

# OS Support for a Commodity Database on PC Clusters — Distributed Devices vs. Distributed File Systems

Felix Rauch<sup>1,2</sup>

Thomas M. Stricker<sup>1,3</sup>

<sup>1</sup> Laboratory of Computer Systems  
Swiss Federal Institute of Technology  
CH - 8092 Zurich  
Switzerland

<sup>2</sup> National ICT Australia (NICTA)  
Locked Bag 6016  
University of New South Wales  
Sydney NSW 1466, Australia  
felix.rauch@acm.org

<sup>3</sup> Google Inc.  
1600 Amphitheatre Parkway  
Mountain View  
CA 94043, USA  
tomstr@acm.org

## Abstract

In this paper we attempt to parallelise a commodity database for OLAP on a cluster of commodity PCs by using a distributed high-performance storage subsystem. By parallelising the underlying storage architecture we eliminate the need to make any changes to the database software. We look at two options that differ in their complexity and features: Distributed devices and distributed file systems. The former aggregates several single disks within the cluster into a RAID device across the network. The latter offers all the features of a real file system at the price of a considerably increased complexity. We configured a Linux version of ORACLE to run on various distributed devices or distributed file systems, respectively, and ran a TPC-D benchmark on our cluster of commodity PCs interconnected by a Gigabit Ethernet. While distributed devices achieve at least the performance of local disks, they offer the benefit of using all surplus storage in a cluster. The distributed file systems seem to run into performance problems due to their increased complexity. We explain the experimental results with an analytic model of the cluster architecture and include a comparison of the same workload on an architecture that distributes the TPC-D queries at a higher level (and not just the underlying storage system). We conclude with suggestions for higher performances in future clusters of commodity PCs.

## 1 Introduction

Modern clusters of commodity PCs are no longer composed of rack-mounted PCs in the basements of compute centres, but comprise the PC nodes on the employees' desks, interconnected by a built-in high-speed commodity network like Gigabit Ethernet. The employees' PC nodes form a more flexible type of a cluster—i.e. a multi-use cluster—and are at the border to the even more distributed computational grids. A typical PC node nowadays contains a large and cheap hard-disk drive which often only stores the operating system (OS) and probably a few applications, while the users' important data is traditionally stored on expensive and centrally-managed specialised file servers. The rest of the disk drives' storage capacity remains largely unused. This unused storage capacity will even increase in the future, as illustrated in Fig-

ure 1<sup>1</sup>. This distributed storage capacity is cheaply available and comes with every node of a multi-use cluster of commodity PCs. Adding hard-disk drives to the nodes of the cluster costs near to nothing, i.e. we can get by with the \$1/GByte pure disk costs and pay almost no infrastructure cost, in particular since the Gigabit Ethernet provided in the cluster can also function as a SAN to carry storage traffic. Given this simple, obvious system architecture, we address the question of how such storage facilities can be used with existing applications.

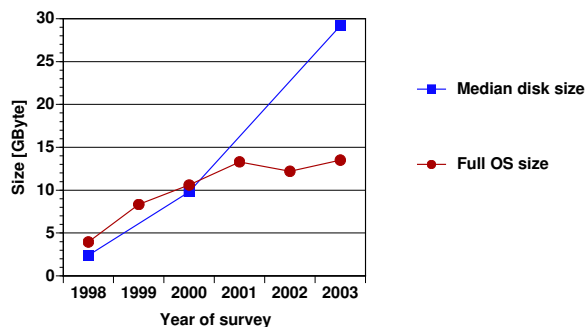


Figure 1: Development of EIDE hard-disk sizes and a full SuSE Linux installation 1998–2003.

The flexibility of multi-use clusters as described in (Rauch 2003) allows to use the free storage capacities according to different usage modes: The excess storage can be used during nights and weekends with no other load on the cluster nodes. This usage mode greatly enhances the utilisation of desktop machines and enables large-scale data mining at almost no additional cost. Alternatively, the storage can also be used while the users work on the cluster nodes, since many typical office and desktop applications have only modest resource requirements.

As one possible use of the surplus storage, we attempt to parallelise a commodity database for OLAP by using a distributed high-performance storage subsystem. By parallelising the storage subsystem, we do not have to make any changes to the database software. We simply distribute the database's data files across the hard-disk drives of the PC nodes in the cluster, thereby increasing their utilisation at no extra cost. Using many disks in parallel also has the potential advantage of using the aggregated throughput of all the drives, which is especially significant for large file reads and should therefore help significantly to speed up large table scans. A small drawback is the slightly increased latency due to crossing the network, which is only significant for small reads. In

Copyright ©2005, Australian Computer Society, Inc. This paper appeared at the 16th Australasian Database Conference (ADC2005), University of Newcastle, Newcastle, Australia. Conferences in Research and Practice in Information Technology, Vol. 39. H.E. Williams and G. Dobbie, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

<sup>1</sup>The hard-disk sizes were extracted from performance-test results of more than 100 consumer disks published by the German consumer magazine "c't Magazin für Computertechnik", issues 1998(14), 2000(16) and 2003(05).

this paper we compare and evaluate different types of distributed storage systems for clusters of commodity PCs as transparent parallelisation techniques for databases running OLAP applications.

The rest of this paper is structured as follows: In Section 2 we discuss related work. In Section 3 we describe distributed storage architectures for clusters of PCs and distinguish between distributed devices and distributed file systems as two options implementing the distribution at different layers in the OS. In Section 4 we discuss our evaluation criteria, describe a simple microbenchmark and reason about our choice of TPC-D as application benchmark. In Section 5 we present our performance measurements. The results are then analysed in Section 6, which also contains suggestions for future high-speed databases on clusters of PCs. Finally, in Section 7 we conclude.

## 2 Related Work

There is much related work in the area of high-performance storage systems for clusters. Other architectures with distributed devices are Petal (Lee & Thekkath 1996) or the Linux network block device (NBD). Petal puts an emphasis on dynamic and transparent reconfiguration as well as load balancing. Both have no special hardware-supported optimisations for higher performance. The Myrinet Distributed Device System MDDS (Rauch 2003) is another prototype system with hardware-assisted optimisations that we developed. It is built with Myrinet gigabit networking technology and uses the programmable NIC and its multiple DMA engines for highly-optimised data transport, but does not satisfy our demand for commodity components like standard Gigabit Ethernet does.

Our specific scenario of using a single off-the-shelf DBMS (like ORACLE) is in sharp contrast to the broader research area of high-performance computing and parallel I/O, where the most interesting aspect of a (parallel) I/O system is the aggregate performance of all nodes running the same parallel application. The exact access pattern depends on the parallel application and can be fine grained or coarse grained and predictable or random. Scientific-computing applications often access data in well defined phases and compute between the I/O phases. More details about recent research results in the area of parallel I/O can be found in (May 2001). According to (May 2001), distributed file systems manage accesses to files from multiple processes, but concurrent accesses are treated as unusual events. On the other hand, parallel file systems used in parallel I/O for high-performance computing are superior and much more scalable. They address several problems far beyond the capabilities of general distributed file systems. In this paper we will focus on the specialised case of a standard DBMS using the parallel I/O facilities of a cluster.

A landmark cluster of commodity PCs with an unprecedented highly attractive cost/performance ratio was built by the Beowulf project (Becker, Sterling, Savarese, Dorband, Ranawake & Packer 1995).

Storage area networks (SANs) provide access to many additional disks over a dedicated additional network for one or more CPUs (Preston 2002). In contrast, distributed devices and distributed file systems for multi-use clusters of commodity PCs use the excess storage capacity of existing disks as well as the already installed commodity high-speed data network.

Other related work includes network-attached (secure) disks (NA(S)D) and network attached storage (NAS). NASDs, described in more detail in (Gibson, Nagle, Amiri, Chang, Feinberg, Gobioff, Lee, Ozceri,

Riedel & Rochberg 1996), are drives that support independent client access to drive-provided object services. They contain a CPU and a small OS to provide client authentication, access-control enforcement, data privacy and data integrity. NAS (Preston 2002) is a device dedicated to sharing files via common network-file-system protocols like NFS or SMB. These systems contain only a minimal CPU to reduce costs, while the nodes of a multi-use cluster are powerful enough to provide the functions offered by NA(S)D and NAS. The multi-use cluster nodes even have enough spare resources for shared use with other applications. Both NA(S)D and NAS require the installation of additional hardware in the network. Our own architecture provides similar services, but uses the economically interesting excess storage of the disks attached to the nodes in the cluster.

Conventional RAIDs (redundant arrays of inexpensive disks, see (Patterson, Chen, Gibson & Katz 1989, Patterson, Gibson & Katz 1988)) are constructed by attaching many similar disks to a server and storing the data striped over the disks including some redundant information (except for RAID level 0 where data is striped without redundant information simply to get highest throughput). The data redundancy allows to reconstruct the data in case a disk fails. In this paper we will not use multiple disks inside a node, but we connect the disks of some nodes together in a “network RAID”. Since our focus is on performance and the understanding thereof, we will only use RAID level 0 which allows to combine the throughputs of multiple (remote) disks at the expense of redundancy—our system does not yet provide redundancy for fault tolerance and is therefore missing on the “R” in classical RAID. But as RAID 0 infrastructure can be extended into RAID 1 and RAID 5, redundancy can be added to our system in exactly the same way without loss of the performance gains.

An emerging technique to store very large amounts of data are Internet storage and grid storage solutions. In these systems, data is stored on servers that are widely distributed over different administrative domains on the Internet. Research in this area encounters other problems such as latency tolerance, ensuring the reliability of data or multi-path data access. An example for a project in this research domain is “Ocean Store” (Kubiatowicz, Bindel, Chen, Czerwinski, Eaton, Geels, Gummadi, Rhea, Weatherspoon, Weimer, Wells & Zhao 2000) with its prototype “Pond” (Rhea, Eaton, Geels, Weatherspoon, Zhao & Kubiatowicz 2003). FARSITE (Adya, Bolosky, Castro, Cermak, Chaiken, Douceur, Howell, Lorch, Theimer & Wattenhofer 2002) is a smaller-scale storage grid targeted towards desktop machines in academic and corporate department settings. It assumes no mutual trust among the client computers and its focus is on availability, reliability and security. In contrast, we focus on performance and the understanding thereof.

## 3 Distributed Storage Architectures for Clusters of PCs

A transparent distributed storage architecture for clusters of PCs can be implemented at different conceptual layers within the OS. Implementations within these layers differ in their transparency and complexity. At the highest layer, the application implements the distribution mechanism itself, which allows to adapt the data distribution to the application, but has the drawback that the application needs to be rewritten or at least adapted. For some classes of applications, like many commercial commodity databases, source codes are not available and this method is

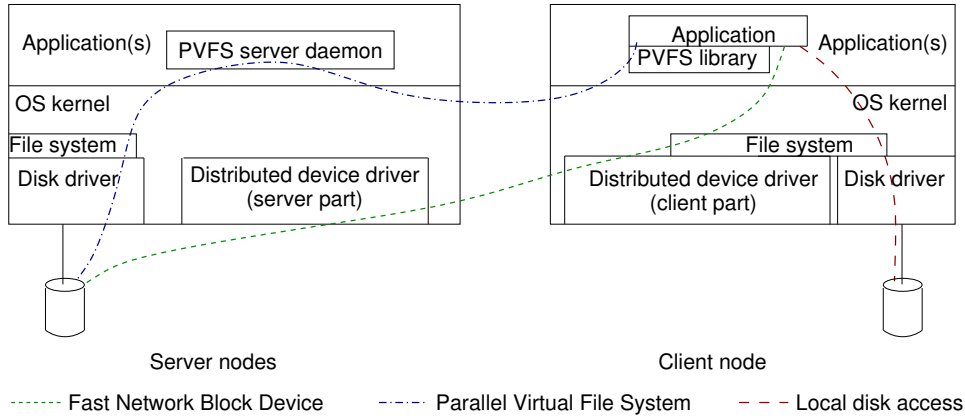


Figure 2: Remote disk access in our two distributed storage architectures for clusters of PCs: Fast Network Block Device (FNBD) and Parallel Virtual File System (PVFS), as well as access to a local disk.

therefore not possible in our scenario. On the next lower layer, the distributed storage is implemented in a dynamically-linked library. The library transparently intercepts the application’s system calls to provide its own implementation of data accesses. In the next lower level, the OS kernel provides a special type of file system. The file system behaves like any other file system, but all file accesses are redirected to some remote servers, which then access their own local file systems to satisfy the data accesses. On the lowest level, below to file-oriented file-system layer, the OS kernel contains a special block-device driver. That driver pretends to offer access to the disk blocks of a local hard-disk drive, but instead maps all disk-block accesses to a remote server which then accesses the respective disk blocks on its own hard-disk drive.

The next subsections briefly discuss the distributed storage architectures used in our experiments and explain their embedding in the OS. More details about the different architectures are in (Rauch 2003). Figure 2 illustrates two architectures as well as a local file access by an application. In the latter case (illustrated by a long-dashed line in Figure 2), the application invokes a system call to the OS kernel to read some file data. The file-system layer then translates the file access to one or more disk-block accesses, which are in turn handled by the hard-disk driver. The driver reads the blocks from the disk and passes the data back to the file-system layer, which possibly keeps a copy of the data in its buffer cache and copies the data back to to application.

### 3.1 Fast Network Block Device (FNBD)

Our own implementation of a distributed-devices system is the Fast Network Block Device (FNBD) which is described in more detail in (Widmer 2001). It transparently maps the blocks of a hard-disk drive’s partition over the high-speed network to another machine. It is implemented as two kernel modules for the Linux kernel. One resides on the client and acts as a block device. It translates any block-requests from higher layers into a request for the server-side kernel module and sends it over the network to one of the corresponding servers. The server part receives the requests and translates it back into a block-request for the server’s disk driver. After the disk read the block, the data travels the same path back to the application on the client.

FNBD is superior to the standard network block device in Linux, as it uses a feature of the commodity Gigabit Ethernet cards to implement automatic defragmentation of incoming packets, thereby eliminating a data-copy operation between the network

stack and the buffer cache. Less copy operations reduce the load on the CPU and the memory subsystem. The defragmentation technique is similar to the one described in (Kurmann, Rauch & Stricker 2001) and can be implemented with many current network-interface cards.

The client has multiple instances of the distributed-device driver, which each map a hard-disk partition of a different server to a local virtual partition. These virtual partitions are then combined with standard OS features into a RAID-0-like array of networked disks. The RAID subsystem stripes the disk blocks on that virtual partition in a round-robin manner over the distributed devices. A file system installed on such a virtual RAID partition therefore sees the aggregated throughput of all the remote disks of the distributed-device system working in parallel.

### 3.2 Parallel Virtual File System (PVFS)

As a distributed file system for clusters we chose the Parallel Virtual File System (PVFS). PVFS is a project in distributed-file-systems research that evolved into a widely used system for PC clusters and is described in more detail in (Ligon & Ross 1999). It provides a distributed file system that can be accessed by any participating node of the cluster. It is implemented as a library which is dynamically linked to applications on startup. The library intercepts standard POSIX system calls and translates the requests to PVFS commands. These commands are then transmitted over the network by using TCP. The server part is a daemon running in user space, which translates the commands to regular file-system accesses to the server’s local file system.

PVFS is a full-featured file system and thus more complex than a distributed device. It does not handle raw disk blocks, but manages files instead. Every file that is stored on PVFS is striped across multiple server nodes. The server nodes store the file-parts as files within a special directory in their local file system. Data is therefore also distributed similar to a RAID 0 system, but managed as files on the server nodes’ disks instead as raw blocks. Applications using the file system should thus ideally be able to see the aggregated throughput of the server nodes’ disks.

### 3.3 Integration with Database

Since both storage architectures described in the previous two subsections are transparent and look like a regular file system to applications, their integration with a commodity database is straight forward:

When installing the database, its data files are simply created on the distributed storage system of the cluster instead of the local disk. For existing databases, the data files can even be copied onto the distributed storage. When the database system boots, it will not see any difference to a file system installed locally on the hard-disk drives. For databases that support only raw disk accesses, only FNBD can be used since it offers direct access to a virtual distributed disk partition, while PVFS works directly at the file system layer and does not offer access to a device.

There are mostly three potential differences a database might notice when it is installed on top of such a distributed storage architecture, as compared to a database with its data files on local hard-disk drives: (a) A higher throughput for large data reads because of the aggregated throughput of multiple combined hard disks working in parallel in the cluster, (b) a higher latency due to the network overhead, and (c) a larger, distributed cache for data files, because the buffer caches of multiple machines are used.

## 4 Evaluation Criteria

Our scenario is specifically targeted at OLAP workloads, which need to read large amounts of data at a high throughput and which do not require any fine-grained write access or fine-grained locking. As benchmarks in our evaluation we use two different workloads: The simplistic micro-benchmark *Speed* and TPC-D as more realistic application benchmark executing in a software system including a commercial Relational Database Management System (RDBMS).

### 4.1 Microbenchmark

A micro-benchmark for our distributed storage architectures must be scalable to large data sets, it must offer read-only experiments, and it must be I/O bound to thoroughly stress the underlying I/O system. *Speed* is such a very simple benchmark developed and used by our masters students. It measures throughputs and block latencies for continuous read or write operations with different user-level block sizes. Latency numbers for continuous read or writes are not so useful in general, because modern systems do read ahead and the latency therefore measures mostly the system-call time. However, in some circumstances the latency numbers helped to find problems with the buffer cache. Since *Speed* fulfills all requirements for a micro-benchmark and was developed in our own laboratory, we use it for the micro-benchmark measurements.

There is a number of other benchmarks for file systems or I/O subsystems. However, they are targeted at NFS servers like *SFS97-R1* (SPEC 2001), test not only the file system but also other OS functionality like *Lmbench* (McVoy & Staelin 1996), use too small working sets for today's data sizes like the Andrew benchmark (Howard, Kazar, Menees, Nichols, Satyanarayanan, Sidebotham & West 1988), offer a very-high scalability like the self-scaling benchmarks (Chen & Patterson 1993) or are focusing on file-system operations on many mostly small files like the Postmark benchmark (Katcher 1997). Finally, *Iozone* (Norcott & Capps 2003) is a file-system benchmark tool that generates and measures a variety of file operations like e.g. read, write, re-read, re-write, read strided, random read and random write. Since we do not need the additional features, we use the *Speed* benchmark.

## 4.2 TPC-D on ORACLE as Application Benchmark

There are a number of widely-accepted benchmarks for database applications standardised by the *Transaction Processing Performance Council* (TPC 2003). Special benchmarks for OLAP applications are the TPC-D, TPC-H and TPC-R (Poess & Floyd 2000). Of these, TPC-D is outdated and has been replaced by TPC-H and TPC-R which both also include updates. Nevertheless we use TPC-D as workload, mostly for historical reasons. We could also use TPC-H or TPC-R without changing the qualitative results of comparing the different storage architectures. Since we are interested in read-only analyses, the older TPC-D will suffice.

TPC-D represents a broad range of decision-support applications that require complex, long running ad-hoc queries against large complex data structures. It determines two metrics: *Power*, which is a statement about the system's capability for a single user, and *throughput* that measures the maximum capability for multiple users. TPC-D consists of 17 different queries. In (Taufers, Stricker & Weber 2002), Taufers et al. describe that some queries of TPC-D in a parallel setup are strictly disk bound, whereas others are CPU or network bound. In this study about the benefits of using distributed storage on clusters of commodity PCs we will put a special emphasis on the queries that are bound by large sequential disk accesses.

## 5 Performance Measurements

The experimental testbed for our performance measurements consists of a 16-node multi-use cluster of PCs called "CoPs". Each node is equipped with two 1 GHz PentiumIII CPUs, 512 MByte of 133 MHz ECC SDRAM and two 10'000 rpm DDYS-T09170M hard-disk drives from IBM with 9.1 GByte storage. As a testbed for high-speed networking the nodes contain two Gigabit Ethernet NICs: A Packet Engines G-NIC II with fiber cables to a Cabletron Smart Switch Router 8600 and an ASANTÉ FriendlyNet (based on the DP83820 chip) with copper cables to a Cisco Catalyst 4000 switch. As OS we use a slightly adapted installation of SuSE Linux 7.1 with a 2.4.3 Linux kernel, and as commodity database an ORACLE RDBMS version 8.0.5.0.0<sup>2</sup>. Unfortunately, the DP83820 chip is badly designed and offers a performance that is much below the full potential of a Gigabit Ethernet network.

### 5.1 Microbenchmarks

As first trivial benchmark we use *Speed* to measure the throughput of a local file system on a single local disk. The resulting throughput for a machine in our CoPs cluster is 33 MByte/s and is independent of the user-level block size because the buffer cache of the file system implements a read-ahead optimisation. The performance of the distributed-devices system (FNBD) initially depended much on the user-level block size of the requests, but increasing the underlying kernel-level request size of the burst-transfer protocol to 32 KByte reduces the the number of network requests and thus the overhead. FNBD now achieves a throughput of 35.7 MByte/s for large-file reads from three servers with a widely-used user-level block size of 4 KByte. For the tests with PVFS we use a standard installation with the PVFS library dynamically linked to the benchmark application. Due

<sup>2</sup>We use this somewhat older version of ORACLE to allow direct comparability of our results with (Taufers et al. 2002).

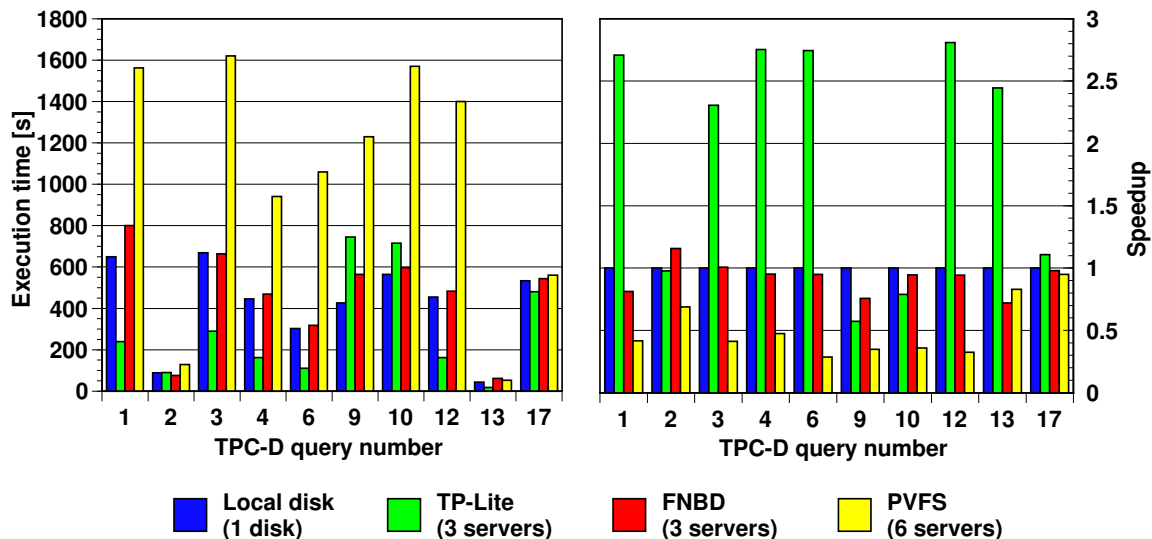


Figure 3: Performance of TPC-D benchmark queries with execution times (left) and speedups (right, relative to a single local disk). All experiments were done on our CoPs cluster comprising cluster nodes with 1 GHz CPUs, Gigabit Ethernet and fast SCSI hard-disk drives.

System name	User-level block size		
	4 KByte	32 KByte	256 KByte
Local disk	32.7	32.4	31.4
FNBD	35.7	37.0	43.1
PVFS	9.8	29.6	36.6

Table 1: Throughput of different storage subsystems according to Speed micro-benchmark in MByte/s. The three tested systems had an Ext2 file system on top of the underlying storage architecture. The nodes have 1 GHz CPUs and are interconnected by Gigabit Ethernet.

to storage constraints with the larger distributed data sets of TPC-D at the time of the measurements, we used six PVFS servers, which bears the possibility of improving the throughput for the client because of higher parallelisation, but should not be a disadvantage. The throughput of PVFS largely depends on the user-level block size. For the typical 4-KByte user-level blocks—which is also the default block size for our ORACLE RDBMS—PVFS reads sequential data with about 9.8 MByte/s and reaches its maximum of 37.1 MByte/s with much larger 128-KByte blocks. The results of the microbenchmarks are summarised in Table 1.

## 5.2 Application Benchmarks with TPC-D

The application benchmarks with TPC-D are executed on all examined architectures and their results are shown in Figure 3 (TP-Lite is introduced and discussed in Section 6). The figure shows the performance for 10 out of 17 TPC-D queries with absolute execution times (left) and speedup numbers relative to a run on a single local disk (right). Queries missing in the figure did not complete on all systems. The results indicate that PVFS has the lowest performance in 9 queries because of its low performance for small user-level block sizes. FNBD surpasses the local-disk system only in query 2 and has otherwise a slightly higher execution time. In the case of query 11 (not shown) FNBD is much faster than the local disk because of the larger distributed buffer cache on FNBD’s servers and thus a reduced number of disk accesses. Contrary to the results of the microbench-

marks, where the large-read throughput of FNBD is higher than a local disk’s, FNBD is slightly slower in almost all queries.

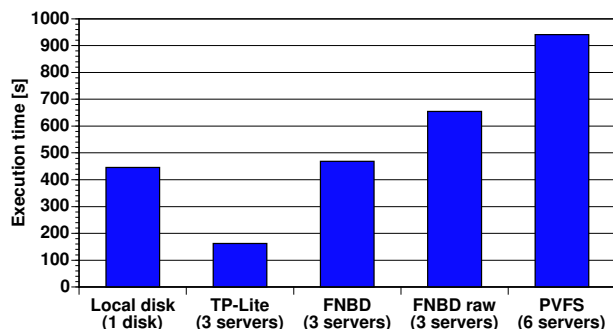


Figure 4: Execution times for disk-limited TPC-D query number four on all examined systems with the ORACLE RDBMS. The nodes are equipped with 1 GHz CPUs and Gigabit Ethernet.

Tauber et al. describe in (Tauber et al. 2002) that the performance of TPC-D query number four is disk bound. Since this study is emphasising systems for large contiguous read operations, we show the results of the different systems for this specific query separately in Figure 4 (“FNBD raw” is discussed in Section 6). The figure indicates that FNBD is able to uphold the performance of OLAP on a single local disk, but PVFS cuts the parallel efficiency in half resulting in a doubled execution time.

It is especially interesting to note that the performance of the TPC-D queries is not higher on FNBD than on a local disk—despite the higher raw read performance of FNBD as measured in the microbenchmarks. The reasons for this paradox are not certain, but we suspect a slightly higher latency for FNBD read operations because of the network and the more involved layers on client and servers. In the next Section we analyse the results in more detail.

## 6 Analysis of Results

To better understand the results from the previous section, we now describe our systems with an analytical model. The model is based on data streams

which flow through the subsystems of a PC node. The model’s focus is solely on large data streams in clusters and the strength of the model is its simplicity. Simpler models result in shorter turn-around times and more guidance during the design of new systems at the potential disadvantage of a slightly reduced accuracy. Despite its simple structure, our model correctly predicts performance trends for different architectures in a data-stream environment, like e.g. large-scale data distribution for partition cloning to install OSs in a cluster. Since large table scan operations also require the transfer of large data streams, we presume the model’s applicability to our workloads. The model is described in more detail in (Rauch, Kurmann & Stricker 2002). We summarise the most important aspects below and discuss its applicability to the workloads described in this paper.

### 6.1 A Data-Stream-Based Analytic Model

The analytic model for all data streams in a cluster considers the streams in the high-speed interconnect of a cluster including the cables, switches (*passive nodes*) and the PC nodes actively involved in handling the data streams (*active nodes*). The solution of the model’s equations evaluates several resource constraints. We assume reliable data streams (e.g. controlled by a higher-level transport protocol) and that there is a fair sharing of resources between the data streams that flow through a system.

The throughput capacity of an active node is modelled by the two network-related limits and four additional resource limits within the active node. The limits of the network are the link capacity (which is taken from the physical specifications of the network technology, e.g. 125 MByte/s for Gigabit Ethernet or 12.5 MByte/s for Fast Ethernet) and the switch capacity of passive nodes (the capacity of the passive nodes is taken from the physical specifications of the network hardware, e.g. the switch(es) of a cluster).

The resource limits of an active node are listed below. The examples of corresponding hardware are based on the CoPs cluster described at the beginning of Section 5.

**Disk system** The maximum throughput of the node’s hard-disk drive (e.g. 33 MByte/s for an IBM DDYS-T09170M 10’000 rpm disk).

**I/O-bus capacity** The sum of data streams traversing the I/O bus must be less than its capacity (e.g. 264 MByte/s on 32-bit 66-MHz PCI-bus-based PC cluster nodes).

**Memory-system capacity** The sum of the data streams to and from the memory system must be less than the memory system’s capacity (e.g. 420 MByte/s on systems with the ServerWorks ServerSet III LE chipset).

**CPU utilisation** The processing power required for the data streams at the different stages. For each operation the fraction coefficient  $1/a_1, 1/a_2, 1/a_3, \dots$  indicates the maximal speed of the stream with exclusive use of 100% CPU. The denominators  $a_1, a_2, a_3, \dots$  correspond to the real throughputs of the respective subsystems as observed in practice. The sum of the fractions of CPU use must be  $< 1$  (= 100%) (Throughputs considered: 160 MByte/s SCSI transfer, 210 MByte/s internal copy memory to memory, 60 MByte/s send or receive over Gigabit Ethernet with the ASANTÉ network interface cards).

Limitations on these four resources result in constraint inequalities for the maximal throughput

achievable through an active node. The minimal throughput resulting from the inequalities is the maximal throughput for the overall data stream.

For example, sending a large amount of data from a typical file server comprises concurrently reading data from the local disk, copying it from the kernel space to a user-space daemon and back, and sending it over the network interface to the client. We assume a one-copy TCP/IP protocol stack as provided by a standard Linux kernel. The corresponding inequalities for the throughput  $b$  of the data stream in this example are then  $b < 33$  MByte/s for the hard disk,  $2b < 264$  MByte/s for the IO bus and  $6b < 420$  MByte/s for the memory. For the CPU, the inequality considers one SCSI transfer, two memory-copy operations and one network transfer:  $(1/160 + 2/210 + 1/100)b < 1$ . The resulting maximal data-stream throughput is 38.9 MByte/s with the CPU as bottleneck.

A related and widely popular performance model for parallel and distributed systems is LogP (Culler, Karp, Patterson, Sahay, Schauer, Santos, Subramanian & von Eicken 1993). As a model for parallel machines and parallel computations it assumes a machine as a collection of essentially complete computers and puts a focus on network congestion or overhead handling messages of NUMA accesses and does not deal properly with accesses to the memory and the I/O bus for auxiliary copies in the system software. The model used in this paper resembles more to the original ideas of the copy-transfer model for communication-system performance (Stricker & Gross 1995).

### 6.2 Modelling OLAP as a Black Box

For the analytic model described in the preceding subsection it is important to know and understand the performance of all involved parts and subsystems, which are partially hardware (CPU, memory, I/O subsystem including NIC and disk), partially OS software (drivers for network interfaces and disks, protocol stacks) and of course the application software (e.g. the ORACLE RDBMS). It will be especially tricky to model the performance of a “black box application”—an application with no source code available and little knowledge about its detailed and complex inner workings.

For ORACLE we use two different cases to model the performance of the application layer: A good case and a bad case, which are deliberately not the best and worst case. The best case in an RDBMS might be a simple index-based lookup of a value and it certainly does not involve a data stream. The worst case on the other hand can be very complex and combine different full table scans, join and sort operations. Thus, as good and bad case that can still be understood, we chose the following two simple queries that are executed on the data set of the TPC-D benchmark: (a) `SELECT MAX(c_acctbal) FROM customer` and (b) `SELECT c_acctbal FROM customer ORDER BY c_acctbal`. Query (a) requires a full table scan of a large fact table to find a maximum value. It therefore has to read a large data stream, but the CPU has otherwise not much to do with the data besides finding the maximum value. Query (b) also has to read a large amount of data, but it additionally has to sort the data, thereby not only reading but also touching and/or copying the data around more often. To find out the throughput of ORACLE alone for these two queries—i.e. without involving many other subsystems—we chose the table `customer` that fits into the buffer cache of the machine running ORACLE. Tests revealed that the system has to read 278.8 MBytes and 280.0 MBytes respectively from

disk<sup>3</sup> during the execution of the above two queries with a cold buffer cache. With a hot buffer cache the execution of the two queries took 2.01 and 36.14 seconds respectively, thus resulting in a throughput of 140 MByte/s and 7.75 MByte/s respectively for ORACLE. We will use these numbers in the following paragraphs to model the overall performance of OLAP on top of the different storage subsystems.

When modelling OLAP—especially with a distributed database in a cluster of PCs—one has to take into account that the RDBMS might filter (or pre-select) the data and pass only the parts to the client application which are relevant to the final result. The size of the relevant data parts depends on the query and can be significantly smaller than the data that is originally read from the disks. This data reduction can be modelled as a data compression in our model. To find out the “compression factor”  $c$  of different TPC-D queries, we analysed some queries as shown in Table 2. The queries are limited by different subsystems like CPU, network, disk or middleware as described by Taufer et al. (Taufer et al. 2002). The results indicate that the RDBMS middleware layer “compresses” data by factors that vary by four orders of magnitude. Even the smallest compression factor encountered is larger than 200, which indicates that we have to handle different types of data streams in our model, depending on their position in the data flow—before or after the filtering by the RDBMS. The amount of data transferred is definitely different before and after the filtering.

The modelling of the rest of the system is possible because the other layers comprise known parts that either have known throughput characteristics for data streams or which can be measured individually (e.g. the SCSI disk, the PCI I/O bus, the memory and the CPU).

### 6.3 Quantitative Performance According to our Model

As a first case we model the performance of an RDBMS executing OLAP queries on a local disk-based file system. The architecture is depicted in Figure 5. The 4GL application `sqlplus` reads queries from the user (or the shell) and then contacts the RDBMS through pipes, which involve at least one copy operation. The RDBMS has to read the data of the database from the file system’s buffer cache, which involves another copy from the kernel space to the user space. The file system—or precisely the underlying disk driver—receives the data by DMA from the SCSI interface of the disk.

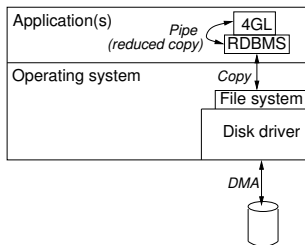


Figure 5: Architecture of OLAP with single local disk.

When inserting the numbers from Subsection 6.1 into the adapted equations of the model, the model predicts a throughput of 33 MByte/s for a trivial large-data read from the local disk. For the slightly more complex cases with the RDBMS we get 33 MByte/s for the good case (trivially limited by the disk) and an overall throughput of 7.1 MByte/s

<sup>3</sup>These values are taken from the `/proc/stat` file.

for the bad case (limited by the CPU). For the disk-bound TPC-D query number four these numbers translate into execution times of 351 and 1631 seconds respectively for the two cases. The rather large difference of a factor of five comes from the very different queries assumed for the good and bad example cases. As you will see in the next paragraphs, the lower and higher numbers still allow to compare the modelled performances of the different architectures.

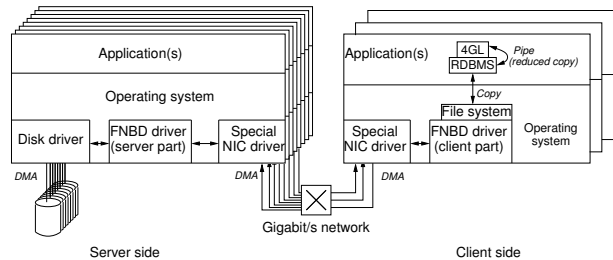


Figure 6: Architectural model of OLAP on the Fast Network Block Device (FNBD).

As a next architecture we model an OLAP system on top of the distributed-devices system FNBD (Figure 6<sup>4</sup>). FNBD is strongly optimised for high-performance data streams and therefore does not involve any unnecessary data copies on the servers—the only data transfers are done by DMA transfers from the SCSI adapter and to the NIC. While the client is also optimised by using special protocols and running completely in kernel space, there is still one last data copy to cross the protection boundary between kernel and user space.

The predicted maximum throughput for a trivial application reading raw sequential data from a FNBD-based file system is predicted at 46.7 MByte/s. With the RDBMS system we get a maximum throughput of 35.0 MByte/s and 6.7 MByte/s for the good and the bad case respectively. The corresponding execution times for the TPC-D query number four are 331 and 1729 seconds (limited by the CPU of the client in both cases).

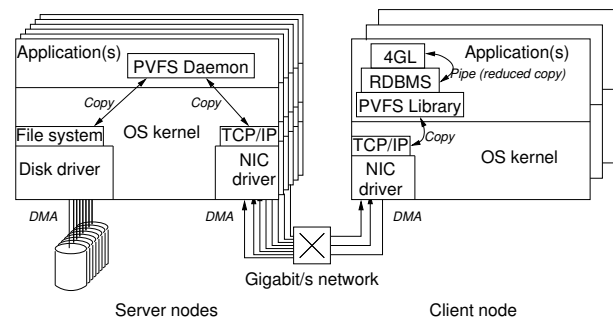


Figure 7: Architectural model of OLAP on the parallel virtual file system (PVFS).

Modelling an OLAP system on top of PVFS is quite similar to FNBD, except that the PVFS servers need additional data copies between their user-space daemon and the OS kernel (see Figure 7). Regarding our model, the client side of PVFS is thus identical to the FNBD-based architecture. The features of the file system might introduce some additional complexity into the system, but this fact is not captured by our stream-oriented model.

The model predicts a maximal throughput of 72.2 MByte/s for a trivial application which reads

<sup>4</sup>In our model (and the previous experiments) we only use a single client. The figures in this section indicate that multiple clients could access the data in our read-only data-mining setup.

TPC-D query	Input read [MByte]			Result output [MByte]	“Compression factor” $c$
	index	data	total		
1	0	8'758	8'758	0.001	7'300'000
2	85	477	562	2.4	234
3	0	10'998	10'998	5.6	1'964
4	1'318	10'265	11'583	<0.001	17'600'000
8	3'412	10'782	14'194	0.017	835'000

Table 2: TPC-D query analysis with ratios of input/output.

contiguous raw data from PVFS (CPU limit of the client). Integrating the RDBMS and 4GL interface into the model, we predict 47.6 MByte/s for the good case and 7.0 MByte/s for the bad case with execution times for the TPC-D query number four of 243 and 1655 seconds respectively (CPU limit of the client in both cases). The predicted numbers are better than in the FNBD case, because we use different Gigabit Ethernet adapters. The hardware of the adapters used for FNBD was limited in its throughput by hardware flaws. As a result of the predicted higher raw-read performance, the model predicts better OLAP execution times for PVFS than for FNBD.

All performance numbers according to our model are summarised in Table 3 (“TP-Lite” will be discussed in the next subsection) and compared with the corresponding measured numbers from our experimental systems. From the table, we note that the difference between the modelled and measured numbers for the raw-read numbers with PVFS is large. We can not explain the reason for the difference, but assume the increased complexity of a full-featured distributed file system. Accordingly, the execution time for TPC-D query four on PVFS is the worst of all examined systems. Further, even though the measured raw-read throughput of FNBD is higher than that of a single local disk, the performance of TPC-D query four is still better on the single local disk. We attribute this anomaly to the higher latency in a distributed system as compared to a local system, even though latency should not influence the throughput of large read operations.

For all the systems examined so far, the limit for non-trivial queries is the CPU of the client nodes. Not only has the CPU to handle the data according to the queries, it must also copy all the data read from the disk from the kernel’s buffers to the user-space applications or libraries. An increase in performance is thus only possible if the load on the clients’ CPU is reduced.

#### 6.4 Improving Database Performance on Clusters of PCs

As a first way of reducing the CPU load on the client, we setup FNBD as a “raw device”. For raw devices, the Linux kernel bypasses the buffer cache and stores data blocks read from the disk directly in the user-space buffers, which eliminates a copy operation. The result shown in Figure 4 (labelled as “FNBD raw”) is worse than for the regular FNBD, because the buffer cache implements read ahead. Without the buffer cache, there is no read ahead and thus a high latency-overhead for every single block read synchronously from FNBD.

A fully transparent way of reducing the CPU load on the client is to implement a true zero-copy technique in the OS. Such a true zero-copy technique is described in detail in (Kurmann et al. 2001). In short, true zero copy defragments incoming data packets from the network directly into page-size buffers, which are then re-mapped from kernel space to user space without much CPU intervention. The client’s

CPU would thus focus its work on the query instead of copying data. This technique is applicable if the application reads and writes data on page boundaries.

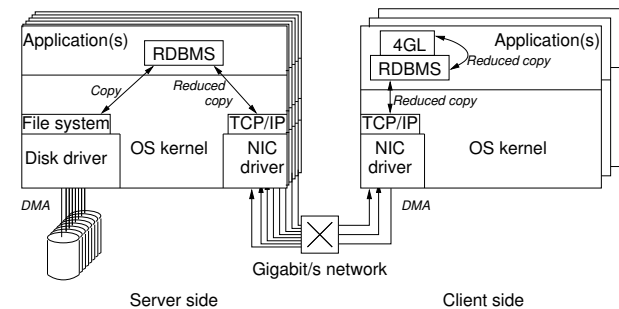


Figure 8: Architectural model of OLAP with TP-Lite, a middleware to distribute queries to multiple remote RDBMSs.

Another technique is to distribute the pre-filtering of the data read from the cluster nodes’ hard-disk drives onto multiple nodes in the cluster. This is done by running the database system itself on multiple machines, distributing the queries to multiple servers and to collect only the required results on the client machine, thereby reducing the amount of data transmitted to and copied by the client node. Two examples for such a system are TP-Lite (Böhm, Grabs, Röhm & Schek 2000) and C-JDBC (Cecchet, Marguerite & Zwaenepoel 2003). TP-Lite supports the database when coordinating and distributing partial queries to other instances of relational database management systems (RDBMSs) running on the server nodes. C-JDBC is a middleware layer that presents a single virtual database to applications through the JDBC interface and is thus limited to applications that support the JDBC interface.

To compare the middleware approach with our distributed storage systems, we examine the TP-Lite middleware. We chose TP-Lite as a reference case in our evaluation, because it is the “poor man’s implementation of a parallel database” and thus suits the commodity philosophy of a multi-use cluster best in terms of off-the-shelf hardware, open-source operating system and standard databases. However, changing the middleware or the application to do the remote disk access is not always a realistic design option, as it is neither trivial nor is it possible for commercial applications in many situations. As depicted in the architectural model in Figure 8, there are only reduced data-copy operations on the clients, thereby reducing the CPU load considerably. Accordingly and as shown in Table 3, the execution times according to our model are much lower for TP-Lite than for all other systems. Also when looking at the performance of different TPC-D queries in Figure 3, TP-Lite achieves best performance for most queries. However, using TP-Lite requires changes to the database system and is therefore not possible in all situations.

In contrast, our architecture with distributed storage is fully transparent and can be used without any

System name	Underlying sub-system raw read		OLAP throughput modelled		TPC-D query 4 modelled			<i>meas. real</i> [s]
	mod. [MByte/s]	<i>meas.</i> [MByte/s]	good [MByte/s]	bad [MByte/s]	good [s]	bad [s]		
Local disk	33	32.7	33	7.1	351	1631	446	
FNBD	47	35.7	35	6.6	331	1729	469	
PVFS	72	9.8	48	7.0	243	1655	938	
TP-Lite	119	109	99	21	117	544	162	

Table 3: Quantitative performance on different architectures according to our analytic model and measurements on our cluster of commodity PCs. The table includes assumed good as well as bad example cases and measurements for throughput as well as execution times of OLAP with ORACLE RDBMS.

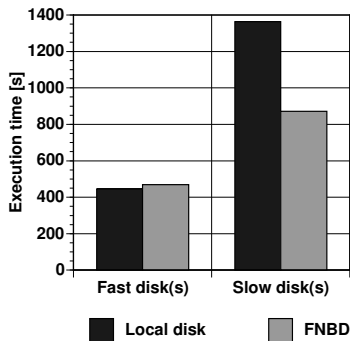


Figure 9: Simplistic simulation of future systems: Execution times for TPC-D query number four with fast (regular SCSI) and slow disks with the SCSI bus set to 10 MByte/s.

changes to the database system. It is important to remark that distributed storage will become increasingly important in the future because of two technological trends: First (as shown in Figure 1) the size of hard-disk drives increases much faster than the size of an OS installation, leaving much excess storage capacities on the disks of PC nodes, and secondly, the gap between the throughputs of commodity networks and hard-disk drives is widening drastically (Rauch 2003). To increase throughputs for storage subsystems, the throughputs of an increasing number of disks on the network must be combined to keep up with the network’s throughput. In such a future system, a database system on top of a distributed-storage architecture will become an increasingly interesting option, as illustrated in Figure 9. In the figure we compare FNBD as discussed throughout the paper (labelled “Fast disk(s)”) with the same system but artificially slowed disks (labelled “Slow disk(s)”). We slowed the disks by changing the SCSI bus’ maximum transfer rate to 10 MByte/s in the BIOS. The figure suggests that a future FNBD will offer a much higher throughput than a system based on a local disk.

## 7 Conclusions

In our paper we studied different high-performance storage architectures for clusters of commodity PCs as a technique to parallelise a database for OLAP applications on clusters. Parallelising a system on the storage layer enables us to use the excess storage capacities on the nodes’ hard-disk drives in a fully transparent way at no additional cost—thereby cost-effectively increasing the storage available to data-mining applications. The OLAP applications access the distributed storage systems we examined in exactly the same way as a local disk, thereby eliminating the need to make *any* changes to the database software. It would be a considerable effort to re-write

all database operations in a genuine parallel way and even then, most applications still contain some sequential work which limits their scalability. It is a real plus to accelerate these remaining sequential sections considerably.

Our detailed performance measurements with two different storage systems revealed that the fully-featured distributed file system PVFS introduces too much additional complexity into the system and thus offers lower performance than a single local disk. The less complex distributed-devices system Fast Network Block Device (FNBD), which simply maps remote hard-disk partitions to the database client and combines them in a RAID 0, offers a higher performance. FNBD’s performance is about the same as a single local disk, but offers a large amount of additional storage at no extra cost. This lower than expected result is partially due to some flawed high-speed network interface cards. In an attempt to increase our understanding of the performance issues, we analytically modelled the systems under study with a data-stream-based model. The analysis revealed that the main scalability-limiting bottleneck are the data copies in the client’s memory.

As a result, we examined the query-distributing middleware TP-Lite in our setup. In TP-Lite, the client distributes partial queries to RDBMS servers running on multiple nodes in the cluster. The client then collects the pre-selected data returned from the servers, which results in much reduced data copies and thus increased performance. However, this performance gain comes at the cost of having to change the database software, which is not possible in all situations. A fully transparent technique to eliminate the costly data copies on the client node is to implement a true zero-copy networking stack. True zero copy would eliminate all unnecessary data copies on the client. The integration of such an optimisation into the system is left as further work and has the potential to increase the performance by a factor of 3–5 for medium-sized clusters (as indicated by the results of the experiments with TP-Lite).

Our contribution to the Australasian Database Conference takes a novel approach to the parallelisation of databases by placing the parallelisation transparently in the storage layer and accessing cheap additional storage on the cluster nodes’ hard-disk drives over the high-speed network. Our paper heightens the understanding of performance issues of such database systems for clusters of PCs.

## References

- Adya, A., Bolosky, W. J., Castro, M., Cermak, G., Chaiken, R., Douceur, J. R., Howell, J., Lorch, J. R., Theimer, M. & Wattenhofer, R. P. (2002), FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment, *in* ‘Proceedings of 5th Symposium on

- Operating Systems Design and Implementation (OSDI 2002)'.
- Becker, D. J., Sterling, T., Savarese, D., Dorband, J. E., Ranawake, U. A. & Packer, C. V. (1995), Beowulf: A Parallel Workstation for Scientific Computation, *in* 'Proceedings of International Conference on Parallel Processing', CRC Press, Boca Raton, FL, USA.
- Böhm, K., Grabs, T., Röhm, U. & Schek, H.-J. (2000), Evaluating the Coordination Overhead of Synchronous Replica Maintenance in a Cluster of Databases, *in* A. Bode, T. Ludwig, W. Karl & R. Wismüller, eds, 'Lecture Notes in Computer Science 1900, Euro-Par 2000 Parallel Processing, 6th International Euro-Par Conference Munich', Springer, Munich, Germany, pp. 435–444.
- Cecchet, E., Marguerite, J. & Zwaenepoel, W. (2003), RAIDb: Redundant Array of Inexpensive Databases, Technical Report 4921, Institut national de recherche en informatique et en automatique (INRIA).
- Chen, P. M. & Patterson, D. A. (1993), A New Approach to I/O Performance Evaluation: Self Scaling I/O Benchmarks, Predicted I/O Performance, *in* 'Proceedings of the 1993 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems'.
- Culler, D., Karp, R., Patterson, D., Sahay, A., Schauser, K. E., Santos, E., Subramonian, R. & von Eicken, T. (1993), 'LogP: Towards a Realistic Model of Parallel Computation', *SIGPLAN Notices* **28**(7), 1–12. Forth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 19–22 May 1993, San Diego, CA, USA.
- Gibson, G. A., Nagle, D. F., Amiri, K., Chang, F. W., Feinberg, E., Gobioff, H., Lee, C., Ozceri, B., Riedel, E. & Rochberg, D. (1996), A Case for Network-Attached Secure Disks, Technical Report CMU-CS-96-142, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.
- Howard, J. H., Kazar, M. L., Menees, S. G., Nichols, D. A., Satyanarayanan, M., Sidebotham, R. N. & West, M. J. (1988), 'Scale and Performance in a Distributed File System', *ACM Transactions on Computer Systems* **6**(1), 51–81.
- Katcher, J. (1997), PostMark: A New File System Benchmark, Technical Report 3022, Network Appliance.
- Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C. & Zhao, B. (2000), OceanStore: An Architecture for Global-Scale Persistent Storage, *in* 'Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)'.
- Kurmman, C., Rauch, F. & Stricker, T. M. (2001), 'Speculative Defragmentation—Leading Gigabit Ethernet to True Zero-Copy Communication', *Cluster Computing: The Journal of Networks, Software Tools and Applications* **4**(1), 7–18.
- Lee, E. K. & Thekkath, C. A. (1996), 'Petal: Distributed Virtual Disks', *ACM SIGPLAN Notices* **31**(9), 84–92.
- Ligon, W. B. & Ross, R. B. (1999), 'An Overview of the Parallel Virtual File System', USENIX Annual Technical Conference, Extreme Linux Workshop. <http://www.usenix.org/events/usenix99/>.
- May, J. M. (2001), *Parallel I/O for High Performance Computing*, Morgan Kaufmann Publishers.
- McVoy, L. & Staelin, C. (1996), Imbench: Portable Tools for Performance Analysis, *in* 'Proceedings of the USENIX 1996 Annual Technical Conference', San Diego, California.
- Norcott, W. D. & Capps, D. (2003), *Iozone Filesystem Benchmark*, <http://www.iozone.org/>.
- Patterson, D. A., Chen, P., Gibson, G. & Katz, R. H. (1989), Introduction to Redundant Arrays of Inexpensive Disks (RAID), *in* 'Digest of Papers. COMPCON Spring '89', IEEE, pp. 112–117.
- Patterson, D. A., Gibson, G. & Katz, R. H. (1988), 'A Case for Redundant Arrays of Inexpensive Disks (RAID)', *ACM SIGMOD Record* **17**(3), 109–116.
- Poess, M. & Floyd, C. (2000), 'New TPC Benchmarks for Decision Support and Web Commerce', *SIGMOD-Record* **29**(4), 64–71.
- Preston, W. C. (2002), *Using SANs and NAS*, O'Reilly & Associates, Inc.
- Rauch, F. (2003), Distribution and Storage of Data on Local and Remote Disks in Multi-Use Clusters of PCs, PhD thesis, Dept. of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Zurich, Switzerland.
- Rauch, F., Kurmann, C. & Stricker, T. M. (2002), 'Optimizing the Distribution of Large Data Sets in Theory and Practice', *Concurrency and Computation: Practice and Experience* **14**(3), 165–181.
- Rhea, S., Eaton, P., Geels, D., Weatherspoon, H., Zhao, B. & Kubiatowicz, J. (2003), Pond: The OceanStore Prototype, *in* 'Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)'.
- SPEC (2001), *SFS 3.0 Documentation Version 1.0*, Standard Performance Evaluation Corporation, 6585 Merchant Place, Suite 100, Warrenton, VA 20187, USA. <http://www.spec.org/>.
- Stricker, T. & Gross, T. (1995), Optimizing Memory System Performance for Communication in Parallel Computers, *in* 'Proc. 22nd Intl. Symposium on Computer Architecture', ACM, Santa Margherita di Ligure, pp. 308–319.
- Taufer, M., Stricker, T. & Weber, R. (2002), Scalability and Resource Usage of an OLAP Benchmark on Clusters of PCs, *in* 'Proceedings of the fourteenth annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'02)', ACM Press, New York, NY, USA, pp. 83–94.
- TPC (2003), 'Transaction Processing Performance Council, TPC', World Wide Web, <http://www.tpc.org/>.
- Widmer, C. (2001), Transparente Zugriffsmethode auf verteilte Festplatten, Diplomarbeit (masters thesis), Swiss federal institute of technology Zürich, laboratory for computer systems. English title: A transparent access method for distributed disks.