Automated System Configuration

John Hewson, Paul Anderson, Andy Gordon*

*Univrsity of Edinburgh. *also Microsoft Research, Cambridge*

cisa | Centre for Intelligent Systems and their Applications

Microsoft Research

THE UNIVERSITY of EDINBURGH informatics

Abstract

The complexity of typical computing installations has increased to the point where automated configuration is desirable. We are developing a high-level, declarative specification language (ConfSolve) for system configuration, from which valid configurations are inferred via compilation of the specification into a Constraint Satisfaction Problem (CSP). A key issue we are focusing on is the minimisation of the number of changes which occur when performing a re-configuration.

Cloudbursting

To examine the problem of minimal changes, we present an example of Cloudbursting - whereby an enterprise scales-up their virtual machine placement into the cloud.

ConfSolve

ConfSolve is object-oriented, in keeping with popular modern system configuration tools such as Puppet, a simple class model describes the datacenter classes and relationships:

```
class Service {
   var host as ref Machine;
}
class Datacenter {
   var machines as Machine[8]
}

class Machine {
}

class Web_Service extends Service {
}

class Worker_Service extends Service {
}

class DHCP_Service extends Service {
}
```

We now declare two datacenters: enterprise and cloud, and 3 services which will be run on them.

```
var cloud as Datacenter
var enterprise as Datacenter

var dhcp as DHCP_Service[2]
var worker as Worker_Service[2]
var web as Web_Service[3]
```

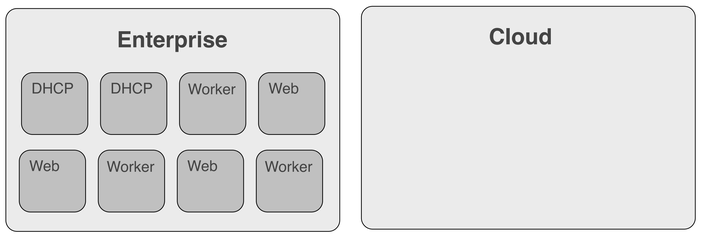
These services will be placed according to the following constraints:

```
// no two services on same machine
var services as ref Service[7]

where foreach (s1 in services) {
   foreach (s2 in services) {
      if (s1 != s2) {
         s1.host != s2.host
      }
   }
}
```

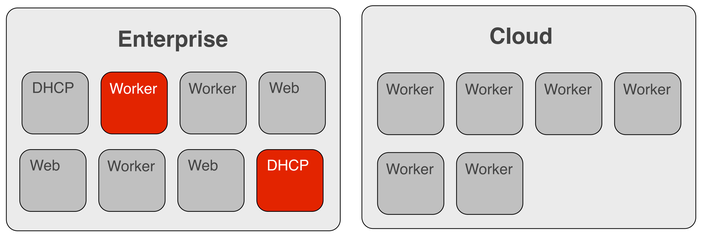
We don't want to scale-up into the cloud if the enterprise datacenter still has capacity, so we add the following constraint:

```
// favour placement of machines in the enterprise
var utilization as int

where utilization == count (s in services where
s.host in enterprise.machines)

maximize utilization
```

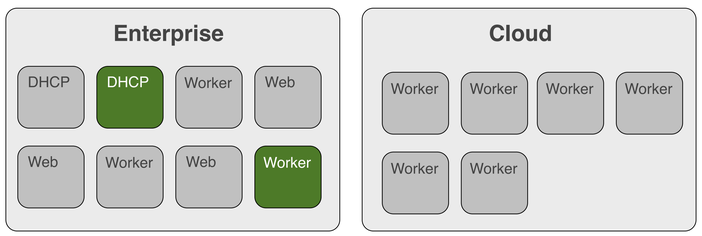
Solving the problem gives us the following configuration:



However, if we scale-up the number of workers, we get an undesirable result - the placement of two machines in the enterprise datacenter is unnecessarily switched:



ConfSolve can automatically add constraints to mimimise the distance form the previous solution, the problem can be avoided, and we get the expected configuration:



The next challenge is to allow the user to declaratively specify which changes are more important than others, and what his preferences are for optimising the original problem, versus minimising the number of changes.

Acknowledgements

This work was supported by Microsoft Research through its European PhD Scholarship Programme.