(65)

# 1

# Three Decades of Multiprocessors

Gordon Bell

## 1.1 Introduction

During the last 25 years, the author has never really considered any alternative to the multiprocessor (mP) for general-purpose, cost-effective, nonminimal computers. This involvement with mPs at Digital Equipment Corporation (DEC), Carnegie Mellon University (CMU), Encore, Stardent, and Kendall Square Research (KSR) included 16 computers. Fourteen were built, including 7 for research or advanced development, and 6 were built for the marketplace. The reasons for designing mPs have been quite compelling.

The only alternatives for general-purpose use that the computing community support are the uniprocessor and evolving networks of computers (e.g., local area network (LAN) connected workstations). A uniprocessor with tens to hundreds of thousands of processing elements controlled by a single instruction stream is relatively ~~easy~~ possible to build and to learn to program. When considering efficiency, such a vector processor is limited to solving very large, simple problems because each processing element performs the same operation. This is equivalent to directing a very large chorus with only one type of voice. Connecting an arbitrary number of computers as a networked multicomputer in order to get higher performance or higher availability requires that computers have to communicate by passing

Connection Machine

messages in a fashion akin to problem solving in a human organization. Programs have to be organized and written for message passing use in the same way one structures a human organization.

In 1990, all of the general-purpose computer classes are implemented as mPs: supers for the highest performance; mainframes for historic reasons; minis in order to become mainframes and to have higher performance than is available from micros; "multis" (for multiple microprocessor) as the mini replacement; and PC-based and workstation-based micros because it is nearly impossible to build a uniprocessor using modern microprocessors that are not multiprocessor capable.

Given that mPs are now standard computer structures, it is important to understand how they got that way, why build them versus other kinds of computers, why the author bet on them for supercomputing in the mid-1990's, what the impediments to their use have been, why they are built now, and the challenge to their evolution. with Hillis

## 1.2 The Evolution of Multiprocessors

Each of the computer classes have evolved mPs at different rates depending on the technology and market need. Mainframe mPs have remained almost unchanged for nearly 30 years except to adopt to the cache (c. 1968) in their second generation, thereby allowing them to improve beyond four processors. Supercomputers have evolved quite rapidly to more than a few processors with the adoption of the mP by Cray Research and the constant need for greater performance. With powerful, low-cost and very small microprocessors, and a "standard" operating system, UNIX, as a base, the multi was born circa 1983. These structures, which have several dozen processors, have evolved rapidly to support parallelism and to provide mainframe performance at relatively low cost. The hierarchical multi with several thousand processors is in the early formation process as it now appears possible to build a very large computer with a shared address space that all processors use. The slow adoption of mPs to date has come from the market and technology cycle operating in reverse: the lack of computers has inhibited the lack of progress and understanding about parallelism and this in turn has inhibited the lack of a market.

### 1.2.1 Mainframes

Exhibits 1.1, 1.2, and 1.3 show the evolution of a number of mPs beginning in 1962 with the Burroughs military (D825) and mainframe (B5000) computers. The exhibits show how the mainframe companies adopted the mP as a means of supplying greater throughput and access to a common database to a large, user group. Mainframes are implemented with a cross-point switch (either located centrally or built as a multiported memory) to connect the processors and input/output (i/o) to the memory. In 1968, the introduction of the cache caused increased the complexity of mP designs in order to maintain memory coherence in a distributed memory. On the other hand, the cache reduced memory access latency, thereby enabling the design of mPs with greater than two processors by reducing the demand to have very high-bandwidth, low-latency memory systems. The result has been a slow, but steady increase in the number of processors with time from 2 to 4 in 1962 to over 10 in 1990 (see Exhibits 1.1, 1.2, and 1.3). Only now are such systems being supplied with software to increase both throughput and single job performance through parallelism.

### 1.2.2 Supers

Cray Research introduced their first mP in 1982 and has moved aggressively to design mPs for both throughput and high performance using parallel processing. Exhibit 1.4 shows the reasons why Cray has adopted the larger mP:

1. The clock has improved only 14% per year on the entire line of Seymour Cray designs. The performance for the Cray 1-based technology (X, Y, C90) has improved less rapidly.
2. The main uniprocessor advances occurred over a decade ago going from simple overlap of instructions and data in the Control Data Corporation CDC 1604 (c. 1960, not shown), to multiple functions units in the 6600, to pipelining in the 7600, and to vector processing in the Cray 1.
3. Increases through the addition of more processors is the fastest way to increase overall performance.

It is important to note that the Cray architecture does not use a cache, and hence building very high-performance computers is mainly a matter of designing very high-bandwidth switches to couple the vector processors to memory. As Exhibit 1.3 shows, all the supercomputer companies have adopted the mP model in order to claim to have the highest peak performance.

Supercomputers are leading the advance to both explicit and implicit (transparent) parallel processing simply because, unlike mainframes, which have evolved very slowly, the existence of supercomputers depends on building machines that have the highest computational performance.

### 1.2.3 Multis for Minicomputers, Workstations, and Personal Computers

In contrast to large machines, the "multi" is derived from technology. Single-bus, multiple microprocessors, called multis [2] began to be introduced in

| *Type* | 1962-63 | 1964 | 1965 | 1966 | 1967-69 |
|---|---|---|---|---|---|
| *Unique* | | | | | |
| | 2:G21 | | | | 10xSafeguard |
| | (Bendix/CMU) | | | | (BTL) |
| *Mainframes* | | | | | |
| B~~o~~urroughs | 2xB5000 | 2xB5500 | | | 4xB6500 |
| | 4xD825 | | | | |
| CDC | | 4x3600 | | 2x6500 | 2x6700 |
| Digital | | | | 2xPDP-6 | |
| Honeywell/Bull | | | 4x635,645 | | |
| IBM | | | | 2x/67 | 2x/65 |
| Univac/Unisys | | | 3x1108 | | |

| *Type* | 1970 | 1971 | 1974 | 1976 | 1979 |
|---|---|---|---|---|---|
| *Unique* | | | | | |
| | | | 16xC.mmp | 50!Cm*(CMU) | 4x11/784 |
| | | | (CMU) | 16xHydra(DEC) | (DEC) |
| *Mainframes* | | | | | |
| B~~o~~urroughs | 2xB5700 | | | | |
| CDC | | | | | |
| Digital | 2xPDP-10 | | | | |
| Honeywell/Bull | | 4x60XX | | | |
| IBM | | | 2x158,168 | | |
| Univac/Unisys | | 4x1110 | | | |

**EXHIBIT 1.1**
The evolution of mP computer families with time: 1962--1979. During the first two decades, mPs were only used to build mainframes. Notation: x = crosspoint using central switch or multiported memory, xx = multistage crosspoint, : = bus connected, ! = hierarchical switch, # = dance hall, multistage switch, v = vector processors.

1983 by Synapse (now defunct). Almost all computers outside of the largest, traditional supercomputers and mainframes are built using this model because microprocessors provide higher performance processing than multiple chip, gate array implementations of minis and mainframes (e.g., the VAX and the IBM 360). Because the structure of the multi, using a single bus, is the simplest computer that can be built, it is virtually impossible to avoid building a mP when connecting a microprocessor to primary memory. Exhibit 1.3 shows the large

| *Type* | 1981 | 1982 | 1983 | 1984 | 1985 |
|---|---|---|---|---|---|
| *Unique* | | | | | |
| | 4xIntel432 | 28:Synapse | | | |
| | | 6:Elexsi | 4x784 | 12?:Firefly | |
| | | 20?HEP | (DEC) | (DEC) | |
| *Mainframes* | | | | | |
| Univac/Unisys | | | | 4x1194 | |
| *Supers/mainframes* | | | | | |
| Cray | 2xxXMP | | 4xxXMP | 4xxCray2 | |
| *Multis* | | | | | |
| Arix | | 4:Arete? | | | |
| Encore | | | | 20:Multimax | |
| Sequent | | | 12:Balance | | |

**EXHIBIT 1.2**
The evolution of mP computer families with time: 1980--1985. Beginning in 1982, mPs began to be used for supercomputers. Notation: x = crosspoint using central switch or multiported memory, xx = multistage crosspoint, : = bus connected, ! = hierarchical switch, # = dance hall, multistage switch, v = vector processors.

number of computers built in this fashion today. For the next decade, the author would expect the majority of nonworkstation, general-purpose, high-performance computers to be of this form.

### 1.2.4 Large (100) Multiprocessors

Several companies have worked on large mPs. IBM Research was the first to declare a 512-processor mP arranged in 8 clusters and called the RP3; only a 64-processor version was built. Evans and Sutherland, Inc., announced and withdrew their 256-node multi arranged in 16 processor clusters, although it built several four processor clusters. Bolt, Bernack and Newman, Inc. has introduced its TC2000, a 64-processor computer based on its Butterfly switch that can be configured with 512 computer modules (each a Motorola 88K with built-in cache and local 4 to 32 megabytes of memory). The TC2000 is clearly a multicomputer since the latency to access memory on another computer is comparatively long. Intergraph has announced a plan for a 500-node mP development with Electronic Data Systems, Inc.

| *Type* | 1986 | 1987 | 1988 | 1989 | 1990 |
|---|---|---|---|---|---|
| *Unique* | | | | | |
| | | 64:M31? | 8xBiin | 4x(8!Cedar) | |
| | | (DEC) | 8!(64!)RP3 | 8x(16x)ES1 | |
| | | | (IBM) | | |
| *Mainframes* | | | | | |
| Digital | | | | | 4vX9000 |
| IBM | | 3x2x3090 | | | 12v? |
| *Supers/mainframes* | | | | | |
| CDC/ETA | | | 8xxETA10 | | |
| Cray | | | 8xxYMP | | 16xxC90 |
| NEC | | | | | 4xxSX3 |
| *Large Multis* | | | | | |
| BBN | #Butterfly | | | #TC2000 | |
| Intergraph/EDS | | | | | 500? |
| KSR | | | | | 1000? |
| *Minisupers* | | | | | |
| Alliant | 8vFX8 | | | | |
| Convex | | | 4vxC240 | | |
| *Minis/superminis* | | | | | |
| Digital | | 4:8800 | 6:6206 | | |
| *Multis* | | | | | |
| Digital | | | 4:6000 | 6:6000 | |
| Sequent | 24?:? | | | | |
| *Workstations* | | | | | |
| Apollo | | | | 4:DN10K | |
| Digital | | | | 2:Firefly | |
| Solborne | | | | 4:? | |
| Silicon Graphics | | | 4:240 | 8:Power Series | |
| Stardent | | | 4v:Titan | | |
| *Personal computers* | | | | | |
| Compaq | | | | 2:8XX86 | |

**EXHIBIT 1.3**
The evolution of mP computer families with time: 1986--1990. By 1990, the mP was used to build general-purpose computers of all types. Notation: x = crosspoint using central switch or multiported memory, xx = multistage crosspoint, : = bus connected, ! = hierarchical switch, # = dance hall, multistage switch, v = vector processors.

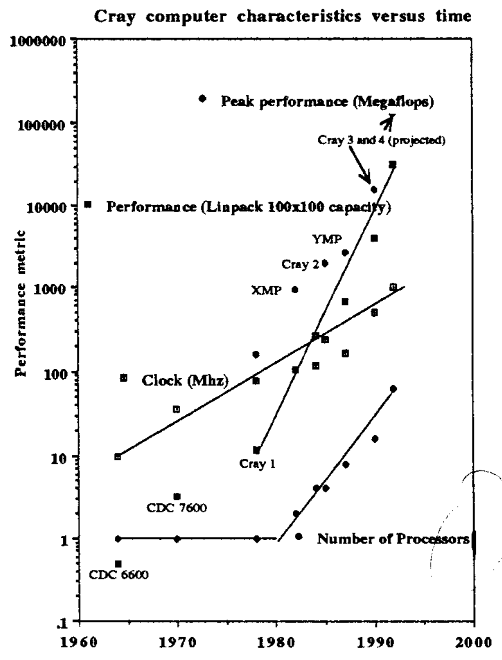

**EXHIBIT 1.4**
Cray research computer performance characteristics is plotted versus introduction date. Characteristics include the clock, number of processors, peak floating point operations per second, and throughput as measured by Linpack 100 × 100 capacity (i.e., unit processor performance multiplied by number of processors).

Kendall Square Research is introducing a hierarchical multi with 1024 processors, each delivering a peak of 40 Mflops. The hierarchical multi provides almost two orders of magnitude more aggregate processing power (10 to 20 billion instructions per second in 1024 10 to 20 mips processors) than any alternative general-purpose computer structure.

## 1.3 Why Build Multiprocessors Versus Other Types of Parallel Computers?

Many approaches are possible to increase performance and/or provide higher reliability. Although the numbers of computers have increased and become real and deployed, the kinds of computers have not changed significantly since the taxonomy posited by Bell and Newell [1]. [1] The suitability of a given machine for an application depends on a number of factors beyond simple performance and price metrics, including the difficulty of adapting existing programs and algorithms, the range of problems a structure can solve, and even the novelty of the structure and supplier. Three kinds of usage are occurring in order of generality: (1) applications-specific hardware/software that is designed to solve one problem within a limited class of problems (e.g., image processing); (2) run-time-defined, application-specific, and monoprogrammed (reprogrammed) to be used on one problem at a time within limited classes of problems; (3) general-purpose computers that are configured with many resources and are used in a multiprogrammed fashion to solve a wide range of applications at a given time.

The uniprocessors and mPs are used in a general-purpose fashion. Multicomputers, such as the transputer or hypercube, can be made at almost any scale but are most often used in a monoprogrammed fashion since problems must be rewritten to exploit them. The single-instruction single-data (SIMD) structure can also be implemented to have an arbitrarily large amount of parallelism, but must be reprogrammed. Finally, machines with fixed function units or a fixed array of processing elements (e.g., systolic arrays) are highly problem specific.

The suitability of a structure to a problem or job mix also depends on the amount and granularity of parallelism inherent in the application. Even the most parallel application is likely to have significant sequential code which limits its overall performance. Thus a SIMD or multicomputer must be built to operate at the highest speed for sequential programs, not from slow components. For example, about half of the 24 Livermore Fortran Kernels are serial. Running just one of the kernels infinitely fast only increases the overall operation rate of the set by 4% whereas running one of the programs 1/25th the speed, lowers the operation rate a factor of two.

Not being multiprogrammed to a significant degree limits the utility of a computer, and hence drives up the cost per application. A uniprocessor, such as the Connection Machine, can be multiprogrammed to make more efficient use of its resources thereby improving the throughput. Multicomputers, on the other hand, are more difficult to multiprogram because each program from a set must be allocated to a particular set of machines. The operating system first has to

[1] The new introductions since 1971 include array and systolic processors.

statically allocate the programs to the computer and then to allocate dynamically the computers to the processing work (unknown a priori) on an efficient basis.

## 1.4 The Bet: Multiprocessors Will Form the Fifth Generation

Given the bias toward general-purpose mP computers, the author bet Danny Hillis that in the last quarter of 1995 more supercomputer power, measured in sustained megaflops, will be shipped by evolving multiple processor supercomputers than by distributed memory computers such as today's SIMD or multicomputer. The bet centers on Hillis's plot (the number of processors versus the per processor power) of the supercomputer race as being run by a few, very fast processors (today's supers) versus a few million, slow processors. [2] mPs are evolving to have more processors and the SIMD is evolving to have higher-speed, more independent instruction streams and more processing elements per instruction stream.

By 1995, if supercomputers continue to exist as very large computers, they will most likely be constructed with a hierarchical memory (i.e., a distributed memory) and hence the bet may be difficult to interpret. For example, the Supercomputer Systems, Inc. computer is built as 4 clusters of 16 processors. A shared memory means a single, common address space and a model of implicit communication of data whereby any processor can access any data item. The opposite would imply multiple, private address spaces for each of the processors and require explicit messages in order to pass data among the processing elements.

## 1.5 Have the Impediments to Multiprocessors Been Removed?

It is essential to understand why a technology as compelling as mPs has been so slow to be adopted and whether all of the issues inhibiting their adoption in mainframes have been addressed as they relate to supercomputer and multi evolution. In this way, we may have some understanding about their future.

CMU's C.mmp. [7] enumerated many of the reasons why mPs were not part of mainstream computer design to rationalize the research for CMU's C.mmp mP. The report conjectured (and countered):

[2]The author also takes issue with the plot because he still considers a SIMD computer such as the Connection Machine to be a vector uniprocessor with a very large number of processing elements connected to a widely dispersed memory inter-connected in a hypercube.

1. High total cost. The high cost of processors and memories limited the market to high-performance systems. Using the best price performance minicomputer components would allow the construction of a lower-cost mainframe.
2. High processor cost. Relative cost of processors such that the incremental processor failed to increase performance/price. With minis, the processor price is a small part of the system.
3. Unreliable, base software. Unreliability and performance degradation of operating systems in 1972. Constructing a more complex system would be futile. Software reliability is improving, and operating systems can be built in an engineered fashion.
4. High switch cost. Inability to construct switches cheaply. Medium Scale Integration (MSI) has enabled the design of low-cost, reliable switches.
5. Memory conflict. Performance loss through memory conflicts and access delays. Strecker's work [6] enabled this to be computed versus having to be simulated.
6. Parallel programming design and algorithms. Dividing a task to operate in parallel is unknown. Work has been started in several places. Indeed, this is one key area of research.
7. Parallel program construction. Difficulty and complexity of constructing programs to operate in a parallel environment. Mechanisms for synchronization exist including fork-join and P and V. Methods for constructing large complex programs are emerging. The experiment will focus on this.

The kinds of results the C.mmp research would address included

1. Processor Memory System (PMS) design to understand the hardware and system design issues
2. Hardware implementation to understand engineering aspects, especially reliability
3. Models of parallel computation, including parallel processing, network and pipelined processing of tasks, functional specialization, multiprogramming, partitioned operation
4. Decomposition of complex programs such as speech processing was somewhat understood and could serve as a user model
5. Operating system design and performance
6. Measurement and analysis of performance
7. Using mPs for reliable computation through higher level redundancy

Wulf et al [7] described the results of the research, which mainly included the design of a capabilities-based operating system (list item 5) Hydra. The central cross-point switch design (items 1 and 2) drew concern from an uninformed academic community, but it proved to be straight-forward and reliable. The principal advantage of a low cost, 16 × 16 cross-point built from a few hundred MSI chips was that only 32 cables were needed to interconnect the 16 processors and 16 memory modules. A multiported memory design would, by comparison, require 256 cables. Only a modest amount was learned about performance because the PDP-11's limited floating-point arithmetic limited its computational power and was of limited use to attract real users. The limited, 16-bit address not only exacerbated operating system and applications design, but it lowered performance. Parallel programming (items 3, 4, 6, and 7) issues were only addressed tangentially. It is important to observe that

> *any computer predicated on using many low power processors to provide very high performance will fail to be used unless the ultimate power is at least one to two orders of magnitude greater than is obtainable on any other computer.*

### 1.5.1 Does Computer Science Research Help?

In retrospect, one wonders how the line of research computers have helped multiprocessing outside of providing some design principles and training computer scientists and engineers who can build computers. In reviewing research results, it has become clear to the author that the "university" mPs could have had a greater impact by simply expressing the understanding about parallelism in units of work performed per unit of time in a comparative fashion, instead of in units of speedup and time (universally poor for experimental machines). In effect, good university research builds "toy" computers because of their scale and speed.

Research computers must express performance metrics in terms of work per clock unit. This allows experimental computers using slow components to be built as "scale models" and "simulators" for "what could be a real computer" so that it can be compared with a "real computer." The experimental researcher is first obliged to understand what a "real computer" does. Thus any newfound knowledge about architecture or parallelism can be compared with reality. That is, does a new structure show promise against what already exists?

One of the flaws in research is the plethora of graphs of speedup versus number of processors with no comparative metrics for real computers and with no corresponding insight or theory. The common fatal flaws in most all research showing speedup are

1. choosing an algorithm that can be sped up over a faster, but serial algorithm.
2. achieving speedup by having a very slow part of the program that would not exist in a "real" system. By having slow, floating point arithmetic the granularity of a problem can be increased, making more parallelism

possible. This can only be solved by building a true, scale model of the entire computation and then normalizing the results to a clock. In this way "what could be a real computer" can be accurately simulated.

3. making a computer from mixed technologies (e.g., CMOS and ECL) to demonstrate scaling that could not exist in a real computer built from the fastest components.
4. holding one part of the computation constant while scaling other parts to test different communication paradigms and faster circuitry.

### 1.5.2 The Mainframe and Minicomputer Industries

Enslow [5] chronicled the early evolution of mPs and enumerated their disadvantages, including

- the complexity, expense, and difficulty of system software
- the hardware-software interaction
- the small and decreasing incremental gain in performance with each processor, hardware limitations for building extensive (> four) processor configurations
- the inability of software to exploit the hardware

Organizational inertia of large, old-line companies was a large factor in the poor adoption of mPs even though such companies were the first to introduce the mP. The inertia comes from having well established, unimaginative large groups and charters within engineering, fear of parallelism (including misunderstanding about memory conflict) and a quarter-century old operating system structure, including its maintainers, that defies change. With a growing market and a large engineering budget, it was always easier to start a new uniprocessor project with another technology (e.g., CMOS, TTL, ECL, GaAs) than to base one on using an existing processor to build a more complex mP. With physically large processors, the problem and cost of switching also made the mP more difficult to build and the gain harder to achieve.

With the introduction of the cache memory, the problem of memory coherence became large and further inhibited development for some time. Finally, in looking back, the myths about mPs (the second one increases performance 60 to 80%, the third 30 to 50% and there is no increase in performance by the fourth processor) failed to understand and segment the various effects, e.g., memory conflict, resource management, locks. Most systems were memory bandwidth-limited because of relatively large and expensive memory modules. More memory bandwidth (i.e., more modules) added significant access delays that a single stream processor was unable to overcome. Strecker's thesis [6] showed that such a system could be built provided enough memory bandwidth was available. Given the increased latency, a way to compensate for the delays would have been to build multistream processors (like the 6600 PPUs or Dennelcor HEP), but these structures would have appeared too radical and risky to mainframe engineers.

## 1.6 Why Build Multiprocessors Now?

Two basic market forces "pull" the introduction of mPs: performance and higher availability. Two technology forces "push" their inevitable adoption: the inherently lower cost of designing and building mPs that yield better performance/price and technology. Exhibit 1.5 shows these forces and factors.

### 1.6.1 Higher Performance and Greater Workload Throughput

Workload. The evolution of the Cray supercomputers [3] (Exhibit 1.4) illustrates the inevitability of mPs. Improvements in clock performance are relatively small and the basic forms of parallelism (pipelining and vector processing) have been extracted for a single job instruction stream. Therefore to increase performance mPs have to be adopted. Since supercomputers are time-shared and serve a large number of users in a multiprogrammed fashion, any increase in computing resources directly improves system effectiveness. Thus a mP provides more power through multiprocessed multiprogramming since each processor works independently on the workload stored in common memory. As previously suggested, unless structures such as the multicomputer can be used efficiently for multiprogramming, the economics will continue to relegate them to specialized tasks.

Although the "Cray" is used in both batch and time-shared modes, a mP is even more applicable for transaction processing because all the users are independent of one another and there is a high likelihood that all the users, such as those in a reservations system, will share common programs and data, i.e., the file system.

Peak Power. Peak power can be made available to a single user through both explicit and implicit forms of parallel processing. In 1987 and 1988, the applications that operated at the highest rates and greatest parallelism used the Cray XMP/416 and YMP8/32 supercomputers [4] operating at over 50% efficiency. Given that most computers use UNIX, the pipes mechanism provides the most

[3]The highest performance computer(s) used for scientific and engineering computation.

[4]Each year the Bell Prize is for parallelism in supercomputers, mPs, multicomputers, and compilers. In 1987, a 1024-node Ncube multicomputer won, operating at almost 100% efficiency.

- ☐ *Higher performance and greater workload throughput*
  - -- A large workload can be processed via multiprogrammed operation of a common job queue and common file system. Greater processing is available through
    - batch processing
    - time-shared, multiprogrammed server to LAN-based, X-windows clients and
    - arbitrarily large transaction-processing capacity with lower overhead (i.e., less context switching) than with a uniprocessor or multicomputer approach.
  - -- Peak performance is available to a single user through transparent parallel processing of
    - multiple jobs
    - multiple, pipelined processes
    - automatic (i.e., compiled) microtasking of blocks
  - -- Peak performance is available via explicit parallel processing for multiple processes.
  - -- Scalability. Incremental growth in performance with incremental growth in price is maintained. Computing power is provided in an incremental and scalable fashion as one computer.
  - -- A common pool of computational resources (processing, primary and secondary memory, networking) is available to trade-off among a large job mix.
- ☐ *Lower costs, providing improved performance/price*
  - -- A small set of PMS-level components forms a complete product line of a range of products in a scalable fashion.
  - -- Fewer component types imply
    - lower production cost and higher quality
    - better service through better spares availability and training
    - lower engineering costs and higher quality
- ☐ *Evolvability with time*
  - -- A well-designed system is evolvable with technology in an asynchronous fashion over several technology generations of processing, primary and secondary memory, and network.
- ☐ *Fault-tolerance and high availability*
  - -- Inherent redundancy using few component types provides higher system availability.
  - -- Nonstop hardware can be built (e.g., Stratus) in an incremental fashion.
- ☐ *Technology forces*
  - -- The mP is an evolutionary form of the computer, generalizing the processing resource in the same fashion as memory and input-output (e.g., terminals).
  - -- Properly designed operating systems provide mP parallelism as a by-product.
  - -- With a somewhat standard operating system, UNIX, the design of operating systems was taken away from engineers. Engineers have to create something, thus they are able to focus on mP versions of UNIX versus building another operating system.
  - -- Parallel processing has evolved to keep up with the evolution of hardware (e.g., processors).
  - -- Modem microprocessors (e.g., MIPS, Motorola) are designed to support multis, and as such it is becoming increasingly difficult to build uniprocessors.
  - -- For the largest machines, fast uniprocessors are technology-limited by scalar performance. mPs are the only way to provide higher throughput and peak power for general-purpose (i.e. "dusty deck") use.

**EXHIBIT 1.5**
Market and Technology Forces for Multiprocessors

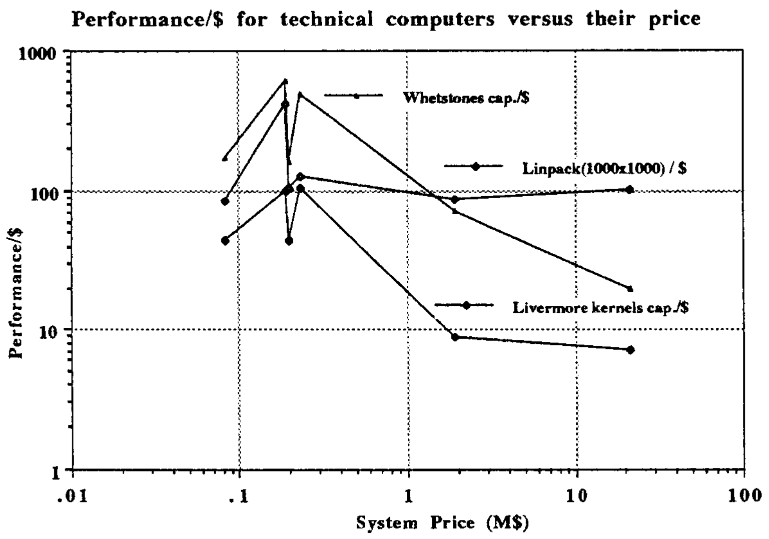

**EXHIBIT 1.6**
Performance characteristics measured by several common benchmarks are plotted against the price of various uni-, multi-, and vector multi-processor computers. The benchmarks are: Linpack running fully parallel, characterizing the best case for a vector mP, Whetstones as the worst case scalar application, and Livermore Kernels which typify a workload of technical computing.

basic form of parallelism, which though explicitly specified by a user, is identical to the uniprocessor case and is therefore used implicitly. With implicit parallelism, the compiler and operating system are responsible for detecting and exploiting the parallelism. The three forms of parallelism are shown in Exhibit @1.8. By explicitly segmenting a program into processes for parallel execution using primitives such as fork and join, a user can further exploit multiple processes.

The proof of the concept is in the machines that form the leading edge of technical computing. Exhibits 1.6, 1.7, and 1.8 compare various performance metrics and performance/price metrics versus the price of six leading edge technical computers.

Several observations can be made from these exhibits:

- Large-scale, highly parallel applications (parallel Linpack) can exploit the peak performance of vector mPs.
- Even for vectorizable applications, having a larger-scale problem (i.e., 1000 × 1000) is essential for high performance. Only on this problem do the

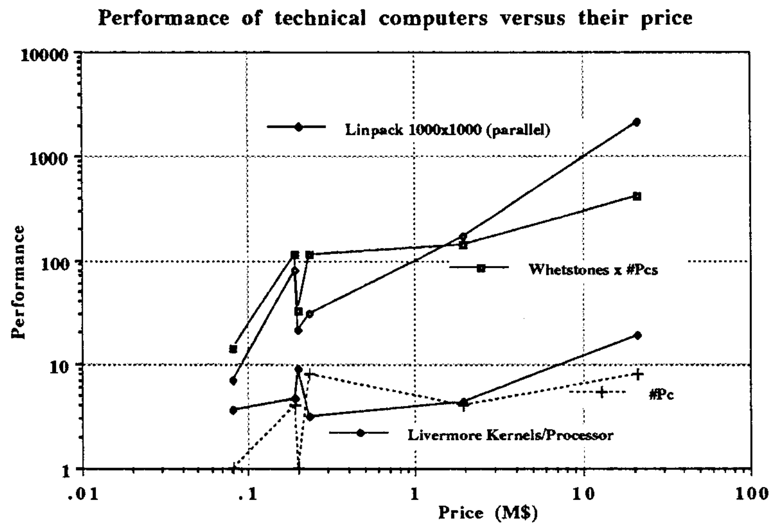

**EXHIBIT 1.7**
Performance per price characteristics measured by several common benchmarks are plotted against the price of various uni-, multi-, and vector multi-processor computers.

multiple, vector processors attain their peak.

- Vector processing, which is a basic parallel construct allowing parallelism to be exploited, doesn't help all applications. A totally scalar application characterized by Whetstones puts a large computer at a significant disadvantage.
- A workload mix of scalar and vector characterized by the Livermore Kernels runs best on the fastest scalar machine. The MIPS R6000 performs half as well as the Cray YMP, which corresponds directly to the clock difference.
- A vector processor improves the operation rate of a workload by only 50% since the amount of code that can be vectorized is limited.
- Fast, Reduced-Instruction Set microprocessors (RISCs) are the basis of the most cost-effective computers. The first four systems are based on the MIPS, Inc. RISC chip. The MIPS 6000 is an ECL microprocessor.
- No economy of scale exists across the range and lower-priced machines are more cost-effective. This reflects using CMOS micros versus ECL technology.

| Computer | Mhz | CPUs | Year | Peak Perf. | Memory | Price(M$) |
|---|---|---|---|---|---|---|
| MIPS 3000 | 25 | 1 | 1988 | 12 | 4 Mw | 0.082 |
| MIPS 6000 | 75 | 1 | 1990 | 38 | 8 Mw | 0.200 |
| SGI 280 | 25 | 8 | 1989 | 100 | 8 Mw | 0.230 |
| Stardent 3000 | 32 | 4v | 1989 | 128 | 8 Mw | 0.190 |
| Convex C240 | 25 | 4v | 1988 | 200 | 16 Mw | 1.900 |
| Cray YMP | 167 | 8v | 1987 | 2667 | 64 Mw | 21.000 |

**EXHIBIT 1.8**
Price and performance metrics for six leading edge technical computers.

- For all but two of the benchmarks, the MIPS 6000 is not as cost-effective as the other MIPS chip-based mP computers. In other words, going faster costs more per operation.
- Several scalar processors (e.g., the Silicon Graphics SGI 280) can be used in parallel for vector processing tasks. Roughly four processors are needed to equal one vector processor for highly vectorized applications.
- The SGI 280 is a classic, eight-processor multi and is most cost-effective for a scalar workload provided the workload does not saturate its low speed bus.
- The most cost-effective use of a mP is usually to run N programs on N processors.

Incremental Growth in Performance with Price. The main advantage of mPs can be seen in how the Cray YMP is constructed to cover a range of price and performance (Exhibit 1.9) in a scalable fashion.

### 1.6.2 Lower Costs, Providing Improved Performance/Price

The Cray YMP line of up to eight processors is constructed in three models: one to two, two to four, and five to eight processors. All use the same basic packaging and power scheme, and common processor and memory modules. The difference in the system comes from building three basic packages that provide various amounts of memory bandwidth depending on the number of processors.

While the pricing is on the basis of processors, the individual processors each have a very small cost. Thus a disproportionate amount of the cost is in the basic package to house and interconnect (i.e., switch) the processors and memory.

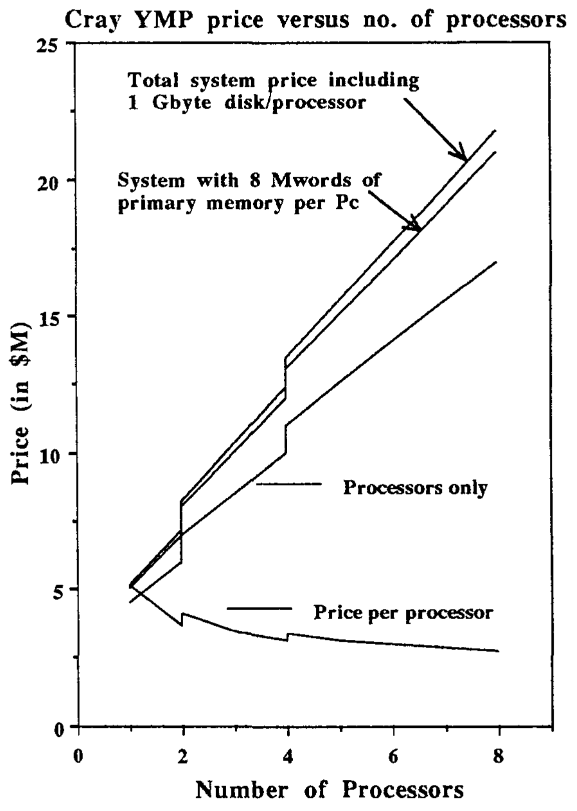

**EXHIBIT 1.9**
Cray YMP price is plotted for one to eight processors.

Nevertheless, Cray is able to use only a few basic components to provide a range of computing power over a range of price and cost.

Fewer Components Reduces Costs. The main benefit to such a scheme comes directly from the learning curve concept. With a learning curve, the cost of a product declines in relationship to the cumulative number of units produced. The amount of learning amounts to a 10 to 20% cost reduction for each doubling of produced units. This would amount to cost differentials of 50% versus comparable uniprocessor families. The author would expect the reliability of units to increase in a similar fashion.

| Model | Date | Cache | Performance range (vs. VAX 11/780) |
|---|---|---|---|
| 120 | 1985 | 32K | 1.5 to 15 |
| 320 | 1987 | 64K | 4.0 to 40 |
| 500 | 1989 | 256K | 17.0 to 170 |

**EXHIBIT 1.10**
Evolution of Multimax product line.

Engineering. Finally, having a few components requires a smaller engineering effort and this alone should produce a higher quality product. While the author believes that the product quality is inversely proportion to the number of people on a project, having a smaller number of systems to engineer will increase the quality provided that all the people who would have been working on alternative systems are not used on the design part.

### 1.6.3 Evolvability with Time

Exhibit 1.10 shows how the Encore Multimax product line evolved with time by just replacing the processor module with the next-generation microprocessor. [5]

Thus, a principle benefit of the multi approach is evolvability. The key to evolution has been the evolution in cache size with time as provided by improved semiconductor density. Models 120 and 320 used a write-through protocol to maintain cache coherence. In Model 500, the cache protocol was changed to write-back on reference so as not to require a bus transaction each time a word needed to be written in memory. Note that only a few mainframe and supercomputers could deliver comparable performance, and none could equal the performance/price.

### 1.6.4 Fault-Tolerance and High Availability

The basic mP provides inherent higher availability than its uniprocessor counterpart by the redundancy of components (i.e., processors, memory, and i/o). In all mPs, system operation is possible with any number of the components being unavailable.

[5]The processing module has two processors. Up to 20 processors can exist in one system. The bus that interconnects processors and memory transfers 64-bit words at a 12.5 Mhz rate (100 Mbytes/sec).

Stratus used the multi structure to provide fault-tolerant computing by constructing a computer with two pair of redundant elements. Each pair votes on a memory by memory reference, and if an error occurs, the other redundant pair is used for the result. Two copies of memory with appropriate error detection and correction are maintained. By having nonfaulting hardware, the software operating system task is significantly simplified.

### 1.6.5 Technology Forces

The mP is merely a generalization of the uniprocessor, providing incremental processing resources in the same way that a computer can have more memory or more terminals. By proper design, the mP introduces no significant design problems to an operating system that are not inherent in a general-purpose, multiprogrammed system, such as UNIX. The standardization of UNIX enabled software engineers to extend an existing system rather than building one from the scratch and thus to avoid the possibility of never getting around to support mPs.

Today, it is virtually impossible to avoid building multis simply because, after a decade of harassing microprocessor suppliers, the support for mPs exist in nearly every micro. Furthermore, a large number of system engineers have mP experience and simply apply market pressure. Finally, as a uniprocessor hits a performance limit, the mP is the only way to provide more performance in an evolutionary fashion.

## 1.7 What Are the Design and Use Issues?

Multiprocessors face various design and use issues in the next decade at all the levels of integration, many of which may be common to other structures (hardware, language, algorithm) employing increasing parallelism.

How are mPs with increasingly large amounts of parallelism measured, and how efficient are they? Before one can embark on design, it is necessary to have a way of measuring the system being designed because metrics should have the greatest effect on the design. Benchmarks and kernels have historically been used to characterize how a computer will perform on a particular user's workload. As machines with more parallelism are constructed, older, small-scale benchmarks fail because a very large, parallel processor is unlikely to run them efficiently and hence is rejected a priori. Thus problem scale is a new parameter in understanding performance. Having a range of benchmarks that correspond to the range of algorithms a machine is likely to encounter is equally important because parallel machines have many new opportunities to stumble, and understanding potential bottlenecks is equally important. If a good way to characterize performance could be found, it would both improve computer designs and at the same time save customers and vendors great expense over the benchmarking that every user and vendor goes through today. SPEC, Illinois (the Perfect Club), the Lawrence Laboratories (Livermore and Los Alamos), the National Aeronautics and Space Administration (NASA), and even the newspaper *Digital Review* are all attempting to improve this critical situation.

What is the design goal of the system in terms of its size and the range of scalability and evolvability with technology and time? The right metrics for all the system design parameters must be specified from the start [4]. Today's systems have to be more effective in all forms of serial and parallel integer and floating point computation, memory management, and all forms of i/o for file management, interactive graphics, and all forms of visualization.

For example, the author posited 11 rules for supercomputer design [4] to reflect the complexity of design today.

1. Performance, performance, and performance are the three objective criteria for a supercomputer design. Rules 2 to 6 relate to performance.
2. Amdahl's law generalized implies that everything matters, a variant of "no chain is stronger than its weakest link," especially when measuring links by harmonic mean.
3. The scalar speed matters most and a super must be the fastest of comparable computers in its class, otherwise the harmonic mean measurement kills it as a super.
4. The vector speed can be arbitrarily high as costs allow. This is the advertised speed of the computer. The vector speed (peak) or advertised speed is the speed that the manufacturer guarantees the computer will not exceed on any application. The past rule of thumb is to have a vector unit that will produce two results per clock tick. IBM's mainframes have vector speeds about four times their scalar speeds. Large increases over the scalar speed beyond a hundred-fold provide a small benefit except for selected, very large scale applications, making the computer special-purpose (e.g., a Connection Machine). The recent NEC SX-3 announcement for a super has a peak speed of 16 times the clock.
5. Allow no holes in the performance space (e.g., arithmetic function, i/o, mass storage) into which a benchmark can step, resulting in large performance losses.
6. Provide peaks in the performance space in order that extraordinary performance for a benchmark will result. Use this single number to advertize (characterize) the machine and to challenge other machines.
7. Obey computer design law no. 1: Provide enough address bits for a decade of constant architecture implementation.
8. Build at least two generations of the architecture. No first design supercom-

puter has ever been perfect. Do it again after the first one.

9. Build on the work of others. Designing a super is hard. Understand exactly why and how every machine works and move forward using this knowledge and any residual software.

10. Make it easy to use. Have a great compiler and diagnostic tools to aid users in vectorization and parallelization. [6]

11. Have slack resources when embarking on a supercomputer design. The fatality rate for companies making machines is at least 50%, and even though a design may be good, it has to be reiterated. Building a new super costs a minimum of $50 million to get a breadboard.

How scalable can a computer be? Scalability is a term that has been introduced into computer architecture in the last decade to connote building a range of computers from a common set of components. While scalability has increased, it is unclear that any computer that will scale effectively over an order of magnitude is possible. We have three scalable computer structures: mPs, microcomputers, and SIMDs (with many processing elements). The mPs are interesting over a range of 1 to 8 today (i.e., the largest machines are from 1 to 8 times the size of the smallest). Multicomputers built from weak, i.e., low performance components start performing competitively at 64 to 128 processors and a few have been built with 1024 elements for a range of 8 to 16. The Connection Machine is offered over a range of only 8 (8K to 64K processing elements). Will subsequent, hierarchical mPs be useful, i.e., operate economically over a wider range and hence be more scalable? More than likely the company supplying a computer limits scalability more than the computer might because each class of computer operates with different cost margins. It is unlikely that any company could be organized to sell a range of products that include low priced minis to large supers covering a $50K to $20M range (or 400:1).

In addition, one must ask, what are the parts and how many are there, how are they interconnected, how do they perform, and how reliable is the resulting structure? This is the PMS level design problem. Small-scale designs are bus-based and supers and mainframes use several levels of switching to supply high memory bandwidth to the processors. For the hundreds of processors that by their nature are advanced machines and that have the potential to advance the state of the art, the alteratives are: the big switch which for the "dance hall" model (processors, usually with both cache and significant local memory, on one side of the switch and memories on the other); or a hierarchy of clusters of computers

[6]Training for supers is missing in academe since computer science departments are not oriented to training people to use computers or deal with computers that produce numbers. No texts exist on programming a parallel, vector processor (i.e., supercomputers), for example, and only few texts address parallelism in general. Only a few universities offer courses to teach supercomputer use and architecture.

such as Cm*, Cedar, E&S 1, KSR, and Ultramax.

How is memory kept coherent? Closely associated with PMS design is the problem of maintaining memory coherence when caches are used. For bus based multi design, the problem is simplest because all parts of the system can easily communicate with one another. The single bus and the cache of the multi serves three functions: caches provide the processors with fast access memory, caches avoid using precious system bandwidth, and the bus allows caches to "snoop" automatically as memory data are written so as to maintain overall memory coherence. As more complex switches are used, the caches become less accessible to one another and hence harder to keep in synchronization with one another except by explicit software control. Supercomputers do not use caches, but they should in order to build systems that are not limited by the "dance hall" switch capacity. A by-product is lower cost.

What are the Instruction Set Processor (ISP) architectural issues for mPs? The ISP design aspects of mPs have been fairly simple in the past with little support outside of basic memory locking primitives and facilities to aide processes to share common parts of memory. With large mPs, reducing the overhead for parallel processing requires operations on shared memory variables such as arithmetic on memory. For example, the Stardent computer has "do loop" variables in memory, which each processor accesses for its indices. Alliant goes further by locking variables that could depend on one another in a loop. The most interesting ISP is the Kendall Square Research computer that segments instructions into memory (including access, locking, data movement) and computation for separate parallel execution. Whether scalars, superscalars, or vector processors are used is a relatively small detail in the design of a large mP, because the overall ISP choice is so highly dependent on the overall design goals and especially the nature of the workload of the computer.

How do you build the hardware? Implementation follows from the previous PMS level decisions and the technology. The challenges today are having the right components (e.g., processors and switch and memory controllers) such that the implementation exploits them without undue overhead. For example, the Stardent processor and 32 or 128 Mbyte memory occupy board area of 2 x 19 inches by 22 inches using off the shelf components, whereas a KSR processor/memory module with the same amount of memory occupies roughly 150 square inches; ~~both operate at roughly the same speed~~. Customization of silicon or GaAs beyond gate arrays is the key to building very large mPs.

the KSR processor/memory pair delivers roughly one-half the workload capacity of a Cray-1.

How do you build computers that don't fail? The implementation of highly reliable computers continues to be an art unto itself. With more recent microprocessors the hardware design of multis for high availability is straightforward and commonplace even for ordinary computers. Operating systems can then evolve accordingly.

What does the software look like to manage mPs? Operating systems have evolved to support parallelism in the language operating environments for small-scale mPs. For mPs with several hundred processors, a different approach may be needed. Today, the leading candidate for such systems is CMU's MACH. Standard extensions such as the Argonne library are needed in all operating systems to provide a common set of primitives for all parallel programming environments to aid portability and simplify training (and texts). To exploit mPs effectively all forms of control for implicit and explicit parallelism are needed, including message passing.

What is the programming environment, including languages, debugging, and profiling tools, that encourage the exploitation of parallelism? Fortran is the language for technical computing but C is becoming common and standard extensions for vectorization and parallelization are needed. For commercial and transaction systems, database systems and C are an alternative to traditional Cobol. Whether traditional languages provide the right paradigm for parallelism is a moot question. Today's environments are all we have, and a research and development community that could develop alternatives is nonexistent.

How do users exploit mPs? By far the most significant limit to mP development is training. Training cannot occur on a wide scale basis until computer science and engineering get involved in parallelism to use, learn, teach, and write texts. Also, until this happens, no alternative to parallelism outside the evolutionary stream will occur.

## 1.8 Conclusions

mPs have been slow to evolve and become a staple of computer engineering. More progress has been made in the last 5 years than ever before because all classes of computers are built in this fashion. The next decade should be the most exciting in terms of challenges for builders and users alike. It is clear that we will be able to build mPs with thousands of processors, but the progress will be difficult because of the complexity of these designs and the difficulty users have in exploiting them for all but transaction processing and embarrassingly parallel programs.

## References

[1] C. G. Bell and A. Newell. *Computers Structures: Readings and Examples*. New York: McGraw-Hill, 1971.

[2] C. G. Bell. Multis: A new class of multiprocessor computers. *Science*, *228*, 462-467, 1985.

[3] C. G. Bell. The 11 rules of supercomputer design. Videotape, University Video Communications, Los Gatos, CA, 1989.

[4] C. G. Bell, C. Mudge, and J. McNamara. *Computer Engineering*. Digital Press, 1978.

[5] P. H. Enslow, ed. *Multiprocessors and Parallel Processing*. John Wiley & Sons, 1974.

[6] W. D. Strecker. An analysis of the instruction execution rate in certain computing structures. Ph.D. dissertation, Carnegie Mellon University, 1971.

[7] W. A. Wulf and C. G. Bell. C.mmp --- a multi-mini-processor. *Proceedings of the Fall Joint Computer Conference*, 1972.