Computer Technology Forecast for Virtual Observatories

Extended Abstract of talk at
Astronomy Virtual Observatories of the Future at
California Institute of Technology, Pasadena, CA, July 2000

Jim Gray

September 2000

Technical Report

MSR-TR-2000-102

Microsoft Research
Microsoft Corporation
301 Howard Street, #830
San Francisco, CA, 94105

Computer Technology Forecast for Virtual Observatories

Extended Abstract of talk at Astronomy Virtual Observatories of the Future, California Institute of Technology, Pasadena, CA, July 2000

Jim Gray
Microsoft Research,
Gray@Microsoft.com or http://research.microsoft.com/~Gray

I was asked, as a computer scientist, to give a sense of what computer technologies the VOF can design for over the next decade. In designing the VOF we need to think in terms of how much storage, bandwidth, and processing power we will have five and ten years from now – because that is the equipment we will actually be using.

The good news is that processing, storage, and networking are improving at an exponential pace. In the limit, they are infinitely fast and capacious, and cost nothing. Unfortunately, most of us do not live in the limit; we live in the present so computers will not be free anytime soon. The bad news is that people are getting more expensive, and indeed the management and programming costs routinely exceed hardware costs.

If everything gets better at the same rate, then noting really changes, but some ratios are changing. The people:computer ratio is changing so that we now try to minimize people costs. Although computers are getting faster, the speed of light is not changing (so if you want to get data from the other coast, you must wait 60 milliseconds (request-response round-trip time)). In addition, WAN costs have not improved much in the last decade. This is a surprise: the cost of WAN networking is much reduced. But, it seems costs and prices are not correlated in the telecommunications industry. Perhaps this will change in the future, but if you live in the present, you want to put the computation "close" to the data to minimize costs – rather than moving huge datasets to the computation. There is a recurring theme in this presentation: processing, storage, and bandwidth are all getting exponentially faster, cheaper, and bigger, but access times are not improving nearly as much, if at all. So the process:access ratios are changing, and forcing us to change our processing strategy by making things more sequential, and by using both pipeline and partition parallelism.

Astronomy is in a paradoxical state, there is not exponential growth in glass, but there is exponential growth in CCDs. So astronomy seems to be tracking Moore's law, doubling data volumes every two years as more and better CCDs are put behind the glass and as new instruments come online.

Computing involves progress in many areas, data visualization, human-computer interfaces (vision, speech, robotics), mobile applications (wireless and batteries), micro-electron-mechanical-systems (MEMS), and the more prosaic areas of networking, processing, and data storage and access. This talk focuses primarily on these classic areas (processing, storage, networking). Even within these classic areas, I am going to go light on topics such as object-oriented programming, databases, data visualization, data mining, and the open source movement. Rather in the 20 minutes I have, I want to just quickly give you a sense of the base architecture issues of server-side processors, memory, storage, and networking. I also want to covey the essential role that parallelism must play in our designs, and point out that we need to place processing next to the data. This may mean that we replicate data near the processors, and it argues against a design that has processors in one place and data stores in another.

How much information is there, and how much astronomy information is there?

To, start, lets talk about how much information there is – and how astronomers fit into the scheme of things. It appears that all the astronomy data adds up to a few petabytes today. Large archives are tens of terabytes (TB). The Sloan will be 40 TB raw and 3 TB cooked – but that is five years off. The radio telescopes generate a few terabytes per day, but at present we are not saving most of that data.

How does this relate to the rest of the world? The public Internet is about 30 TB toady (Sept 2000). The Library of Congress in ASCII is about 20 TB (20 million books at 1MB/book). If you include all the film, all the music, all the photos, and images of all the books, then LC is a few tens of Petabytes. Each year the

disk industry builds a few exabytes of disk. Mike Lesk (following Alan Newell) points out that ALL the recorded information is a few tens of exabytes. He points out that we can now record everything on disk or tape, and that the precious resource is human attention. People will never see most data. Computers will read, analyze, and summarize this data for us. Human attention is the scarce resource in this new world [Lesk].

So, to put things in perspective, astronomy data sets are huge, but not so huge that they are impossible. They are Wal-Mart size, not impossible size. They come in units of 10TB and sum to a few PB (petabytes). They can all be put online (see below) for a few million dollars as a federated virtual observatory. Each member of the federation can contribute its part to the overall corpus. Some members of the federation must serve the unifying role of coordinating access, much as Yahoo! does for the current Internet.

I think astronomy data is the best playpen for us computer scientists to experiment with data analysis and summarization techniques. You astronomers are unique: You share your data. That is surprisingly rare (for a variety of reasons data in physics, in geology, in medicine, in finance, and in the Internet is encumbered by privacy and by commercial interests). Your data has rich correlations, many of which you have discovered. And you have so much data, and it has such high dimensionality that you need help from computers and computer scientists and statisticians (and you know you need help). So, it seems you astronomers are willing to tolerate us computer scientists. I have formed personal relationships with some of you, and I really like the community. And, lastly, as each of you knows, Astronomy is just such an amazing study – you are always on the threshold of discovering whole new realms.

Moore's, Gilder's, Metcalf's, and Amdahl's Laws

We computer scientists observe phenomena and coin "laws" to explain them. We know, and you know, that these laws are made to be broken (they reflect the *current* technology universe.) Memory chip capacity has been doubling every 18 months since 1965 – so that now 64megabit chips are common and 256Mb chips are available in small quantities. This doubling is called *Moore's law*. It gives a hundred-fold improvement every 10 years.

Moore's law applies more generally to microprocessor clock speeds (they have been doubling every 18 months for a long time), disk capacity (doubling every 12 months since 1990), and network bandwidth (doubling every 8 months according to George Gilder (and others) [Gilder].

An interesting thing about these laws, is that in the next doubling period, you install as much capacity as in the history of time ($\int_0^t = \int_t^{t+1}$). That is amazing. Among other things it means that if you can afford to store the data for one time period (a year) then storing it forever only doubles the hardware cost. If you compute something today, the computation in *3t* years will run 8 times faster and be 8 times cheaper (so do not bother to store it, just re-compute it if you do not expect to access it in the interim.) I will come back to some of these rules-of-thumb later.

Processors

Figure 1 shows how processing has gotten less expensive since 1880. Electro mechanical devices performance/price improved at a rate of 2x per 7.5 years. With the advent of vacuum tubes and then transistors, the doubling time dropped to 2.3 years. With the advent of integrated circuits, the doubling time dropped to one year! That is the cost of doing an n-bit operation has been dropping in half each year. I can cite lots of evidence for this (see for example progress on sorting at [Sort Benchmark]). About 50% of this speedup comes from faster devices (disks, cpus, memories, …). The other 50% comes from

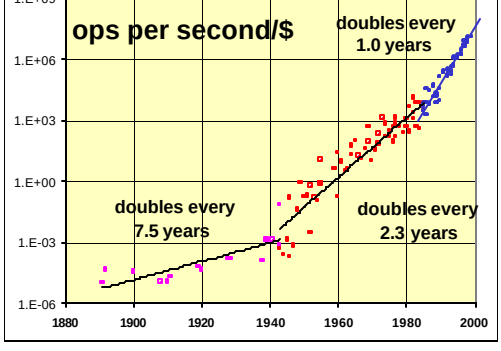

Figure 1: Operations/s/$ doubling times.

parallelism In 1980 processors executed a few million instructions per second, while current processors deliver about 500 to 1000 million instructions per second. That's "only" a 100 fold to 1000-fold improvement. But doubling for 20 years is a growth of 2^{20} ~ 1,000,000. So, where did the other factor of 1,000 come from? Well, it came from *parallelism*. The fastest sort uses 2,000 processors! Each of those processors is moving 64-bit words rather and 16-bit words in parallel. Each of those processors is heavily pipelined with much parallelism inside and outside the processor. It also uses about 5,000 disks. That's parallelism. You will find that I harp on this a lot, here and forever. Parallelism is your friend. Tom Sterling's Beowulf clusters are your friends. The 8,000 cpus at Hotmail are your friends, all running in parallel. The 6,000 processors at Goggle are your friends, all running in parallel. You could not do Hotmail, or Goggle, or Yahoo, or MSN or AOL on an SMP – because people are waiting. You can do all your astronomy on an SMP if you are patient, but it will go a LOT faster, and a lot cheaper if you use commodity clusters of inexpensive disks and processors.

The processor consumes programs and data from memory and sends answers (data) to memory. In a perfect world, whenever a processor wanted a piece of data, that datum would be sitting there instantly available. We do not live in that perfect world. Often, the processor does not know what data it will need until the last minute. For example if it is following a linked list, then the next data is not know until the current node of the list is examined. So, how far away is the data in, if it is a surprise? I think the picture in Figure 2 is a good way of thinking about this question.

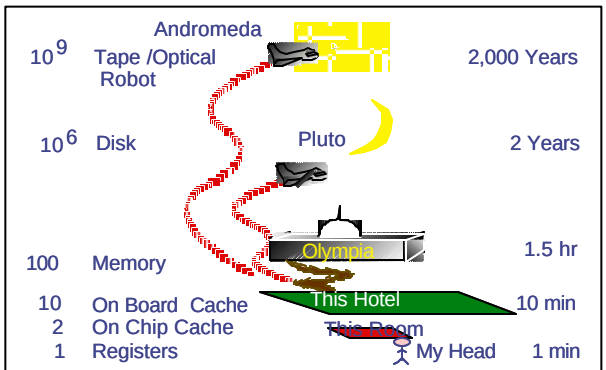

Figure 2: how far away is your data?

Processors live by clock cycles: today a clock cycle is a nano-second (for a 1GHz processor). In that time, a modern processor can execute 4 instructions in parallel, if all the data is available (a quad issue processor). But, what if the data is not in the processor registers? Then the processor must look in the processor cache (typically a fraction of a megabyte of data near the processor (e.g. on the processor chip/package)). If the data is there, great, that just costs 2 clocks. If the data is not there, the processor has to go to the level 2 cache (a few megabytes) that are typically 10 clocks away. If it is not there, the processor has to go to the L3 cache (typically main memory of a few Gigabytes). RAM data is 100 or more clocks away. If this is a big SMP, some memory can be even further away (say 400 clocks in a NUMA). If the data is not there, it must be on disk. Disk data is 10,000,000 clocks away. And if it is not there, then it must be on tape that is a trillion clocks away. To put this in human terms, our clocks run in minutes. If I ask you a question, you say: "Just a minute, you compute and you tell me the answer." If you do not know, you ask someone in this room (cache) and it is two minutes. If you have to go to cache it is like going somewhere on your campus. If you have to go to main memory, it is an hour's drive there and back. If you have to go to disk, it is 2 years (like going to the Pluto!), and if you have to go to disk it is like going to Andromeda (there is at least one joke associated with that).

If your program is doing list processing on disk, your program will execute 100 instructions per second – and you will be back with the 1940's class computers. If you want a modern computer you need to have the data ready for the processor. That means that you have to have good program locality (no surprises in the data stream). You have to go sequentially through the data so that the system can prefetch it and pipeline it to the processor. If you program in this streaming style, you get a 4 billion instructions per second rather than 100 instructions per second. In practice, people are achieving about one clock per instruction (there are some surprises in the data, but there is also lots of regularity.)

In the future we expect to see each "chip" contain several processors. In the mean time, there is a tendency to put many processors on a shared memory (Shared Memory Processors or SMP). Figure 3 shows this. You can get "hot" uniprocessors and you can get fairly high-performance 2-way 4-way, 8-way and even 64-way multiprocessors. You will notice that an N-way multiprocessor costs much more than N times

more than N uni-processors, and that the N-way multiprocessor is typically not much more powerful than a N/2 way system. For example and 8-way Intel is about 3 times slower than a 64 way Sun E10000 (so 40 processors are wasted), and an 8 way Intel is only 4x faster than an Intel uniprocessor, so half the processors are wasted (on a standard transaction processing benchmark.)

Figure 3: System balance: what is the right ratio of cpus:memory:disk:networks? Amdahl's law suggests that 1 million instructions per second processor needs 1 MB of ram and 1 mbps of IO. These laws are still approximately true and suggest that a billion instructions per second needs a gigabyte of storage and about 25-50 disk drives, and a Gigabit Ethernet connection. See [Gray] for more information on this.

Memory and the Memory Hierarchy

So, now we come to the memory hierarchy. You might think that Moore's law makes memory boring: it just doubles in size every 18 months. That's true. Memory is even more boring than that. It comes in various speeds: slow, slower, slower still, very slow, very very slow, and so on. Typical memory is a few hundred nanoseconds to access (when it is packaged and mapped). This is 10x faster than the memory of 1960. So, main memory speeds have improved about 10x in 40 years. That compares poorly to the return on Treasury bills (they grow at 5%). As I said before, if everything gets faster at the same rate, then the ratios do not change and noting really changes. One ratio that is changing is the speed and bandwidth to memory. Processor speeds are rising much faster than memory speeds, and processor bandwidth is rising much faster than memory bandwidth. The net of this is that it does not make a lot of sense to put two infinitely fast processors on a finite speed memory. Computer architects are struggling with this problem, (put the processors next to the memory); but today there is not much progress. My simple advice is to stick with low degrees of SMP and work hard on program locality so that you have streamed access to the memory.

In the past, main memory was a precious resource. In the future terabyte memories will be economic (a few tens of thousands of dollars). This will simplify some programming tasks (right now a TB of RAM costs a $1M to $100M depending on the vendor.)

Memory addresses are a key to the way we program. We are in a transition from 32-bit addressing to 64-bit addressing. Since memory doubles every 18 months, the memory needs an extra address bit every 18 months. Right now 4 gigabytes of memory costs about $5,000. Notice that 4GB (=2^{32}) needs 32 bit addresses, exhausting the 32-bit pointers used in most of our software. The MIPS, Alpha, PowerPC, Sparc and (someday) Itanium processors all support 64-bit addressing, as does IRIX, VMS, HPUX, Solaris, and AIX, but Linux, SCO, Windows, Netware, MacOS, and MVS do not yet have a 64bit memory option. Also most compilers and software systems have not upgraded to 64-bit addressing. This transition will give you (and me) considerable pain over then next 5 years. There is now a crash program underway to overhaul our software to be 64-bit friendly (the Microsoft Windows2000 and Linux solutions are undergoing beta test today – but that is just the first layer of the software stack). Now for the good news, *32/1.5= 20*. So it will be 20 years before we run out of 64-bit addresses and have to do this all over again. If I am lucky I will get to do this three times in my life (I had fun going from 16/24 bits to 32 bits).

Of course, the main good news is that large and inexpensive main memory lets you write your programs more simply: but remember Figure 2. This memory is getting *relatively* slower with every processor generation. In 10 years it will look about 100x slower in relative terms.

This naturally segues to disks and file systems and database systems. Again, many file systems have been late to convert to 64-bit addressing, so for them the largest file is 2GB (Linux, NFS, and Informix are examples). But most file systems and database systems are 64-bit (e.g. Windows, Solaris, AIX, IRIX, HPUX, Oracle, DB2, SQL Server,…).

Disks are relatively inexpensive $100 to $1000 each. Well-balanced systems have many disks (about one for every 10 MIPS, or about 40 disks per cpu with today's ratios. Managing that many disks might take

some effort, so most file systems aggregate many disks into one logical volume. The resulting volume looks like one big disk (of about a terabyte).

Each disk has a mean time to failure of 50 years, but when you have 50 of them, the group has a one-year mean time to failure. To eliminate these problems, the hardware or software duplicates the data on the disks (either duplexing it or using a parity scheme). The trend is to go towards duplexing because it is simpler and better utilizes the disk arms. The net of this is that 2 terabytes of "raw" disk becomes 1 terabyte of "cooked" (fault tolerant, file system mapped, …) storage.

Just as with RAM, disks hate surprises. A single disk can deliver about 30 MBps of data if the disk is read sequentially. If you read the same disk in random order (8KB pages), it will deliver about 120 reads per second and so about 1 MBps. This is thirty times slower than the sequential access pattern. This sequential:random bandwidth ratio is increasing. Therefore is increasingly important that large computations lay the data out on disk in a sequential order, and scan it sequentially if possible. Of course if you are looking at just one bit of data (if you just want to find a particular galaxy) you need not read all the data, you can just look the page address up in an index and then go directly to the page that holds that galaxy. But, as a rule of thumb, if you are going to read 1% or more of the pages, you should read all the pages sequentially.

Like memories, disks have gotten faster: in the last 5 years the sequential access speed has gone from 5MBps to 30 MBps, and from 70 random accesses per second to 125 random accesses per second. These 6x and 2x improvements are modest gains compared to the 30x increase in capacity (from 2GB to 60GB). Disk technologies now in the laboratory allow disks with current form factors to have 1 TB capacity, 200 random accesses per second, and 100 MBps bandwidth. Notice that it will take a LONG time to scan such a disk. We may end up storing the data 3 times on 3 disks for fault tolerance. This will allow us to scan the logical disk in 1,000 seconds, and will allow any single disk to fail without degrading performance.

So, here comes another pitch for parallelism. Suppose you have a Terabyte store and you read it sequentially. How long will it take? Well 1TB/(30MBps) = 30,000 seconds or about 8 hours. But that terabyte is probably stored on 40 disks (each 50 GB but each paired with it's mirror). So, if you read in parallel, the scan will take 833 seconds or 13 minutes rather than 8 hours. Sequential access gives a 30x speedup; parallel access gives a 40x speedup (or 10,000x speedup if you use 10,000 streams).

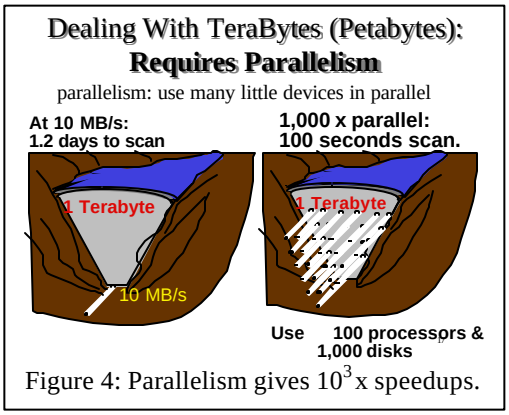

Figure 4: Parallelism gives 10^3 x speedups.

I have lunatic-fringe ideas about disk architecture. So, to be fair, I should start by saying that the conventional view is that you should buy disk cabinets from an established vendor like IBM, Compaq, EMC, or others. These cabinets house hundreds of disks, and about 10TB today. They then connect to your processors via a system area network (storage area network often built from fiber-channel). This is how the TerraServer (http://terraserver.microsoft.com/) is built. But, this approach is not inexpensive, and I think that the future may make this approach less attractive. First, these storage boxes are expensive: the empty box can cost $500k and a populated box can cost $1M per terabyte, compared to $0.7k to $15k for the disks themselves (wow! what a markup!). Second, the fiber channel interconnect does not offer bandwidth comparable to several hundred disks (each fiber channel link is guaranteed not to exceed 100MBps).

I advocate that you buy your disks with your processors, packaged together in Beowulf style. I have been building these storage bricks for our day-to-day work on the TerraServer, SDSS, and some other projects. The basic plan is to stuff a no-name PC with inexpensive disks. The folks at the Internet Archive (http://www.archive.org/) and many web-centric stores are taking a similar approach. The Archive has a 30TB store that cost them about 7K$/TB (including the processing and networking). The basic idea is that one buys inexpensive disks (right now 60 GB disks cost 200$ in quantity 1) and put 8 of them in an inexpensive server, as in Figure 5.

Now we come to tape. Let me start by saying that I hate tape and tape hates me. My experience with tape is that it is slow, unreliable, and the software to manipulate it is full of bugs (because the software is not used very much.) So, I try to avoid using tape – and I caution my friends and family to avoid it. Replace your tapes with disk.

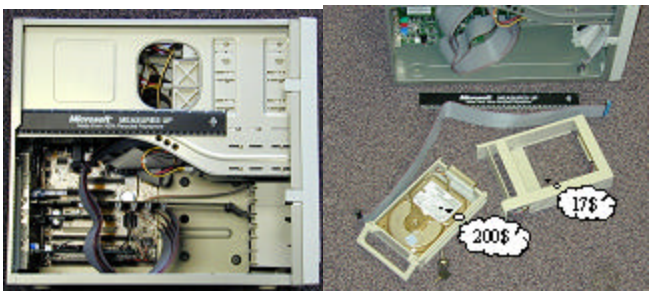

Figure 5: Inexpensive terabytes. A no-name PC includes a 700Mhz processor, 256MB ram, gigabit Ethernet, and housing for 8 disks. This costs 1k$ for the computer and 2k$ for the disks (and controller). The net is 7K$ per TB. The disks are hot-pluggable. They can be used for data interchange or for backup, eliminating the need for tape.

But, the simple fact is that offline tape is cheaper than disk. Again, taking the TerraServer example, we back it up to DLT7000 tapes, each is 40 GB and each costs us $60. So the backup of a terabyte costs about 25*$60 =$1,500. In practice, we get some fragmentation so it costs more like $2,000. Now $2,000 is less than the $7,000 in figure 5, and half t cost of the raw disk price. But when one includes the cost of a tape robot ($10k to $1M), then the price for nearline tape rises to $10k/TB, which exceeds the price for online disk. Technology trends will exacerbate this trend.

Tape access times are measured in minutes (not milliseconds). Tape bandwidths are 5x worse than disks. So tapes are a relatively slow and inconvenient way to store data that you want to read. I can scan all the data on the disks in my system in an hour or two. It takes a week to scan the data in a tape robot.

You no doubt use tape for archive and for data interchange, and are probably wondering how I expect to perform those functions with disks. For data interchange, I want to use the network to move small (multi-gigabyte) objects. For larger objects, I propose to ship disks as in the right side of figure 5.

As for archive, it takes a LONG time to restore a petabyte database from tape. With current tapes (6MBps), it will take 5.4 tape years (166M tape seconds) to restore a petabyte. Parallelism could speed this up by 1,000x if you had 1,000x more tape readers. But, still it is an unacceptably long time at an unacceptably high price. So, I want to store all the data online at two places. This means that each place stores the data twice (on 2 mirrored disks), and then we do site mirroring. Each observatory will count on others in the federation to back up its data, and conversely, the observatory will store some part of the federation's backup data at its site. This has to be part of the VO architecture. There is an initial 4x cost multiplier to do this reliable storage, but it comes with 4x more read bandwidth (since you can read all four copies in parallel.) Hence, if you need the read bandwidth, this redundancy may come for free if disk capacities are so large that disk bandwidth is the only limit to how much you put on a drive.

Having two (or more) online copies of the same data offers some interesting possibilities. Recall that the data should be laid out sequentially. The two copies can have different sequential orders: one could be spectral, the other spatial or some other variant.

Summarizing the storage discussion: storage access times and bandwidth are the scarce resources. Storage capacities are doubling every year or 18 months and are predicted to continue this trend for at least the next 5 years. You can optimize storage access by using sequential (predictable) access patterns so that pipeline parallelism is possible. You can leverage inexpensive processors and disks by partitioning your task into many independent parallel streams that can execute in parallel.

Communications

The trends in data communications are exaggerated forms of the trends in processing and storage. At the technology level, the speed of light is not changing, but there are both software and hardware revolutions in our ability to deliver bandwidth. For short distances (tens of meters) one can use point-to-point copper and

coax to send signals at up to 10Gbps. For longer distances, optical networks can carry signals at about 10 Gbps per wavelength, and can carry 500 different wavelengths, for a total of about 1 Tbps per fiber. These fibers can come in bundles of a thousand, so there is lots of bandwidth to go around. In 1995, George Gilder coined the law of the telecosm: *deployed bandwidth grows 300% each year*. So far, his prediction has held true, and some people are beginning to believe him [Gilder].

In the local area, gigabit Ethernet is about to become ubiquitous and inexpensive. This delivers 100 MBps to each processor and each device. If that is not enough then you can get multiples of these links. The cost of such links is about $1,000 now (NIC+cable+port at a switch). This is expensive compared to about $100 for a switched 100Mbps Ethernet, but the link price for GbE will drop to $100 over the next 2 years. Indeed, 10GbE is now emerging from the laboratories.

In parallel to this, most of the hardware vendors (IBM, Intel, Sun, Compaq, Dell, …) are working on a system area network called Infiniband (http://www.infiniband.org). This is a 10Gbps processor-to-processor link that is not as flexible as GbE, but has the virtue that the NIC is built onto the chip. Future machines will likely be a combination of 10GbE and Infiniband interconnects. (This will likely displace ATM and FiberChannel and MyriNet and GigaNet and ServerNet and …), mostly because these latter technologies are low volume and hence relatively expensive.

The sad fact is that local networks have been software limited: the software costs of sending a message have dominated the hardware latencies. The basic problem has been that tcp/ip and the sockets interface were designed for a wide area network with long latencies, high error rates, and low bandwidth. The local area network (LAN) performance suffered with this common software stack. Over the last 5 years, the software stacks have improved enormously, mostly driven by the need to improve web server performance and partly driven by the use of tcp/ip in parallel programming environments. Now most operating systems are willing to offload much of the tcp/ip protocol to the NIC (checksums, DMA, interrupt moderation, buffering) so the software is much less of a bottleneck.

As a demonstration of this, colleagues at ISI-east (in Arlington Virginia), U. Washington (Seattle, Washington), and Microsoft (Redmond Washington) have been showing off high-speed desktop-to-desktop WAN bandwidths in the 100MBps (mega byte per second) range. This is not so impressive in a LAN, but they have also demonstrated it coast-to-coast using commodity workstations and out-of-the-box Windows2000 software. The Unix community is working along similar lines. Indeed, the Windows2000 demo interoperates with a Linux box at similar speeds.

These are "hero" benchmarks that employ very high-speed links (2.5 Gbps), expensive switches (Juniper) costing hundreds of thousands of dollars each, and of gurus (who are priceless). The challenge now is to close this guru gap: to allow ordinary scientists and high-school students to get this same high-performance. The empirical fact is that most scientists are getting less than 100KBps to their desktop, 1,000 times less than the "advertised" performance. Companies like Microsoft are working on this problem for their products – one goal is for Windows2001 to deliver this performance with no user tuning. I believe other companies are also working to close the guru gap. The USG has given several million dollars to the Pittsburgh Supercomputer Center to build an open source auto-tuner for Linux (http://www.web100.org/).

You can expect that links in the wide area network will be able to move data at about 1GBps (10 Gbps). This means that it will be fairly easy to move astronomy datasets of less than a terabyte from place to place within an hour. In that same timeframe, disks will be about 1TB and will have bandwidth of about 200MBps, so the technologies will be reasonably well matched (you will use parallel disks).

Now, the bad news: Networking costs have been dropping fast. Public network prices have not improved much in the last 10 years. The retail cost of a megabit link has remained about the same since 1990. While the price of the technology has dropped about 1,000x the cost of the right-of-way, trucks, cones, and people has increased in that period. One might expect therefore, that you could buy a gigabit link in 2000 for the price of a megabit link on 1990. But, sadly, that is not the case. Someone is making a lot of money on

these artificially high prices. The hope is that competition will bring prices in line with costs over the next 5 years, but that hope failed over the last 5 years (see page 134 of [Odlyzko]).

Software

If you have watched the evolution of computing, it is easy to believe that there have been exponential improvements in hardware and linear improvements in software. After all: there is not that much difference between Java and Fortran.

In some ways I agree with this view, in some ways I disagree. If you want to compute GCD (greatest common devisor using the Chinese Remainder Theorem) then there has not been much progress in software. On the other hand if you want to sort, then there has been huge progress (50% of the sorting speedup is due to parallelism). If you want to do a database application, then there has been huge progress in the tools space. And if you want to build a web site, there has been huge progress in tools.

The VOF effort is more like a database problem or a web site problem than it is like GCD. The traditional Fortran/Cobol/Algol/C/Pascal/Ada/C++/Smalltalk/Java community has made substantial progress on traditional programming. I use these tools almost every day. But the real quantum leap has been to raise the level of abstraction so that you are programming at a meta-level. PERL-Unix operates at this meta-level. More generally, scripting languages like JavaScript, VBscript combined with powerful libraries seem to offer huge productivity gains over traditional approaches. John Ousterhout has been making this point for a long time [Ousterhout]. By now, I think most of us get it.

The traditional programming languages (including the scripting languages) run tasks sequentially. They do not make parallel execution easy to specify or to execute. Throughout this talk, I have tried to emphasize the need to execute things in parallel, if you want to be able to process large quantities of data quickly. The scientific community has tried to recast the Fortran model in a parallel style, first with High Performance Fortran (HPF) and more recently with PVM and MPI [MPI]. These are laudable efforts, and MPI is the workhorse of Beowulf clusters. But, MPI requires great skill on the part of the programmer. Meanwhile, the database community has figured out ways to automatically extract parallelism from SQL queries and execute those queries in parallel. Companies like Wal-Mart have huge clusters of machines (thousands of processors, 10s of thousands of disks) storing and searching the Wal-Mart inventory database.

Don Slutz and I have been working with Alex Szalay to apply these techniques to the Sloan Digital Sky Survey data. I think it is fair to say that we have made great progress. Alex is able to read data in parallel at about 200 MBps per node using an SQL interface. A 10-node cluster costing about 200k$ should be able to read data at 4 GBps and be able to answer most questions within 20 minutes. If we can put a Graphical query interface on the front of this, and a good data visualization system on the output, we should be able to allow astronomers to do data analysis without having to write programs. Rather the astronomers would just write the equations that we then execute in parallel against all the relevant objects [Szalay]. We hope that this approach will bring the benefits of parallelism to the VOF community.

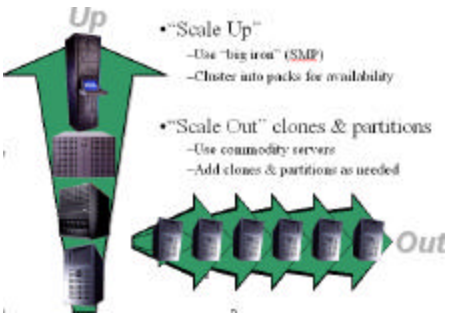

Figure 6. By parallel database techniques on many inexpensive processors and disks to search the astronomy archives in parallel, we hope to give astronomers automatic parallelism. Searching terabyte datasets in a few minutes.

We expect that groups will set up servers on the Internet that allow that group's data to be searched. Astronomers will be able to send queries to the server and get back answers either immediately (if it is a simple question) or within an hour (if the question requires scanning the entire archive). If an astronomer is interested in a particular part of the dataset, she can copy that part to a local parallel server (costing 1k$/terabyte of disk including the processors to manage and search that terabyte).

There have also been great advances in data mining technology for massive data sets – mostly being used in the commercial sector. Wall-mart, Amazon, AOL, and Microsoft are each investing heavily in these technologies to help their marketing and their Internet business. Similar, even more confidential, efforts are underway in the financial community and in the security agencies. Much of this work is proprietary – and so not generally understood. On the surface, it seems that these tools ought to be useful to Astronomy, since they are designed to find patterns in huge datasets of very high dimension with considerable noise in the data. It is also possible that advances in the Astronomy community could help these commercial interests. This is all quite speculative, but the public nature of the Astronomy datasets makes them an excellent testbed for new data mining algorithms.

How will the members of the Virtual Observatory exchange data? How will anyone find anything? How can the archives cooperate? The current answer is FITS (Flexible Image Transport System) files. This is actually an OK answer for some questions. Certainly FITS is a good way of representing astronomy data sets for data interchange. The more trendy approach would be to use XML as the data representation. That replaces ASCII with UNICODE and replaces all binary numbers with their Unicode representation. The result will be a bigger file (which compresses nicely) but the result will not be semantically any different from FITS. There are many tools that work with XML and XML is more widely known than FITS – both of these are important points, and probably justify a FITS-XML marriage.

In order for the archives to cooperate and in order for astronomers to compare data from multiple archives, the data in each archive needs to have a clear description of what it means. The units, coordinate system, precision, accuracy needs to be specified if other scientists are to be able to use the data. It is a difficult and thankless task to construct this *meta-data*. This fact means that a virtual observatory is not possible unless the scientists are willing to perform this difficult and thankless task. It is hard for me to guess the outcome. But, I must assume that the astronomers went to great pains to gather the data, so they will exert the effort to preserve it, and make it available to future generations.

Let's assume that the metadata has been built for a few important data sets. Then what? How will you or I find the data and access it? The best current story seems to be a combination of OpenGIS and SOAP (sorry to be talking in acronyms – but that is how it goes with us geeks). OpenGIS is defining standard ways to access geo-spatial data (GIS stands for Geographical Information System.) They have defined a standard meta-data framework for spatial data and are doing an XML representation now. This also includes logic for accessing the data: extracting data from an archive, and adding new data to it. It seems likely that there are strong ties between this spatial data and the spatial data in the astronomy community. Some combination of the FITS work with OpenGIS will probably emerge within the next 5 years.

The next step is to have a worldwide catalog of the metadata for all the astronomy data sets. Rather than bore you with more acronyms, I will just say that the commercial sector is cooperating to define such a "yellow pages" and "green pages" for Internet services. The Astronomy community can piggyback on this effort (XML is a core technology for these directory services). The Astronomy community already has a good start on this with the Vizier project (http://vizier.u-strasbg.fr/doc/astrores.htx).

SOAP (Simple Object Access Protocol) is a way to call subroutines anywhere in the Internet – from any language to any language. It maps your subroutine call into an XML message, and sends it to the server. It maps the answer onto XML and sends it back to your program. SOAP works for Fortran, Cobol, C++, Java, JavaScript, VB, and C#, and your favorite programming language (it is agnostic about languages). It uses web servers like Apache or IIS as the ORB, replacing the need for a CORBA or DCOM ORB.

As Tony Hoare observes "algorithms are procedures plus data." XML is just data, SOAP is a way to externalize (publish) your procedures. The Astronomy Archives need to define the procedures they want to publish and support. These procedures would return the data you (the client) want from the archive. There is not time to go into it here, but we are doing something like this for the TerraServer as a textbook case of how to externalize information from a web services to clients. This is a big part of Microsoft's .NET initiative.

Summary

I have tried to covey three main messages:

(1) Technology progress in processors, storage, and networks is keeping pace with the growth of Astronomy datasets.

(2) Access times are not improving so rapidly, hence we must use parallelism to ameliorate this gap. With parallelism, we will be able to scan the archives within an hour. This parallelism should be automatic. I believe database technology can give this automatic parallelism.

(3) The Virtual Observatory will be a federation of database systems. As such, the federation needs to define interoperability standards. FITS is a good start, but the community could benefit by looking at progress in the commercial sector in using directories, portals, XML, and SOAP for data interchange.

References

[Chung] L. Chung, J. Gray, **"**Windows 2000 Disk IO Performance," MSR-TR-2000-55, http://research.microsoft.com/scripts/pubs/view.asp?TR_ID=MSR-TR-2000-55

[Devlin] B. Devlin, J. Gray, B. Laing, and G. Spix, "Scalability Terminology: Farms, Clones, Partitions, and Packs: RACS and RAPS," MSR-TR-2000-85 http://research.microsoft.com/scripts/pubs/view.asp?TR_ID=MSR-TR-99-85

[FITS] Flexible Image Transport System: http://fits.gsfc.nasa.gov/

[Gilder] G. Gilder, *Telecosm: How Infinite Bandwidth Will Revolutionize Our World, Free Press, ISBN: 068480930, 2000*, or "Fiber Keeps its Promise," Forbes, 7 April, 1997, http://www.forbes.com/asap/97/0407/090.htm or http://www.gildertech.com/

[Gray] J. Gray, P. Shernoy "Rules of Thumb in Data Engineering," ICDE 2000, April 20000, San Diego, or http://research.microsoft.com/scripts/pubs/view.asp?TR_ID=MSR-TR-99-100

[Lesk] M. Lesk, "How Much Information Is There In the World?" http://www.lesk.com/mlesk/ksg97/ksg.html

[MPI] Message Processing Interface Forum http://www.mpi-forum.org/

[OpenGIS] http://www.opengis.org/index.htm

[Ousterhout] http://www.scriptics.com/people/john.ousterhout/

[Odlyzko] A. M. Odlyzko, *The history of communications and its implications for the Internet*, http://www.research.att.com/~amo/doc/ history.communications0.pdf.) page 134.

[Sort Benchmark] The sort benchmark web site: http://research.microsoft.com/barc/SortBenchmark.

[SOAP] http://www.w3.org/TR/SOAP or http://www.develop.com/soap or

[Szalay] A. Szalay, P. Z. Kunszt, A. Thakar, J. Gray, D. R. Slutz: "Designing and Mining Multi-Terabyte Astronomy Archives: The Sloan Digital Sky Survey", ACM SIGMOD Conference 2000: 451-462, http://research.microsoft.com/scripts/pubs/view.asp?TR_ID=MSR-TR-99-30