BCFG: Validation and Change Detection

Narayan Desai
desai@mcs.anl.gov
10/25/03

Initial Observations

- Existing tools don't scale well to diverse administrator groups
  - Some administrators aren't comfortable “programing” the environment.
  - Tools don't recover gracefully from chaos.
  - Method consensus is rare.
- Existing user interfaces only partially address daily administration tasks
  - Imaging systems allow ad-hoc problem solving on a live configuration, but don't allow abstraction across multiple configurations
  - Metadata-based systems require programming mentality for configuration updates. However, configuration is reusable across multiple configurations.

Goals

- Imaging system style user interface
- Underlying metadata-based configuration repository
- Suitable for use on a testbed cluster
- Research platform for configuration management research

BCFG Architecture

- Client/Server
- Client has no access to configuration repository
- The configuration repository is metadata based
- A classing mechanism is built into all aspects of the configuration repository
- The server-side configuration for a client is defined to be correct, despite actual state of client configuration

Idealized Update Model

- Client on-disk configuration updated
- Changes tested
- Configuration put tool run
  - Changes detected
  - Administrator interrogated to categorize changes
  - Changes, tagged with metadata, uploaded to the server
  - Changes integrated into the configuration repository

Configuration Change Detection

- Changes detected using verification
- Verification process is two pass
  - First pass is verification of specified client configuration
    - Detail verification
    - Can find many configuration changes
  - Second pass detects extra configuration
    - Configuration details not previously contained in the client configuration
    - Heuristics abound
- Configuration changes consist of any failures in pass 1 and any configuration detected by pass 2

Change Detection Requirements

- Client configuration requirements
  - Comprehensive
  - Composed of verifiable entities
    - Entities also comprehensive
    - **No opaque entities**
    - **No logic**
  - Defines state, not process

Generators

- Comprehensive client configuration requirement means all configuration must transit the configuration management system
- Generators are server side executable logic
- Used when system configuration not well suited by model
  - Allows integration of external data sources
  - Provides an external interface to high level changes
  - Provides triggers on data access
  - Context specific constructs
- Example generator: user account configuration on cluster nodes
  - Interface to user managent system
  - Access to compute nodes dependent on scheduler state

Heuristics

- Heuristics are designed to find configuration changes made in good faith
  - This is not a tripwire replacement
- Simple heuristics implemented
  - Check for any extra files in /etc
  - Check for extra packages installed
  - Check for extra active services
- These are sufficient, but not perfect
- More complicated algorithms could easily be implemented, but so far there has been no need

Benefits of Verification

- Most configuration processes simplified
  - Manual change assimilation accelerated
  - Initial configuration generation automatable
- Results of fire-fighting sessions summarized
- Chaotic activity can be recovered from
- Permanent configuration skew can be avoided
- Automated change assimilation now possible

Next Steps

- Automated change assimilation
- Automate initial configuration build
- More heuristics with input from other users
- Change notification interface to allow sparse configuration propagation
- Accommodate other problem solving styles