CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Laser Interferometer Gravitational Wave Observatory (LIGO) Project

To/Mail Code:
From/Mail Code: Daniel Sigg
Phone/FAX:
Refer to: LIGO-T000108-00
Date: October 2, 2000

Subject: **Frame Performance Investigations and Recommendations**

Summary:
The way frames are currently written to file is inefficient—both in respect to disk space usage and speed in reading channel data back. The main culprit seems to be the overhead associated with reading 'small' amounts of data. With todays fast RAID systems block sizes have to be of order MByte to reach optimal performance. By increasing the frame size speed improvements between one and two order of magnitudes could be observed. Furthermore, in-frame data compression becomes much more attractive.

Recommendations:
- Increase the size of frame files by increasing the length of each frame:
  full frames → 128 sec, second-trend frames → 1 hour, minute-trend frames → 1 day.
  Reduced data sets working with decimated time series may have to use 1024 sec frames.
- Use compressed data vectors within the frames by default.
- Update all frame readers to use the TOC (table of contents), at least for partial requests.

An additional advantage of longer frame files is that they fit more naturally into the archive structure. With a data rate of 6MB/s (LHO) and a frame length of 128 sec full frames will become about 400MB large (assuming a compression factor of 2); the size of trend frames are estimated to become of order 120MB (second-trend) and 60MB (minute-trend).

Speed Issues:
This investigation was triggered by the fact that reading old data from disk through NDS typically performed at a speed of 10 times real-time. This was essentially independent on the number of channels and data rates as long as the part which had to be read was small compared to the full frame size. This was despite the fact that the frame reader has been improved by caching the frame structure between successive reads so that only the requested data had to be read from disk. One would typically expect read speeds of several MB/s from a stripped disk system, whereas the above numbers would indicate an effective throughput of a few 10kB/s only.
As a consequence the following tests were performed on the fortress machine reading from the internal RAID5 disk system: (1) read a single channel from a series of frame files each of them 1 second long—present situation, (2) read a single channel from a series of frames concatenated together into a single file, 60 at a time, and (3) read a single channel from a series of frame files

each of them 30, 60, 120 or 240 seconds long. Each test typically read from 1–2 hours worth of frame files which were written at 1.7MB/s containing about 60 channels each. The test was using the newly introduced table of contents to accelerate frame access. The test program worked as follows: make a list of all frame files in a directory, map the first file into memory, read the seekTOC value, jump to the TOC and read it, loop over the supplied channel list and copy the data into a float array (look up their start address in the TOC, skip the ADC structure, read the vector structure, read the data, convert to float if necessary), continue with the next frame file until finished. The listed execution times do not include the first step of reading the list of files. Test 3 was performed with the same frames as test 2, since I don't have a tool which can create longer frames. Test 3 read the TOC as test 2, but then only read data from the first ADC channel instance using 30, 60 120 and 240 times the number of data points (this will read invalid data but should be representative what speed improvements can be expected from longer frames). Special care was taken to ensure that file access was not cached. I am pretty sure that the main data was not read from cache, however, there is no guarantee that some (small) overhead, such as the directory containing the frames, was not cached. The column headings are as follows: column 1 – test number, column 2 – sampling rate and byte-per-sample, column 3 – number of files, number of frames per file, length of frames in sec, column 4 – total measurement interval, column 5 – size of read data, 6 – time it took to read the data, and column 7 – effective data rate.

test	chn f_S/ bps	files #/frms/ f_dt	T	size [MB]	dt [sec]	rate [kB/s]
1	-	3600/1/1	1h	TOC	39	
1	16k/4	3600/1/1	1h	225	134	1700
1	2k/4	3600/1/1	1h	28	125	230
1	256/2	3600/1/1	1h	2	78	23
2	-	117/1/60	1h	TOC	1.1	
2	16k/4	117/60/1	2h	439	130	3500
2	2k/4	117/60/1	2h	54	127	440
2	256/2	117/60/1	2h	3	58	61
3 (30)	16k/4	117/1/30	1h	219	12	19000
3 (30)	2k/4	117/1/30	1h	27	4.4	6400
3 (30)	256/2	117/1/30	1h	2	2.4	730
3 (60)	16k/4	117/1/60	2h	439	19	24000
3 (60)	2k/4	117/1/60	2h	54	6.6	8500
3 (60)	256/2	117/1/60	2h	3	2.2	1600
3 (120)	16k/4	117/1/120	4h	878	34	26000
3 (120)	2k/4	117/1/120	4h	110	10.0	11000
3 (120)	256/2	117/1/120	4h	7	3.4	2100
3 (240)	16k/4	117/1/240	8h	1755	60	30000
3 (240)	2k/4	117/1/240	8h	219	13	17000
3 (240)	256/2	117/1/240	8h	14	4.0	3500

Bottom line: Frames have to be significant larger than 1 second, otherwise the read rate from high performance disk systems is limited by overhead.

Space Issues:
Some numbers to estimate the required space for a trend archive. Bottom line, we have to turn frame compression on. Here are the numbers (simply using gzip on the files):

	size	compressed	ratio
full frames	10858M (1h)	5355M	49%
second frames	3652M (1d)	682M	19%
minute frames	71M	19M	27%
minute CDS fmt.	25707M (all)	5796M	23%

As far as I understand the frame format compression is implemented on data only, meaning headers will not be compressed. Using FrDump I conclude that the overhead for each channel is about 160 Bytes (~95 B for FrAdcData and ~65 B for FrVec). Assuming a compression factor of 5, assuming each data point takes 6 Bytes (float/doubles mixed), and requiring that the frame overhead must be less than 10%, >1300 data points have to be included in each frame vector structure. This means second frames have to include an hour or more, whereas minute frames have to include a day or more, and compression has to be turned on. We currently have 1000 channels as part of the trend. Assuming this number will increase to 2000 channels/ifo when the system is completed, the data rate becomes (assuming 5 vectors are needed per trend channel; mean, min, max, rms, n):

second frames:	1GB/d/ifo	30GB/m/ifo	350GB/y/ifo
minute frames:	17MB/d/ifo	500MB/m/ifo	6GB/y/ifo

So, in principle it should be possible to keep second trend on disk forever, if you can afford to buy 1–1.5TB disk space per year.

Tape Issues:
While backing up data from fortress to the AIT-2 robot, one would typically use a command like this: "tar -cf /dev/rmt/0bn ....". Data throughput numbers of 2–3MB/s write and slightly more for read were achieved this way. Changing the block size to 1024 (512kB) instead of the default 20 (10kB) leads to greatly improved throughput numbers, i.e., "tar -cbf 1024 /dev/rmt/0bn ....". For example: write: compressed 50GB (32GB uncompressed); 110' => 7.7MB/s (4.8MB/s raw); read: same as before, 90' => 9.5MB/s (5.9MB/s raw); and write uncompressed 36GB; 100' => 6MB/s. Contrary to what is written in the man pages, the block size during read doesn’t have to match the one used during write. It seems that the block size only affects system and SCSI overhead, but not the way data is written to tape.

cc:
LDAS group CDS group GDS group

Document Control Center