Low-Cost Image Generation for Immersive Multi-Screen Environments

Felix Rabe[1] Christian Fröhlich[1] Marc Erich Latoschik[1]

[1]AI & VR Lab, Bielefeld University

Abstract: This paper describes the configuration of a cost-efficient monolithic render server aimed at multi-screen Virtual Reality display devices. The system uses common Of-The-Shelf (OTS) PC components and feeds up to 6 independent screens via 3 graphics pipes with the potential to feed up to 12 screens. The internal graphics accelerators each use at least 8 PCIe lanes which results in sufficient bandwidth. Performance measurements are provided for several benchmarks which compare the system's performance to well established network based render clusters.

Keywords: Low-cost image generation, render hardware, multi-screen rendering

1 Introduction

Since the middle of the 90th, professional graphics capabilities are entering the low-cost PC market. Mainly driven by entertainment and gaming, the 3D graphics industry has primarily focussed on dedicated PC graphics accelerator cards which support single or dual displays as commonly expected to be found in desktop based systems. To utilize these graphics engines for Immersive Projection Technology (IPT) systems is the main goal of graphics cluster technology. The idea is to substitute the dedicated and proprietary intra-node CPU-to-graphics high-speed interconnects of professional multi-pipe graphics hardware by OTS (Of the Shelf) network interconnects to cluster multiple common PC based render systems with 3D graphics accelerator cards for multi-screen environments. The technical challenge here is how to distribute the render tasks between the interconnected nodes to minimize bandwidth usage and latency and hence to conserve the limited network resources.

In this paper, we introduce the hardware and software setup of a PC based image generator (shown in figure 1). Its main differences from past approaches is its utilization of the now available PC based serial intra-node interconnects instead of a network based interconnect. Cluster-based graphics systems are usually considered low priced in comparison to dedicated hardware. For example, a simple system driving 6 screens may be configured for less then 15.000,-EUR including the network infrastructure. Since our system a) does not require every node to be a fully equipped PC and b) does not need any network infrastructure it can be assembled for less then 3.000,-EUR. Still, the current system is configured to feed 6 independent displays using 3 independent hardware pipes with the potential to drive up to 12 independent displays.

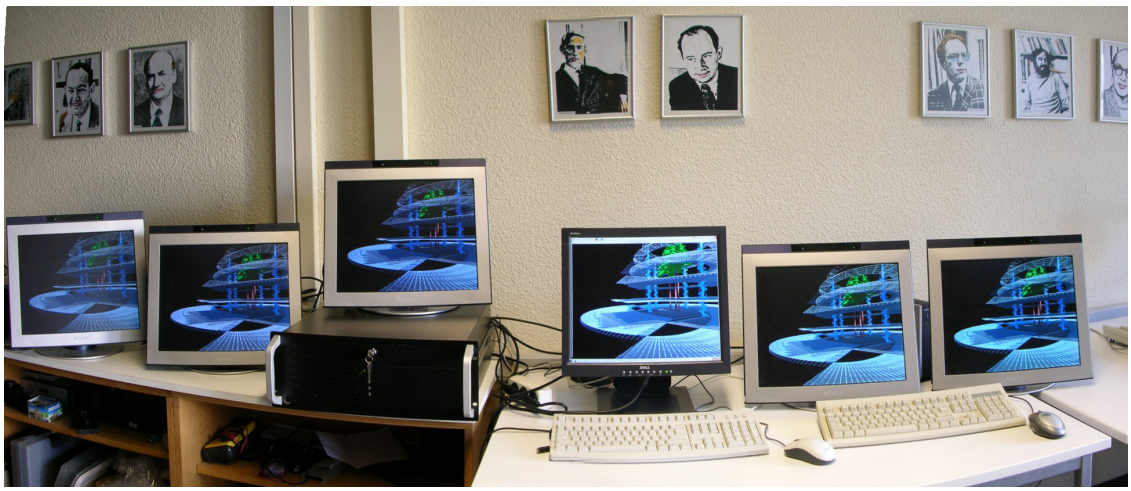

Figure 1: The new system with 6 screens attached. It runs SPECviewperf® and distributes the graphics output to each screen using Chromium.

2 Related Work

In general, generating images for multiple screens can be achieved using several ways. Figure 2 illustrates the basic interconnection scheme of a standard PC supporting one graphics card. One limiting factor in the upcoming discussions is of course the pure graphics performance of the graphics accelerator. As we will later see, we neglect this bottleneck since—compared to the multi-pipe interconnection schemas discussed—it is of minor importance. The limiting factors in the figure are the memory access and the peripheral access which may both be determined by the underlying transport mechanism of the utilized chipset and/or the CPU in case the CPU provides built-in communication facilities.

At the current time, a major technology shift in PC based computer architectures can be observed. First, high-speed network interconnects like 10G Ethernet or Infiniband become available to the mass-market. Second, CPU core clock rates have reached a limit which complicates brute force clock rate acceleration. This is one of the main factors which motivates current multi-core CPU architectures for future performance increases. Third, intra-node connects like PCIe (PCI Express) reach a performance high enough to allow for multi-pipe

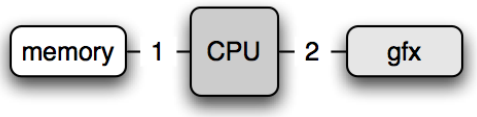

Figure 2: Simple example of the interconnection scheme of a typical PC. the CPU is connected to the memory utilizing a dedicated communication channel (1). The graphics card is connected by the supported peripheral interconnect (2). Both connections may be limited by the given chipset's and/or CPU's communication system.

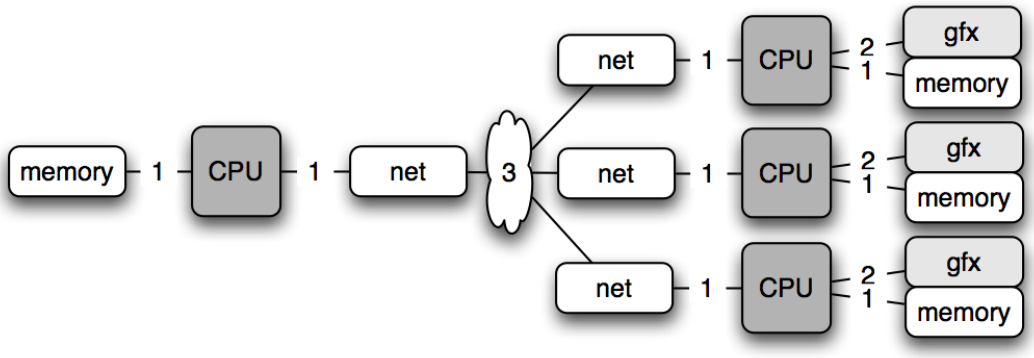

Figure 3: Network based graphics cluster. Multiple nodes with local memory and graphics cards are interconnected by a network infrastructure (3).

systems being assembled by commodity PC hardware. Key-numbers of different interconnect types are given in table 1.

Figure 3 illustrates a typical setup of nowadays network based graphics clusters. Several nodes are interconnected using a network layer. Each node has its own memory and possibly its own graphics card. Their local connection scheme is defined by the same boundary conditions as in figure 2. The final image generation has to be synchronized between the nodes using the available inter-node connection. For distributing graphics to generate multiple images there are several principle ways:

- Distributed application: Each node runs the same application in parallel and renders the scene from a specific view. Only buffer swap synchronization data is send over the network.

- Distributed scene graph: Shared graph-based graphics data structure. Only attribute changes are synchronized over the network. See, e.g., AVANGO [Tra99] or OpenSG [RVB02][OSG].

- Distributed display-traversal: A multi-pipe aware application directs drawing commands to the appropriate pipes during the draw traversal. See, e.g., SGI®'s OpenGL Performer™framework, SGI®'s OpenGL Multipipe™, or OpenSceneGraph [Com07].

- Distributed graphics command: One (or sometimes multiple) node(s) produce(s) rendering commands which are packed and sent over the network to be executed by the graphics cards of the receiving nodes. See, e.g., Chromium [HHN+02].

- Distributed images: One (or sometimes multiple) node(s) produce(s) final images which are sent over the network and are displayed by the receiving nodes. See, e.g., SGI®'s OpenGL Vizserver™

These graphics distribution schemes imply several possible bottlenecks which should be reflected on the utilized hardware platform. The given distribution schemes greatly vary in

Type	Theoretical bandwidth	bidirectional / full-duplex / dual channel
SGI® NUMAlink™4	3,2GB/s	6,4GB/s
HyperTransport™2.0	11,2GB/s	20,4GB/s
HyperTransport™3.0	20,8GB/s	41,6GB/s
Gigabit Ethernet	125MB/s	250MB/s
Myrinet 10G	1,2GB/s	2,3GB/s
10G Ethernet	1,2GB/s	2,4GB/s
Infiniband 12x	6GB/s	12GB/s
SGI® XIO	1,6GB/s	3,2GB/s
AGP 8x	2,1GB/s	
PCIe 1 lane	250MB/s	500MB/s
PCIe 4 lanes	1GB/s	2GB/s
PCIe 8 lanes	2GB/s	4GB/s
PCIe 16 lanes	4GB/s	8GB/s
DDR2-667 Memory	5,3GB/s	10,6GB/s
DDR2-800 Memory	6,4GB/s	12,8GB/s

Table 1: Comparison chart between different inter-node and intra-node interconnects. The upper top describe backplane interconnects. The upper middle entries describe inter-node interconnects. The lower middle entries describe peripheral interconnects. At the bottom DDR memory bandwidth is shown. The numbers represent theoretical net rates. Effective throughput may vary greatly due to additional protocol overhead.

bandwidth requirements. The given list is sorted in increasing demand, where distributed application has almost no bandwidth requirement and image distribution has the highest. On the other hand, the schemas imply an inverse application complexity.

The serial intra-node connect PCIe has far more potential. NVIDIA®'s SLI™or AMD™'s (former ATI™'s) CrossFire™technology already demonstrates the usage of two graphics accelerators in one PC. To use just 2 cards is more a matter of the market demand for desktop based 3D acceleration then a technological limit, as demonstrated by one of the same company's latest products, the Quadro® Plex.

3 System Layout

In our prototype render system, we have assembled three high-end consumer graphics cards into a system which supports 3 PCIe 16 lanes slots. The general idea behind this architecture is illustrated in figure 4. The network based inter-node connects of today's cluster architectures like, e.g., in [AMBR05][DRE06], are substituted by intra-node connects. Intra-node

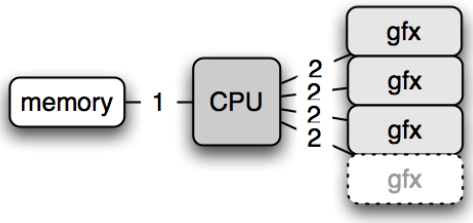

Figure 4: Intra-node multi-pipe system.

connects provide higher bandwidth and at the same time a lower latency due to unnecessary network protocol overhead.

It took considerable effort to assemble a compatible system following that scheme due to the up-to-dateness of the required components—most just arrived on the the market. There were several obstacles, the first one was to find a motherboard that not only provided the three PCIe 16 lanes slots, but which physically separated all three slots far enough to let 3 double-width graphics cards to fit in. The second one was about power supply and cooling. The system produces a lot of heat due to the closely packed graphics cards where each card consumes a remarkable amount of energy (more then 140 watts). The third obstacle was the installation of the linux operating system and the X-server configuration, since the board required some special tweaks to enable its on-board peripherals like the network adapter. The latter is a requirement since installing the 3 graphics cards does not enable any other expansion card to be inserted into the system (see figure 5).

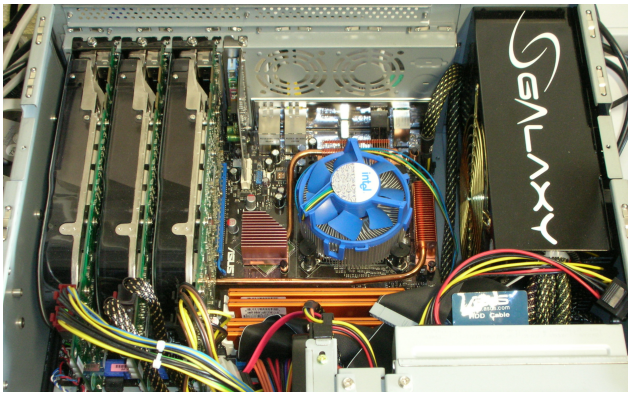

Figure 5: Top view of the assembled system. The expansion slot area (top left) is fully equipped by the three double width consumer graphics cards (NVIDIA 8800GTX). The 1kW power supply directly supports up to 2 cards, the additional power lines are taken from normal periphery connects.

The list of the main components assembled is as follows: 1 x Motherboard ASUS P5N32E-SLI, 1 x CPU Intel Core™2 Quad Q6700 (2.66 GHz), 4 x RAM GEIL 1GB DDR2-667, 3 x Graphics cards ASUS 8800GTX, 1 x Power supply Galaxy 1kW, 1 x Server case with 120mm inblowing front fan.

The system runs with Ubuntu 6.10 Edgy Eft with an self-optimized Vanilla Kernel 2.6.20 rc2, and a X.Org 7.1.1 X-Server with NVIDIA-Linux-x86-1.0-9755 binary drivers. The X-server is configured with six displays from :0.0 to :0.5. Furthermore Chromium 1.8 with some own improvements for our CAVE environment is used on the new system to get comparable results to the comparison system.

3.1 Principles

As illustrated, the assembled system follows an interconnection scheme which was historically supported by dedicated image generators like SGI® (Onyx™Infinite Reality series). This suggest to utilize an appropriate software graphics distribution scheme as formerly developed for these types of hardware architectures, that is to use a distributed display-traversal. For this initial setup this choice was currently deferred in favor for a distributed graphics command approach. This approach has the advantage to let enable the utilization of unmodified applications. Additionally, since it produces a remarkable communication overhead, it is a good benchmark for the system's throughput. If the machine sufficiently handles this scheme, it will perform even better in the distributed scene graph or distributed display-traversal.

3.2 Comparison System

Our comparison system is based on a client-server architecture. It consists of one application server and six renderclients, one for every image generated for our three-sided CAVE environment. The server's configuration is as follows: 2 x AMD Opteron™ Processors 248 2.2 GHz, 4 x 1GB RAM, 1 x Graphics card GeForce 6600GT AGP, 1 x Server case with 120mm inblowing front fan. The six renderclients are configured identically with the following components: 1 x CPU AMD Athlon™ 64 Processor 3000+, 1 x GB RAM, 1 x Gainward GeForce 6800GT PCIe 256MB RAM, 1 x Mainboard ASUS A8N-SLI, 1 x Server case with 120mm inblowing front fan.

The computers are interconnected via a Gigabit Ethernet LAN. We are currently in the process of upgrading our network components to an Infiniband Architecture, but have not done any testing up until now. As mentioned above, we are running a three-sided CAVE with this architecure with two possible distribution paradigms. On the one hand we are using distributed OpenGL with Chromium and on the the other hand a distributed scene-graph using OpenSG. In the following sections we will show some Benchmarks comparing our networked cluster with the low cost PC.

4 Benchmark I

The first benchmark approach was designed to evaluate the rendering performance of the system—locally as well as distributed—and to compare it to our current rendering-cluster setup. To achieve this, the SPECviewperf® 9 benchmark set [Sta07] and ID Software's

Quake III Arena's timedemo benchmark were run. Both benchmarks were run locally on one screen and afterwards distributed with Chromium from one to six screens. The distribution paradigm of Chromium is based on a client server architecture. At the application node a stream processing unit (SPU)—in our network setup a tilesort SPU—sends the graphics commands to server nodes with render SPUs. This setup was adapted on the new server so that it runs the application and distributes the graphics commands to local servers since one server is needed for each screen to render.

The SPECviewperf® 9 benchmark is an OpenGL benchmark program. It is distributed with different kinds of viewsets reassembling real-world applications. The main goal of using SPECviewperf® was to have reliable and comparable benchmarking results for our system.

Quake III Arena's timedemo benchmark was chosen, because the SPECviewperf® benchmark does not apply for the applications that are currently run in our VR-Lab, e.g. there are no highly complex models. Also the new system has consumer graphic cards which are not intended to run CAD applications. Furthermore the Quake III Arena engine is distributable with Chromium since it just requires OpenGL 1.5, in contrary to newer OpenGL 2.0 based games like Doom 3.

The SPECviewperf®, 9.0.3 for Linux/Unix, did not need any configuration, it runs in 1280x1024 resolution, which is the default resolution for the CAVE setup. It was run locally on our new server, with Chromium distributing to a single screen, and with Chromium distributing to six screens as shown in figure 1. To compare the results we ran it on one screen and on all six screens in our current CAVE setup.

Quake III Arena, point-release 1.32, was also configured to render at 1280x1024. Furthermore the configuration included high geometric- and full texture detail, 32 bit color depth and texture quality, as well as trilinear texture filtering. GL-extensions had to be turned off since Chromium is not capable of handling them correctly, which resulted in major image-errors. The timedemo was run with the standard four.dm_68 demo included with the game. First it was run locally, then distributed with Chromium to one to six screens. Again, to compare the results to the current CAVE setup it was run locally on our application-server and then distributed with Chromium to one to six render-servers.

4.1 Results

The SPECviewperf® results (figure 6) and the Quake III Arena's timedemo results (figure 7) show both that the new system outperforms our comparison system. The results lead also to the conclusion, that the distribution with Chromium is a big loss in performance, like the other benchmarks in the following sections show, too.

5 Benchmark II

After knowing the basic system performance the second benchmark approach is designed to evaluate how the new system performs in a setup resembling our CAVE environment. This

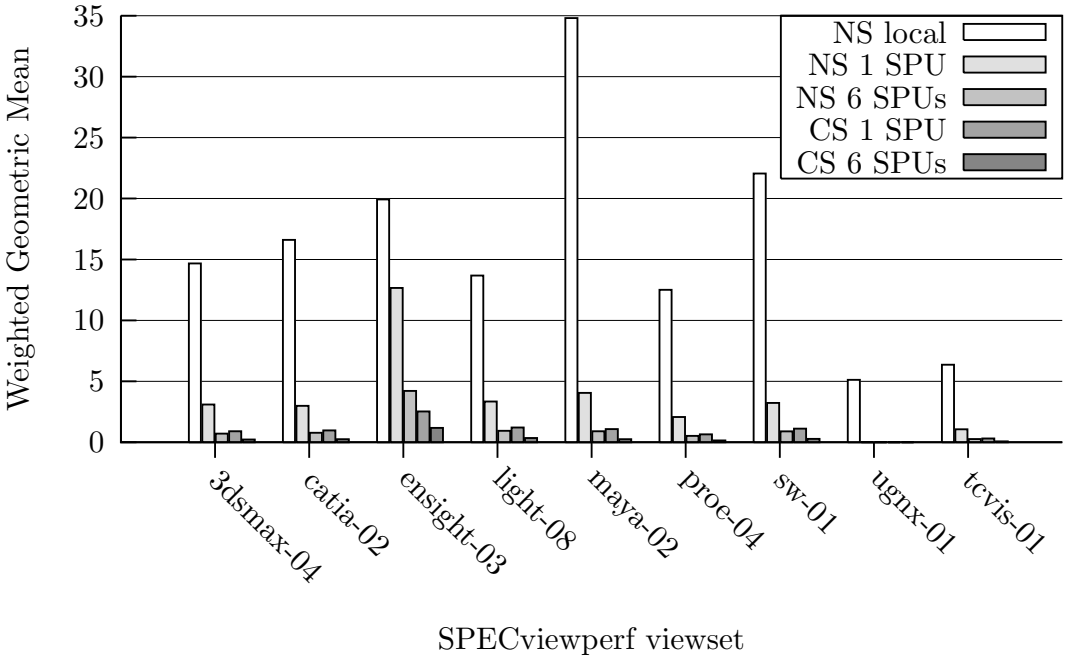

Figure 6: SPECviewperf® benchmark results. NS refers to our new system and CS to our comparison system, the number of SPUs refers here to the number of Chromium render SPUs.

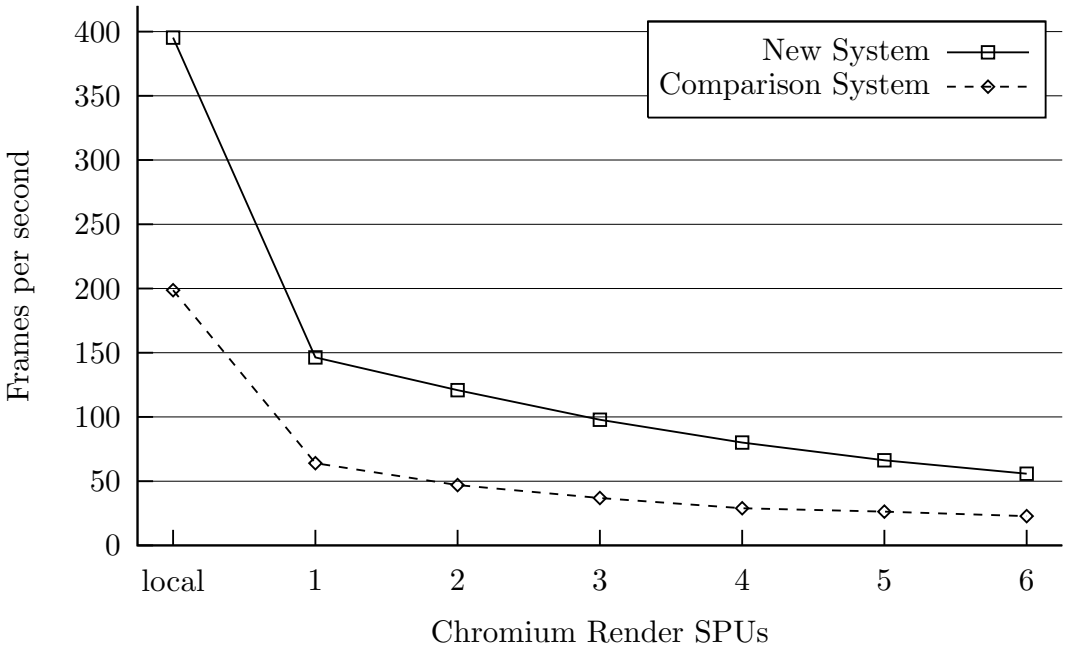

Figure 7: Quake III Arena benchmark results.

means stereo projection onto two walls and the floor. In contrary to the first benchmark now Chromium is used to set the view frustum for each render SPU. Furthermore the tilesort SPU was configured in two ways. On the one hand it can send all graphics commands to each renderclient, on the other hand it can perform culling and send only the graphics commands which are needed to draw what is inside the defined view frustum.

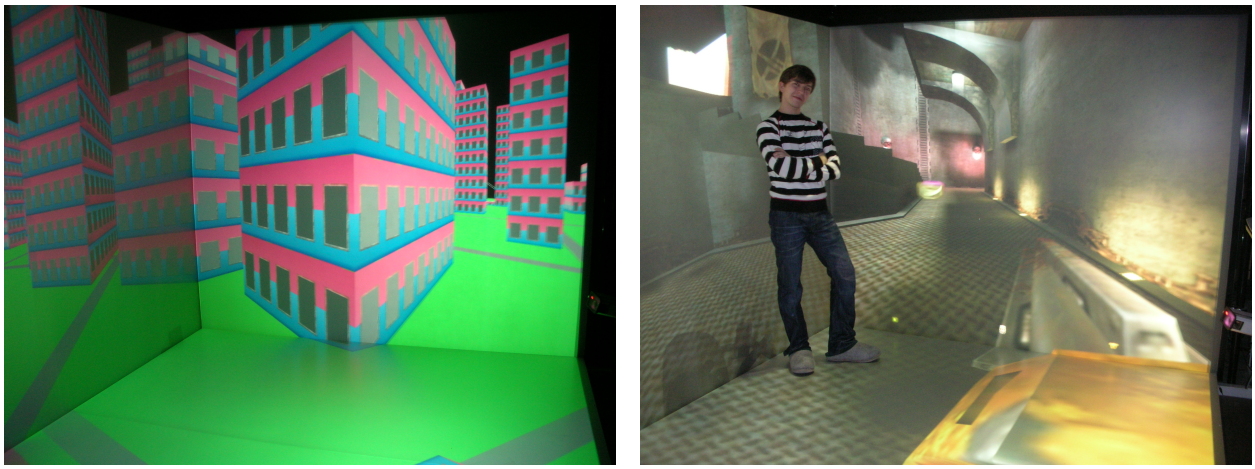

Figure 8: a) Running Chromium city demo and b) running Quake III Arena in our CAVE environment.

We used two benchmark programs. The first one was the city demo, which is distributed with chromium. This program generates a city scene rotating around the user (see figure 8 a) and measures the frames per second rendered. The second one was Quake III Arena's timedemo (see figure 8 b).

To see the influence of the view frustum, we also tested with the city program for differences by using the same view frustum for all render SPUs in comparison to three different view frustums like in our CAVE environment.

5.1 Results

The results for the city demo (table 2) still show that the new system outperforms the comparison system. Furthermore it shows that the feature of Chromium's tilesort SPU to cull before sending the graphics commands to the renderclient improves the performances a lot. It even slightly improves the performance to look into three different directions like in our CAVE environment. Projecting onto the floor in case of the city demo means a great amount vertices being culled, but this may very for different scenes.

The Quake III Arena's timedemo results show also an improvement in framerate if the tilesort SPU is culling. But since the Quake III Arena engine already performs culling, which cannot be disabled, the results do not show the same improvement as using culling with the city demo.

	Broadcast		Frustum	
	1 View	3 Stereo-views	1 View	3 Stereo-views
New system: city FPS	71.37	66.94	162.56	176.3
Standard deviation	1.68	2.31	9.42	7.08
Comparison system: city FPS	34.39	34.12	84.98	92.48
Standrad deviation	0.05	0.13	4.7	2.63
New system: timedemo FPS	—	54	—	88
Comparison system: timedemo FPS	—	22	—	36

Table 2: Results of running Chromium's city program and Quake III Arena's timedemo in a CAVE-like setup (3 stereo-views). Note that since Quake III Arena has been already excessively tested during the first benchmark it is run here only with different views.

6 Benchmark III

Finally the third benchmark has been designed to evaluate the network performance and to examine the slowdown of the system when using Chromium. The first test simply measures network throughput. Chromium provides a small test-suite to perform this task. The npclient program will run on the application server and send packages to npservers over the network. Again here we compared the new low-cost system to the network cluster system.

Another idea is to test different Chromium configurations, how the graphics commands are distributed. The first one is to ignore Chromium's client server architecture and just to have a render SPU at the application node, so that there is no network distribution. The second one is to have a pack SPU at the application node which simply packs the graphics commands and sends them over a network connection—in this and the following cases over the loopback device—to a server node with a render SPU. The third and fourth configuration run a tilesort SPU at the application node which distributes the graphics commands to a server node with a render SPU. The two tilesort configurations differ in the point of distribution over the network: The frustum and broadcast method were tested. As application we used again Quake III Arena's timedemo so the results tie in with the preceding benchmarks.

6.1 Results

The network performance test results (table 3) show that the comparison system's throughput is limited by the Gigabit Ethernet (5 runs, tiny variation). In contray the new system's limits (10 runs, huge variation) could result from packing and unpacking the data.

The overall performance is also affected Chromium's distribution method configuration (figure 9). If there is no distribution over a network the application performs as without Chromium. If Chromium is configured with an application node that just packs the graphics commands and sends them over a network connection there is a first big performance loss of

Bytes/buffer	10000	100000	1000000	10000000	100000000
New system (MB/s)	916.15	1325.4	894.33	701.32	676.58
Std. dev.	219.93	394.24	124.03	72.86	75.21
Comparison system (MB/s)	102.96	101.29	98.73	98.72	99.7
Std. dev.	0.13	0	0	0	0

Table 3: Total network performance measurement with Chromium.

nearly 100 FPS (25%). Furthermore if Chromium is used for culling there is an even greater performance loss of about 230 FPS (57.5%).

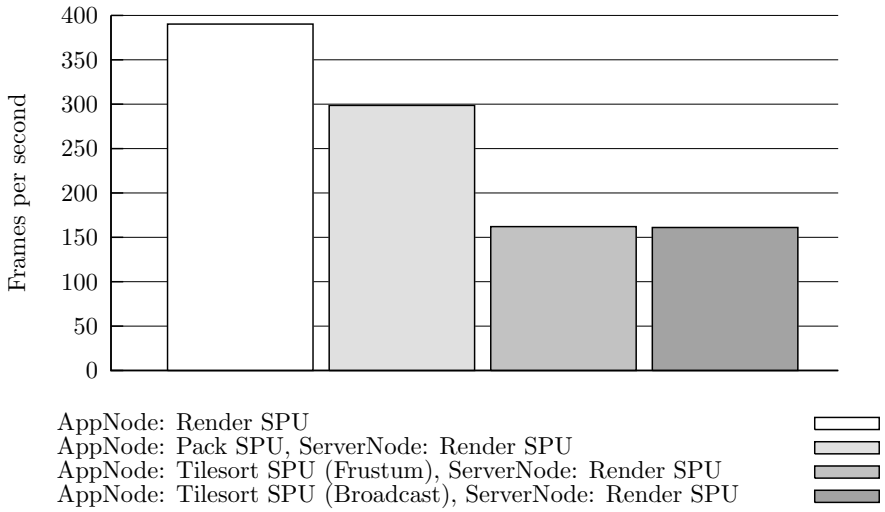

Figure 9: Running Quake III Arena's timedemo with different Chromium configurations.

7 Conclusion & Future Work

In this Paper we have shown a low-cost image-generation solution for immersive multi-screen environments, using a single computer based on regular consumer hardware. The presented system feeds up to 6 independent screens and can be extended to feed up to 12 screens using NVIDIA Quadro 4500 graphics boards. The system outperforms our current setup, which is based on a networked client-server architecture by 120%. Still, the initial performance gain of our prototype is not as high as we first expected. First of all one bottleneck is based on the sysytem's architecture. Though the Chromium renderserver as well as the application itself are running on different CPU cores, they share the system's main interconnection and memory access. Other than with dedicated SMP systems, which often provide crossbar-like architectures between the CPUs and the periphal components, I/O and interrupt/signal-handling in consumer hardware is usually managed by only one node and possibly core. This can result—as has already been discussed in sections 2 and 3—in several bottlenecks at the communication channel from the CPU to the main memory and the periphal components

(the graphics cards). Secondly, another bottleneck is caused by Chromium's distribution paradigm. Distributing OpenGL commands to three internal graphics cards using the IP-stack is a major overhead, especially for that type of architectures, where main—even local loopback—network access may be dedicated to one specific core.

But nontheless the systems performance was all in all very satisfying. Consdering the circumstances—taking into account the bottlenecks discussed above—the results are impressive. Additonally thinking of the low costs, compared to a networked cluster or dedicated image generators, it presents a very reasonable alternative with a high performance.

Future Work: Future work consists of using different graphics distribution paradigms, for example a distributed scene graph or distributed display-traversal. The performance gain towards a networked system should be higher than with Chromium's distributed OpenGL technique. Since Chromium's distribution paradigm is not optimal for our system, we are trying to implement some changes, to reduce the high data of OpenGL commands flow over the IP-stack. In the next step we are going to compare this setup with a high performance Infiniband network architecture.

References

[AMBR05] Jérémie Allard, Clément Ménier, Edmond Boyer, and Bruno Raffin. Running large vr applications on a pc cluster: the flowvr experience. In *Immersive Projection Technology*, October 2005.

[Com07] The OSG Community. OpenSceneGraph. http://www.openscenegraph.com, April 2007.

[DRE06] Aidan Delaney and Karina Rodriguez-Echavarria. A low-cost linux based graphics cluster for cultural visualisation in virtual environments. In *Linux 2006, University of Sussex, Brighton*, june 2006.

[HHN+02] G. Humphreys, M. Houston, Y. Ng, R. Frank, S. Ahern, P. Kirchner, and J. Klosowski. Chromium: A stream processing framework for interactive graphics on clusters, 2002.

[OSG] OpenSG. http://www.opensg.org.

[RVB02] Dirk Reiners, Gerrit Voß, and Johannes Behr. OpenSG: Basic Concepts. www.opensg.org/OpenSGPLUS/symposium/Papers2002/Reiners_Basics.pdf, february 2002.

[Sta07] Standard Performance Evaluation Corporation. SPECviewperf® 9. http://www.spec.org/gpc/opc.static/vp9info.html, April 2007.

[Tra99] Henrik Tramberend. A distributed virtual reality framework. In *IEEE Virtual Reality Conference*, pages 14–21, 1999.