

Programming the Datacentre

Challenges in System Configuration



Paul Anderson <dcspaul@ed.ac.uk>

The Rise and Rise of the Declarative Datacentre
May 12-13, 2008, Microsoft Research Cambridge

[http://homepages.inf.ed.ac.uk/dcspaul/
publications/r2d2-2008-talk.pdf](http://homepages.inf.ed.ac.uk/dcspaul/publications/r2d2-2008-talk.pdf)

Programming the datacentre

- The difficulties of configuring a datacentre have been increasing steadily over the years
- Virtualisation has recently increased the complexity
 - *but it also makes the entire infrastructure programmable and raises the possibility of complete automation*
- The problem seems similar to the development of high-level programming languages
 - *can we learn anything from exploring this analogy?*
- Modern high-level languages developed out of collaboration between theory and practice
 - *perhaps this is required to make progress on datacentre configuration*

Overview

- System configuration
 - *configuration as programming*
- A declarative approach
 - *the target machine*
 - *the source language*
 - *trust*
 - *deployment*
 - *specifying behaviour*
 - *state transitions*
- Constraints
- Aspects

System Configuration

**SERVICE
SPECIFICATION**



Configuration as programming

- Most installations are manually managed at a low level
 - *humans decide the details of the configuration file contents*
- What might the input to a “configuration compiler” look like?
 - *machine X is a web server*
 - *there should be at least two web servers in each building*
 - *all HTTP requests should be serviced in 2 seconds*
 - *if possible, service X should only run on hardware Y*
 - *customers should be allowed to define their own values for their web server parameters, as long as the security specialist doesn't prevent this*

A declarative approach

- A declarative approach seems even more natural for specifying configurations than for traditional programs
 - *describe the desired state*
- But automatically computing and implementing the changes between states is difficult
- Most system administrators are used to planning and implementing these changes themselves
 - *so they tend to think in terms of procedures, rather than goals*
- Most practical tools are concerned with deployment
 - *rather than specification*

The target machine

- The machine code of a traditional processor presents a clear model for the target of a programming language
 - *this is not the case for configuration tools ...*
- Different tools target different levels of abstraction:
 - *opaque disk images*
 - *detailed configuration file contents*
 - *something in-between*
- And assume different levels of structure:
 - *rich, common schema*
 - *simple parameters*

The source language

- The source language needs to be
 - *able to support varying levels of abstraction*
 - *clear!*
- Different categories of users have different requirements
 - *in the case of programming languages, many different languages can be combined into the same object code*
 - *in the configuration case, different communities have their own, incompatible approaches*
- Is a common approach possible?
 - *this would concentrate the effort*

Trust

- Administrators need confidence in any tool
 - *the consequences of errors can be serious*
 - *adequate testing is difficult*
- Most (current) tools are capable of destroying themselves given the wrong input!
 - *similar to self-modifying code*
- Administrators will want to see the explicit actions generated by any compiler
 - *early programmers would often inspect the machine code generated by a compiler*

Deployment

- The “target machine” is highly distributed
 - *the programmer needs to be isolated from the details*
 - *distributed reasoning may be involved*
- The system must be considered as highly unreliable
 - *including the deployment system itself*
- Latencies must be considered
 - *the “actual configuration” is almost always different from the “desired configuration”*
- Feedback is also unreliable
 - *there is always a degree of uncertainty about the actual state*

Specifying behaviour

- At the highest level, we are interested in the behaviour of the datacentre
 - e.g. *I want a 2s response time to http requests*
- There is no obvious way to generate configuration parameters directly from this type of specification
 - *it requires feedback and “autonomics”*
 - *the results depend on external factors*
- Solutions to specific problems are fairly straightforward
 - *but these occur at multiple levels*
 - *and a general approach is less obvious*

State transitions

- In a conventional program, we do not usually care about the state of the system during the computation
 - *we do care about transitional states during a configuration change*
- Changing a server from A to B involves
 - *configure server B*
 - *move all the clients*
 - *decommission server A*
- This is a planning problem
 - *with declarative constraints on the intermediate stages*

Constraints

- Many high-level requirements translate quite naturally into logical constraints
 - e.g. *I want redundant DHCP servers in different buildings*
- If these can be implemented directly, then we can have confidence that the resulting configurations match the intent
 - *the constraints can be interpreted in isolation*
 - *they do not depend on order*
- Specifying requirements as constraints is also important for autonomies
 - *the system has the freedom to reconfigure while still meeting the requirements*

Issues with constraints

- Slight changes in the constraints should not produce wildly different results
 - *I don't want the database server to move unnecessarily when I change the mail server requirements*
- A typical configuration would have constraints with varying degrees of “hardness”
 - *satisfying them all would not be possible*
 - *but they would have priorities*
- Writing constraints requires more thought
 - *it is easy to over- or under- constrain values*
 - *there can easily be unintended consequences*
- Current solvers are not adequate

Aspects

- Many people are involved in specifying some aspect of the datacentre configuration
- But these people have overlapping concerns
 - *they cannot be partitioned into independent modules*
- We would like people to be able to specify their requirements independently
 - *this has some analogies with aspect-oriented programming*
- Specifying constraints solves many of these problems
 - *constraint union is often appropriate*
 - *but the composition operations are sometimes more complex - e.g. numerical addition*

Programming the Datacentre

Challenges in System Configuration



Paul Anderson <dcspaul@ed.ac.uk>

The Rise and Rise of the Declarative Datacentre
May 12-13, 2008, Microsoft Research Cambridge

[http://homepages.inf.ed.ac.uk/dcspaul/
publications/r2d2-2008-talk.pdf](http://homepages.inf.ed.ac.uk/dcspaul/publications/r2d2-2008-talk.pdf)