Best Practices

Enterprise EXX00 Systems

Troubleshooting and Diagnostics

Jaffar Annab, Clive King

with contributions from

Kirk Strong CSSE, Paul Kaplan CSSE, Wayne Titchen TSE

Introduction

This document is an attempt at defining a bestpractices standard for trouble shooting Sun Enterprise E6X00, E5X00, E4X00, E3X00 system fatal errors. These types of errors can be difficult to isolate, especially given the architectural design limitations of the Enterprise systems.

This document is based on our experiences in dealing with these systems in the field and meant to provide others with a basic, methodical approach to help reduce the amount of effort and customer downtime incurred as a result of these problems.

Throught these documents, we provide refrences to information that we have used to helps us understand, diagnose and isolate the faulty component that caused the initial problem. It is our hope that this document will contiune to evolve and become more refined thru reader input and experience.

Diagnostics Summary – Analysing Failure Messages

This section covers only the analysis of hardware error messages. Software errors should be dealt with via the Solution Centre Operating System or kernel group depending on the process in place for your GEO (AMERICAS, EMEA,APAC). If in doubt, always use the Solution Centre both for diagnosis of software errors, and as a sounding board or to check your analysis.

Engineers with experience in the analysis of particular types of messages may wish to skip those areas some sections.

This document takes the approach that for systems which have experienced multiple problems, the Field Replaceable unit should be the system board.

Classes of messages

The first step must be to understand the origin of the message. Ask the questions

1. What part of the system reported the message? e.g. Kernel, OBP (this is not what generated it)
2. Where did the message originate from e.g. a disk driver would be reported as faulty by the sd or ssd disk target device driver and not the disk drive directly. This question attempts to identify that component(hardware or software) which noticed the problem.
3. What part(s) does the message implicate. If no individual part can be identified, then list FRU's for which a failure could cause this message and for each FRU, how did its failure cause the message seen.

Avoid jumping straight to step 3.

Deal with each message instance as an individual and don't assume two similar messages point to the same component.

It is important to keep in mind that a message may not be significant to the problem in hand. This is often true of messages logged on the console. It is very important to establish that the message seen is indeed related to the problem seen by the customer without ignoring significant messages is a judgement call.

syslog messages
These are messages reported in the following locations–

1. The console while the OS is running
2. In messages files in /var/adm

Examples are

1. Panic messages
2. Fiber Loop online/offline notices
3. Disk drive retry notices
4. Persistent memory error notices

`Solaris Common Error Messages (alphabetized) Info Docs: 11371` describes a number of the common and significant messages and is worth consulting first. Other messages may require reading kernel source code to understand there meaning which is the domain of the Solution Centre Engineer, SPG CPR Engineering and Solaris Sustaining.

It is very rare that the significance of a message can not be determined.

OBP and POST messages
These are reported by the Open Boot Prom and POST. These include Fatal resets.
The OS is not running at the time these messages are reported.
The output from prtdiag –v (included in Explorer) reports components which have been failed by POST. It should also be inspected for issues such as power supply failure and over temperature components.

Post accumulates a list of failed components across resets. However failed component information is cleared after a reset–por, after a physical power–cycle, and on board replacement.

<u>Understanding common hardware error types</u>

ECC memory errors
Enterprise Server memory is protected by ECC (Error Correcting Check). ECC allows the detection of both single and multiple bit errors and the correction of soft errors. Some memory errors can not be corrected and will cause the system to panic.

ECC memory messages are reported to the console and may appear in the messages file. By default, Correctable error (CE) messages are only reported when a particular DIMM has incurred more max_ce_err CE events. By default the default value of max_ce_err is 5

from kernel updates 2.5.1 103640−23 and form 2.6 105181−08 and in Solaris 7 & 8. See bugid 4127499 & 4127499 for further background.

Intermittent soft errors (CE Errors) are a fact of life and should be ignored unless the error rate exceeds that specificed for that part.

Setting max_ce_err to 1 in /etc/system will report all CE errors generated. A small number of CE errors does not mean a DIMM is faulty.

Documents available from the following link describe the causes of ECC errors and appropriate actions on detail.

`http://scs-cte.eng/product_info/allen.html`

The FRU is the board, even if a single DIMM is identified, if the system is question has been the subject of multiple or difficult to diagnose failures.

Fatal Errors
Fatal errors result from the inability of one component to communicate with another, an illegal configuration or an inconstency in communication between components. Fatal Errors are detected by the Ex000 Hardware i.e the Address Controllers (AC) and the Data Controllers (DC).

In general, any error detected by the AC or DC will be fatal unless masked. The Ex000 hardware does provide the ability to generate a reset on error, however only the errors that cause an illegal hardware state are issued a hardware reset.

Typical errors that would trigger a hardware reset are :

UPA Address Parity Error
FireTruck Address Parity Error
FireTruck Control Parity Error
DTAG Parity Error
UPA Master Port Timeout
Internal Error

Internal Errors normally indicate an inconsistent hardware state such as a coherence error or queue overflow as well as errors detected be the DC's such as a parity error on a control line from the AC's to DC's.

Ecache Parity Errors
This topic has been covered significant depth and breadth elsewhere. Suggested reading is given at the end of this document.

Understanding POST messages
Most post messages are self explanatory. When POST tests a FRU there are 1 of 3 outcomes

1. No problem is found with the FRU
2. A problem is identified with the FRU and it is not mapped out by the OBP
3. A problem is identified with the FRU and it is mapped out by the OBP

Therefore, the success of Extended POST in identifying faulty components can not be judged by the parts which are mapped out as the aim pf POST is to configure a system which will boot. It is *very* important to review the whole extended POST log generated on the console and to get a second pair of eyes to look at the output.

Externally Initiated Reset (XIR)
For E3000 to E6500 systems, it is possible to reset the system or cycle the power from a remote console providing the console is connected to Serial Port A on the Clock Board, and the key switch is in either the ON or DIAGNOSTIC setting. In addition, the remote sequences will be ignored if the key is in the OFF or SECURE position and Security Features (e.g OpenBoot security−mode) should be disabled.

You must enter the sequence character by character at a speed between 0.5 and 5 seconds between characters

The following Commands are available

Command Key Sequence:

Remote Power ON/OFF	(Return Key) (Return Key) (~ Key) (Control−Shift−p Key)
Remote system reset	(Return Key) (Return Key) (~ Key) (Control−Shift−r Key)
Remote XIR (CPU) Reset	(Return Key) (Return Key) (~ Key) (Control−Shift−x Key)

(You may need to press the ~ Key twice in the key sequence as the ~ can be interpreted as an ESCAPE character to some console servers)

The remote External Initiated Reset (XIR) command can be used as a limited aid to software debugging of hung systems. Currently XIR stores the following information for each CPU:

TL (Trap Level)
TT (Trap Type)
TPC (Program Counter)
TNPC (Next Program counter)
TSTATE (Trap State Register)

This information is reported by typing .xir−state−all at the OBP (You will need to send a BREAK to the machine to stop the machine from rebooting in order to issue this command)

This information should be of value to the Solution Centre engineer in moving the identification of root cause of the fault forward for both hardware and software issues.

It is hoped that this process never needs to be used.

Tools
What : AFSR Decoder
Where : `http://cte-www.uk/cgi-bin/afsr/afsr.pl`
When : For panic strings from kernels which do not include the SSRE functionality.

The software recovery project (often called the scrubber, but it is much more than that) has made a significant improvement in the reporting of ecache related issues. The logic used in the web tool is the same as that used for in the kernel update.

Newer versions of the ACT crash dump tool will also interpret the ASFR value and suggest an appropriate action.

What : Fatal Error Decoder
Processes output from a when FATAL ERROR is detected on an Ex000 server
Where : `http://cte-www.uk/cgi-bin/fatal-cgi.tcl`
When : When you get a fatal error and Extended POST does not identify the component at fault

What : Remote Power Resets/XIR
Processes XIR output from .xir−state−all on an Enterprise Server
Where : `http://cte-www.uk/cgi-bin/xir-cgi.tcl`
When :
Use this approach when the system is hung and will not respond to other types of "break" such as a break from the console.

Troubleshooting Process Flow

Aim of this section is to describe an approach to troubleshooting system failures on large enterprise systems where at least 1 fix has been attempted and a reliability problems still exist.

It does not contend to be the only approach to trouble shooting or a complete handbook to troubleshooting hardware problems on Enterprise systems with issues. It should be used as advise where the assembled team needs a common approach and used in sympathy with there collective experience.

Many of the suggestions state the obvious. Often in the heat of a customer situation, the obvious is missed or avoided.

Each system problem must be treated on its own merits as each is different. It is not possible to to provide a cookbook of actions which will resolved each and every situation. However, provided are some techniques which used in combination with the experience and common sense will resolve the vast majority of difficult Enterprise Class System persistent problems and will at worst always move the resolution of the problem forward. The sections are in the order which will usually be correct, but the order of execution will be dictatedby the situation. However, missing section out as they look like

they will take too long is not a sensible approach.

The most important attributes in good trouble shooting are

1. Determine what the problem(s) you are trying to solve really is
2. What facts do we know
3. What facts do we need to know
4. What are our possible causes
5. Do the facts we have allow our possible cause to stand
6. Not jumping to cause

Background

Large Enterprise Servers can prove difficult to diagnose if some of the following conditions exist

1. Multiple problems
2. Limited down time
3. Imprecise diagnostics output
4. Inability to pin−point failure

While such failures are rare, they do inflict considerable pain on the customer concerned and cost to Sun.

Triggers for use

A number of triggers are suggested

1. One attempt at fix has failed to rectify the problem reported by the customer
2. Multiple problems are suspected on the system and we do not know for sure what they are
3. The system has a history of previous failure
4. We can not describe in 1 sentence what the problem is with the system
5. We are not 100% sure it is a hardware problem or we could have a class issue where a genuine product defect means that component replacement like−for−like will not resolve the issue in hand.

Prerequestis

System Console

```
While ( no console on system ) {
        attach console
}
```

Attach a console which can keep a persistent log of the console output. This is vital for catching failure messages while the customer has the system in production or during stress testing and also for capturing the extended POST log.

Do not miss this step out and do not move on before this step is completed.

What do we know?

State in order, with dates and time if possible, all events that we know of since the first

failure occurred. Also look at least 1 month prior to the first failure.

Do we have crash dumps, panic messages, POST logs? What do they tell us and have we really asked the right person yet as to what they mean?

All future changes (hardware or software) must be recorded and all involved should be made aware of any change in advance.

What is the behaviour of other customer systems at the same site over the past year. The intent here is to expose site based problems such as the environment and power.

What is the system doing when it fails?

Account Management

The hardest part is often managing the customer. Setting the correct expectation that this is a hard problem and we don't yet know the root cause. However, we have a tried and tested approach to this type of problem and it will be resolved in a timely manner. The cooperation of the customer is key to timely resolution and we will work with the customer to minimise impact on there business.

One option to consider is a short term loan system of a configuration sufficient to keep the customer business running. See the HostSparing section of this document

Experience suggests that keeping the customer in the dark about the true situation does not work well in the long term and customers are most often calmed by honesty.

Getting into a known state

This section addresses the need of getting the system and our trouble shooting into a known state.

Hardware Baselining

Undertake a hardware audit which will require a limited amount of downtime.

1. What parts are in the system and where did they all come from?
2. Are all the parts qualified for that system?
3. Which parts have been installed/changed prior to or after the first failure?
4. Who has done the recent work and why did they do it?
5. What were the last parts to be added/changes on the system?
6. Check that all Field Replaceable Units part numbers in the system are compatible with the system type by the Sun System Handbook at `http://infoserver.central/data/sshandbook/`
7. Ensure all parts are in good visual order
8. Components such a load boards, sufficient power supplies are in place.

Aim for minimal change/distrurbance of hardware. No actions should be taken such as cleaning/re–torquing CPU's at this stage. This type of action should only by used as the

result of an ATS specification which suggest a cause for which this is an appropriate action. For example where visual inspection revealed 50% of heat sinks were loose.

The configuration which the system left the factory can be checked on
Newark – `http://sfepdpkg2.ebay:8000/request/printcis` For EXX00 systems

Beverton –
`http://bops.west/Starfire/ManufacturingTools/ReprintFruList.html` For E10000 systems

An amber light on a board does not always mean it is faulty and is the root cause of the problem in hand. Disk boards in EXX00 systems always have the amber light on!

Extended Post
Execute 5 loops of extended POST. Most faulty hardware will be identified using Extended POST and this step must not be avoided and should be done as soon as a potential multi–problem system is identified. This may take some time to run(the bigger the system, the longer it takes), but is an essential baseline.

The diag switch must be set to the diag position and power cycle the system.

At the OBP, set the configuration–policy to be board. Make sure that all the post logs are reviewed, not just that post reports a clean system. Post may report failures or warnings, but still configure the component.

If POST detects a suspect component, disable the board at the OBP and rerun POST. This would be positioned as the first down time slot if this has not been done before with *all* the parts which are in the system at this time.

Explorer
Take a current explorer and review its contents. Files which should be reviewed in detail include

- sysconfig/prtdiag–v.out – Inconsistencies
- sysconfig/eeprom.out – OBP setting are suitable for hardware debugging
- messages/* – Interesting messages starting with error, warning or panic for example
- etc/init.d/syssetup – check crash dump configuration
- etc/system – Look for unreasonable settings

A number of good tools exist for analysis of explorer. The best known examples is Sentinal (`http://spider.aus/sentinal`). Take the explorer itself and/or the output of such a tool and look for obvious anomalies such as over temperature boards, firmware revisions which are very out of date, parts which have open FIN's against them.

In addition, HealthScreen (`http://otis.uk`) may also report potential significant issues.

This gives an idea of the overall state of the system, and is useful in pushing back on the customer where their administration of thesystem in question is not what it might have

been.

At this stage, also install ACT (`http://otis.uk`) on the customer system to enable more rapid initial analysis of crash dumps by Solution Centre OS or kernel engineers.

Analytic Troubleshooting

Complex problems generate a large amount of information which can swamp the individual. Multiple problems provide a greater challenge. The engineer/management may choose to engage the services of an ATS facilitators if they feel the following conditions exist

1. An intention exists to escalate to CTE. From September 2001 allescalation to CTE must include an completed ATS specification.
2. No one can give a concise technical description of the problems to be solved
3. The volume of data has overwhelmed those involved.
4. The divide between fact and assumption has become ill–defined.
5. We do not know for sure how many problems we have and if they are hardware or software.
   See `http://sun-resolve.central` for details.

ATS can assist the team in standing back and addressing the following

1. What is the problem I am trying to solve?
2. What facts do we know?
3. What facts don't we know?
4. What sharp comparisons can we draw?
5. What has changed?
6. What could have caused out problem?
7. Reduce the list of possible causes?
8. Identify the most plausible possible cause?
9. How do we eliminate each possible cause?
10. What risks do we take in eliminating a possible or implementing a fix?
11. Determine how confident we are that we have found our root cause?
12. Protect future plans?
13. Think beyond our current problem?

Stress testing

If the root cause of the problem has not been found by this stage, then consideration must be given to reducing the configuration of the system and stressing the system in such as way as to reproduce the problem.

Suggested tools are included in this document

It is essential that any stress tests are only run with the system in question out of production and with data filesystems unmounted.

Additional message reporting
Before starting stress testing, add the following entries in /etc/system, set the following to enable more verbose reporting of errors by the kernel and then reboot.

```
set snooping=1
set obpdebug=1
set forthdebug=1
set ce_debug=1
set ce_show_data=1
set ce_verbose=1
set report_ce_console=1
set report_ce_log=1
set reset_debug=1
set ue_debug=1
set sd:sd_error_level=1
set max_ce_err=1;
```

Not all settings are valid on all version of Solaris so some warning may be given out at boot time.
These must be removed before the system goes back into production as they can have a performance impact.

Strategy – Divide and Conquer
The divide and conquer strategy allows the field engineer to home in on the faulty components by eliminating good components as suspects.

This approach is valid for systems with a single faulty component, particular if the reporting of the failure appears to move between components. However, it is often the only way to proceed when a system has multiple faults (often combined hardware and software). Finding the root cause of one problem does not mean that all root causes have been found.

It is essential to remember that the component that reports a failure is not necessarily the component which is the root cause of the failure.

The Field Replaceable Unit (FRU) must be considered to be the system board level. Individual system boards must be off–lined at the OBP by setting the disable–board–list OBP property. This avoid the potential of introducing additional problems through contact with the system. DO NOT REMOVE BOARDS.

The "divide and conquer" strategy requires a log to be made of all changes. No hardware should be physically removed/changed in the system until a firm failure is narrowed down to a signle board. When a problem board is identified, it should be replaced with a "warm" board which has been run–in on a lab system and then tested with extended POST and suitable period of stress testing.

Set the configuration-policy for POST to be board. For example

```
setenv configuration-policy board
setenv disabled-board-list 45ef (disable boards in slot 4, 5, 14 and 15)
```

To reset a list to null, enter the following at the ok prompt:

```
ok set-default disabled-board-list
```

These variables take effect on the next reset or power-on.

For E450/E250 use the .asr commands in OBP to disable CPUs and prevent them being seen by Solaris.

```
asr-disable
.asr lists components and their state
asr-enable reactivates them
```

Here is a sample output of prtdiag -v:

```
========================= CPUs =========================
                    Run  Ecache   CPU   CPU
Brd CPU  Module  MHz    MB    Impl.  Mask
--- --- ------- ----- ------  ------ ----
 0   0     0     400    8.0   US-II  10.0
 0   1     1     400    8.0   US-II  10.0
 2   4     0     400    8.0   US-II  10.0
 2   5     1     400    8.0   US-II  10.0
 4   8     0     400    8.0   US-II  10.0
 4   9     1     400    8.0   US-II  10.0
 6   12    0     400    8.0   US-II  10.0
 6   13    1     400    8.0   US-II  10.0

========================= Memory =========================

                                                 Intrlv.  Intrlv.
Brd   Bank   MB    Status   Condition  Speed   Factor   With
---  -----  ----  -------  ----------  -----  -------  -------
 0     0    1024   Active      OK        60ns    8-way     A
 0     1    1024   Active      OK        60ns    8-way     A
 2     0    1024   Active      OK        60ns    8-way     A
 2     1    1024   Active      OK        60ns    8-way     A
 4     0    1024   Active      OK        60ns    8-way     A
 4     1    1024   Active      OK        60ns    8-way     A
 6     0    1024   Active      OK        60ns    8-way     A
 6     1    1024   Active      OK        60ns    8-way     A

========================= IO Cards =========================
     Bus   Freq
Brd  Type  MHz   Slot  Name                              Model
---  ----  ----  ----  --------------------------------  ----------------------
1    SBus   25    0    QLGC,isp/sd (block)               QLGC,ISP1000
1    SBus   25    1    network                           SUNW,sbus-gem
1    SBus   25    2    SUNW,qfe                          SUNW,sbus-qfe
1    SBus   25    2    SUNW,qfe                          SUNW,sbus-qfe
1    SBus   25    2    SUNW,qfe                          SUNW,sbus-qfe
1    SBus   25    2    SUNW,qfe                          SUNW,sbus-qfe
1    SBus   25    3    SUNW,hme
1    SBus   25    3    SUNW,fas/sd (block)
1    SBus   25   13    SUNW,socal/sf (scsi-3)            501-3060
3    SBus   25    0    QLGC,isp/sd (block)               QLGC,ISP1000
3    SBus   25    2    network                           SUNW,sbus-gem
3    SBus   25    3    SUNW,hme
3    SBus   25    3    SUNW,fas/sd (block)
3    SBus   25   13    SUNW,socal/sf (scsi-3)            501-3060
```

Boards 1 & 3 have I/O, so eliminate the other boards first.
For example, use the following steps

1. off-line boards 4, 6
2. Reboot
3. Stress
4. If failure exclude board 2. If no failure include board 4.
5. Reboot
6. Stress system

This process is repeated until the board(s) with the fault(s) are isolated. This process of reducing to a minimal configuration may have to continue outside the maintenance window. Often simple performance tuning of the OS can be given to the customer to compensate for the loss of system resources provided it is not peak processing time for the customer. In addition, many customer system are overconfigured for the workload outside of peak load.

A difficult question to answer is how long should diagnostics be run for. There are a number of possible right answers

- As long as you have a period of down time for
- As long as the customer lets you
- 2 or 3 hours

One problem with this approach is that potential exists for a faulty board to not exhibit a problem as the period of stress testing has not been long enough. There is no way round this other than extending the stress testing period or returning a reduced configuration system to production.

Patching
For the purpose of error reporting, if possible suggest to the customer to upgrade to the lastest kernel update for the OS and OBP revision if there is a significant deviation from the current. Stress to the customer that this is not intended to fix the problem, but to improve error reporting. In essence ensure that the system has a current OBP and the Software Recovery functionality(Scrubber) in the kernel patch.

Tools
The choice of stress test should be driven by the type of work the customer does when the system fails. See the tools section of the document for details of tools which can be used to stress systems.

Tidyup
Remember to reset all values to their previous (unless we know them to be wrong and the customer is both aware and in agreement).

Class issues
Once a individual system has been diagnosed and is running without fault, it is important to identify other systems which may be exposed to the same root cause with that customer an other customers.

The action will depend on the problem type.

If a product defect (as opposed to a faulty componet) is possible, an escalation to CTE should be raised to investigate this further. The triggers for this may include

1. Multiple systems showing the same problem across a range of customers or individual customer sites.
2. New systems having a high failure rate

Stress Testing Tools

Here we provide a summary of a selection of tools and techniques which may be found useful when stressing systems which have exhibited problems which are expected to be caused by faulty hardware, but faulty component has not yet been identified.

Diagnostics Clearing House
`http://dch.eng`
You need to register before you can get access to this site.

Caveat : If you have not used these tools in a safe environment such as a lab, then don't use them on a customer site.

General Tools
Thes tools test both I/O and CPU memory

SYSTEST
This is a general load generator available from `http://otis.uk`. It should not be used on customer's production systems as it tends to fill the root filesystem up.

SUNVTS
SunVTS provides hardware fault detection, fault isolation and system exercise capabilities
`http://diagnosis.eng/sunvts/sunvts_intro.shtml`

compress
Put a large file in /tmp and run compress and uncompress on it in a loop. Large is at least as large as physical memory.

CPU/Memory
These tools tend to exercise the CPUs on a system

SETI
SETI@home is a scientific experiment that uses Internet−connected computers in the Search for Extraterrestrial Intelligence (SETI).

It is very good at keeping CPU's busy.

`http://setiathome.ssl.berkeley.edu/`

CPU and Memory
These are cache–specific tests which are multi–threaded, with each thread binding to different CPUs and then running the specific tests. The same is true for the dcachtest1.c and dcachtest2.c.

`http://dch.eng`

SSO
Tests all of the CPU modules in the system simultaneously. The diagnostics are executed in an order where the most diagnostics which are the most likely to fail are executed first and the most often. The actual run times of the individual diagnostics are scaled based on CPU frequency and external cache size in order to keep the actual run times relatively constant.

The most likely failure mode of these diagnostics is a parity error. When this occurs it will cause the system to panic. The CPU which reports the parity error should then be blacklisted or removed before the system is rebooted and the test restarted. All failures will be found sequentially, with each failure requiring a system reboot.

SSO runs a sequence of utilities including Linpack.

***WARNING: Do not use SSO on 501–4995 processors as good modules will give eache partity errors. Running SSO requires examining the processors first to determine the part number. ***

***WARNING: Do not change the processor speed as suggested in the SSO documentation ***

Available from `http://dch.eng`

savecore –L
Aim here is to scan all of memory every few minutes

1. 1. Delete the swap devices with swap –d
2. 2. Set up the dump device as a partition different from the swap partition which is just larger than physical memory. This might be a challenge on some systems.
   ```
   dumpadm -d /dev/dsk/cnnnnnnn
   ```
3. 3. Set the system to dump the whole of physical memory used, not just the kernel.
   ```
   dumpadm -c all
   ```
4. Run this script
   ```
   #!/bin/ksh

   while true
   do
     date
     savecore -L
     /bin/rm /var/crash/`uname -n`/*
   ```

```
    sleep 300 # system does other stuff
done
```

An alternative is `wc -c /dev/mem`

I/O

These test are used to generate an I/O load on the system under scrutiny

dd

The aim is to read across each controller in the system many times. As an example for a system with 3 I/O controllers. Pick any disk on these controller, but all controllers must be covered, not all disks.

```
#!/bin/ksh
while true
do
   dd if=/dev/dsk/c0t0d0s0 bs=8k of=/dev/null &
   dd if=/dev/dsk/c1t0d0s0 bs=8k of=/dev/null &
   dd if=/dev/dsk/c2t0d0s0 bs=8k of=/dev/null &
   wait;
done
```

Block size (bs) can be varied between 0 and 1 meg.

Diskomizer

Diskomizer is a program for testing and verifying disk subsystems. It uses multiple processes to do asynchronous writes and reads to devices that are specified and then verifies the data that is read back is the data that was written to that block. Every block of data has a unique header each time it is written and the body of the data changes every time the block is written.

Diskomizer will find broken devices and paths to devices software bugs and latent faults in hardware. It does not break these devices, it simply finds faults that are already there. It knows nothing about the under lying storage devices, and hence can be used just as well to generate load on NFS file systems as any other device. Diskomizer can be run as an ordinary user as long as that user has permission to open the files and or devices that are being used.

`http://cte-www.uk.sun.com/~gerhard/diskomizer_new/latest/diskomizer.html`

Substantial system resources may be required to drive a large storage subsystem.

Diskomizer is generally used for destructive testing.

Stortools

Storetools is a diagnostic, configutration and stress testing set of tools developed for use with most of Sun's Storedge storage products, see

`http://webhome.central/stortools/` for more information. Dex32, the program described below is part of storetools.

dex32
The dex32 program generates I/O transactions to a device.The command−line options allow you to specify:

- Test information including the type of test, test−related flags, and the number of I/O transactions per pass
- The range across the media to test
- The number of passes to run
- The type of I/O transaction
- Whether to stop after a certain number of errors or an elapsed time frame
- The data pattern used during testing

You can use dex32 to exercise disk drives, tape drives, file systems, and memory. dex32 is best run in conjuction with Stortools to flush out any problems with FCAL type storage.

Example:
Run the Sequential test, write only, across 100 Mbytes on the specified device. The test performs eight I/O transac− tions per pass.

```
dex32 −S8 100m −w /dev/rdsk/c1t2d0s6
```

Network
It has been found useful that when running CPU intensive stress tests to also generate some periodic network traffic. This will interrupt the processor execution and has been known to expose certain type of CPU and network card problems.

Modify this script to suit your requirements

```
#!/bin/ksh
while true
do
   spray <host under test>
   sleep 1
done
```

How much load?

I/O
iostat -xn 5
The queue length (wait) should be about equal to the queue depth of the disk. For Photon this is 15.

CPU
vmstat and/or mpstat should show all CPU being busy with a 0 in the idle column and 100 divided between the sys and usr columns. If a lot of I/O is being done, some time may be

spent in the wt column.

Stress tests should have a positive figure in the r column of vmstat to indicate that the system has more work to do than it can manage. However, tests such as seti or linpack will want to run 1 job to 1 CPU.

Where to go for help
Some tools have authors who may be able to assist.

Escalation Process

You've done all of the above, but the system is still crashing, and the customer is not very happy, what next: ESCALATE. The procedure to escalate is well defined in the Quality Managment System. It describes the steps necessary to initiate and progress a Field Escalation from the District level, to the Region level and, if necessary, to the Area level via the online escalation tool – WebESC.

If the case is still not resolved, then it should be further escalated into CTE using the appropriate process.

Here is a brief synopsis of how the process it iplemented in the Americas Geo:

```
Case requires escalation -> Escalate Case to Region ->
         Develop Region Escalation Action Plan ->
         Region Escalation Closure ->
         If necessary, Escalate Case to Area ->
         Develop  Escalation Action Plan ->
         Area  Escalation Closure ->
         If necessary, further escalated outside Area via CCC to CTE

Note: CCC -> Customer Care Center
          CTE -> Corprate Technical Escalations
```

The following links provide details on the above process for the Americas:
Field Escalation
`http://americasqual.central/qms/us/isodoc/fld/FLDP1033.html`
Escalating to CTE
`http://americasqual.central/qms/us/isodoc/ato/CCCW2211.html`

Sample Errors and Solutions:

Here is an example of a FATAL error that can lead you down the wrong track:

```
18-MAR-2001 06:09:05.17 Fatal Reset
18-MAR-2001 06:09:06.55 0,0>FATAL ERROR
18-MAR-2001 06:09:07.00 0,0>    At time of error: System software was  running.
18-MAR-2001 06:09:07.00 0,0>    Diagnosis: Board 0, centerplane pin, connector pin, AC
18-MAR-2001 06:09:07.22 0,0>    Diagnosis: Board 2, UPA PORT Device, AC
18-MAR-2001 06:09:07.23 0,0>Log Date: Mar 18 11:15:33 GMT 2001
18-MAR-2001 06:09:07.23 0,0>
18-MAR-2001 06:09:07.23 0,0>RESET INFO for CPU/Memory  board in slot 0
```

```
18-MAR-2001 06:09:07.44 0,0>    AC ESR 00000400.00000000 FT_ARBERR
18-MAR-2001 06:09:07.44 0,0>    DC[0] 00
18-MAR-2001 06:09:07.44 0,0>    DC[1] 00
18-MAR-2001 06:09:07.44 0,0>    DC[2] 00
18-MAR-2001 06:09:07.45 0,0>    DC[3] 00
18-MAR-2001 06:09:07.45 0,0>    DC[4] 00
18-MAR-2001 06:09:07.45 0,0>    DC[5] 00
18-MAR-2001 06:09:07.45 0,0>    DC[6] 00
18-MAR-2001 06:09:07.45 0,0>    DC[7] 00
18-MAR-2001 06:09:07.66 0,0>    FHC  CSR 00050200 LOC_FATAL SYNC NOT_BRD_PRES
18-MAR-2001 06:09:07.66 0,0>    FHC RCSR 02000000  FATAL
18-MAR-2001 06:09:07.67 0,0>
18-MAR-2001 06:09:07.67 0,0>RESET INFO for CPU/Memory  board in slot 2
18-MAR-2001 06:09:07.67 0,0>    AC ESR 00000000.00600001 IPREP FERR UPA_A_ERR
18-MAR-2001 06:09:07.88 0,0>    DC[0] 00
18-MAR-2001 06:09:07.88 0,0>    DC[1] 00
18-MAR-2001 06:09:07.88 0,0>    DC[2] 00
18-MAR-2001 06:09:07.88 0,0>    DC[3] 00
18-MAR-2001 06:09:07.88 0,0>    DC[4] 00
18-MAR-2001 06:09:08.10 0,0>    DC[5] 00
18-MAR-2001 06:09:08.10 0,0>    DC[6] 00
18-MAR-2001 06:09:08.10 0,0>    DC[7] 00
18-MAR-2001 06:09:08.11 0,0>    FHC  CSR 00050030 LOC_FATAL SYNC BRD_LED_M BRD_LED_R
18-MAR-2001 06:09:08.11 0,0>  FHC RCSR 02000000  FATAL
```

In this situation POST flagged CPU/MEM board 0 and placed it in low power mode, but if you look at the actual error on board 2, you will see that an error occured on UPA a which is CPU module 0.

As it turns out CPU 0 on board 2 suffered a DTAG parity error, board 0 was a victim in this case. The action is to replace CPU0 on board 2, always following the Best Practices Guide for CPU replacement.

Here is another example:

```
Fatal Reset
0,0>FATAL ERROR
0,0>    At time of error: System software was running.
0,0>    Diagnosis: Board 0, AC, CPU, Dtags
0,0>Log Date: Apr  9  8:56:22 GMT 2000
0,0>
0,0>RESET INFO for CPU/Memory  board in slot 0
0,0>    AC ESR 00000000.00001002 TAGVICT UPA_B_ERR
0,0>    DC[0] 00
0,0>    DC[1] 00
0,0>    DC[2] 00
0,0>    DC[3] 00
0,0>    DC[4] 00
0,0>    DC[5] 00
0,0>    DC[6] 00
0,0>    DC[7] 00
0,0>    FHC  CSR 00050200 LOC_FATAL SYNC NOT_BRD_PRES
0,0>    FHC RCSR 02000000  FATAL
0,1>
```

Here the error was due to a faulty CPU on UPA port B on system board 0. This was verified by placing the cpu on a new board on UPA port A and seeing the same error

however UPA_B_ERR became UPA_A_ERR

Required Reading

Best Practices Addressing: E−cache Parity Errors
`http://bestpractices.central/`

FIN #: I0570−3
SYNOPSIS: Provide instructions and guidelines to CORRECTLY identify Ecache panic fault diagnosis.

FIN #: I0616−1
SYNOPSIS: Ecache Memory Parity Error

FIN #: I0587−1
SYNOPSIS: Fatal Reset Errors occurring on the Enterprise Servers may not be diagnosed as E−Cache Tag Parity Errors.

FIN #: I0377−1
SYNOPSIS: ECC Memory Error Guideline for Enterprise Servers.

`http://leika.eng/Firmware/POST/` − General information on POST.

Useful Reading

`http://cte-www.uk` − Follow the technical library link.

UPA interconnect 960−1156−01

Sunfire Architecture Manual 950−2241−1

About the authors:

Clive King is an engineer with the Corprate Technical Escalations working in the Systems Product Group in the UK.

Jaffar Annab is a Regional System Support Engineer in the Mid Atlantic region in the US.