PETER BAER GALVIN

Pete's all things Sun: Solaris System Analysis 101

Peter Baer Galvin is the Chief Technologist for Corporate Technologies, a premier systems integrator and VAR (www.cptech.com). Before that, Peter was the systems manager for Brown University's Computer Science Department. He has written articles and columns for many publications and is coauthor of the *Operating Systems Concepts* and *Applied Operating Systems Concepts* textbooks. As a consultant and trainer, Peter teaches tutorials and gives talks on security and system administration worldwide.

pbg@cptech.com

AIRPLANE PILOTS MUST EXECUTE A pre-flight checklist before taking off. This list ensures that no steps are missed as the pilot prepares for flight. Over time, these checklists have been standardized and edited by many pilots and aircraft designers to the point that they are complete, logical, useful, and indispensable. Systems administrators are lacking such consensus documents, for the most part. Rather, some sysadmins have no useful checklists, doing all work ad hoc. Others have their own lists or work with groups that have documented methodologies that they follow. Frequently these lists have limited scope or assume site-specific details.

This month I'm unveiling one of my lists, "System Analysis 101." This list has been built over time via experience and trial-and-error. It is constantly expanding as new problems are encountered and solutions determined. Hopefully this list will grow into a consensus set of steps that both new and experienced sysadmins can use to save time and to avoid missing important aspects of a system that could be causing a problem. Input is welcome, and USENIX has set up a wiki to allow collaboration on this living, expanding checklist. Please visit it at http://wiki.sage.org/pbg and contribute.

This is the list that I used to diagnose "it's slow" or "it's not working right" kinds of problems. For more specific problems the list can be abbreviated, but carefully. Each of the entries comes from personal experience (or second-hand examples) in which that step wasn't taken and time was wasted. When a system or facility is not working right, time to resolution of the problem is of the essence, and counter-intuitively, this long list of steps can quickly lead to the diagnosis of the problem, the first step in getting it resolved.

This list may seem long and in some steps overly basic. When I approach a broken system I try to step through the list religiously, just as pilots execute their pre-flight checklist. Usually, lives are not at risk, but certainly time, and frequently money, gets lost when problems occur, and a systematic approach is the best way to resolving the issue. When I have been tempted into skipping steps, frequently I've regretted it as the steps skipped sometimes would have been useful in solving the problem.

Many of the steps here are general system-administration tasks that could be used on any system. Some of them are UNIX-specific, some are Solaris-specific, and some are Solaris 10–specific. If the problem is not occurring on a Solaris 10 system then other, equivalent (where possible) steps should be substituted.

Finally, the order of the steps need not be exactly as listed here, but the overall flow should be preserved, going from general to specific and from data gathering through making system changes.

Phase 0: Prepare for Problems

The time to learn how to use tools, to understand your facility architecture and performance, and to learn administration and debugging techniques is not when in the midst of a production problem. Rather, these things need to be part of your DNA, ready for when they are needed. To prepare for the inevitable problem, consider doing the following:

- Join helpful organizations, especially local user groups, to both learn and build your network.
- Take classes and tutorials (from organizations such as USENIX, of course).
- Read books on system administration and practice what they preach. My personal favorites include *UNIX System Administration Handbook* (latest version) by Nemeth et al., *The TCP/IP Guide* by Kozierok, *Solaris Performance and Tools* by McDougall et al., and *Solaris Internals,* 2nd ed., by McDougall and Mauro.
- Execute phase 4 when the facility is running normally (assuming there is such a thing) or at least at steady state. One of the easiest ways to determine problem details is to compare the state of the broken facility against the state when it was working better.
- Practice all of the other steps in this document to ensure that tools and documents are in place for when they are needed, that they work in your environment, and that you understand how to use them, what they do, and what their results mean.

Phase 1: Capture the Problem Definition as Succinctly as Possible

Capturing the problem definition as succinctly as possible helps keep focus on the problem and helps communicate the problem as needed. It also helps avoid the "death spiral," in which while exploring one problem other (potential) problems, or red herrings, are found.

Areas to capture include:

- When did the problem start?
- What invokes the problem?
- What avoids the problem?
- What is the problem, exactly?
- What changes were made before the problem started? (This is usually the key question!)
- What debugging/analyzing/testing changes have been done since the problem started? (Answering this can avoid wasting time and following those red herrings.)
- What existing diagnostics are available? These can include performance trends, performance monitoring tools, errors in log files, core dumps, angry users, and so on.

Phase 2: Phone a Friend

The timing of this phase is variable and generally occurs throughout the other phases. If more than one person is working on the problem, this phase can be delegated—it can be time-consuming.

- Place service calls on the problem components, including, for example, the application that is having a problem, the operating system software, and any hardware it is running on. Perform this early in the project if support contracts are in place and so the service calls are at no cost.
- Get in touch with whoever sold you the facility that is having the problem (for example, a reseller).
- Get in touch with anyone (if other than you) who was active in the implementation of the facility.
- Search for similar problems that other people have written about (even better, solved). General, technical, and product search engines are useful. For example, for a Solaris 10 on Sun hardware problem you could search at http://www.opensolaris.org/os/discussions/ and at http://sunsolve.sun.com/show.do?target=home. Looking for and exploring resources is a great thing to have done before the problem occurred, in all your "spare time."
- Get in touch with experienced and helpful sysadmins to whom you have been helpful in the past. (This is one of the many reasons to be helpful to other sysadmins.)

Phase 3: Determine Available Testing/Resolution Resources

- Are there any similar development, Q/A, or business continuance systems available? (Watch out that "similar" might be different enough that the problem cannot be reproduced there.)
- Can the problem be reproduced? If so, capture the steps to re-creation.
- Can we make changes in production? If so, capture the details (e.g., downtime windows, change limits such as validated system and production lockdown times and low-impact times).
- If the problem only occurs under load, do we have the ability to test under load and to generate a sufficient load to cause the problem? The worst problems are those that only occur under load and when the load cannot be generated artificially.
- What is the change deployment method and cycle in case changes need to be made to resolve the problem?
- Is testing in production possible?
- Is having an impact on performance in production acceptable?
- Is the use of unsupported tools in the production environment allowable?

Phase 4: Capture the State of the Problem Environment

For each component in the problem environment (certainly computers, but this could also extend to storage, networking, and security components), do the following:

- Capture the state and configuration details with the "best" tools available. For Solaris, that means running explorer (if possible), which is now part of the Services Tools Bundle and is available for free from http://www.sun.com/download/products.xml?id=47c7250a.
- Capture state with GUDS, which is a tool similar to explorer but more complete; unfortunately, apparently it is only available from Sun Support on an as-needed basis.

- If using Solaris 10, use dexplorer while the problem is occurring, if possible and allowed. dexplorer is part of the indispensable DTrace-Toolkit available from http://opensolaris.org/os/community/dtrace/dtracetoolkit/. Note that it is free and that the tools it uses are supported, but the toolkit and its scripts themselves are not supported.

Some additional things to capture if not already recorded by these actions include the following:

- What are the operating system, firmware, and hardware versions and patch levels?
- What are the pertinent application release and patch level levels?
- Are the versions of the applications supported on the versions of the hardware and operating systems? If not an upgrade cycle is probably going to be necessary to bring all of the components into compliance before support organizations will help resolve the problem.

Note that support organizations are likely to push for changes even in supported configurations before escalating a problem. For example, a very common scenario is that technical support will encourage or try to require installation of the latest patches, upgrading to the latest firmware, or even upgrading operating system or application versions. This eliminates variables for them, but it's also how they try to get you off the phone. Such work can take hours or days and frequently is not necessary—the problem frequently still exists after the work. Push back on support, depending on the level of effort and time their recommendations would take and your level of support. Ask support if they have any evidence that the problem will be solved by the changes recommended. If they have no proof, try to force them to continue working on the problem without first making their suggested changes. Note that politeness, firmness, and mention of how much you pay them for support is much more effective than poor behavior. If satisfaction is not received, then (politely) try to get to the next level of support management. Also, whoever sold you the facility has a vested interest in having you as a happy customer, so have them help increase the priority of the problem within the support organizations.

Phase 5: Track Down the Problem

Tracking the problem down is, of course, the meat of the project and the most difficult part. But with the preparation done as outlined here, many variables have been eliminated and plenty of information is now readily available to help diagnose the problem. This phase can vary immensely depending on the kind of problem being worked on, the scale of the problem, and all of the details determined in the previous phases. Some first steps for Solaris 10 systems are listed here, but other lists for other operating systems and devices should be compiled for your site (or be found to have already been published).

Generally, compare the results attained during the problem against the same information from when the system was healthy (see Phase 0).

- Scan through log files such as /var/adm/messages and via dmesg. Don't ignore anything odd—it could be the canary indicating the problem.
- Run `svcs -a` to check for services that have failed or are disabled.
- Check for full disks or changed mount information via `df` and `mount`.
- Run `ifconfig -a` and look for any errors; run `kstat` and grep through the output for the network interface names (such as e1000g0) to check network parameters such as duplex and speed.

Read through /etc/system and look for settings copied from other systems or left behind during an upgrade. Note that /etc/system should never be copied or left intact between operating system or application upgrades; such events should cause an audit of the file for entries to remove or update. Check the *Solaris Tunable Parameters Reference Manual* at http://docs.sun.com/app/docs/doc/817-0404. This document is updated for every Solaris release.

- Check /etc/projects for any resource management settings that could be affecting system or application performance.
- Check the `stat` commands and look for anomalies:
  - `iostat -x 10`—are there large response times?
  - `mpstat 10`—how many threads are in which states?
  - `vmstat 10`—do the thread counts and scan rate indicate memory shortage?
  - `vmstat -p 10`—look for systemwide memory operations.
  - `prstat`—are there any resource hogs?
  - `prstat -Lmp <pid>`—look at detailed state information about a specific process.
  - `pmap -x <pid>`—explore the memory map of a problem process.
  - DTraceToolKit and DTrace scripts—look at specific suspect aspects of the system.

Some general areas to consider, especially on Solaris:

- Are you running the most appropriate scheduler for each system in your environment?
- Are you using the best-fit page sizes?
- Is your I/O well balanced and spread across enough devices (disks, network ports, etc.)?
- Are you using the best CPU for the workload? (Are there few fast threads or many slower threads?)
- If processes are contending with each other, implement resource management (e.g., containers, project sets, and dynamic resource pools).
- Since rebooting the system can reset the state to a known starting point and remove variables, consider it as appropriate after current state information is captured.
- Finally, if the code is your own, did you use the best current compiler to generate the machine code?

Next Time

My list might be a bit controversial. Don't like it? Have a better one? Have a checklist for some other sysadmin activity? System administration is a difficult activity because not much of it can be learned in school or from books. It's a journeyman's trade. Help your fellow sysadmins and help make the world a better place by documenting your accomplishments and making them available. Contribute to the SAGE Web site, create your own blog, contribute to the wiki of this column at http://wiki.sage.org/pbg, join (and attend!) a user group, and get sharing!

Next month, in Solaris Systems Analysis 102, I'll dive deeper into Phase 5, showing DTrace and system command examples and how to tune areas such as the scheduler class and resource management. Until then, there should be plenty here to add to the sysadmin to-do list.