Overview of recent supercomputers

Aad J. van der Steen
Dept. of Computational Physics
Utrecht University
The Netherlands
steen@fys.ruu.nl
www.fys.ruu.nl/~steen

NCF/Utrecht University February 1998

Abstract

In this report we give an overview of parallel an 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 an MIMD machines are discerned. The information about each machine is kept as compact as possible. Moreover, no attempt is made to quote price information as this is often even more elusive than the performance of a system.

This documents reflects the technical state of the supercomputer arena as accurately as possible. However, the author nor NCF take any responsibility for errors or mistakes in this document. We encourage anyone who has comments or remarks on the contents to inform us, so we can improve this report.

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

Contents

1 Introduction and account

This is the eighth 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 is 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. Like last year, the present report will be somewhat shorter than earlier ones: at this moment systems are disappearing at a faster rate than new ones replace them. The reasons for this seem to be threefold:

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

These effects make the high performance computing scene somewhat more clear (and also somewhat less adventurous). Still, the supercomputer market is very dynamic and we cannot hope to 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 is as follows: they should be either available commercially at the time of appearance of this report, or within 6 months thereafter. This excludes interesting resarch systems like the Intel ASCI Option Red system at Sandia National Laboratory (with a measured performance of 1.3 Tflop/s), the CP-PACS at the University of Tsukuba (measured performance of 368 Gflop/s) and the Numerical Wind Tunnel at the National Aerospace Lab. in Japan (230 Gflop/s), because they are not marketed and only available at the institutes mentioned and, therefore, of not much benefit to the supercomputer community at large.
The rule that systems should be available within a timespan of 6 months 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 C4600 series or the Cray C90 series, or the Thinking Machines CM-5 anymore although these systems are still used widely. Generally speaking, we include machines that are still marketed or will be marketed within 6 months. To add to the information given in this report, we quote the Web addresses of the vendors because the information found there may be more recent than what can be provided here. On the other hand, such pages should be read with care because it will not always be clear what the status is of the products described there.
Some vendors offer systems that are identical in all respects except in the clock cycle of the nodes (examples are the Digital Alphaserver and the Fujitsu AP3000). In these cases we always only mention the models with the fastest clock as it will be always possible to get the slower systems and we presume that the reader is primarily interested in the highest possible speeds that can be reached with these systems.

The availability of relatively low-cost (RISC) processors and network products to connect these processors together with standardised communication software have with has stimulated the building of home-brew clusters computers as an alternative to complete systems that offered by vendors. Obviously, we cannot list all the variants that are and have been built by groups all over the world.

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 the best measured performance will not always reflect the performance of the reader's favourite 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) biased, this is entirely the responsibility of the author of this report. He is quite willing to be corrected or to receive additional information from anyone who is in the position to do so.

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

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

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

Lastly, we should point out that also a WWW version is available. The URLs are:
`www.netlib.org/utk/papers/advanced-computers/paper.html` (USA).
`www.nwo.nl.ncf/` (Europe).
`www.fys.ruu.nl/~steen` (Europe).

2 Architecture of high performance computers

Before going on to the descriptions of the machines themselves, it is important to consider some mechanisms that are or have been used to increase the performance. The hardware structure or *architecture* determines to a large extent what the possibilities and impossibilities are in speeding up a computer system beyond the performance of a single CPU. Another important factor that is considered in combination with the hardware is the capability of compilers to generate efficient code to be executed on the given hardware platform. In many cases it is hard to distinguish between hardware and software influences and one has to be careful in the interpretation of results when ascribing certain effects to hardware or software peculiarities or both. In this chapter we will give most emphasis to the hardware architecture. For a description of machines that can be considered to be classified as "high-performance" one is referred to [20] and, for more recently available systems, [18].

2.1 The main architectural classes

Since many years the taxonomy of Flynn [4] 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 is described.

- **SISD** machines: These are the conventional systems that contain one CPU and hence can accommodate one instruction stream that is executed serially. Nowadays many large mainframes may have more than one CPU but each of these execute instruction streams that are unrelated. Therefore, such systems still should be regarded as (a couple of) SISD machines acting on different data spaces. Examples of SISD machines are for instance most workstations like those of DEC, Hewlett-Packard, and Sun Microsystems. The definition of SISD machines is given here for completeness' sake. We will not discuss this type of machines in this report.
- **SIMD** machines: Such systems often have a large number of processing units, ranging from 1,024 to 16,384 that all may execute the same instruction on different data in lock-step. So, a single instruction manipulates many data items in parallel. Examples of SIMD machines in this class are the CPP DAP Gamma II and the Alenia Quadrics.

  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. An example of such systems is for instance the Hitachi S3600.
- **MISD** machines: Theoretically in these type of machines multiple instructions should act on a single stream of data. As yet no practical machine in this class has been constructed nor are such systems easily to conceive. We will disregard them in the following discussions.
- **MIMD** machines: These machines execute several instruction streams in parallel on different data. The difference with the multi-processor SISD machines mentioned above lies in the fact that the instructions and data are related because they represent different parts of the same task to be executed. So, MIMD systems may run many sub-tasks in parallel in order to shorten the time-to-solution for the main task to be executed. There is a large variety of MIMD systems and especially in this class the

Flynn taxonomy proves to be not fully adequate for the classification of systems. Systems that behave very differently like a four-processor Cray Y-MP T94 and a thousand processor nCUBE 2S fall both in this class. In the following we will make another important distinction between classes of systems and treat them accordingly.

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

Although the difference between shared and distributed-memory machines seems clear cut, this is not always entirely the case from user's point of view. For instance, the late Kendall Square Research systems employed the idea of "virtual shared-memory" on a hardware level. Virtual shared-memory can also be simulated at the programming level: A specification of High Performance Fortran (HPF) was published in 1993 [9] which by means of compiler directives distributes the data over the available processors. Therefore, the system on which HPF is implemented in this case will look like a shared-memory machine to the user. Other vendors of Massively Parallel Processing systems (sometimes called MPP systems), like HP/Convex and SGI/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. To the user it seems that such systems have one *global address space* spanning all of the memory in the system. In addition, packages like TreadMarks ([1]) provide a virtual shared-memory environment for networks of workstations.

Another trend that has came up in the last few years is *distributed processing*. This takes the DM-MIMD concept one step further: instead of many integrated processors in one or several boxes, workstations, mainframes, etc., are connected by Ethernet, FDDI, or otherwise and set to work concurrently on tasks in the same program. Conceptually, this is not different from DM-MIMD computing, but the communication between processors is often orders of magnitude slower. Many packages to realise distributed computing are available. Examples of these are PVM (standing for Parallel Virtual Machine) [5], and MPI (Message Passing Interface, [14]). PVM and MPI have been adopted for instance by HP/Convex, SGI/Cray, and IBM for the transition stage between distributed computing and MPP on the clusters of their favorite processors and they are available on a large amount of distributed memory MIMD systems and even on shared-memory MIMD systems for compatibility reasons. In addition there is a tendency to cluster shared-memory systems, for instance by HIPPI channels, to obtain systems with a very high computational power. E.g., the NEC SX-4, and the Convex Exemplar SPP-2000X have this structure. In addition, the latter system has a software environment that allows for virtual shared-memory addressing.

2.2 Shared-memory SIMD machines

This subclass of machines is practically equivalent to the single-processor vectorprocessors, although other interesting machines in this subclass have existed (viz. VLIW machines [15]). In the block diagram in Figure

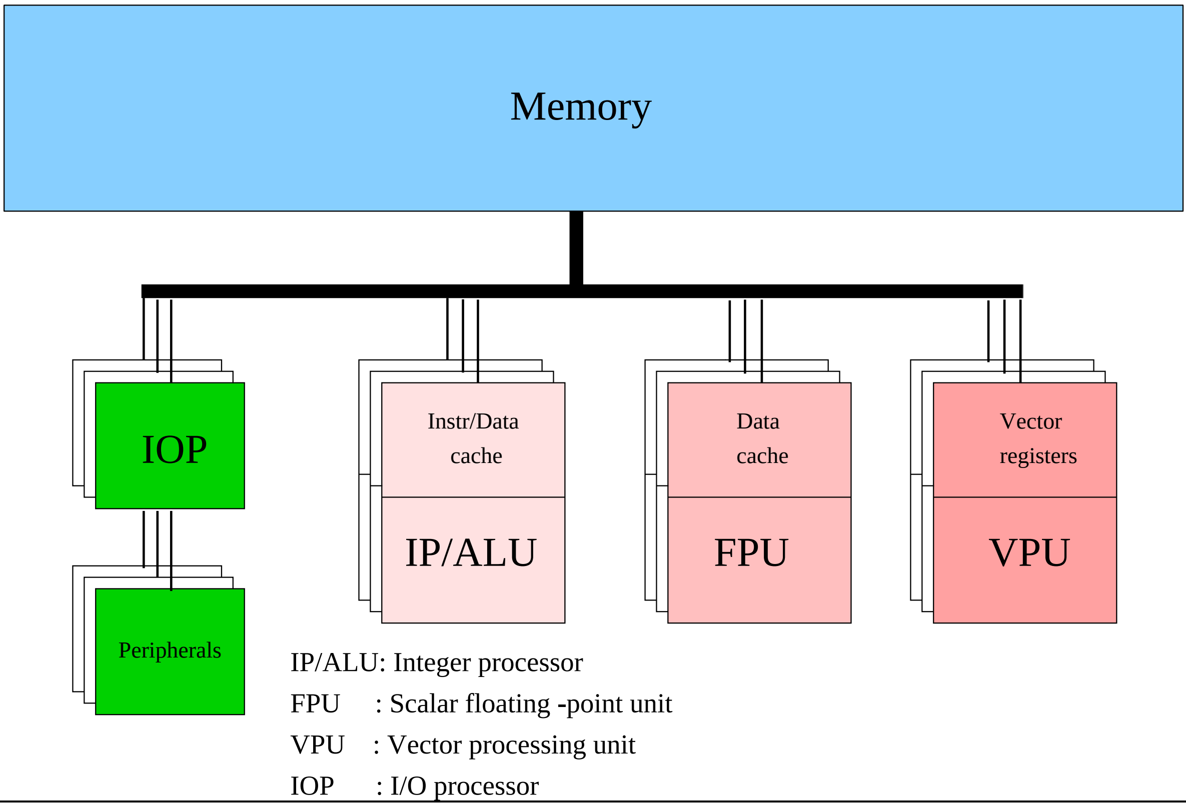

Figure 2.1: *Block diagram of a vector processor.*

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

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

Because of the generic nature of Figure 2.1 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. The influence of the number of load/store paths for the dyadic vector operation $c = a + b$ (a, b, and c vectors) is depicted in Figure 2.2. 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 VPUs are shown as a single block in Figure 2.1. Yet, again there is a considerable diversity in the structure of VPUs. Every VPU consists of a number of vector functional units, or "pipes" that fulfill one or several functions in the VPU. Every VPU will have pipes that are designated to perform memory access functions, thus assuring the timely delivery of operands to the arithmetic pipes and of storing the results

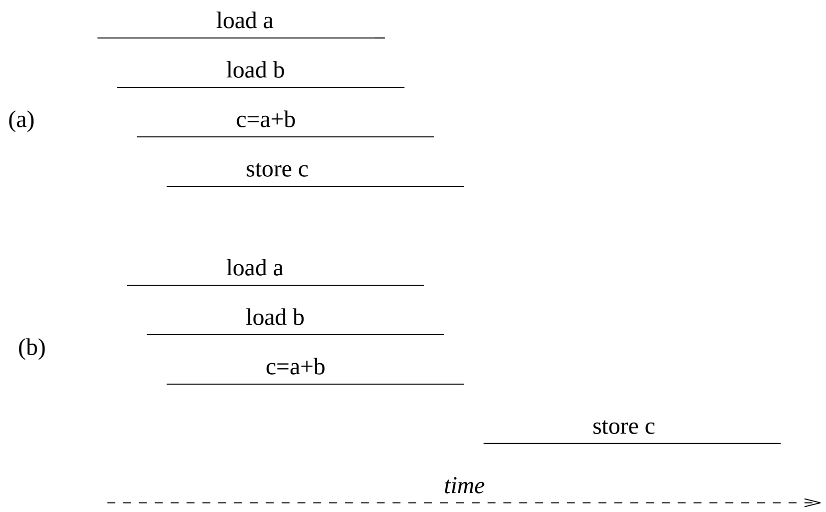

Figure 2.2: *Schematic diagram of a vector addition. Case (a) when two load- and one store pipe are available; case (b) when two load/store pipes are available.*

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. Division is performed by an iterative procedure, table look-up, or a combination of both using the add and 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 up to 4-fold replication occurs). Ideally, this will increase the performance per VPU by the same factor provided the bandwidth to memory is adequate.

2.3 Distributed-memory SIMD machines

Machines of this type are sometimes also known as *processor-array* machines [7]. Because the processors of these machines operate in lock-step, i.e., all processors execute the same instruction at the same time (but on different data items), no synchronisation between processors is required. This greatly simplifies the design of such systems. A *control processor* issues the instructions that are to be executed by the processors in the processor array. All currently available DM-SIMD machines use a front-end processor to which they are connected by a data path to the control processor. Operations that cannot be executed by the processor array or by the control processor are off-loaded to the front-end system. For instance, I/O may be through the front-end system, by the processor array machine itself or both. Figure 2.3 shows a generic model of a DM-SIMD machine of which actual models will deviate to some degree. Figure 2.3 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

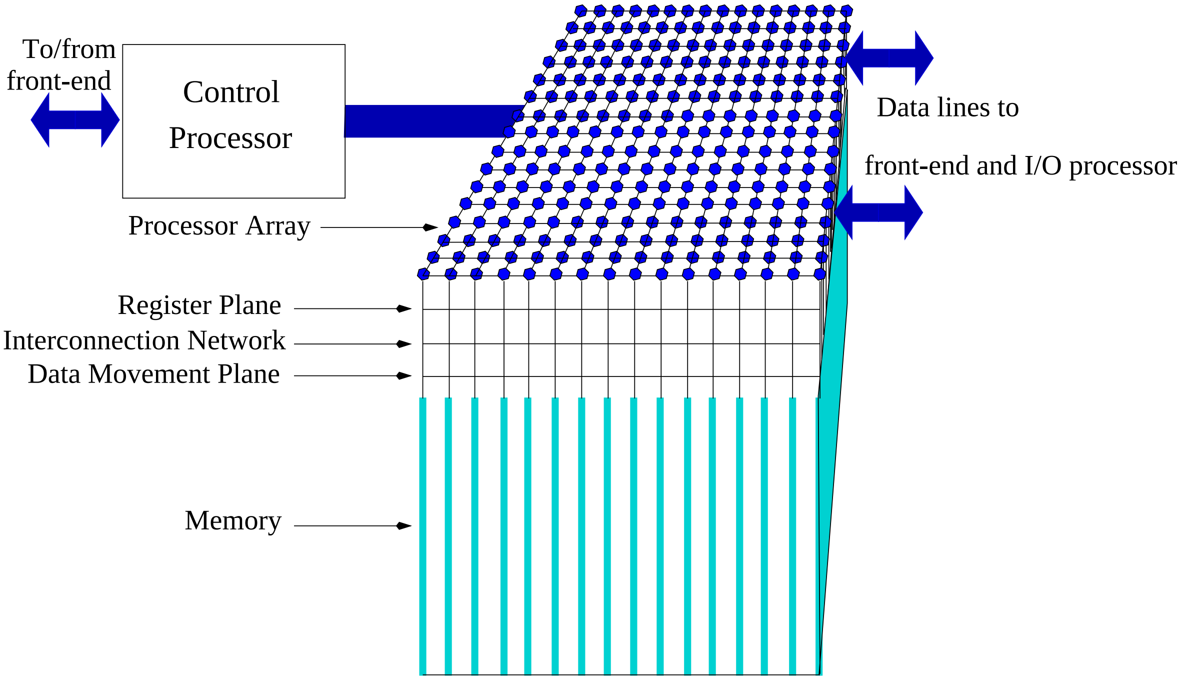

Figure 2.3: *A generic block diagram of a distributed-memory SIMD machine.*

of Monte Carlo simulations where virtually no data exchange between processors is required and exactly the same type of operations is done on massive datasets with a size that can be made to fit comfortable in these machines.

The control processor as depicted in Figure 2.3 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., operations on integers are produced by software routines on these simple bit-serial processors which takes at least as many cycles as the operands are long. So, a 32-bit integer result will be produced two times faster than a 64-bit result. For floating-point operations a similar situation holds, be it that the number of cycles required is a multiple of that needed for an integer operation. 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 slower operation on floating-point numbers can be often compensated for by their number, while the cost per processor is quite low as compared to full floating-point processors. In some cases, however, floating-point coprocessors are added to the processor-array. Their number is 8–16 times lower than that of the bit-serial processors because of the cost argument. An advantage of bit-serial processors is that they may operate on operands of any length. This is particularly advantageous for random number generation (which often boils down to logical manipulation of bits) and for signal processing 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

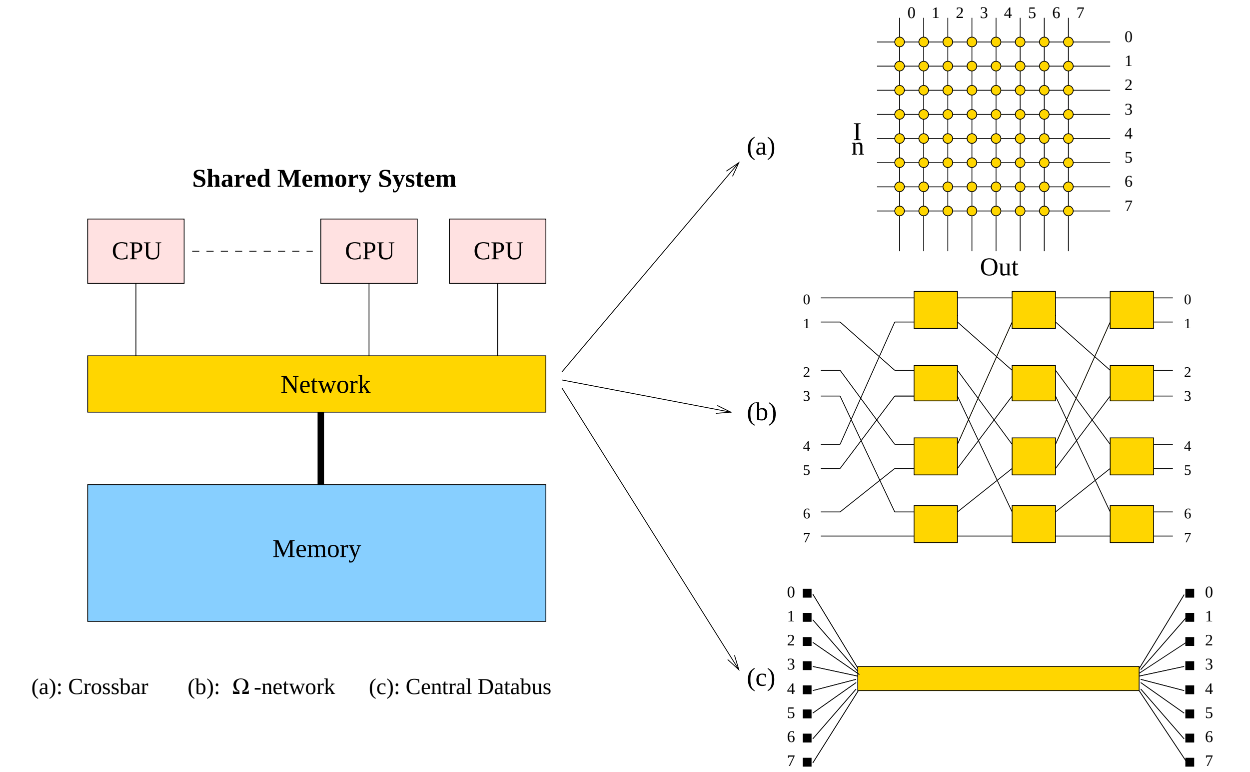

Figure 2.4: *Some examples of interconnection structures used in shared-memory MIMD systems.*

operands, this may result in significant speedups.

2.4 Shared-memory MIMD machines

In Figure 2.1 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 $\mathcal{O}(n^2)$ while increasing the number of processors with $\mathcal{O}(n)$. So, various alternatives have been tried. Figure 2.4 shows some of the interconnection structures that are (and have been) used.

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

The bus connection is the least expensive solution, but it has the obvious drawback that bus contention may occur thus slowing down the computations. Various intricate strategies have been devised using caches associated with the CPUs to minimise the bus traffic. This leads however to a more complicated bus structure which raises the costs. In practice it has proved to be very hard to design buses that are fast enough, especially where the speed of the processors have been increasing very quickly and it imposes an upper bound on the number of processors thus connected that in practice appears not to exceed a number of 10-20. In 1992, a new standard (IEEE P896) for a fast bus to connect either internal system components or to external systems has been defined. This bus, called the Scalable Coherent Interface (SCI) should provide a point-to-point bandwidth of 200-1,000 Mbyte/s. It is in fact used in the HP/Convex SPP-2000, but could also be used within a network of workstations. The SCI is much more than a simple bus and it can act as the hardware network framework for distributed computing, see [10].

The Ω-network is a structure which is situated somewhere in between a bus and a crossbar which respect to potential capacity and costs. At this moment of the commercially available machines the IBM SP2, the Meiko CS-2, and the Cenju-3 use this network structure, but a number of experimental machines also have used this or a similar kind of interconnection. The BBN TC2000 that acted as a virtual shared-memory MIMD system used an analogous type of network (a Butterfly-network) and it is quite conceivable that new machines may use it, especially as the number of processors grows. For a large number of processors the $n \log_2 n$ connections quickly become 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. Obviously, not only the *structure* but also the *width* of the links between the processors is important: a network using 16-bit parallel links will have a bandwidth which is 16 times higher than a network with the same topology implemented with serial links.

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

Whichever network is used, the type of processors in principle could be arbitrary for any topology. In practice, however, bus structured machines do not have vector processors as the speeds of these would grossly mismatch with any bus that could be constructed with reasonable costs. All available bus-oriented systems use RISC processors. The local caches of the processors can sometimes alleviate the bandwidth problem if the data access can be satisfied by the caches thus avoiding references to the memory.

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 synchronise via the shared memory. Generally, this is much slower but may still be acceptable when the synchronisation occurs relatively seldom. Of course in bus-based systems communication also has to be done via a bus. This bus is mostly separated from the databus to assure a maximum speed for the synchronisation.

2.5 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. The latter type of machines are processor-array systems in which the data structures that are candidates for parallelisation are vectors and multi-dimensional arrays that are laid out automatically on the processor array by the system software. For shared-memory systems the data distribution is completely transparant to the user. This is quite different for DM-MIMD systems where the user has to distribute the data over the processors and also the data exchange between processors has to be performed explicitely. The initial reluctance to use DM-MIMD machines seems to have been decreased. Partly this is due to the now existing standard for communication software ([5, 14]) 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 sys-

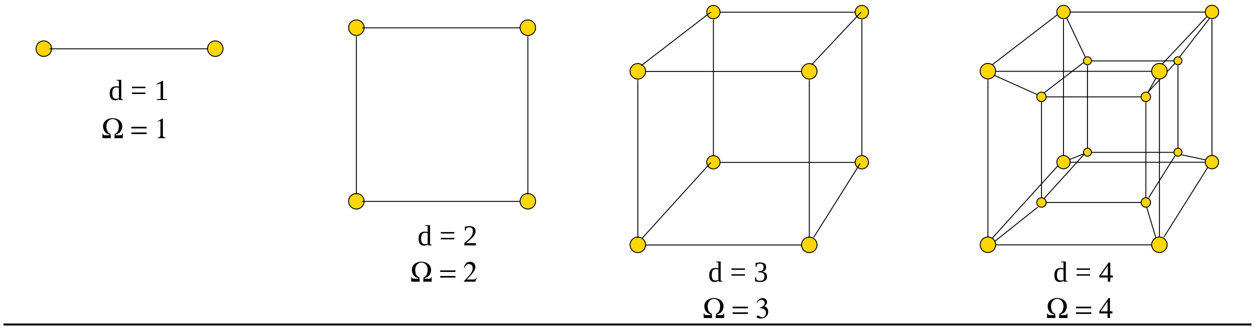

Figure 2.5: *1-, 2-, 3-, and 4-dimensional hypercube connections.*

tems 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 paragraph above that also for DM-MIMD machines both the topology and the speed of the data paths are of crucial importance for the practical usefulness of a system. Again, as in the section on SM-MIMD systems¡/A¿, the richness of the connection structure has to be 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 2.5.

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 node i to node j a header message is sent from i to j, resulting in a direct connection between rthese nodes. As soon as this connection is established, the data proper is sent through this connection without disturbing the operation of the intermediate nodes. Except for a small amount of time in setting up the connection between nodes, the communication time has become virtually independent of the distance between the nodes. Of course, when several messages in a busy network have to compete for the same paths, waiting times are incurred as in any network that does not directly connect any processor to all others and often rerouting strategies are employed to circumvent busy links.

A fair amount of 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 [8].

A large 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 such systems.

As with SM-MIMD machines, a node may in principle consist of any type of processor (scalar or vector) for computation or transaction processing together with local memory (with or without cache) and, in almost all cases, a separate communication processor with links to connect the node to its neighbours. Nowadays, the node processors are mostly off-the-shelf RISC processors sometimes enhanced by vector processors. A problem that is peculiar to this DM-MIMD systems 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 (almost) available systems

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

3.1 Shared-memory SIMD systems

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

3.1.1 The Hitachi S3600 series.

Machine type: Vectorprocessor
Models: S3600/120, S3600/140, S3600/160, S3600/180
Operating system: VOS3/HAP/ES (IBM MVS compatible) and OSF/1
Compilers: FORT77/HAP vectorising Fortran 77, C, C++.
Vendors information Web page: none.
Year of introduction: 1994.

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	0.25 Gflop/s	1.0 Gflop/s	2.0 Gflop/s
Main memory	128-256 MB	256-512 MB	256-512 MB	512-1024 MB
Extended memory	$\leq$ 6 GB	$\leq$ 16 GB	$\leq$ 16 GB	$\leq$ 16 GB

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.

A unique feature of the S3600, as in its direct predecessor the S-820, is that all machines of the series are air cooled. All other machines in this class relied at least on water cooling.
Unlike the S-820 series, the S3600 series is also marketed worldwide, not only in Japan. This is also the case for the 3.3.3 SM-MIMD machines.

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

3.2 Distributed-memory SIMD systems

3.2.1 The Alenia Quadrics.

Machine type: Processor array.
Models: Quadrics Qx, QHx, $x = 1, \ldots, 16$.
Front-end: Almost any 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.
Vendors information Web page: `www.quadrics.com`
Year of introduction: 1994.

System parameters:

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

Remarks:
The Quadrics is a commercial spin-off of the APE-100 project of the Italian National Institute for Nuclear Physics. Systems are available in multiples of 8 processor nodes in the Q-model where up to 16 boards can be fitted into one crate or in multiples of 128 nodes in the QH-model by adding up to 15 crates to the minimal 1-crate system. The interconnection topology of the Quadrics is a 3-D grid with interconnections to the opposite sides (so, in effect a 3-D torus). The 8-node floating-point boards (FPBs) are plugged into the crate backplane which provides point-to-point communication and global control distribution. The FPBs are configured as 2^3 cubes that are connected to the other boards appropriately to arrive at the 3-D grid structure.
The basic floating-point processor, the so-called MAD chip, contains a register file of 128 registers. Of these registers the first two hold permanently the values 0 and 1 to be able to express any addition or multiplication as a "normal operation", i.e., a combined multiply-add operation, where a multiplication is of the form, $a \times b + 0$ and an addition is $a \times 1 + b$. In favourable circumstances the processor can therefore deliver two floating-point operations per cycle. Instructions are centrally issued by the controller at a rate of one instruction every two clock cycles.
Communication is controlled by the Memory Controller and the Communication Controller which are both housed on the backplane of a crate. When the Memory Controller generates an address it is decoded by the Communication Controller. In case non-local access is desired, the Communication Controller will provide the necessary data transmission. The memory bandwidth per processor is 50 MB/s which means that very 2 cycles an operand can be shipped in or out a processor. The bandwidth for non-local communication turns out to be only four times smaller then local memory access.
The Quadrics communicates with the front-end system via a T805 transputer-based interface system, called the Local Asynchronous Interface (LAI). The interface can write and read the memories of the nodes and the Controller. Presently, the bandwidth of the interface to the front-end processor is not very large (1 MB/s). It is expected that this can be improved by about a factor of 30 in the near future. I/O has to be performed via the front-end system and will therefore be relatively slow.
The TAO language has several extensions to employ the SIMD features of the Quadrics. Firstly, floating-point variables are assumed to be local to the processor that owns them, while integer variables are assumed

to be global. Local variables can be promoted to global variables. Other extensions are the `ANY`, `ALL`, and `WHERE/END WHERE` keywords that can be used for global testing and control. Processors that not meet a global condition effectively skip the operation(s) that are associated with it. For easy referencing nearest-neighbour locations special constants `LEFT`, `RIGHT`, `UP`, `DOWN`, `FRONT`, and `BACK` are provided. In addition, new data types and operators on these data types are supported together with overloading of operators. This enables very concise code for certain types of calculations.

Measured performances:
No measured performances have been reported for this machine.

3.2.2 The Cambridge Parallel Processing Gamma II.

Machine type: Processor array.
Models: Gamma II Plus 1000, Gamma II Plus 4000.
Front-end: DEC, HP, or Sun workstation, stand-alone for dedicated applications.
Operating system: Internal OS transparent to the user, Unix on front-end.
Connection structure: 2-D mesh, row- and column datapaths (see remarks).
Compilers: FORTRAN-PLUS (a Fortran 77 compiler with some Fortran 90 and some proprietary array extensions), C++.
Vendors information Web page: `www.cppus.com`
Year of introduction: 1995.

System parameters:

Model	Gamma II Plus 1000	Gamma II Plus 4000
Clock cycle	33 ns	33 ns
Theor. peak performance		
Per Proc. (32-bits, Mflop/s)	0.6	0.6
Maximal (32-bits, Gflop/s)	0.6	2.4
1-bit (Gop/s)	30.7	122.8
8-bit (Gop/s)	30.7	122.8
Program memory	≤4 MB	≤4 MB
No. of processors	≤128 MB	≤512 MB
Internal communication speed		
Across row, column	120 MB/s	480 MB/s
Memory to PE	3.84 GB/s	15.4 GB/s

Remarks:
In November 1995 the new Gamma II Plus models have been announced by CPP. In essence there is not much difference with its predecessor the DAP Gamma. However, the clock cycle has tripled to 33 ns with an equivalent rise in the peak performance of the systems.
The Gamma II is presented as the fourth generation of this type of machine. Indeed, the macro architecture of the systems has hardly changed since the first ICL DAP (the first generation of this system) was conceived. As in the ICL DAP in the Gamma 1000 models the 1024 processors are ordered in a 32×32 array, while the Gamma 4000 has 4096 processors arranged in a 64×64 square.
The systems are able to operate byte parallel on appropriate operands to speed up floating-point operations, however, for logical operations bit-wise operations are possible, which makes the machines quite fast in this respect. As the byte parallel code consists of separate sequences of microcode instructions, the bit processor plane and the byte processor plane are in fact independent and can work in parallel. This is also the case for I/O operations. Also character-handling can be done very efficiently. This is the reason that Gamma systems are often used for full text searches.
As in all processor-array machines, the control processor (called the Master Control Unit (MCU) in the DAP) has a separate memory to hold program instructions while the data are held in the data memory associated with each Processing Element (PE) in the processor array. So, for a Gamma 1000 with 128 MB of data memory each PE has 128 KB of data memory directly associated to it. To access data in other PE's memories these must be brought up to the data routing plane and shifted to the appropriate processor.

As already mentioned under the heading of the connection structure, there are two ways of connecting the PEs. One is the 2-D mesh that connects each element to its North-, East-, West-, and South neighbour. In addition there are row- and column data paths that enable the fast broadcast of a row or column to an entire matrix by replication. Conversely, they can be used for row or column-wise reduction of matrix objects into a column or row-vector of results from, e.g., a summing or maximum operation.
Separate I/O processors and disk systems can be attached to the Gamma directly thus not burdening the front-end machine (and the connection between front-end and DAP) with I/O operations and unnecessary data transport. One of these I/O devices is the GIOC that can transport data to the data memory at a sustained rate of 80 MB/s transposing the data to the vertical storage mode of the data memory on the fly. Also, a direct video interface is available to operate a frame buffer.
A nice (non-standard) feature of the FORTRAN-PLUS compiler is the possibility to use logical matrices as indexing objects for computational matrix objects. This enables a very compact notation for conditional execution on the processor array. In addition, recently C++ is available.

Measured Performances:
In [3] the speed of matrix multiplication on various DAP models (precursors of the Gamma systems) is analyzed. The documentation states 32-bit floating-point add speed of 1.68 Gflop/s on 4096 PEs, while a 32-bit 1,024 complex FFT would attain 2.49 Gflop/s. No independent performance figures for the Gamma II Plus systems are available.

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 the section on 3.1 sytems where they could have been treated.

Furthermore, we have included systems here that have a shared-memory model as a basis but that may be extended by coupling several of these systems together in a distributed-memory way. The distinction is not always very clear when looking at the end product: a multi-frame DEC AlphaServer and an HP/Convex X-class look very much alike. The difference lies in the integration. In the latter machine the distributed-memory extension of multi-CPU nodes was a basis for the architecture while in the former DEC system it is more like an added feature.

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

Machine type: Shared-memory multi-vectorprocessor.
Models: Cray J90, J90se, T90.
Operating system: UNICOS (Cray Unix variant).
Compilers: Fortran 90, C, C++, Pascal, ADA.
Vendors information Web page: `www.cray.com/products/systems`
Year of introduction: J90: 1994, T90: 1995, J90se: 1997.

System parameters:

Model	Cray J90(se)	Cray T90
Clock cycle	10 ns	2.2 ns
Theor. peak performance		
Per Proc.	200 Mflop/s	1.8 Gflop/s
Maximal	6.4 Gflop/s	58 Gflop/s
Memory	≤4 GB	≤8 GB
No. of processors	4–32	1–32
Memory bandwidth		
Single proc. bandwidth	1.6 GB/s	24 GB/s

Remarks:
Cray Research Inc. (CRI) has been taken over by Silicon Graphics (SGI) but for the next few years Cray will maintain separate product lines from SGI. Here we discuss the Cray-inherited vector systems.
Cray supports at this moment 3 product lines. Two of these are multi-headed vector processors which are discussed here. The third is the T3E, a DM-MIMD machine that will be described in section 3.4.
The Cray J90 series is the entry level model marketed by CRI announced in September 1994. The J90 series is based on CMOS technology which has a low power consumption (all J90s are air cooled) and low production costs. The machine is binary compatible with the high-end systems. It has one multiply and add vector pipe set per CPU at a clock cycle of 10 ns which results in a theoretical peak performance of 200 Mflop/s. Furthermore, a cache has been added to speed up scalar processing. 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. Recently, an enhancement of the scalar speed is available with the J90se processors where "se" stands for "scalar enhanced". The vector performance of these processors is exactly the same as that of the original J90 processors. However, the scalar performance is twice as fast. This should diminish the gap in performance on scalar code between the J90 systems and the performance of pipelined fast RISC processors as employed in high-end workstations.
The Cray T90 series is built in ECL logic and has therefore a much lower clock cycle (2.2 ns) and correspondingly faster SRAM memory. As its direct predecessor, the Cray C90, every CPU contains two vector add and multiply pipes. This gives rise to a maximum of 4 floating-point results/clock cycle/CPU equivalent to a theoretical peak performance of 1.8 Gflop/s per CPU or 58 Gflop/s for a maximal system.

The Cray T90 machines are at this moment the only ones with a memory bandwidth as seems optimal for vector processors: two operands can be loaded and one result can be stored in one cycle for each pipe set. For the T90 this meant that the relative bandwidth has to be 48 bytes/cycle/CPU. This has indeed been accomplished and observed results indicate that for the T90 the performance scales up with the clock cycle and the number of functional units (see measured performances below). For the J90 series the bandwidth is lower: 16 bytes/cycle. This is regrettably less than was available in its predecessors, Y-MP EL machines, and this adversely affects the efficiency.
Another property that is unique for the Cray T90 systems is that they do not have a separate scalar processor but that scalar- and vector code have to share the same functional units. However, a small scalar cache is added to speed up scalar calculations. As mentioned, the Cray J90 series has separate scalar processors. Theoretically, the absence of separate scalar processors might impair the throughput speed (Hitachi even adds an extra scalar processor in the S-3800 series to combat excessive context switching). However, in practice the drawbacks seem rather limited.
Contrary to earlier high-end Cray systems, the T90 now features compatibility with the IEEE 754 floating-point standard. Formerly, Cray-specific floating-point arithmetic was employed which could give rise to problems in data exchange with other systems and in different computational results due to the difference in arithmetic.

Measured Performances:
On the T90 in [2] a speed of 29.4 Gflop/s was found on a 32 processor machine for the solution of an order 1000 dense linear system and of 36.6 Gflop/s for a system of order 16384. This means that efficiencies of 51 and 64%, respectively. However, in the latter case Strassen's algorithm was used for the matrix multiplications involved which gives a slightly more favourable view of the performance than when a "standard" matrix multiplication would have been used.
For a J90 series machine with 32 processors a speed of 4.486 Gflop/s was observed with linear system solving for an order 1000 system, which amounts an efficiency of 70%. Also using Strassen's algorithm for a system of order 19456 a speed of 7.6 Gflop/s was attained.

3.3.2 The Digital Equipment Corp. AlphaServer.

Machine type: RISC-based shared-memory multiprocessor.
Models: AlphaServer 8200 5/625, 8400 5/625, Cluster.
Operating system: Digital Unix (DEC's flavour of Unix).
Compilers: Fortran 77, HPF, C, C++.
Vendors information Web page: `www.digital.com/info/hpc`
Year of introduction: 1997.

System parameters:

Model	8200/625	8400/625	Cluster
Clock cycle	1.63 ns	1.63 ns	1.63 ns
Theor. peak performance			
Per Proc.	1.23 Gflop/s	1.23 Gflop/s	1.23 Gflop/s
Maximal	7.3 Gflop/s	17.2 Gflop/s	68.6 Gflop/s
Memory	$\leq$ 12 GB	$\leq$ 28 GB	$\leq$ 112 GB
No. of processors	$\leq$ 6	$\leq$ 14	$\leq$ 56
Memory bandwidth			
Processor/memory	1.87 GB/s	1.87 GB/s	1.87 GB/s
Between cluster nodes	—	—	100 MB/s

Remarks:
The AlphaServers are symmetric multi-processing systems which are based on the Alpha 21164 processor. The 8200 model is a somewhat smaller copy of the 8400 model: in the 8200 a maximum of 6 CPUs can be accomodated while this number is 14 for the 8400 model. Also, there is room for at most 12 GB of memory in the 8200 while the 8400 can house 28 GB. However, the amount of CPUs and memory is not independent. For instance, the 8400 has 9 system slots. One of these is reserved for I/O and one will have to contain at

least one CPU module which can contain 1 or 2 CPUs. From the remaining slots 7 can be used either for memory or for a CPU module. So, one has to choose for either higher computational power or for more memory. This can potentially be a problem for large applications that require both.
AlphaServers can be clustered using PCI bus Memory Channel link cables that are connected to a hub. The systems need not be of the same model. The bandwidth of this interconnect is slightly over 100 MB/s. Up to four systems can be coupled in this way. To support this kind of cluster computing, HPF and optimised versions of PVM and MPI are available.

Measured Performances:
In [2] a speed of 3.6 Gflop/s for an 8-processor AlphaServer 8400 5/625 system is reported for the solution of a dense linear system of order 1000. For a system of size 30712 a speed of 24.7 Gflop/s is given in the same report on a cluster of 64 processors of 8400 5/440s (of these systems 8 could be clustered). A single 8400 5/625 with 12 processors could attain 7.3 Gflop/s on a system of order 9548.

3.3.3 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: Fortran 77, HPF, C, C++.
Vendors information Web page: `www.hitachi.co.jp/Prod/comp/hpc/index.html`
Year of introduction: 1993.

System parameters:

Model	S-3800/x60	S-3800/y8z
Clock cycle VPU	2 ns	2 ns
Clock cycle scalar proc.	6 ns	6 ns
Theor. peak performance		
Per Proc.	4 Gflop/s	8 Gflop/s
Maximal	8 Gflop/s	32 Gflop/s
Main Memory	256–1024 MB	0.5–2 GB
Extended Memory	$\leq$ 16 GB	$\leq$ 32 GB
No. of processors		
Scalar	1–2	1–4
Vector	1–2	1–4

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 was offered in the former Fujitsu VPX200 series and for the same reason: context switching delay between jobs should be reduced by this scheme. The smallest model, the S-3800/160 has 4 multi-functional multiply/add pipes which may deliver up to 8 results per clock cycle. This is equivalent to 4 Gflop/s. In the /180 the number of pipes is doubled to 8 with a corresponding peak performance of 8 Gflop/s. All models feature one or more separate divide pipes. As the multi-headed systems can work in parallel, the top model, the S-3800/480, may theoretically attain a speed of 32 Gflop/s.
Hitachi 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.
The S-3800 can be clustered and thus used as message-passing DM-MIMD machines.

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 [17]). In [2] a speed of 27.2 Gflop/s on 4 processors is reported for the solution of an order 31,200 dense linear system. The efficiency is 85%. On a cluster of 3 4-processor S-3800/180s a speed of 78.2 Gflop/s was attained which amounts to an efficiency of 81%.

3.3.4 The NEC SX-4 series.

Machine type: Vectorprocessor.
Models: SX-4B/eA, SX-4B/A, SX-4A, SX-4.
Operating system: Super-UX (Unix variant based on BSD V.4.3 Unix).
Compilers: Fortran 90, HPF, ANSI C, C++.
Vendors information Web page: `www.nec.co.jp/english/product/computer/sx/`
Year of introduction: 1995.

System parameters:

Model	SX-4B/eA	SX-4B/A	SX-4A	SX-4
Clock cycle	8.8 ns	8.8 ns	8 ns	8 ns
Theor. peak performance				
Per Proc. (64-bits)	0.9 Gflop/s	1.8 Gflop/s	2 Gflop/s	2 Gflop/s
Maximal				
Single frame:	0.9 Gflop/s	7.2 Gflop/s	32 Gflop/s	64 Gflop/s
Multi frame:	—	—	0.5 Tflop/s	1 Tflop/s
Main Memory (per frame)	1–4 GB	2–8 GB	1–32 GB	1–16 GB
Ext. Memory (per frame)	$\leq$ 8 GB	$\leq$ 16 GB		$\leq$ 32 GB
No. of processors	1	1–4	4–256	4–512

Remarks:
The SX-4 series is comprised of a large range of machine sizes. The smallest of these is the SX-4B/eA. This machine has one CPU housing 4 vector pipe sets. As the clock cycle is 8.8 ns and each pipe set is able to deliver 2 floating-point results per cycle, the total maximum performance is 0.9 Gflop/s for this system. In all other systems the replication factor of the pipe sets is 8 which doubles the speed per CPU to a maximum of 1.8 Gflop/s in the SX-4B series and to 2 Gflop/s for the SX-4A and SX-4 series. The bandwidth from memory to the CPUs is 16 64-bit words per cycle per CPU. With a replication factor of 8 this is enough to provide two operands per pipe set but it is not sufficient to transport the results back to the memory at the same time. So, some trade-offs with the re-use of operands have to be made to attain the peak performance. The distinction between the SX-4A and the SX-4 lies in the type of memory used: in the SX-4A Synchronous DRAM memory is used while in the SX-4 Synchronous SRAM is employed. SSRAM is faster but bulkier than SDRAM, so, the maximum amount of memory per frame in an SX-4 is 16 GB while it is 32 GB per frame in the SX-4A.
The technology used is CMOS. This lowers the fabrication costs and the power consumption appreciably (the same approach is being used in the Fujitsu VPP700, see section 3.4.5) and all models are air cooled. This enables the placement of up to 32 CPUs in one frame for the SX-4 model and 16 in the SX-4A. The placement of less CPUs in the SX-4A frame is a consequence of the slower memory, not of power consumption. Beyond the maximum single frame system, it is possible to couple up to 16 frames together to form a distributed-memory system. This is equivalent to the AlphaServer cluster idea (see 3.3.2). There are two ways to couple the SX-4 frames: NEC provides a full crossbar, the so-called IXS crossbar to connect the various frames together at a speed of 16 GB/s for point-to-point out-of-frame communication (128 GB/s bisectional bandwidth for a maximum configuration). In addition, a HiPPI interface is available for interframe communication at lower cost and speed.
For distributed computing there is an HPF compiler and for message passing an optimised MPI (MPI/SX) is available. The SX-4 is the only system that supports three floating-point number systems: IBM-compatible, Cray-compatible, and the IEEE 754 standard.

Measured Performances:
In [2] a speed of 122.2 Gflop/s was reported for the solution of a full linear system of order 30080 on a 64-processor multiframe configuration. This amounts to to a very high efficiency of 95%. The author measured speeds of 1.83 Gflop/s for a matrix-vector multiplication on a single CPU (not yet published) which also shows an efficiency of over 90%.

3.3.5 The Sun E10000 Starfire.

Machine type: RISC-based shared-memory multiprocessor.
Models: E10000 Starfire.
Operating system: Solaris (Sun's Unix flavour).
Compilers: Fortran 77, Fortran 90, C, C++.
Vendors information Web page: `//www.sun.com/servers/ultra_enterprise/10000/`
Year of introduction: 1997.

System parameters:

Model	E10000
Clock cycle	4 ns
Theor. peak performance	
Per Proc. (64-bits)	500 Mflop/s
Maximal	32 Gflop/s
Main Memory	$\leq$ 64 GB
No. of processors	16–64

Remarks:
The Starfire E10000 is the largest of a series of Ex000 servers, where x can be 3, 4, 5, 6, 10. We only discuss this largest model as Sun has clearly positioned this machine themselves as a system for large-scale high performance computing. The basic processor is a 4 ns cycle UltraSPARC processor with a Theoretical Peak Performance of 500 Mflop/s. Up to 64 processors are connected by a 64×64 crossbar, the largest crossbar employed commercially. This crossbar, called the Gigaplane-XB, also makes it different from the lower-end models from the Ex000 series as these systems use a bus interconnect between processors. The system is built up from system boards each containing up to 4 processors, 2 level-2 caches ($\leq$ 4 MB) and 4 memory banks that plug into the Gigaplane crossbar which thus acts as a backplane. The caches are kept coherent by a "snoopy bus" protocol, i.e., each cache is aware of the (in)validation of data by continuous monitoring the data on the backplane and updating their copies accordingly.
The Gigaplane crossbar connects to the processors with separate data and address lines which recognises the fact that most data transfers are essentially point-to-point transfers while addresses often have to be broadcasted to many or all processors. The effective aggregate bandwidth for data is 102.4 GB/s with a point-to-point speed of 1.6 GB/s (theoretical peak).
The Starfire is a typical SMP machine with provisions for shared-memory parallelism in the Fortran and C(++) compilers by directives in the source code. A the time of writing it is not yet clear whether Sun will join the OpenMP consortium for standardising the shared-memory programming model. Of course it is possible to cluster E10000s as has for instance been done with E6000 servers and use such a cluster in a DM-MIMD way (see results for E6000 in [2]).

Measured Performances:
In [2] a speed of 26.45 Gflop/s is reported for a 64 processor machine in solving an order 19968 linear system. The efficiency for this problem is 83%. As yet no results of clustering E10000s are available.

3.3.6 The Tera MTA.

Machine type: Distributed-memory multi-processor.
Models: MTA-xC, $x = 1, \ldots, 256$.
Operating system: Unix BSD4.4 + proprietary micro kernel.
Compilers: Fortran 77 (Fortran 90 extensions), HPF, ANSI C, C++.
Vendors information Web page: `www.tera.com/`
Year of introduction: 1997.

System parameters:

Model	MTA-xC
Clock cycle	3 ns
Theor. peak performance	
Per Proc. (64-bits)	1 Gflop/s
Maximal	256 Gflop/s
Main Memory	$\leq$ 256 GB
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 in section 4).
The connection network connects a 3-D cube of p processors with sides of $p^{1/3}$ of which alternately the x- or y axes are connected. Therefore, all nodes connect to four out of six neighbours. In a p processor system the worst case latency is $4.5p^{1/2}$ cycles; the average latency is $2.25p^{1/2}$ cycles. 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 amounts to a bandwidth of 5.33 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 compiler directives, library routines, including synchronisation, barrier, and reduction operations on defined groups of threads. Controlled and uncontrolled parallelism approaches may be freely mixed. Furthermore, each variable has a full/empty bit associated with it which can be used to control parallelism and synchronisation with almost zero overhead. HPF will also be supported for SPMD-style programming.

Measured Performances:
The company planned to deliver a 2-processor system to the San Diego Supercomputing Center by the end of 1997 but it did not receive production network boards for a multiprocessor system in time. Tera expects to receive the boards shortly, and thereafter upgrade the system in stages to larger configurations. So, at this moment actual performances cannot yet be reported.

3.4 Distributed-memory MIMD systems

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

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

3.4.1 The Avalon A12.

Machine type: RISC-based distributed-memory multi-processor.
Models: Avalon A12.
Operating system: AVALON micro kernel based Unix (Image compatible with Digital Unix).
Connection structure: Multistage variable (see remarks).
Compilers: Fortran 77, Fortran 90 extensions, HPF, ANSI C.
Vendors information Web page: `www.teraflop.com/`
Year of introduction: 1996.

System parameters:

Model	A12
Clock cycle	2.5 ns
Theor. peak performance	
Per Proc. (64-bits)	800 Mflop/s
Maximal	1.3 Tflop/s
Memory/node	$\leq$ 1 GB
Memory/maximal	1.7 TB
No. of processors	12–1680
Communication bandwidth	
Point-to-point	128–400 MB/s
Bisectional (maximal)	10 GB/s

Remarks:
The A12 is be based on the DEC Alpha 21164 RISC procesor. The processor used in the system has a clock cycle of 2.5 ns. However, most of the information given at the vendors Web page still uses the data for a node processor with a 3.3 ns clock to describe the configuration properties. The Web information is therefore internally somewhat inconsistent. Because the Alpha 21164 has dual floating-point arithmetic pipes it will deliver a theoretical peak performance of 800 Mflop/s. The maximum configuration of the system is given as 1680 processors The first and second level cache that reside on chip, a 1 MB third level cache is provided on each A12 CPU card. The bandwidth to/from the first level cache is sufficient to transport two operands to the CPU and to ship one result back in one cycle. The second level cache has two-thirds of is bandwidth, while the third level cache has the capability of providing an 64-bit word every two cycles. The bandwidth to/from memory is 400 MB/s or one 64-bit word every 8 cycles. The memory has two-way interleaved banks of a memory that can be up to 1 GB/node.

Each CPU card contains a Alpha 21164 processor, the third level or B cache and the local memory for that node. Twelve CPU cards can be housed in one crate which has a full crossbar backplane. This yields a internode bandwidth of slightly under 400 MB/s between the cards within one crate. Apart from the 12 slots for CPU cards, there are two extra dual channel slots that can accomodate communication cards that provide the connections with other crates. For the in-crate crossbar CMOS technology is used. However, for the intercrate connections ECL logic is employed. The actual connections between crates are made by coaxial cables. This way of connection provides a large flexibility in the overall interconnection topology: one could build trees or toruses or a secondary level crossbar (is the last case one crate should be filled entirely with communication cards to build a 144 processor system). The communication speed between crates is less fast (but still respectable): 128 MB/s. Various configurations are described at the Web-address given above.
I/O can be configured in various ways: It is possible to put 32-bit or 64-bit PCI expansion cards on each CPU card to obtain what Avalon calls "Type 1 I/O nodes". Also, a direct switch connection via a variant of the communication card can be made to the outside world. Depending on the number of cards the bandwidth is 400 or 800 MB/s for this type 3 I/O node. The type 2 I/O node is in fact a dedicated TCP/IP connection as needed for the control workstation as required by the system.

Measured Performances:
An A12 was said to be sold by the end of 1996 although Avalon was not willing to reaveal its customer, nor were any performance figures available.

3.4.2 The C-DAC PARAM OpenFrame system.

Machine type: RISC-based distributed-memory multi-processor.
Models: PARAM OpenFrame 9000 system.
Operating system: PARAS micro kernel based Unix (compatible with Sun's Solaris).
Connection structure: Multistage variable (see remarks).
Compilers: Fortran 77, Fortran 90, HPF, ANSI C, C++.
Vendors information Web page: `www.soft.net/cdac/html/newprod.html#PARAM`
Year of introduction: 1996.

System parameters:

Model	OpenFrame 9000
Clock cycle	6 ns?
Theor. peak performance	
Per Proc. (64-bits)	300 Mflop/s ?
Maximal	300 Gflop/s ?
Memory/node	—
Memory/maximal	—
No. of processors	1–1024
Communication bandwidth	
Point-to-point	3.2 GB/s
Bisectional (maximal)	—

Remarks:
The OpenFrame 9000 system is the fourth generation of CDAC machines that is developed by CDAC, the Centre for Development of Advanced Computing, an institute in India that has as its mission to develop an manufacture "state-of-the-art open architecture supercomputers". This system is the second generation that is marketed abroad. In the predecessor, the PARAM 9000/SS SuperSPARC II processors were used. In the present model the UltraSPARC is employed. Also, the maximum possible number of processors has been increased from 200 to 1024. The OpenFrame 9000 uses Sun's UltraSPARC processors but it is not clear from the scarce documentation at what clock cycle(s) they are running. Assuming the same basic processors as Sun employs in most of its Enterprise servers, the estimates as given in the parameter list above seem the most probable. The documentation also does not reveal any details of the type of the interconnection network. Only a point-to-point communication bandwidth is quoted.

The amount of available software shows that the PARAM 9000/SS is not a first-generation system. Apart from Fortran 77, Fortran 90, HPF, and C++ are available and the CORE, MPI, and PVM message passing interfaces are available. There is a parallel debugger, a proprietary performance evaluation tool called AIDE, while TOTALVIEW can be delivered on request.
In addition, a library of parallel routines, PARUL, is available. This library contains PVM versions of dense linear algebra routines, eigenvalue routines, and FFTs.

Measured Performances:
No measured performances of the PARAM OpenFrame 9000 are available at this moment.

3.4.3 The Cray Research Inc. T3E.

Machine type: RISC-based distributed-memory multi-processor.
Models: T3E, T3E-900, T3E-1200.
Operating system: UNICOS/mk (micro kernel-based Unix).
Connection structure: 3-D Torus.
Compilers: Fortran 77, Fortran 90, HPF, ANSI C, C++.
Vendors information Web page: `www.cray.com/products/systems/crayt3e/`
Year of introduction: T3E, T3E-900: 1996, T3E-1200: 1997.

System parameters:

Model	T3E	T3E-900	T3E-1200
Clock cycle	3.3 ns	2.2 ns	1.67 ns
Theor. peak performance			
Per Proc. (64-bits)	600 Mflop/s	900 Mflop/s	1200 Mflop/s
Maximal	1229 Gflop/s	1843 Gflop/s	2458 Gflop/s
Memory/node	$\leq$ 2 GB	$\leq$ 2 GB	$\leq$ 2 GB
Memory/maximal	$\leq$ 4 TB	$\leq$ 4 TB	$\leq$ 4 TB
No. of processors	16–2048	6–2048	6–2048
Communication bandwidth			
Point-to-point	300 MB/s	300 MB/s	300 MB/s

Remarks:
The T3E is the second generation of DM-MIMD systems from CRI. Lexically, it follows in name after its predecessor T3D which name referred to its connection structure: a 3-D torus. In this respect it has still the same interconnection structure as the T3D. In many other respects, however, there are quite some differences. A first and important difference is that no front-end system is required anymore (although it is still possible to connect to a Cray T90). The systems up to 128 processors are air-cooled. The larger ones, from 256-2,048 processors, are liquid cooled.
The T3E uses the DEC Alpha 21164 RISC processor for the T3E and the 21164A processor for the T3E-900 for its computational tasks just like the Avalon A12 (see 3.4.1). Recently, a T3E-1200 is introduced that uses 21164A processors at a clock rate of only 1.67 ns but that identical in any other aspect to the T3E-900. Cray stresses, that the processors are encapsulated in such a way that they can be exchanged easily for any other (faster) processor as soon as this would be available without affecting the macro-architecture of the system.
Each node in the system contains one processing element (PE) which in turn contains a CPU, memory, and a communication engine that takes care of communication between PEs. The bandwidth between nodes is quite high: 300 MB/s. Like the T3D, the T3E has hardware support for fast synchronisation. E.g., barrier synchronisation takes only one cycle per check.
In the microarchitecture most changes have taken place with the transition from the T3D to the T3E. Firstly, there is only one CPU per node instead of two, which removes a source of asymmetry between processors. Secondly, the new node processor has a 96 KB 3-way set-associative secondary cache which may relieve some of the problems of data fetching that were present in the T3D where only a primary cache was present. Third, the Block Transfer Engine has been replaced by a set of E-registers and streaming registers that are much more flexible and that remove some odd restrictions on the size of shared arrays and the number

processes when using Cray-specific PVM. An interesting additional feature is the availability of 32 contexts per processor which opens the door for multiprocessing.
In the T3D all I/O had to be handled by the front-end, a system at least from the Cray Y-MP/E class. In the T3E distributed I/O is present. For every 8 PEs an I/O channel can be configured in the air-cooled systems and 1 I/O channel per 16 nodes in the liquid-cooled systems. The maximum bandwidth for a channel is about 1 GB/s, the actual speed will be in the order of 700 MB/s.
The T3E supports various programming models. Apart from PVM3 and MPI for message passing and HPF for data distribution, a Cray proprietary one-sided communication library, the so-called `shmem` library can be employed for message passing. In addition, the BSP library (see [6]), also a one-sided message passing library is available. The `shmem` library is implemented close to the hardware and shows very low latency of only 1.6 μs.

Measured Performances:
In [2] a speed of 670 Gflop/s is reported for the solution of a dense linear system of order 128832 on a T3E-900 with 1320 processors. The efficiency for such an exercise is 56%.

3.4.4 The Fujitsu AP3000.

Machine type: RISC-based distributed-memory multi-processor.
Models: AP3000.
Operating system: Cell OS (transparent to the user) and Solaris (Sun's Unix variant) on the front-end system.
Connection structure: 2-D torus.
Compilers: Parallel Fortran/AP, Fortran 90, HPF, C, C++.
Vendors information Web page:
`www.fujitsu.co.jp/hypertext/Products/Info_process/hpc/ap3000-e/index.html`
Year of introduction: 1996.

System parameters:

Model	AP3000
Clock cycle	3.3 ns
Theor. peak performance	
Per Proc. (64-bits)	600 Mflop/s
Maximal	614 Gflop/s
Memory/node	$\leq$ 2 GB
Memory/maximal	$\leq$ 2 TB
No. of processors	4–1024
Communication bandwidth	
Point-to-point	200 MB/s

Remarks:
The AP3000 is the sucessor of the earlier AP1000 system. Although the name could suggest otherwise, few characteristics of the AP1000 have been retained except that Sun SPARC processors are used in the nodes. No front-end processor is required anymore as in the former system.
Also the communication network has been simplified considerably with respect to that in the earlier model: where three different networks were present in the AP1000 (see [8]), in the AP3000 the nodes are connected in a 2-D torus structure with a bi-directional bandwidth of 200 MB/s. The maximum amount of memory is huge: a full 1024 node system can accomodate 2 TB.
Another difference with the AP1000 system is that the fastest nodes (the (U300 nodes described here) can have either 1 or 2 CPUs as opposed to only one CPU in the AP1000. The two CPUs share the on-board memory.
The available software for the AP3000 is extensive: Parallel Fortran/AP is a Fortran 77 with extensions that offers a shared-memory-like programming model for the system. In addition, HPF is available and the machine can also be used with a message passing model as customised MPI/AP and PVM/AP are offered. As sequential languages to be used with the message passing libraries Fortran 90, C and C++ are available.

Measured Performances:
The system has been announced in March 1996 but as yet no performance figures are known.

3.4.5 The Fujitsu VPP700 series.

Machine type: Distributed-memory vector multi-processor.
Models: VX, VPP300, VPP700.
Operating system: UXP/V (a V5.4 based variant of Unix).
Connection structure: Full distributed crossbar.
Compilers: Fortran 90/VP (Fortran 90 Vector compiler), Fortran 90/VPP (Fortran 90 Vector Parallel compiler), C/VP (C Vector compiler), C, C++.
Vendors information Web page:
`www.fujitsu.co.jp/hypertext/Products/Info_process/hpc/vx-e/index-e.html`
Year of introduction: VX, VPP300: 1995, VPP700: 1996.

System parameters:

Model	VX	VPP300	VPP700
Clock cycle	7 ns	7 ns	7 ns
Theor. peak performance			
Per Proc. (64-bits)	2.28 Gflop/s	2.28 Gflop/s	2.28 Gflop/s
Maximal	9.2 Gflop/s	36.5 Gflop/s	583.6 Gflop/s
Memory/node	$\leq$ 2 GB	$\leq$ 2 GB	$\leq$ 2 GB
Memory/maximal	$\leq$ 8 GB	$\leq$ 32 GB	$\leq$ 512 GB
No. of processors	1–4	1–16	8–256
Memory bandw./proc.	18.2 GB/s	18.2 GB/s	18.2 GB/s
Communication bandwidth			
Point-to-point	570 MB/s	570 MB/s	570 MB/s

Remarks:
The VPP300 is a sucessor to the earlier VPP500. It is a much cheaper CMOS implementation of its predecessor with some important differences. First, no VPX200 front-end system is required anymore. Second, the crossbar that is used to connect the vector nodes is distributed. Therefore, the cost of a system is scalable: one does not need to buy a complete enclosure with the full crossbar for only a few nodes. The VX series is in fact a smaller version of the VPP300 with a maximum of 4 processors. Both the VX machines and the larger VPP300 systems are air-cooled.
The architecture of the VPP300 nodes is almost identical to that of the VPP500: Each node, called a Processing Element (PE) in the system is a powerful (2.28 Gflop/s peak speed with a 7 ns clock) vector processor in its own right. The vector processor is complemented by a RISC scalar processor with a peak speed of 285 Mflop/s dependent. The scalar instruction format is 64 bits wide and may cause the execution of three operations in parallel. Each PE has a memory of up to 2 GB while a PE communicates with its fellow PEs at a point-to-point speed of 570 MB/s. This communication is cared for by separate Data Transfer Units (DTUs). To enhance the communication efficiency, the DTU has various transfer modes like contiguous, stride, sub array, and indirect access. Also translation of logical to physical PE-ids and from Logical in-PE address to real address are handled by the DTUs. When synchronisation is required each PE can set its corresponding bit in the SR. The value of the SR is broadcast to all PEs and synchronisation has occurred if the SR has all its bits set for the relevant PEs. This method is comparable to the use of synchronisation registers in shared-memory vector processors and much faster than synchronising via memory.
The VPP700 is a logical extension of the Fujitsu VPP300. While the processors in the latter machine are connected by a full crossbar, the maximum configuration of a VPP700 consists of 16 clusters of 16 processors connected by a level-2 crossbar. So, a fully configured VPP700 consists in fact of 16 full VPP300s. Because the diameter of the network is 2 (for the larger configurations) instead of 1 as in the VPP300, the communication time between processors will be slightly larger. At the moment this worst case increase is not exactly known to the author.
The Fortran compiler that comes with the VPP300/700 has extensions that enable data decomposition by

compiler directives. This evades in many cases restructuring of the code. The directives are different from those as defined in the High Performance Fortran Proposal but it should be easy to adapt them. Furthermore, it is possible do define parallel regions, barriers, etc., via directives, while there are several intrinsic functions to enquire about the number of processors and to execute `POST/WAIT` commands. Furthermore, also a message passing programming style is possible by using the PVM or PARMACS communication libraries that are available. Of course the software for the VPP700 and the VPP300 is exactly the same and the systems can run each others executables.

Measured Performances:
In [2] results for the VX, the VPP300, and the VPP700 are given. The speed for solving dense linear system of sizes 28,800 59,200, and 111,360 was 8.6, 34.1, and 213 Gflop/s on a 4 proc. VX, a 16 proc. VPP300, and a 116 proc. VPP700, respectively.

3.4.6 The Hitachi SR2201.

Machine type: RISC-based distributed-memory multi-processor.
Models: SR2201.
Operating system: HI-UX/MPP (Micro kernel Mach 3.0).
Connection structure: Hyper crossbar (see remarks).
Compilers: Fortran 77, Fortran 90, Parallel Fortran, HPF, C, C++.
Vendors information Web page: `www.hitachi.co.jp/Prod/comp/hpc/eng/sr1.html`
Year of introduction: 1996.

System parameters:

Model	SR2201
Clock cycle	6.7 ns
Theor. peak performance	
Per Proc. (64-bits)	300 Mflop/s
Maximal	307 Gflop/s
Memory/node	$\leq$ 256 MB
Memory/maximal	$\leq$ 256 GB
No. of processors	32–1024
Communication bandwidth	
Point-to-point	300 MB/s

Remarks:
The SR2201 is the second generation of distributed-memory parallel systems of Hitachi. The basic node processor is again an Hitachi implementation of the PA-RISC architecture of HP running at a clock cycle of 6.7 ns. However, in contrast with its predecessor, the SR2001, in the SR2201 the node processors are somewhat modified to allow for "pseudo vector processing" (both hardware and instructions). This means that for operations on long vectors one does not have to care about the detrimental effects of cache misses that often ruin the performance of RISC processors unless code is carefully blocked and unrolled. Experiments have shown that this idea seems to work quite well. The system supports distributed I/O with a possibility to connect disks to every node.
As in the earlier SR2001, the connection structure is a hyper (3-D) crossbar which connects all nodes directly at high speed (300 MB/s point-to-point). In February 1996 two 1024-node systems were installed at the Universities of Tokyo and Tsukuba respectively. The latter has been extended to the (non-commercial) CP-PACS system that has 2048 processors but less memory (128 MB) per processor.
Like in some other systems as the Cray T3E (3.4.3), Meiko CS-2 (3.4.9), and the NEC Cenju-3 (3.4.11), one is able to directly access the memories of remote processors. Together with the very fast hardware-based barrier synchronisation this should allow for writing distributed programs with very low parallelisation overhead.
The following software products are supported in addition to those already mentioned above: PVM, MPI, PARMACS, Linda, and FORGE90. In addition several numerical libraries like NAG and IMSL are offered.

Measured Performances:
In [2] a speed of 232.3 Gflop/s is reported to solve a dense linear system of size 138,240 on a 1024 processor system. Also some results of class A NAS parallel benchmarks show that the SR2201 runs at about 5.7 Gflop/s on the MG benchmark, at 5.9 Gflop/s on the BT benchmark, and at 5.4 Gflop/s on the SP benchmark, all on 256 processors ([11]).

3.4.7 The HP/Convex Exemplar SPP-2000.

Machine type: RISC-based distributed-memory multi-processor.
Models: SPP-2000K, SPP-2000S, SPP-2000X, Exemplar V-Class.
Operating system: 2000K/S/X: HI-UX/MPP (Micro kernel Mach 3.0), Exemplar V: HP-UX (HP's usual Unix flavour).
Connection structure: Ring.
Compilers: Fortran 77, Fortran 90, Parallel Fortran, HPF, C, C++.
Vendors information Web page: 2000K/S/X: `www.hp.com/wsg/products/servers/servpfo.html`, Exemplar V-Class: `www.hp.com/gsy/products/vclass.html`

Year of introduction: 2000K/S/X: 1996, Exemplar V-Class: 1997.

System parameters:

Model	SPP-2000K	SPP-2000S	SPP-2000X	Exemplar V
Clock cycle	5.55 ns	5.55 ns	5.55 ns	5 ns
Theor. peak performance				
Per Proc. (64-bits)	720 Mflop/s	720 Mflop/s	720 Mflop/s	800 Mflop/s
Maximal	2.9 Gflop/s	11.5 Gflop/s	46.1 Gflop/s	12.8 Gflop/s
Memory/node	≤1 GB	≤1 GB	≤1 GB	≤1 GB
Memory/maximal	≤4 GB	≤16 GB	≤64 GB	≤16 GB
No. of processors	1–4	4–16	16–64	1–16
Communication bandwidth				
Aggregate (see remarks)	3.84 GB/s	15.4 GB/s	15.4/3.84 GB/s	15.4 GB/s

Remarks:
The SPP-2000 systems form the family of successors of the SPP-1200/1600. There are significant differences with respect to the preceding SPP-1200 generation. The SPP-2000K and S are shared-memory machines. Of these the SPP-2000K connects its maximally 4 PA-RISC 8000 processors by a bus while the S-class system has up to 16 of these processors which are connected by a crossbar. Each processor has a peak performance of 720 Mflop/s and because the processors feature out-of-order execution of instructions it may be expected that memory latency effects can be evaded or diminished in a good many cases. This should make the impact of cache misses much less severe. Data and instruction caches are large (1 MB both) which also will help in minimising cache misses.
One SPP-2000S can be viewed as the successor of a hypernode in the earlier SPP-1200/SPP-1600 systems. As such the number of processors within a hypernode has doubled. Also the amount of memory per system has increased 8-fold from 8×256 MB to 16×1 GB. The internal aggregate bandwidth is still 15.36 GB/s for the 2000S and 3.84 GB/s for the 2000K. I/O can be done at an aggregate rate of 960 MB/s.
As in the earlier SPP-1200/1600 systems, the hypernodes are connected by uni-directional SCI rings with an aggregate bandwidth of 3.84 GB/s. This makes the SPP-2000X a NUMA machine when operates in a shared-memory fashion. Note that, although the speed of computation has increased significantly with respect to the SPP-1200/1600 (more than a factor of 2), the interconection bandwidth did not grow.
The introduction of the new Exemplar V-class system is a bit confusing. Technically, there very little difference with the SPP-2000S system, except that the basic processor is a 5 ns clock PA-RISC 8200 one. The other difference is the operating system: this is HP-UX, the same Unix flavour that is also used on all other HP Unix platforms except the SPP-2000K/S/X systems. This enables the V-class systems to run all applications that are available for the HP systems, including commercial ones. From the presentation it seems that HP tries to position this system in the commercial market and less so for technical/scientific use.

The Exemplar programming environment as was available for the SPP-1200/1600 carries over to the SPP-2000K/S/X without changes. This environment includes a message passing programming model (PVM and MPI) and a virtual shared-memory model which allows the user to have a shared-memory view of the system. Of course the shared-memory model is not surprising for a symmetrical multiprocessor machine like the SPP-2000S but it is still valid in the SPP-2000X systems which effectively clusters four SPP-2000S systems. The (virtual) shared-memory model supported by HP/Convex has significant syntactical differences with models supported by other vendors. However, HP has agreed to support the OpenMP initiative [12] to standardise the shared-memory model across vendors.

Measured Performances:
In [2] a speed of 27.6 Gflop/s is reported for a 64 proc. system when solving a 29,956-order dense linear system, an efficiency of 60% on this problem. For the EuroBen `mod2a` matrix-vector multiplication benchmark a speed of 417 Mflop/s is found on 16 processors. This is however for straight Fortran 77 code with PVM and without the use of library routines. On a 16-processor V-class machine a speed of 9.2 Gflop/s was measured in solving a 14944-order dense linear system.

3.4.8 The IBM 9076 SP2.

Machine type: RISC-based distributed-memory multi-processor.
Models: IBM 9076 SP2.
Operating system: AIX (IBM's Unix variant).
Connection structure: Ω-switch.
Compilers: XL Fortran (Fortran 90), HPF, XL C, C++.
Vendors information Web page: `www.rs6000.ibm.com/hardware/largescale/index.html`.
Year of introduction: 1997.

System parameters:

Model	9076 SP2
Clock cycle	6.25 ns
Theor. peak performance	
Per Proc. (64-bits)	640 Mflop/s
Maximal	variable (see remarks)
Memory/node	≤1/2 GB (see remarks)
Memory/maximal	≤1 TB
No. of processors	8–512
Communication bandwidth	
Point-to-point	150 MB/s

Remarks:
Presently, three types of node processors are available for the SP2. The P2SC thin nodes, P2SC wide nodes and the 604e "High node". The latter is in fact a cluster of up to 8 604e processor connected to a common memory in an SMP fashion. The processors have a clock cycle of 5 ns, however, they can only deliver 2 floating-point results/cycle at maximum, while the P2SC nodes can deliver 4 floating-point results per clock cycle. as the fastest of the P2SC nodes has a clock cycle of 6.25 ns, it has a peak performance of 640 Mflop/s while a single 604e processor can attain 400 Mflop/s at maximum. It seems that the 604e processor-based systems are more targetted to the commercial market. The P2SC-based systems are primarily meant for the technical/scientific market. In the parameter list above we included the presently fastest P2SC processor.
The SP2 configurations are housed in columns that each can contain 8–16 processor nodes. This depends on the type of node employed: a thin nodes occupies half of the space of a wide node. Although the processors in these nodes are basically the same there are some differences. At the time of writing no 6.25 ns clock wide nodes were available yet. The fastest in this class feature a clock cycle of 7.4 ns giving a peak speed of 540 Mflop/s. Also thin nodes with a clock cycle of 8.3 ns are still around, having a peak speed of 480 Mflop/s.
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 1 GB. IBM envisions the wide node more or less as server for a column and recommends configurations of one

wide node packaged with 14 thin nodes per column (although this may differ with the needs of the user). The SP2 is accessed through a front-end control workstation that also monitors system failures. Failing nodes can be taken off line and exchanged without interrupting service. In addition, fileservers can be connected to the system while every node can have up to 2 GB. This can greatly speed up applications with significant I/O requirements.
Although we mentioned only the highest speed option for the communication, the high-performance switch, there is a wide range of other options that could be chosen instead: Ethernet, Token Ring, FDDI, etc., are all possible. The new version of the high-performance switch has greatly improved over the former model with a bandwidth of 150 MB/s point-to-point instead of the former 40 MB/s. Unfortunately, the online (semi)technical information of IBM is not very helpful in providing more detailed information with regard to the other properties of the switch so, for instance, a bisection bandwidth cannot be provided at this point. The high-performance 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 MPI. Also High Performance Fortran is supported, both a proprietary version and a compiler from the Portland Group. 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. Also the MPI implementation, MPI-F, is optimised for the SP2 systems.

Measured Performances:
In [2] a performance of 151.8 Gflop/s is mentioned for a 460 processor 8.33 ns P2SC thin node machine in solving a 61000-order dense linear system. A 128-processor system using 6.25 ns processors solved a similar problem of order 39000 at a speed of 57.2 Gflop/s which amounts to an effiency of 70%.

3.4.9 The Meiko Computing Surface 2.

Machine type: Distributed-memory multi-vectorprocessor.
Models: Computing Surface 2.
Operating system: Internal OS transparent to the user, Solaris (Sun's Unix variant) on the front-end system.
Connection structure: Multistage crossbar.
Compilers: Extended Fortran 77, ANSI C.
Vendors information Web page:
`www.meiko.com/info/TechnicalDescription/TechnicalDescription.html`.
Year of introduction: 1994.

System parameters:

Model	Computing Surface 2
Clock cycle	20 ns
Theor. peak performance	
Per Proc. (64-bits)	200, 40 Mflop/s
Maximal	204.8 Gflop/s
Memory/node	32–128, 32–512 MB
Memory/maximal	≤128 GB
No. of processors	8–1024
Communication bandwidth	
Point-to-point (bi-directional)	50 MB/s

Remarks:
The CS-2 features 8-1,024 processor elements (PEs) which can be either scalar or vector nodes. Apart from a separate communications module, these PEs contain either a SuperSparc or a SuperSparc + 2 μVP vectorprocessors. The speed of a scalar PE is estimated to be 40 Mflop/s (at a 20 ns clock) and 200 Mflop/s for the vector PEs for 64-bit precision. The μVP modules are manufactured by Fujitsu. The speed at 32-bit precision is doubled with respect to 64-bit operation and, unlike the early 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 compiler a offers HPF data distribution directives. Furthermore, some optimised standard linear algebra and FFT routines are offered via a proprietary numerical library.
In the USA the machine will be marketed by Meiko. In 1996 Meiko has merged with Alenia, the same firm that also markets the Alenia Quadrics. Although the new marketing policy has not been made clear, it may assumed that Alenia will market the system in Europe and the rest of the world (see `www.quadrics.com`).

Measured Performances:
In [2] a speed of 5.0 Gflop/s on a 64 processor CS-2 is reported for the solution of an order 18688 dense linear system. From the NAS parallel benchmarks [11] some results on a 128 processor machine are given for class B problems: EP took 21.16 seconds while 6.52 seconds was measured for the MG problem.

3.4.10 The nCUBE 2S.

Machine type: Distributed-memory multi-processor.
Models: nCUBE 2S.
Operating system: Internal OS transparent to the user, Solaris (Sun's Unix variant) on the front-end system.
Connection structure: Hypercube.
Compilers: Fortran 77, ANSI C.
Vendors information Web page: `www.ncube.com/products/`.
Year of introduction: 1993.

System parameters:

Model	nCUBE 2S
Clock cycle	40 ns
Theor. peak performance	
Per Proc. (64-bits)	3.0 Mflop/s
Maximal	19.7 Gflop/s
Memory/node	≤64 MB
Memory/maximal	≤256 GB
No. of processors	8–8192
Communication bandwidth	
Point-to-point	2.75 MB/s

Remarks:
The nCUBE 2S is, apart from the SGI Origin 2000, 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 20speed 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 accomodate 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.
Some years ago nCUBE had plans to develop a successor to the nCUBE 2S, the nCUBE 3 but no firm date the new product is given. Details of the nCUBE 3 processor are given at `www.ncube.com/products/cpu.html` Presently, nCUBE is more directing its interest to the multi-media server market, where it offers several products. Scientific and technical work does not seem to have a high priority anymore.

Measured Performances:
For a 1024-processor nCUBE 2 a speed of 1.9 Gflop/s was observed for a 21,376-order linear system in 64-bit precision ([2]), about 77% of the quoted peak performance. No other independent performance figures are available.

3.4.11 The NEC Cenju-3.

Machine type: RISC-based distributed-memory multi-processor.
Models: Cenju-3S, Cenju-3.
Operating system: Cenju-3 OS.
Connection structure: Multistage crossbar.
Compilers: Fortran 77, HPF (subset), ANSI C.
Vendors information Web page:
`www.ccrl-nece.technopark.gmd.de/Cenju.html`.
Year of introduction: 1994.

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	533 Mflop/s	12.8 Gflop/s
Memory/node	≤64 MB	≤64 MB
Memory/maximal	≤1 GB	≤16 GB
No. of processors	8–16	8–256
Communication bandwidth		
Point-to-point	40 MB/s	40 MB/s

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 MIPS R4400 RISC processor. All processors have, apart from their on-chip primary cache, a secondary cache of 1 MB to mitigate the problems that arise in the high data usage of the CPU.
The interconnection type used in the Cenju is a multistage crossbar build from 4×4 modules that are pipelined. So, in a full configuration the maximal number of levels in the crossbar to be traversed is four. The peak transfer rate of the crossbar is quoted as 40 MB/s irrespective of the data placement.
The system needs a front-end processor of the EWS4800 type (functionally equivalent to Silicon Graphics workstations). The I/O requirements have to be fulfilled by the front-end system as the Cenju does not have local (distributed) I/O capabilities.
There is some software support that should make the programmer's life somewhat easier. The library PARALIB/CJ contains proprietary functions for forking processes, barrier synchronisation, remote procedure calls, and block transfer of data. Like on the Cray T3E (3.4.3), the Hitachi SR2201 (3.4.6), and on the Meiko CS-2 (3.4.9) the programmer has the possibility to write/read directly to/from non-local memories which avoids much message passing overhead.

Measured Performances:
Delivery of the systems have started in the second quarter of 1994 but no systematic performance figures are ever published for the Cenju-3. A comparative performance study has been done for various sorting algorithms on the Cenju-3 ([13])and the SP2 and a performance evaluation of MPP LS-DYNA3D ([19]).

3.4.12 The Parsytec CC series.

Machine type: RISC-based distributed-memory multi-processor.
Models: Parsytec CCi, Parsytec CCe, Parsytec Power Mouse.
Operating system: Unix on host processor, Parix (CC OS, transparent to the user).
Connection structure: . . . , Grid (on Power Mouse).
Compilers: Fortran 77, Fortran 90, ANSI C, C++.
Vendors information Web page: `www.parsytec.com`.
Year of introduction: 1996.

System parameters:

Model	CCi	CCe	Power Mouse
Clock cycle	— ns	4.4 ns	5 ns
Theor. peak performance			
Per Proc. (64-bits)	— Mflop/s	450 Mflop/s	400 Mflop/s
Maximal	— Gflop/s	— Gflop/s	— Gflop/s
Memory/node	—	64 MB	64 MB
Memory/maximal	— GB	— GB	— GB
No. of processors	—	—	—
Communication bandwidth			
Point-to-point	10 MB/s	75 MB/s	10 MB/s

Remarks:
The standard technical information as provided by Parsytec via its Web pages is too incomplete for a useful description. It is sufficient to infer that the systems exist and are marketed but not much more. From the CCi models only is made known that they are using Intel processors but no other details are available.
A CCe with 40 processors is located at the University of Amsterdam. In this system a CLOS-network is used to connect the nodes, but it is not sure whether this is a standard configuration. Parsytec also works as a system integrator, so other interconnections/configurations might be possible. The Parsytec CCe series is equipped with ATM HS links. The link speed is up to about 75 MB/s.
The Power Mouse is built from 4-node bricks that communicate over 4 serial links/node at 10 MB/s per link (bi-directional) using T425 transputers. The documentation states the scalability as being unlimited. In view of this discrepancy between the computational speed and the communication speed this might, however, not be of much practical value. A Sun SPARC front-end is required to communicate with the systems. I/O also has be done via the front-end system.

Measured Performances:
No independently confirmed performance figures from either system are available at this moment.

3.4.13 The Silicon Graphics Origin series.

Machine type: RISC-based distributed-memory multi-processor.
Models: Origin 200, Origin 2000.
Operating system: IRIX (SGI's Unix variant).
Connection structure: Crossbar, hypercube (see remarks).
Compilers: Fortran 77, Fortran 90, HPF, C, C++, Pascal.
Vendors information Web page: `www.sgi.com/Products/hardware/servers/index.html`.
Year of introduction: 1996.

System parameters:

Model	Origin 200	Origin 2000
Clock cycle	5.4 ns	5.12 ns
Theor. peak performance		
Per Proc. (64-bits)	370 Mflop/s	390 Mflop/s
Maximal	1.48 Gflop/s	49.9 Gflop/s
Memory	$\leq$4 GB	$\leq$256 GB
No. of processors	1–4	2–128
Communication bandwidth		
Point-to-point	780 MB/s	780 MB/s
Aggregate peak	3.1 GB/s	99.8 GB/s
Bisectional	1.6 GB/s	82 GB/s

Remarks:
The Origin 2000 is the newest high-end parallel server marketed by SGI. The basic processor is the MIPS R10000. A maximum of 128 processors can be configured in the system. The interconnection is somewhat hybrid: 4 CPUs on two node cards can communicate directly with the memory partitions of each other via the hub, a 4-ported non-blocking crossbar. Hubs can be coupled to other hubs in a hypercube fashion. ¡P¿
The structure of the machine makes it somewhat difficult to classify: SGI prefers to call it a shared-memory non-uniform memory architecture system. The memory is physically distributed over the node boards but the system has one system image. Because of the structure of the system, the bisectional bandwidth of the system remains constant from 4 processors on: 82 GB/s. This is a large improvement over the earlier PowerChallenge systems which possessed a 1.2 GB/s bus.
The Origin 200 is a smaller configuration, using the same crossbar as the Origin 2000 but without the need for the hypercube connections used in the latter. Effectively, it is a SMP system because of the uniform access of the memory modules. Therefore, also the bisectional bandwidth is identical to the point-to-point bandwidth: 1.6 GB/s.
Parallelisation is done either automatically by the (Fortran or C) compiler or explicitly by the user, mainly through the use of directives. All synchronisation, etc., has to be done via memory. This may cause potentially a fairly large parallelisation overhead. Also a message passing model is allowed on the Origin using the optimised SGI versions of PVM and MPI. Programs implemented in this way will possibly run very efficiently on the system.
A nice feature of the new system is that it may migrate processes to nodes that should satisfy the data requests of these processes. So, the overhead involved in transferring data across the machine are minimised in this way. The technique is reminiscent of the late Kendall Square Systems although in these systems the data were moved to the active process. SGI claims that the time for non-local memory references is on average about 3 times longer than for local memory references.

Measured Performances:
In [2] a speed of 10.9 Gflop/s out of 12.5 was measured on a system with 32 processors, an efficiency of 87%. Furthermore, for an MPI version of a matrix-vector multiplication from the EuroBen benchmark ([16]) a speed of 7.2 Gflop/s was attained, an efficiency of 60%.

4 Systems disappeared from the list

As already stated in the introduction the list of systems is not complete. On one hand this is 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 wass very similar but not equivalent to the former C-DAC system and there are numerous other examples). On the other hand there many systems that are still in operation around the world, often in considerable quantities that for other reasons are excluded. The most important reasons are:

- The system is not marketed anymore. This is generally for one of two reasons:
  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 the first time we include the year of introduction and the year of exit of the systems when they were readily accessible. These timespans could give a hint of the dynamics that governs this very dynamical branch of the the computer industry.

Machine: The Alex AVX 2.
Year of introduction: 1992.
Year of exit: 1997.
Type: RISC-based distributed-memory multi-processor.
Theoretical Peak performance : 3.84 Gflop/s.
Reason for disappearance: System is obsolete, there is no new system planned.

Machine: Alliant FX/2800.
Year of introduction: 1989.
Year of exit: 1992.
Type: Shared-memory vector-parallel, max. 28 processors.
Theoretical Peak performance: 1120 Mflop/s.
Reason for disappearance: Manufacturer out of business.

Machine: The AxilSCC.
Year of introduction: 1996.
Year of exit: 1997.
Type: RISC-based distributed-memory system, max. 512 processors.

Theoretical Peak performance : 76.8 Gflop/s.
Reason for disappearance: System is not marketed anymore by Axil.

Machine: BBN TC2000.
Year of introduction: —.
Year of exit: 1990.
Type: Virtual shared-memory parallel, max. 512 processors.
Theoretical Peak performance: 1 Gflop/s.
Reason for disappearance: Manufacturer has discontinued marketing parallel computer systems.

Machine: Cambridge Parallel Processing DAP Gamma.
Year of introduction: 1986.
Year of exit: 1995.
Type: Distributed-memory processor array system, max. 4096 processors.
Theoretical Peak performance: 1.6 Gflop/s (32-bit).
Reason for disappearance: replaced by newer Gamma II Plus series (3.2.2).

Machine: C-DAC PARAM 9000/SS.
Year of introduction: 1995.
Year of exit: 1997.
Type: Distributed-memory RISC based system, max. 200 processors.
Theoretical Peak performance: 12.0 Gflop/s.
Reason for disappearance: replaced by newer PARAM OpenFrame series (3.4.2).

Machine: Convex SPP-1000/1200/1600.
Year of introduction: 1995 (SPP-1000).
Year of exit: 1996 (SPP-1600).
Type: Distributed-memory RISC based system, max. 128 processors.
Theoretical Peak performance: 25.6 Gflop/s
Reason for disappearance: replaced by newer SPP-2000 series (3.4.7).

Machine: Cray Computer Corporation Cray-2.
Year of introduction: 1982.
Year of exit: 1992.
Type: Shared-memory vector-parallel, max. 4 processors.
Theoretical Peak performance: 1.95 Gflop/s.
Reason for disappearance: Manufacturer out of business.

Machine: Cray Computer Corporation Cray-3.
Year of introduction: 1993.
Year of exit: 1996.
Type: Shared-memory vector-parallel, max. 16 processors.
Theoretical Peak performance: 16 Gflop/s.
Reason for disappearance: Manufacturer out of business.

Machine: Cray Research Inc. APP.
Year of introduction: 1993.
Year of exit: 1995.
Type: Shared-memory RISC based system, max. 84 processors.
Theoretical Peak performance: 6.7 Gflop/s.
Reason for disappearance: Product line discontinued, gap expected to be filled by Cray J90 (3.3.1).

Machine: Cray T3D.
Year of introduction: 1994.

Year of exit: 1996.
Type: Distributed-memory RISC based system, max. 2048 processors.
Theoretical Peak performance: 307 Gflop/s.
Reason for disappearance: replaced by newer Cray T3E (3.4.3).

Machine: Cray Research Inc. Cray Y-MP, Cray Y-MP M90.
Year of introduction: 1989 (Cray Y-MP).
Year of exit: 1994 (Cray Y-MP M90).
Type: Shared-memory vector-parallel, max. 8 processors.
Theoretical Peak performance: 2.6 Gflop/s.
Reason for disappearance: replaced by newer T90 (3.3.1).

Machine: Cray Y-MP C90.
Year of introduction: 1994.
Year of exit: 1996.
Type: Shared-memory vector-parallel, max. 16 processors.
Theoretical Peak performance: 16 Gflop/s.
Reason for disappearance: replaced by newer T90 (3.3.1).

Machine: Digital Equipment Corp. Alpha farm.
Year of introduction: —.
Year of exit: 1994.
Type: Distributed-memory RISC based system, max. 4 processors.
Theoretical Peak performance: 0.8 Gflop/s.
Reason for disappearance: replaced by newer AlphaServer clusters (3.3.2).

Machine: Fujitsu AP1000.
Year of introduction: 1991.
Year of exit: 1996.
Type: Distributed memory RISC based system, max. 1024 processors.
Theoretical Peak performance: 5 Gflop/s.
Reason for disappearance: replaced by the AP3000 systems (3.4.4).

Machine: Fujitsu VPP500 series.
Year of introduction: 1993.
Year of exit: 1995.
Type: Distributed-memory multi-processor vectorprocessors, max. 222 processors.
Theoretical Peak performance: 355 Gflop/s.
Reason for disappearance: replaced by the VPP300/700 series (3.4.5).

Machine: Fujitsu VPX200 series.
Year of introduction: —.
Year of exit: 1995.
Type: Single-processor vectorprocessors.
Theoretical Peak performance: 5 Gflop/s.
Reason for disappearance: replaced by the VPP300/700 series (3.4.5).

Machine: Hitachi SR2001 series.
Year of introduction: 1994.
Year of exit: 1996.
Type: Distributed-memory RISC based system, max. 128 processors.
Theoretical Peak performance: 23 Gflop/s.
Reason for disappearance: Replaced by the newer SR2201 (3.4.6).

Machine: HP/Convex C4600 series.
Year of introduction: 1994.
Year of exit: 1997.
Type: Shared-memory vector-parallel, max. 4 processors.
Theoretical Peak performance: 3.24 Gflop/s.
Reason for disappearance: HP does not market the vector product line anymore.

Machine: IBM ES/9000 series.
Year of introduction: 1991.
Year of exit: 1994.
Type: Shared-memory vector-parallel system, max. 6 processors.
Theoretical Peak performance: 2.67 Gflop/s.
Reason for disappearance: IBM does not pursue high-performance computing by this product line anymore.

Machine: IBM Power/4.
Year of introduction: 1992.
Year of exit: 1994.
Type: Shared-memory RISC based system, max. 4 processors.
Theoretical Peak performance: 336 Mflop/s.
Reason for disappearance: Product line discontinued, gap expected to be filled by the newer SP2 (3.4.8).

Machine: IBM SP1 series.
Year of introduction: 1992.
Year of exit: 1994.
Type: Distributed-memory RISC based system, max. 64 processors.
Theoretical Peak performance: 8 Gflop/s.
Reason for disappearance: Replaced by the newer SP2 (3.4.8).

Machine: Intel iPSC/860.
Year of introduction: 1989.
Year of exit: 1992.
Type: Distributed-memory RISC based system, max. 128 processors.
Theoretical Peak performance: 7.7 Gflop/s.
Reason for disappearance: replaced by newer Intel Paragon XP (MP) series (but see below).

Machine: Intel Paragon XP.
Year of introduction: 1992.
Year of exit: 1996.
Type: Distributed-memory RISC based system, max. 4000 processors.
Theoretical Peak performance: 300 Gflop/s.
Reason for disappearance: Except for a non-commercial research system (the ASCI Option Red system at Sandia National Labs.) Intel is not in the business of high-performance computing anymore.

Machine: Kendall Square Research KSR2.
Year of introduction: 1992.
Year of exit: 1994.
Type: Virtually shared-memory parallel, max. 1088 processors.
Theoretical Peak performance: 400 Gflop/s.
Reason for disappearance: Kendall Square has terminated its business.

Machine: Kongsberg Informasjonskontroll SCALI.
Year of introduction: 1996.
Year of exit: 1997.

Type: Distributed-memory RISC based system, max. 512 processors.
Theoretical Peak performance: 76.8 Gflop/s.
Reason for disappearance: Kongsberg does not market the system anymore.

Machine: MasPar MP-1, MP-2.
Year of introduction: 1991 (MP-1).
Year of exit: 1996.
Type: Distributed-memory processor array system, max. 16384 processors.
Theoretical Peak performance: 2.4 Gflop/s (64-bit, MP-2).
Reason for disappearance: Systems are not marketed anymore.

Machine: Matsushita ADENART.
Year of introduction: 1991.
Year of exit: 1997.
Type: Distributed-memory RISC based system, 256 processors.
Theoretical Peak performance: 2.56 Gflop/s.
Reason for disappearance: Machine is obsolete and no new systems are developed in this line.

Machine: Meiko CS-1 series.
Year of introduction: 1989.
Year of exit: 1995.
Type: Distributed-memory RISC based system.
Theoretical Peak performance: 80 Mflop/s per processor.
Reason for disappearance: Replaced by the newer CS-2 (3.4.9).

Machine: nCUBE 3.
Year of introduction: — (see remarks at 3.4.10).
Year of exit: —
Type: Distributed-memory system, max. 10244 processors.
Theoretical Peak performance: 1 Tflop/s.
Reason for disappearance: Was announced several times but was never finished. Development has been abandoned but may be revived some time. The nCUBE 2S is the available product (3.4.10).

Machine: NEC SX-2.
Year of introduction: 1990.
Year of exit: 1993.
Type: Single-processor vector processors.
Theoretical Peak performance: 1.3 Gflop/s.
Reason for disappearance: replaced by newer SX-4 series (3.3.4).

Machine: NEC SX-3R.
Year of introduction: 1993.
Year of exit: 1996.
Type: Shared-memory multi-processor vector processors, max. 4 processors.
Theoretical Peak performance: 25.6 Gflop/s.
Reason for disappearance: replaced by newer SX-4 series (3.3.4).

Machine: Parsys SN9000 series.
Year of introduction: 1993.
Year of exit: 1995.
Type: Distributed-memory RISC based system, max. 2048.
Theoretical Peak performance: 51.2 Gflop/s.
Reason for disappearance: Replaced by the newer TA9000 (but see below).

Machine: Parsys TA9000 series.
Year of introduction: 1995.
Year of exit: 1996.
Type: Distributed-memory RISC based system, max. 512 processors.
Theoretical Peak performance: 119.3 Gflop/s.
Reason for disappearance: Parsys does not offer complete system anymore. Instead it sells node cards based on the TA9000 for embedded systems.

Machine: Parsytec GC/Power Plus.
Year of introduction: 1993.
Year of exit: 1996.
Type: Distributed-memory RISC based system.
Theoretical Peak performance: 266.6 Mflop/s per processor.
Reason for disappearance: System has been replaced by the Parsytec CC systems (3.4.12).

Machine: Siemens-Nixdorf VP2600 series.
Year of introduction: —.
Year of exit: 1995.
Type: Single-processor vectorprocessors.
Theoretical Peak performance: 5 Gflop/s.
Reason for disappearance: replaced by the VPP300/700 series (3.4.5).

Machine: Silicon Graphics PowerChallenge.
Year of introduction: 1994.
Year of exit: 1996.
Type: Shared-memory multi-processor, max. 36 processors.
Theoretical Peak performance: 14.4 Gflop/s.
Reason for disappearance: replaced by the SGI Origin 2000 (3.4.13).

Machine: Stern Computing Systems SSP.
Year of introduction: 1994.
Year of exit: 1996.
Type: Shared-memory multi-processor, max. 6 processors.
Theoretical Peak performance: 2 Gflop/s.
Reason for disappearance: Vendor terminated its business just before delivering first systems.

Machine: Thinking Machine Corporation CM-2(00).
Year of introduction: 1987.
Year of exit: 1991.
Type: SIMD parallel machine with hypercube structure, max. 64K processors.
Theoretical Peak performance: 31 Gflop/s.
Reason for disappearance: was replaced by the newer CM-5 (but see below).

Machine: Thinking Machine Corporation CM-5.
Year of introduction: 1991.
Year of exit: 1996.
Type: Distributed-memory RISC based system, max. 16K processors.
Theoretical Peak performance: 2 Tflop/s.
Reason for disappearance: Thinking Machine Corporation has stopped manufacturing hardware and hopes to keep alive as a software vendor.

5 Systems under development

Although we mainly want 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 SGI Origin 2000). We recognise the wishes of such vendors (it is generally wise not to stretch the expectation of potential customers too long) and will not disclose such information.
Below we discuss systems that may lead to commercial systems to be introduced on the market between somewhat more than half a year to a year from now. The commercial systems that result from it will sometimes deviate significantly from the original research models depending on the way the development is done (the approaches in Japan and the USA differ considerably in this respect) and the user group with is targeted.

5.1 Hitachi

Although no details are publicly known, a new system Hitachi is expected in the first half of 1998. This machine should replace the S3600/S3800 series of vector processors and at the same time be scalable to a considerable number of processors. In all probability, the structure of the system will be much like the NEC SX-4 (3.3.4), clustering the a modest amount of vector processors in an SMP fashion and connecting the clusters by a crossbar network. As said, no details of individual processor speed, connection type, or connection capacity are known at this stage.
Of a successor for the Hitachi SR2201 massively parallel processor product line, presently nothing is known.

5.2 NEC

NEC has, like Hitachi, a massively parallel processor product line and a vector processor line. The former, the Cenju-3 (3.4.11), is not very actively marketed and information is scarce. Still a successor is planned featuring the MIPS R10000 a its basic processor. This would speed up the speed per node with a factor of about 10, as a 500 Mflop/s R10000 will be available soon. Presently, no details, like the number of processors, and the network capacity are disclosed. However, one can assume that the machine structure largely will be retained.
This year a successor, the SX-5, of the reasonably successful NEC SX-4 (3.3.4) system is expected. The structure is believed to be almost identical to the that of the SX-4. Probably, no models with the expensive static RAM memories will be offerred anymore as was the case with the SX-4 models. It may be assumed that the processor speed will grow four-fold to about 8 Gflop/s when keeping in line with the general growth in speed over time. As the clustered shared-memory systems have worked well for NEC, the SX-5 undoubtly will have the same structure as the SX-4 systems.

5.3 SGI/Cray

It is not sure when a new SGI/Cray will come out, only that this will be in 1998. In view of the success and the cost/performance of the Fujitsu VPP300/700 (3.4.5) and the NEC SX-4 (3.3.4) it stands to reason that SGI/Cray will offer a similar machine based on CMOS technology and with vector processors. SGI/Cray

has already experience with such systems, the Cray Y-MP EL and the Cray J90 (3.3.1), be it that the per processor speed is lower than that of the Japanese systems which are targeted to the top of the market while the J90 is a typical midrange system. To be interesting, the new system should be at least in the 1 Gflop/s per processor range as present-day RISC processors are also approaching this speed. Also the maximum number of processors will have to be higher than what now is offerred in the Cray vector systems (32 max.) but a definite number of processors is not known to the author yet.
For the T3E product line it is not expected that 1998 will bring a new system. With the T3E-1200 (3.4.3) that was announced in November 1997 SGI/Cray probably has enough leeway to defer the presentation of a really different system in this sector for at least a year. When such a system is announced, it will contain almost certainly MIPS-built processors instead of the DEC Alhpa processors that are used now. As such a new system would be on the market by 1999/2000, the per processor peak performance should be at least be in the 2 Gflop/s range or higher.

Acknowledgments

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 us when necessary. Therefore, we will have to thank them here collectively but not less heartily for their support.
Special thanks are due to David Presberg from the Cornell Theory Center, USA, who provided pointers for the new larger product range of IBM.

References

[1] C. Amza, A.L. Cox, S. Dwarkadas, P. Keleher, H. Lu, R. Rajamony, W. Yu, W.Zwaenepoel, *TreadMarks: Shared Memory Computing on Networks of Workstations*, to appear in IEEE Computer (also: `www.cs.rice.edu/~willy/TreadMarks/papers.htm`)

[2] J.J. Dongarra, *Performance of various computers using standard linear equations software*, Computer Science Technical Report CS-89-85, Univ. of Tennessee, November 1997.

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

[4] M.J. Flynn, "Some computer organisations and their effectiveness, *IEEE Trans. on Comp.*, Vol. C-21, 9, (1972) 948–960.

[5] A. Geist, A. Beguelin, J. Dongarra, R. Manchek, W. Jaing, and V. Sunderam, *PVM: A Users' Guide and Tutorial for Networked Parallel Computing*, MIT Press, Boston, 1994.

[6] J.M.D. Hill, W. McColl, D.C. Stefanescu, M.W. Goudreau, K. Lang, S.B. Rao, T. Suel, T. Tsantilas, R. Bisseling, *BSPlib: The BSP Programming Library*, Technical report PRG-TR-29-9, Oxford University Computing Laboratory, May 1997. (Compressed Postscript with ANSI C examples, 142K; Compressed Postscript with Fortran 77 examples, 141K)

[7] R. W. Hockney, C. R. Jesshope, *Parallel Computers II*, Bristol: Adam Hilger, 1987.

[8] T. Horie, H. Ishihata, T. Shimizu, S. Kato, S. Inano, M. Ikesaka, *AP1000 architecture and performance of LU decomposition*, Proc. Internat. Symp. on Supercomputing, Fukuoka, Nov. 1991, 46–55.

[9] High Performance Fortran Forum, *High Performance Fortran Language Specification*, Scientific Programming, **2**, 13, (1993) 1–170.

[10] D.V. James, A.T. Laundrie, S. Gjessing, G.S. Sohi, *Scalable Coherent Interface*, IEEE Computer, **23**, 6, (1990),74–77. See also:
Scalable Coherent Interface: `http://sunrise.scu.edu/`

[11] Web page for the NAS Parallel benchmarks NPB2: `http://science.nas.nasa.gov/Software/NPB/`

[12] OpenMP Forum, *Fortran Language Specification, version 1.0*, Web page:`www.openmp.org/`, October 1997.

[13] D. Sanders, Y. Park, V. Govindan, *Parallel sorting on the NEC Cenju-3 and IBM SP2*, Proc. HPC Asia '97, IEEE, Apr 1997.

[14] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra, *MPI: The Complete Reference*, MIT Press, Boston, 1996.

[15] A.J. van der Steen, *Exploring VLIW: Benchmark tests on a Multiflow TRACE 14/300*, Academic Computing Centre Utrecht, Technical Report TR-31, April 1990.

[16] A.J. van der Steen, *The benchmark of the EuroBen Group*, Parallel Computing **17** (1991) 1211–1221.

[17] A.J. van der Steen, *Benchmark results for the Hitachi S-3800*, Supercomputer, **10**, 4/5, (1993) 32–45.

[18] A.J. van der Steen, ed. *Aspects of computational science*, NCF, The Hague, 1995.

[19] T. Sugawara, G. Skinner, Y. Park, K. Tatsukawa, A. Konagaya, *Performance evaluation of MPP LS-DYNA3D on Cenju-3/DE*, Proc. First Cenju Workshop, Report UH-CS-97-1, University of Houston, Apr 1997.

[20] A. Trew and G. Wilson: *Past, Present, Parallel: A Survey of Available Parallel Computer Systems*, Heidelberg: Springer-Verlag, 1991.