Michael Stonebraker

An Overview of the Sequoia 2000 Project

The Sequoia 2000 project is the joint effort of computer scientists, earth scientists, government agencies, and industry partners to build a better computing environment for global change researchers. The objectives of this widely distributed project are to support high-performance I/O on terabyte data sets, to put all data in a database management system, and to provide improved visualization tools and high-speed networking. The participants developed a four-level architecture to meet these objectives. Chief among the lessons learned is that the Sequoia 2000 system must be considered an end-to-end solution, with all pieces of the architecture working together. This paper describes the Sequoia 2000 project and its implementation efforts during the first three years. The research was sponsored by Digital Equipment Corporation.

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. These researchers investigate issues such as global warming, ozone depletion, environment toxification, and species extinction and are members of earth science departments at universities and national laboratories. A more detailed conception for the project appears in the Sequoia 2000 technical report "Large Capacity Object Servers to Support Global Change Research."[1]

The participants in the Sequoia 2000 project are investigators of four types: (1) computer science researchers, (2) earth science researchers, (3) government agencies, and (4) industry partners.

Computer science researchers are responsible for building a prototype environment that better serves the needs of the target clients. Participating in the Sequoia 2000 project are investigators from the Computer Science Division at the University of California, Berkeley; the Computer Science Department at the University of California, San Diego; the School of Library and Information Studies at the University of California, Berkeley; and the San Diego Supercomputer Center.

Earth science researchers must explain their needs to the computer science investigators and use the resulting prototype environment to perform better earth science research. The Sequoia 2000 project comprises earth science investigators from the Department of Geography at the University of California, Santa Barbara; the Atmospheric Science Department at the University of California, Los Angeles (UCLA); the Climate Research Division at the Scripps Institution of Oceanography; and the Department of Earth, Air, and Water at the University of California, Davis.

To ensure that the resulting computer environment addresses the needs of the Sequoia 2000 clients, government agencies that are affected by global change matters participate in the project. The responsibility of these agencies is to steer Sequoia 2000 research toward achieving solutions to their problems. The government agencies that participate are the State of California Department of Water Resources (DWR),

the State of California Department of Forestry, the Coordinated Environment Research Laboratory (CERL) of the United States Army, the National Aeronautics and Space Administration (NASA), the National Oceanic and Atmospheric Administration (NOAA), and the United States Geologic Survey (USGS).

The task of the industry participants is to use the Sequoia 2000 technology and to offer guidance and research direction. In addition, they are a source of free or discounted computing equipment. Digital Equipment Corporation was the original industry partner. Recently, Epoch Systems, Hewlett-Packard, Hughes, Illustra, MCI, Metrum Systems, PictureTel, RSI, SAIC, Siemens, and TRW have become participants.

The purpose of this paper is to present the goals of the Sequoia 2000 project and to discuss how we achieved these goals and the results we accomplished during the first three years. The paper describes the architecture that we decided to pursue and the state of the software efforts in the various areas. The most important lesson we have learned is that the Sequoia 2000 system must be considered an end-to-end solution. Hence, clients can be satisfied only if all pieces of the architecture work together in a harmonious fashion. Also, many services required by the clients must be provided by every element of the architecture, each working with the others. We illustrate this end-to-end characteristic of Sequoia 2000 by discussing three issues that cross all parts of the system: guaranteed delivery, abstracts, and compression. We then indicate other specific lessons that we learned during the first three years of the project. The paper concludes with the current state of the project and its future directions.

The Sequoia 2000 Architecture

The Sequoia 2000 architecture is motivated by four fundamental computer science objectives:

1. *Support high-performance I/O on terabyte data sets.* The Sequoia 2000 clients are frustrated by current computing environments because they cannot effectively store the massive amounts of data desired for research purposes. The four academic clients plus DWR collectively want to be able to store approximately 100 terabytes of information, much of which is common data sets used by multiple investigators. These clients would like high-performance system software that would allow sharing of assorted tertiary memory devices. Unlike the I/O activities of most other scientific computing users, their activity involves primarily random access. For example, DWR is digitizing the agency's library of 500,000 slides and is putting it on-line using the Sequoia 2000 system. This data set has some locality of reference but will have considerable random activity.
2. *Put all data in a database management system (DBMS).* To maintain the metadata that describe their data sets and thus aid in the retrieval of information, the Sequoia 2000 clients want to move all their data to a DBMS. More important, using a DBMS will facilitate the sharing of information. Because a DBMS insists on a common schema for shared information, it will allow the researchers to define a schema. Then all researchers must use a common notation for shared data. Such a system will be a big improvement over the current situation where every data set exists in a unique format and must be converted by every researcher who wishes to use it.
3. *Provide improved visualization tools.* Sequoia 2000 clients use popular scientific visualization tools such as Explorer, Khoros, AVS, and IDL and are eager to use a next-generation toolkit.
4. *Provide high-speed networking.* Sequoia 2000 clients realize that a 100-terabyte storage server (or 100-terabyte servers) will not be located on each of their desktops. Moreover, the storage is likely to be located at the other end of a wide area network (WAN), far from their client machines. Since the clients' visualization scenarios invariably involve animation, for example, showing the last 10 years of the ozone hole by playing time forward, the clients require ultrahigh-speed networking to move sequences of images from a server machine to a client machine.

To meet these objectives, we adopted the four-level architecture illustrated in Figure 1. The architecture comprises the footprint layer, the file system layer, the DBMS layer, and the application layer. This section discusses our efforts at each of the levels and then concludes with a discussion of the Sequoia 2000 networking that connects the elements of the architecture.

The Footprint Layer

The footprint layer is a software system that shields higher-level software, such as file systems, from device-specific characteristics of robotic devices. These characteristics include specific robot commands, block sizes, and media-specific issues. The footprint layer can be thought of as a common robot device driver. A footprint implementation exists for each of the four tertiary memory devices used by the project, namely, a Sony write once, read many (WORM) optical disk jukebox, an HP rewritable optical disk jukebox, a Metrum VHS tape jukebox, and an Exabyte 8-millimeter tape jukebox. Collectively, these four devices and the CPUs and disk storage systems in front of them were named Bigfoot, after the legendary, very tall recluse spotted occasionally in the Pacific Northwest.

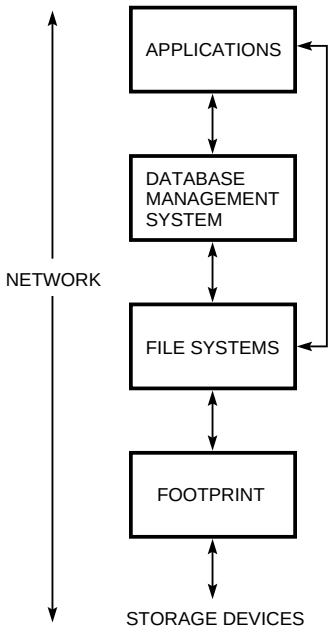


Figure 1
The Sequoia 2000 Architecture

The File System Layer

On top of the footprint layer is the file system layer. Two file systems manage data in the Bigfoot multilevel memory hierarchy. The first file system is Highlight, which extends the Log-structured File System (LFS) pioneered for disk devices by Ousterhout and Rosenblum to tertiary memory.[2,3] The original LFS treats a disk device as a single continuous log onto which newly written disk blocks are appended. Blocks are never overwritten, so a disk device can always be written sequentially. Hence, the LFS turns a random-write environment into a sequential-write environment. In particular problem areas, this may lead to much higher performance. Benchmark data support this conclusion.[4] In addition, the LFS can always identify the last few blocks that were written prior to a file system failure by finding the end of the log at recovery time. File system repair is then very fast, because potentially damaged blocks are easily found. This approach differs from conventional file system repair, where a laborious check of the disk must be performed to ascertain disk integrity.

Highlight extends the LFS to support tertiary memory by adding a second log-structured file system on top of the footprint layer. This file system also writes tertiary memory blocks sequentially, thereby obtaining the performance characteristics of the LFS. The Highlight file system adds migration and bookkeeping code that treats the disk LFS file system as a cache for the tertiary memory file system. In summary, Highlight should provide good performance for workloads that consist of mainly write operations. Since Sequoia 2000 clients want to archive vast amounts of data, the Highlight file system has the potential for good performance in the Sequoia 2000 environment.

The second file system is Inversion.[5] Most DBMSs, including the one used for the Sequoia 2000 project, support binary large objects (BLOBs), which are arbitrary-length byte strings of variable length. Like several commercial systems, Sequoia's data manager POSTGRES stores large objects in a customized storage system directly on a raw storage device.[6] As a result, it is a straightforward exercise to support conventional files on top of DBMS large objects. In this way, the front end turns every read or write operation into a query or an 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 addition, 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 simulating files on top of a DBMS is poor performance. As reported by Olson, Inversion performance is exceedingly good when large blocks of data are read and written, as is characteristic of the Sequoia 2000 workload.[5]

At the present time, Highlight is operational but very buggy. Inversion, on the other hand, is used to manage production data on Sequoia's Sony WORM jukebox. Unfortunately, the reliability of the prototype system has not met user expectations. Sequoia 2000 clients have a strong desire for commercial off-the-shelf (COTS) software and are frustrated by documentation glitches, bugs, and crashes.

As a result, the Sequoia 2000 project team has also deployed two commercial file systems, Epoch and AMASS. The Epoch file system is quite reliable but does not support either of Sequoia's large-capacity robots. Hence, it is used heavily but only for small data sets. The AMASS file system is just coming into production use for Sequoia's Metrum robot and replaces an earlier COTS system, which was unreliable. Given the experience of the Sequoia 2000 team with tertiary memory support, tertiary memory users should carefully test all file system software.

The DBMS Layer

To meet Sequoia 2000 client needs, a DBMS must support spatial data such as points, lines, and polygons. In addition, the DBMS must support the large spatial arrays in which satellite imagery is naturally stored. These characteristics are not met by popular, general-purpose relational and object-oriented DBMSs.[7] The best fit to client needs is a special-purpose Geographic Information System (GIS) or a next-generation object-relational DBMS. Since it has one such object-relational system, namely

POSTGRES, the Sequoia 2000 project elected to focus its DBMS efforts on this system.

To make the POSTGRES DBMS suitable for Sequoia 2000 use, we require a schema for all Sequoia data. This database design process has evolved as a cooperative exercise between various database experts at Berkeley, the San Diego Supercomputer Center, CERL, and SAIC. The Sequoia schema is the collection of metadata that describes 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
- 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 "The Sequoia 2000 Benchmark."[7]

Raster data constitute the bulk of the 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.

We have tuned the POSTGRES DBMS to meet the needs of the Sequoia 2000 clients. The interface to POSTGRES arrays has been improved, and a novel chunking strategy is now operational.[8] Instead of storing an array by ordering the array indexes from fastest changing to slowest changing, this system chooses a stride for each dimension and stores chunks 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.

Sequoia 2000 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 percent snow and that were observed subsequent to June 1992. In SQL, this query is expressed as follows:

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

The first clause in the predicate requires the CPU to evaluate millions of instructions, whereas the second clause requires only a few hundred instructions. The DBMS must be cognizant of the CPU cost of clauses when constructing a query plan, a cost component that has been ignored by most previous optimization work. We have extended the POSTGRES optimizer to deal intelligently with expensive functions.[9]

It is highly desirable to allow popular expensive functions to be precomputed. In this way, the CPU need only evaluate each such function once, rather than once for each query in which the function appears. Our approach to this issue is to allow databases to contain indexes on a function of the data and not on just the data object itself. Hence, the database administrator can specify that a B-tree index be built for the function AREA (SNOW(content)). Areas of images are arranged in sort order in a B-tree, so the first clause in the above query is now very inexpensive to compute. Using this technique, the function is computed at data entry or data update time and not at query evaluation time. A consequence of function indexing is that inserting a new image into the database may be very time-consuming, since function computation is now included in the load transaction. To deal with the undesirable lengthy response times for some loads, we have also explored lazy indexing and partial indexing. Thus, index building does not need to be synchronous with data loading.

The feedback from the Sequoia 2000 clients regarding POSTGRES is that it is not reliable enough to serve as a base for production work. Moreover, the documentation is inadequate, and no facility exists to train users. Our users want a COTS product and not a research prototype. Consequently, the Sequoia 2000 project has migrated to the commercial version of POSTGRES, namely the Illustra system, to obtain a COTS DBMS product. Migration to this system required reloading all project data, a task that is now nearly complete.

The Application Layer

The application layer of the Sequoia 2000 architecture contains five elements:

1. An off-the-shelf visualization tool
2. A visualization environment

3. A browsing capability for textual information
4. A facility to interface the UCLA General Circulation Model (GCM) to the POSTGRES/Illustra system
5. A desktop videoconferencing or "picturephone" facility

For the off-the-shelf visualization tool, we have converged around the use of AVS and IDL for project activities. AVS has an easy-to-use "boxes-and-arrows" user interface, whereas IDL has a more conventional linear programming notation. On the other hand, IDL has better two-dimensional (2-D) graphics features. Both AVS and IDL allow the user to read and write file data. To connect to the DBMS, we have written an AVS–POSTGRES bridge. This program allows the user to construct an ad hoc POSTGRES query and pipe the result into an AVS boxes-and-arrows network. Sequoia 2000 clients can use AVS for further processing on any data retrieved from the DBMS. IDL is being interfaced to AVS by the vendor. Consequently, data retrieved from the database can be moved into IDL using AVS as an intermediary. Now that we have migrated to the Illustra DBMS, we are considering porting this AVS bridge to the Illustra application programming interface (API).

AVS has some disadvantages as a visualization tool for Sequoia 2000 clients. First, its type system, which is different from the POSTGRES/Illustra type system, has no direct knowledge of the common Sequoia 2000 schema. In addition, AVS consumes significant amounts of main memory. Architecturally, AVS depends on virtual memory to pass results between various boxes. It also maintains the output of each box in virtual memory for the duration of an execution session. The user can thus 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 generally produce very large intermediate results, consume large amounts of both virtual and real memory. In fact, clients report that 64 megabytes of real memory on a workstation is often not enough to enable serious AVS use. Furthermore, AVS does not support zooming in to investigate 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 that is too primitive for many Sequoia 2000 clients.

Consequently, we have designed two new visualization environments. The first system, called Tecate, is being built at the San Diego Supercomputer Center. The Tecate infrastructure enables the creation of applications that allow end users to browse for and visualize data from networked data sources. This software platform capitalizes on the architectural strengths of current scientific visualization systems, network browsers, database management system front ends, and virtual reality systems, as discussed in a companion paper in this issue of the *Journal*.[10]

The other system, Tioga, is a boxes-and-arrows programming environment that is DBMS-centric, i.e., the environment type system is the same as the DBMS type system. The Tioga user interface gives the user a flight simulator paradigm for browsing the output of a network. In this way, the visualizer can navigate around data and then zoom in to obtain additional data on items of particular interest. The preliminary Tioga design was presented at the 1993 Very Large Databases Conference.[11] A first prototype, described by Woodruff, is currently running.[12]

A commercial version of the Tioga environment has also been implemented by Illustra. The Sequoia 2000 project is making considerable use of this tool, which is named Object-Knowledge. Early user experience with both Tioga and Object-Knowledge indicates that these systems are not easy to use. We are now exploring ways to improve the Tioga system. The objective is to build a system that a scientist with minimal training in the environment can use without a reference manual.

The third element of the application layer is a browsing capability for textual information of interest to our clients. This capability is a cornerstone of the Sequoia 2000 architecture. Initially, we converted a stand-alone text retrieval system called Lassen to our DBMS-centric view. The first part of the Lassen system is a facility for constructing weighted keyword indexes for the words in a POSTGRES document. This indexing system, Cheshire, builds on the pioneering work of the Cornell Smart system and operates as the action part of a POSTGRES rule, which is triggered on each document insertion, update, or removal.[1,13] The second part of the Lassen system is a front-end query tool that understands natural language. This tool allows a user to request all documents that satisfy a collection of keywords by using a natural language interface. The Lassen system has been operational for more than a year, and retrievals can be requested against the currently loaded collection of Sequoia 2000 documents.

In addition, we have moved Lassen to Z39.50, a popular protocol oriented toward information interchange and information retrieval.[14] The client portion of Lassen has been changed to emit Z39.50, and we have written 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 the Cheshire system.

With our move to the Illustra DBMS, we have converted the client side of Lassen to work with Illustra.

Illustra has an integrated document data type with capabilities similar to the extensions we made to POSTGRES.

A related Berkeley project is focused on digitizing all the Berkeley Computer Science Technical Reports. This project uses a Mosaic client to access a custom World Wide Web server called Dienst, which stores technical report objects in a UNIX file system. In a few months, we expect to convert Dienst to store objects in the Sequoia 2000 database, rather than in files. When this system, nicknamed Database Dienst, is operational, Mosaic/Dienst service will be available for all textual objects in the Sequoia schema.

Our fourth thrust in the application layer is a facility to interface the UCLA General Circulation Model (GCM) to the POSTGRES/Illustra system. This program is a "data pump" because it pumps data out of the simulation model and into the DBMS. We named the program "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 (3-D) 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 from 0.1 to 10.0 megabytes per second (MB/s) output. The purpose of the big lift is to install the output data into a database in real time. UCLA scientists can then use ObjectKnowledge, Tioga, Tecate, AVS, or IDL to visualize their simulation output. The big lift will likely 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 fifth application system is a conferencing system. Since Sequoia 2000 is a distributed project, we learned early that face-to-face meetings that required participants to travel to other sites and electronic mail were not sufficient to keep project members working as a team. Consequently, we purchased conference room videoconferencing equipment for each project site. This technology costs approximately $50,000 per site and allows multiway videoconferences over integrated services digital network (ISDN) lines.

Although the conference room equipment has helped project communication immensely, it must be set up and taken down at each use because the rooms it occupies at the various sites are normally used as classrooms. Therefore, videoconferencing 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 videoconferencing. A video compression board, a microphone, speakers, a network connection, a video camera, and the appropriate software can turn a conventional workstation into a desktop videoconferencing facility. In addition, video can be easily transmitted over the network interface that is 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 migrating most of our videoconferencing activities to desktop technology. This effort, which is called Hollywood, strives to further improve the ability of Sequoia 2000 researchers to communicate.

Note that the Sequoia 2000 researchers do not need groupware, i.e., the ability to have common windows on multiple client machines separated by aWAN, in which common code can be run, updated, and inspected. Rather, our researchers need a way to hold impromptu discussions on project business. They want a low-cost multicast picturephone capability, and our desktop videoconferencing efforts are focused in this direction.

Sequoia 2000 Networking

The last topic of this section on the Sequoia 2000 architecture is the networking agenda. Regarding 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 of the Sequoia 2000 architecture assumes a local UNIX socket connection or a local area network (LAN) or WAN connection using the transmission control protocol/internet protocol (TCP/IP). Actual connections among Sequoia 2000 sites use either the Internet or a dedicated T3 network, which the University of California provides as part of its contribution to the project.

The networking team judged Digital's Alpha processors to be fast enough to route T3 packets. Hence, the project uses conventional workstations as routers; custom machines are not required. Furthermore, the Sequoia 2000 network 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 sends information at a rate that does not exceed the rate specified in the contract. These algorithms, which are based on the work of Ferrari, require a setup phase for a connection that allocates bandwidth on all the lines and in all the switches.[15]

Lastly, the network researchers are concerned that the Digital UNIX (formerly DEC OSF/1) operating system copies every byte four times in between retrieving it from the disk and sending it out over a network connection. The efficient integration of networking services into the operating system is the topic of a companion paper by Pasquale et al. in this issue.[16]

Sequoia 2000 as an End-to-End Problem

The major lesson we have learned from the Sequoia 2000 project is that many issues facing our clients cannot be isolated to a single layer of the Sequoia 2000 architecture. This section describes three such end-to-end problems: guaranteed delivery, abstracts, and compression.

Guaranteed Delivery

Clearly, guaranteed delivery must be an end-to-end contract. Suppose a Sequoia 2000 client wishes to visualize a specific computation; for example, the client wants to observe Hurricane Andrew as it moves from the Bahamas to Florida to Louisiana. Specifically, the client wishes to visualize appropriate satellite imagery at a resolution of 500 × 500 in 8-bit color at 10 frames per second. Hence, the client requires 2.5 MB/s of bandwidth to his screen. The following scenario might be the computation steps that take place.

The DBMS must run a query to fetch the satellite imagery. The query might require returning a 16-bit data value for each pixel that will ultimately appear on the screen. The DBMS must therefore agree to execute the query in such a way that it guarantees output at a rate of 5.0 MB/s.

The storage system at the server will fetch some number of I/O blocks from secondary and/or tertiary memory. DBMS query optimizers can accurately guess how many blocks they need to read to satisfy the query. The DBMS can then easily generate a guaranteed delivery contract that the storage manager must satisfy, thus allowing the DBMS to satisfy its contract.

The network must agree to deliver 5.0 MB/s over the network link that connects the client to the server. The Sequoia 2000 network software expects exactly this type 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 MB/s.

In short, guaranteed delivery is a collection of contracts that must be adhered to by the DBMS, the storage system, the network, and the visualization package. One approach to architecting these contracts was presented at the 1993 Very Large Databases Conference.[11]

Abstracts

One aspect of the Sequoia 2000 visualization process is the necessity of abstracts. Consider the Hurricane Andrew example. The client might initially want to browse the hurricane at 100 × 100 resolution. Then, on finding something of interest, the client would probably like to zoom in and increase the resolution, usually to the maximum available in the original data. This ability to dynamically change the amount of resolution in an image is supported by abstracts.

Note that providing abstracts is a much more powerful construct than merely providing for 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 an early DBMS visualization system called the Spatial Data Management System (SDMS).[17]

Sequoia 2000 clients wish to have abstracts; however, it is clear that they can be managed by the visualization tool, the DBMS, the network, or the file system. In the former case, abstracts are defined for boxes-and-arrows networks.[11] In the DBMS, abstracts would be defined for individual data elements or for data classes. If the network manages abstracts, 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) is available. In the file system, abstracts would be defined for files. Sequoia 2000 researchers are 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.

Compression

The Sequoia 2000 clients are adamant on the issue of compression—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 future phenomena. Since it is not possible to predict what these phenomena might be or where they might occur, the Sequoia 2000 scientists want access to all data at full resolution.

Some Sequoia 2000 data cannot be compressed economically and should be stored in uncompressed form. The inclusion of abstracts offers a mechanism to lower the bandwidth required between the storage device and the visualization program. No saving of tertiary memory through compression is available for such data.

Other data ought to be stored in compressed form. The question of when compression and decompression should occur can be handled by using a just-in-time decompression strategy. For example, if the storage manager compresses data as they are written and then decompresses them on a read operation, the network manager may then recompress the data for transmission over a WAN to a remote site where they will be decompressed a second time. Obviously, data should be moved in compressed form and decompressed only when necessary. In general, decompression will occur 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. If an application resides on the same machine as the storage manager, the file system must be in charge of decompressing the data. All software modules in the Sequoia 2000 architecture must cooperate to perform just-in-time decompression and as-early-as-possible compression. Like guaranteed delivery, compression is a task that requires all software modules to cooperate.

Specific Lessons Learned

In addition to the end-to-end issues, we learned other lessons from the first three years of the Sequoia 2000 experience, as discussed in this section.

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

We learned early in the project that electronic mail and travel between sites would not result in the desired degree of cooperation from geographically dispersed researchers from different disciplines. Consequently, we made a significant investment in infrastructure. This included meetings for all the Sequoia 2000 participants, which are now held twice a year, and videoconferencing equipment at each site. Through this video link, project members interact by holding a weekly distributed seminar, semimonthly operations committee meetings, occasional steering committee meetings, and meetings between researchers with common interests. The video quality of the project's current videoconferencing equipment is not high, and to achieve success when participants are located far apart, specially trained individuals must operate the equipment. Nevertheless, the equipment has proven to be valuable in generating cohesion in the dispersed project. We have installed desktop videoconferencing systems on 30 Sequoia 2000 workstations and expect to replace our current conference room equipment with next-generation desktop technology.

In addition, we conducted a learning experiment in which a course taught by one of the Sequoia 2000 faculty members at the Santa Barbara campus was broadcast over our videoconferencing equipment to four other sites. Students could take the course for credit at their respective campuses. Of course, the overhead of setting up such a course was substantial. A new course had to be added at each campus, and every step in the approval process required briefings on the fact that the instructor was from a different campus and on the way everything was going to work. This experiment was popular, and students have requested additional courses taught in this manner.

On the other hand, we also tried to run a computer science colloquium using this technology. We broadcast from various sites to six computer science departments around the U.S. Initial student interest was high because of the lineup of eminent speakers. Such speakers could be recruited easily, because they only had to locate the nearest compatible equipment and then get to that site. No air travel was required. The experiment failed, however, because attendance decreased throughout the semester and ended at an extremely low level.

The basic problem was that, typically, speakers were not skilled in using the medium—they would put too much information on slides and then flip though the slides before remote sites could get a complete transmission. Also, the question-and-answer period could not be very interactive because of the many sites involved. The experiment ended after one semester and will not be repeated.

Lesson 2: There was often a mismatch between the expectations of the earth scientists and those of the computer scientists.

The computer scientists on the Sequoia 2000 team wanted access to knowledgeable application specialists who could describe their problems in terms understandable to the computer scientist. The computer scientists then wanted to think through elegant solutions, verify with the earth scientists that the solutions were appropriate, and then prototype the results. The earth scientists wanted final COTS solutions to their problems; they were unsympathetic about poor documentation, bugs, and crashes.

With considerable effort, the expectations are converging. The ultimate solution is to move to COTS software modules as they become available for portions of the system and augment the modules with in-house prototype code.

We have found that the best way to make forward progress was to ensure that each earth science group using Sequoia 2000 prototype code had one or more sophisticated staff programmers who could deal successfully with the quirks of prototype code. With computer science expertise surrounding the earth scientists, the problems in this area became much more manageable. We also discovered that we could distribute such expertise. In fact, support programmers for Sequoia 2000 clients are often not at the same physical location as the client.

Lesson 3: Interdisciplinary research is fundamentally difficult.

One lengthy discussion on the construction of a Sequoia 2000 benchmark eventually led to the discussion presented in the 1993 ACM SIGMOD conference paper entitled "The Sequoia 2000 Benchmark," which we referred to previously.[7] The computer science researchers were arguing strongly for a representative abstract example of earth science data access, i.e., the "specmark of earth science." On the other hand, the earth scientists were equally adamant that the benchmark convey the exact data accesses.

Finally, the computer scientists and the earth scientists realized that the word "benchmark" has a different meaning for each of the two groups of researchers. To earth scientists, a benchmark is a scenario, whereas to computer scientists, a benchmark is an abstract example. This vignette was typical of the experience these two disciplines had trying to understand one another. Fundamentally, this process is time-consuming, and ample interaction time should be planned for any project that must deal with multiple disciplines.

The Sequoia 2000 project participants made effective use of "converters." A converter is a person of one discipline who is planted directly in the research group of another discipline. Through informal communication, this person serves as an interpreter and translator for the other discipline. Converters are encouraged by the existence of a formal exchange program, whereby central Sequoia 2000 resources pay the living expenses of the exchange personnel.

Lesson 4: Database technology is a major advance for earth scientists.

Our initial plan was to introduce database technology into the project with the expectation that the earth scientists would pick it up and use it. Unfortunately, they are accustomed to data being in files and found it very difficult to make the transition to a database view. The earth scientists are becoming increasingly aware of the inherent advantages of DBMS technology.

In addition, we appointed the earth scientist with the most computer science knowledge as leader of the database design effort. This person chaired a committee of mainly computer scientists who were charged with producing a schema.

This technique failed for several reasons. First, the computer scientists disagreed about 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, such as the Hierarchical Data Format (HDF) and the network Common Data Form (netCDF), are of the first form, and there was substantial enthusiasm for simply choosing one of these formats.[18,19] On the other hand, some computer scientists argued that an on-the-wire representation mixes the data (e.g., a satellite image) and the metadata that describe 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 was that numerous legacy formats are currently in use, and some earth scientists did not want to change the formats they were using. This led to many arguments about the merits of one legacy format over another, which in turn caused the opposing sides to conclude that both formats under discussion should be supported in addition to a neutral representation.

A third problem was that earth science data are fundamentally quite complex. For example, earth scientists store geographic points, which are 3-D positions on the earth's surface. There are approximately 20 popular projections of 3-D space onto 2-D space, including (latitude, longitude), Mercator projection, and Lambert Equal Azimuthal projection. With every instance of a geographic point, it is necessary to associate the projection system that is being used. Another data problem is related to units. Some geographic data are represented as integers, with miles as the fundamental unit; other data are represented as floating-point numbers, 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 2-D representation. Many details of this cooking process must be recorded for all imagery. This dramatically expands the metadata about imagery as well as forces the earth scientist to write down all the extra data elements.

Schema design turned out to be laborious and very difficult. The earth scientists did not have a good understanding of database design and thus were not prepared to take on the extreme complexity of the task. As a result, we have reconstructed our database design effort. Now, two computer scientists are responsible for producing a schema. They interact with the earth scientists when such action helps to accomplish the task.

Lesson 5: Project management is a substantial problem.

Sequoia 2000 is a large project. About 110 people attended the last general meeting. The attendees included approximately 30 computer scientists, 40 earth scientists, and 40 visitors from industry. Multiple efforts on multiple campuses must "plug and play." Synchronizing distributed development is an extreme challenge. Furthermore, the skill of project management is not fostered in a university environment, nor is it rewarded in a university faculty evaluation.

The principal investigators viewed the time spent on project management as time that could be better invested in research activities. An obvious solution would be for the Sequoia 2000 project to hire a professional project manager. Unfortunately, it is impossible to pay a nonfaculty person the market rates normally received by such skilled persons. One strategy we attempted to use was to solicit a visitor with the desired skill mix from one of our industrial sponsors. Our efforts in this direction failed, and we were never able to recruit project management expertise for

the Sequoia 2000 effort. 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.

Lesson 6: Multicampus projects are extremely difficult to implement.

Sequoia 2000 work is taking place in seven different organizations within the University of California educational system. There is a constant need to transfer money and people among these organizations. Accomplishing such moves is a difficult and slow process, however, because of the bureaucracy within the system. In addition, the personnel rules of the University are often in conflict with the needs of the Sequoia 2000 project. As a result, multi-institution projects, where participants are in different and often distant locations, are extremely difficult to carry out.

Status and Future Plans

The Sequoia 2000 project is more than three years old and has nearly accomplished its objectives. We have a common schema in place for all Santa Barbara and UCLA data, and all participants have agreed to use the schema. This schema serves as leverage for the standards efforts under way in the spatial arena.[20] The infrastructure is in place to enable this schema to evolve as more data types, user-defined functions, and operators are included in the future.

The combination of Object-Knowledge, Illustra, Epoch, and AMASS is proving robust and meets our clients' needs. Lastly, we have ample resources to move our prototype into production use at UCLA and Santa Barbara during the next several months.

We are also extending the scope of the prototype in two different directions. First, we will recruit additional earth scientists to utilize our system. This will require extending our common schema to meet their needs and then installing our suite of software at their site. We expect to recruit two to three new groups during the next year.

Second, a companion project, the Electronic Repository, has as one of its objectives to use the Sequoia 2000 technology to support an environmental digital library of aerial photography, polygonal data, and text for the Resources Agency of the State of California.[21] This electronic library project is extending the reach of Sequoia 2000 technology from earth scientists toward a broader community.

Our research activities are also very active. As noted earlier, we are continuing our visualization activities and anticipate an improved Tioga system. The Sequoia 2000 clients have made it clear that they want seamless access to distributed data, and we have evolved POSTGRES to a wide-area distributed DBMS that makes decisions based on an economic paradigm. This system is called Mariposa.[22] In our COTS system, a bad impedance mismatch exists between the DBMS and the tertiary memory file systems. We have therefore shifted our research focus to constructing an intelligent mass storage interface that properly supports a DBMS.

Finally, the Sequoia 2000 network currently supports service guarantees, but there is no economic framework in which to place multiple levels of service. As a result, our networking research is focused on construction of this type of framework.

We anticipate a robust production environment for earth science researchers by the end of 1995. In addition, we expect to continue to improve the Sequoia 2000 environment with future research results in the above areas.

References and Notes

1. M. Stonebraker and J. Dozier, "Large Capacity Object Servers to Support Global Change Research," Sequoia 2000 Technical Report 91/1, Berkeley, California (July 1991).
2. J. Kohl et al., "Highlight: Using a Log-structured File System for Tertiary Storage Management," *Proceedings of the 1993 Winter USENIX Meeting,* San Diego, California (January 1993).
3. M. Rosenblum and J. Ousterhout, "The Design and Implementation of a Log-structured File System," *ACM Transactions on Computing Systems (TOCS)* (February 1992).
4. M. Seltzer et al., "An Implementation of a Log-structured File System for UNIX," *Proceedings of the 1993 Winter USENIX Meeting,* San Diego, California (January 1993).
5. M. Olson, "The Design and Implementation of the Inversion File System," *Proceedings of the 1993 Winter USENIX Meeting,* San Diego, California (January 1993).
6. M. Stonebraker et al., "The Implementation of POSTGRES," *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (March 1990).
7. M. Stonebraker et al., "The Sequoia 2000 Benchmark," *Proceedings of the 1993 ACM SIGMOD Conference,* Washington, D.C. (May 1993).
8. S. Sarawagi and M. Stonebraker, "Efficient Organization of Large Multidimensional Arrays," *Proceedings of the 1993 IEEE Data Engineering Conference,* Houston, Texas (February 1993).
9. J. Hellerstein and M. Stonebraker, "Predicate Migration: Optimizing Queries with Expensive Predicates," *Proceedings of the 1993 ACM SIGMOD Conference,* Washington, D.C. (May 1993).

10. P. Kochevar and L. Wanger, "Tecate: A Software Platform for Browsing and Visualizing Data from Networked Data Sources," *Digital Technical Journal,* vol. 7, no. 3 (1995, this issue): 66–83.

11. M. Stonebraker et al., "Tioga: Providing Data Management for Scientific Visualization Applications," *Proceedings of the 1993 VLDB Conference,* Dublin, Ireland (August 1993).

12. A. Woodruff et al., "Zooming and Tunneling in Tioga: Supporting Navigation in Multidimensional Space," Sequoia 2000 Technical Report 94/48, Berkeley, California (March 1994).

13. R. Larson, "Classification, Clustering, Probabilistic Information Retrieval and the On-Line Catalog," *Library Quarterly* (April 1991).

14. *Information Retrieval Application Service Definition and Protocol Specification for Open Systems Interconnection,* ANSI/NISO Z39.50-1992 (revision and redesignation of ANSI Z39.50-1988) (New York: American National Standards Institute/National Information Standards Organization, 1992).

15. D. Ferrari, "Client Requirements for Real-time Communication Services," *IEEE Communications* (November 1990).

16. J. Pasquale et al., "High-performance I/O and Networking Software in Sequoia 2000," *Digital Technical Journal,* vol. 7, no. 3 (1995, this issue): 84–96.

17. C. Herot, "SDMS: A Spatial Data Base System," *ACM Transactions on Computing Systems (TOCS)* (June 1980).

18. The National Center for Supercomputing Applications (NCSA) at the University of Illinois developed the Hierarchical Data Format (HDF) as a multiobject file format.

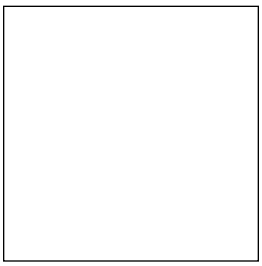
19. Network Common Data Form (netCDF) is an interface for scientific data access and a freely distributed software library that provides an implementation of the interface. netCDF was developed by Glenn Davis, Russ Rew, and Steve Emmerson at the Unidata Program Center in Boulder, Colorado. The netCDF library defines a machine-independent format for representing scientific data. Together, the interface, the library, and the format support the creation, access, and sharing of scientific data.

20. J. Anderson and M. Stonebraker, "Sequoia 2000 Metadata Schema for Satellite Images," *SIGMOD Record,* Vol. 23, No. 4 (December 1994).

21. R. Larson et al., "The Sequoia 2000 Electronic Repository," *Digital Technical Journal,* vol. 7, no. 3 (1995, this issue): 50–65.

22. M. Stonebraker et al., "An Economic Paradigm for Query Processing and Data Migration in Mariposa," *Proceedings of IEEE Parallel and Distributed Information Systems Conference,* Austin, Texas (September 1994).

Biography



Michael Stonebraker
Michael Stonebraker is a professor of electrical engineering and computer science at the University of California, Berkeley, where he has been employed since 1971. He was one of the principal architects of the INGRES relational database management system and subsequently constructed Distributed INGRES. For the last six years, Michael has been developing POSTGRES, a next-generation DBMS that can manage objects and rules, as well as data. Michael is a founder of INGRES Corporation, the founder of Illustra Information Technologies, a past chairman of ACM SIGMOD, and the author of many papers on DBMS technology. He lectures widely and was the winner of the first ACM SIGMOD innovations award in 1992.