CMST SI(1) 29-32 (2006) DOI:10.12921/cmst.2006.SI.01.29-32

COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY **Special Issue 2006**, 29-32

Performance Evaluation of the CXFS File System on the HPC/Storage Complex for data-intensive computing at the TU Dresden

Michael Kluge

Center for Information Services and High Performance Computing
Technische Universität Dresden, Dresden, Germany
Michael.Kluge@tu-dresden.de

(Rec. 14 September 2006)

Abstract: In the 3rd quarter 2005 the Technische Universität Dresden started to install the first phase of a new SGI supercomputer and a PC farm build by LinuxNetworX. The final phase of the installation will take place in July/August 2006. One main focus of the installation is to have not just outstanding I/O-Performance for the disk to memory channel but for the tape to disk channel too. In this paper we will show how our setup looks like and what our expectations for the performance are. We will also present some measurements done on the system and show the performance boundaries of the current setup.

Key words: Parallel File Systems, CXFS, data-intensive computing

1. INTRODUCTION

The Technische Universität Dresden is in the process of installing a new supercomputer for data intensive computing. The new system has been set up to provide a well balanced mixture of a classical HPC component and a PC farm, both connected to a SAN with high throughput. The idea is to have a HPC component with at least 4 TB main memory and to be able to fill that memory within 10 minutes. The requirements for the PC farm are mainly driven by the demands from our users which have lots of jobs that are in the range of one to eight CPUs.

Within this paper we will show how our I/O setup for the new installation has been planned for fall 2006 and how it looks in spring 2006. We will then show results from some benchmarks we ran on the preliminary production system, especially the maximum I/O bandwidth with large and small strides and different numbers of processes as well as how many create/unlink operations per second our setup can handle.

2. I/O SETUP AT TU DRESDEN AS PLANED FOR FALL 2006

The installation has split up into three part. The first part has been installed at the end of the 3rd quarter 2005. The second and the third part will be installed in July/August 2006. An overview of the I/O demands of the final installation is given in Fig. 2. CXFS [1] will be used as the file system for the HPC component, an SGI Altix 4700 system. DMF [2] will handle the transfer between the CXFS file system and the tapes. The home directories from the HPC component will be visible on the PC farm too via several NFS servers. The central file system on the PC farm will be Lustre [3].

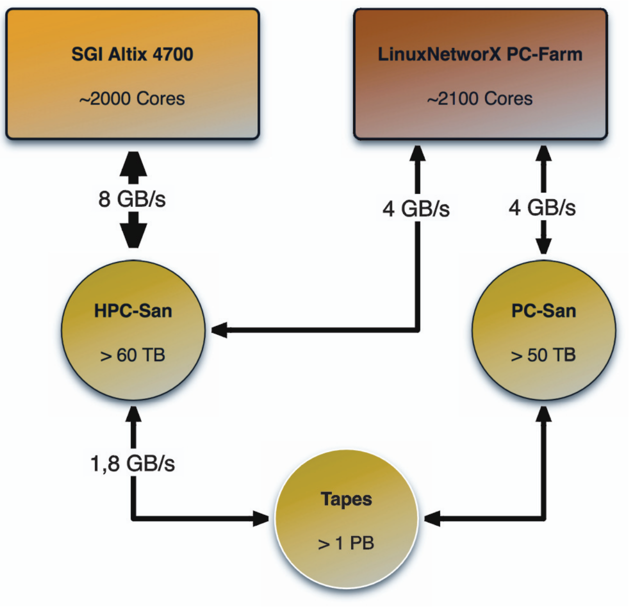

Fig. 1. The I/O setup at TU Dresden as planned for fall 2006

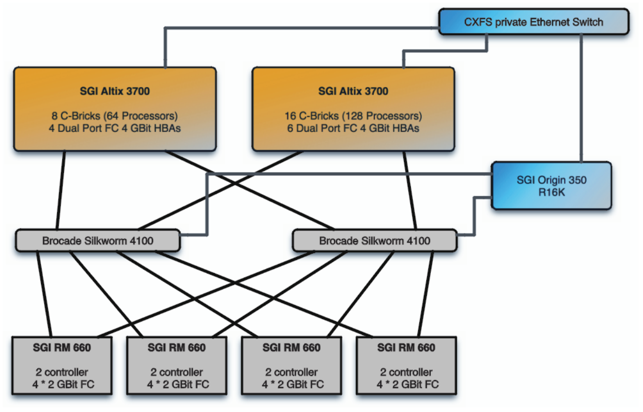

Fig. 2. The I/O setup at TU Dresden / May 2006

Table 1. Key components in the final installation phase

Part	Components
HPC component	1024 Intel Montecito (2048 Cores)
PC-Farm	728 Nodes, 2592 Cores AMD Athlon
HPC SAN	5 couplets SGI InfiniteStorage 6700 Controller; 68 TB Disks
PC SAN	3 couplets SGI InfiniteStorage 6700 Controller; 50 TB Disks
NFS Gateway	12 SGI Altix 350 (2 * Intel Itanium II)
MDS/DMF Server	1 SGI Altix 350 (24 * Intel Itanium II)
CXFS interconnect	GBit Ethernet
FPGA	1 Blade SGI RASC 100

An overview of the components to be built into the final system is given in Table 1.

3. CURRENT (I/O) SETUP AT TU DRESDEN

The current (spring 2006) setup in Dresden for the SGI Altix 3700 machine consists of 192 Intel Itanium II processors running at 1.5 Ghz. The machine is divided into two partitions, one (merkur) has 64 and the second (venus) 128 processors. Each partition has one IX- and one PR-Brick. The PR-Bricks are equipped with four dual port 2GBit FC adapters. The venus partition has two additional dual port FC adapters. The DDN Raid system is installed in two racks. Each rack has two pairs of SGI's RM 660 raid controllers where each controller has four 2 GBit host adapters. These two parts, the Altix System and the RM 660's are connected using two Brocade Silkworm 4100 switches in a redundant setup. A picture of the installation is given in Fig. 3. So, the theoretical bandwidth from the RM660 altogether to the FC-Switches is 7.45 GByte/s. The theoretical bandwidth between the FC-Switches and merkur is 1.86 GByte/s and the bandwidth to venus is 2.79 GByte/s. The major acceptance criteria for the I/O subsystem was to deliver a I/O bandwidth of 2.7 GByte per second when reading and writing files in large chunks. A benchmark especially for this purpose has been developed at the TU Dresden. The bandwidth actually measured was around 2.8 GByte/s in October 2005. The whole two RM 660 racks delivered one large file system for this acceptance test. The heterogeneity in terms of number of CPU's per partition (64 vs. 128) and the fact, that the number of bytes written from each processor is constant, was the biggest challenge here.

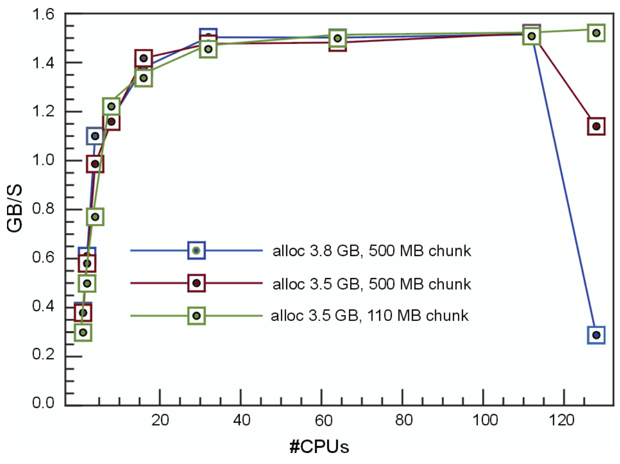

Fig. 3. Write performance, each process writes to its own file large chunks of data (measurements done by Ulf Markwardt)

For the production environment, the following setup is used (where each LUN provides 1.1 Terabyte disk space):

- fastfs: 24 LUNs, 6 from each controller pair,
- work: 4 LUNs, 1 from each controller pair,
- archive: 1 LUN from the first controller.

4. MAXIMUM I/O BANDWIDTH

Maximum I/O bandwidth is only achievable by reading and writing files in large chunks. When small chunks are used, too many overhead due to the operating system and operations on the MDS are involved. Measurements done in two different environments will be shown next. The first data we got on the setup used for the acceptance test (all controllers create one file system), the second part of the data are gathered on the production system (as described above).

One experiment done on the big filesystem used during the acceptance test was to allocate per processor the whole amount of free memory and to write large chunks of data to the disks. As it can be seen in Fig. 4 the performance will drop when this benchmark uses all available processors. This behavior is due to the Linux kernel. When an application allocates almost every byte of the available free memory nothing is left for kernel buffers. But when large chunks of memory are written onto disk then the kernel needs memory at the size of this large chunk. At this time the machine will start to swap pages from the user space to disk and the performance observed for the benchmark drops.

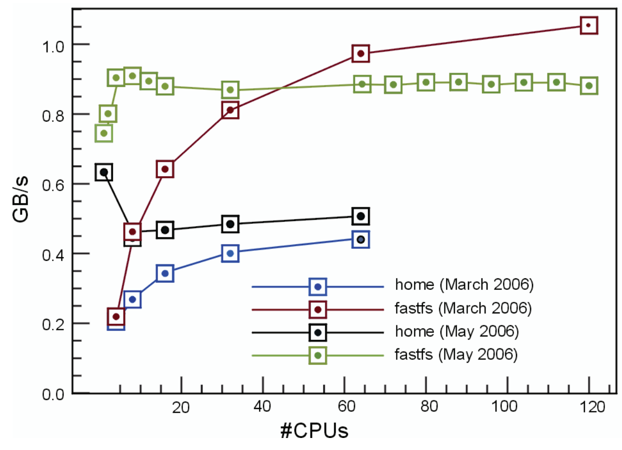

Fig. 4. Write performance to different file systems with different configurations

On the production system we have also tested our CXFS file systems by having each process writing large chunks of data to its own file. We have varied the number of processes used and the total file size written by each process (7 GB) as well as the chunk size written by each call to fwrite() (500 MB) has been kept constant. The results are given in Fig. 4. For each of the two used file systems (fastfs and home) two curves are shown. Between the curves labeled with 'March' and the curves labeled with ‘May’ the FC cables between the two ports on the two additional FC HCAs in the venus partition have been exchanged. Those wrong cabling has had an big impact on the I/O performance of applications that uses a small number processes to read data. After correcting this one single task is now able to read data with about 700 MB/s from fastfs. The experiments with smaller number of processors does benefit significantly from the large main memory. The OS is able to buffer most of the I/O within the local memory.

5. I/O BANDWIDTH ON A SINGLE FILE

One of the applications running on the Altix system is GeneIndex [4]. GeneIndex indexes and matches patterns in DNA sequences and is used in inference of amino acid sequences. The interesting part of GeneIndex from an I/O point of view is the way how the input file (approx. 1 GB) is read. It is read line by line (80 bytes) with buffered I/O. For a single task about 80 MB/s can be reached for reading this file, which is about 15% of the maximum performance when reading a file with one task in large chunks. One idea to improve the performance here was to split up the reading process to different processes, where each tasks read a contiguous part of the file. The bandwidth with an increasing number of processes reading from the 1 GB file is shown in Fig. 5. The problem with an increasing number of processes is that the portion of the file read by each process decreases. With 32 processes each process reads just 30 MB from the file. This seems not to be enough to get good performance. The check this assumption, we have increased the file size step by step up to 20 GB. As one can see, the maximum bandwidth for this scenario is about 1.2 GByte per second an can never be reached with the 1 GB file. To see the influence of the chunk size on the performance the file size had been fixed with 1 GB and the chunk size varied between 80 bytes and 16 KBytes. From Fig. 5 we can take as an conclusion, that with 32 processors the 30 MB to each processor will never be read faster than with 600 MB/s. By using larger chunk sizes (but still small ones compared to the 500 MB used in section 4) we do not see any significant increase in the performance. Only really large chunk sizes seems to have an impact here.

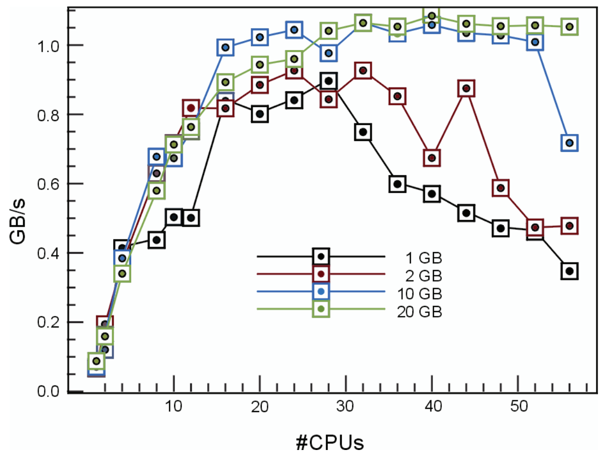

Fig. 5. Read performance, reading a single file (1 GB) with a chunk size of 80 bytes and different number of processors

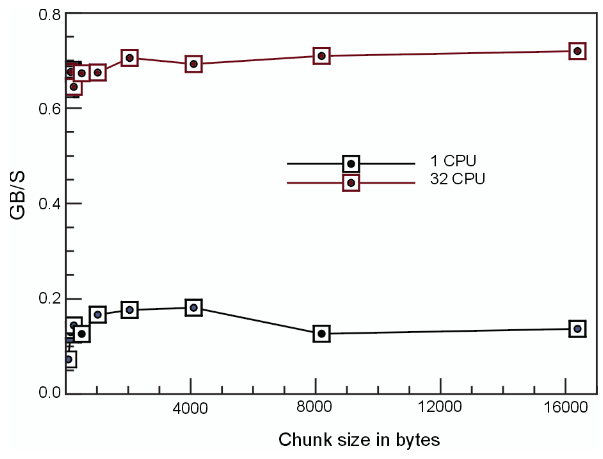

Fig. 6. Read performance, reading a single file (1 GB) with a various chunk sizes between 80 bytes and 16K with one and 32 processes

6. CREATE/UNLINK BENCHMARK

One other benchmark figure commonly used is the number of files that can be created and removed within a certain amount of time. This number is commonly re ferred to as the operations per second for a file system.

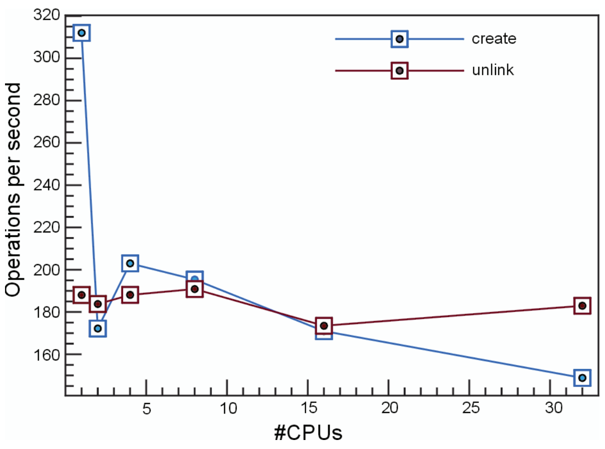

Fig. 7. create/unlink operations per second in the CXFS file system

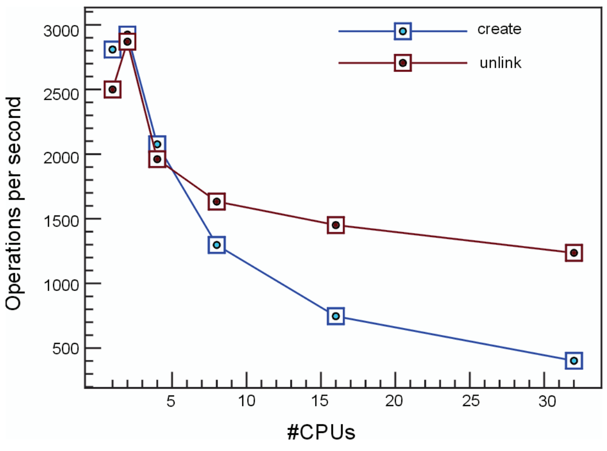

Fig. 8. create/unlink operations per second in the XFS file system

Under CXFS (and any other file system that uses an MDS) this benchmark should show the performance of the MDS as well as scalability issues. Our setup for this test is toopen 100 000 files in a single directory and to remove them afterwards. For this test the number of processes used have been changed between 1 and 32. At first we would expect here (Fig. 7) a number of 3000+. This is an average number usually seen on other sites and other parallel file systems utilizing and MDS. As the MDS should be the bottleneck here, we have no explanation for the increase of the run-time of the benchmark (and the decrease in the operations/second) other than that the virtual LAN that is used for the communication between the MDS and the clients needs to be improved by a dedicated switch.

7. CONCLUSION

Especially being able to read and write with 700 MB/s with a single client is very useful for application like Gaussian that often write large checkpoint files from a single process. For compilation processes the file creation rate of max. 300 operations per second might be too less. We have not yet amply tested our DMF configuration which will be done in fall 2006.

References

[1] L. Shepard and E. Eppe, SGI InfiniteStorage Shared Filesystem CXFS: A High-Performance, Multi-OS File-system from SGI. Technical report, SGI (2004).

[2] L. Shepard, SGI InfiniteStorage Data Migration Facility (DMF); A New Frontier in Data Lifecycle Management. Technical report, SGI, 2004.

[3] Cluster Filesystems Inc. Selecting a Scalable Cluster File System. Technical report, Cluster Filesystems Inc., Nov 2005.

[4] Craig A. Stewart, D. Hart, R. Sheppard and R. Cruise, *Parallel computing in biomedicine: current examples and the search for peta-scale applications.* In: G. R. Joubert *et. al.* (eds.) Advances in Parallel Computing, **13**, 719-726, Elsevier (2004).

MICHAEL KLUGE holds an M.Sc. in Information System Technologies from the TU Dresden. He is responsible for performance analysis in the SAN environment.