Argonne NATIONAL LABORATORY

Configuration Management Using Bcfg2

Narayan Desai

desai@mcs.anl.gov

FNAL

February 7th, 2008

THE UNIVERSITY OF CHICAGO

Office of Science U.S. DEPARTMENT OF ENERGY

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

Overview

- Background
- Motivating Problems and Goals
- Bcfg2 Architecture
  - Major components
- Experiences

Software-Related System Administration Categories

- Software-specific domain knowledge
  - Linux Kernel
  - Apache
  - Globus
  - ...
- Putting this software together
  - Building coherent services
  - Correctness
  - Robustness
  - Security
  - Fault Tolerance/Fail over

Configuration

- Union of software factors that influence the behavior and performance of computer systems

What do we do with Configuration?

- Creation (Deploy new things)
- Modification (Update existing things)
- Analysis (Acclimation and Troubleshooting)
- Validation (Audits)

So where's the problem?

- Configuration is large
  - Distributed across a large number of devices
- Disorganized
- Inconsistent

- Scaling factors
  - Client count
  - Configuration diversity
  - Number of administrators

Unwieldy and hard to work with

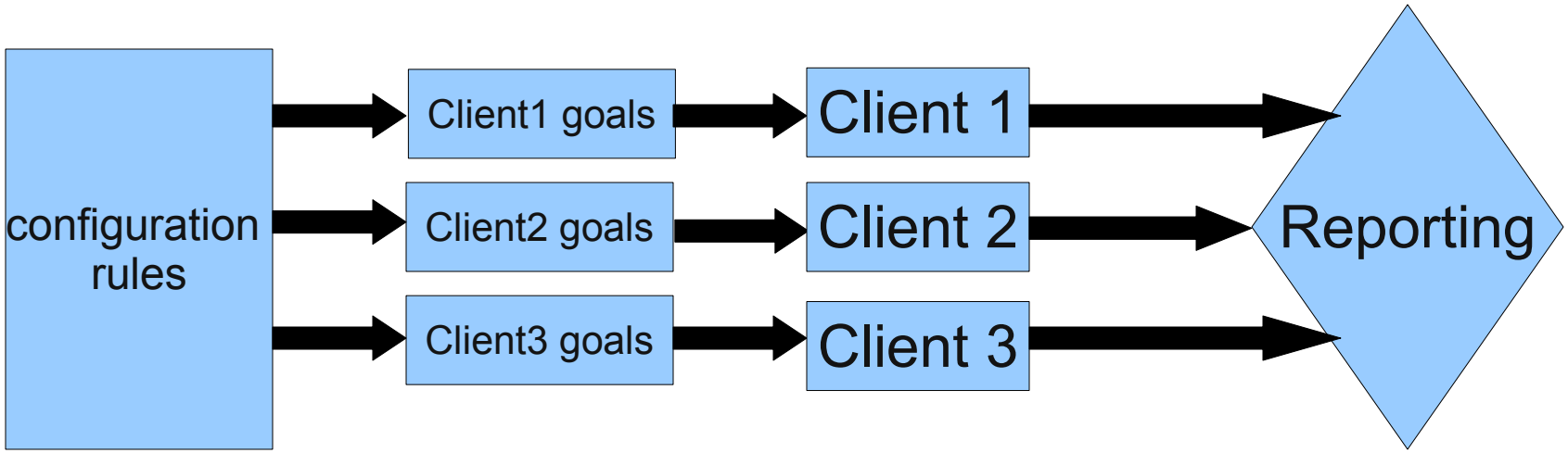
What is needed?

- A *useful* representation of configuration goals
  - Accurate
  - Verified
  - Compact
  - Centralized
- And reconciliation with reality
  - Flexible
  - Intuitive

The Bcfg2 Architecture

- Built on verification
- Configuration is discoverable
- Configuration rules used to build per-client configuration goals
- Clients compare goals with current state and reconcile
- Per-client statistics used to construct full system reports

The Bcfg2 Architecture (cont)



Configuration Goals

- Built of collections of entries
- Entries correspond to familiar types
  - ConfigFile, Package, Service, etc
  - Verifiable
  - Idempotent
  - Installable
- Overall goals built of typed collections of entries
  - Dependent and Independent
- Comprehensive
- Complete
- Literal
- Typical client goals range from 200-2500 entries

Bcfg2 Client Functions

- Executes local state probes
- Retrieves goals from the server
- Compares current local state to goals
- Determines which goals to attempt
- Attempts to install those goals
- Report on local state back to the server

Client Execution Triggers

- Init script (boot time)
- Cron jobs (hourly or daily)
- Job prologue/epilogue (in HPC environments)
- Agent mode

Client Verification Process

- 2 phase
  - Per-entry verification of goals
  - Heuristic discovery of unspecified configuration
- Describes goal mismatches
- Discovers (in part) unspecified goals

Client Statistics Upload

- Entry counts (total/good/bad)
- Detailed entry information
  - Bad Entries
  - Modified Entries
  - Extra Entries
- Detailed Activity information
- Performance data

Reconciliation

- Goals don't always match reality
  - Goals are incomplete or wrong
  - Client configurations are wrong
- Without a convention, it is impossible to know
  - Administrators need to make decisions
- Tools need to expect this, since reality doesn't always live up to expectations

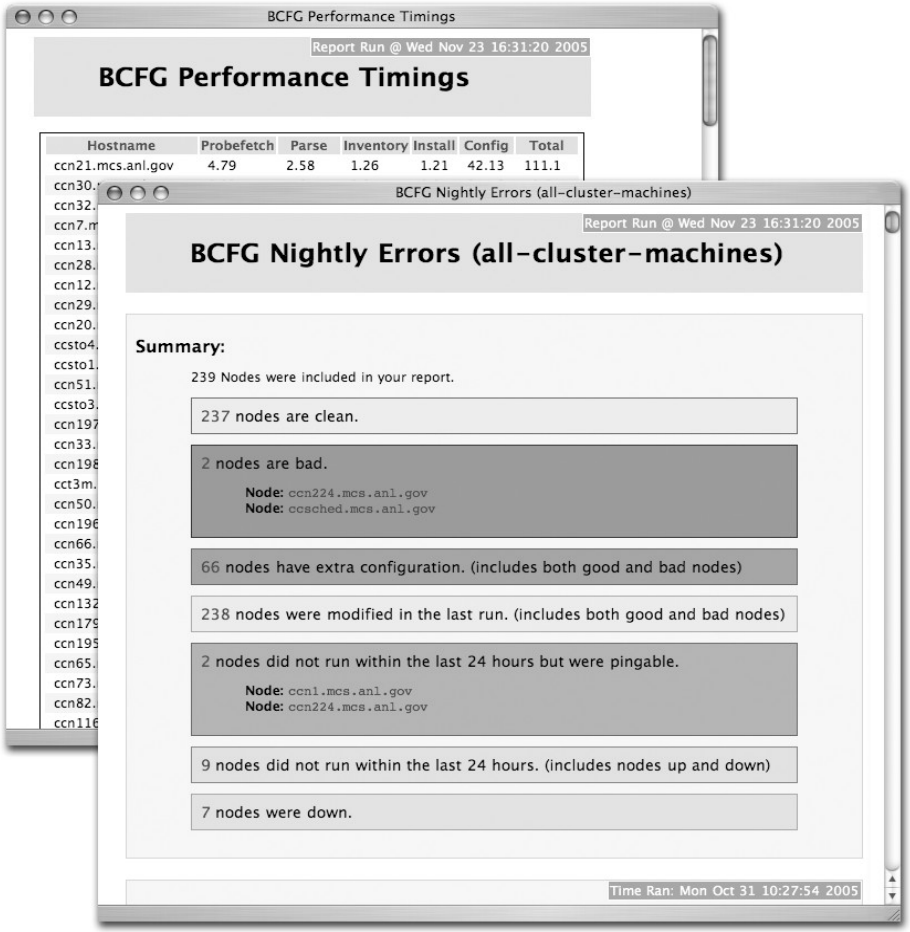
Action Determination

- Normal mode (all pending changes made)
- Dry run mode (no changes made)
- Interactive mode (user prompted for each change)
- Extra entry removal
- Continuity mode (*)
- Centralized decision mode (*)

Reporting System

- A bird's eye view of goal conformance
- Describe reconciliation sites in detail
  - Grouped by pathology
- Multiple output formats (Mail/RSS/Web)
- Provides a good set of metrics for conformance

Reports System Screen shots



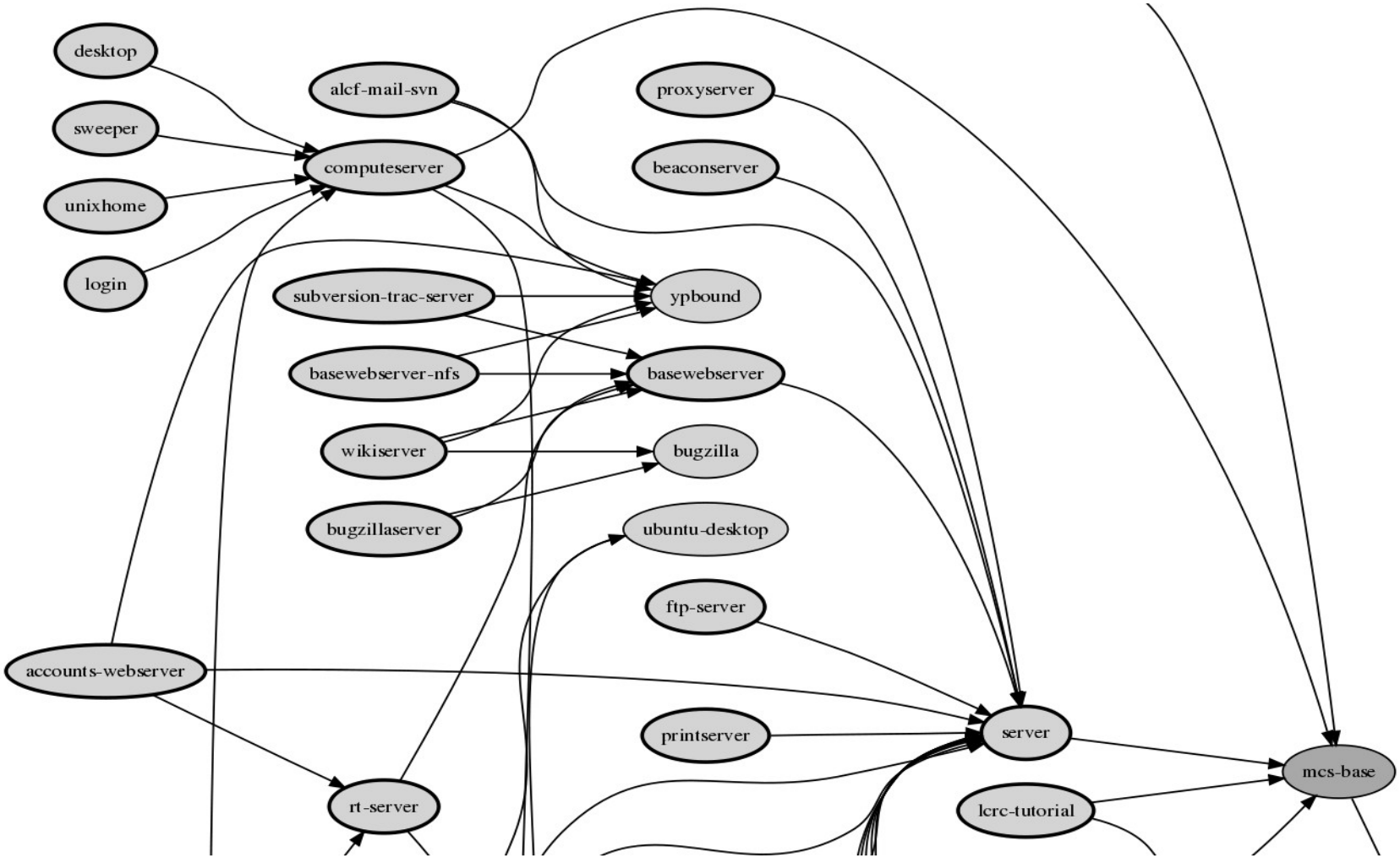
The Bcfg2 Server

- Serves data using XML-RPC over HTTPS
- Two main tasks
  - Rendering configuration rules into per-client goals
  - Routing client statistics to the reporting system
- Holds client metadata and configuration rules
- Uses a plugin mechanism for extensible rule implementation
- Goal construction is a 2 step process
  - Determine which entries should be included
  - Find the appropriate version of each entry

Metadata

- Describes the structure of configuration similarities and differences
  - Arbitrary specificity
  - Extension through inclusion
- Includes
  - Client Identifier (hostname or uuid)
  - Top-level group (profile)
  - List of component groups

Metadata Diagrams



Repository Structure

- Usually located in /var/lib/bcfg2
- Each plugin owns a subdirectory (by name)
- Plugins
  - Purpose-built
    - *Manage a given type of entry*
    - *Perform a particular task*
  - Specify a domain-specific rule format
  - Can contribute to either or both parts of the goal construction process

Plugins

- Cfg
  - Manages configuration files
- Pkgmgr
  - Manages package versions and metadata
- Rules
  - Manages service information
- SSHbase
  - Manages ssh keys and known_hosts files
- Tcheetah/TGenshi
  - Provides a templating interface to generate configuration files
- Base/Bundler/SGenshi
  - Specify which entries should be included in goals

Experiences

- Made us much more efficient
- Provides a useful analogue to documentation
  - Externally verified
  - Automatically updates
  - Close to the mental representation administrators already use
  - Brings new administrators up to speed faster
  - Shared mental model for the Configuration
  - Centralized source of accurate information
- Visibility into the configuration process
  - We know if it is working!

Bcfg2 Factoids

- Open Source licensed
- Included with most major linux distributions
- Widely deployed across all sectors of IT
  - Research/Education/Finance/Fortune500/Defense)
- Source code, documentation papers and much more available at:
  - http://www.bcfg2.org

Questions?

http://www.bcfg2.org