Comparison of Remote Visualization Strategies for Interactive Exploration of Large Data Sets

Lori A. Freitag Raymond M. Loy

Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439
{freitag,rloy}@mcs.anl.gov

Abstract

We compare three remote visualization strategies used for interactive exploration of large data sets: image-based rendering, parallel visualization servers, and subsampling. We review each strategy and provide details for an adaptive multiresolution subsampling technique that we have developed. To determine the problem regimes for which each approach is most cost effective, we develop performance models to analyze the costs of computation and communication associated with the common visualization task of isosurface generation. Using these models, we investigate a number of hardware system configurations and task complexity scenarios when parameters such as problem size, visualization demands, and network bandwidth change. For one particular strategy, subsampling, we further investigate the tradeoffs between multiresolution and uniform grid methods in terms of performance and approximation errors.

Keywords: Remote Visualization, Interactive Visualization, Large Data Set Exploration, Performance Models

1 Introduction

Tera- and petabyte size data sets are becoming more common as scientists gain access to ever increasing computational resources. Interactively exploring these data sets is an extremely challenging task, particularly for scientists whose primary access to visualization resources is a desktop graphics workstation. To address this problem, researchers are exploring a number of approaches that provide interactive navigation and exploration of very large data sets. The approaches combine remote computational or visualization resources with high-speed networks to deliver the images or reduced geometries to the local graphics workstation.

In this paper, we consider three commonly used strategies for performing interactive, remote data exploration: image-based rendering, parallel visualization servers, and subsampling of the original data set. Each approach has inherent advantages and disadvantages, and we give an overview of these strategies in Section 2. We also describe our approach, which is based on an adaptive, general-purpose, multiresolution data subsampling technique.

To determine the most cost-effective strategy for particular system or problem configurations, we examine the performance characteristics of each by developing theoretical performance models. These models estimate the costs of computation and communication when parameters such as network bandwidth, problem size, and visualization demands change. We describe the models in Section 3 and analyze them in Section 4 for a variety of problem scenarios.

For the subsampling approaches, we develop two theoretical models: one for uniform grids and one for multiresolution grids. Uniform grids are advantageous for two reasons: (1) they can be represented with a comparatively small amount of data, and (2) visualization algorithms on uniform grids generally outperform their nonuniform grid counterparts. Therefore, a larger number of grid points can be used with uniform grid subsampling than with multiresolution techniques. However, multiresolution techniques are designed to minimize the approximation error associated with subsampling by placing more points where the data is changing rapidly. To examine the tradeoffs between these two subsampling methods, we compare the results for several different application data sets.

2 Remote Data Exploration Strategies

For each of the three strategies considered for remote data exploration, we briefly describe the fundamental concepts and give an overview of the method's advantages and disadvantages. We then describe a particular data reduction strategy that we have developed for multiresolution subsampling using a parallel octree data structure.

2.1 Overview

Image-based rendering techniques use two or more reference images from multiple viewpoints to reconstruct either the geometry in a scene or new images of the scene as the user's viewpoint changes.[1] In this paper, we consider image warping techniques that use reference images containing color, depth, and surface normal information to "warp" or change the input images to the desired output image (see, for example, [22, 19, 7, 18, 2]). Typically, several reference images are needed to reconstruct the scene for arbitrary viewpoints; if only a few reference viewpoints are available, the reconstructed image is more likely to contain approximation errors or holes generated by surfaces not represented in the original images.

The primary advantage of this technique for remote data exploration is that the amount of data transmitted and manipulated locally is independent of the complexity of the scene or original data set. Thus the costs are fixed as data set sizes increase. In addition, if remote resources are used to generate the derived visualization entities using the full data set, no approximation errors are associated with the reference images. However, as the user rotates the scene or otherwise changes the view perspective, errors associated with the reconstruction process can misrepresent the original data set. In addition, these techniques are still moderately expensive, and even fairly sophisticated hierarchical techniques can require 1.5 seconds/frame using four reference images [2]. Finally, the addition of a temporal dimension, although not significant for the results of this paper, remains an open area of research.

Parallel visualization servers utilize remote computational resources to visualize full-resolution data sets either as the computation proceeds (e.g., [8, 26]) or as a post-processing step (e.g., [3, 4, 16, 9]). The geometries of the derived visualization entities, rather than images, are extracted and communicated to the graphics workstation for display. Typically, lower-dimensional entities such as isosurfaces or streamlines are targeted for use with these systems because their transmission and memory requirements are much smaller than the full-dimensional data set.

The primary advantage of these techniques is that the derived visualization entities have no subsampling or other approximation errors. In addition, once the geometry is loaded into the local graphics workstation, it may be freely rotated or manipulated without reconstruction errors. The primary disadvantage is that the geometries transmitted to the local graphics workstation are functions of the overall problem size. Thus, as the problem size increases, the demands on computational resources, both remotely and locally, and on network bandwidths also increase.

Subsampling and clustering techniques create smaller, full-dimensional data sets by sampling the original data at specified locations or by averaging clusters of points from the original data set. The simplest approach to subsampling is to create a uniform grid representation of the original data set, and this is common in practice. Alternatively, a hierarchical, multiresolution representation of the data can be constructed using, for example, quadtrees or octrees [15, 11, 14], progressive meshes [13, 12], wavelets [23], or other clustering approaches [10, 25]. The level of detail in each region is controlled through a variety of mechanisms, such as error tolerance bounds that control fidelity to the original model, or user input, such as field of view.

[1] We note that simply sending each image as the object is manipulated from the remote resources requires more network bandwidth than is generally available.

The primary advantage of subsampling or clustering approaches is that they are useful for fast, local exploration of the reduced data set. The computation and transmission costs are fixed regardless of the number of visualization tasks. The primary disadvantage is that the maximum resolution of the reduced data set is limited by the memory size and speed of the local graphics workstation. Thus, as the original problem size increases, a smaller percentage of points can be used in the reduced data set, resulting in higher approximation errors. This disadvantage can be somewhat mitigated, at the cost of more communication, by an adaptive approach such as the one described in the next section.

2.2 Adaptive Multiresolution Subsampling Using a Parallel Octree

Our approach to interactive remote data exploration is to use a parallel octree infrastructure to create a general-purpose tool for adaptive, multiresolution subsampling of the original data set. Currently, our system allows file-based scalar field input and inserts each data point into the appropriate leaf octant. That leaf is then evaluated according to a specified criterion and refined if necessary with its associated data points reassigned to the new leaf octants. Our subsampling code is general purpose and requires only spatial coordinate information from the original data set, with no connectivity information necessary. We have successfully used our approach with unstructured tetrahedral and hexahedral, block adaptive, and uniform meshes. We provide default routines to support reductions that meet user-specified bounds in standard or maximum deviation, or to automatically determine the error bound on these quantities given a performance constraint specified by a maximum target number of leaf octants. Additionally, we provide stubs to allow custom, user-defined insertion criteria to ensure wide applicability of our software.

To provide an indication of the error associated with the reduced data set, for each leaf octant we compute and store statistical values such as the standard deviation, σ, and maximum deviation from the mean, e. These values are normalized by the mean to yield σ_n and e_n, respectively. One or both of these values are included as additional scalar fields to be visualized so that the user has an indication of the fidelity of the reduced data set to the original data set. These measures of error also serve to highlight potential regions of interest; the cells with a large deviation from the average value are likely to have fine-scale structure that was not adequately captured by the reduction process. The computed scalar fields are either stored to a file for later processing or communicated directly to the graphics workstation for visualization.

The graphics application we have developed uses JAVA Swing components [5] to provide a GUI to vtk classes [24]. The user interface supports adaptive level-of-detail requests to the parallel octree code so that the user may interactively change the leaf criterion and thereby the resolution of the reduced data. The new criterion may be applied either globally or in a specified region of interest. In this way, the user can "zoom in" with high-resolution views in local subregions without sacrificing graphics performance. Communication between the octree code and the desktop graphics application is performed using the ALICE Memory Snooper (AMS) [1] from Argonne National Laboratory, which uses a client/server model based on TCP/IP and Unix sockets.

Additional details about the parallel octree algorithms, our software architecture, and results obtained on large data sets can be found in [6].

3 Performance Models

For each of the remote data exploration strategies discussed in Section 2, we now develop a theoretical performance model describing the computation and communication costs. The visualization task used in these models is isosurface generation, which we chose for three reasons:

1. Isosurfaces are one of the most commonly used visualization techniques for exploring scientific data sets.
2. Much research in the visualization community has targeted efficient isosurface generation for both uniform and multi-resolution data sets and for both serial and parallel computers.
3. The fact that isosurfaces are lower-dimensional entities ensures that all three strategies are fairly considered.

Other commonly used visualization techniques such as cutting planes, streamlines, vector glyphs, and volume visualization could be easily modeled by replacing the isosurface-specific information.

The costs included in our models include the time to compute a specified number of isosurfaces, the time to transmit information over a wide area network to the local graphics workstation, and, for the subsampling techniques, the time to compute the reduced data sets.[2] Because the data sets of interest are large, we assume that the scientist has access to a remote parallel computer and that any remote isosurface or subsampling computations are done scalably in parallel. We assume that the original data is preloaded and distributed across the processors of the remote parallel computer, because this is common to each approach and does not differentiate the models. For all models, we assume that the original computational mesh is nonuniform.

The parameters and cost variables used in our models are defined in Table 1. To define the hardware characteristics of our remote visualization system, we use P to define the number of remote processors available for parallel computation and R and L to define the network bandwidth and latency in Mbs and seconds, respectively. The data set size, N, and the number of isosurfaces to be computed and rendered, I, define the complexity of our visualization task. The parameters X and N_R define the size of the images in pixels and the reduced data sets for image-based rendering and subsampling, respectively. The parameters C_{R_u} and C_{R_m} give the total serial cost of subsampling for uniform and multiresolution grids, respectively. The parameters C_{I_u} and C_{I_m} give the cost per element of isosurface generation on uniform and multiresolution grids, respectively. A detailed analysis of these costs will be given in Section 4.

[2]Our models do not address the costs associated with loading and processing the resulting data sets on the local graphics workstation.

Table 1: Variables and parameters used in the cost models

Symbol	Definition
P	Number of Remote Processors
R	Network Bandwidth (Mbs)
L	Network Latency (s)
N	Number of Elements
I	Number of Isosurfaces
X	Pixels/Image
N_R	Number of Subsampled Elements
System-Dependent Computational Costs	
C_{R_u}	Uniform Grid Subsampling Cost
C_{R_m}	Multiresolution Subsampling Cost
C_{I_u}	Uniform Grid Isosurface Cost/Element
C_{I_m}	Multiresolution Isosurface Cost/Element

3.1 Model 1: Image-based Rendering

For image-based rendering techniques, we assume that the isosurfaces are computed in parallel and that six depth images are used for reconstruction on the local graphics workstation [22]. To determine the number of bits that must be transmitted for each pixel in the depth image, we use the best-case scenario information given in [17] for a postrendering warping technique. In particular, we assume 24 bits for color, 16 bits for depth, and 8 bits for surface orientation information, for a total of 48 bits per pixel. We assume that a new set of reference depth images is required for each new isosurface generated.[3] The total cost of the image-based rendering technique is

$$M_I = \frac{N\ C_{I_m}\ I}{P} + \frac{48 \cdot 6\ X\ I}{10^6\ R} + 2\ L\ I. \tag{1}$$

The first term gives the time required to compute the isosurfaces in parallel on the original data set. The second term gives the transmission time required to send the depth images associated with each isosurface. The final term gives the network latency for new isosurface requests. We assume that the cost of generating the six depth images is negligible and the volume of information to request new isosurfaces is minimal.

3.2 Model 2: Parallel Visualization Servers

The parallel visualization server also computes the isosurfaces in parallel, so the first term of this model is identical to the first term in Equation 1. The transmission costs for each isosurface are three spatial coordinates for each data point as well as the

[3]We note that if multiple isosurfaces are generated during each request, the number of reference depth images does not necessarily increase, which has the potential to increase the attractiveness of this approach.

connectivity information for each triangle. If we assume that each isosurface contains $N^{\frac{2}{3}}$ triangles, the total cost of the remote visualization server is

$$M_V = \frac{N\, C_{I_m}\, I}{P} + \frac{32 \cdot 6\, N^{\frac{2}{3}}\, I}{10^6\, R} + 2\, L\, I. \qquad (2)$$

In this model we assume 32 bit information for all scalar and integer values and use the fact that the numbers of vertices and elements in a triangular mesh are approximately equal.

3.3 Model 3: Uniform Subsampling

For uniform grid subsampling, we assume that the original data is partitioned such that the reduction operations may be performed with minimal communication and will scale linearly as a function P. The transmission costs for a uniform grid include the cost of sending the origin, grid spacing, and problem size in each of the three dimensions, and the scalar data and error associated with each element. No additional spatial or coordinate information is required.

The total cost of uniform subsampling is

$$M_U = \frac{C_{R_u}}{P} + N_R\, C_{I_u}\, I + \frac{32 \cdot (9 + 2N_R)}{10^6\, R} + 2L. \qquad (3)$$

The first two terms are the computational costs associated with data reduction and isosurface generation on the reduced uniform grid, respectively. The third and fourth terms are the transmission and latency costs for a single request for a subsampled grid, respectively.

3.4 Model 4: Multiresolution Subsampling

The model for multiresolution subsampling is similar to Equation 3. Because we are using an octree data representation, the spatial coordinates and connectivity information for the reduced mesh must be transmitted in addition to the scalar information. Given a number of octant leaves, the number of associated vertices cannot be determined *a priori*. An upper bound for the number of vertices is $8N_R$, which is the case when no vertex is shared between octants. A lower bound for the number of vertices is N_R, which is the case when the vertices are maximally shared by octants, that is, when the leaf octants form a uniform mesh. For the purposes of our model, we assume an average case of $4N_R$. Thus, the total amount of information that must be transmitted is $3 \times 4N_R$ for the spatial coordinates plus $10N_R$ for the connectivity, scalar, and error data. With these assumptions, our cost model for multiresolution subsampling is

$$M_M = \frac{C_{R_m}}{P} + N_R\, C_{I_m}\, I + \frac{32 \times (22N_R)}{10^6\, R} + 2L. \qquad (4)$$

Note the use of the nonuniform grid costs for data reduction and isosurface generation, C_{R_m} and C_{I_m}, in the first two terms of the model.

4 Results

To determine the regimes for which the models developed in Section 3 are most cost effective, we first determine typical values for the parameters C_{I_u}, C_{I_m}, C_{R_u}, and C_{R_m}. We then analyze the performance models for various values of N, P, I, and R and determine the breakeven points between them. Finally, we compare the uniform grid and multiresolution subsampling techniques in terms of performance and approximation error for several different application problems. All experiments were performed on one or more processors of an SGI Origin with 250 MHz R10000 chips, and all timings were performed using the Unix subroutine `gettimeofday()`.

Network bandwidth and latency estimations are based on the vBNS and ESnet networks. Each of these networks consists of OC3 lines for a maximum throughput of 155 Mbs, although expected performance is often far less [21]. Latency measurements listed on the vBNS net traffic web page [20] range from 3 to 40 ms depending on the destination/origination combination. For the purposes of this paper we consider bandwidth rates ranging from R=.5 Mbs to 100 Mbs and use a latency value of $L = 20$ ms.

4.1 Determining Cost Parameter Values

We first determine the cost per element of isosurface generation on uniform and multiresolution grids, C_{I_u} and C_{I_m}, respectively. We used vtk's `vtkContourMarchingFilter`, which uses a fast marching cubes algorithm for uniform structured point sets and a general algorithm for all other mesh types, including our unstructured octree representation [24]. We tested the algorithms on the same uniform data set, changing only the way it was represented in vtk data structures. In particular, we used `vtkStructuredPoints` to determine C_{I_u} and `vtkUnstructuredGrid` to determine C_{I_m}. We present the timing results for computing the isosurface on a cost per element basis for five different problem sizes in Table 2.

Table 2: Isosurface Generation Costs (ms)

N	C_{I_u}	C_{I_m}	$\frac{C_{I_u}}{C_{I_m}}$
20^3	.0036	.0243	6.75
30^3	.0026	.0246	9.46
40^3	.0022	.0240	10.91
50^3	.0018	.0240	13.33
60^3	.0016	.0239	14.93

The cost parameter C_{I_u} decreases as the problem size increases, reflecting the fact that the cost of processing the data structure is far less than the cost of processing cells containing contour values. Thus, as the percentage of cells containing contour data decreases, the average cost per cell decreases. Plotting the results for C_{I_u} shows that the parameter is asymptotically approaching a value of approximately $.0015$ ms/cell as N increases, and this is the value used in our model. In contrast, the cost of using general data structures in vtk, along with the

Table 3: Data reduction costs for uniform subsampling (s)

N	$N_R = 1$	$N_R = 5^3$	$N_R = 10^3$	$N_R = 20^3$	Add. Cost
50^3	.108	.111	.142	.618	$6.57 \cdot 10^{-5} N_R$
75^3	.368	.371	.403	.885	$6.66 \cdot 10^{-5} N_R$
100^3	.873	.874	.908	1.41	$6.94 \cdot 10^{-5} N_R$

Table 4: Data reduction costs for multiresolution subsampling (s)

N	$N_R = 1$	N_R	Time	N_R	Time	Add. Cost
50^3	3.76	512	4.61	32768	6.38	$4.34 \cdot 10^{-5} N_R$
75^3	12.67	512	15.55	23696	18.85	$11.0 \cdot 10^{-5} N_R$
100^3	31.74	512	41.05	18404	43.60	$26.2 \cdot 10^{-5} N_R$

associated virtual function calls, results in the parameter C_{I_m} remaining fixed at approximately $.024$ ms/cell, a factor of 16 greater than C_{I_u}. We note that this factor should be significantly less for routines specific to octree data types and is also implementation dependent.

To determine C_{R_u}, we subsampled a tetrahedral mesh onto a uniform grid of different sizes and monitored the time required to collect the reduced data set averages and compute the normalized standard deviations for each subsampled grid point. We also computed the average and maximum deviation over the entire reduced data set. Timing results in seconds are given in Table 3 for $N = 50^3$, 75^3, and 100^3 and for $N_R = 1, 5^3, 10^3$, and 20^3. Linear least squares analysis for $N_R = 1$ yields a base cost for visiting every point in the original data set of $8.75 \cdot 10^{-7} N$. To obtain the additional cost of data reduction per point as N_R increases, we performed a least squares analysis on each row of Table 3. The results are given in the last column and show that the total data reduction costs grow as a function of both N and N_R. Linear least squares analysis on the coefficient of N_R given in the last column of Table 3 determines the dependence on N. Our final expression for C_{R_u} is

$$C_{R_u} = 8.75 \cdot 10^{-7} N + (6.50 \cdot 10^{-5} + 4.31 \cdot 10^{-12} N) N_R.$$

To determine C_{R_M}, we performed a similar analysis using the octree data reduction technique described in Section 2. The timing results in seconds for various values of N and N_R are given in Table 4. Linear least squares analysis for the $N_R = 1$ case yields a base cost of $3.21 \cdot 10^{-5} N$. Again, the costs of data reduction increase as a function of both N and N_R. A linear least squares analysis on the coefficients from the last column of Table 4 yields our final formula for C_{R_m}:

$$C_{R_m} = 3.21 \cdot 10^{-5} N + (.87 \cdot 10^{-5} + 2.5 \cdot 10^{-10} N) N_R.$$

4.2 Performance Model Comparisons

To determine the regimes for which each model is the most cost effective, we first consider the image-based rendering and the parallel visualization servers models. The computational costs for isosurface generation are identical; the models are differentiated only by the amount of data transmitted. By equating M_I and M_V, we can easily derive the breakeven function that relates X and N:

$$X = \frac{2}{3} N^{\frac{2}{3}}. \quad (5)$$

For pixels sizes less that X, $M_I < M_V$. Note that this expression is independent of both the number of isosurfaces and the network bandwidth and latency. A few representative data pairs are $(N, X) = (128^3, 104^2)$, $(256^3, 209^2)$, $(512^3, 418^2)$ and $(1024^3, 836^2)$. Thus, if the resolution of the images used is fixed, image-based rendering becomes increasingly attractive as the data set sizes increase.

For the subsampling techniques, the costs will far exceed the costs of image-based rendering or parallel visualization servers as $N_R \longrightarrow N$. Therefore, we determine the breakeven points for these models by finding a value of N_R such that the costs of the subsampling model are equal to the smaller of either Model 1 or Model 2 costs. We consider a large number of scenarios, and in Figure 1 we show the breakeven percentages, $\mathcal{P}_N = \frac{N_R}{N} \cdot 100$, for both uniform and multiresolution subsampling. We choose a representative set of base parameters and vary N, P, R, and I to examine the effect on $\mathcal{P}_N$. We choose base values of $R = 10$ and $I = 64$ for all cases; the top two figures show uniform and multiresolution cases for base parameters $N = 128^3$ and $P = 8$, and the bottom two figures show a larger problem whose base parameters are $N = 256^3$ and $P = 16$. Percentage values, $\mathcal{P}_N$, below the curves indicate the regimes for which subsampling is more cost effective than either parallel visualization servers or image-based rendering at a resolution of $X = 512^2$.

For both problem sizes, a given case that employs uniform grid subsampling can use a higher percentage of grid points than can be used with multiresolution grid subsampling. For the smaller problem size, a full-resolution view resampled to a uniform grid is often the most cost effective of all strategies considered. In contrast, the multiresolution subsampling is cost effective for subsampling percentages less than 12 in the smaller problem size, and less than 5 for the larger problem size. In all cases, as N increases, $\mathcal{P}_N$ decreases as a result of the increased cost of creating the reduced data sets. Similarly, as P increases, $\mathcal{P}_N$ decreases because the parallel isosurface com-

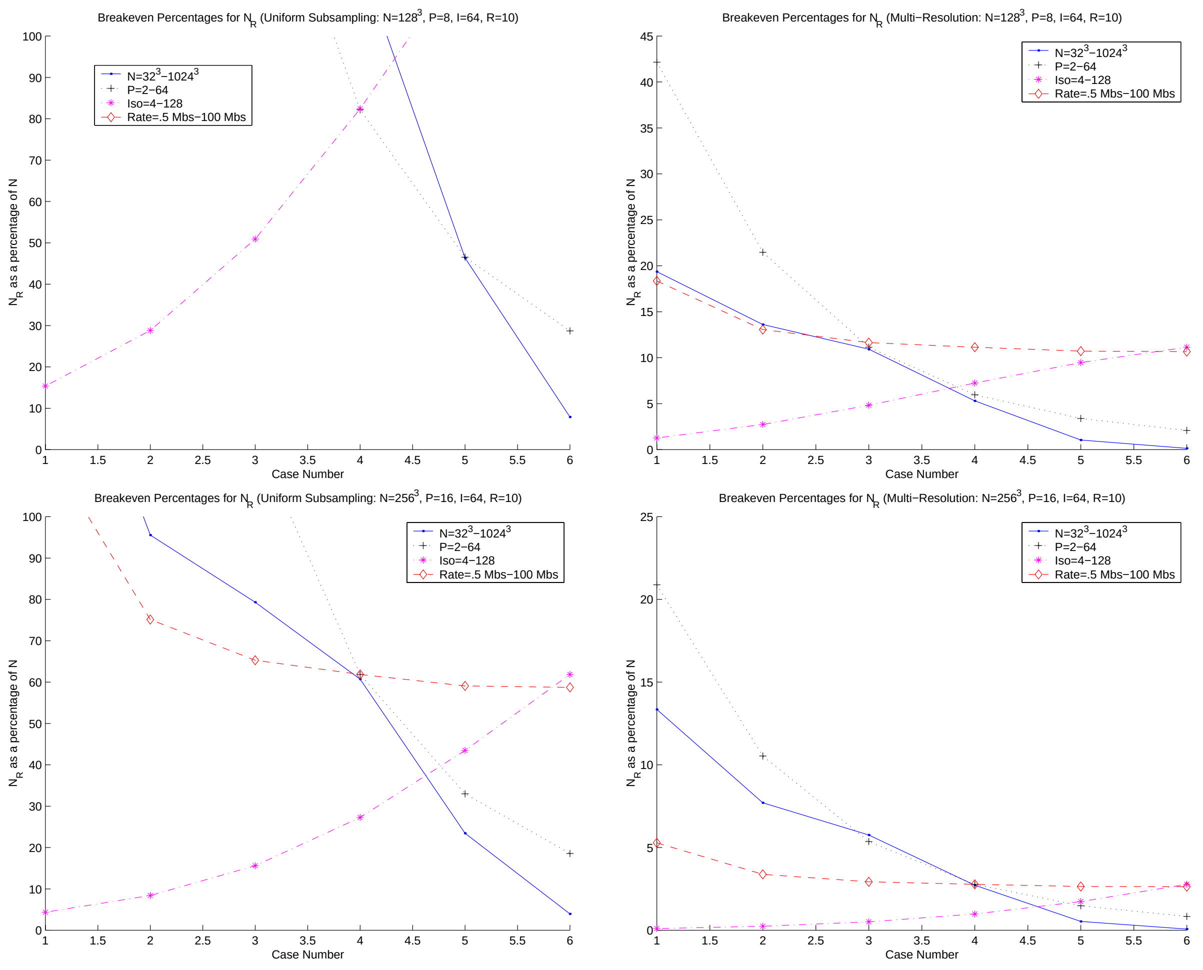

Figure 1: The breakeven graphs for uniform and multiresolution subsampling; the base parameters for the first row are are $N = 128^3$, $P = 8$, $R = 10$ Mbs, $I = 64$; the base parameters for the second row are $N = 256^3$, $P = 16$, $R = 10$ Mbs, $I = 64$

putation costs in Models 1 and 2 decrease. Network bandwidth has very little effect on $\mathcal{P}_N$, except for small values, showing that transmission costs are not the limiting factor for any of the models. Finally, as expected, as the number of isosurfaces generated, I, increases, $\mathcal{P}_N$ also increases.

4.3 Uniform and Multiresolution Subsampling

For a given set of parameters, the graphs in Figure 1 clearly show that uniform grid resampling is significantly more cost effective than multiresolution subsampling. To determine relative performance benefits, we plot the ratios of the uniform and multiresolution subsampling breakeven percentages in Figure 2. We see that the uniform grid is about 12 and 22 times better than the multiresolution grid for the smaller and larger problem sizes, respectively. We note that the primary difference between the two methods is the large difference in C_{I_u} and C_{I_m}. In fact, if we consider a case in which C_{I_m} is $5.0 \cdot 10^{-6}$ or about four times slower than C_{I_u}, the corresponding performance ratios are 4 and 17, respectively.

Thus, the multiresolution technique will outperform the uniform grid method only if the subsampling approximation errors are reduced by the same amount or more using significantly fewer grid points. To explore these tradeoffs, we subsample three different application data sets of varying size, dimensionality, and mesh type. The first two data sets are from Rayleigh-Taylor (R-T) simulations in both two and three dimensions and contain $N = 5.3 \cdot 10^4$ and $N = 3.7 \cdot 10^5$ data points, respectively. These data sets are characterized by a contact discontinuity between two fluids of different density and represent a broad class of applications whose primary features are sharp discontinuities which are typically local, lower-dimensional phenomena. The third data set is from a three-dimensional simulation of hairpin vortices developing in flow around a hemisphere and contains $N = 2.05 \cdot 10^6$ data points. This problem is representative of a class of applications in which the scalar field of interest describes fully three-dimensional features.

For each problem, we create reduced data sets of approximately 5, 10, 25, and 50 percent in the uniform grid case and approximately 1.5, 3, 6, 12, and 25 percent in the multiresolution case. For multiresolution subsampling, N_R was specified, and the

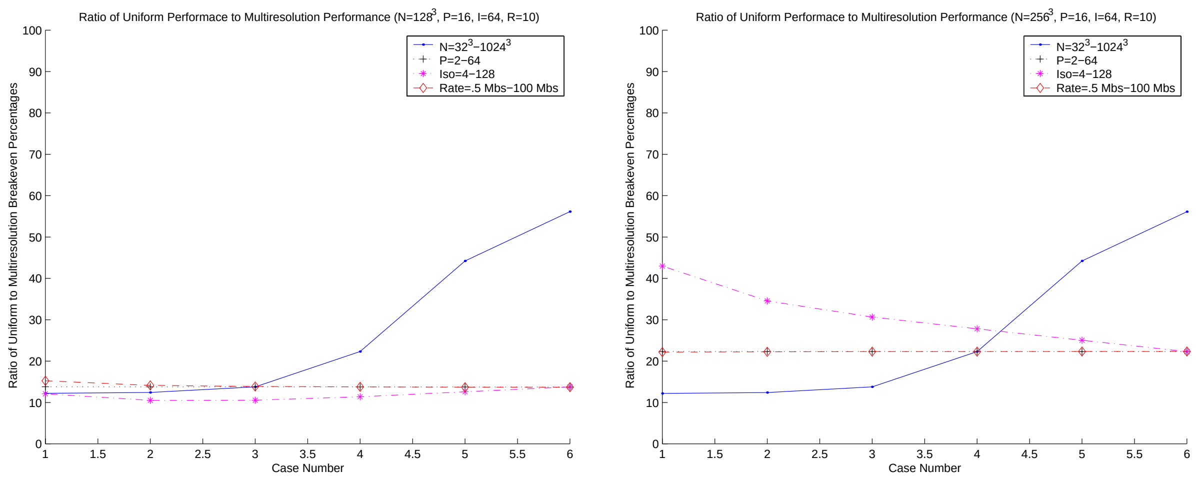

Figure 2: The relative outperformance of the uniform grid subsampling compared to multiresolution subsampling for a fixed value of N_R and the base parameters used in Figure 1.

code automatically determined the best decomposition to minimize one of the two different error criteria defined in Section 2: the standard deviation, σ_n, in the leaf octants, and the maximum deviation, e_n, in the leaf octants. In Table 5, we report the the average σ_n and maximum e_n over all octants for each case. The first value gives a measure of the overall fidelity of the reduced data set to the original data set; the latter value gives a worst-case measure of fidelity.

In all cases, we achieve the same average and maximum errors using far fewer multiresolution grid points than uniform grid points. The number of cells required to achieve similar results depends on both the insertion criterion used and the error values used to make the comparison. For example, for the 2D Rayleigh-Taylor problem, if we use average standard deviation as our insertion criterion, the same average error is achieved by approximately a factor of five fewer grid points, but the maximum error is approximately doubled. Similarly, if we use an insertion criterion based on the maximum deviation, the same maximum error can be achieved in approximately a factor of 30 fewer grid points, but the average standard deviation is increased by about a factor of six. For the 3D Rayleigh-Taylor problem, the corresponding results are decreases by a factor of two and ten for the average and maximum error insertion criteria, with corresponding increases of about 10 percent and double for the error measure not targeted. For the hairpin vortex data set, the scalar field is interesting in that only values less than negative one are of interest; all other values are disregarded as noise. These noise values can vary dramatically outside the regime of interest, rendering the maximum deviation error measure ineffective. By comparing the results achieved with the average error measure, we find that we can obtain the same errors with roughly a factor of ten fewer grid points.

In the left image in Figure 3, we show the two-dimensional Rayleigh-Taylor data set subsampled using a uniform grid containing 15616 data points. On the right, we show the same data set subsampled with the multiresolution technique using 4102 data points. The average errors are .0347 and .0339, respectively. In Figure 4, we show isosurfaces from the hairpin vortex set for vorticity indicator of -.28. The left figure shows uniform grid subsampling using 34798 grid points which results in an average error of 3.18. The right figure shows multiresolution subsampling with 34725 grid points which results in an average error of 1.46.

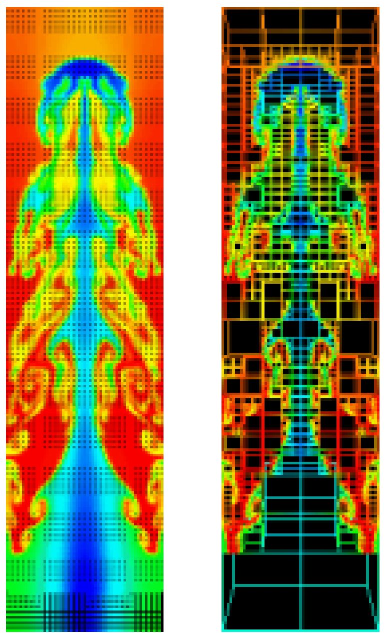

Figure 3: The left figure shows the results of uniform subsampling using 15616 grid points for the two-dimensional Rayleigh-Taylor; the right figure shows the results of multiresolution subsampling using 4102 grid points

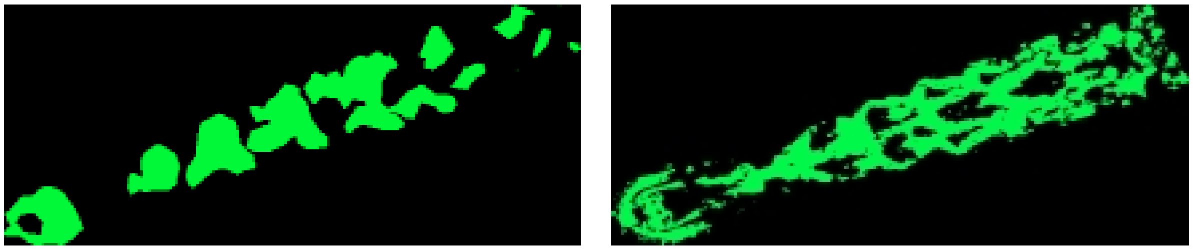

Figure 4: The left figure shows the results of uniform subsampling using 34798 grid points for the hairpin vortex applications; the right figure shows the results of multiresolution subsampling for 34725 grid points

5 Conclusions

Much useful information can be obtained by studying performance cost estimates for a variety of techniques that accomplish the same goal. From our analysis of the computation and communication costs associated with three different strategies for remote, interactive exploration of large data sets, we find that each has regimes for which is it is the most cost effective approach. Our results show that using a full-resolution view resampled to a uniform grid is often the most cost effective approach. However, many computations are performed on nonuniform, adaptive grids which implies that a uniform grid resampling will have more error in regions containing a large number of grid points which are typically the areas containing features of interest. Multiresolution approaches address this difficulty; large numbers of resampled points may be concentrated in the same regions as the original computation. In addition, our tests showed that they can also be the most cost effective approach when the metric used is the cost to achieve a given error level.

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

References

[1] Ibrahima Ba, Christopher Malon, and Barry Smith. Design of the ALICE Memory Snooper, http://www.mcs.anl.gov/ams, 1999.

[2] Chun-Fa Chang, Gary Bishop, and Anselmo Lastra. LDI tree: A hierarchical representation for image-based rendering. In *Proceedings of SIGGRAPH 99*, pages 291–298, Los Angeles, California, August 1999.

[3] T. W. Crockett and T. Orloff. A MIMD rendering algorithm for distributed memory architectures. In *Proceedings of the Parallel Rendering Symposium*, pages 35–42, 1993.

[4] Thomas Crockett. Beyond the renderer: Software architecture for parallel graphics and visualization. Technical Report ICASE Report No. 96-75, Institute for Computer Applications in Science and Engineering, 1996.

[5] Robert Eckstein, Marc Loy, and Dave Wood. *JAVA Swing*. O'Reilly and Associates, Sebastopol, California, 1998.

[6] Lori Freitag and Raymond Loy. Adaptive multiresolution visualization of large data sets using parallel octrees. In *Proceedings of SC99*. ACM SIGARCH and IEEE Computer Society, 1999.

[7] Steven Gortler, Li wei He, and Michael Cohen. Rendering layered depth images. Technical Report 97-09, Microsoft Research, March 1997.

[8] Robert Haimes. pV3: A distributed system for large-scale unsteady CFD visualization. *AIAA paper*, 94-0321, January 1994.

[9] C. Hanson and P. Hinker. Massively parallel isosurface extraction. In *Proceedings of Visualization 92*. IEEE Computer Society, 1992.

[10] Bjoern Heckel, Gunther Weber, Bernd Hamann, and Kenneth Joy. Construction of vector field hierarchies. In *Proceedings of IEEE Visualization 99*, pages 19–26, October 1999.

[11] Brian Von Herzen and Alan Barr. Accurate triangulations of deformed, intersecting surfaces. In *Computer Graphics Proceedings, SIGGRAPH 87*, volume 21, pages 103–110. ACM, 1987.

[12] Hugues Hoppe. Progressive meshes. In *Computer Graphics SIGGRAPH 96 Proceedings*, pages 99–108, 1996.

[13] Hugues Hoppe. Efficient implementation of progressive meshes. Technical Report MSR-TR-98-02, Microsoft Research, Microsoft Corporation, 1998.

[14] Eric LaMar, Bernd Hamann, and Kenneth Joy. Multiresolution techniques for interactive texture-based volume rendering. In *Proceedings of IEEE Visualization 99*, pages 355–362, October 1999.

[15] Peter Lindstrom, David Koller, William Ribarsky, Larry Hodges, Nick Faust, and Gregory Turner. Real-time, continuous level of detail rendering of height fields. In *Computer Graphics Proceedings SIGGRAPH 96, Annual Conference Series*, pages 109–118. ACM, 1996.

[16] Kwan-Lui Ma. Parallel rendering of 3D AMR data on SGI/Cray T3E. In *Proceedings of the Frontiers 99 Conference*, pages 138–145, 1999.

[17] William Mark. Post-rendering 3D image warping: Visibility, reconstruction, and performance for depth-image warping. Technical Report TR99-022, University of North Carolina, April 1999.

[18] Willianm Mark. Efficient reconstruction techniques for post-rendering 3D image warping. Technical Report TR98-011, University of North Carolina, March 1998.

[19] Nelson Max. Hierarchical rendering of trees from precomputed multi-layer Z-buffers. In *Rendering Techniques '96: Proceedings of 7th Eurographics Workshop on Rendering*, pages 165–174, New York, 1996. Springer.

[20] MCI WorldCom. MCI WorldCom and NFS's very High Speed Backbone Network Service, http://www.vbns.net/, 2000.

[21] G. J. Miller, K. Thopson, , and R. Wilder. Performance measuresments on the vBNS. In *Proceedings of the Interop '98 Engineering Conference*, Las Vegas, 1998.

[22] Manuel Oliveira and Gary Bishop. Image-Based Objects. In *Proceedings of 1999 ACM Symposium of 3D Graphics*, pages 191–198, Atlanta, Georgia, April 1999.

[23] Jos Roerdink and Michel Westenberg. Wavelet-based volume visualization. Technical Report IWI 98-9-06, Institute for Mathematics and Computing Science, University of Groningen, 1998.

[24] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit, An Object-Oriented Approach to 3D Graphics*. Prentice Hall PTR, Upper Saddle River, New Jersey, 1998.

[25] Alexandru Telea and Jarke van Wijk. Simplified representation of vector fields. In *Proceedings of IEEE Visualization 99*, pages 35–42, October 1999.

[26] Arsi Vaziri and Mark Kremenetsky. Visualization and tracking of parallel CFD simulations. In *Proceedings of HPC 95*. Society of Computer Simulation, 1995.

Table 5: Results for uniform grid and multiresolution data reduction techniques

Method	N_R	$\mathcal{P}_N$	Avg σ_n	Max e_n
2D Rayleigh Taylor				
Uniform	4096	7.5	.0652	.932
Uniform	7179	14.0	.0506	.763
Uniform	15616	28.9	.0347	.569
Uniform	23560	43.6	.0230	.555
Criteria = Avg. σ_n				
Multi-Res	808	1.5	.114	1.24
Multi-Res	1630	3.0	.0703	1.24
Multi-Res	3268	6.0	.0397	1.08
Multi-Res	6550	12.1	.0238	1.05
Multi-Res	13108	24.2	.0083	.515
Criteria = Max e_n				
Multi-Res	808	1.5	.131	.538
Multi-Res	1630	3.0	.108	.457
Multi-Res	3268	6.0	.0817	.362
Multi-Res	6550	12.1	.0405	.241
Multi-Res	13108	24.2	.0122	.116
3D Rayleigh Taylor				
Uniform	24344	6.5	.0262	.448
Uniform	43750	11.8	.0225	.311
Uniform	95152	25.7	.0159	.245
Uniform	156309	42.4	.0103	.152
Criteria = Avg. σ_n				
Multi-Res	5835	1.6	.0482	.460
Multi-Res	11669	3.1	.0391	.392
Multi-Res	23345	6.3	.0286	.271
Multi-Res	46697	12.6	.0192	.271
Multi-Res	93397	25.2	.0100	.165
Criteria = Max e_n				
Multi-Res	5836	1.6	.0530	.265
Multi-Res	11672	3.1	.0456	.210
Multi-Res	23348	6.3	.0354	.146
Multi-Res	46697	12.6	.0260	.103
Multi-Res	93396	25.2	.0148	.059
3D Hairpin Vortices				
Uniform	202799	9.9	1.08	–
Uniform	474443	23.3	.701	–
Uniform	800602	39.1	.699	–
Uniform	1212327	59.2	.268	–
Criteria = Avg. σ_n				
Multi-Res	62642	3.1	.568	–
Multi-Res	124667	6.1	.343	–
Multi-Res	250790	12.2	.176	–
Multi-Res	502772	24.5	.080	–