UnixWorld Online: Technical Feature Article: No. 001

Riding the Distributed Management Trail

Years of Unix system administration have given our guest author a unique perspective on problems with distributed tools

by Dinah McNutt

After working as a system administrator for nine years, I jumped at the opportunity to become a consultant for a small company. I had been wrestling with the problem of never being in the mainstream of my employer's business, big oil. Your career is limited at an oil company if your expertise is not in oil exploration or production, and now a company wanted me for my expertise and to help them deploy a new technology.

At Tivoli Systems, I was project manager for a new release of the Tivoli Management Environment and then spent the next year and a half helping Tivoli customers deploy the system. During this experience, I learned what living with your mistakes really means!

This article is not about Tivoli's product nor do the problems and solutions I describe necessarily apply to the latest release of the Tivoli software. As an advocate of tools that make system administrators more productive, my goal is to share my experiences deploying a distributed systems management tool. I will describe some of the problems we encountered, solutions to those problems, and my personal opinions.

Hopefully, the ideas described here will help developers of such tools produce better tools and let potential customers know what questions to ask when that software salesperson comes to call.

Background

First, we need to define distributed systems management. This ambiguous term is applied to everything from `rdist` on to vendor software. For our purposes, distributed systems management describes a collection of programs running on a group of machines, as well as the files that these programs modify and the database(s) associated with the programs.

Let's look at an example of how a distributed systems management tool might evolve using `rdist` as an example because many people are familiar with it. This program is used to distribute files from one system to another, but traditionally involves only two machines at any one time. You can do parallel distributions with some versions of `rdist` and you can use it to send files from host A to host B to host C. However, when you examine the details of the distribution, files are basically just being transferred from one host to another.

Now imagine extending `rdist` so that the remote systems (hosts B or C) request files from host A. You need an easy way to keep track of which files have been updated, which are available from host A, and which need to be updated. In addition, you can obtain the files from either host A or some other host D, depending which host has the lowest system load.

I have taken a simple example of distributing files from one host to another and made it more complex by adding state information and delegating the task from the file server to the client system. Problem solving when a failure occurs is now more complicated because you must determine from which host (A or D) the file transfer was occurring. This scenario can be made more complicated by several orders of magnitude by adding hundreds of clients, at which point you will be able to appreciate the complexity of a truly distributed system management application.

Functionality Issues

Do you want a Swiss Army knife or a screwdriver? The Unix approach to tools has been one tool, one job. First, you need to identify the problem you are trying to solve. If you want an integrated solution, then you probably want a Swiss Army knife. If you just want to distribute files, then the screwdriver is probably best.

An infinitely flexible piece of software can be too complicated to use. Emacs is a very flexible text editor, but there is definitely a learning curve that must be overcome before you can take advantage of some of its features. Alternatively, a point-and-click style editor may be easy to use, but you may not be able to read USENET news or browse the World Wide Web from within it like you can with Emacs.

My experience has been that not all sites perform system administration tasks the same way. For instance, one site may locate home directories in `/home/`*username*, while another site may use `/home/`*hostname*`/`*username*. These differences may be historical (for instance, it's the way the system administrator who installed the system set it up) or for valid business reasons.

If a vendor makes assumptions about how a particular task is performed, then customers may have to adapt to the vendor's philosophies or find another tool. An alternative is to have a tool that is customizable so it can by adapted to your environment. However, this means you have to learn how to customize the software before you can use it. It can be frustrating to buy software and have it sit on the shelf because you don't have time to configure it properly.

Sun Microsystems' Admintool is a good example of a tool that is not flexible, yet is fairly simple to use. (I use Admintool as an example because it is available on all Solaris 2 systems. Many of my comments here apply to similar tools found on other systems.) If you want to add a user on a single system, Admintool is a good way to do so quickly without having to read manuals. However, if you want to do something a little bit different from the way the Sun engineers envisioned, you're out of luck.

In addition, you can only perform the tasks using the graphical user interface (GUI). However, experienced system administrators typically prefer to type commands into the shell rather than wait for window systems. This is particularly true if you are creating many users at once. One of my first jobs as a consultant was to write a batch shell script using the command line interface (CLI) that would create many users from information stored in an ASCII file. No one wants to create 200 users using a GUI because it would take too long.

My other primary complaint about Admintool is that it's not distributed: Host-based tasks only work on a single system. I am convinced that any useful administration product needs to be distributed. Managing a single system is not very hard or challenging, but managing hundreds or thousands of systems is.

One mistake we made at Tivoli was to provide only a subset of the functionality that was available from the GUI in the CLI commands. Correcting this deficiency was a major goal for subsequent releases of the Tivoli software.

Political Issues

Most competent system administrators want tools to help them manage systems. They don't want tools that infringe on the way they manage their systems. I know they want a chance to be involved in the decision-making process when it comes to selecting tools. I have seen sites where department or company management would make a decision about system management software and then expect the system administrator to implement the software. The system administrators are the people who best understand how the systems are being managed, and they need to be "in the loop" if for no other reason than to explain how much time and effort will be required to implement the software.

One of the first lessons I learned while at Tivoli was that sometimes companies will purchase distributed system management software with the expectation that it is a replacement for a system administrator. My response is that no software will help you when your system won't boot. You still need someone available (even if it is on an on-call basis) to troubleshoot problems.

In addition, you may need to attend product training classes, depending on how comprehensive the software you purchased is. We found that training classes were needed to address deployment issues including planning, customization requirements, procedural changes, and so forth.

When new versions of the software come out, you have to carefully plan how to transition to the new software. If you have 1,000 systems, it would be ideal to migrate a few machines at a time. The software vendors should make the transition as easy as possible for their customers.

Installation Issues

Even if you are installing your own home-grown system management software, you must address the following issues:

- How do I distribute the software? Will remote root access be required? Can I use NFS, and if so, are the appropriate file systems mounted? Is there enough disk space on the file system(s) I have chosen?
- How do I execute the software? At system boot time? As a cron job? Remotely from a central server?
- Does the software need to be customized as I distribute it with information like the host's IP address and operating system version?
- What happens if a machine goes down during the installation?

Remember, these problems are not necessarily specific to a particular application. Early on at Tivoli, we wrote a script to check all root remote shell (`rsh`) accesses, NFS mounts, and so forth. We found it was faster to do the checks up front than to fix problems as they occurred because many of the systems would fail one or more of the checks performed by the script.

One of the perceptions our customers sometimes had was that the software was difficult to install. However, the only problems that occurred were incorrect NFS mounts. Customers often did not understand that the problems would have occurred regardless of the software being installed. This was most often true at sites that were new to Unix. We found ourselves educating people on host name space management just to get the software installed.

As a result, everyone on our customer support staff was required to have system administration knowledge because it was not enough to know how to use the actual product. We became creative at remote troubleshooting over the phone when we did not have either e-mail or Telnet access to a customer's systems.

At one point, one of our customers at a large communications company was trying to update her "hosts" (`/etc/hosts`) file. She used Telnet to connect to the target machine and add the needed entry. Each time she used `vi` to view the file, the entry was not there, so she kept adding it. I suggested using `cat` or `more` to view the file, and sure enough--there were 10 identical entries! She had come from a mainframe environment and did not know about terminal emulation and how to set the terminal type correctly. Resolving this problem took more than 30 minutes on the phone. We had a similar problem with another customer who used Backspace instead of the Delete key and ended up with unprintable control characters in his `/etc/hosts` file. Try troubleshooting that problem when you have no way to access the machine remotely.

Sometimes a novice user can make you think about things differently. I had a customer at a Federal organization (whose name I cannot mention or I would have to shoot you ;-)) who was describing a problem over the phone by starting with "I clocked the window..." At first we thought she meant she used a stopwatch to time how long the operation took, but she meant she hit the Apply button and made the OpenWindows busy cursor appear (which is a clock.) This terminology has made its way into the Tivoli culture and now clocking a window is an actual term used internally at Tivoli.

Needing root `rsh` access to distribute files and remotely issue commands during installation was a great security concern at many sites (especially customers on "Wall Street") although the access was only required for a few minutes during the install process. The solution to this problem was to offer three installation methods:

- Provide root remote shell access.
- Require the root password of the remote machine for remote command execution.
- Bootstrap the remote system using local media devices to load the "core" files (sites that were this security conscious did not run NFS).

Handling system crashes gracefully was a more difficult problem. The Tivoli software uses a distributed database for which each host stores information pertinent to itself. This improves performance because you don't need to make queries across the network to a database server. The downside is that without sophisticated transaction processing, if one host goes down, you could run into problems with database consistency across all hosts.

During installation, there is a critical period where the software is installed, but the database is not fully configured with host-specific information. If a network connection is lost or the host goes down, you must be able to detect the failure and recover. We handled this by taking snapshots of the databases only on the systems involved in the installation and restoring those databases upon failure. We also developed an "fsck"-like program for the database to detect references for database objects that no longer existed.

We eventually recommended that customers back up the database daily. Because it was a distributed database, we developed a special `dbtar` program to back up data to a single system where it could then be backed up using standard Unix backup utilities.

Implementation Issues

The biggest implementation issue was how much to customize the product. It's a Swiss Army knife solution, but that doesn't mean you have to use (or need) all the features. I strongly believe that a system administration product is more useful if it can be adapted to your site's policies and procedures. Doing this can be time consuming, however.

It may be more cost effective for some sites to adapt their policies to match those provided by off-the-shelf products. Many sites don't have written policies, and one of the first things you need to do to properly deploy a product that does user management is to define what those policies are: Where do home directories go? What mail aliases should be created? What user IDs are available? What does a login name look like?

Futures

There are more and more system administration tools on the market. Here are the things I look for when evaluating them:

- Is it distributed? If not, I stop considering it unless it solves a very specific problem and does it better than anything else. The single-host solution is not very interesting.
- Does it scale? Can I perform operations on thousands of hosts in a reasonable amount of time without a lot of data input?
- Does it meet your requirements? If not, is it extensible or customizable? Not being flexible doesn't mean it's not the right tool for your site.
- Does it compartmentalize tasks? I want the ability to group a set of tasks and allow a person (or persons) to perform those tasks, but not do other tasks. I want to define the grouping and have as many of them as I want.

I am not aware of any tools that meet all these criteria, but I have hopes that they are coming. If I never add another user to a host (or write a script to do it for me), I think I will be happy.

Edited by Becca Thomas / Online Editor / UnixWorld Online / beccat@wcmh.com