Overview of recent supercomputers

Aad J. van der Steen
HPC Research
NCF
P.O. Box 9
6800 AA Arnhem
The Netherlands
steen@hpcresearch.nl
www.hpcresearch.nl

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 development phase. The machines are described according to their macro-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 systems information given in this report.

This document reflects the technical momentary 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 18th 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 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 [54], for a long time was 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 was realised 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, high-end graphical cards 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.

Still, 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.* [11] 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 has 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 to be reported do not allow the report to be shorter than in former years: in the order of 80+ 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 are the heart of the systems. We do that in section 2.8. In addition, from 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. Now, for the first time we include a section on computational accelerators. Their use has grown significantly the last few years

and it is certain the this trend will continue for quite some years to follow. So, it seemed appropriate to discuss them, however superficial this necessarily must be.

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 various places in the world (all with measured performances in the range of 100+ Tflop/s), the Japanese Earth Simulator system (with a performance around 40 Tflop/s), and the IBM Cell-based Roadrunner system, the first system to exceed a Petaflop/s at Los Alamos, 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 earlier 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.

All systems described are 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. It 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/PDF 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 18 years, this information will still be available in full at URL `http://www.phys.uu.nl/~steen/web08/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, networks, and accelerators 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. So, not all features of the systems discussed will be present. We also omit systems that may be characterised as "high-performance" in the fields of database management, real-time computing, or visualisation, scientific and technical computing being our primary interest. Still we think (and certainly hope) that the impressions

obtained from the entries of the individual machines may be useful to many. Furthermore, we have set a threshold of about 300 Gflop/s for systems to appear in this report as, at least with regard to theoretical peak performance, single CPUs often exceed 3 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/web08/overview.html` (Europe).
`www.euroben.nl/reports/web08/overview.html` (Europe).
`www.top500.org/in_focus/orsc`.

From the 16^{th} issue on we *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 [11, 47].

2.1 The main architectural classes

Since many years the taxonomy of Flynn [18] 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 about all large servers 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 were the CPP Gamma II and the Quadrics Apemille which are not marketed anymore since about a 2 years. Nevertheless, the concept is still interesting and it is recurring these days as a co-processor in HPC systems be it in a somewhat restricted form, for instance, a Graphical Processing Unit (GPU).

  Another subclass of the SIMD systems are the vectorprocessors. Vectorprocessors act on arrays of similar data rather than on single data items using specially structured CPUs. When data can be manipulated by these vector units, results can be delivered with a rate of one, two and — in special cases — of three per clock cycle (a clock cycle being defined as the basic internal unit of time for the system). So, vector processors execute on their data in an almost parallel way but only when executing in vector mode. In this case they are several times faster than when executing in conventional scalar mode. For practical purposes vectorprocessors are therefore mostly regarded as SIMD machines. Examples of such systems are for instance the NEC SX-9B and the Cray X2.
- **MISD** machines: Theoretically in these types 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-9 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 but that may have an impact on the performance. 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 [25] 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 [14]. Since 2006 Intel markets its "Cluster OpenMP" (based on TreadMarks) as a commercial product. It allows the use of the shared-memory OpenMP parallel model [36] to be used on distributed-memory clusters. Lastly, so-called Partitioned Global Adrress Space (PGAS) languages like Co-Array Fortran (CAF) and Unified Parallel C (UPC) are gaining in popularity due to the recently emerging multi-core processors. With proper implementation this allows a global view of the data and one has language facilities that make it possible to specify processing of data associated with a (set of) processor(s) without the need for explicitly moving the data around.

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) [19], and MPI (Message Passing Interface, [30, 31]) 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-9, and the Cray X2 have this structure. So, within the clustered nodes a shared-memory programming style can be used while between

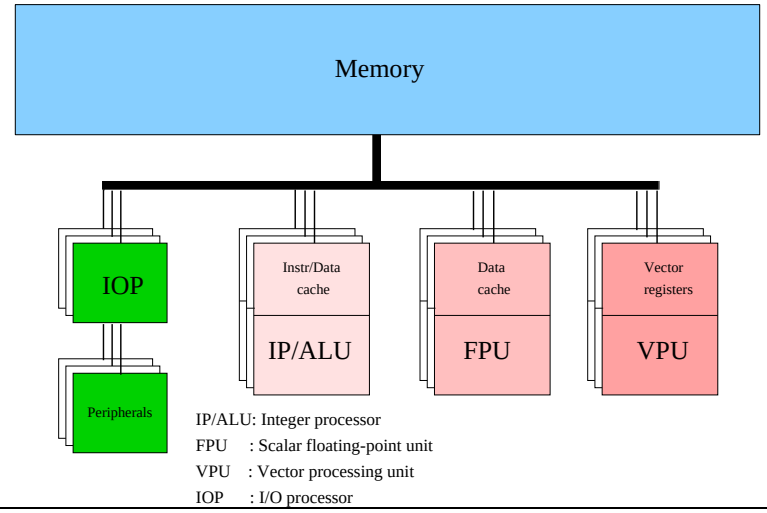


Figure 2.1: *Blockdiagram of a vector processor.*

clusters message-passing should be used. It must be said that PVM is not used very much anymore and the development has stopped. MPI has now more or less become the *de facto* standard.
For SM-MIMD systems we mention OpenMP [36, 9, 10], 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 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 [44]) and may emerge again in the near future. 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 here 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. Of late, however, this tendency is reversed because of the increasing gap in speed between the memory and the processors: the Cray X2 has a cache and the follow-on of NEC's SX-9 vector system may well have one.

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

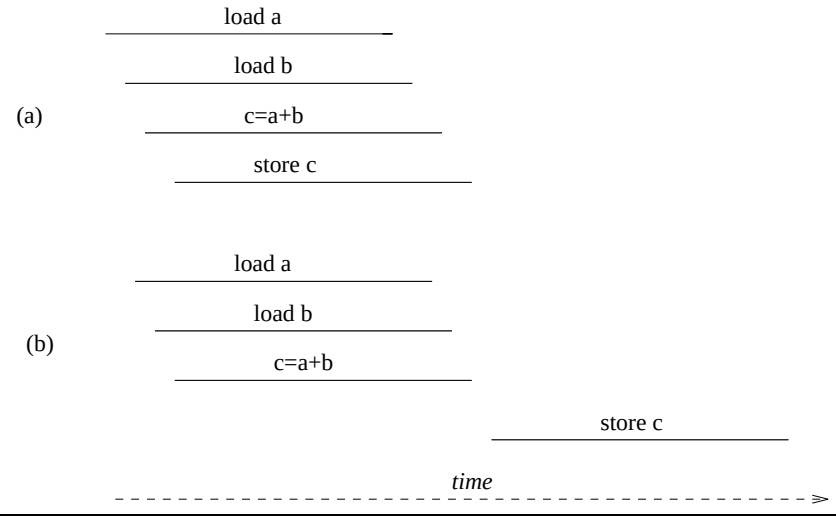


Figure 2.2: *Schematic diagram of a vector addition. Case (a) when two load- and one store pipe are available; case (b) when two load/store pipes are available.*

$c = a + b$ (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 (i.e., two load operations and one store operation at the *same* time). Only Cray Inc. in its former Y-MP, C-series, and 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 the DM-SIMD type are sometimes also known as *processor-array* machines [22]. 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 worthwhile 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

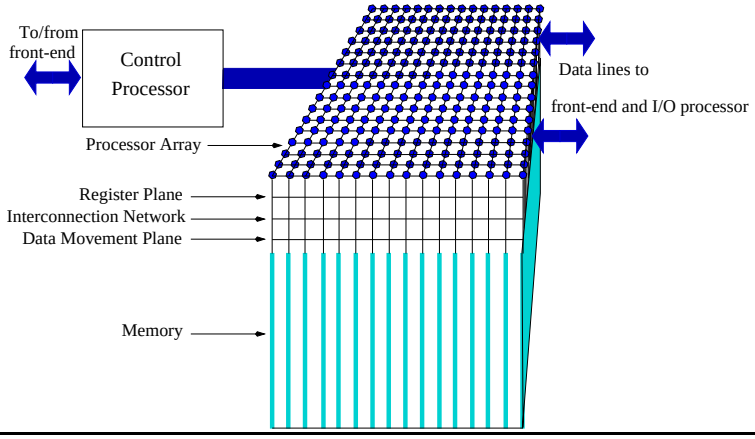


Figure 2.3: *A generic block diagram of a distributed-memory SIMD machine.*

will deviate to some degree. Figure 2.3 might suggest that all processors in such systems are connected in a 2-D grid and indeed, the interconnection topology of this type of machines always includes the 2-D grid. As opposing ends of each grid line are also always connected the topology is rather that of a torus. 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 data sets 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 device 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 bit-wise, irrespective of their type. So, e.g., operations on integers are produced by software routines on these simple bit-serial processors which takes at least as many cycles as the operands are long. So, a 32-bit integer result will be produced two times faster than a 64-bit result. For floating-point operations a similar situation holds, be it that the number of cycles required is a multiple of that needed for an integer operation. As the number of processors in this type of systems is mostly large (1024 or larger, the Quadrics Apemille was a notable exception, however), the

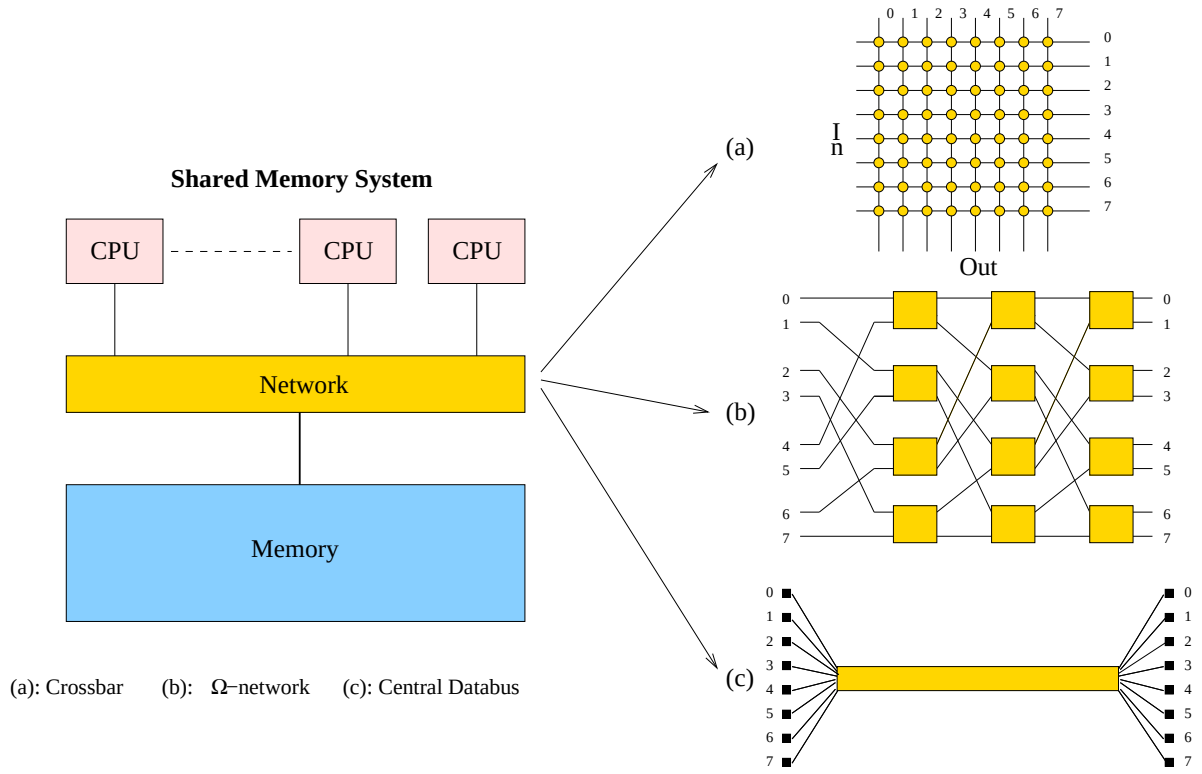


Figure 2.4: *Some examples of interconnection structures used in shared-memory MIMD systems.*

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 co-processors 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. because, as mentioned, the execution time for bit-serial machines is proportional to the length of the operands, this may result in significant speedups.

Presently there are no DM-SIMD systems are on the market but some types of computational accelerators (see 2.9) share many characteristics with DM-SIMD systems that have existed until shortly. We will briefly discuss some prperties of these accelerators later.

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) provides 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 [26]. It now has been effectively superseded by InfiniBand, however (see 2.10).

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 4700, 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 as far as they still exist. 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. Some Cray vectorprocessors employed in the past special communication registers within the CPUs (the X-MP and Y-MP/C series) by which they could communicate directly with the other CPUs they have to synchronise with. This is, however, not practised anymore as it is viewed too costly a feature. 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 undoubtedly 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 generally 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 when using the so-called message passing parallelisation model (which is the case in the vast majority of programs). 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 ([19, 30, 31]) and partly because, at least theoretically, this class of systems is able to outperform all other types of machines. Alternatively, instead of message passing a Partitioned Global Address Space parallelisation model may be used with a programming language like UPC [56] or Co-Array Fortran [6]. In this case one still has to be aware where the relevant data are but no explicit sending/receiving between processors is necessary. This greatly simplifies the programming but the compilers are still either fairly immature or even in an experimental stage which does not always guarantee a great performance to say the least.

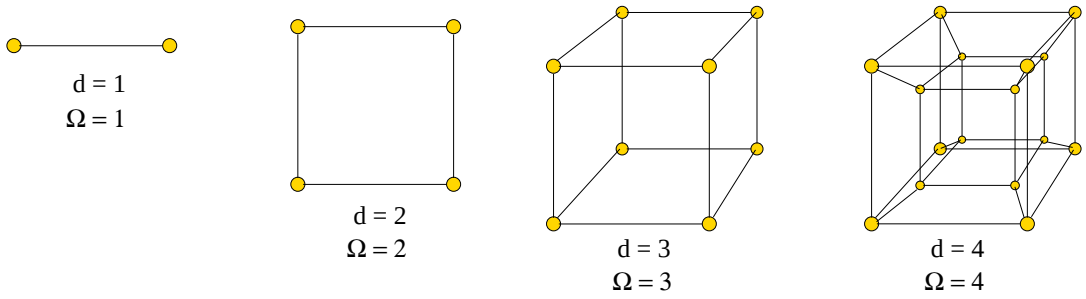
The advantages of DM-MIMD systems are clear: the bandwidth problem that haunts shared-memory systems is avoided because the bandwidth scales up automatically with the number of processors. Furthermore, the speed of the memory which is another critical issue with shared-memory systems (to get a peak performance 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.

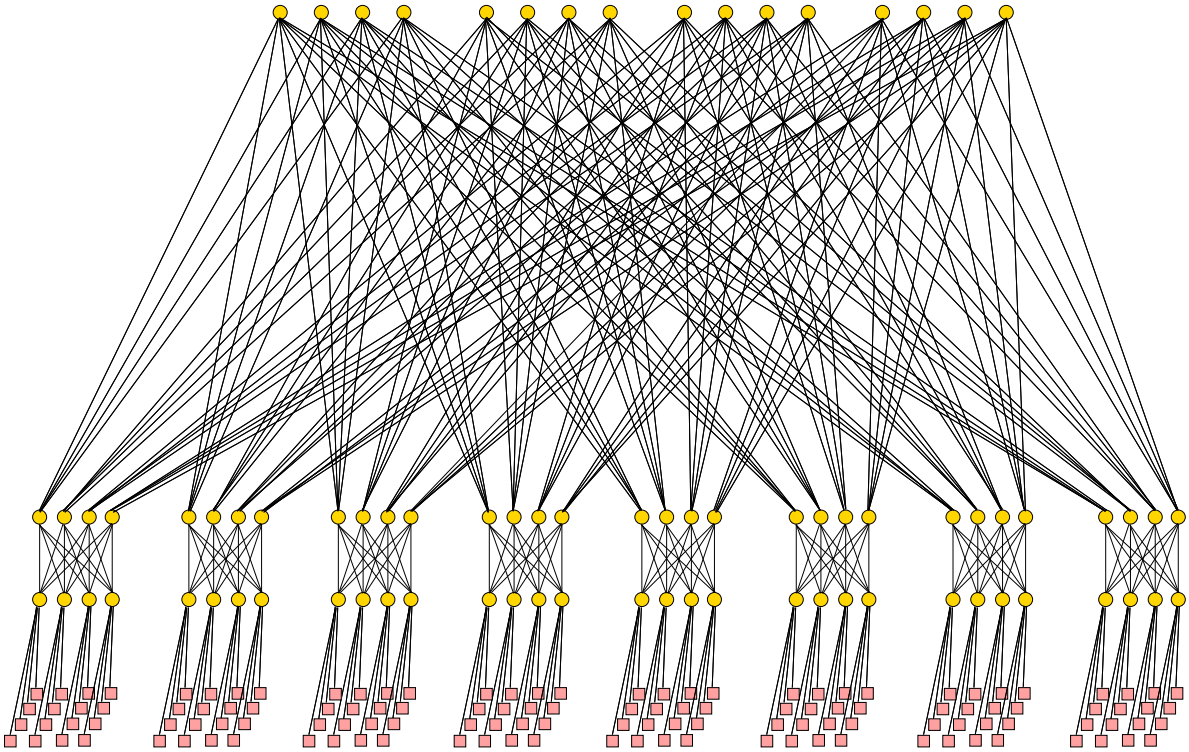
It will be clear from the paragraph above that also for DM-MIMD machines both the topology and the speed of the data paths are of crucial importance for the practical usefulness of a system. Again, as in the section on SM-MIMD systems, the richness of the connection structure has to be balanced against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

Another cost-effective way to connect a large number of processors is by means of a *fat tree*. In principle



(a) Hypercubes, dimension 1−4.



(b) A 128−way fat tree.

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

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 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 [23]. 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 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

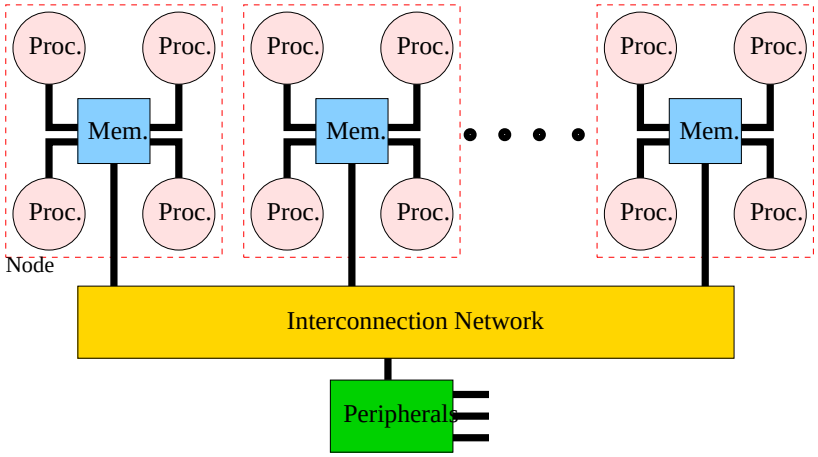


Figure 2.6: *Block diagram of a system with a "hybrid" network: clusters of four CPUs are connected by a crossbar. The clusters are connected by a less expensive network, e.g., a Butterfly network.*

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 (like the Bull NovaScale 5000 series, HP Superdome, and SGI Altix 4700) 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 cross-section, 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 [55]. Also books how to build and maintain clusters have greatly added to their popularity [53, 43]. As the cluster scene has become a mature and 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 (a small minority these days).

The number of vendors that sell cluster configurations has become so large that it is not possible to include all their 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 used to be mostly operated through the Linux operating system (a small minority using

Microsoft Windows) where these operating systems either missed the tools or these tools were 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 too, particularly because nowadays there is a plethora of cluster management and monitoring tools. In [49] is looked at some of the aspects that are necessary conditions for this kind of use like available cluster management tools and batch systems. The systems then assessed are now quite obsolete but many of the conlusions are still valid: 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 cluster nodes have become available with 4–8 processors where each processor also may have 2–4 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 capability-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 ccNUMA systems. Interestingly, in clusters the ccNUMA memory access model is turning up now in the cluster nodes as for the larger nodes it is not possible anymore to guarantee symmetric access to all data items for all processor cores (evidently, for a core a data item in its own local cache will be faster available than for a core in another processor).

Fortunately, there is nowadays a fair choice of communication networks available in clusters. Of course Gigabit Ethernet or 10 Gigabit Ethernet are always possible, which are attractive for economic reasons, but have the drawback of a high latency ($\approx$ 10–40 μs). Alternatively, there are networks that operate from user space at high speed and with a latency that approaches these of the networks in integrated systems. These will be discussed in section 2.10.

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 ones. 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. Still, we will be a bit more conservative in this section than in the description of the systems in general. The reason is processors are turned out at a tremendous pace while planning ahead for next generations takes years. We therefore tend to stick to the really existing components in this section or when already a β version of a processor is being evaluated.

The RISC processor scene has shrunken significantly in the last few years. The Alpha and PA-RISC processors have disappeared in favour of the Itanium processor product line and, interestingly, the MIPS processor line that disappeared some years ago now re-appears in the SiCortex systems (see section 3.1.19). 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 FPGAs or other computational accelerators because their possibilities in performance, price level, power consumption, and ease of use has improved to a degree that they offer attractive alternatives for certain application fields.

The notion of "RISC processor" has eroded somewhat in the sense that the processors that execute the Intel x86 (CISC) instruction set now have most of the characteristics of a RISC processor. Both the AMD and Intel x86 processors in fact decode the CISC instructions almost entirely into a set of RISC-like fixed-length instructions. Furthermore, both processor lines feature out-of-order execution, both are able to address and

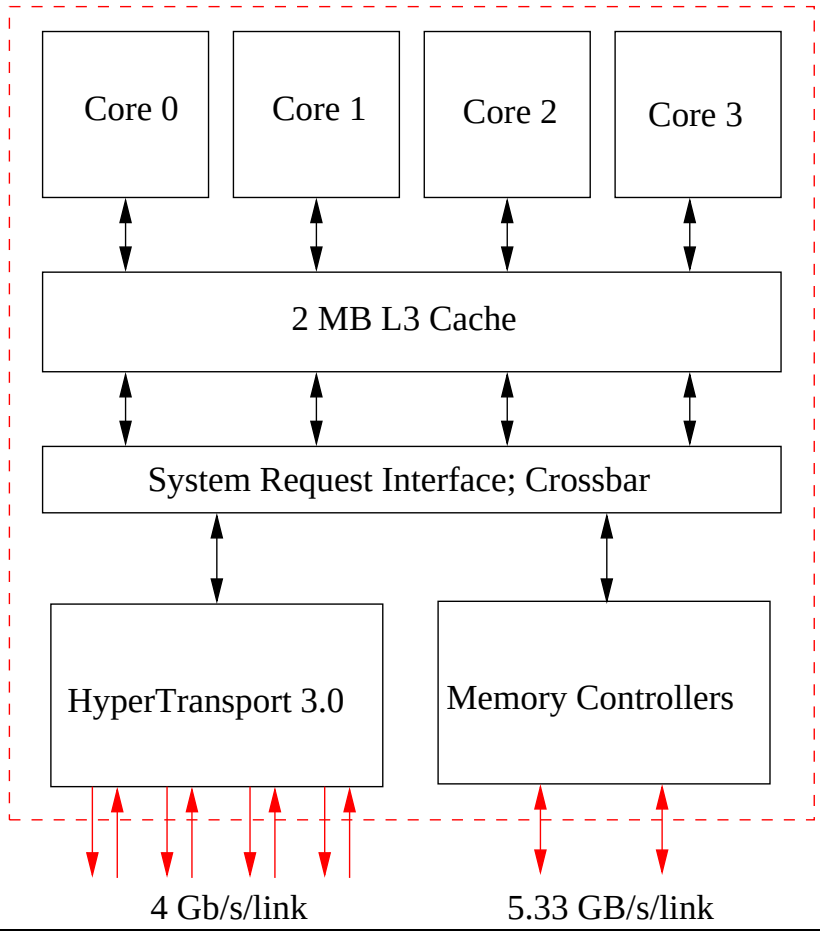


Figure 2.7: *Block diagram of an AMD Phenom (Barcelona) processor.*

deliver results natively in 64-bit length, and the bandwidth from memory to the processor core(s) have become comparable to those of RISC/EPIC processors. A distinguishing factor is still the mostly much larger set of registers in the RISC processors.
Another notable development of the last few years 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 Phenom

The AMD Athlon, Opteron, and Phenom processors are clones with respect to Intel's x86 Instruction Set Architecture, and especially the dual-core Opteron has been (and is) quite popular for use in clusters. In addition, the Opteron and Phenom are used in Cray and the Liquid Computer systems that are presented later. The Phenom is a quad core chip (also known under the code name "Barcelona") that has come onto the market by the end of 2007. We will discuss this new processor in the following. The first version of the processor contained an error in the L2 TLB that could be patched at the expense of the performance. As of April 2008 there is a new version that does not have TLB problems and works without the performance penalty. In contrast to the Opteron, a third level of cache has been added that is common to the four cores as can be seen in Figure 2.7. The size is 2 MB. As already mentioned, the AMD processors have many features that are also present in modern RISC processors: they support out-of-order execution, have multiple floating-point units, and can issue up to 9 instructions simultaneously. A block diagram of the Phenom processor core is shown in Figure 2.8. The four 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).

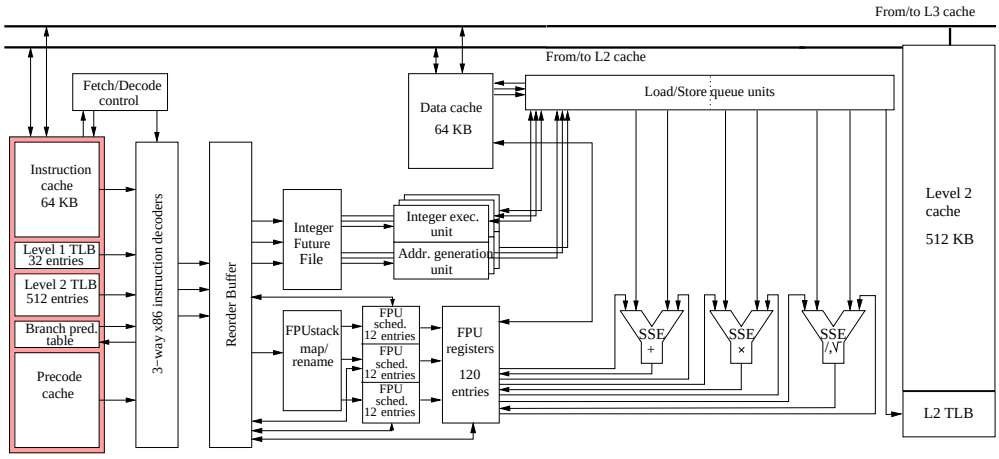


Figure 2.8: *Block diagram of an AMD Phenom processor core.*

The figure shows that the processor has three pairs of Integer Execution Units and Address Generation Units that via an 32-entry Integer Scheduler takes care of the integer computations and of address calculations. Both the Integer Future File and the Floating-Point Scheduler are fed by the 72-entry Reorder Buffer that receives the decoded instructions from the instruction decoders. The decoding in the Phenom core has become more efficient than in the earlier processors: SSE instructions decode now into 1 micro-operation (μop) as are most integer and floating-point instructions. In addition, a new piece of hardware, called the sideband stack optimiser, has been added (not shown in the figure) that takes care of the stack manipulations in the instruction stream thus making instruction reordering more efficient thereby increasing the effective number of instructions per cycle.
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 Reorder Buffer and reorders them if necessary before handing them over to the FPU Scheduler. The Floating-Point Register File is 120 elements deep on par with the number of registers as available in RISC processors [1].
The floating-point part of the processor contains three units: Floating Add and Multiply units that can work in superscalar mode, resulting in two floating-point results per clock cycle and a unit handling "miscelaneous" operations, like division and square root. 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 directly accessible from higher level languages, like Fortran 90 or C(++). Both instruction sets were originally meant for massive processing of visualisation data but are increasingly used to also for standard dense linear algebra operations.
Due to the shrinkage of technology to 45 nm each core can harbour a secondary cache of 512 KB. 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. In the Phenom the third generation, HyperTransport 3.0 is used which in principle can transfer 10.4 GB/s per directional link. However, because the present sockets do not yet support this speed the throughput is still that of the earlier HyperTransport 1.1 link, 4 GB/s/link/direction.
The clock frequency is in the range of 2.2–2.5 GHz which makes the Phenom 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.

[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.

2.8.2 IBM POWER6

Recently, in the systems that feature as IBM's supercomputer line, the p575 series, the nodes contain the POWER6 chip as the computational engine. There are quite some changes with respect to its predecessor, the POWER5+, both in the chip lay-out and in the two cores that reside on a chip. Figure2.9 shows the layout of the cores, caches, and controllers on the chip. Already here are significnant changes: instead of

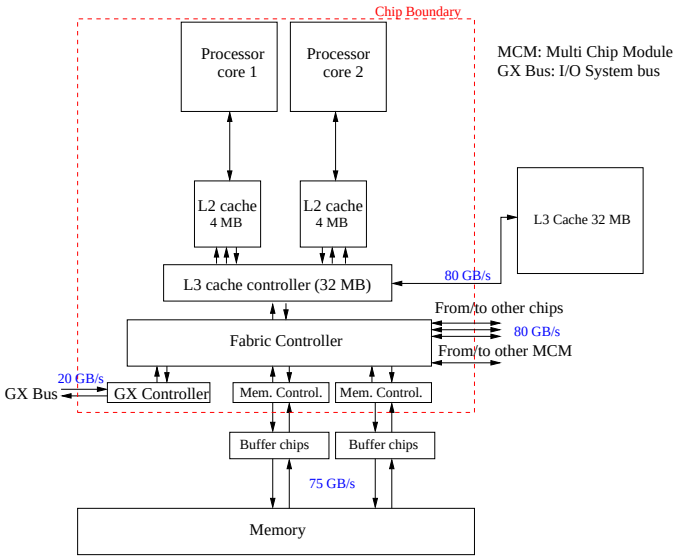


Figure 2.9: *Diagram of the IBM POWER6 chip layout*

a 1.875 MB shared L2 cache now each core has its own 4 MB 8-way set-associative L2 cache that operates at half the core frequency. In addition, there are 2 memory controllers that via buffer chips connect to the memory and, depending on the amount of buffer chips and data widths (both are variable) can have a data read speed $\leq$ 51.2 GB/s and a write speed of $\leq$ 25.6 GB/s, i.e., with a core clock cycle of 4.7 GHz up to 11 B/cycle for a memory read and half of that for a memory write. Furthermore, the separate busses for data and coherence control between chips are now unified with a choice of both kinds of traffic occupying 50% of the bandwidth or 67% for data and 33% for coherence control. The off-chip L3 cache has shrunk from 36 to 32 MB. It is a 16-way set-associative victim cache that operates at 1/4 of the clock speed.
Also the core has changed considerably. It is depicted in Figure 2.10. The clock frequency has increased from 1.9 GHz in the POWER5+ to 4.7 GHz for the POWER6 (water cooled version), an increase of almost a factor 2.5 while the power consumption stayed in the same range of that of the POWER5+. This has partly come about by a technology shrink from a 90 nm to a 65 nm feature size. It meant also that some features of the POWER5+ have disappeared. For instance, the POWER6 largely employs static instruction scheduling, except for a limited amount of floating-point instruction scheduling because some of these can sometimes be fit in empty slots left by division and square root operations. The circuitry required for dynamic instruction scheduling that thus could be removed has however been replaced by new units. Besides the 2 Fixed Point Units (FXUs) and the 2 Binary Floating-Point Units (BFUs) that were already present in the POWER5+ there are now also a Decimal Floating-Point Unit (DFU) and a VMX unit, akin to Intel's SSE units for handling multimedia instructions. In fact, the VMX unit is inherited from the IBM PowerPC's Altivec unit. The Decimal Floating-Point Unit is IEEE 754R compliant. It is obviously for financial calculations and is hardly of consequence for HPC use. Counting only the operations of the BPUs both executing fused multipy-adds (FMAs), the theoretical peak performance in 64-bit precision is 4 flop/cycle or 18.8 Gflop/s/core. Also new is a Checkpoint Recovery Unit has been added that is able to catch faulty FXU and FPU (both binary and decimal) instruction executions and reschedule them for retrial. Because of the large variety of functional units a separate Instruction Dispatch Unit ships the instructions that are ready for execution to the appropriate units while a siginificant part of instruction decoding has been pushed into the Instruction Fetch Unit, including updating the Branch History Tables.

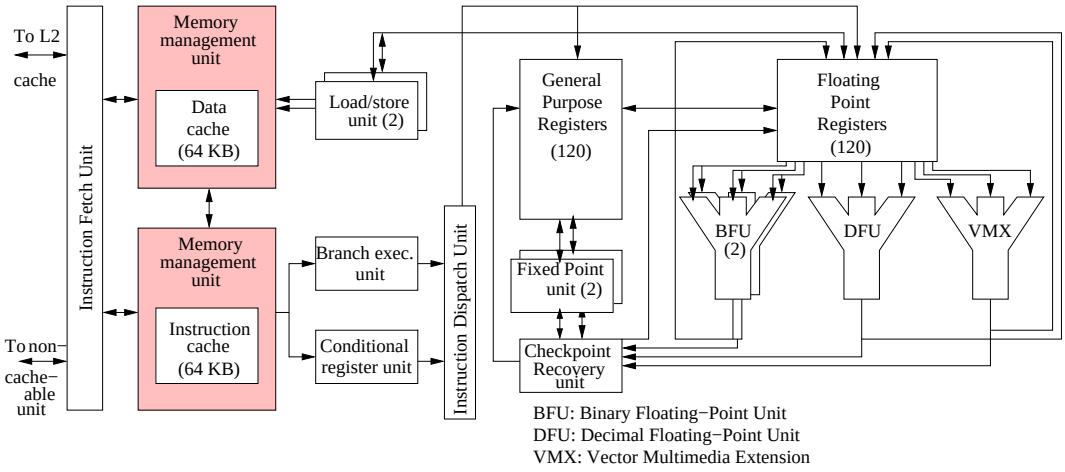


Figure 2.10: *Block diagram of the IBM POWER6 core.*

The BFUs not only execute the usual floating-point instructions like add, multiply, and FMA. They also take care of division and square root operations. A new phenomenon is that integer divide and multiply operations are also executed by the BFUs again saving on circuitry and therefore power consumption. In addition, these operations can be pipelined in this way and yield a result every 2 clock cycles.
The L1 data cache has been doubled in comparison of the POWER5+ and is now 64 KB like the L1 instruction cache. Both caches are 4-way set-associative.
The Simultaneous Multi-Threading (SMT) that was already present in the POWER5+ has been retained in the POWER6 processor and has been improved by a higher associativity of the the L1 I and D caches and a larger dedicated L2 cache. Also instruction decoding and dispatch are dedicated for each thread. By using SMT the cores 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 especially where only very limited dynamic scheduling is possible.
Because the much higher clock cycle and the fact that the memory DIMMs are attached to each chip is it not possible anymore to maintain a perfect SMP behaviour within a 4-chip node, i.e., it matters whether data is accessed from a chip's own memory or from the memory of a neighbouring chip. Although the data is only one hop away there is a ccNUMA effect that one have to be aware of in multi-threaded applications.

2.8.3 IBM PowerPC 970 processor

A number of IBM systems are built from JS21 blades, the largest being the Mare Nostrum system at the Barcelona Supercomputing Centre. On these blades a variant of the large IBM PowerPC processor family is used, the PowerPC 970MP. It is a dual-core processor with a clock cycle of 2.3 GHz. A block diagram of a processor core is given in Figure 2.11.

A peculiar trait of the processor is that the L1 instruction cache is two times larger than the L1 data cache, 64 against 32 KB. This is explained partly by the fact that up to 10 instructions can be issued every cycle to the various execution units in the core. Apart from two floating-point units that perform the usual dyadic operations, there is an AltiVec vector facility with a separate 80-entry vector register file, a vector ALU that

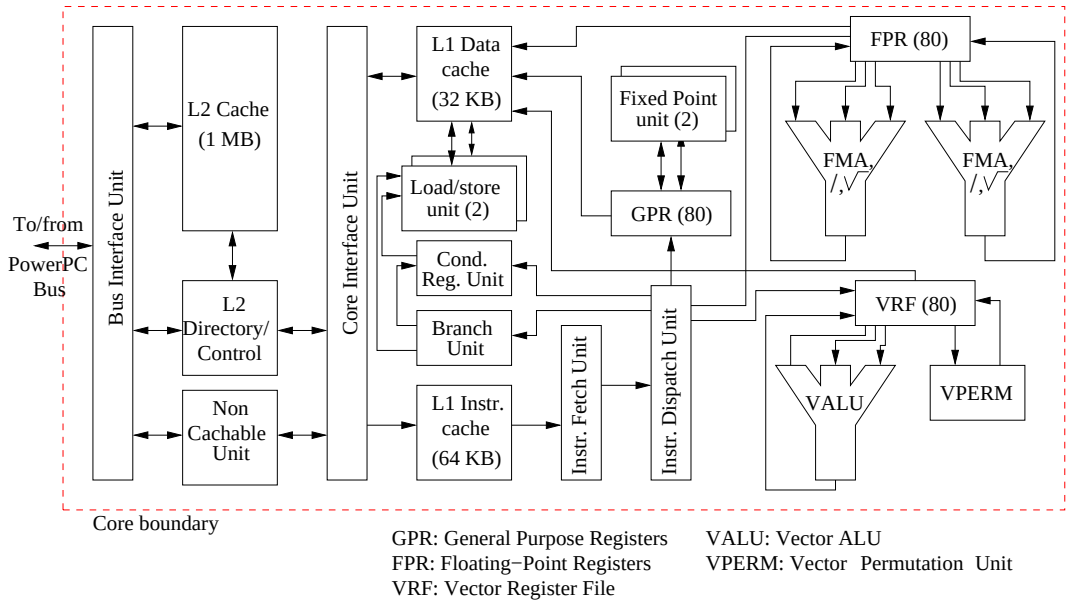


Figure 2.11: *Block diagram of the IBM PowerPC 970 MP core.*

performs (fused) multiply/add operations, and a vector permutation unit that attempts to order operands such that the vector ALU is used optimally. The vector unit was designed for graphics-like operations but works quite nicely on data for other purposes as long as access is regular and the operand type agrees. Theoretically, the speed of a core can be 13.8 Gflop/s/core when both FPUs turn out the results of a fused multiply-add and the vector ALU does the same. One PowerPC 970 MP should therefore have a theoretical peak performance of 27.6 Gflop/s. The floating-point units also perform square-root and division operations. Apart from the floating-point and vector functional units two integer fixed-point units and two load/store units are present in addition to a conditional register unit and a branch unit. The latter uses two algorithms for branch prediction that are applied according to the type of branch to be taken (or not). The success rate of the algorithms is constantly monitored. Correct branch prediction is very important for this processor as the pipelines of the functional units are quite deep: from 16 for the simplest integer operations to 25 stages in the vector ALU. So, a branch miss can be very costly. The L2 cache is integrated and has a size of 1 MB. To keep the load/store latency low, hardware-initiated prefetching from the L2 cache is possible and 8 oustanding L1 cache misses can be tolerated. The operations are dynamically scheduled and may be out-of-order. In total 215 operations may be in flight simultateously in the various functional units, also due to the deep pipelines.
The two cores on a chip have common arbitration logic to regulate the data traffic from and to the chip. There is no third level cache between the memory and the chip on the board housing them. This is possible because of the moderate clock cycle and the rather large L2 cache.

2.8.4 IBM BlueGene processors

At the time of writing this report two BlueGene types of systems have become available: the BlueGene/L and the BlueGene/P, the successor of the former. Both feature processors based on the PowerPC 400 processor family.

2.8.4.1 BlueGene/L 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. The modification lies in tacking on floating-point units (FPUs)that are not part of the standard processor but can be connected to the 440's APU bus. Each FPU contains two floating-point functional units capable of performing 64-bit multiply-adds, divisions and

square-roots. Consequently, the theoretical peak performance of a processor core is 2.8 Gflop/s. Figure 2.12 shows the embedding of two processor cores on a chip. As can be seen from the figure the L2 cache is very

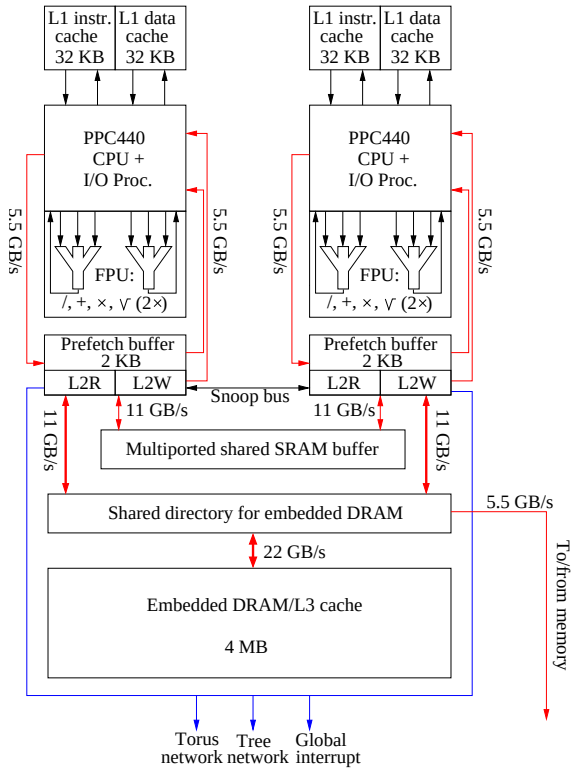


Figure 2.12: *Block diagram of an IBM BlueGene/L processor chip.*

small: only 2 KB divided in a read and a write part. In fact it is a prefetch and store buffer for the rather large L3 cache. The bandwidth to and from the prefetch buffer is high, 16 B/cycle to the CPU and 8 B/cycle to the L2 buffer. The memory resides off-chip with a maximum size of 512 MB. The data from other nodes are transported through the L2 buffer, bypassing the L3 cache in first instance. The packaging of the 2-CPU nodes in the BlueGene/L is discussed in section 3.1.12.

2.8.4.2 BlueGene/P processor

Like the BlueGene/L processor the BlueGene/P processor is based on the PowerPC core, the PowerPC 450 in this case at a clock frequency of 850 MHz and with similar floating-point enhancements as applied to the PPC 440 in the BlueGene/L. The BlueGene/P node contains 4 processor cores which brings the peak speed to 13.6 Gflop/s/node. The block diagram in Figure 2.13 shows some details. As can be seen from the Figure the structure of the core has not changed much with respect to the BlueGene/L. The relative bandwidth from the L2 cache has been maintained: 16 B/cycle for reading and 8 B/cycle for writing. In contrast to the BlueGene/L, the cores operate in SMP mode through multiplexing switches that connect pairs of cores to the two 4 MB L3 embedded DRAM chips. So, the L3 size has doubled. Also, the memory per node has increased to 2 GB from 512 MB. Like for the BlueGene/L, the packaging and network details are discussed in section 3.1.12.

2.8.5 Intel Itanium 2

The Itanium 2 is a representative of Intel's IA-64 64-bit processor family and as such the second generation. A block diagram is shown in 2.14.

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

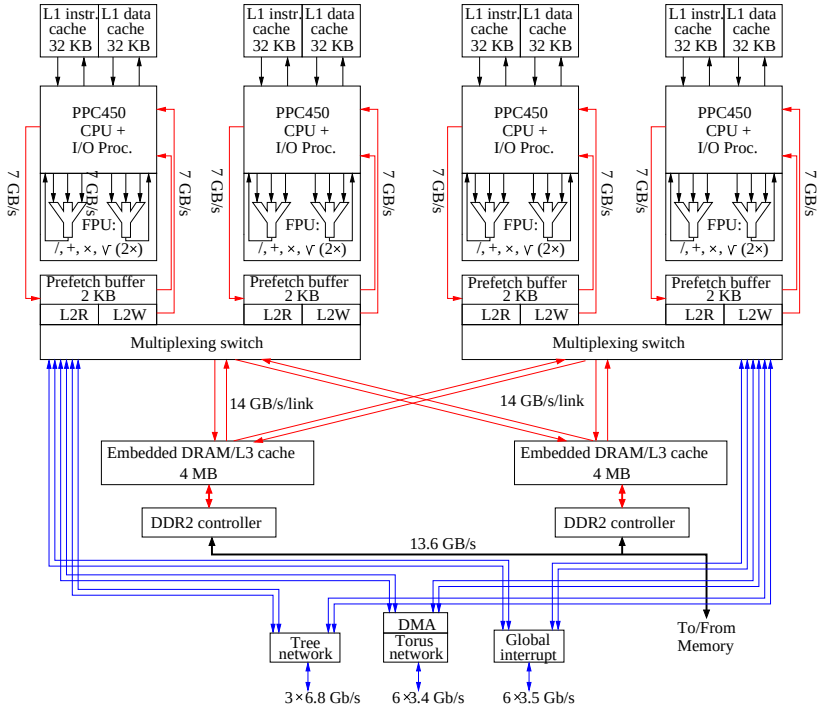


Figure 2.13: *Block diagram of an IBM BlueGene/P processor chip.*

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 1.3–1.66 GHz. At time of writing this report the current generation, the Montvale processor it around for about a year and will be replaced by its successor, code named Tukwila by the end of 2008. The processor core is almost unaltered with respect to predecessor the Montecito processor and built with 90 nm feature size. The two cores are put onto a chip working at a clock frequency of 1.66 GHz at maximum. The Montvale is a dual core processor like many other processors these days. Another feature that cannot be shown in the block diagram in Figure 2.14 is that it is dual-threaded, be it not so fine-grained as the IBM POWER6. The power requirements are lower than for the Madison processor, slightly over 100W even with two processor cores on the chip. The differences with its predecessor, the Montecito, are small: instead of a 533 MHz frontside bus a 667 MHz frontside bus is offered. It is clear that the Montvale is the end of the line for the Itanium family using a frontside bus for data access. The next generation Tukwila processor will be connected to the memory by the QuickPath Interface (QPI) at much higher speeds than with a frontside bus. The QPI bandwidth is expected to be 25.6 GB/s.

The Itanium family of processors has characteristics that are different from the RISC chips presented elsewhere in this section. Figure 2.14 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 15 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 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 validity of an operation depending

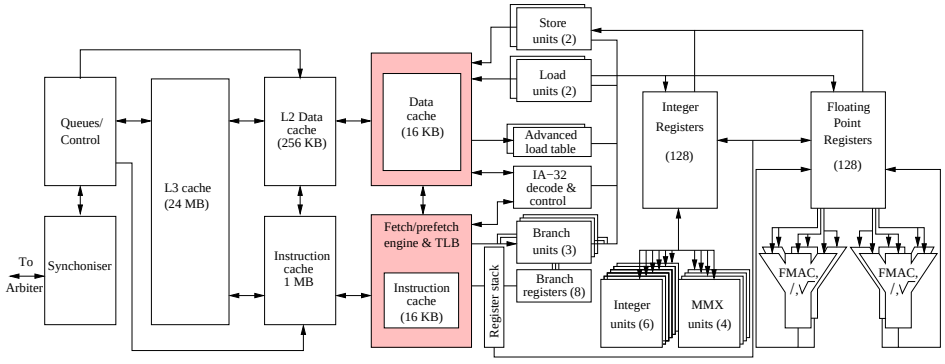


Figure 2.14: *Block diagram of the Intel Itanium 2 processor core.*

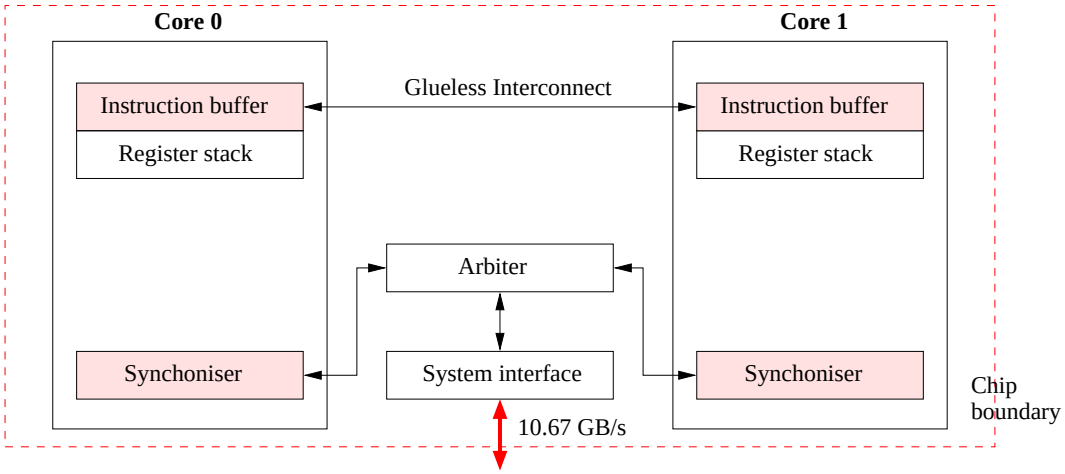


Figure 2.15: *Block diagram of 2 processor cores on a Montvale chip.*

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 2.14 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 6.7 Gflop/s at a clock frequency of 1.66 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 Montvale the latter bandwidth is no luxury.
As already remarked before, the Montvale is dual-threaded. So, when the control logic (located at the upper left in Figure 2.14) decides that no progress will be made with one thread it dispatches the other thread 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 2.15). 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.

2.8.6 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 majority 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 Intel 64[2] processor family, as this is the type to be found in clusters, mostly in 2-socket nodes but increasingly in 4-socket and 8 socket nodes.
The processor versions discussed here will replaced by the end of 2008 by a new set of processors that will not have a Front Side Bus (FSB) anymore but, analoguous to AMD's HyperTransport, a direct connection to the memory via on-chip memory controllers, named the QuickPath Interface (QPI). At the time of writing this report, however, all Xeon-based systems still use processors with a Front Side Bus and therefore we will only discuss these.
As of early 2006 Intel has introduced an enhanced micro-architecture for the IA-32 instruction set architecture called the Core architecture. The server version with the code name Woodcrest was the first implementation of this new micro-architecture. The current most aggressive dual-core processor is the X5272 Wolfdale DP. A technology shrink from 60 to 45 nm was done and the clock cycle went up to 3,4 GHz. Also the FSB bandwidth was increased going from a frequency of 1066 MHz to 1600 MHz. Thanks to the technology shrink the power requirements are the same of that of the earlier Woodcrest processor: 80 W.
In November 2007 Intel introduced a quad-core processor, code name Clovertown. In fact it featured to two Woodcrest processors on one chip while sharing the 1333 MHz frontside bus to the memory. Also the Clovertown has been succeeded. In this case by the X5482 Harpertown. Like the dual-core Wolfdale DP it has a 45 nm feature size. The clock frequency is 3.2 GHz and again the FSB frequency has gone up to 1600 MHz. The energy consumption is slightly less than double that of the Wolfdale DP: 150 W. In the following we discuss mostly the Wolfdale structure as this is essentially the same as for the quad-core Harpertown. A fairly extensive evaluation of the Clovertown processor can be found at [52].
In Figure 2.16 a block diagram of a Wolfdale 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, while the connection to the memory via the off-chip memory controller is depicted in Figure 2.17.
To stay backwards compatible with the x86 (IA-32) Instruction Set Architecture which comprises a CISC

[2]Not to be confused with the IA-64 architecture, which name indicates the Itanium processor family

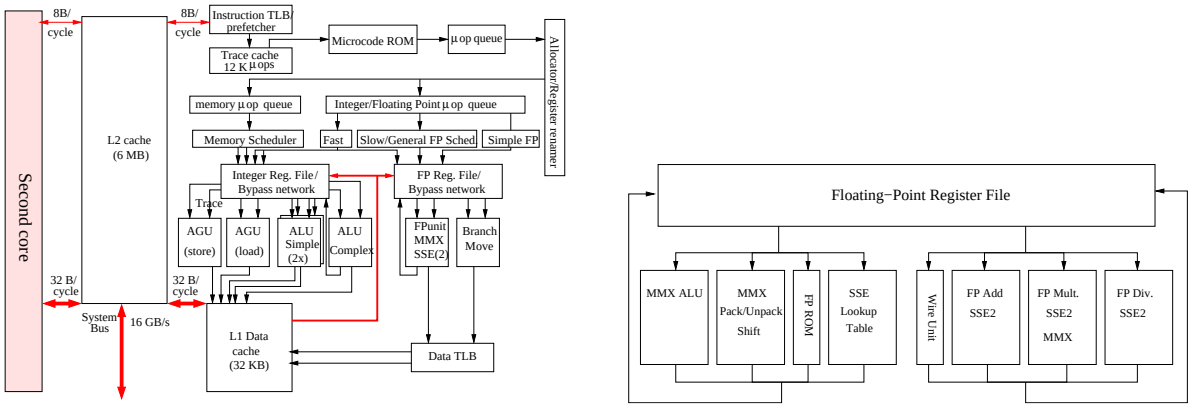


Figure 2.16: *Block diagram of an Intel Core processor core and floating-point unit.*

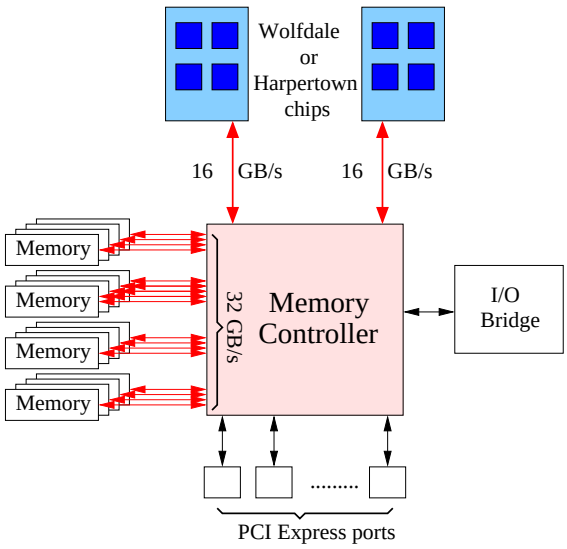


Figure 2.17: *Diagram of the connection of the memory to Wolfdale or Harpertown processors.*

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 micro-architecture. Furthermore, 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 2.16 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 4 (SSE4) repertoire of instructions, a 189-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 SSE4 units. This makes it in principle possible to achieve a 2–3 times higher floating-point performance.
The present Xeon do not support multi-threading. In the next generation, called Nehalem, multi-threading will be re-introduced (in the earlier Netburst architecture multi-threading was supported under the name of Hyperthreading)
The secondary cache has a size of 6 MB for the Wolfdale and Harpertown implementation of the Core processor. The two cores share the 1600 MHz frontside bus with a bandwidth of 16 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 extra general purpose registers(GPRs), 8 extra registers for SSE4 support, and 64-bit wide GPRs and instruction pointers.
As in the dual-core Montvale (see 2.8.5) the two cores need an arbiter on the chip to provide fair bandwidth utilisation between them. Figure 2.15.
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 or quad-core processor.

2.8.7 The MIPS processor

Quite unexpectedly the MIPS processor has turned up again in the HPC area because it is employed in SiCortex machines, see section 3.1.19. The last time a MIPS processor featured in an HPC system it was the R14000 variant that populated the SGI Origin 3000 series. Now it is back not because it is particularly fast but because of its very low power requirement at the clock frequency that is used: 500 MHz. The processor looks very much like the late R14000 processor and as such is a typical representative of simple straight-forward RISC processors. A block diagram is shown in Figure 2.18.
There are two independent floating-point units for addition and multiplication and, additionally, two units that perform floating division and square root operations (not shown in Figure 2.18). The latter, however, are not pipelined and with latencies of about 20–30 cycles are relatively slow. In all there are 5 pipelined functional units to be fed: an address calculation unit which is responsible for address calculations and loading/storing of data and instructions, two ALU units for general integer computation and the floating-

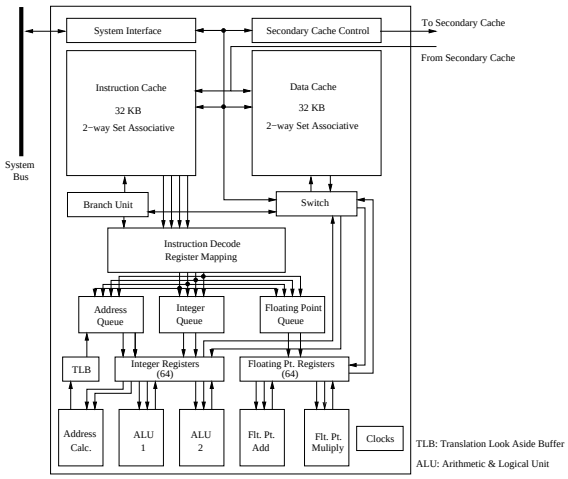


Figure 2.18: *Block diagram of a MIPS processor.*

point add and multiply pipes already mentioned.
The level 1 instruction and data caches have a moderate size of 32 KB and are 2-way set-associative. In contrast, the secondary cache can be very large: up to 16 MB but in the system they are now employed in it has a 256 KB section in a 1.5 MB shared L2 cache. Both the integer and the floating-point registers have a physical size of 64 entries, however, 32 of them are accessible by software while the other half is under direct CPU control for register re-mapping.
As already remarked, the clock frequency of the processor is quite low, 500 MHz and therefore the theoretical peak performance is 1 Gflop/s. This is not all bad. It has in fact chosen to be that low intentionally in order to have a very low power dissipation: only slightly less than 1 Watt. To achieve this the processor design had to be somewhat simplified in comparison to the former Rx000 series of processors. The instruction execution is done in-order which potentially can lower the number of instructions that can be processed per cycle. On the other hand, the discrepancy between the processor speed and the memory speed is much smaller than in other processors which leads to a higher average efficiency. Especially in codes with irregular but intensive memory access patterns the actual speed per processor might not be very much lower than experienced on other systems because of this smaller memory gap.

2.8.8 The SPARC processors

Sun has shelved its own plans to produce mainstream UltraSPARC V and VI processor by April 2004 in favour of processor designs with many (up to 16) processor cores, each capable of handling two execution threads. This so-called Rock processor should become available in the second half of 2008 and for the present the SPARC development is in the hands of its partner Fujitsu that will advance with its own SPARC64 implementation. Both Fujitsu/Siemens and Sun market servers based on the latter processor. As Sun does not actively market its UltraSPARC IV+ based servers anymore we refrain from a description of this processor and only give details of Fujitsu's SPARC64 processor line.

For quite some time Fujitsu is making its own SPARC implementation, called SPARC64. Presently the SPARC64 is in its seventh generation, the SPARC64 VII. Obviously, the processor must be able to execute the SPARC instruction set but the processor internals are rather different from Sun's implementation. Figure 2.19 shows a block diagram of one core of the quad-core SPARC64 VII.
Actually, the core achitecture has not changed from the SPARC64 VI but thanks to the decrease of the feature size from 90 nm to 65 nm, now 4 cores can be placed on a chip while the highest available clock frequency is raised from 2.4 GHz to 2.52 GHz.
The L1 instruction and data caches are 64 KB, two times smaller than in the SPARC64 VI core and both

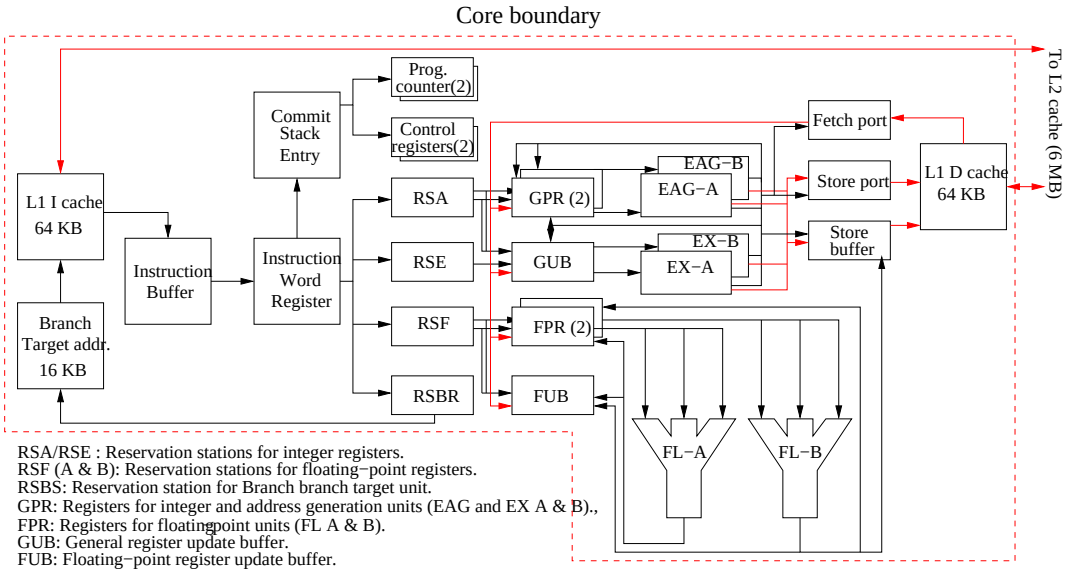


Figure 2.19: *Block diagram of a Fujitsu SPARC64 VII processor core.*

2-way set-associative. This decrease in size is somewhat surprising and probably due to the technology shrink to 65 nm feature size. 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 and -B also perform divide and square root operations in contrast to the SPARC4+ that has a separate unit for these operations. Because of their 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 VII does not have a third level cache but on chip there is a large (6 MB) unified L2 12-way set-associative write-through cache that is shared by the 4 cores in a processor as can be seen in Figure 2.20. Note that the system bandwidth is the highest available. For the lower end systems this bandwidth is about 8 GB/s.

The Memory Management Unit (not shown in Figure 2.19) 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.

What cannot be shown in the diagrams is that, like the IBM and Intel processors, the SPARC VII is dual-threaded per core. The type of multithreading is similar to that found in the Intel processors and is called Simultaneous Multithreading, differing from the type of multithreading present in the IBM processors but with the same name. At this moment the highest clock frequency SPARC64 available is 2.52 GHz. As already remarked, the floating-point units are capable of a fused multiply-add operation, like the POWER and Itanium processors, and so the theoretical peak performance is presently 10.08 Gflop/s/core

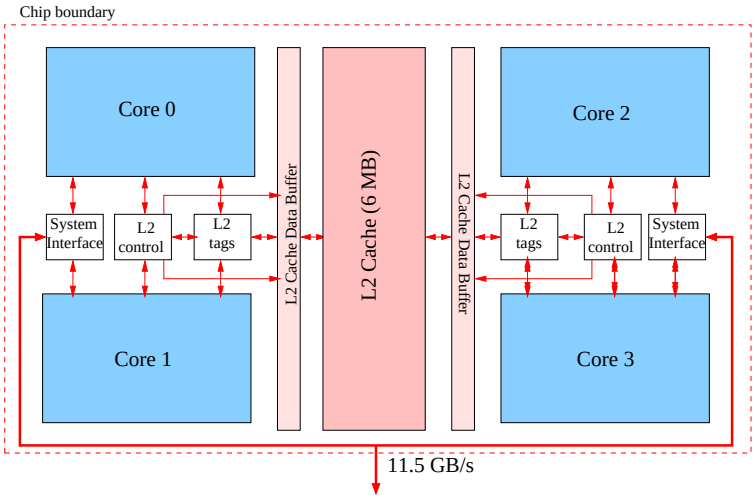


Figure 2.20: *Block diagram of a Fujitsu SPARC64 VII processor chip. Four cores share the L2 cache.*

and consequently 40.3 Gflop/s/processor.

2.9 Computational accelerators

In the last 2–3 years computational accelerators have emerged and have taken a firm foothold now. They come in various forms of which we will discuss some general characteristics. Accelerators are not a new phenomenon: in the 1980's, for instance, Floating Point Systems sold attached processors like the AP120-B with a peak performance of 12 Mflop/s, easily 10 times faster than the general purpose systems they were connected to. Also the processor array machines described in section 2.3 could be regarded as accelerators for matrix-oriented computations in their time. A similar phenomenon is on us at the moment. HPC users never tend to be content with the performance of the machines they have at their disposal and are continuously looking for ways to speed up their calculations or parts of them. Accelerator vendors are complying to this wish and presently there is a fair amount of products that, when properly deployed, can deliver significant performance gains.
The scene is roughly divided in three unequal parts:

1. Graphical cards or Graphical Processing Units (GPUs as opposed to the general CPUs).
2. General floating-point accelerators.
3. Field Programmable Gate Arrays.

The appearance of accelerators is believed to set a trend in HPC computing. Namely, that the processing units should be diversified according to their abilities. Not unlike the occurence of different functional units within a CPU core[3]. In a few years this will lead to hybrid systems that incorporate different processors for different computational tasks. Of course, processor vendors can choose to (attempt to) integrate such special purpose processing units within their main processor line but as of now it is not sure if or how this will happen.
When speaking of special purpose processors, *i.e.*, computational accelerators, one should realise that they are indeed good at some specialized computations while totally unable to perform others. So, not all applications can benefit of them and those which can, not all to the same degree. Futhermore, using accelerators effectively is not at all trivial. Although the Software Development Kits (SDKs) for accelerators have improved enormously lately, for many applications it is still a challenge to obtain a significant speedup. An important factor in this is that data must be shipped in and out the accelerator and the bandwidth of

[3]In principle it is entirely possible to perform floating-point computations with integer functional units, but the costs are so high that no one will attempt it.

the connecting bus is in most cases a severe bottleneck. One generally tries to overcome this by overlapping data transport to/from the accelerator with processing. Tuning the computation and data transport task can be cumbersome. This hurdle has been recognised by at least two software companies, Acceleware and Rapidmind. They offer products that automatically transform standard C/C++ programs into a form that integrates the functionality of GPUs, multi-core CPUs (which are often also not used optimally), and, in the case of Rapidmind, of Cell processors.
There is one other and important consideration that makes accelerators popular: in comparison to general purpose CPUs they all are very power-effective. Sometimes orders of magnitude when expressed in flop/Watt. Of course they will do only part of the work in a complete system but still the power savings can be considerable which is very attractive these days.
We will now proceed to discuss the three classes of accelerators mentioned above. It must be realised though that the developments in this field are extremely rapid and therefore the information given here will be obsolete very fast and will be of an approximate nature.

2.9.1 Graphical Processing Units

Graphics processing is characterised by doing the same (floating-point) operation on massive amounts of data. To accommodate for this way of processing Graphical Processing Units (GPUs) consist of a large amount of relatively simple processors, fast but limited local memory, and fast internal buses to transport the operands and results. Until recently all calculations, and hence the results, were in 32-bit precision. This is hardly of consequence for graphics processing as the colour of a pixel in a scene may be a shade off without anyone noticing it. HPC users often have similar computational demands as those in the graphical world: the same operation on very many data items. So, it was natural to look into GPUs with their many integrated parallel processors and fast memory. The first adopters of GPUs from the HPC community therefore disguised their numerical program fragments as graphical code (e.g., by using the graphical language OpenGL) to get fast results, often with remarkable speedups. Another advantage is that GPUs are relatively cheap because of the enormous amounts that are sold for graphical use in virtually every PC. A drawback is the 32-bit precision of the usual GPU and, in some cases more important, there is no error correction available. By carefully considering which computation really needs 64-bit precision and which does not and adjusting algorithms accordingly the use of a GPU can be entirely satisfactorily, however. GPU vendors have been quick in focussing on the HPC community. They tended to rename their graphics cards to GPGPU, general-purpose GPU, although the product was identical to the graphics cards sold in every shop. But there also have been real improvements to attract HPC users: recently the first 64-bit GPUs have come to the market. In addition, it is no longer necessary to reformulate a computational problem into a piece of graphics code. Both ATI/AMD and NVIDIA claim IEEE 754 compatibility (being the floating-point computation standard) but neither of them support it to the full. Also the error correction as is usual for general purpose CPUs is not (yet) available. So, users of these new products have an extra responsibility in validating the results of their computations. There are C-like languages and runtime environments available that makes the life of a developer for GPUs much easier. In the following we describe some high-end GPUs that are more or less targeted to the HPC community.
When one develops a code for a particular GPU platform it cannot be transferred to another without a considerable effort in rewriting the code. This drawback is taken up by the GPU vendors (and not only them). There is an initiative to define a language, OpenCL, that should be platform independent, thus protecting the development effort put into the acceleration of a program. Presently, Apple, ATI/AMD, Intel, and NVIDIA are members of the consortium that are trying to define the language.

2.9.1.1 ATI/AMD

The latest product from ATI (now wholly owned by AMD) is the ATI Firestream 9170 card. There is not enough information available for a block diagram but we list the most important features of the processor: It is expected that in the third quarter of 2008 its successor, the Firestream 9250 will come out. This card will have roughly double the performance of the Firestream 9170 and presumably will use ≤ 150 W. The specifications given indicate that per core 2 floating-point results per cycle can be generated, presumably the result of an add and a multiply operation. Whether these results can be produced independently or result from linked operations is not known because of the lack of information.

Table 2.1: *Some specifications for the ATI/AMD Firestream 9170 GPU*

Feature size	55 nm
Number of processors	320
Memory (GDDR3)	2 GB
Clock cycle	775 MHz
Peak performance	497 Gflop/s
Power requirement	≤ 100 W
Interconnect (PCIe Gen2)	$\times$16, 8 GB/s
Floating-point support	Partial (32/64-bit)

Like its direct competitor, NVIDIA, ATI offers a C-like language, BROOK+, and the accompanying run time system to ease the use of the card. The SDK containing these products is free and can be installed both for Linux (RedHat and SuSE) and Windows environments. Objects that have to be handled by the card are declared in a special syntax and there are library functions to put the data onto the card and retrieve results from it. Functions that should be performed on the card are called "Kernels". They typically operate on the stream objects defined as such in the BROOK+ program. Although this looks simple, to get an optimum performance one should tune the amount of computation carefully with the data transport, for however fast the PCIe bus might be that must transport the data to/from the GPU, there is still a significant amount of time involved in shipping the data on and off the card. BROOK+ is as yet rather restricted in its functionality. To help out in situations that are not covered by BROOK+ a assembly language, CAL can be used. This is, however, far from easy.

2.9.1.2 NVIDIA

NVIDIA is the other big player in the GPU field with regard to HPC. Its latest product is the C1060 as an individual card but it is also possible to have 4 of these cards in a 1U rack enclosure, obviously with four times the performance. Such rack-mounted systems are primarily made with the HPC community in mind. Again, we do not have enough information to provide a reliable block diagram but the most important details are given below: From these specifications can be derived that 3 floating-point results per core per cycle can

Table 2.2: *Some specifications for the NVIDIA C1060 GPU*

Number of processors	240
Memory (GDDR3)	4 GB
Clock cycle	1.3 GHz
Peak performance	936 Gflop/s
Power requirement	225 W peak, 160 W typical
Interconnect (PCIe Gen2)	$\times$8, 4 GB/s;$\times$16, 8 GB/s
Floating-point support	Partial (32/64-bit)

be delivered. Because of the scant information on the core structure it is not clear how this comes about. The power requirement given may not be entirely appropriate for HPC workloads. A large proportion of the work being done will be from the BLAS library that is provided by NVIDIA, more specifically, the dense matrix-matrix multiplication in it. This operation occupies any computational core to the full and one may expect a somewhat higher power consumption than what is considered as typical for other kinds of work.
Like ATI, NVIDIA provides an SDK comprised of a compiler named CUDA, libraries that include BLAS and FFT routines, and a runtime system that accomodates both Linux (RedHat and SuSE) and Winodws. CUDA is a C/C++-like language with extensions and primitives that cause operations to be executed on the card instead of on the CPU core that initiates the operations. Transport to and from the card is done via library routines and many threads can be initiated and placed in appropriate positions in the card memory so as not causing memory congestion on the card. This means that for good performance one needs knowledge

of the memory structure on the card to exploit it accordingly. This is not unique to the C1060 GPU, it pertains to the ATI Firestream GPU and other accelerators as well.

2.9.2 General computational accelerators

Although we have looked at the GPUs in the former section primarily from the point-of-view of computational accelerators, they are of course full-blown high-end graphical processors in the first place. However, several vendors have developed accelerators that did not have graphical processing in mind as the foremost application to be served (although they might not be bad in this respect when compared to general CPUs). Below we discuss two of these accelerators.

2.9.2.1 ClearSpeed

ClearSpeed is presently probably the only company that specifically makes computational accelerators for HPC computing. It has done this for some time which means that at this moment the ClearSpeed products are in their 3rd generation. Unlike the GPUs, the ClearSpeed processors were made to operate on 64-bit floating-point data from the start and full error correction is present in the ClearSpeed processors. The latest processor is the CSX700 chip that is packaged in a number of products. The most common is the e710 card that fits in a PCIe slot of any PC or server unit. A variant with a different form factor but with the same functionality is the e720 card that can be put into blade servers. In addition, so-called CATS units are sold that package 12 e710 cards in a 1U rack unit. The peak speed of such a CATS unit is about 1.1 Tflop/s.
There is another feature that is peculiar to the e720 card: its power consumption is extremely low, 25 W maximal, 15 W typical. This is partly due to the low clock frequency of 250 MHz. The e710 card contains, apart from the CSX700 processor, 2 GB DDR2 SDRAM, and an FPGA that manages the data traffic to and from the card. As said, the interconnect to the host system is compliant with PCIe 8×, amounting to a bandwidth of 2 GB/s. ClearSpeed is quite complete in giving technical details. So, we are able to show a block diagram of the CSX processor in Figure 2.21. Two so-called Multi-Threaded Array Procesor (MTAP)

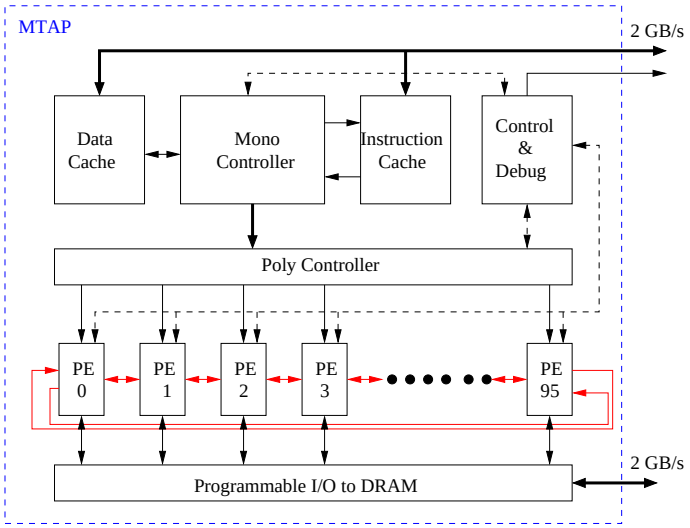


Figure 2.21: *Block diagram of a ClearSpeed MTAP unit. Two of these units reside on a CSX700 chip.*

units are located on one CSX700 chip. As can be seen an MTAP contains 96 processors (with 4 redundant ones per MTAP). The are controlled via the Poly Controller, “poly” being the indication for the data types that can be processed in parallel. The processing elements themselves are able to communicate fast between themselves via a dedicated ring network. Every cycle a 64-bit data item can be shifted to the right or to the left through the ring. In Figure 2.22 we show the details of a processing element. A maximum of two 64-bit floating-point results can be generated per cycle. As one MTAP contains 96 PEs and there are 2 MTAPs

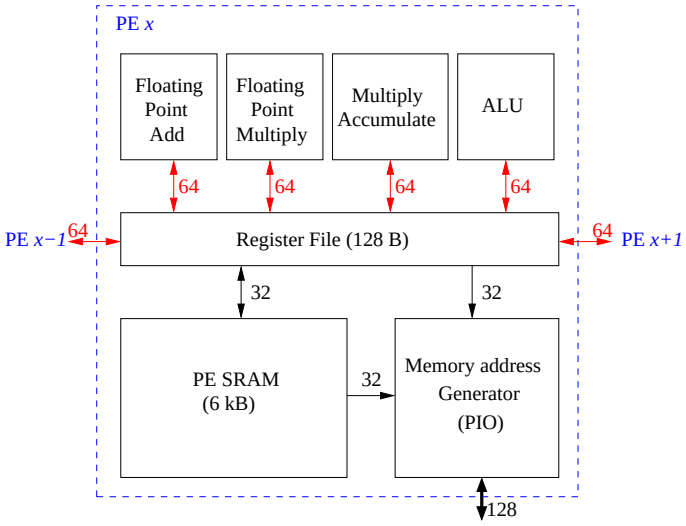


Figure 2.22: *Block diagram of a PE in an MTAP of a CSX700 chip. The numbers near the arrows indicate the number of bits that can be transferred per cycle.*

on a chip the peak performance of a CSX700 chip is 96 Gflop/s at a clock frequency of 250 MHz.
Note the Control & Debug unit present in an MTAP. It enables debugging within the accelerator on the PE level. This is a facility that is missing in the GPUs and the FPGA accelerators we will discuss later.
Also ClearSpeed employs an extended form of C, called C^n, for program development on the card. The extension is very slight, however. The keywords `mono` and `poly` are added to indicate data that should be processed serially or in parallel, respectively. Because ClearSpeed is in the accelerator trade for quite some time, the SDK is very mature. Apart from the C^n compiler already mentioned, it contains a library with a large set of the BLAS/LAPACK routines, FFTs, and Random Number generators. For dense linear algebra there is an interface that enables calling the routines from a host program in Fortran. Furthermore, a graphical debugging and optimisation tool is present that may or may not be embedded in IBM's Eclipse Integrated Development Environment (IDE) as a plug-in.

2.9.2.2 The IBM/Sony/Toshiba Cell processor

The Cell processor, offically called the Cell Broadband Engine (Cell BE), was designed at least partly with the gaming industry in mind. Sony uses it for its PS3 gaming platform and to be successful it has to deliver high performance for the graphical part as well in doing a large amount of floating-point computation to sustain the rapidly changing scenes that occur during a game. The Cell processor is therefore not a pure graphics processor but considerably more versatile than a GPU. A testimony to this is that Mercury computers, specialised in systems for radar detection, etc., markets a product with two Cell processors, instead of dedicated DSPs (i.e., Digital Signal Processors), while Toshiba incorporates the Cell in HDTV sets and considers to bring out notebooks with a Cell processor. The Cell processor is able to operate in 32-bit as well as in 64-bit floating-point mode, though there is a large performance difference: in single precision the peak speed is 204.8 Gflop/s while in double precision it is about 14 Gflop/s. From the start there was a keen interest from the HPC community. It also restarted the discussion of the necessity of using 64-bit precision calculation all the way through an application or, by reformulation some key algorithms it would not be possible to get results with acceptable accuracy when parts would be carried out in single precision [29]. At least for the Cell processor this discussion has become of less importance as at present the variant is available under the name of PowerXCell 8i which is developed by IBM, probably expressly targeted at the HPC area. In the PowerXCell the speed for 64-bit precision has increased considerably to 102.4 Gflop/s, half the speed of the single precision computations. Also it is produced in 65 nm instead of 90 nm technology and it employs DDR2 memory instead of Rambus memory which is used in the original Cell processor. Figure 2.23 shows a diagram of this rather complicated processor. As can be seen, the processor is hybrid in the

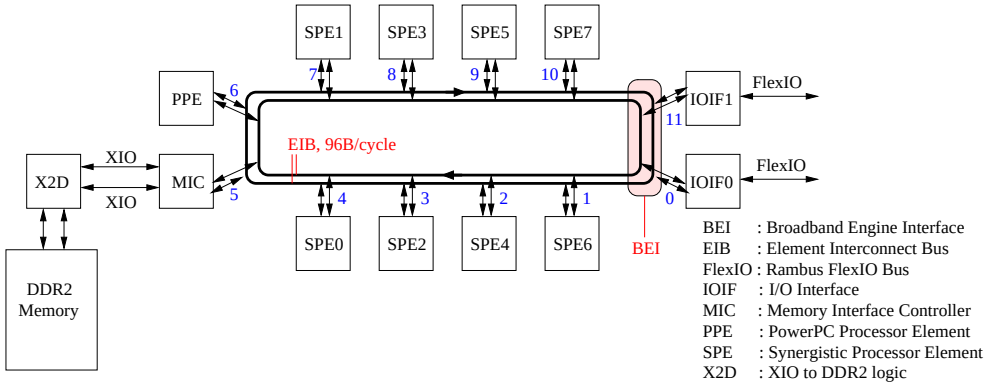


Figure 2.23: *Block diagram of an IBM PowerXCell processor. The blue numbers in the figure indicate the device numbering used for delivering data via the Element Interconnect Bus.*

sense that it contains two different kinds of processors: The PPE, which is essentially a PowerPC core as discussed in section 2.8.3, and 8 SPEs all running at a clock frequency of 3.2 GHz. The SPEs are meant to do the bulk of the computation, while the PPE takes care of operating system tasks and coordinating the work to be done by the SPEs. All devices in the processor are connected by the Element Interconnect Bus. The EIB in fact consists of four 16B wide rings that transport data in opposite directions as to minimise the distance between the devices in the processor. The devices connected to the EIB are numbered to allow data to be transferred from one device to another can easily be delivered. Up to 96 B/cycle can be transferred, amounting to 307.2 GB/s. Although the PowerXCell uses DDR2 memory, the processor proper is designed for use with Rambus memory. This has been taken care of by including the X2D device that translates the DDR memory requests into Rambus requests and vice-versa. The two I/O Interfaces are controlled through the Broadband Engine Interface (BEI). They have different functions: IOIF1 takes care of the usual external I/O devices via the IOIF protocol while IOIF0 is able to use the internal I/O protocol, BIF that is also used on the EIB rings. In this way it is possible to connect to other Cell processors.
The SPEs are the computational workhorses in the Cell processor. We show the internals of an SPE in Figure 2.24. Roughly, there are three important parts in an SPE: the SXU (Synergistic Execution Unit)

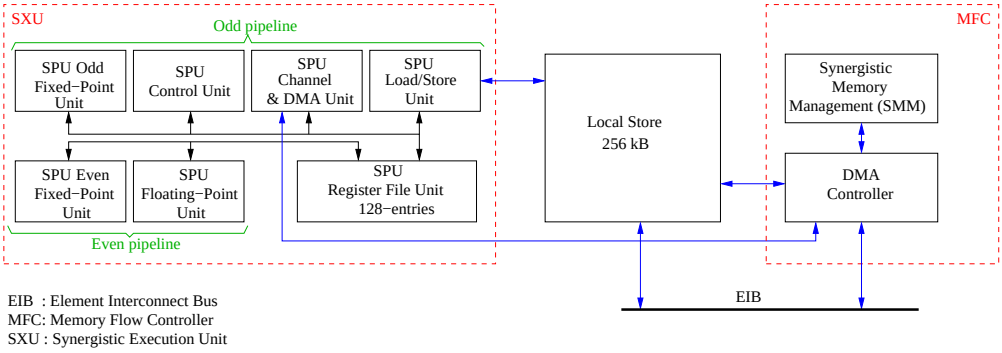


Figure 2.24: *Block diagram of an IBM PowerXCell Synergistic Processing Element (SPE).*

which contains the functional units for computation, load/store, and DMA-control, the Local Store that contains the local data to be operated on, and the Memory Flow Controller that in turn contains the DMA Controller and the memory management unit. As shown in Figure 2.24 in the SXU the functional units are organised in an odd and an even pipeline. Two instructions can be issued every cycle, one for each of these pipelines. This also implies that one floating-point instruction can be issued per cycle. Depending on type of the operands this can yield four 32-bit results or two 64-bit results per cycle (in the PowerXCell, in the original Cell processor a 64-bit result can be delivered every 13 cycles, hence the much lower double

precision performance). Note that SPE does not have any form of cache. Rather, data is brought in from external memory by DMA instructions via the EIB. This leads to much lower memory latency when a data item is not in the Local Store. Up to 16 DMA requests can be outstanding for any of the SPEs. As all SPEs are independent up to 128 DMA requests can be in flight. Of course, this explicit memory management makes not for easy programming. So, one must be careful in managing the data to get (close to) optimal performance.
IBM has put much effort into a Software Development Kit for the Cell processor. It is freely available and apart from the necessary compilers, there is an extensive library for managing the data transport both from the PPE to the SPEs, between SPEs, initiating the processes on the SPEs, retrieving the results, and managing program overlays. As the Local Stores in the SPEs are small, the old concept of overlays has been revived again: The program is divided into units that depend on each other but do not constitute the whole program. By loading and unloading the units in the correct sequence one can still execute the total program. In addition, there are debugging and performance analysis tools. The total of the program development can be done using IBM's IDE, Eclipse.
The PowerXCell 8i won its share of fame for it's use in the Roadrunner system at Los Alamos National Laboratory. In this system 3240 so-called triblades are connected by InfiniBand. A triblade consists of 2 QS22 blades each containing 2 PowerXCell processors and an LS21 blade with 2 Opteron processors. This configuration was the first to break the LINPACK Petaflop barrier. This fact certainly helped in increasing the interest in the Cell processor as an accelerator platform. Presently, there are many research projects under way to assess the applicability of Cell BE accelerators and to make the learning curve for employing them effectively, less steep.

2.9.3 FPGA-based accelerators

An FPGA (Field Programmable Gate Array) 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. Moreover, it is possible to configure a multiple of these units on an FPGA that work in parallel. So, potentially, FPGAs may be good candidates for the acceleration of certain applications. Because of their versatility it is difficult to specify where they will be most useful. In general, though, they are not used for heavy 64-bit precision floating-point arithmetic. Excellent results have been reported in searching, pattern matching, signal- and image-processing, encryption, etc. The clock cycle of FPGAs is low as compared to that of present CPUs: 100–550 MHz which means that they are very power effective. All vendors provide runtime environments and drivers that work with Linux as well as Windows.
Tradionally, FPGAs are configured by describing the configuration by means of a hardware description language (HDL), like VHDL or Verilog. This is very cumbersome for the average programmer as one not only has to explicitely define such details as the placement of the configured devices but also the width of the operands to be be operated on, etc. This problem has been recognised by FPGA-based vendors and a large variety of programming tools and SDKs have come into existence. Unfortunately, they differ enormously in approach and the resulting programs are far from compatible. Also for FPGA-based accelerators, like for GPUs, there is an initiative to develop a unified API that will assure compatibility between platforms. The non-profit OpenFPGA consortium is heading this effort and at the time of writing this report a 0.4 version of the API is out for comment by vendors and users. Meanwhile one has to make do with the many dialects and APIs that are around in this area.
The two big players on the FPGA market are Altera and Xilinx. However, in the accelerator business one seldom will find these names mentioned, because the FPGAs they produce are packaged in a form that makes them usable for accelerator purposes.
It is not possible to fully discuss all vendors that offer FPGA-based products. One reason is that there is a very large variety of products ranging form complete systems to small appliances housing one FPGA and the appropriate I/O logic to communicate with the outside world. To complicate matters further, the FPGAs themselves come in many variants, e.g., with I/O channels, memory blocks, multipliers, or DSPs already configured (or even fixed) and one can choose for FPGAs that have for instance a PowerPC405 embedded. Therefore we present the FPGA accelerators here only in the most global way and necessarily incomplete.
We will now discuss some of the accelerator products on the market and in what way they are made useful.

2.9.3.1 DRC

DRC sells various products based on the Xilinx Virtex-5 FPGA. These products range from a co-processor, called the Accelium 2000, that can be directly connected to AMD's HyperTransport, via 2U and 4U rack-mounted servers to a tower workstation in which up to 3 co-processors can be accomodated. There is an integrating software environment that contains the DRC API and driver on the CPU side (in practice an Opteron processor) and DRC's hardware OS on the co-processor side. DRC modules with the Xilinx Virtex-4 LX200 are employed in the Cray XR1 blades that fit into the Cray XT5 infrastructure, thus making it a hybrid computer.
Software development for the DRC processors can, e.g., be done using Handel-C or Mitrion-C that can generate loadable code for Altera as well as Xilinx FPGAs. On one hand there are, however, restrictions with regard to standard C, on the other hand there are extensions, for instance to describe operand width. It therefore is highly chip-specific whether the code will run satisfactorily (or sometimes at all). Good Developments Kits, like that of Mitrion for instance, will notify the user, however, when he/she wants the impossible.

2.9.3.2 Nallatech

Nallatech offers an enhancement for IBM BladeCenter hardware, both as complete cards and as an expansion card. The H102 cards contain 2 Xilinx Virtex-4 LX-100 FPGAs. In addition a general accelerator card, H101, with 1 Virtex-4 is available that can be connected via a PCI-X slot. Floating-point arithmetic is pre-configured. For the H102 cards the peak speed is 12.8 Gflop/s in 64-bit presicion and 40 Gflop/s in 32-bit precision while it is half this speed for the PCI-X-based H101 card.
Nallatech has its own API, FUSE, to manage the configuration, control, and communication with the rest of the system. With it comes Dimetalk that for some functions transform restricted type of C code, DIME-C, into VHDL and can generate the communication network between the various algorithm blocks, memory blocks I/O interfaces.

2.9.3.3 Pico

Pico markets its so-called E-12 cards in two varieties: the E-12 LX25 contains a Xilinx Virtex-4 LX25 FPGA while the E-12 FX12 has a similar FPGA but with a 450 MHz PowerPC 405 core embedded. The latter can be convenient when one also wants operating systems tasks to be done of which the results directly impact the operation on the FPGA. This is, however, generally not the case in HPC applications. Furthermore, a tower model workstation with up to 15 E-12 LX25 cards is available as well as a 4U rack-mounted model with room for 77 of the larger Virtex-4 LX50 cards. Performance measures given are typical for FPGA systems: 32×10^9 18×18 fixed-point multiplies/s (a standard in image processing), 400 million DES encryption keys/s, etc.
As Pico exclusively uses Xilinx Virtex FPGAs it is not surprising that it offers the full Xilinx standard toolset, EDI, EDK, and Platform EDK. The capabilities of these tools remains, however, unclear as the information content of the online product description is minimal.

2.9.3.4 SGI RASC

Since a few years SGI has its RASC RC100 blade on the market. It contains 2 Xilinx Virtex-4 LX200 FPGAs as well as NumaLink connection ports to fit it into SGI's Altix 4700 and 450 systems as accelerator boards. It can be regarded as an expression of SGI's interest in hybrid computing systems. Because the product is there already for some years there is a reasonable software base in the form of a library, RASClib, and various developing platforms are supported, including Handel-C, Impulse Codeveloper/Impulse-C, and Mitrion-C.

2.9.3.5 SRC

SRC is the only company that sells a full stand-alone FPGA accelerated system, named the SRC-7. Besides that the so-called SRC-7 MAP station is sold, the MAP being the processing unit that contains 2 Altera Stratix II FPGAs. Furthermore, SRC has the IMAP card as a product that can be plugged in a PCIe slot

of any PC.
SRC has gone to great length to ban the term FPGA from its documentation. Instead it talks about implicit vs. explicit computing. In SRC terms implicit computing is performed on standard CPUs while explicit computing is done on its (reconfigurable) MAP processor. The SRC-7 systems have been designed with the integration of both types of processors in mind and in this sense it is a hybrid architecture also because shared extended memory can be put into the system that is equally accessible by both the CPUs and the MAP processors. We show a sketch of the machine structure in Figure 2.25. It shows that CPUs and MAP

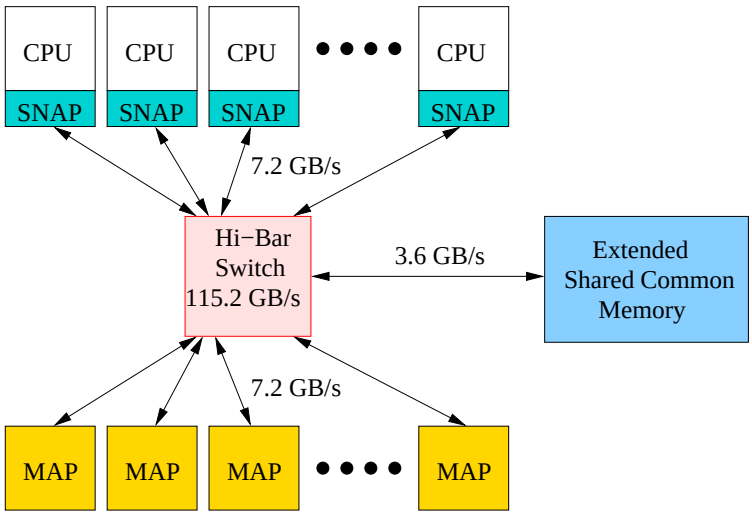


Figure 2.25: *Approximate machine structure of the SRC-7.*

processors are connected by a 16×16 so-called Hi-Bar crossbar switch with a link speed of 7.2 GB/s. The maximum aggregate bandwidth in the switch 115.2 GB/s, enough to route all 16 independent data streams. The CPUs must be of the x86 or x86_64 type. So, both Intel and AMD processors are possible. As can be seen in the Figure the connection to the CPUs is made through SRCs proprietary SNAP interface. This accommodates the 7.2 GB/s bandwidth but isolates it from the vendor-specific connection to memory. Instead of configuring a MAP processor, also common extended memory can be configured. This allows for shared-memory parallelism in the system across CPUs and MAP processors.
The MAP station is a shrunk version of the SRC-7: it contains a x86(_64) CPU, a MAP processor, and a 4×4 Hi-Bar crossbar that allows Common Extended memory to be configured.
SRC is the only accelerator vendor that supports direct support for Fortran through its development environment Carte. In addition, of course also C/C++ is available. The parallelisation and acceleration are largely done by putting comment directives in Fortran code and pragmas in C/C++ code. Also, explicit memory management and prefetching can be done in this way. The directives/pragmas cause a bitstream to by loaded onto the FPGAs in one or more MAP processors that configures them and executes the target code. Furthermore, there is an extensive library of functions, a debugger and a performance analyzer. When one wants to employ specific non-standard funtionality, e.g., computing with arithmetic of non-standard length, one can create a so-called Application Specific Funtional Unit. In fact, one then configures one or more of the FPGAs directly and one has to fall back on VHDL or Verilog for this configuration.

2.10 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 that was used (mostly Ethernet). 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 (GbE) 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. Furthermore, also 10 Gigabit Ethernet (10 GigE) is increasingly offered. The adoption of Ethernet is hampered by the latencies that are incurred when the TCP/IP protocol is used for the message transmission. In fact, the transmission latencies without this protocol are much lower: about 5 μs for GbE and 0.5 μs for 10 GigE. Using the TCP/IP protocol, however, gives rise to latencies of somewhat less than 40 μs and in-switch latencies of 30–40 μs for GbE and roughly a 10 μs latency for 10GbE. This makes that the applicability is too restricted to be the network of choice (except perhaps for price reasons). Various vendors, like Myrinet and SCS, have circumvented the problem with TCP/IP by implementing their own protocol thus using standard 10GigE equipment but with their own network interface cards (NICs) to handle the proprietary protocol. In this way latencies of 2–4 μs can be achieved: well within the range of other network solutions.

We restrict ourselves here to networks that are independently marketed as the proprietary networks for systems like those of Cray 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 presently mostly superseded by its successor PCI-Express 1.1 (PCIe). 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 fast enough for the host bus adapters of any communication network vendor sofar. So, for the networks discussed below often different bandwidths are quoted, depending on the PCI bus type and the supporting chip set. Therefore, when speeds are quoted it is always with the proviso that the PCI bus of the host node is sufficiently wide/fast.
Lately, PCIe 2, commomly known as PCIe Gen2 has emerged with a two times higher bandwidth. Currently PCIe Gen2 is mostly used within servers to connect to high-end graphics cards (including GPUs used as computational accelerators) at speeds of 4–8 GB/s but evidently it could also be used to connect to either other computational accelerators or network interface cards that are designed to work at these speeds.

An idea of network bandwidths and latencies for some networks, both propriety and vendor-independent is given in Table 2.10. Warning: The entries are only approximate because they also depend on the exact switch and host bus adapter characteristics as well as on the internal bus speeds of the systems. The circumstances under with these values were obtained was very diverse. So, there is no guarantee that these are the optimum attainable results. In Table 2.10 the Kautz graph [4] is another interesting logarithmic

Table 2.3: *Some bandwidths and latencies for various networks as measured with an MPI Ping-Pong test.*

	Bandwidth	Latency
Network	GB/s	μs
Cray SeaStar2 (measured)	2.1	4.5
IBM (Infiniband) (measured)	1.2	4.5
SiCortex Kautz graph (stated)	2.0	1.0
SGI NumaLink (measured)	2.7	1.2
Infiniband (measured)	1.3	4.0
Infinipath (measured)	0.9	1.5
Myrinet 10-G (measured)	1.2	2.1
Quadrics QsNet^{II} (measured)	0.9	2.7

network that was not discussed before and which appears in the new SiCortex systems. It has a diameter Ω that grows logarithmically with its size and has high connectivity which makes it resilient against network faults. A small non-trivial Kautz network is shown in Figure 2.26.

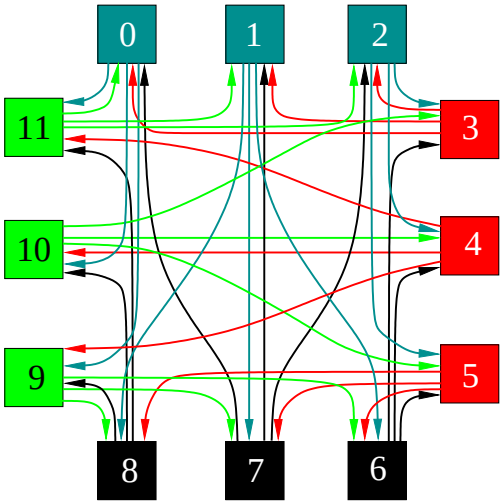


Figure 2.26: *A $\Omega = 2$, degree 3 Kautz graph.*

2.10.1 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 [42]. 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 has become relatively inexpensive because the high volume of sellings is presently 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 normally offered with 8–288 ports and presently mostly at a speed of 1.25 GB/s. However, Sun is now providing a 3456-port switch for its Constellation cluster systems. Switches

and HCAs accommodating double this speed (double data rate, DDR) are now common. 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 depends of course 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 experiment on Infiniband-based clusters with different MPI implementations has shown bandwidths of 1.3 GB/s and an MPI latency of 4 μs for small messages is quoted by Mellanox, one of the large Infiniband vendors. The in-switch latency is typically about 200 ns. For the 2.5 GB/s products the MPI bandwidth indeed about doubles while the latency stays approximately the same. At the time of writing this report, quad data rate (QDR) Infiniband products are starting to become available but no reliable bandwidth and latency results are known yet. A nice feature of QDR Infiniband is that it provides dynamic routing which is not possible with the earlier generations. In complicated communication schemes this feature should alleviate the contention at some data paths by letting take the message an alternative route.
Because of the profusion of Infiniband vendors of late, the price is now at par with or lower than those of other fast network vendors like Myrinet (2.10.3) and Quadrics (2.10.4).

2.10.2 InfiniPath

InfiniPath only provides Host Channel Adapters with a 4-wide (1.25 GB/s) Infiniband link on the network side and connecting to a HyperTransport bus or PCI-Express at the computer side. For systems with AMD processors on board the HyperTransport 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 InfiniPath 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 [12] 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.

2.10.3 Myrinet

Until recently Myrinet was the market leader in fast cluster networks and it is still one of the largest. The Myricom company which sells Myrinet started in 1994 with its first Myrinet implementation, [33], 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.
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 of PCI Express bus of the host they are attached to. 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 damage 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

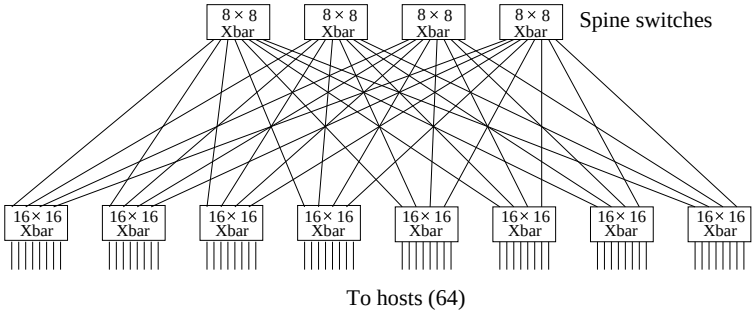


Figure 2.27: *An 8×16 Clos network using 8 and 16 port crossbar switches to connect 64 processors.*

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 2.27. 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: just over 2 μs .

2.10.4 QsNet

QsNet is the main 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 into 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 three 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 three sizes: with 8, 32, and 128 ports but nothing between 32 and 128 ports. Of course one can put together networks for other sizes but this goes at the cost of compactness and speed. 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 [38] 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. With PCI Express this limitation is over and can go up to just over 1 GB/s. 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 allows to write/read to/from 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.

3 Recount of (almost) available systems

In this section we give a recount of the types of systems 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 Fujitsu systems under entries M9000 and PRIMEQUEST because they have a different structure.

As already remarked in the Introduction we will not discuss clusters here but restrict ourselves to "integrated" parallel systems. However, the distinction between clusters and integrated systems becomes less and less clear as the network speed of clusters and integrated systems are (almost) comparable and the number of cores in a cluster node also approaches (or in the case of the Cray XT4/5 is larger than) the number found in the integrated systems. The criteria we use consist of the special measures a vendor has taken to more tightly integrate the nodes in the system than what is found in the standard cluster. Such measures may be in hardware, like special barrier registers, or software, like a vendor-optimised fast intra-node MPI, or both. It may well be that even these kinds of distinctions will disappear over time. In that case the difference will have disappeared completely and we will add the (former) clusters to this overview. Still, we want to mention a few systems here that do not fall within the criteria maintained in the regarding clusters because they represent important systems but cannot be described easily in the usual terms for one or more of various reasons. So, below one will find a few entries that give some information about such configurations but cannot always provide the details one should like to give. Just because we want the reader to be aware of these systems we include them. Because the criteria are so vague, we cannot claim consistency in this respect.

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 ought 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) 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 5325.
Operating system: Linux, Windows Server 2008, 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 5325
Clock cycle	1.6 GHz
Theor. peak performance	204.8 Gflop/s
Main Memory	16–512 MB
No. of processors	8–32
Communication bandwidth	
Point-to-point	6.4 GB
Aggregate peak	25.6 GB

Remarks:
The NovaScale 5005 series is the second generation of Itanium-2 based systems targeting the HPC field. Besides the model listed under System Parameters it also includes the 5085, 5165, and 5245 models which we do not discuss separately as they are simply models with a maximum of 8, 16, and 24 processors, respectively. The main difference with the first generation, the 5230 system, 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. In about all other regards the new series is equal to the first generation.
The NovaScales are ccNUMA systems. 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 2.10.4). Bull provides its own MPI implementation which turns out to be very efficient (see "Measured Performances" below and [50]).
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.

Bull has introduced a new server series by mid 2006, the NovaScale 3005. The largest one, the 3045, contains 4 Montvale processors and as such due to its 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. Bull offers the 3005 nodes with Quadrics QsNet^{II} as the standard network medium for large configurations. 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 [50]).
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 ([54]).

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 2.10.3). 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 ([54]) 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. 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/index.html`
Year of introduction: 2004.

System parameters:

Model	Cray XT3
Clock cycle	2.6 GHz
Theor. peak performance	
Per processor	5.2/10.4 Gflop/s
Per Cabinet	499.2/998.4 Gflop/s
Max. Configuration	160/319 Tflop/s
Memory	
Per Cabinet	$\leq$ 768 GB
Max. Configuration	239 TB
No. of processors	
Per Cabinet	96
Max. Configuration	30,508
Communication bandwidth	
Point-to-point	$\leq$3.8 GB/s
Bisectional/cabinet	333 GB/s

Remarks:
The Cray XT3 is the commercial spinoff of the 26000+ processor Red Storm machine, built by Cray for Sandia Laboratories. Although the successors, the XT4 and the XT5 are already on the market, Cray seems to see value in it as product for it is still figuring as such on their web pages. The structure of the XT3 is similar to that of the Red Storm, 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 an AMD Opteron 100, at 2.6 GHz or its dual core follow-on, hence the double entries in the systems parameters table. Cray initially had chosen for the uniprocessor version 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 6.4 GB/s 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 and the relatively ease of connecting the processor 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 [37]). 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, has been upgraded to a dual core XT3-like system, at a clock frequency of 2.4 GHz. In [54] a speed of 101.4 Tflop/s out of 127.4 Tflop/s is reported on a 26,544 core system: an efficiency of 80%. ORNL in the USA reports in [54] a performance of 101.7 Tflop/s on a 23,016-processor regular XT3/XT4 system mixture for a linear system of size 2,220,160 with an efficiency of 85%.

3.1.4 The Cray Inc. XT4

Machine type: Distributed-memory multi-processor.
Models: XT4.
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/xt4/index.html`
Year of introduction: 2006.

System parameters:

Model	Cray XT4
Clock cycle	2.6 GHz
Theor. peak performance	
Per processor	5.2/10.4 Gflop/s
Per Cabinet	499.2/998.4 Gflop/s
Max. Configuration	160/319 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	667 GB/s

Remarks:
Architecturally the Cray XT4 is very much alike its predecessor, the Cray XT3, which is still marketed next to the XT4 (see 3.1.3). The main structure with respect to the internode network and the nodes themselves are still the same. The only substantial difference, being the point-to-point network speed, has doubled to a 7.6 GB/s bi-directional link. In addition, the memory bandwidth is doubled from 6.4 GB/s to 12.8 GB/s. However, the speed to the communication router, the SeaStar2 has stayed the same at 6.4 GB/s, the standard HyperTransport speed.
Like in the XT3, on the compute nodes Cray's UNICOS/lc microkernel is employed and for the same reason: to combat OS-jitter.

Measured Performances:
As already given in the Performance section of 3.1.3, ORNL in the USA reports in [54] a performance a performance of 101.7 Tflop/s on a 23,016-processor regular XT3/XT4 system mixture for a linear system of size 2,220,160 with an efficiency of 85%. Furthermore, additional benchmark results for this system are available from Kuehn *et. al.*, [28].

3.1.5 The Cray Inc. $XT5_h$

Cray has come a long way in the realisation of true hybrid systems, meaning that processors of different type can work seamlessly together in one computer infrastructure. Cray is consistently working towards a hardware infrastructure that allows for this ideal situation. The current product is named the Cray $XT5_h$ that can harbour scalar Opteron-based XT5 nodes, X2 vector nodes, and XR1 nodes that house a Xilinx FPGA. The $XT5_h$ is not yet able to let them work all together at one computational task but SIO (Service and I/O nodes) can log on to the system and deal with any of the different types of nodes within the system. The common infrastucture is formed by the 3-D torus network implemented by the SeaStar2+ communication nodes. Each of the 6 ports of the SeaStar2+ provides a bandwidth of 9.6 GB/s and is so connected to the X2, XT5, and XR1 blades. The bandwidth to the X2 blades is 4.8 GB/s for reasons not disclosed in Cray's documentation.
Below we will discuss the different types of blades of the $XT5_h$. It is not possible to give the maximum performance of the system as a whole but we can, at least for the X2, and XT5 blades do so. Obviously this is not possible for the XR1 blades as the performance is totally dependent on the algorithm that is executed.

3.1.5.1 The Cray Inc. X2

Machine type: Shared-memory multi-vectorprocessor.
Models: Cray X2 blade.
Operating system: UNICOS/lc, Cray's microkernel Unix.
Connection structure: Fat Tree.
Compilers: Fortran 95, C, C++, Co-Array Fortran, UPC.
Vendors information Web page: `www.cray.com/products/xt5/index.html`
Year of introduction: 2007.

System parameters:

Model	Cray X2
Clock cycle	1.6 GHz
Theor. peak performance	
Per processor	25.6 Gflop/s
Per node	102.4 Gflop/s
Per blade	204.8 Gflop/s
Maximal	3.3 Pflop/s
Memory/node	32/64 GB
No. of processors	$\leq$32576
Memory bandwidth	28.5 GB/s

Remarks:
The X2 vector processor blade fits in Cray's $XT5_h$ infrastructure where $_h$ stands for "hybrid". The processor is not very different from its predecessor, the X1E. The clock frequency has gone up from 1.125 GHz to 1.6 GHz. A vector pipe set, containing an add-, multiply-, and a miscellaneous functions pipe, can generate 2 floating-point results/cycle. As there are 8 pipe sets in a processor this yields a peak performance of 25.6 Gflop/s. Four CPUs are housed in one node with 32 or 64 GB memory and operate in SMP mode. One X2 blade in turn houses two nodes for a total peak performance of 204.8 Gflop/s. Like in its predecessors, the bandwidth from/to the memory of 28.5 GB/s is not sufficient to support the the operation of the processors at full speed. So, the scalar processor in a CPU has 2-way set-associative L1 instruction and data caches, while the unified 512 KB, 16-way set-associative L2 cache and the 8 MB L3 cache are shared by all data. Part of the bandwidth discrepancy has been met by decoupling the vector functional units from each other. For instance, the load/store unit is able to issue store instructions before the associated results are present. In this way functional units incur minimal waiting times because they do not have to synchronise unnecessarily. To integrate the X2 blade fully in the $XT5_h$ infrastructure it is connected to a SeaStar2+ router that connects it to the other types of nodes in the system, however at half the normal speed link at 4.8 GB/s. The X2 blades have also their own very high bandwidth interconnect, implemented as a dense fat tree. Cray uses its proprietary so-called YARC router chips. The density lies in the fact that the radix, i.e., the amount of ports/router is high: 64 ports/YARC chip. Four of these chips constitute one Rank 1 router, that connects 32 CPUs at level 1. A rank 2 router can in turn connect 128 rank 1 routers, etc., up to the maximum of 32K processors. A unique feature of the network is that it allows *side links* that connect rank 1 routers in order to connect subtrees statically. One may have reasons to configure the network in such a way, especially when one has a number of X2 boards that naturally matches with such a configuration. The network topology can become rather complicated in this way, however. The worst case distance $\Omega = 7$ in the maximally configured network which means that only 7 hops are needed to connect two of the most distant processors in a 32K processor configuration. The point-to-point bandwidth between processor is quite high: 15 GB/s or almost 60% of the local memory bandwidth within a CPU. An extensive description of this interesting interconnect can be found in [40]. In an SMP node `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:
Results of a subset of the synthetic HPCC benchmark, some application kernels 7 application code are compared to an Intel Woodcrest processor (HPCC subset) and an Cray XT4 are discussed in [41].

3.1.5.2 The Cray Inc. XT5

Machine type: Distributed-memory multi-processor.
Models: XT5.
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/xt5/index.html`
Year of introduction: 2007.

System parameters:

Model	Cray XT5
Clock cycle	—
Theor. peak performance	
Per processor	—
Per Cabinet	7 Tflop/s
Max. Configuration	—
Memory	
Per Cabinet	$\leq$ 6.14 TB
Max. Configuration	—
No. of processors	
Per Cabinet	192
Max. Configuration	—
Communication bandwidth	
Point-to-point	$\leq$9.6 GB/s
Bisectional/cabinet	842 GB/s

Remarks:
The quality of the layout of the online product information of Cray has improved siginificantly over the years. Unfortunately, the information content, at least for the XT5, has been decreasing at the same rate. So, no more information is given about the processor on the XT5 blade than that AMD 2000 processors are used and may be either dual- or quad-core. No per processor performance is given either, only that a 192-processor, quad-core, cabinet will have a peak performance of 7 Tflop/s. This amounts to a peak speed/core of 9.11 Gflop/s. As the fastest 2.5 MHz quad-core Phenom processor is capable of 5.0 Gflop/s/core with its regular floating-point units, it turns out that Cray conveniently has counted in the SSE capability of the processors. A practice that has not been exercised before, nor should be recommended. Also no information is given about the maximum configuration as is done for its predecessors, the XT3 and XT4.
There *is* information about the interconnection network: is it based on the SeaStar2+ router. The structure is the same as that of the SeaStar2 router but faster by about 25%. The bi-directional link speed has gone up from 7.6 GB/s to 9.6 GB/s. The bandwidth from the SeaStar2+ to the processors is still the same: 6.4 GB/s, the generic speed of AMD's HyperTransport 1.1.
At the moment no XT5 systems have been installed yet. However, in a $XT5_h$ also XT4 blades can be accommodated for those who cannot wait.

Measured Performances:
As yet, there are no performance results known for XT5 systems.

3.1.5.3 The Cray Inc. XR1

Machine type: Distributed-memory multi-processor.
Models: XR1.
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/xt5/index.html`
Year of introduction: 2007.

System parameters:

Model	Cray XR1
Clock cycle	500 MHz
Memory	
Per FPGA	200,448 LUTs
Bandwidth CPU–FPGA	3.2 GB/s
No. of processors	
Per blade	2
Max. Configuration	—
Communication bandwidth	
Point-to-point	≤9.6 GB/s

Remarks:
An XR1 blade contains two Xilinx Virtex-4 LX400 FPGAs, each packaged as DRC's Reconfigurable Processing Units (RPUs). This means that DRC already has preconfigured the I/O connection between the AMD opteron on the board and the two RPUs. The connection is directly via AMD's HyperTransport 1.0 protocol, making the bandwidth off and on the FPGAs quite high in comparison to most accelerator platforms. The AMD processor is used for connection to the SeaStar2+ router in the $XT5_h$ infrastructure and as an intermediate processor from where the RPUs are activated. As the algorithms to be configured and run on an XR1 board can be so different no sensible performance figures can be given, even as an upper bound.
As already discussed in section 2.9.3.1 various programming interfaces can be used, including Handel-C, Mitrion-C, etc., for the development of routines to be run on the XR1 boards.

Measured Performances:
As said, because of the enormous difference in speed between one application and the other no meaningful performance number can be defined. Only in terms of speedup against a CPU-based version one can reasonably assess what the benefits of an XR1-accelerated system are.

3.1.6 The Cray Inc. XMT

Machine type: Shared-memory multi-processor.
Models: XMT.
Operating system: MTK, a multi-threaded Unix variant (Linux on service nodes).
bf Connection structure: 3-D Torus.
Compilers: C, C++.
Vendors information Web page: `www.cray.com/products/xmt/`
Year of introduction: 2006.

System parameters:

Model	Cray XMT
Clock cycle	500 MHz
Theor. peak performance	
Per processor	1.5 Gflop/s
Per Cabinet	144 Gflop/s
Max. Configuration	12.0 Tflop/s
Memory	
Per Cabinet	$\leq$ 768 GB
Max. Configuration	64 TB
No. of processors	
Per Cabinet	96
Max. Configuration	8024
Communication bandwidth	
Point-to-point	$\leq$7.6 GB/s
Bisectional/cabinet	667 GB/s

Remarks:
The macro architecture of the Cray XMT is very much alike those of the Cray XT3 and the XT4. However, the processors used are completely different: They are made for massive multithreading and resemble the processors of the late Cray MTA-2 (see 4 and [51]).
Let us look at the architectural features: Although the memory in the XMT is physically distributed, the system is emphatically presented as a shared-memory machine (with non-uniform access time). The latency incurred in memory references is hidden by *multi-threading*, i.e., usually many concurrent program threads (instruction streams) may be active at any time. Therefore, when for instance a load instruction cannot be satisfied because of memory latency the thread requesting this operation is stalled and another thread of which an operation can be done is switched into execution. This switching between program threads only takes 1 cycle. As there may be up to 128 instruction streams per processor and 8 memory references can be issued without waiting for preceding ones, a latency of 1024 cycles can be tolerated. References that are stalled are retried from a retry pool. A construction that worked out similarly was to be found in the late Stern Computing Systems SSP machines (see in section 4).
An XMT processor has 3 functional units that together can deliver 3 flops per clock cycle for a theoretical peak performance of 1.5 Gflop/s. There is only one level of caches, data and instruction, because due to the nature of the applications at which the machine is directed more cache levels would be virtually useless. The high degree of latency hiding through massive multi-threading is the mechanism of choice here to combat memory latency.
Unlike the earlier MTA-2 there is no Fortran compiler anymore for the XMT. Furthermore, the new 3-D torus network, identical to that of the Cray XT4 and the faster clock cycle of 500 MHz makes the machine highly interesting for applications with very unstructured but massively parallel work as for instance in sorting, data mining, combinatorial optimisation and other complex pattern matching applications. Also algorithms like sparse matrix-vector multiplications will perform well.

Measured Performances:
As yet no independent performance results for this new machine are available to prove the value of this interesting architecture.

3.1.7 The Fujitsu/Siemens M9000 series.

Machine type: RISC-based shared-memory multi-processor.
Models: M9000/32, M9000/64.
Operating system: Solaris (Sun's Unix variant).
Connection structure: Crossbar.
Compilers: Fortran 90, OpenMP, C, C++.
Vendors information Web page:
`www.fujitsu-siemens.com/products/unix_servers/sparc_enterprise/sparcent_enterprise.html`
Year of introduction: 2008.

System parameters:

Model	M9000/32	M9000/64
Clock cycle	2.52 GHz	2.52 GHz
Theor. peak performance		
Per core (64-bits)	10.1 Gflop/s	10.1 Gflop/s
Maximal	1.29 Tflop/s	2.58 Tflop/s
Memory/node	$\leq$ 128 GB	$\leq$ 128 GB
Memory/maximal	$\leq$ 1 TB	$\leq$ 2 TB
No. of processor cores	8–64	8–128
Communication bandwidth		
Point-to-point	$\geq$8 GB/s	$\geq$8 GB/s
Aggregate	367.5 GB/s	737 GB/s

Remarks:
We only discuss here the M9000/32 and M9000/64 as the smaller models like the M8000s have the same structure but less processors. We also mention here that the same models are available with a somewhat slower dual-core processors at 2.28 and 2.4 GHz. The M9000 systems now represent the high-end servers of Fujitsu-Siemensand Sun and as such replace both the Fujitsu-Siemens PRIMEPOWER series as well as Sun's E25K server (see 3.1.20).
The quad-core SPARC64 VII processors (see section 2.8.8) have a theoretical peak speed of 10.08 Gflop/s/core and are packaged in four-processor CPU Memory Units (CMUs). Apart from the four processors a CMU also houses a part of the total memory, up to 128 GB/CMU. All components of a CMU, CPUs and memory controllers are directly connected to each other in a crossbar fashion. The CMUs, each residing on one board, are in turn again connected by a crossbar, connecting 8 or 16 of them in the M9000/32 and M9000/64, respectively. The M9000-64 is a 2-cabinet version of the M9000/32.
The system interconnect is called the Jupiter bus by Fujitsu. It connects via the crossbar both to the CPU and the I/O boards. From the information provided by Fujitsu the point-to-point bandwidth cannot be exactly derived but should be more than 8 GB/s. The aggregate bandwidth is however stated for both configurations. Because of the structure of an CMU the memory access will be uniform between CPUs but it is not clear this also the case for memory access from other boards. Fujitsu does not state a NUMA factor for the systems although it is highly probable memory access is non-uniform within the entire system. From other sources it can be gathered that with respect to the earlier PRIMEPOWER series the crossbar is doubled and when one of them fails communication proceeds at half of the total bandwidth. The aggregate bandwidth is impressive: 737 GB/s for the M9000/64.
Fujitsu-Siemens positions the Mx000 servers for the commercial market and seems not interested to market it for HPC-related work although the specifications look quite good. On the other hand, the systems are fitted with extreme RAS features, like memory mirroring and a double crossbar, that will be much appreciated in commercial environments but which makes the systems relatively costly.

Measured Performances:
No performance results in the technical/scientific area are known to us to date. This is not only due to the newness of the system but also to Fujitsu's lack of interest in the scientific HPC realm for this platform.

3.1.8 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.computers.us.fujitsu.com/www/products_primequest.shtml?products/servers/primequest/index`
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	$\leq$ 512 GB	$\leq$ 2 TB
No. of processors	4–16	4–32
Communication bandwidth		
Point-to-point	4.3 GB/s	4.3 Gb/s
Aggregate	68.2 GB/s	136.4 GB/s

Remarks:
We only discuss here the PRIMEQUEST 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 offered these days. The older 400 series that was on the market for less than a year is now replaced by the 500 series that contains the dual-core Montvale 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 from the 540 or 580 nodes are not offered as was the case for the late PRIMEPOWER systems (see 4) probably because the PRIMEQUESTs use a standard Linux distribution for an operating system instead of the Solaris variant that Fujitsu employs in the SPARC64-based M9000 systems. So, when one (or Fujitsu) is prepared to build a large configuration based on the PRIMEQUEST one would have to used a third party network like Infiniband, Myrinet, or Quadrics.
The PRIMEQUEST is not a ccNUMA 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 fulfilled 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 Montvale-based systems. Only in parallel OpenMP and MPI programs the differences should show.
Fujitsu tries to stand out with respect to other vendors in offering fail-safe 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:
An ensemble of 10 PRIMEQUEST systems has, according to the TOP500 list of November 2006 attained a speed of 3119 Gflop/s for a linear system of size $N = 499200$. This amounts to an efficiency of 76%.

3.1.9 The Hitachi BladeSymphony.

Machine type: RISC-based distributed-memory multi-processor.
Models: BladeSymphony.
Operating system: Linux (RedHat EL4), Windows Server 2008.
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 core (64-bits)	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 3.1.13). 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 3.1.8 and 3.1.16) 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, indicative of the fact that the system is not primarily regarded as an HPC platform by Hitachi.

3.1.10 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: 2005.

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	65.5 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). We discuss here the latest model, the SR11000 K1. There is a J1 model which is identical to 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 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 did for its own 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 XT4 (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 first 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. A model K1 result was reported in the TOP500 list of June 2007: 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%.

3.1.11 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 (SuSE SLES 10), Microsoft Windows Server 2008, OpenVMS.
Compilers: Fortran 77, Fortran 90, HPF, C, C++.
Vendors information Web page: `h20341.www2.hp.com/integrity/cache/342254-0-0-0-121.html`
Year of introduction: 2004.

System parameters:

Model	Integrity Superdome
Clock cycle	1.6 GHz
Theor. peak performance	
Per core (64-bits)	6.4 Gflop/s
Maximal	819.2 Gflop/s
Memory	$\leq$2 TB
No. of processors	6–64
Communication bandwidth	
Cell controller–CPU	8.5 GB/s
Aggregate (cell–backplane)	34.6 GB/s

Remarks:
The Integrity Superdome is HP's investment in the future for high-end servers. Within a time span of a few years it should replace the PA-RISC-based HP 9000 Superdome completely. 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 through the crossbar backplane. Every cell connects to the backplane at a speed of 34.6 GB/s while the bandwidth from a cell to a processor is 8.5 GB/s. In fact, the inter-cell crossbar is implemented as 3 parallel crossbars which makes it highly resistent against failure.

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 17.1 GB/s, to the backplane crossbar at 34.6 GB/s, and to two ports that each serve two processors at 8.5 GB/s/port. As each processor houses two CPU cores the available bandwidth per CPU core is 2.125 GB/s. Like the SGI Altix systems (see section 3.1.18), 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. Unfortunately, the largest configuration with 64 processors is only offered with HP-UX, HP's proprietary OS. This does not help spreading in the HPC community. For the other operating systems the maximum size has 32 processors/64 cores.

Measured Performances:
In the TOP500 list of June 2007 a Myrinet-connected cluster of Superdomes was reported to attain 4972 out of 5990 Gflop/s on 936 cores (468 processors) with HP-UX as the operating System. The efficiency was 82%.

3.1.12 The IBM BlueGene/L&P.

Machine type: RISC-based distributed-memory multi-processor.
Models: IBM BlueGene/L&P.
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/bluegene/.`
Year of introduction: 2004 for BlueGene/L, 2007 for BlueGene/P.

System parameters:

Model	BlueGene/L	BlueGene/P
Clock cycle	700 MHz	850 MHz
Theor. peak performance		
Per Proc. (64-bits)	2.8 Gflop/s	3.4 Gflop/s
Maximal	367/183.5 Tflop/s	1.5/3 Pflop/s
Memory/card	512 MB	2 GB
Memory/maximal	$\leq$16 TB	$\leq$442 TB
No. of processors	$\leq$2×65,536	$\leq$4×221,184
Communication bandwidth		
Point-to-point (3-D torus)	175 MB/s	$\approx$350 MB/s
Point-to-point (Tree network)	350 MB/s	$\approx$700 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.

BlueGene/P
In the second half of 2007 the second generation BlueGene system, the BlueGene/P was realised and several systems have been installed. The macro-architecture of the BlueGene/P is very similar to that of the L model, except that about everything in the system is faster and bigger. The chip is a variant of the PowerPC 450 family and runs at 850 MHz. As, like in the BlueGene/L processor 4 floating-point instructions can be performed per cycle, the theoretical peak performance is 3.4 Gflop/s. Four processor cores reside on a chip

(as opposed to 2 in the L model). The L3 cache grew from 4 to 8 MB and the memory per chip increased four-fold to 2 GB. In addition, the bandwidth in B/cycle has doubled and became 13.6 GB/s. Unlike the dual-core BlueGene/L chip the quad-core model P chip can work in true SMP mode, making it amenable to the use of OpenMP.
One board in the system carries 32 quad-core chips while again 32 boards can be fitted in one rack with 4,096 cores. A rack therefore has a theoretical peak performance of 13.9 Tflop/s. The IBM Press release sets the maximum number of cores in a system to 884,736 in 216 racks and a theoretical peak performance of 3 Pflop/s. The higher bandwidth of the main communication networks (torus and tree) also goes up by a factor of about 2 while the latency is halved.
Like the BlueGene/L the P model is very energy-efficient: a 1024-processor (4096-core) rack only draws 40 KW.

In both the BlueGene/L and /P the compute nodes run a reduced-kernel type of Linux to reduce the OS-jitter that normally occurs when very many nodes are involved in computation. Interface nodes for interaction with the users and providing I/O services run a full version of the operating system.

Measured Performances:
In [54] a speed of 478.2 Tflop/s on the HPC Linpack benchmark for a BlueGene/L is reported, solving a linear system of size $N = 2{,}456{,}063$, on 212,992 processor cores. processors amounting to an efficiency of 80.1%.
In the same report a speed of 180 Tflop/s out of a maximum of 222.82 Tflop/s for a 65,536-core BlueGene/P was published, again with an efficiency of 80.1% but on a smaller linear system of size $N = 1{,}766{,}399$.

3.1.13 The IBM eServer p575.

Machine type: RISC-based distributed-memory multi-processor.
Models: IBM eServer p575.
Operating system: AIX (IBM's Unix variant), Linux (SuSE SLES 10).
Connection structure: Variable (see remarks).
Compilers: XL Fortran (Fortran 90), (HPF), XL C, C++.
Vendors information Web page:
`www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=AN&subtype=CA&htmlfid=897/ENUS107-675&appname=USN`.
Year of introduction: 2008 (32-core POWER6 SMP).

System parameters:

Model	eServer p575
Clock cycle	4.7 GHz
Performance	
Per Proc. (2 cores)	37.6 Gflop/s
Per node (32 cores)	601.6 Gflop/s
Per 14-node frame	8.42 Tflop/s
Maximal	$\geq$ 100 Tflop/s
Memory	
Memory/node	256 GB
Memory maximal	—
Communication bandwidth	
Node-to-node (see remarks)	—

Remarks:
There is a multitude of high end servers in the eServer p-series. However, IBM singles out the POWER6 based p575 model specifically for HPC. The eServer p575 is the successor of the earlier POWER5+ based systems. 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. Four dual-core POWER6 processors are housed in a Multi-

Chip Module (MCM) while four of these constitute a p575 node. So, 32 cores make up a node. The 4 MCMs are all directly connected to each other at a bandwidth of 80 GB/s. The inter-MCM links are used to reach the memory modules that are not local to a core but within a node. Therefore, all memory in a node is shared by the processor cores, although the memory access is no longer uniform as in earlier p575 models. As yet no NUMA factor is published but, given the node structure, it should be moderate. Obviously, within a node shared-memory parallel programming as with OpenMP can be employed.
In contrast to its earlier p575 clusters, IBM does not provide its proprietary Federation switch anymore for inter-node communication. Instead, one can choose to configure a network from any vendor. In practice this will turn out to be InfiniBand in most cases, but also switched Gigabit Ethernet, Myrinet or a Quadrics network is enterily possible. For this reason it is not possible to give inter-node bandwidth values as this is to be chosen by the user.
At this moment nowhere is to be found what the maximum configuration of a POWER6-based p575 configuration would be. The online information at present is not definitive and consistently speaks of *planned* characteristics of the POWER6-based systems, although several machines already have been installed and are in operation. At ECMWF, UK a 156 Tflop/s system is installed, at NCAR, USA a 71 Tflop/s, and a 60 Tflop/s at SARA in The Netherlands. Because of this lack of information, we cannot give details about maximum performance, memory, etc.
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. Because of the very dense packaging of the units that house the POWER6 processors are water cooled.
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: the professional version of SuSE Linux is available for the p575 series.

Measured Performances:
In [54] a performance of 80.3 Tflop/s for a 8,320 core system at ECMWF, Reading, UK, is reported for solving a dense linear system of unspecified order with an efficiency of 51%.

3.1.14 The IBM System Cluster 1350.

Machine type: RISC-based distributed-memory multi-processor.
Models: IBM System Cluster 1350.
Operating system: Linux (RedHat EL4/5, SuSE SLES 10), Windows Server 2008.
Connection structure: Variable (see remarks).
Compilers: XL Fortran (Fortran 90), (HPF), XL C, C++.
Vendors information Web page:
`www.ibm.com/systems/clusters/hardware/1350.html`.
Year of introduction: 2005–2008, dependent on blade/rack type.

System parameters:

Model	IBM System Cluster 1350
No. of processors	2–1024

Remarks:
The IBM System Cluster 1350 is one of the systems reffered to in the introduction of this section. The choice of components is so wide that not a single description of the system can be given. The only constant factor that can be given about the system is the amount of processors. The system can house a bewildering number of different rack units or blades, including models with AMD Opterons, PowerPC 970MPs, POWER6s, or Cell BE processors. The choice of the interconnect network can also be more or less arbitrary: Gigabit Ethernet, InfiniBand, Myrinet, etc. A very large system build on this technology is the Mare Nostrum

machine at the Barcelona Supercomputing Centre which has a cluster of ten 1350 Cluster Systems based on the JS21 blade. The PowerPC 970MP variant (see section 2.8.3). With 10,240 processors the Theoretical Peak Performance is just over 94 Tflop/s.
Cell BE boards (QS21 and QS22) can be accommodated in the System Cluster 1350. So, potentially, one can build a hybrid system in this way with the Cell processors as computational accelerators.

3.1.15 The Liquid Computing LiquidIQ system.

Machine type: Distributed-memory multi-processor system.
Models: LiquidIQ.
Operating system: Linux (RedHat EL4/5, SuSE SLES 10), Windows Server 2008.
Connection structure: Crossbar.
Compilers: Fortran 95, ANSI C, C++, Berkeley UPC.
Vendors information Web page: `www.liquidcomputing.com/product/product_product.php`
Year of introduction: 2006.

System parameters:

Model	LiquidIQ
Clock cycle	2.5 GHz
Theor. peak performance	
Per core (64-bits)	5.0 Gflop/s
Maximal	10.0/20.0 Tflop/s
Main Memory (per chassis)	$\leq$1.28 TB
No. of processors	$\leq$960
Communication bandwidth	
Point-to-point	16 GB/s
Aggregate (per chassis)	16 GB/s

Remarks:
Liquid Computing announced the LiquidIQ system in 2006. The systems are very densely built with up to 20 4-processor compute modules in a chassis. A maximum of 12 chassis can be put together to form a system with very high bandwidth (16 GB/s) and very low communication latency. The basic processor in the compute modules is a dual-core or quad-core AMD Opteron from the 2200/8200 or 2300/8300 series, respespctively. The 8 or 16 cores within a compute module can access the maximally 64 GB in the module in SMP mode. So, the system is capable of supporting hybrid OpenMP/MPI programming style.
The chassis that house the modules are cable-less: the modules connect to a backplane that acts as a crossbar between the modules. According to the documentation the guaranteed throughput bandwidth of the backplane is 16 GB/s (2 GB/s per communication plane of which there are 8). The inter-chassis bisection bandwidth ratio is 1.1. So, the inter-chassis bandwidth is 17.6 GB/s in order to combat the delays between chassis. The latency as given in the documentation is low: 2.5 μs irrespective of the relative positions of the communicating processors. This is brought about by a combination of the chassis backplanes and Liquid's multi-chassis switches. Liquid provides its own MPI implementation which is as yet at the level of MPICH 1.2. OpenMP is available via the supported compiler set and also Berkeley's UPC is supported.
Liquid Computing prides itself in making the systems rather energy-efficient: about 14 KW/chassis peak. So, a maximally configured system (not counting the I/O configuration) has a maximal power consumption of 168 KW.

Measured Performances:
There are as yet no independent performance results available for the system. Liquid Computing itself has published some partial results from the HPCC Benchmark [24] in press releases but there is no official entry for the system on the HPCC web site yet.

3.1.16 The NEC Express5800/1000 series.

Machine type: Shared-memory ccNUMA system.

Models: Express5800 1160Xf, 1320Xf.
Operating system: Linux, Windows Server 2008.
Connection structure: Crossbar.
Compilers: Fortran 95, HPF, ANSI C, C++.
Vendors information Web page: `www.nec.co.jp/express/products/enterprise/index.html`.
Year of introduction: 2006.

System parameters:

Model	1160Xf	1320Xf
Clock cycle	1.6 GHz	1.6 GHz
Theor. peak performance		
Per core (64-bits)	6.4 Gflop/s	6.4 Gflop/s
Maximal	204.8 Gflop/s	409.6 Gflop/s
Main Memory (per frame)	≤256 GB	≤512 GB
No. of processors	16	32
Communication bandwidth		
Point-to-point	—	—
Cell-to-cell (see remarks)	6.4 GB/s	6.4 GB/s
Aggregate	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 e.g., also 3.1.1, 3.1.18, 3.1.11, and 3.1.9) that is presently on the market. The largest configuration presently offered is the 1320Xf with 32 1.6 GHz Montecito processors 24 MB L3 cache/processor. 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-9 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 3.1.8, 3.1.9) NEC very much emphasizes the RAS features of the system, targeting mission-critical operating environments and, in fact, have implemented memory-mirroring in the 1160Xf and 1320Xf models.

Measured Performances:
As yet no performance results have been published for this system.

3.1.17 The NEC SX-9 series.

Machine type: Shared-memory multi-vectorprocessor.
Models: SX-9B SX-9A, SX-9xM$y, y = 2, \ldots, 512$.
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/hardware/index.html`.
Year of introduction: 2007.

System parameters:

Model	SX-9B	SX-9A	SX-9xMy
Clock cycle	3.2 GHz	3.2 GHz	3.2 GHz
Theor. peak performance			
Per Proc. (64-bits)	102.4 Gflop/s	102.4 Gflop/s	102.4 Gflop/s
Maximal			
Single frame:	819.2 Gflop/s	1.6 Tflop/s	—
Multi frame:	—	—	838.9 Tflop/s
Main Memory, DDR2-SDRAM	256–512 GB	512–1024 GB	≤512 TB
Main Memory, FCRAM	128–256 GB	256–512 GB	≤256 TB
No. of processors	4–8	8–16	32–8192

Remarks:
The NEC SX-9 is a technology shrunken version of its predecessor the SX-8 (see 4). As a result the clock cycle has increased from 2.0 to 3.2 GHz, the density of processors/frame has doubled and the power consumption almost halved. The structure of the CPUs, however, has stayed the same. The SX-9 series is basically offered in three models as displayed in the table above. 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) and two of each are present, the peak performance of a pipe set at 3.2 GHz is 12.8 Gflop/s. Because of the 8-way replication a single CPU can deliver a peak performance of 102.4 Gflop/s. The official NEC documentation quotes higher peak performances because the peak performance of the scalar processor (rated at 6.4 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 3.2 GHz has a theoretical peak of 6.4 Gflop/s. The peak bandwidth per CPU is 160 B/cycle. This is sufficient to ship 20 8-byte operands back or forth, enough to feed 5 operands every 2 cycles 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 4–8 processors but is in virtually all other respects equal to the larger SX-8A that can house 8–16 processors, 16 being the maximum number of processors fitting in a frame. There is one difference connected to the maximal amount 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 system parameters table, the entries for FCRAM are two times lower than for SDRAM. 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-9A models fit up to 16 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: 512 GB/s/port. The fully configured frame can therefore attain a peak speed of 1.6 Tflop/s.
In addition, there are multi-frame models (SX-9xMy) where $x = 32, \ldots, 8192$ 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. To couple the SX-9 frames NEC provides a full crossbar, the so-called IXS crossbar to connect the various frames together at a speed of 128 GB/s for point-to-point unidirectional out-of-frame communication. When connected by the IXS crossbar, 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:
Presently no independent performance results for the SX-9 are known.

3.1.18 The SGI Altix 4000 series.

Machine type: RISC-based ccNUMA system.
Models: Altix 4700.
Operating system: Linux (SuSE SLES9/10, RedHat EL4/5) + 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 core (64-bits)	6.64 Gflop/s
Maximal	6.8 Tflop/s
Memory	≤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 latest Altix Itanium-based version is the 4000 series succeeding the Altix 3700. The difference is mainly in the support of the type of Intel Itanium processors and the communication network. The Altix 4700 supports the dual-core Montvale 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 accommodate 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 (1024 cores). So OpenMP programs with up to 1024 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.
The SGI RASC blade (see 2.9.3.4) with FPGAs can be incorporated in an Altix 4700 system, as a first step to SGI's version of a hybrid system (see section 5 for SGI's plans in that respect).

Measured performances:
In [54] a variety of Altix 4700 entries are present, the highest ranking of which is that of the system at the Leibniz Rechenzentrum in Munich. It reports a speed of 56.5 Tflop/s solving a linear system of size N =1,583,232 with an efficiency of 91%.

3.1.19 The SiCortex SC series.

Machine type: RISC-based distributed-memory multi-processor.
Models: SC648, SC5832.
Operating system: Linux (Gentoo + extensions).
Connection structure: Kautz graph.
Compilers: Fortran 95, ANSI C, C++.
Vendors information Web page:`www.sicortex.com/products`
Year of introduction: 2006.

System parameters:

Model	SC648	SC5832
Clock cycle	500 MHz	500 MHz
Theor. peak performance		
Per Proc. (64-bits)	1.0 Gflop/s	1.0 Gflop/s
Maximal	648 Gflop/s	5.8 Tflop/s
Memory	$\leq$864 GB	$\leq$7.8 TB
No. of processors	648	5832
Communication bandwidth		
Point-to-point	2 GB/s	2 GB/s
Aggregate peak	648 GB/s	5.8 TB/s

Remarks:
SiCortex is a new player on the HPC field, the systems being announced end 2006 and with first available systems in June 2007. The systems are unusual in a number of aspects: First, like in the IBM BlueGene systems SiCortex has chosen for a processor with a low clock frequency in order to have low power consumption. According to the documentation this goal has been achieved as the SC648 only uses 2 kW and the SC5832 18 kW. As such they are by far the most energy efficient HPC systems marketed today. The downside is of course the that theoretical peak performance of the processor is also low by today's standards: 1 Gflop/s. The processors are from the MIPS family that disappeared from the HPC scene with the SGI's Origin systems (see section 4). Now a MIPS64 variant is re-introduced in the SC models (see section 2.8.7 for some details). The low clock frequency is not all bad: the mismatch between the memory speed and the processor speed is much smaller than in most other systems which may lead to a higher relative efficiency for codes that suffer a fair proportion of cache misses. Furthermore, the MIPS compilers used to be of very high quality, also potentially leading to high relative efficiency.
The SC systems are built out of 6-processor nodes where all devices of a node are integrated on one chip. This makes the packaging of the systems very compact. A node further contains a common coherent L2 cache for the 6 processors, a DMA engine, two memory controllers, a network fabric switch, and a PCI Express controller. The communication network is logarithmic but unlike other networks employed. It is a so-called Kautz graph which combines a small Ω with a high connectivity, [4] (see also Figure 2.26 in section 2.10). The point-to-point bandwidth is stated to be high: 2 GB/s while the latency is extremely low: 1 μs. Because of the constraints of the network we end up with the rather unusual number of processors. In the SC648 108 nodes are connected by a Kautz network with $\Omega = 4$ while in the SC5832 972 nodes are connected with a diameter $\Omega = 6$ network.
On the MIPS processors the PathScale compilers for Fortran and C/C++ are used while a proprietary MPI library is provided. The operating system is based on the Linux Gentoo distribution with some extensions by SiCortex. The file system offered is Lustre as is done by a growing number of vendors (Bull, Cray, Liquid Computing, and many clusters to name some).

Measured performances:
At this moment no independent performance results are known yet.

3.1.20 The Sun M9000.

Machine type: RISC-based shared-memory multi-processor.
Models: M9000-32, M9000-64.
Operating system: Solaris (Sun's Unix variant).
Connection structure: Crossbar.
Compilers: Fortran 90, OpenMP, C, C++.
Vendors information Web page: `www.sun.com/servers/highend/m9000/`
Year of introduction: 2007.

The systems are identical to those of Fujitsu-Siemens. For a description see section 3.1.7.

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 (4.1).

Machine: Cray X1E.
Year of introduction: 2004.
Year of exit: 2007.
Type: Shared-memory vector-parallel, max. 8192 MSP processors.
Theoretical Peak performance: 147.2 Gflop/s.
Reason for disappearance: replaced by newer Cray X2 (3.1.5.1).

Machine: Cray XD1.
Year of introduction: 2004.
Year of exit: 2006.
Type: Distributed-memory multi-processor, max. 144 processors.
Theoretical Peak performance: 663 Gflop/s.
Reason for disappearance: relevant components will re-appear in Cray's heterogeneous systems in the near future.

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.(3.1.11).

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.

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 PRIMEQUEST series. (3.1.8).

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 (3.1.10).

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. (3.1.11).

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. (3.1.13).

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 (see below).

Machine: NEC SX-8.
Year of introduction: 2004.
Year of exit: 2007.
Type: Distributed-memory cluster of SM-MIMD vector processors, max. 4096 processors.
Theoretical Peak performance: 90.1 Tflop/s.
Reason for disappearance: replaced by newer SX-9 series (3.1.17).

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 Itanium-based SGI Altix 4700 (3.1.18).

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 (3.1.20).

Machine: SUN Fire E25K.
Year of introduction: 2004.
Year of exit: 2007.
Type: Shared-memory multi-processor, max. 72 processors.
Theoretical Peak performance: 432 Gflop/s.
Reason for disappearance: replaced by the M9000 models (see 3.1.7 and 3.1.20).

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, it is good to look ahead a little and try to see what may be in store for us in the near future.
Below we discuss systems that may lead to commercial systems or components (i.e., mainly processors) 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 was, at the time, of significance was the introduction of Intel's IA-64 Itanium processor family. Six vendors are offering Itanium 2-based systems at the moment and it is known that HP has ended the marketing of its Alpha and PA-RISC based systems in favour of the Itanium processor family. Likewise SGI stopped the further development of MIPS processor based machines. The only vendor going against this trend is SiCortex that re-introduced a MIPS processor based machine. This means that the processor base for HPC systems has become rather narrow. However, the shock that was caused in the USA by the advent of the Japanese Earth Simulator system has helped in refueling the funding of alternative processor and computer architecture research of which we see the consequences in the last few years.
In section 2.9 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, etc. Within the near future a HPC cannot afford *not* to include somehow such accelerators into their architectures. One also cannot expect general processor and HPC vendors to ignore this trend. In some way they will either integrate the emerging accelerator capability into their system (as is in the road maps of, e.g., Cray and SGI, see below), try to incorporate accelerating devices on the chips themselves (as seems the way Intel in going), or provide ways to thightly integrate accelerator hardware with a CPU and memory via a fast direct connection. This we already see with AMD processors and will shortly be the case also with Intel processors. We briefly review the status of these developments below.

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 primarily based on the Cascade project. This is the project that has started with help of DARPA's High Productivity Computer Systems initiative (HPCS) that has as one of its goals that 10 Pflop/s systems (sustained) should be available by 2010. This should not only entail the necessary hardware but also a (possibly new) language to productively program such systems. Cascade was Cray's answer to this initiative. Together with IBM Cray has continuing support from the HPCS program (HP, SGI, and SUN, respectively have fallen out).
Cray seems reasonably on track with its Cascade project: The $XT5_h$ system (see 3.1.5) is already capable of housing X2, XT5, and XR1 boards within one infrastructure and it may well be that XMT processors (see 3.1.6) also will be included in following generations. At the moment, however, the different processor types cannot yet work seamlessly together in the sense that one can freely mix code in one program that will be dispatched to a set of the most appropriate processor type. This is the ultimate goal to be realised in a sequence of future systems. The follow-on systems bear imaginative names like "Baker" (about end 2008, begin 2009), "Granite" and "Marble", ultimately leading to a system that should be able to deliver 10 Pflop/s sustained by 2010.
Until recently Cray was to some extent dependent on AMD with respect to the scalar processors. In fact, Cray was hurt somewhat by the delay of AMD's quad-core Barcelona processor but this has not led to

a major slip in the scheduled plans and Cray recently has entered into discussions with Intel which may well lead to a change of processor in the scalar-type nodes or at least to a diversification. Cray's interest in the Intel processor has presumably been fuelled by the adoption of the QuickPath interface in the next generation of Intel's processors which would allow them to be integrated into the Cray systems in a way that is very similar to that which is used with AMD's HyperTransport interface.

5.2 IBM

IBM has been working for some years on its BlueGene systems. Many of these first models, the BlueGene/L, have been installed in the last few years (see 3.1.12). The BlueGene/L follow-up the BlueGene/P has been available for about a year now first and several /P systems have been installed in Europe as well as in the USA. Theoretically the BlueGene/P will attain a peak speed of 3 Pflop/s and the BlueGene/Q, the next generation will have a peak speed of around 10 Pflop/s. The BlueGene systems are hardly meant for the average HPC user but rather for a few special application fields that are able to benefit from the massive parallelism that is required to apply such systems successfully.
Of course the development of the POWERx processors also will make its mark: the POWER6 processor has the usual technology-related advantages over its predecessor, and the first POWER6-based systems are in operation since a few months. Furthermore, it is a subject of research how to couple 8 cores 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 late Cray X1E. This road will take some years to go and may appear in the POWER7 processor and extend to the next generation(s) of the POWERx.
The Cell processor, and in particular the PowerXCell 8i, will be further integrated in the systems provided by IBM. This has already been done in the Roadrunner systems with the tri-blade configuration and as evident from section 3.1.14, the current cluster systems can already include the accelerator blades along with other processor blades. Like, with Cray, it is not yet possible to have one code targetting a random mix of processor blades but this could be a next step.
Like Cray, IBM is one of the two vendors that are still supported by the HPCS program from DARPA. Although this support is less important for IBM than for Cray, parts of the research that now is done regarding porting applications to BlueGene-type systems, the viability of the ViVA concept, and the integration of Cell processors is certainly helped by this support. A system based on the POWER7 should be IBM's answer to DARPA's request for a machine able of a 10 Pflop/s sustained performance in 2010.

5.3 Intel-based systems

All systems that are based on the Itanium line of processors, i.e., Bull, Fujitsu, Hitachi, HP, NEC, and SGI, are critically dependent on the ability of Intel to timely deliver the Tukwila processor, which is slated for late 2008. Not only the number of cores in this processor will double to four while the modest clock frequency will go up in the 2 GHz realm, most importantly, the processor finally will get rid of the front-side bus that is a serious bottleneck in the access of data from the memory. The Tukwila processor will use the QuickPath Interface (QPI), formerly known as the Common System Interface (CSI) which presumably will provide a bandwidth of 25.6 GB/s. Of course this is necessary because of the increased number of cores/chip. In addition, the QPI specification will be open like AMD has done with its HyperTransport bus in the Torrenza initiative. This means that both low-latency networks and attached computational accelerators can be connected directly at high speed. This in turn will allow vendors to diversify their products, possibly to optimise them for specific application areas similar to Cray's future plans (see 5.1).
Furthermore the QPI will also be available for the Xeon line of processors the next one being the Nehalem processor, which would even allow for mixing them with Itanium-based systems components. In fact, SGI plans to do so in its Altix systems and Hitachi already provides it in their BladeSymphony systems (see 3.1.9). Other vendors may follow this trend as it is another way of system diversification.
Intel is not insensitive with respect to the fast increase of the use of computational accelerators, GPUs in particular. An answer might be the many-core processors that are part of Intels future plans. The Larabee processor that is expected to be available in 2009 unites general CPU cores and about 16 simple GPU-like

512-bit wide SIMD cores on a chip with a feature size of 45 nm at a clock frequency of about 2–2.5 GHz. The SIMD core would be connected by a 1024 bit wide ring. In all the structure is somewhat reminiscent of the Cell processor. A rough estimate of the performance would be about 1 Tflop/s (for the right type of computations), in the same league as the high-end GPUs now sold by ATI an NVIDIA. Intel is not expected to stop here and will develop other many-core processors in the near future.

5.4 SGI

SGI has 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 (see 2.9.3.4. A next step is mixing Itanium-based components and Xeon-based components in the same system. Once the QuickPath Interface is available (see 5.3 this should be doable without excessive costs because the QPI chipset will support both the Itanium and Xeon processor variants. The idea is to further diversify the future systems, ultimately into a system with the codename "Ultraviolet". Development of such systems is quite costly and unlike Cray and IBM, SGI does not have support from DARPA's HPCS program, so it remains to be seen whether such plans will pass the stage of intentions in regard of the less luxury financial position of SGI.

5.5 SUN

Like Cray and IBM, SUN had been awarded a grant from DARPA to develop so-called high-productivity systems in DARPA's HPCS program. In 2007 SUN has fallen out of this program and so it determined to concentrate even more on developing heavily multi-threaded processors. The second generation of the Niagara chip, the T2 is in production and is on the market for some time now. It supports 64 threads with 8 processor cores and has a floating-point unit attached to each core. This is a large improvement in comparison of the former T1 that had only one floating-point unit/chip. Still, the T2 is not geared for the HPC area. This will be reserved for Sun's Rock processor which is to come out somewhere in 2008. It has 16 processor cores in a 4×4 grid on the chip, each core supporting 2 threads. Each 4-core part shares 32 KB of L1 data cache and L1 instruction cache together with 512 KB of integrated L2 cache. The L1 and L2 caches are connected by a 4×4 crossbar. The Rock processor *will* be geared to HPC work. The clock frequency is believed to be around 2.3 GHz. SUN plans to produce two server variants: "Peble", a one-socket version and "Boulder" that may have 2, 4, or 8 sockets.
There are several other novel features in the processor. For one, the dual threads in a core are used for "scouting", i.e., one thread runs ahead to look for possiblities to prefetch operands that will be needed by the other thread. When the active computation thread stalls for whatever reason, the roles are reversed. Furthermore, SUN will be probably implement Transactional Memory in the Rock processor. This would enable exclusive memory access for the threads within a program without the need of explicit locks in the threads. This would greatly simplify many multi-threaded applications and give much less overhead.
For the mainstream market SUN continues to rely on the UltraSPARC64 developments from Fujitsu-Siemens as present in Fujitsu-Siemens and SUN M9000 systems.

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

API:API: API stands for Application Program(mer) Interface. Ususally it consists of a set of library functions that enable to access and/or control the functionality of certain non-standard devices, like an I/O device or a computational accelerator.

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 (L1) 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 (L2) cache is sometimes 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 quarter of a nanosecond.

Clock frequency: Reciproke of the clock cycle: the number of cycles per second expressed in Hertz (Hz). Typical clock frequencies nowadays are 1–4 GHz.

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

Co-Array Fortran: Co-Array Fortran (CAF) is a so-called Partitioned Global Address Space programming language (PGAS language, see below) that extends Fortran by allowing to specify a processor number for data items within distributed data structures. This allows for processing such data without explicit data transfer between the processors. CAF will be incorporated in Fortran 2003, the upcoming Fortran standard.

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 40 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 instance 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.

CSI: Common System Interface: See QPI.

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 CPU cores and (possibly common) caches. Due to the progression of the integration level more devices can be fitted on a chip. AMD, Fujitsu, IBM, and Intel make dual core chips and the first quad-core chips (with 4 CPU cores) are already available from AMD, Fujitsu, and Intel.

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 all other processors and without this modification contention would occur near the root.

Feature size: The typical distance between the various devices on a chip. By now processors with a feature size of 45 nanometer (10^{-9} m, nm) are on the market. Lowering the feature size is beneficial in that the speed of the devices on the chip will increase because of the smaller distances the electrons have to travel. The feature size cannot be shrunk much more though, because of the leaking of current that can go up to unacceptable levels. The lower bound is now believed to be around 7–8 nm for Silicon.

Flop: floating-point operation. Flop/s are used as a measure for the speed or performance of a computer. Because of the speed of present day computers, rather Megaflop/s (Mflop/s, 10^6 flop/s), Gigaflop/s (Gflop/s, 10^9 flop/s), Teraflop/s (Tflop/s, 10^{12} flop/s), and Petaflop/s (Pflop/s, 10^{15} flop/s) are used.

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.

Frontside Bus: Bus that connects the main memory with the CPU core(s) via a memory controller (only in scalar processors, not in vector processors). The frontside bus has increasingly become a bottleneck in the performance of a system because of the limited capacity of delivering data to the core(s).

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, also known as Host Channel Adaptor (specifically for InfiniBand). 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.

HPCS: Abbrevation of High Productivity Computer Systems: a program initiated by DARPA, the US Army Agency that provides large-scale financial support to future (sometimes futuristic) research that might benefit the US Army in some way. The HPCS program was set up to ensure that by 2010 computer systems will exist that are capable of a performance of 1 Pflop/s in real applications as opposed to the Theoretical Peak Performance which might be much higher. Initially Cray, HP, IBM, SGI, and SUN participated in the program. After repeated evaluations only Cray and IBM still get support from the HPCS program.

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.

HyperTransport: An AMD-developed bus that directly connects a processor to its memory without use of a Frontside Bus at high speed. It also can connect directly to other processors and because the specification is open, also other types of devices, like computational accelerators can be connected in this fashion to the memory. Intel will provide a similar type of bus, called CSI (see above) from 2008 on.

IDE Integrated Development Environment. A software environment that integrates several tools to write, debug, and optimise programs. The added value of an IDE lies (should lie) in the direct feedback from the tools. This should shorten the development time for optimised programs for the target architecture. The most well-kown IDE is probably IBM's Eclipse.

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.

LUT: Look-up table. A measure for the amount of memory cells on an FPGA.

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 highest bandwidth PCI-X, a common PCI bus version is $\approx$ 1 GB/s, while its successor, PCI Express, now normally is available with a 2–4 GB/s bandwidth. The newest version PCI Express, Generation 2 allows for a maximum of 8 GB/s for a 16×connection.

PGAS Languages: Partitioned Global Address Space languages. A family of languages that allow to specify how data items are distributed over the available processes. This gives the opportunity to process these data items in a global fashion without the need for explicit data transfer between

processors. It is believed that at least for a part of the HPC user community this makes parallel programming more accessible. The most known languages are presently Unified Parallel C (UPC) and Co-Array Fortran (CAF). Also Titanium, a Java-like language is employed. Apart from these languages that are already in use (be it not extensively) Chapel, developed by Cray and X10, developed by IBM, both under a contract with the US Department of Defense, have PGAS facilities (and more). However, the latter languages are still in the development phase and no complete compilers for these languages are available yet.

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.

Quad-core chip: A chip that contains four CPU cores and (possibly common) caches. Due to the progression of the integration level more devices can be fitted on a chip. AMD, Fujitsu, and Intel make quad-core chips soon to be followed by so-called many-core chips that will hold several tens to a hundred cores.

QPI: QuickPath Interface: Formerly known as Common System Interface (CSI). A bus structure in development by Intel and available by late 2008 that directly connects a (variety of) processor(s) of a system at high speed to its memory without the need for a Frontside Bus. It also can be used for connecting processors to each other. A similar type of interconnection, HyperTransport, is already provided for some years by AMD for its Opteron processors (see 2.8.1 for details).

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.

SDK: Software Development Kit. The term has become more common as many vendors that sell computational accelerator hardware also need to provide the software that is necessary to make the accelerator effectively usable for non-specialist HPC users.

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 some other 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.

U: A unit used in defining the height of a component in a standard 19-inch wide rack system. 1 U is 44.5 mm or 1.75 inch.

UPC: Unified Parallel C (UPC). A PGAS language (see above). UPC is an extension of C that offers the possibility to specify how data items are distributed over the available processors, thus enabling processing these data without explicit data transfers between the processors.

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 Intel's IA-64 chip architecture, there called EPIC (see above).

Acknowledgments

Again, 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 us 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.sandia.gov/ASC/`.

[4] J.-C. Bermond, N. Homonobo, C. Peyrat, *Large Fault-Tolerant Interconnection Networks*, Graphs and Combinatorics **5**, (1989), 107–123.

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

[6] The home page for Co-array Fortran can be found at: `http://www.co-array.org/`.

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

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

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

[10] B. Chapman, G. Jost, R. van der Pas, *Using OpenMP*, MIT Press, Boston, 2007.

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

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

[13] 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.

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

[15] 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.

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

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

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

[19] 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.

[20] 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`.

[21] 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)

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

[23] 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.

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

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

[26] 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/`

[27] 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.

[28] J.Kuehn, J. Larkin, N. Wichmann, *An Analysis of HPCC Results on the Cray XT4*, `info.nccs.gov/_media/resources/jaguar/docs/kuehn_paper.pdf?↩`
`id=resources:jaguar:docs&cache=cache` , 2007.

[29] Julie Langou, Julien Langou, P. Luszczek, J. Kurzuk, J.J. Dongarra, *Exploiting the Performance of 32-Bit Floating Point Arithmetic in Obtaining 64-Bit Accuracy*, Proceedings of SC06, Tampa, Nov. 2006.

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

[31] 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.

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

[33] 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.

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

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

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

[37] 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.

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

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

[40] S. Scott, D. Abts, J. Kim, W.J. Dally, *The Black Widow High-Radix Clos Network*, Proc. ISCA'06, Boston, 2006.

[41] *The Advantages of First-Generation Heterogeneous Computing on the Cray XT5h*, SciDAC Review, Summer 2008, 42–49.

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

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

[44] 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.

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

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

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

[48] 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/`

[49] 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/`.)

[50] 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/`.)

[51] A.J. van der Steen, *Overview of recent supercomputers*, June 2005, `www.euroben.nl`, directory `reports/`.)

[52] A.J. van der Steen, *Evaluation of the Intel Clovertown Quad Core Processor*, Technical Report HPCG-2007-02, Utrecht University, High Performance Computing Group, April 2007. (Also available through `www.euroben.nl`, directory `reports/`.)

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

[54] H.W. Meuer, E. Strohmaier, J.J. Dongarra, H.D. Simon, *Top500 Supercomputer Sites*, 29th Edition, June 2007, The report can be downloaded from: `www.top500.org/`.

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

[56] The home page of UPC can be found at: `http://upc.gwu.edu/`.