Network Requirements for 3D Flying in a Zoomable Brain Database

M. Claypool, J. Riedl, J. Carlis, G. Wilcox, R. Elde, E. Retzel, A. Georgopoulos, J. Pardo, K. Ugurbil, B. Miller, C. Honda

University of Minnesota
Medical School and Computer Science Department

In laboratories around the world, neuroscientists from diverse disciplines are exploring various aspects of brain structure. Because of the size of the domain, neuroscientists must specialize, making it difficult to fit results together and causing some research efforts to be duplicated because of lack of sharing of information. We have begun a long-term project to build a neuroscience research database for brain structure. One aspect of the database is the ability to visualize high-quality, high-resolution micro-graphs montaged together into three-dimensional structures as they were in the living brain. As demonstrated in this paper's analysis, realistic presentation of these visualizations across computer networks will stress current and proposed gigabit networks. Image compression can reduce network loads, but wide-spread use of the visualizations will still require networks capable of sustaining terabits/second of throughput.

I. Introduction

Gradually, in laboratories around the world, neuroscientists from diverse disciplines are exploring various aspects of brain structure. Since there is so much research to be done on the nervous system, neuroscientists must specialize. An undesirable result of this specialization is that it is difficult for individual neuroscientist to investigate how their results fit together with results from other scientists. Moreover, they sometimes duplicate research efforts since there is no easy way to share information.

To enhance the work of neuroscientists, we propose a zoomable database of images of the brain tissue. We begin with the acquisition of 3D structural maps of the nervous system using high-field, high-resolution magnetic resonance imaging (MRI). The MR images show the entire brain in a single dataset and preserve spatial relationships between structures within the brain. However, even high resolution MRI cannot show individual cells within the brain. We therefore anchor confocal microscope images to these 3D brain maps. Because of their higher resolution, the confocal images are of smaller regions of the brain. Many such images are montaged into larger images by aligning cellular landmarks between images.[1] These montages are then aligned with structural landmarks from the MR images, so the high-resolution images can be anchored accurately within the MR image. In addition to the image data, other types of brain data can be linked to the 3D structure.

[1] We have created a WWW server that includes an example of a montage of confocal micro-graphs at http://ftp.cs.umn.edu/neural/doc/proposal.html.

This paper focuses on the system requirements for the interface to this database. Section II of this paper describes related work. Section III introduces the user requirements for the database interface. Section IV predicts the network requirements based on the user requirements. Section V details predicted network benefits from compression. Section VI describes the disk requirements. Section VII presents the processor requirements. Section VIII explores quality considerations. And section IX summarizes possible topology considerations and lists future work.

II. Related Work

A. Scientific Visualization

Hibbard, Paul, Santek, Dyer, Battaiola and Voidrot-Martinez design an interactive, scientific visualization application [1]. They seek to bridge the barrier between scientists and their computations and allow them to experiment with their algorithms.

Elvins describes five foundation algorithms for volume visualization in an intuitive, straight-forward manner [2]. His paper includes references to many other papers that give the algorithm details.

Singh, Gupta and Levoy show that shared-address-space multiprocessors are effective vehicles for speeding up visualization and image synthesis algorithms [3]. Their article demonstrates excellent parallel speedups on some well-known sequential algorithms.

We propose an application that will use techniques developed in other scientific visualization applications. In particular, we use performance results from Singh, et al. and may implement some of the algorithms that Elvins describes.

B. Neuroscience

Carlis, et al, present the database design for the Zoomable Brain Database [4]. We have developed data models for novel neuroscience. One model focuses on MR and Confocal microscopy images, the connections between them, and the notions of macroscopic and microscopic neural pathways in the brain. We have implemented this "connections" data model in the Montage DBMS, and have populated the schema with neuroscience data.

Kandel, Schwartz and Jessel discuss in detail the fundamentals behind neural science [5].

Slotnick and Leonard have an extensive photo atlas of an albino mouse forebrain [6]. The ideas of a zoomable, digitized rat brain came from atlases such as this one.

The Zoomable Brain Database is based on neuroscience fundamentals. At its heart is a digital brain atlas. Our work presents network, processor and disk performance analysis in accessing the digital images.

C. Compression

Wallace presents the Joint Photographic Experts Group (JPEG) still picture standard [7]. He describes encoding and decoding algorithms and discusses some relations to the Motion Picture Experts Group (MPEG).

Patel, Smith and Rowe design and implement a software decoder for MPEG video bitstreams [8]. They give performance comparisons for several different bit rates and platforms. They claim that memory bandwidth is the bottleneck for the decoder, not processor speed. They also give a cost analysis of three different machines for doing MPEG.

We expand compression research by providing careful CPU load measurements of JPEG compression and decompression on a Sun IPX. In addition, we predict the effects of compression on network, processor and disk performance.

D. Networks

Hansen and Tenbrink investigate gigabyte networks and their application to scientific visualization [9]. They explore various system topologies centered around the HIPPI switch. They analyze network load under some possible visualization applications and hypothesize on the effects of compression.

Lin, Hsieh, Du, Thomas and MacDonald study the performance characteristics of several types of workstations running on a local Asynchronous Transfer Mode (ATM) network [10]. They measure the throughput of four different application programming interfaces (API). They find the native API achieves the highest throughput, while TCP/IP delivers considerably less.

Frankowski and Riedl study the effects of delay jitter on the delivery of audio data [11]. They develop a heuristic for managing the audio playout buffer and compare it to several alternative heuristics. They find none of the heuristics is superior under all possible arrival distributions.

E. Processors

Claypool and Riedl study audioconference processor load [12]. We focus on the effects of silence deletion. We find silence deletion improves audioconference CPU load more than most alternatives, including 10 times faster processors and multicasting.

Dongarra compares the performance of different computer systems in solving dense systems of linear equations [13]. He compares the performance of approximately 100 computers, from a CRAY Y-MP to scientific workstations such as Apollos and Suns to IBM PC's.

Our work resents an alternative method for measuring CPU performance. We expand upon our model measurements of the effects of compression on CPU performance.

F. Disks

Ruwart and O'Keefe describe a hierarchical disk array configuration that is capable of sustaining 100 MBytes/second transfer rates [14]. They show that virtual memory management and striping granularity play key roles in enhancing performance.

III. User Level Requirements

The brain database will be embedded in three dimensions. The user starts a typical investigation by navigating through the MR images in a coarse 3D model of the brain to a site of interest. The user then zooms to higher resolution confocal images embedded in the MRI landscape. This real-time navigating and zooming we call "flying."

The scientific value of the data and the distributed nature of the database impose a series of user-level requirements that the flying interface should satisfy:

- *Wide Spread Use.* We estimate the number of users who may be interested in using the database to be 10,000 (half the number of members of the Society for Neuroscience in 1994), the average day to be 10 work hours, the average work week to be 5 days and the average work month to be 4 weeks. We can predict the average number of simultaneous database users based on some possible usage amounts:

Uses the Database	Simultaneous Database Users
1 hour/day	1000
1 hour/week	200
1 hour/month	50

- *Twenty-four Bit Color.* As often as possible, the database will express experimental data in its purest form, so expensive raw data can be analyzed and reanalyzed by researchers worldwide. Images include up to three layers of eight bit gray-scale, with each layer representing one brain chemical. The final images include 24 bits of color, all important to the scientists who view it.
- *Three Dimensions.* Since the anchoring MR images are three dimensional, flying must be allowed in all three dimensions. This necessitates computing two dimensional frames from the three dimensional images. The computation can take place by two different methods:
  - *Remote Flying:* In remote image processing the server does the image computation and transfers only a 2D frame to the client. We estimate Remote Flying will require sending 24 Mbits (a mega-pixel) of data per frame.
  - *Local Flying:* In local image processing the server transfers the 3D data and the client does the image computation. We estimate Local Flying will require sending 384 Mbits (the 3D region of the brain) of data per frame.
- *Smooth Navigation.* Flying must be very smooth even over a varied, non-dedicated network. This necessitates a motion picture quality rate of 30 frames per second, and jitter control to compensate for network

variance.

- *Adaptability.* Flying must adapt to a wide variety of resources, including varying CPU and disk types and non-dedicated variable bandwidth networks.

IV. Network Requirements

Since the database is distributed, the above user-level requirements determine the network requirements. For one user, we can predict the network load that flying will induce under each type of image processing:

Flying Type	Mbits/second
Remote	720
Local	11520

We can determine the minimal network required for flying using projected network bandwidths. The following table lists the Optical Carrier (OC) network rates:

Protocol	Mbits/second
OC-1	52
OC-3	155
OC-12	622
OC-24	1243

We infer from the above two tables that we need more bandwidth than even an OC-12 network can deliver to support just one remote flying user. Since local flying appears to be very difficult under all projected network bandwidths, we will assume remote flying in the rest of this paper.

Network bandwidth will increase further with additional simultaneous users. Figure 1 shows the load predictions for a variable number of users.

To counteract the huge bandwidth requirements, we can increase the number of servers. If we assume that each user flies through the database for one hour per week, we have an average of 200 simultaneous users (see section III). Figure 2 shows the load predictions for a variable number of servers.

With 200 servers, there is, on average, one server per user. We take 200 to be the upper limit on the number of servers. But with 200 servers, each connected to clients by OC-12s, each server can only support one user. Clearly, there is a need to find ways to reduce network load.

V. Compression

We turn to compression to reduce network bandwidth. If each frame is compressed before sending, the network bandwidth will be reduced.

We assume JPEG compression for flying. The Joint Photographic Experts Group (JPEG) have been working to establish the first international compression standard for continuous-tone still images, both grayscale and color [7]. Although JPEG is intended as a still picture standard, it has greater flexibility for video editing, and is likely to become a "de facto" intraframe motion standard as well.

The quality factor in JPEG lets you trade off compressed image size against quality of the reconstructed image: the higher the quality setting, the closer the output image will be to the original image and the larger the JPEG file.[2] A quality of 100 will generate a quantization table of all 1's, eliminating loss in the quantization step (but there is still information loss in subsampling, as well as roundoff error).

Since the data is to be as close to the original data as possible, we assume a quality of 100 must be used for flying compression. Pilot tests show that JPEG with a quality of 100 reduces the size of PPM[3] images by about 70%. In all subsequent predictions, we assume a compression ratio of 70%.

Figure 3 shows our predictions of the effects of compression versus the number of simultaneous users. Figure 4 shows our predictions of the effects of compression versus the number of servers.

With compression and 75 servers, we can satisfy the user-level network bandwidth requirements for 200 simultaneous users.

VI. Disk Requirements

Compression may also be used for reducing the size of images on disk. The most obvious effect is the decrease in storage space. A fully-mapped mouse brain will take approximately 24 terabits of data, while a rat brain will take approximately 80 terabits of data. A 70% compression rate would reduce the required storage space to about 7.2 terabits and 24 terabits respectively.

Compressed disk images will also increase the disk flying throughput. With an image compression of 70%, disk drives can supply over 3 times more frames/second. Using the disk bandwidths obtained from [4], we can determine the disks required for meeting the requirements for flying throughput. Recent studies at the Army High Performance Computing Research Center at the University of Minnesota (AHPCRC) have measured the performance of single disk arrays. When going through a file system, normal I/O runs at about 6 Mbits/second. Direct I/O which bypasses the file system buffer cache and beams data directly to the user's application buffer, runs at 14.5 Mbits/second on a non-fragmented file. Direct I/O and striping across multiple disk arrays can achieve up to N times 14.5 Mbits/second where N is the number of arrays striped over. The AHPCRC currently stripes over 4 or 8 arrays, producing transfer rates around 40 Mbits/second for 4 arrays. In the 3-5 year time frame they expect to see the single array speed go to 100 Mbits/second and possibly 1 Gbits/second for large file transfers. If the physical design matches users' needs effectively, the database retrieval rate may approach the maximum disk rate, but applications cannot exceed this retrieval speed.

Figure 5 shows the disk drive throughputs compared to the increase in bandwidth as simultaneous users increase. At least 8 direct I/O disk arrays are required to meet the

[2] Quality values below about 25 generate 2-byte quantization tables, which are considered optional in the JPEG standard. Some commercial JPEG programs may be unable to decode the resulting file.

[3] Jef Poskanzer's PBMPLUS image software can be obtained via FTP from export.lcs.mit.edu (contrib/pbmplus*.tar.Z) or ftp.ee.lbl.gov (pbmplus*.tar.Z).

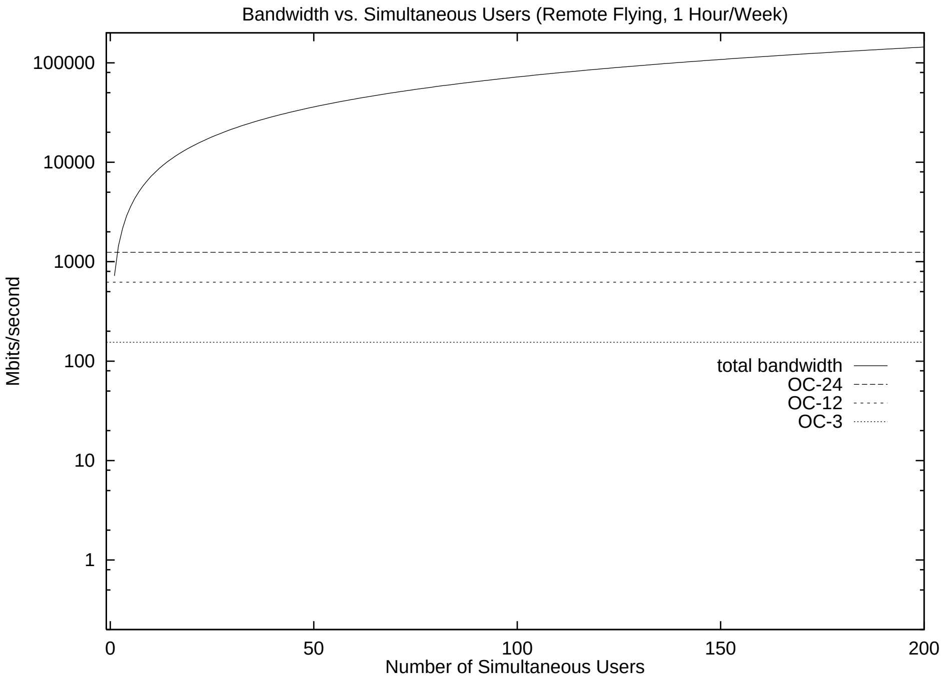

Fig. 1. **Bandwidth versus Number of Simultaneous Users. The curve is the total bandwidth required. The horizontal lines are various Optical Carrier (OC) network bandwidths. On this, and subsequent graphs, the Mbits/second axis is in log 10 scale.**

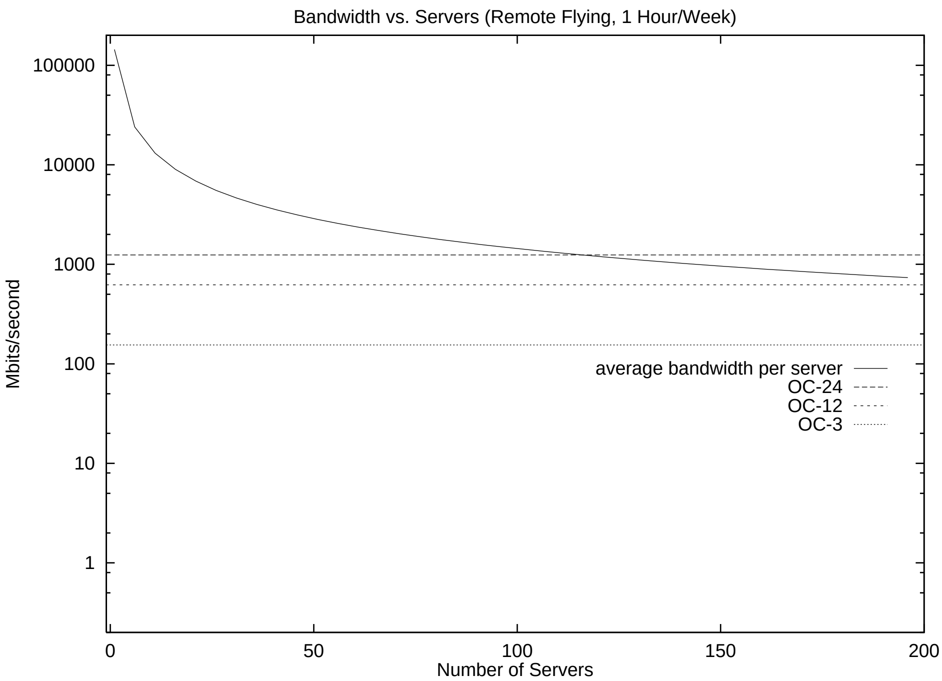

Fig. 2. **Bandwidth versus Servers. The curving is the average bandwidth per server. The horizontal lines are OC network bandwidths.**

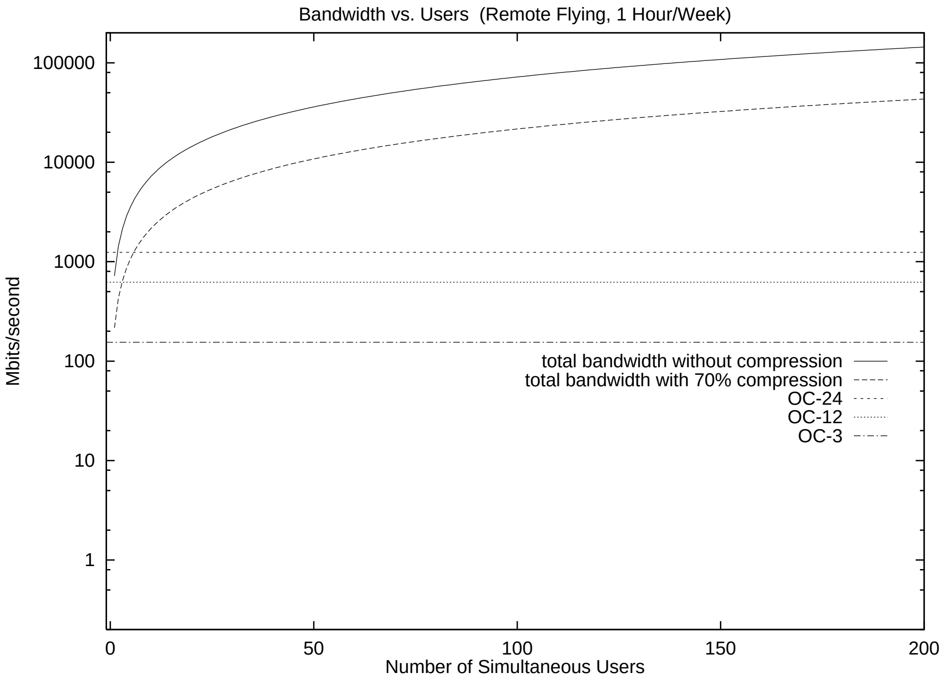

Fig. 3. Bandwidth versus Simultaneous Users. The top curve depicts the total bandwidth without compression. The lower curve depicts the total bandwidth with 70% compression. The horizontal lines are OC network bandwidths.

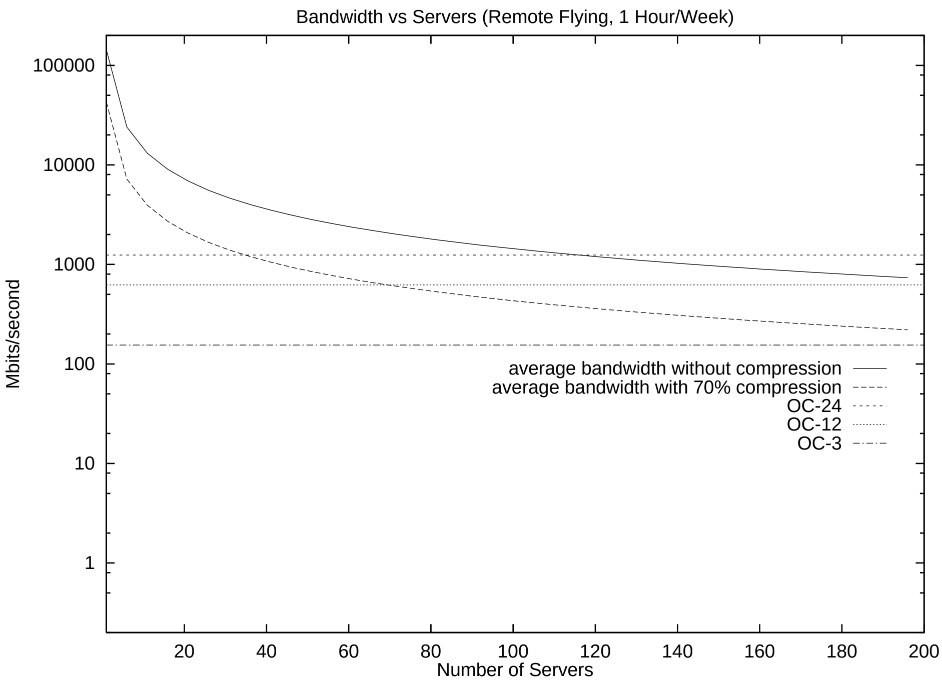

Fig. 4. Bandwidth versus Servers. This graph depicts the average bandwidth per server for a variable number of servers. The top curving line represents bandwidth without compression. The lower curving line represents bandwidth with compression. The horizontal lines are OC network bandwidths.

flying requirements for one user. However, with compression, one user's flying need can be met with 4 direct I/O disk arrays.

Figure 6 shows the disk rates compared to the decrease in bandwidth as servers increase. A 1 disk array will not satisfy even a single user's flying requirements. An 8 disk array will satisfy user flying requirements only with compression and 40 servers. Without compression, an 8 disk array will need 140 servers to satisfy user flying requirements. A 4 disk array will need compression and 100 or more servers to meet flying requirements.

A drawback of compression is that it will decrease CPU frame throughput. Compressed images on disk must be uncompressed before the CPU can perform image computation to generate the 2D frame. And compressing before sending takes additional CPU time.

VII. CPU Requirements

We can predict the CPU throughput required for flying by using the CPU load for individual components as in [12]. We obtain the flying throughput for a 40 MHz Sun IPX to obtain a baseline for predictions to faster machines.

To do this, we obtained the CPU load for doing JPEG compression and decompression. We modified the source code for *cjpeg* and *djpeg*[4] to perform compression and decompression in separate processes. We followed experimental techniques that were identical to those used to obtain the CPU loads for the previous components. As in [12], we used a counter process that incremented a double variable in a tight loop to measure the CPU load of the JPEG components. We compressed and decompressed images to and from PPM files (see the footnote in section V).

The independent variable in our measurements was the JPEG compression quality. Figure 7 depicts the CPU load for compression and decompression versus quality. All points are shown with 95% confidence intervals.

Using linear regression, we can derive the JPEG CPU load in milliseconds per bit of the original PPM image:

JPEG	msec/bit
compression	0.000338
decompression	0.000341

To estimate the effects of new high-performance graphical workstations on the CPU load for image calculation, we use the results reported in [3]. They report that image calculation from a `256x256x256` voxel volume data set (the same size we assume for our 3D region) takes about 5 seconds per frame on a 100 MHz Silicon Graphics Incorporated (SGI) Indigo 2 workstation using Levoy's ray-casting algorithm and about a second per frame using a new shear-warp algorithm. We will assume a CPU load of one second per frame for an Indigo workstation.

We compare the processing power of the Sun IPX to the SGI Indigo by comparing their performance under the Systems Performance Evaluation Cooperative (SPEC) benchmark integer suite.[5] SPEC is a non-profit corporation formed by leading computer vendors to develop a standardized set of benchmarks. Founders, including Apollo/Hewlett-Packard, DEC, MIPS and Sun, have agreed on a set of real programs and inputs that all will run. The benchmarks are meant for comparing CPU speeds. The SPEC numbers are the ratio of the time to run the benchmarks on a reference system and the system being tested. The SPECint value for a Sun IPX is 21.8 and the SPECint value for the 150 MHz Indigo 2 is 92.2 [15]. Roughly, the Indigo 2 is 4 times faster than IPX, so we assume the IPX takes 4 seconds to do the 2D image calculation from the 3D region.

We can now predict the flying throughput for the Sun IPX CPU server. There are three different methods of flying:

1. *No compression.* Flying with no compression has three CPU load components: *read* is the CPU load for reading the image from the disk; *calculate* is the CPU load for computing the 2D frame from the 3D image; and *send* is the CPU load for sending the frame to the user.
2. *Network compression.* Flying with network compression has four CPU load components: *read* is the CPU load for reading the image from the disk; *calculate* is the CPU load for computing the 2D frame from the 3D image; *compress* is the CPU load for compressing the frame; and *send* is the CPU load for sending the compressed frame to the user.
3. *Network and disk compression.* Flying with network compression and disk compression has five CPU load components: *read* is the CPU load for reading the compressed image from the disk; *decompress* is the CPU load for decompressing the image; *calculate* is the CPU load for computing the 2D frame from the 3D image; *compress* is the CPU load for compressing the frame; and *send* is the CPU load for sending the compressed frame to the user.[6]

Table I gives the Sun IPX CPU throughput predictions for the above 3 methods for the server to provide flying.

Network compression slightly reduces CPU frame throughput. Disk compression, however, reduces CPU frame throughput by more than 80%. Most importantly, it would take a CPU *over nine-hundred times faster* than a Sun IPX to satisfy the flying requirements for even one user! Even an SGI Indigo 2 with 20 processors would still only have a flying throughput of 59.2 Mbits/second, not even enough for 1 user! Clearly, there is a need to find ways to increase CPU flying throughput.

One such method may be specialized hardware. Currently, there are co-processors that perform JPEG compression and decompression. Similarly, many computers have specialized graphics rendering hardware. To estimate

[4] The "official" archive site for this software is ftp.uu.net (Internet address 137.39.1.9 or 192.48.96.9). The most recent released version can always be found there in directory graphics/jpeg. The particular version we used is archived as jpegsrc.v4.tar.Z.

[5] The SPEC primer is frequently posted to comp.benchmarks. SPEC questions can also be sent to `spec-ncga@cup.portal.com`.

[6] A clever image computation algorithm might do the calculation on the compressed images.

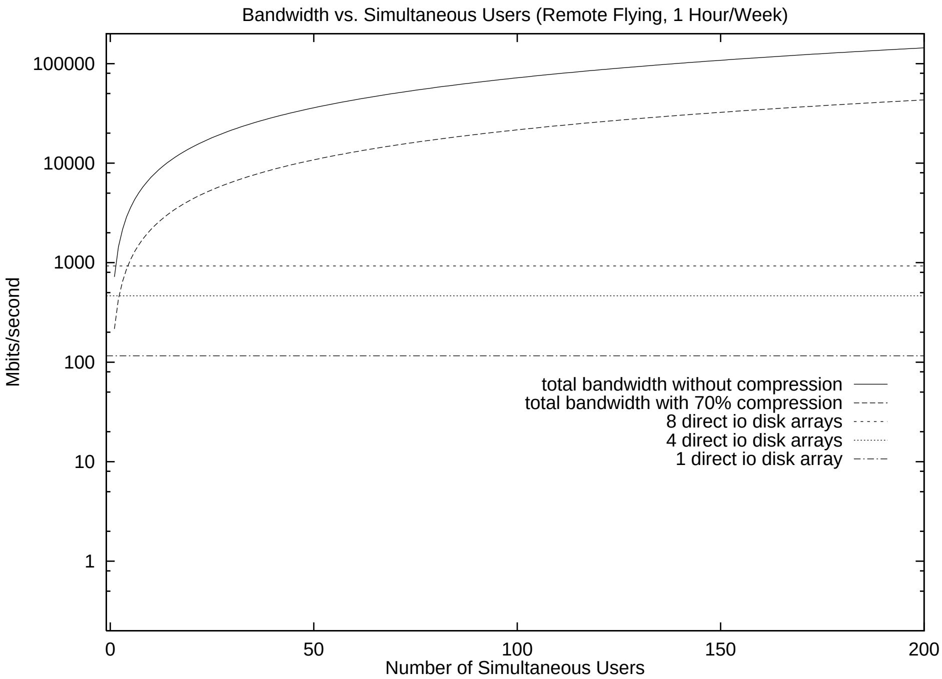

Fig. 5. Bandwidth versus Simultaneous Users. The Mbits/second axis is a log 10 scale. The increasing curves are bandwidths without compression and with compression. The horizontal lines are all disk throughput rates.

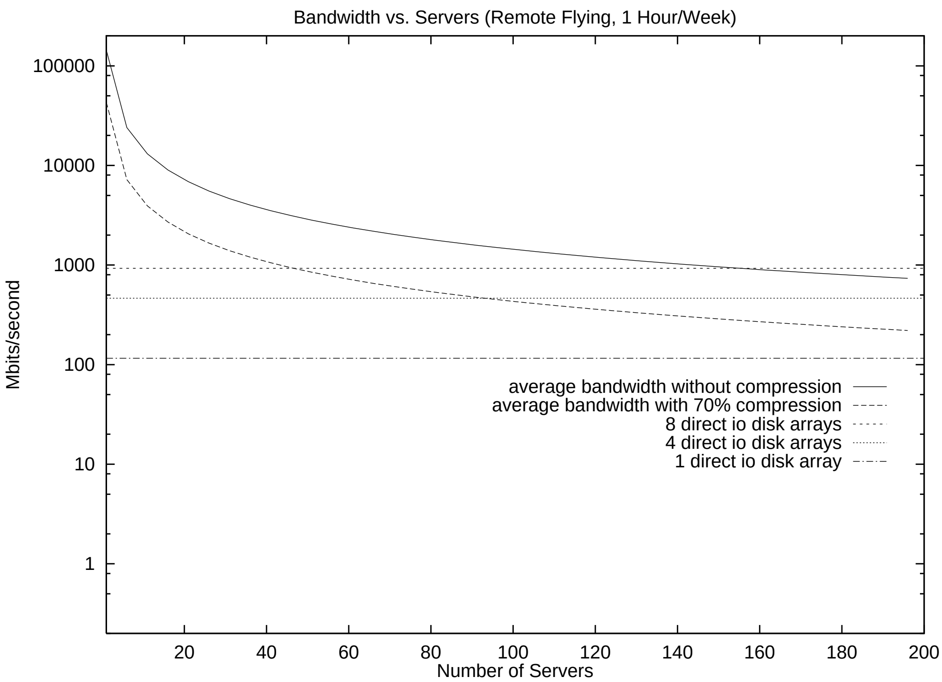

Fig. 6. Bandwidth versus Number of Servers. The decreasing curves are bandwidths with compression and without compression. The horizontal lines are disk rates.

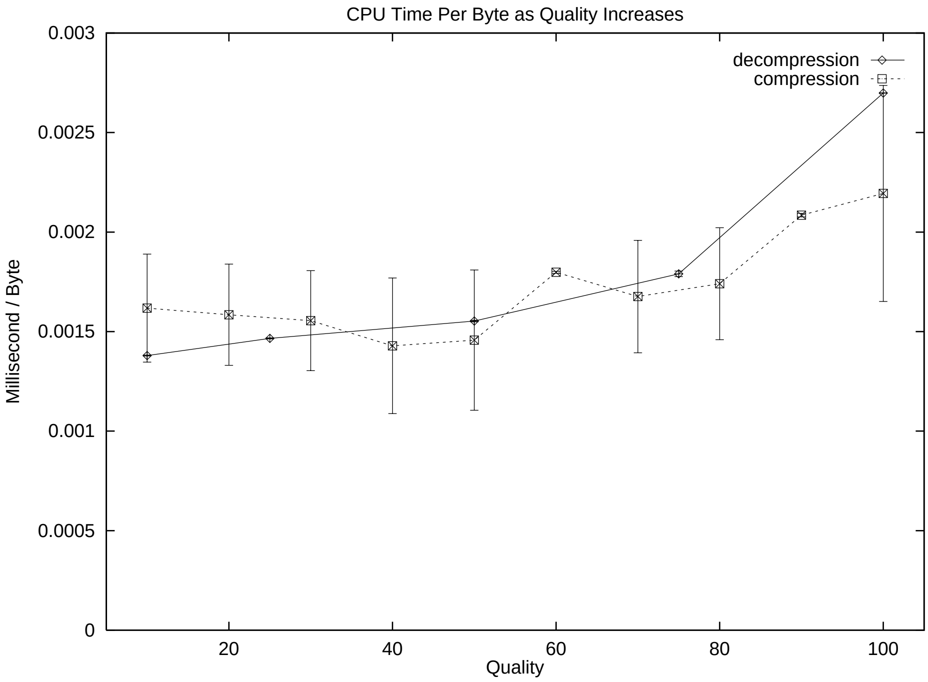

Fig. 7. CPU load for JPEG Compression and Decompression versus Quality. The vertical axis is CPU time in milliseconds/byte. The horizontal axis is the JPEG quality setting. All points are shown with 95% confidence intervals.

Compression	Operations	Seconds/frame	Mbits/second
None	read calculate send	32.6	0.74
Network	read calculate compress send	40.4	0.59
Network and Disk	read decompress calculate compress send	170	.14

TABLE I

Flying Throughput. Predicted Sun IPX CPU loads with the three ways to use compression for flying.

the CPU load for using specialized hardware, we assume that accessing specialized hardware is equivalent to one kernel call and that calls can take place in block sizes of 100 Kbytes. We also assume that all hardware is sufficiently fast to keep up with the CPU. Table II analyzes the Sun IPX CPU improvements from using such hardware for each flying component operating on one frame. Table III gives the CPU throughput predictions for an SGI Indigo 2 workstation using hardware support for the flying components.

We can now predict the CPU throughput required for flying. Figure 8 shows the load predictions versus simultaneous users and Figure 9 shows the load predictions versus servers. Flying throughput for a Sun IPX would be at the very bottom of these graphs. Even a 20 processor 100 MHz SGI Indigo 2 using compression cannot satisfy user flying requirements for even one user. We therefore consider the use of specialized hardware and kernel support for reading, compressing, calculating and sending.

Component	Software Load	Hardware Load
read uncompressed image	28.2 seconds	0.03 seconds
read compressed image	8.5 seconds	0.009 seconds
decompress image	131 seconds	-
compress frame	8.2 seconds	-
calculate frame	4 seconds	0.03 seconds
send uncompressed frame	3.8 seconds	0.006 seconds
send compressed frame	1.1 seconds	0.0002 seconds

TABLE II

Flying Component Loads. Sun IPX CPU loads induced by various flying components when done in software and hardware.

VIII. Quality

There is an old network saying: "Bandwidth problems can be cured with money. Latency problems are harder because the speed of light is fixed – you can't bribe God." David Clark, MIT.

So far we have explored the necessary network, disk, and CPU requirements to completely satisfy our user requirements. But when user requirements cannot be completely satisfied, there is often a tradeoff in system choices. For

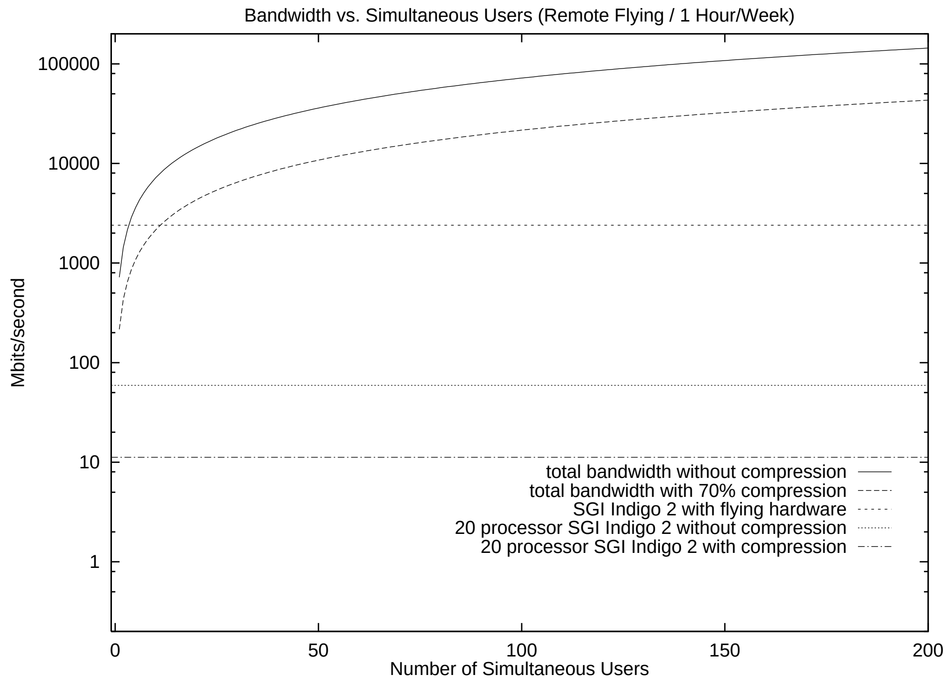

Fig. 8. Bandwidth versus Simultaneous Users. The upward sloping curves are total bandwidths with compression and without compression. The horizontal lines are CPU flying throughput rates. The top horizontal line is the predicted flying throughput of a 2 processor SGI Indigo 2 with specialized flying hardware.

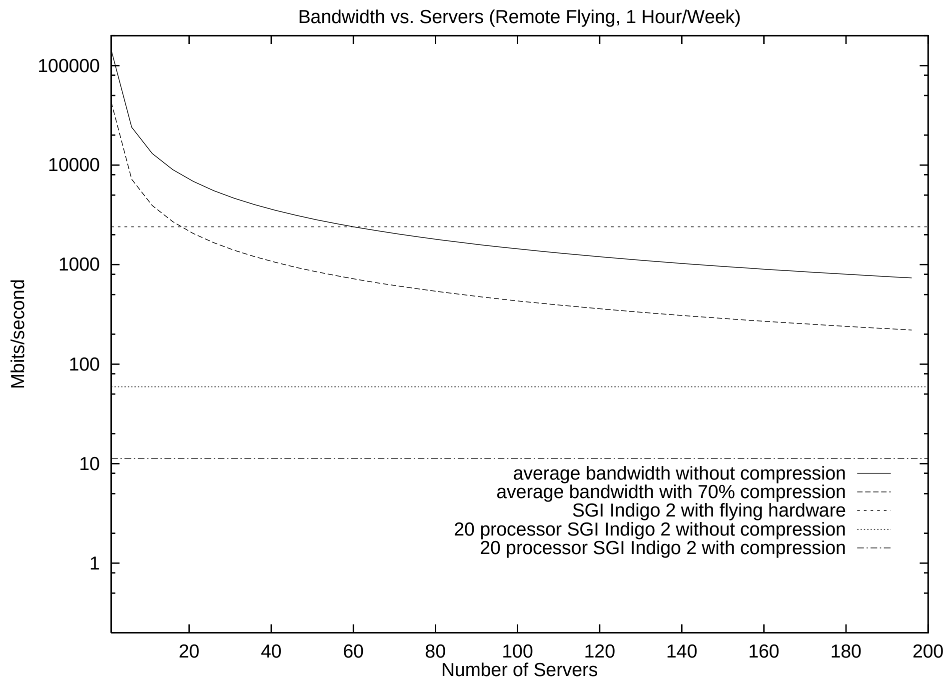

Fig. 9. Bandwidth versus Number of Servers. The upward sloping curves are the average bandwidths per server with compression and without compression. The horizontal lines are CPU flying throughput rates. The top horizontal line is the predicted flying throughput of a SGI Indigo 2 with specialized flying hardware.

Compression	Operations	Seconds/frame	Mbits/second
None	read calculate send	0.0170	1416
Network	read calculate compress send	0.0150	1608
Network and Disk	read decompress calculate compress send	0.0066	2400

TABLE III

Hardware Flying Throughput. Possible SGI Indigo 2 CPU loads induced by the three forms of flying when the CPU is equipped with specialized flying hardware.

example, if more compression is applied, bandwidth requirements for network and disk are reduced, but the CPU may become overloaded with the added decompression cost. One quantitative trade-off measure may be in terms of the application quality.

We have identified three components that influence application quality. There may be others:

- **Latency.** The time it takes the server to start processing a 3D image until the time the client displays the 2D frame we call latency.
- **Jitter.** Distributed applications on packet-switched networks invariably have variable packet inter-arrival times, causing gaps in the stream playout [16]. We call these gaps jitter. Jitter can be reduced or eliminated by buffering ahead in the data stream. When a packet arrives late, the next frame is already in the data buffer and no gap appears in the playout stream. The buffer size must be carefully chosen. Too large a buffer, and latency becomes intolerable. Too small a buffer, and there is jitter. There have been several heuristics used to determine buffer size with audio [17], but [11] has shown that none of these heuristics can control playout quality under all packet arrival distributions. In addition, choosing the *right* data to buffer can be extremely difficult for some applications. For flying, buffering may be done along a well-known neural pathway in anticipation of the user following it. However, in the event that the user turns away from this pathway, the server would have to send additional data to re-establish the buffer.
- **Data Reduction.** In order to reduce latency and/or jitter, or to reduce processor loads, an application may voluntarily accept some data reduction. Such reduction can take many forms, from reduced color, to jumbo pixels and smaller images to dropped frames.

Kleinrock and Naylor suggest a quality metric for audio streams based on jitter and latency [18]. We define the quality of our application based on latency, jitter and data reduction. We form our quality metric by assigning units and relative weights to each component. Using each component as one axis, we create a 3-dimensional quality space. One configuration of an application lies at one point in this space. The quality of the application can be computed by taking the Euclidean distance of the point from the origin. By assigning a weight of "1" to the upper bound of acceptability along each axis, we can create an acceptable quality plane of all points that have a quality of 1; all points inside the curve (towards the origin) will have acceptable quality while points outside will not.

Our model does not support jitter predictions yet. Jitter is difficult to model in visualization domains because the data are non-linear. Users may move in different directions at any time, making buffering difficult. Thus, in creating our model for application quality we consider only latency and data reduction.

We chose an equal weighting of milliseconds of latency and Mbits/second of data reduction. We assume more than 250 milliseconds of latency to be unacceptable, a level of tolerability suggested by other applications [11]. We also assume fewer than 20 Mbits/second of data provides too little data to be useful, and more than 720 Mbits/second (30 frames/second and 24 bits of color) provides no more useful data. Our quality is the Euclidean distance from the origin normalized over 250 milliseconds and 700 Mbits/second.

Figure 10 depicts our measure of quality. The individual points are all SGI Indigo 2 clients with a different number of processors. In this figure, we have not assumed any specialized flying hardware. We assume that the servers will be able to provide the bandwidth requested by all the clients to simplify the computation.

Figure 11 depicts quality versus the number of clients for both flying with compression and without. Clients are assumed to be 20 processor Indigo 2's without specialized hardware. The server is assumed to be an SGI Indigo 2 workstation with specialized hardware (see Section VII). The arrows indicate points at which the server can no longer keep up with the bandwidth requests by the clients. At this point, application performance decreases, as depicted by the increasing quality values. Thus, for fewer than 4 clients, compression decreases client-side quality, mostly because of the latency increase from the clients decompressing the images. However, for 5 or more clients, compression increases application quality because the server can meet the bandwidth requirements of more clients.

Note that this graph depicts the quality tradeoffs with one particular system configuration. Different system configurations may have different quality tradeoffs.

IX. Conclusions

Since the users of the Zoomable Brain Database will be distributed across the country, the system stress from flying will be spread over the underlying network. If the network topology appears as a backbone with six Network Access Points (NAP's), like the proposals for the national data

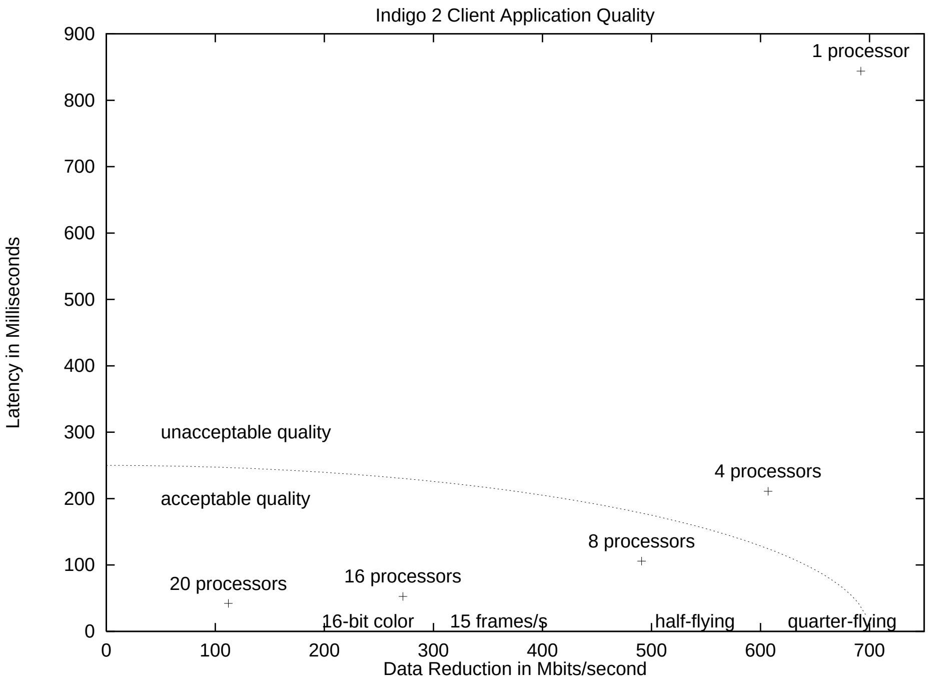

Fig. 10. Client Application Quality. A measure of quality for the client. The horizontal axis is the number of Mbits/second of data reduction received by the client. The vertical axis is the latency added by the client. The points are SGI Indigo 2's clients with different numbers of processors. The curve represents an acceptable level of quality; all points inside the curve will have acceptable quality while points outside will not. Note that the clients are not equipped with any special flying hardware.

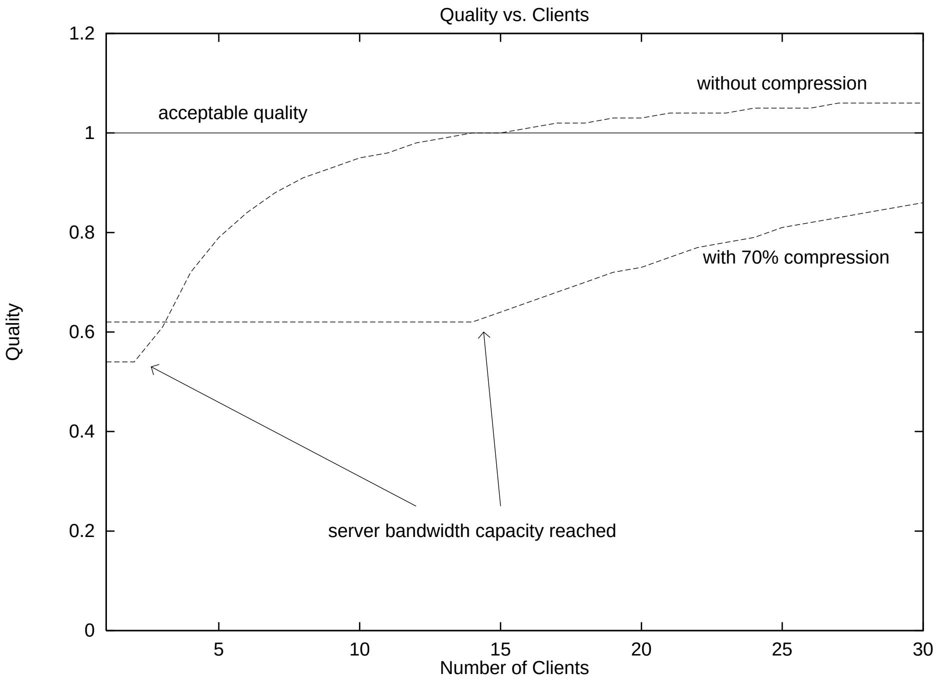

Fig. 11. Quality versus Clients. Clients are 100 processor SGI Indigo 2 workstations with no specialized hardware. The server is an SGI Indigo 2 workstation with specialized hardware (see Section VII). The arrows indicate points at which the server can no longer keep up with the bandwidth requests by the clients. At this point, perceived application performance decreases, as depicted by the increasing quality values.

highway [19], we can determine the bandwidth required for each link.

We assume:

- An equal distribution of users among servers,
- 50 servers, each directly connected to the backbone, and
- 70% compression.

We can use the analysis from previous sections to determine the system requirements:

Component	Mbits/second
Backbone:	43200
NAP's:	7200
Individual Connections:	216
50 CPU Servers:	4680

Figure 12 depicts the network for the constructed Zoomable Brain Database. Unfortunately, some of the proposals for the national data highway have bandwidths like those in figure 13.

We can only hope to fly on such a network by reducing the user-level flying requirements. [4] describes a number of modifications that can be done to the user requirements that will ease the system requirements. The server can reduce the frame resolution in each dimension by one-half (half-flying) to a one-fourth (quarter-flying), reducing the data needed by 1/8th and 1/64th respectively. Likewise, fewer bits of color decrease the data needed for each pixel. The table below summarizes the system benefits of several of these modifications:

	Reduction		
Modification	I/O	CPU	Network
8 bit color	0	2/3	2/3
half-flying	0	0	3/4
quarter-flying	0	0	15/16
3 frames/sec	9/10	9/10	9/10

Processing and sending 8-bit color reduces CPU and Network requirements by 2/3. Yet many neuroscientists may find scientific analysis difficult under such conditions. In half- or quarter-flying, one frame pixel represents 4 or 16 display pixels, respectively. Thus, network data rates are reduced by 3/4 and 15/16 respectively, but at the expense of image quality. Sending fewer frames per second reduces all system components. However, motion displayed at 3 frames/second has a much rougher flow than does motion displayed at 30 frames/second.

The data reduction from the above modifications can also be combined. If we modify the user-requirements to allow 8 bit color, 3 frames/second and 1/2 flying, the system requirements for the Zoomable Brain Databases would appear as in Figure 14. These reductions in user requirements make it possible to use the proposed infrastructure to achieve some of our goals, but will not be suitable for some neuroscience analysis.

All of our predictions have done the image computation at the server (*remote flying*) as opposed to image computation done at the client (*local flying*). But remote flying, while inducing significantly less bandwidth than local flying, increases the amount of server computation. With many database users, the server computation may become prohibitive. As client workstations become increasingly powerful, they may be able to readily perform the needed computations at a sufficient flying rate, reducing server load. In other environments, server load has proven critical for performance [20].

There are many areas for future research. A cost analysis of the tradeoffs between network bandwidths, disk data rates and CPU throughputs can determine what system configuration will be the most economical. Algorithms to manage playout buffers and reduce jitter will be important to preserve flying image quality. A careful study of the effects of flying hardware will determine bottlenecks in current proposed flying systems. The analysis techniques presented in this paper apply to other applications that can be described with analogous user requirements. As in [12], experiments to justify that predicted system requirements accurately meet user requirements are needed to validate the analysis techniques.

Advances in networking, such as those described in this paper, are needed to enable a new era of neuroscience. Researchers will be able to directly access high-quality pristine data from all areas of the brain. Hypotheses will be formulated, evaluated, and scientifically tested through interaction with data collected by scientists on the other side of the world. As we have shown, realizing this dream fully will require gigabits per second of sustained throughput per user, and terabits per second of aggregate bandwidth.

X. Acknowledgements

We would like to thank the reviewers for their valuable suggestions and Mike Stein and Joe Habermann for their many helpful comments.

References

[1] William L. Hibbard, Brian E. Paul, David A. Santek, Cahrles R. Dyer, Andre L. Battaiola, and Marie-Francoise Voidrot-Martinez, "Interactive visualization of earth and space science computations", *IEEE Computer*, vol. 27, no. 7, pp. 65 – 72, July 1994.

[2] T.T. Elvins, "A survey of algorithms for volume visualization", *Computer Graphics*, vol. 26, no. 3, pp. 194 – 201, Aug. 1992.

[3] Jaswinder Pal Singh, Anoop Gupta, and Marc Levoy, "Parallel visualization algorithms: Performance and architecural implications", *IEEE Computer*, vol. 27, no. 7, pp. 45 – 55, July 1994.

[4] J. Carlis, J. Riedl, A. Georgopoulos, G. Wilcox, R. Elde, J. H. Pardo, K. Ugurbil, E. Retzel, J. Maguire, B. Miller, M. Claypool, T. Brelje, and C. Honda, "A zoomable DBMS for brain structure, function and behavior", in *International Conference on Applications of Databases*, June 1994.

[5] E.R. Kandel, J.H. Schwartz, and T.M. Jessel, *Principles of Neural Science, 3rd Ed.*, Appleton & Lange, Norwalk, CT, 1991.

[6] B. M. Slotnick and C. M. Leonard, *A Stereotaxic Atlas of the Albino Mouse Forebrain*, Superintendent of Documents, US Government Printing Office, Washington DC, 1975.

[7] Gregory K. Wallace, "The JPEG still picture compression standard", *Communications of the ACM*, Apr. 1991.

[8] Ketan Patel, Brian C. Smith, and Lawrence A. Rowe, "Performance of a software mpeg video decoder", in *Proceedings of ACM Multimedia*, Anaheim, 1993.

[9] Charles Hansen and Stephen Tenbrink, "Impact of gigabit network research on scientific visualization", *IEEE Computer*, May 1993.

[10] Mengjou Lin, Jenwei Hsieh, David H.C. Du, Joseph P. Thomas, and James A. MacDonald, "Distributed network computing over

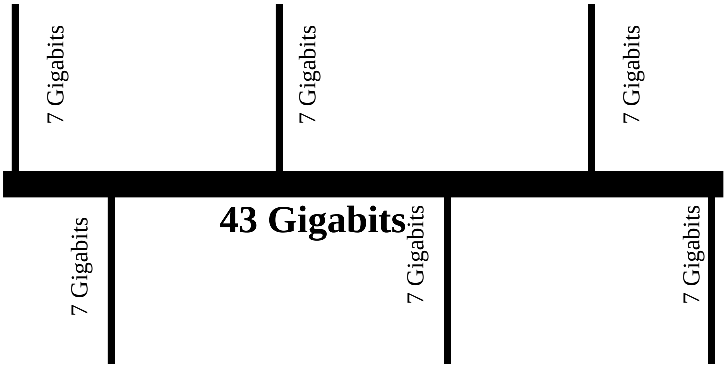

Fig. 12. Bandwidth requirements for the Zoomable Brain Database system on a topology similar to a possible national data highway.

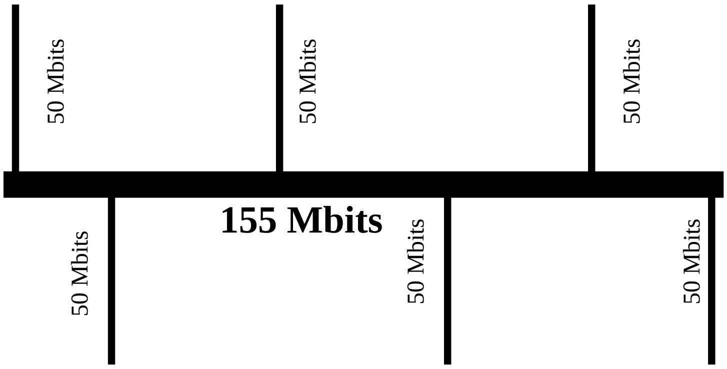

Fig. 13. A proposed national data highway network. In order to allow flying in this environment, user-level requirements must be relaxed.

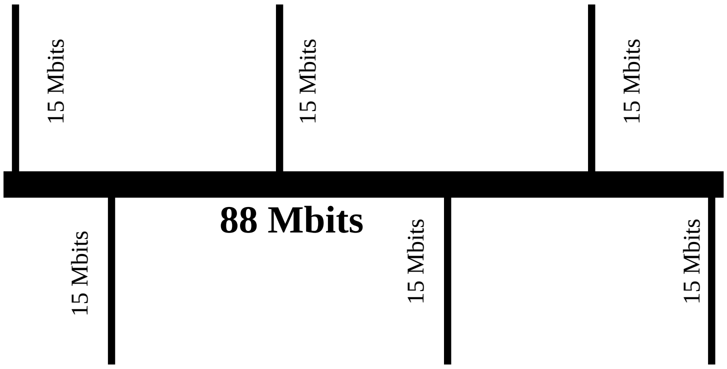

Fig. 14. Bandwidth requirements for the Zoomable Brain Database system on a topology similar to a proposed national data highway. User requirements have been reduced to ease system requirements.

local atm networks", *ATM LANs: Implementation and Experience with An Emerging Technology*, 1995.

[11] Daniel Frankowski and John Riedl, "Hiding jitter in an audio stream", Tech. Rep. Technical Report 93-50, University of Minnesota Department of Computer Science, 93.

[12] Mark Claypool and John Riedl, "Silence is golden? The effects of silence deletion on the CPU load of an audio conference", in *Proceedings of IEEE Multimedia*, Boston, May 1994.

[13] Jack J. Dongarra, "Performance of various computers using standard linear equations software", Tech. Rep. CS-89-85, University of Tennessee, Feb. 1994, To obtain a postscript copy, send email to netlib@ornl.gov with message body: send performance from benchmark.

[14] Thomas M. Ruwart and Matthew T. O'Keefe, "Storage and interfaces '94", in *Proceedings of the Storage and Interfaces '94*, Santa Clara, CA, Jan. 1994.

[15] gsnow@clark.edu Gary Snow, "Specint and specfp 1992 numbers", *comp.benchmarks*, Nov. 1993.

[16] Domenico Ferrari, "Delay jitter control scheme for packet-switching internetworks", *Computer Communications*, vol. 15, no. 6, pp. 367–373, July 1992.

[17] Henning Schulzrinne, "Voice communications across the internet: A network voice terminal", Tech. Rep., University of Massachussetts Department of Electrical Engineering, Aug. 1992.

[18] Kleinrock and Naylor, "Stream traffic communication in packet switched networks: Destination buffering considerations", *IEEE Transactions on Communications*, vol. COM-30, no. 12, pp. 2527–2534, Dec. 1982.

[19] Andy Reinhardt, "Building the data highway", *Byte*, Mar. 1994.

[20] Edward D. Lazowska, John Zahorjan, David R. Cheriton, and Willy Zwaenepoel, "File access performance of diskless workstations", *ACM Transactions on Computer Systems*, Aug. 1986.