# Scalable Isosurface Visualization of Massive Datasets on COTS* Clusters

Xiaoyu Zhang† Chandrajit Bajaj† William Blanke‡ Donald Fussell†

† Department of Computer Science,
‡ Department of Electrical and Computer Engineering,
University of Texas at Austin

## Abstract

Our scalable isosurface visualization solution on a COTS cluster is an end-to-end parallel and progressive platform, from the initial data access to the final display. In this paper we focus on the back end scalability by introducing a fully parallel and out-of-core isosurface extraction algorithm. It partitions the volume data according to its workload spectrum for load balancing and creates an I/O-optimal external interval tree to minimize the number of I/O operations of loading large data from disk. It achieves scalability by using both parallel processing and parallel disks. Interactive browsing of extracted isosurfaces is made possible by using parallel isosurface extraction and rendering in conjunction with a new specialized piece of image compositing hardware called Metabuffer. We also describe an isosurface compression scheme that is efficient for isosurface transmission.

**Keywords:** Parallel Rendering, Metabuffer, Multi-resolution, Progressive mesh, Parallel and Out-of-core Isocontouring

## 1 Introduction and Related Works

Today's real time visualization applications increasingly are dealing with larger and larger data streams. In order to find the critical areas of interest within these datasets and understand the underlying physics, it is necessary that the application allows the user to navigate quickly and easily around the object space to explore different views. Given a Scalar Field, $w(\mathbf{x})$, defined over a $d$-dimensional bounded volume mesh ($\mathbf{x} \in \mathbf{R^d}$), we often visualize the data by rendering $(d-1)$-dimensional surfaces satisfying $w(\mathbf{x}) = const$. This visualization technique is popularly known as Isocontouring and is widely used method for volume data visualization.

Isocontour visualization for extremely large datasets poses challenging problems for both computation and rendering with guaranteed frame rates. To achieve scalable isosurface visualization in a time critical manner, we decompose the time-critical isosurface extraction and rendering into three pipeline stages2.

First, isosurfaces are to be extracted efficiently from those large datasets. As the size of the input data increases, isocontouring algorithms need to be executed out-of-core and/or on parallel machines for both efficiency and data accessibility. Hansen and Hinker describe parallel methods for isosurface extraction on SIMD machines [17]. Ellsiepen describes a parallel isosurfacing method for FEM data by dynamically distributing working blocks to a number of connected workstations [13]. Shen, Hansen, Livnat and Johnson implement a parallel algorithm by partitioning load in the span space [31]. Parker et al. present a parallel isosurface rendering algorithm using ray tracing [26]. Chiang and Silva give an implementation of out-of-core isocontouring using the I/O optimal external interval tree on a single processor [7, 8]. Bajaj et al. use range partition to reduce the size of data that are loaded for given isocontour queries and balance the load within a range partition [3]. In this paper, we propose and implement a parallel and out-of-core isocontouring algorithm using parallel processors and parallel I/O, which would be fully scalable to arbitrarily large datasets.

Second, The interactive aspect of the application demands that the scene is rendered quickly in order to provide responsive feedback to the user. At the same time, if an interesting area is found within the dataset, it is imperative that the application show the scene at the highest level of detail possible. In some cases, the detail allowed by a single high performance monitor may not be adequate for the resolution required. An even more common problem is that the dataset itself may be too large to store and render on a single machine.

Many research groups have studied the problem of parallel rendering [6, 12, 18, 20, 22, 28–30]. Schneider analyzes the suitability of PCs for parallel rendering for four parallel polygon rendering scenarios: rendering of single and multiple frames on symmetric multiprocessors and clusters [30]. Samanta et al. discuss various load balancing schemes for a multi-projector rendering system driven by multiple PCs [29]. Heirich and Moll demonstrate how to build a scalable image composition system using off-the-shelf components [18]. In general, most parallel rendering methods can be classified based on where data is sorted from object-space to image-space [23].

In the sort-first approach, the display space is broken into a number of non-overlapping display regions, which can vary in size and shape. Because polygons are assigned to the rendering process before geometric processing, sort-first methods may suffer from load imbalance in both the geometric processing and rasterization if polygons are not evenly distributed across the screen partitions. Crocket [10] describes various considerations to build a parallel graphics library using the sort-first method. The Princeton University SHRIMP project [29] uses the sort-first approach to balance the load of multiple PC graphical workstations.

The sort-last approach is also known as image composition. Each rendering process performs both geometric processing and rasterization independent of all other rendering processes. Local images rendered on the rendering processes are composited together to form the final image. The sort-last method makes the load balancing problem easier since screen space constraints are removed. However, compositing hardware is needed to combine the output of the various processors into a single correct picture. Such approaches have been used since the 60's in single-display systems [15, 24, 25], and more recent work includes [14, 18].

Our solution to image compositing problem is the Metabuffer, whose architecture is shown in figure 1. This is a sort-last multi-display image compositing system with several unique features such as multi-resolution and antialiasing [5]. A very similar project, though currently without stressing multi-resolution support, exists at Stanford University and is called Lightening-2 [19]. The Metabuffer hardware supports a scalable number of PCs and an independently scalable number of displays–there is no *a priori* cor-

*Common of the shelf