Data integrity

Bernd Panzer-Steindel, CERN/IT
Draft 1.3 8. April 2007

Executive Summary

We have established that low level data corruptions exist and that they have several origins. The error rates are at the 10^{-7} level, but with complicated patterns. To cope with the problem one has to implement a variety of measures on the IT part and also on the experiment side. Checksum mechanisms have to implemented and deployed everywhere. This will lead to additional operational work and the need for more hardware.

Introduction

During January and February 2007 we have done a systematic analysis of data corruption cases in the CERN computer center. The major work in the implementation of probes and automatic running schemes were done by Tim Bell, Olof barring and Peter Kelemen from the IT/FIO group. There have been similar problems reported in Fermilab and Desy and information exchange with them was done.
The following paper will provide results from this analysis, a judgment of the situation and a catalogue of measures needed to get the problem under control.
It is also to be seen as a starting point for further discussions with IT, the experiments and the T1 sites.

Status

There have been several activities to accumulate statistics and understand the underlying problems for the seen data corruptions:

1. **Disk errors**
   A special program was developed by Olof Barring and than Peter Kelemen improved it considerably and he is now responsible for the program This program writes a ~2 GB file containing special bit patterns and afterwards reads the file back and compares the patterns. This program was deployed on more than 3000 nodes (disk server, CPU server , data bases server, etc.) and run every 2h.
   About 5 weeks of running on 3000 nodes revealed 500 errors on 100 nodes.

Three different types of errors were seen :

- single bit errors (some correlation with ECC memory errors) (10 % of all errors)
- sector- or page-sized regions of corrupted data (10% of all errors)
- 64k regions of corrupted data, one up to 4 blocks (large correlation with the 3ware-WD disk drop-out problem) (80% of all errors) different 'style' of corruption between SLC4 and SLC3

2. RAID 5 verification

The RAID controllers don't check the 'parity' when reading data from RAID 5 file systems. In principle the RAID controller should report problems on the disk level to the OS, but this seems not always to be the case. The controller allows to run the 'verify' command which reads all data from disk and re-calculates the RAID5 checksum and corrects the discovered bad RAID5 blocks (block size is 64 KB). But of course it does not have a notion of what is 'correct' data from the user point of view.

Running the verify command for the RAID controller on 492 systems over 4 weeks resulted in the fix of ~300 block problems. The disk vendors claims about one unrecoverable bit error in 10^{14} bits read/written. The 492 systems correspond to about 1.5 PB of disk space. To get the real number of physical bits touched during the process one has to include the fact that the data on disk contain an ECC overhead (~17%) and that the 'parity' disk is also used, which leads to an increase of 35%. Thus over 4 weeks in total

$1.5 * 10^{15}$ (bytes) * 1.35 (effective) * 8 (bits) * 4 (weeks)

→ ~$650 * 10^{14}$ bits were read by the verify process.

During the same period about $200 * 10^{14}$ bits were read/written by the applications.

The first thing to notice is that the verify command stresses the disks 3 times more than the actual physics applications, by just running it once per week.

The second observation is the that we measured about 300 errors while from the mentioned BER (Bit Error Rate) and the usage one would expect about 850 errors. As the vendor BER numbers are normally on the 'safe' side the measurement is close to the expectation.

3. Memory errors

We have 44 reported memory errors (41 ECC and 3 double bit) on ~1300 nodes during a period of about 3 month. The memory vendors quote a Bit Error Rate of 10^{-12} for their memory modules. The 1.5 PB of disk servers run during that period at an average IO rate of 800 MB/s. To transfer a byte from the network to the disk or vice versa needs on average 6 memory read/write operations (read from NIC buffer – write to kernel memory → read from kernel memory – write to application user memory → read from application memory – write to file buffer kernel memory → read from file memory – write to disk). And of course the same transfers take place on the worker node and the tape server. Thus the 800 MB/s translate into 800 MB/s * 8 bit * 6 * 2 = $7.7 * 10^{10}$ bits/s of memory read/written per second. With the quoted vendor error rate of 10^{-12} one would

expect during 3 month a total of 600000 ECC single bit errors reported. Thus the observed error rate is 4 orders of magnitude lower than expected for the single bit errors, while one would expect no double bit errors. But there is also a problem with the correct reporting of ECC errors to the IPMI level, which is motherboard dependent and we might have not 'seen' all errors.
Single bit errors don't lead to data corruptions as they are corrected, only double bit error do cause problems.

4. **CASTOR data pool checksum verification**
   All the previously mentioned error will of course result in the corruption of user data. To assess the size of the problem, files on a disk pool were checked and the previously calculated checksum on tape was compared with another adler32 calculation. During a test 33700 files were checked (~8.7 TB) and 22 mismatches found.
   That translates into an error rate of one bad file in 1500 files. Assuming that the majority are 64KB problems and on average there are 2 wrong blocks then this yields a byte error rate of $3 * 10^{-7}$.

   Remark : This number is only true for non-compressed files. A test with 10000 compressed files showed that with a likelihood of 99.8 % a SINGLE bit error makes the whole file unreadable, thus the data loss rate would be much higher for compressed files

There are some correlations with known problems, like the problem where disks drop out of the RAID5 system on the 3ware controllers. After some long discussions with 3Ware and our hardware vendors this was identified as a problem in the WD disk firmware. We are currently updating the firmware of about 3000 disks.
We also had a case of memory incompatibility where memory modules on 120 nodes had to be exchanged. In this area one can watch in industry the problem of shorter and shorter periods of 'throwing' new chipsets on the market and a growing memory-chipset problem-matrix. During our tests we have also seen errors on high-end hardware and the recent publications about disk reliabilities also indicate no major difference between different types of hardware.
http://labs.google.com/papers/disk_failures.pdf
http://www.usenix.org/events/fast07/tech/schroeder/schroeder_html/index.html

There is a clear distinction between 'expected' errors based on the vendors reliability figures and obvious bugs/problems in the hardware and software parts.

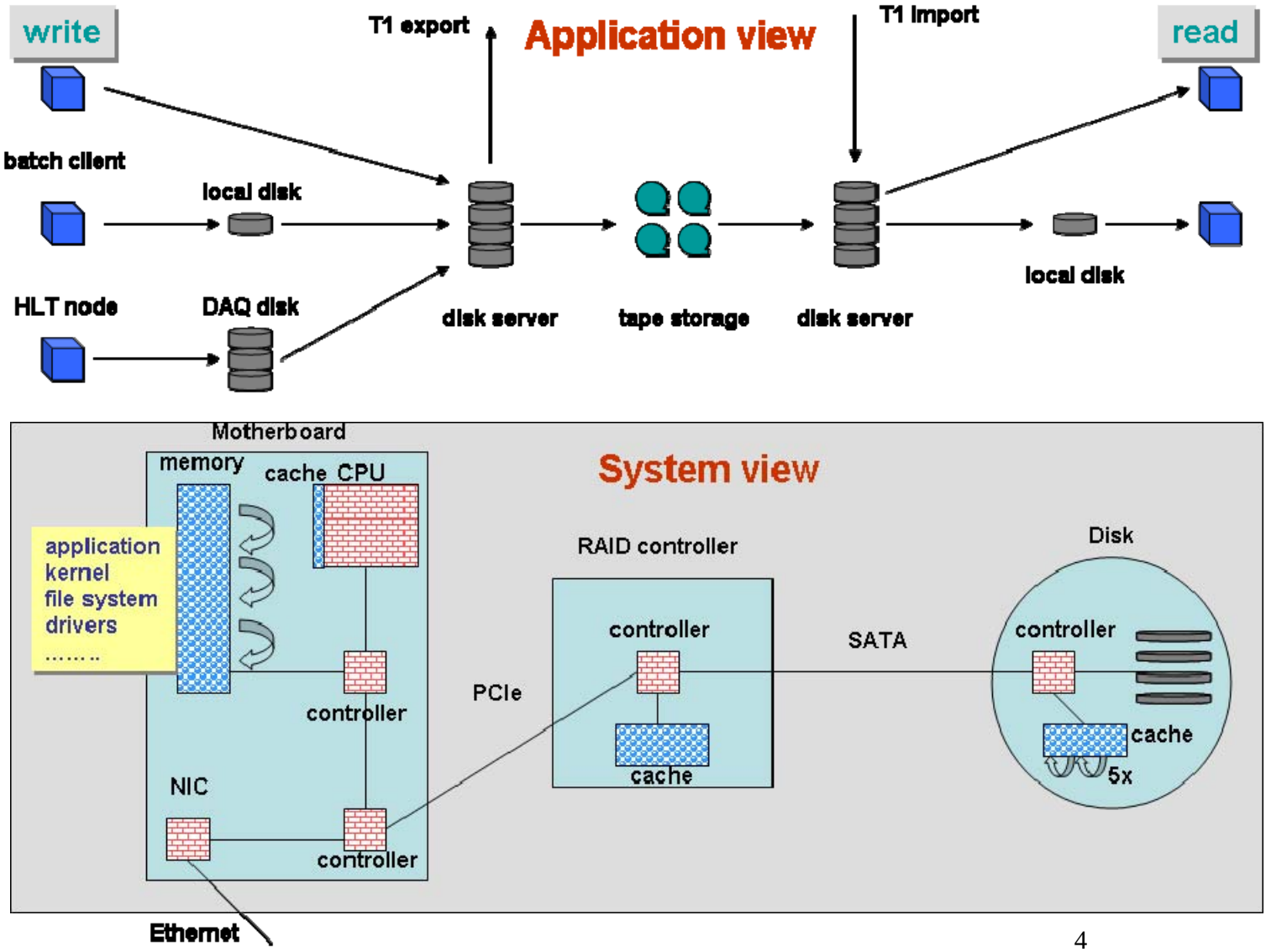
The fact that the disk probe reports regularly corruption errors as described in point 1 shows clearly that the different protection mechanisms don't work 100% , i.e. not every errors in the data flow is correctly treated and reported to the upper layers.

The problem here is the large amount of controllers/memory/chipsets we have in the computing system. Data has to pass through a long chain of equipment in its lifetime.

In principle the whole data flow chain is protected through the implementation of ECC (Error Correction Code) and CRC (Cyclic redundancy Check) :

1. The memory is capable of correcting single bit error
2. the cache in the processor is ECC protected
3. PCIe and SATA connections have CRC implemented
4. the disk cache has ECC memory and the physical writing to disk has as well ECC as CRC in a complicated manner implemented to correct up to 32 byte errors (per 256 bytes) and detect any data corruption. The data is actually 5 times encoded before it reaches physically the disk.

The following picture shows a principle layout of the data flow inside the farm and inside a node:



Measures

The ensure data integrity and have an early warning system the following actions need to be taken on several different layers :

1. continuously running probes in the background to check for errors on disk and in memory
2. regular scrubbing of the disk and memory systems
3. close monitoring of error messages (logs, IPMI, etc.) and improvements in this area
4. the applications need to deploy CRC checks during writing and before reading of any data

Client part

There are several possibilities to ensure data integrity on the client side, but not all of them can really be implemented :

1. **Writing**
   the application calculates the checksum of every block/event, writes it to disk, syncs the disk and reads it back for checksum comparison. Checksums for all blocks/events are stored in a data base.
   → destroys completely the disk performance, large data base needed

2. **Writing**
   the application calculates the checksum of every block/event, writes it to disk, syncs the disk and reads it back for checksum comparison. The application integrates the checksum itself into the data stream, so that the data are self-describing.
   → Still requires a re-read of the data and considerable changes in the application

3. **Writing**
   creates a running checksum for the whole file to be written to disk, re-read the file at the end of the job for checksum comparison
   → doubles the needed IO performance, only possible for file writing during processing and re-processing, does not work for event-building

4. **Writing**
   All data are encoded with an ECC algorithm which is capable to correct multiple 64 Kbyte blocks of data.
   → must be a special algorithm as normal ones are coping with 10s of byte corrections only. At least 20% more data data needs to be written. More CPU resources needed to do the encoding.

5. **Reading**
   the application reads the file from disk first for a checksum verification and then again for processing
   → doubles the read performance needed, assumes that no corruption appears in the time–interval between the first check and the end of the processing (jobs can run for 10h or longer)

Block level checksums need quite a sophisticated infrastructure and seems not to be a reasonable option and the same is true for ECC data encoding. All this would need quite some effort from the experiment application side, which seems today not a feasible option.

As an example for the DAQ-T0-T1 chain the following checks need to be done to ensure data integrity:

1. the checksum of a file is calculated at the experiment before it is written to the online buffer
   → if it is wrong when checked in the online disk buffer , there is no way to recover

2. the checksum is verified when the experiment application uses the file for processing
   → if it is wrong it needs to be re-copied from the online buffer

3. the checksum is verified while the file is written to tape
   → but the checksum is calculated on-the-fly while the file is copied to tape, thus a wrong file needs a long procedure of cleaning in CASTOR and re-copying from the online buffer.

4. the checksum is verified at the tape level on the T1 site
   → needs re-transfer if the checksum is wrong

A prerequisite to these procedures is the propagation of the checksum and file-id from the online part to the T0 and the T1 sites, which can be part of the LFC and bookkeeping experiment data bases.

Server part

The following procedures must run constantly as background processes :

- disk probe on each node for each file system, writing and cross-checking a file every 2 h (ok)
- memory probe on each node, checking 1 GB of memory every 4h (partly ok)
- RAID5 consistency checks once per week on each disk server (ok)

- IPMI sensor to report all errors (cable CRC, memory ECC, etc.) and provide alarms (partly ok)
- CASTOR pool checksum verification of every file, once per week (not yet done)

  ok means these are already implemented and running today

If the probes discover problems than the corresponding node needs to be stopped and investigated. There are still some more questions to be answered :
e.g. At time t1 the node was still okay and than reported problems at time t2, should one than invalidate all files written during that time period ?! this requires some sophisticated tracking which is not yet in place.

Summary

We have established that low level data corruptions exist and that they have several origins. There is some effort to reduce them, but it is very unlikely that they will disappear completely. We will rather see from time to time (new hardware, software, firmware, etc.) an increase in the corruption cases and a constant careful monitoring of the situation is required.
We have deployed already several means to control the situation, but more probes and monitoring needs to be done. The involvement of the experiment application level is absolutely necessary.

The implementation of all these means will lead to a doubling of the original required IO performance on the disk servers and also needs an increase of the available CPU capacity on the disk servers (50% ?!). This will of course have an influence on the costing and sizing of the CERN computing facility.

We have also to continue to investigate the causes of all these errors and have constantly monitor the systems. All this will of course create additional operational load on the people.

The effort to cope with this problem has to start right now.