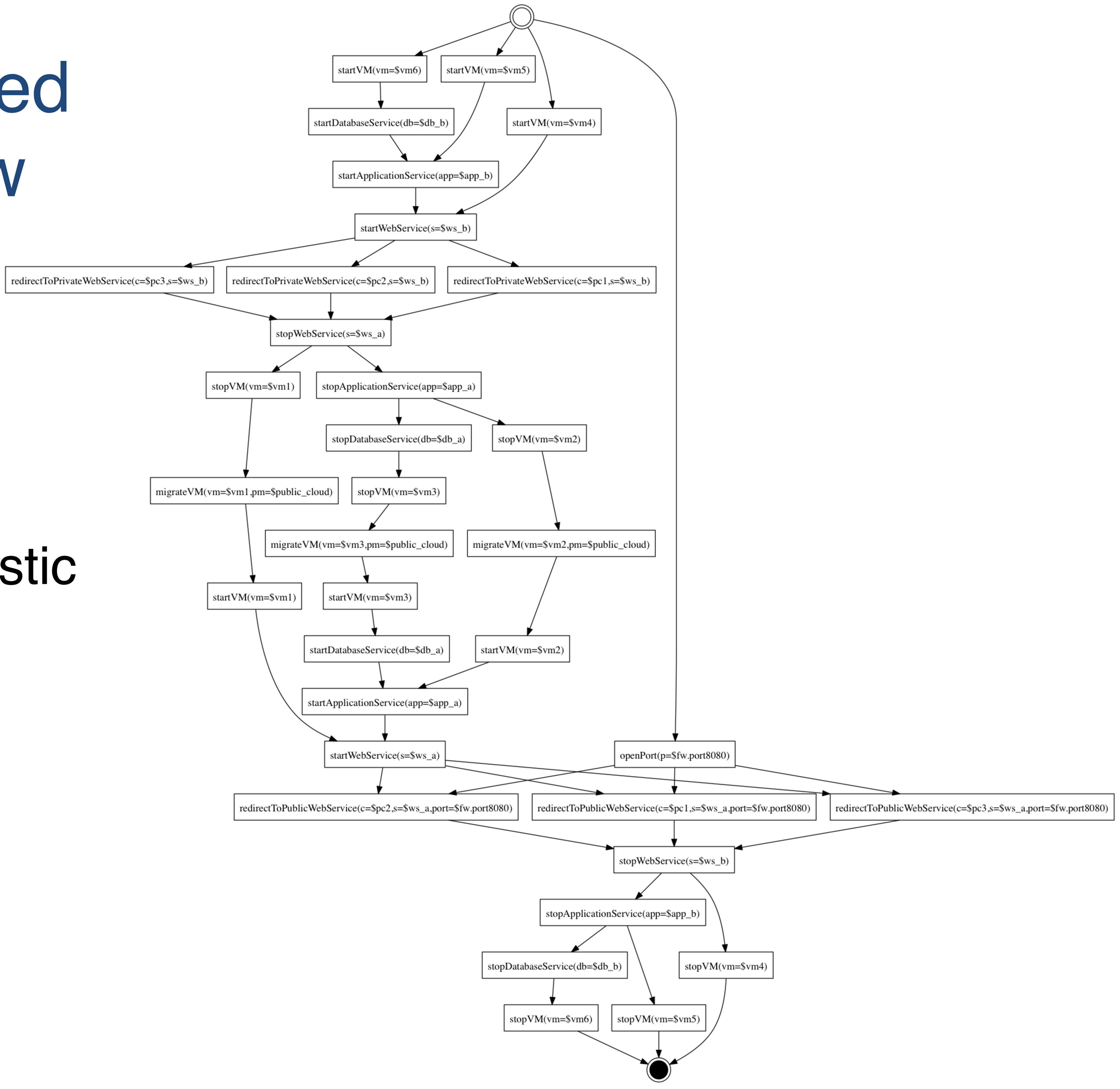


Example: Cloud-Burst



Generated Workflow

- FF Heuristic


startVM(vm=$vm6)
startVM(vm=$vm5)
startDatabaseService(db=$db_b)
startVM(vm=$vm4)
startApplicationService(app=$app_b)
startWebService(s=$ws_b)
redirectToPrivateWebService(c=$pc3,s=$ws_b)
redirectToPrivateWebService(c=$pc2,s=$ws_b)
redirectToPrivateWebService(c=$pc1,s=$ws_b)
stopWebService(s=$ws_a)
stopVM(vm=$vm1)
stopApplicationService(app=$app_a)
stopDatabaseService(db=$db_a)
stopVM(vm=$vm2)
migrateVM(vm=$vm1,pm=$public_cloud)
stopVM(vm=$vm3)
migrateVM(vm=$vm3,pm=$public_cloud)
migrateVM(vm=$vm2,pm=$public_cloud)
startVM(vm=$vm1)
startVM(vm=$vm3)
startDatabaseService(db=$db_a)
startVM(vm=$vm2)
startApplicationService(app=$app_a)
startWebService(s=$ws_a)
openPort(p=$fw.port8080)
redirectToPublicWebService(c=$pc2,s=$ws_a,port=$fw.port8080)
redirectToPublicWebService(c=$pc1,s=$ws_a,port=$fw.port8080)
redirectToPublicWebService(c=$pc3,s=$ws_a,port=$fw.port8080)
stopWebService(s=$ws_b)
stopApplicationService(app=$app_b)
stopDatabaseService(db=$db_b)
stopVM(vm=$vm4)
stopVM(vm=$vm6)
stopVM(vm=$vm5)

Conclusion & Future Works

- SFplanner generates workflow to attain the goal state and also preserving global constraints
- Classical planner could solve configuration problems
- Ongoing/future works
  - Soft global constraint
  - Larger size of use-cases (still collecting)
  - More distributed and localized approach to increase resilient

Source codes: http://github.com/herry13

Thank you

Q & A