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

Lori A. Freitag
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439
freitag@mcs.anl.gov

Raymond M. Loy
ASCI Flash Center
University of Chicago
Chicago, IL 60637
rloy@flash.uchicago.edu

Abstract

We compare three remote visualization strategies used for interactive exploration of large data sets in distributed environments: 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 regimes for which each approach is most cost-effective, we develop performance models to study the computation and communication costs associated with the common visualization task of isosurface generation. Using these models, we analyze a number of scenarios in which the hardware system configurations and task complexity change and offer general recommendations for the use of each strategy. For one particular strategy, subsampling, we further investigate the tradeoffs between multiresolution and uniform grid methods in terms of performance and approximation errors.

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 combine remote, high-end resources with high-speed networks to provide interactive navigation and exploration of very large data sets.

In this paper, we consider three common strategies for improving performance of interactive data exploration using remote resources: image-based rendering, parallel visualization servers, and subsampling of the original data set. We give an overview of these strategies in Section 2. Each approach attempts to improve performance by targetting a different stage of the visualization process, and each has its advantages and disavantages. For this analysis we consider each method individually, but note that they are not inherently exclusive and combinations are possible.

To determine the most cost-effective strategy for particular hardware 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 two 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.

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. (Simply sending each image as the object is manipulated requires more network bandwidth than is generally available.) In this paper, we focus on 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 e.g., [1, 14]). 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 communication and local computation 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 [1].

Parallel visualization servers utilize remote computational resources to visualize full-resolution data sets either as the computation proceeds (e.g., [5, 18]) or as a post-processing step (e.g., [2, 6]). 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 size of 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 remote and local, 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 [10, 9], progressive meshes [8], wavelets [15], or other clustering approaches [7, 17]. 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.

The primary advantage of subsampling or clustering approaches is that they are useful for fast, local exploration of the reduced data set. The remote computation and transmission costs are fixed regardless of the number of visualization tasks. The primary disadvantage is the difficulty in maintaining fidelity to the original data set; 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 windowing approach. Furthermore, forming derived visualization entities on the interpolated data set may be difficult in the multiresolution approach; for example, isosurfaces may contain undesired "cracks."

3. Performance Models

For each remote data exploration strategy, 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 multiresolution 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. We note that the performance of the parallel visualization server strategy can be dramatically affected by the choice of visualization techniques studied. For example, the cost of transmitting streamline information is very small and would likely improve the relative performance of the parallel visualization server compared to the other two strategies.

The costs 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. 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.

We note that our models serve to highlight the basic performance of each remote visualization strategy and some simplifying assumptions have been made. First, our models do not address the costs associated with loading and processing the resulting data sets on the local graphics workstation. Second, in the case of parallel visualization servers, the number of primitives contained in the isosurface may be reduced through various decimation proceedures (see, for example, [3]); we do not include these strategies in our models. Finally, our models do not address the general question of data compression during the communication phase between the remote resources and the local desktop graphics computer.

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 number of elements in the data set, N, and the number of isosurfaces to be computed and rendered, I, define the complexity of our visualization task. The parameter X defines the image size in pixels for image-based rendering, and N_R defines the number of elements in the reduced data sets for subsampling. 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.

Table 1. Cost model variables

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 [14]. 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 [11] 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.[1] 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 needed 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 that in Equation 1. The data transmitted for each isosurface are three spatial coordinates for each data point (3 32 bit floats) as well as the connectivity information for each triangle (3 32 bit integers) . Using the fact that the numbers of vertices and elements in a triangular mesh are nearly equal and that the average isosurface will contain $N^{\frac{2}{3}}$ triangles, the total cost of the parallel visualization server is

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

3.3. Model 3: Uniform Subsampling

For uniform grid subsampling, we assume that the original data is partitioned such that the reduction operations

[1] 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.

may be performed with minimal communication and will scale linearly as a function of 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 (9 floats), and the scalar data and error associated with each element ($2 \cdot N_R$ floats). No additional spatial or coordinate information is required.

The total cost of uniform subsampling is

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

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

There are many approaches for multiresolution subsampling, and we develop our cost model by examining one particular approach that uses parallel octrees. In this method, points from the original data set are inserted into the appropriate leaf octants. These are then evaluated according to a specified criterion and refined if necessary with associated data points reassigned to the new leaf octants. When all data has been inserted, an average data value is computed for each leaf, and this constitutes the reduced data set.

To provide an indication of the error associated with the reduced data set, we compute normalized standard deviation and maximum deviation from the mean for each leaf octant. These values can 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 user may specify the insertion criterion as an error bound, or alternatively, the error bound may be determined automatically given a performance constraint such as maximum target number of leaves. The user may interactively change the leaf criterion and thereby the resolution of the reduced data. A criterion may be applied either globally or in a specified region of interest to "zoom in" without sacrificing graphics performance. Additional details about the parallel octree algorithms, our software architecture, and results obtained on large data sets can be found in [4].

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 $3 \cdot 4N_R$ floats for the spatial coordinates plus $8N_R$ integers for the connectivity data and $2N_R$ floats for the scalar and error data. With these assumptions, our cost model for multiresolution subsampling is

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

The primary difference between Models 3 and 4 is the amount of data that must be transfered for the reduced data set, and 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 each of the models developed in Section 3 are most cost-effective, we first determine typical values for the cost 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. Finally, we consider the uniform grid and multiresolution subsampling techniques in more detail and examine their performance as a function of the subsampling error for two 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 [13]. Latency measurements listed on the vBNS net traffic web page [12] 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 use 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 [16]. 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}	$\frac{C_{I_m}}{C_{I_u}}$
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 empty cells 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 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 this factor would be less for routines specific to octree data types and is also implementation-dependent.

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

N	N_R				Add. Cost
	1	5^3	10^3	20^3	
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	$N_R = 1$	$N_R = 512$	N_R	Time	Add. Cost
50^3	3.76	4.61	32768	6.38	$4.34 \cdot 10^{-5} N_R$
75^3	12.67	15.55	23696	18.85	$11.0 \cdot 10^{-5} N_R$
100^3	31.74	41.05	18404	43.60	$26.2 \cdot 10^{-5} N_R$

To determine the uniform subsampling cost, C_{R_u}, we subsampled a tetrahedral mesh onto a uniform grid of different sizes and monitored the time required to insert the data and compute the average value and normalized standard deviation for each subsampled grid point. We also include the time required to compute 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 the multiresolution subsampling cost, C_{R_M}, we performed a similar analysis using the octree data reduction technique described in Section 3.4. 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 server models, Models 1 and 2, respectively. The computational costs for isosurface generation are identical; the models are differentiated only by the amount of data transmitted. By equating the two models, we derive the breakeven function that relates the number of pixels in the image, X, and the data set size, N:

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

Thus, for pixel sizes less than $\frac{2}{3} N^{\frac{2}{3}}$ image-based rendering will be faster than the parallel visualization server. It is interesting to note that this expression is independent of both the number of isosurfaces and the network bandwidth and latency.

Conclusion 1: *If the resolution of the images used in image-based rendering is fixed, this strategy becomes increasingly attractive compared to parallel visualization servers as the data set size increases.*

We analyze the performance of the subsampling techniques by finding the value of the reduced data set size, N_R, reported as a percentage of N, $\mathcal{P}_N = \frac{N_R}{N} \cdot 100$, such that the costs of the subsampling model are equal to the smaller of either Model 1 or 2 costs. For both uniform and multiresolution grid subsampling, we start with a representative set of base parameters $N = 128^3$, $P = 8$, $R = 10$ and $I = 64$.

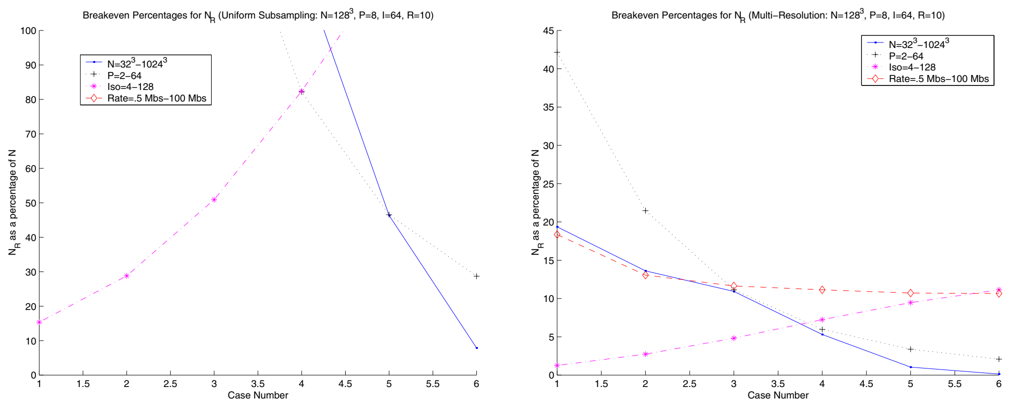

Figure 1. The breakeven graphs for uniform and multiresolution subsampling with base parameters $N = 128^3$, $P = 8$, $R = 10$ Mbs, $I = 64$.

To examine the effect of these parameters on $\mathcal{P}_N$, we vary each one individually while holding the others fixed, and plot the results in Figure 1. Values of $\mathcal{P}_N$ 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 $X = 512^2$.

As expected, uniform grid subsampling can use a higher percentage of grid points than can be used with multiresolution grid subsampling. In fact, for this base problem size it is often the most cost-effective strategy for remote visualization. In contrast, the multiresolution subsampling is cost-effective when $\mathcal{P}_N < 12$. However, the multiresolution approach can cluster points in regions of interest so that fewer grid points are needed to achieve the same fidelity to the original data set. These tradeoffs are examined in more detail in Section 4.3. In general, the following trends are evident:

Conclusion 2: *The network speed is not the primary bottleneck in any of these strategies as varying the value of the parameter R has a minimal effect on the results.*

Conclusion 3: *Subsampling approaches are best used for moderate problems sizes and for problems with a large number of visualization tasks. As the problem size increases, the size of reduced data set that gives good performance relative to the other two strategies decreases causing larger approximation errors. We note that this effect can be somewhat mitigated by the use of a windowing functionality that allows the user to obtain higher resolution only in regions of interest.*

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 uniform subsampling is about a factor of 12 more cost-effective than the multiresolution subsampling We note that the primary difference between the two methods is the large difference in the cost of generating isosurfaces, 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 ratio is 4.

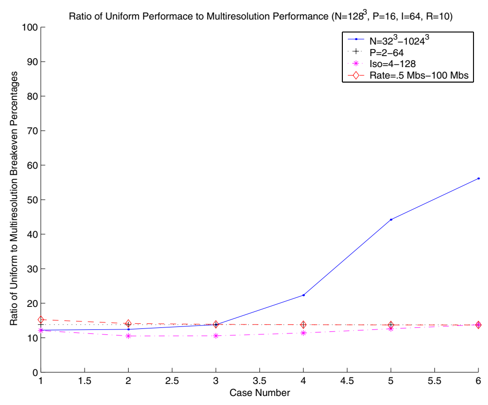

Figure 2. Relative performance of the uniform grid subsampling compared to multiresolution subsampling.

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 two different application data sets. The first data set is from a two-dimensional Rayleigh-Taylor (R-T) simulation and contains $N = 5.3 \cdot 10^4$ data points. This

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
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	–

data set is characterized by a contact discontinuity between two fluids of differing density and represents a broad class of applications whose primary features are sharp discontinuities which are typically local, lower-dimensional phenomena. The second 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 code automatically determined the best decomposition to minimize the standard deviation, σ_n. In Table 5, we report the the average σ_n and maximum e_n over all octants. 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. 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.

In both cases, we achieve the same average errors using far fewer multiresolution grid points than uniform grid points. For the R-T problem, the same average error is achieved by approximately a factor of five fewer grid points; for the hairpin vortex data set, the factor is ten.

In the left image in Figure 3, we show the R-T data set subsampled using a uniform grid containing 15616 points. On the right, we show the same data set subsampled with the multiresolution technique using 4102 points; points are clustered at the discontinuity. 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 points which results in an average error of 3.18. The right figure shows multiresolution subsampling with 3472 grid points which results in an average error of 1.46.

Conclusion 4: *For the cost parameters C_{I_u} and C_{I_m} considered here, uniform grid subsampling outperforms multiresolution subsampling when the average error criteria is used. As techniques for generating multiresolution isosurfaces improve in speed, multiresolution subsampling approaches will become increasingly cost-effective.*

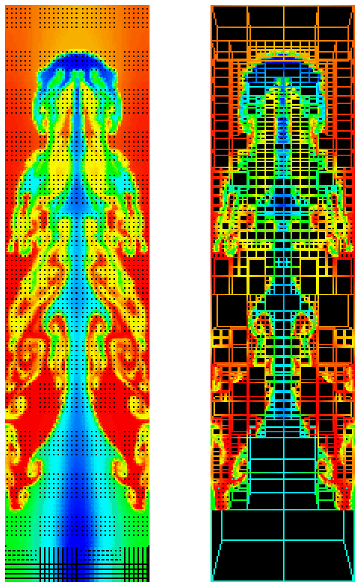

Figure 3. The R-T problem subsampled by uniform (left, 15616 points) and multiresolution (right, 4102 points)

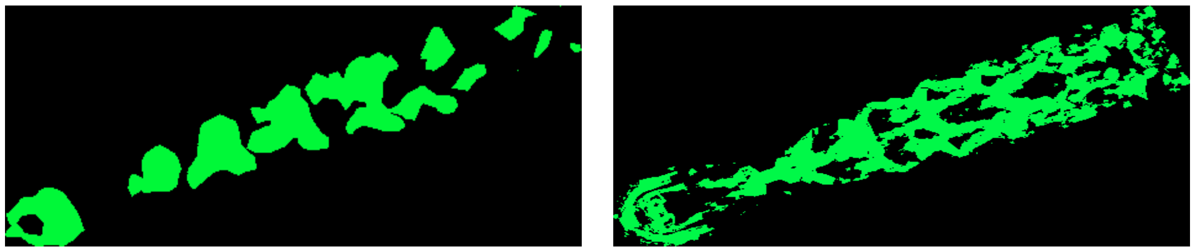

Figure 4. The hairpin vortex data set subsampled by uniform (left, 34798 points) and multiresolution (right, 34725 points) methods.

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. In general, the subsampling approaches will be the most cost-effective for moderate problems sizes containing tens of millions of elements, followed by parallel visualization servers and image-based rendering as the problem size increases.

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, and by the ASCI/Alliances Center for Astrophysical Thermonuclear Flashes at the University of Chicago under DOE subcontract B341495.

References

[1] C.-F. Chang, G. Bishop, and A. Lastra. LDI tree: A hierarchical representation for image-based rendering. In *Proc. SIGGRAPH 99*, pages 291–298, Los Angeles, California, August 1999.

[2] T. Crockett. Beyond the renderer: Software architecture for parallel graphics and visualization. Technical Report 96-75, ICASE, 1996.

[3] K. Engel, R. Westermann, and T. Ertl. Isosurface extraction techniques for web-based volume visualization. In *Proc. IEEE Visualization 99*, pages 139–146, October 1999.

[4] L. Freitag and R. Loy. Adaptive multiresolution visualization of large data sets using parallel octrees. In *Proc. SC99*. ACM SIGARCH and IEEE Computer Society, 1999.

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

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

[7] B. Heckel, G. Weber, B. Hamann, and K. Joy. Construction of vector field hierarchies. In *Proc. IEEE Visualization 99*, pages 19–26, October 1999.

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

[9] E. LaMar, B. Hamann, and K. Joy. Multiresolution techniques for interactive texture-based volume rendering. In *Proc. IEEE Visualization 99*, pages 355–362, October 1999.

[10] P. Lindstrom, D. Koller, W. Ribarsky, L. Hodges, N. Faust, and G. 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.

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

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

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

[14] M. Oliveira and G. Bishop. Image-Based Objects. In *Proc. 1999 ACM Symposium of 3D Graphics*, pages 191–198, Atlanta, Georgia, April 1999.

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

[16] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit, An Object-Oriented Approach to 3D Graphics*. Prentice Hall PTR, Upper Saddle River, New Jersey, 1998.

[17] A. Telea and J. van Wijk. Simplified representation of vector fields. In *Proc. IEEE Visualization 99*, pages 35–42, October 1999.

[18] A. Vaziri and M. Kremenetsky. Visualization and tracking of parallel CFD simulations. In *Proc. HPC 95*. Society of Computer Simulation, 1995.