

Best Practices for Solving Problems in Non-z/OS Environments

How problem solving is different in mainframe and non-z/OS environments, and how you can accelerate troubleshooting convergence.

(this article is reprinted from "Enterprise Strategies" - www.esj.com - December 4, 2007)

by Dan Skwire

The IBM MVS (z/OS, OS/390, etc.) operating system environments were built as a continuation of IBM business products. They had to work reliably in business production environments, and problems had to be solved the first time they occurred (a primary MVS design principle), otherwise the business that was dependent on the system would suffer. Even a failure to attempt to recover from a failure was a defect that IBM expected to be fixed!

MVS's predecessor had an internal "system trace-table" comparable to airplane industry "black-box" flight data-recorders. This trace table is captured at the time of a system (or application) failure in a storage dump. Subsequent releases of MVS and associated hardware and firmware continued the "system trace table" and added other "trace tables." IBM continuously enhanced many of these MVS first-fault problem resolution features.

In today's production environment, non-z/OS production systems also perform significant work. Their operating systems don't have "trace tables" or many of the other significant first-fault tools that z/OS does because Windows, UNIX, and Linux, were originally built for non-production environments. They were excellent development operating environments, but many, due to cost and convenience (among many reasons), perform significant work in *your* data center.

In all fairness, the Unix, Linux, and Windows environments have made great strides in their service tools and in reliability in general. A few applications and isolated system programs have been known to include "trace tables". Even so, problem-solving is different in Unix, Linux, and Windows. They foster a very different mind-set, continuing from their original heritage, just like z/OS practitioners have inherited their particular mind-set, culture, and tools built around that culture.

You need to be sure your team has maximized its problem-solving results in these non-z/OS environments and can present a significant, cohesive package of data to the relevant Unix, Linux, and Windows support organizations. A well-constructed package of diagnostic data fosters respect in support organizations and accelerates successful problem resolution. Experience has shown that these support organizations are actually accustomed to a large number of call-ins with poor preparation, and a minimum of problem diagnostic data. . "System-down" ("severity 1") problems are reported, with hardly any accompanying documentation.

A well-prepared support event yields significant results. This article will help you ensure your team's diagnostic package is as complete as possible when they contact a support group. Support organizations often expect that the site perform a "do-over" (recreate the incident in a lab or in production), and activate or add diagnostic data-gathering tools (a "second-fault"

analysis tool) to capture the data one expects a z/OS-type system to automatically capture the first time. Often the support organization also attempts to duplicate the problem in their lab environments; someone has to get more problem documentation than was originally obtained. At times, a Unix or Linux site will be asked to "instrument the product" (modify the software to collect data), which, of course, could be a disruptive act.

To minimize the need for do-overs, you and your staff must use your mainframe-mindset thinking: "expect" a problem to occur. If you and your IT staff have expected problems to occur, you will need to ensure:

- Data cohesiveness (be sure all systems, storage arrays, and other devices have the same timestamp so logs are not confusing to support staff)
- System and application dumps ("crash" dumps and "core" dumps, respectively) are collected (the default is they are *not* collected in Unix, Linux, and Windows)
- Performance is monitored with data permanently recorded
- Error events are collected and monitored (for hardware and software)

In addition, you must:

- Use data-collection summary tools when available ("snap" in AIX, "explorer" in Solaris, "home-grown" scripts in Linux)
- Follow any operating systems and application product vendor (Oracle, IBM, etc.) recommendations for data collection

Let's look more closely at some of these issues.

Dumps

Linux environments were expected to be "hardware-vendor independent," so system crash-dump support was actually initially omitted from Linux, due to dump device-dependencies. Dump support has been added, in various means, via "dump to swap," "dump to disk," and "dump to a second system" configured to receive the dump ("netdump"). A fourth means, a hybrid kind of "standalone dump" (a built-in "mini-dump"), is being implemented or may be available by the time you read this. Expect dump configuration in Linux to be problematic at best. It has not been Linux's strength. Hopefully, this characteristic will change.

Performance Monitors

A vendor product, "Performance Sentry" by Demand Technologies, is available for Windows environments and can collect performance data records (think "RMF/SMF"). Also available free from Winternals (now part of Microsoft), is an online real-time monitor, perfmon, written by Mark E. Russinovich and David A. Solomon, who also wrote a Windows crash-dump formatter/analysis tool, and a book, *Microsoft Windows Internals* (Microsoft Press, 2005).

Data Collection

It would be helpful if comprehensive data collection were performed automatically at the time of an error, or immediately after it. Many vendor-supported Unix implementations do include a manually-initiated data collector, such as SUN Solaris' "explorer" report, and IBM AIX's "snap." These tools capture the output of "display" commands and they copy configuration

files to capture hardware and software installed, parameters, many key message logs, and similar indicators of the "state of the system."

The intent is to run these data collectors immediately after a significant problem occurs. Running the data collectors soon after system bootup could be very useful for establishing a baseline of data for that IPL; you can much more easily answer the problem-solver's key questions: "Has anything changed?" followed by "What changed?" when a problem occurs. Better yet, run the data collector regularly, perhaps once a day, to better establish a change history. The data collectors will also interrogate installed software "packages."

Although not a standard part of Linux, a basic Linux data collector is available from Mark Wilding and Dan Behman, authors of *Self-Service Linux: Mastering the Art of Problem Determination* (Prentice-Hall PTR, 2005). They recommend customizing the source for one's own site, to capture additional application and/or site-specific data. (The script can be found at <http://static.scribd.com/docs/i5db7wt38smfe.txt>.)

Although these data collectors may seem trivial (and they do provide capture of some very standard data), they perform significant data capture tasks, and that capture can be initiated with a one-line command. It would take a long time to manually run the commands and copy the files identified by these commands. No one wants to be responsible for typing a series of commands after an unnerving system crash. It's better to have a single command to capture it all right after a major crash than have to think and selectively run commands that may (or may not) provide needed data. Run the command automatically right after system bootup and you have the best non-crash data collected for your site's recent history on that server.

I recommend you verify that you have the "latest" level of the data collector, your staff knows who should run it and how to run it, and how to make its data available to your vendor support organizations.

Consider a vendor-supported black box such as AppSight from BMC (formerly independent Identify software). It has been successful in Windows and Java environments and is a true "data recorder" with telemetry for the application environment.

Summary

Mission-critical Unix, Linux, and Windows servers have different mind-sets, are managed by different tools, and require different methods to support them. The environments also instill different thinking of many technical people solving problems in these environments. Resolving system problems on these platforms in your data center takes pre-planning, but is possible, and you can greatly improve troubleshooting by applying the same successful thinking that you have used and employed on the z/OS side of your data center.

- - -

Dan Skwire is the founder and principal consultant at First Fault Problem Resolution Technologies (www.firstfaultproblemresolution.com) where he is responsible for business planning, performing consulting engagements, and research and development. You can reach Dan at dskwire@mindspring.com .