Directing Change with Bcfg2

Narayan Desai, Rick Bradshaw, Joey Hagedorn, Cory Lueninghoener

Narayan Desai

desai@mcs.anl.gov

LISA06

December 7th, 2006

A U.S. Department of Energy laboratory managed by The University of Chicago

Changes...

- Are unavoidable in today's networks
  - User demands
  - Security patches
  - Integration with externally controlled systems
- Network sizes continue to grow
- Service scale continues to increase
- Security continues to be an ongoing issue
- *Complexity is not going away*

Supporting Change

- Three main factors
  - Low per-change cost
  - Fine-grained deployment control
  - *Representing changes over time*
- Configurations aren't static, so tools should not treat them as if they are
- Goals:
  - Robust client reconfiguration workflows
  - Understanding how configurations change and propagate
  - Controlling change staging and deployment

Bcfg2

- Client/Server architecture
  - Server specification describes our desired configuration
  - Client configuration state information reflects reality
  - Comparison yields a configuration specification accuracy metric
- Group-based description mechanism
- Client-side state feedback
- Mature software
- In production use across many sites
  - Research, Academia, Corporate, Finance

Change Support Requirements for Tools

- Time as an independent variable
  - Without this, administrators can only interact with the current specification state
- Client-side state feedback
  - Needed for understanding change propagation
  - Must be time-sensitive
  - Coordinated with specification
- Not tool specific in any way
  - We have implemented this with bcfg2, but other tools can implement it just as easily

Constructing Time as an Independent Variable

- Specification Revision Control (with subversion)
  - Specification is put under revision control
  - Yields a per-repository revision unique identifier
  - Server can be pegged at a given revision
- Server-side Modifications
  - Revision identifier tracking
  - Per-client configuration tagging
- Client-side Modifications
  - Statistics tagging with source identifier
- Reporting system modifications
  - Reflecting identifier in system state summaries
- Net result is time correlated feedback, from specification to deployment

New Capabilities

- Opened up a number of exciting new possibilities
  - Fine-grained Change Management
  - Change Propagation Analysis
  - Change Orchestration
- Allowed us to think about the process differently

This approach is applicable to any tool!

Change Management

- Control change creation and deployment individually
  - Administrators can commit to the repository at any time
  - Server consumption of the repository controlled independently
- Allows reliable implementation of change windows
- Supervised change performance

Change Analysis

- Correlation of specification consumption and deployment allows observation of change propagation
- Understand change patterns
  - While systems change often and why
  - Patterns of client updates
- Detailed change reporting

Change Orchestration

- Describing a complete workflow is now possible
  - Put each of the states into the specification subversion repository
  - Describe preconditions for each state
  - Deliberately advance the state when changes are sufficiently propagated
- Workflow guidance is possible, as well as execution
  - The list of all failing predicates => tasks that remain in a step
  - The deployment process can be performed in as manual or automatic fashion as desired, with automatic bookkeeping in any case
- Enables "fire and forget" reconfiguration tasks

Results

- This technique can be applied to any configuration tool, regardless of overall architecture
- It provides an explicit representation of the overall processes we all manually track
- *Everything is now observable*

Questions?