Bcfg2: Use, Care, and Feeding

Narayan Desai
desai@mcs.anl.gov
January 7, 2005

Outline

- Bcfg2 Overview
- Client Usage
- Server Architecture
  - Common Messages and Problems
- Examples

Bcfg2 Overview - Features

- metadata based client configurations
- change detection
- improved server performance
- 2 way configuration verification
- sparse configuration installation
- probes
- dry-run mode
- profiles

Bcfg2 Overview – Benefits

- Changes are made in a single location
- Configuration fragments are composeable
- Machine functionality can be easily duplicated for testing or upgrades
- Configuration specifications are designed to be reusable
  - Each decision is only encoded once
- Installation process is automated
  - Services are automatically restarted (when possible)
  - Complete service configurations are rechecked whenever touched (needed on redhat)

Bcfg2 Overview - Architecture

- Client / Server
- “dumb” client
  - almost no client side processing
  - minimizes client bootstrap dependencies
- logic is server side
  - served configuration describes complete machine configuration
  - no client race condition
  - can completely rebuild machines without locate state dependence

Bcfg2 Overview – Configuration Process

- Client requests probes
- Server provides a list of probes appropriate to <client>
- Client probes and uploads results
- Server routes probe results to generators as needed
- Client requests configuration
- Server generates and serves the configuration (described in detail later)
- Client inventories, locates configuration updates, and makes them
  - Client loops while making progress
- Client determines local state (clean/dirty) and uploads it
- Server logs last seen client state

Bcfg2 Overview - Integration

- Run from the systemimager miniroot
  - All systems up to date upon build completion
  - latest bcfg2 client is used
  - client is run in “build mode”
  - can specify client profile

Bcfg2 Overview – Upcoming Features

- More toolsets
  - Redhat support (completed)
  - Solaris support
  - Gentoo support
- Improved statistics support
  - automatically locate configuration problems
- Templating module
  - remove the need for bcfg2 extension for some complex tasks
  - improve probe reusability
  - provide compatibility with high-level reasoning systems
- Change management support
  - Revision Control
  - Change ingress control mechanism

Bcfg2 – Useful Client Options

- -v: verbose operation
- -d: debugging info (add information about decision making processes during the client run)
- -n: dry-run mode (no changes are made to the client)
- -c: cache a copy of the served configuration
- -b: only install specified bundle
- -q: only check package versions, do not check checksums (this options substantially speeds up execution and is automatically used in build mode)
- -B: run in build mode (do not check file checksums, and do not enable services)
- more documented in the man page

Client Usage Scenarios

- Workstation environment
  - Run in build miniroot
  - Run on boot
  - Run through cron
- Cluster environment
  - Run in build miniroot
  - Run on boot
  - Limited scope run during job startup and completion
  - Deliberate change flushing runs between jobs
  - Run through cron on non-scheduled nodes

Example Runs

- Client already correct (no changes made)
- Client misconfigured
  - Reconfiguration possible – end result clean state
  - Reconfiguration impossible – end result dirty state
  - Extra configuration present – not currently removed

Bcfg2 Server - Functions

- Store metadata and configuration repository
  - Provide classing mechanism for client description
- Provide configurations to clients
- Track current state of all clients
- Coherently cache repository data
  - Uses fam for update
  - No server restarts are required for configuration changes

Bcfg2 Server - Repository

- Location determined in /etc/bcfg2.conf
- Metadata and other single files in <repo>/etc
- Each generator requiring a file repository has its own directory
  - ie, <repo>/<generator>
- Each generator repository is structured for its task (discussed later)

Bcfg2 Server - Metadata

- Defines all information about clients
  - Hostname – hostname
  - Image – base distribution
  - Classes – set of functional classes
  - Attributes – modifications to classes
  - Profiles – set of classes and attributes
- stored in <repo>/etc/metadata.xml
- Whole configurations (minus host-specific information) encapsulated in profiles
- Default configurations added for new hosts upon initial connection

Bcfg2 Server – Configuration Generation

- Client requests configuration
- Server performs metadata lookup
- Server queries structures for abstract configuration fragments
  - Abstract configuration fragments are configuration elements grouped into dependent groups (ie, bundles) without client specific information
- For each entry in all AC fragment, a generator is called to bind element information (package version, cf contents, etc) into the configuration
  - Failures (which can occur due to configuration repository mistakes) logged
- The bound configuration is marshalled into XML, and served to the client
- Configuration generation time is logged

Bcfg2 Server - Structures

- Generate abstract configuration fragments
- Two included with stock distribution (using a standard API)
  - Bundler
    - Implements dependent groupings of configuration information
      - ssh bundle
    - Allows class specific additions of configuration (image specific, etc)
  - Base
    - Independent groupings of configuration information
      - the pile of packages needed to make machines work
    - Allows class specific additions of configuration

Bcfg2 Server - Generators

- Used to bind specific information into client configuration
  - Configuration element source
- Can implement any site-specific policy as needed
  - Imperative encoding of decisions about site configuration
- Enabled in /etc/bcfg2.conf
- Several generators included in standard distribution
  - Cfg – configuration file repository
  - Pkgmgr – available packages
  - Servicemgr – service to client mappings
  - Chiba – generate files for chiba
  - SSHbase – manage ssh keys (including public key export)
  - Debconf – manage debconf settings
  - Accounts – setup accounts for chiba

Generators - Cfg

- Configuration file repository (similar to cfg-get)
- based in <repo>/Cfg
- File repositories based using relative paths
  - ie file /path/to/file repository will be in <repo>/Cfg/path/to/file
- Repositories can contain base files or deltas
  - base files are complete instances
  - deltas are diffs
  - each can be class/image/hostname/attribute specific
- Most specific basefile is used
- More specific deltas are applied
- Cat file support implemented, diff support can be

Generators - SSHBase

- Keeps public and private for all hosts
- Builds new keys for any new host that connects
- Pre-generates known_hosts file when new keys are added
- Adds a host-specific line for localhost when known_hosts fetched
- Reads external public key definitions from *.static
- Pre-generated ssh_known_host file can be exported to other repositories as <system>.static

Generators - Pkgmgr/Servicemgr

- State defined in a single XML file
- Definitions are scoped based on metadata classes

Bcfg2 Server - Logging

- Normal run:
  - Generated config for ccsto1 in 0.2335436 seconds
  - Client 140.221.67.51 reported state clean
- Failed run:
  - Client 140.221.67.51 reported state dirty
- Configuration generation failures
  - Failed to FetchRecord Service:mpd
  - Failed to FetchRecord ConfigFile:/etc/passwd
- Generator Failure
  - Unexpected failure in BindStructure
    - Will include traceback
    - Mail it to me

Bcfg2 Server – Repository Tasks

- Updating a package version
  - place in package repository
  - update entry in <repo>/Pkgmgr/<image>.xml
  - Automatically done for debian
  - Rick has a script in testing for redhat
- Changing a configuration file
  - Global change
    - put new file in <repo>/Cfg/path/to/file/file
  - Class specific change
    - put new file in <repo>/Cfg/path/to/file/file.C<priority>_class_name
  - Host specific change
    - put new file in <repo>/Cfg/path/to/file/file.H_<hostname

Bcfg2 Server – Repository Tasks (cont)

- Creating a new bundle
  - Put configuration elements in repository
    - Config files in Cfg/
    - Package definitions in Pkgmgr/
    - Service definitions in etc/services.xml
  - Write new bundle
  - Associate bundle with client in <repo>/etc/metadata.xml
  - Run ValidateBcfg2Repo
  - Reconfigure client

Bcfg2 Server – XML Validation

- Bcfg2 includes schema definitions for all xml files used in the repository
- ValidateBcfg2Repo will validate each file in the repository against the schemas
- This command can safely be run at any time
- This will catch typos
- Should be run any time xml files are modified by hand