Overview of Recent Supercomputers

Aad J. van der Steen *
and
Jack J. Dongarra †

Abstract

In this report we give an overview of parallel- and vector computers which are currently available or will become available within a short time frame from vendors; no attempt is made to list all machines that are still in the research phase. The machines are described according to their architectural class. Shared- and distributed memory SIMD- and MIMD machines are discerned. The information about each machine is kept as compact as possible. Moreover, no attempt is made to quote prices as these are often even more elusive than the performance of a system. This document reflects the technical state of the supercomputer arena as accurately as possible. However, the authors nor their employers take any responsibility for errors or mistakes in this document. We encourage anyone who has comments or remarks on the contents to inform us, so we can improve this work. [1]

*Academic Computing Centre Utrecht, PO Box 80.011, 3508 TA Utrecht, The Netherlands, actstea@cc.ruu.nl

†Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301, and Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, TN 37831

[1]NCF, the National Computing Facilities Foundation, supports and furthers the advancement of technical and scientific research with and into advanced computing facilities and prepares for the Netherlands national supercomputing policy. Advanced computing facilities are multi-processor vectorcomputers, massively parallel computingsystems of various architectures and concepts and advanced networking facilities.

Contents

1 Introduction and account

This is the sixth edition of a report in which we attempt to give an overview of parallel- and vector systems that are commercially available or are expected to become available within a short time frame (typically a few months to half a year). We choose the expression "attempt" deliberately because the market of parallel- and vector machines is highly evasive: the rate with which systems are introduced — and disappear again — is very high and therefore the information will probably be only approximately valid. Nevertheless, we think that such an overview will be useful for those who want to obtain a general idea about the various means by which these systems strive at high performance, especially when it is updated on a regular basis.

We will try to be as up-to-date and compact as possible and on these grounds we think there is a place for this report. The present report will be somewhat shorter than earlier ones: at this moment systems are disappearing at a faster rate than new ones replace them. The reasons for this seem to be threefold:

- The competition is very fierce and only companies that can offer up-to-date systems, both hardware and software wise can keep in business.
- Generally, less money is available worldwide for purchasing new high performance systems oriented to scientific and technical computing. This makes life more difficult for both existing companies and for potential starters.
- Because of price/performance considerations the number of companies offering systems with custom-made processors is decreasing because they cannot capitalise on large volume sales as is the case with RISC processor based systems.

These effects make the high-performance computing scene somewhat more clear (and also somewhat less adventurous). Still, the supercomputer market is very dynamic and we too, cannot hope give a complete report for the reason already mentioned above: the speed with which companies and systems appear and disappear makes this almost impossible. However, by updating the report we can at least follow the main trends in popular and emerging architectures.

The rules for including systems in this report are as follows: they should be either available commercially at the time of appearance of this report, or within 6 months thereafter. This is to avoid confusion by describing systems that are announced much too early, just for marketing reasons and that will not be available to general users within a reasonable time. We also have to refrain from including all generations of a system that are still in use. Therefore, for instance, we do not include the Convex C3000 series, the Cray Y-MP series, or the Thinking Machines CM-5 anymore although these systems are still used widely. Generally speaking, we include machines that are still marketed or will be marketed within 6 months. In this issue we add for the first time Wold Wide Web addresses of vendors. The information on the Web pages of the vendors may be more recent than what can be provided in this report. On the other hand such pages should be read with care because it will not always be clear what the status is of the products decscribed there.

We order the systems by their various architectural classes, which should facilitate to find the information of systems that belong to a certain class. We also omit the price information which in most cases is next to useless. If available, we will give some information about performances of systems based on user experiences instead of only giving theoretical peak performances. Here we have adhered to the following policy: We try to quote *best measured performances*, if available, thus providing a more realistic upper bound than the theoretical peak performance. We hardly have to say that the speed range of supercomputers is enormous, so also the best measured performance will not always reflect the performance of a reader's favorite application. When we give performance information, it is not always possible to quote all sources and in any case if this information seems (or is) biassed, this is entirely the responsibility of the author of this report. He is quite willing to be corrected or to receive additional information from anyone who is in the position to do so.

Before giving a recount of the systems proper, we first define the architectural classes and some other terms in section 2 which will be used in section 3 in the description of the machines. In section 4 some systems are listed that disappeared from the market and in section 5 we present some systems that are under development and have a fair chance to appear on the market.

The overview given in this report concentrates on the computational capabilities of the systems discussed. To do full justice to all assets of present days high-performance computers one should list their I/O performance and their connectivity possibilities as well. However, the possible permutations of configurations even for one model of a certain system often are so large that they would multiply the volume of this report, which we tried to limit for greater clarity. So, not all features of the systems discussed will be present. Still we think (and certainly hope) that the impressions obtained from the entries of the individual machines may be useful to many. We also omitted some systems that may be characterised as "high-performance" in the fields of database management, real-time computing, or visualisation. Therefore, as we try to give an overview for the area of general scientific and technical computing, systems that are primarily meant for database retrieval like the AT&T GIS systems or concentrate exclusively on the real-time user community, like Concurrent Computing Systems, are not discussed in this report.

Although most terms will be familiar to many readers, we still think it is worthwhile to give some of the definitions in section 2 because some authors tend to give a meaning that may slightly differ from the idea the reader already has acquired.

2 The Main Architectural Classes

Since many years the taxonomy of Flynn [5] has proven to be useful for the classification of high-performance computers. This classification is based on the way of manipulating of instruction- and data streams and comprises four main architectural classes. We will first briefly sketch these classes and afterwards fill in some details when each of the classes are described separately.

- **SISD** machines: These are the conventional systems that contain one CPU and hence can accommodate one instruction stream that is executed serially. Nowadays many large mainframes may have more than one CPU but each of these execute instruction streams that are unrelated. Therefore, such systems still should be regarded as (a couple of) SISD machines acting on different data spaces. Examples of SISD machines are for instance most workstations like those of DEC, Hewlett-Packard, and Sun Microsystems. The definition of SISD machines is given here for completeness' sake. We will not discuss this type of machines in this report.
- **SIMD** machines: Such systems often have a large number of processing units, ranging from 1,024 to 16,384 that all may execute the same instruction on different data in lock-step. So, a single instruction manipulates many data items in parallel. Examples of SIMD machines in this class are the CPP DAP Gamma and the MasPar MP-2.
- Another subclass of the SIMD systems are the vectorprocessors. Vectorprocessors act on arrays of similar data rather than on single data items using specially structured CPUs. When data can be manipulated by these vector units, results can be delivered with a rate of one, two and — in special cases — of three per clock cycle (a clock cycle being defined as the basic internal unit of time for the system). So, vector processors execute on their data in an almost parallel way but only when executing in vector mode. In this case they are several times faster than when executing in conventional scalar mode. For practical purposes vectorprocessors are therefore mostly regarded as SIMD machines. Examples of such systems are for instance the Convex C410, and the Hitachi S3600.
- **MISD** machines: Theoretically in these type of machines multiple instructions should act on a single stream of data. As yet no practical machine in this class has been constructed nor are such systems easily to conceive. We will disregard them in the following discussions.
- **MIMD** machines: These machines execute several instruction streams in parallel on different data. The difference with the multi-processor SISD machines mentioned above lies in the fact that the instructions and data are related because they represent different parts of the same task to be executed. So, MIMD systems may run many sub-tasks in parallel in order to shorten the time-to-solution for the main task to be executed. There is a large variety of MIMD systems and especially in this class the Flynn taxonomy proves to be not fully adequate for the classification of systems. Systems that behave very differently like a four-processor Cray Y-MP T94 and a thousand processor nCUBE 3 fall both in this

class. In the following we will make another important distinction between classes of systems and treat them accordingly.

- **Shared memory systems**: Shared memory systems have multiple CPUs all of which share the same address space. This means that the knowledge of where data is stored is of no concern to the user as there is only one memory accessed by all CPUs on an equal basis. Shared memory systems can be both SIMD or MIMD. Single-CPU vector processors can be regarded as an example of the former, while the multi-CPU models of these machines are examples of the latter. We will sometimes use the abbreviations SM-SIMD and SM-MIMD for the two subclasses.
- **Distributed memory systems**: In this case each CPU has its own associated memory. The CPUs are connected by some network and may exchange data between their respective memories when required. In contrast to shared memory machines the user must be aware of the location of the data in the local memories and will have to move or distribute these data explicitly when needed. Again, distributed memory systems may be either SIMD or MIMD. The first class of SIMD systems mentioned which operate in lock step, all have distributed memories associated to the processors. For the distributed memory MIMD systems again a subdivision is possible: those in which the processors are connected in a fixed topology and those in which the topology is flexible and may vary from task to task. For the distributed memory systems we will sometimes use DM-SIMD and DM-MIMD to indicate the two subclasses.

Although the difference between shared- and distributed memory machines seems clear cut, this is not always entirely the case from user's point of view. For instance, the late Kendall Square Research systems employed the idea of "virtual shared memory" on a hardware level. Virtual shared memory can also be simulated at the programming level: The first draft proposal for High Performance Fortran (HPF) was published in November 1992 [6] which by means of compiler directives distributes the data over the available processors. The proposal was fixed by May 1993. Therefore, the system on which HPF is implemented will act in this case as a shared memory machine to the user. Other vendors of Massively Parallel Processing systems (the buzz-word MPP systems is fashionable here), like Convex and Cray, also support proprietary virtual shared-memory programming models which means that these physically distributed memory systems, by virtue of the programming model, logically will behave as shared memory systems. In addition, packages like TreadMarks [1] provide a virtual shared memory environment for networks of workstations.

Another trend that has came up in the last few years is *distributed processing*. This takes the DM-MIMD concept one step further: instead of many integrated processors in one or several boxes, workstations, mainframes, etc., are connected by Ethernet, FDDI, or otherwise and set to work concurrently on tasks in the same program. Conceptually, this is not different from DM-MIMD computing, but the communication between processors is often orders of magnitude slower. Many packages to realise distributed computing, commercial, and non-commercial are available. Examples of these are

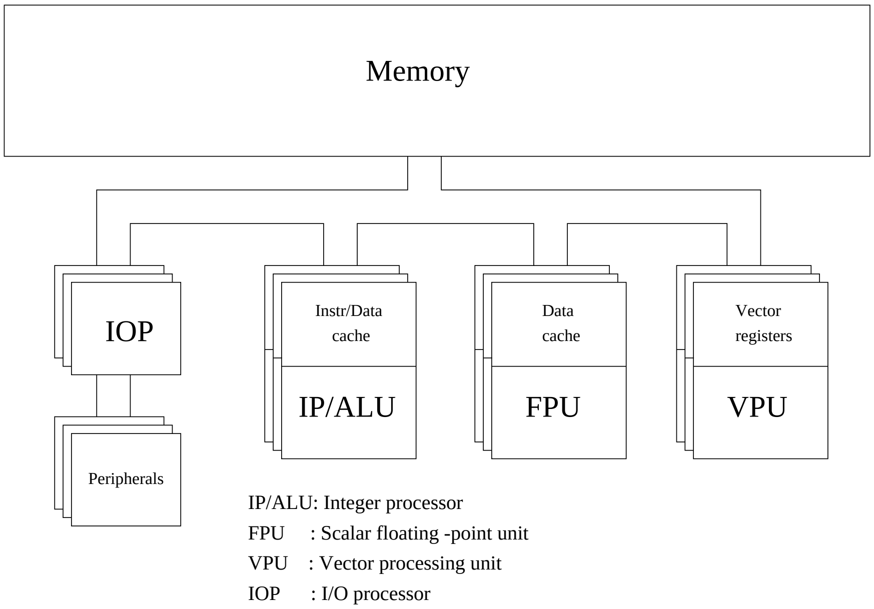

Figure 1: Block diagram of a vector processor.

Parasoft's Express (commercial), PVM (standing for **P**arallel **V**irtual **M**achine, non-commercial) citepvm, and MPI (**M**essage **P**assing **I**nterface, [14] also non-commercial). PVM and MPI have been adopted for instance by Convex, Cray, IBM and Intel for the transition stage between distributed computing and MPP on the clusters of their favorite processors and they are available on a large amount of distributed memory MIMD systems and even on shared memory MIMD systems for compatibility reasons. In addition there is a tendency to cluster shared memory systems, for instance by HIPPI channels, to obtain systems with a very high computational power. E.g., Silicon Graphics is already providing such arrays of systems, the Intel Paragon with the MP (**M**ulti **P**rocessor) nodes, and the NEC SX-4 also have this structure. The Convex Exemplar SPP-1200 could be seen as a more integrated example (although the software environment is much more complete and allows shared memory addressing).

2.1 Shared-memory SIMD machines

This subclass of machines is practically equivalent to the single-processor vectorprocessors, although other interesting machines in this subclass have existed (viz. VLIW machines). In the block diagram in Figure 1 we depict a generic model of a vector architecture.

The single-processor vector machine will have only one of the vectorprocessors depicted and the system may even have its scalar floating-point capability shared with the vector processor (as is the case in Cray systems, see 3.3.1). It may be noted that the VPU does not show a cache. The majority of vectorprocessors do not employ a cache anymore. In many cases the vector unit cannot take advantage of it and execution speed may even be unfavourably affected because of frequent cache overflow.

Although vectorprocessors have existed that loaded their operands directly from memory and stored the results again immediately in memory (CDC Cyber 205, ETA-10), all present-day vectorprocessors use vector registers. This usually does not impair the speed of operations while providing much more flexibility in gathering operands and manipulation with intermediate results.

Because of the generic nature of Figure 1 no details of the interconnection between the VPU and the memory are shown. Still, these details are very important for the effective speed of a vector operation: when the bandwidth between memory and the VPU is too small it is not possible to take full advantage of the VPU because it has to wait for operands and/or has to wait before it can store results. When the ratio of arithmetic to load/store operations is not high enough to compensate for such situations, severe performance losses may be incurred. Because of the high costs of implementing these datapaths between memory and the VPU, often compromises are sought and the number of systems that have the full required bandwidth (i.e., two load operations and one store operation at the *same* time) is limited.

The VPU is shown as a single block in Figure 1. Yet, again there is a considerable diversity in the structure of VPUs. Every VPU consists of a number of vector functional units, or "pipes" that fulfill one or several functions in the VPU. Every VPU will have pipes that are designated to perform memory access functions, thus assuring the timely delivery of operands to the arithmetic pipes and of storing the results in memory again. Usually there will be several arithmetic functional units for integer/logical arithmetic, for floating-point addition, for multiplication and sometimes a combination of both, a so-called compound operation. The division is usually approximated in the multiply pipe. In addition, there will almost always be a mask pipe to enable operation on a selected subset of elements in a vector of operands. Lastly, such sets of vector pipes can be replicated within one VPU (2- and 4-fold replication are common). Ideally, this will increase the performance per VPU by the same factor.

2.2 Distributed-memory SIMD machines

Machines of this type are sometimes also known as *processor-array* machines [7]. Because the processors of these machines operate in lock-step, i.e., all processors execute the same instruction at the same time (but on different data items), no synchronisation between processors is required. This greatly simplifies the design of such systems. Figure 2 shows a generic model of a DM-SIMD machine of which actual models will deviate to some degree.

All currently available DM-SIMD machines use a front-end processor to which they are connected by a datapath. I/O may be through the front-end system, by the processor array machine itself or both.

Figure 2 might suggest that all processors in such systems are connected in a 2-D grid and indeed, the interconnection topology of this type of machines always includes the 2-D grid. As opposing ends of each grid line are also always connected the topology is rather that of a torus. For several machines this is not the only interconnection scheme: They might also be connected in 3-D, diagonally, or more complex structures.

It is possible to exclude processors in the array from executing an instruction on

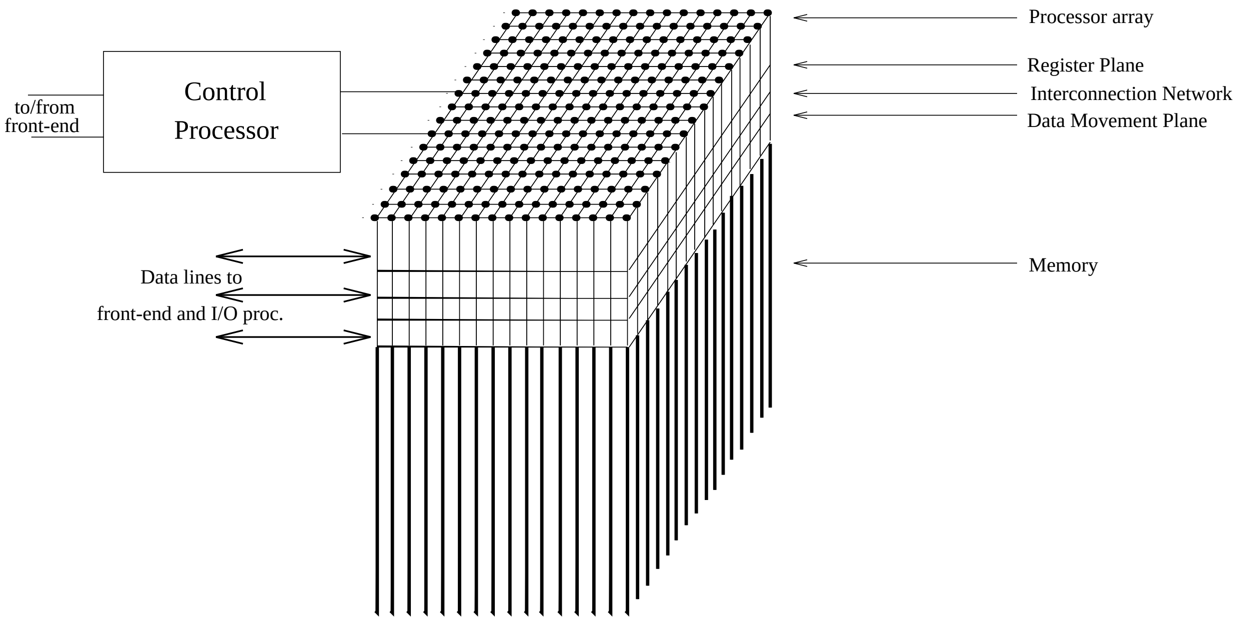

Figure 2: A generic block diagram of a distributed memory SIMD machine.

certain logical conditions, but this means that for the time of this instruction these processors are idle (a direct consequence of the SIMD type operation) which immediately lowers the performance. Another factor that may adversely affect the speed occurs when data required by processor i resides in the memory of processor j (in fact, as this occurs for all processors at the same time this effectively means that data will have to be permuted across the processors). To access the data in processor j, the data will have to be fetched by this processor and then send through the routing network to processor i. This may be fairly time consuming. For both reasons mentioned DM-SIMD machines are rather specialised in their use when one wants to employ their full parallelism. Generally, they perform excellently on digital signal and image processing and on certain types of Monte Carlo simulations where virtually no data exchange between processors is required and exactly the same type of operations is done on massive datasets with a size that can be made to fit comfortable in these machines.

The control processor as depicted in Figure 2 may be more or less intelligent. It issues the instruction sequence that will be executed by the processor array. In the worst case (that means a less autonomous control processor) when an instruction is not fit for execution on the processor array (e.g., a simple print instruction) it might be offloaded to the front-end processor which may be much slower than execution on the control processor. In case of a more autonomous control processor this can be avoided thus saving processing interrupts both on the front-end and the control processor. Most DM-SIMD systems have the possibility to handle I/O independently from the front/end processors. This is not only favourable because the communication between the front-end and back-end systems is avoided. The (specialised) I/O devices for the processor-array system is generally much more efficient in providing the necessary data directly to the memory of the processor array. Especially for very data-intensive applications like radar- and image processing such I/O systems are very important.

A feature that is peculiar to this type of machines is that the processors sometimes are of a very simple bit-serial type, i.e., the processors operate on the data items bitwise, irrespective of their type. So, e.g., floating-point operations have either to be implemented in software, or to be dealt with by floating-point coprocessors. As the number of processors in this type of systems is mostly large (1024 or larger, the Alenia Quadrics is a notable exception, however), the natural slowness of the processors can be often offset by their number, while the cost per processor is quite low as compared to full floating-point processors. When floating-point coprocessors are added their number is usually much lower because of the cost argument. An advantage of bit-serial processors is that they may operate on operands of any length. Both for random number generation (which often boils down to logical manipulation of bits) and for signal processing this is fortunate because in both cases operands of only 1–8 bits are abundant. As the execution time for bit-serial machines is proportional to the length of the operands, this may result insignificant speedups.

2.3 Shared-memory MIMD machines

In Figure 1 already one subclass of this type of machines was shown. In fact, the single-processor vector machine discussed there was a special case of a more general type. The figure shows that more than one FPU and/or VPU may be possible in one system.

The main problem one is confronted with in shared-memory systems is that of the connection of the CPUs to each other and to the memory. As more CPUs are added, the collective bandwidth to the memory ideally should increase linearly with the number of processors, while each processor should preferably communicate directly with all others without the much slower alternative of having to use the memory in an intermediate stage. Unfortunately, full interconnection is quite costly, growing with $O(n^2)$ while increasing the number of processors with $O(n)$. So, various alternatives have been tried. Figure 3 shows some of the interconnection structures that are (and have been) used.

As can be seen from the figure, a crossbar uses n^2 connections, an Ω-network uses $n \log_2 n$ connections while, with the central bus, there is only one connection. This is reflected in the use of each connection path for the different types of interconnections: for a crossbar each datapath is direct and does not have to be shared with other elements. In case of the Ω-network there are $log_2 n$ switching stages and as many data items may have to compete for any path. For the central databus all data have to share the same bus, so n data items may compete at any time.

The bus connection is the least expensive solution, but it has the obvious drawback that bus contention may occur thus slowing down the computations. Various intricate strategies have been devised using caches associated with the CPUs to minimise the bus traffic. This leads however to a more complicated bus structure which raises the costs. In practice it has proved to be very hard to design buses that are fast enough, especially where the speed of the processors have been increasing very quickly and it imposes an upper bound on the number of processors thus connected that in practice appears not to exceed a number of 10–20. In 1992, a new standard (IEEE P896) for a fast bus to connect either internal system components or to external systems has been defined. This bus, called the Scalable Coherent Interface (SCI) should provide a point-to-point bandwidth of 200–1,000 Mbyte/s. It is in fact used to in the HP/Convex SPP-1200, but could also be used within a network of workstations. The SCI is much more than a simple bus and it can act as the hardware network framework for distributed computing, see [10].

The Ω-network is a structure which is situated somewhere in between a bus and a crossbar which respect to potential capacity and costs. At this moment of the commercially available machines the IBM SP2, the Meiko CS-2, and the Cenju-3 use this network structure, but a number of experimental machines also have used this or a similar kind of interconnection. The BBN TC2000 that acted as a virtual shared-memory MIMD system used an analogous type of network (a Butterfly-network) and it is quite conceivable that new machines may use it, especially as the number of processors grows. For a large number of processors the $n \log_2 n$ connections become quickly more attractive than the n^2 used in crossbars. Of course, the switches at the intermediate levels

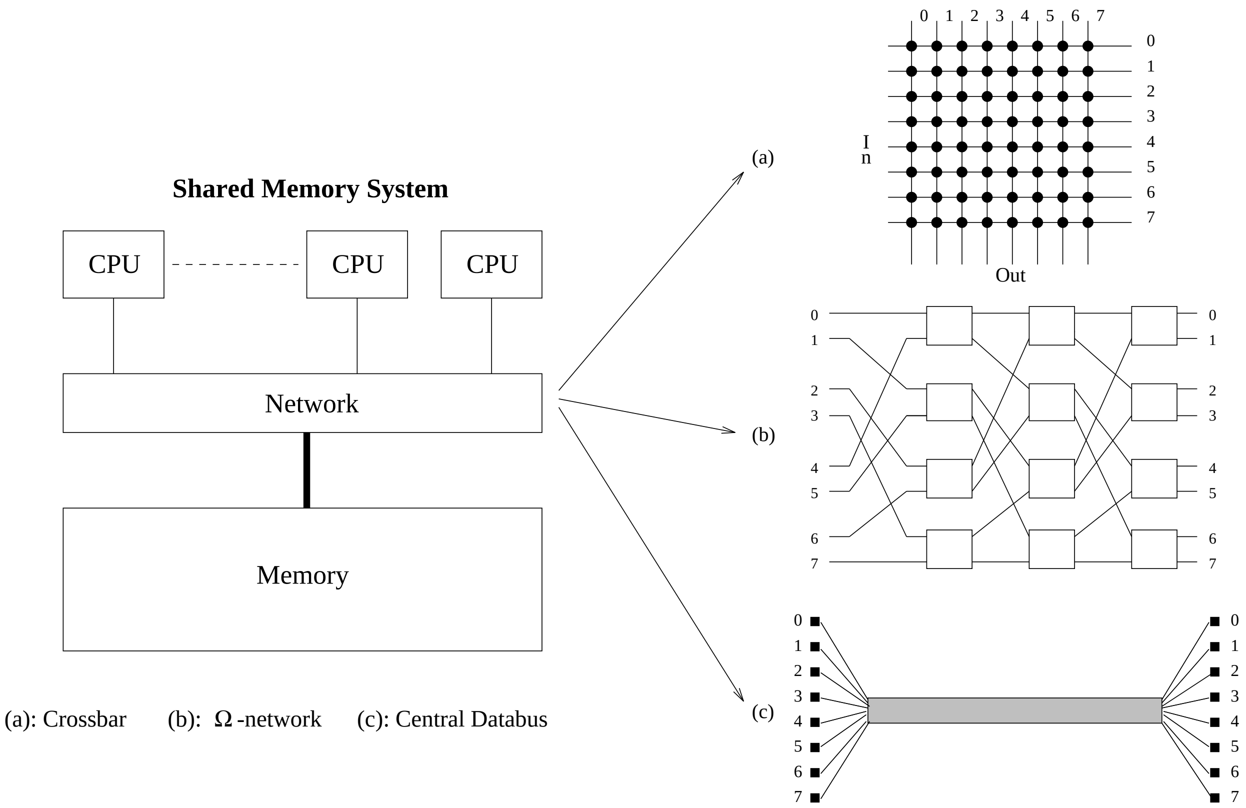

Figure 3: Some examples of interconnection structures used in shared-memory MIMD systems.

should be sufficiently fast to cope with the bandwidth required.

In all present-day multi-processor vectorprocessors crossbars are used. This is still feasible because the maximum number of processors in a system is still rather small (32 at most presently). When the number of processors would increase, however, technological problems might arise. Not only it becomes harder to build a crossbar of sufficient speed for the larger numbers of processors, the processors themselves generally also increase in speed individually, doubling the problems of making the speed of the crossbar match that of the bandwidth required by the processors.

Whichever network is used, the type of processors in principle could be arbitrary for any topology. In practice, however, bus structured machines do not have vector processors as the speeds of these would grossly mismatch with any bus that could be constructed with reasonable costs. RISC processors are however employed for bus-oriented systems. Caches can sometimes alleviate the bandwidth problem, but only when the data-access pattern allows them to be used sufficiently.

The systems discussed in this subsection are of the MIMD type and therefore different tasks may run on different processors simultaneously. In many cases synchronisation between tasks is required and again the interconnection structure is here very important. Most vectorprocessors employ special communication registers within the CPUs by which they can communicate directly with the other CPUs they have to synchronise with. A minority of systems does synchronisation via the shared memory. Generally, this is much slower but may still be acceptable when the synchronisation occurs relatively seldom. Of course for bus-based systems communication also have to be done via a bus. This bus is mostly separated from the databus to assure a maximum speed for the synchronisation.

2.4 Distributed-memory MIMD machines

The class of DM-MIMD machines is undoubtly the fastest growing part in the family of high-performance computers. Although this type of machines is more difficult to deal with than shared-memory machines and DM-SIMD machines (where the distribution of data is mostly obvious and/or transparent). The initial reluctance to use DM-MIMD machines seems to have been decreased. Partly this is due to greatly improved software and partly because, at least theoretically, this class of systems is able to outperform all other types of machines.

The advantages of DM-MIMD systems are clear: the bandwidth problem that haunts shared-memory systems is avoided because the bandwidth scales up automatically with the number of processors. Furthermore, the speed of the memory which is another critical issue with shared-memory systems (to get a peak performance that is comparable to that of DM-MIMD systems, the processors of the shared-memory machines should be very fast and the speed of the memory should match it) is less important for the DM-MIMD machines, because more processors can be configured without the afore mentioned bandwidth problems.

Of course, DM-MIMD systems also have their disadvantages: The communication between processors is much slower than in SM-MIMD systems, and so, the synchronisation overhead in case of communicating tasks is generally orders of magnitude higher

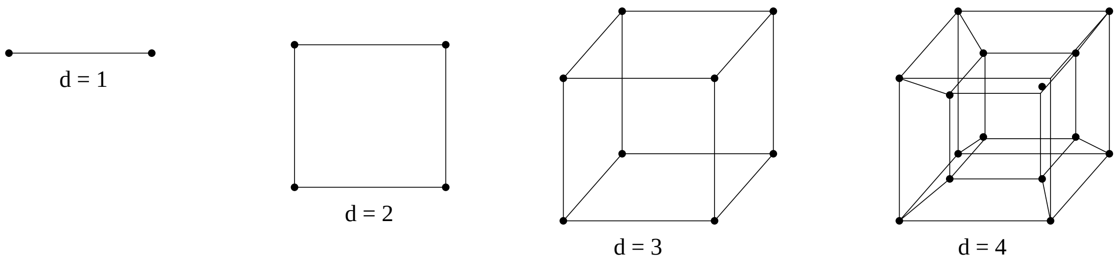

Figure 4: 1-, 2-, 3-, and 4-dimensional hypercube connections.

than in shared-memory machines. Moreover, the access to data that are not in the local memory belonging to a particular processor have to be obtained from non-local memory (or memories). This is again on most systems very slow as compared to local data access. When the structure of a problem dictates a frequent exchange of data between processors and/or requires many processor synchronisations, it may well be that only a very small fraction of the theoretical peak speed can be obtained. As already mentioned, the data- and task decomposition are factors that mostly have to be dealt with explicitly, which may be far from trivial.

It will be clear from the paragraph above that also for DM-MIMD machines both the interconnection and the speed of the datapaths are of crucial importance for the practical usefulness of a system. Again, as in section 2.3, the richness of the connection structure has to balanced against the costs. Of the many conceivable interconnection structures only a few are popular in practice. One of these is the so-called hypercube topology as depicted in Figure 4.

A nice feature of the hypercube topology is that for a hypercube with 2^d nodes the number of steps to be taken between any two nodes is at most d. So, the dimension of the network grows only logarithmically with the number of nodes. In addition, theoretically, it is possible to simulate any other topology on a hypercube: trees, rings, 2-D and 3-D meshes, etc. In practice, the exact topology for hypercubes do not matter too much anymore because all systems in the market today employ what is called "wormhole routing". This means that a message is send from one node to another node that it wants to communicate with to set up a direct connection between them. As soon this connection is established, the data proper is sent through this connection without disturbing the operation of the intermediate nodes. Except for a small amount of time in setting up the connection between nodes, the communication time has become virtually independent of the distance between the nodes. Of course, when several messages in a busy network have to compete for the same paths, waiting times are incurred as in any network that does not directly connect any processor to all others.

Many of the newly introduced massively parallel DM-MIMD systems seem to favour a 2- or 3-D mesh (torus) structure. The rationale for this seems to be that most large-scale physical simulations can be mapped efficiently on this topology and that a richer interconnection structure hardly pays off. However, some systems maintain (an) additional network(s) besides the mesh to handle certain bottlenecks in data distribution and retrieval.

A non-negligible fraction of systems in the DM-MIMD class employ crossbars. For relatively small amounts of processors (in the order of 64) this may be a direct or 1-stage crossbar, while to connect larger numbers of nodes multi-stage crossbars are used, i.e., the connections of a crossbar at level 1 connect to a crossbar at level 2, etc., instead of directly to nodes at more remote distances in the topology. In this way it is possible to connect in the order of a few thousands of nodes through only a few switching stages. Butterfly-, Ω-, or shuffle-exchange networks are often employed in this case.

As with SM-MIMD machines, a node may in principle consist of any type of processor together with local memory (with or without cache) and, in almost all cases, a separate communication processor and the links to connect the node to its neighbours. Nowadays, the node processors are mostly off-the-shelf RISC processors sometimes enhanced by vector processors. A problem that is peculiar to this type of system is the mismatch of communication vs. computation speed that may occur when the node processors are upgraded, without also speeding up the intercommunication. In some cases this may result in turning computational-bound problems into communication-bound problems.

3 Recount of the (almost) available systems

In this section we give a recount of all types of systems as discussed in the former section. When of a certain system various models are available, we will discuss them all at once. So, for instance, we will discuss Convex systems under one entry, SM-MIMD systems, even if a one-processor model of such a system strictly should be discussed under the SM-SIMD heading. We rather regard such systems a special cases of a general product line.

3.1 Shared-memory SIMD systems

In this class only one system is still actively marketed demonstrating the fact that this type of machines is only interesting for a rather small (but not unimportant) group of customers that have to do high-speed production work with well vectorised codes.

3.1.1 The Hitachi S3600 series.

Machine type: Vectorprocessor.
Models: S3600/120, S3600/140, S3600/160, S3600/180.
Operating system: VOS3/HAP/ES (IBM MVS compatible) and OSF/1.
Compilers: FORT77/HAP vectorising Fortran 77.
System parameters:

Model	S3600/120	S3600/140	S3600/160	S3600/180
Clock cycle VPU	4 ns	4 ns	4 ns	4 ns
Clock cycle scal. proc.	8 ns	8 ns	8 ns	8 ns
Theor. peak performance	0.25 Gflop/s	0.5 Gflop/s	1.0 Gflop/s	2 Gflop/s
Main memory	128–256MB	256–512MB	256–512MB	512–1024MB
Extended memory	$\leq$6GB	$\leq$16GB	$\leq$16GB	$\leq$16GB

Remarks:
The speed differences between the different models stem from replication of the multiply/add pipe in the models S3600/120–180. The /160 and /180 models have respectively two- and four-fold sets of a separate add- and a multifunctional multiply/add vector pipes. This should lead to a maximum of 3 results per clock cycle per pipe set. So, contrary to the information given by the vendor, the maximum performance of, e.g., the /180 should in some situations be 3 Gflop/s instead of 2.

Note that the clock cycle of the scalar processor is twice that of the VPU. The memory bandwidth from the memory from/to the CPU is 2 operands per clock cycle via 1 load and 1 load/store pipe per arithmetic pipe set), which is somewhat less than optimal. It is not possible to load two operands and store one result in one cycle. The /120 model lacks a separate load pipe, only a load/store pipe is present.

A unique feature of the S3600, as in its direct predecessor the S-820, is that all machines of the series are air cooled. All other machines in this class relied at least on water cooling.

Unlike the S-820 series, the S3600 series is also marketed worldwide, not only in Japan. This is also the case for the S3800 SM-MIMD machines (3.3.2).

Measured performances:

In [2] a speed of 851 Mflop/s for the solution of a full linear system of order 1000 is reported for the S3600/160. The S3600/180 attains a performance of 1672 Mflop/s on the same problem.

3.2 Distributed-memory SIMD systems

3.2.1 The Alenia Quadrics.

Machine type: Processor array.
Models: Quadrics Qx, QHx, $x = 1, 16$.
Front-end: Almost any Unix workstation.
Operating system: Internal OS transparent to the user, Unix on front-end
Connection structure: 3-D mesh, (see remarks).
Compilers: TAO: a Fortran 77 compiler with some Fortran 90 and some proprietary array extensions.
System parameters:

Model	Qx	QHx
Clock cycle	40 ns	40 ns
Theor. peak performance		
Per Proc. (32-bits)	50 Mflop/s	50 Mflop/s
Maximal (32-bits)	6.4 Gflop/s	100 Gflop/s
Memory	≤2 GB	≤32 GB
No. of processors	8–128	128–2048
Communication bandwidth		
Per Proc.	50 MB/s	50 MB/s
Aggregate local	≤6 GB/s	≤96 GB/s
Aggregate non-local	≤1.5 GB/s	≤24 GB/s

Remarks:
The Quadrics is a commercial spin-off of the APE-100 project of the Italian National Institute for Nuclear Physics. Systems are available in multiples of 8 processor nodes in the Q-model where up to 16 boards can be fitted into one crate or in multiples of 128 nodes in the QH-model by adding up to 15 crates to the minimal 1-crate system. The interconnection topology of the Quadrics is a 3-D grid with interconnections to the opposite sides (so, in effect a 3-D torus). The 8-node floating-point boards (FPBs) are plugged into the crate backplane which provides point-to-point communication and global control distribution. The FPBs are configured as 2^3 cubes that are connected to the other boards appropriately to arrive at the 3-D grid structure.

The basic floating-point processor, the so-called MAD chip, contains a register file of 128 registers. Of these registers the first two hold permanently the values 0 and 1 to be able to express any addition or multiplication as a "normal operation", i.e., a combined multiply-add operation, where an addition is of the form, $a \times 1 + b$ and a multiplication is $a \times b + 0$. In favourable circumstances the processor can therefore deliver two floating-point operations per cycle. Instructions are centrally issued by the controller at a rate of one instruction every two clock cycles.

Communication is controlled by the Memory Controller and the Communication Controller which are both housed on the backplane of a crate. When the Memory

Controller generates an address it is decoded by the Communication Controller. In case non-local access is desired, the Communication Controller will provide the necessary data transmission. The memory bandwidth per processor is 50 MB/s which means that very 2 cycles an operand can be shipped in or out a processor. The bandwidth for non-local communication turns out to be only four times smaller then local memory access.

The Quadrics communicates with the front-end system via a T805 transputer-based interface system, called the Local Asynchronous Interface (LAI). The interface can write and read the memories of the nodes and the Controller. Presently, the bandwidth of the interface to the front-end processor is not very large (1 MB/s). It is expected that this can be improved by about a factor of 30 in the near future. I/O has to be performed via the front-end system and will therefore be relatively slow.

The TAO language has several extensions to employ the SIMD features of the Quadrics. Firstly, floating-point variables are assumed to be local to the processor that owns them, while integer variables are assumed to be global. Local variables can be promoted to global variables. Other extensions are the `ANY`, `ALL`, and `WHERE/END WHERE` keywords that can be used for global testing and control. Processors that not meet a global condition effectively skip the operation(s) that are associated with it. For easy referencing nearest-neighbour locations special constants `LEFT`, `RIGHT`, `UP`, `DOWN`, `FRONT`, and `BACK` are provided. In addition, new data types and operators on these data types are supported together with overloading of operators. This enables very concise code for certain types of calculations.

3.2.2 The Cambridge Parallel Processing Gamma II.

Machine type: Processor array. **Models**: Gamma II 1000, Gamma II 4000. **Front-end**: Sun of HP workstation, stand-alone for dedicated applications.
Operating system: Internal OS transparent to the user, Unix on front-end
Connection structure: 2-D mesh, row- and column datapaths (see remarks).
Compilers: FORTRAN-PLUS (a Fortran 77 compiler with some Fortran 90 and some proprietary array extensions), C++.
Vendors information Web page: `http://TechCentral.www.com`.
System parameters:

Model	Gamma II 1000	Gamma II 4000
Clock cycle	33 ns	33 ns
No. of processors	1024	4096
Theor. peak performance		
Per Proc. (Mflop/s)	1.2	1.2
1-bit Gop/s	30.7	122.8
8-bit Gop/s	30.7	122.8
Gflop/s (32-bit) total	1.2	4.8
Program memory	$\leq$ 4 MB	$\leq$ 4 MB
Data memory	$\leq$ 32 MB	$\leq$ 128 MB
Int. comm. speed		
Across row, column	120 MB/s	480 MB/s
Memory to PE	3.84 GB/s	15.4 GB/s

Remarks:

In November 1995 the new Gamma II models has been announced by CPP. In essence there is not much difference with its predecessor the DAP Gamma. However, the clock cycle has tripled to 33 ns with an equivalent rise in the peak performance of the systems.

The Gamma II is presented as the fourth generation of this type of machine. Indeed, the macro architecture of the systems has hardly changed since the first ICL DAP (the first generation of this system) was conceived. As in the ICL DAP in the Gamma 1000 models the 1024 processors are ordered in a 32×32 array, while the Gamma 4000 has 4096 processors arranged in a 64×64 square.

The systems are able to operate byte parallel on appropriate operands to speed up floating-point operations, however, for logical operations bit-wise operations are possible, which makes the machines quite fast in this respect. As the byte parallel code consists of separate sequences of microcode instructions, the bit processor plane and the byte processor plane are in fact independent and can work in parallel. This is also the case for I/O operations. Also character-handling can be done very efficiently. This is the reason that Gamma systems are often used for full text searches.

As in all processor-array machines, the control processor (called the Master Control Unit (MCU) in the DAP) has a separate memory to hold program instructions while the data are held in the data memory associated with each Processing Element (PE) in the processor array. So, for a Gamma 1000 with 32 MB of data memory each PE has 32 KB of data memory directly associated to it. To access data in other PE's memories these must be brought up to the data routing plane and shifted to the appropriate processor.

As already mentioned under the heading of the connection structure, there are two ways of connecting the PEs. One is the 2-D mesh that connects each element to its North-, East-, West-, and South neighbour. In addition there are row- and column data paths that enable the fast broadcast of a row or column to an entire matrix by replication. Conversely, they can be used for row- or column wise reduction of matrix objects into a column- or row vector of results from, e.g., a summing- or maximum operation.

Separate I/O processors and disk systems can be attached to the Gamma directly thus not burdening the front-end machine (and the connection between front-end and DAP) with I/O operations and unnecessary data transport. One of these I/O devices is the GIOC that can transport data to the data memory at a sustained rate of 80 MB/s transposing the data to the vertical storage mode of the data memory on the fly. Also, a direct video interface is available to operate a frame buffer.

A nice (non-standard) feature of the FORTRAN-PLUS compiler is the possibility to use logical matrices as indexing objects for computational matrix objects. This enables a very compact notation for conditional execution on the processor array. In addition, recently C++ is available.

Measured Performances:

In [4] the speed of matrix multiplication on various DAP models (precursors of the Gamma systems) is analyzed. The documentation states 32-bit floating-point add speed of 1.68 Gflop/s on 4096 PEs, while a 32-bit 1,024 complex FFT would attain 2.49 Gflop/s. No independent performance figures for the Gamma II systems are available.

3.2.3 The Digital Equipment Corp. MPP series

These machines are identical to the MasPar machines described in 3.2.4 and 3.2.5.

3.2.4 The MasPar MP-1.

Machine type: Processor array.
Models: MP1101, MP1102, MP1104, MP1208, MP1216.
Front-end: DECstation 5000 or DEC VAX.
Operating system: Internal OS transparent to the user, Ultrix or VMS on front-end.
Connection structure: 2-D mesh, crossbar (see remarks).
Compilers: MPL: (C with extensions), MPF: (Fortran 90-like with extensions).
System parameters:

Model	MP1101	MP1102	MP1104	MP1208	MP1216
Clock cycle	83 ns	83 ns	83 ns	83 ns	83 ns
No. of processors	1024	2048	4096	8192	16384
Theor. peak performance					
Per proc. (Mflop/s)	0.034	0.034	0.034	0.034	0.034
Mop/s (32-bit)	1600	3200	6400	13000	26000
Mop/s (64-bit)	800	1600	3200	6400	13000
Mflop/s (32-bit)	75	150	300	600	1200
Mflop/s (64-bit)	34	69	138	275	550
Program memory	1–4MB	1–4MB	1–4MB	1–4MB	1–4MB
Data memory	16–64MB	32–128MB	64–256MB	128–512MB	256–1GB
Int. comm. speed					
Via Xnet (n. neighbour)	1.4 GB/s	2.8 GB/s	5.7 GB/s	11.5 GB/s	23.0 GB/s
Via global router	80 MB/s	160 MB/s	320 MB/s	640 MB/s	1.28 GB/s

Remarks:
The Processing Elements (PEs) of the MP-1 are more intricate than those from the DAP Gamma II (3.2.2). Each PE contains a 4-bit parallel Arithmetic/Logic Unit together with a 1-bit functional unit, a 16-bit exponent unit and a 64-bit mantissa unit. These units may be operated separately or in concord (e.g., for floating-point calculations). Because of the hardware implementation of the PEs, only 1-, 8-, 16-, 32-, and 64-bit data types are allowed.

Unlike on the DAP Gamma II, on the MP-1 it is possible to address data items in the data memories indirectly. This greatly facilitates manipulation of matrix objects indexed through an index matrix.

One type of interconnection of the PEs is a 2-D rectangular mesh (with wrap-around). This is however implemented by connecting the PEs diagonally via 3-way switches. As the setting of the switches only takes 1 cycle, this means that every PE can reach it 8 surrounding neighbours in 1 cycle. For more general routing schemes a Global Router is available. This acts, in principle, as a three-stage crossbar. PEs are arranged in clusters of 4×4, which connect to other clusters through the first level of the crossbar. All clusters connect via an intermediate stage to the target stage (again at cluster level). The ports from the clusters are multiplexed to the individual PEs within a cluster. As this type of communication is fairly intricate, it is much slower than via the Xnet (see system parameter list above).

As with the DAP Gamma II, there are provisions for connecting a frame buffer and/or disks directly to the MP-1. Also like the DAP, the MP-1 is essentially a single-user machine, that is, only one user at a time can have a task on the MP-1. Of course, tasks can be scheduled via a multi-user interface on the front-end system.

The MP-1 features a very nice X-window based programming environment, MPPE, which integrates an interactive source debugger, a profiler, and output windows in one environment.

Measured Performances:

In [2] the solution of a full linear system was reported on a 16384 PE machine with a speed of 440 Mflop/s. The same report estimated the peak performance to be 580 Mflop/s in 64-bit precision.

3.2.5 The MasPar MP-2.

Machine type: Processor array.
Models: MP2201, MP2202, MP2204, MP2208, MP2216.
Front-end: DECstation 5000 or DEC VAX.
Operating system: Internal OS transparent to the user, Ultrix or VMS on front-end.
Connection structure: 2-D mesh, crossbar (see remarks).
Compilers: MPL: (C with extensions), MPF: (Fortran 90-like with extensions).
System parameters:

Model	MP2201	MP2202	MP2204	MP2208	MP2216
Clock cycle	80 ns	80 ns	80 ns	80 ns	80 ns
No. of processors	1024	2048	4096	8192	16384
Theor. peak performance					
Per proc. (Mflop/s)	0.15	0.15	0.15	0.15	0.15
Mop/s (32-bit)	4250	8500	17000	34000	68000
Mop/s (64-bit)	2100	4250	8500	17000	34000
Mflop/s (32-bit)	400	800	1600	3200	6300
Mflop/s (64-bit)	150	300	600	1200	2400
Program memory	1–4MB	1–4MB	1–4MB	1–4MB	1–4MB
Data memory	64MB	128MB	256MB	512MB	1GB
Int. comm. speed					
Via Xnet (n. neighbour)	1.25 GB/s	2.5 GB/s	5.0 GB/s	10.0 GB/s	20.0 GB/s
Via global router	80 MB/s	160 MB/s	320 MB/s	640 MB/s	1.28 GB/s

Remarks:
Apart from being roughly 2.5 times faster than the MasPar MP-1 (3.2.4), there is little difference between the MP-2 and the MP-1. The gain in speed relative to the MP-1 is accomplished by using a processor with a higher degree of bit-parallelism. This resulted in a higher performance at the expense of the number of data formats: only 32-bit and 64-bit data are allowed on the MP-2.

MasPar will market both the MP-1 and MP-2. In the view of the company there is a market for both. With a fixed budget one can make a choice for a system with larger memory, but slower or a faster system with a smaller memory.

Software and tools are exactly the same for both types of machines.

Measured Performances:

In [2] a speed of 1.6 Gflop/s was reported for a 16,384 processor machine in solving a 11,264 order full linear system.

3.3 Shared-memory MIMD systems

For some systems in this category it will not always be possible to discuss all models of a particular product line because the number of configurations is simply too large. However, we attempt to give the representative configurations for such systems. In

addition, when single-processor versions of a certain machine are available, this is regarded as a special case of a multi-processor version and the one-processor versions are omitted from section 3.1 where they could have been treated.
Furthermore, we have included systems here that a have shared-memory model as a basis but that may be extended by coupling several of these systems together in a distributed memory way. The distinction is not always very clear when looking at the end product: a multi-frame DEC AlhpaServer (3.3.4) and an HP/Convex SPP-1200 (3.4.8) look very much alike. The difference lies in the integration. In the latter machine the distributed memory extension of multi-CPU nodes was a basis for the architecture while in the former DEC system it is more like an added feature.

3.3.1 The Cray Research Inc. Cray J90-series, T90 series.

Machine type: Shared-memory multi-vectorprocessor.
Models: Cray J90, T90.
Operating system: UNICOS (Cray Unix variant).
Compilers: Fortran, C, C++, Pascal, ADA.
Vendor information Web page: `http://www.cray.com`.
System parameters:

Model	Cray J90	Cray T90
Clock cycle	10 ns	2.2 ns
Theor. peak performance		
Per processor	200 Mflop/s	1.8 Gflop/s
Maximal	3.2 Gflop/s	58Gflop/s
Main memory	≤4 GB	≤8 GB
Memory bandwidth		
Single proc. bandwidth	1.6 GB/s	24 GB/s
No. of processors	4–32	1–32

Remarks:
Cray Research Inc. (CRI) supports at this moment 3 product lines (apart from the SuperSparc-based CS6400 which is targeted to the commercial market and is not discussed in this report). Two of these are multi-headed vector processors which are discussed here. The third is the T3E, a DM-MIMD machine that will be described in section 3.4.

The Cray J90 series is the entry level model marketed by CRI announced in September 1994. The J90 series is based on CMOS technology which has a low power consumption (all J90s are air cooled) and low production costs. The machine is binary compatible with the high-end systems. It has one multiply and add vector pipe set per CPU at a clock cycle of 10 ns which results in a theoretical peak performance of 200 Mflop/s. Furthermore, a cache has been added to speed up scalar processing (as in the Convex C4 series, see 3.3.3). It is interesting to note that the strategy of using more (four) multi-functional pipes as in the predecessor, the Y-MP EL has been left again to return to the classic two-pipe/CPU design.

The Cray T90 series is built in ECL logic and has therefore a much lower clock cycle (2.2 ns) and correspondingly faster SRAM memory. As its direct predecessor, the Cray C90, every CPU contains two vector add and multiply pipes. This gives rise to a maximum of 4 floating-point results/clock cycle/CPU equivalent to a theoretical peak performance of 1.8 Gflop/s per CPU or 58 Gflop/s for a maximal system.

The Cray T90 machines are at this moment the only ones with a memory bandwidth as seems optimal for vector processors: two operands can be loaded and one result can be stored in one cycle for each pipe set. For the T90 this meant that the relative bandwidth has to be 48 bytes/cycle/CPU. This has indeed been accomplished and observed results indicate that for the T90 the performance scales up with the clock cycle and the number of functional units (see measured performances below). For the J90 series the bandwidth is lower: 16 bytes/cycle. This is regrettably less than was available in its predecessors, Y-MP EL machines, and it might adversely affect the efficiency.

Another property that is unique for the Cray T90 systems is that they do not have a separate scalar processor but that scalar- and vector code have to share the same functional units. However, a small scalar cache is added to speed up scalar calculations. The Cray J90 series has separate scalar processors. Theoretically, the absence of separate scalar processors might impair the throughput speed (Hitachi (3.3.2) even adds an extra scalar processor in the S-3800 series to combat excessive context switching). However, in practice the drawbacks seem rather limited.

Contrary to earlier high-end Cray systems, the T90 now features compatibility with the IEEE 754 floating-point standard. Formerly, Cray-specific floating-point arithmetic was employed which could give rise to problems in data exchange with other systems and in different computational results due to the difference in arithmetic.

Measured Performances:

On the T90 in [2] a speed of 29.4 Gflop/s was found on a 32 processor machine for the solution of an order 1000 dense linear system. For a J90 series machine with 32 processors a speed of 4.486 Gflop/s was observed for the same problem, which amounts to efficiencies of 51 and 70% for the T90 and the J90, respectively.

3.3.2 The Hitachi S3800 series.

Machine type:Vectorprocessor.
Models: S-3800/x60, S-3800/$y8z$; $x = 1, 2 \quad y = 1, 2, 4 \quad z = 0, 2$.
Operating system: VOS3/HAP/ES (IBM MVS compatible) and OSF/1.
Compilers: FORT77/HAP vectorising Fortran 77.
System parameters:

Model	S3800/x60	S3800/$y8z$
Clock cycle VPU	2 ns	2 ns
Clock cycle scal. proc.	6 ns	6 ns
Theor. peak performance	4–8 Gflop/s	8–32 Gflop/s
No. of processors		
Scalar	1–2	1–4
Vector	1–2	1–4
Main memory	256–1024MB	512–2048MB
Extended memory	≤16GB	≤32GB

Remarks:
The S3800 is the current top-end system of Hitachi's S-3000 series. Five different models are offered: The 160 and the 260 in which the 260 is simply the 2-CPU version of the the 160. Furthermore, there is a sub-series 180, 280, and 480, of which the 280 and 480 are again 2-CPU and 4-CPU versions of the 180. However, in addition, there is a model 182 with 2 scalar processors and 1 vector processor as is offered in the Fujitsu VPX200 series and for the same reason: context switching delay between jobs should be reduced by this scheme. The smallest model, the S-3800/160 has 4 multi-functional multiply/add pipes which may deliver up to 8 results per clock cycle. This is equivalent to 4 Gflop/s. In the /180 the number of pipes is doubled to 8 with a corresponding peak performance of 8 Gflop/s. All models feature one or more separate divide pipes. As the multi-headed systems can work in parallel, the top model, the S-3800/480, may theoretically attain a speed of 32 Gflop/s.

Hitachi now delivers an auto-parallelising compiler, which features parallelising compiler directives similar to those of Cray and NEC. The OSF/1 system can be run under the MVS-like VOS3/HAP/ES, but it can also be run as a native operating system.

Measured Performances:
The first S-3000 system, a S-3800/480, was installed in January 1993 at the University of Tokyo. Tests with the EuroBen benchmark were done on this system in July-September 1993. During these tests a speed of 5.7 Gflop/s was observed for the evaluation of a 9$^{\rm th}$ degree polynomial on a single processor. In matrix-vector multiplication, speeds of 6.5 Gflop/s on one processor were measured (see [15, 16]). In [2] a speed of 28.4 Gflop/s on 4 processors is reported for the solution of an order 15,500 dense linear system. The efficiency is here 89%.

3.3.3 The HP/Convex C4 series.

Machine type: Shared-memory multi-vectorprocessor.
Models: C46x0, $x = 1, \ldots, 4$
Operating system: ConvexOS (Convex's Unix variant).
Compilers: Fortran, C, C++, ADA, Lisp.
Vendors Information Web page: `http://www.convex.com/prod_serv/prod_serv.html`.

System parameters:

Model	C4600
Clock cycle	7.41 ns
Theor. peak performance	
Per proc. (64-bit prec.)	810 Mflop/s
Per proc. (32-bit prec.)	1620 Mflop/s
Maximal, 64-bit precision	3240 Mflop/s
Maximal, 32-bit precision	6480 Mflop/s
No. of processors	1–4
Main memory	≤4GB
Memory bandwidth	
Single proc. bandwidth	1080 MB/s

Remarks:

Recently (November 1995), Convex Computer Corp. has become a subsidiary of Hewlett Packard. This has, at least for the moment no impact on the products that are marketed by HP/Convex. Both the vectorprocessors and the Exemplar SPP series (see section 3.4) will stay on the market. The C4600 series is the fourth generation of vectorprocessors from Convex. Unlike in the former C3800 series, with a maximum of 8 processors, the highest number of processors is four in the C4640 model. A major difference with the former generations is that more functional unit sets per CPU are present: six general purpose functional units. This brings the number of floating-point results per cycle to 6 in the ideal case. Because the floating-point units are general the opportunities for linking or independent processing are increased with respect to specialised multiply and add pipes which increases the scheduling density of operations. In addition, some logical operations can be done in the functional units which enables 32-bit convolutions to be done in excess of 1 Gflop/s (this is called the "extended architecture" in Convex jargon).

As in the former C3400 and C3800 GaAs components are used to arrive at the cycle time of 7.41 ns. Also like in these former models, there is difference in speed of a factor of two between single precision (32 bits) and double precision (64 bits) calculations.

As for the Convex Exemplar SPP-1200 (see 3.4.8) an "application compiler" is available that is capable of interprocedural analysis. This can greatly enhance the vectorisability of some codes and in general is beneficial in optimising large codes.

Measured performances:

Traditionally, Convex systems are able to obtain a significant fraction of their theoretical peak performance. On a C220 (functionally equivalent to a C3220) 77.6 and 88.9 Mflop/s out of the theoretical 100 Mflop/s have been observed for a Fortran 77 and a library implementation of a linear system solver, respectively [12]. The C4600 proves to be no exception: on one processor the solution of a dense linear system of order $N = 1000$ shows a speed of 683 Mflop/s on one processor for 64-bits precision and of 1320 Mflop/s on a C4620. At 32-bits precision speeds of resp. 1227 and 2252 Mflop/s were found on the C4610 and the C4620. In [2] a speed 1.933 Gflop/s out of 3.24 Gflop/s at maximum is reported.

3.3.4 The Digital Equipment Corp. AlphaServer.

Machine type: RISC-based distributed-memory multi-processor.

Models: AlphaServer 8200 5/300, 8400 5/300.

Operating system: Digital Unix (DEC's flavour of Unix).

Connection structure: Crossbar.

Compilers: Fortran 77, HPF, C, C++.

Vendors information Web page: `http://www.digital.com:80/info/hpc`.

System parameters:

Model	8200	8400
Clock cycle	3.3 ns	3.3 ns
Theor. peak performance		
Per proc. (64-bit)	600 Mflop/s	600 Mflop/s
Maximal (64-bit)	3.6 Gflop/s	7.2 Gflop/s
Main memory	≤6 GB	≤14 GB
Memory bandwidth		
Processor/memory	1.6 GB/s	1.6 GB/s
No. of processors	6	12

Remarks:

The AlphaServers are symmetric multi-processing systems which are based on the Alpha 21164 processor. The 8200 model is a somewhat smaller copy of the 8400 model: in the 8200 a maximum of 6 CPUs can be accomodated while this number is 12 for the 8400 model. Also, there is room for at most 6 GB of memory in the 8200 while the 8400 can house 14 GB. However, the amount of CPUs and memory is not independent. For instance, the 8400 has 9 system slots. One of these is reserved for I/O and one will have to contain at least one CPU module which can contain 1 or 2 CPUs. From the remaining slots 6 can be used either for memory or for a CPU module. So, one has to choose for either higher computational power or for more memory. This can potentially be a problem for large applications that require both.

As with the SGI PowerChallenge (see 3.3.6), AlphaServers can be clustered using a PCI bus Memory channel for interconnection of the systems. The systems need not be of the same model. The bandwidth of this interconnect is 100 MB/s. Eight systems can be coupled in this way. To support this kind of cluster computing, HPF and optimised versions of PVM and MPI are available.

Measured Performances:

As yet, only single system results for the AlphaServer 8400 are available. No cluster results are known. In [2] a speed of 5.0 Gflop/s for an 8-processor system are reported for the solution of a dense linear system of order 9548. In [13] results for the NAS parallel benchmarks are given. For the class B EP benchmark a time of 78.43 seconds was measured for 8 processors. For the class B LU, SP, and BT benchmarks times of 296.19, 364.54, and 458.21 seconds were found, respectively.

3.3.5 The NEC SX-4.

Machine type: Distributed-memory multi-vector processor.
Models: SX-4C, SX-4.
Operating system: EWS-UX/V (Unix variant based on Unix System V.4).
Connection structure: Multi-stage crossbar (see Remarks).
Compilers: Fortran 77, Fortran 90, HPF, ANSI C, C++.
Vendors information Web page: `http://www.nec.co.jp/english/product/computer/sx`.
System parameters:

Model	SX-4Ce	SX-4C	SX-4
Clock cycle	8 ns	8 ns	8 ns
Theor. peak performance			
Per Proc. (64 bits)	1 Gflop/s	2 Gflop/s	2 Gflop/s
Single frame:			
Maximal (64 bits)	1 Gflop/s	8 Gflop/s	64 Gflop/s
Multi frame: Maximal (64 bits)	—	—	1 Tflop/s
Main memory	$<$ 2 GB	$<$ 2 GB	$<$ 128 GB
Communication bandwidth (see Remarks)	—	—	—
No. of processors	1	1–4	4–512

Remarks:

The SX-4 series is comprised of a large range of machine sizes. The smallest of these is the SX-4Ce. This machine has one CPU housing 4 vector pipe sets. As the clock cycle is 8 ns and each pipe set is able to deliver 2 floating-point results per cycle, the total maximum performance is 1 Gflop/s for this system. In all other systems the replication factor of the pipe sets is 8 which doubles the speed per CPU to a maximum of 2 Gflop/s. The bandwidth from memory to the CPUs is 16 64-bit words per cycle per CPU. With a replication factor of 8 this is enough to provide two operands per pipe set but it is not sufficient to transport the results back to the memory at the same time. So, some trade-offs with the re-use of operands have to be made to attain the peak performance.

The technology used is CMOS. This lowers the fabrication costs and the power consumption appreciably (the same approach is being used in the Fujitsu VPP300, see 3.4.6) and all models are air cooled. This enables the placement of up to 32 CPUs in one frame (for the SX-4 model). Beyond this maximum single frame system, it is possible to couple up to 16 frames together to form a distributed memory system. This is equivalent to the PowerChallenge Array idea (see 3.3.6). There are two ways to couple the SX-4 frames: NEC provides a full crossbar, the so-called IXS, crossbar to connect the various frames together at a speed of 16 GB/s for poit-to-point out-of-frame communication (128 GB/s bi-sectional bandwidth for a maximum configuration). In addition, a HiPPI interface is available for interframe communication at lower cost and speed.

For distributed computing there is an HPF compiler and for message passing an optimised MPI (MPI/SX) is available. The SX-4 is the only system that supports three floating-point number systems: IBM-compatible, Cray-compatible, and the IEEE 754 standard.

Measured Performances:

The SX-4 will be available from the first quarter of 1996. Therefore, at this moment no performance results are available.

3.3.6 The Silicon Graphics Power Challenge

Machine type: Shared-memory multi-processor.
Models: Power Challenge L, XL.
Operating system: IRIX (SGI's Unix variant).
Compilers: Fortran 77, C, C++ , Pascal.
Vendors information Web page: `http://www.sgi.com/Products/hardware/Power/index.html`.
System parameters:

Model	Model L	Model XL
Clock cycle	13.3 ns	13.3 ns
Theor. peak performance		
Per proc. (64-bit)	300 Mflop/s	300 Mflop/s
Maximal (64-bit)	1.8 Gflop/s	5.4 Gflop/s
Main memory	≤6 GB	≤16 GB
Memory bandwidth		
Proc. to cache/proc.	1.2 GB/s	1.2 GB/s
Main memory/cache	1.2 GB/s	1.2 GB/s
No. of processors	2–6	2–18

Remarks:

The Power Challenge systems are shared-memory multiple-instruction multiple-data parallel (MIMD) computers. So, several different instructions can be going on at the same time using different data items in these instructions. All data are stored in a single shared memory from which the multiple processors draw the data items they need and in which the results are stored again. In most high performance systems the main problem is to provide the CPUs with enough data and to transport the results back at such a rate that they can be kept busy continuously. In this, the Powerchallenge is no exception. The data is transported from the main memory to the CPUs by a central bus. The so-called POWERpath-2 bus is 256 bits wide and has a bandwidth of 1.2 GB/s. This is very fast as busses go but even then the data rates that are needed by the CPUs cannot possibly be fulfilled when no special provisions would exist. These provisions are present in the form of data and instruction caches for each of the CPUs.

The Power Challenge series uses MIPS R8000 RISC processors (formerly called the TFP processor standing for True Floating Point) with a nominal peak speed of 300 Mflop/s. Although the clock rate of this processor is two times lower than that of its predecessor, the R4400, the performance is 4 times higher. As the need for data is even higher than that of the R4400 processors with this speed of processing, there is a special extra cache called the "Streaming cache" of up to 16 MB. This is very large and it should reduce the bus traffic as much as possible. All floating-point operations are done by streaming the operands from this large off-chip cache to the floating-point registers. In contrast to the R4400 processor, the R8000 is able to do a combined multiply-add operation which in many cases doubles the operation speed. In addition, the floating-point functional units are doubled with respect to the R4400 which should explain the four-fold increase in performance with respect to this predecessor.

Power Challenge systems can be coupled by HiPPI channels to form a cluster of systems using very efficient "shared-memory" PVM and MPI implementations that can be used homogeneously (for the user) both within a single Power Challenge system and between them. This could be used for the solution of extremely large application problems. Such clusters are called Power Challenge Arrays by SGI. SGI wants to extend this technique by providing faster coupling and switching between the systems. This trend is also to be seen with other vendors (see 3.4.6 and 3.3.5(SX-4)).

Power Challenge systems can be coupled by HiPPI channels to form a Parallelisation is done either automatically by the (Fortran or C) compiler or explicitly by the user, mainly through the use of directives. As synchronisation, etc., has to be done via memory the parallelisation overhead is fairly large. In fact, experiments as reported in citebmtut show that a distributed memory implementation of the same problem can be much faster on a single PowerChallenge.

Measured Performances:

On a SGI PowerChallenge Array equipped with 128 processors a performance of 26.7 Gflop/s was measured when solving an order 53,000 dense linear system [2]

3.3.7 The Tera MTA

Machine type: Distributed-memory multi-processor.
Models: MTA.
Operating system: Unix BSD4.4 + proprietary micro kernel.
Compilers: Fortran 77 (Fortran 90 extensions), HPF, C, C++.
Vendors information Web page: `http://www.tera.com`.
System parameters:

Model	MTA-xC
Clock cycle	< 3 ns
Theor. peak performance	
Per proc. (64-bit)	1 Gflop/s
Maximal (64-bit)	256 Gflop/s
Main memory	$\leq$16 GB
Memory bandwidth	
CPU-to-memory	> 8 GB/s
No. of processors	16–256

Remarks:

Although the memory in the MTA is physically distributed, the system is emphatically presented as a shared memory machine (with non-uniform access time). The latency incurred in memory references is hidden by *multi-threading*, i.e., usually many concurrent program threads (instruction streams) may be active at any time. Therefore, when for instance a load instruction cannot be satisfied because of memory latency the thread requesting this operation is stalled and another thread of which an operation can be done is switched into execution. This switching between program threads only takes 1 cycle. As there may be up to 128 instruction streams and 8 memory references can be issued without waiting for preceding ones, a latency of 1024 cycles can be tolerated. References that are stalled are retried from a retry pool. A construction that works out similarly is to be found in the Stern Computing Systems SSP machines (see 3.3.9).

The connection network connects a 3-D cube of p processors with sides of $p^{\frac{1}{3}}$ of which alternately the x- or y axes are connected. Therefore, all nodes connect to four out of six neighbours. Furthermore, there is an I/O port at every node. Each network port is capable of sending and receiving a 64-bit word per cycle which amount to a bandwidth of 22.6 GB/s per port. In case of detected failures, ports in the network can be bypassed without interrupting operations of the system.

Although the MTA should be able to run "dusty-deck" Fortran programs because parallelism is automatically exploited as soon as an opportunity is detected for multi-threading, it may be (and often is) worthwhile to explicitly control the parallelism in the program and to take advantage of known data locality occurrences. MTA provides handles for this in the form of library routines, including synchronisation, barrier, and reduction operations on defined groups of threads. Controlled and uncontrolled parallelism approaches may be freely mixed. HPF will also be supported for SPMD-style programming.

Measured Performances:

The MTA will be benchmarkable from the beginning of 1996, therefore, no performance figures are available yet.

3.4 Distributed-memory MIMD systems

In particular for this class of systems we cannot claim completeness of this overview. This has two reasons: First, at present this is the most dynamic area of development of new machines and it is quite probable that already new systems appear on the market while this report goes to print. This in no way implies that the systems not mentioned here should be in any way inferior to the ones that appear in this section. It is rather felt that many of these systems are in some sense equivalent and listing (almost) all of the systems would be counterproductive in the sense that the descriptions of the systems might lead to confusion.

For distributed-memory MIMD machines obviously the internode bandwidth and latency are very important system parameters. Unfortunately, it is very hard to come by reliable figures for these parameters. Therefore, we only can state the internode bandwidth point-to-point for the majority of systems, not for all. Where we do not have these figures we give the aggregate bandwidth which is less informative but better than nothing. We were not able to give latency figures for the systems for two reasons: manufacturers mostly state hardware latencies which, regrettably, does not say very much about the actual latency, except that the hardware latency is a guaranteed lower bound. The second reason is that the actual (software) latency, even if known at some point in time, decreases very fast, as better implementation of the communication software occurs continuously. Therefore, stating figures for this system parameter is next to useless at the moment even when very much desired.

3.4.1 The Alex AVX 2.

Machine type: RISC-based distributed-memory multi-processor.
Models: AVX series 2.
Operating system: Trollius (Unix-like variant with extentions).
Connection structure: Crossbar.
Compilers: Fortran, C, C++, Linda.
System parameters:

Model	AVX series 2
Clock cycle	25 ns
Theor. peak performance	
Per proc. (64-bit)	60 Mflop/s
Maximal (64-bit)	3.84 Gflop/s
Main memory	$\leq$1,280 MB
Memory/node	$\leq$20 MB
Memory bandwidth	5 GB/s
Communication bandwidth	$\leq$10 MB/s
No. of processors	8–64

Remarks:

The AVX system is reminiscent to the Meiko i860 CS (see 3.4.12) in that the system can be configured with and without Intel i860 processors in the processing nodes while each node always contains a T805 transputer which is responsible for inter-node communication. The i860s are used for computational intensive tasks. Instead of computational nodes with or without i860s special function nodes, like SCSI interface nodes or graphics nodes may be installed to make the machine more balanced with respect to I/O requirements or graphics performance.

Up to 8 simultaneous users can be allocated on the (virtual) machine leaving the users the freedom to define the topology for their machine in software.

Third party programming environments available are Perihelion's Helios which should be able to attract users which migrate from transputer based machines and Parasoft's Express which runs on many DM-MIMD platforms.

Measured Performances:

Alex is fairly new in the field, so no measured performance figures were available at the moment of writing.

3.4.2 The Avalon A12.

Machine type: RISC-based distributed-memory multi-processor.
Models: Avalon A12.
Operating system: AVALON micro kernel based Unix (Image compatible with Digital Unix).
Connection structure: Multistage variable (see remarks).
Compilers: Fortran 77, Fortran 90, HPF, ANSI C.
System parameters:

Model	A12
Clock cycle	3.3 ns
Theor. peak performance	
Per proc. (64-bit)	600 Mflop/s
Maximal (64-bit)	—
Memory/node	—
Memory (maximal)	—
Communication bandwidth	
Point-to-point	128–400 MB/s
Bisectional (full system)	—
No. of processors	—

Remarks:

The Avalon technical documentation is not entirely helpful in providing complete information with regard to system configurations. Therefore the list of system parameters above is somewhat incomplete. The A12 will be based on the DEC Alpha 21164 RISC procesor. This processor has a clock cycle of 3.3 ns. Because the Alpha 21164 has dual floating-point arithmetic pipes it will deliver a theoretical peak performance of 600 Mflop/s. The total performance of the system, however, cannot be specified because the maximum number of processors is not given. In addition to the usual first and second level cache that reside on chip, a 1 MB third level cache is provided on each A12 CPU card. The bandwidth to/from the first level cache is sufficient to transport two operands to the CPU and to ship one result back in one cycle. The second level cache has two-thirds of is bandwidth, while the third level cache has the capability of providing an 64-bit word every two cycles. The bandwidth to/from memory is 400 MB/s or one 64-bit word every 6 cycles. The memory has two-way interleaved banks but the size of the memory is not specified in the documentation.

Each CPU card contains a Alpha 21164 processor, the third level or B cache and the local memory for that node. Twelve CPU cards can be housed in one crate which has a full crossbar backplane. This yields a internode bandwidth of slightly under 400 MB/s between the cards within one crate. Apart from the 12 slots for CPU cards, there are two extra dual channel slots that can accomodate communication cards that provide the connections with other crates. For the in-crate crossbar CMOS technology is used.

However, for the intercrate connections ECL logic is employed. The actual connections between crates are made by coaxial cables. This way of connection provides a large flexibility in the overall interconnection topology: one could build trees or toruses or a secondary level crossbar (is the last case one crate should be filled entirely with communication cards to build a 144 processor system). The communication speed between crates is less fast (but still respectable): 128 MB/s.

I/O can be configured in various ways: It is possible to put 32-bit or 64-bit PCI expansion cards on each CPU card to obtain what Avalon calls "Type 1 I/O nodes". Also, a direct switch connection via a variant of the communication card can be made to the outside world. Depending on the number of cards the bandwidth is 400 or 800 MB/s for this type 3 I/O node. The type 2 I/O node is in fact a dedicated TCP/IP connection as needed for the control workstation as required by the system.

Measured Performances:

The A12 is expected to be available by the first quarter of 1996. As yet no systems are benchmarkable. So, no performance figures are known at this moment.

3.4.3 The C-DAC PARAM 9000/SS.

Machine type: RISC-based distributed-memory multi-processor.
Models: P9S/4–P9S/200.
Operating system: PARAS 9000/SS (Mach-like micro-kernel).
Connection structure: Multistage crossbar.
Compilers: Fortran 77, Fortran 90, HPF, ANSI C, C++ (soon).

System parameters:

Model	P9S
Clock cycle	16.6 ns
Theor. peak performance	
Per proc. (64-bit)	60 Mflop/s
Maximal (64-bit)	12 Gflop/s
Memory/node	$\leq$128 MB
Memory (maximal)	$\leq$25.6 GB
Communication bandwidth	
Point-to-point	10–40 MB/s
Bisectional (full system)	3.2 GB/s
No. of processors	4–200

Remarks:

The PARAM 9000/SS is the third generation of systems that is produced by C-DAC, the Centre for Development of Advanced Computing, an institute in India that has as its mission to develop an manufacture "state-of-the-art open architecture supercomputers". This system, however, is the first one to be marketed abroad. The machine is based on the Sun SuperSparC II as a processing node. The nodes are connected by a multistage crossbar with dynamically adaptive wormhole routing which is highly useful in terms of fault-tolerance. The point-to-point bandwidth is 10 MB/s per link. With a maximum of 4 links this bandwidth can be scaled up to 40 MB/s. The bisectional bandwidth for a full 200-node system is a very respectable 3.2 GB/s. For every four compute nodes one I/O node can be configured for distributed I/O.

The amount of available software shows that the PARAM 9000/SS is not a first-generation system. Apart from Fortran 77, Fortran 90, HPF, and C++ are available and the CORE, MPI, and PVM message passing interfaces are available. There is a parallel debugger, a proprietary performance evaluation tool called AIDE, while TOTALVIEW can be delivered at request.

In addition, a library of parallel routines, PARUL, is available. This library contains PVM versions of dense linear algebra routines, eigenvalue routines, and FFTs.

Measured Performances:

No measured performances of the PARAM 9000/SS are available. The performance of the computing node is rather optimistically estimated to be 60 Mflop/s for a 60 MHz processor. It is not very likely that the processing node will attain even half of this performance in practice. Even then, the system could be quite interesting in terms of price/performance.

3.4.4 The Cray Research Inc. T3E.

Machine type: RISC-based distributed-memory multi-processor.
Models: T3E.
Operating system: UNICOS MAX (micro-kernel Unix).
Connection structure: 3-D Torus.
Compilers: CFT77_M (Fortran 77 with extensions), C.
Vendors information Web page: `http://www.cray.com/PUBLIC/T3E/`.
System parameters:

Model	T3E
Clock cycle	3.3 ns
Theor. peak performance	
Per proc. (64-bit)	600 Mflop/s
Maximal (64-bit)	1229 Gflop/s
Main memory	$\leq$4096 GB
Memory/node	$\leq$ 2 GB
Communication bandwidth	300 MB/s
No. of processors	16–2048

Remarks:

The T3E is the second generation of DM-MIMD systems from CRI. Lexically, it follows in name after its predecessor T3D which name referred to its connection structure: a 3-D torus. In this respect it has still the same interconnection structure as the T3D. In many other respects, however, there are quite some differences. A first and important difference is that no front-end system is required anymore (although it is still possible to connect to a Cray T90). The systems up to 128 processors are air-cooled. The larger ones, from 256–2,048 processors, are liquid cooled.

The T3E uses the DEC Alpha 21164 RISC processor for its computational tasks just like the Avalon A12. Cray stresses, however, that the processors are encapsulated in such a way that they can be exchanged easily for any other (faster) processor as soon as this would be available without affecting the macro-architecture of the system.

Each node in the system contains one processing element (PE) which in turn contains a CPU, memory, and a communication engine that takes care of communication between PEs. The bandwidth between nodes is quite high: 300 MB/s. Like the T3D, the T3E has hardware support for fast synchronisation. E.g., barrier synchronisation takes only one cycle per check.

In the microarchitecture most changes have taken place with the transition from the T3D to the T3E. First, there is only one CPU per node instead of two, which removes a source of asymmetry between processors. Second, the new node processor has a 96 KB 3-way set-associative secondary cache which may relieve some of the problems of data fetching that were present in the T3D where only a primary cache was present. Third, the Block Transfer Engine has been replaced by a set of E-registers that are believed to be much more flexible and at least removes some odd restrictions on the size of shared arrays and the number of processes when using Cray-specific PVM. An interesting additional feature is the availability of 32 contexts per processor which opens the door for multiprocessing.

In the T3D all I/O had to be handled by the front-end, a system at least from the Cray Y-MPE class. In the T3E distributed I/O is present. For every 8 PEs an I/O channel can be configured in the air-cooled systems and 1 I/O channel per 16 nodes in the liquid-cooled systems. The maximum bandwidth for a channel is about 1 GB/s, the actual speed will be in the order of 700 MB/s.

The T3E supports various programming models. Apart from PVM 3.x for message passing and HPF for data distribution, a Cray proprietary work sharing model, called CRAFT, can be employed. Cray views HPF and Fortran 90 array syntax as subsets of the CRAFT model. Within this model data can be exchanged implicitly, thus looking effectively as a shared-memory system to the user. As several other vendors, Cray has extended/altered the implementation of PVM to enhance the communication performance. For small messages this can give an improvement of a factor 3 (20–25 μs instead of 70–80 μs). For SPMD programs channel send/receive functions can be used which reduces the communication time to 4–5 μs. The faster implementations are not portable, however.

Measured Performances:

The Cray T3E has only recently been announced (November 1995). At this moment no performance figures are available.

3.4.5 The Fujitsu AP1000.

Machine type: RISC-based distributed-memory multi-processor.
Models: AP1000.
Operating system: Cell OS (transparent to the user) and SunOS (Sun's Unix variant) on the front-end system.
Connection structure: T-net (2-D torus), B-net (common bus + hierarchical ring), S-net (tree) (see remarks).
Compilers: Fortran 77 and C with extensions.
System parameters:

Model	AP1000
Clock cycle	40 ns
Theor. peak performance	
Per proc. (64-bit)	12.5 Mflop/s
Maximal (64-bit)	12.8 Gflop/s
Main memory	≤16 GB
Memory/node	16 MB
Communication bandwidth	
B-net	50 MB/s
T-net	25 MB/s
No. of processors	8–1024

Remarks:

The AP1000 is put together from computing cells each of which contains a 25 MHz SPARC processor (IU) and an additional floating-point processor (FPU). The processor cells are complemented by routing- and message controllers, a B-net interface (see below), cell memory, and cache memory (128 KB). The peak performance of the FPU is estimated to be 12.5 Mflop/s which brings the aggregate peak rate to 12.8 Gflop/s for a full 1024 cell system. The system is front-ended by a Sun 4 machine.

Fujitsu has attempted to diminish the communication problems that are inherent to DM-MIMD machines by implementing different networks for broadcasting and collection of data (the B-net), for synchronisation (the S-net), and for communication on the processor grid (the T-net). As the broadcasting or multicasting (i.e., broadcasting to a selected subset) of data often constitutes a bottleneck in the execution of a computational task, the B-net has a two times higher bandwidth than the interprocessor T-net (50 vs. 25 MB/s). Because the gather and scatter of data over the processors is generally less structured a combination od a common bus and a hierarchical ring structure is used. The B-net interface has FIFO buffers and scatter-gather controllers to allow for sending/receiving data independent the other active components in the cell. The message controller seeks to minimise the overhead for data transfer setup and relieves the IU from doing the message passing proper.

For the T-net which connects the cells in a 2-D grid the transfer speed is two times lower than that of the B-net, but as data movement will often be more regular, it is expected to give good throughput, especially as a new conflict-free wormhole routing scheme has been implemented by allocating routed messages to alternating buffer pairs in the intermediate cells. Experiments have shown relatively low message overhead for this system [9].

There is a tree-structured S-net for barrier synchronisation of processes with again quite low overheads (a maximum of 5.2 μs for a full configuration).

Recently an entry model of the AP1000, the AP1000C, is being offered. The AP1000C starts at a configuration of 8 processor cells instead of the original 64. Also the housing has been made more compact for this model, saving a factor 3 in space.

Measured Performances;

In [8] the performance on the solution of a full linear system on a 256 cell machine is given. A system of order 100 performed at about 40 Mflop/s, an order 300 system attained 180 Mflop/s, while a 1000×1000 system reached more than 300 Mflop/s. In [2] a speed of 2.3 Gflop/s on a dense system of order 25,600 on 512 cells.

3.4.6 The Fujitsu VPP300 series.

Machine type: Distributed-memory vector multi-processor.
Models: VX series, VPP300.
Operating system: UXP/VPP (a V5.4 based variant of Unix).
Connection structure: Full distributed crossbar.
Compilers: Fortran 90/VP (Fortran 90 Vector compiler), Fortran 90/VPP (Fortran 90 Vector Parallel compiler),C/VP (C Vector compiler), C, C++.
Vendors information Web page:
`http://www.fujitsu.co.jp/hypertext/Products/Info_process/vpp300/vpp300br.html`.
System parameters:

Model	VX	VPP300
Clock cycle	7/10 ns	7/10 ns
Theor. peak performance		
Per proc. (64-bit)	1.6/2.2 Gflop/s	1.6/2.2 Gflop/s
Maximal (64-bit)	6.4/8.8 Gflop/s	25.6/35.2 Gflop/s
Main memory	≤8 GB	≤32 GB
Memory/node	≤2 GB	≤2 GB
Memory bandwidth		
Memory banwidth/proc.	12.8/18.2 GB/s	12.8/18.2 GB/s
Communication bandwidth	400/570 MB/s	400/570 MB/s
No. of processors	1–4	1–16

Remarks:

The VPP300 is a succesor to the earlier VPP500. It is a much cheaper CMOS implementation of its predecessor with some important differences. First, no VPX200 front-end system is required anymore. Second, the crossbar that is used to connect the vector nodes is distributed. Therefore, the cost of a system is scalable: one does not need to buy a complete enclosure with the full crossbar for only a few nodes. The VX series is in fact a smaller version of the VPP300 with a maximum of 4 processors. Both the VX machines and the larger VPP300 systems are air-cooled. The systems are marketed either with a 10 ns or a 7 ns clock.

At this moment the VPP300 is officially only available with 16 processors connected by a direct crossbar. However, it is presumed that an announcement of larger systems will be made in the first quarter of 1996 in which multiple 16-processor machines are connected by a second level crossbar.

The architecture of the VPP300 nodes is almost identical to that of the VPP500: Each node, called a Processing Element (PE) in the system is a powerful (2.2 Gflop/s peak speed with a 7 ns clock) vector processor in its own right. The vector processor is complemented by a RISC scalar processor with a peak speed of 200 or 285 Mflop/s dependent on the clock speed. The scalar instruction format is 64 bits wide and may cause the execution of three operations in parallel. Each PE has a memory of up to

2 GB MB while a PE communicates with its fellow PEs at a point-to-point speed of 400 or 570 MB/s. This communication is cared for by separate Data Transfer Units (DTUs). To enhance the communication efficiency, the DTU has various transfer modes like contiguous, stride, sub array, and indirect access. Also translation of logical to physical PE-ids and from Logical in-PE address to real address are handled by the DTUs. When synchronisation is required each PE can set its corresponding bit in the SR. The value of the SR is broadcast to all PEs and synchronisation has occurred if the SR has all its bits set for the relevant PEs. This method is comparable to the use of synchronisation registers in shared-memory vector processors and much faster than synchronising via memory.

The Fortran compiler that comes with the VPP300 has extensions that enable data decomposition by compiler directives. This evades in many cases restructuring of the code. The directives are different from those as defined in the High Performance Fortran Proposal but it should be easy to adapt them. Furthermore, it is possible do define parallel regions, barriers, etc., via directives, while there are several intrinsic functions to enquire about the number of processors and to execute `POST/WAIT` commands. Furthermore, also a message passing programming style is possible by using the PVM or PARMACS communication libraries that are available.

Measured Performances:

The first VPP300 systems will be delivered in the first quarter of 1996 (first only with the 10 ns clock). Therefore, no performance figures are available yet.

3.4.7 The Hitachi SR2201 series.

Machine type: RISC-based distributed memory multi-processor.
Models: SR2201.
Operating system: HI-UX/MPP (Micro kernel Mach 3.0).
Connection structure: Hyper crossbar.
Compilers: Fortran 77, Fortran 90, Parallel Fortran, HPF, C, C++.
System parameters:

Model	SR2201
Clock cycle	6.7 ns
Theor. peak performance	
Per proc. (64-bit)	300 Mflop/s
Maximal (64-bit)	307 Gflop/s
Main memory	≤256 GB
Memory/node	≤256 MB
Communication bandwidth	300 MB/s
No. of processors	32–1024

Remarks:

The SR2201 is the second generation of distributed memory parallel systems of Hitachi. The basic node processor is again an Hitachi implementation of the PA-RISC architecture of HP running at a clock cycle of 6.7 ns. However, in contrast with its predecessor, the SR2001, in the SR2201 the node processors are somewhat modified to allow for "pseudo vector processing" (both hardware and instructions). This means that for operations on long vectors one does not have to care about the detrimental effects of cache misses that often ruin the performance of RISC processors unless code is carefully blocked and unrolled. First experiments have shown that this idea seems to work quite well. THe system supports distributed I/O with a possibility to connect disks to every node.

As in the earlier SR2001, the connection structure is a hyper (3-D) crossbar which connects all nodes directly at high speed (300 MB/s point-to-point). In February 1996 two 1024-node systems will be to in stalled at the Universities of Tokyo and Tsukuba respectively.

Like in some other systems as the Cray T3E (3.4.4) and the Meiko CS-2 (3.4.12), and the NEC Cenju-3 (3.4.14), one is able to directly access the memories of remote processors. Together with the very fast hardware-based barrier synchronisation this should allow for writing distributed programs with very low parallelisation overhead.

The following software products will be supported in addition to those already mentioned above: PVM, MPI, PARMACS, Linda, Express, FORGE90, and PARALLELWARE. In addition a numerical libraries (MATRIX/MPP, MATRIX/MPP/SSS) will be offered. These libraries support basic linear algebra operations with dense and band matrices, Fast Fourier Transformations, and skyline solvers.

Measured Performances:

Some preliminary (but not yet officially certified) results of class A NAS parallel benchmarks show that the SR2201 runs at about 1.3 Gflop/s on 16 processors for the MG benchmarks and about 700 Mflop/s for the CG benchmark also on 16 processors ([11]

3.4.8 The HP/Convex Exemplar SPP-1200.

Machine type: RISC-based distributed-memory multi-processor.
Models: SPP-1200.
Operating system: SPP-UX, based on OSF/1 AD microkernel.
Connection structure: Ring.
Compilers: Fortran, C.
Vendors information Web page: `http://www.convex.com/prod_serv/exemplar/exemplar.html`.
System parameters:

Model	SPP-1200
Clock cycle	8.3 ns
Theor. peak performance	
Per proc. (64-bit)	240 Mflop/s
Maximal (64-bit)	30.7 Gflop/s
Main memory	≤32 GB
Memory/node	≤256 MB
Communication bandwidth	
aggregate (see remarks)	16 GB/s, 4GB/s
No. of processors	4–128

Remarks:

The SPP-1200 is the second generation in Exemplar SPP series. In fact, in almost every respect the system is identical to its predecessor, the SPP-1000 except the clock cycle (10 instead of 8.3 ns) and the use of PA/RISC 7200 instead of 7100 processors. Because of the prefetch and poststore capabilities of the 7200 processors the number of floating-point operations per cycles should be somewhat higher than in the 7100 processor, thus increasing the floating-point performance beyond the amount that is caused by the reduction of the clock cycle. Up to 8 HP PA/RISC 7200 processors can be placed in what is called a *hypernode* by Convex. A maximal system consists of 16 nodes, i.e., 128 processors.

Within each hypernode up to 2 GB of memory can be accommodated which can be reached by the local processors via a crossbar with an aggregate bandwidth of 16 GB/s. The hypernodes in turn are connected to each other by a crossbar with an aggregate bandwidth of 4 GB/s. So, the system concept is somewhat hybrid: within a hypernode the machine is effectively a shared-memory system, while between hypernodes it is a distributed memory system. Each node supports local I/O, while external global I/O can be done at an aggregate rate of 4 GB/s.

The Exemplar programming environment complements the SPP-1200 at the software side. This environment includes a message passing programming model (PVM) and a virtual shared memory model which allows the user to have a shared-memory view of the system. The underlying communication is hidden from the user, thus enabling the execution of standard Fortran 77, C, or C++ programs. The efficiency of

this mode of operation is determined by the extent to which the original code is parallelisable. In many cases it might be enhanced using another (possibly message passing) implementation. The application compiler included in the Exemplar environment may help in parallelising the original program and in generating the necessary parallel code.
Measured Performances:

First results for the solution of a linear system of order $N = 1000$ are 123, 213, 383, and 656 Mflop/s for 1, 2, 4, and 8 processors (within one hypernode), respectively. In [2] also a speed of 3.72 Gflop/s is reported for the solution of a dense system of order 25,100.

3.4.9 The IBM 9076 SP2

Machine type: RISC-based distributed-memory multi-processor cluster.
Models: IBM9076 SP2.
Operating system: AIX (IBMs Unix variant).
Connection structure: Dependent on type of connection (see remarks).
Compilers: XL Fortran, XL C, XL C++.
Vendors information Web page: `http://ibm.tc.cornell.edu/ibm/pps/sp2/sp2.html`.
System parameters:

Model	9076 SP2
Clock cycle	15 ns
Theor. peak performance	
Per Proc. (64-bit)	267 Mflop/s
Maximal (64-bit)	34.1 Gflop/s
Memory/node	64–512/2048 MB (see remarks)
Communication bandwidth	
Point-to-point	20+ MB/s
Bisectional	25 GB/s
No. of processors	8–128

Remarks:

As a basis for the computational nodes in the SP2 RS/6000 processors with a clock cycle of 15 ns are used. This amounts to a peak performance of 266 Mflop/s per node because the floating-point units of the SP2 processors can deliver up to 4 results/cycle. The SP2 configurations are housed in columns that each can contain 8–16 processor nodes. This depends on the type of node employed: there are two types, thin nodes and wide nodes. Although the processors in these nodes are basically the same there are some differences. Wide nodes have the double amount of microchannel slots (8 instead of 4) as compared to the thin nodes. Furthermore, the maximum memory of a wide node can be 2 GB whereas the maximum for thin nodes is 512 MB. More important in terms of performance is the fact that the data cache of a wide node is four times larger than that of a thin node (256 KB instead of 64 KB) and that the memory bus is two times wider than that of a thin node (8 instead of 4 words/cycle). The latter differences explain than a performance gain of a factor 1.5 has been observed for wide nodes over the thin nodes. However, the newer Thin-node2 is except with regard to the number of micro-channel slots almost identical to a wide node. Also the performance is very simlar to that of a wide node (see Measured performance). IBM envisions the wide node more or less as server for a column and recommends configurations of one wide node packaged with 14 thin nodes per column (although this may differ with the needs of the user). The SP2 is accessed through a front-end control workstation that also monitors system failures. Failing nodes can be taken off line and exchanged without interrupting service. In addition, fileservers can be connected to the system while every node can have up to 2 GB. This can greatly speed up applications with significant I/O requirements.

There is a choice in the way communication is done: Ethernet, Token Ring, FDDI, etc., are all possible. However, it is also possible to connect the processors by an optional high-speed switch with a speed of 40 MB/s. Therefore, depending on the communication type the speed can range from 1–40 MB/s. The high-speed switch has some redundancy built into it for greater reliability. The structure is that of a multi-stage crossbar (Ω-switch).

Applications can be run using PVM or Express. FORGE 90 MIMDizer can be used to assist in parallelising the code by generating the necessary calls to PVM or Express communication routines. Under Express Fortran 77 or 90, C, and C++ can be used. Also High Performance Fortran is supported. IBM uses its own PVM version from which the data format converter XDR has been stripped. This results in a lower overhead at the cost of generality. Recently an optimised version of MPI has also become available.

Measured Performances:

In [2] a performance of 88.4 Gflop/s in solving a dense linear system of order N =73,500 with 512 Thin-node2 nodes. In [13] it appears that at 128 nodes the Thin-node2 is consistently slower than the Wide-node1. The differences range from 4-20% with an average of about 9%. The Wide-node1 times for the Class B problems EP, MG, CG, FT, IS, LU, SP, and BT are 4.99, 2.46, 25.44, 14.52, 1.98, 47.8, 54.8, and 67.0 seconds, respectively.

3.4.10 The Intel Paragon XP.

Machine type: RISC-based distributed-memory multi-processor.
Models: Paragon XP/S (MP), XP/E
Operating system: OSF/1, SunMos.
Connection structure: 2-D mesh (torus).
Compilers: Fortran 77, ADA.
Vendors information Web page: `http://www.ssd.intel.com/pubs.html`.
System parameters:

Model	Paragon XP/S	Paragon XP/E
Clock cycle	20 ns	20 ns
Theor. peak performance		
Per Proc. (64-bits)	75 Mflop/s	75 Mflop/s
64-bits precision	300 Gflop/s	2.1 Gflop/s
Main memory	≤128 GB	≤4.5 GB
Memory/node	≤128 MB	≤128 MB
Communication bandwidth	200 MB/s	200 MB/s
No. of processors	64–4000	4–32

Remarks:

The Paragon is a commercialised offspring of the experimental Touchstone Delta system. The latter machine was built for the Concurrent Supercomputing Consortium at CalTech. The Delta system used i860 processors as computational elements in its nodes but, unlike its predecessor, the iPSC/860, the nodes were not arranged in a hypercube topology but in a 2-D grid (for many physical simulation phenomena, as well as for the solution of linear systems this is a quite natural topology). The Delta system proved to be quite fast for a variety of problems (a speed of 11.9 Gflop/s was reported for an order 20,000 full linear system). The Paragon machine should do better because of the faster i860/XP processor that is used in the nodes. In addition, the i860/XP has processor communication hardware on-chip which makes the communication bandwidth faster.

In November 1993 the Paragon XP/E was introduced. This is an entry-level system with the same characteristics as the XP/S and up to 32 processors. The maximal configuration of the XP/E, the XP/E-28N has 32 nodes of which 28 are compute nodes. The others are used for assisting the routing, I/O, and other operating system tasks.

The Paragons retain compatibility with the former iPSC/860 systems, an Intel hypercube system preceding them. In particular the the transparent parallel Distributed File System can be used in applications migrated from the iPSC/860. The Paragon has its own parallel file system.

In 1995 the MP (Multi Processor) node was introduced. In such an MP node 3 i860/XP processors reside on one board and the processors share one address space.

Fortran and C compilers take care of the automatic parallelisation within a MP node. The Intel-provided information claims a better performance than with single processor nodes. Until now this seems consistently but not spectacularly true (see Measured Performances).

Measured Performances:

As on many systems a results are available for the solution of a large dense linear system. In [2] a speed of 281.1 Gflop/s is reported for a system of size 128,600 on a 6768-node ensemble of XP/S MP systems. No actual systems of this size are in operation. Results as quoted above are obtained by systems that are put together for the occasion. In [13] results for the class B EP, MG, and FT benchmarks, the times obtained on 512 processors were 3.98, 7.01, and 16.17 seconds for the single-node XP, while on the MP-node XP of the same size these times were 2.98, 6.72, and 12.4 seconds, respectively.

3.4.11 The Matsushita ADENART.

Machine type: RISC-based distributed-memory multi-processor.
Models: ADENART64, ADENART256.
Operating system: Internal OS transparent to the user, SunOS (Suns Unix variant) on the front-end system.
Connection structure: HX-net (see remarks).
Compilers: ADETRAN, an extended Fortran 77.
System parameters:

Model	ADENART64	ADENART256
Clock cycle	50 ns	50 ns
Theor. peak performance		
Per Proc. (64 bits)	10 Mflop/s	10 Mflop/s
Maximal (64 bits)	0.64 Gflop/s	2.56 Gflop/s
Main memory	0.5GB	0.5GB
Memory/node	8MB	2MB
Communication bandwidth	20 MB/s	20MB/s
No. of processors	64	256

Remarks:

The ADENART has an interesting interconnection structure that is somewhere halfway between a crossbar and a grid. The processors are organised in planes, where for each plane all processors are connected by a crossbar. Between planes there is a connection structure that connects each crossbar node in a plane directly with its corresponding counterpart on all other planes. So, for a processor (i, j) in plane data that are required by processor (k, j) in the same plane can be transported by simply shifting it through the in-plane crossbar which can be accomplished in one step. For processors in different planes the number of steps is at most two. In the first step the data is routed to the right crossbar node in one plane and after being send to the plane where the target processor resides, send there from the corresponding crossbar node to the processor that requires them. The connection structure is called HX-net by Matsushita. Because of the connection structure the number of processors is constrained to be of the form 2^{2n} and presently in the two model numbers available n is 3 or 4 (a machine with 1024 processors, $n = 5$, is being considered). As remarked, the complexity of the network is lower than that of a crossbar: $O(n^{3/2})$ instead of $O(n^2)$ while the efficiency is half of that of a crossbar: a maximum of 2 steps instead of 1.

The processors consist of a proprietary RISC processor with a peak speed of 20 Mflop/s in perfect pipeline mode, however, a "sustained speed" of 10 Mflop/s is quoted by Matsushita to arrive at the peak performance given in the system parameters list above. The inter-processor bandwidth is 20 MB/s, which is quite reasonable with respect to the processor speed. At this moment nothing is known about the message setup overhead however. Curiously enough, the amount of memory per node is 4 times larger for the ADENART64 than for the 256-processor model (8MB against 2MB per node). The latter memory size seems fairly small for a processor node that is meant to process large amounts of data. The front-end machine that hosts the ADENART is a Solbourne (Sun 4 compatible) workstation.

Measured Performances:

In [3] a speed of 475 Mflop/s for a PDE solver using a Splitting-up Conjugate Gradient algorithm was reported for an ADENART256. Also, results for some Livermore kernels were given of which the highest reported speed was 520.1 Mflop/s. In the article there are some complaints about the rigidness of existing benchmark codes which should be a disadvantage for massively parallel computers. It could of course also be argued that massively parallel machines are too rigid to run general codes well. In [13] some class A results for the ADENART256 are quoted: EP, FT, IS, SP, and BT times are 32.9, 72.7, 46.6, 209.9, 314.1 seconds respectively.

3.4.12 The Meiko Computing Surface 2.

Machine type: Distributed-memory multi-vectorprocessor.
Models: Computing Surface 2.
Operating system: Internal OS transparent to the user, SunOS (Sun's Unix variant) on the front-end system.
Connection structure: Multistage crossbar.
Compilers: Extended Fortran 77, ANSI C.
Vendors information Web page: `http://www.meiko.com`.
System parameters:

Model	Computing Surface 2
Clock cycle	20 ns
Theor. peak performance	
Per Proc. (64 bits)	200, 40 Mflop/s
Maximal (64 bits)	204.8 Gflop/s
Main memory	$\leq$ 128 GB
Memory/node	32–128, 32–512MB
Communication bandwidth	—
No. of processors	8–1024 PEs

Remarks:

The CS-2 features 8-1,024 processor elements (PEs) which can be either scalar or vector nodes. Apart from a separate communications module, these PEs contain either a SuperSparc or a SuperSparc + 2 μVP vectorprocessors. The speed of a scalar PE is estimated to be 40 Mflop/s (at a 20 ns clock) and 200 Mflop/s for the vector PEs for 64-bit precision. The μVP modules are manufactured by Fujitsu. The speed at 32-bit precision is doubled with respect to 64-bit operation and, unlike the earlier Fujitsu VP products, use IEEE 754 floating-point format. The memory has 16 banks and to avoid memory bank conflicts the CS-2 has the interesting option to have scrambled allocation of addresses, thus guaranteeing good access at potential problematic strides 2, 4, etc.

The point-to-point communication speed is 100 MB/s (50 MB/s in each direction). Because the communication happens through multi-level crossbars, called "layers" by Meiko, the aggregate bandwidth of the system scales with the number of PEs, with a very respectable latency of 200 ns per layer. As the maximum configuration of the machine contains 1,024 PEs, the theoretical peak performance at 64-bit precision is 200 Gflop/s. It is possible to connect each PE to its own I/O devices to have scalable parallel I/O with the scaling of other resources.

The Portland Group which has won some renown for its excellent i860 compilers has developed the compilers for the CS-2. These include Fortran 77 and ANSI C but also Fortran 90. The current compiler already offers data distribution directives as proposed in [6].

In the USA the machine will be marketed by Meiko, however, in Europe and the rest of the world marketing is done by Parallel Computing Industries, a consortium of Meiko, Parsys, and Telmat.

Measured Performances:

In [2] a speed of 5.0 Gflop/s on a 64 processor CS-2 is reported for the solution of an order 18688 dense linear system. From the NAS parallel benchmarks some results on a 128 processor machine are given for class B problems: EP took 21.16 seconds while 6.52 seconds was measured for the MG problem.

3.4.13 The nCUBE 3.

Machine type: Distributed-memory multi-processor.
Models: nCUBE 3.
Operating system: Internal OS transparent to the user, SunOS (Sun's Unix variant) on the front-end system.
Connection structure: Hypercube.
Compilers: Extended Fortran 77, ANSI C, C++.

System parameters:

Model	nCUBE 3
Clock cycle	10 ns
Theor. peak performance	
Per Proc. (64-bits)	100 Mflop/s
Maximal(64-bits)	1024 Gflop/s
Main memory	$\leq$1 TB
Memory/node	$\leq$1 GB
Communication bandwidth	
Point-to-point	115 MB/s
No. of processors	8–10244

Remarks:

The nCUBE 3 is presently the only commercially available machine with a hypercube structure. The nCUBE uses in-house developed processors implemented in 0.5 μm CMOS which have a performance of 100 Mflop/s in 64-bit precision (in contrast to the former 2S model the new processor is entirely 64-bit wide). The node processor has 8 KB instruction an data caches, both 2-way set associative. Furthermore, each processor has miss and write buffers of four operands deep that allows for 4 cache misses (or deferring four cache writes) before disturbing the data cache.

There are 16 outward DMA channels per node (8 send and 8 receive) for inter-processor communication while an additional one is used for the distributed I/O system which therefore has the nice property that it scales with the number of nodes. The speed of these I/O nodes is 20 MB/s full-duplex. The communication latency is quite low: about 3 μs while the single channel bandwidth is 50 MB/s. By "folding" multiple channels higher point-to-point bandwidth can be achieved. For the instruction cache an autoprefetch facility is implemented while prefetch for the data cache is compiler directed. On 1024 processors with 6 ports folded a bisectional bandwidth of 45.5 GB/s can be realised.

Apart from the fixed wormhole routing scheme that already was employed in the former nCUBE systems, a new fault-tolerant adaptive routing scheme is available. This scheme is also essentially a wormhole routing but with the additional constraint that after the first hop the distance to the target node should strictly decrease. Therefore, no cycles can occur and delivering of a message is guaranteed to be done in a finite number of hops.

Within the hypercube sub-cubes can be allocated to accommodate more users. A queue of tasks is set up with (sub)-cubes of the required size. Programs may be written to determine the sub-cube dimensions just before execution.

Measured Performances:

The first system is expected to be realised in the 2nd quarter of 1995, so no real performance figures are available. Simulations showed a single node speed of 96 Mflop/s on a matrix-matrix multiply and of 40 Mflop/s on a matrix-vector multiply.

3.4.14 The NEC Cenju-3.

Machine type: RISC-based distributed-memory multi-processor.
Models: Cenju-3S, Cenju-3.
Operating system: EWS-UX/V (Unix variant based on Unix System V.4).
Connection structure: Multi-stage crossbar.
Compilers: Fortran 77, ANSI C.
System parameters:

Model	Cenju-3S	Cenju-3
Clock cycle	20 ns	13.3 ns
Theor. peak performance		
Per Proc. (64 bits)	33 Mflop/s	50 Mflop/s
Maximal (64 bits)	533 Mflop/s	12.8 Gflop/s
Main memory	< 1 GB	< 16 GB
Memory/node	< 64 MB	< 64 MB
Communication bandwidth	40 MB/s	40MB/s
No. of processors	8–16	16– 256

Remarks:

The name Cenju-3 suggests that there have been predecessors, Cenju-1 and Cenju-2. This is indeed the case but these systems have only been used internally by NEC for research purposes and were never officially marketed. The Cenju-3 is based on the same RISC processor as the Silicon Graphics Challenge, the MIPS R4400 processor (see 3.3.8). It is confusing that the peak performance of the processor is rated differently by Silicon Graphics and NEC respectively. The lower estimates of 33 vs. 50, and 50 vs. 75 Mflop/s as quoted by NEC seem to be more realistic. All processors have apart from their on-chip primary cache a secondary cache of 1 MB to mitigate the problems that arise in the high data usage of the CPU.

The interconnection type used in the Cenju is a multistage crossbar build from 4 × 4 modules that are pipelined. So, in a full configuration the maximal number of levels in the crossbar to be traversed is four. The peak transfer rate of the crossbar is quoted as 40 MB/s irrespective of the data placement.

The system needs a front-end processor of the EWS4800 type (functionally equivalent to Silicon Graphics workstations). The I/O requirements have to be fulfilled by the front-end system as the Cenju does not have local (distributed) I/O capabilities.

There is some software support that should make the programmer's life somewhat easier. The library PARALIB/CJ contains proprietary functions for forking processes, barrier synchronisation, remote procedure calls, and block transfer of data. Like on the Cray T3D (3.4.4) and on the Meiko CS-2 (3.4.13) the programmer has the possibility to write/read directly to/from non-local memories which avoids much message passing overhead.

Measured Performances:

Delivery of the systems have started in the second quarter of 1994 but no performance figures are available ever published for the Cenju-3.

3.4.15 The Parallel Computing Industries system.

This system is identical to the Meiko CS-2 (see 3.4.12 for details). Parallel Computing Industries does the marketing of this system in Europe, while it is offered as the CS-2 in the USA and the rest of the world.

3.4.16 The Parsys TA9000.

Machine type: Distributed-memory multi-processor.
Models: TA9800 (TA9400, TA9500).
Operating system: Idris (a real-time sub-Unix variant).
Connection structure: Multi-stage crossbar.
Compilers: Extended Fortran 77, ANSI C, Pascal, Modula 2.
System parameters:

Model	TA9800
Clock cycle	4.3 ns
Theor. peak performance	
Per Proc. (32 bits)	233 Mflop/s
Maximal (32 bits)	119.3 Gflop/s
Main memory	≤32 GB
Memory/node	≤64 MB
Communication bandwidth	25 MB/s/link
No. of processors	≤512

Remarks:

The Parsys TransAlpha TA9000 series systems are the successors of the Parsys SN9000 machines. The latter had the Thomson T9000 transputer as their basic processors. The new TA9000 systems use the DEC Alpha 21066 transputers for that purpose.

The TA9000 is roughly 10 times faster than its predecessor, the SN9000, which had a maximal speed of roughly 25 Mflop/s per node. However the communication speed has remained the same still using T9000 transputers for the internode communication. The use of the T9000 as a communication engine enables employment of the fast C104 communication switch. The same multistage crossbar switch is also used in the Meiko CS-2 (see 3.4.12) and allows for very good latency and bandwidth characteristics (although at this stage no figures are available to show how much of these are realised).

Apart from the largest model, the SN9800, there are smaller models, like the desktop models SN9400 and SN9500. The latter houses up to 12 processors and can be driven either as a Sun SparC back-end system or as a network device via Ethernet. An optimised PVM library is available for standard message passing programs.

Measured Performances:

Although the TransAlpha machines were announced and available from May 1995, until now no measured performances are known to the author.

3.4.17 The Parsytec GC/Power Plus.

Machine type: RISC-based distributed-memory multi-processor.
Models: Parsytec GC/Power Plus.
Operating system: Unix on host processor, Parix (GC OS, transparent to the user).
Connection structure: 2-D grid.
Compilers: Fortran 77, ANSI C, Pascal, Modula-2.
Vendors information Web page: `http://www.parsytec.de/products/gc1.html`.
System parameters:

Model	GC/Power Plus
Clock cycle	7.5 ns
Theor. peak performance	
Per proc. (64-bits)	266 Mflop/s
Maximum (64-bits)	...Gflop/s
Main memory	... GB
Memory/node	16–128 MB
Communication bandwidth	
Point-to-point	8.8 MB/s
No. of processors	8–...

Remarks:

The Parsytec GC Power Plus system (GC standing for Grand Challenge) is based on the IBM/Motorola MC604 processor. Initially, the CG series was planned to be built around the T9000 transputer. However, the T9000 is still not available in sufficient quantity and quality. Therefore, the MC601 was chosen to replace the T9000. It is not clear what is the maximum configuration for the Power Plus system. Therefore we could not give maximum performance, memory capacity, etc.

Each "node" (the term node not used here in the usual sense) of a GC Power Plus system contains two MC601 processors and 4 T805 transputers which are responsible for the communication. Two nodes are placed on one board, while 4 boards are interconnected with 16 C004 static routers to maintain the intercommunication within a so-called GC-cube. For every 4 nodes one redundant node is present for fault-tolerance. To complement the computing power, a parallel I/O system, the Parallel Storage System is available to aid in the handling of large-scale applications which require massive I/O.

The communication speed of the system is presently not particularly high with respect to the processor speed (although Parsytec from its PowerStone project claims that the present choice of computational versus communication capacity is optimal from the viewpoint of cost-effectiveness). There are plans to use T9000 transputers for communication instead of the present T805s and to replace the C004 routers by its successor, the C104. This would speed up the communication by at least a factor of 10, making the computation/communication speeds more balanced.

In the Parsytec CC series, also equipped with the MC604 but with ATM HS links, the link speed is up to about 75 MB/s. The CC series, however, is primarily directed to the embedded systems market.

The PARIX operating system is Unix-like. It allows to specify various virtual topologies onto the actual 2-D grid topology to match possible natural application topologies. Besides Parsytecs own communication library, PVM and PARMACS are available. An MPI communication library is presently developed.

Measured Performances:

Early experiments have been done on a 64-processor system. On 4 processors the solution of an order $N = 1000$ dense linear system attained a speed of 141 Mflop/s. For a scaled-up system on 32 processors a speed of 1007 Mflop/s was found, while for the NAS Embarassingly Parallel benchmark (see [13] a speed of 2.8 Gflop/s was observed on 64 processors.

4 Systems Disappeared from the List

As already stated in the introduction the list of systems is not complete. On one hand this caused by the sheer number of systems that are presented to the market and are often very similar to systems described above (for instance, the Volvox system not listed is very similar but not equivalent to the listed Alex AVX system and there are numerous other examples). On the other hand there many systems that are still in operation around the world, often in considerable quantities that for other reasons are excluded. The most important reasons are:

- The system is not marketed anymore. This is generally for one of two reasons:
  - The manufacturer is out of business.
  - The manufacturer has replaced the system by a newer model of the same type or even of a different type.
- The system has become technologically obsolete in comparison to others of the same type. Therefore, listing them is not sensible anymore.

Below we present a table of systems that fall into one of the categories mentioned above. We think this may have some sense to those who come across machines that are still around but are not the latest in their fields. It may be interesting at least to have an indication how such systems compare to the newest ones and to place them in context.

It is good to realise that although systems have disappeared from the section above they still may exist and are actually sold. However, their removal stems in such cases mainly from the fact that they are not serious candidates for high-performance computing anymore.

The table is, again, not complete and admittedly somewhat arbitrary. The data are in a highly condensed form: the system name, system type, theoretical maximum performance of a fully configured system, and the reason for their disappearance is given. The arbitrariness lies partly in the decision which systems are still sufficiently of interest to include and which are not. For instance, the Convex C-1 is not included, while the Alliant FX/80 is. The reason is that the C-1 is conceptually is not different from the later generations of single-processor Convex vector processors, while the Alliant FX/80 was fairly different from its successor the Alliant FX/2800.

Machine: Alliant FX/2800.
Type: Shared memory vector-parallel, max. 28 processors.
Theoretical Peak performance: 1120 Mflop/s
Reason for disappearance:Manufacturer out of business.

Machine: BBN TC2000.
Type: Virtual shared memory parallel, max. 512 processors.
Theoretical Peak performance: 1 Gflop/s
Reason for disappearance: Manufacturer has discontinued marketing parallel computer systems.

Machine: Cambridge Parallel Processing DAP Gamma.
Type: Distributed memory processor array system.
Theoretical Peak performance: 1.6 Gflop/s (32-bit)
Reason for disappearance: replaced by newer Gamma II series (see 3.2.2).

Machine: Convex C3200, C3400, C3800.
Type: Shared memory vector-parallel, max. 8 processors (C3880).
Theoretical Peak performance: 960 Mflop/s
Reason for disappearance: replaced by newer C4 series (see 3.3.3).

Machine: Convex Meta Series.
Type: Distributed memory network of workstations.
Theoretical Peak performance: 200 Mflop/s per processor
Reason for disappearance: replaced by newer SPP-1200 series (see 3.4.8).

Machine: Convex SPP-1000.
Type: Distributed memory RISC based system, max. 128 processors.
Theoretical Peak performance: 25.6 Gflop/s
Reason for disappearance: replaced by newer SPP-1200 series (see 3.4.8).

Machine: Cray Computer Corporation Cray-2.
Type: Shared memory vector-parallel, max. 4 processors.
Theoretical Peak performance: 1.95 Gflop/s
Reason for disappearance: Manufacturer out of business.

Machine: Cray Computer Corporation Cray-3.
Type: Shared memory vector-parallel, max. 16 processors.
Theoretical Peak performance: 16 Gflop/s
Reason for disappearance: Manufacturer out of business.

Machine: Cray Research Inc. APP.
Type: Shared memory RISC based system, max. 84 processors.
Theoretical Peak performance: 6.7 Gflop/s
Reason for disappearance: Product line discontinued, gap expected to be filled by Cray J90 (see 3.3.1).

Machine: Cray T3D.
Type: Distributed memory RISC based system, max. 2048 processors.
Theoretical Peak performance: 307 Gflop/s
Reason for disappearance: replaced by newer T3E (see 3.4.4).

Machine: Cray Research Inc. Cray Y-MP, Cray Y-MP M90.
Type: Shared memory vector-parallel, max. 8 processors.
Theoretical Peak performance: 2.6 Gflop/s
Reason for disappearance: replaced by newer T90 (see 3.3.1).

Machine: Cray Y-MP C90.
Type: Shared memory vector-parallel, max. 16 processors.
Theoretical Peak performance: 16 Gflop/s
Reason for disappearance: replaced by newer T90 (see 3.3.1).

Machine: Digital Equipment Corp. Alpha farm.
Type: Distributed memory RISC based system, max. 4 processors.
Theoretical Peak performance: 0.8 Gflop/s
Reason for disappearance: replaced by newer AlphaServer clusters (see 3.3.4).

Machine: Fujitsu VPP500 series.
Type: Distributed memory multi-processor vectorprocessors, max. 222 processors.
Theoretical Peak performance: 355 Gflop/s
Reason for disappearance: replaced by the VPP300 series (see 3.4.6).

Machine: Fujitsu VPX200 series.
Type: Single-processor vectorprocessors.
Theoretical Peak performance: 5 Gflop/s
Reason for disappearance: replaced by the VPP300 series (see 3.4.6).

Machine: Hitachi SR2001 series.
Type: Distributed memory RISC based system, max. 128 processors.
Theoretical Peak performance: 23 Gflop/s
Reason for disappearance: Replaced by the newer SR2201 (see 3.4.7).

Machine: IBM ES/9000 series.
Type: Shared memory vector-parallel system, max. 6 processors.
Theoretical Peak performance: 2.67 Gflop/s
Reason for disappearance: IBM does not pursue high-performance computing by this product line anymore.

Machine: IBM Power/4.
Type: Shared memory RISC based system, max. 4 processors.
Theoretical Peak performance: 336 Mflop/s
Reason for disappearance: Product line discontinued, gap expected to be filled by SP2 (see 3.4.9).

Machine: IBM SP1 series.
Type: Distributed memory RISC based system, max. 64 processors.
Theoretical Peak performance: 8 Gflop/s
Reason for disappearance: Replaced by the newer SP2 (see 3.4.9).

Machine: Intel iPSC/860.
Type: Distributed memory parallel hypercube, max. 128 processors.
Theoretical Peak performance: 7.7 Gflop/s
Reason for disappearance: replaced by newer Intel Paragon XP (MP) series (see 3.4.10).

Machine: Kendall Square Research KSR2.
Type: Virtually shared memory parallel, max. 1088 processors.
Theoretical Peak performance: 400 Gflop/s
Reason for disappearance: Kendall Square has terminated its business.

Machine: Meiko CS-1 series.
Type: Distributed memory RISC based system.
Theoretical Peak performance: 80 Mflop/s per processor
Reason for disappearance: Replaced by the newer CS-2 (see 3.4.12).

Machine: NEC SX-2.
Type: Single-processor vector processors.
Theoretical Peak performance: 1.3 Gflop/s
Reason for disappearance: replaced by newer SX-4 series (see 3.3.5).

Machine: NEC SX-3R.
Type: Shared memory multi-processor vector processors, max. 4 processors.
Theoretical Peak performance: 1.3 Gflop/s
Reason for disappearance: replaced by newer SX-4 series (see 3.3.5).

Machine: Parsys SN9000 series.
Type: Distributed memory RISC based system, max. 2048.
Theoretical Peak performance: 51.2 Gflop/s
Reason for disappearance: Replaced by the newer TA9000 (see 3.4.16).

Machine: Siemens-Nixdorf VP2600 series.
Type: Single-processor vectorprocessors.
Theoretical Peak performance: 5 Gflop/s
Reason for disappearance: replaced by the VPP300 series (see 3.4.6).

Machine: Stern Computing Systems SSP.
Type: Shared memory multi-processor, max. 6 processors.
Theoretical Peak performance: 2 Gflop/s
Reason for disappearance: Vendor terminated its business just before delivering first systems.

Machine: Thinking Machine Corporation CM-2(00).
Type: SIMD parallel machine with hypercube structure, max. 64K processors.
Theoretical Peak performance: 31 Gflop/s
Reason for disappearance: was replaced by the newer CM-5 (but see below).

Machine: Thinking Machine Corporation CM-5.
Type: Distributed memory RISC based system, max. 16K processors.
Theoretical Peak performance: 2 Tflop/s
Reason for disappearance: Thinking Machine Corporation has stopped manufacturing hardware and hopes to keep alive as a software vendor.

Machine: Transtech Paramid series.
Type: Distributed memory RISC based system, max. 64 processors.
Theoretical Peak performance: 6.4Gflop/s
Reason for disappearance: Transtech now mostly manufactures PC extension boards with IBM MC603 processors as performance boosters..

---

5 Systems under development

Although we wanted mainly to discuss real, marketable systems and no experimental, special purpose, or even speculative machines, we want to include a section on systems that are in a far stage of development and have a fair chance of reaching the market. For inclusion in section 3 we set the rule that the system described there should be on the market within a period of 6 months from announcement. The systems described in this section will in all probability appear within one year from the publication of this report.

However, there are vendors who do not want to disclose any specific data on their new machines until they are actually beginning to ship them (an example is the Convex C4). We recognise the wishes of such vendors (it is generally wise not to stretch the expectation of potential customers too long) and will not disclose such information.

Below we discuss systems that are expected to appear on the market between somewhat more than half a year to a year from now.

5.1 The Fujitsu VPP300 successor

No new name or details has been disclosed for this system (the new product announcement is scheduled for the 1st quarter of 1996). The following facts are known about the system: it will connect several of the 16-processor VPP300 as its basic units together. Very probably a second level will be added to the crossbar that now connects the processors in a VPP300 frame. The number of processors is also as yet unknown.

Acknowlegdements

It is not possible to thank all people that have been contributing to this overview. Many vendors and people interested in this project have been so kind to provide me with the vital information or to correct me when necessary. Therefore, I will have to thank them here collectively but not less heartily for their support.

References

References

[1] Amza C., A.L. Cox, S. Dwarkadas, P. Keleher, R. Rajamony H. Lu, W. Yu, and W.Zwaenepoel. Treadmarks: Shared memory computing on networks of workstations. *to appear in IEEE Computer, (draft copy:* `www.cs.rice.edu/~willy/TreadMarks/papers.html`.

[2] Jack Dongarra. Performance of various computers using standard linear equations software in a fortran environment. Technical Report CS–89–85, University of Tennessee, Computer Science Dept., February 1996. The report is available electronically, the url is `ftp://www.netlib.org/benchmark/performance.ps`.

[3] H. Kadota et al. Parallel computer adenart — its architecture and application. In *Proc. of the ACM International Conference on Supercomputing*, pages 1–8. ACM Press, June 1991.

[4] P. Flanders. Matrix multiplication on 'c' series daps. Technical Report AMT Document TR40, January 1991.

[5] M.J. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computing*, 21:948–960, 1972.

[6] High Performance Fortran Forum. High Performance Fortran language specification, version 1.0. Technical Report CRPC-TR92225, Center for Research on Parallel Computation, Rice University, Houston, Tex., 1993.

[7] R. Hockney and C. Jesshope. *Parallel Computers: Architecture, Programming and Algorithms*. Adam Hilger, Ltd., Bristol, United Kingdom, 1988.

[8] T. Horie, H. Ishihata, T. Shimizu, S. Kato, S. Inano, and M. Ikesaka. Ap1000 architecture and performance of LU decomposition. In *Proceedings of the 1991 International Conference on Parallel Processing*, volume I, Architecture, pages I–634–I–635, Boca Raton, FL, August 1991. CRC Press.

[9] H. Ishihata, T. Horie, T. Shimizu, S. Kato, S. Inano, and M. Ikesaka. Third generation message passing computer ap1000. In *International Symposium on Supercomputing*, pages 46–55, November 1991.

[10] David V. James, Anthony T. Laundrie, Stein Gjessing, and Gurindar S. Sohi. Scalable coherent interface. *IEEE Computer*, 23(6):74–77, June 1990. Scalable Coherent Interface,`http://sunrise.scu.edu/`.

[11] S. Kawabe. Private Communication.

[12] P.P.M. Rijk. The linear algebra codes using blas levels 2 and 3. In *Proceedings of the 2nd EuroBen Workshop*, pages 121–126, October 1991.

[13] S. Saini and D.H. Bailey. The nas parallel benchmark results 12-95. Technical Report Report NAS-95-021, NASA Ames Research Center, Moffett Field, CA 94035-1000, 1995. The report is available electronically, the url is `http://www.nas.nasa.gov/NAS/TechReports/NASreports/NAS-95-021/NAS-95-021.html`.

[14] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra. *MPI: The Complete Reference*. The MIT Press, 1996. The book is available electronically, the url is `http://www.netlib.org/utk/papers/mpi-book/mpi-book.html/`.

[15] A.J. van der Steen. The benchmark of the euroben group. *Parallel Computing*, 17:1211–1221, 1991.

[16] A.J. van der Steen. Benchmark results for the hitachi s3800. *Supercomputer*, 10(4/5):32–45, 1993.