Super-Servers:
Commodity Computer Clusters Pose a Software Challenge

Jim Gray
310 Filbert Street, San Francisco, CA. 94133-3206
Gray @ crl.com

Abstract: Technology is pushing the fastest processors onto single mass-produced chips. Standards are defining a new level of integration: the Pizza Box – a one board computer with memory, disk, baseware, and middleware. These developments fundamentally change the way we will build computers. Future designs must leverage commodity products. Clusters of computers are the natural way to build future mainframes. A simple analysis suggests that such machines will have thousands of processors giving a tera-op processing rate, terabytes of RAM storage, many terabytes of disc storage, and terabits-per-second of communications bandwidth. This presages 4T clusters. To an iron monger or software house: the T stands for Terror! To customers it stands for Tremendous! These computers will be ideally suited to be super-servers in future networks. Software that extracts parallelism from applications is the key to making clusters useful. Client-server computing has natural parallelism: many clients submit many independent requests that can be processed in parallel. Database, visualization, and scientific computing applications have also made great strides in extracting and exploiting parallelism within a single application. These promising first steps bode well for cluster architectures. The challenge remains to extend these techniques to general purpose systems.

Outline:

Introduction

Computers are a key force in the evolution of human civilization. They change the way we communicate, the way we act, the way we play, the way we do science, the way we learn, and even the way we think. I believe that the revolution has just begun -- there is much more coming. As such they are key to the fabric of each society.

Some view the computer is the hardware embodiment -- the box. This paper argues that the boxes will be a commodity by the end of the decade. In the next century, the computer industry will be dominated by the software that animates these boxes with new applications.

The most exciting software will be the new clients: the super-phone, the intelligent-TV, the intelligent-car, the intelligent house, and most exciting of all, the intelligent assistant. These artifacts will all be part the intelligent universe predicted by Herb Simon. In that world, all our artifacts will have behavior and will be programmed to adapt to and assist people.

These billions of clients will need millions of servers. The servers will store, process, and communicate information for the smaller and mobile clients. This paper focuses on the construction of such servers. They will come in many sizes, most will be small. Some servers will need to be very powerful **super-servers**. This paper argues that these servers must be constructed from commodity hardware. Economics form the basis of these arguments, so the paper touches on the new structure of the computer industry.

This paper was invited by the German ACM. It is good news for Germany and for the EU. Clearly, Europe does not dominate the current hardware or software industries. But, the new software industry is wide open. It is quite reasonable for Europe, with its recognized talent for innovation and design excellence to lead the application-oriented software industry. This paper focuses on the need to design software for super-servers. There is a corresponding need to design software for information appliances (super-clients).

These ideas have been evolving for many years. Gordon Bell is their main and most articulate proponent. This paper grew out of an 1990 taskforce at Digital Equipment chaired by Barry Rubinson. Participants included Bob Bean, Andrew Birell, Verell Boaen, Barry Goldstein, Bill Laing, Richie Lary, Alan Nemeth, Ron Obermarck, Tom Rarich, Dave Tiel, and Cathy van Igen. A confidential version spread widely in the Internet, so in 1992 a public version as Digital SFSC Technical Report 92.1. This is the 1994 revision of that never-published paper.

The 1992 version had two major changes over the 1990 version. (1) High-speed networks were mentioned (gigabit LANs and megabits WANs). This was recognized as the BIG change in computer architecture. Other parts of the computer were getting only ten to one hundred times cheaper and faster in the next decade. Networking was getting thousands or millions of times faster and cheaper in the next decade. (2) Clusters were contrasted with distributed systems. Clusters are simple distributed systems (homogeneous, single site, single administration).

The 1994 version showed four years progress: (early 1991 to late 1994). NT replaces POSIX as the darling operating system. Generic, *Middleware*, replaces the failed POSIX (=UNIX), SAA, and NAS initiatives. Networking promises are more real. Disks and tapes exceeded my technology forecasts. Cpus are on schedule; but RAM is evolving more slowly, more in

step with the pessimistic predictions of 4x every 4 years rather than 4x every 3 years. Tape technology and tape robots had been ignored, but are now included in the discussion.

Standards Are Coming!

By the end of the decade, boatloads of NT or POSIX systems, complete with software and hardware, will be arriving in ports throughout the world. They will likely be ten times more powerful than today's Pentium workstation, and will cost less than 10,000$ each, including a complete Microsoft software base (front and back office). No doubt they will come in a variety of shapes and sizes, but typically these new super-computers will have the form factor of a PC or VCR. These products will be inexpensive because they will exploit the same software and hardware technologies used by mass-market consumer products like, HDTV, telephones, desktop teleconferencing, voice and music processors, super-FAX, and personal computers.

How can traditional computer companies add a hundred billion dollars of value to these boxes each year? Such added value is needed to keep computer industry giants like AT&T, Bull, Digital, HP, Hatachi, Fujitsu, IBM, ICL, NEC, Olivetti, SNI, and Unisys alive.

I believe that the 100B$/year will come from three main sources:

Manufacture: Provide the **hardware** and **software components** in these boxes.

Distribute: **Sell, service,** and **support** these platforms for corporations. Although the boxes will be standard, corporations will want to out-source the expertise to install, configure and operate them and the networks that connect them. Much as they outsource car rentals.

Integrate: Build **corporate electronics**, by analogy to consumer electronics, prepackaged or turnkey application systems that directly solve the problems of large corporations or provide mass-market services to consumers. The proliferation of computers into all aspects of business and society will create a corresponding demand for **super-servers** that store, analyze, and transmit data. Super-servers will be built from hundreds of such boxes working on common problems. These super-servers will need specialized application software to exploit their cluster architecture. Database search and scientific visualization are two examples of such specialize application software.

As in the past, most revenue will come from manufacturing and distribution – the traditional computer business. The high profit margins will be in integrated systems that provide unique high-value products. For example, in 1993 Compaq made 7B$ of revenue and .5B$ of profit on Microsoft-based systems. Microsoft made only 4B$ of revenue on those sales but more than 1B$ in profit – 7% profit versus 28% profit. Similarly, Microsoft made as much profit on the average Apple system as Apple Computer did. Adobe's margins are higher than HP's on HP postscript printers.

Integration is not a new business for traditional computer companies, but the business structure will be different. There will be more emphasis on using commodity (outside) products. The development cost of standard products will have to be amortized across the maximum number of units. These units will be marketed to both competitors and to customers. Development of non-standard products will only be justified for items that make a unique contribution with order-of-magnitude payoffs. The cost of me-too products on proprietary platforms will be prohibitive.

This phenomenon is already visible in the PC-marketplace. In that market, standardized hardware with provides the bulk of the revenue, but has low profit margins. A few vendors dominate the high-margin software business (notably Microsoft, Novell, and Lotus).

I conclude from this that the application software business will be the most innovative and most financially attractive sector of the computer industry in the year 2000.

Business Strategy In An Era Of Commodity Components

Profit margins on manufacturing commodity hardware and software products will be modest, but the volumes will be enormous. So, it will be a good business for a few large producers, but a very competitive one. There will continue to be a brisk business for peripherals such as displays, scanners, mass storage devices, and the like. But again, this will be a commodity business with narrow profit margins – much like the commodity PC industry of today.

Why even bother with such a low-margin business? The reasons are simple, jobs and technology. For many nations and companies it is essential to be in the high-volume business. The revenues and technology from this high-volume business fund the next generation and cross-fertilize new products and innovations. This can already be seen in the integrated circuit business where DRAM manufacturing refines the techniques needed for many other advanced devices.

There is a software analogy to this phenomenon visible within IBM, Lotus, Novell, Microsoft, and Oracle. There are economies-of-scale in advertising, distributing, and supporting software. Microsoft's Windows products demonstrate the importance of an installed base and of a distribution network. In addition, the pool of software expertise in developing one product is a real asset in developing the next.

On the other hand, observe that IBM could not afford to do all of SAA and that Digital could not afford to do all of NAS. These projects are so huge that they were stretched-out over the next decade. In fact, they are so huge, that alliances were formed to spread the risk and the workload. This is a root cause of the many consortia (e.g., OSF, COSE, OMG, ...). For IBM and Digital to recover the development costs for SAA and NAS, their software efforts will have to become ubiquitous. NAS and SAA must run on millions of non-Digital and non-IBM hardware platforms. This outcome seems increasingly implausible.

There is no longer room for dozens of companies building me-too products. For example, each operating system now comes with a SQL engine (DB2 on AIX, OS/2, and MVS, Rdb on VMS, SQLserver on NT, NonStop SQL on Guardian,...). It will be hard to make a profit on a unique SQL engine – SQL is now commodity software. A company or consortium must either build an orders-of-magnitude-better unique-but-portable SQL product, or form an alliance with one of the portable commodity SQL vendors. Put glibly: each company has a choice, either (1) build a database system and database tools that will blow away Oracle, Sybase, Informix, and the other portable database vendors, or (2) form an alliance with one of these commodity vendors.

Networks show a similar convergence. The need for computers from many vendors killed IBM's SNA and Digital's DECnet. Customers are moving away from these proprietary protocols to use the TCP/IP protocol instead. The need for interoperability, and especially the need to support desktop and client-server computing has driven this trend more quickly than predicted.

There is confusion about standards. There are committee standards and there are industry standards. For example, the ISO-OSI standards have had almost no impact -- rather it has been a de facto standard (TCP/IP) driven by the PC, UNIX, and Internet that became pervasive. We return to this issue in a later section.

In general, each computer company will both build and buy. This probably represents the way things will be in the future; no company can afford to do everything. No single company can produce the best implementation of all standards. Even Microsoft has its limits: it has 85% of the desktops but Novell has 70% of the servers. Lotus dominates the Mail and Workflow components of the Microsoft desktop.

There will be a good business to migrate legacy systems to commodity platforms -- but that will be a small part of the business of using these new platforms.

System Integration And Service In A Commodity World

The costs of designing, implementing, deploying, and managing applications has always dominated hardware costs. Traditionally, data centers spent 40% of their budget on capital, and 60% on staff and facilities. As hardware and software prices plummet, there is increasing incentive to further automate design, implementation, and management tasks.

Cost-of-ownership studies for client-server computer systems show that most of the money goes to system management and operations. A full-time support person is needed for every 25 workstations. Just that cost exceeds the workstation cost after a year or two.

This is reminiscent of the 1920 situation when a human operator was needed to complete each telephone call. It was observed then that by 1950 everyone would be a telephone operator. Ironically the prediction was correct, direct dialing made us all telephone operators.

If computers are to become ubiquitous, we are all going to become system designers, administrators, and operators. Computer software designers are going to have to automate and elevate the programming process by presenting visual (object-oriented) metaphors for task parameters and sequencing. This should allow "ordinary" people to program, manage, and use information appliances.

Automating the programming, operation, and use of servers and super-servers is equally important. As shown below, the super-server will have thousands of components. Software must manage and exploit these components automatically.

Where will this automated software come from? The computer industry is rapidly moving to a horizontally structured industry as diagrammed in Figure 1. In this model, rather than having one company provide all the services, the customer contracts with a systems integrator who combines products from many vendors into a solution tailored the customer. The customer may operate the resulting system, or may contract with someone to operate it.

Function	Example
Operation	AT&T
Integration	EDS
Applications	Computer Associates
Middleware	Oracle
Baseware	Microsoft
Systems	Compaq
Silicon & Oxide	Intel & Segate

Figure 1: The horizontal structure of the new information industry. In a vertically integrated industry one company provides the complete solution. In a horizontal industry, providers at each level select the best components from the lower levels to provide a product at their level. Few companies are competitive at more than one level.

The super-server will primarily be an applications and integration business. It will not be a shrink-wrapped, mass-market business. It will be more like the business of building bridges, airports, hospitals, or oil refineries. Each is a separate industry.

Systems integrators and applications designers need deep application-knowledge to implement application-specific super-servers. Each problem domain has different needs. There are big differences between a document super-server, a consumer shopping super-server, a stock and commodities trading super-server, and a scientific data storage and analysis super-server. They need some common middleware, but mostly they need domain specific knowledge to build applications and middleware on top of commodity products. It is

likely that companies will be built around one or another problem domain -- one specializing in documents, another specialized on geographic data, another specialized on financial systems, and so on. These companies will add considerable value, and so should be profitable. They will write software to adapt commodity middleware, baseware, and systems to the particular problem domain.

4B Machines: Smoking Hairy Golf Balls

Today, the fundamental computer hardware building blocks are cpus, memory chips, discs, tapes, print engines, keyboards, displays, modems, and Ethernet. Each is a commodity item. Computer vendors add value by integrating these building blocks and by adding software to form workstations, mid-range computers, and to some extent mainframes. Apple, AT&T, Compaq, Digital, HP, IBM, Sequent, SGI, SNI, Sun, and Tandem, all follow this model. They use commodity components. Proprietary product lines are shrinking.

The unit of integration has gone from vacuum tube to chip. The next step in integration will be a minimal hardware/software package. By the end of this decade, the basic processor building blocks will be commodity boards running commodity software. The boards will likely have one or more 1 bips cpus (billion instructions per second), 1 GB (Giga byte) of memory, and will include a fairly complete software system. This is based on a technology forecast shown in Table 1.

Table 1: Hardware component technology forecast.

Year	1 Chip CPU Speed	1 Chip[1] DRAM	1GB Disc	Tape	LAN	WAN
1990	10 mips	4 Mb	8"	.3 GB	10 mbps Ethernet	64kbps ISDN
1995	100 mips	16+ Mb	3"	10. GB	150 mbps ATM	1mb/s T3
2000	1000 mips	64+ Mb	1"	100. GB	850 mbps ATM	1 gbps fiber

This forecast is fairly conservative. It also estimates the following costs for the various 2000 components (see Table 2.)

Table 2: Cost forecast for year 2000 hardware components.

	CPU	DRAM	1GB Disc	tape robot	LAN	WAN
unit cost	500$	15$	200$	1,000$	200$	200$
1,000$ buys	2 cpus	.5 GB	5x10 GB disk =50 GB array	1TB tape robot	5 x LAN	5 x WAN

These costs must be inflated by about 2x to package the components into a mass-market product. Given these costs, one could build a processor, .5GB of RAM, several high-speed communications chips, and ten discs, package and power them for a few thousand dollars.

Such computers are called **4B machines** (Billion instructions per second, Billion bytes of DRAM storage, and a Billion bytes per second of IO bandwidth, and a Billion bits per second of communications bandwidth). A **5B machine** will support a Billion bit display, that is 4000x4000 pixels and each pixel 32 bits of shading and color[2]. These machines are the natural evolution of the 5M machines that drove the PC revolution of the 1980's (mip, megabyte of ram, megapixel display, 10 megabit per second LAN, and a mouse).

[1] There is good evidence that DRAMs are evolving more slowly than they have in the past. This slower evolution comes from reduced demand and increased capital costs. If recent trends continue, in 1999 DRAMS chips will be at 64Mb and will cost about 15$ each. Thanks to Steve Culled of Digital for this observation.

[2] Some prefer to call these 4G and 5G machines using Giga instead of Billion.

To minimize memory latency these 4B machines will likely be **smoking-hairy-golf-balls**[3]. The processor will be one large chip wrapped in a memory package about the size of a golf ball. The surface of the golf ball will be hot and hairy: hot because of the heat dissipation, and hairy because the machine will need many wires to connect it to the outside world.

Dramatic changes are also expected for storage and networks.

Disc farms will be built from mass-produced 1" discs placed on a board; much as DRAMs are placed on memory boards today. A ten-by-ten array of such discs will store about 100 GBytes. Disc array technology will give these disc-boards very high performance and very high reliability[4].

Tape farms will be built from arrays of inexpensive tape robots. Each robot will have about a hundred tapes. Each tape will store 100GB of data. This will give an inexpensive way to store 10 TB nearline. The use of many such robots exploits commodity components, and minimizes queuing delays for tape transports. Multiple transports also increases bandwidth with parallel transfers.

Networks: Networks will be much faster. Fiber based communications will be able to deliver gigabit data rates, but at a high price. Commodity fiber-optic interfaces will run at gigabit speeds. Local communication (LANs) will be able to use this bandwidth, but long haul bandwidth will still be expensive. So, although gigabit-WANs will be possible, and may form the backbones of the Internet, it seems likely that megabit-WANs will be more typical. The transition from the low speed WANs of today running at 64kbps, to the higher-speed commodity ATM WANs of 1999 running at 155 mbps (OC3) will be a major architectural shift for data communications. These changes in network performance and network economics will be key enablers for super-servers. Such networks will allow almost instant access to data and images distributed all over the world.

Baseware: The base software for 4B machines will contain all the elements of X/Open, POSIX, DCE, SAA, and NAS. In particular it will include some standard descendants of Motif, C++, SQL, OSI, DCE-UNIX, X/Open, and so on. The NT operating system along with its GUI, integral database engine, and system management tools is emerging as the commodity baseware of choice.

Perhaps more significant, I believe that Microsoft's OLE standard will become ubiquitous. OLE will be present hundreds of millions of desktops. It's class libraries will be the standard representation for data capture and data display. This will drive all other systems to support OLE. The result is that the OLE class libraries define the standard format for documents, sounds, images, videos, spreadsheets, and other common datatypes. OLE also sets the template for creating new datatypes and representations. It is already the standard way to capture, store, and display data. It will be the mechanism we use to extend databases, operating systems, and viewers.

These basic building blocks will be commodities. That is, the hardware will be mass produced and so will have very low unit price. Standard operating systems, window systems,

---

[3] Frank Worrell used this metaphor in 1985. Frank is now working at LSI Logic.

[4] Patterson, D. A., G. Gibson and R. Katz. (1988). *A Case for Redundant Arrays of Inexpensive Disks (RAID).* Proc .ACM SIGMOD. 109-116. or Schulze, M., G. Gibson, R. Katz and D. A. Patterson. (1989). *How Reliable is a RAID*. 34th IEEE Compcon 89. 118-123.

compilers, class libraries, database systems, and transaction monitors will have high volumes and so will also have low unit prices. This can already be seen in the workstation world. There, NT, OS/2 and NetWare provide complete software systems (database, network, and tools) for less than a thousand dollars.

Today, most applications are not portable from one family to another (e.g., from Intel to Alpha). NT makes applications portable among hardware platforms and provides limited portability from UNIX to NT. The stable interfaces will be software interfaces: windows interfaces, programming languages, file systems, class libraries, databases, and network protocols.

Future Mainframes: 4T Machines

In a classic paper Gordon Bell and Dave Nelson defined the basic laws of computing[5]. One of their key observations is that there are seven tiers to the computer business. These tiers are roughly categorized by the dollar value of the computers:

10$: wrist watch computers
100$: pocket/ palm computers
1,000$: portable computers
10,000$: personal computers (desktop)
100,000$: departmental computers (closet)
1,000,000$: site computers (glass house)
10,000,000$: regional computers (glass castle)

Bell and Nelson observed that each decade, computers from one tier move down a notch or two. For example, current portables have the power and capacity approximating that of a 1970 glass-house machine. Machines with the power of 1980 workstations are now appearing as palmtop computers.

Bell and Nelson observed that service workers can be capitalized at about 10,000$ of computer equipment per person on average. That more or less defines the price of the typical workstation.

The costs of departmental, site, and regional servers can be amortize over many more people, so they can cost a lot more.

What will the price structure look like in the year 2000? Will there be some super-expensive super-fast neural-net computer that costs ten million dollars? If future processors and discs are very fast and very cheap, how can one expect to build an expensive computer? What will a main-frame look like?

One theory is that the mainframe of the future will be 10,000$ of hardware and 990,000$ worth of software. Being a software guy, I like that model. Fighter planes work this way. Each new fighter is smaller and lighter – yet costs much more because it is filled with fabulously expensive software and design. It's unlikely that similar mechanisms will operate for commodity super-servers.

OK, so the 99% software theory is blown. What else? Perhaps the customer will pay for 990,0000$ worth of maintenance or service on his 10,000$ box? Probably not. He will probably just buy two, and if one breaks, discard it and use the other one.

I conclude that the mainframe itself will cost about a million dollars in hardware. What will a million dollars buy? It will buy (packaged and powered) about:

~	1,000	processors	= 1 TOP	(tera-op: trillion instructions per second) or
~	100,000	DRAMs (@64Mb+)	= 5 TB	(half a terabyte RAM) or
~	10,000	discs (@1GB)	= 10 TB	(ten terabytes disc) or
~	10,000	net interfaces (@1Gbps)	= 10 Tb	(10 terabits of networking)

So, the mainframe of the future is a 4T machine!

---

[5] See C.G. Bell and J.E. MacNamera, *High Tech Ventures,* Addison Wesley, 1991, pp. 164-167

Who Needs a 4T Super-Server?

What would anyone do with a 4T machine? Perhaps the mainframe of the future is just a personal computer on each desk. A thousand 4B PCs would add up to a 4T "site" computer. The system is the network! This is in focus of the NOW (Network Of Workstations) project at Berkeley[6]

Each worker will probably have one or more dedicated 4B computers, but there will be some jobs that require more storage or more processing than a single processor, even one of these super-powerful 4B ones.

Consider the problem of searching the 25 terabyte Library of Congress database looking for a for all documents similar to a specified one. A single 4B processor using current software (e.g., Rdb) would take about a month to do this search. By using a thousand 4B processors in parallel, the search would take about a half hour and would cost about 20$. Such searches on a 2 TB database are common today in marketing applications.

Similar observations apply to applications that analyze or process very large bodies of data. Database search is prosaic compared to data visualization algorithms mapping vast quantities of data to a color image. These search and visualization problems lend themselves to parallel algorithms. By doubling the number of processors and memories, one can **scaleup** the problem (solve twice as big a problem), or **speedup** the solution (solve the problem twice as fast).

Some believe that the 4B machines spell the end of machines costing much more than 10,000$. I have a different model. **I believe that the proliferation of inexpensive computers will increase the need for super-servers. Billions of clients mean millions of servers.**

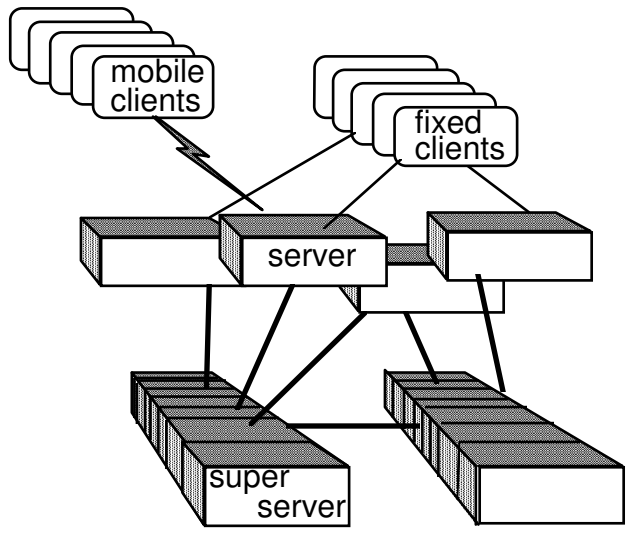

Figure 2: The structure of computer systems. Mobile clients will have low-bandwidth access to the network. All others will have high-speed and low-cost access to local servers and to remote super-servers. Local servers and remote servers will be built from the same hardware and software components. The super-servers will simply be large arrays of processors, disks and tapes. Each server will be functionally specialized to some task like document search and dissemination, video service, intelligent agent shopping, and so on.

[6] *Proceeedings of NOW Workshop*, Hennesy, J., & Patterson, D. (ed.), ACM, ASPLOS Conference, San Jose, CA, Sept. 1994.

A fraction, say 25%, of future computer expenditures will go for super-servers. The typical strategy today is to spend half the budget on workstations, and half on print, storage, and network servers. In the end, the split may be more like 90-10, but servers will not disappear. The main arguments in favor of centralized servers are:

Power: The bandwidth and data storage demands of servers supporting hundreds or thousands of 4B machines will be enormous. The servers will have to be more powerful than the clients. **Fast clients want faster servers**.

Control: The proliferation of machines and bandwidth will make it possible, even easy, to access centralized services and resources. No longer will you go to the video store to get a videotape, you will download it. No longer will you search paper libraries for information, you will have a server do it for you. These resources (movies, libraries,...) will contain valuable information. Central utilities (or at least regional utilities) will want to control access to them. They will set up super-servers that offer an client-server interface to them.

Manageability: People do not want to manage their own data centers. Yet, the trends above suggest that we will all own a personal data center in 1999. Each PC and perhaps each mobile telephone will be a 4B machine. There will be a real demand for automatic data archiving and automatic system management. This will likely be a centralized service. A simple example of this is visible today with the tendency to use X-terminals to move management issues from the desktop to the closet.

What Are The Key Properties Of Super-Servers?

Servers must have the following properties:

Programmable: It is easy to write client and server applications for the server.

Manageable: It is easy to manage the server.

Secure: The server can not be corrupted or penetrated by hackers.

Highly available: The server does not lose data and is always "up".

Scaleable: The server's power can grow arbitrarily by adding hardware.

Distributed: The server can interoperate with other super-servers.

Economic: The server must be built from commodity components to be inexpensive.

Clusters – The Key To 4T Machines

Servers need to be as powerful or more powerful than their clients. They must serve hundreds or millions of clients. How can powerful servers with all these properties be built from commodity components? How can a collection of hundreds of 4B machines be connected to act as a single server? What kind of architecture is needed? What kind of software is needed?

I believe the answer is **clusters**: Tandem, Teradata, and Digital currently offer clusters that scale to hundreds of processors. Their clusters have excellent programming tools, are a single management entity, and are secure. Clients access servers on the cluster not knowing where the servers are running or where the data resides. So the cluster is scaleable. Processors, storage, and communications bandwidth can be added to the cluster while it is operating. These clusters are fault-tolerant; they mask faults with failover of discs and communications lines. Teradata's TOS, Tandem's Guardian, and Digital's VMS have the transaction concept integrated into the operating system. These clusters are built from commodity processors (Intel x86, MIPS 4000, Alpha), commodity disks and memories, and are among the most economic servers available today as measured by the TPC A, and C benchmarks. In addition they hold the performance records on all those benchmarks because they scale so well.

Well, that is the official marketing story; and there is a grain of truth to it. But, the details of the Teradata, Tandem, and Digital clusters do not deliver on most of these promises. Digital and Tandem clusters do not currently scale much beyond a hundred processors (Teradata scales to about 500.) The programming and management tools of the Digital and Tandem clusters not offer much transparency; each component is managed individually. Few of the tools use more than one-processor-at-a time in running an application; this dramatically limits the ability to scaleup or speedup applications by adding hardware. The cluster price is not especially economic when compared to PC-based servers -- it only compares well to UNIX and mainframe prices. Lastly there are single points of failure in the software and operations aspects of these cluster systems: e.g., software upgrades are not hot-pluggable.

But, these clusters are certainly a step in the right direction. Clusters are the direction that most vendors have adopted. Notable examples are:

AT&T Teradata builds clusters out of the Intel x86 family and proprietary software. These clusters act as back-end SQL servers to mainframes and LANs. Teradata systems feature economy, scaleability, and fault-tolerance. The largest clusters have over four hundred processors and two thousand discs. AT&T hopes to build systems that scale from the palm to the super-computer by building clusters of Intel x86 processors[7].

Digital has long offered the VMScluster and is not evolving it to the AlphaCluster. They have ported the VMS cluster technology to UNIX-OSF and to NT.

IBM Sysplex is a cluster of up to forty eight 390 processors. At present there is very little software to support this cluster hardware. In addition, the IBM AIX system (their UNIX clone) running on the PowerPC-SP2 hardware has a software cluster concept. It supports an array of scientific parallel programming packages and IBM has a large efforts to add parallelism to its DB2/2 products on AIX-SP2 and DB2 MVS.

---

[7] *NCR's 486 Strategy,* Moad, J., *Datamation*, V36.23, 1 Dec. 1990, pp. 34-38.

Intel is building a hyper-cubes of a thousand processors. Unlike the other machines mentioned so far, software, fault tolerance, and input/output seem to be afterthoughts -- rather they are focused on the small high-end scientific computing market. Similar comments apply to **Cray's** T3D, **TMI**'s CM5, and **KSR**'s machines.

Oracle, leveraging its experience on VAX clusters, cloned the VMS distributed lock manager and has ported it to most UNIX systems. Oracle reports excellent scaleup on nCUBE, Sequent, and SP2 clusters. **Informix** and **Sybase** report similar scaleups on Sequent and AT&T clusters.

Tandem builds clusters out of a proprietary hardware-software combination running as network servers. Tandem system features match the super-server list above. The systems scale to about 100 processors and to a few hundred discs. Customer complaints center on system manageability. Tandem is working hard on that issue.

Sequent sells a shared memory multiprocessor based on Intel processors. Originally it could scale to about 30 processors, but with the faster Pentium processors, the design peaks at about 9 processors. Next generation processors will require a more partitioned design. Cray T3D, KSR, Encore and SGI all have designs that circumvent these problems.

Silicon Graphics builds shared memory processor arrays based on its MIPS processors. These clusters scale to about 30 processors. SGI hopes to scale to hundreds of processors in the next generation design[8].

Looked at in this light, no one is ready to build a thousand processor 4T machine *and the associated software*. Some are ahead of others. Teradata and Tandem are the leaders today, but IBM's SP2 and Informix or Oracle on SGI are also promising.

[8] M. Heinrich, et. al., "The Performance Impact and Flexibility of the Stanford FLASH Multiprocessor, " 6th ASPLOS, Oct. 1994.

Cluster Software – The Key To 4T Clusters

It is important to understand the virtue of clusters. The idea is to add more discs, more processors, more memory, and more communications lines, and get more work out of the system. This speedup and scaleup should go from one processor-memory-disc-communications module to several thousand modules.

Traditionally, multiple processors have been connected by sharing a common memory: Shared Memory Multi-Processors (SMP). The SMP approach does not scale well in a world of smoking-hairy golfballs. The event-horizon of a smoking hairy golfball is on the processor chip; signals from one ball cannot get to the next ball before the processor goes on to the next instruction. A memory shared by two such golfballs looks more like a communications line or a remote processor. SMP designers are aiming for ten-way parallelism. The hundred-fold and thousand-fold speedups available using commodity processors in a cluster have much higher payoff. Most of the machines mentioned above have a cluster architecture. They communicate via messages rather than via shared memory.

Perhaps SMP systems will solve their scaleability problems, but many believe this is not possible. In addition, they must solve the fault-isolation problem. If one part of an SMP fails, it must not contaminate the other parts. In the end, a synthesis of the shared everything and shared-nothing designs will probably prevail[9].

The goal of cluster software is to divide-and-conquer large problems. It must do three things:
1. break the computation into many small jobs,
2. spread the jobs among many processors and memories executing in parallel, and
3. arrange that traffic among the jobs does not swamp the network or create interference.

The challenge has been to extract parallelism from applications. Certain applications like timesharing and transaction processing have natural parallelism. Each client represents a separate and independent request. Each request can go to a separate processor. So, servers with many clients have inherent parallelism. If the number of clients doubles, and if there are no bottlenecks in the hardware or software design, then doubling the number of servers, storage devices, and communications lines will give good scaleup. This is what VMSclusters, Teradatas, and Tandems do today.

The real challenge is recognizing and extracting parallelism *within* applications (big batch jobs). This is an ad hoc field today. SQL servers have discovered how to extract parallelism from large database queries – they search each disc of the database in parallel, they sort in parallel, they join tables in parallel, and so on. These systems display good speedup and scaleup to a hundred processors. Notable commercial examples of this are Teradata and Tandem[10].

Beyond that there have been few successes. Today, recognizing parallelism is an application-specific task. The application programmer must program parallelism into his application by inventing new and innovative algorithms. Automatic extraction of parallelism from applications stands as a major research challenge.

---

[9] Mike Stonebraker, *The Case for Shared Nothing*, IEEE Database Engineering, Vol. 9, No. 1, 1986.

[10] David DeWitt and Jim Gray, *Parallel Database Systems: The Future of Database Processing or a Passing Fad?*, CACM, Vol. 35, No. 6, June 1992.

The current situation is (1) 4T machines have a bright future as parallel SQL servers and (2) servers get natural parallelism and scaleup from having many clients. So, no scientific breakthroughs are needed to get the parallelism needed to use 4T machines as data servers. These servers will have lots of opportunities for parallelism.

Cluster Software – Is It a Commodity Business?

Once the parallelism problem is "solved" innovation is still needed to make clusters manageable, secure, and highly available. Evolving cluster software to solve any of the major problems (parallel software, manageability, security, fault-tolerance) will be a major software initiative.

There is a fundamental question about whether these initiatives should be based on a proprietary system (NT) or on an Open System (OSF DCE UNIX). I am unclear on the answer to this question. NT is the property of Microsoft and so can only be evolved by them. OSF is in the public domain and so allows much more innovation and experimentation by many different groups. Both are portable to the instruction-set-of-the-month; and both are very large.

The easy way out of this is to base future clusters on UNIX. UNIX is portable and standard. The problem is that UNIX is like stone soup[11]. You have to add a lot to get what you want. If we add 1M lines to UNIX for fault tolerance, 1M lines for distributed databases, and 10M lines for manageability, do we still have UNIX? Have you built a commodity product? Will super-servers be a commodity product – probably not.

Super-Servers will use commodity hardware and proprietary software. Super-servers will have sales volumes measured not in millions of units, but in tens of thousands of units – one super-server per thousands of clients. Super-server operating and management software will have demanding requirements that will not be satisfied by commodity client software. There will be a few server operating systems that offer an open interface (e.g., SAA, NAS, POSIX, X/Open, or the like), run on clusters of commodity devices, but that are proprietary. The software will have many unique performance and management features. It would be best to base this software on NT so that it could run client software at the server. At a minimum, the server will have to support the NT and UNIX application programming interfaces.

This is good news for anyone who wants to make a business of super-servers. If they were easy to build and had huge volumes then there would be a lot of competition for them and margins would be very slim. The key to a successful business is having a product that everybody needs but that few people can build. The software that goes into super-servers may well be such a product.

The next section tries to explain why standard software (e.g., vanilla UNIX-DCE) is unlikely to produce a competitive cluster architecture.

[11] The recipe for stone soup calls for a stone to be placed in a large pot of boiling water. Each guest is requested to bring an additional ingredient (e.g., onions, carrots, ...). The quality of the soup depends on the quality of the guests.

Standards: Tell Me It Isn't SO (Snake Oil)

Some believe that all this Open-UNIX-Standards stuff is Snake Oil (SO for short). I do too – well perhaps its not all snake oil, but there is a lot of hype about standards This is an unpopular view – or at least a reactionary one; but it deserves a fair hearing. The SO view proceeds as follows.

Standards are Boring: Customers always want some leading edge feature. They use this as a competitive advantage. Leading edge features have not made it into standards. Parallelism, fault-tolerance, manageability, and high-performance tricks are unlikely to become standards.

Standards are Incomplete: It is standard to see the seven-layer ISO protocol stack. You have seen the 1000-page SQL standard. You have seen the multi-volume X-Windows books. Guess what? They are the tip of the iceberg.

- The ISO protocol stack has an elevator shaft running down the side called network management. That elevator shaft is not standard. There are implementations that are de facto standards (e.g., NetView-SNMP), but they are not standard. ISO and SNMP ignore issues like security and performance.
- The SQL standard looks the other way about most errors (they just define a few simple ones), performance (no performance monitor), utilities (no load/dump, import/export,...), and administration (no accounting, space management,...).

The SO reactionaries believe that computing is fractile: there is complexity in every corner of it. Workstations hide this complexity by dealing with a single user and ignoring system management. Consequently, the cost of managing a workstation on a LAN now routinely exceeds the cost of the workstation. SuperServers cannot ignore these problems.

When building a workstation, one aims for simplicity. Microsoft has an *open* standard MS/DOS - a single code body that is its own spec. Apple's Macintosh is a similar story. The UNIX world has a standard *open* application programming interface that allows many simple stand-alone programs to be easily ported from one platform to another. The CICS world has 300,000 programmers who know and love the CICS application programming interface – that is its own spec. These systems are all *open* and standard. Their programming interfaces are published and do not change much, they just grow.

But there is a separate world. There is no real open-standard operations interface for a network of PCs, or for the applications that run them. All the tools to do these operations tasks are proprietary. The CICS operations interface is not well documented, is not open, and it changes from release to release. It is the elevator shaft.

Certainly, the standards organizations have place-holder bodies that are "working" on these elevator shafts, but the SO reactionaries believe such efforts are doomed. These big-systems issues are too specific to become commodity standards or products.

Standards have poor-performance: The standard NFS protocol stack from SUN has poor performance. SUN and other vendors have *deep* ports of this standard code that are much faster. They sell the fact that they have the best NFS. Someone who offers a vanilla NFS server will have a difficult time competing.

Similar comments apply to SQL. Rdb is a deep port of SQL to VMS (actually Rdb was written explicitly for VMS). Other SQL systems do not take advantage of VMS. Rdb consequently displaced other SQL systems on VMS. It was better, less expensive, and came from the hardware vendor. These performance issues are more important for servers than for clients, since servers are resource-poor compared to clients (even in a cluster). I predict a similar scenario for Microsoft's SQLserver on NT -- it will be difficult for "portable" implementations to compete with this native database system.

There is no question that the mass-marketed systems will run standard, mass-marketed software. But, if my model is correct, then a super-server cluster will have commodity hardware and proprietary software extensions. The software may be "open" in the POSIX-Windows sense that applications are portable to it and can interoperate with it. But it will not be the commodity OSF software; it will have LOTS of value added in the areas of scaleability, security, manageability, and availability. In this model, the clients (millions of them) will be running commodity software, but the servers will not.

This reactionary SO view matches the current situation: today the clients are commodity DOS, Windows, or UNIX systems. The small servers are also commodity systems: NetWare. But, past a certain threshold the commodity servers hit a wall. The hardware does not scale up to clusters and neither does the software.

Server vendors have a choice; they can build this super-server cluster software and call it anything they want. They can call it UNIX or NT++. It will be mostly new code. It will surely be X/Open branded, POSIX compliant, and support DCE; so it will be UNIX. But it will have a large body of code that is in neither NT nor UNIX today.

Clusters versus Distributed Systems, What's the Difference?

Why isn't a cluster just a distributed system? Why won't all the wonderful software we have been developing for distributed systems apply directly to clusters? Won't DCE solve the cluster problem? After all, DCE means Distributed Computing Environment.

Well, right! A cluster is a distributed system. Everything, even an isolated PC is a distributed system. That is the virtue of distributed systems, they encompass all and integrate everything.

A cluster a *special* kind of distributed system. It has properties that make it qualitatively different.

Homogeneous hardware and software: A distributed system necessarily involves many types of computers with many different software systems. This heterogeneity comes at a cost. General purpose algorithms are needed to communicate among nodes. Things on one node are slightly different than things on another. It is expensive, if not impossible, to offer transparent access to all data at all nodes.

In a cluster, all the nodes are running the same software and have approximately the same hardware. This simplicity has huge benefits, both for performance and for transparency. It is relatively easy to give the illusion that the entire cluster is a single computer.

Single administrative domain: Distributed systems are designed to cross organizational and geographic boundaries. Each node is considered an independent member of a federation. Since boundaries among nodes of the network are explicit, designers of distributed systems make many design choices that allow fine-grain (node-level) control and authorization.

A cluster is more like a single node of a distributed system. The cluster may consist of thousands of devices, but it is managed as a single authentication domain, a single performance domain, and a single accounting domain. The cluster administrator views it as a single entity with no internal boundaries.

Ideal communication: In a distributed system communication is slow, expensive, and unreliable. The finite speed of light and long distances make it slow - 100 ms round trip is typical. The long distances and huge capital costs of common carriers imply that communications lines are the most expensive part of a distributed computer system. Public networks lose individual connections, and occasionally deny service for extended periods.

By contrast, communication within a cluster is ideal. The distances are short, less than 100 meters; so the speed of light delay is short, less than a microsecond. Bandwidth is plentiful and inexpensive in a cluster. One can just add more ports and fibers. The short distances and low communication complexity within a cluster give highly reliable communication. Since all members of the cluster speak the same language, very efficient communications protocols can be used.

In summary, a cluster is a special kind of distributed system. Distributed systems techniques help build clusters, but the differences make clusters both simpler and faster than distributed systems. In a sense, it is much easier to build a cluster than to build a distributed system.

Of course, a cluster acting as a server will be a key part of a distributed system. It will be a super-server node of the distributed system.

Conclusion

The PC marketplace shows how mass-production and economies-of-scale can mask the engineering costs that dominate minicomputer and mainframe prices today. Technology is pushing the fastest processors onto single mass-produced chips. Standards are defining a new level of integration: the POSIX box. These developments fundamentally change the way we will build computers. Future designs must leverage commodity products.

Clusters of computers are the natural way to build the mainframe of the future. A simple analysis suggests that such machines will have thousands of processors, terabytes of RAM, many terabytes of disc, and terabits-per-second of communications bandwidth. This gives rise to the 4T clusters. These computers will be ideally suited to be super-servers in future networks.

Software that extracts parallelism from applications is the key to making clusters useful. Client-server computing has natural parallelism: many clients submit many independent requests that can be processed in parallel. Database, visualization, and scientific computing applications also have made great strides in extracting and exploiting parallelism. These promising first steps bode well for cluster architectures.

Anyone with software and systems expertise can enter the cluster race. The goal in this race is to build a 2000-processor 4T machine in the year 2000. You have to build the software to make the machine a super-server for data and applications -- **this is primarily a software project.** The super-server software should offer good application development tools - it should be as easy to program as a single node. The 4T cluster should be as manageable as a single node and should offer good data and application security. Remote clients should be able to access applications running on the server via standard protocols. The server should be built of commodity components and be scaleable to thousands of processors. The software should be fault-tolerant so that the server or its remote clone can offer services with very high availability.