Visualization Walls Using Commodity Clusters: Rocks Viz Roll

Student: Aaron Robinson
Advisor: Dr. Bruce Segee

August 4, 2009

Abstract

High-resolution data visualization is used for viewing large data while maintaining finer detail. The need for high-resolution arises from the very large data sets generated by many scientific models. If combined with building clusters from commodity hardware, this tool can be available to a larger number of researchers. This paper discusses using the Rocks Linux cluster distribution for easily creating visualization walls from commodity hardware. In addition, it addresses our experience at the University of Maine with implementing the framework on both newer and older hardware.

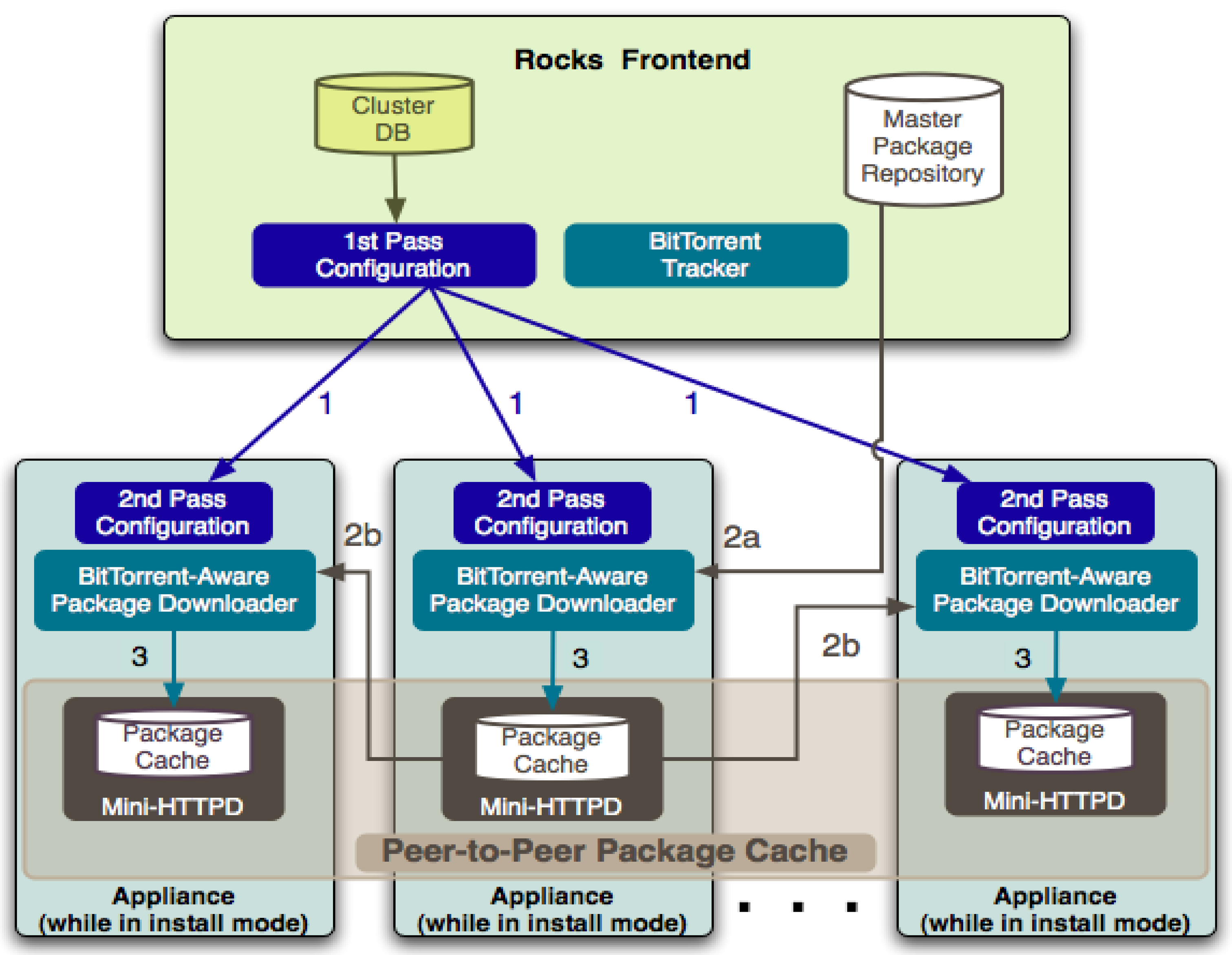


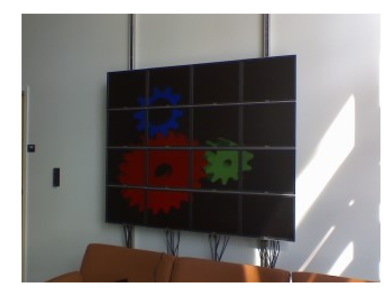
Figure 1: Rocks Avalanche Installer Process Flow (From [3])

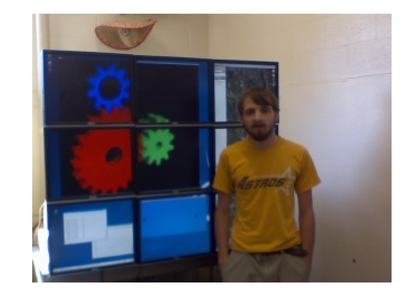
Rocks

When Rocks is installed on the front-end machine, an SQL database and a master package repository are created for streamlining software installation on new cluster nodes. When a machine connects to the front-end via CD or PXE boot, the front-end makes a first pass configuration. After which the new node makes a second pass configuration which generates a Red Hat Kickstart File. Kickstart is an installation method which automates parts of installing a new OS, such as partitioning and boot loader configuration [4]. Once the new node has its Kickstart File, it begins downloading packages from the master package repository using a peer-to-peer file sharing package downloader. As more nodes connect to the front-end, they are configured using the same method as the first node. However, instead of getting packages from the front-end the new machines use peer-to-peer file sharing to download the cached packages from another node (See Figure 1). Once installation is done, all machines have the same configuration even if the cluster is heterogeneous. This has a distinct advantage over traditional imaging because this method only works for homogeneous clusters [3].

DMX & Xinerama

DMX, also refereed to as Xdmx, provides support for multiple displays from multiple machines. When combined with Xinerama, the user is provided with a single unified display. DMX works by creating an X proxy on the front-end machine which stitches together the X servers on the cluster nodes (See Figure #) [4].





SAGE

SAGE is used to stream graphics to a display which is driven by a render cluster while providing a collaborative environment. The objective being to create an interactive environment for multiple users on high-resolution displays which allows for collaboration on multiple distinct data sets [9]. SAGE is composed of the Free Space Manager, SAGE Application Interface Library (SAIL), SAGE Receiver and User Interface. The key component is the Free Space Manager which handles all messaging related to SAGE. It interacts with the UI by sending SAGE status messages and receiving user commands. SAGE Receivers and SAILs communicate status messages to the Free Space Manager. When a SAGE application is executed, a SAIL is created for each rendering node. Each display node has a SAGE Receiver. Pixels are streamed from SAILs to the SAGE Receiver where they are displayed (See Figure 2)[10].

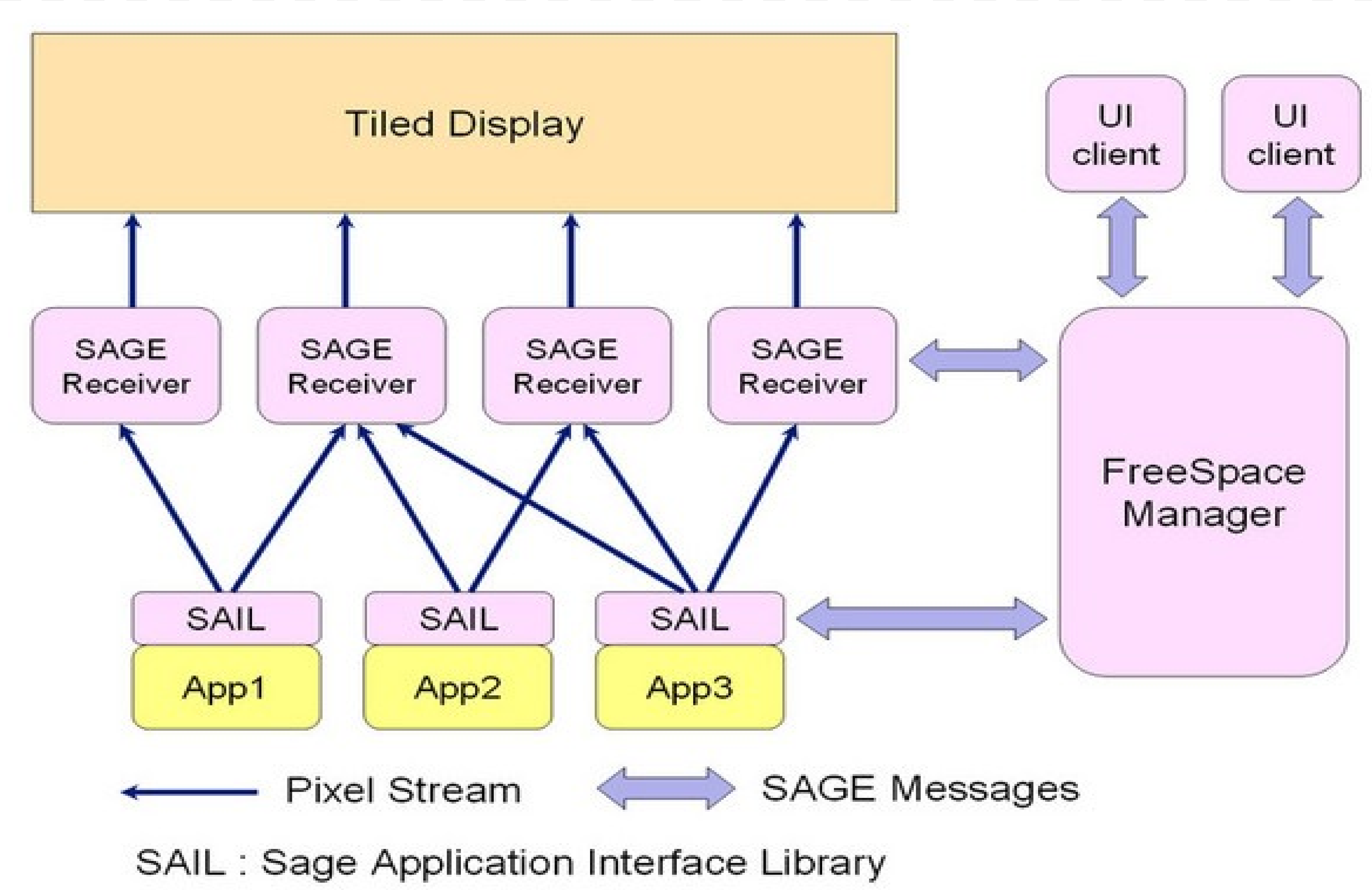
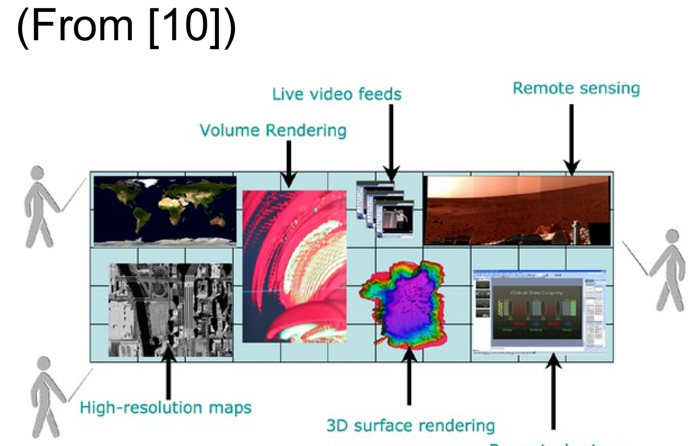
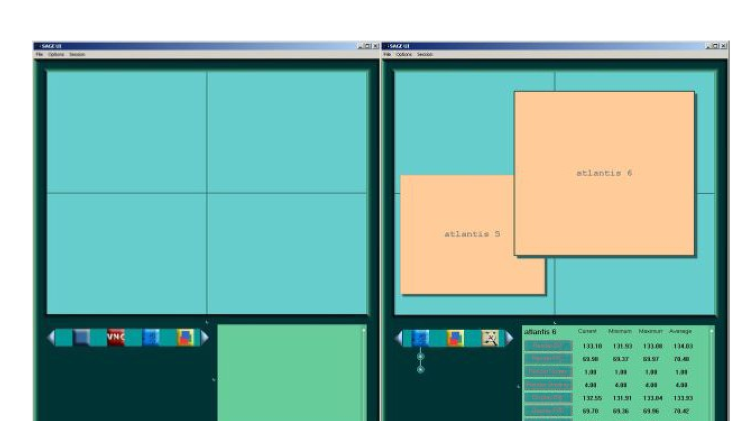


Figure 2-4: SAGE Framework, SAGE Free Space Manager, and Sage UI (From [10])





Chromium

Chromium manipulates OpenGL command streams which allows rendering to be done on a graphics cluster [6]. The basic unit of Chromium is the Stream Processing Unit (SPU). When Chromium is used to render to a display wall using DMX, each back-end runs a crserver which connect to the front-end running the mothership. The crserver provides a render SPU which will be used for doing any rendering on that part of the display wall. After all the back-end machines have connected to the mothership, the crappfaker is started which manipulates the OpenGL. The crappfaker configures a tilesort SPU (See Figure 3). When OpenGL calls are made, they execute Chromium's libGL faker rather than the normal libGL. If a new window is created, the tilesort SPU communicates with DMX to get the window position and then the SPU directs the rendering to the back-end render SPUs [7].

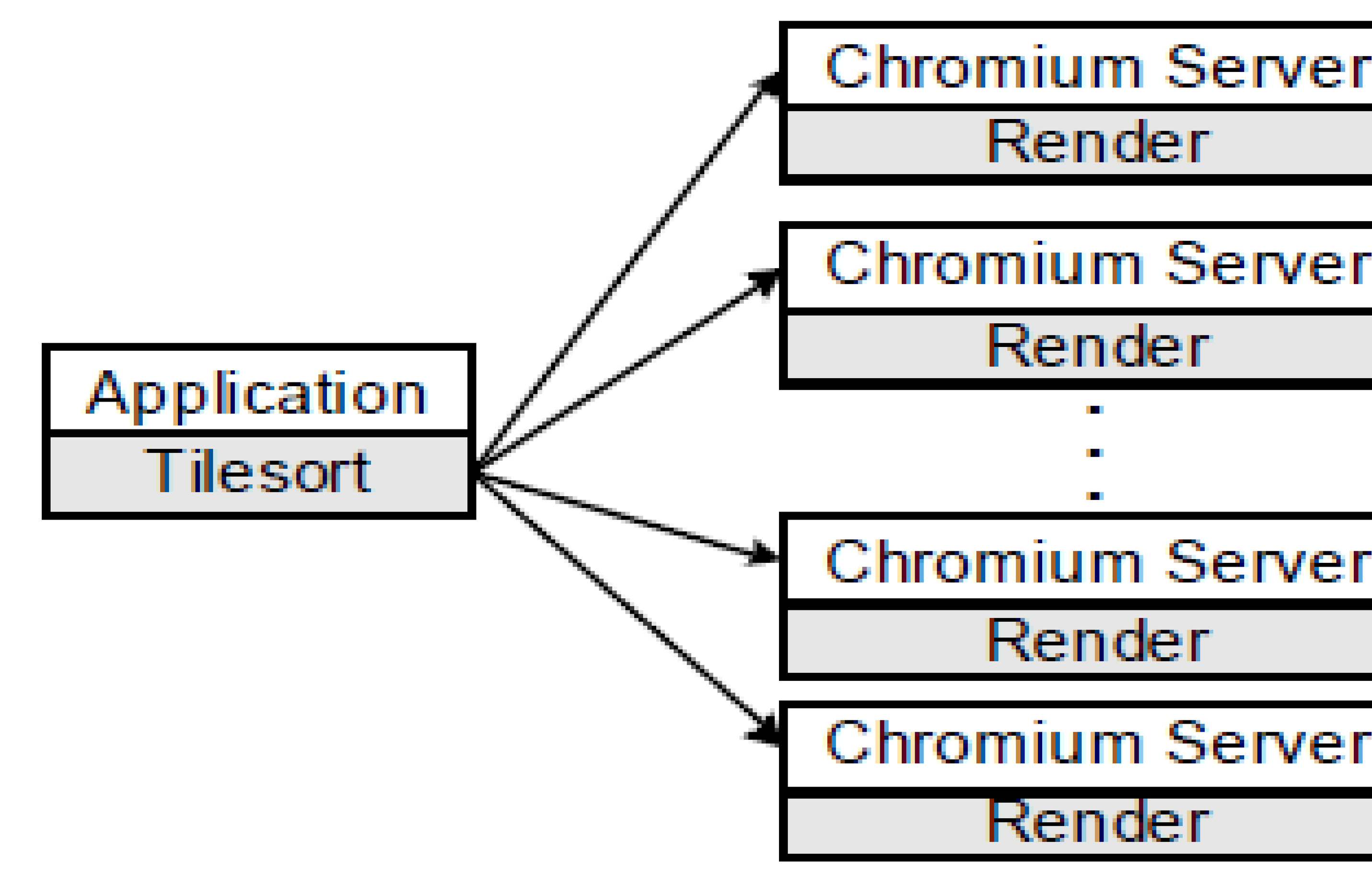


Figure 5: SPU configuration for a DMX display wall (From [8])

References

[1] G. Bruno, M. J. Katz, F. D. Sacerdoti, and P. M. Papadopoulos, "Rolls: Modifying a Standard System Installer to Support User-Customizable Cluster front-end Appliances," presented at IEEE International Conference on Cluster Computing, San Diego, USA, 2004. Available: http://www.rocksclusters.org/rocks-doc/papers/ieee-cluster-2004/paper.pdf. [Accessed: Jul. 20, 2009].

[2] "Rocks Clusters: About," [Online]. Available: http://www.rocksclusters.org/wordpress/?page_id=57. [Accessed: Jul. 20, 2009].

[3] "The Rocks Avalanche Installer," Jun. 19, 2008. [Online]. Available: http://www.rocksclusters.org/rocks-doc/papers/two-pager/paper.pdf. [Accessed: Jul. 29, 2009].

[4] "How Kickstart Works," Mar. 1, 2004. [Online]. Available: http://kickstart-tools.sourceforge.net/howkickstartworks.html. [Accessed: Jul. 30, 2009].

[5] "Distributed Multihead X Project," Jun. 13, 2004. [Online]. Available: http://dmx.sourceforge.net/. [Accessed: Jul. 30, 2009]

[6] "Chromium Homepage," Aug. 15, 2005. [Online]. Available: http://chromium.sourceforge.net/. [Accessed: Jul. 31, 2009]

[7] "Using Chromium with DMX," Sep. 1, 2006. [Online]. Available: http://chromium.sourceforge.net/doc/dmx.html. [Accessed: Jul. 31, 2009]

[8] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. Kirchner, and J. T. Klosowski, "Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters," Apr. 10, 2002. [Online]. Available: http://graphics.stanford.edu/papers/cr/cr_lowquality.pdf.

[9] "Scalable Adaptive Graphics Environment," [Online]. Available: http://www.evl.uic.edu/cavern/sage/index.php. [Accessed: Jul. 31, 2009]

[10] "SAGE :: DESCRIPTION," [Online]. Available: http://www.evl.uic.edu/cavern/sage/description.php. [Accessed: Jul. 31, 2009]