Performance and Bus Transfer Influences

S. Schönberg F. Mehnert Cl.-J. Hamann L. Reuther H. Härtig

Dresden University of Technology
Faculty of Computer Science
D-01062 Dresden, Germany
email: schoenbg@os.inf.tu-dresden.de

Introduction

Real-time systems which offer guaranteed response times need to schedule all resources, including CPU cycles, caches and bandwidths of buses. In this review we present a methodology how to schedule real-time requests for SCSI bus based IO and show measurements about the influences between SCSI, ATM and CPU traffic on the PCI bus and the local memory bus. We propose a slowdown factor $\mathcal{F}_{sd}$ to characterise applications for their sensitivity to bus interference which is used in CPU scheduling analysis. This work is embedded in the DROPS project [1] that uses simple and powerful standard hardware as platform; it tempts to organise the coexistence of real-time and time-sharing applications by providing managers for all resources. These managers enforce reservations for real-time applications and leave the rest to the time-sharing applications.

We mainly focus on bus bandwidth as resource, whereas the conflicting points in a standard PCI-based system are (I) the SCSI bus, (II) the PCI bus, and (III) the memory bus.

Experimental work showed that by use of our SCSI access method we can achieve guaranteed SCSI bandwidth of 86% of the maximum bandwidth. The slowdown factor for the application with worst-case (i.e., highest) memory bandwidth is around 2.5, for applications such as Quicksort it is 1.22, and 1.08 for DES.

For measurement, we generate maximum SCSI load for a predictable access method for SCSI bus systems. Next, we induce PCI load by a master capable PCI card that actively transfers data into the host memory, and measure maximum memory bus read and write bandwidth. Therefore, we use an FORE PCA200E card, since the on-board i960-CPU is freely programmable. Hereafter, we combine PCI load with SCSI bus to see impacts on the PCI bus, and last but not least, PCI load and host memory transfer.

Finally, we determine the worst-case slowdown factor for processes running on the processor. For that purpose, we measure a worst-case (i.e., highest memory bandwidth) process and its slowdown.

Time-Driven SCSI Bus Operation

As mentioned, the major objective of DROPS is to combine real-time and time-sharing applications and their data on a single machine. Hence, we cannot rely on the exclusive access on pre-allocated data streams.

We shortly describe a model to provide guaranteed bandwidth even under these constraints using a time-driven access method.

One read/write operation consists of the following steps: (1) bus connect (arbitration), (2) command request, (3) disconnect, (4) drive seek, (5) bus connect, (6) data transmission, and, finally, (7) bus disconnect. If the drive cannot send all data in one transfer, it repeats disconnect and reconnect cycles to the bus until all data has been transmitted. The net data transfer time can be calculated from the bus bandwidth, the data size and the command time. The worst-case disconnect time is defined in the SCSI standard. Time for step 4 depends on the drive's characteristics and the current head position. The time when the drive starts sending its data over the bus is not predictable. The drive begins its data transfer, when enough data is read into the internal cache. Bus arbitration of the SCSI bus is done by a prioritisation scheme; bus conflicts occur when several drives independently reconnect to the bus at the same time. Hence, transfers may be delayed by other higher priorized transfers.

The general approach taken is to use the SCSI bus in a time-driven manner. Commands are issued periodically at precise points in time. The exact time-driven command execution allows to maximise the throughput even under real-time constraints and worst-case access patterns.

Comparison & Results

Our model reveals that for currently available drive technology and a bandwidth of 40 $^{MB}/_{s}$ for 16-bit Ultra-Wide SCSI, 21 drives need to be connected; this is beyond our current technical capabilities. But we believe in the near future with faster drives our model describes a realistic scenario.

To verify our assumptions, we measure and simulate the critical values with restricted bus throughput of 10 $^{MB}/_{s}$ and 128 KB and 256 KB clusters. This allows us to compare our technique with an access pattern that does not give guarantees on the same disk system.

While for the 10 $^{MB}/_{s}$ restricted bus the fastest possible access without slotting resulted in timeouts (which means some blocks could not be transferred to the host adapter); our technique provided 8.6 $^{MB}/_{s}$ of guaranteed transfer bandwidth.

We measured the maximum possible SCSI bandwidth of our system with 35.5 $^{MB}/_{s}$ when the block is read only from the drive's cache and 10.2 $^{MB}/_{s}$ bandwidth for a random read.

PCI Influences

The memory bus benchmark tasks operate on two arrays, one for source data and one for destination data, each 8 MB of size. The ATM card uses a dedicated 8 MB space. This avoids cache snoop hits which imply L1 and L2 cache write-backs on dirty cache-lines. This scenario can be regarded as realistic, since device drivers usually operate exclusively on a given buffer space and no application works *simultaneously* on the same buffer.

We expect to get the worst-case factor for a test that only focuses on memory operations. For measurement we made three different runs with two 8 MB buffers; read, write, and copy. A read consisting of four `mov` instructions reveals the highest memory performance (155.26 $^{MB}/_{s}$). Write (69.88 $^{MB}/_{s}$) and copy (96.16 $^{MB}/_{s}$) performed best using the CPU instructions `rep stosd`, and `rep movsd`.

Note, that the bandwidth for copy is higher than that for write, since always 8 MB are transferred, which means 4 MB are read *and* written. For comparison, we made an additional test that generates accesses to cache-line aligned memory in random order to avoid that CPU combines contiguous read cycles to memory bus bursts, resulting in 143.78 $^{MB}/_{s}$. Figure 1 confirms the statement that the highest memory bus load is generated when accessing the memory in ascending or descending order.

We use the i960 processor on the ATM adaptor card to generate bursts between 1.5 $^{MB}/_{s}$ and 36.0 $^{MB}/_{s}$. Figure 1 depicts the $\mathcal{F}_{sd}$ for the load/store/copy memory benchmarks.

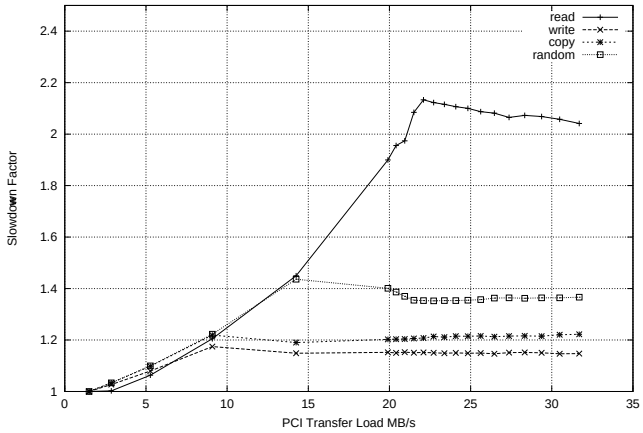

Figure 1: Memory Benchmarks (Slowdown)

Application Benchmarks

Following, we now try to obtain $\mathcal{F}_{sd}$ for three different types of applications. (1) DES - a very CPU-intensive task with high cache hit rate, (2) Quicksort - a more memory-intensive task, and (3) MPEG I decoding, a task with both high CPU and memory utilisation. Figure 2 illustrates these slowdown factors.

(1) Measurements of the influence of high PCI load in parallel to DES showed that a CPU intensive benchmark is slowed down for about 9% or $\mathcal{F}_{sd}$ = 1.09.

(2) [4] describes a non-recursive version of a fast sorting algorithm. For our measurements we filled the 8 MB buffer with numbers in descending order. This leads to the algorithmic worst case and subsequently to the highest possible memory load. The slowdown factor here is $\mathcal{F}_{sd}$ = 1.22.

(3) MPEG [2] is used for video and audio compression in our system. The MPEG decoding measurement was done using a PIIPII-coded video, 384x288 pixel with 24 bit color and a compression factor of 60 for 33 I-frames and 169 for 66 P-frames. The slowdown factor here is $\mathcal{F}_{sd}$ = 1.36. Variation in the graph in Figure 2 can be explained by the different time the CPU takes to decode each picture group.

Conclusion / Future Work

The technique for time-driven access method for the SCSI bus allows to guarantee predictable access times and bandwidths even under a worst-case scenario. Bus utilisation up to 91.4% can be achieved.

Interference on the PCI bus under loads up to 60 $^{MB}/_{s}$ are at about 8%. Since time-sharing loads exist besides real-time loads, planning and scheduling of the PCI bus seems not feasible.

The factor by which an application performing memory operations is slowed down ($\mathcal{F}_{sd}$) allows to describe the influence of PCI bus loads on this application. In the worst-case, we measured a value of $\mathcal{F}_{sd}$ = 2.2. $\mathcal{F}_{sd}$ allows us to scale the part of the CPU an application needs under idle load to that under worst-case conditions.

Future work need to be done to develop a tool that automatically obtains all worst-case disk parameters. Tests shall be repeated with newer drive and bus technologies to verify our simulations.

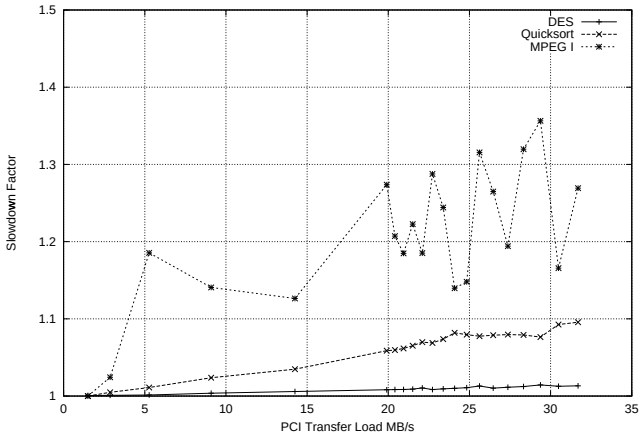

Figure 2: Application Benchmarks (Slowdown)

Acknowledgements

Jean Wolter, Michael Hohmuth, Jochen Liedtke, Volkmar Uhlig, and Robert Baumgartl provided helpful feedback and commentary on earlier versions of this paper. Special thanks to Uwe Dannowski for his support on the ATM driver and i960 firmware.

References

[1] R. Baumgartl, M. Borriss, H. Härtig, Cl.-J. Hamann, M. Hohmuth, L. Reuther, S. Schönberg, and J. Wolter. DROPS - Dresden Realtime Operating System, 1998.

[2] D. Le Gall. MPEG: a video compression standard for multimedia applications. *Communications of the ACM*, 34, 4:46–58, 1991.

[3] Robert Sedgewick. Implementing Quicksort programs. *Communications of the ACM*, 21(10):847–857, October 1978. See corrigendum [4].

[4] Robert Sedgewick. Corrigendum: "Implementing Quicksort Programs". *Communications of the ACM*, 22(6):368–368, June 1979. See [3].