Overview of recent supercomputers

Aad J. van der Steen
high-performance Computing Group
Utrecht University
P.O. Box 80195
3508 TD Utrecht
The Netherlands
steen@phys.uu.nl
www.phys.uu.nl/~steen

Abstract

In this report we give an overview of high-performance 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. In addition, some general information about high-performance computer architectures and the various processors and communication networks employed in these systems is given in order to better appreciate the system information given in this report.

This document 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 16th edition of a report in which we attempt to give an overview of high-performance computer 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 high-performance machines is highly volatile: the rate with which systems are introduced — and disappear again — is high (although not as high as a few years ago) and therefore the information may 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. At this moment systems are disappearing from the market in a certain sense balance against the ones that are newly appearing. This is because the spectrum of integrated systems has lost a few members but also some (few) new systems have appeared. Besides that, an enormous growth in the use of clusters, some very large, can be observed. A larger amount of new systems may be expected in the next few years because of the renewed interest in computer architectures both on the processor level and the macro-architecture level. This new interest was sparked by the introduction of the Earth Simulator system in Japan which by TOP500 standards, see [45], for a long time the most powerful system in the world. As a result a new discussion emerged in the USA to look at new (and old) architectures as the gap between Theoretical Peak Performance and application performance for systems born from the ASCI (see [3]) initiative has grown steadily and for many users of such systems to an unacceptable level. Programs like the DARPA-funded HPCS program should curb this trend which cannot be done by staying on the same track of SMP-clustered RISC processor technology without further enhancements in memory access and intra- and inter-node bandwidth. Furthermore it has been realised that that no one processor type is best for all possible types of computation. So, a trend is emerging of diversifying processor types within a single system. A first sign of this is the appearance of FPGAs and other computation accelerators in systems with standard processors. We may expect that this trend will continue in the coming years which will make the high-performance computer landscape more diverse and interesting.

The majority of systems still look like minor variations on the same theme: clusters of RISC(EPIC)-based Symmetric Multi-Processing (SMP) nodes which in turn are connected by a fast network. Culler *et.al.* [8] consider this as a natural architectural evolution. However, it may also be argued that the requirements formulated in the ASCI programs has steered these systems in this direction and that this will change in the coming years for the reasons given above.

The supercomputer market is a very dynamic one and this is especially true for the cluster world that have emerged at a tremendous rate in the last few years. The number of vendors that sell pre-configured clusters has boomed accordingly and, as for the last few issues, we have decided *not* to include such configurations in this report: the speed with which cluster companies and systems appear and disappear makes this almost impossible. We will briefly comment on cluster characteristics and their position relative to other supercomputers in section 2.7 though.
For the tightly-coupled or "integrated" parallel systems, however, we can by updating this report at least follow the main trends in popular and emerging architectures. The details of the systems be reported do not allow the report to be shorter than in former years: between 60–70 pages.

As of the 11th issue we decided to introduce a section that describes the dominant processors in some detail. This seems fit as the processors the heart of the systems. We do that in section 2.8. In addition, as in the 13th issue on we include a section that discusses specific network implementations, being also constituents of primary importance apart from the general discussion about communication networks.

The rule 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 some interesting cluster systems at the Sandia, Los Alamos, and Lawrence Livermore National Laboratories in the USA (all with measured performances in the range of 6–19 Tflop/s) and the Japanese Earth Simulator system (with a performance around 40 Tflop/s) because they are not marketed or represent standard cluster technology (be it on a grand scale).
The rule that systems should be available within a time-span 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 euarler IBM SP or the Cray T90 series anymore although some of these systems are still in use. Generally speaking, we include machines that are presently 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 SGI Altix series and the Fujitsu PRIMEQUEST). 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.

Until the eighth issue of this report we ordered the systems by their architectural classes as explained in section 2.1. However, this distinction became more and more artificial as is explained in the same section. Therefore all systems described are simply listed alphabetically. In the header of each system description the machine type is provided. There is referred to the architectural class for as far this is relevant. We omit 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 favorite application. In fact, when the HPC Linpack test is used to measure the speed it is almost certain that for the average user the application performance will be significantly lower. 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.

Although for the average user the appearance of new systems in the last years tended to become rapidly more and more alike, it is still useful to dwell a little on the architectural classes that underlie this appearance. It gives some insight in the various ways that high-performance is achieved and a feeling why machines perform as they do. This is done in section 2 which will be referred to repeatedly in section 3 in the description of the machines.
Up till the 10^{th} issue we included a section (4) on systems that disappeared from the market. We reduced that section in the printed and PostScript versions because it tends to take an unreasonable part of the total text. Still, because this information is of interest to a fair amount of readers and it gives insight in the field of the historical development of supercomputing over the last 14 years, this information will still be available in full at URL `http://www.phys.uu.nl/~steen/web06/gone.html`. In section 5 we present some systems that are under development and have a fair chance to appear on the market. Because of the addition of the section on processors that introduces many technical terms, also a glossary is included.

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. Furthermore, we have set a threshold of about 100 Gflop/s for systems to appear in this report as, at least with regard to theoretical peak performance, single CPUs often exceed 2.5 Gflop/s although their actual performance may be an entirely other matter.

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 web version is available. The URLs are:
`www.arcade-eu.org/overview` (Europe).
`www.phys.uu.nl/~steen/web06/overview.html` (Europe).
`www.euroben.nl/reports/web06/overview.html` (Europe).
`www.netlib.org/utk/papers/advanced-computers/` (USA).

From this issue on we will *attempt* to keep the web version up-to-date by refreshing the contents more frequently than once a year. So, the printed version may lag a little behind the web version over the year.

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 on the hardware architecture. For a description of machines that can be considered to be classified as "high-performance" one is referred to [8, 40].

2.1 The main architectural classes

Since many years the taxonomy of Flynn [15] 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 separately.

- **SISD** machines: These are the conventional systems that contain one CPU and hence can accommodate one instruction stream that is executed serially. Nowadays many large mainframes may have more than one CPU but each of these execute instruction streams that are unrelated. Therefore, such systems still should be regarded as (a couple of) SISD machines acting on different data spaces. Examples of SISD machines are for instance most workstations like those of Hewlett-Packard, IBM, and SGI. 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 Gamma II and the Quadrics Apemille which are not marketed anymore since about a year.

  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 NEC SX-8B.
- **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 easy 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 NEC SX-8 vector system and a thousand-processor IBM p575 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. As we will see, distributed-memory MIMD systems exhibit a large variety in the topology of their interconnection network. The details of this topology are largely hidden from the user which is quite helpful with respect to portability of applications. For the distributed-memory systems we will sometimes use DM-SIMD and DM-MIMD to indicate the two subclasses.

As already alluded to, 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 [22] 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 and SGI, also support proprietary virtual shared-memory programming models due to the fact that these physically distributed memory systems are able to address the whole collective address space. So, for the user such systems have one *global address space* spanning all of the memory in the system. We will say a little more about the structure of such systems in section 2.6. In addition, packages like TreadMarks ([1]) provide a "distributed shared-memory" environment for networks of workstations. A good overview of such systems is given at [11].

Distributed processing takes the DM-MIMD concept one step further: instead of many integrated processors in one or several boxes, workstations, mainframes, etc., are connected by (Gigabit) Ethernet, or other, faster networks 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 can be much slower. Packages that initially were made to realise distributed computing like PVM (standing for Parallel Virtual Machine) [16], and MPI (Message Passing Interface, [25, 26]) have become *de facto* standards for the "message passing" programming model. MPI and PVM have become so widely accepted that they have been adopted by all vendors 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 by a fast communication network to obtain systems with a very high computational power. E.g., the NEC SX-8, and the Cray X1E have this structure. So, within the clustered nodes a shared-memory programming style can be used while between clusters message-passing should be used. It must be said that PVM is not used very much anymore and that MPI has more or less become the *de facto* standard.
For SM-MIMD systems we mention OpenMP [31, 7], that can be used to parallelise Fortran and C(++) programs by inserting comment directives (Fortran 77/90/95) or pragmas (C/C++) into the code. OpenMP has quickly been adopted by the all major vendors and has become a well established standard for shared memory systems.

Note, however, that for both MPI-2 and OpenMP 2.5, the latest standards, many systems/compilers only implement a part of these standards. One has therefore to inquire carefully whether a particular system has

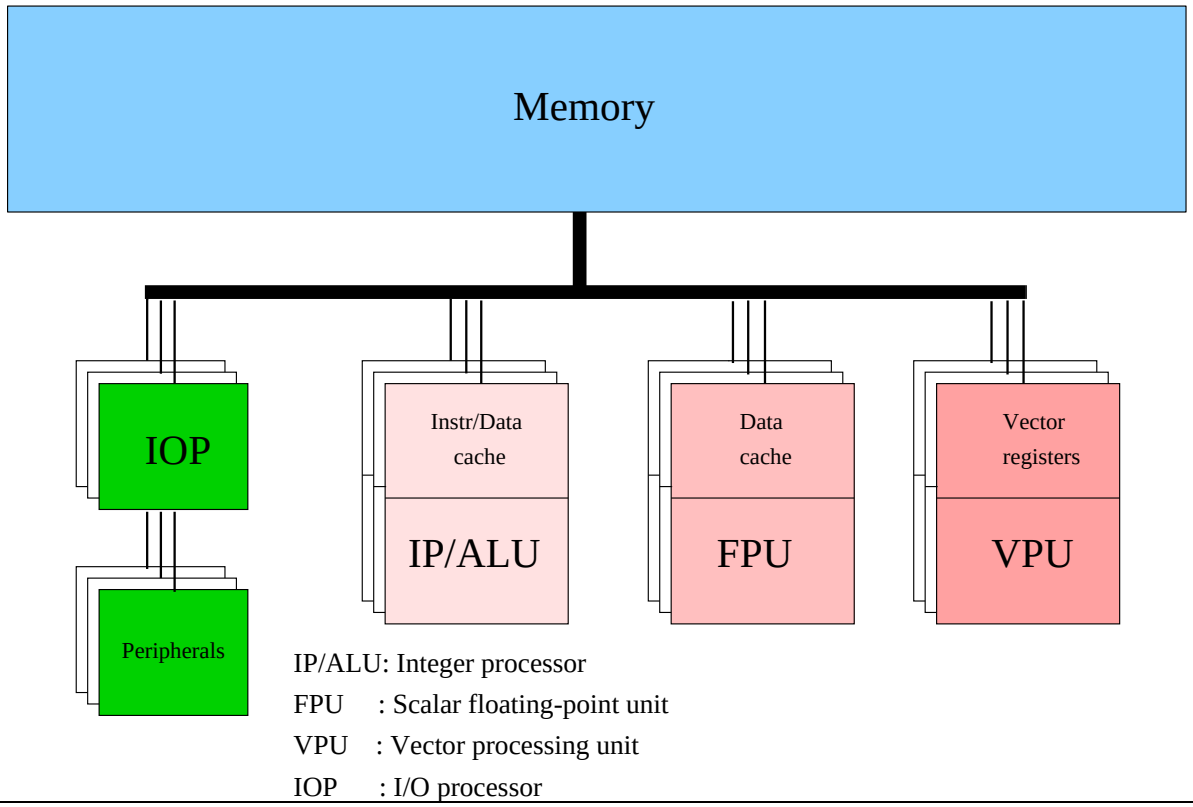


Figure 2.1: *Blockdiagram of a vector processor.*

the full functionality of these standards available. The standard vendor documentation will almost never be clear on this point.

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 [37]). In the block diagram in Figure 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 was the case in some Cray systems). It may be noted that the VPU does not show a cache. Vectorprocessors may have a cache but in many cases the vector unit cannot take advantage of it and execution speed may in some cases 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), 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 data paths between memory and the VPU, often compromises are sought and the full required bandwidth

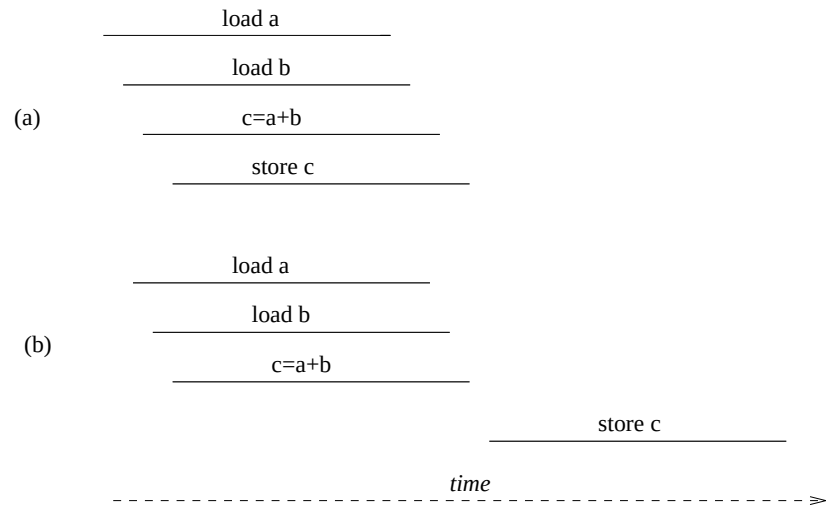


Figure 2.2: *Schematic diagram of a vector addition. Case (a) when two load- and one store pipe are available; case (b) when two load/store pipes are available.*

(i.e., two load operations and one store operation at the *same* time) Only Cray Inc. in its former Y-MP, C-series, T-series employed this very high bandwidth. Vendors rather rely on additional caches and other tricks to hide the lack of bandwidth.

The VPUs are shown as a single block in Figure 2.1. Yet, there is a considerable diversity in the structure of VPUs. Every VPU consists of a number of vector functional units, or "pipes" that fulfill one or several functions in the VPU. Every VPU will have pipes that are designated to perform memory access functions, thus assuring the timely delivery of operands to the arithmetic pipes and of storing the results in memory again. Usually there will be several arithmetic functional units for integer/logical arithmetic, for floating-point addition, for multiplication and sometimes a combination of both, a so-called compound operation. 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 16-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 [19]. 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. Presently, no commercially available machines of the processor-array type are marketed. However, because of the shrinking size of devices on a chip it may be worthwile to locate a simple processor with its network components on a single chip thus making processor-array systems economically viable again. In fact, common Graphical Processing Units (GPUs) share many characteristics with processor array systems.

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 offloaded to the front-end system. For instance, I/O may be through the front-end system, by the processor array machine itself or by 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.

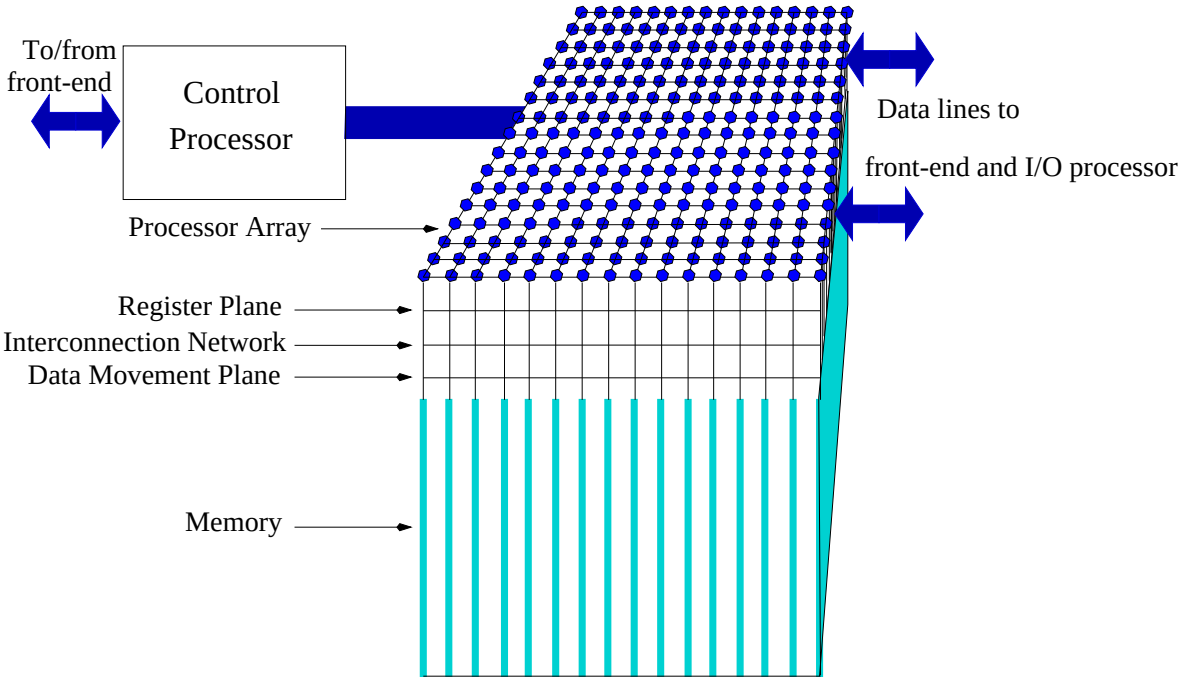


Figure 2.3: *A generic block diagram of a distributed-memory SIMD machine.*

As opposing ends of each grid line are also always connected the topology is rather that of a torus. This is not the only interconnection scheme: They might also be connected in 3-D, diagonally, or more complex structures.

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

The control processor as depicted in 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. A (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.,

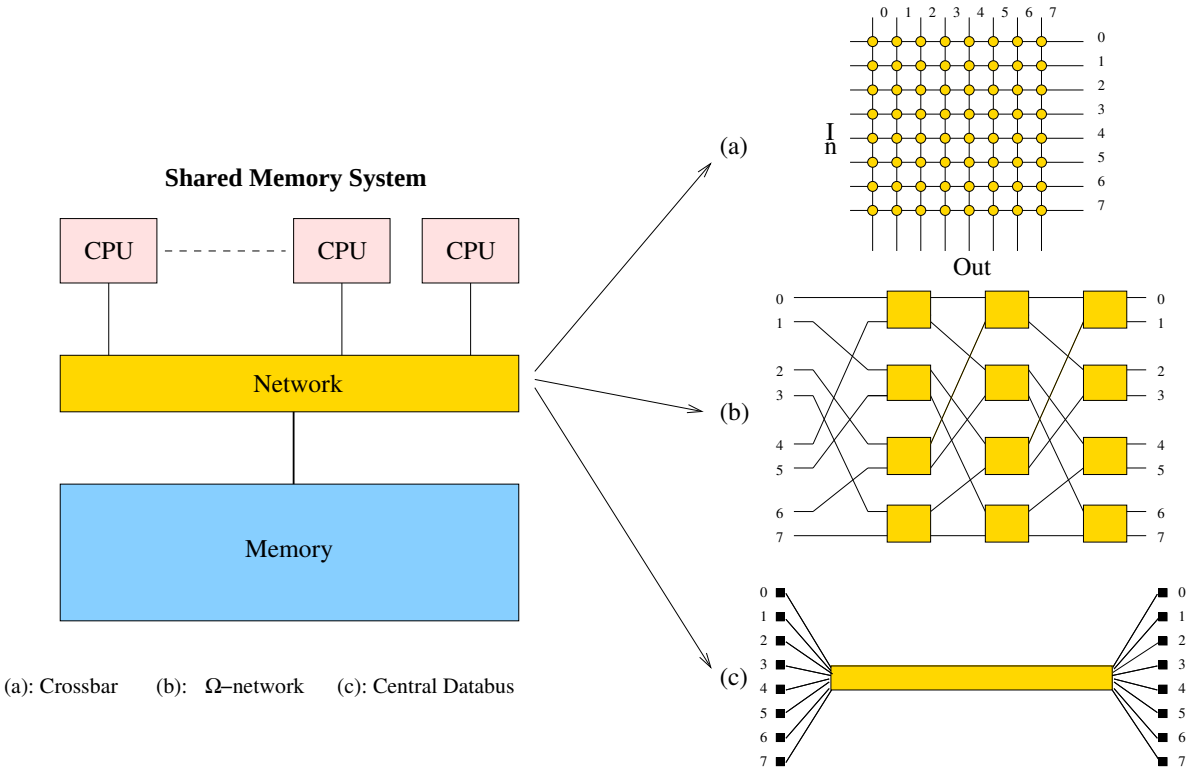


Figure 2.4: *Some examples of interconnection structures used in shared-memory MIMD systems.*

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 Quadrics Apemille was 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 were added to the processor-array. Their number was 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. beacuse, as mentioned, the execution time for bit-serial machines is proportional to the length of the 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 data path 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 data bus 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 has 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 MB/s. It has been used in the HP Exemplar systems, but also within a cluster of workstations as offered by SCALI. The SCI is much more than a simple bus and it can act as the hardware network framework for distributed computing, see [23].

A multi-stage crossbar is a network with a logarithmic complexity and it has a structure which is situated somewhere in between a bus and a crossbar with respect to potential capacity and costs. The Ω-network as depicted in figure 2.4 is an example. Commercially available machines like the IBM eServer p575, the SGI Altix 3000, and many others use(d) such a 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 likely that new machines will 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 within in a system node is still rather small (16 at most presently). When the number of processors would increase, however, technological problems might arise. Not only it becomes harder to build a crossbar of sufficient speed for the larger numbers of processors, the processors themselves generally also increase in speed individually, doubling the problems of making the speed of the crossbar match that of the bandwidth required by the processors.

Whichever network is used, the type of processors in principle could be arbitrary for any topology. In practice, however, bus structured machines do not have vector processors as the speeds of these would grossly mismatch with any bus that could be constructed with reasonable costs. 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 very important here. Most vectorprocessors employ special communication registers within the CPUs by which they can communicate directly with the other CPUs they have to synchronise with. The systems may also synchronise via the shared memory. Generally, this is much slower but it can 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 data bus to ensure 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 transparent 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 explicitly. 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 ([16, 25, 26]) and partly because, at least theoretically, this class of systems is able to outperform all other types of machines.

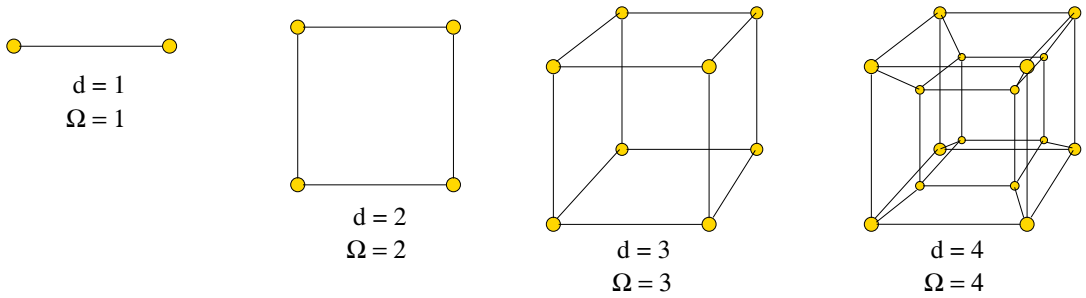
The advantages of DM-MIMD systems are clear: the bandwidth problem that haunts shared-memory systems is avoided because the bandwidth scales up automatically with the number of processors. Furthermore, the speed of the memory which is another critical issue with shared-memory systems (to get a peak performance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

Of course, DM-MIMD systems also have their disadvantages: The communication between processors is much slower than in SM-MIMD systems, and so, the synchronisation overhead in case of communicating tasks is generally orders of magnitude higher than in shared-memory machines. Moreover, the access to data that are not in the local memory belonging to a particular processor have to be obtained from non-local memory (or memories). This is again on most systems very slow as compared to local data access. When the structure of a problem dictates a frequent exchange of data between processors and/or requires many processor synchronisations, it may well be that only a very small fraction of the theoretical peak speed can be obtained. As already mentioned, the data and task decomposition are factors that mostly have to be dealt with explicitly, which may be far from trivial.

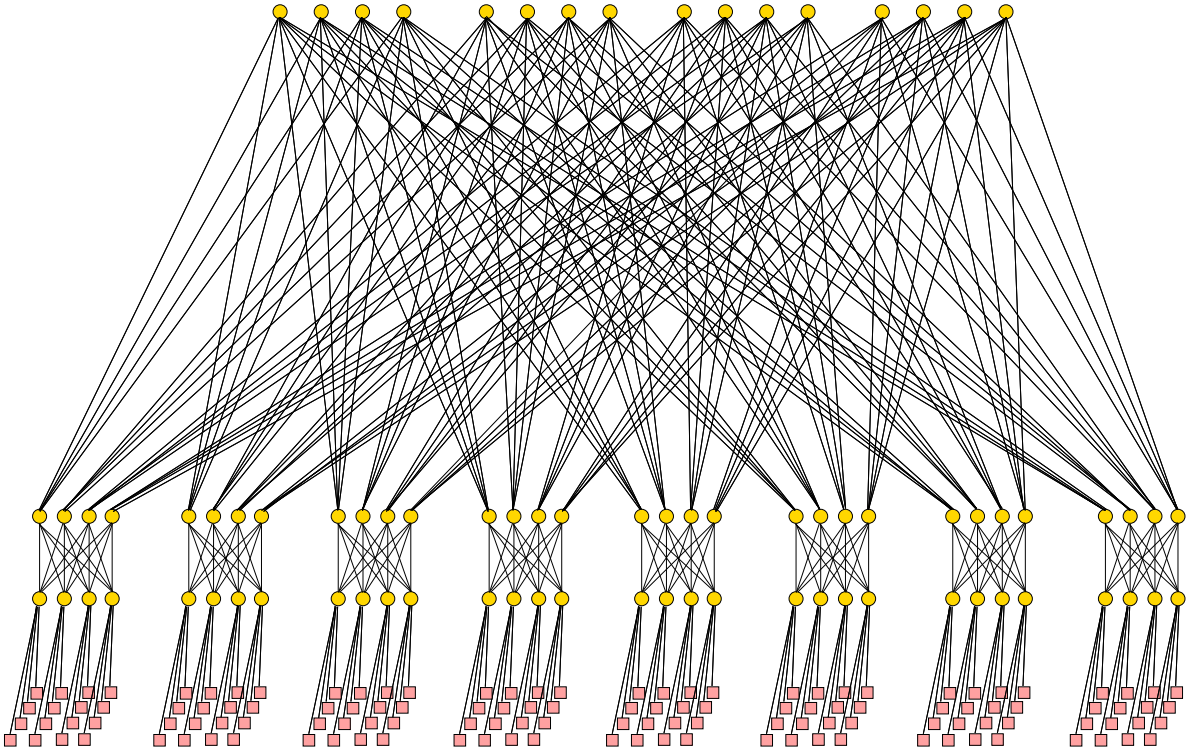
It will be clear from the paragraph above that also for DM-MIMD machines both the topology and the speed of the data paths are of crucial importance for the practical usefulness of a system. Again, as in the section on SM-MIMD systems, the richness of the connection structure has to be balanced against 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).

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 does 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 these 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.

Another cost-effective way to connect a large number of processors is by means of a *fat tree*. In principle a simple tree structure for a network is sufficient to connect all nodes in a computer system. However, in practice it turns out that near the root of the tree congestion occurs because of the concentration of messages that first have to traverse the higher levels in the tree structure before they can descend again to their target nodes. The fat tree amends this shortcoming by providing more bandwidth (mostly in the form of multiple connections) in the higher levels of the tree. One speaks of a N-ary fat tree when the levels towards the roots are N times the number of connections in the level below it. An example of a quaternary fat tree with a bandwidth in the highest level that is four times that of the lower levels is shown in Figure 2.5 (b).



(a) Hypercubes, dimension 1–4.



(b) A 128–way fat tree.

Figure 2.5: *Some often used networks for DM machine types.*

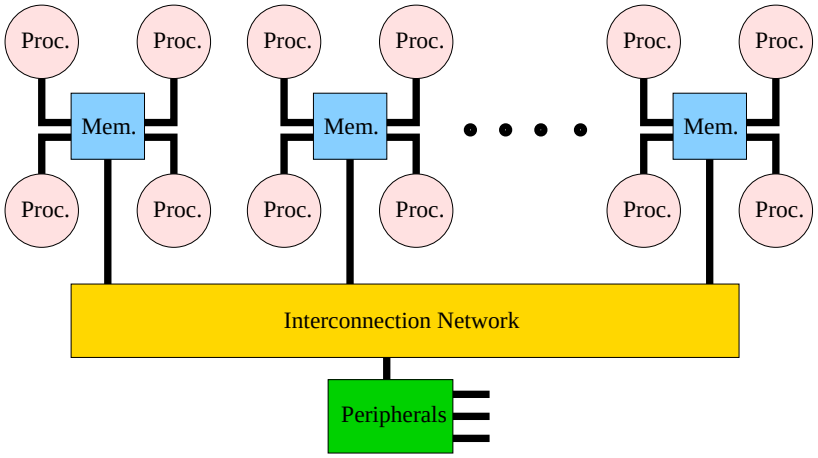


Figure 2.6: *Block diagram of a system with a "hybrid" network: clusters of four CPUs are connected by a crossbar. The clusters are connected by a less expensive network, e.g., a Butterfly network.*

A number 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 [20]. Also on IBM's BlueGene systems this philosophy has been followed.

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 many thousands of nodes through only a few switching stages. In addition to the hypercube structure, other logarithmic complexity networks like Butterfly, Ω, or shuffle-exchange networks and fat trees 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 many cases this may result in turning computational-bound problems into communication-bound problems.

2.6 ccNUMA machines

As already mentioned in the introduction, a trend can be observed to build systems that have a rather small (up to 16) number of RISC processors that are tightly integrated in a cluster, a Symmetric Multi-Processing (SMP) node. The processors in such a node are virtually always connected by a 1-stage crossbar while these clusters are connected by a less costly network. Such a system may look as depicted in Figure 2.6. Note that in Figure 2.6 all CPUs in a cluster are connected to a common part of the memory. This is similar to the policy mentioned for large vectorprocessor ensembles mentioned above but with the important difference that all of the processors can access all of the address space if necessary. The most important ways to let the SMP nodes share their memory are S-COMA (Simple Cache-Only Memory Architecture) and ccNUMA, which stands for Cache Coherent Non-Uniform Memory Access. Therefore, such systems can be considered as SM-MIMD machines. On the other hand, because the memory is physically distributed, it cannot be guaranteed that a data access operation always will be satisfied within the same time. In S-COMA systems

the cache hierarchy of the local nodes is extended to the memory of the other nodes. So, when data is required that does not reside in the local node's memory it is retrieved from the memory of the node where it is stored. In ccNUMA this concept is further extended in that all memory in the system is regarded (and addressed) globally. So, a data item may not be physically local but logically it belongs to one shared address space. Because the data can be physically dispersed over many nodes, the access time for different data items may well be different which explains the term non-uniform data access. The term "Cache Coherent" refers to the fact that for all CPUs any variable that is to be used must have a consistent value. Therefore, is must be assured that the caches that provide these variables are also consistent in this respect. There are various ways to ensure that the caches of the CPUs are coherent. One is the *snoopy bus protocol* in which the caches listen in on transport of variables to any of the CPUs and update their own copies of these variables if they have them and are requested by a local CPU. Another way is the *directory memory*, a special part of memory which enables to keep track of all the copies of variables and of their validness.

Presently, no commercially available machine uses the S-COMA scheme. By contrast, there are several popular ccNUMA systems (Bull NovaScale, HP Superdome, and SGI Altix 3000) commercially available. An important characteristic for NUMA machines is the *NUMA factor*. This factor shows the difference in latency for accessing data from a local memory location as opposed to a non-local one. Depending on the connection structure of the system the NUMA factor for various parts of a system can differ from part to part: accessing data from a neighbouring node will be faster than from a distant node in which possibly a number of stages of a crossbar must be traversed. So, when a NUMA factor is mentioned, this is mostly for the largest network crosssection, i.e., the maximal distance between processors.

For all practical purposes we can classify these systems as being SM-MIMD machines also because special assisting hardware/software (such as a directory memory) has been incorporated to establish a single system image although the memory is physically distributed.

2.7 Clusters

The adoption of clusters, collections of workstations/PCs connected by a local network, has virtually exploded since the introduction of the first Beowulf cluster in 1994. The attraction lies in the (potentially) low cost of both hardware and software and the control that builders/users have over their system. The interest for clusters can be seen for instance from the IEEE Task Force on Cluster Computing (TFCC) which reviews on a regular basis the current status of cluster computing [46]. Also books how to build and maintain clusters have greatly added to their popularity [44, 36]. As the cluster scene becomes relatively mature and an attractive market, large HPC vendors as well as many start-up companies have entered the field and offer more or less ready out-of-the-box cluster solutions for those groups that do not want to build their cluster from scratch.

The number of vendors that sell cluster configurations has become so large that it is not possible to include all these products in this report. In addition, there is generally a large difference in the usage of clusters and their more integrated counterparts that we discuss in the following sections: clusters are mostly used for *capability computing* while the integrated machines primarily are used for *capacity computing*. The first mode of usage meaning that the system is employed for one or a few programs for which no alternative is readily available in terms of computational capabilities. The second way of operating a system is in employing it to the full by using the most of its available cycles by many, often very demanding, applications and users. Traditionally, vendors of large supercomputer systems have learned to provide for this last mode of operation as the precious resources of their systems were required to be used as effectively as possible. By contrast, Beowulf clusters are mostly operated through the Linux operating system (a small minority using Microsoft Windows) where these operating systems either miss the tools or these tools are relatively immature to use a cluster well for capacity computing. However, as clusters become on average both larger and more stable, there is a trend to use them also as computational capacity servers. In [42] is looked at some of the aspects that are necessary conditions for this kind of use like available cluster management tools and batch systems. In the same study also the performance on an application workload was assessed, both on a RISC (Compaq Alpha) based configuration and on Intel Pentium III based systems. An important, but not very surprising conclusion was that the speed of the network is very important in all but the most compute bound

applications. Another notable observation was that using compute nodes with more than 1 CPU may be attractive from the point of view of compactness and (possibly) energy and cooling aspects, but that the performance can be severely damaged by the fact that more CPUs have to draw on a common node memory. The bandwidth of the nodes is in this case not up to the demands of memory intensive applications.
As recently cluster nodes have become available with 4 processors where each processor also may have 2 processor cores, this issue has become all the more important and one might have to choose for capacity-optimised nodes with more processors but less bandwidth/processor core or capabililty-optimised nodes that contain less processors per node but have a higher bandwidth available for the processors in the node. This choice is not particular to clusters (although the phenomenon is relatively new for them), it also occurs in the integrated CC-NUMA systems.

Fortunately, there is nowadays a fair choice of communication networks available in clusters. Of course 100 Mb/s Ethernet or Gigabit Ethernet is always possible, which is attractive for economic reasons, but has the drawback of a high latency ($\approx 50 - -100\mu$s). Alternatively, there are for instance networks that operate from user space, like Myrinet [27], Infiniband, QsNet and SCI [23]. The first two have maximum bandwidths in the order of 250 MB/s, 850 MB/s, and 900 MB/s, respectively, and a latency in the range of 6–9 μs. SCI has a bandwidth of 400–500 MB/s theoretically and a latency under 3 μs. The latter solution is more costly but is nevertheless employed in some cluster configurations. The network speeds as shown by Myrinet, and, certainly, QsNet and SCI is more or less on par with some integrated parallel systems as discussed later. So, possibly apart from the speed of the processors and of the software that is provided by the vendors of DM-MIMD supercomputers, the distinction between clusters and this class of machines becomes rather small and will undoubtly decrease in the coming years.
An interesting new player is PathScale's Infinipath. In this network the Infiniband fabric is combined with HyperTransport (AMD's processor-to-processor bus/protocol, see section 2.8.1) not using the Infiniband protocol but rather PathScale's own. For AMD Opteron processors (and presently only those) this results in a very high bandwidth of $>$ 900 MB/s and a very low latency of about 1.3 μs.

2.8 Processors

In comparison to 10 years ago the processor scene has become drastically different. While in the period 1980–1990, the proprietary processors and in particular the vectorprocessors were the driving forces of the supercomputers of that period, today that role has been taken over by common off-the-shelf processors. In fact there are only two companies left that produce vector systems while all other systems that are offered are based on RISC/EPIC CPUs or x86-like. Therefore it is useful to give a brief description of the main processors that populate the present supercomputers and look a little ahead to the processors that will follow in the coming year.

The RISC processor scene has shrunken significantly in the last few years. chip will only be marketed until 2006 due to HP's decision to replace these processors by the Itanium line of processors. The PA-RISC 8800 is still made for the Superdome 9000 machine but it will disappear shortly, although officially a PA-RISC 8900 still will be produced. Still, we will not discuss this processor in this section as should be regarded as a kind of service to a shrinking community of users that is critically dependent on the PA-RISC architecture rather than a vital processor development path.
Although at this moment the RISC processors still play a significant role many of these will disappear from the scene: the Alpha, PA-RISC and MIPS processors disappear(ed) all in favour of the Itanium processor product line. The disappearance of RISC processor families demonstrates a trend that is both worrying and interesting: worrying because the diversity in the processor field is decreasing severely and, with it, the choice for systems in this sector. On the other hand there is the trend to enhance systems having run-of-the-mill processors with special-purpose add-on processors in the form of preconfigured FPGAs or DSPs because their possibilities in performance, price level, and ease of use has improved to a degree that they offer attractive alternatives for certain application fields.

The RISC processors generally have a clock frequency that is lower than that of the Intel Pentium 3/4 processors or the corresponding AMD Intel look-alikes. However, they have a number of facilities that put them ahead in the speed of floating-point oriented applications. Firstly, all RISC processors are able to

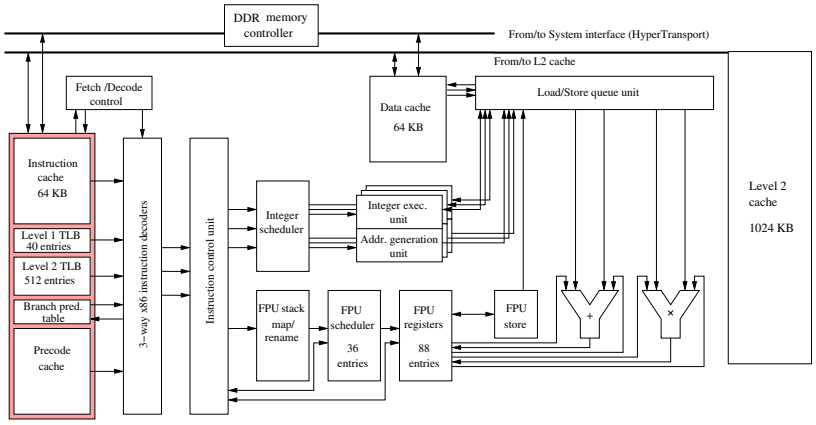


Figure 2.7: *Block diagram of an AMD Opteron processor core.*

deliver 2 or more 64-bit floating-point results in one clock cycle. Secondly, all of them feature out-of-order instruction execution, which enhances the number of instructions per cycle that can be processed (although the newer AMD processors also have 2-way floating-point instruction issuing and out-of-order execution, they are somewhat limited by their adherence to the Intel x86 instruction set). The bandwidth from the processor to the memory, in case of a cache miss, used to be larger than that of the Intel(-like) processors. For the AMD Opteron, however, this is not the case anymore, because of its on-chip memory controller and HyperTransport bus from memory to CPU. Notwithstanding the commonalities between the various RISC processors, there are also differences in instruction latencies, number of instructions processed, etc., which we will address below. We provide block diagrams for each of the processors to give a schematic idea of their structure. However, these figures do not reflect the actual layout of the devices on the respective chips.
Other developments are the placement of more than one processor core on a processor chip and the introduction of some form of multi-threading. We will discuss these developments for each of the processors separately.

2.8.1 AMD Opteron

The Opteron dual core processor is the newest development in the AMD Opteron family. It is manufactured in 90 nm feature size and available since the end of 2005. It is a clone with respect to Intel's x86 Instruction Set Architecture, and it is becoming more and more popular for use in clusters. In addition it is used in two types of Cray systems that we will discuss later.
The Opteron processor has many features that are also present in modern RISC processors: it supports out-of-order execution, has multiple floating-point units, and can issue up to 9 instructions simultaneously. In fact, the processor core in very similar to that of the Athlon processor. A block diagram of the processor core is shown in Figure 2.7. Note that in the dual core version two of these cores are housed on one CPU chip. The two cores are connected by an on-chip crossbar (see next section) that also connects to the memory controller and to other processors on the board (if present).

The figure shows that the processor has three pairs of Integer Execution Units and Address Generation Units that via an 24-entry Integer Scheduler takes care of the integer computations and address calculations. Both the Integer Scheduler and the Floating-Point Scheduler are fed by the 96-entry Instruction Control Unit that receives the decoded instructions from the instruction decoders. An interesting feature of the Opteron is the pre-decoding of x86 instructions in fixed-length macro-operations, called RISC Operations (ROPs), that can be stored in a Pre-decode Cache. This enables a faster and more constant instruction flow to the instruction decoders. In comparison to the Athlon the instruction decode pipeline has been deepened by two stages to 12 to enable a higher yield of ROPs. Like in RISC processors, there is a Branch Prediction Table assisting in branch prediction.
The floating-point units allow out-of-order execution of instructions via the FPU Stack Map & Rename unit. It receives the floating-point instructions from the Instruction Control Unit and reorders them if necessary

before handing them over to the FPU Scheduler. The Floating-Point Register File is 88 elements deep which approaches the number of registers as is available on RISC processors [1].
The floating-point part of the processor contains three units: a Floating Store unit that stores results to the Load/Store Queue Unit which also performs some miscellaneous floating-point operations, and Floating Add and Multiply units that can work in superscalar mode, resulting in two floating-point results per clock cycle. Because of the compatibility with Intel's Pentium 4 processors, the floating-point units also are able to execute Intel SSE2/3 instructions and AMD's own 3DNow! instructions. However, there is the general problem that such instructions are not accessible from higher level languages, like Fortran 90 or C(++). Both instruction sets are meant for massive processing of visualisation data and only allow for 32-bit precision to be used.
Due to the shrinkage of components a core can harbour a secondary cache of 1024 KB with an 8-cycle latency and the memory controller. This, together with a significantly enhanced memory bus can deliver up to 6.4 GB/s of bandwidth to/from the memory. This memory bus, called HyperTransport by AMD, is derived from licensed Compaq technology and similar to that employed in HP/Compaq's former EV7 processors. It allows for "glueless" connection of several processors to form multi-processor systems with very low memory latencies. HyperTransport 1, the present bus, can transport 8 GB/s, including cache coherency traffic, I/O, and interprocessor data.
The clock frequency is in the range of 2.2–2.6 GHz which makes the Opteron an interesting alternative for the few RISC processors that are still available at this moment. Especially the HyperTransport interconnection possibilities makes it highly attractive for building SMP-type clusters. A very large (over 11,000 processor) Opteron cluster, the Red Storm system, has been built at Sandia National Laboratory by Cray Inc. which uses the HyperTransport facility because it is an open standard that makes the connection with the Cray network much simpler.

2.8.2 IBM POWER5+

In the systems that feature as IBM's supercomputer line, the p575 series the nodes still contain the POWER4+ chip as the computational engine although the sucessor, the POWER5+ chip already is available in other server lines of IBM. Undoubtly the POWER5 will eventually appear in the p575-like systems but presently this is not yet the case. Here we discuss the POWER5+. It should, according to IBM's official road map. be replaced quite soon by the POWER6 processor which should be fairly different from the POWER5+ but no details of the latter are available yet.
At the time of writing, the clock frequency of the POWER5+ is in the range 1.5–1.9 GHz. The POWER5+ chip it in fact a re-engineering of the POWER5 in 90 nm technology instead of 130 nm and has no really new features other than that the clock frequency might become slightly higher during its lifetime. The techonology shrink enables IBM to place two processor cores on a chip since the introduction of the POWER4 as shown in Figure 2.8a. The L1 instruction cache has a size of 64 KB and is direct-mapped while the L1 data cache is 32 KB and is 4-way set associative. The chip also harbours 1.875 MB of secondary cache divided over three modules of 0.625 MB each which are 10-way set associative while the 36 MB off-chip L3 cache is 12-way set associative.

The L2 cache modules are connected to the processors by the Core Interface Unit (CIU) switch, a 2×3 crossbar with a bandwidth of 40 B/cycle per port. This enables to ship 32 B to either the L1 instruction cache or the data cache of each of the processors and to store 8 B values at the same time. Also, for each processor there is a Non-cacheable Unit that interfaces with the Fabric Controller and that takes care of non-cacheable operations. The Fabric Controller is responsible for the communication with three other chips that are embedded in the same Multi Chip Module (MCM), to L3 cache, and to other MCMs. The bandwidths at 1.9 GHz are 19.8, 13.2, and 9.9 GB/s, respectively. The chip further still contains a variety of devices: the L3 cache directory and the L3 and Memory Controller that should bring down the off-chip latency considerably, the GX Controller that responsible for the traffic on the GX bus. This bus transports data to/from the system and in practice is used for I/O or the fast communication network. Some of the integrated devices, like the Performance Monitor, and logic for error detection and logging are not shown in Figure 2.8a. The L2 caches of two neighbouring chips are connected and the L3 caches are directly connected

[1] For the x86 instructions 16 registers in a flat register file are present instead of the register stack that is usual for Intel architectures.

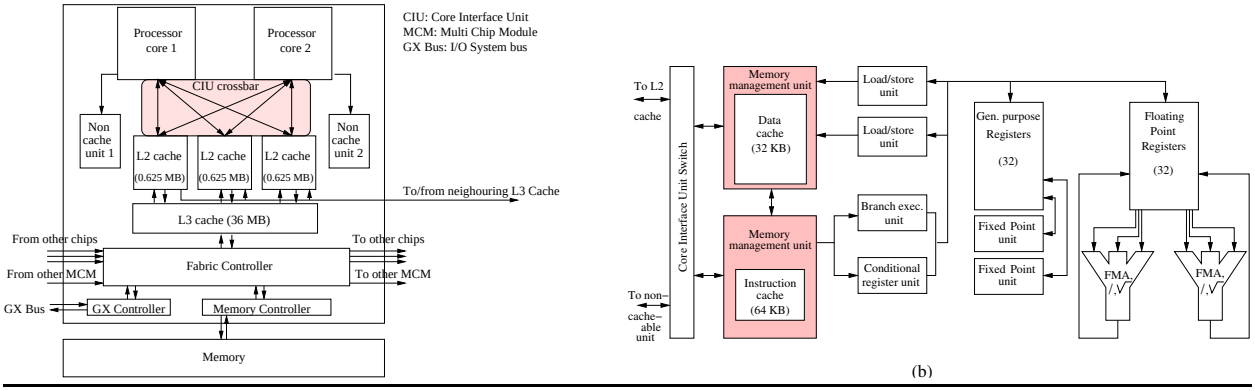


Figure 2.8: *Diagram of the IBM POWER5+ chip layout (a) and of the processor core (b).*

to the L2 caches. The L3 caches are also partitioned in three parts that each serve as a "spill cache" for their L2 counterpart, i.e., data that has to be flushed out of the L2 cache is transferrred to the corresponding L3 cache part. The L3 cache has a latency of 80 cycles as opposed to the main memory that has a latency of 220 cycles.
The better cache characteristics leads to less waiting time for regular data access: evaluation of a high order polynomial and matrix-matrix multiplication attain 90% or better of the peak performance while this is 65–75% on the POWER4+ chip. There is another feature of the POWER5+ that does not help for regular data access but which can be of benefit for programs where the data access is not so regular: Simultaneous Multithreading (SMT). The POWER5 CPUs are able to keep two process threads at work at the same time. The functional units get instructions for the functional units from any of the two threads whichever is able to fill a slot in an instruction word that will be issued to the functional units. In this way a larger fraction of the functional units can be kept busy, improving the overall efficiency. For very regular computations single thread (ST) mode may be better because in SMT mode the two threads compete for entries in the caches which may lead to trashing in the case of regular data access. Note that SMT is somewhat different from the "normal" way of multi-threading. In this case a thread that stalls for some reason is stopped and replaced by another process thread that is awoken at that time. Of course this takes some time that must be compensated for by the thread that has taken over. This means that the second thread must be active for a fair amount of cycles (preferably a few hundred cycles at least). SMT does not have this drawback but scheduling the instructions of both threads is quite complicated.

A block diagram of the processor core is shown in Figure 2.8b. In many ways the POWER5+ processor core is still similar to the former POWER3 processor: there are 2 integer functional units instead of 3 (called Fixed Point Units by IBM) and instead of a fused Branch/Dispatch Unit, the POWER5+ core has a separate Branch and Conditional Register Unit, 8 execution units in all. The execution units have instruction queues associated with them that enables the out-of-order processing of up to 200 instructions in various stages. Having so many instructions simultaneously in flight calls for very sophisticated branch prediction facilities. Instructions are fetched from the Instruction Cache under control of the Instruction Fetch Address Register which in turn is influenced by the branch predict logic. This consists of a local and a global Branch History Table (BHT), each with 16 K entries and a so-called selector table which keeps track of which of the BHTs has functioned best in a particular case in order to select the prediction priority of the BHTs for similar cases coming up.
Unlike in the former POWER3, the fixed point units performs integer arithmetic operations that can complete in one cycle as well as multi-cycle operations like integer multiply and divide. There are no separate floating-point units for operations that require many cycles like divisions and square roots. All floating-point operations are taken care of in the FP units and there is an instruction to accommodate the `axpy` operation, called Fused Multiply Add (FMA) at IBM's which could deliver 2 floating-point results every cycle. This brings the theoretical peak performance at 7.6 Gflop/s at the current clock frequency. The composition of the floating-point operations should be such that the units have indeed enough FMAs to perform. Otherwise the performance drops by a factor of 2.

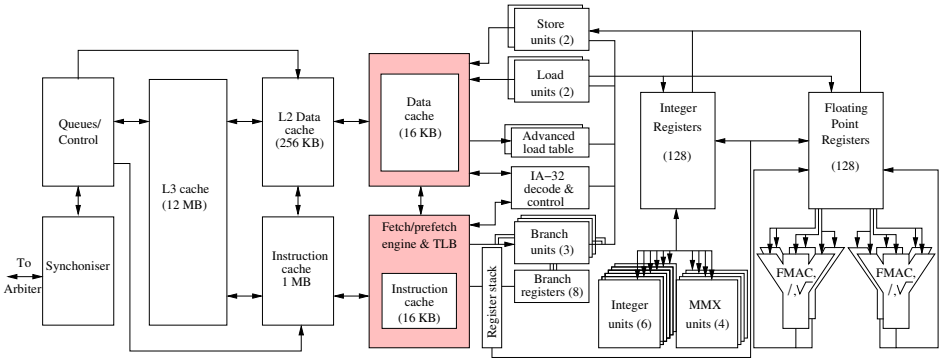


Figure 2.9: *Block diagram of the Intel Itanium 2 processor core.*

Although here the dual core version of the chip is described that is positioned for general processing, also a single core version is marketed that is recommended for HPC use. The reason is that in this case the bandwidth from the L2 cache does not have to be shared between the CPUs and a contention-free transfer of up to 108.8 GB/s can be achieved while in the dual core version a peak bandwidth of 163.2 GB/s is to be shared between both CPUs.

It is interesting to see that presently two vendors, AMD and IBM, have facilities that enable glueless coupling of processors although the packaging and implementation is somewhat different. All implementations allow for low-latency SMP nodes with a considerable number of processors stimulating the trend to build parallel systems based on SMP nodes.

2.8.3 IBM BlueGene processor

This processor is in fact a modified PowerPC 440 processor which is made especially for the IBM BlueGene family. It runs presently at a speed of 700 MHz. Because of its rather specialised nature we will discuss it when describing the BlueGene system proper in section 3.1.12.

2.8.4 Intel Itanium 2

The Itanium 2 is a representative of Intel's IA-64 64-bit processor family and as such the second generation. The first Itanium processor came out in 2001, but has not spread widely, primarily because the Itanium 2 would follow quickly with projected performance levels up to twice that of the first Itanium. The first Itanium 2 implementation ran at 0.8–1.0 GHz and has been followed quickly by a technology shrink (code name Madison) to the present-day Itanium 2 with clock frequencies in the range 1.3–1.6 GHz. At time of writing this report the next generation is becoming available. The processor core is almost unaltered with respect to the Madison processor but it is now built with 90 nm feature size instead of 130 nm and two cores are put onto a chip working at a clock frequency of 1.8 GHz. So, the new processor, code name Montecito, is a dual core processor like the new AMD Opteron, the IBM POWER5+, and the SUN SPARC4+. Another major change that is not immediately obvious from the block diagram in Figure 2.9 is the it is dual-threaded, be it not so fine-grained as the IBM POWER5+. Because of the technology shrink the power requirements are lower than for the Madison processor, 100W, even if there are two processor cores on the chip.
The Itanium family of processors has characteristics that are different from the RISC chips presented elsewhere in this section. A block diagram of the Itanium 2 is shown in 2.9.

Figure 2.9 shows a large amount of functional units that must be kept busy. This is done by large instruction words of 128 bits that contain 3 41-bit instructions and a 5-bit template that aids in steering and decoding the instructions. This is an idea that is inherited from the Very Large Instruction Word (VLIW) machines that have been on the market for some time about ten years ago. The two load/store units fetch two instruction words per cycle so six instructions per cycle are dispatched. The Itanium has also in common with these systems that the scheduling of instructions, unlike in RISC processors, is not done dynamically at run time but rather by the compiler. The VLIW-like operation is enhanced with predicated execution

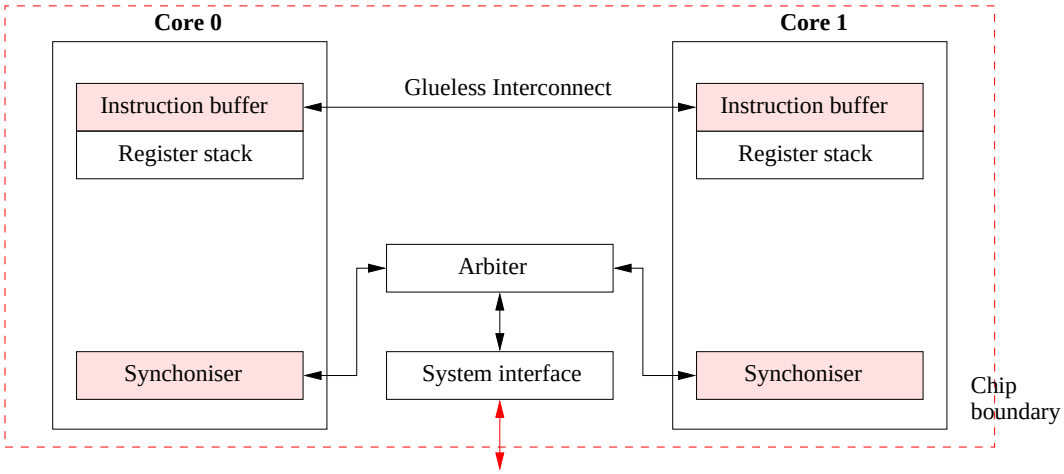


Figure 2.10: *Block diagram of 2 processor cores on a Montecito chip.*

which makes it possible to execute instructions in parallel that normally would have to wait for the result of a branch test. Intel calls this refreshed VLIW mode of operation EPIC, Explicit Parallel Instruction Computing. Furthermore, load instructions can be moved and the loaded variable used before a branch or a store by replacing this piece of code by a test on the place it originally came from to see whether the operations have been valid. To keep track of the advanced loads an Advanced Load Address Table (ALAT, and there are two of them) records them. When a check is made about the validness of an operation depending on the advanced load, the ALATs are searched and when no entry is present the operation chain leading to the check is invalidated and the appropriate fix-up code is executed. Note that this is code that is generated at compile time so no control speculation hardware is needed for this kind of speculative execution. This would become exceedingly complex for the many functional units that may be simultaneously in operation at any time.

As can be seen from Figure 2.9 there are four floating-point units capable of performing Fused Multiply Accumulate (FMAC) operations. However, two of these work at the full 82-bit precision which is the internal standard on Itanium processors, while the other two can only be used for 32-bit precision operations. When working in the customary 64-bit precision the Itanium has a theoretical peak performance of 6 Gflop/s at a clock frequency of 1.5 GHz. Using 32-bit floating arithmetic, the peak is doubled. In the first generation Itanium there were 4 integer units for integer arithmetic and other integer or character manipulations. Because the integer performance of this processor was modest, 2 integer units have been added to improve this. In addition four MMX units are present to accommodate instructions for multi-media operations, an inheritance from the Intel Pentium processor family. For compatibility with this Pentium family there is a special IA-32 decode and control unit.

The register files for integers and floating-point numbers is large: 128 each. However, only the first 32 entries of these registers are fixed while entries 33–128 are implemented as a register stack. The primary data and instruction caches are 4-way set associative and rather small: 16 KB each. This is the same as in the former Itanium processors. The L1 cache is full speed: data and instructions can be delivered every clock cycle to the registers. An enhancement with respect to the Madison processor is that the L2 cache has been split: instead of a unified secondary cache with a size of 256 KB now the data cache alone has that size while an L2 instruction cache of 1 MB is added. The code for VLIW/EPIC processors tends to be larger than equivalent RISC code by a factor of 2–3, so this enhancement was welcome, also because the processor is now dual-threaded and therefore the instruction cache may contain instructions from both threads. Both L2 caches are 8-way set-associative. Floating-point data are loaded directly from the L2 cache to registers. Moreover, the L3 cache resides on the chip and is no less than 12 MB/core. The bus is 128 bits wide and operates at a clock frequency of 400 MHz or 667 MHz totaling to 6.4 or 10.6 GB/s, respectively. For the data hungry Montecito the latter bandwidth is no luxury.

As already remarked before, the Montecito is dual-threaded. So, when the control logic (located at the upper left in Figure 2.9) decides that no progress will be made with one thread it dispatches the other thread

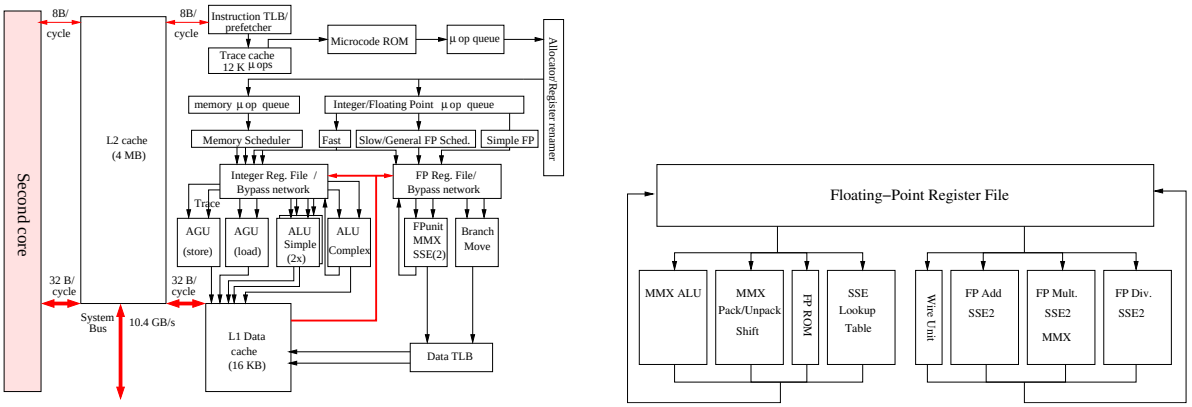


Figure 2.11: *Block diagram of an Intel Pentium IV processor core and floating-point unit.*

to minimise the idle stages in the instructions that are executed. This will most often happen with very irregular data access patterns where it is impossible to load all relevant data in the caches beforehand. The switch between threads is based on an "urgency level" ranging from 0–7. When the urgency of the active thread falls below that of the inactive one the latter becomes active and vice versa.
Because now two cores are present on a chip some provisions had to be added to let them cooperate without problems. The synchronisers in the core feed their information about read and write requests and cache line validity to the arbiter (see Figure 2.10). The arbiter filters out the unnecessary requests and combines the snooping information from both cores before handing the requests over to the system interface. In addition, the arbiter assures a fair access of both cores to the system interface.

The introduction of the first Itanium has been deferred time and again which quenched the interest for use in high-performance systems. With the availability of the Itanium 2 in the second half of 2002 the adoption has sped up. Apart from HP also Bull, Fujitsu, Hitachi, NEC, SGI, and Unisys are offering now multiprocessor systems with this processor replacing the Alpha, PA-RISC, SPARC, and MIPS processors as were employed by HP, Fujitsu, and SGI.

2.8.5 Intel Xeon

Although the Intel Xeon processors are not applied in integrated parallel systems these days, they play a major role in the cluster community as the mojority of compute nodes in Beowulf clusters are of this type. Therefore we briefly discuss also this type of processor. We concentrate on the Xeon, the server version of the IA-32 processor family, as this is the type to be found in clusters, mostly in 2-processor nodes.
As of 2006 Intel has introduced an enhanced microarchitecture for the IA-32 instruction set architecture called the Core architecture. The server version with the code name Woodcrest is a first implementation of this new microarchitecture. The Woodcrest processor has two processor cores as now all high-end processors do. In addition, many improvements have been made to increase the performance and at the same time to decrease the power requirements.
In Figure 2.11 a block diagram of the processor is shown with one of the cores in some detail. Note that the two cores share one second level cache while the L1 caches and TLBs are local to each of the cores.
To stay backwards compatible with the x86 (IA-32) Instruction Set Architecture which comprises a CISC instruction set Intel developed a modus in which these instructions are split in so-called micro operations (μ-ops) of fixed length that can be treated in the way RISC processors do. In fact the μ-ops constitute a RISC operation set. The price to be payed for this much more efficient instruction set is an extra decoding stage.
Many of the improvements of the Core architecture are not evident from the block diagram. For instance in the Core architecture 4 μ-ops/cycle can be scheduled instead of 3 as in the former microarchitecture. Futhermore, some macro-instructions as well as some μ-ops can be fused, resulting in less instruction handling, easier scheduling and better instruction throughput because these fused operations can be executed in a single cycle.

As can be seen in Figure 2.11 the processor cores have an execution trace cache which holds partly decoded instructions of former execution traces that can be drawn upon, thus foregoing the instruction decode phase that might produce holes in the instruction pipeline. The allocator dispatches the decoded instructions, the μ-ops, to the appropriate μ-op queue, one for memory operations, another for integer and floating-point operations.
Two integer Arithmetic/Logical Units are kept simple in order to be able to run them at twice the clock speed. In addition there is an ALU for complex integer operations that cannot be executed within one cycle. The floating-point units contain also additional units that execute the Streaming SIMD Extensions 2 and 3 (SSE2/3) repertoire of instructions, a 144-member instruction set, that is especially meant for vector-oriented operations like in multimedia, and 3-D visualisation applications but which will also be of advantage for regular vector operations as occur in dense linear algebra. The length of the operands for these units is 128 bits. The throughput of these SIMD units has been increased by a factor of 2 in the Core architecture which greatly increase the performance of the appropriate instructions. The Intel compilers have the ability to address the SSE2/3 units. This makes it in principle possible to achieve a 2–3 times higher floating-point performance.
The Xeons boast so-called Hyperthreading: with the processor two threads can run concurrently under some circumstances. In this it is not unique anymore as all main processor makers now provide some form of multi-threading. It may for instance be used for speculative execution of `if` branches. Experiments have shown that up to 30% performance improvements can be attained for a variety of codes. In practice the performance gain about 3–5%, however.
The secondary cache has a size of 4 MB for the Woodcrest implementation of the Core processor. The two core share a forntside bus with a bandwidth of 10.6 GB/s.
Since its predecessor, the Nocona processor the Intel processors have the ability to run (and address) 64-bit codes, thereby following AMD, in fact copying the approach used in the AMD Opteron and Athlon processors. The technique is called Extended Memory 64 Technology (EM64T) by Intel. In principle it uses "unused bits" from in the instruction words of the x86 instruction set to signal whether an 64-bit version of an instruction should be executed. Of course some additional devices are needed for operating in 64-bit mode. These include 8 new general purpose registers(GPRs), 8 new registers for SSE2/3 support, and 64-bit wide GPRs and instruction pointers.
As in the dual-core Montecito (see 2.8.4) the two cores need an arbiter on the chip to provide fair bandwidth utilisation between the them. . Figure 2.10.
It will depend heavily on the quality of the compilers whether they will be able to take advantage of all the facilities present in the dual-core processor.

2.8.6 The SPARC processors

The situation with respect to the SPARC processors is rather unclear. Sun markets systems that it positions in the HPC market based on the UltraSPARC4+. Sun has shelved its own plans to produce UltraSPARC V and VI processor by April 2004 in favour of processor designs with many (≥ 8) processor cores, each capable of handling several execution threads. This so-called Rock processor is still some the SPARC development in the hands of its partner Fujitsu that will advance with its own SPARC64 implementation. Fujitsu markets a server based on the latter processor but at the same time also sells a newer, similar system based on the Itanium-2 processor. So, it is far from sure what the future of the various SPARC implementations will be. Nevertheless, systems with these processors are still here and therefore we discuss them in this section.

2.8.6.1 The UltraSPARC IV+

The UltraSPARC IV+ is the fifth generation of the UltraSPARC family. Like virtually all processor makers also Sun has put two processor cores on a chip from the UltraSPARC IV on. The CPU cores in the UltraSPARC IV were in fact slightly modified UltraSPARC III processors. The UltraSPARC IV+ is a technology shrink of the UltraSPAR IV fabricated with a 90 nm feature size. We show a block diagram of the processor core and its embedding in the UltraSPARC IV+ chip in Figure 2.12.
The processor is characterised by large large amount of caches of various sorts as can be seen in Figure 2.12a. The Data Cache Unit (DCU) contains apart from a 4-way set associative cache of 64 KB also a write and a prefetch cache, both of 2 KB. All these L1 caches operate at half speed: loads and stores from the processor

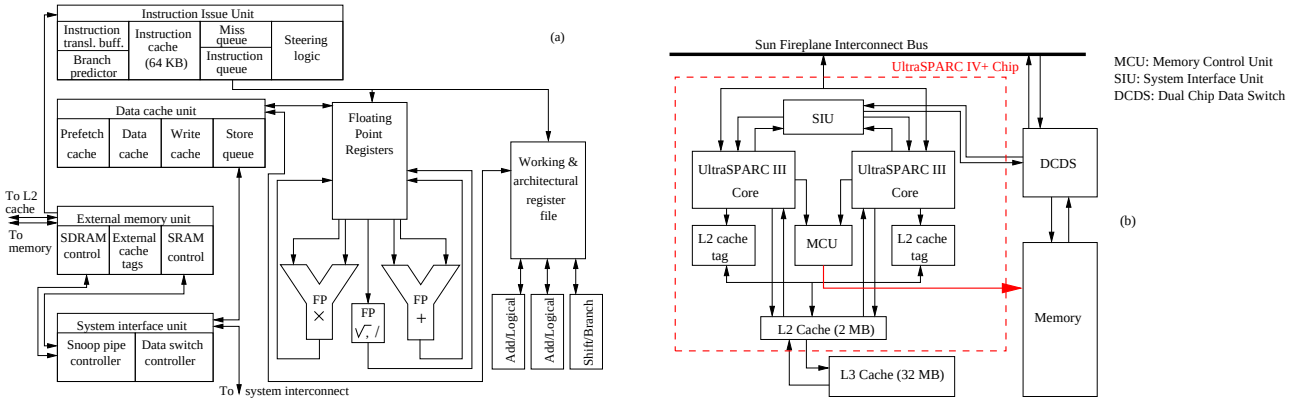


Figure 2.12: *Block diagram of the UltraSPARC IV+ processor core and chip layout.*

can be done in 2 cycles. The prefetch cache is independent from the data cache and can load data when this is deemed appropriate. The write cache defers writes to the L2 cache and so may evade unnecessary writes of individual bytes until entire cache lines have to be updated. The Instruction Issue Unit (IIU) contains a 64 KB 4-way set associative instruction cache together with the instruction TLB which is called Instruction translation buffer in Sun's terminology. The size of the instruction cache could be doubled thanks to the technology shrink in comparison with the UltraSPARC III and IV. In addition a 2 MB L2 cache and an L3 Tag Cache could be placed on the chip while a 32 MB L3 cache was added off chip. The IIU also contains a so-called miss queue that holds instructions that are immediately available for the execute units when a branch has been mis-predicted. Branch prediction is fully static in the UltraSPARC-III. It is implemented as a 16 KB table in the IIU that is pipelined because of its size.

The Integer Execute Unit (IEU) has two Add/Logical Units and a branch unit. Integer adds and multiplies are pipelined but the divide operation is not. It is performed by an Arithmetic Special Unit (not shown in the figure) that does not burden the pipelines for the ALUs. The integer register file is effectively divided in two and is called the Working and Architectural Register File by SUN. Operands are accessed and results stored in the working registers. When an exception occurs, the results to be undone in the working registers are overwritten by those from the architectural file. One of the enhancements with respect to the original UltraSPARC III design is the adding of hash indexing for the write cache. This should decrease the number of write misses and thus leave more write store bandwidth for results that need storing.

The floating-point unit (FPU) has two independent pipelined units for addition and multiplication and a non-pipelined unit for floating division and square-root computation that require in the order of 20–25 cycles. The FPU also contains graphics hardware (not shown in Figure 2.12) that shares the pipelined adder and multiplier with general 64-bit calculations. For the chips delivered at 1.5 GHz, the theoretical peak performance is 3.0 Gflop/s per processor core. It is expected that the UltraSPARC IV+ technology can be shrunk to reach a clock frequency that is slightly more by the end of its life cycle. In the UltraSPARC IV+ the FPUs are enhanced by adding hardware support of handling for IEEE 754 floating-point errors (which can be very costly otherwise when properly handled).

As is evident from Figure 2.12b the Memory Control Unit (MCU) is on chip as well as the L3 cache controller (in the MCU) and the L3 cache tags. This shortens the latency of accesses from all memory levels. In addition, both controllers communicate with the System Interface Unit (SIU), also on-chip to keep in touch with the snoop pipe controller in the SIU. The processor has been built with multi-processing in mind and the snoop controller keeps track of data requests in the whole system to ensure coherency of the caches when required.

The UltraSPARC IV+ is around since half 2005. Sun refers to having the two processor cores on a chip and running one execution thread on each of them as Chip Multithreading (CMT). This is not quite what one would normally would understand as multi-threading because one would then expect more execution threads per processor core. So, the CMT terminology is somewhat confusing and one would hope that Sun will drop it in favour of the common use of the term.

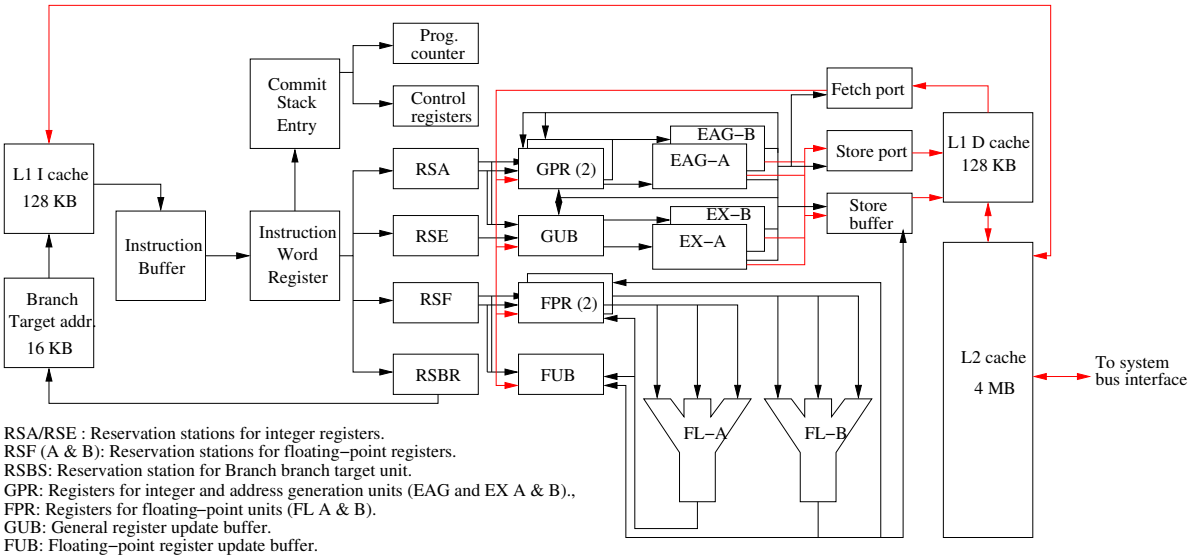


Figure 2.13: *Block diagram of the Fujitsu SPARC64 V processor.*

2.8.6.2 SPARC64

For quite some time Fujitsu is making its own SPARC implementation, called SPARC64. Presently the SPARC64 is in its fifth generation, the SPARC64 V. Obviously, the processor must be able to execute the SPARC instruction set but the processor internals are rather different from Sun's implementation. Figure 2.13 shows a block diagram of the SPARC64 V.

Notwithstanding the mutual compatibility of the different SPARC implememtations, there is quite some difference in the actual realisation of the processors as can be seen by comparing the SPARC IV+ processor core and the SPARC64 V diagrams in Figures 2.12 and 2.13, respectively. The L1 instruction and data caches are 128 KB, two times larger than in the SPARC4+ core and both 2-way set-associative. There is also an Instruction Buffer (IBF) than contains up to 48 4-byte instructions and continues to feed the registers through the Instruction Word Register when an L1 I-cache miss has occurred. A maximum of four instructions can be scheduled each cycle and find their way via the reservation stations for address generation (RSA), integer execution units (RSE), and floating-point units (RSF) to the registers. The two general register files serve both the two Address Generation units EAG-A, and -B and the Integer Execution units EX-A and -B. The latter two are not equivalent: only EX-A can execute multiply and divide instructions. There also two floating-point register files (FPR), that feed the two Floating-Point units FL-A and FL-B. These units are different from those of Sun in that they are able to execute fused multiply-add instructions as is also the case in the POWER and Itanium processors. Consequently, a maximum of 4 floating-point results/cycle can be generated. In addition, FL-A an -B also perform divide and square root operations in contrast to the SPARC4+ that has a separate unit for these operations. Because of there iterative nature the divide and square root operations are not pipelined. The feedback from the execution units to the registers is decoupled by update buffers: GUB for the general registers and FUB for the floating-point registers.

The dispatch of instructions via the reservation stations,that each can hold 10 instructions, gives the opportunity of speculative dispatch: i.e., dispatching instructions of which the operands are not yet ready at the moment of dispatch but will be by the time that the instruction is actually executed. The assumption is that it results in a more even flow of instructions to the execution units.

The SPARC64 V does not have a third level cache but on chip there is a large (4 MB) unified L2 cache that is a 4-way set-associative write-through cache. Furthermore, the Memory Management Unit (not shown in Figure 2.13) contains separate sets of Translation Look aside Buffers (TLB) for instructions and for data. Each set is composed of a 32-entry μTLB and a 1024-entry main TLB. The μTLBs are accessed by high-speed pipelines by their respective caches.

At this moment the highest clock frequency SPARC64 available is 2.08 GHz. As already remarked, the

floating-point units are capable of a fused multiply-add operation, like the POWER and Itanium processors, the peak floating-point and so the theoretical peak performance is presently 8.32 Gflop/s. Fujitsu plans to bring out a dual core SPARC64 VI by the end of 2006 in which the core is essentially the same as in the SPARC64 V with a clock frequency of 2.4 GHz and for 2007 even a quad-core SPARC VII is scheduled. However, not much about its structure is known yet.

2.9 Networks

Fast interprocessor networks are together with fast processors the decisive factors for both good integrated parallel systems and clusters. In the early days of clusters the interprocessor communication, and hence the scalability of applications, was hampered by the high latency and the lack of bandwidth of the network (mostly Ethernet) that was used. This situation has changed very much and to give a balanced view of the possibilities opened by the improved networks a discussion of some of these networks is in order. The more so as some of these networks are, or have been employed also in "integrated" parallel systems.
Of course Gigabit Ethernet is now amply available and with a maximum theoretical bandwidth of 125 MB/s would be able to fulfill a useful role for some applications that are not latency-bound in any way. But with latencies in the order of somewhat less than 50 μs and in-switch latencies of 35–45 μs the applicability is too restricted to be the network of choice (except perhaps for price reasons).
We restrict ourselves here to networks that are independently marketed as the proprietary networks for systems like those of IBM and SGI are discussed together with the systems in which they are incorporated. We do not pretend to be complete because in this new field players enter and leave the scene at a high rate. Rather we present main developments which one is likely to meet when one scans the high-performance computing arena.

A complication with the fast networks offered for clusters is the connection with the nodes. Where in integrated parallel machines the access to the nodes is customised and can be made such that the bandwidth of the network matches the internal bandwidth in a node, in clusters one has to make do with the PCI bus connection that comes with the PC-based node. The type of PCI bus which ranges from 32-bit wide at 33 MHz to 64-bit wide at 66 MHz determines how fast the data from the network can be shipped in and out the node and therefore the maximum bandwidth that can be attained in internode communication. In practice the available bandwidths are in the range 110–480 MB/s. Since 1999 PCI-X is available, initially at 1 GB/s, in PCI-X 2.0 also at 2 and 4 GB/s. Coupling with PCI-X is now common in PC nodes that are meant to be part of a cluster. Recently PCI Express became available. This provides a 200 MB/s bandwidth per data lane where 1×, 2×, 4×, 8×, 12×, 16×, and 32×multiple data lanes are supported: this makes it amply sufficient for the host bus adapters of any communication network vendor. So, for the networks discussed below often different bandwidths are quoted, depending on the PCI bus type and the supporting chip set which bandwidth could be attained. Therefore, when speeds are quoted it is always with the proviso that the PCI bus of the host node is sufficiently wide/fast.

2.9.1 Infiniband

Infiniband has become rapidly a widely accepted medium for internode networks. The specification was finished in June 2001. From 2002 on a number of vendors has started to offer their products based on the Infiniband standard. A very complete description (1200 pages) can be found in [35]. Infiniband is employed to connect various system components within a system. Via Host Channel Adapters (HCAs) the Infiniband fabric can be used for interprocessor networks, attaching I/O subsystems, or to multi-protocol switches like Gbit Ethernet switches, etc. Because of this versatility, the market is not limited just to the interprocessor network segment and so Infiniband is expected to become relatively inexpensive because a higher volume of sellings can be realised. The characteristics of Infiniband are rather nice: there are product definitions both for copper and glass fiber connections, switch and router properties are defined and for high bandwidth multiple connections can be employed. Also the way messages are broken up in packets and reassembled as well as routing, prioritising, and error handling are all described in the standard. This makes Infiniband independent of a particular technology and it is, because of its completeness, a good basis to implement a communication library (like MPI) on top of it.

Conceptually, Infiniband knows of two types of connectors to the system components, the Host Channel Adapters (HCAs), already mentioned, and Target Channel Adapters (TCAs). The latter are typically used to connect to I/O subsystems while HCAs does more concern us as these are the connectors used in interprocessor communication. Infiniband defines a basic link speed of 2.5 Gb/s (312.5 MB/s) but also a 4×and 12 ×speed of 1.25 GB/s and 3.75 GB/s, respectively. Also HCAs and TCAs can have multiple ports that are independent and allow for higher reliability and speed.

Messages can be sent on the basis of Remote Memory Direct Access (RDMA) from one HCA/TCA to another: a HCA/TCA is permitted to read/write the memory of another HCA/TCA. This enables very fast transfer once permission and a write/read location are given. A port together with its HCA/TCA provide a message with a 128-bit header which is IPv6 compliant and that is used to direct it to its destination via cut-through wormhole routing: In each switching stage the routing to the next stage is decoded and send on. Short messages of 32 B can be embedded in control messages which cuts down on the negotiation time for control messages.
Infiniband switches for HPC are offered with 8–288 ports and presently mostly at a speed of 1.25 GB/s. However, recently the first switches and HCAs accomodating double this speed have become available. Obviously, to take advantage of this speed at least PCI Express must be present at the nodes to which the HCAs are connected. The switches can be configured in any desired topology but in practice a fat tree topology is almost always preferred (see Figure 2.5b, section 2.5). It obviously depends on the quality of the MPI implementation put on top of the Infiniband specifications how much of the raw speed can be realised. A Ping-Pong experiments on Infiniband-based clusters with different MPI implementations has shown bandwidths of around 850 MB/s and an MPI latency of < 6 μs for small messages. The in-switch latency is typically about 200 ns. For the 2.5 GB/s products no reliable MPI bandwidth and latency data are available yet.
Because of the profusion of Infiniband vendors of late, the price is now at par with those of other fast network vendors like Myrinet (2.9.3) and Quadrics (2.9.4).

2.9.2 InfiniPath

InfiniPath only provides Host Channel Adapters with a 4-wide (1.25 GB/s) Infiniband link on the network side and connects to a HyperChannel bus or PCI-Express at the computer side. For systems with AMD processors on board the HyperChannel option is particularly attractive because of the direct connection to the host's processors. This results in very low latencies for small messages. PathScale, the vendor of the InfiniPath HCAs quotes latencies as low as 1.29 μs. Obviously, this type of HCA cannot be used with systems based on non-AMD processors. For these systems the HCAs with PCI-Express can be used. They have slightly higher, but still low latency of 1.6 μs. The effective bandwidth is also high: a uni-directional bandwidth of $\approx$ 950 MB/s can be obtained using MPI for both types of HCA.
The InifiniPath HBAs do not contain processing power themselves. Any processing associated with the communication is done by the host processor. According to PathScale this is an advantage because the host processor is usually much faster than the processors employed in switches. An evaluation report from Sandia National Lab [9] seems to corroborate this assertion.
PathScale only offers HCAs (and the software stack coming with it) and these can be used by any Infiniband switch vendor that adheres to the OpenIB protocol standard which are pretty much all of them.

2.9.3 Myrinet

Until recently Myrinet was the market leader in fast cluster networks and is still one of the largest. The Myricom company which sells Myrinet started in 1994 with its first Myrinet implementation, [28], as an alternative for Ethernet to connect the nodes in a cluster. Apart from the higher bandwidth, around 100 MB/s at that time, the main advantage was that it entirely operated in user space, thus avoiding Operating System interference and the delays that come with it. This meant that the latency for small messages was around 10–15 μs. Latency and bandwidth compared nicely with the proprietary networks of integrated parallel systems of Convex, IBM, and SGI at the time. Although such a network came at a non-negligible cost, in many cases it proved a valuable alternative to either an Ethernet connected system or an even costlier integrated parallel system.

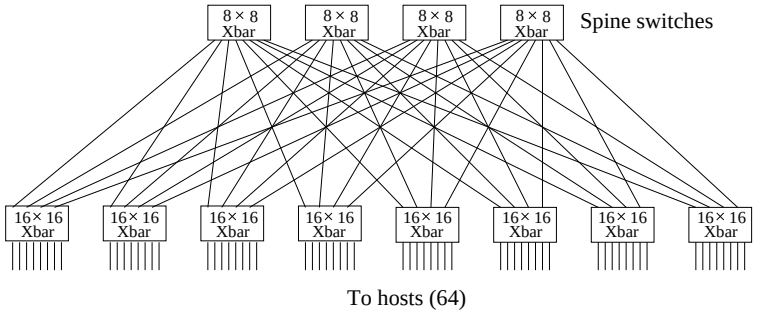


Figure 2.14: *An 8×16 Clos network using 8 and 16 port crossbar switches to connect 64 processors.*

Since then hardware upgrades and software improvements have made Myrinet the network of choice for many cluster builders and until recently there was hardly an alternative when a fast, low-latency network was required.

Like Infiniband, Myrinet uses cut-through routing for an efficient utilisation of the network. Also RDMA is used to write to/read from the remote memory of other host adapter cards, called Lanai cards. These cards interface with the PCI-X bus of the host they are attached to. The Myrinet allows copper cables or fibers as signal carriers. The latter form gives a high flexibility in the connection and much headroom in the speed of signals but the fiber cables and connectors are rather delicate which can lead to dammage when cluster nodes have to be serviced.
Myrinet offers ready-made 8–256 port switches (8–128 for its newest product, see below). The 8 and 16 port switches are full crossbars. In principle all larger networks are build form these using a Clos network topology. An example for a 64-port systems is shown in Figure 2.14. A Clos network is another example of a logarithmic network with the maximum bi-sectional bandwidth of the endpoints. Note that 4 ports of the 16×16 crossbar switches are unused but other configurations need either more switches or connections or both.
Since the start of 2006 Myricom provides, like many Infiniband switch vendors, a multi-protocol switch (and adapters): The Myri-10G. Apart from Myricom's own MX protocol it also supports 10 Gigabit Ethernet which makes it easy to connect to external nodes/clusters. An ideal starting point for building grids from a variety of systems. The specifications as given by Myricom are quite good: ≈ 1.2 GB/s for the uni-directional theoretical bandwidth for both its MX protocol and about the same for the MX emulation of TCP/IP on Gigabit Ethernet. According to Myricom there is no difference in bandwidth between MX and MPI and also the latencies are claimed to be the same: 2 μs .

2.9.4 QsNet

QsNet is the main (and only) product of Quadrics, a company that initially started in the 1990s as the British firm Meiko that made parallel systems called the Computer Surface 1 and 2. In the CS-2 a very good logarithmic network was included that was made in an separate product after Meiko closed down and the network part was taken over by the Italian company Alenia and put into the independent company Quadrics. The success came when Digital/Compaq chose QsNet as the network for its large high-performance computer products, the AlphaServer SC line. Some very large configurations of these machines were sold, e.g., at the Pittsburgh Supercomputing Centre and the French CAE. QsNet proved to be a very fast and reliable network and since about two years QsNet is also offered for cluster systems.

Like Infiniband and Myrinet the network has effectively two parts: the ELAN interface cards, comparable to Infiniband Host Bus Adapters or Myrinet's Lanai interface cards, and the Elite switch, comparable to an Infiniband switch/router or a Myrinet switch. The topology that is used is a (quaternary) fat tree like in most Infiniband switches, see Figure 2.5b, section 2.5 for an example. The ELAN card interfaces with the PCI-X port of a host computer.
Ready-made Elite switches for clusters come in two sizes: with 16 and 128 ports but nothing in between. Of course one can put together networks for other sizes but this goes at the cost of compactness and speed.

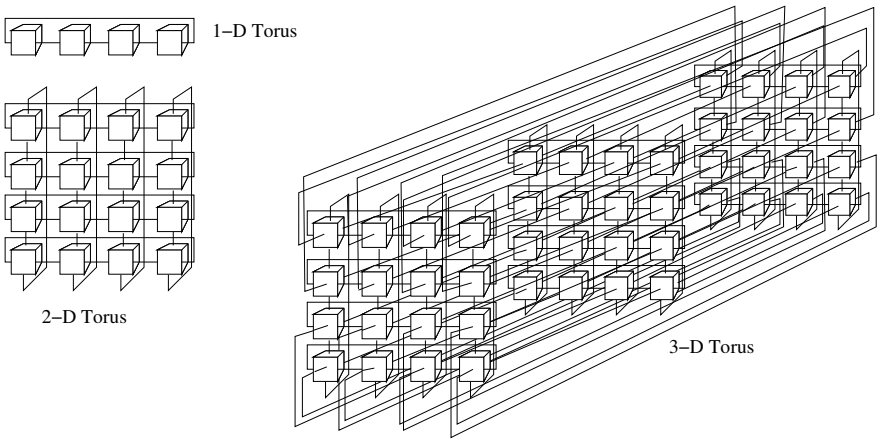


Figure 2.15: *SCI networks arranged as 1-D, 2-D, and 3-D toruses.*

A difference with the other switches lies in the providing *two* virtual bi-directional channels per link. Since the end of 2003 Quadrics sells its second generation QsNet, QsNet II. The structure, protocols, etc., are very similar to those of the former QsNet but much faster: where in [33] a speed of 300 MB/s for an MPI Ping-Pong experiment was measured while QsNet II has a link speed of 1.3 GB/s. In this case the PCI-X bus speed is a limiting factor: it allows for somewhat more than about 900 MB/s. However, with the advent of PCI Express this limitation may be over. Also the latency for short messages has improved from ≈ 5 μs to close to 1 μs. Furthermore, the switch supports two priority levels which greatly helps in a fair distribution of message packets. This is more than Myrinet provides but less than the 16 priority levels of Infiniband.

Like Infiniband, QsNet(II) has RDMA capabilities that allow to write/read from/to remote memory regions on the ELAN cards. This can however be extended to memory regions of the host processor itself. So, in principle, one would be able to view a QsNet-connected system as a virtual shared memory system. As yet, this has not been realised nor on integrated systems, nor on clusters. Nevertheless, it could be an attractive alternative for the much slower memory page-based virtual shared memory systems like TreadMarks [1]. A Cray-style `shmem` library is offered that enables one-sided communication via `put` and `get` operations as well as an MPI-2 implementation that supports one-sided communication calls.
Since early 2006 Quadrics also offers 10 Gbit Ethernet cards and switches under the name QSTenG. These products capitalise on the experience obtained in the development of the two generations of QsNet but they are not meant for inter-processor communication. As yet Quadrics does not seem to consider developing multi-protocol products like Myricom's Myri-10G and Infiniband.

2.9.5 SCI

SCI, standing for Scalable Coherent Interface, is the oldest of the networks discussed here. SCI became an IEEE/ANSI standard by October 1992. It was born as a reaction to the limitations that were encountered in bus-based multiprocessor systems in these days and a working group from various vendors and from universities tried to devise a solution that would do away with these limitations once and for all. A nice introduction into the rationale and the design choices for SCI can be found in [17]. Further discussions can be found in [23].
SCI was largely designed to avoid the usual bus limitations which resulted in a ring structure to connect the hosts. In fact pairs of rings are used to enabled reverse signaling between a send and receive host. This is because of one of the design principles of SCI: signals flow continuously in one direction enabling a very high signal rate without noise interference. A consequence is that the ring is always active, sending zero-length payload messages when no actual messages have to be transferred. The information length in a packet has a fixed sizes of 0, 16, 64, and 256 bytes and a header of 16 bytes or 32 bytes while the packet is closed by a 2-byte error correcting CRC code. By having the error correcting code at the end of the packet it is immediately known whether the data are corrupted or not and immediate action can be taken. The limited packet format enables fast reception and checking of the packets.

A special feature of SCI is its ability keep the caches of the processors it connects to coherent. This is also

a consequence of its design history in the sense that SCI should be able to replace buses in multi-processor systems where such buses via a snoopy-bus protocol (see section 2.6) should keep the processor caches up-to-date or *coherent*. So, like with QsNet, one can use it to implement virtual shared memory. Unlike with QsNet, this is (have been) indeed been done: the late Convex Exemplar systems used SCI to connect its nodes, as did Data General. Presently the NUMA-Q systems of former Sequent, now IBM, use it as a memory interconnect medium. In clusters SCI is always used as just an internode network. Because of the ring structure of SCI this means that the network is arranged as a 1-D, 2-D, or 3-D torus as shown in Figure 2.15.

The torus network has some risks in comparison to other networks like the fat tree or the Clos network: when a node adapter fails it incapacitates all the nodes that share the ring on which it lies. So, for instance, Dolphin networks, one of the SCI vendors provides software to reconfigure the torus in such a way that the minimal number of nodes become unavailable. Furthermore, it is not possible to add or remove an arbitrary number of nodes in the cluster because of the torus topology.
Bandwidths are reported for the SCI-based clusters: up to about 320 MB/s for a Ping-Pong experiment over MPI with very low latencies of 1–2 μs for small messages.

3 Recount of (almost) available systems

In this section we give a recount of all types of systems as discussed in the former section and that are marketed presently or will appear within 6 months from now. When vendors market more than one type of machine we will discuss them in distinct subsections. So, for instance, we will discuss Cray systems under entries, X1E and XD1 because they have a very different structure.
A reservation with respect to the word "available" in the heading of this section is in order: rather *theoretically* available is a better description of the situation. This has nothing to do with the technical ability of the vendors to produce the systems described but everything with the ability of the vendor to invest in selling the system: when a large system is purchased this aught to come with the required maintenance and support. Some vendors cannot or will not sell a system in certain parts of the world because they are not willing or able to provide the necessary maintenance and/or support for the system that is theoretically available. For instance, it is not possible to buy a large Fujitsu(-Siemens) PRIMEPOWER or PRIMEQUEST or a Hitachi SR11000 or BladeSymphony in Europe. Nevertheless we still discuss these systems in the section below for general interest and for those that are located in a part of the world where these vendors deem the selling of their systems economically viable.

3.1 System descriptions

The systems are presented alphabetically. The "Machine type" entry shortly characterises the type of system as discussed in the former chapter: Processor Array, ccNUMA, etc.

3.1.1 The Bull NovaScale.

Machine type: ccNUMA system.
Models: NovaScale 5165, 5325.
Operating system: Linux, WindowsServer 2003, GCOS 8
Connection structure: Full crossbar.
Compilers: Intel's Fortran 95, C(++).
Vendors information Web page: `www.bull.com/novascale/`
Year of introduction: 2005.

System parameters:

Model	NovaScale 5165	NovaScale 5325
Clock cycle	1.6 GHz	1.6 GHz
Theor. peak performance	102.4 Gflop/s	204.8 Gflop/s
Main Memory	8–256 GB	16–512 MB
No. of processors	4–16	8–32
Communication bandwidth		
Point-to-point	6.4 GB	6.4 GB
Aggregate peak	25.6 GB	25.6 GB

Remarks:

The NovaScale 5005 series is the second generation of Itanium-2 based systems targetting the HPC field. Besides the two models listed under System Parameters it also includes the 5085 and 5245 models which we

do not discuss separately as they are simply models with a maximum of 8 and 24 processors, respectively. The main difference with the first generation, the 5160 and 5230 systems, is the doubling of the density: where the 5230 had to be housed in two 40 U racks, a 5325 systems fit in one rack and the 5165 can be housed in a 19 U rack. In about all other regards the new series is equal to the first generation.
The NovaScales are therefore ccNUMA SMPs They are built from standard Intel Quad Building Blocks (QBBs) each housing 4 Itanium 2 processors and a part of the memory. The QBBs in turn are connected by Bull's proprietary FAME Scalability Switch (FSS) providing an aggregate bandwidth of 25.6 GB. For reliability reasons a NovaScale 5165 is equipped with 2 FSSes. This ensures that when any link between a QBB and a switch or between switches fails the system is still operational, be it on a lower communication performance level. As each FSS has 8 ports and only 6 of these are occupied within a 5165 system, the remaining ports can be used to couple two of these systems thus making a 32-processor ccNUMa system. Larger configurations can be made by coupling systems via QsNet II (see 2.9.4). Bull provides its own MPI implementation which turns out to be very efficient (see "Measured Performances" below and [43]).
A nice feature of the NovaScale systems is that they can be partitioned such that different nodes can run different operating systems and that repartitioning can be done dynamically. Although this is not particularly enticing for HPC users, it might be interesting for other markets, especially as Bull still has clients that use their proprietary GCOS operating system.
The documentation from Bull states that the systems are "Montecito ready" which is true in the sense that the same socket can be used for the Itanium 2 and its successor, the dual-core Montecito (see 2.8.4). And in fact, Montecito-based systems are expected within a few months (around July, August 2006). There is a proviso with respect to replacement of Itanium 2 processors by Montcitos: the latter are because of their dual cores obviously more bandwidth-hungry than the Itanium 2 processors. Because of that Montecitos with a Front Side Bus (FSB) of 10.6 GB/s will become available. However, when one simply exchanges processors one cannot use these higher bandwidth chips because the original E8870 chipset within the systems has a maximum bandwidth of 6.4 GB/s. Therefore, the benefit for bandwidth-limited application would also be limited in this case.
At the time of writing this report Bull has introduced a new server series, the NovaScale 3005. The largest one, the 3405, contains 4 Montecito processors and as such due to it peak of 49.2 Gflop/s has no place in this report. However, the form factor makes it very compact and very fit to be used in heavy-node clusters. We thought it to be of sufficient interest to mention it here.

Measured Performances:
In the spring of 2004 rather extensive benchmark experiments with the EuroBen Benchmark were performed on a 16-processor NovaScale 5160 with the 1.3 GHz variant of the processor. Using the EuroBen benchmark, the MPI version of a dense matrix-vector multiply was found to be 13.3 Gflop/s on 16 processors while both for solving a dense linear system of size N =1,000 and a 1-D FFT of size N =65,356 speeds of 3.3–3.4 Gflop/s are observed (see [43]).
For the recently installed Tera-10 system at CEA, France, a Linpack performance of 42,900 Gflop/s out of 55,705.6 Gflop/s installed is reported. An efficiency of 77% on a linear system of unknown rank ([45]).

3.1.2 The C-DAC PARAM Padma.

Machine type: RISC-based Distributed-memory multi-processor.
Models: C-DAC PARAM Padma.
Operating system: AIX, Linux.
Connection structure: Clos network.
Compilers: Fortran 90, C, C++.
Vendors information Web page: `www.cdac.in/html/parampma.asp`
Year of introduction: 2003.

System parameters:

Model	PARAM Padma
Clock cycle	1 GHz
Theor. peak performance	
Per Proc.	4 Gflop/s
Maximal	992 Gflop/s
Memory	500 GB
No. of processors	248
Communication bandwidth	
Aggregate	4 GB/s
Point-to-point	312 MB/s
Full duplex	235 MB/s

Remarks:
The PARAM Padma is the newest systems made by the Indian C-DAC. It is built somewhat asymmetrically from 54 4-processor SMPs and 1 32-processor node. All nodes employ 1 GHz IBM POWER4 processors. As an interconnection network C-DACs own PARAMnet-II is used for which a peak bandwidth of 2.5 Gb/s (312 MB/s) is given with a latency for short messages of $\approx 10\mu$s. The network is build from 16-port PARAMnet-II switches and has a Clos64 topology, very similar to the structure used by Myrinet (see 2.9.3). No MPI results over this network are available.
C-DAC has already a long tradition of building parallel machines and it has always provided its own software to go with them. Therefore, the Padma comes with Fortran 90, C(++), MPI, and a Parallel File System.

Measured Performances:
The Padma performs at 532 Gflop/s with the HPC Linpack Benchmark ([45]) for a linear system of size N =224,000 on a 62-node machine with a theoretical peak of 992 Gflop/s. That amounts to an efficiency of 53.6% for this benchmark.

3.1.3 The Cray Inc. X1E

Machine type: Shared-memory multi-vectorprocessor.
Models: Cray X1E (cluster).
Operating system: UNICOS/mp (Cray Unix variant).
bf Connection structure: Modified 2-D torus (see remarks).
Compilers: Fortran 95, C, C++, Co-Array Fortran, UPC.
Vendors information Web page: `www.cray.com/products/x1e/index.html`
Year of introduction: 2004.

System parameters:

Model	Cray X1E AC	Cray X1E LC	Cray X1E MF
Clock cycle	1.125 GHz	1.125 GHz	1.125 GHz
Theor. peak performance			
Per Proc.	4.5/18 Gflop/s	4.5/18 Gflop/s	4.5/18 Gflop/s
Frames	1 air-cooled	1 liquid-cooled	≤64 liquid-cooled
Maximal	576 Gflop/s	2.3 Tflop/s	147.2 Tflop/s
Memory	≤128 GB	≤512 GB	≤32 TB
No. of processors	≤32 (MSP, see below)	≤128	≤8192
Memory bandwidth	34.1 GB/s	34.1 GB/s	34.1 GB/s
Cache–CPU	76.8 GB/s	76.8 GB/s	76.8 GB/s
Communication bandwidth			
Maximum aggregate	816 GB/s	3.2 TB/s	204.8 TB/s

Remarks:
Each processor board of a X1E contains 4 CPUs that can deliver a peak rate of 4 floating-point operations per cycle, amounting to a theoretical peak performance of 4.5 Gflop/s per CPU. However, 4 CPUs can be

coupled *across* CPU boards in a configuration to form a so-called Multi Streaming Processor (MSP) resulting in a processing unit that has effectively a theoretical peak performance of 18 Gflop/s. The reconfiguration into MSPs and/or single CPU combinations can be done dynamically as the workload dictates. The vector start-up time for the single CPUs is smaller than for MSPs, so for small vectors single CPUs might be preferable while for programs containing long vectors the MSPs should be of advantage. MSP mode is regarded as the standard mode of operation. This is also visible in the processor count given in the data sheets of Cray: the maximum number within one frame is 32 MSP processors for the air-cooled model while it is 128 MSPs in the liquid-cooled variant. In the present Cray optimisation documentation it is said that the Cray Programming Environment is as yet not optimised for SSP processing which is not to say that suitable programs would not run efficiently in SSP mode.
The logical hardware structure of the Cray X1E is largely identical to that of the former Cray X1 (see 4.1). However, technology improvements made it possible to increase the density twofold by placing two MSPs on one multi-chip module (MCM). Four of these MCMs are located in one X1E Compute Module comprised of two (logical) 4-MSP nodes. The 8 MCMs are connected to routing logic in the Compute Module which contains 32 network ports to connect it to other Compute Modules. The clock frequency of the processors was raised from 800 MHz in the X1 to 1.125 GHz in the X1E and the amount of memory and the bandwidth to the processors is increased.
The relative bandwidth both from memory to the CPU boards and from the cache to the CPUs has improved in comparison to the predecessor X1: from memory to the CPU board 5.3 8-byte operands can be transferred. From the cache the peak bandwidth to the CPUs is 12 8-byte operands, enough to sustain dyadic operations. The cache structure is rather complex: each of the 4 SSPs on a board have their own 16 KB 2-way set-associative L1 data and instruction cache. The L1 data cache only stores scalar data. The L2 cache is 2 MB in size and is shared by the SSP processors on the CPU board.
New features that are less visible to the user are adherence to the IEEE 754 floating-point standard arithmetic and a new vector instruction set that can make better use of the new features like caches and addressability and synchronisation of remote nodes. This is because every cabinet can be regarded as a node in a cluster of which a maximum of 64 can be configured in what is called a modified 2-D torus topology. Cray itself regards a board with 4 MSPs as a "node". Each node has two connections to the outside world. Odd and even nodes are connected in pairs and the other connection from the board is connected via a switch to the other boards. Thus requiring at most two hops to reach any other MSP in the cabinet. The aggregate bandwidth in such a fully populated cabinet is 400 GB/s. Multi-cabinet configurations are extending the 2-D torus into a 3-D torus structure much like the late Cray T3E 4.1 Latency and bandwidth data for point-to-point communication are not provided but various measurements in an MPI environment have been done, see the Measured Performances below.
On a 4-SSP CPU board `OpenMP` can be employed. When accessing other CPU boards one can use Cray's `shmem` library for one-sided communication, MPI, Co-Array Fortran, etc.

Measured Performances:
In [45] a speed of 15,706 Gflop/s is reported for solving a 494,592-order linear system on a 1020-(MSP)processor machine at the Korea Meteorological Administration. This amounts to an efficiency of 85%.
A more extensive evaluation of 504-MSP system at ORNL system is reported in [12]. Here a point-to-point bandwidth of 13.9 GB/s was found within a 4-MSP node and 11.9 GB/s between nodes. MPI latencies for small messages were 8.2 and 8.6 μs , respectively. For `shmem` and Co-Array Fortran the latencies were only 3.8 and 3.0 μs .

3.1.4 The Cray Inc. XD1

Machine type: Distributed-memory multi-processor.
Models: XD1.
Operating system:Linux (kernel 2.4.21 with Cray HPC enhancements).
bf Connection structure: Variable (see remarks).
Compilers: Fortran 95, C, C++.
Vendors information Web page: `www.cray.com/products/xd1/`
Year of introduction: 2004.

System parameters:

Model	Cray XD1
Clock cycle	2.2 GHz
Theor. peak performance	
Per Chassis	52.8+ Gflop/s (see remarks)
Per Rack	663+ Gflop/s (see remarks)
Memory	
Per Chassis	96 GB
Per Rack	1.2 TB
No. of processors	
Per Chassis	12
Per Rack	144
Communication bandwidth	
Point-to-point	$\leq$2.9 GB/s
Aggregate, per chassis	96 GB/s

Remarks:
The Cray XD1 is a product that was originally developed by Octigabay until this company was taken over by Cray. A distinctive factor in the Octigabay systems was the possibility to add FPGAs (see Glossary) to the compute boards of the systems to accelerate algorithms that are of special interest to the user, like massive FFTs or DNA sequence alignments. Hence the plus symbols in the entries for the Theorectical Peak Performance in the System Parameters list above. Cray turned the system into a product by adding its special communication networking capability to connect the compute boards and the nodes, called "chassis" by Cray by means of its proprietary Rapid Array Network.
The general structure of an XD1 is as follows: one chassis houses up to 6 compute cards. Each compute card has 2 AMD Opterons at 2.2 GHz and one or two RapidArray Processors (RAPs) that handle the communication. The two Opterons on a card are connected via AMD's HyperTransport with a bandwidth of 3.2 GB/s forming a 2-way SMP. Because of the high bandwidth of the HyperTransport bus the memory access does not suffer from using two processors on a board, unlike in most 2 processor/node clusters. Optionally an application acceleration processor (FPGA) can be put onto a compute board. With 2 RAPs/board a bandwidth of 8 GB/s (4 GB/s bi-directional) between boards is available via a RapidArray switch. This switch has 48 links of which half is used to connect to the RAPs on the compute boards within the chassis and the others can be used to connect to other chassis. Twelve chassis fit into a standard rack and because of the number of free links per RapidArray switch the chassis in two racks may be connected directly. Of course larger configurations can be put together by connecting the links in a more sparsely connected network, like a 3-D torus or a fat tree.
The RAPs offload the Opteron processors from communication tasks and have hardware support for MPI, Cray-style `shmem`, and Global Arrays (a virtual shared memory system). The communication characteristics for MPI via the RapidArray network as stated by Cray are impressive: 2.9 GB/s for long messages and a 1.6 μs latency for small messages.
An extra feature of the Cray-enhanced Linux OS is the synchronisation of tasks in the system. The random scheduling of tasks within the system (by the OS or otherwise) can result in large latencies (see [32]) that may be detrimental to the MPI performance. By task synchronisation this problem can be evaded.

Measured Performances:

An evaluation of a 6-chassis test has been done at ORNL. See the website of Tom Dunigan [13]. The largest XD1 configuration today is that at the Naval Research Lab in the USA which uses a 144-chasis (864-processor) XD1. In [45] a Linpack performance of 3,041 out of 3,801.6 Gflop/s was reported; An efficiency of 80.0%.

3.1.5 The Cray Inc. XT3

Machine type: Distributed-memory multi-processor.
Models: XT3.

Operating system: UNICOS/lc, Cray's microkernel Unix.
bf Connection structure: 3-D Torus.
Compilers: Fortran 95, C, C++.
Vendors information Web page: `www.cray.com/products/xt3/`
Year of introduction: 2004.

System parameters:

Model	Cray XT3
Clock cycle	2.4 GHz
Theor. peak performance	
Per processor	4.8 Gflop/s
Per Cabinet	460.8 Gflop/s
Max. Configuration	147 Tflop/s
Memory	
Per Cabinet	$\leq$ 768 GB
Max. Configuration	196 TB
No. of processors	
Per Cabinet	96
Max. Configuration	30,508
Communication bandwidth	
Point-to-point	$\leq$7.6 GB/s
Bisectional/cabinet	333 GB/s

Remarks:
The Cray XT3 is the commercial spinoff of the 10,000+ processor Red Storm machine, built by Cray for Sandia Laboratories. The structure is similar, be it that there are no provisions are made to have a "classified" and an "unclassified" part in the machine. The basic processor in a node, called PE (Processing Element) in Cray jargon, is the AMD Opteron 100, at 2.4 GHz. Cray has chosen for this uniprocessor version of the chip because of the lower memory latency (about 60 ns) in contrast to the SMP-enabled versions that have a memory latency that can be up to 2 times higher. Per PE up to 8 GB of memory can be configured, connected by a 6.4 HyperTransport to the processor. For connection to the outside world a PE harbours 2 PCI-X busses, a dual-ported FiberChannel Host Bus Adaptor for connecting to disk, and a 10 GB Ethernet card.
The Opteron was also chosen because of the high bandwidth the relatively ease of connecting the processor of to the network processor, Cray's SeaStar chip. For the physical connection another HyperTransport channel at 6.4 GB/s is used. The SeaStar has 6 ports with a bandwidth of 7.6 GB/s each (3.8 GB/s, incoming and outgoing). Because of its 6 ports the natural interconnection mode is therefore a 3-D torus.
Like for the earlier Cray T3E (see 4), Cray has chosen to use a microkernel approach for the compute PEs. These are dedicated to computation and communication and are not disturbed by other OS tasks that can seriously influence the scalability (see [32]). For tasks like communicating with users, networking, and I/O special PEs are added that have versions of the OS that can handle these tasks.
The XT3 is obviously designed for a distributed memory parallel model, supporting Cray's MPI 2.0 and its one-way communication `shmem` library that date back to the Cray T3D/T3E systems but is still popular because of its simplicity and efficiency. The system comes in cabinets of 96 PEs, including service PEs. For larger configurations the ratio of service PEs to compute PEs (generally) can be lowered. So, a hypothetical maximal configuration of 30,508 PEs would need only 106 service PEs.

Measured Performances:
The Red Storm machine at Sandia National Lab, USA, can be regarded as a prototype for the XT3 systems, be it at a clock frequency of 2.0 GHz instead of 2.4 GHz. In [45] a speed of 36,190 out of 43,520 Gflop/s is reported: an efficiency of 83%. ORNL in the USA reports in [45] a performance of 20,527 Gflop/s on a 5200-processor regular XT3 for a linear system of unknown size. The efficiency is 82.2% in this case.

3.1.6 The Fujitsu/Siemens PRIMEPOWER.

Machine type: RISC-based shared-memory multi-processor.
Models: PRIMEPOWER 900, 1500, 2500.
Operating system: Solaris (Sun's Unix variant).
Connection structure: Crossbar.
Compilers: Fortran 90, OpenMP, C, C++.
Vendors information Web page: `www.fujitsu.com/services/computing/server/unix/`
Year of introduction: 2002.

System parameters:

Model	PRIMEPOWER 1500	PRIMEPOWER 2500
Clock cycle	1.89 GHz	1.82 GHz
Theor. peak performance		
Per Proc. (64-bits)	3.78 Gflop/s	3.64 Gflop/s
Maximal	121.0 Gflop/s	466 Gflop/s
Memory/node	$\leq$ 4 GB	$\leq$ 4 GB
Memory/maximal	$\leq$ 128 GB	$\leq$ 512 GB
No. of processors	4–32	8–128
Communication bandwidth		
Point-to-point	—	—
Aggregate	17.2 GB/s	133 GB/s

Remarks:
We only discuss here the PRIMEPOWER 1500 and 2500 as the smaller models have the same structure but less processors (maximally 16 in the 900 model). In many respects this machine is akin to the SUN Fire E25K, section 3.1.16. The processors are 64-bit Fujitsu implementations of SUN's SPARC processors, called SPARC64 V processors and they are completely compatible with the SUN products. Fujitsu's SPARC64 processors have extensive RAS features for all CPU components, caches, registers, execution units, etc., that make them very reliable. The processors are available in a 1.89 GHz and 1.82 GHz for the 1500 and 2500 models respectively. Also the interconnection of the processors in the PRIMEPOWER systems is like the one in the Fire E25K: a crossbar that connects all processors at the same footing, i.e., it is *not* a NUMA machine. In fact, in the PRIMEPOWER 2500 the communication is maintained by two crossbars each at a speed of 66.5 GB/s. If one of the crossbars fails it is possible by reconfiguration to operate at half the speed. Unfortunately, there is no sound technical information available beyond the data sheets that are provided via Fujitsu's web site. These data sheets omit almost all information about the bandwidth of the interconnect be it point-to-point, bi-sectional, or aggregate. Only for the PRIMEPOWER 2500 an aggregate bandwidth is quoted: a very respectable 133 GB/s stemming from the two 66.5 GB/s crossbars. Judging from the available information the 1500 system is more positioned as a commercial server than as a high-performance computer while the 2500 model is also targeted at the HPC market. Fujitsu offers the possibility to link many PRIMEPOWER 2500s together by a very fast (4×4 GB/s) optical link to build very large configurations.

Note: Large HPC configurations of the PRIMEPOWER are not sold in Europe as they are judged to be of insufficient economical interest by Fujitsu-Siemens.

Measured Performances:
In [45] a cluster of 18 fully configured PRIMEPOWER 2500s with the older 1.3 GHz processors is used to solve a linear system of order $N = 658{,}800$. This yields a performance of 5.4 Tflop/s with an efficiency of 45% on 2,304 processors.

3.1.7 The Fujitsu/Siemens PRIMEQUEST 500.

Machine type: Shared-memory SMP system.
Models: PRIMEQUEST 520, 540, 580.
Operating system: Linux (RedHat EL4 or SuSE SLES 9/10).
Connection structure: Crossbar.

Compilers: Fortran 90, OpenMP, C, C++. (Intel)
Vendors information Web page: www.fujitsu.com/global/services/computing/server/primequest/
Year of introduction: 2006.

System parameters:

Model	PRIMEQUEST 540	PRIMEQUEST 580
Clock cycle	1.6 GHz	1.6 GHz
Theor. peak performance		
Per Proc.core (64-bits)	6.4 Gflop/s	6.4 Gflop/s
Maximal	204.8 Gflop/s	409.6 Gflop/s
Memory/maximal	$\leq$ 1 TB	$\leq$ 2 TB
No. of processors	4–16	4–32
Communication bandwidth		
Point-to-point	—	—
Aggregate	68.2 GB/s	136.4 GB/s

Remarks:
We only discuss here the PRIMEPOWER 540 and 580 as the smaller model, the 520 has the same structure but less processors (maximally 4). The PRIMEQUEST is one of the many Itanium-based machines that are offerred these days. The older 400 series that is on the market for less than a year is now replaced by the 500 series that contains the dual-core Montecito variant, or Itanium 9000 as it is named officially. As the clock frequency is 1.6 GHz the peak performance is just over 200 Gflop/s for a model 540 and about 410 Gflop/s for a model 580.
The proprietary network to build clusters of the 540 or 580 nodes are not offerred unlike that is the case for the PRIMEPOWER systems (see 3.1.6) probably because the PRIMEQUESTs use a standard Linux distribution for an operating system instead of the Solaris variant that Fujitsu employs on the SPARC64-based PRIMEPOWER systems. So, when one (or Fujitsu) is prepared to build a large configuration based on the PRIMEQUEST one would have to use a third party network like Infiniband, Myrinet, or Quadrics.
Just like the PRIMEPOWER and the related SUN Sunfire systems, the PRIMEQUEST is not a cc-NUMA machine but a SMP system where all processors have equal access to the common and potentially large memory. Such a choice dictates a modest amount of processors (32 maximum here) and a high-speed crossbar. The latter requirement is fullfilled by a full crossbar with an aggregate bandwidth of 136.4 GB/s for the PRIMEQUEST 580.
Obviously, the compiler suite provided by Intel is used on the processors and therefore will there be virtually no difference between a single processor of the PRIMEQUEST and other Montecito-based systems. Only in parallel OpenMP and MPI programs the differences should show.
Fujitsu tries to stand out with respect to other vendors in offerring failsafe systems. For instance it is possible to have the crossbar doubled. Would one crossbar fail, the other would take over without interrupt of service thus securing the safe completion of the active tasks. Apart from the second crossbar also other components can be doubled, of course at a price, to make the system highly reliable.

Note: Large HPC configurations of the PRIMEQUEST are not sold in Europe as they are judged to be of insufficient economical interest by Fujitsu-Siemens.

Measured Performances:
The PRIMEQUEST is a very new system of which no indepedent performance measurements are available.

3.1.8 The Hitachi BladeSymphony.

Machine type: RISC-based distributed-memory multi-processor.
Models: BladeSymphony.
Operating system: Linux (RedHat EL4), MS Windows.
Connection structure: Fully connected SMP nodes (see remarks).
Compilers: Fortran 77, Fortran 95, Parallel Fortran, C, C++.
Vendors information Web page: www.hitachi.co.jp/products/bladesymphony_global/products03.html

Year of introduction: 2005.

System parameters:

Model	BladeSymphony
Clock cycle	1.66 GHz
Theor. peak performance	
Per processor core	6.64 Gflop/s
Per Frame of 64 proc.s	850 Gflop/s
Memory/frame	≤128 GB
No. of processors	4–64
Communication bandwidth	
Point-to-point	—

Remarks:
The Hitachi BladeSymphony is one of the many Itanium based parallel servers that are currently on the market. Still there are some differences with most other machines. First, a BladeSymphony frame can contain up to 4 modules which contain a maximum of 8 two-processor blades. The 16 processors in a module constitute an SMP node, like the nodes of the IBM eServer p-series (see 3.1.11). Four of such modules are housed in a frame and can communicate via a 4×4 crossbar. Unfortunately Hitachi nowhere mentions bandwidth data for the communication between modules nor within a module. Hitachi offers the blades with processors of various speeds. The fastest of these runs at 1.66 GHz from which it can be derived that the dual-core Montecito processor is used. This makes the Theoretical Peak speed for a 64-processor frame 850 Gflop/s.
Another distinctive feature of the BladeSymphony is that also blades with 2 Intel Xeon processors are offered. In this case, however, only 6 blades can be housed in a module. Theoretically modules with Itanium processors and Xeon processors can be mixed within a system, although in practice this will hardly occur.
Hitachi makes no mention of a connecting technology to cluster frames into larger systems but this can obviously been done with third party networks like Infiniband, Quadrics, etc. In all, there is the impression that Hitachi is hardly interested in marketing the system in the HPC area but rather for the high-end commercial server market.
Like the other Japanese vendors Hitachi (see 3.1.7 and 3.1.13) very much stresses the RAS features of the system. About all failing components may be replaced while the system is in operation which makes it very resilient against system-wide crashes.

Note: Large HPC configurations of the BladeSymphony are not sold in Europe as they are judged to be of insufficient economical interest by Hitachi.

Measured Performances:
The BladeSymphony was introduced in November 2005 and as yet no independent performance figures are available.

3.1.9 The Hitachi SR11000.

Machine type: RISC-based distributed-memory multi-processor.
Models: SR11000 K1.
Operating system: AIX (IBM's Unix variant).
Connection structure: Multi-dimensional crossbar (see remarks).
Compilers: Fortran 77, Fortran 95, Parallel Fortran, C, C++.
Vendors information Web page: `www.hitachi.co.jp/Prod/comp/hpc/SR_e/11ktop_e.html`
Year of introduction: 2003.

System parameters:

Model	SR11000 K1
Clock cycle	2.1 GHz
Theor. peak performance	
Per Proc. (64-bits)	134.4 Gflop/s
Maximal	68.8 Tflop/s
Memory/node	≤128 GB
Memory/maximal	16.4 TB
No. of processors	4–512
Communication bandwidth	
Point-to-point	12 GB/s (bidirectional)

Remarks:
The SR11000 is the fourth generation of distributed-memory parallel systems of Hitachi. It replaces its predecessor, the SR8000 (see 4). Presently two models are available, the SR11000 J1 and the SR11000 K1, discussed here. The J1 model is identical to the the K1 model except for the clock cycle which is 1.9 GHz. The J1 and K1 systems replace the H1 model that had exactly the same structure but was based on the 1.7 GHz IBM POWER4+ processor instead of the POWER5.
The basic node processor in the K1 model is a 2.1 GHz POWER5 from IBM. Unlike in the former SR2201 and SR8000 systems no modification of the processor its done to make it fit for Hitachi's Pseudo Vector Processing, a technique that enabled the processing of very long vectors without the detrimental effects that normally occur when out-of-cache data access is required. Presumably Hitachi is now relying on advanced prefetching of data to bring about the same effect.
The peak performance per basic processor, or IP, can be attained with 2 simultaneous multiply/add instructions resulting in a speed of 8.4 Gflop/s on the SR11000. However, 16 basic processors are coupled to form one processing node all addressing a common part of the memory. For the user this node is the basic computing entity with a peak speed of 134.4 Gflop/s. Hitachi refers to this node configuration as COMPAS, Co-operative Micro-Processors in single Address Space. In fact this is a kind of SMP clustering as discussed in sections 2.1 and 2.6. In contrast to the preceding SR8000 it does not contain a an SP anymore, a system processor that performed system tasks, managed communication with other nodes and a range of I/O devices. These tasks are now performed by the processors in the SMP nodes themselves.
The SR11000 has a multi-dimensional crossbar with a single-directional link speed of 12 GB/s. Also here IBM technology is used: the IBM Federation Switch fabric is used, be it in a different topology than IBM does for its own former p695 servers. From 4–8 nodes the cross-section of the network is 1 hop. For configurations 16–64 it is 2 hops and from 128-node systems on it is 3 hops.
Like in some other systems as the Cray XD1 (3.1.4), and the late AlphaServer SC, (4) and NEC Cenju-4, one is able to directly access the memories of remote processors. Together with the fast hardware-based barrier synchronisation this should allow for writing distributed programs with very low parallelisation overhead.
Of course the usual communication libraries like PVM and MPI are provided. In case one uses MPI it is possible to access individual IPs within the nodes. Furthermore, in one node it is possible to use OpenMP on individual IPs. Mostly this is less efficient than using the automatic parallelisation as done by Hitachi's compiler but in case one offers coarser grained task parallelism via OpenMP a performance gain can be attained. Hitachi provides its own numerical libraries to solve dense and sparse linear systems, FFTs, etc. As yet it is not known whether third party numerical libraries like NAG and IMSL are available.

Note: Large HPC configurations of the SR11000 are not sold in Europe as they are judged to be of insufficient economical interest by Hitachi.

Measured Performances:
The SR11000 was introduced by the end of 2003. A few model H1 systems have been sold in Japan in the mean time but there are no performance results available for these systems. Presently only one model K1 result is available at [45]: for a 80-node system at Hitachi's Enterprise Server Division a Linpack speed of 9,036 Gflop/s was measured for a 542,700 order linear system on a 10,752 Gflop/s peak performance system. Thus attaining an efficiency of 84.0%.

3.1.10 The HP Integrity Superdome.

Machine type: RISC-based ccNUMA system.
Models: HP Integrity Superdome.
Connection structure: Crossbar.
Operating system: HP-UX (HP's usual Unix flavour), Linux.
Compilers: Fortran 77, Fortran 90, HPF, C, C++.
Vendors information Web page: `h20341.www2.hp.com/integrity/cache/342370-0-0-0-121.html`
Year of introduction: 2004.

System parameters:

Model	Integrity Superdome
Clock cycle	1.6 GHz
Theor. peak performance	
Per Proc. core (64-bits)	6.4 Gflop/s
Maximal	409.6 Glop/s
Memory	$\leq$1 TB
No. of processors	$\leq$64
Communication bandwidth	
Aggregate (global)	64 GB/s
Aggregate (cell–backplane)	8 GB/s
Aggregate (within cell, see below)	16 GB/s

Remarks:
The Integrity Superdome is HP's investment in the future for high-end servers. Within a timespan of a few years it should replace the PA-RISC-based HP 9000 Superdome. HP has anticipated on this by giving it exactly the same macro structure: cells are connected to a backplane crossbar that enables the communication between cells. For the backplane it is immaterial whether a cell contains PA-RISC or Itanium processors. The Superdome has a 2-level crossbar: one level within a 4-processor cell and another level by connecting the cells the crossbar backplane. Every cell connects to the backplane at a speed of 8 GB/s and the global aggregate bandwidth for a fully configured system is therefore 64 GB/s.
As said, the basic building block of the Superdome is the 4-processor cell. All data traffic within a cell is controlled by the Cell Controller, a 10-port ASIC. It connects to the four local memory subsystems at 16 GB/s, to the backplane crossbar at 8 GB/s, and to two ports that each serve two processors at 6.4 GB/s/port. As each processor houses two CPU cores the available bandwidth per CPU core is 1.6 GB/s. Like the SGI Altix systems (see section 3.1.15), the cache coherency in the Superdome is secured by using directory memory. The NUMA factor for a full 64 processor systems is by HP's account very modest: only 1.8.
The Integrity Superdome, like its predecessor, is a ccNUMA machine. It therefore supports OpenMP over its maximum of 64 processors. As the Integrity Superdome is based on the Itanium 2 for which much Linux development is done in the past few years, the system can also be run with the Linux OS. In fact, because the machine can be partitioned, it is possible to run both Linux and HP-UX in the different complexes of the same machine. One can even mix the old PA-RISC processors with Itanium processors within one system: cells with different types of processors, making the system a hybrid Integrity and HP 9000 Superdome.
HP at the time of writing this report still does not offer Integrity Superdome machines based on the dual-core Montecito processors like Bull, Fujitsu, Hitachi, and SGI do. This is somewhat surprising as HP was one of the original developers of the EPIC processors of which the Itanium and Montecito are representatives.

Measured Performances:
There are no performance results of the newest version of the system. However, there is a result for an older 1.5 GHz processor machine: In [45] a speed of 1716.5 Gflop/s is reported for solving a full linear system of unspecified size on a system with 384 1.5 GHz Itanium 2 processors. As the theoretical peak performance of such a cluster is 2304 Gflop/s the efficiency is 75%.

3.1.11 The IBM eServer p575.

Machine type: RISC-based distributed-memory multi-processor.
Models: IBM eServer p575.
Operating system: AIX (IBM's Unix variant), Linux.
Connection structure: Ω-switch.
Compilers: XL Fortran (Fortran 90), (HPF), XL C, C++.
Vendors information Web page:
`www-03.ibm.com/systems/p/hardware/enterprise.html`.
Year of introduction: 2005 (8/16-CPU POWER5+ SMP).

System parameters:

Model	eServer p575 Single Core	eServer p575 Dual Core
Clock cycle	2.2 GHz	1.9 GHz
Theor. peak performance		
Per Proc. (64-bits)	8.8 Gflop/s	7.6 Gflop/s
Per node	70.4 Gflop/s	121.6 Gflop/s
Per 12-node frame	844.8 Gflop/s	1.46 Tflop/s
Max. (512 node system)	36 Tflop/s	62 Tflop/s
Memory/node	256 GB	256 GB
No. of processors/frame	8–96	8–192
Communication bandwidth		
Node-to-node (bi-directional)	2 GB/s	2 GB/s

Remarks:
There is a multitude of high end servers in the eServer p-series. However, IBM singles out the POWER5+ based p575 model specifically for HPC. The eServer p575 is the successor of the RS/6000 SP. It retains much of the macro structure of this system: multi-CPU nodes are connected within a frame either by a dedicated switch or by other means, like switched Ethernet. The structure of the nodes, however, has changed considerably, see 2.8.2. Up to 8 Dual Chip Modules (DCMs) are housed in a node totaling 8 or 16 cores in a node depending on whether the dual or single core version of the chip is used. For High-Performance Computing IBM recommends to employ the 8 CPU, single core nodes because a higher effective bandwidth from the L2 cache can be expected in this case. For less data intensive work that primarily uses the L1 cache the difference would be small while there is a large cost advantage using the 16-CPU nodes. The dual-core chips run at a lower clock frequency because of power dissipation considerations. The p575 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.
The so-called Federation switch is the fourth generation of the high-performance interconnects made for the p575 series. The Federation switch is, like its predecessors, an Ω-switch as described in section 2.4. It has a bi-directional link speed of 2 GB/s and an MPI latency of 5–7 μs . Although we mention only the highest speed option for the for inter-node communication here, there is a wide range of other options that could be chosen instead, e.g., Gbit Ethernet is also possible.
Applications can be run using PVM or MPI. IBM used to support High Performance Fortran, both a proprietary version and a compiler from the Portland Group. It is not clear whether this is still the case. 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 p575-based systems. As the nodes are in effect shared-memory SMP systems, within the nodes OpenMP can be employed for shared-memory parallelism and it can be freely mixed with MPI if needed. In addition to its own AIX OS IBM also supports some Linux distributions: both the professional versions of RedHat and SuSE Linux are available for the p575 series.

The standard commercial models that are marketed contain up to 128 nodes. However, on special request systems with up to 512 nodes can be built. This largest configuration is used in the table above.

Measured Performances:

In [45] a performance of 75.8 Tflop/s for a 12,208 processor system, the ASC Purple, is reported for solving a dense linear system of order $N = 1{,}383{,}600$, yielding an efficiency of 82%. Note that this machine is a 763-node dual-core system that falls even outside the range of the commerically offered maximum of 512 nodes.

3.1.12 The IBM BlueGene/L.

Machine type: RISC-based distributed-memory multi-processor.
Models: IBM BlueGene/L.
Operating system: Linux.
Connection structure: 3-D Torus, Tree network.
Compilers: XL Fortran 90, XL C, C++.
Vendors information Web page:
`www-1.ibm.com/servers/deepcomputing/.`
Year of introduction: 2004.

System parameters:

Model	BlueGene/L
Clock cycle	700 MHz
Theor. peak performance	
Per Proc. (64-bits)	2.8 Gflop/s
Maximal	367/183.5 Tflop/s
Memory/card	512 MB
Memory/maximal	$\leq$16 TB
No. of processors	$\leq$2$\times$65,536
Communication bandwidth	
Point-to-point (3-D torus)	175 MB/s
Point-to-point (Tree network)	350 MB/s

Remarks:
The BlueGene/L is the first in a new generation of systems made by IBM for very massively parallel computing. The individual speed of the processor has therefore been traded in favour of very dense packaging and a low power consumption per processor. The basic processor in the system is a modified PowerPC 400 at 700 MHz. Two of these processors reside on a chip together with 4 MB of shared L3 cache and a 2 KB L2 cache for each of the processors. The processors have two load ports and one store port from/to the L2 caches at 8 bytes/cycle. This is half of the bandwidth required by the two floating-point units (FPUs) and as such quite high. The CPUs have 32 KB of instruction cache and of data cache on board. In favourable circumstances a CPU can deliver a peak speed of 2.8 Gflop/s because the two FPUs can perform fused multiply-add operations. Note that the L2 cache is smaller than the L1 cache which is quite unusual but which allows it to be fast.
The packaging in the system is as follows: two chips fit on a compute card with 512 MB of memory. Sixteen of these compute cards are placed on a node board of which in turn 32 go into one cabinet. So, one cabinet contains 1024 chips, i.e., 2048 CPUs. For a maximal configuration 64 cabinets are coupled to form one system with 65,356 chips/130,712 CPUs. In normal operation mode one of the CPUs on a chip is used for computation while the other takes care of communication tasks. In this mode the theoretical peak performance of the system is 183.5 Tflop/s. It is however possible when the communication requirements are very low to use both CPUs for computation, doubling the peak speed; hence the double entries in the System Parameters table above. The number of 360 Tflop/s is also the speed that IBM is using in its marketing material.
The BlueGene/L possesses no less than 5 networks, 2 of which are of interest for inter-processor communication: a 3-D torus network and a tree network. The torus network is used for most general communication patterns. The tree network is used for often occurring collective communication patterns like broadcasting, reduction operations, etc. The hardware bandwidth of the tree network is twice that of the torus: 350 MB/s against 175 MB/s per link.

Measured Performances:
Recently IBM has reported to have attained a speed of 280.6 Tflop/s on the HPC Linpack benchmark in solving a linear system of size N = 1,769,471, with 131,072 processors, see [45]. The efficiency for this exercise is 76%.

3.1.13 The NEC Express5800/1000 series.

Machine type: Shared-memory ccNUMA system.
Models: Express5800 1080Xe, 1160Xe, 1320Xe.
Operating system: Linux, Windows Server 2003.
Connection structure: Crossbar.
Compilers: Fortran 95, HPF, ANSI C, C++.
Vendors information Web page: `www.necam.com/servers/products/model.cfm?model=10`
Year of introduction: 2006.

System parameters:

Model	1080Xe	1160Xe	1320Xe
Clock cycle	1.6 GHz	1.6 GHz	1.6 GHz
Theor. peak performance			
Per Proc. (64-bits)	12.8 Gflop/s	12.8 Gflop/s	12.8 Gflop/s
Maximal	102.4 Gflop/s	204.8 Gflop/s	409.6 Gflop/s
Main Memory (per frame)	≤64 GB	≤256 GB	≤512 GB
No. of processors	8	16	32
Communication bandwidth			
Point-to-point	—	—	—
Cell-to-cell (see remarks)	6.4 GB/s	6.4 GB/s	6.4 GB/s
Aggregate	102.4 GB/s	102.4 GB/s	102.4 GB/s

Remarks:
The Express5800 series is more or less a renaming of the earlier TX7 series. The structure of the system has stayed the same but instead of the former Itanium 2 processors the new machines are offered with the Montecito processors. It is another of the Itanium 2-based servers (see also 3.1.1, 3.1.15, 3.1.10, and 3.1.8) which recently appeared on the market. The largest configuration presently offered is the 1320Xe with 32 1.6 GHz Montecito processors. NEC had already some experience with Itanium servers offering 16-processor Itanium 1 servers under the name AsuzA and the already mentioned TX7 systems. So, the Express5800 systems can be seen as a third generation.
Processors are housed in 4-processor cells that connect via a flat crossbar. The bandwidth of the crossbar links is 6.4 GB/s. Unfortunately the documentation does not mention the bandwidth of the links between processors and memory within a cell. Although NEC still calls the machines SMP systems they are in fact ccNUMA systems with a low NUMA factor. The documentation speaks about "near-uniform high speed memory access".
Unlike the other vendors that employ the Itanium processors, NEC offers its own compilers including an HPF compiler which is probably available for compatibility with the software for the NEC SX-8 because it is hardly useful on a shared-memory system like the Express5800. The software also includes MPI and OpenMP.
Like the other Japanese vendors (see 3.1.7, 3.1.8) NEC very much emphasizes the RAS features of the system, targeting mission-critical operating environments.

Measured Performances:
The Express5800/1000 series is quite new, so no performance results are known yet for these systems.

3.1.14 The NEC SX-8 series.

Machine type: Shared-memory multi-vectorprocessor.

Models: SX-8B SX-8A, SX-8xMy, $y = 2, \ldots, 512$.
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.hpce.nec.com/572.0.html`
Year of introduction: 2004.

System parameters:

Model	SX-8B	SX-8A	SX-8xMy
Clock cycle	2 GHz	2 GHz	2 GHz
Theor. peak performance			
Per Proc. (64-bits)	16 Gflop/s	16 Gflop/s	16 Gflop/s
Maximal			
Single frame:	64 Gflop/s	128 Gflop/s	—
Multi frame:	—	—	90.1 Tflop/s
Main Memory, DDR2-SDRAM	32–64 GB	32–128 GB	≤16 TB
Main Memory, FCRAM	16–32 GB	32–64 GB	≤8 TB
No. of processors	1–4	4–8	8–4096

Remarks:
The SX-8 series is offered in numerous models but most of these are just frames that house a smaller amount of the same processors. We only discuss the essentially different models here. All models are based on the same processor, an 8-way replicated vector processor where each set of vector pipes contains a logical, mask, add/shift, multiply, and division pipe (see section 2.2 for an explanation of these components). As multiplication and addition can be chained (but not division) the peak performance of a pipe set at 2 GHz is 4 Gflop/s. Because of the 4-way replication a single CPU can deliver a peak performance of 16 Gflop/s. The official NEC documentation quotes higher peak performances because the peak performance of the scalar processor (rated at 4 Gflop/s, see below) is added to the peak performance of the vector processor to which it belongs. We do not follow this practice as a full utilisation of the scalar processor along with the vector processor in reality will be next to non-existent. The scalar processor that is 2-way super scalar and at 2 GHz has a theoretical peak of 4 Gflop/s. The peak bandwidth per CPU is 64 B/cycle. This is sufficient to ship 8 8-byte operands back or forth and just enough to feed one operand to each of the replicated pipe sets. Unlike from what one would expect from the naming the SX-8B is the simpler configuration of the two single-frame systems: it can be had with 1–4 processors but is in virtually all other respects equal to the larger SX-8A that can house 4–8 processors. There is one difference connected to the maximal amout of memory per frame: NEC now offers the interesting choice between the usual DDR2-SDRAM or FCRAM (Fast Cycle Memory. The latter type of memory can a factor of 2–3 faster than the former type of memory. However, because of the more complex structure of the memory, the density is about two times lower. Hence that in the systemparamemters table, the entries for FCRAM are about two times lower than for SDRAM. The lower bound for SDRAM in the SX-8A and SX-8B systems are the same: 32 GB. For the very memory-hungry applications that are usually run on vector-type systems, the availability of FCRAM can be beneficial for quite some of these applications.
In a single frame of the SX-8A models fit up to 8 CPUs. Internally the CPUs in the frame are connected by a 1-stage crossbar with the same bandwidth as that of a single CPU system: 64 GB/s/port. The fully configured frame can therefore attain a peak speed of 128 Gflop/s.
In addition, there are multi-frame models (SX-8xMy) where $x = 8, \ldots, 4096$ is the total number of CPUs and $y = 2, \ldots, 512$ is the number of frames coupling the single-frame systems into a larger system. There are two ways to couple the SX-8 frames in a multi-frame configuration: NEC provides a full crossbar, the so-called IXS crossbar to connect the various frames together at a speed of 8 GB/s for point-to-point unidirectional out-of-frame communication (1024 GB/s bisectional bandwidth for a maximum configuration). Also a HiPPI interface is available for inter-frame communication at lower cost and speed. When choosing for the IXS crossbar solution, the total multi-frame system is globally addressable, turning the system into a NUMA system. However, for performance reasons it is advised to use the system in distributed memory mode with MPI.
For distributed computing there is an HPF compiler and for message passing optimised MPI (MPI/SX) is available. In addition for shared memory parallelism, OpenMP is available.

Measured Performances:
In the TOP500 [45] the SX-8 from Stuttgart University shows a performance of 8.9 Tflop/s with a 576-processor machine on a system of size N = 829,440 with an efficiency of 97%.

3.1.15 The SGI Altix 4000 series.

Machine type: RISC-based ccNUMA system.
Models: Altix 4700.
Operating system: Linux (SuSE SLES9/10, RedHat EL4) + extensions.
Connection structure: Fat Tree.
Compilers: Fortran 95, C, C++.
Vendors information Web page: `www.sgi.com/products/servers/altix/4000/`.
Year of introduction: 2006.

System parameters:

Model	Altix 4700
Clock cycle	1.66 GHz
Theor. peak performance	
Per Proc. (64-bits)	13.3 Gflop/s
Maximal	6.8 Tflop/s
Memory	$\leq$512 GB
No. of processors	4–512
Communication bandwidth	
Point-to-point	3.2 GB/s
Aggregate peak/64 proc. frame	44.8 GB/s

Remarks:
The newest Altix version is the 4000 series succeeding the Altix 3700. Although the latter is still marketed we will not discuss it here as the functionality is largely the same. The difference is mainly in the support of the type of Intel Itanium processors and the communication network. The new Altix 4700 supports the dual-core Montecito processor with the new, faster 533 and 667 MHz frontside buses. Furthermore, where the model 3700 used NUMAlink3 for the connection of the processor boards, the Altix uses NUMAlink4 with twice the bandwidth at 3.2 GB/s, unidirectional. Also the structure of the processor boards has changed: instead of the so-called C-bricks with four Itanium 2 processors, 2 memory modules, two I/O ports, and two SHUBs (ASICs that connect processors, memory, I/O, and neighbouring processor boards), the Altix 4700 uses processor blades that houses 1 or 2 processors. SGI offers these two variants to accomodate different types of usage. The blades with 1 processor support the fastest frontside bus of 677 MHz thus giving a bandwidth of 10.7 GB/s to the processor on the blade. This processor blade is offered for bandwidth-hungry applications with irregular but massive memory access. The 2-processor blade, called the density option, uses the slower 533 MHz frontside bus for the processors and the slightly slower 1.6 GHz Montecito. The latter blade variant is assumed to satisfy a large part of the HPC users more cost-effectively.
The Altix is a ccNUMA system which means that the address space is shared between all processors (although it is physically distributed and therefore not uniformly accessible). In contrast to the Altix 3700 the bandwidth on the blades is as high as that of the off-board connections: NUMAlink4 technology is employed both on the blade and off-board.
SGI does not provide its own suite of compilers. Rather it distributes the Intel compilers for the Itanium processors. Also the operating system is Linux and not IRIX, SGI's former own Unix flavour although some additions are made to the standard Linux distributions, primarily for supporting SGI's MPI implementation and the CFXS file system.
Frames with 32 processor blades can be coupled with NUMAlink4 to form systems with a single-system image of at most 512 processors. So OpenMP programs with up to 512 processes can be run. On larger configurations, because Numalink allows remote addressing, one can apart from MPI also employ the Cray-style `shmem` library for one-sided communication.

Measured performances:
At this moment no large Altix 4700 installations are operational yet (the first two being installed at the time of writing). For the Altix 3700 the are results. In the TOP 500 list, [45], a complex of 10160 processors connected by Infiniband attained a speed of 51.78 Tflop/s solving a 1,290,240-order linear system. The efficiency for this complex is 85%.

3.1.16 The Sun Fire E25K.

Machine type: RISC-based shared-memory multiprocessor.
Models: Sun Fire E25K.
Operating system: Solaris (Sun's Unix flavour).
Connection structure: Crossbar (see remarks).
Compilers: Fortran 77, Fortran 90, C, C++.
Vendors information Web page: `www.sun.com/servers/highend/sunfire_e25k/index.xml`
Year of introduction: 2004.

System parameters:

Model	Sun Fire E25K
Clock cycle	1.5 GHz
Theor. peak performance	
Per Proc. (64-bits)	6.0 Gflop/s
Maximal	432 Gflop/s
Main Memory	$\leq$ 576 GB
No. of processors	$\leq$72
Communication bandwidth	
Aggregate	172.8 GB/s

Remarks:

In the E25K is the successor of the 3800-15K system. It employs the newest UltraSPARC IV+ processors at a clock frequency of 1.5 GHz. The structure of the systems is exactly the same as for its predecessor. In fact, the backplane is identical and one can turn one system into the other by just exchanging the processor boards. The processor/memory boards are plugged into a backplane that is an 18×18 flat crossbar. Each board contains four 1.5 GHz UltraSPARC IV+ processors and a maximum of 32 GB of memory. The maximum number of processors is therefore 72. Where in the 3800-15K there is the possibility to use slots of an additional crossbar to plug in extra 2-CPU boards to extend the computational power at the cost of I/O capacity, this possibility is *not* offered for the E25K system. Because of the flat crossbar memory access is uniform and the aggregate bandwidth of the crossbar is 172.8 GB/s. This is equivalent to 2.4 GB/s/processor or 1.6 B/cycle. So, an 8-byte operand needs 5 cycles to be shipped to the processor.
The main (almost only) difference with its predecessor is that the E25K system employs the dual-core UltraSPARC IV(+) processors. This formally doubles the peak performance per processor, although because of bandwidth constraints the effective performance increase will be lower. Sun maintains a somewhat confusing terminology with respect to multi-threading: it uses the term to express the fact that the two processor cores on an UltraSPARC IV(+) chip each do their processing independently. It calls this chip multi-threading (CMT). Yet the cores are not capable of switching between process threads as what is normally understood by multi-threading.
The E25K is a typical SMP machine with provisions for shared-memory parallelism with OpenMP over the full range of processors in the system.
As explained in section 2.8.6 Sun presently does not develop new implementations of the processor but leaves it to Fujitsu with the SPARC64. So, the Sun Fire 25K seems to be the end of this line of Sun systems.

Measured Performances:
In the TOP500 list of June 2004 a speed of 891.4 Gflop/s was reported for a 7-way cluster with a total of 672 processors (SPARC4) in solving a dense linear system of unspecified size. The efficiency for this problem is 74%.

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 was very similar but not equivalent to the former C-DAC system and there are numerous other examples). On the other hand, there are 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. Furthermore, we have reduced the contents of this section (in comparison to the former version) to systems that very probably are still actually in use in this printed (PostScript) version.
A more comprehensive, "full" version is available in the web version of this document at:

`www.phys.uu.nl/~steen/web04/gone.html`

This full information may be of interest for a subset of readers that want to know about the historical development of supercomputing over the last 11 years.
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.
We include also both the year of introduction and the year of exit of the systems when they were readily accessible. These time-spans could give a hint of the dynamics that governs this very dynamical branch of the computer industry.

4.1 Disappeared machines

Machine: Avalon A12.
Year of introduction: 1996.
Year of exit: 2000.
Type: RISC-based distributed memory multi-processor, max. 1680 processors.
Theoretical Peak performance: 1.3 Tflop/s.
Reason for disappearance: Avalon is not in business anymore.

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 (4.1).

Machine: Cambridge Parallel Processing DAP Gamma II Plus.
Year of introduction: 1995.
Year of exit: 2003.
Type: Distributed-memory processor array system, max. 4096 processors.
Theoretical Peak performance: 2.4 Gflop/s (32-bit).
Reason for disappearance: system became too slow, even for its specialised tasks. (4.1).

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 OpenFrame series (see below).

Machine: C-DAC PARAM Openframe series.
Year of introduction: 1996.
Year of exit: 1999.
Type: Distributed-memory RISC based system, max. 1024 processors.
Theoretical Peak performance: Unspecified.
Reason for disappearance: The system is not actively marketed anymore by C-DAC.

Machine: C-DAC PARAM 10000 Openframe series.
Year of introduction: 2000.
Year of exit: 2002.
Type: Distributed-memory RISC based system, max. processors: Unspecified.
Theoretical Peak performance: Unspecified.
Reason for disappearance: The system is replace by the newer C-DAC PARAM Padma, see section 3.1.2.

Machine: Cray T3E Classic.
Year of introduction: 1996.
Year of exit: 1997.
Type: Distributed-memory RISC based system, max. 2048 processors.
Theoretical Peak performance: 1228 Gflop/s.
Reason for disappearance: replaced Cray T3Es with faster clock. (4.1).

Machine: Cray MTA-2.
Year of introduction: 2001.
Year of exit: 2005.
Type: Distributed-memory multi-processor, max. 256 processors.
Theoretical Peak performance: 192 Gflop/s.
Reason for disappearance: The system is not actively marketed anymore by Cray.

Machine: Cray T3E 1350.
Year of introduction: 2000.
Year of exit: 2003.
Type: Distributed-memory RISC based system, max. 2048 processors.
Theoretical Peak performance: 2938 Gflop/s.
Reason for disappearance: Cray does not market the system anymore.

Machine: Cray J90.
Year of introduction: 1994.
Year of exit: 1998.
Type: Shared-memory vector-parallel, max. 32 processors.
Theoretical Peak performance: 6.4 Gflop/s.
Reason for disappearance: replaced by newer Cray SV1 (4.1).

Machine: Cray Y-MP T90.
Year of introduction: 1995.
Year of exit: 1998.
Type: Shared-memory vector-parallel, max. 32 processors.
Theoretical Peak performance: 58 Gflop/s.
Reason for disappearance: replaced by newer Cray SV1 (see below).

Machine: Cray SV-1(ex).
Year of introduction: 2000.
Year of exit: 2004.
Type: Shared-memory vector-parallel, max. 32 processors(per frame).
Theoretical Peak performance: 64 Gflop/s.
Reason for disappearance: replaced by newer Cray X1 (4.1).

Machine: Cray X1.
Year of introduction: 2000.
Year of exit: 2004.
Type: Shared-memory vector-parallel, max. 64 MSP processors.
Theoretical Peak performance: 819 Gflop/s.
Reason for disappearance: replaced by newer Cray X1E (3.1.3).

Machine: Digital Equipment Corp. AlphaServer 8200 & 8400.
Year of introduction: —.
Year of exit: 1998.
Type: Distributed-memory RISC based systems, max. 6 processors (AlphaServer 8200) or 14 (AlphaServer 8400).
Theoretical Peak performance: 7.3 Gflop/s, resp. 17.2 Gflop/s.
Reason for disappearance: after take-over by the HP AlphaServer SC and presently by the newer HP Integrity Superdome.(3.1.10).

Machine: Fujitsu AP3000.
Year of introduction: 1996.
Year of exit: 2003.
Type: Distributed memory RISC based system, max. 1024 processors.
Theoretical Peak performance: 614 Gflop/s.
Reason for disappearance: Fujitsu does not market the system anymore in favour of its PrimePower series (3.1.6).

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 (4.1).

Machine: Fujitsu VPP300/700 series.
Year of introduction: 1995/1996.

Year of exit: 1999.
Type: Distributed-memory multi-processor vectorprocessors, max. 256 processors.
Theoretical Peak performance: 614 Gflop/s.
Reason for disappearance: replaced by the VPP5000 series (4.1).

Machine: Fujitsu VPP5000 series.
Year of introduction: 1999.
Year of exit: 2002.
Type: Distributed-memory multi-processor vectorprocessors, max. 128 processors.
Theoretical Peak performance: 1.22 Tflop/s.
Reason for disappearance: Fujitsu does not market vector systems anymore, this machine line is replaced by the PRIMEPOWER series. (3.1.6).

Machine: Hitachi S-3800 series.
Year of introduction: 1993.
Year of exit: 1998.
Type: Shared-memory multi-processor vectorprocessors, max. 4 processors.
Theoretical Peak performance: 32 Gflop/s.
Reason for disappearance: Replaced by the SR8000 (4.1).

Machine: Hitachi SR2201 series.
Year of introduction: 1996.
Year of exit: 1998.
Type: Distributed-memory RISC based system, max. 1024 processors.
Theoretical Peak performance: 307 Gflop/s.
Reason for disappearance: Replaced by the newer SR8000 (4.1).

Machine: Hitachi SR8000 series.
Year of introduction: 1998.
Year of exit: 2000–2003 (different models).
Type: Distributed-memory RISC based system, max. 512 processors.
Theoretical Peak performance: 7.3 Tflop/s.
Reason for disappearance: Replaced by the newer SR11000 (3.1.9).

Machine: The HP Exemplar V2600.
Year of introduction: 1999.
Year of exit: 2000.
Type: Distributed-memory RISC based system, max. 128 processors.
Theoretical Peak performance: 291 Gflop/s.
Reason for disappearance: Replaced by the HP Integrity Superdome. (3.1.10).

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 RS/6000 SP (4.1).

Machine: IBM RS/6000 SP series.
Year of introduction: 1999.
Year of exit: 2001.
Type: Distributed-memory RISC based system, max. 2048 processors.
Theoretical Peak performance: 24 Gflop/s.
Reason for disappearance: Replaced by the newer eServer p575. (3.1.11).

Machine: Meiko CS-2.
Year of introduction: 1994.
Year of exit: 1999.
Type: Distributed-memory RISC based system.
Theoretical Peak performance: 200 Mflop/s per processor.
Reason for disappearance: Quadrics Supercomputers World Ltd. does not market the system anymore. The updated network technology is now offered for other systems like Bull NovaScale (see 3.1.1).

Machine: NEC Cenju-3.
Year of introduction: 1994.
Year of exit: 1996.
Type: Distributed-memory system, max. 256 processors.
Theoretical Peak performance: 12.8 Gflop/s.
Reason for disappearance: replaced by newer Cenju-4 series (4.1).

Machine: NEC Cenju-4.
Year of introduction: 1998.
Year of exit: 2002.
Type: Distributed-memory system, max. 1024 processors.
Theoretical Peak performance: 410 Gflop/s.
Reason for disappearance: NEC has withdrawn this machine in favour of a possible successor. Specifics are not known, however.

Machine: NEC SX-4.
Year of introduction: 1995.
Year of exit: 1996.
Type: Distributed-memory cluster of SM-MIMD vector processors, max. 256 processors.
Theoretical Peak performance: 1 Tflop/s.
Reason for disappearance: replaced by newer SX-5 series (see below).

Machine: NEC SX-5.
Year of introduction: 1998.
Year of exit: 2002.
Type: Distributed-memory cluster of SM-MIMD vector processors, max. 512 processors.
Theoretical Peak performance: 5.12 Tflop/s.
Reason for disappearance: replaced by newer SX-6 series (see below).

Machine: NEC SX-6.
Year of introduction: 2002.
Year of exit: 2005.
Type: Distributed-memory cluster of SM-MIMD vector processors, max. 1024 processors.
Theoretical Peak performance: 9.2 Tflop/s.
Reason for disappearance: replaced by newer SX-8 series (3.1.14).

Machine: Quadrics Appemille.
Year of introduction: 1999.
Year of exit: 2004.
Type: Processor array, max. 2048 processors.
Theoretical Peak performance: 1 Tflop/s.
Reason for disappearance: Not marketed anymore.

Machine: Silicon Graphics Origin2000.
Year of introduction: 1996.

Year of exit: 2000.
Type: Shared-memory multi-processor, max. 128 processors.
Theoretical Peak performance: 102.4 Gflop/s.
Reason for disappearance: replaced by the SGI Origin 3000 (see below).

Machine: Silicon Graphics Origin3000.
Year of introduction: 2003.
Year of exit: 2005.
Type: Shared-memory multi-processor, max. 512 processors.
Theoretical Peak performance: 819 Gflop/s.
Reason for disappearance: replaced by the Itaium-based SGI Altix 4700 (3.1.15).

Machine: SUN E10000 Starfire.
Year of introduction: 1997.
Year of exit: 2001.
Type: Shared-memory multi-processor, max. 64 processors.
Theoretical Peak performance: 51.2 Gflop/s.
Reason for disappearance: replaced by the Fire 3800-15K (see below).

Machine: SUN Fire 3800-15K.
Year of introduction: 2001.
Year of exit: 2004.
Type: Shared-memory multi-processor, max. 106 processors.
Theoretical Peak performance: 254 Gflop/s.
Reason for disappearance: replaced by the Fire E25K (3.1.16).

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. We recognise the wishes of such vendors (it is generally wise not to stretch the expectation of potential customers too long) and they 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 which is targeted.
A development that has shown to be of significance is the introduction of Intel's IA-64 Itanium processor family. Already six vendors are offering Itanium 2-based systems at the moment and it is known that HP will end the marketing of its Alpha and PA-RISC based systems in favour of the Itanium processor family. SGI stopped the further development of MIPS processor based machines. This means that in a few years only AMD, IBM, Intel, and SUN will produce RISC-like processors for HPC systems. This will make the HPC system field much less diverse and interesting. On the other hand, the shock that was caused in the USA by the advent of the Japanese Earth Simulator system may help in refueling the funding of alternative processor and computer architecture research. Indeed, some initiatives in that direction are already under way but these will not bear real new results in one or two years (except maybe with the IBM Blue Gene, see below).
In section 2.8 we already noted the considerable interest generated by systems that provide acceleration by means of FPGAs or other special computational accelerators like those from ClearSpeed. For selected algorithms such accelerators can sometimes speed up the applications containing them by an order of maginitude or more. The interest has so much increased because of the much improved software interfaces to program the accelerators which makes them accessible to the large community of application program developers. Some vendors make systems built entirely around the accelerator. An example is SRC Computer Inc. that makes clusters in which each node contains a FPGA, while Cray markets its XD1 and SGI its RASC blades (see below). It is to be expected that at within the near future a HPC cannot afford *not* to include somehow such accelerators into their architectures.

5.1 Cray Inc.

In the end of 2002 the next generation vector processor, the X1, from Cray Inc. was ready to ship. It built on the technology found in the Cray SV-1s. Cray widely publicises a roadmap of future systems as far as around 2010. It remains to be seen how much can be realised, however, at least 2 of these systems are have reached the market: first the Cray X1E, which is nothing more than the present X1 system in which the clock cycle of the processor is raised from 800 MHz to 1.2 GHz. The other one is the Cray XT3, a commercialised version of the AMD Opteron-based (11,648 processors) Red Storm machine that is presently built by Cray for Sandia Laboratories. There is much interest for this type of system because of the cheap basic processor and the fast network based on AMD's HyperTransport and Cray's SeaStar router ASIC.
Further away lies the BlackWidow a follow on to the X1E, scheduled for 2007. The system will have a new type of infrasturucture, code name "Rainier". As the same infrastructure will be used for the next generation XT3 it is possible to combine vectorprocessors and AMD processors in one configuration and

let them share common resources like I/O processors. Most likely such a system cannot operate yet as one integrated system in the first phase but this is the ultimate goal to be reached in the Cascade system in 2010. In this system it must be possible to freely add either type of processor and FPGAs and massively multi-threaded processors as well.
The latter type of processor is still developed by Cray as a follow-on to the former MTA-2 systems of which very few were actually sold. Still, there is considerable interest in this processor architecture, especially in national security circles. The MTA-3, code name Eldorado, will have the macro-architecture of the Cray XT3 but with the AMD processors on the board replaced by MTA processor at a clock frequency of 500 MHz instead of the 200 MHz in the MTA-2. At the time of writing prototypes exist but the product is kept low-profile and it is not known presently whether it will become a publicly marketed product.

5.2 Hewlett-Packard/Intel

HP and Intel will have a great influence in the next few years with their Itanium processor family. A dual core processor based on the Itanium is already on (or beyond) the drawing board and will hit the market in a year or two. As dual core processors usually have a relatively poorer performance than their single core equivalents, the performance improvement will not be spectacular. The system architecture will be much more important. Also a diversification of the processors themselves may help to boost the performance. Because of HP/Intel's experience with VLIW processors (as the Itanium essentially is), one might expect that the research will go in the direction of processors with even longer instruction words and possibly including specialised devices for high level operations like FFTs or sparse Matrix-Vector multiplies as well. When and how such improvements would turn up in future systems is however speculative. It will certainly not happen within the next two years. As yet no radically different system architectures are known to be on HP's drawing boards. Instead it may try to penetrate more in the cluster field were it already has installed some large Itanium-based systems.

5.3 IBM

IBM has been working for some years on its BlueGene systems. Of which the first model, the BlueGene/L, has been installed now in various places (see 3.1.12). Other BlueGene follow-ups are planned called the BlueGene/P with a peak speed of 1 Pflop/s and the BlueGene/Q with a peak speed of 3 Pflop/s, respectively. All these systems are hardly meant for the average HPC user but they may help in finding suitable architectural features for systems for the general market.
Of course the development of the POWERx processors also will make its mark: the POWER5+ processor has the usual technology-related advantages over its predecessor, and now the first prototypes of the POWER6 processor are being tested. When this processor becomes generally available the clock frequency is expected to be 3.5–4 GHz, yielding a processor with a performance of 14–16 Gflop/s/core. Futhermore, it is a subject of research how to couple 8 of them such that a virtual vector processor with a peak speed of around 120 Gflop/s can be made. This approach is called the ViVA (Virtual Vector Architecture). It is reminiscent of Hitachi's SR8000 processors (which used POWER5 processors) or the MSP processors in the Cray X1E. This road will take some years to go also after the POWER6 processor has become available and will extend to the next generation(s) of the POWERx.
In addition, the cell processor, developed with Sony might become a factor in HPC systems. This processor has 8 computational cores and a control processor. Although it is in first instance targetted at the gaming industry (hence Sony's interest) numerical experiments with the processor proved it to be extremely performant in this area, be it in 32-bit precision. Future generations could however be adapted to numerically intensive work. When it is possible to maintain similar performance characteristics it could become an important building block for HPC systems.

5.4 SGI

Two years ago SGI made it known that it would stop producing its MIPS-based systems. Therefore, the difference they would like to make with respect to other vendors that also offer Itanium-based systems would

have to lie in the macro-architecture of their systems. Improvements will be realised in the speed of the network (NUMALink3 to NUMAlink4 and beyond) and systems with a large amount of processors in a single system image (SSI). In that respect SGI has a track record with its MIPS-based Origin 3000 systems which may be extended for its future Altix $x000$ systems where at present SSIs of 512 are realised.
Further in the future SGI seems to have plans that are more or less similar to Cray's Cascade project: coupling of heterogeneous processor sets through its proprietary network, in this case a successor of the NUMAlink4 network architecture. A first step in that direction is the availability of the so-called RASC blades that can be put into the Altix 4700 infrastructure. Each RASC blade features 2 FPGAs that can be used as computational accelerators for certain algorithms in applications. The idea is to further diversify the future systems, ultimately into a system with the codename "Ultraviolet". Development of such systems is very costly, so it remains to be seen whether such plans will pass the stage of intentions in regard of the present difficult financial position of SGI.

5.5 Sun

Like Cray and IBM Sun has been awarded a grant from DARPA to develop so-called high-productivity systems in DARPA's HPCS program. Up till now Sun has concentrated on developing heavily multi-threaded processors, the first product being the Niagara chip and the next, the still more multi-threaded Rock processor. The first implementation of the Niagara processor is a ready and now called the T2000 processor. It is however not meant for computational work but rather as a throughput Web or I/O server.The chip harbours 8 CPU cores of which each core is 4-way multi-threaded. Systems based on the Rock processor would not be available before 2008. For its near future mainstream high-end systems Sun will therefore rely on Fujitsu's SPARC64 processors.

6 Glossary of terms

This section contains the explanation of some often-used terms that either are not explained in the text or, by contrast, are described extensively and for which a short description may be convenient.

6.1 Glossary

Architecture: The internal structure of a computer system or a chip that determines its operational functionality and performance.

Architectural class: Classification of computer systems according to its architecture: e.g., distributed memory MIMD computer, symmetric multi processor (SMP), etc. See this glossary and section 2.1 for the description of the various classes.

ASCI: Accelerated Strategic Computer Initiative. A massive funding project in the USA concerning research and production of high-performance systems. The main motivation is said to be the management of the USA nuclear stockpile by computational modeling instead of actual testing. ASCI has greatly influenced the development of high-performance systems in a single direction: clusters of SMP systems.

ASIC: Application Specific Integrated Circuit. A chip that is designed to fulfill a specific task in a computer system, e.g. for routing messages in a network.

Bank cycle time: The time needed by a (cache-)memory bank to recover from a data access request to that bank. Within the bank cycle time no other requests can be accepted.

Beowulf cluster: Cluster of PCs or workstations with a private network to connect them. Initially the name was used for do-it-yourself collections of PCs mostly connected by Ethernet and running Linux to have a cheap alternative for "integrated" parallel machines. Presently, the definition is wider including high-speed switched networks, fast RISC-based processors and complete vendor-preconfigured rack-mounted systems with either Linux or Windows as an operating system.

Bit-serial: The operation on data on a bit-by-bit basis rather than on byte or 4/8-byte data entities in parallel. Bit-serial operation is done in processor array machines where for signal and image processing this mode is advantageous.

Cache — data, instruction: Small, fast memory close to the CPU that can hold a part of the data or instructions to be processed. The primary or level 1 caches are virtually always located on the same chip as the CPU and are divided in a cache for instructions and one for data. A secondary or level 2 cache is mostly located off-chip and holds both data and instructions. Caches are put into the system to hide the large latency that occurs when data have to be fetched from memory. By loading data and or instructions into the caches that are likely to be needed, this latency can be significantly reduced.

Capability computing: A type of large-scale computing in which one wants to accommodate very large and time consuming computing tasks. This requires that parallel machines or clusters are managed with the highest priority for this type of computing possibly with the consequence that the computing resources in the system are not always used with the greatest efficiency.

Capacity computing: A type of large-scale computing in which one wants to use the system (cluster) with the highest possible throughput capacity using the machine resources as efficient as possible. This may have adverse effects on the performance of individual computing tasks while optimising the overall usage of the system.

ccNUMA: Cache Coherent Non-Uniform Memory Access. Machines that support this type of memory access have a physically distributed memory but logically it is shared. Because of the physical difference of the location of the data items, a data request may take a varying amount of time depending on the location of the data. As both the memory parts and the caches in such systems are distributed a mechanism is necessary to keep the data consistent system-wide. There are various techniques to enforce this (directory memory, snoopy bus protocol). When one of these techniques is implemented the system is said to be cache coherent.

Clock cycle: Fundamental time unit of a computer. Every operation executed by the computer takes at least one and possibly multiple cycles. Typically, the clock cycle is now in the order of one to a few nanoseconds.

Clock frequency: Reciproke of the clock cycle: the number of cycles per second expressed in Hertz (Hz). Typical clock frequencies nowadays are 400 MHz–1 GHz.

Clos network: A logarithmic network in which the nodes are attached to switches that form a *spine* that ultimately connects all nodes.

Communication latency: Time overhead occurring when a message is sent over a communication network from one processor to another. Typically the latencies are in the order of a few μs for specially designed networks, like Infiniband or Myrinet, to about 100 μs for (Gbit) Ethernet.

Control processor: The processor in a processor array machine that issues the instructions to be executed by all the processors in the processor array. Alternatively, the control processor may perform tasks in which the processors in the array are not involved, e.g., I/O operations or serial operations.

CRC: Type of error detection/correction method based treating a data item as a large binary number. This number is divided by another fixed binary number and the remainder is regarded as a checksum from which the correctness and sometimes the (type of) error can be recovered. CRC error detection is for instances used in SCI networks.

Crossbar (multistage): A network in which all input ports are directly connected to all output ports without interference from messages from other ports. In a one-stage crossbar this has the effect that for instance all memory modules in a computer system are directly coupled to all CPUs. This is often the case in multi-CPU vector systems. In multistage crossbar networks the output ports of one crossbar module are coupled with the input ports of other crossbar modules. In this way one is able to build networks that grow with logarithmic complexity, thus reducing the cost of a large network.

Distributed Memory (DM): Architectural class of machines in which the memory of the system is distributed over the nodes in the system. Access to the data in the system has to be done via an interconnection network that connects the nodes and may be either explicit via message passing or implicit (either using HPF or automatically in a ccNUMA system).

Dual core chip: A chip that contains two CPUs and (possibly common) caches. Due to the progression of the integration level more devices can be fitted on a chip. HP, IBM, and Sun make dual core chips and other vendors may follow in the near future.

EPIC: Explicitly Parallel Instruction Computing. This term is coined by Intel for its IA-64 chips and the Instruction Set that is defined for them. EPIC can be seen as Very Large Instruction Word computing with a few enhancements. The gist of it is that no dynamic instruction scheduling is performed as is done in RISC processors but rather that instruction scheduling and speculative execution of code is determined beforehand in the compilation stage of a program. This simplifies the chip design while potentially many instructions can be executed in parallel.

Fat tree: A network that has the structure of a binary (quad) tree but that is modified such that near the root the available bandwidth is higher than near the leafs. This stems from the fact that often a root processor has to gather or broadcast data to stands for Host Bus Adapter. It is the part in an external network that constitutes the interface between the network itself and the PCI bus of the

compute node. HBAs usually carry a good amount of processing intelligence themselves for initiating communication, buffering, checking for correctness, etc. HBAs tend to have different names in different networks: HCA or TCA for Infiniband, LANai for Myrinet, ELAN for QsNet, etc. all other processors and without this modification contention would occur near the root.

FPGA: FPGA stands for Field Programmable Gate Array. This is an array of logic gates that can be hardware-programmed to fulfill user-specified tasks. In this way one can devise special purpose functional units that may be very efficient for this limited task. As FPGAs can be reconfigured dynamically, be it only 100–1,000 times per second, it is theoretically possible to optimise them for more complex special tasks at speeds that are higher than what can be achieved with general purpose processors.

Functional unit: Unit in a CPU that is responsible for the execution of a predefined function, e.g., the loading of data in the primary cache or executing a floating-point addition.

Grid — 2-D, 3-D: A network structure where the nodes are connected in a 2-D or 3-D grid layout. In virtually all cases the end points of the grid are again connected to the starting points thus forming a 2-D or 3-D torus.

HBA: HBA stands for Host Bus Adapter. It is the part in an external network that constitutes the interface between the network itself and the PCI bus of the compute node. HBAs usually carry a good amount of processing intelligence themselves for initiating communication, buffering, checking for correctness, etc. HBAs tend to have different names in different networks: HCA or TCA for Infiniband, LANai for Myrinet, ELAN for QsNet, etc.

HPF: high-performance Fortran. A compiler and run time system that enables to run Fortran programs on a distributed memory system as on a shared memory system. Data partition, processors layout, etc. are specified as comment directives that makes it possible to run the processor also serially. Present HPF available commercially allow only for simple partitioning schemes and all processors executing exactly the same code at the same time (on different data, so-called Single Program Multiple Data (SPMD) mode).

Hypercube: A network with logarithmic complexity which has the structure of a generalised cube: to obtain a hypercube of the next dimension one doubles the perimeter of the structure and connect their vertices with the original structure.

Instruction Set Architecture: The set of instructions that a CPU is designed to execute. The Instruction Set Architecture (ISA) represents the repertoire of instructions that the designers determined to be adequate for a certain CPU. Note that CPUs of different making may have the same ISA. For instance the AMD processors (purposely) implement the Intel IA-32 ISA on a processor with a different structure.

Memory bank: Part of (cache) memory that is addressed consecutively in the total set of memory banks, i.e., when data item $a(n)$ is stored in bank b, data item $a(n+1)$ is stored in bank $b+1$. (Cache) memory is divided in banks to evade the effects of the bank cycle time (see above). When data is stored or retrieved consecutively each bank has enough time to recover before the next request for that bank arrives.

Message passing: Style of parallel programming for distributed memory systems in which non-local data that is required explicitly must be transported to the processor(s) that need(s) it by appropriate send and receive messages.

MPI: A message passing library, Message Passing Interface, that implements the message passing style of programming. Presently MPI is the *de facto* standard for this kind of programming.

Multithreading: A capability of a processor core to switch to another processing thread, i.e., a set of logically connected instructions that make up a (part of) a process. This capability is used when a process thread stalls, for instance because necessary data are not yet available. Switching to another thread that has instructions that can be executed will yield a better processing utilisation.

NUMA factor: The difference in speed of accessing local and non-local data. For instance when it takes 3 times longer to access non-local data than local data, the NUMA factor is 3.

OpenMP: A shared memory parallel programming model in which shared memory systems and SMPs can be operated in parallel. The parallelisation is controlled by comment directives (in Fortran) or pragmas (in C and C++), so that the same programs also can be run unmodified on serial machines.

PCI bus: Bus on PC node, typically used for I/O, but also to connect nodes with a communication network. The bandwidth varies with the type from 110-480 MB/s. Newer upgraded versions PCI-X and PCI Express are (becoming) available presently.

Pipelining: Segmenting a functional unit such that it can accept new operands every cycle while the total execution of the instruction may take many cycles. The pipeline construction works like a conveyor belt accepting units until the pipeline is filled and than producing results every cycle.

Processor array: System in which an array (mostly a 2-D grid) of simple processors execute its program instructions in lock-step under the control of a Control Processor.

PVM: Another message passing library that has been widely used. It was originally developed to run on collections of workstations and it can dynamically spawn or delete processes running a task. PVM now largely has been replaced by MPI.

Register file: The set of registers in a CPU that are independent targets for the code to be executed possibly complemented with registers that hold constants like 0/1, registers for renaming intermediary results, and in some cases a separate register stack to hold function arguments and routine return addresses.

RISC: Reduced Instruction Set Computer. A CPU with its instruction set that is simpler in comparison with the earlier Complex Instruction Set Computers (CISCs). The instruction set was reduced to simple instructions that ideally should execute in one cycle.

Shared Memory (SM): Memory configuration of a computer in which all processors have direct access to all the memory in the system. Because of technological limitations on shared bandwidth generally not more than about 16 processors share a common memory.

`shmem`: One-sided fast communication library first provided by Cray for its systems. However, `shmem` implementations are also available for SGI and HP AlphaServer systems.

SMP: Symmetric Multi-Processing. This term is often used for compute nodes with shared memory that are part of a larger system and where this collection of nodes forms the total system. The nodes may be organised as a ccNUMA system or as a distributed memory system of which the nodes can be programmed using OpenMP while inter-node communication should be done by message passing.

TLB: Translation Look-aside Buffer. A specialised cache that holds a table of physical addresses as generated from the virtual addresses used in the program code.

Torus: Structure that results when the end points of a grid are wrapped around to connect to the starting points of that grid. This configuration is often used in the interconnection networks of parallel machines either with a 2-D grid or with 3-D grid.

Vector register: A multiple entry register (typically 128–256) that hold the operands to be processed by a vector unit. Using vector registers controlled by vector instructions guarantees the production of results every cycle for the amount of operands in the register.

Vector unit (pipe): A pipelined functional unit that is fed with operands from a vector register and will produce a result every cycle (after filling the pipeline) for the complete contents of the vector register.

Virtual Shared Memory: The emulation of a shared memory system on a distributed memory machine by a software layer.

VLIW processing: Very Large Instruction Word processing. The use of large instruction words to keep many functional units busy in parallel. The scheduling of instructions is done statically by the compiler and, as such, requires high quality code generation by that compiler. VLIW processing has been revived in the IA-64 chip architecture, there called EPIC (see above).

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.

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] W.Anderson, D.W. Hess, P. Briggs, A. Khoklov, R. Rosenberg, C.S. Hellberg, M. Lanzagorta, *Early Experience with Scientific Programs on the Cray MTA-2*, Proc. SC2003, November 2003, Phoenix, AZ, USA.

[3] The ASCI program: `http://www.llnl.gov/asci/`.

[4] S.H. Bokhari, J.R. Sauer, *Sequence Alignment on the Cray MTA-2*, Proc. HICOMB2002, April 2002, Fort Lauderdale, USA.

[5] L. Carter, J. Feo, A. Snavely, *Performance and Programming Experience on the Tera MTA*, Proc. SIAM Conf. on Parallel Processing, March 1999.

[6] K. Cassirer, B. Steckel, *Block-Structured Multigrid on the Cenju*, 2nd Cenju Workshop, October 1998, Sankt Augustin, Germany.

[7] R. Chandra, L. Dagum, D. Kohr, D. Maydan, J. McDonald, R. Menon, *Parallel Programming in OpenMP*, Morgan Kaufmann Publishers Inc., January 2001.

[8] D.E. Culler, J.P. Singh, A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann Publishers Inc., August 1998.

[9] D.W. Doerfler, *An Analysis of the Pathscale Inc. Infiniband Host Channel Adapter, InfiniPath*, Sandia Report, SAND2005-5199, August 2005.

[10] J.J. Dongarra, *Performance of various computers using standard linear equations software*, Computer Science Technical Report CS-89-85, Univ. of Tennessee, August 1, 2006.

[11] Overview page of Distributed Shared Memory systems: `http://www.ics.uci.edu/~javid/dsm.html`.

[12] T.H. Dunigan, M.R. Fahey, J.B. White, P.H. Worley, *Early Evaluation of the Cray X1*, Proc. SC2003, November 2003, Phoenix, AZ, USA.

[13] T.H. Dunigan, ONRL Cray XD1 Evaluation, `www.csm.ornl.gov/~dunigan/xd1/`.

[14] Directory with EuroBen results: `www.euroben.nl/results`.

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

[16] 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.

[17] D.B. Gustavson, Q. Li, *Local-Area MultiProcessor: the Scalable Coherent Interface*, SCIzzL Report, Santa Clara University, Dept. of Computer Engineering, 1995. Available through: `www.scizzl.com`.

[18] 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)

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

[20] 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.

[21] HPC Challenge Benchmark, `icl.cs.utk.edu/hpcc/`.

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

[23] 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/`

[24] D.J.Kerbyson, A. Hoisie, S. Pakin, F. Petrini, H.J. Wassermann, *A performance evaluation on an Alpha EV7 processing node*, Int. J. of High Perf. Comp. Appl., **18**(2), 199–209, Summer 2004.

[25] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra, *MPI: The Complete Reference Vol. 1, The MPI Core*, MIT Press, Boston, 1998.

[26] W. Gropp, S. Huss-Ledermann, A. Lumsdaine, E. Lusk, B. Nitzberg, W. Saphir, M. Snir *MPI: The Complete Reference, Vol. 2, The MPI Extensions*, MIT Press, Boston, 1998.

[27] `http://www.myrinet.com`

[28] N.J. Boden, D. Cohen, R.E. Felderman, A.E. Kulawik, C.L. Seitz, J.N. Seizovic, Wen-King Su, *Myrinet – A Gigabit-per-second Local Area Network*, IEEE Micro, **15**, No. 1, Jan. 1995, 29–36.

[29] W.E. Nagel, *Applications on the Cenju: First Experience with Effective Performance*, 2nd Cenju Workshop, October 1998, Sankt Augustin, Germany.

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

[31] OpenMP Forum, *OpenMP Application Interface, version 2.5*, Web page: `www.openmp.org/`, May 2005.

[32] F. Petrini, D.J. Kerbyson, S. Fakin, *The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q*, Proc. SC2003, November 2003, Phoenix, AZ, USA.

[33] F. Petrini, W. Feng, A. Hoisie, S. Coll, E. Frachtenberg, *The Quadrics Network: High-performance clustering technology*, IEEE Micro Jan/Febr 2002, 46–57.

[34] T. Sato, *The Earth Simulator*, Proc. SC2002, Nov. 2002, IEEE Press. (Proceedings only on CD-ROM.)

[35] T. Shanley, *Infiniband Network Architecture*, Addison-Wesley, Nov. 2002.

[36] D.H.M. Spector, *Building Unix Clusters*, O'Reilly, Sebastopol, CA, USA, July 2000.

[37] 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.

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

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

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

[41] A.J. van der Steen, *Benchmarking the Silicon Graphics Origin2000 System*, Technical Report WFI-98-2, Dept. of Computational Physics, Utrecht University, The Netherlands, May 1998. The report can be downloaded from: `www.euroben.nl/reports/`

[42] A.J. van der Steen, *An evaluation of some Beowulf clusters*, Technical Report WFI-00-07, Utrecht University, Dept. of Computational Physics, December 2000. (Also available through `www.euroben.nl`, directory `reports/`.)

[43] A.J. van der Steen, *An evaluation of Itanium 2-based high-end servers*, Technical Report HPCG-2004-04, Utrecht University, High Performance Computing Group, November 2004. (Also available through `www.euroben.nl`, directory `reports/`.)

[44] T.L. Sterling, J. Salmon, D.J. Becker, D.F. Savaresse, *How to Build a Beowulf*, The MIT Press, Boston, 1999.

[45] H.W. Meuer, E. Strohmaier, J.J. Dongarra, H.D. Simon, *Top500 Supercomputer Sites*, 27th Edition, June 2006, The report can be downloaded from: `www.top500.org/`.

[46] Task Force on Cluster Computing home page: `www.clustercomputing.org`.