European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies
A Network of Excellence funded by the European Commission

Practical evaluation of custom technology-based vs commodity technology-based Storage Elements

Maciej Brzezniak and Norbert Meyer
Poznan Supercomputing and Networking Center
61-704 Poznan, Noskowskiego 12/14, Poland
Email: {maciekb,meyer}@man.poznan.pl

Michail Flouris and Angelos Bilas
Institute of Computer Science (ICS)
Foundation for Research and Technology – Hellas (FORTH)
P.O. Box 1385, Heraklion, GR-71110, Greece
Email: {flouris,bilas}@ics.forth.gr

CoreGRID Technical Report
Number TR-0168
July 21, 2008

Institute on Knowledge and Data Management

CoreGRID - Network of Excellence
URL: http://www.coregrid.net

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

Practical evaluation of custom technology-based vs commodity technology-based Storage Elements

Maciej Brzezniak and Norbert Meyer
Poznan Supercomputing and Networking Center
61-704 Poznan, Noskowskiego 12/14, Poland
Email: {maciekb,meyer}@man.poznan.pl

Michail Flouris and Angelos Bilas
Institute of Computer Science (ICS)
Foundation for Research and Technology – Hellas (FORTH)
P.O. Box 1385, Heraklion, GR-71110, Greece
Email: {flouris,bilas}@ics.forth.gr

CoreGRID TR-0168
July 21, 2008

Abstract

Scalable and cost-effective Storage Elements are essential components of Grid systems. An increasing number of data-intensive applications and services in Grids make the cost-effective scalability of capacity and performance of the storage infrastructure a very hot research topic. In this paper, we present the practical evaluation of the performance features of two classes of storage systems. We tested an example custom technology-based storage system, the FC-SATA disk matrix and a commodity-based storage solution, the Violin system, developed at FORTH. Using a block-level benchmark, we examined the performance limits and the scalability features of both classes of systems.

1 Introduction

Scaling the capacity and performance of Storage Elements while reducing their costs is an important issue in Grid systems, due to the increasing number of data-intensive Grid applications and services. Storage Elements traditionally rely on DAS (Directly Attached Storage), NAS (Network Attached Storage) or SAN (Storage Area Network) architectures. To satisfy application requirements on performance and capacity scalability, traditionally Grid Storage Elements are built using specialised, aggressive SAN-based storage systems. At the same time, the DAS and NAS-based storage become less popular mainly due to their capacity and performance scalability limits.

Custom, SAN-based storage systems are to a large extend centralised solutions. Multiple disk drives are typically connected to one or more central matrix controllers, which processes I/O requests coming from the user applications and implements various storage virtualisation functions, such as RAID structures and snapshots. This centralised, controller-based architecture facilitates providing strong reliability guarantees as well as simplifies system management, including planning, deployment and day-to-day maintenance. However, centralisation induces scalability limitations in terms of both capacity and performance.

Another important feature of custom storage systems is that they use specialised devices, optimised for I/O processing purposes. Similarly, the communication protocols used in these systems (mainly the Fibre Channel protocol)

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

are designed for carrying I/O traffic through a dedicated, low-latency SAN. This guarantees high performance and reliability of particular components and the whole system operation, however results in high cost both due to technology and market reasons.

An emerging trend is to exploit numerous, low-cost, commodity-based components for building scalable storage systems. In this approach, storage nodes built out of dozens of disks connected to PCs act as storage controllers. They are interconnected with a general-purpose, low-cost network, e.g. 1-10 GBit/s Ethernet. Key features of this approach are (1) the de-centralisation of storage I/O processing and (2) the ability to follow technology curves for general-purposes systems (CPUs, memory and interconnects) and thus, achieve high cost efficiency. However, several challenges must be addressed before commodity-based storage systems are used transparently in real applications. Open issues include seamless integration, interoperability, automated management tools, strong reliability guarantees over a distributed system, and robust security mechanisms in de-centralised architectures.

In our previous work [3][2] we have performed a qualitative analysis of the two approaches (FC-based and commodity-based systems). Our results show that commodity storage systems can scale both capacity and performance at lower cost compared to custom systems, mainly by using larger number of components.

In this work we examine quantitatively the performance of two such systems. We use two setups, an FC-based setup and a commodity-based research prototype and measure block-level performance using a simple micro-benchmark. We find that performance of processing the sequential and random I/O in both examined systems is similar.

The rest of this paper is organised as follows. Section 2 discusses the methodology we use in our evaluation. Section 3 presents our experimental results and their analysis. Section 4 discusses related work. Finally, Section 5 draws our conclusions.

2 Methodology

The two systems we use in this work are: (a) Nexsan SATABeast [9], FC-to-SATA matrix that belongs to a low-end segment of custom storage (b) The Violin research prototype [5] of commodity-based storage components.

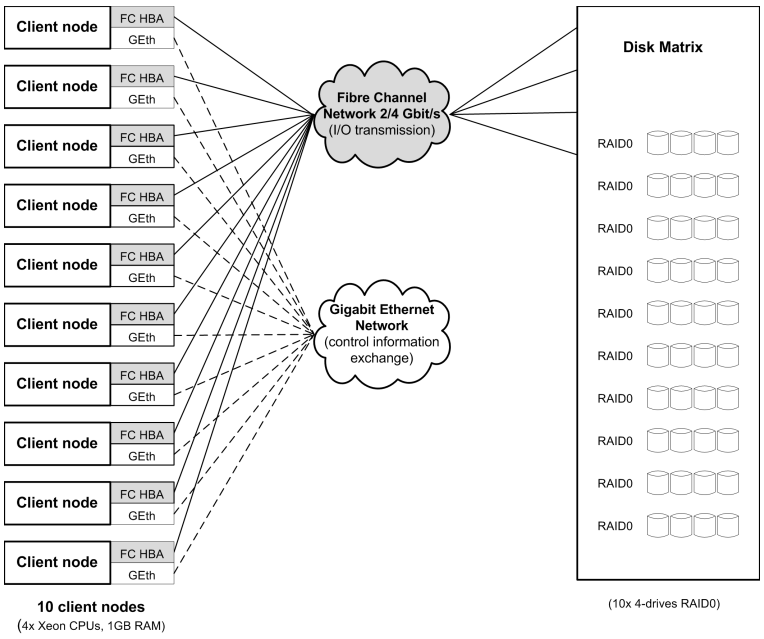

Figure 1: Custom technology-based storage testbed

Figure 1 presents the configurations of the custom storage test-bed. It contains the Nexsan SATABeast matrix, equipped with 42 SATA drives (500 GB of capacity each, 40 drives used for testing), two RAID controllers (each has 1 GB of cache memory), and 4 Fibre Channel front-end ports. The matrix is connected to a 4 Gbit/s FC switch using four FC links working in 2 Gbits/s mode (due to a matrix ports speed limitation). As storage clients we use ten PCs, each equipped with four Xeon 3GHz CPUs, 4GB of RAM and 1 Fibre Channel 4Gbit HBA. A separate Gigabit

Ethernet network is used for coordinating the benchmark operation in client nodes. The matrix operation mode is set to active-active (both controllers active). We enable cache mirroring between controllers and set cache optimisation policy to *mixed sequential/random.*

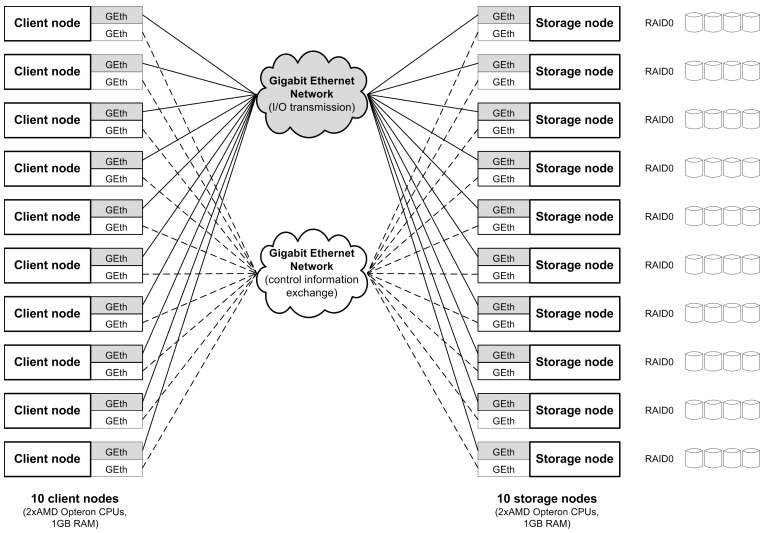

Figure 2: Custom technology-based storage testbed

The commodity storage test-bed is shown in Figure 2. It contains ten storage servers, each equipped with 4 SATA drives, controlled by Violin, a software-based block-level virtualisation layer. As storage clients we use ten PCs equipped with two AMD Opteron 2 GHz CPUs and 1 GByte of memory. Servers and clients are interconnected through two separate, dedicated Gigabit Ethernet networks: one for I/O requests and one for exchanging control information among Violin nodes and between storage clients. All systems used in both test-beds run Red Hat Enterprise Linux Server system (release 5).

In this work, we have evaluate the performance of RAID10 structures. We configure RAID1 arrays on every 4 drives in the FC matrix. Similarly, we configure 4-drive RAID1 structures on each Violin-based storage node. We then stripe the RAID1 arrays (10 for each test-bed) on the client side using the Linux MD mechanism (RAID0). Thus, each system is configured with five 8-drive RAID10 structures.

In each test-bed we use a load-balancing mechanism: In the FC setup, distribution of I/O requests over matrix controllers and front-end links is guaranteed by an appropriate assignment of RAID arrays to matrix controllers and addressing the logical volumes on the client side. In the Violin setup, load-balancing of the storage nodes is performed in the Violin middle-ware, both on the server and the client side.

In our micro-benchmark tests we use `xdd` [6] for generating the test I/O traffic, including both sequential and random reads and writes. In our experiments we vary the number of outstanding IOs per client (from 1 to 8) and the IO request size (between 4KB-1024KB). We run `xdd` in direct I/O mode, where requests are sent directly to a block device, by-passing the file system layer. This eliminates potential influence of the Linux file system caching on the test results.

3 Results

Figure 3 shows our results from both our experimental setups. Throughput values refer to the overall achieved throughput as measured on the client side. Each curve refers to different parameters.

The results of sequential workloads show that performance trends are similar in both classes of storage systems. Sequential I/O throughput grows with increasing block size and, in most cases, reaches its maximum for block sizes larger than 512kB.

Sequential write throughput in the FC-based setup is limited to a maximum of about 150MB/s. This limit is caused by internal the connectivity bottleneck of the matrix. Non-matrix-related reasons for the observed bottleneck in the FC-based setup are eliminated by checking CPU loads in the client machines as well as monitoring the distribution

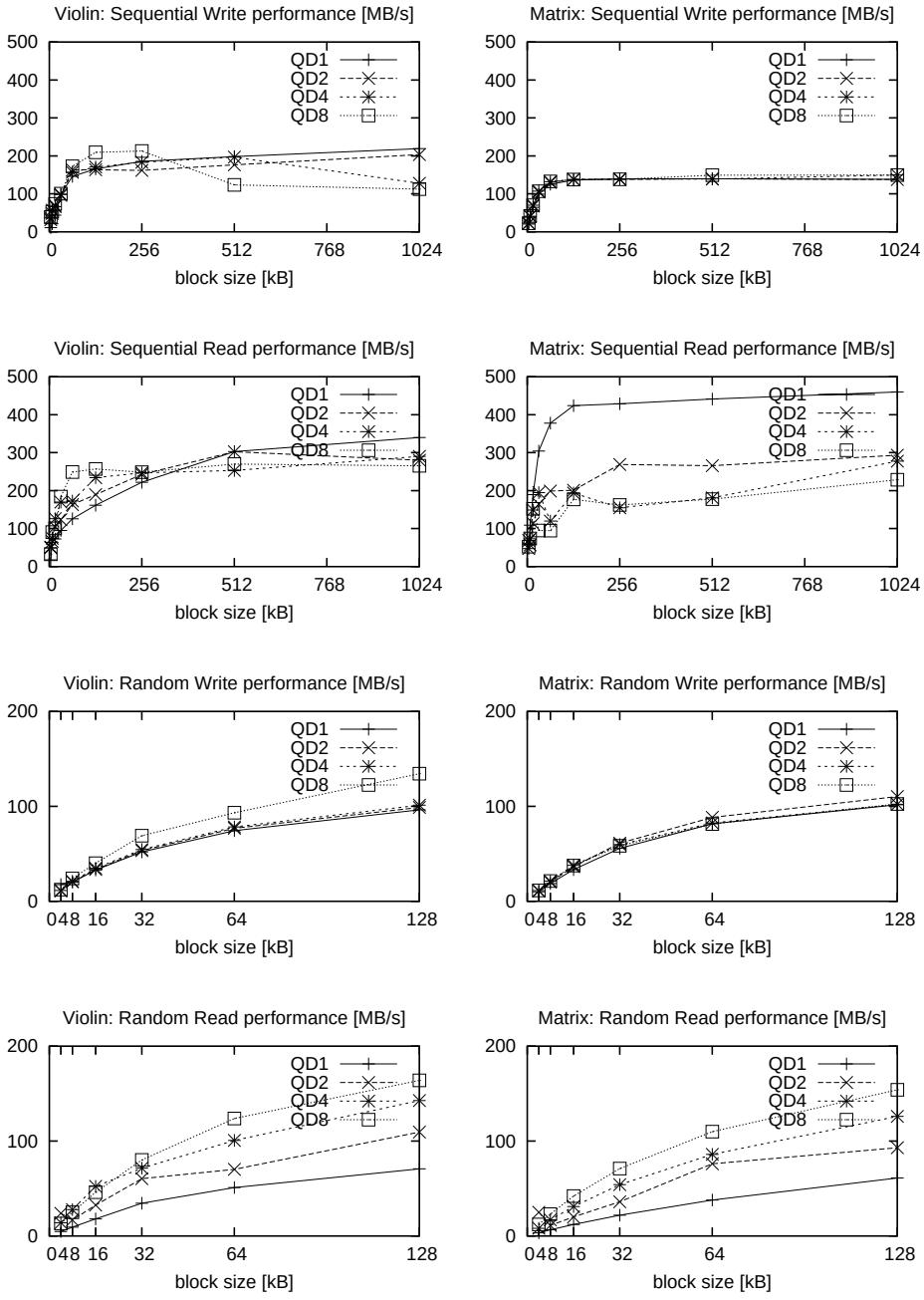

Figure 3: Performance results from the two experimental setups.

of the I/O traffic among the client-matrix FC links and the balancing of the load of the matrix controllers. Opposite, the commodity-based storage system benefits from its high controller-disks throughput, available due to the system decentralisation and the fact that it has multiple, independent RAID controllers. Violin's write performance reaches a maximum of 219MB/s for small queue depths, which is better than the matrix result. However, for high I/O queue depths and large block sizes the benchmark performance drops under 150MB/s (block size larger than 512kB for QD=4 and the block size larger than 256 kB for QD=8). These irregularities result from the implementation problems in Violin.

On the other hand, the centralised architecture of the matrix results in effective read-ahead caching for small number of I/O threads, resulting in relatively high read throughput, up to up to 460MB/s. However, for higher number of concurrent I/Os, read operations in the matrix perform similarly (QD=2) or even worse (QD=3, QD=4) compared to the commodity-based setup. This is due to the centralised architecture of the FC-based setup and the inability to distribute I/O traffic among multiple controllers.

Measurements with random I/O requests show similar trends in both classes of storage systems. Results in both systems between 4K and 128K request sizes are within 10% in most cases. The main trend is that small blocks sizes result in poor performance in both setups, and performance improves with request size. Random read performance drops under 100MB/s for block sizes smaller than 64kB in most our tests. Overall, in most random I/O tests, the commodity-based setup performs slightly better than the FC-based setup.

4 Related work

Traditionally, building scalable storage systems that provide storage sharing for multiple applications relies on layering a distributed file-system on top of a pool of block-level storage, typically a SAN. This approach is dictated by the fact that block-level storage has limited semantics that do not allow for performing advanced storage functions and especially they are not able to support transparent sharing without application support. Recently, there has been a lot of research work in enabling, cluster-based networked storage systems.

In our previous work [3] we indicated potential performance bottlenecks of custom, centralised storage architectures. We also examined the potential of commodity-based systems in terms of performance scalability, due to the distributed nature of their architecture. Our cost analysis suggests that the price-performance ratio is significantly better for commodity-based storage than for FC-based systems. This work performs a preliminary performance evaluation of the two base approaches.

Efforts in this direction include distributed *cluster file systems* often based on VAXclusters [7] concepts that allow for efficient sharing of data among a set of storage servers with strong consistency semantics and fail-over capabilities. Such systems typically operate on top of a pool of physically shared devices through a SAN. However, they do not provide much control over the system's operation. Modern cluster file-systems such as the Global File System (GFS) [12] and the General Parallel File System (GPFS) [11] are used extensively today in medium and large scale storage systems for clusters. However, their complexity makes them hard to develop and maintain, prohibits any practical extension to the underlying storage system, and forces all applications to use a single, almost fixed, view of the available data. The Federated Array of Bricks (FAB) [10] discusses how storage systems may be built out of commodity storage nodes and interconnects and yet compete (in terms of reliability and performance) with custom, high-end solutions for enterprise environments. Ursa Minor [4], a system for object-based storage bricks coupled with a central manager, provides flexibility with respect to the data layout and the fault-model (both for client and storage nodes). These parameters can be adjusted dynamically on a per data item basis, according to the needs of a given environment. Such fine grain customisation yields noticeable performance improvements.

Previous work has also investigated a number of issues raised by the lack of a central controller and the distributed nature of cluster-based storage systems, e.g. consistency for erasure-coded redundancy schemes [1] and efficient request scheduling [8].

5 Conclusions

Traditionally, scalable storage systems are being built using custom, SAN-based technologies. An alternative approach is to use commodity components interconnected with commodity networks, such as 1 or 10 GBit/s Ethernet and provide storage virtualisation functionality in the software I/O path. In this work we perform a preliminary evaluation of the two approaches. Although systems perform comparably, there are certain observations to make: We see that in

the custom, FC-based system, the centralised matrix controller is a major performance bottleneck. At the same time, a heavy communication protocol (TCP/IP), even at the kernel level, is an important factor limiting the performance of I/O in the commodity-based, Violin research prototype. Another aspect of the performance scalability in the compared systems is the granularity of the available scaling options. In the FC-based system only adding matrix controllers or whole matrices can improve performance, which results in poor cost scalability. Instead, in the commodity-based prototype, performance scaling is possible by adding low-cost, commodity storage components, such as disk drives, memory modules, or CPUs.

Overall, given the price difference between FC-based and commodity-based systems (about ten times higher) and the experimental results we observe, we believe that commodity-based systems have a lot of potential in cost-sensitive environments and applications.

In the future, we plan to examine larger setups for both systems. The matrix we examine in this work belongs to a low-end segment of custom storage market; it is implemented in FC-to-SATA technology (FC front-end ports, SATA disk drives). A further step is to examine the mid-range-class disk matrix implemented in FC-to-SATA or in FC-to-FC (FC front-end ports, FC disk drives). Similarly, it is important to examine larger-scale commodity-based system. Finally, we plan to use file system and application-level benchmarks in both classes of storage systems to examine their behavior in more realistic setups.

References

[1] K. A. Amiri, G. A. Gibson, and R. Golding. Highly Concurent Shared Storage. In IEEE, editor, *Proceedings of 20^{th} International Conference on Distributed Computing Systems (ICDCS'2000)*, pages 298–307, Taipe, Taiwan, R.O.C, April 2000. IEEE Computer.

[2] M. Brzezniak, T. Makiela, N. Meyer, R.Mikolajczak, M. Flouris, R. Lachaize, and A.Bilas. An Analysis of GRID Storage Element Architectures: High-end Fiber-Channel vs Emerging Cluster-based Networked Storage. Technical Report 0088, CoreGRID, May 2007.

[3] M. Brzezniak, N. Meyer, M. Flouris, R. Lachaize, and A.Bilas. An Analysis of GRID Storage Element Architectures: High-end Fiber-Channel vs. Cluster-based Networked Storage. In *Proceedings of the CoreGRID Workshop on Grid Middleware. Dresden, Germany. Held with the International Supercomputing Conference (ISC07)*, June 2007.

[4] M. Abd-El-Malek et al. Ursa Minor: Versatile Cluster-Based Storage. In *Proceedings of the 4th USENIX Conference on File and Storage Technology*, San Francisco, CA, USA, December 2005.

[5] Michail D. Flouris, Renaud Lachaize, and Angelos Bilas. A Framework for Extensible Block-level Storage. In *D. Talia, A. Bilas, M. Dikaiakos (eds.): Knowledge and Data Management in Grids*, CoreGRID series, Vol. 3, pages 83–98. Springer Verlag, 2007.

[6] I/O Performance Inc. The xdd. http://www.ioperformance.com/products.htm.

[7] Nancy P. Kronenberg, Henry M. Levy, and William D. Strecker. Vaxcluster: a closely-coupled distributed system. *ACM Transactions on Computer Systems*, 4(2), 1986.

[8] C. R. Lumb, R. Golding, and G. R. Ganger. D-SPTF: Decentralized Request Distribution in Brick-Based Storage Systems. In *Proceedings of the 11th ACM ASPLOS Conference*, Boston, MA, USA, October 2004.

[9] Nexsan Technologies. SATABeast. http://www.nexsan.com/satabeast.php.

[10] Y. Saito, S. Frolund, A. Veitch, A. Merchant, and S. Spence. FAB: Enterprise storage systems on a shoestring. In *Proc. of the ASPLOS 2004*, October 2004.

[11] Frank Schmuck and Roger Haskin. GPFS: A Shared-disk File System for Large Computing Centers. In *USENIX Conference on File and Storage Technologies*, pages 231–244, Monterey, CA, January 2002.

[12] S. Soltis, G. Erickson, K. Preslan, M. O'Keefe, and T. Ruwart. The Global File System: A File System for Shared Disk Storage, October 1997.