NARAYAN DESAI, RICK BRADSHAW, AND JOEY HAGEDORN

# system management methodologies with Bcfg2

Narayan Desai is a programmer and system administrator in the Mathematics and Computer Science Division of Argonne National Laboratory. His current research interests include system management and HPC system software issues.

*desai@mcs.anl.gov*

Rick Bradshaw is a system administrator in the Mathematics and Computer Science Division of Argonne National Laboratory. He helps to maintain HPC resources, experimental computing resources, and general UNIX infrastructure.

*bradshaw@mcs.anl.gov*

Joey Hagedorn is a student in Computer Science at the University of Illinois at Urbana-Champaign. When not studying, he spends time working on several programming projects.

*hagedorn@mcs.anl.gov*

As UNIX networks continue to grow in size and complexity, system management methods must evolve as well. In this article we discuss a typical deployment of Bcfg2 and describe its sophisticated configuration management capabilities. We present information about the environment at Argonne National Laboratory's Mathematics and Computer Science Division and the common tasks we in the systems group must perform, providing an overview of the tools used in our implementation of Bcfg2. We then discuss the procedural and qualitative impact that Bcfg2 has had on the way we manage our systems. Our aim is to describe what an environment with a comprehensive configuration management infrastructure looks like and to explain why one might want to invest the time needed to set it up.

## Background

Configuration management is an area of intense interest in the system administration community. Although this area has seen substantial effort over the past 15 years, a consensus on configuration management methods has not yet been reached. While a limited form of configuration management is widespread, relatively few organizations have adopted a comprehensive approach. Similarly, few practical accounts of tool adoption and results are available.

During the past year, the systems group in the Mathematics and Computer Science Division of Argonne National Laboratory redefined its methods for building, maintaining, and reconfiguring UNIX machines. The deployment process was quite involved, including substantial input from all 12 members of the systems group, and it altered the way that many tasks are accomplished. Two aspects of the process proved especially interesting: the social aspects of tool adoption and the technical aspects resulting from changes in systems management. In a companion paper, published at LISA this year (see Resources, below), we discussed the social issues, focusing on the nontechnical problems we faced. This article focuses on the technical issues, particularly the architecture we deployed, the changes we made in our management process, and the tools we chose.

## BCFG2

Bcfg2 provides a declarative interface to system configuration. It was designed and implemented in-house at Argonne but has matured to the point that external sites have begun using it. Its configuration specifications describe a literal configuration goal state for clients. In this architecture, the Bcfg2 client tool is responsible for determining what, if any, configuration operations must occur and then performing those operations. The client also uploads statistics and client configuration state information.

All complicated processing occurs on the Bcfg2 server. It uses an abstract, aspect-based classing system to represent patterns in system configuration. These abstract classes typically correspond to functional characteristics of the configuration. For example, a class may contain a description of the configuration needed to produce a Web server, Samba server, or an ntp client. Other, more abstract classes can also be created. These tend to be more site-specific—for example, "desktop" or "user-login." The configuration needed to fulfill these goals will vary greatly from site to site. This classing system allows administrators to employ a cookbook-style approach to building new configuration profiles, once various classes are built. Administrators can decide to include features on a profile-by-profile basis.

The other main function of the Bcfg2 server is to provide a reporting system that describes details about client execution. Several different types of statistics are collected during each client execution. Also recorded are overall client configuration state and lists of configuration entries that were either modified or remain incorrect. Timestamps are stored so that inactive clients can be detected. The reporting system has a major impact on how Bcfg2 can be used. It provides sufficient feedback for administrators that they can solely use Bcfg2 for deploying changes on all machines.

## ENVIRONMENT

In our division we have about 100 researchers, with large numbers of collaborators who frequently need access to our machines. During the summer, we have numerous temporary research aides and co-op students. The requirements of these collaborators and visitors strongly affects our network configuration:

- Many users access our resources from offsite. This access is vital for collaboration, but it means we cannot depend solely on a firewall for security.
- Our desktop environment is constantly in a state of flux because of constant staffing changes. Such changes are particularly pronounced at the beginning and end of the summer, when students arrive and leave. For this reason, the machine build process must be streamlined and easy.
- Our management system must continue to work and remain secure independently, in order to allow system administrators to focus on more pressing issues.

All said, our environment is fairly typical of most academic and research environments. The principal exception is that we have slightly more stringent security requirements than many sites, because of our government affiliation.

## Deployment

Deploying Bcfg2 took substantial time and effort, including work by nearly all members of our systems group. Adopting a new set of tools and methodologies was a challenge, both technically and socially.

Most of the social issues had technical issues at their root, many of which were tool-specific. Administrators were not comfortable that Bcfg2 would *do the right thing* when reconfiguring systems. What followed was a six-month process of identifying what the *right thing* was and ensuring that Bcfg2 did it.

The other main task during deployment was the construction of a configuration specification that Bcfg2 could use to generate proper client specifications for our network. This task moved in jumps; some configuration aspects were quickly transcribed, while other, more subtle ones took much longer to get right.

### TOOL REQUIREMENTS

During our group discussions about system management strategy, several key issues emerged. Administrator confidence in Bcfg2 was the most important issue. Administrators need to trust a tool, in terms of both generating proper configurations and performing correct reconfiguration operations on the client. Without such trust, administrators won't use a tool for anything important.

To address this issue, we chose to make Bcfg2's behavior as observable as possible. Specifically, we implemented a comprehensive dry-run mode in the Bcfg2 client. This allowed our administrators to experiment with the tool without undue pressure; once they were comfortable that the pending changes were reasonable, they were willing to commit to adopting the tool. Similarly, high levels of debug output were added, documenting all decisions the client makes while determining what operations should be performed. The availability of this information fostered confidence in the client, because the administrator could watch the tool in operation and understand why it performed the way it did.

Another issue we addressed was management of client configurations. Bcfg2 had to be able to handle all aspects of client configuration and reconfiguration without manual intervention. It also needed to be robust in the face of manual client reconfiguration. (Who hasn't made several changes debugging a problem, only to cause a new problem later?) To this end, we designed Bcfg2's reporting system so that it can describe all aspects of Bcfg2's actions and can provide salient information about client configuration. This reporting system gives administrators the ability to consider client configurations in a class-based way, using the Bcfg2's configuration specification for all nodes. The reporting system then reports all deviations from that specification. We augmented the Bcfg2 client to detect configuration elements on the system that weren't specified in its configuration. These *extra* configuration elements are also reported back to the server.

The third issue we considered was convenience. We streamlined several common tasks, including the machine build process, and we made the configuration profile selectable from the boot disk menu. These small measures typically reduced the interactive time substantially. Most important, they were vital in convincing administrators that it was worthwhile to spend time learning how Bcfg2 works.

## CONFIGURATION SPECIFICATION

In parallel with our technical discussions, we devised a Bcfg2 configuration specification that describes our network. We started with the desktop system. This is, by far, the largest basic type of system in our division and thus has the most uniform configuration. Building a specification for our desktop systems consisted of identifying services and software on each machine, recording these in the configuration specification, and then testing this configuration in stages.

Next we turned to the servers. This process took much longer for a number of reasons. Server configurations varied much more than desktop systems. Desktops had been managed in an organized way, while servers were managed in an ad hoc fashion. Servers also had much more complicated service definitions. Many of these systems had specific owners who had performed manual modifications over time. Most important, these machines provided a large number of user-visible services and represented the infrastructure on which the entire division functioned. Servers were one of the primary drivers for many of our technical discussions. Once these issues were resolved, however, the specification process was quite similar to that of the desktop process.

The basic procedure for incorporating new classes into the specification comprises writing a description of all the interrelated configuration that provides a service, and collecting relevant configuration, such as configuration file contents and permissions. This process can be expedited by using the Bcfg2 client in dry-run mode. Once all configuration information is integrated, the Bcfg2 client can be used to detect whether any reconfiguration on the system is needed.

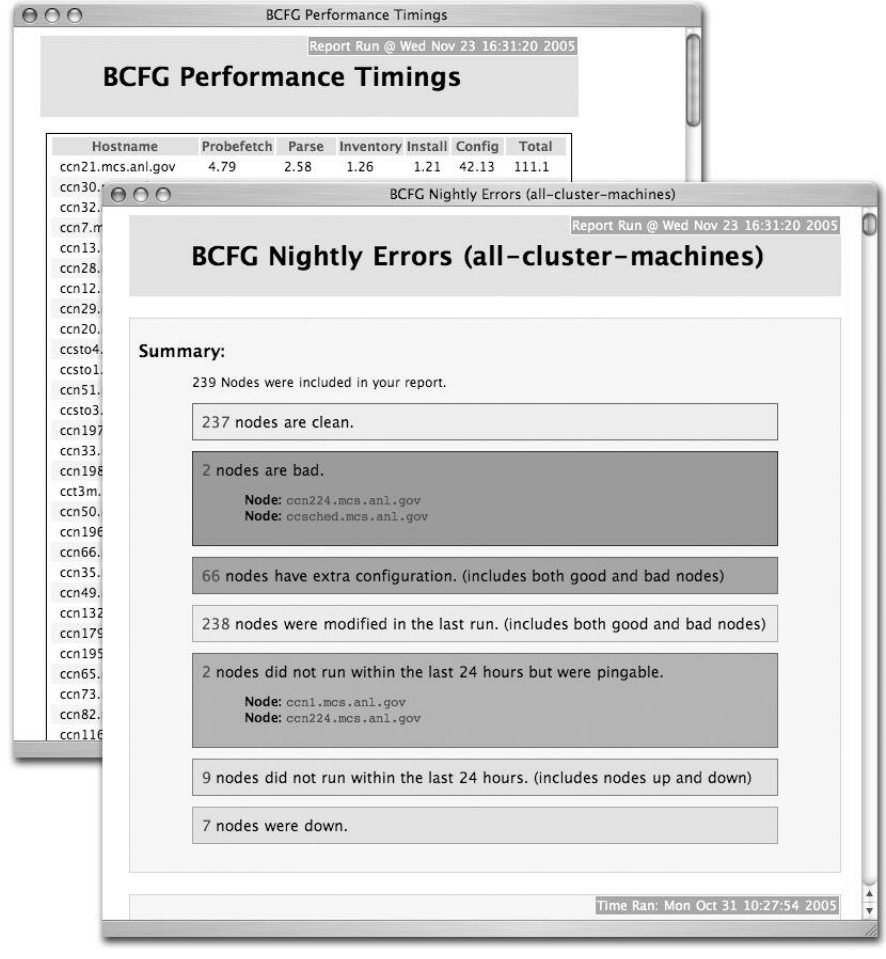
Another Bcfg2 feature that allowed smooth migration was the ability to incorporate information about unmanaged hosts. Bcfg2 stores statistics about all aspects of a client configuration that do not match the configuration specification. If the Bcfg2 client is run on a machine, statistics describing its configuration deviations are uploaded to the server and included in system reports, even if no changes occurred. These reports can be used to find areas where the configuration specification is incomplete.

## REPORT-BASED CONFIGURATION MONITORING

Bcfg2 configuration reports provide an impedance-matching mechanism between the configuration specification and the actual configuration state of all clients. Discrepancies between the two cause a variety of latent management problems. Most importantly, if a service-providing machine has a running state that does not match the configuration specification, it cannot be rebuilt or duplicated. Its state also cannot be reasoned about by Bcfg2.

This system also allowed us to administrate our servers in a more interactive fashion. We run the Bcfg2 client on each server in dry-run mode. The client inventories the local machine state, determines what operations should happen, and uploads this information to the server. Administrators view the resulting reports daily, and can supervise the execution of the Bcfg2 client on critical servers when it is needed.

The reporting system provides a bird's-eye view of the overall configuration state of all clients. This view exposes configuration specification problems, allowing their repair before they cause problems. We now feel confident in our understanding of all machines that are properly described in the con-




figuration specification, show a clean configuration state, and have no extra configuration detected. *All* of our administrators now have a deep understanding of all of our machines' configurations, or clear indicators for situations where they do not.

## Impact on Administration

Rebuilding our management infrastructure had a dramatic effect on our daily lives. Many everyday procedures were simplified, and powerful new mechanisms for automation became available. Moreover, the system administration process was changed in a qualitative way that transcends particular tasks.

### PROCEDURAL CHANGES

Our new management infrastructure enabled several categories of procedural changes. Some tasks disappeared altogether. Many more changed in basic character. The fundamental unit of automation in our old environment was the venerable shell script. Scripts are useful for a variety of purposes but are lacking in one major way: scripting multi-machine processes is fault-prone, and error handling is difficult. Moreover, these scripts generally have a lot of local topology information hardcoded inline. This approach prevents them from being portable across sites.

Bcfg2's model—specifically the existence of a central, declarative configuration specification that can be programmatically modified—makes simple scripts considerably more powerful. Administrative applications need only calculate final results and can leave all error handling to the Bcfg2 client. These applications can be adapted to the Bcfg2 server plug-in interface, which is called during client configuration generation. This plug-in interface has access to add or alter configuration elements on any managed client. We have adapted several administrative applications to use this interface:

- Controlling user access to batch-scheduled nodes
- Managing SSH keys and creating a correct ssh_known_hosts file
- Balancing virtual hosts across several Web servers

In each of these cases, the plug-in logic needed encapsulated a near-literal transcription of policies or configuration generation rules. The previous implementations of each were uniformly complicated and non-portable. In all cases, conversion to this API reduced the code volume by 75% or more. This reduction occurred because much of the heavy lifting is now handled by the Bcfg2 client.

Good experiences with basic automation have led us to attempt much more complicated workflows. For example, we are adapting our IP/host management application to directly feed Bcfg2 with DNS and DHCP configurations. Once this system is integrated, it will be easy to correlate with any effects from other plug-ins. We are confident that such efforts will automate many of the remaining daily reconfiguration operations requested by our users.

## QUALITATIVE CHANGES

More important than the procedural changes, several qualitative changes affected the administration process overall. These transcended the performance of particular tasks and changed the character of system administration in our division.

The most striking change was that configuration management tasks became a proactive part of the environment. At regular intervals, all clients check against the central specification for configuration changes, and may (depending on their settings) apply configuration changes. In any case, statistics describing their current state are uploaded. Bcfg2 serves as a steady-state deployment engine that can detect and correct configuration inconsistencies.

This change in model allows administrators to focus on changing the configuration specification and inspecting reports describing the results without having to worry about deployment details. Having a comprehensive deployment engine also greatly reduces the cost of individual reconfiguration operations. The availability of cheap reconfiguration operations expands the range of options open to administrators. We found that our environment now has some daily churn of configuration changes. Automated scripts can make changes based on external stimulus, such as the release of software updates, and deploy appropriate configuration changes across clients.

These changes resulted in an environment where we can make reasoned judgments about how we wanted changes to propagate to our environment. The time freed up by deployment automation allowed us to design a system with a more measured approach to change management and test-

ing. Our changes now migrate to desktop machines first; on these systems we value security more than anything, because of the large number of clients. In contrast, we are willing to wait for administrators to run the Bcfg2 client on servers, so that they can ensure that everything is still running properly. This approach costs more than the one employed for desktop machines, but we think that it is worthwhile for server machines. Most important, we were able to make a local determination about how we wanted changes to propagate and implement that exact system. This is a policy matter that will greatly vary from site to site, and one size will never fit all.

Overall, these changes resulted in a much more deliberate system management process in our environment. Administrators were freed from repetitive tasks, and we were able to exploit the new tools to make the decisions that only experts can make effectively.

## Conclusions

Configuration management needs to be globally adopted. It can dramatically reduce the time spent performing repetitive tasks. Initial efficiency gains can be fed back into improving configuration management capabilities. The net effect is a large time savings for system administrators—time that can always be used to improve services for users.

Bcfg2 provides a good framework for automating complex workflows. This infrastructure offers an interface with simple and reliable reach throughout your environment. This enables easy automation at a scale not previously possible. Complex, network-wide policies can be implemented from a central location. Moreover, the central configuration specification and statistics can be mined for a variety of information.

This redesign of our infrastructure took a substantial amount of time and effort. Nevertheless, we recommend that others attempt the same. The long-term benefits far overshadow any short-term costs.

### RESOURCES

The Bcfg2 Web site: http://www.mcs.anl.gov/cobalt/bcfg2 (information about Bcfg2, including a manual, mailing list archives and sources).

Narayan Desai et al., "A Case Study in Configuration Management Tool Deployment," *Proceedings of the Nineteenth System Administration Conference (LISA '05)* (Berkeley, CA: USENIX Association, 2005). This paper describes the social issues encountered by a large system administrator group during the adoption of new tools and management procedures. The paper provides the social counterpoint to the technical account of this process provided here.

Paul Anderson, *Configuration Management* (Berkeley, CA: USENIX Association, forthcoming). This book provides a primer in configuration management, as both a practice and a research area.

The lssconf mailing list: http://homepages.informatics.ed.ac.uk/group/lssconf/. This mailing list provides vigorous discussion of configuration management tools and the techniques they employ. Many configuration management tool developers subscribe to this list.

**ACKNOWLEDGMENTS**

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38.