FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61-6402

Interner Bericht

Intel Paragon XP/S - Architecture and Software Environment

Rüdiger Esser, Renate Knecht

KFA-ZAM-IB-9305

April 1993
(Stand 26.04.93)

Erscheint in: Supercomputer ’93, Proceedings zum Seminar, Mannheim, 24.-26.6.1993

Intel Paragon XP/S - Architecture and Software Environment

Rüdiger Esser and Renate Knecht

Central Institute for Applied Mathematics
Research Centre Jülich (KFA)
P.O.Box 1913, D-5170 Jülich, Germany
e-mail: r.esser@kfa-juelich.de, r.knecht@kfa-juelich.de
April 26, 1993

Abstract. The paper describes the hardware and software components of the Intel Paragon XP/S system, a distributed memory scalable multicomputer. The Paragon processing nodes, which are based on the Intel i860 XP RISC processor, are connected by a two-dimensional mesh with high bandwidth. This new interconnection network and the new operating system are the main differences between the Paragon and its predecessor, the iPSC/860 with its hypercube topology. The paper first gives an overview of the Paragon system architecture, the node architecture, the interconnection network, I/O interfaces, and peripherals. The second part outlines the Paragon OSF/1 operating system and the program development environment including programming models, compilers, application libraries, and tools for parallelization, debugging, and performance analysis.

1 Introduction

The Paragon, which was first delivered in September 1992, is a product of Intel Corporation's Supercomputer Systems Division. As its predecessors, the prototypical Touchstone Delta system and the iPSC/860, the Paragon is a scalable distributed multicomputer. Its nodes are also based on Intel's i860 RISC processor and it also primarily supports message-passing as a programming model. The most significant differences between the Paragon and the iPSC/860 with its hypercube topology are the new fast rectangular interconnection network and the new OSF/1 based operating system.

This paper intends to give a tutorial survey of the Paragon hardware and software architecture. At the time of writing (April 1993), more than 25 Paragon systems have already been delivered to customers. The system, however, is still under development. This is especially true for the operating system, the first commercial version of which will be available in May, 1993. The first release of the complete operating system is announced for fall 1993. At this time most of the hardware features and tools for software development will also be available. As it is difficult to give a valid description in such a fast changing situation, we chose to describe the Paragon system as it will exist in late 1993. The features which are currently missing are listed in a separate chapter.

2 System architecture

The Paragon's processing nodes are arranged in a two-dimensional rectangular grid. The memory is distributed among the nodes. The system contains nodes for three different tasks: compute nodes, service nodes, and I/O nodes. Compute nodes are used for the execution of parallel programs; service nodes offer the capabilities of a UNIX system, including compilers and program development tools, thus making a traditional front-end computer unnecessary; and I/O nodes are interfaces to mass storage or external networks. All nodes are uniformly integrated in the interconnection network. The network provides fast routing of messages. The bandwidth between two nodes is 200 MB/s in each direction, virtually independent of the distance between the nodes. The communication uses wormhole routing with a deterministic routing algorithm. The start-up latency for a message issued by a program written in a high-level language is 30 μs.

Mass storage devices and external networks are attached to special I/O nodes. Disk arrays (RAIDs) having a capacity of 4.8 GB each provide internal disk space. They are built into the Paragon cabinets; each one is connected to a single I/O node with 5 MB/s bandwidth. For external disks, tapes, and networks SCSI-1, HiPPI, and Ethernet interfaces are available. A built-in Diagnostic Workstation is used for diagnostics and for booting the system. It is connected to the nodes by a separate network.

Each Paragon cabinet has a footprint of 56×107 cm, and can contain 64 nodes, each on a separate board, and up to 8 RAIDs or 6 RAIDs plus the Diagnostic Workstation. A full cabinet has a power consumption of about 5 kW; the system is air-cooled. Intel offers Paragon systems with up to 1024 nodes.

2.1 Node architecture

Compute nodes, service nodes, and I/O nodes are all realized by the same General Purpose (GP) node hardware (cf. Fig. **??**). All components of the GP node's compute and network interface parts are connected by a 32-bit wide address bus and a 400 MB/s 64-bit wide data bus.

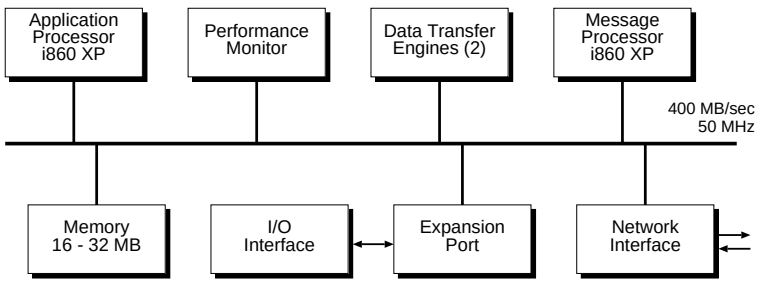

Fig. 1. Paragon node components

Compute part. The compute part includes an Intel i860 XP microprocessor, whose clock speed is 50 MHz (20 ns per cycle), and 16 or 32 MB of memory. The memory is constructed from 4 Mbit, 60 ns DRAM chips. It is organized in two banks and has single-bit error correction and double-bit error detection. The peak speed of data transfer between the memory and the processor caches is 64 bits per cycle, i.e. 400 MB/s.

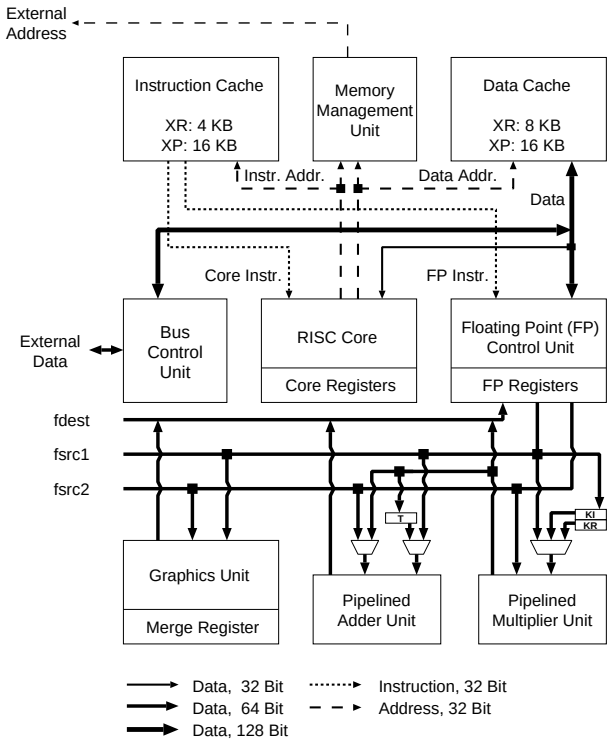

Fig. 2. Components of the Intel i860 processor (from [12])

The i860 XP is a fast compute-oriented RISC processor in 0.8 micron CMOS technology and contains more than 2.5 million transistors (cf. Fig. ?? and [?]). Address paths are 32 bits wide, and data paths are 32, 64, or 128 bits wide. Besides the RISC core, it has two 16 KB caches, one for data and one for instructions, and two integrated vector pipelines for 32-bit and 64-bit IEEE floating-point add and multiply. Furthermore, it has an on-chip memory management unit supporting the caches and virtual memory. Page sizes are 4 KB and 4 MB; the Paragon operating system uses only the small pages.

The data cache and the floating-point registers are connected by a 128-bit wide data path, enabling a data rate of 128 bits per cycle or 800 MB/s. As the pipelines can work simultaneously and the adder can deliver one result every

clock cycle and the multiplier one result every two clock cycles the theoretical peak performance of the i860 XP is 75 MFLOPS (64-bit arithmetic).

Remark. Practice has shown that programming the i860 is rather difficult. On the i860 XR, which is used in the iPSC/860 and has a clock frequency of 40 MHz and smaller caches, code generated by the Fortran compiler seldom exceeds 5 MFLOPS even when it vectorizes well, whereas assembler routines may run at nearly 40 MFLOPS [**?**]. For the i860 XP, up to 10 MFLOPS for pure Fortran code and more than 40 MFLOPS for assembler code can be expected. LINPACK performance for the solution of a 100×100 dense system of linear equations is 22 MFLOPS.

Network interface part. The interface to the interconnection network consists of a message processor, a network interface controller, and two Direct Memory Access (DMA) controllers. The message processor is a second i860 XP processor operating in parallel and sharing the memory with the application processor. Its task is to perform the details of inter-node communication and to provide global communication functions. Thus the application processor is not interrupted by message-passing operations; context switching and code turbulence and draining of the floating-point pipelines are avoided.

For outgoing messages, the message processor splits longer messages into packages, adds protocol information, and initiates the transfer. Incoming messages are autonomously received and the application processor is informed when a message has completely arrived in memory. The message processor also handles global operations independently including broadcast to and synchronization of a group of nodes as well as reduction operations on integer, floating-point or logical operands (e.g. global sum, global minimum, global and). The program executed by the message processor fits in its instruction and data caches, thus keeping the start-up latency for communication low.

The actual transmission of data between the memory and the network is effectuated by a special Network Interface Controller (NIC). It is assisted by two DMA controllers, one for inbound and one for outbound messages, which can operate in parallel. This aggregate provides a bandwidth of 200 MB/s for traffic in each direction.

Additional hardware. The Paragon GP node contains a data capture chip (RPM) that non-intrusively collects node performance data by monitoring bus activities. The data are read by the message processor and transmitted to a service node through the interconnection network. They are made available to the user through the Performance Visualization System (iPVS).

Furthermore, every GP node is equipped with an expansion port. On I/O nodes, this is used to attach a special I/O interface card.

2.2 Interconnection network

The Paragon has two communication networks: the high-speed data network and the diagnostic network. The diagnostic network is used for booting and diagnostics. It conforms to the IEEE 1149.1 JTAG specification [?]. This implements serial scan strings which provide access to the various Paragon hardware components.

The data network is the main communication vehicle between all nodes. For its topology, Intel has chosen a 2-dimensional mesh (cf. Fig. ??). It is constructed from Paragon Mesh Routing Chips (iMRCs) which are connected by high-speed channels. The channels are 16 bits wide and support a bandwidth of 200 MB/s. To each iMRC, one node may be attached. The iMRCs can, however, route messages autonomously and are independent of the attached node, and in most Paragon systems there is a number of iMRCs which have no node attached to them. Routers and channels are combined into backplanes carrying 16 routers (4 rows and 4 columns). Four of these active backplanes accomodate the nodes of one cabinet.

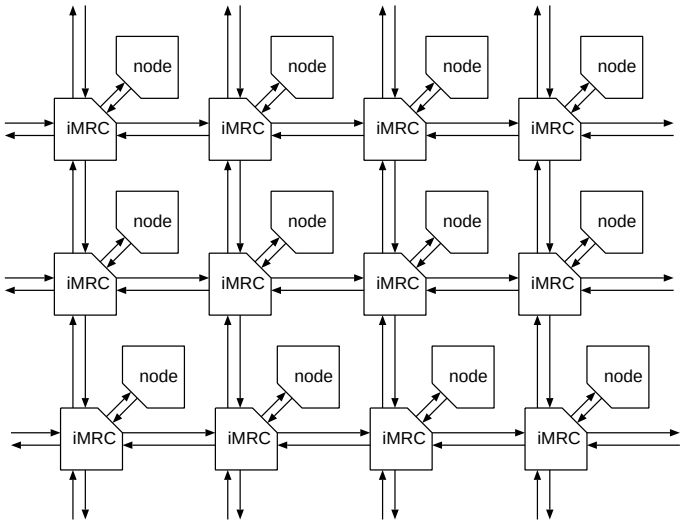

Fig. 3. Paragon interconnection network

The Paragon Message Routing Chip (iMRC). The iMRC is a message routing chip with 5 input and 5 output channels. One input and one output channel are connected to each of the 4 neighbouring nodes (east, west, north, south). The last pair of channels is connected to the Network Interface Controller (NIC) on the attached node. A 16-bit buffer on the iMRC is assigned to every input channel. The iMRC can route a 16-bit quantity of data from an

input buffer to any output channel. If the direction does not change, it takes 40 ns to make an individual routing decision and to close the resulting switches. The iMRC hardware supports broadcasting by automatically routing a message to all nodes in the rectangle between the sending and the receiving nodes.

Wormhole routing. The unit to be transmitted between nodes is one *packet*. The message processor on the node partitions the messages into packets of 8 to 1984 bytes (user controlled, default 1024 bytes); it also adds the necessary routing information to the packets and controls the transmission of packets between the network and the node.

Wormhole routing has been introduced as a fast switching technique for direct networks by Seitz and Dally [?, ?] to combine a pipelined transmission of packages with low storage requirements on the routers. In wormhole routing a packet is divided into a sequence of *flits* (flow control digits). The flit size in the Paragon system is 16 bits. A flit can be transmitted between adjacent routers in parallel in a single step.

The header flits of a package determine its path in the network. As the header advances along the specified route, the remaining flits follow in a pipelined fashion. When the header is blocked, because a channel that it wants to use is already in use by another packet, the whole pipeline stops. The sequence of flits remains in place, i.e. in the input buffers of the router chips until the requested channel is free. A packet's header thus reserves a path for its packet, and the individual channels are owned by the packet until the last flit has been transmitted.

The pipelined nature of wormhole routing makes network latency nearly independent of the distance between the sending and the receiving node, at least for longer messages. Furthermore, it requires only very small buffers on the routers. As it is difficult to synchronize a large network, synchronization is replaced by a handshake protocol between neighbouring routers. To implement such a *self-timed network* the Paragon uses a request line and an acknowledge line in addition to the data lines of each channel.

A serious problem with wormhole routing is deadlock. It can easily occur because packets travelling through the network constantly request new resources (channels) while occupying others. Deadlock is avoided by selecting adequate routing algorithms. The Paragon uses a simple but effective approach: messages are first sent in the horizontal direction and then in the vertical direction. A change of direction is allowed only once on the path. This algorithm is *minimal* (it uses a path of minimal length) and *deterministic* (as opposed to adaptive algorithms which can take collisions into account and make detours).

In detail, routing on the Paragon works as follows. A packet has two header flits containing the orientation of the path and the number of intermediate hops to be passed. The first flit contains the orientation and the number of hops in the horizontal direction. After deciding at the sending node whether to go left or right, the flit proceeds from router to router being decremented by one at every router it passes. When its value is zero, the first header flit is stripped off,

the direction is changed to vertical, and the second flit is used to determine the orientation and the number of hops to the destination.

Network performance. The bandwidth of each channel is 200 MB/s. This figure is virtually independent of the path length for longer messages. The start-up time for a message is 30 μs. This time is mainly used by the sending and the receiving nodes, e.g. for forming packets, adding routing information, recombining the packages, etc. The bisection bandwidth of the Paragon with fully equipped backplanes is $400\sqrt{n}$ MB/s (n number of nodes). This is a great advantage over the iPSC/860 that uses a circuit-switched network with a hypercube topology and has a bisection bandwidth of $2.8n$ MB/s. For 128 nodes, this means 4500 MB/s for the Paragon vs. 358 MB/s for the iPSC/860.

2.3 I/O and mass storage

Specialized I/O nodes located anywhere in the processor network act as interfaces between the processing nodes on one side, and mass storage and external networks on the other side. Their number is in principle arbitrary and does not depend on the number of compute and service nodes. An I/O node is built from a regular GP node by plugging a specialized adaptor card into the node's expansion port.

For different I/O requirements, two types of I/O nodes are available: the MIO node and the HiPPI node. The MIO node provides a SCSI-1 interface (5 MB/s, e.g. for internal disk arrays), an Ethernet interface (10 Mbit/s), and a V24 interface. The HiPPI node (100 MB/s) consists of two physical nodes; it serves to attach external disk arrays or frame buffers as well as FDDI networks (100 Mbit/s).

Every I/O request issued by a node is serviced by an I/O node. When, for example, a program requires data from a disk file, the data are read and buffered by the I/O node to which the physical device is attached, and are transferred to the requesting node via the interconnection network. All this happens transparently to the user program.

The primary mass storage systems of the Paragon are disk arrays (RAIDs) which are integrated in the Paragon cabinets. Each disk array has its dedicated MIO node to which it is connected via the SCSI interface. The RAIDs consist of five 3.5-inch commodity disk drives with a capacity of 1.56 GB each, which hold the data and parity information. The data are striped bytewise over the disks. In total one disk array can accomodate 4.8 GB of data. The RAID controller offers Levels 3 and 5 RAID functionality. In the event of a drive failure, the Paragon operating system, together with the controller, provides routines to rebuild the data. The file system remains intact and can be used while the reconstruction is performed.

In addition to the internal disk systems, connections and services to support external disk storage and backup systems are also available for the Paragon, including HiPPI, FDDI, and UniTree. Furthermore, there is a QIC-150 streamer

on the Diagnostic Workstation, and another internal 4 mm streamer tape can be attached to an MIO node.

Remark. For a medium size Paragon configuration with, for example, 140 compute nodes and a theoretical peak performance of 10 GFLOPS, Intel recommends four RAID systems, each attached to one MIO node. According to an optimistic estimate, one MIO node can sustain a bandwidth of 5 MB/s. Thus the I/O bandwidth of the whole system would be 20 MB/s. Divided by the number of nodes, this yields 140 KB/s for a single node. This rate is very low compared to the floating point performance of 10 to 20 MFLOPS per node usually achieved by well tuned applications. As all the I/O traffic has to go through the interconnection network with its 200 MB/s bandwidth concurrently to the ordinary message-passing, a simple increase in the number of MIO nodes with additional RAID systems will probably not alleviate the problem for large systems. The situation can even be aggravated if virtual storage that is offered by the operating system is actually used by programs and increases the I/O load. So the I/O performance may prove to be a major bottleneck of the Paragon.

2.4 Fault tolerance

The strategy on the Paragon for achieving a certain level of fault tolerance is twofold: there are several means for online diagnostics on one hand, and there are ways of concurrent repair and operation on the other hand.

A separate diagnostic network controlled by a Diagnostic Workstation monitors important system components including CPUs and memories on the nodes, iMRCs, I/O interfaces, and power supplies. The Diagnostic Workstation is also used for booting the system. Online diagnostics can test groups of nodes without disturbing the computational activities of other nodes. The interconnection network provides error detection capabilities in both hardware and the message-passing protocols.

When a fault has been detected in a node, a disk array, or a communcation component, the system administrator can reconfigure the system so that the rest of the machine can still be used. A faulty node is marked so that it is no longer allocated to users. This does not affect the communication network, since the mesh routing chip to which the node is attached can continue operation. The system must, however, be powered down, when the node is replaced. In case of a faulty disk in an internal RAID system the disk can be replaced during operation of the rest of the machine and the contents of the disk can be rebuilt online while normal disk I/O goes on.

3 Operating system

The Paragon operating system was designed to provide an application interface compatible with the OSF/1 operating system developed by the Open Software Foundation [**?**], the NX message-passing interface compatible with the iPSC/860, and a parallel file system extending the Concurrent File System of the iPSC/860.

3.1 Paragon OSF/1

To realize the application interface an "Advanced Development" multiprocessor version of OSF/1, called OSF/1AD, was developed. Like OSF/1, OSF/1AD is based on the Mach 3 *microkernel* developed at Carnegie Mellon University [**?**]. However, to support an environment without shared memory, the NORMA (NO Remote Memory Access) version is used. The Paragon operating system architecture is shown in Fig. **??**.

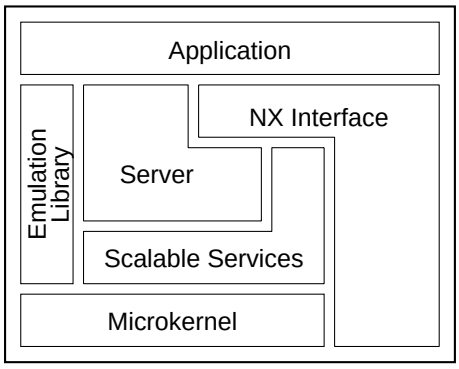

Fig. 4. Paragon OSF/1 operating system architecture

The Mach kernel provides threads, tasks, and ports as key mechanisms for building higher-level system services. These mechanisms are not usually directly exposed to the user. A *thread* constitutes a basic unit of execution. A *task* is a virtual address space in which a set of related threads execute with protected access to system resources. Threads can be executed concurrently with other threads, even within the same task. The conventional notion of a *process* is, in Mach, represented by a task with a single thread of control. Communication between tasks is based on a client/server system structure in which tasks (clients) access services by making requests of other tasks (servers) via messages sent over a communication channel called *port* (Inter Process Communication, IPC). A port is a unidirectional channel consisting of a queue holding messages managed and protected by the kernel. A task holds rights to these ports that specify its ability to send or receive messages. Only one task can hold the receive right for a port.

Each Paragon node runs the microkernel that supports basic system services. An OSF/1 *server*, implemented outside the kernel, runs on every node and provides access to all OSF/1 services including process management, file system, and network access.

In addition, an *emulation library* is linked to an application, implementing those parts of the operating system that can be executed in the same task as the UNIX process. There are three kinds of operations:

- Those which can be performed completely locally to the task, such as returning the process identifier.
- Those which use Mach services directly, such as creating a new thread.
- Those which call on the OSF/1 server.

These OSF/1 servers and libraries cooperate to offer the view of a single UNIX-like system to the user.

The *NX interface* provides a superset of the NX/2 message passing-interface. Applications use the fast kernel-level NX interface rather than the microkernel IPC for message-passing.

The Paragon OSF/1 supports virtual memory on all nodes.

File systems. The Paragon OSF/1 file system is based on the Berkeley 4.3 Virtual File System (VFS). VFS provides an abstract layer interface to different UNIX file system types, especially to the UNIX File System (UFS) and to the Network File System (NFS). UFS is compatible with the Berkeley 4.3 Tahoe release. On top of the VFS, which is a single-processor file system, the Distributed File System (DFS) has been built. This provides a common logical view of the file system structure for every process on every node. There is only one global directory tree which transparently combines local file systems of RAID disks and NFS mounted file systems.

In addition to its support for UNIX-type file systems, the Paragon operating system offers the Parallel File System (PFS), which provides file services at high data transfer rates by striping files across multiple I/O nodes and their attached RAID systems. PFS can be used with standard OSF/1 system calls and commands. For parallel applications there are also parallel I/O system calls including those of the Concurrent File System (CFS) that exists on the iPSC/860.

The UniTree client interface is available for access to servers for file exchange as well as backup and restore.

Single system image. A single system image across the multicomputer system for management of processes, files, user authorization, accounting, etc. is realized by *scalable services* which integrate the individual services of all nodes. For example, users are able to obtain status information regarding all processes in the system using the *ps* utility, killing a process is possible wherever the process may be. The DFS allows all I/O devices to be equally accessed from any node. The system is managed as a conventional one-processor system.

3.2 Partitioning and scheduling

The Paragon operating system allows the processor mesh to be divided into sets of nodes, called partitions. Partitions provide a means of restricting access to portions of the mesh for particular users or types of jobs and a way to specify different scheduling characteristics on different portions of the machine. In principle, a partition may comprise any arbitrary set of nodes. At least the following

partitions which are established by the system administrator must always be present (cf. Fig. ??):

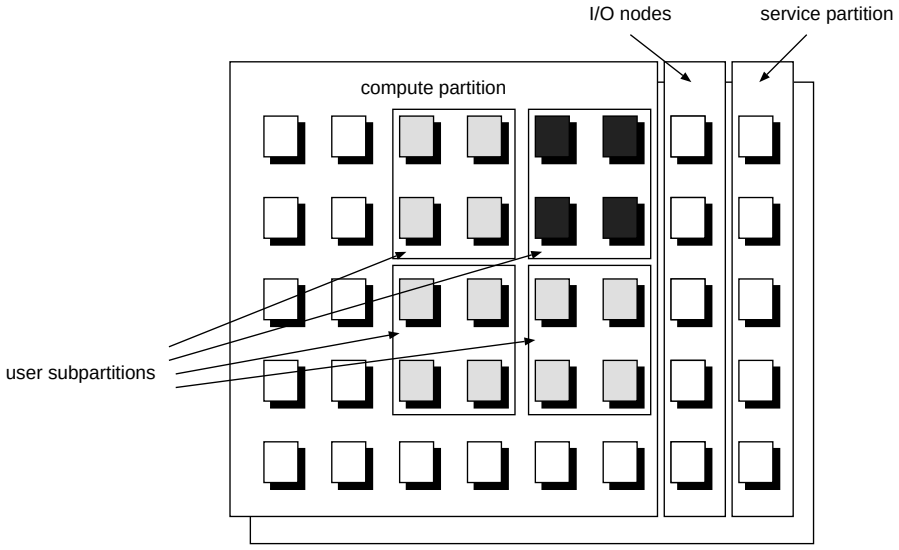

Fig. 5. Paragon partitions

- The root partition; this consists of all the nodes in the machine.
- The service partition; this supports general user services, such as editors, compilers and UNIX shells. Operating system services, such as the file server, are physically located on service nodes. Load levelling is supported within the service partition by automatically moving processes to less loaded nodes.
- The compute partition; this consists of the majority of the nodes in the system. Here the users' parallel applications are executed. The compute partition is hierarchical; i.e. it may be divided into subpartitions, which themselves may have subpartitions and so on. Subpartitions may also overlap. Subpartitioning of the compute partition can be done by the system administrator as well as by the user.

The I/O nodes can be grouped into an I/O partition. Normally, however, these nodes are added to the service partition.

Subpartitions are defined by specifying the parent partition from which to allocate the nodes, the specific nodes to allocate, access permissions, and the scheduling characteristics of the partition. Attributes similar to the UNIX file system modes control the access to partitions (for user, group, all):

- r allows the subpartitions of a partition to be displayed,

- w allows the attributes of the partition to be changed and to create and remove subpartitions, and
- x allows applications to run in the partition.

If access permissions or scheduling characteristics are not explicitly assigned they are inherited from the parent partition.

Depending on the type of the partition, different scheduling mechanisms are available. In the service partition, processes are scheduled in the usual UNIX-style timesharing mode, in which they run for a short period of time (typically 100 ms) or until they issue a blocking system call.

Parallel applications will usually run in the compute partition and will there be scheduled according to the *gang scheduling* mechanism. In this model all the processes which make up an application are scheduled at once on all the nodes on which the application has been loaded. The application will run until the end of its roll-in quantum, at which time the system will determine whether there is another application with equal or higher priority ready to run. Applications are not rolled out, when they issue system calls. Therefore asynchronous, i.e. non-blocking, I/O is provided. As the paging facilities of the operating system are used to page an application in and out, small applications need not be moved between disk and memory. The roll-in quantum is an amount of time specific of the partition and will be in the range of several minutes. Applications in overlapping partitions are always scheduled in a way such that only one application runs on any single node at one time.

In the compute partition, an additional scheduling mechanism is available, called *tenniscourt scheduling*. This allows a fixed timeslot in a partition to be reserved for a specific application. When on a single node more than one process is associated with an application, this set of processes is scheduled by the normal UNIX scheduling, while the application is active.

Remark. Subpartitioning of the compute partition together with gang scheduling and the Network Queueing System (NQS) provide a flexible and effective means to manage a workload of jobs with different resource requirements. It will, for example, be possible to create separate subpartitions for jobs requesting different numbers of nodes. To accomodate both long and short running jobs utilizing a large portion of the machine, gang scheduling with long time-slices seems to be a good choice as it avoids both frequent context switching and waiting for messages from other processes which are currently inactive.

3.3 System access and accounting

The user can develop and run parallel applications on the Paragon system either interactively with remote login facilities or by submitting batch jobs. The Paragon appears as a stand-alone UNIX system on the network connected to his workstation. After login, the user is running his shell somewhere in the service partition. Although some cross development tools are offered, the Paragon

requires no front-end, neither for program development nor for system administration.

The Paragon utilizes the Multi-User Accounting, Control, and Scheduling (MACS) system developed by the San Diego Supercomputer Center to manage the system resources and the Network Queueing System (NQS) developed at NASA Ames to manage batch jobs. It provides flexible, automated job scheduling schemes for assigning system resources. MACS allows simultaneous batch and interactive scheduling and control, using separate partitions for each. The scheduler allows jobs to be executed from the NQS queues. The system administrator specifies for each queue the number of nodes, the priority, and the number of jobs permitted to wait for execution. MACS can monitor and account for system resource usage, e.g. number of nodes and execution time of the job, producing data for analysis and reporting purposes. MACS includes facilities for automatic or selective preemption of jobs that have exceeded their resource allocations.

3.4 Message-passing

Processes in parallel applications use several facilities both in hardware and in software to exchange information. As mentioned earlier, these include the message processor, a second i860 processor on every node. The message processor shares memory with the application processor, and the two processors communicate with each other via shared variables. When the application processor wants to make a message-passing call, it places the parameters into these variables. The message processor regularly polls them and executes the call when it finds the information. The basic message-passing software, which, e.g. packetizes the messages and controls the transfer, runs on the message processor at the kernel level and can directly address the hardware. Thus, the involvement of the OSF/1 operating system in message-passing is minimal keeping the startup latency low.

Outgoing messages are sent directly out of the process memory. For incoming messages, a system buffer is provided on every node. It contains a distinct buffer for each node in the Paragon system. Every sending node knows the status of its associated buffers on all receiving nodes. A process sends the first packages of a message into this buffer. As soon as a receive is posted, the message processor will transfer the data into process memory, thus freeing space for subsequent packages. It will also inform the application processor when the message has completely arrived by setting a shared variable. If a receive has been posted before the message arrives, the packets go directly to the process memory. The various buffer sizes can be selected by the application.

4 Program development environment

Diverse programming models are supported by the Paragon system's development environment. A broad range of programming languages, optimized mathematical libraries, and a set of tools assisting the user to create new applications

or to port existing codes are available from Intel or as third party products. Most tools can be used on the Paragon service nodes or as cross-development tools on Silicon Graphics and Sun workstations. Several components of the Paragon system's development environment have been ported from the iPSC/860.

4.1 Programming models

The Paragon system supports several programming models. Besides the message-passing model, which is basic for most distributed memory multicomputers and is also the Paragon's primary programming model, the Paragon supports the data parallel model through High Performance Fortran (HPF) and the shared memory model through Shared Virtual Memory (SVM).

Message-passing. In this programming model independent processes are running asynchronously. They communicate by explicit message-passing. Messages are also used to synchronize processes. On the Paragon, several processes belonging to one application can run concurrently on one compute node.

In principle, the processes can be totally different programs. The most common programming style, however, is the *Single Program Multiple Data (SPMD)* style where the same program runs on each node in the application, but each node works on only part of the data. For *perfectly parallel problems*, each process can do its work without access to data held by other processes. For other types of problems, the processes must exchange data with each other to do their work. Another popular programming style is the *manager/worker* concept. One manager process starts several worker processes and assigns them their tasks. As soon as a worker process has finished its task it reports to the manager which accepts and interprets the results and assigns it a new task.

Data parallel. For the data parallel programming model, Intel provides High Performance Fortran (HPF) [?], an extension to Fortran 90. Parallelism is expressed in the program by array operations. Data distribution directives describe how arrays are to be distributed to the parallel processes. The programmer also specifies a mesh of processes which is then mapped to the real hardware. The compiler takes responsibility for inserting the explicit communication instructions required for running the code on a distributed memory system. The amount of communication and load balancing is determined by the mapping of data to processes. Each process is responsible to perform the computation for its assigned data. Therefore, the programmer can control the communication costs and load balancing with the help of the distribution directives. HPF provides two important benefits for the parallel programmer: a familiar programming model and portability by machine-independent specification of the data distribution.

Shared memory. The Paragon system also incorporates Shared Virtual Memory (SVM) [?] that allows building parallel applications in which data are logically shared between processes on one or more nodes. The distributed physical

memory forms a uniform global address space accessible on every node. There is no need of explicit message-passing. This facility simplifies the porting of large application programs but will not usually yield the performance obtainable by using the message-passing routines directly. On the Paragon, SVM can be used via the standard UNIX System V shared segment interface. Shared segments can be mapped into the virtual address space of different processes.

4.2 Message-passing libraries

The message-passing programming model is widespread, and many message-passing libraries have been developed. Some of these are related to a special distributed memory machine, e.g. Intel's NX message-passing library. Others are portable in a sense that they allow the user to specify processes and inter-process communication in a machine-independent way.

NX message-passing library. The Paragon OSF/1 operating system includes the NX message-passing library which is known from the iPSC series. Some extensions are supported on the Paragon including the possibility of allocating more than one process of a parallel application to a node. Synchronous (csend, crecv) and asynchronous (isend, irecv) messages, as well as messages producing interrupts (hsend, hrecv) are supported. Additionally, global operations for performing operations that use data from every node, e.g. for a global sum, are available.

Other message-passing libraries. Portable message-passing libraries on the Paragon include PVM of Oak Ridge National Laboratory, EXPRESS of Parasoft, PARMACS of the Gesellschaft für Mathematik und Datenverarbeitung (GMD), and TCGMSG of Argonne National Laboratory.

4.3 Languages and compilers

The Paragon system offers a set of programming languages. The languages are interoperable, allowing compiler output to be linked irrespective of the source language. Compiler switches allow different code generation strategies (e.g. scalar code, software pipelined loops, and vector code) to be selected, as well as different optimizations (e.g. scalar optimization, loop transformation, and cache management). The compilers exploit the advanced hardware features of the i860, such as dual instruction mode, dual operation instructions, and the arithmetic and load pipelines. They were originally developed for the iPSC/860 and have been adapted to the Paragon.

Fortran 77 and High Performance Fortran. Version 4.0 of the Paragon Fortran 77 compiler for Paragon OSF/1 is available. Compilation can be performed on the service nodes and using the cross compiler on Sun or SGI workstations. A precompiler for HPF, called xHPF, has been announced by Applied

Parallel Research. It takes as input a subset of Fortran 90 with HPF directives and produces a Fortran 77 program with embedded calls to a communication library.

C, C++, and Ada. Version 4.0 of the Paragon optimizing C compiler for Paragon OSF/1 is available on the service nodes and as cross compiler on Sun or SGI workstations. For object oriented programming Paragon C++ is available which is based on the AT&T cfront preprocessor which translates C++ into C. For the installation of this compiler a license of AT&T is necessary. An Ada compiler for Paragon OSF/1 is also available.

4.4 Program development tools

The tools available on Paragon are based on the Tools Application Monitor (TAM). This is a per-node, per-application server for tools that perform application process monitoring on the Paragon system. Figure 6 shows the co-operation of the program development tools.

Program analysis and restructuring. For the parallelization step Applied Parallel Research has developed the FORGE 90 parallel CASE tools [?], a set of interactive tools for parallel programming. These tools help the user to convert sequential Fortran programs into parallel programs and to design and implement new parallel algorithms. FORGE 90 is an analytical tool for understanding the concept and structure of sequential Fortran programs and supports the user in program restructuring and parallelization. FORGE 90 is available on the Paragon as a third party product.

Debugging. The Interactive Parallel Debugger (IPD) is a source-level debugger for large parallel application programs written in Fortran, C, or Assembler. The IPD allows context debugging to access and control selected groups of processes spread across multiple nodes. The IPD has a data reduction mechanism and facilities to examine the message-passing events and structures. Breakpoints can be set in some or all of the application's processes. The current version of the IPD on the Paragon is the same as the one available on the iPSC/860. An enhanced version, the IPD GUI supports a graphical user interface based on Motif.

Performance analysis. The runtime profiling tool prof860, which is a special version of the UNIX profiler prof for the Intel i860 processor, can be used to analyze an application program. By specifying a compiler switch, profiling data are collected on every node during the run. These are presented in tables and provide the user with information about the number of subroutine calls and the execution time of these routines.

For performance tuning of large parallel applications Intel has developed the Performance Visualization System (iPVS). It uses the hardware performance

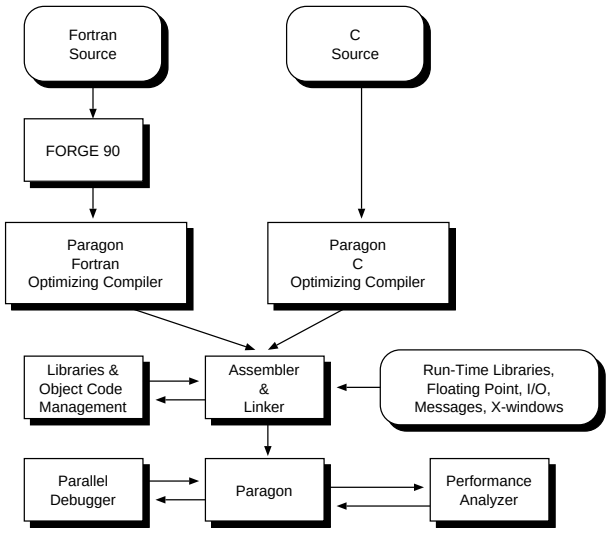

Fig. 6. Paragon Program Development

monitor RPM on each compute node, to collect data on runtime hardware performance and to provide low-overhead collection of software events. The user can control the data collection through compiler switches. The results can be presented in tables, graphs, or bar charts. Furthermore, the Paragon's front panel animation of the message-passing traffic and node activity can also be viewed in more detail on a workstation.

For software-based performance monitoring of applications, the Paragon offers a tool built on the ParaGraph display system developed at the Oak Ridge National Laboratory. This system presents an animation of the execution of parallel applications as derived from trace information gathered during program execution. In addition, the user can request graphical summaries and statistical analyses of overall program behaviour in a variety of display formats. ParaGraph which uses Motif replaces the Performance Analysis Tools (PAT) which are provided on the iPSC/860 for identification of time intensive parts in an application program.

Utilities. A parallel make utility, pmake, that maintains up-to-date versions of target files and performs shell programs in parallel, is available on the Paragon. It is an extension of GNU make. The pmake command updates multiple target files in parallel. Parallel execution may occur either in the service partition or the compute partition. In the service partition, pmake relies on process migration and load balancing to ensure efficient parallel execution. In the compute partition, pmake places commands on the available nodes within a partition and executes as a parallel application.

4.5 Optimized mathematical libraries

To ease the process of porting software to the Paragon and to provide means of exploiting the machine's computational power, Intel offers several libraries of mathematical routines, both node libraries with sequential routines and parallel libraries.

Node libraries. As the Paragon Fortran compiler does not always produce code that makes the most of the i860 processor, it is mandatory in order to achieve high node performance to employ optimized library routines, at least for the mathematical kernels of the applications.

The main library for this purpose is the Basic Math Library, an implementation for the i860 processor of the CLASSPACK Basic Math Library from Kuck and Associates. It contains the Basic Linear Algebra Subroutines (BLAS) Levels 1, 2, and 3 and also several FFT routines and solvers for tridiagonal and pentadiagonal linear systems. The Basic Math Library forms the basis for other libraries such as the public domain Linear Algebra PACKage (LAPACK) [?] that contains routines for the solution of dense linear systems and eigenvalue problems or for general commercial libraries like the NAG library.

The Signal Processing Library (SEGlib) includes a collection of signal processing routines that are compatible with the SEGlib Seismic Subroutine Standard Library.

The performance of the routines in the Basic Math Library ranges from 10 to over 40 MFLOPS (64-bit arithmetic). First timings for some BLAS routines gave the following results (MFLOPS, 64-bit arithmetic).

n	daxpy	ddot	dgemm
100	17.5	29.2	41.5
300	21.2	34.8	43.1
500	21.9	40.7	44.0

Parallel libraries. Intel has developed the ProSolver software package for the solution of large systems of linear equations. The matrices can be stored either on disk or in local memory. There are three separate products: ProSolver-DES applies a direct method to dense matrices, ProSolver-SES is a skyline direct solver for sparse matrices, and ProSolver-IES is an iterative solver for general sparse matrices using the conjugate gradient algorithm. The dense solver is expected to achieve 35 MFLOPS per node, the skyline solver 20 MFLOPS per node. For the iterative solver, which has been newly developed for the Paragon, a special distributed matrix interface has been designed including routines for basic manipulations of distributed matrices.

There is a collection of mathematical software developed for Intel machines by institutions outside Intel. References to this software can be found in the software catalog [?].

4.6 Visualization tools

The Paragon system provides several levels of support for network-transparent, client/server visualization. The X Window System X11 Release 5, PEX, and Motif are included in the system software. These tools enable client applications to run on the Paragon system and direct their output to a graphics workstation server. Additionally, the Distributed Graphics Library (DGL) is available for interactive graphical viewing of application results.

5 Hardware and software status, future developments

More than 25 Paragon systems have been installed to date (April 1993), one with 512 nodes and the others with 56 to 192 nodes. The Paragon hardware and software, however, are still under development and do not yet meet all design goals. Two milestones for the future development are the delivery of Releases 1.0 and 1.1 of the Paragon OSF/1 operating system. Release 1.0 is the first commercial version and will be delivered in May, 1993 and Release 1.1 is the first complete version and planned for fall, 1993.

Hardware. The hardware of the Paragon is fairly complete and stable. Hardware features scheduled for summer 1993 include the 32 MB memory extension, the HiPPI I/O node, and the 10 MB/s SCSI-2 I/O interface.

The most important enhancement planned for 1994 will be the multiprocessor node which will contain four i860 application processors sharing 64–128 MB of memory. Furthermore, Intel plans to implement a fast SCSI-2 I/O interface (20 MB/s) and a special network with nodes dedicated to performance monitoring.

Operating system. The operating system has been newly developed for the Paragon and is currently not yet up to its specifications. A significant problem for Paragon systems with 16 MB node memory is the fact that the operating system and associated buffers require a large part of the memory on the compute nodes. In the near future this amount probably cannot be reduced below 7 MB. Part of this is migrated to disk, but current experience shows that programs with more than 10 MB per node start paging.

High communication performance is currently prevented by the fact that instead of the message processor the application processor is still used for controlling the message traffic. This will change in Release 1.1.

The Parallel File System will be supported in Release 1.1. Currently the operating system provides a UFS-based emulation of the Concurrent File System (CFS) from the iPSC/860. Files greater than 2 GB are not supported. The asynchronous I/O calls (iread, iwrite) are provided for source compatibility, but do not operate asynchronously. I/O in general is rather slow, as it uses Mach IPC calls for the communication with the I/O nodes.

The NQS batch system is supported in the current release, but the MACS resource control and accounting system will only be available in Release 1.1. Gang scheduling will also be supported in Release 1.1.

A Shared Virtual Memory interface for Fortran has not yet been determined. In a future release there will be a possibility to exchange messages between different programs running on a single Paragon system or running on different systems connected via ethernet (Remote Message Passing).

Programming tools. Many components of the Paragon's programming environment were developed for the predecessor systems and have been adapted for the Paragon. The following table gives an overview of the availability of software scheduled by Intel for 1993.

Product	Release 1.0 May 1993	Release 1.1 Fall 1993
NX	•	
Fortran 77	•	
C	•	
C++ preprocessor	•	
IPD	•	
IPD GUI		•
prof860	•	
iPVS		•
ParaGraph		•
parallel make	•	
BLAS	•	
FFT	•	
LAPACK		•
SEGlib		•
ProSolver		•
DGL	•	
X11 Motif	•	

From independent software vendors, the NAG library and FORGE 90 are available in Release 1.0. Intel's plans for 1994 include a compiler for High Performance Fortran (HPF).

Acknowledgement

The authors would like to thank H. Bast and J. Finger from Intel for their substantial support. Fruitful discussions with our colleagues, especially R. Berrendorf, U. Detert, H.M. Gerndt, and I. Gutheil are gratefully acknowledged.

References

1. Anderson, E., et al.: LAPACK User's Guide. Philadelphia, SIAM, 1992

2. Applied Parallel Research: FORGE 90, Version 8.0, User's Guide. 1992
3. Dally, W.J., Seitz, C.L.: The Torus Routing Chip. J. Distributed Computing, Vol. 1, No. 3 (1986) 187–196
4. Esser, R., Knecht, R. (eds.): Applications on KFA's Intel iPSC/860. Interner Bericht, KFA-ZAM-IB-9218, Forschungszentrum Jülich, 1992
5. High Performance Fortran Forum: High Performance Fortran Language Specification (DRAFT), Version 0.4. Rice University, Houston TX, 1992
6. IEEE STD 1149.1-1990: IEEE Standard Test Access Port and Boundary Scan Architecture.
7. Intel Corporation: i860 Microprocessor Family Programmer's Reference Manual. Order No. 240875-002, 1992
8. Intel Supercomputer Systems Division: iPSC/860 Software Resource Catalog. Beaverton OR, December 1991
9. Lee, K.: On the Floating Point Performance of the i860 Microprocessor. Int. J. High Speed Computing, Vol. 4, No. 4 (1992) 251–267
10. Li, K.: Shared Virtual Memory on Loosely-coupled Multiprocessors. PhD Thesis, Yale University, Technical Report YALEU-RR-492, October 1986
11. Loepere, K.: Mach 3 Kernel Principles, Open Software Foundation and Carnegie Mellon University, 1992
12. Mlynski-Wiese, A.: Die Architektur der Prozessorfamilie i860. Interner Bericht KFA-ZAM-IB-9214, Forschungszentrum Jülich, 1992
13. Ni, L.M., McMinley, P.K.: A Survey of Wormhole Routing Techniques in Direct Networks. IEEE Computer, February 1993, 62–76
14. OSF/1 Operating System, User's Guide. Open Software Foundation, Prentice Hall, 1992

This article was processed using the LaTeX macro package with LLNCS style