Overview of recent supercomputers

by

Aad J. van der Steen
Academic Computing Centre Utrecht
PO Box 80.011, 3508 TA Utrecht
The Netherlands
Email: actstea@cc.ruu.nl

Publication of the NCF

Stichting Nationale Computer Faciliteiten
PO Box 93120
2509 AC 's-Gravenhage
The Netherlands

Fourth revised edition

September 1994

Contents

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.

1. Introduction and account

This is the fourth 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. For instance the information from `netlib` [4] compiled periodically by Dongarra and Duff is compact and fairly complete but many machines are listed which do not exist anymore or of which the information is still only partly valid. Although this gives an interesting view of the historical development in this field, we rather would like to concentrate on systems that are available now. The excellent book [24] was up-to-date at the beginning of 1991 but the continuous appearance and disappearance of systems will make it outdated when no update is issued shortly. The book is fairly comprehensive and therefore not complete (in this case the 400 page book would probably have been tripled in volume). 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 C200 series, the Cray Y-MP series, or the Thinking Machines CM-200 anymore although these systems are still used widely. Generally speaking, we include machines that are still marketed or will be marketed within 6 months.

Unlike [4] 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 Teradata 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 [6] 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 the Bull DPX 5000 series, the Control Data 4000 series, and 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 AMT DAP 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 3210, and the NEC SX-3/11.
- **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 two-processor Cray Y-MP C92 and a thousand processor Parsys SN1000 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 in some way 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 [12] 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.

Another trend that came up recently 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. Probably the most notable ones are Parasoft's Express (commercial) and PVM (standing for Parallel Virtual Machine, non-commercial) [2]. PVM has been adopted by Convex, and IBM for the transition stage between distributed computing and MPP on the clusters of their favorite processors. A proposal for a message passing standard, called MPI for Message Passing Iinterface, has recently been fixed and may help in making message passing programs portable both for distributed processing and DM-MIMD parallel processing [16].

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 been in existence. In the block diagram in Figure 1 we depict a generic model of a vector architecture.

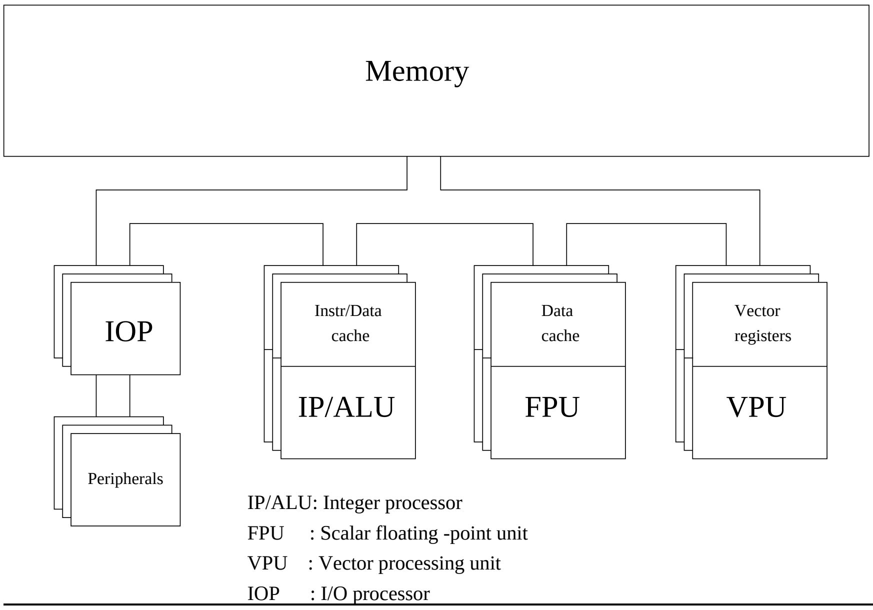

Figure 1. Block diagram of a vector processor.

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 3.3.4). 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 Fig. 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 Fig. 1. Yet, again there is a considerable diversity in the structure of VPUs. Every VPU is constituted 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 [10]. 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 slightly.

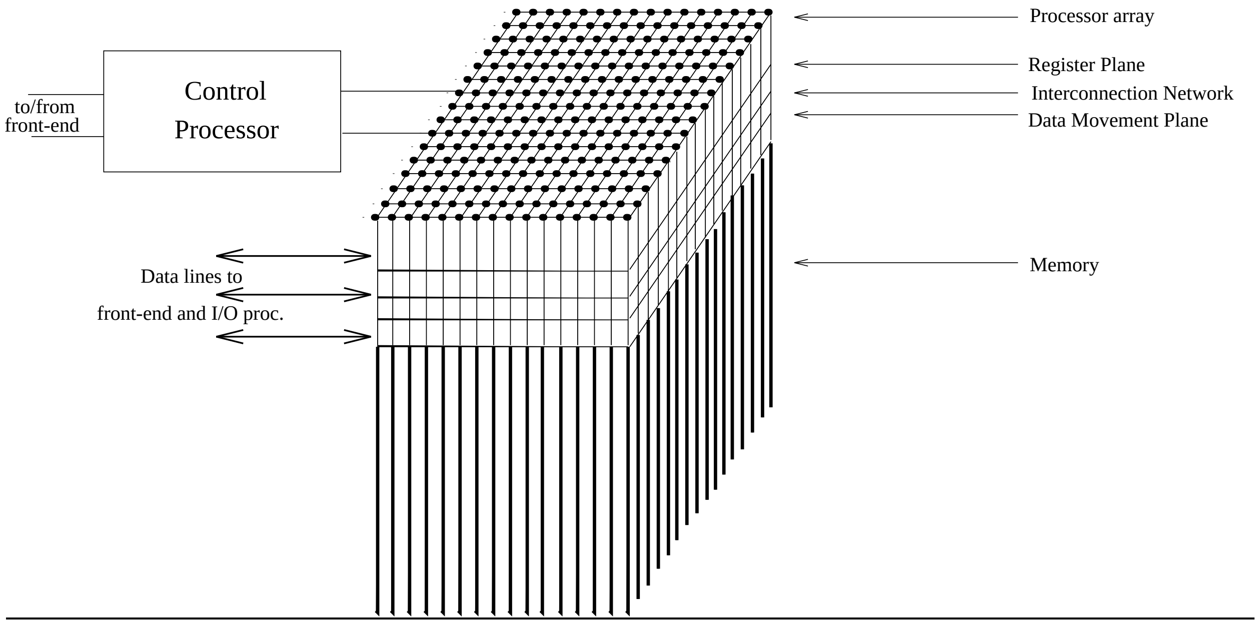

Figure 2. A generic block diagram of a distributed memory SIMD machine.

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 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 Fig. 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, significant speedups may result from this.

2.3 Shared-memory MIMD machines

In Fig. 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 the central bus 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.

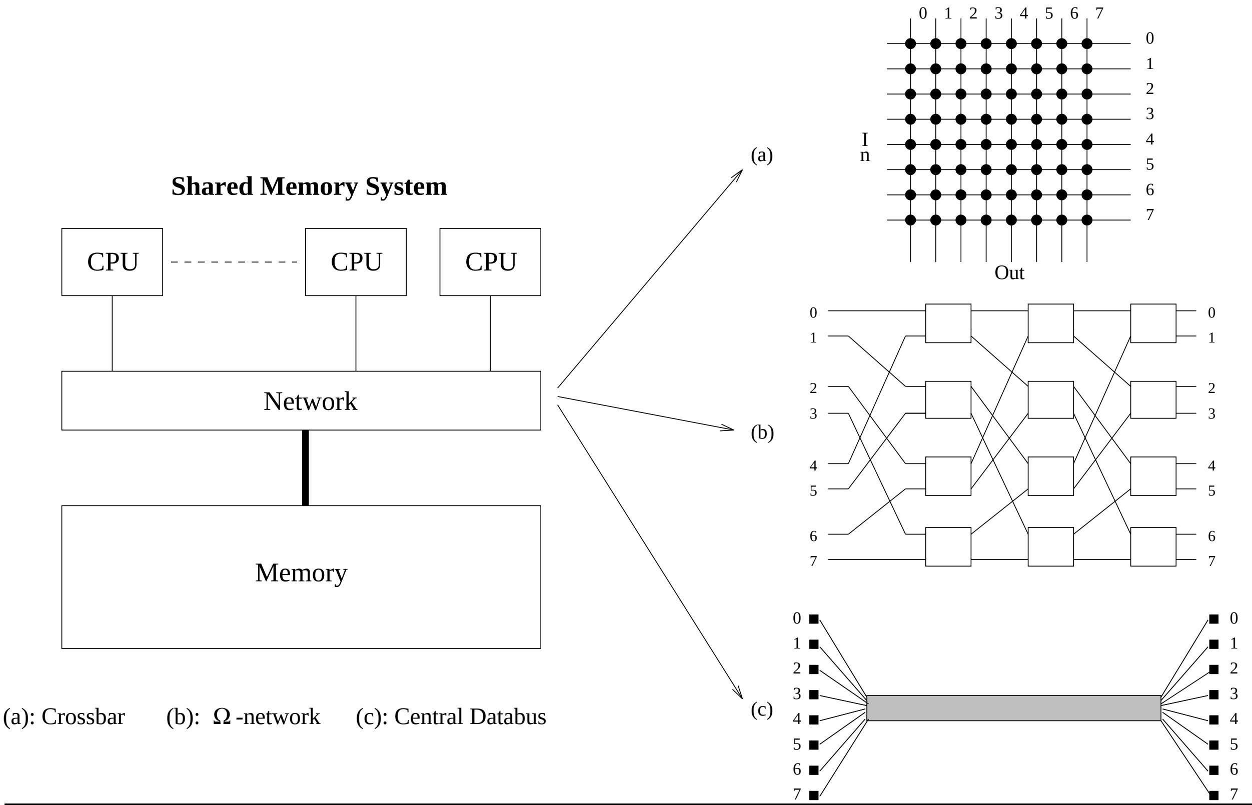

Figure 3. Some examples of interconnection structures used in shared-memory MIMD systems.

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 cannot exceed much a number of 10–20.

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 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 (16 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 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 former paragraph 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.

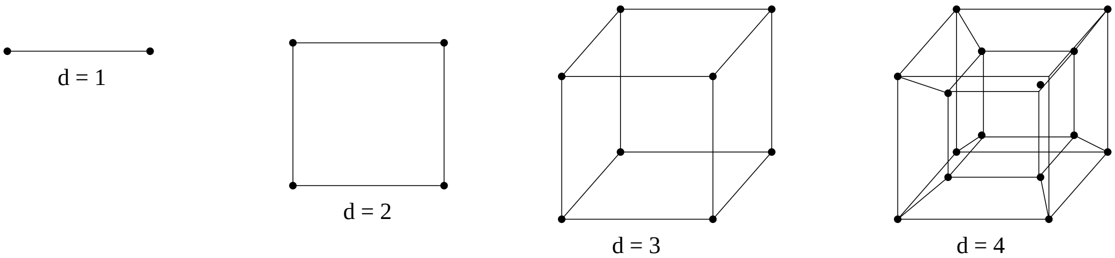

Figure 4. 1-, 2-, 3-, and 4-dimensional hypercube connections.

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.

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 of-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

3.1.1 The Fujitsu VPX200 series.

Machine type: Vectorprocessor.
Models: VPX210/10, VPX210/20, VPX220/10, VPX220/20, VPX220/40, VPX240/10, VPX240/20, VPX240/40, VPX260/10, VPX260/20.
Operating system: VSP/S (IBM MVS compatible) or UXP/M (a Unix variant acquired from Amdahl).
Compilers: Vectorising Fortran 77/VP both under VSP and UXP, C.

System parameters:

Model	VPX210/10 VPX210/20	VPX220/10 VPX220/20 VPX220/40	VPX240/10 VPX240/20 VPX240/40	VPX260/10 VPX260/20
Clock cycle VPU	3.2 ns	3.2 ns	3.2 ns	3.2 ns
Clock cycle scal. proc.	6.4 ns	6.4 ns	6.4 ns	6.4 ns
Theor. peak performance	0.625 Gflop/s	1.25 Gflop/s	2.5 Gflop/s	5 Gflop/s
Main memory	64–1024MB	128–1024MB	256–1024MB	512–2048MB
Extended memory	1–32GB	1–32GB	1–32GB	1–32GB

Remarks:

The speed differences between the different models stem from replication of the multi-functional multiply/add pipes in the models 210–260. The VPX210 has one pipe set, the VPX220 two, the VPX240 four, and the VPX260 eight. The VPX200 series replaces the VP2000 series which was virtually identical. However the models VP2100–2400 had a clock cycle of 4 ns for the vectorprocessor instead of 3.2 ns in the current VPX models. The VP2600 had already a clock cycle of 3.2 ns and is identical to the VPX260.

The multiply-add pipes as employed in the VPX200 series is of a type that is seen in an increasing number of vectorprocessors. It is capable of a combined multiply-add operation in which case 2 results per clock cycle are delivered. The same pipe also does single adds and multiplies. A separate pipe is used for the floating division operation. The clock speed of the scalar processors is two times lower than that of the VPU. With regard to Amdahl's Law which asserts that the slowest component of the system will be the decisive factor this design choice seems strange. For highly vectorisable programs the effect is small, however.

Just as in the former Fujitsu VP series (VPx00, VP2000) the vector registers are not fixed in length, but can be reconfigured according to the vector length encountered. This may avoid or minimise stripmining thus increasing the performance.

In the VPX200 systems each vectorprocessor can be accommodated with 2 or even 4 scalar processors (in which case the suffix /20 or /40 is used. The 1-scalar-processor models have /10 as a suffix). Multiple scalar processors should give a considerable better throughput when many users access the system because of less frequent context switching. A factor of 1.5 is quoted by Fujitsu for systems with two scalar processors.

Measured performances: In [3] results for the solution of a full linear system of order 1000 is given for the VP2600, a former model which is in fact equivalent to the VPX260. The speed turns out to be 4009 Mflop/s out of the theoretical 5 Gflop/s. For the VPX240 a speed of 2110 Mflop/s is reported from the same source, while the VPX220 attains 1048 Mflop/s .

3.1.2 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	≤6GB	≤16GB	≤16GB	≤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 again the clock cycle of the scalar processor is twice that of the VPU. The memory bandwidth is better than for the Fujitsu VP2000 series (1 load and 1 load/store pipe per arithmetic pipe set against 1 load/store pipe per arithmetic pipe set), but it is still not 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 rely at least on water cooling.

Unlike the S-820 series, the S3600 series will probably not only be marketed in Japan but worldwide. This should also be the case for the S3800 SM-MIMD machines (3.3.5).

Measured performances: In [3] 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.

3.1.3 The Siemens-Nixdorf Sx00 series.

These machines are identical to those listed under 3.1.1. The model names correspond to those of Fujitsu by changing VPX to S and by dropping the first figure in the model number and adding a zero at the back. So, for instance VPX220/20 translates to S200/20.

In the so-called ODIN program Siemens has cooperated with several academic customers to develop efficient numerical software for the S-series. This has resulted in excellent libraries for linear algebra and Fast Fourier Transforms [7].

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
No. of processors	8–128	128–2048
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

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 old 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 bandwidth for non-local communication turns out to be only four times smaller then local memory access (no local bandwidth figures are given, though).

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 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 Active Memory Technology Distributed Array Processor.

Machine type: Processor array.
Models: DAP510, DAP610, DAP510C, DAP610C
Front-end: Sun workstation or DEC VAX.
Operating system: Internal OS transparent to the user, SunOS or VMS 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.

System parameters:

Model	DAP510	DAP610	DAP510C	DAP610C
Clock cycle	100 ns	100 ns	100 ns	100 ns
No. of processors	1024	4096	1024	4096
Theor. peak performance				
Per Proc. (Mflop/s)	0.015	0.015	0.14	0.14
1-bit Mop/s	10240	40960	10240	40960
8-bit Mop/s	400	1600	5000	20000
Mflop/s (32-bit)total	15	60	140	560
Program memory	0.5–4MB	0.5–4MB	0.5–4MB	0.5–4MB
Data memory	4–32MB	16–128MB	4–32MB	16–128MB
Int. comm. speed				
Across row, column	40 MB/s	160 MB/s	40 MB/s	160 MB/s
Memory to PE	1.28 GB/s	5.12 GB/s	1.28 GB/s	5.12 GB/s

Remarks:

Apart from the name of the vendor, very little has changed with AMT and its products. AMT has been taken over by the USA-based Cambridge Technology Group. This has no influence on the product line or policy. Just the name has been changed to "Cambridge Active Memory Technology".

In the DAP510 models the 1024 processors are ordered in a 32×32 array, the '5' in 510 indicating the exponent of 2 of the edge size, while the '10' stands for the processor speed in MHz. Likewise has the DAP610 4096 processors arranged in a 64×64 square.

The models 510C and 610C are able to operate byte parallel on appropriate operands. Operations done in this fashion speed up by a factor of nearly 10 over the otherwise equivalent 510 and 610 models. Bit-serial speed is not affected.

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 DAP510 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 DAP directly thus not burdening the front-end machine (and the connection between front-end and DAP) with I/O operations and unnecessary data transport. 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.

Measured Performances: In [5] the speed of matrix multiplication on various DAP models is analyzed. For matrices of order 512 speeds of 94 and 340 Mflop/s (32-bit precision) were measured on the DAP510C and DAP610C, respectively.

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 AMT DAP (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 AMT DAP, 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 AMT DAP, 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 [3] 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 [3] 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 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.

3.3.1 The 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.

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:

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-1 (see 3.4.3) 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 [19]. 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.

3.3.2 The Cray Computer Corporation Cray-3

Machine type: Shared-memory multi-vectorprocessor.
Models: Cray-3
Operating system: UNICOS (CCC's Unix variant).
Compilers: Fortran,C.

System parameters:

Model	Cray-3
Clock cycle	2 ns
Theor. peak performance	
Per processor	1 Gflop/s
Maximal	16 Gflop/s
Main memory	$\geq$ 4GB
Memory bandwidth	
Single proc. bandwidth	8 GB/s
No. of processors	1–16

Remarks:

Cray Computer Corporation (CCC) has been established by Seymour Cray to build the Cray-3, of which the first stages were done under the wings of the mother company, Cray Research Inc. (see 3.3.3, 3.3.4, and 3.4.4). It was decided that it was too much of a burden to develop two independent high-end product lines, the Cray Y-MP (see 3.3.4) and the Cray-3. So, CCC split off and took then available Cray-2 with it as a marketable product. As might be expected the Cray-3 is reminiscent to the Cray-2. There are notable differences however: As in the Cray-2 there is a foreground processor which performs system tasks, I/O handling, and synchronisation of the background processors, the vectorprocessors proper. The maximum number of processors has grown from 4 to 16 in the Cray-3 and clock cycle is halved from 4 to 2 ns. The Cray-3 relies heavily on the use of GaAs components (CCC is with Convex the only company who actually uses these) to be able to condense the components to such an extent that the required clock cycle speed can be sustained.

There are additional improvements in the Cray-3 design as compared to the Cray-2. A main bottleneck in the Cray-2 was that there only was one port to main memory per processor where the memory itself was relatively slow. This required at least a high data re-use to minimise the memory traffic but this was not helped at all by the inability of the CPUs to chain operations. All of these weak points have been mended in the Cray-3 in some way: each processor has now two memory ports, the memory has been speeded up over a factor of two with respect to the fastest SRAM model of the Cray-2 and, although there is still no chaining of operations in the Cray-3, there is a technique call "tailgating",i.e., shifting in of elements of a new vector in a register at the places of elements that already have been used in an operation instead of waiting for the completion of an operation on a complete vector register. These improvements should make the machine much more usable than the Cray-2 which often was bought mainly for the (then) large memory of 512 MB. We could not get information on the existence of local memories in the background processors as were available in the Cray-2. In these machines the local memories, more or less used as a cache memory, were not accessible from a high level language. Library routines, using the local memory could greatly improve the performance in many cases. Another point that has improved in the Cray-3 is the the instruction rate has doubled to one instruction per clock cycle. Apart from the decreased clock cycle this speeds the scalar performance up by another factor of two.

Presently, a Cray-3 system is on trial at NCAR in Boulder, Colorado.

Measured Performances: In [3] a speed for the solution of a dense linear system of order $N = 100$ is given of 391 Mflop/s.

3.3.3 The Cray Research Inc. Attached Parallel Processor.

Machine type: RISC-based multiple shared-memory processor
Models: CRI APP

Operating system: SunOS (Sun variant of Unix), Cray APP micro kernel transparent to the user.
Compilers: Fortran, C.

System parameters:

Model	APP
Clock cycle	25ns
Theor. peak performance	
Per processor (64-bit)	40 Mflop/s
32-bit precision	6.7 Gflop/s
64-bit precision	3.3 Gflop/s
Main memory	32MB–1GB
Memory bandwidth	160 MB/s/path (see remarks)
Single proc. bandwidth	640 MB/s
No. of processors	7–84

Remarks:

One of the assets in taking over Floating Point Systems late 1991 was the FPS 500 system. This was a system that could harbour two scalar processors, two vector processors, and the so-called Matrix Co-Processor, an accelerator box with up to 84 Intel i860 processors. This last element has now been made by Cray into a separate product, the Cray APP. The basic architecture has not been changed: an 8-way crossbar connects 7 buses or paths each of which can be accommodated with 12 i860 processors. The bandwidth for a processor bus is 160 MB/s. The eighth connection on the crossbar is used for I/O channels like a VME bus and HIPPI channels. The front-end to the APP is a Sun Sparc workstation. To make the APP independent, its was fitted with its own memory of up to 1 GB.

The APP uses auto-parallelising compilers and Cray-style microtasking. In addition, up to twelve of these systems can be clustered via a global memory of up to 8 GB (not unlike the IBM System Four, see 3.3.6). With 1008 processors such a full system should theoretically deliver 40 Gflop/s at 64-bit precision. Together with the local memories of the individual APPs the aggregate memory would amount to 20 GB.

Measured Performances: Almost no independent results are reported yet. Cray quotes several applications in image-, sonar-, and radar processing with speeds of 1–2.3 Gflop/s for one APP. In [3] a speed of 548 Mflop/s is reported for the solution of an order 1000 full linear system.

3.3.4 The Cray Research Inc. Cray J-series, Y-MP C90, Y-MP M90.

Machine type: Shared-memory multi-vectorprocessor.
Models: Cray J, Y-MP C90, Y-MP M90.
Operating system: UNICOS (Cray Unix variant).
Compilers: Fortran, C, C++, Pascal, ADA.

System parameters:

Model	Cray J9	Cray Y-MP C90	Cray Y-MP M90
Clock cycle	10 ns	4 ns	6 ns
Theor. peak performance			
Per processor	200 Mflop/s	1 Gflop/s	333 Mflop/s
Maximal	3.2 Gflop/s	16Gflop/s	2.67 Gflop/s
Main memory	$\leq$4 GB	$\leq$16 GB	$\leq$32 GB
Memory bandwidth			
Single proc. bandwidth	1.6 GB/s	12 GB/s	4 GB/s
No. of processors	4–16	2–16	2–8

Remarks:

Cray Research Inc. (CRI) supports at this moment 3 product lines (apart the SuperSparc-based CS6400 which is targeted to the commercial market and is not discussed in this report). All models are binary compatible and run exactly the same software. The Cray Y-MP/16 M90 is the latest of these products (announced late 1991). It is a variation of the former Y-MP series with a much larger memory: 1–32 GB.

In terms of speed the M90 is not up to the fastest that can be produced by Cray. The C90 has a lower clock cycle and a faster memory. The M90 series is meant for applications that are more restricted by lack of memory than by the lack of speed. Very often memory and speed are interchangeable in this respect. Cray expects to have a market for these systems besides the fast C90 series.

In the C90 the number of vector pipe sets is doubled with respect to the M90 models. Therefore, the performance of a C90 CPU is trice that of a Y-MP CPU, a factor 1.5 because of the lower clock cycle and an additional factor of 2 because of the doubling of the functional units. The doubling of the number of CPUs makes the maximal C90 6 times faster than the fully configured Y-MP M90.

The Cray J9 series is the entry level model marketed by CRI announced in September 1994. 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 3.3.1). 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 C90 and M90 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 C90 this meant that the relative bandwidth to the CPUs had to be doubled from 24 to 48 bytes/cycle. This has indeed been accomplished and observed results indicate that for the C90 the performance scales up with the clock cycle and the number of functional units (see measured performances below). For the J9 series the bandwidth is lower: 16 bytes/cycle. This is regrettably less than was available in the Y-MP EL machines and it might adversely affect the efficiency.

Another property that is unique for the Cray C90 and M90 systems is that they do not have separate scalar processors but that scalar- and vector code have to share the same functional units. The Cray J9 series does have separate scalar processors. Theoretically, the absence of separate scalar processors might impair the throughput speed (Fujitsu and Hitachi (3.1.1, 3.3.5) even add an extra scalar processor in the VPX200 and S-3800 series to combat excessive context switching). However, in practice the drawbacks seem rather limited.

Measured Performances: Measurements with the EuroBen Benchmark [22] have been done at CRI on a test machine of the Y-MP/16 C90 series and in [3] a speed of 10.7 Gflop/s was found for solving an order 10,000 dense linear system. The J9 series machine is still quite new and no results are available yet.

3.3.5 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 VP2000 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^{th} degree polynomial on a single processor. In matrix-vector multiplication, speeds of 6.5 Gflop/s on one processor were measured (see [23]).

3.3.6 The IBM Power/4.

Machine type: RISC-based multiple shared-memory processor
Models: Power/4
Operating system: AIX/6000 (IBM variant of Unix).
Compilers:Fortran, C. **System parameters**:

Model	Power/4
Clock cycle	23.8 ns
Theor. peak performance	
Per proc. (64-bit)	84 Mflop/s
Maximal (64-bit)	336 Mflop/s
Main memory	≤1.4GB (see remarks)
Memory bandwidth	
Single proc. bandwidth	336 MB/s
No. of processors	1–4

Remarks:

This system has been announced at Supercomputing '92 and apart from private talks with IBM representatives and the official press release from IBM, very little information is to be had. This even extends to the name of the system proper: while the machine on display at Supercomputing '92 was presented as "Power/4", the press release consistently spoke of the "Shared Memory POWERparallel system" and it was uncertain which name was correct.

The new feature of the system is its shared memory: 7 card slots provide space for 7×128 MB of shared memory, while each of the processors also may contain 16–128 MB of local memory. So, the aggregate memory can be up to 1408 MB. In many respects the machine seems similar to the Cray APP (3.3.3), but a distinction is the interconnection which is a direct 4×4 crossbar here, while in the APP up to 12 processor have to shared a bus that is connected to a crossbar.

Although the system can use PVM for intercommunication (for those that already used an RS/6000 cluster), this is not necessary. The Fortran compiler is able to parallelise programs over the combined processors in which case synchronisation is done via the shared memory. Parallel tasks that have an irregular nature will again have to fall back to PVM, but as communication is now through memory rather than Ethernet, the communication speed is very much higher. I/O may be distributed over the processors and has an aggregate bandwidth of up to 320 MB/s.

The system is not promoted heavily: one year after its announcement at Supercomputing '93 the only information on it was that it still existed and was marketed. No user experiences were known.

Measured Performances: No performance figures are available for this system.

3.3.7 The NEC SX-3R series.

Machine type: Shared-memory multi-vectorprocessor.
Models: SX-3/11R, SX-3/12R, SX-3/14R, SX-3/22R, SX-3/24R, SX-3/42R, SX-3/44R.
Operating system: SXOS, SUPER-UX (NECs Unix variant).

Compilers: Fortran, C. **System parameters**:

Model $x = 1, 2, 4$	SX-3/x1R	SX-3/x2R	SX-3/x4R
Clock cycle	2.5 ns	2.5 ns	2.5 ns
Theor. peak performance			
64-bit precision	1.6–6.4 Gflop/s	3.2–12.8 Gflop/s	6.4–25.6 Gflop/s
No. of processors	1–4	1–4	1–4
Main memory	≤4GB	≤4GB	≤8GB
Extended memory	≤16GB	≤16GB	≤16GB

Remarks:

The SX-3R series is the second generation of NECs vector supercomputers, the first being the SX-2. The many model numbers deserve some explanation: the suffix $/xy$ stands for the number of processors x and the number of vector pipe sets y, respectively. In this terminology the largest SX-2 would have been a SX-2/14R because it has four pipe sets. With the new generation of SX-3s multiple processors were introduced. Because a pipe set under the right circumstances can produce four results per clock cycle at a cycle time of 2.5 ns, the theoretical peak speed is 1.6 Gflop/s per pipe set which leads to the quoted performances in the system parameters list. Although the clock cycle for the scalar processor is identical to that of the VPUs, an instruction only can be issued every two cycles, so effectively the situation is not very different from that of the other Japanese vectorprocessors where the scalar unit has a clock rate of twice that of the VPUs.

Apart from the arithmetic processors (APs) which contain the VPUs and a scalar processor, each system can be configured with 1 or 2 Control Processors (CPs). The CPs are dedicated to systems tasks, interactive processing, and controlling I/O requests (which are handled by separate I/O processors). The APs are thus freed from any menial tasks and can concentrate on computational work.

As in the Cray and Convex machines (3.3.1–4) synchronisation of parallel processes is performed via special communication registers in order to minimise synchronisation overhead.

It is not easy to derive the CPU-to-memory bandwidth in the SX-3 from the NEC documentation. One is left with the impression that up to the SX-3/24R the bandwidth is enough to sustain a stream of 2 operands/pipe-set/cycle and shipping one result/pipe-set/cycle back to memory.

The parallelisation model as used on the SX-3 is conceptually not very different from that of other vendors in the field, although NEC has its own terminology (like all others).

Measured Performances: Results from the SX-3R series are given in [3]: On an SX-3/44R a speed of 15.1 Gflop/s was obtained on an $N = 1000$ order dense linear system. The SX-3 needs rather long vectors to get to high fractions of its peak speed but when in this situation it is quite fast.

3.3.8 The Silicon Graphics (Power) Challenge

Machine type: Shared-memory multi-processor.
Models: (Power)Challenge M, L, XL.
Operating system: IRIX (SGI's Unix variant).
Compilers: Fortran 77, C, C++ , Pascal.

System parameters:

Model	Model M	Model L	Model XL
Challenge			
Clock cycle	10 ns	6.7 ns	6.7 ns
Theor. peak performance			
Per proc. (64-bit)	50 Mflop/s	75 Mflop/s	75 Mflop/s
Maximal (64-bit)	50 Mflop/s	900 Mflop/s	2.7 Gflop/s
Main memory	≤384 MB	≤6 GB	≤16 GB
Memory bandwidth			
Proc. to cache/proc.	0.8 GB/s	1.2 GB/s	1.2 GB/s
Main mem. to cache	1.2 GB/s	1.2 GB/s	1.2 GB/s
No. of processors	1	2–12	2–36
Power Challenge			

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 mem. to cache	—	1.2 GB/s	1.2 GB/s
No. of processors	—	2–6	2–18

Remarks:

The 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 Challenge 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 Challenge series uses MIPS R4400 RISC processors with a nominal peak speed of 75 Mflop/s (except in the M model which has one 50 Mflop/s processor). In the Power Challenge the R8000 processor (formerly called the TFP processor standing for True Floating Point) will be used (available from half of 1994). Although the clock rate of this processor is two times lower than that of the R4400, the performance should be 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.

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.

Measured Performances: At this moment Power Challenge systems are just becoming available to customers. Experiments with the EuroBen Benchmark on a Challenge L system indicated that on a single processor 55–65 Mflop/s is possible for linear algebra operations. The overhead on 4 processors proved to be large when using automatic parallelisation but may improve significantly with hand optimisation. Early results as provided by Silicon Graphics are: 101 Mflop/s for the solution of an $N = 100$ dense linear system on one processor. For $N = 1000$ a speed of 260 Mflop/s is given and on 16 processors a speed of 3.6 Gflop/s was reported for a system of order 14000.

3.3.9 The Stern Computing Systems SSP

Machine type: Shared-memory multi-processor.
Models: SSP/Enterprise, SSP/Application.
Operating system: OSF/1.
Compilers: Fortran 77, C.

System parameters:

Model	Application	Enterprise
Clock cycle	5 ns	6 ns
Theor. peak performance		
Per proc. (64-bit)	200 Mflop/s	266Mflop/s
Maximal (64-bit)	1.2 Gflop/s	2 Gflop/s
Main memory	≤14 GB	≤8 GB
No. of processors	1–6	1–6

Remarks:

The SSP/Application and the SSP/Enterprise (SSP standing for Shared Sustained Performance) have different structures that are assumed to reflect the different needs of customers. The SSP/Application is a bus-based machine in which a 128 bits wide 600 MB/s bus connects up to six processors to 7 memory modules that each can have a size of up to 2Gbyte. The basic processor of the SSP/Application is a DEC Alpha processor with a peak speed of 200 Mflop/s , 8KB primary data- and instruction caches and a 4 MB secondary cache. These caches may be sufficient to ensure that on most applications the bus speed will not be a bottleneck for the performance.

The SSP/Enterprise has a different structure in many respects. Firstly, the processors ($\leq$ 6) are interconnected to 8 memory modules by a crossbar with an aggregate bandwidth of 30 GB/s instead of by a bus. Secondly, the processors are a proprietary development of Stern. They have some interesting features that are aimed at sustainment of the data flow to the various units in the processor. Each processor contains units, called the Control Unit (CU), the Address Unit (AU), and the Data Unit (DU). The latter in turn contains 2 floating-point and integer pipelines, and four divide/square root units. At every cycle the DU (ideally) accepts two operands and produces one or two results per cycle. The Control Unit generates instruction fetch blocks that are dispatched to the Address Unit and the Data Unit. The AU generates memory load and store addresses which are matched with data to loaded and stored to/from the DU by means of queues: Load and Store Address Queues and Load and Store Data Queues. These queues allow for a large amount of latency hiding and therefore a largely uninterrupted flow of operands to the Data Unit. Effectively, the generation of instructions, addresses, and the resulting operations are decoupled to a large extent. Operation of the CU is sustained by a data cache.

The SSP/Enterprise might be thought of as a systems that is specifically targeted at continuous high-speed operation, while the SSP/Application is geared at running large standard applications (which are rather well available for DEC Alpha processors, also the large memory is useful in that respect).

Measured Performances: Because the systems are still very new performance figures are very scarce. For the SSP/Application Stern reports a speed of 43 Mflop/s for the solution of a dense $N = 100$ linear system and of 751 Mflop/s for a system of order $N = 1000$. For the SSP/Enterprise no figures are available yet.

3.3.10 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++.

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 marketed from the beginning of 1995, 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. Another reason is the host of transputer-based products that has been developed in the last few years. Although the majority of these systems are very interesting, it is impossible to keep track of them all and the same is true for Intel i860-based machines that have been developed from them. We had therefore no other reasonable choice than to select a few representatives from these sub-classes and discuss them. This in no way implies that the systems not mentioned ere 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, do 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	≤1,280MB
Memory/node	≤20MB
Memory bandwidth	5GB/s
Communication bandwidth	≤10MB/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 Convex Meta series.

Machine type: RISC-based distributed-memory multi-processor.
Models: Meta series.
Operating system: ConvexOS (Convex Unix variant) on front-end, OSF/1 on cluster nodes.
Connection structure: Ring.
Compilers: Fortran, C.

System parameters:

Model	Meta series
Clock cycle	16 ns
Theor. peak performance	
Per proc. (64-bit)	67 Mflop/s
Maximal (64-bit)	—
Main memory	—
Memory/node	$\leq$128 MB
Communication bandwidth	1.25 MB/s
No. of processors	4– …

Remarks:

The Meta series can be seen as Convex's first step to its MPP system, although Convex thinks that there may be a place for the Meta series in its own right. As yet, the Meta series is very similar to a workstation cluster with some packaging and software support, and a front-end machine from Convex. As nodes in the cluster, processors from Hewlett-Packard's PA-RISC line have been chosen. Presently, the 16 ns cycle version is used but within a few month the 10 ns version will replace these. In the same time frame the Ethernet-based inter-node communication will be replaced by FDDI with an estimated communication speed of at least 4 MB/s. Another new feature that will be introduced within half a year is the VIOP-2 (for Virtual I/O Processor). The VIOP-2 should act as a kind of shared memory between the cluster nodes. In addition, it is connected to the C-series front-end. Both maximum size and communication speed of the VIOP-2 are not yet specified, but will presumably boost the inter-node communication significantly. In this respect, the machine is akin to the IBM System Four machine (see 3.3.6) in which also the shared memory can be used to accelerate programs that employ PVM as a message passing medium.

As mentioned, a Convex of the C series (C3200, C3400, C3800) is required as a front-end. Up to 8 cluster nodes can be housed in a cabinet and, at least conceptually, the cabinets can be extended indefinitely. For this reason no peak performance was quoted in the system parameters section.

Programming the cluster is done by using PVM. It is possible for the C-series processor(s) to participate as heterogeneous nodes in the the PVM cluster. This could be advantageous when jobs contain well-vectorisable tasks to assigned to the C-series vector processors while scalar processing in the job is left to the HP processors. Convex will actively support PVM (which is why it is marketed as ConvexPVM) at a charge. Furthermore, ConvexNQS+, an enhanced NQS version, enables batch job submission on the heterogeneous system and the mathematical library ConvexMLIB should ensure efficient computation of basic algorithms like linear algebra operations and FFTs on the Meta series.

Measured Performances: The system is quite new and therefore no performance numbers are known at this stage. Convex mentioned an efficiency for matrix multiplication routine `DGMMEV` of 7.8 on 8 processors, although the (very relevant) size of the problem was not given.

3.4.3 The Convex Exemplar SPP-1.

Machine type: RISC-based distributed-memory multi-processor.
Models: SPP-1.
Operating system: SPP-UX, based on OSF/1 AD microkernel.
Connection structure: Ring.
Compilers: Fortran, C.

System parameters:

Model	SPP-1
Clock cycle	10 ns
Theor. peak performance	
Per proc. (64-bit)	200 Mflop/s
Maximal (64-bit)	25.6 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-1 can be seen as a logical next step of integration after the Meta Series (see 3.4.2). Where the latter is a cluster of workstation that still uses external interconnection channels like Ethernet or FDDI, the SPP-1 has its communication structure integrated. As in the Meta Series the CPUs used are HP's PA-RISC processors. Up to 8 of those 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 2GB 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-1 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 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 111, 231, 398, and 650 Mflop/s for 1, 2, 4, and 8 processors (within one hypernode), respectively.

3.4.4 The Cray Research Inc. T3D.

Machine type: RISC-based distributed-memory multi-processor.
Models: T3D.
Operating system: UNICOS MAX (micro-kernel Unix).
Connection structure: 3-D Torus.
Compilers: CFT77_M (Fortran 77 with extensions), C.

System parameters:

Model	T3D
Clock cycle	6.6 ns
Theor. peak performance	
Per proc. (64-bit)	150 Mflop/s
Maximal (64-bit)	307 Gflop/s
Main memory	≤128GB
Memory/node	64MB
Communication bandwidth	300MB/s
No. of processors	32–2048

Remarks:

The T3D is the first generation of DM-MIMD systems from CRI. The name T3D refers to its connection structure: a 3-D torus. The system needs one of Cray's larger vectorprocessors as a front-end. A Cray Y-MP/E or Y-MP C90 or M90 will do, but not a Y-MP EL because it lacks the Cray proprietary low-speed (6 MB/s) and high-speed (200 MB/s) channels which are required for coupling with the back-end system.

The computational tasks are executed by DEC alpha MC21064 RISC processors. 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 two processing elements (PEs) which in turn contain a CPU, memory, and a communication engine that takes care of communication between PEs. The bandwidth between nodes is quite high: 300 MB/s. Furthermore, care has been taken to hide the latency to a large extent. There is a prefetch queue for asynchronous fetches of small messages and a remote store that allows direct writing in memory addresses of non-local PEs with very low latency. Another device that can be used for general data re-distribution is the Block Transfer Engine. It should make rearrangement of data, for instance in case of matrix transposition, a relatively painless operation. The T3D has hardware support for fast synchronisation. E.g., barrier synchronisation takes only one cycle per check.

Presently the I/O is handled by the front-end processor. Cray intends to make the second generation stand-alone by adding I/O-nodes to the T3D itself.

The T3D 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: On a 4096×4096 matrix multiplication a speed of about 8 Gflop/s was found for 128 PEs. Furthermore, favourable performances for the NAS parallel benchmarks have been reported by Cray, suggesting that a 128 PE system was fastest on all NAS codes as compared to TMC, Intel Paragon, and KSR-1 systems with the same amount of nodes. These figures still have to be confirmed independently.

3.4.5 The Digital Equipment Corp. Alpha AXP Farms.

Machine type: RISC-based distributed-memory multi-processor.
Models: AXP Farm 1000, 2000, and 3000 (see remarks).
Operating system: OSF/1.
Connection structure: Crossbar (see remarks).
Compilers: Fortran 77 , HPF, C.

System parameters:

Model	1000	2000	3000
Clock cycle	5.7 ns	5.7 ns	5.0 ns
Theor. peak performance			
Per proc. (64-bit)	150 Mflop/s	175 Mflop/s	200 Mflop/s
Maximal (64-bit)	600 Gflop/s	700 Gflop/s	800 Gflop/s
Main memory	≤1 GB	≤2 GB	≤4 GB
Memory/node	≤256 MB	≤512 MB	≤1 GB
Communication bandwidth			
Agregate bandwidth	450 MB/s	450 MB/s	450 MB/s
No. of processors	4	4	4

Remarks:

Like Convex,(see 3.4.2) has reacted on the current trend of distributed computing by marketing a cluster of DEC Alpha workstations in one rack. With this rack comes, if desired, a fast non-blocking crossbar switch based on FDDI, called the GIGAswitch. Because of the speed of the switch (450 MB/s aggregate bandwidth) which is greater than those of many internal switches of integrated parallel systems, the distinction between an Alpha Farm and many parallel systems becomes rather theoretical.

The different models derive from the type of machines employed in the cluster: The model 1000 uses the DEC 3000/300 processor, the model 2000 uses the 3000/500 processor, and the model 3000 uses the 3000/800 processor, the 500 processor being somewhat faster than the 300 processor and the 800 processor in turn faster than the 500.

Of course, of the most usual software available for distributed computing is also available on the Alpha Farms: PVM, Express, and Linda are supported. Futhermore High Performance Fortran is supported as well as software that provide load levelling capabilities: the Load Sharing Facility (a Digital product), NQS, DQS, and DNQS.

Measured Performances: As yet, only single processor results are known for the nodes constituting the various Farm models: for the solution of an order $N = 1000$ dense linear system speeds of 80.6, 130.1, and 148.3 Mflop/s were found for the 300, 500, and 800 processors respectively.

3.4.6 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 [13].

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 [11] 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 [3] a speed of 2.3 Gflop/s on a dense system of order 25,600 on 512 cells.

3.4.7 The Fujitsu VPP500 series.

Machine type: Distributed-memory vector multi-processor.
Models: VPP500.
Operating system: UXP/VPP (a V5.4 based variant of Unix).
Connection structure: Multi-stage crossbar.
Compilers: Fortran 77+ (Fortran 77 with data decomposition extensions), C.

System parameters:

Model	VPP500
Clock cycle	10 ns
Theor. peak performance	
Per proc. (64-bit)	1.6 Gflop/s
Maximal (64-bit)	355 Gflop/s
Main memory	≤56.8GB
Memory/node	≤256MB
Communication bandwidth	400MB/s
No. of processors	7–222

Remarks:

The VPP-500 is front-ended by a machine of the VP2000 series. The system itself can be used as a batch processor from the front-end. Each node, called a Processing Element (PE) in the system is a powerful (1.6 Gflop/s peak speed) vector processor in its own right. The vector processor is complemented by a RISC scalar processor with a peak speed of 200 Mflop/s. The scalar instruction format is 64 bits wide and may cause the execution of three operations in parallel. Each PE has a memory of 128–256 MB while a PE communicates with its fellow PEs at a point-to-point speed of 400 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.

Because the network is a multistage crossbar, the complexity only grows logarithmically. The network has some intelligence built in in the form of a Synchronisation Register (SR). 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.

Communication to the outside world is realised via the SSU of the front-end VP2000 or VPX200 system. Also all I/O is handled by the front-end. On the VPP side traffic is controlled by one or two Control Processors (CPs). Other tasks of the CPs is to keep track of the availability and state of the PEs and to allocate work on them. The allocation is dynamic and three access modes of the PEs are possible: "simplex", meaning that on request of N processors, N PEs are exclusively granted as soon as they are available with one process per processor. The "exclusive" mode again claims N PEs but in this case 1 or 2 processes may run on each PE in the complex. In "shared" mode many processes may share the requested PEs but at least $\frac{1}{2}N$ are guaranteed during processing.

The Fortran compiler that comes with the VPP500 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.

A first trial system will be installed by the end of 1992 at the National Aerospace Laboratory in Tokyo. Other systems may be installed in the course of 1993.

Measured Performances: An impression of the performance of the VPP500 can be obtained from results measured at the Numerical Wind Tunnel (NWT) at NAL, Tokyo, Japan. This machine is an experimental system closely resembling the VPP500. With 140 processors a speed of 116 Gflop/s was observed for a 3-D CFD application [14]. Furthermore a speed of 124.5 Gflop/s for the solution of a large dense linear system was reported in [3], again on 140 processors.

3.4.8 The Hitachi SR2001 series.

Machine type: RISC-based distributed memory multi-processor.
Models: SR2001.
Operating system: HI-UX/MPP (Micro kernel Mach 3.0).
Connection structure: Hyper crossbar.
Compilers: Fortran 77, Fortran 90, HPF, C, C++.

System parameters:

Model	SR2001
Clock cycle	11 ns
Theor. peak performance	
Per proc. (64-bit)	180 Mflop/s
Maximal (64-bit)	23 Gflop/s
Main memory	≤33.8 GB
Memory/node	≤256 MB
Communication bandwidth	100 MB/s
No. of processors	8–128

Remarks:

The SR2001 is the first generation of distributed memory parallel systems of Hitachi. The basic node processor is an Hitachi implementation of the PA-RISC architecture of HP running at a clock cycle of 11 ns. The connection structure is a hyper (3-D) crossbar which connects all nodes directly at high speed (100 MB/s point-to-point). The system is very new: presently a prototype is running at the Computing Centre of the University of Tokyo for evaluation. First results show that the single node performance is up to the specifications.

The following software products will be supported in addition to those already mentioned above: PVM, 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: As yet, no quotable performance figures are available for the SR2001.

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++.

System parameters:

Model	9076 SP2
Clock cycle	15 ns

Theor. peak performance	
Per Proc. (64-bit)	266 Mflop/s
Maximal (64-bit)	8 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. 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 2GB. 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.

Measured Performances: No performance results for the SP2 are available yet, however, in [3] a single node performance of 236 Mflop/s is mentioned for the basic processor in solving a dense linear system of order $N = 1000$.

3.4.10 The Intel Paragon XP.

Machine type: RISC-based distributed-memory multi-processor.
Models: Paragon XP/S, XP/E
Operating system: OSF/1.
Connection structure: 2-D mesh (torus).
Compilers: Fortran 77, ADA.

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.

Measured Performances: As on many systems a results are available for the solution of a large dense linear system. In [3] a speed of 72.9 Gflop/s is reported for a system of size 55,000 on a 1872-node XP/S. For the XP/E no results are available yet.

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 [15] 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.

3.4.12 The Meiko i860 Computing Surface.

Machine type: RISC-based distributed-memory multi-processor.
Models: Infinity Series Computing Surface.
Operating system: Internal OS transparent to the user, SunOS (Suns Unix variant) on the front-end system.
Connection structure: Multistage crossbar.
Compilers: Extended Fortran 77, ANSI C.

System parameters:

Model	i860 Computing Surface
Clock cycle	25 ns
Theor. peak performance	
Per proc. (64 bits)	60, 4 Mflop/s
Maximal (64 bits)	—
Main memory	—
Memory/node	32, 128MB
Communication bandwidth	6 MB/s
No. of processors	2, 1– …… ∞

Remarks:

The i860 Computing Surface is a heterogeneous processor machine. The present model uses i860-, SPARC-, and Inmos T800- processors in one system. Other processors could in principle be added. Although the T800s can be used for computation, they usually will be used as communication engines between processor boards. The Computing Surface system needs at least one SPARC processor which acts as a host. Furthermore, boards can be put in system cabinets (either for 10 or for 40 boards) that each contain 2 i860 processors, two T800s (called Network Interface Processors in this context), 64 KB of on-board cache and up to 32 MB of memory per board. The Computing Surface Network has another 2 MB of buffer memory on board. An alternative board contains a SPARC processor, 4 Network Interface Processors and up to 128 MB of memory (the second entries in the the system parameter list above shows the speed as well as the memory size for the different boards). There is, for compatibility reasons, also a board with only one T800 processor but this is not very interesting from a performance point of view. Meiko contends that any mix of the three types of boards can be configured and no upper limit for configurations is given. Therefore, no peak performance is given in the list above.

The communication speed on- and off board are the same and is about 6 MB/s at most. This speed can be achieved by combining all the links of the intermediate T800s. Still, to feed the i860 processors this speed is quite low. The Computing Surface Network that connects the boards works conceptually as a multi-stage crossbar, however, all types of configurations can be set up in software using CS tools. This enables to accommodate several users that in principle all can deal with different topologies

Measured Performances: [3] shows a speed of 1.3 Gflop/s for a configuration of 62 i860 processors on, an order 8500 full linear system.

3.4.13 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.
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 Computing Surface 2 (CS-2) is sufficiently different from the i860 Computing Surface (3.4.12) to warrant a separate entry in this section. 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 [12].

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: At this moment no performance figures are available. Only a few systems with vector nodes are shipped at present and no reliable results are known from these systems.

3.4.14 The nCUBE 2S.

Machine type: Distributed-memory multi-processor.
Models: nCUBE 2S.
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 2S
Clock cycle	40 ns
Theor. peak performance	
Per Proc. (64-bits)	3.0 Mflop/s
Maximal(64-bits)	19.7 Gflop/s
Main memory	≤256GB
Memory/node	≤64MB
Communication bandwidth	2.75 MB/s
No. of processors	8–8192

Remarks:

The nCUBE 2S is presently the only commercially available machine with a hypercube structure. The nCUBE uses in-house developed processors which have a performance of 4.1 Mflop/s in 32-bit precision and of 3.0 Mflop/s in 64-bit precision. The 2S model is virtually identical to the former nCUBE 2 series, however, the clock cycle of the processor has been speeded up by 20%. The communication speed is 2.75 MB/s and a wormhole routing technique is used that makes the communication speed almost independent of the distance between nodes in the system. The maximum size of the nCUBE 2S is 8192 processors. There are 14 outward connections per node of which 13 are used for inter-processor communication while the fourteenth is used for the distributed I/O system which therefore has the nice property that it scales with the number of nodes.

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: For 1024-processor nCUBE 2 a speed of 1.9 Gflop/s was observed for a 21,376-order linear system in 64-bit precision [3], about 77% of the quoted peak performance.

3.4.15 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 pipelined. So, 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: No performance figures are available yet for the Cenju-3.

3.4.16 The Parallel Computing Industries system.

This system is identical to the Meiko CS-2 (see 3.4.13 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.

3.4.17 The Parsys SN9000.

Machine type: Distributed-memory multi-processor.
Models: SN9800 (SN9400, SN9500, SN9600).
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	SN9800
Clock cycle	20 ns
Theor. peak performance	
Per Proc. (32 bits)	25 Mflop/s
Maximal (32 bits)	51.2 Gflop/s
Main memory	≤128 GB
Memory/node	≤64 MB
Communication bandwidth	20 MB/s/link
No. of processors	≤2048

Remarks:

The Parsys SN9000 series systems are the successors of the Parsys SN1000 machines. The latter had the INMOS T800 transputer as their basic processors. The new SN9000 systems use the T9000 transputers for that purpose. The production and marketing of the new systems has mainly waited for the limited availability of the T9000 processors which presumably recently have been overcome.

The T9000 is roughly 10 times faster than its predecessor, the T800 while the communication speed per link has increased 8 times. Although this is not overly fast as current node processors go, the T9000 has the nice feature of integrated communication and the capability of using the fast C104 communication switch. The same multistage crossbar switch is also used in the Meiko CS-2 (see 3.4.13) 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, smaller models, like the desktop models SN9400 and SN9500. The latter houses up to 32 processors and can be driven either as a Sun SparC back-end system or as a network device via Ethernet.

Measured Performances: Because of the late availability, information on the SN9000 systems is scarce and no performance figures are available yet for these systems.

3.4.18 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.
System parameters:

Model	GC/Power Plus
Clock cycle	12.5 ns
Theor. peak performance	
Per proc. (64-bits)	80 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 MC601 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.

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 [1]) a speed of 2.8 Gflop/s was observed on 64 processors.

3.4.19 The Thinking Machine CM-5.

Machine type: Distributed-memory multi-vectorprocessor.
Models: CM-5.
Operating system: CMost (TMCs Unix variant).
Connection structure: Hypercube, tree (see remarks).
Compilers: CM Fortran (Fortran 90 like), C* (C with parallel extensions), *Lisp (Lisp with parallel extensions).

System parameters:

Model	CM-5
Clock cycle	31.25 ns
Theor. peak performance	
64-bits precision	2 Tflop/s
Main memory	≤524GB
Memory/node	≤32MB
Communication bandwidth	5–20 MB/s
No. of processors	16–16,384

Remarks:

At the time of writing, it is not yet completely sure whether TMC will survive in some way. Presently, they have stopped development and production of new systems. At the moment the possibility of continueing business is dependent on possible cooperation with another vendor of parallel computer systems. As the outcome is not yet completely sure (but not very hopeful), we still include the CM-5 entry.

Contrary to its predecessor (CM-2, CM-200) the CM-5 can be used as a DM-MIMD machine, which makes operation much more flexible. The very simple bit-sliced processors as used in the CM-2 have disappeared. The basic computational node now consists of a SPARC processor of which in turn the performance can be enhanced by adding up to 4 vector units. Such a maximal node should deliver 128 Mflop/s in 64-bit precision.

Apart from the computational processors, so-called Control Processors (CPs) can be configured (the amount depending on the size of the configuration). CPs handle system tasks, I/O requests, etc. For I/O processors there is an analog possibility.

The CM-5 has several networks. There are a control- and a data network. As the name suggests, the first is used for synchronisation and control, while the second one is used for massive data movement. A "fattened tree" topology is employed, i.e., a tree structure that scales the bandwidth with the distance to the root in three levels, each nearer level being having a bandwidth which is twice that of the more distant level. This overcomes to some extent the bandwidth problems as found in normal trees. At the first level the communication speed is 20 MB/s (10 MB/s/wire; 2 wires/node), at the second level this speed is halved to 10 MB/s, while for all higher levels the speed is again halved to 5 MB/s.

The technical specifications allow a machine to be built with 16,384 processors. However, this machine would probably be too expensive for anyone to buy: a price of 330 M$ was mentioned for such a system.

Measured Performances: In [3] a speed of 59.7 Gflop/s is reported on the solution of a dense linear system of order 52224 on a 1024 node system. For a matrix multiplication a speed of 18 Gflop/s was found for a 256-node system, while 70 Gflop/s on a 1024-node machine was attained on a code that solves a simplified Boltzman equation.

3.4.20 The Transtech Paramid.

Machine type: RISC-based distributed-memory multi-processor.
Models: GF1, PM.
Operating system: Unix on front-end system, back-end OS transparent to the user .
Connection structure: Bus.
Compilers: Fortran 77, ANSI C.
System parameters:

Model	GF1	PM
Clock cycle	20 ns	20 ns
Theor. peak performance		
per proc. (32-bits)	100 Mflop/s	100 Mflop/s
per proc. (64-bits)	75 Mflop/s	75 Mflop/s
Maximal (32-bits)	1 Gflop/s	6.4 Gflop/s
Main memory	≤160 MB	≤2 GB
Memory/node	≤16 MB	≤32 MB
Communication bandwidth	2.0 MB/s/link	
No. of processors	10	4–64

Remarks

In many respects the Transtech machines are comparable to the CS-1 series (see 3.4.12) from Meiko. Both use Intel i860 processors for computation, while T805 transputers are employed for communication. A difference lies in the fact that Transtech uses the i860XP (similar to the Intel Paragon, see 3.4.10), and that the models are not presented as being infinitely scalable. The maximum number of users per system is 4 for the GF1 and 16 for the PM model. Like Parsytec (see 3.4.18) Transtech hopes to use T9000 transputers instead of the current T805s to boost the communication speed by a factor of 10.

If desired, I/O can be done via the Paramid systems themselves, instead of using the I/O system of the front-end workstation. The Paramids are connected to the front-end systems by a SCSI-2 bus, not needing any special interface at the front-end side. The Paramid systems support up to 4 disks.

The Paramid systems support the PVM and PARMACS libraries for communication. Furthermore, numerical libraries are offered to make the most of the processing speed of the i860XP which are known to be difficult to program efficiently.

Measured Performances: The speed observed for the solution of an order $N = 1000$ dense linear system on 16 processors was 320 Mflop/s. For the GR1 benchmark with $n = 251$ (see [9]), a speed of 71 Mflop/s was found on 16 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 quatities 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:
  1) The manufacturer is out of business.
  2) 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/80.
Type: Shared memory vector-parallel, max. 8 processors.
Theoretical Peak performance: 190 Mflop/s
Reason for disappearance: replaced by newer FX/2800 (see below).

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 out of business.

Machine: Convex C200.
Type: Shared memory vector-parallel, max. 4 processors (C240).
Theoretical Peak performance: 200 Mflop/s
Reason for disappearance: replaced by newer C4 series (see 3.3.1 and below).

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 3.3.1 and below).

Machine: Cray S-MP.
Type: Shared memory vector-parallel, max. 8 processors.
Theoretical Peak performance: 267 Mflop/s
Reason for disappearance: replaced on one hand by the CS6400 (symmetric multi-processing machine with $\leq$64 SuperSparc processors) which is not targeted at the high-performance computing market. On the other hand the Cray Y-MP EL fills the gap with respect to vector processing. In short: a marketing strategic reason.

Machine: Cray Computer Corporation Cray-2.
Type: Shared memory vector-parallel, max. 4 processors.
Theoretical Peak performance: 1.95 Gflop/s
Reason for disappearance: replaced by newer Cray-3 (see 3.3.2).

Machine: Cray Y-MP.
Type: Shared memory vector-parallel, max. 8 processors.
Theoretical Peak performance: 2.6 Gflop/s
Reason for disappearance: replaced by newer Y-MP C90 (see 3.3.4).

Machine: Digital Equipment Corp. VAX 9000 series.
Type:Shared memory vector-parallel system, max. 4 processors.
Theoretical Peak performance: 500 Mflop/s
Reason for disappearance: DEC does not pursue high-performance computing in this product line anymore.

Machine: Encore S/90 series.
Type: Bus-based parallel system, max. 32 processors.
Theoretical Peak performance: 128 Mflop/s
Reason for disappearance: technologically obsolete for high-performance computing.

Machine: Fujitsu VP2000 series.
Type: Single-processor vectorprocessors.
Theoretical Peak performance: 5 Gflop/s
Reason for disappearance: replaced by newer VPX series (see 3.1.1).

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 in this product line anymore.

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 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: NEC SX-2.
Type: Single-processor vector processors.
Theoretical Peak performance: 1.3 Gflop/s
Reason for disappearance: replaced by newer SX-3R series (see 3.3.7).

Machine: Sequent Symmetry series.
Type: Bus-based parallel system, max. 30 processors.
Theoretical Peak performance: 75 Mflop/s
Reason for disappearance: technologically obsolete for high-performance computing.

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:replaced by the newer CM-5 (see 3.4.19).

Machine: Wavetracer Zephyr.
Type: SIMD parallel machine, max. 16K processors.
Theoretical Peak performance: — Mflop/s
Reason for disappearance: Manufacturer out of business.

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 two machines that will probably come on the market by the end of 1994.

5.1 The nCUBE 3

The nCUBE 3 has a structure that is very similar to that of its predecessor, the nCUBE 2. It has also a hypercube structure, but unlike the nCUBE 2, the maximum number of processors is 65,536 instead of the present maximum of 8192. Furthermore, the peak performance per node it expected to be much higher: 100 Mflop/s instead of 3.0 Mflop/s at a clock rate of 20 ns. The (very) theoretical peak performance of such a system would become 6.5 Tflop/s. Also the communication bandwidth is scaled up from 2.75 MB/s (point-to-point) to roughly 115 MB/s. The usual compilers, like Fortran 77, C, C++ will be supported, as well as High Performance Fortran and Data Parallel C.

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

[1] Bailey D.H., J. Barton, T. Lasinski, H. Simon, *The NAS parallel benchmarks*, NASA Ames Research Center, Report RNR-91-02, Moffett Field, CA 94035, Jan. 1991.

[2] Beguelin A., J.J. Dongarra, A. Geist, R. Manchek, V. Sunderam, *A users' guide to PVM Parallel Virtual Machine*, Oak Ridge National Laboratory, ORNL/TM-11826, Jan. 1992.

[3] Dongarra J.J., *Performance of Various Computers Using Standard Linear Equations Software in a Fortran Environment*, `netlib` version of Nov. 15 1992, Oak Ridge National Laboratory, Technical Report CS-89-85.

[4] Dongarra J.J., I. Duff, `netlib` *report on high-performance computers*, Oak Ridge National Laboratory, version as of August 1992.

[5] Flanders P.,*Matrix Multiplication on 'C' series DAPs*, AMT Document TR40, Jan. 1991.

[6] Flynn M.J., *Some computer organizations and their effectiveness*, IEEE Trans. Comput. **C-21**, 1972, 948-960.

[7] Geers N., W. Wälde, *Highly efficient basic numerical software for supercomputers*, Proc. International Symposium on Supercomputing, Fukuoka, Japan, Nov. 1991, 219–228.

[8] Grassl C.M., Cray Research Inc., private communication.

[9] Hey A.J.G, *The Genesis distributed memory benchmarks*, Parallel Computing 17, 1991, 1275-1283.

[10] Hockney R.W., C.R. Jesshope, *Parallel Computers 2*, Adam Hilger Ltd, Bristol, 1988.

[11] Horie T., H. Ishihata, T. Shimizu, S. Kato, S. Inano, M. Ikesaka, *AP1000 architecture and performance of LU decomposition*, Proceedings of the International Conference on Parallel Processing, 1991, I-634–635.

[12] High Performance Fortran Forum, *Draft High Performance Fortran Language Specification*, Version 0.4, Rice University, Houston, November 1992.

[13] Ishihata H., T. Horie, T. Shimizu, S. Kato, S. Inano, M. Ikesaka, *Third Generation Message Passing Computer AP1000*, Proceedings of the International Symposium on Supercomputing, Fukuoka, November 1991, 46–55.

[14] Iwamiya T., M. Fukuda, *The Numerical Wind Tunnel of the National Aerospace Laboratory*, Proc. of the Workshop on Benchmarking and Performance evaluation in High Performance Computing, Tokyo, July 1993, 53–58.

[15] Kadota H. *et al.*, *Parallel Computer ADENART — Its Architecture and Application —*, Proc. of the ACM International Conference on Supercomputing, June 1991, Cologne, 1–8.

[16] Message Passing Interface Forum, *Draft document for a standard message-passing interface*,`netlib`, `netlib@ornl.gov`, November 1993.

[17] Pas R. van der, Convex Computer Corporation, private communication.

[18] Prevost L. *Experimentation and evaluation of parallel multiprocessor machines on industrial codes*, in Proceedings of the 2nd EuroBen Workshop, Oct. 1991, 59–80.

[19] Rijk P.P.M. de, *The Linear Algebra codes using BLAS levels 2 and 3*, in Proceedings of the 2nd EuroBen Workshop, Oct. 1991, 121–126.

[20] Schreiber R.S., *Application case studies — sparse matrix methods*, in Tutorial 11 of IEEE Supercomputing '91: Using Massively Parallel Supercomputers: An Applications Perspective, Albuquerque, Nov. 1991.

[21] Simmons M.L., H.J. Wassermann, O.M. Lubeck, C. Eoyang, R. Mendez, H. Harada, M. Ishiguro, *A Performance Comparison of Three Supercomputers: Fujitsu VP-2600, NEC SX-3, and Cray Y-MP*, in Proceedings of IEEE Supercomputing '91, Albuquerque, Nov. 1991, 150–157.

[22] Steen van der A.J., *The benchmark of the EuroBen Group*, Parallel Computing 17, 1991, 1211-1221.

[23] Steen van der A.J., *Benchmark results for the Hitachi S3800*, Supercomputer **10**, No. 4/5, 1993, 32–45.

[24] Trew A., G. Wilson, *Past, Present, Parallel*, Springer-Verlag, London, 1991.