BRINGING MPP
TO THE MAINSTREAM CUSTOMER

BY BOB MASSON
CONVEX COMPUTER CORPORATION

WHITE PAPER SERIES

ABSTRACT

Using a large number of commodity microprocessors as a basis for high-performance computer systems seems to hold the promise of solving supercomputer-class problems more affordably than ever before, while at the same time increasing by at least 100-fold the range of problem sizes that can be tackled. But the road to applying Massively Parallel Processing (MPP) to everyday applications is fraught with twisty curves and arduous ascents. At least two companies offering MPP technology have come and gone recently, and the industry press[1] has begun to question the rationale of producing systems that may be complicated to use and inefficient [9]. Why, then, are computer manufacturers scrambling to produce these systems, and who is going to use them?

In this paper we will examine the promises and potential of MPP systems, some of the reasons why such systems are being developed, and offer reasons why customers would be willing to purchase them. In addition, we will contrast the Convex vision of providing a supercomputer-class problem-solving tool to the everyday "mainstream" computer user, with the realities of deploying MPP technology. Insights are provided into the company's plans of "bringing MPP to the mainstream customer."

---

[1]The reader is referred to a number of recent articles in the press: "In Supercomputing, Superconfusion," *Business Week*, March 22, 1993; "Never the twain shall meet? : IS poised to harness the power of parallel processing," *Information Week*, January 4, 1993; "Massively parallel processors win attention, but how about sales?" Electronic News, July 6, 1992; "Parallel Processing, is it all promises?" *Computing Canada*, January 20, 1992.

MULTIPROCESSOR COMPUTER SYSTEMS

There are a wide variety of technologies for applying multiple microprocessors to increase the performance of a computer system. There are different base architectures, different processor interconnect schemes, and different programming models. To further complicate matters, systems employing multiple processors are able to provide different *types* of performance for different applications. As we will explore later, these different types of performance have a major bearing on how a consumer is able to efficiently use an MPP system.

It is not our intent to provide a tutorial on computer architectures. What we are interested in is how these computer systems may be used as tools to solve everyday problems for what we call the "mainstream" computer user.

MPP ARCHITECTURES

MPP is an acronym for Massively Parallel Processing. This definition has generally been used to describe systems that use "hundreds or thousands of processors," presumably on a single problem. However, things have gotten confusing of late; the "M" in MPP has also been used to stand for "Modest," "Moderate," and "Masochistic."[2] For our purposes it is less important for us to be concerned about the number of processors as it is what work may be accomplished with those processors.

MPP systems are generally classified by the method in which they process data and instructions (that is, in singular or multiple data or instruction streams). As shown in Table 1, there are four combinations of ways of processing instructions and data.

TABLE 1. DIFFERENT METHODS OF PROCESSING INSTRUCTIONS AND DATA IN MPP SYSTEMS.

SISD	Single-instruction, single-data
SIMD	Single-instruction, multiple-data
MISD	Multiple instruction, single-data
MIMD	Multiple-instruction, multiple-data

The two most common architectures are SIMD and MIMD. There are tradeoffs with each, but the MIMD architecture seems to be enjoying the most popularity lately. This is largely due to the flexibility possible with a MIMD model; multiple CPUs in the system are able to independently execute different pieces of code.

Besides the method that is used to process instructions and data, there is the view of the machine that the application developer sees when migrating a program to or developing a new program on an MPP system (the programming model). Many MPP systems today require the programmer to make a radical change in the way the machine is viewed, leading to a steep learning curve and slow migration of existing programs to the new architecture.

SMP ARCHITECTURES

It is useful to describe here another model of multiprocessor system, that of Symmetric Multiprocessing, or SMP. Symmetric multiprocessing involves using a number of processors (probably not a "massive" number, for reasons we will describe) to increase aggregate throughput of a computer system. For the most part, SMP systems are not used for parallelization of individual jobs, although that trend appears to be changing with the acceptance of *de facto* standard message-passing parallelization tools. [12]

Most multiprocessing computer systems in use today are used in an SMP environment. There are many benefits to having a number of

[2]Steve Wallach, a founder of Convex, refers to greater than 1,000 processors as "masochistic."

processors sharing the same cabinet, let alone the same memory subsystem. (This is in contrast to an equivalent number of uniprocessor workstations distributed throughout an organization.)

For example, the system may be kept more constantly busy than a number of workstations. In addition, by connecting the processors to a common memory system and I/O subsystem, the processors may share peripherals, eliminating redundancy and wasted disk space.

THE POTENTIAL

Before delving into the intricacies (or even definition) of MPP, it is useful to relate the promises that appear on the horizon for this new technology. It has been said that MPP will revolutionize the computing industry the way microcomputers did in the early 1980s [8]. This may be true, but the keyword here is "revolutionize."

We describe a revolutionary product change as one that requires the consumer to fundamentally change the way she or he does business. The majority of consumers are loath to product revolutions. For example, when was the last time that there was a successful revolution in, say, the "user interface" of automatic automobile transmissions? The author recalls a push-button automatic transmission on a 1960's Plymouth, which, while very high technology, was rather slow to be adopted by the general public (to say the least).

Both fortunately and unfortunately, computer systems consumers have grown up with drastic revolutions in their technology. *Fortunately*, because this has led to tremendous advances in the power of computing tools to solve everyday problems. *Unfortunately*, because every new "revolutionary" technology reverberates throughout the consumer's organization, lowering productivity and increasing costs (at least at first).

So, in true high technology tradition, let's look at the promises of this revolutionary new technology of massively parallel processing, and discuss the pitfalls later on in our discourse. Also in this tradition, we will use terms such as "performance" in a nebulous fashion; remember, our intent is to be optimistic in the extreme.

PRICE/PERFORMANCE

The overwhelming advantage of MPP systems is that they promise unprecedented performance for the price. In a fall 1992 survey of high-performance computer systems users, price/performance was ranked the number one criterion considered in the selection of a high-performance system [10]. Additionally, out of 50 respondents, 37 ranked this criterion as 4 or 5 on an importance scale of 1-5 (5 being most important).

In a June 1992 issue of *Datamation* and the research firm of Cowen & Co. concluded that "although capacity requirements [of supercomputer sites] are doubling every 2.5 years, corresponding budgets will grow only 5 to 10% over the long term." [2]

MPP systems definitely have the potential of satisfying the need of more performance for less price, especially compared with the mainframes and traditional supercomputers today. Figure 1 illustrates the improvement in price/performance for several types of computer processors over about the last decade. Here, we have used the 100x100 LINPACK benchmark [5] on a single processor as a performance indicator, knowing this may not be representative of "performance." Our intent is to illustrate that price/performance has improved dramatically over these years for *at least* one type of problem.

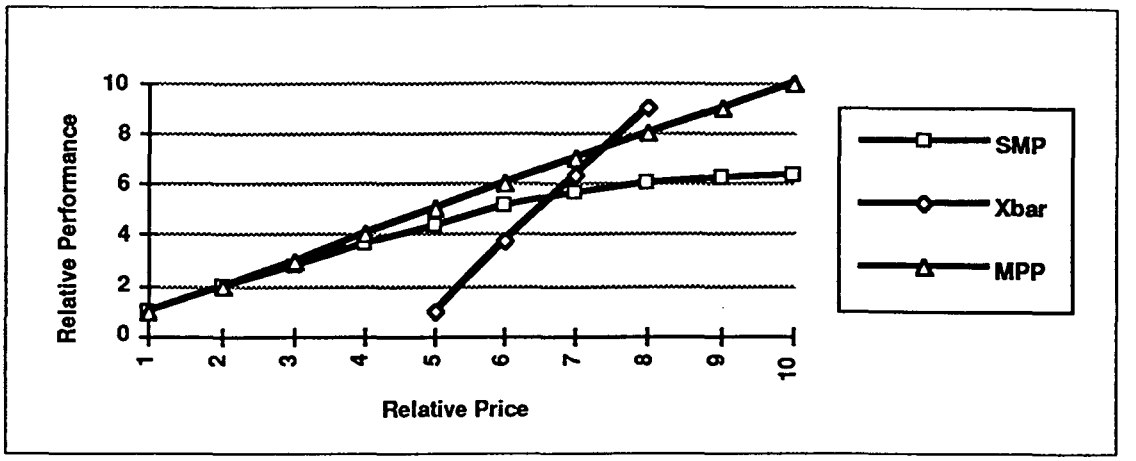

FIGURE 2. RELATIVE SCALABILITY FOR A NUMBER OF ARCHITECTURES.

Further investment protection is promised in MPP systems by means of inter-generational scalability. Given that microprocessor generations improve in performance quite quickly, some architectures allow next-generation CPUs to be added that coexist with the previous generation(s), or are easily replaced at the chip level; in effect, a system with an extra-long lifespan.

TERAFLOPS AND TERABYTES

Probably the most important promise of MPP is the ability to reach levels of performance that would otherwise be impossible with single-processor systems; indeed, it is generally acknowledged that an MPP architecture will eventually be *required* to get additional performance out of a computer system. Computer CPU clock cycles are ultimately limited by the physical speed of electrons through a conductor. The fastest computer systems today are in the 300-500 MHz (2-3 ns cycle) range. Even a single order of magnitude of performance increase (excluding such factors as pipelining and superscalar features) would require 3-5 GHz processors! Until the electromechanical issues of cooling a computer system that is roughly a 1-inch cube are conquered, these systems are impossible. Figure 3 plots the clock cycles of three classes of computer systems since 1975; it is obvious that the performance of single CPUs is fast approaching some physical limit.

In January of 1992 President Bush signed the High Performance Computing and Communications (HPCC) Act of 1991. The HPCC implements a five-year, $3 billion program intended to further America's economic competitiveness and lead in computing technologies. One of the four project areas is development of computer systems capable of performing trillions of floating operations per second (TFLOPS). These computer systems are intended to address a series of grand challenges in science and engineering. Defined as "fundamental problems with potentially broad economic, political and/or scientific impact," these grand challenges include climate modeling, superconductivity, vehicle design, and plasma dynamics for fusion technology [7].

We will spare the reader another incarnation of the "Grand Challenge" graphic. It is

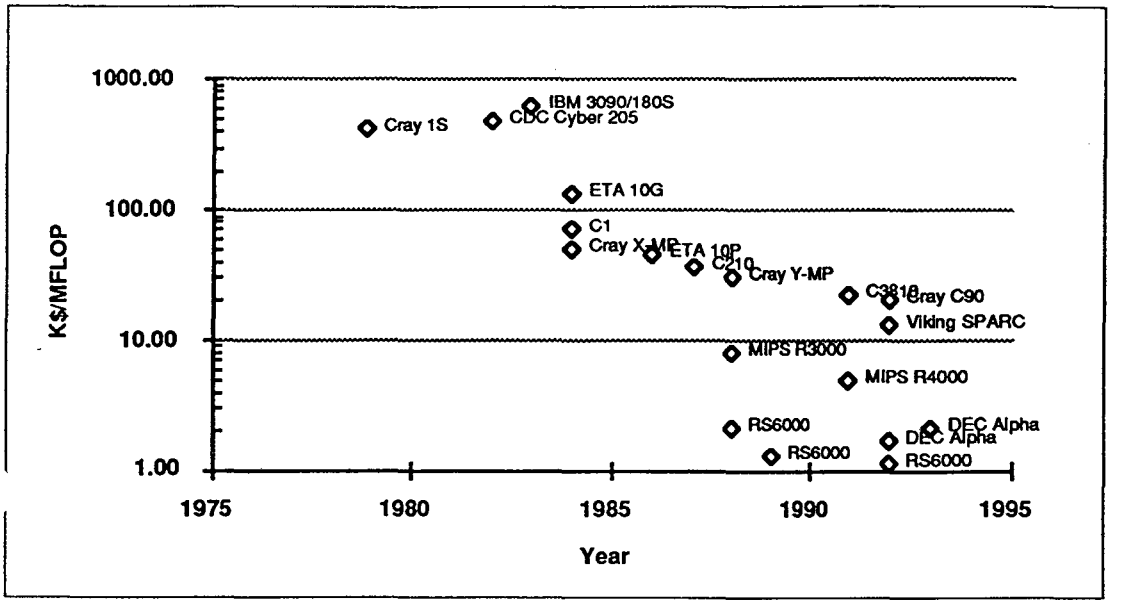

FIGURE 1. ILLUSTRATION OF THE IMPROVEMENT IN PRICE/PERFORMANCE FOR THE 100x100 LINPACK BENCHMARK.

SCALABILITY

Another important benefit of MPP systems is their *scalability*. For our uses, scalability is "the property of a computer system to be expanded without introducing a fundamental bottleneck." [3] In the not-too-distant past (arguably, now!), a computer purchaser would have to trade-in his old system and acquire the next "generation" system — just for performance gains. This is the ultimate in "non-scalability."

In a way, scalability is closely related to price/performance, since it involves protecting your investment as you expand your computer system. A truly scalable system will permit incremental performance at a consistent price (excluding the effects in price or performance due to changes in the technology of the incremental components). For example, bus-based multiprocessing systems typically allow incremental processors to be added to the base system, but at a declining price/performance, since each incremental processor will provide diminishing levels of performance as the bus becomes saturated with traffic.

Most supercomputers, which have more robust memory systems (usually some variant of a crossbar), allow incremental processors to be added without the effects of saturating the memory system. However, in these systems, the base configuration (and thence the price!) usually already has the memory subsystem for the maximum allowable number of processors. In very general terms, Figure 2 illustrates curves of relative scalability for a number of architectures.

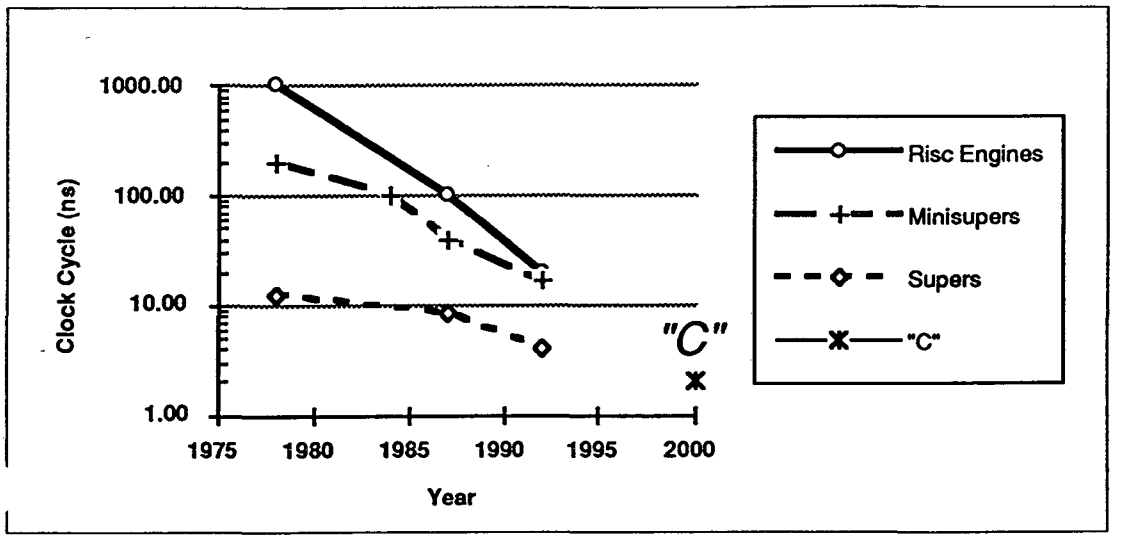

FIGURE 3. PROCESSOR CLOCK CYCLES HAVE DIMINISHED OVER TIME, BUT WILL EVENTUALLY BE LIMITED BY THE LAWS OF PHYSICS.

sufficient to state that, for reasons cited earlier, there is a set of problems that will not be solvable any other way than with MPP technology. As defined, these problems are so large that they require TeraFLOPS (10^{12} floating operations per second) of performance and Terabytes (10^{12} bytes) of storage. Let's quickly put that in perspective: the current high-end Cray system, the Y-MP C90, has a peak theoretical performance of 16 GFLOPS (give or take a FLOP). Selling for $30,000,000 each, we would have to buy 62.5 full-up Cray C90's ($1,875,000,000 worth of computers) to achieve one TeraFLOP of *peak* performance, clearly something outside of the normal industrial computer hardware budget.

Having said that, it is the author's observation that the majority of problems being solved by prospective MPP users are *not* grand challenge problems; rather it is the everyday problems faced by engineers and scientists that reflect a more pragmatic nature of scientific computing.[3] Hence, the promise of MPP performance gets down to the *potential* of achieving TFLOP performance, even if actually acquiring it is something of a pipe dream.

Open systems

To a lesser extent, one of the promises of MPP systems is that they are based on commodity microprocessors available from major semiconductor or computer systems vendors. As a result, they should be able to execute programs targeted for these microprocessors (commonly called "open systems," since they generally run some derivative of UNIX and accept reasonably portable code). To date, however, none of the current MPP vendors actually achieve this promise, even when using off-the-shelf microprocessors.

The goal is worth noting, however, since it promises dramatic savings in development costs, application porting, and retraining of users and system administrators.

[3]Based on fifteen years of observing the type of customer that purchases high performance systems.

MPP MARKET

We will sum this section up with some market projections from the industry pundits. Almost all projections show the market for MPP systems growing at a rate faster than the rest of the computer industry. However, we will also preface this with a quote from IDC (International Data Corporation):

> "If parallel suppliers can effectively demonstrate available third-party applications and clear approaches to integrating their products into production environments, then the anticipated strong growth for this technology [MPP] will materialize. If users, after careful evaluation of this technology, decide that its practical implementation is still "another five years off," then it will be difficult to maintain current levels of market interest and growth." [6]

In other words, if all of this works, people will buy it, if not, they won't! Having said that, in Table 2 we present a couple of the market analysts' forecasts for the MPP "market" in the coming years.

TABLE 2. PUNDIT PROJECTIONS FOR THE MPP MARKETPLACE, 1992-1996 (IN MILLIONS).

	1992	1993	1994	1995	1996
IDC	263	300			
Dataquest	290	405	515	653	825
Smaby Group					911
SPCS				500	

MPP PITFALLS

Having reviewed the glowing promises of MPP systems, we must now come down to earth with the realities of successfully exploiting such technology. While not inclusive, the "problems" that are described in succeeding paragraphs represent the majority of economical and emotional resistance to promoting MPP systems.

PROGRAMMING MODELS

We refer to the view of the computer system that an application developer sees as the "programming model." The simplest programming model is that of a single processor connected to a single memory system. In such a single-instruction single-data (SISD) system, instructions are executed singly, and the programmer barely has to worry about the specifics of the hardware.

With MPP systems, it is often necessary to know where memory is "located" (in relation to the stream of instructions that are being executed). The memory may be attached to a processor, and any references to data beyond that next to the processor must be explicitly stated in the program. Alternatively, the memory may be distributed throughout the machine but appear to be accessible by all programs executing within the system.

In addition to being concerned about memory, the MPP programmer has to worry about the distribution of the different paths of code that he is attempting to parallelize. Somehow, all of these "threads" must be dispatched to different processors and assigned the proper data to execute. Furthermore, the different threads must synchronize their execution and their reference to data that is shared amongst all the threads.

All of these problems face the user of an MPP system today. This is one of the primary pitfalls of MPP technology, and leads to another pitfall: that of application availability.

THIRD-PARTY APPLICATIONS AVAILABILITY

Many computer systems users rely on software that is produced elsewhere and sold or leased to the user. Vendors of such software, often called

independent software vendors (ISVs), have the most difficult time with new architectures (or even with multiple implementations of similar architectures!). Their continuing battle is to produce quality software that runs on the widest range of platforms available, thus increasing sales possibilities.

Whenever a new architecture is introduced in the computer industry, the ISV will port his code to this new system when it appears that there is a substantial potential for sales; in other words, when there is a fairly large number of systems employing the new architecture installed. Since a large percentage of users of high-performance computer systems rely on third-party software,[4] a "Catch-22" situation arises [3]. The third-party code developer is not economically motivated to port his product to the new architecture (because there are not many installed), and the computer user cannot justify purchasing the new machine because the third-party application doesn't execute on the new architecture!

This situation introduces a considerable amount of resistance to the general acceptance of a new architecture. As we will explore later, the only practical way of breaking this Catch-22 situation is to provide tools that mask the idiosyncrasies of the new technology while letting the benefits shine through.

Limitations of Parallelization

MPP systems promise performance gains by employing parallelism, or the ability to execute portions of an application at the same time on many processors. But how realistic is this promise?

In 1967, G. Amdahl proposed that the performance of a computer system is ultimately limited by the speed of its slowest component [1]. In the context of MPP systems, the performance of the application will be limited by the serial, or non-parallel portions of the code. Assuming perfect parallelization (completely unrealistic, of course), the execution time of an application is represented by the non-parallelizable portion of the code plus the parallelizable portion. The effects of Amdahl's law are surprising, and result in the conclusion that a program must be *highly* parallelizable to be able to take advantage of any substantial number of processors. Figure 4

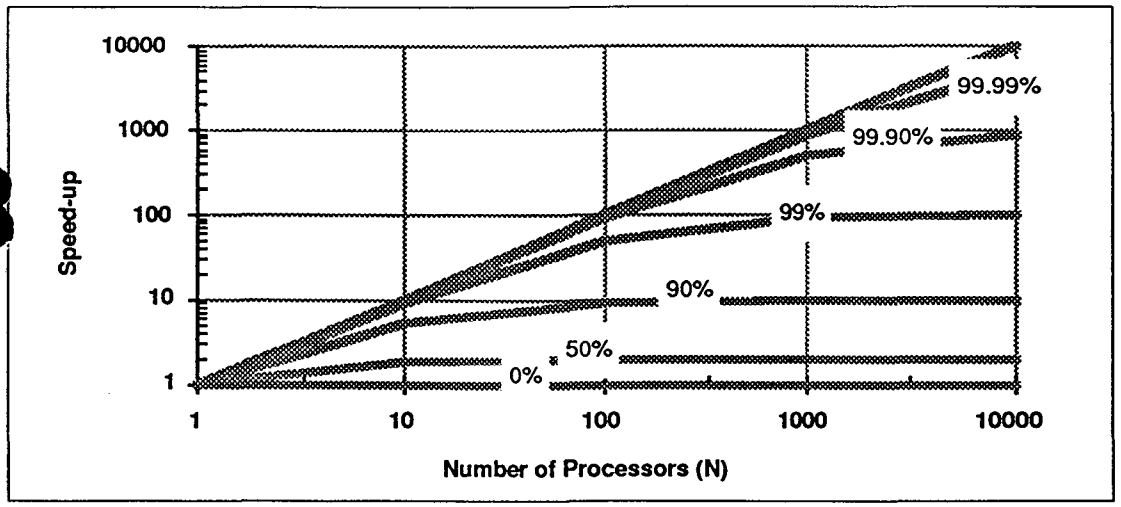

Figure 4. The amount of speed-up for different percentages of parallelization as determined by Amdahl's Law.

[4]The majority of Convex systems that are shipped are sold in conjunction with some third party application. Indeed, Convex maintains a substantial staff of developers that assist these vendors in porting and tuning their applications for the Convex architecture.

illustrates the performance speed-ups that are theoretically attainable for different percentages of ideal parallelization. For example, if 90% of the execution time of an application my be reduced by parallelization, the speedup only approaches 10X, regardless of the number of processors! Furthermore, after 100 processors have been utilized, the remaining processors are used very inefficiently.[5]

The point of this exercise is that we must be careful with the MPP promise of performance, regardless of the underlying architecture.

MPP FOR THE MAINSTREAM

It's been said that today's computer manufacturers have the impression of "build it and they will come" [11] when pursuing the market for MPP machines. They (in this case, application developers) *will* come if these promises, reviewed below, are born out. However, we have examined many of the pitfalls of MPP, and we know it will be some time before many of these obstacles are overcome.

TABLE 3. THE PROMISES AND PITFALLS OF MPP.

Promises
• Price/performance
• TFLOPS and Tbytes
• Scalability
• Common Open Systems Environment
Pitfalls
• Programming models
• Architectural standards
• Algorithm parallelization and performance
• Language standardization
• Applications availability

Given these promises and pitfalls, how then will the mainstream customer benefit from this new technology?

THE MAINSTREAM CUSTOMER

First, let us revisit the definition of "mainstream customer." We have stated that this customer is one who is more concerned with getting his job done than with the specifics of the tools he uses; a "production-oriented" computer user might be a more descriptive synonym.

Another way of viewing the mainstream customer is his reaction to revolutionary technology (as defined earlier). A model that is commonly used is that of the "technology adoption curve" as shown in Figure 5. The curve represents the different reactions to new technology — from energetic early adopter to the technology laggard. We picture the mainstream customer as the center of the bell curve; it happens that statistically this represents the majority of consumers.

The task of "bringing MPP to the mainstream customer" involves living up to as many of the promises of MPP as possible, while solving, circumventing, or hiding as many of the pitfalls as possible. The Convex approach to balancing these requirements, and ultimately placing MPP technology into the hands of the everyday customer, is described in the following paragraphs.[6]

SPP (SCALABLE PARALLEL PROCESSING)

First, we need to define a new term. Convex refers to its RISC multiprocessor machine as an SPP — Scalable Parallel Processor. SPP is a computer system that has the software environment and ease-of-use of a general-purpose computer, with the architecture and performance of a massively parallel processing system. This redefinition is necessary for a number of reasons:

- The term MPP is ambiguous today (as we have discussed),
- The implementation of the Convex machine is a superset of the traditional MIMD parallel processing model,

[5]For an expansion of this example and further examples, the reader is urged to refer to G. Astfalk, "Fundamentals and practicalities of MPP."

[6]Whether it is a "true" MPP machine is left to the interpretation of the reader. We merely acknowledge that the system is scalable and composed of 10's to 100's of processors, which adheres to our earlier definition of "MPP."

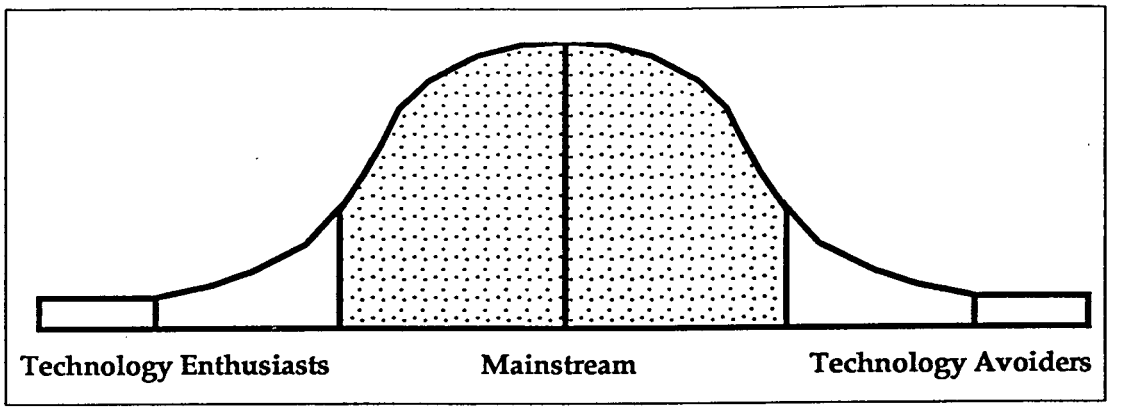

FIGURE 5. THE TECHNOLOGY ADOPTION CURVE.

- The programming model of the machine lends itself to being viewed as a number of MPP implementations, as well as the traditional shared-memory SMP machine,
- Because it is geared to the mainstream customer, the Convex SPP can be viewed as the next step in the evolution of MPP.

The Convex SPP machine is intended to satisfy the promises of MPP systems as we have described them, while still maintaining the look and feel of a contemporary workstation or SMP system. As illustrated in Figure 6 this is accomplished by "borrowing" many of the architectural features of today's MPP systems and combining them with the programming model, ease-of-use, and electromechanical technologies of workstations.[7]

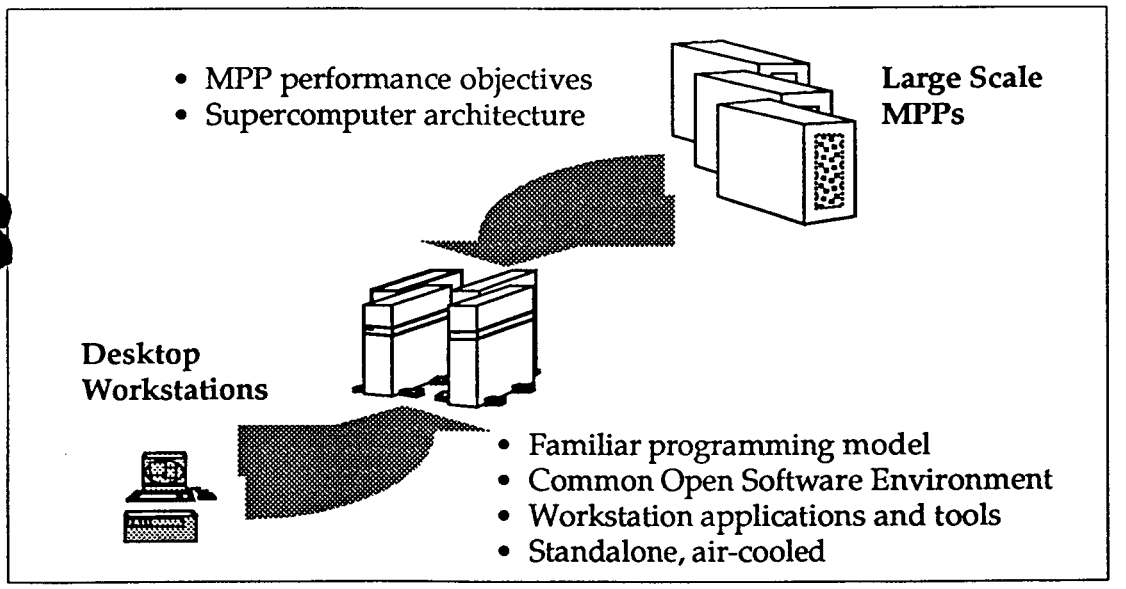

FIGURE 6. THE POSITIONING OF THE CONVEX SPP SYSTEM.

[7]Something Convex has done before; when the first Convex product was shipped in 1984 it married the supercomputer architecture of the Cray with the usability of the DEC VAX.

The result is a system that utilizes many commodity microprocessors, yet has the programming and user interface "look and feel" of a workstation. This is accomplished through a combination of hardware and software features.

KEY FEATURES OF THE CONVEX SPP SYSTEM

PROGRAMMING MODEL

The basis for the usability of the Convex SPP system lies in the programming model, or the view of the system as seen from the code developer. As was discussed earlier, the plethora of different parallel computer architectures is one of the pitfalls of using an MPP machine, and leads us to the Catch-22 of providing "standard" third party application codes on the system.

The Convex system mitigates the Catch-22 situation by providing several options to the application developer. The first option is to simply recompile existing codes and let the compiler detect whatever parallelization it is able. This option is most useful for the many "utility" programs an installation often has; these codes are run only occasionally and any extensive effort to optimize them is unwarranted. The result, however, is that the application will at least execute on the machine, and further optimizations may be carried out as necessary. The parallelization with this option is known as shared memory parallel, or SM.[8] SM is most often seen today on tightly-coupled shared-memory systems such as the Convex C Series and the Cray Y-MP systems. The Convex FORTRAN and C compilers will attempt to generate SM code (when parallelization is requested). SM typically results in fairly fine-grained parallelization (for example, at the loop level).

The second option is the support of programs that explicitly support parallelization through message passing (MP). Several *de facto* standards exist today for MP programming; the most common seems to be PVM (Parallel Virtual Machine). Additionally, there are several proposed language standards for the MP programming model that Convex has committed to supporting.[9] The use of the MP programming model through PVM is that the application will often execute on a variety of platforms,[10] including the Convex Meta Series, which may be used as a development platform for these parallel codes. The two options may be combined to provide two levels of parallelism within an application. Figure 7 illustrates that the SPP architecture provides performance for a wide range of parallelism.

PERFORMANCE

Performance is one key to the successful utilization of an MPP system. As we discussed earlier, precious few applications exist today which will take advantage of ten processors, let alone hundreds! To that end, the Convex SPP system supports a variety of features to provide performance.

First, the system is a good symmetric multiprocessing system in its own right. That means that the system will provide throughput performance for a mixture of parallel and not-so-parallel jobs.

Second, the operating system supports a scheduling mechanism whereby all of the system's processors are divided into sub-complexes. Each sub-complex is scheduled and time-shared independently; the result is the ability to schedule jobs on sub-complexes that match the parallelization of each job. For example, a sub-complex may be defined that consists of eight processors that accept general timeshared traffic. Thus, 20 or 30 users may timeshare on these eight processors, while the system's remaining processors (say, 40 of them) are allocated to highly parallel tasks such as seismic processing.

[8] This is also known as data parallel on SIMD machines; we wish to distinguish the shared memory data parallel approach from that of SIMD DP.

[9] Currently, HPF (High Performance FORTRAN) appears to be the most popular choice for a set of extensions to the FORTRAN 77 standard language. HPF supports directives to permit MP parallelization.

[10] PVM version 3.0 is alleged to run on at least thirty different platforms.

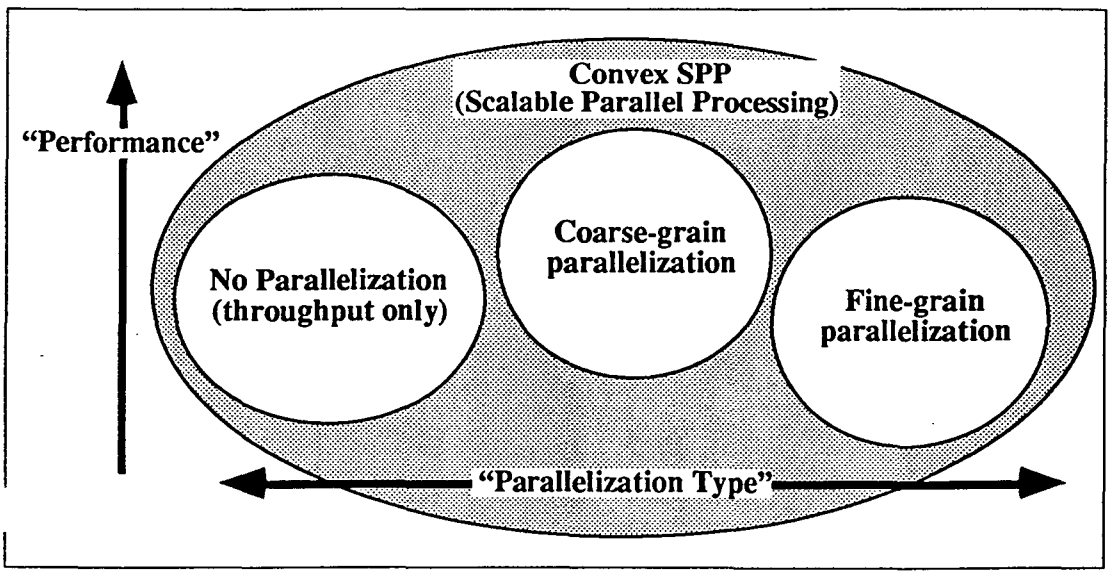

FIGURE 7. THE SPP SYSTEM PROVIDES A PROGRAMMING MODEL THAT SUPPORTS DIFFERENT LEVELS OF PARALLELIZATION.

The scalability of this scheduling mechanism is supported through a distributed microkernel operating system, in this case, the Open System Foundation's OSF1 Mach microkernel. The microkernel provides system services to every processor, but requires very little overhead. Because of its distributed nature, the microkernel performance scales with the number of processors in the system.

HP/UX ABI COMPATIBILITY

Earlier we discussed code portability and the benefits of adhering to open systems standards. The Convex SPP operating system contains a layer that permits the user-level and application interfaces to execute HP/UX binary executables. As a result, the entire suite of HP/UX "middleware" is available to a user of the Convex SPP system. Among other things, this permits any program that uses the HP/UX system calls of this layer to execute unmodified on the SPP system (obviously, unless recompiled, the executable will not utilize more than a single processor!). The layers of the Convex SPP operating system are illustrated in Figure 8.

Probably the most important feature of this compatibility is the standardization that is realized. In April of 1993, an unprecedented agreement was announced by Hewlett-Packard, IBM, The Santa Cruz Operation (SCO), SunSoft, Univel and UNIX System Laboratories Inc. (USL). The announcement indicated that these companies would agree upon, and produce, a Common Open Software Environment (COSE), encompassing the adoption of specifications and standards in the areas shown in Table 4.

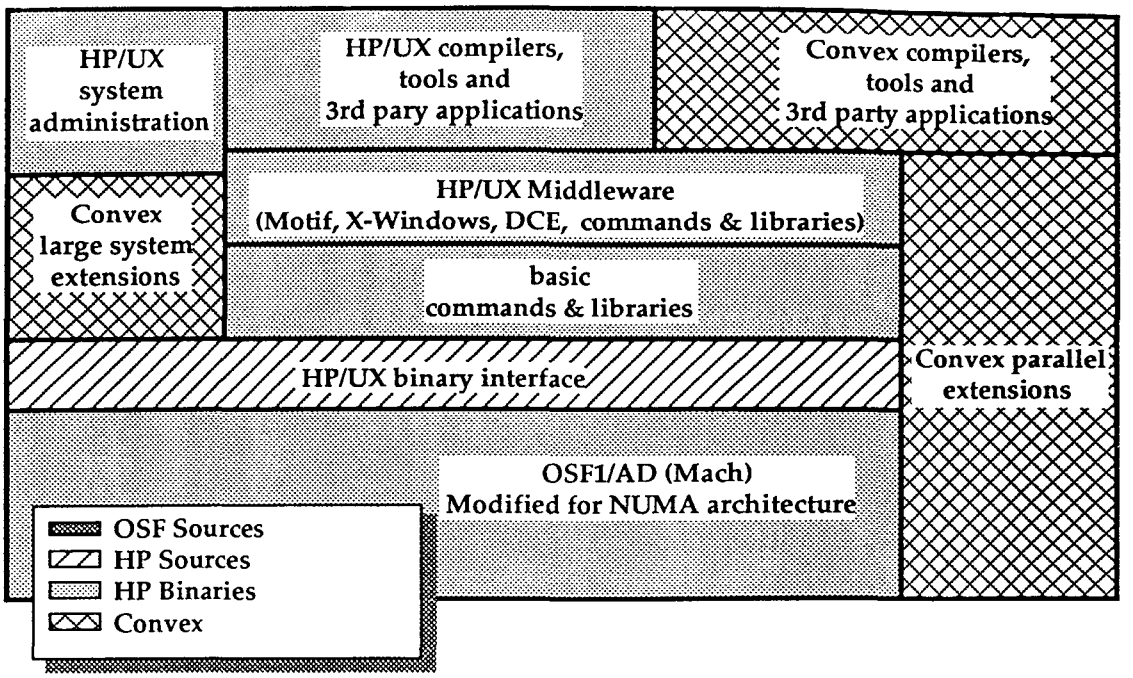

FIGURE 8. THE LAYERS OF THE CONVEX SPP OPERATING SYSTEM.

TABLE 4. AREAS OF STANDARDIZATION PROMISED BY COSE.

• Common Desktop Environment
• Networking
• Graphics
• Multimedia
• Distributed Object Technology
• Systems Management

As a result of this agreement, and the HP/UX ABI compatibility of the SPP operating system, the SPP user will realize the benefits of the COSE agreement.

SCALABILITY

Scalability of the Convex SPP system is provided through the distributed operating system (discussed earlier) and the implementation of the memory subsystem. The system is based on a series of nodes, each node consisting of up to eight HP PA-RISC microprocessors and up to two gigabytes of memory. The nodes are connected by a fast, low-latency ring interconnect. The result is a hybrid memory system of a crossbar for internode performance, and a high performance, cache-coherent intranode connection for scalability. Figure 9 depicts a system block diagram for a system with sixteen nodes (128 processors).

AFFORDABILITY

Of course, what literary piece would be complete without a mention of the affordability of such a system? The use of commodity microprocessors, which are sold in huge quantities for the desktop, leverages an economy of scale for the CPU portion of the system. That, combined with the use of other off-the-shelf components (such as memory),

allows the system to track the price/ performance of the PA-RISC microprocessor. Of course, the system will not be as inexpensive as the equivalent number of PA-RISC-based workstations; this is offset by many of the features we have described. (Of course, if you have but one application, and it runs merrily on a stack of workstations connected by Ethernet, the acquisition of *any* hardware beyond that necessary would be folly!)

Metacomputing

The SPP system we have described is a component of an overall system envisioned by Convex, that of the metacomputer [4]. The metacomputer empowers users by providing them with the ability to dispatch a task with no thought given to the platform(s) on which it will ultimately be executed. The metacomputer vision employs heterogeneous elements connected by a high-speed network that appear to be one system, with software that maximizes users' efficiency.

Summary

Computer systems will necessarily evolve into those that contain many general-purpose processors, with a programming model that permits the appropriate performance to be extracted from these processors.

In the end, the mainstream customer simply wants to get the job done, without regard to the intricacies of the interior of the system. While not a reality today, over time hardware and software will eliminate many of the MPP pitfalls we have highlighted, and the computer industry (and computer system consumers!) will have survived yet another technology revolution.

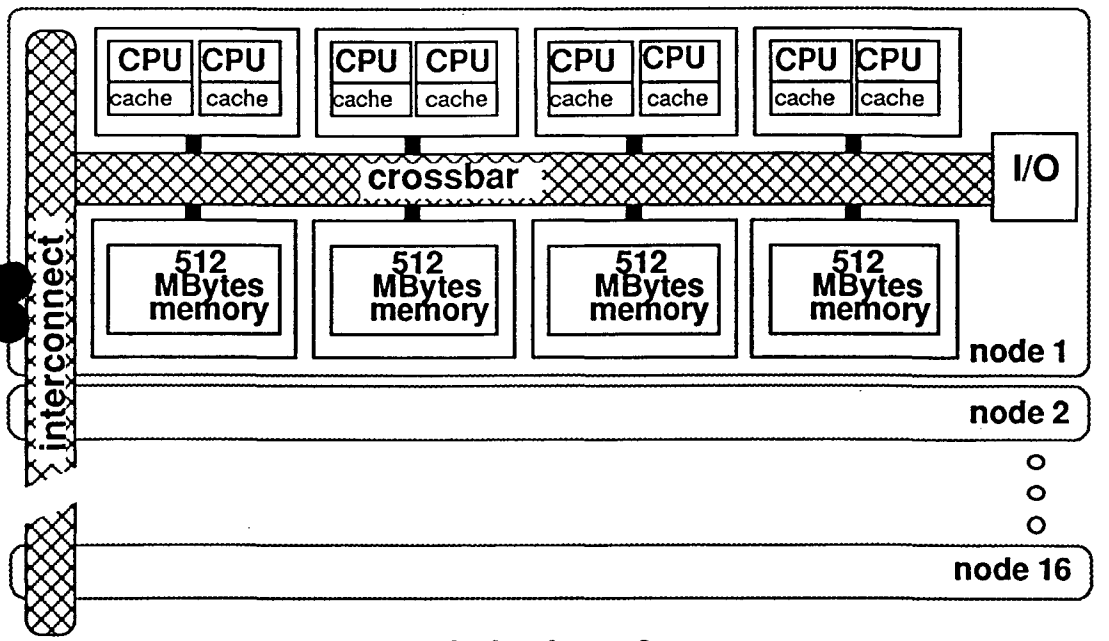

FIGURE 9. SPP SYSTEM OVERVIEW.

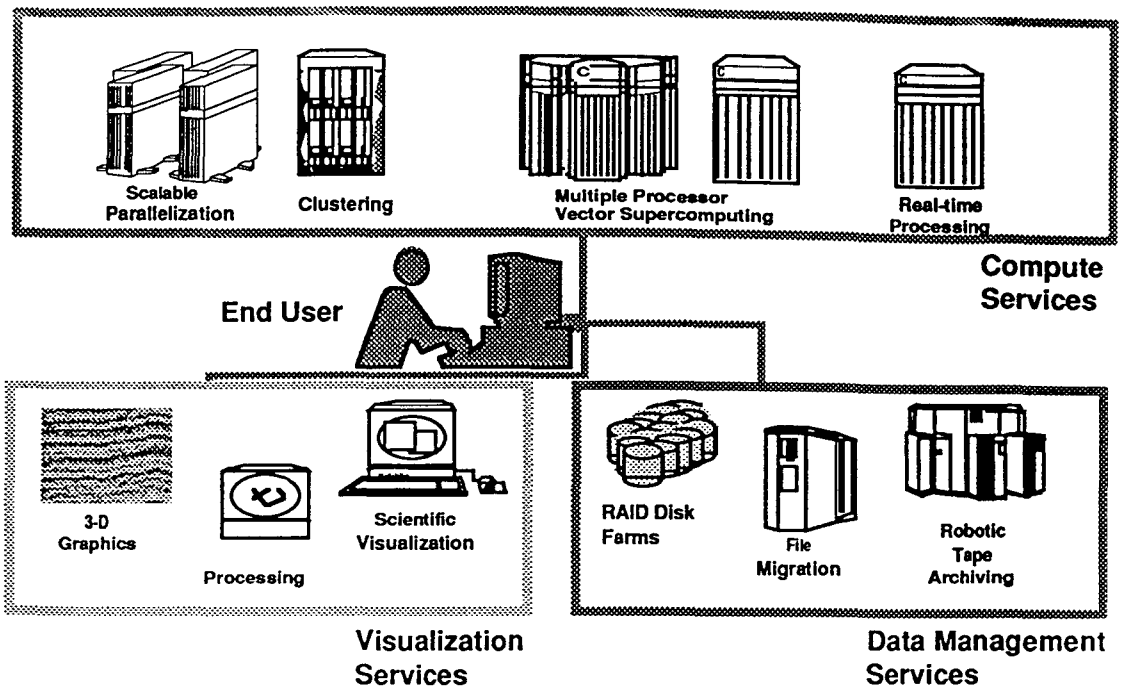

FIGURE 10. THE CONVEX METACOMPUTING VISION.

REFERENCES

[1] G. AMDAHL, *Validity of the single-processor approach to achieving large scale computing capabilities*, in Proceedings of the AFIPS Conference, Atlantic City, NJ, AFIPS Press, Reston, 1967, pp. 483-485.

[2] E. APPLETON, "Massive solutions for massive problems," *Datamation*, June 1 1992.

[3] G. ASTFALK, "Fundamentals and Practicalities of MPP," to be published.

[4] C. CATLETT, "Metacomputing, a network of heterogeneous, computational resources linked by software in a way that makes them easy to use," Communications of the ACM, June 1992.

[5] J. DONGERRA, "Performance of various computers using standard linear equations software," University of Tennessee, Knoxville, TN CS-89-85 May, 1992.

[6] D. GOLDFARB AND C. WILLARD, "Worldwide High-Performance Computing Market: 1992 Review and Outlook," International Data Corporation Analysis, March, 1993.

[7] *Grand Challenges: High Performance Computing and Communications*, available from the Office of Science and Technology Policy, National Science Foundation, 1800 G Street, NW, Washington DC 20550.

[8] J. MARKOFF, "Foray into Mainstream for Parallel Computing," *New York Times*, June 15, 1992.

[9] S. MCCARTHY, "Vector processing is second to none," *Government Computer News*, September, 1992.

[10] RCI, Ltd., 1992 RCI European HPC User Survey, Fourth Quarter 1992.

[11] S. VARNEY, "MPP systems set to tackle complex problems," *Digital Review*, June 8, 1992.

[12] F. VITALIANO, "Cut through massively parallel clutter," *Digital News & Review*, October 12, 1992.

For more information or for the location of your local Convex sales office, contact:

U.S.A. CORPORATE HEADQUARTERS

Convex Computer Corporation
3000 Waterview Parkway
P.O. Box 833851
Richardson, Texas 75083-3851
Phone: 214/497-4000

EUROPEAN HEADQUARTERS

Convex Computer, Ltd.
Randalls Research Park
Randalls Way
Leatherhead
Surrey, U.K. KT22 7TS
Phone: 44-372-386696

FAR EAST

Convex Computer PTE, Ltd.
1 Scotts Road
#25-06 Shaw Centre
Singapore 0922
Phone: 65-733-4355

AUSTRALIA

Convex Computer PTY, Ltd.
Level 20, Como Office Tower
644 Chapel Street
South Yarra, Victoria 3141
Phone: 61-3-823-6216

Printed in U.S.A.
080-002251-000