The VVand: A Two-Tier System Design for High-Resolution Stereo Rendering

Abstract
We present a high-resolution tiled display designed for optimised stereo output. The display is driven by a two-tier cluster with InfiniBand interconnect consisting of dedicated nodes for display and rendering each. The network structure is aimed at providing increased bandwidth in accordance with data aggregation towards the display. This way we optimise the available compute power per output pixel and enable the future scaling of compute resources via additional nodes.

Author Keywords
Tiled display, high-resolution display, graphics cluster.

ACM Classification Keywords

Introduction
The data sets which visualisation researchers need to tackle today are steadily growing. This is a direct consequence of sensors like digital cameras or industrial computed tomography (CT) improving in resolution. More importantly, the computational power available to engineers, physicists, chemists, or biologists through clusters is also continuously increasing, which allows them to run ever larger simulations. As a result, not
only the amount of data that needs to be processed by the visualisation increases, but oftentimes also the visual complexity of the representation of the data. From that the interest in output devices with more pixels arises. As the number of pixels available through a single display is only growing slowly, systems providing a lot of screen real estate are usually tiled displays.

Such systems are currently either built using arrays of liquid crystal displays (LCDs) or various kinds of video projectors. Both approaches have advantages and disadvantages: the main advantages of LCD-based solutions are the installation and operating costs. The relatively low price of a single unit makes it easy to achieve a huge overall resolution, and compared to projectors maintenance and operating costs are negligible [1]. The main disadvantage of LCD walls are the visible bezels between the screens which thwart immersion, but also diminish the need for a good colour and brightness matching of the devices.

Systems like the original CAVE use video projectors which can be blended seamlessly to achieve highest levels of immersion [2] [3], often increased by stereoscopy. However, projection-based tiled displays are expensive in acquisition, difficult to calibrate, and have high operating costs. These are due to the high energy consumption of the projectors themselves and the resulting cooling requirements and to direct maintenance of the devices – mainly lamp replacement costs.

From the software side, only small tiled displays can be powered by a single computer and thus just run arbitrary existing visualisation programmes. Most powerwalls require a graphics cluster, be it only for providing a sufficient number of display connectors for all the projector inputs. The software running on these clusters is frequently custom-made, e.g. by extending existing in-house code to run on a cluster. Comprehensive visualisation tools like ParaView sometimes provide built-in support for running on clusters, too [4].

A variety of frameworks and libraries has been suggested to aid developers in writing software for graphics clusters or extending the available programmes. The Holy Grail of cluster middleware is the transparent parallelisation of any existing software without source code access – a goal that Chromium comes very close to by intercepting and distributing the OpenGL command stream to rendering servers which then execute it [5]. Others try to mimic widely used APIs like the OpenGL Utility Toolkit (GLUT) as Doerr and Kuester do with their Cross Platform Cluster Graphics Library (CGLX) [6]. On the implementation side, CGLX follows the synchronised execution pattern running an instance of the same application on every node. Few of the proposed libraries have found wider distribution like the CAVElib or Equalizer, which have become commercially supported products [7]. The advent of high-speed network interconnects like InfiniBand (IB) made distributing high-resolution imagery in real-time a feasible solution. The Scalable Adaptive Graphics Environment (SAGE) implements a window manager for tiled displays which can stream imagery via such high-speed networks from various sources [8].

**Design Goals**

An important application area we designed our tiled display for is the visualisation of large particle-based simulations like molecular dynamics [9]. The rendering that we employ for those data sets (see Figure 4) is ray-casting of glyphs on the graphics processing unit...
**Figure 1:** Schematic view of setup with two exemplary vertices and their projections on the virtual screen. Dimensions in scene coordinates are subscripted \( v \) while real-world coordinates are denoted with \( r \).

Positive parallax \( p \): the object is perceived as being behind the virtual screen.

Negative parallax \( p' \): the object is perceived as being in front of the virtual screen.

In stereo output the illusion of depth comes from two slightly different images of a vertex at distance \( v \) generated by two virtual cameras with an offset of \( e_v \). This difference is the parallax \( p \), which is zero for vertices at the convergence distance \( c \). To avoid squinting the maximum parallax \( p_{\text{max}} \) must be at most the distance \( e_v \) of the eyes [10]. Objects appear infinitely far away in this case. While the virtual coordinates are almost freely scalable, their real-world equivalents are not: the inter-ocular distance \( e_c \) is defined by human physiology and about 6.5 cm; hence the maximum parallax in screen space is fixed, too, and discretised according to the pixel density of the display [11]. With \( R \) being the horizontal resolution of the display,

\[
d(x) = \frac{p}{[x]} c, x \in [-p_{\text{max}}, p_{\text{max}}], \text{ with } p = \frac{e_c R}{w_r}
\]

denotes the perceived depth of a vertex. Figure 2 illustrates the few achievable depth steps \( d(x) \) for a lower resolution display resulting in what we call the "card-board standup effect". Figure 3 depicts the improved depth discretisation for our system. Hence, our design goal of a pixel size equivalent to standard desktop monitors does not only come from the need to display significant amounts of information, but also from improving the "displayable discretised depth resolution" of the system.

Our applications are frequently rasterisation-bound ones, most notably volume rendering and the above mentioned GPU-based ray-casting. The goal for our graphics cluster therefore was not falling below the GPU per pixel ratio of our workstation systems, ideally providing more computational power and keeping the installation scalable. A remote rendering approach with multiple GPUs in each node should fulfil these requirements. The network required for any remote rendering solution should provide sufficient bandwidth to allow for
a minimum of 20 – 25 frames per second, while employing at most lossless compression methods. We believe that compression artefacts contradict the idea of a high-resolution display. While being primarily designed for driving the tiled display, the cluster should be a multi-use system, which can be employed for general-purpose computing (using CUDA) or even for traditional CPU-based parallel applications as well.

**Hardware Setup**

**Tiled Display**

Our display is made of five portrait-oriented strips, thus avoiding that more than two projectors overlap at any point. For each strip, we use two of JVC’s DLA-SH4K projectors – one for each eye –, which results in a net resolution of 10,800 × 4,096 pixels per eye that are projected on a screen the size of approximately 6 × 2.2 m. The projectors are mounted in upright position and the image is deflected by a mirror in order to achieve the portrait orientation while operating the device within its specifications (Figure 7).

The separation of the stereo channels is realised using interference filters (Infitec). As Infitec filters the colour spectrum to achieve separation, the lack of colour fidelity is its main disadvantage. Conversely, it is a passive technology which does not require refresh rates above 60 Hz like shutter glasses, and it can be used with a nearly-Lambertian screen material — in contrast to polarisation. The latter is important for avoiding hot spots which would make the seams between the projectors clearly visible [12]. Such a diffuse screen material does, however, also result in reduced image sharpness, which in turn is disadvantageous when the goal is achieving a high pixel density. Therefore, the pixels on
our screen are hardly discernible even if standing directly in front of the screen (Figure 6).

**Graphics Cluster**
Our graphics cluster comprises two parts: a display cluster, which the GPUs of are connected to the projectors, and a rendering cluster, which is solely used for off-screen rendering. All nodes are connected with a Gigabit Ethernet network for management purposes and a DDR InfiniBand (IB) interconnect with full bisectional bandwidth as application network.

The ten display nodes are each equipped with two NVIDIA Quadro 6000 GPUs and one GSync board that provides frame locking capabilities. Four outputs of each machine are connected to one projector to power all of its quadrants. These nodes also contain two dual-port IB adapters, thus providing a dedicated port for each active GPU output.

The rendering cluster consists of 64 nodes, which are equipped with two NVIDIA GeForce GTX480 each – mainly for budgetary reasons. These nodes also have only one IB port reflecting the fact that we expect a large number of rendering nodes sending small image portions to a few display nodes. Thus, the image composition pyramid is reflected in the available interconnect bandwidth.

**Software**
While our display nodes provide a reasonable amount of computational power to directly run applications in a synchronised execution mode, the whole setup is designed for remote rendering and transferring the final pixels. A last step for all applications, therefore, is copying their imagery from the rendering nodes to the correct location in the “distributed frame buffer” of the tiled display and presenting it. Although we do want to allow application developers to freely experiment with workload partitioning schemes on the render nodes, we nevertheless provide a client library which implements this last step for several reasons besides avoiding repeated work: we want to relieve the user from establishing the correct number and type of network connections for transferring the pixels. Moreover, we want this part to be exchangeable to e.g. transparently change between IP over InfiniBand (IPoIB) and RDMA transfer. There is furthermore a variety of points to tweak like whether small image rectangles are coalesced for transfer or not, whether they are compressed or not, whether the upload on the display servers is done by mapping the whole texture or changing only a subpart etc. Table 1 lists e.g. the beneficial influence of the lossless LZO compression on the frame rate. However, when the image contains two colour channels with random data, the compression ratio drops below 1 : 2 and the required additional CPU time by far outweighs the benefits of smaller packets. Finally, the hardware frame lock is notoriously difficult to get done right for large systems like ours. To summarise, even someone developing new distributed rendering methods on a GPU cluster does not want to care about these details when using the VVand.

**Conclusions**
We built a high-resolution stereo projection system powered by a scalable two-tier cluster. With respect to the goal regarding stereo presentation for our applications results are satisfactory, and the network design works better than anticipated. The resolution limit for a rear-projection on a diffuse screen has been reached

Table 1: Average frame rates for a tile of 4096 × 2048 px remotely rendered on four nodes. The overall time includes the download of the image from the GPU, the “network-only” column measures only the time the application runs in our client library. The compression ratio for the 2D gradient test image used was about 1 : 10.

<table>
<thead>
<tr>
<th></th>
<th>Overall</th>
<th>Network-only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw</td>
<td>37.47 fps</td>
<td>44.70 fps</td>
</tr>
<tr>
<td>LZO</td>
<td>59.26 fps</td>
<td>70.59 fps</td>
</tr>
</tbody>
</table>

Figure 7: View of the projection room.
with our current setup as individual pixels already start to blur (see Figure 5).

Practical experience shows that building nodes with two consumer GPUs for off-screen rendering is rarely possible: running two X servers under Linux only works with selected driver and X server combinations, while under Windows the second GPU cannot be addressed using OpenGL due to the unavailability of vendor-specific GPU affinity extensions. Only Direct3D 10+ and CUDA can explicitly address all hardware.

We believe that we did not yet hit the sweet spot for image transmission between the two parts of the cluster. For future work we would like to add automatic configuration of the transmission mode based on continuous measurements. For information visualisation applications, we already started adding bindings for Windows Forms, WPF, and Java which enable copying directly from their graphics contexts.

References


