SEQUOIA 2000 -- A REFLECTION ON THE FIRST THREE YEARS

Michael Stonebraker
EECS Department
University of California, Berkeley

Abstract

This paper describes the SEQUOIA 2000 project and its implementation efforts during the first three years. Included are the objectives we had, how we chose to address them and some of the lessons we learned from this endeavor.

1. INTRODUCTION

The purpose of the SEQUOIA 2000 project is to build a better computing environment for Global Change Researchers, hereafter referred to as SEQUOIA 2000 clients. Such researchers investigate issues such as global warming, ozone depletion, environment toxification, and species extinction, and are members of Earth Sciences Departments at Universities and National Laboratories. SEQUOIA 2000 is the Digital Equipment Corporation (DEC) flagship research project for the 1990's, replacing Project Athena at MIT, and its original conception is described in [STON91].

The participants in SEQUOIA 2000 are four types of investigators:

Computer Science Researchers. Their charge is to build a prototype environment which better serves the needs of the target clients. Investigators are associated with SEQUOIA 2000 from the Computer Science Division at Berkeley, the Computer Science Department at UC/San Diego, the School of Library and Information Studies at Berkeley and the San Diego Supercomputer Center.

Earth Science Researchers. Their charge is to explain their needs to the Computer Science investigators and to use the resulting prototype environment to perform better science. Earth Science investigators are associated with SEQUOIA 2000 from the Department of Geography at UC/Santa Barbara, the Atmospheric Science Department at UCLA, the Climate Research Division at the Scripps Institution of Oceanography, and the Department of Earth, Air and Water at UC/Davis.

Governmental agencies. In order to ensure that the "rubber meets the road" rather than the sky, governmental organizations have been recruited which are impacted by global change matters. The responsibility of these members is to steer SEQUOIA 2000 research, so it moves in a direction ensuring its applicability to their problems. Governmental participation in SEQUOIA 2000 comes from the State of California Department of Water Resources (DWR), the State of California Department of Forestry, the Co-ordinated Environment Research Laboratory (CERL) of the US Army, NASA, NOAA and the United States Geologic Survey (USGS).

Industry. Their charge is to use the SEQUOIA 2000 technology and offer guidance and research directions. In addition, they are a source of free or discounted computing equipment. Current industrial participants are Epoch, Hewlett-Packard, Hughes, Illustra, Metrum, MCI, Picturetel, RSI, SAIC, Siemens, and TRW.

The purpose of this paper is to reflect on our goals, how we decided to attempt to achieve them and the results we accomplished during the first three years. As a result, Section 2 first indicates the Sequoia 2000 goals and the architecture that we decided to pursue to achieve these goals. Moreover, it discusses the state of our software efforts in the various areas. The most important lesson that we have learned is that SEQUOIA 2000 must be considered as an **end-to-end** problem. Hence, clients can only be satisfied if all pieces of our architecture work together in a harmonious fashion. Also, many services required by the clients must be provided by all elements of the architecture, each working together. In Section 3 we illustrate this end-to-end characteristic of SEQUOIA 2000 by discussing three issues that cross all parts of the system. Then Section 4 indicates other specific lessons that we have learned from the first three years. Lastly, Section 5 concludes the paper.

2. THE SEQUOIA 2000 ARCHITECTURE

2.1. Introduction

Our architecture is motivated by four fundamental Computer Science objectives, which we present in turn.

1) Support high performance I/O on terabyte data sets

Our clients are frustrated by current computing environments because they cannot effectively store the massive amounts of data desired for research purposes. Our four academic clients plus DWR collectively would like to store about 100 Terabytes of information. Much of this is common data sets used by multiple investigators. As a result, they would like high performance system software

This research was sponsored Digital Equipment Corporation under Contract Number 1243

that would effectively allow sharing of assorted tertiary memory devices. Unlike other scientific computing users, much of their I/O activity is random access. For example, DWR is digitizing the agency's 500,000 slide library and is putting it on line using the SEQUOIA 2000 environment. This data set has some locality of reference but will have considerable random activity.

2) Put all data in a DBMS

Our clients have been sold on the merits of moving all their data to a Data Base Management System (DBMS). In this way, the meta data that describes their data sets can be maintained, assisting them with the ability to retrieve needed information. More importantly, it will facilitate sharing of information. Because a DBMS will insist on a common schema for shared information, it will allow the researchers to define this schema, and then all must use a common notation for shared data. This will be a big improvement over the current situation where every data set exists in a unique format, and must be converted by any researcher who wishes to use it.

3) Provide better visualization tools

Our clients are users of the popular scientific visualization tools such as Explorer, Khoros, AVS and IDL. They are frustrated by aspects of these products and are anxious for a next generation toolkit.

4) Provide high speed networking

Our clients realize that a 100 terabyte storage server(s) will not be located on their desktop. Moreover, it is likely to be at the other end of a wide area network (WAN) from their client machines. Since their visualization scenarios invariably involve animation, for example showing the last 10 years of the ozone hole by playing time forward, they require ultra high speed networking to move sequences of images from a server machine to a client machine.

To address these concerns, we adopted the four-level architecture described in Figure 1. In the next four subsections we discuss our efforts at each of these levels. We then close the section with a discussion of SEQUOIA 2000 networking, which provides the "glue" to connect the elements of the architecture.

2.2. The Footprint Layer

This software system shields higher level software, such as file systems, from device specific characteristics of robotic devices. These include specific robot commands, block sizes and media specific issues. As such, Footprint can be thought of as a common robot device driver. At the moment we have a footprint implementation for each of the four tertiary memory devices used by the project. These are a Sony WORM optical disk jukebox, an HP rewritable optical disk jukebox, a Metrum VHS tape

Collectively these four devices and the CPUs and disk storage systems in front of them have been named **Bigfoot** after the legendary very tall recluse spotted occasionally in the Pacific Northwest. Bigfoot is deployed on DECsystem hardware running the Ultrix operating system. In the near future we expect to port our software environment to the Alpha platform running the Alpha/NT operating system.

2.3. The File System Layer

On top of Footprint we have written two file systems that manage data in the Bigfoot multi-level memory hierarchy. The first file system is Highlight [KOHL93]. It is an extension of the Log Structured File System (LFS) pioneered for disk devices by Ousterhout and Rosenblum [ROSE92]. Specifically, LFS treats a disk device as a single continuous **log** onto which newly written disk blocks are appended. Moreover, blocks are never overwritten, so a disk device can always be written sequentially. Hence, LFS turns a random write environment into a sequential write environment. In particular problem areas, this may lead to much higher performance, and benchmark data which supports this conclusion can be found in [SELT93]. In addition, LFS can always identify the last few blocks that were written prior to a crash by finding the end of the log at recovery time. Repair of the file system is then very fast, because potentially damaged blocks are easily found. This approach should be contrasted to conventional file systems where a laborious check of the disk must be performed to ascertain disk integrity.

We have extended LFS to support tertiary memory by adding a second log structured file system on top of Footprint for tertiary memory. This file system also writes tertiary memory blocks sequentially, obtaining the performance characteristics of LFS. Lastly, Highlight adds migration and bookkeeping code that treats the disk LFS file system as a cache for the tertiary memory one. In summary, Highlight should give very good performance on a workload that is "write-mostly". Since SEQUOIA 2000 clients want to archive vast amounts of data, Highlight has the potential for good performance in the SEQUOIA 2000 environment.

The second file system is Inversion [OLSO93]. Most Data Base Management Systems, including the one we are using for SEQUOIA 2000, support binary large objects (blobs), which are arbitrary length, variable-length byte strings. Like several commercial systems, our data manager POSTGRES [STON90] stores large objects in a customized storage system directly on a **raw** storage device. As a result, it is a straightforward exercise to support conventional files on top of DBMS large objects. In this way, every read or write is turned by the front end into a query or update, which is processed directly by the DBMS. Simulating files on top of DBMS large objects has several advantages. First, DBMS services such as transaction management and security are automatically supported for files. In additional novel characteristics of our next generation DBMS, including **time travel,** and an extensible type system for all DBMS objects, are automatically available for files. Of course, the possible disadvantage of files on top of a DBMS is poor performance. As reported in [OLSO93], Inversion performance is exceedingly good when large amounts of data are read and written, a characteristic of the SEQUOIA 2000

The SEQUOIA 2000 Architecture
Figure 1

workload.

At the present time, Highlight is more or less operational on 4.3BSD Unix. However, the effort to port the code to Ultrix turned out to be rather difficult, and we have not succeeded in generating a working Highlight on Ultrix. Inversion, on the other hand, is deployed on Ultrix and used to manage our Sony WORM jukebox. Unfortunately, the reliability of our prototype system has not measured up to user expectations. Our clients have a strong desire for Commercial Off-The-Shelf (COTS) software, and are frustrated by documentation glitches, bugs, and crashes.

As a result, we have also deployed two commercial file systems, Epoch and Unitree. Epoch has proved to be quite reliable but does not support either of our large capacity robots. Hence, it is used heavily, but only for small data sets. There are many versions of Unitree, and the one we are running (Alpha/OSF-1 Unitree for the Metrum) has not met user expectations. As such, we face a conundrum with regard to tertiary memory file systems. Our prototypes are not sufficiently stable to warrant "prime time"; however COTS packages available for the Alpha/Metrum combination have the same problem. We are scratching our head on how to move forward, and our advice to other tertiary memory users is to test any software you are considering very carefully.

2.4. The DBMS Layer

As noted in Figure 1, some users will simply run application programs against the file system, and will have no use for DBMS technology. Others will store their data in a DBMS. In order to have any chance of meeting SEQUOIA 2000 client needs, a DBMS must support spatial data such as points, lines, and polygons. In addition it must support the large spatial arrays in which satellite imagery is naturally stored. As noted in [STON93], these characteristics are not met by popular general purpose relational and object-oriented DBMSs. The best fit to client needs is a special purpose Geographic Information Systems (GIS), or a next general prototype DBMS. Since we have one such next-generation system within the project, we have elected to focus our DBMS work on this system, POSTGRES.

To make POSTGRES suitable for SEQUOIA 2000 use, we require a **schema** for all SEQUOIA data. This data base design process is evolving as a cooperative exercise between various data base experts at Berkeley, SDSC, CERL and SAIC. The Sequoia schema is the collection of **metadata** describing the data stored in the POSTGRES DBMS on Bigfoot. Specifically, these metadata comprise:

- a standard **vocabulary** of terms with agreed-upon definitions that are used to describe the data;
- a set of **types,** instances of which may store data values;
- a hierarchical collection of **classes** that describe aggregations of the basic types; and
- **functions** defined on the types and classes.

The SEQUOIA 2000 schema accommodates four broad categories of data: scalar, vector, raster, and text. Scalar quantities are stored as POSTGRES types and assembled into classes in the usual way. Vector quantities are stored in special line and polygon types. Vectors are fully enumerated (as opposed to an arc-node representation) to take advantage of POSTGRES indexed searches. The advantages of this representation are discussed in more detail in [STON93].

Raster data comprises the bulk of SEQUOIA 2000 data. These data are stored in POSTGRES multidimensional arrays objects. The contents of textual objects (in PostScript, or scanned page bitmaps) are stored in a POSTGRES document type. Both documents and arrays make use of a POSTGRES large object storage manager that can support arbitrary length objects.

In addition, we are in the process of loading this schema with several terabytes of client data. This load process is expected to continue for the duration of the project. In addition, schema migration must be planned for as the schema evolves. How to reformat a multiterabyte data base in finite time is currently an open question that is troubling us.

Furthermore, we have tuned POSTGRES to meet the needs of our clients. The interface to POSTGRES arrays has been improved and a novel **chunking** strategy [SARA93] is now operational. Instead of storing an array by ordering the array indices from fastest to slowest changing, this system chooses a **stride** for each dimension and stores "hyperslabs" of the correct stride sizes in each storage object. When user queries inspect the array in more than one way, this technique results in dramatically superior retrieval performance.

Moreover, SEQUOIA clients typically run queries with user-defined functions in the predicate. Moreover, many of the predicates are very expensive in CPU time to compute. For example, The Santa Barbara group has written a function, SNOW, that recognizes the snow covered regions in a satellite image. It is a user-defined POSTGRES function that accepts an image as an argument and returns a collection of polygons. A typical query using the SNOW function for the table:

IMAGES (id, date, content)

would be to find the images that were more than 50% snow observed subsequent to June 1992. In SQL, this query is expressed as:

```
select id
from IMAGES
where AREA (SNOW (content)) > 0.5
and date > "June 1, 1992"
```

The first clause in the predicate consumes millions of instructions to evaluate, while the second requires a few hundred. As such, the DBMS must be cognizant of the CPU cost of clauses when constructing a query plan, a cost component ignored by most previous optimization work. In [HELL93], we indicate our work in extending the POSTGRES optimizer to deal intelligently with expensive functions. Furthermore, it is highly desirable to allow popular expensive functions to be **precomputed.** In this way they can be evaluated once, rather than once per query in which they appear. Our approach to this issue is to allow data bases indexes on a function of the data, rather than just on the data object itself. Hence, the data base administrator can specify that a B-tree index be built for the function, AREA (SNOW(content)). As such, areas of images are arranged in sort order in a B-tree and the first clause in the above query is now very cheap to compute. Using this technique the function is computed at data entry or data update time and not a query evaluation time. A consequence of function indexing is that it may be very time consuming to insert a new image into the data base, since function computation is now included in the load transaction. To deal with the undesirability of having very lengthy response time for loads, we have also explored **lazy** indexing and **partial** indexing. In this way, index building does not need to be synchronous with data loading.

2.5. The Application Layer

There are five elements of our application layer, which we consider in turn. The first is the use of an off-the-shelf visualization tool, and we have converged around the use of IDL and AVS for project activities. AVS is liked for its easy-to-use "boxes and arrows" user interface, while IDL has a more conventional linear programming notation. On the other hand, IDL is liked for its better 2-D graphics features. Both IDL and AVS allow a user to read and write file data. To connect to the DBMS, we have written an AVS-POSTGRES bridge. This program allows one to construct an ad-hoc POSTGRES query and pipe the result into an AVS boxes-and-arrows network. As a result, our clients can use AVS for further processing on any data retrieved from the DBMS. Finally, IDL is being interfaced to AVS by the vendor. As a result data retrieved from the data base can be moved into IDL using AVS as an intermediary.

AVS has a collection of severe disadvantages as a visualization tool for our clients. First, it has a type system that is different from the POSTGRES type system and has no direct knowledge of the common SEQUOIA 2000 schema. In addition AVS has a severe appetite for main memory. Architecturally, AVS depends on virtual memory to pass results between various boxes. In addition, it maintains the output of each box in virtual memory for the duration of an execution session. In this way, the user can change a run-time parameter somewhere in the network, and AVS will recompute only the "downstream" boxes by taking advantage of the previous output. As a result, SEQUOIA 2000 clients, who produce very large intermediate results, consume large amounts of both virtual and real memory. In fact, they report that 64 Megs of real memory on a workstation is often not enough to enable serious AVS use. Furthermore, AVS has no support for "zooming" into data of interest to obtain higher resolution, nor does it keep track of the history of how any given data element was constructed, i.e. the so-called **data lineage** of an item. Lastly, AVS has a "video player" model for animation, which it too primitive for many SEQUOIA 2000 clients.

To correct these deficiencies, we have designed a new boxes-and-arrows programming environment that is "DBMS-centric", i.e. the environment type system is the same as the DBMS type system. Moreover, the user interface gives the user a "flight simulator" paradigm for browsing the output of a network. In this way, the visualizer can "navigate" around his data and then zoom in to obtain additional data on items of particular interest. This environment, named Tioga, is a joint project between Berkeley and SDSC, and its preliminary design is presented in [STON93B]. Moreover, a first prototype (early Tioga) is currently running and its features are described in [WOOD94].

A third cornerstone of our architecture is a browsing capability for textual information of interest to our clients, called **Lassen.** This text system has two components. The first is a facility for constructing weighted keyword indexes for the the words in a POSTGRES document. This indexing systems, Cheshire [LARS91], builds on the pioneering work of the Cornell Smart system and operates as the action part of a POSTGRES rule [STON90] which is triggered on each document insertion, update or removal. The second piece of Lassen is a natural language understanding front-end query tool that allows a user to ask for all documents which satisfy a collection of keywords by inquiring in a subset of Natural

English. Lassen is now operational, and retrievals can be requested against the currently loaded collection of SEQUOIA 2000 documents.

In addition, we expect to move Lassen to a Z39.50 protocol. In this way, the client portion of Lassen would emit Z39.50 and we would write a Z39.50 to POSTGRES translator on the server side. In this way, the Lassen client code can access non-SEQUOIA 2000 information and the SEQUOIA 2000 server can be accessed by text retrieval front ends other than Cheshire.

Our fourth thrust in the application layer is a facility to interface the UCLA General Circulation Model (GCM) to POSTGRES. This program is a "data pump" because it pumps data out the simulation model and into POSTGRES. As such, it has been named **the big lift** after the DWR pumping station that raises Northern California water over the Tehachapi Mountains into Southern California.

Basically, the UCLA GCM produces a vector of simulation output variables for each time step of a lengthy run for each **tile** in a three dimensional grid of the atmosphere and ocean. Depending on the scale of the model, its resolution, and the capability of the serial or parallel machine on which the model is running, the UCLA GCM can produce anywhere for 0.1 - 10.0 Mbytes/sec of output. The purpose of the big lift is to install this data into a POSTGRES data base, in real time. UCLA scientists can then use AVS or eventually Tioga to visualize their simulation output. It is likely that the big lift will have to exploit parallelism in the data manager, if it is required to keep up with the execution of the model on a massively parallel architecture.

The last application system is called **Hollywood.** Since SEQUOIA 2000 is a distributed project, we learned early that airplane tickets and electronic mail were not sufficient to keep project members working coherently as a distributed team. As a result, we purchased conference room videoteleconferencing equipment for each project site. This technology costs around $50,000 per site and allows multiway teleconferences over ISDN lines.

Although the conference room equipment has helped project communication immensely, it must be set up and torn down at each use because it occupies rooms at each site, otherwise used as classrooms. As such, it tends to be used for arranged conferences, and not for "spur of the moment" interactions. To alleviate this shortcoming, SEQUOIA 2000 has also invested in desktop videoteleconferencing. A video compression board, microphone, speakers, network connection, video camera, and appropriate software will turn a conventional workstation into a desktop teleconferencing facility. In addition, video can be easily transmitted over the network interface, present in virtually all SEQUOIA 2000 client machines. We are using the Mbone software suite to connect about 30 of our client machines in this fashion, and are moving most of our teleconferencing activities over to desktop technology. This effort, named Hollywood, strives to further improve the ability of SEQUOIA 2000 researchers to communicate.

It should be clearly noted that the SEQUOIA 2000 researchers do not really need "groupware", i.e. the ability to have common windows on multiple client machines separated by a WAN, in which common code can be run, updated and inspected. Rather, our researchers need a way to hold impromptu discussions on project business. As such they want a low-cost multicast "picturephone" capability, and our desktop videoteleconferencing efforts are focused in this direction.

2.6. The Network Layer

The last topic to discuss is the SEQUOIA 2000 networking agenda. In Figure 1, it is possible for the implementation of each layer to exist on a different machine. Specifically, the application can be remote from the DBMS which can be remote from the file system which can be remote from the storage device. Each layer assumes a local UNIX socket connection or a LAN or WAN connection using TCP/IP. Actual connections among SEQUOIA 2000 sites use either the internet or a dedicated T3 network, provided to the project as part of the University of California contribution to the SEQUOIA 2000 project.

The networking team has taken the position that DEC Alphas are plenty fast enough to route T3 packets. Hence, the project is using conventional workstations as routers instead of "custom iron". Furthermore, SEQUOIA/net has installed a unique **guaranteed delivery** service, through which an application can make a **contract** with the network that will guarantee a specific bandwidth and latency if the client agrees not to try to send faster than the contract. These algorithms require a "set-up" phase for a connection that will allocate bandwidth on all the lines and in all the switches, and are based on the work in [FERR90].

Lastly, the network researchers are concerned that Ultrix and OSF-1 copy every byte four times in between retrieving it from the disk and sending it out over a network connection. Even Alphas may not be fast enough to endure this kind of backplane bandwidth. As such, we have made modifications to Ultrix that will "fast path" the network connection through the operating system to cut down on senseless overhead.

3. SEQUOIA 2000 AS AN END-TO-END PROBLEM

The major lesson that we have learned from SEQUOIA 2000 is that many issues faced by our clients cannot be isolated to a single layer in the SEQUOIA 2000 architecture. In this section, we illustrate three such **end-to-end** problems, guaranteed delivery, compression and abstracts.

3.1. Guaranteed Delivery

It is clear that guaranteed delivery must be an end-to-end contract. Suppose a SEQUOIA 2000 client wishes to visualize a specific computation, for example, say he wants to observe Hurricane Andrew as it moves from the Bahamas to Florida to Louisiana. Specifically, he wishes to visualize appropriate satellite imagery at 500 x 500 resolution in 8 bit color at 10 frames per second. Hence, he requires 2.5 Mbytes/sec of bandwidth to his screen. The following scenario might be the computation steps that occur:

The DBMS must run a query to fetch the satellite imagery. It might require returning a 16 bit data value for each pixel that will ultimately go to the screen. Hence, the DBMS must agree to execute the query in such a way that it returns 5.0 Mbytes/sec.

The storage system at the server will fetch some number of I/O blocks from secondary and/or tertiary memory. DBMS query optimizers can make an accurate guess for how many blocks they need to read to satisfy the query. It is an easy extension for the DBMS to generate a guaranteed delivery contract which the storage manager must satisfy that will in turn allow the DBMS to satisfy its contract.

The network must agree to deliver 5.0 Mbytes/sec. over the link connecting the client to the server. The SEQUOIA/net software expects exactly this sort of contract request.

The visualization package must agree to translate the 16 bit data element into an 8 bit color and render the result onto the screen at 2.5 Mbytes/sec.

In short, guaranteed delivery is a collection of contracts which must be adhered to by the DBMS, the visualization package, the storage system and the network. One approach to architecting these contracts is discussed in [STON93B].

3.2. Abstracts

One aspect of the SEQUOIA 2000 visualization process is the the necessity of abstracts. Consider the above Hurricane Andrew example. The client might initially want to browse the Hurricane at 100 x 100 resolution. Then, if he found something of interest, he would like to **zoom** in and increase the resolution, usually to the maximum available in the original data. This ability to change the amount of resolution in an image dynamically has been termed **abstracts.**

It should be clearly noted that abstracts are a much more powerful construct than merely providing resolution adjustment. Specifically, obtaining more detail may entail moving from one representation to another. For example, one could have an icon for a document, zoom in to see the abstract and then zoom in further to see the entire document. Hence, zooming can change from iconic to textual representation. This use of abstracts was popularized in the DBMS community by SDMS [HERO80].

SEQUOIA 2000 clients wish to have abstracts. However, it is clear that they can be managed by the visualization tool, the network, the DBMS, or the file system. In the former case abstracts are defined for boxes-and-arrows networks as noted in [STON93B]. In the DBMS, abstracts would be defined for individual data elements or for data classes. Furthermore, if the network manages abstracts, then it will use them to automatically lower resolution to eliminate congestion. Much research on the optimization of network abstracts (called hierarchical encoding of data in that community) has been presented. Lastly, in the file system abstracts would be defined for files. There are SEQUOIA 2000 researchers pursuing all four possibilities, and it is expected that this notion will be one of the powerful constructs to be used by SEQUOIA 2000 software, perhaps in multiple ways.

3.3. Compression

The SEQUOIA 2000 clients are adamant on this matter; they are open to any compression scheme as long as it is lossless. As scientists, they believe that ultimate resolution may be required to understand some future phenomenon. Moreover, it is not possible to predict what or where this phenomenon might occur. Hence, the only alternative is to keep everything at full resolution.

Some SEQUOIA 2000 data is not economically compressible, and should be stored in clear form. For such data, the inclusion of abstracts offers a mechanism to lower the bandwidth required between the storage device and the visualization program. However, no saving of tertiary memory space via compression is available for such data.

On the other hand, certain data is compressible, and should be stored in compressed form. As such, the question arises as to when compression and decompression should occur. The only concept that makes any sense is the principle of **just in time** decompression. For example, if the storage manager compresses data as it is written and then decompresses it on a read, then the network manager may then recompress the data for transmission over a WAN to a remote site where it will be decompressed a second time. Obviously, data should be moved in compressed form and only decompressed when absolutely necessary. In general, this will mean in the visualization system on the client machine. If search criteria are performed on the data, then the DBMS may have to decompress the data to perform the search. Lastly, it is possible that an application resides on the same machine as the storage manager. If so, the file system must be in charge of decompressing the data. As should be clear, all software modules in the SEQUOIA 2000 architecture must co-operate to perform just-in-time decompression and as-early-as-possible compression. Like guaranteed delivery, compression is is a task in which everybody must cooperate.

4. SPECIFIC LESSONS LEARNED

In this section we discuss other specific lessons that we have learned from the first three years of the SEQUOIA 2000 experience.

1) Infrastructure is necessary, time-consuming and very expensive.

It was crystal clean within a short period of time that e-mail and airplane tickets would not result in the desired degree of co-operation from geographically dispersed researchers from different disciplines. As such, a significant investment in infrastructure was made. This included "all hands" meetings, which are now held twice a year, and video teleconferencing equipment at each site. Over this video link, we interact by holding a weekly distributed seminar, semi monthly operations committee meetings, occasional steering committee meetings, and meetings between researchers with common interests. Current video teleconferencing equipment has lousy video quality, and it takes special skills to run "distance meetings". Nevertheless, the equipment has proved valuable in generating cohesion in the dispersed project. We have now installed desktop video teleconferencing systems on 30 Sequoia workstations, and expect to replace our current conference room equipment with next-generation desktop technology.

In addition, we have run a "distance learning" experiment in which a course taught by one of the SEQUOIA 2000 faculty at the Santa Barbara campus was broadcast over our video teleconferencing equipment to 4 other sites, where students could take the course for credit at their respective campuses. This experiment proved very popular and students have requested additional courses taught in this manner. Of course, the overhead of setting this up was overwhelming; a new course had to be added at each campus, and every step in the approval process required briefings on the fact that the real instructor was from a different campus, and how everything was going to work.

On the other hand, we also tried running a Computer Science Colloquium using this technology. In all, we broadcast from various sites to six Computer Science departments around the country. Student interest started off very high, because they could listen to a "star-studded" lineup of speakers. Such speakers could be recruited easily, because they only had to locate the nearest compatible equipment and then get to that site. In this way, no airplane travel was required. However, this experiment failed badly because attendance dropped off badly as the semester wore on. By the end, attendance was down to embarassing levels.

The basic problem was that speakers were typically not skilled in using the medium. Hence, they would put way too much information on slides and then flip though slides before remote sites could get a complete transmission. In the hands of untrained presenters, the technology is a disaster. Also, the question and answer period could not be very interactive because of the multitude of sites. As such, the experiment ended after one semester and will not be repeated.

2) There was often a mismatch between the expectations of the Sequoia Earth Scientists and Computer Scientists.

Rather simplistically, the Computer Scientists on the SEQUOIA 2000 team wanted access to knowlegable application specialists who could describe their problem in terms understandable to the Computer Scientist. The "CS type" then wanted to think through an elegant solution, verify with the "ES type" that it was appropriate, and then prototype the result.

The ES types wanted bulletproof "production quality" solutions to their problems. In short, the ES types wanted a Commercial Off The Shelf (COTS) solution while the CS types wanted to write a proof of concept prototype. Put differently, the ES types were unsympathetic about poor documentation, bugs, and crashes.

It has taken considerable time to get the expectations of everybody "onto the same page". This has been especially true for ES use of POSTGRES, which suffers from all three of the above characteristics.

We found that the best way to make forward progress was to ensure that each ES group using SEQUOIA 2000 prototype code had one or more sophisticated staff programmers who could deal successfully with the quirks of prototype code. With CS expertise surrounding the ES types, the problems in this area became much more manageable. We found that such expertise could be distributed; in fact support programmers for SEQUOIA 2000 clients are often not at the same physical location as the client.

3) Interdisciplinary Research is fundamentally difficult.

I recall one endless discussion on the construction of a SEQUOIA 2000 benchmark that led eventually to the one in [STON93]. The CS types were arguing strongly for a representative abstract example of ES data access, i.e. the "specmark" of Earth Science. On the other hand, the ES types were equally adamant that the benchmark convey the exact data accesses of one of their real applications.

The ultimate result was the realization that the word "benchmark" means different things to CS and ES people. To ES people it means a "scenario", while to CS types, it means an abstract example. This vignette was typical of the struggle for each of two disciplines to understand the other. Fundamentally, this is a time consuming process, and ample interaction time should be planned for any project that must deal with multiple disciplines.

In SEQUOIA 2000, we made very effective use of "converters", i.e. a person of one discipline planted directly in the research group of the other discipline. This person, using the "informal interchange in the hall" communication system could serve as an interpreter and translator for the other discipline. Converters were encouraged by setting up a formal "exchange program", whereby central SEQUOIA 2000 resources would pay the living expenses of any exchange personnel.

4) Data base technology is a "big leap" for Earth Scientists.

Our initial plan was to simply drop data base technology into the project and the Earth Scientists would pick it up and use it. Unfortunately, they are strongly cultured in a "data is in files" mentality, and it was a very difficult transition for them to move to a data base view. Only after 3 years are the inherent advantages of DBMS technology beginning to sink in.

In addition, we followed the standard wisdom of appointing the most technically literate Earth Scientist as the leader of the data base design effort. This person chaired a committee of mostly CS types, charged with producing a schema.

This technique failed dismally. First, there was considerable disagreement among the CS types whether we were designing an interchange format, by which sites could reliably exchange data sets, (i.e. an on-the-wire representation) or a schema for stored data at a site. Most Earth Science standards (e.g. HDF, NetCDF) are of the first form, and there was substantial enthusiasm for simply picking one of these formats. On the other hand, other CS types argued that an on-the-wire representation mixes the data (e.g. a satellite image) and the meta data describing it (e.g. the frequency of the sensor, the date of the data collection and the name of the satellite) into a single highly encoded bit string. A better design would separate the two kinds of data and construct a good stored schema for it.

A second problem is the multitude of legacy formats currently in use and the desires by various Earth Scientists not to have to change the format they were currently using. This led to many arguments about the merits of one legacy format versus another, and usually led to the conclusion that both should be supported in addition to a neutral representation with clean semantics.

A third problem is that Earth Science data is fundamentally very complex. For example, Earth Scientists store geographic points, which are 3-D positions on the face of the Earth. There are some 20 popular projections of 3-D space onto 2-D. These include (latitude, longitude), Mercator projection and Lambert Equal Azimuthal projection. It is necessary to associate with every instance of a geographic point the projection system that is being used. Another problem is units. Some geographic data are represented as integers, with miles as the fundamental unit; other as floats with meters as the underlying unit. In addition, satellite imagery must be massaged in a variety of ways to "cook" it from raw data into a usable form. Cooking includes converting imagery from a one-dimensional stream of data recorded in satellite flight order into a two-dimensional representation. Many details of this cooking process must be recorded about all imagery. This blows up the meta data about imagery as well as forces the Earth Scientist to simply write down all the extra data elements.

Schema design turned out to be laborious and very difficult. Moreover, the Earth Scientists didn't understand data base design very well and hence were not well equipped to take on the extreme complexity of the task. As a result, we have reconstructed our data base design effort and put it in the hands of a 2 person team of CS types. Their mission is to produce a schema by interacting with the ES types, and being as dictatorial as necessary to get the job done.

5) Tertiary memory file systems are immature

At the beginning of the project, we were commited to an "all DEC" hardware environment, since they were the major project sponsor. However, there was no COTS tertiary memory file system that ran on DEC hardware and supported our storage devices. Rather than relax our hardware environment, we elected to write our own file system. As noted earlier, Inversion and Highlight were the result of this effort. However, neither worked with reliability the clients found suitable. Later we found two tertiary memory file systems that ran on DEC iron. Neither proved any more reliable than our "homebrew" solutions. Moreover, the importation of COTS software had the effect of stopping the development of our internal file systems, since the appropriate developers saw that their efforts were, at best, a backup strategy. Finally, we moved to running tertiary memory on non-DEC servers, and managed to get more reliable file systems. Unfortunately, even with this hardware relaxation, we have still not been able to find reliable software for our largest tape jukebox. Conversations with other users have largely confirmed our frustrating experience, and reliable tertiary memory software is hard to find.

In the future, we expect to test COTS software rigorously before we commit to it. Moreover, we would be more inclined to stick with homebrew software, since it is under our control. If it doesn't work, at least we can do something about it (i.e. try to fix it), rather than just plead with the COTS vendor.

6) Project management is a serious problem

SEQUOIA 2000 is a large project. At the last "all hands" meeting, there were about 110 people, broken down as 30 Computer Scientists, 40 Earth Scientists and 40 visitors from industry. There are multiple efforts on multiple campuses that must "plug and play". Keeping distributed development in synchronization is an extreme challenge. Furthermore, project management is not a skill fostered in a University environment. Additionally, it is not one that is rewarded in the University evaluation of faculty.

As such, the principal investigators viewed project management as a drain on their time that could be better invested in research activities. An obvious solution would be for SEQUOIA 2000 to hire a professional project manager. Unfortunately, it is impossible to pay a non-faculty person the going rates for such skilled persons. Another strategy which we attempted was to solicit a visitor from one of our industrial sponsors with the desired skill mix. Unfortunately, our efforts in this direction did not succeed. Hence, we were never able to recruit project management expertize to SEQUOIA 2000.

As a result project management was performed poorly at best. In any future large project, this component should be addressed satisfactorily up front by project personnel.

7) Multicampus projects are extremely painful

SEQUOIA 2000 work is ongoing in 7 different organizations within the University of California system. As such, there is a constant need to transfer money and people between these organizations. Moving either has proved unbelievably tedious within the University of California bureaucracy. It consumes a full time staff person to deal with these matters, and even then, tends to take forever. Moreover, it seems that SEQUOIA 2000 is constantly bending the personnel rules of the University.

As a result, multi-institution projects are extremely painful. There is a lot to be said for "put everybody in one place".

5. CONCLUSIONS

The SEQUOIA 2000 project plans a software distribution consisting of Footprint, Highlight, Inversion, POSTGRES, the AVS-POSTGRES bridge, the big lift, guaranteed delivery routing software, Lassen, and Tioga during late 1994. In addition, DEC and the University of California have agreed to form the cornerstone of a partnership to sponsor a second 3 year phase of SEQUOIA 2000 research and development. The exact goals of phase 2 are currently under investigation.

REFERENCES

[FERR90] Ferrari, D., "Client Requirements for Real-time Communication Services," IEEE Communications, November 1990.

[HELL93] Hellerstein, J. and Stonebraker, M., "Predicate Migration: Optimizing Queries with Expensive Predicates," Proc. 1993 ACM SIGMOD Conference on Management of Data, Washington, D.C., May 1993.

[HERO80] Herot, C., "SDMS: A Spatial Data Base System," ACM TODS, June 1980.

[KOHL93] Kohl, J. et. al., "Highlight: Using a Log-structured File System for Tertiary Storage Management," Proc. 1993 Winter USENIX Meeting, San Diego, Ca., Jan 1993.

[LARS91] Larson, R., "Classification, Clustering, Probabilistic Information Retrieval and the On-Line Catalog," Library Quarterly, April 1991.

[OLSO93] Olson, M., "The Design and Implementation of the Inversion File System," Proc. 1993 Winter USENIX Meeting, San Diego, Ca., Jan 1993.

[ROSE92] Rosenblum, M. and Ousterhout, J., "The Design and Implementation of a Log-structured File System," ACM TOCS, Feb. 1992.

[SARA93] Sarawagi, S. and Stonebraker, M., "Efficient Organization of Large Multidimensional Arrays," Proc. 1993 IEEE Data Engineering Conference, Houston, Tx., Feb 1993.

[SELT93] Seltzer, M. et. al., "An Implementation of a Log-structured File System for UNIX," Proc. 1993 Winter USENIX Meeting, San Diego, Ca., Jan 1993.

[STON90] Stonebraker, M. et al., "The Implementation of POSTGRES," IEEE TKDE, March 1990.

[STON91] Stonebraker, M. and Dozier, J., "Large Capacity Object Servers to Support Global Change Research," SEQUOIA 2000 Technical Report 91/1, Berkeley, Ca., July 1991.

[STON93] Stonebraker, M. et. al., "The SEQUOIA 2000 Benchmark," Proc. 1993 ACM SIGMOD Conference on Management of Data, Washington, D.C., May 1993.

[STON93B] Stonebraker, M. et. al., "Tioga: Providing Data Management for Scientific Visualization Applications," Proc. 1993 VLDB Conference, Dublin, Ireland, August 1993.

[WOOD94] Woodruff, A. et. al., "Zooming and Tunneling in Tioga: Supporting Navigation in Multidimensional Space," SEQUOIA 2000 Technical Report 94/48, Berkeley, Ca., March 1994.