

NPU-Based Graphics Composition

LINE-RATE GRAPHICS COMPOSITION FOR LARGE-SCALE DATA-VISUALIZATION CLUSTERS

Visualization is an integral part of scientific computation and simulation. Large-scale data visualization often involves workload distribution across a cluster of computational nodes because the data sets are too large to fit on a single graphics processor. This situation results in the generation of a series of images that must be composited to form the final image to be displayed.

Previous CPU-based image composition reuses the rendering nodes for composition¹. Each image is broken into subareas and sent to the node responsible for compositing the images for that subarea. Given an m -node cluster, each node must send not just the n bytes of data that it has rendered but receive n bytes of data to composite and output n/m bytes of final data.

With video card I/O performance increasing quickly, we believe this internode I/O will become a crucial bottleneck in cluster-visualization performance. By offloading the network-intensive compositing task onto this network processor, we reduce the network bandwidth requirements for each node by more than 50% and also remove the image-composition task from the CPU workload.

Graphics compositing is a network-intensive bottleneck for viz clusters. Common compositing operations such as Z buffering and alpha blending perform simple computations on enormous amounts of data. We use network processors to achieve line-rate graphics compositing in the network.

We are using off-the-shelf hardware, an Intel IXDP2800 Development Platform with two IXP 2850 processors.

Each 2850 has 16 RISC microengines capable of running a total of 128 threads on a single die. Each microengine runs at 1.4 GHz and is responsible for the computationally intensive portions of this application. There is no cache and all memory management is done explicitly.

We developed our graphics compositing functionality using Teja's development environment for the IXP. Viz nodes send UDP packet streams to the network processor. These full-packet payloads are loaded into DRAM and transferred to local memory and registers on each microengine. Using 32 parallel threads, we then construct a new output packet stream to be displayed by the display node. We use 8-bit integer values for RGB, Z, and Alpha.

GRAPHICS COMPOSITION

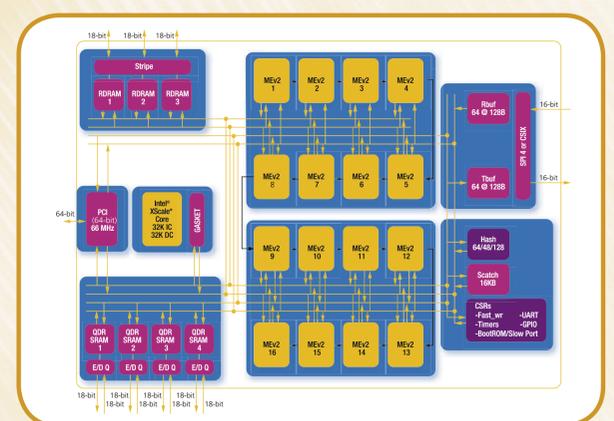
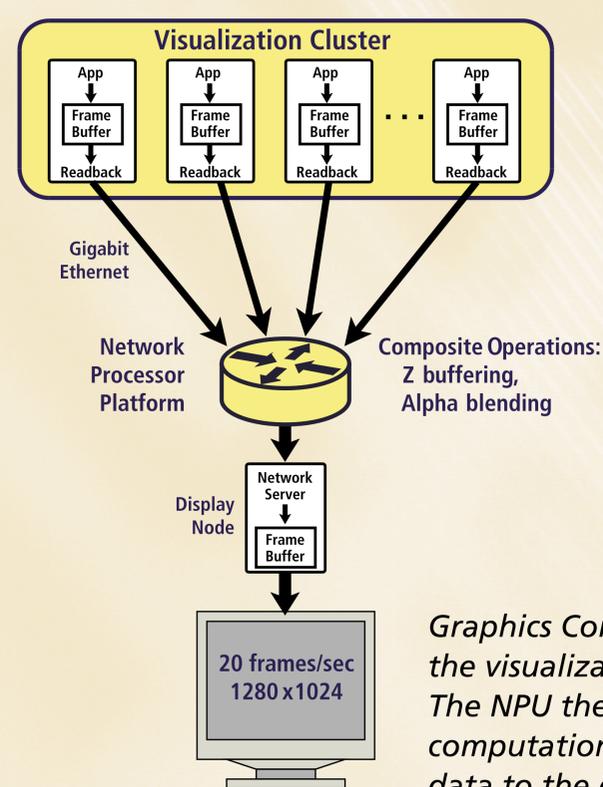
Z buffering - Each input pixel's RGB value is accompanied by a Z value representing the distance of that pixel's color from the viewer. To compose the image, we find, for each pixel location, the input pixel that is "in front."

Alpha blending - Each input pixel has not only RGB and Z values but an alpha (α) value representing the pixel's transparency. Using the Z values, we find the order of the input pixels from back to front and blend iteratively according to the following formula:

$$p = p_{\text{front}} \cdot \alpha_{\text{front}} + p_{\text{back}} \cdot (1 - \alpha_{\text{front}})$$

MORE INFORMATION

Networked Systems Research Team
<http://public.lanl.gov/netsys> ■



IXP Architecture (figure by Intel).

Graphics Composition Architecture. Data flows from the visualization cluster to the network processor. The NPU then performs the Z-buffer or alpha-blending computations and sends the final composited image data to the display node. All this is performed at line rate.

1. Interactive Texture-Based Volume Rendering for Large Data Sets, J. Kniss, P. McCormick, J. Ahrens, J. Painter, A. Keahey, and C. Hansen, *IEEE Computer Graphics and Applications*, July/August 2001; v.21, No. 4, p.52-61