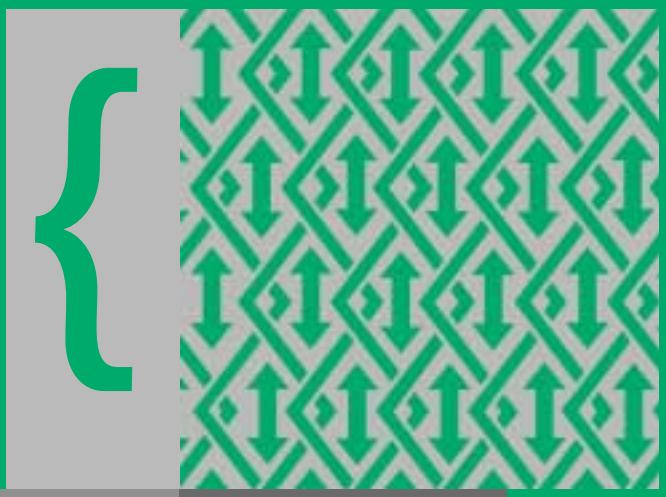
;login:

THE MAGAZINE OF USENIX & SAGE
October 2000 • volume 25 • number 6



inside:

SYSADMIN
Ode to Risk Management

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

ode to risk management

By Geoff Halprin

Geoff Halprin is a member of the SAGE Executive Committee, and the Principal Consultant at The SysAdmin Group, a systems administration consultancy.

<geoff@sysadmin.com.au>

It was Y2K, and a change was afoot. It was a simple change, to apply some Y2K patches to a production Solaris server. These patches had already been applied to a development server, so there was a reasonable confidence in their correctness. The change was scheduled to commence at 9:00 am on a Sunday.

The change plan, as approved, called for the mirrors to be split as a regression strategy. (All production systems are fully mirrored, using Solstice Disk Suite.) This was in addition to the cold backup that was to have been performed prior to the change.

The change commenced. The patches were applied. The machine was rebooted.

"ld.so: file not found". Oh dear. The time: 09:15.

I was called in to resolve this continuing problem at: 23:15.

The teleconference revealed that in attempting to resolve the above problem (which seemed to affect only a token ring driver), the administrator booted from CD-ROM and copied /usr/lib/ld.so from the CD-ROM to the /usr partition (A-side sub-mirror), then rebooted. Surprisingly, this did not resolve the problem.

Yes, I can hear a few of you gasping.

Once I got past the initial incredulity at the course of action, I asked whether they were aware of the consequences of this chosen course.

Did they know that the copy of /usr/lib/ld.so on the CD-ROM would almost certainly be a different binary from the patched Solaris revision, and therefore incompatible with the system? Did they know that /usr/lib/ld.so is somewhat critical to a system, and if it really couldn't be found, the entire system would not work (or boot past this error message)? Did they know how many copies of ld.so reside on a Solaris system, and where they live?

Oh yes, and that niggly one, did they know that mounting and touching one sub-mirror taints the entire mirror and effectively creates two randomly different disks which, in a round-robin read-balancing scheme (such as Disk Suite uses by default), this would cause (at best) a random panic from the kernel for reasons that would be next to impossible to trace?

Once I had teased the detail of their actions from them, I declared the change a wash, and asked them about the change plan's regression strategies. Turns out that, seeing as the change had gone so well on the test machine, they hadn't bothered to split the mirrors per the change plan, and so only had the full backup to revert to.

This was, it turns out, a blessing. The systems administrators involved had no idea of how to recover a system from split mirrors. The organisation had no "bare metal recovery procedure" for split mirrors. Some practical regression strategy that turned out to be!

So I asked them about their "full, cold" backup. It turns out, that this was an ADSM backup, performed with the system in multi-user mode (init level 3). Oh well, it's all we have, so here we go . . . . (It seemed a common ailment of that company to not understand the meaning of "cold," but that's another story.)

Now, I have nothing against ADSM as a product. I believe it to be an excellent backup solution. It is, however, inappropriate for "bare metal" recovery scenarios.

Our principal objective in making changes to a production environment is to ensure the success of those changes.

Why? Well, what is the general recovery scenario from a UFSDUMP of system partitions?

1. Boot from CD-ROM.
2. Restore to the A-side sub-mirrors (corrupted partitions only).
3. Mount the A-side sub-mirrors and remove references to mirror devices.
4. Boot the A-side (raw devices).
5. Bootstrap the mirrors from the A-side sub-mirrors.

This scenario should take around an hour or so. And from a third-party backup solution?

1. Boot from CD-ROM.
2. Suninstall a basic Solaris onto the B-side sub-mirrors.
3. Boot the B-side sub-mirrors (raw devices actually).
4. Load the ADSM client software.
5. Restore to the A-side sub-mirrors (devices).
6. Mount the A-side sub-mirrors and remove references to mirror devices.
7. Boot the A-side (raw devices).
8. Bootstrap the mirrors from the A-side sub-mirrors.

This scenario took in excess of 24 hours to complete.

So, at one level this change was a series of catastrophic mistakes and errors in judgment. But on another level, it all stems from a single fault – the lack of risk assessment and risk mitigation, skills that are essential to managing production computing environments.

Risk Management

Our principal objective in making changes to a production environment is to ensure the success of those changes – as planned, with no unanticipated side effects. They should be completed within the designated timeframe, and with no unforeseen impact on the user community.

The only way to do that is with careful planning. We are, of course, fighting against Murphy's Law. So it is important to plan for things to go wrong, things that we cannot predict. The only counter we have to an unknown threat is time – time to resolve unknown problems, and (planning for the worst case) time to admit defeat and back-out the entire change. So our change plan must include this contingency and regression time in calculating the change window required.

It is also vital that the system is never left in an indeterminate state (I hold this to be an axiom of change management), so this acts as a primary constraint on our risk-mitigation strategy selection. The state prior to the change is, by definition, a known state, with known functionality and known problems. This is why a cold backup forms a baseline for change regression, should things go awry.

Time

Assuming a stable baseline (backups), then our risks and costs are pretty much always time – time to back a change out, time to reschedule and reapply the change, lost productivity.

So our goal in risk management is to minimize these costs.

In evaluating our risk-mitigation options, we must always consider (a) how much (time) does this regression step cost to implement? (E.g., how long will the backups

take?) And (b) how much (time) does this regression plan cost to execute? (E.g., how long will the restore take?)

Splitting and recovering from split mirrors is far faster than filesystem restore from backup media (no slow tapes to deal with), and therefore a good strategy. It does, of course, come at increased risk – a disk failure during a change window would require a full restore from tape. If you want hard guarantees of return-to-service time, then a third mirror might be appropriate. Given the cost of disks and RAID units these days, doing cold backups to a mirrored disk array could be an effective alternative to tape in some environments!

Of course, copying file hierarchies sideways is an even cheaper risk-mitigation strategy (only the relevant files are copied), but not always a viable one.

Whatever regression strategy is chosen, be sure to reduce to rote procedure the steps to follow to back the change out. You do not want to have to work this out at 3:00 am under pressure of a failed change.

Further Reducing Risk

Have a good look at your change plan and identify all those steps that can be performed prior to the change window. Not only does this reduce the time required for the main change window itself, but it reduces pressure on the systems administrator, and so also reduces the risk inherent in the change.

Can steps be automated? Scripting steps ahead of a change has several advantages:

- A peer can inspect your work, and provide that essential QA review without any time pressure.
- You can create test scaffolding to find errors that the human eye is less tuned to spot.
- You greatly reduce the room for human error (such as fat-finger problems) to cause a failed change. How often have you typed in the wrong disk-partition number or cylinder number?

Geoff's Rule: Script anything where numbers or disk partitions are involved.

Of course, another aspect of reducing risk is to trial the change on a nonproduction platform. As the systems administrators in our story learned, however, this is no assurance that the production platform will behave identically to the test platform.

Problem Containment

Another important aspect of risk control is to ensure that you really do know the bounds of the exercise and can identify the point at which a problem was introduced.

Although it was never determined (don't get me started on the lack of root cause analysis), it was highly likely that the patches were nothing to do with the observed problem. The fact that the host was not rebooted prior to the application of the patches means that there was no guarantee of a stable baseline prior to the change. Just think how much time could have been saved if the host had been rebooted prior to the change, and this problem encountered at that time?

Any nontrivial change plan should have regular checkpoints scattered throughout it (including at the beginning). These are points at which correct system behavior can be confirmed, and thus problems detected early, and attributed to their cause faster and more accurately. The earlier we detect a problem, the better position we are in to take corrective action and continue with the change. By contrast, if we do not detect the

The earlier we detect a problem, the better position we are in to take corrective action and continue with the change.

Systems administrators must understand the risks inherent in the changes they are performing and plan for things to go wrong.

problem until the acceptance tests at the conclusion of the change, then our only course of action is to enter a protracted troubleshooting exercise, or back the entire change out.

When is Cold Not Really Cold?

Finally, another hard-learned lesson. I always thought that what constituted "cold" was pretty obvious. It turns out that not everyone shares this insight.

A "cold" backup (as opposed to a "warm" or "hot" backup) means a known, stable point in the system being backed up. We should always be able to recover from a cold backup and know that the system will operate properly, and the integrity of the data is assured.

The only way to perform a cold backup is with the relevant object (e.g., database, filesystem) quiescent, with no changes occurring. This is why cold backups are normally performed in single-user mode on UNIX.

I have seen people performing "cold" backups in multi-user mode (run level 3), and with a database live!

It is important to note that even with no users on a system, you are not guaranteed a stable cold snapshot if you do not ensure that the host is in the appropriate run mode, or that other steps are taken to guarantee the machine (or relevant subsystem) is quiescent. For example, CRON can kick off jobs that alter the filesystem during a backup, and upstream applications can FTP new transactions onto the host. Should either of these happen during a "cold" backup, we have created the potential for data loss.

Conclusion

Change management is a critical skill for systems administrators, whether you are running systems for a Fortune 500 company, an e-business that is exploding in capacity every week, or even a small business with relatively few systems to manage. Businesses rely on computers as mission-critical resources. It is our job as systems administrators to minimize service disruption. Change management is a core strategy and discipline for achieving that end.

A core element of change management is risk management. No change should happen to a production environment without appropriate risk assessment and planning. The weight of such an assessment should be proportional to the cost of the change failing and the time required to take corrective actions.

Systems administrators must understand the risks inherent in the changes they are performing and plan for things to go wrong. The same skills that we use in reactively troubleshooting system failures should be employed in proactively planning system changes.