Tender to III/97/31 Lot 5, Deliverable 1.1 DISCO Report on the state-of-the-art of PC Cluster Computing

M. Bertozzi[1], G. Chiola[2], G. Ciaccio[2], G. Conte[1], P. Marenzoni[3], A. Poggi[1], P. Rossi[4]

DISI-TR-98-09, December 1998

[1]DII, Università di Parma, viale delle Scienze, 43100 Parma, Italy. Email: {bertozzi,conte,poggi}@ce.unipr.it

[2]DISI, Università di Genova, via Dodecaneso 35, 16146 Genova, Italy. Email: {chiola,ciaccio}@disi.unige.it

[3]Email: marenzoni@dsdata.it

[4]ENEA HPCN-BOL, via Martiri di Monte Sole, Bologna, Italy. Email: rossi@bologna.enea.it

Abstract

This report surveys the state of the art of Cluster Computing based mainly on low-cost PC or workstations technology. Real industrial applications as well as EU funded and international University/Research projects are taken into account in order to provide an overall (although necessarily not exhaustive) view of the current situation and trends.

Cluster computing is mainly obtained by providing a set of PCs or workstations connected by a more or less sophisticated local area network (LAN) with appropriate communications libraries and/or operating system (OS) primitives, and by adopting appropriate application-level languages, libraries, and/or environments to support parallel processing. After a first chapter reviewing already existing industrial applications (that should be regarded as main practical motivations to pursue this approach), we present in subsequent chapters the enabling technologies that have been adopted at the various application, LAN, OS and system library levels. Another chapter reviews past and currently active EU supported R&D efforts in the field, with the intent of reaching a better understanding whether the EU policy in this field is adequate and/or conformant to the one followed by other developed countries. Finally, we conclude with a chapter discussing costs, performance, and trade-offs among the two, in the context of Cluster Computing.

Contents

Chapter 1

Analysis of Experiments with Cluster Computing

Cluster computing is commonly viewed as a specialized method of computing used only by the scientific community for research in odd, extremely complex, computational intensive applications. This view has been supported by the rising popularity of computer clusters as a low-cost alternative to traditional supercomputers in scientific laboratories, universities and supercomputing centers. Hence this perception of cluster computing has become a widespread de-facto definition.

However this figure presents a very narrow view of cluster computing. It excludes a large percentage of today's cluster computing applications aimed at commercial uses. If we expand our definition of a cluster to a network of machines viewed as an aggregate computing resource, we can encompass both scientific and commercial computing worlds.

With the desire to put valuable information at the employees' fingertips and reduce mainframe computing maintenance costs, businesses have moved the computer out of the computer room and onto the desktop. Several commercial companies have replaced mainframes with networks of workstations, resulting in the birth of commercial clusters.

While a common definition can meaningfully be used to describe both scientific and commercial cluster computing, the two operating environments are actually very different. Scientific cluster environments were born from the desire to create a low cost alternative to the supercomputer. As a result, scientific environments are homogeneous and typically composed of a not very large number of high powered machines of the same make and model. In contrast, commercial clusters, typically referred to simply as "networks," are composed of hundreds of workstations. Most of these networks have expanded over time, and contain many makes, models, and configurations of workstations from a variety of hardware vendors. Although the collection of such individual machines alltogether form powerful computing networks, these networks have not been viewed yet as a single, aggregate cluster of computing resources. That view, however, is changing.

The desire to move mission-critical applications off of mainframes has fueled the development of larger, more sophisticated applications for workstation networks. The resource requirements of these applications vary widely. Some applications are data intensive, others are CPU intensive, and still others are memory intensive.

As a matter of fact, commercial applications of this technology are not limited to the assembly and use of high-end workstation networks. They also start to exploit less expensive clusters of high performance PCs. In the last few years the impressive growth of the computational power expressed by personal computers made the PC technology extremely favorable with respect to traditional, more expensive workstations for its price/performance ratio.

So far, the wide practical applicability of networks of PC's into commercial projects has encountered one main obstacle: the availability of (really distributed) operating systems (OSs) that can be efficiently exploited by useful applications. Two major OS environments are currently available on PC-based platforms, namely Windows NT and Linux. The technical background needed to assemble and configure networks as well as to develop and implement distributed, computing and/or data intensive applications is traditionally founded on Unix systems. Only recently the Windows NT OS has appeared as a viable alternative for commercial, computing intensive, parallel or distributed applications. On the other hand, a wide applicability of Linux-based systems for commercial uses encountered two major problems: difficulties to install, configure and maintain the network and the lack of commercially supported applications running on Linux platforms.

In scientific environments, personal computers with Linux OS currently are a widely assessed substitute of more expensive traditional workstations. In this context, a deep knowledge of all the details involved by installing and maintaining such rapidly evolving systems are widespread in the community. On the contrary, there are yet high barriers for accepting Linux in the corporate world:

Support and legal responsibility are the first. Corporate MIS wants someone to take responsibility for applications running on their computing environments. Linux instead has incredible support through the Internet, but that is not the same as having someone's name on a legal document guaranteeing support. This is the main reason why some commercial Linux vendors like Caldera, Red Hat and VA Research recently appeared. The MIS people needs to see a large stable company that will be there to back them up in every situation.

Availability of applications is another problem. Ninety-five percent of the corporate world doesn't care what OS the desktop runs, but whatever OS it is, they want it to support the latest version of widespread, industry standard commercial applications.

Perception sometimes can also be a barrier. The perception in the corporate world is that Linux is good for hackers only. In corporate domains, systems ready to install and easy to be configured and maintained are preferred. Newer releases of Linux (e.g., Red-Hat) of course addressed the guided installation problem in a quite satisfactory way, however common perception is very slow to change once established.

These considerations limited the number of commercial applications based on PC clusters, in spite of extremely favorable price/performance appeals. The public domain availability of the Linux OS for platforms other than PC's allows its exploitation also for commercial large scale projects based on workstation clusters. The most important project based on a very large Alpha cluster running Linux has been recently carried out: the production of visual effects for the "Titanic" movie. We will describe some details of this interesting, very computing intensive project.

Furthermore, this chapter will present a list of other commercial experiences performed on workstation clusters and PC clusters. Several European Union funded projects exploiting these technologies and carried out with industrial partners have also been undertaken. However we prefer to consider these projects in a separate chapter of this report, since usually the research and institutional technology transfer efforts are dominating the pure enterprize spirit in them.

1.1 Simulation, technical and scientific computing

The original idea of exploiting clusters of workstations or low-cost PCs for scientific computing applications originated in the U.S.A., both in University and in advanced research centers. In this context, the focus was of course in identifying and trying to solve the problems that prevented the naive exploitation of such platforms, namely the inter-processor communication bottlenecks.

The experience gained allowed both to develop more and more efficient software platforms for the exploitation of low-cost hardware, and to identify applications that could better tolerate the intrinsic limitations of such low-cost platforms. In the higly dynamic U.S.A. society, it was not difficult also for high-tech private enterprises to try and exploit the new technologies developed by Universities, NASA and other National Labs for commercial applications

The Titanic movie production, performed with a very wide cluster of Alpha workstations running Linux, is maybe the most important "spontaneous" commercial experience built upon public domain Linux systems. Always in the same application field we will briefly summarize the production of "Toy Story," one of the first significant experiences carried out with a large workstation cluster rather than traditional supercomputers.

1.1.1 U.S.A. university projects

The Berkeley NOW is probably one of the first and best known world-wide university research projects in the field. In the context of this project the NOW acronym (Network Of Workstation) was invented. The idea was to substitute expensive massively parallel machines with a cluster of workstations connected by (more or less) off-the-shelf, high-speed local area networks for interprocessor communications. At the time the project started SPARC workstations and

ATM technology was considered enough low-cost as compared to massively parallel platforms. Within the framework of this project the new concept of "Active Messages" was also invented in order to support higher performance communications.

Similar research projects also started at Cornell University, leading to the development of the U-Net prototype, a fast messaging system whose software components were intended to be detached from the Operating System in order to avoid the inefficiency of classical TCP/IP communication protocols usually adopetd for LAN communications. Eventually U-Net was also ported over Fast Ethernet technology, where it could demonstrate all the potentials of lowest cost interconnects for clusters of PCs. The experience of Cornell's group eventually resulted in the definition of a new standard adopted by major hardware manufacturers for high-speed interconnects, called the Virtual Interface (VI) Architecture.

The Illinois Fast Message (FM), in co-operation with the Berkeley NOW group, eventually developed an optimized messaging system based on Myrinet technology, a Gigabit per second experimental network characterized by extremely low hardware switching latency.

1.1.2 Beowulf and its descendants

Also major research labs in the USA soon realized the potentials of low-cost clusters, after the pioneering work on the Berkeley NOW and Cornell's U-Net. In this case the approach was much more pragmatic than in case of University projects, and the idea was to immediately exploit the cluster prototypes for computation as well as data intensive applications, traditionally addresses by means of parallel platforms. The first effort in this sense was the Beowulf project carried out at NASA CESDIS Labs, where a cluster of extremely low cost PCs connected by Fast Ethernet technology was assembled in order to address large database problems. The emphasis was in this case on the low cost of the hardware platform rather than the performance obtained (that was nevertheless considered enough for the application, and in any case higher than using single processor techniques). In the context of the Beowulf project, large emphasis was also devoted to the adoption of free software, the project being based on the Linux OS kernel and the adoption of a GNU environment. New, efficient Linux drivers were developed for 100Mb/s Fast Ethernet Network Interface Cards (NICs), thus contributing to the diffusion of this technology.

Other famous clusters such as the Avalon prototype were developed at the Los Alamos National Laboratory with the aim of obtaining high computational power at lowest cost for Physics simulation applications. Also in this case the interconnection technology (that happened to be not very critical for the considered applications) was 100 Mb/s Fast Ethernet. Also in this case the emphasys was on the use of cheapest, off-the-shelf hardware and software components to address problems that traditionally required massively parallel platform technologies. Also in this case a strong emphasis was devoted to the adoption of the Linux/GNU operating system environment.

As a results of these efforts, a special release of Linux (called the "Extreme

Linux") incorporating some of the optimized communication drivers and applications developed for Beowulf is now available. This, in principle, allows so called "out-of-the-box" replications of the above clusters in other organizations, facilitating the diffusion of the idea of exploitation of PC clusters for real applications in various domain of interest.

1.1.3 Titanic

The production of the film Titanic required a large number of challenging visual effects. The work was performed by Digital Domain, an advanced production studio located in Venice, California, selected to produce the Titanic.

Digital visual effects constitute a large portion of the whole work. For many digital effects shots, original photographic images are first shot on film (using conventional cinematic methods) and then scanned into the computer. Each "cut" or "scene" is set up as a collection of directories with an "element" directory for all the photographic passes that contribute to the final scene. Each frame of film is stored as a separate file on a central file server. A digital artist then begins working on the shot. The work may involve creating whole new elements such as animating and rendering 3D models or modifying existing elements such as painting out a wire or isolating the areas of interest in the original film. This work is done at the artist's desktop (often on an Silicon Graphics or NT workstation). Once the setup for this work is done, the process is repeated for each frame of the shot. This batch processing is done on all the available CPUs in the network, often in parallel, and requires a distributed file system and uniform data overview. A goal of this processing is to remain platform independent whenever possible.

Once all the elements are created, the final image is "composited." During this step, the individual elements are color corrected to match the original photography, spatially coordinated and layered to create the final image. Again, the set up for compositing work is usually done on a desktop Silicon Graphics, and the batch processing is done exploiting the cluster.

Since building a full-scale model of the Titanic would have been prohibitively expensive, only a portion of the ship was built full size (by the production staff), and miniatures were used for the rest of the scenes. To this model other elements of the scene were added, such as the ocean, people, birds, smoke and other details that make the model appear to be docked, sailing or sunk in the ocean. To this end, a 3D model and photographed 2D elements to simulate underwater, airborne and land-based photography have been built.

The work on Titanic has been performed on a network comprizing approximately 350 Silicon Graphics CPUs and 200 DEC Alpha CPUs, all connected by a 100Mbps or faster network, with 5 terabytes of disk space overall. A DEC Alpha-based cluster was the computing platform chosen by Digital Domain for the previously mentioned batch-processing needs. The main motivation given by Digital Domain to support their decision is the attractive price points offered by that platform, considering the high floating-point performance levels achieved in their intensive applications. Moreover, the Alpha systems can be configured with large amounts of memory and fast networking.

The other main problem concerned the choice of the operating system to use. Windows/NT, DEC UNIX or Linux were the considered alternatives. Windows NT had several shortfalls for Digital Domain needs. First, their standard applications, which normally run on SGI hardware, were not available under NT. NT also has several other limitations: it did not support an automounter, NFS or symbolic links. Other third-party applications were available to fill some of these holes, but they increased the total cost and, in many cases, did not perform well in handling the general computing needs. Digital UNIX performs very well, but with disadvantages due to cost and lack of flexibility. Linux was the operating system preferred. The flexibility of the existing devices and the availability of source code gave Linux a definitive advantage, in order to be adopted by Digital Domain for the Titanic production. The Linux distribution used was Red Hat 4.1 and 2.1.42 kernel with a few patches. The motivations typically adopted by companies to discard Linux as their computing platform, that is configuration difficulties, maintenance problems and lack of technical support during operations, did not influence at all the choice made by Digital Domain.

The cluster was build out of 160 433MHz DEC Alpha systems. Of those 160 machines, 105 of the machines were running Linux, the other 55 were running NT. The machines were connected by 100Mbps Ethernet to each other and to the rest of the LAN.

The first part of the work was dedicated to simulate and render the water elements, a computationally very expensive task. These jobs computed for approximately 45 minutes and then generated several hundred megabytes of image data to be stored on central storage servers. Intermediate data were stored on the local SCSI disk. As the water rendering completed, the task load then switched to compositing.

Digital Domain experts consider the experience made using Linux systems very effective. They worked well for the problem and the cost/benefit ratio was overwhelmingly positive. They concluded that the choice of coupling commodity Alpha workstations with public domain software OS as Linux yielded good results for this commercial application. Indeed this commercial, computationally intensive application of the public domain Linux OS can be considered as one of the most important experiences for this system.

1.1.4 Toy Story

In 1995 the first important commercial application appeared where price versus performance considerations led to the exploitation of a network of high performance workstations instead of expensive supercomputers: the production of visual effects for the "Toy Story" movie.

The first fully computer animated film in the world, Toy Story, was written, produced, and directed by Pixar Animation Studios. They exploited for that graphical rendering a large cluster of Sun Microsystems SPARCstations. The movie's final image rendering was accomplished on a "farm" of 87 dual-processor and 30 quad-processor 100-MHz SPARCstation 20s, representing at that time more computing power than 300 Cray 1's. Even with that, rendering the film's

110,000 frames required the equivalent of 46 days of continuous processing. In other words, rendering each frame took one to three hours of SPARC processor time. Each workstation was equipped with 192 to 384 megabytes of RAM and three to five gigabytes of local disk storage.

The cluster of SPARCstations was connected using hubs of 10BaseT and 100BaseT links. There was also a SPARCserver 1000e in the network (connected to the rest of the company's network) storing models of different objects retrieved as needed by other workstations.

1.2 Data management, data bases and WWW

Some important applications based on network of workstations can be found when addressing the problem of indexing and retrieving data from very large databases, such as when building web-based search engines. Two among the widest known search engines on the Web are based on network of commodity workstations: Altavista by Digital, and HotBot by Inktomy.

The Web is the graphically-oriented portion of the Internet that is currently experiencing exponential growth both in the number of users and in the amount of available data. In Spring 1998 there were over 50 million HTML pages on the Web with an aggregate size of about 0.5 terabytes. The numbers increase dramatically when including images, movies, and audio files. One popular service for searching pages on the Web is "Yahoo!," a directory of Web pages. It provides a hierarchical organization of approximately 100,000 web pages that were assembled individually by the editors of "Yahoo!." The other type of service that is available today is an index.

HotBot and AltaVista are the most comprehensive indexes on the Internet, covering respectively about 50 million and 30 million pages. They represent full-text indexes, i.e., they index every word in each document. Both search engines apply parallel computing technology in order to build indexes that as a matter of fact are not limited to a mere fraction of the Web. In particular, both applications run on cluster technology implemented in the form of workstation networks.

1.2.1 Altavista

AltaVista is a very popular search engine on the Internet, developed by Digital in 1995. AltaVista Search Public Service runs exclusively on several Digital Equipment Corporation Alpha computers. The choice by Digital to exploit their own proprietary workstations is obvious, but it also allows high performance at considerably low costs. Alpha computers, as mentioned during the overview of the Titanic production, are sophisticated systems which are particularly suited for storing huge amounts of information and serving that information quickly when the user submits a search query. The AltaVista system is able to support and serve user queries searching on the AltaVista's entire Web index (composed approximately by 200 Gigabytes of data) in less than a second.

All of the software on AltaVista systems is developed entirely within Digital's research laboratories. The software, which is written in C under DIGITAL

UNIX, makes heavy use of multi-threading and the Alpha's 64-bit addressing capabilities.

There are six main functionalities implemented within the Alpha cluster.

1. The query interface runs on a set of three Dec AlphaStations 500/500s, each with 1 GB of RAM and 6 GB of hard disk space, running a custom multi-threaded Web server that accepts the queries and sends the queries along to the index servers. These systems, relatively small by AltaVista Search standards, forward millions of hits per day to the AltaVista Search index servers, with roughly 90 percent of the queries going to the Web and the remaining 10 percent to newsgroups.

2. The query engine is the "kernel" of the cluster. It is composed by a set of sixteen AlphaServer 8400 5/440 systems, each with 12 processors, 8 GB of RAM, and 300 GB of hard disk in a RAID array. (RAID systems are sets of hard disks that work together and ensure that if a hard disk fails, no information is lost.) Each server holds a portion of the Web index and provides response times of less than a second.

3. The Scooter (or the super-spider) is an AlphaServer 4100 5/300, with 1.5 GB memory and 30 GB RAID of hard disk space. All Scooter has to do is roam the Web, retrieve information, and send it to the indexing system (Vista), which compiles the index.

4. Vista, the Web indexer, is an AlphaServer 4100 5/300, with 2 processors, 2 GB memory, and a 180 GB RAID disk. This machine indexes Scooter's output and serves as a central distribution point for the new index data.

5. The News Indexer is an AlphaServer 4100 5/440 with two processors, 1 GB of memory and 13 GB of disk space. This machine keeps an up-to-date index of the news spool. Because new articles appear and old articles expire all the time, it remains constantly quite busy, even though the index it serves is much smaller than the Web index.

6. The News Server is an AlphaServer 4100 5/440 with two processors, 1 GB of memory and 24 GB of disk space. This system maintains a current news spool for the News Indexer as well as serving the articles via the Web to those who don't want to know about news servers but want to read USENET news.

Moreover, another characteristic of the AltaVista architecture, due to the very large overall RAM memory available within the workstation cluster, is that the AltaVista system is able to hold a lot of information in RAM so that the cluster can produce search results in very small times. Ideally, the AltaVista cluster is able to hold the most frequently accessed information in RAM so it doesn't have to waste time fetching new items from disk memory. With this architecture, a typical query takes 0.2 seconds to be answered.

1.2.2 HotBot

The other experience considered in this section exploits primarily commodities computing resources (hardware and software) instead of proprietary workstations, such as those exploited in ALtaVista. HotBot by Inktomi is another very powerful Web search engines present on the Internet. The technology behind HotBot wants to combine on one hand cost-effectiveness and on the other hand such features as scalability, fault tolerance, and mass customization. As a result HotBot represents the first use of commodity building blocks for parallel computing in a very large Internet server.

The goal is to bring leading-edge parallel computing technology to the Internet using commodity building blocks that provide very favorable cost and performance, nearly unlimited scalability, and high levels of fault tolerance and availability. The HotBot search engine, developed in collaboration with HotWired, is one among the first products on this path and represents one of the most important engines to exploit multi-machine parallel computing.

For HotBot, this edge directly enables one of the largest Web indexes and creates a system that can easily grow with the Web by just adding more commodity building blocks. Moreover, the clustered nature of the service leads to high availability even in the presence of hardware or software faults. It also lets HotBot staff hot-swap disks, machines, software, and even the database while the service is in active use. Finally, the cost-effective use of computing has led to incorporate extensive support for dynamic content (including database support, user customization, and browser independence).

The technology implemented by the HotBot engine can be viewed as the combination of four previously independent technologies into a coherent whole:

- The use of Networks of Workstations (NOWs) to build large-scale cluster of computers out of commodity workstations and high-speed local-area networks. This technology enables low-cost systems with unlimited scalability: the performance or database size can increase simply by adding more commodity building blocks (machines, disks, or memory) to the collection.

- The development of user-level networking and application software that enables a collection of machines, disks, and network switches to mask the faults of individual components. The nature of unlimited scale implies that the number of building blocks is large enough that some amount of failure is inevitable; it is critical that application and networking software mask these faults from the user.

- The integration of traditional databases into scalable Web servers. Although the primary database for the HotBot search engine is custom-built for high performance, an integrated multi-machine Informix database is used for tracking user preference profiles and add placement and accounting. Informix provides multi-platform parallel database queries that fit well with the building-block model used by Inktomi: each server has the full power of SQL transactions and information is replicated to provide fault tolerance.

- The pervasive use of dynamic HTML generation to allow every user to see a customized page. The use of mass customization requires scalable computing resources and database integration, but also requires new tools and technology.

1.2.3 Summary

The main difference between the two approaches described above is that, while HotBot wants to apply a highly scalable technology by assembling only cost effective commodities building blocks, AltaVista wants to address the issue of achieving the highest computation powers by exploiting high performance proprietary hardware and software components. The HotBot choice excludes the use of either a single big computer (as in InfoSeek's recently announced Ultra-Seek) or a set of independent high performance machines (as used by Alta Vista) with completely distinct tasks that do not help with managing the database size. Like the single big machines, the HotBot engine has the flexibility of leveraging huge amounts of memory and disk for a single unified database but it is not limited by the size of the single machine. Inktomi's underlying parallel processing provides a level of fault tolerance that is inherently beyond a single large computer. To increase performance or expand a database, another commodity building block can be added. If a building block fails, the responsibility of that block falls to the others which perform the necessary functions unbeknownst to the user.

To summarize, a NOW is the correct base even for a data intensive Web service because it is high performance, cost-effective, incrementally scalable, and fault-tolerant. The relative balance of CPU power, memory, and disk performance can be tuned simply by picking the correct off-the-shelf components and a leading-edge network can be used to speed-up interprocessor communications.

Chapter 2

Hardware and Software Technology: state-of-the-art and trend

The role played by the interconnection hardware when implementing NOW or cluster structures is crucial. While the use of industry-standard, off-the-shelf processor has already consolidated also in the domain of MPP architectures and nobody questions this choice anymore, it is a consolidated fact that in the last ten years the main efforts of all the companies producing commercial high-performance multiprocessor systems have been devoted to the interconnection structure, both in the case of distributed (message passing) systems and of shared memory systems.

Traditional operating system support for communications in local area networks is usually considered not appropriate for NOW/cluster computing, due to extremely poor user-level performance as compared to the potentials of hardware technology. Therefore low-level software architecture for message passing systems specifically oriented towards NOW/clusters have been developed, that either improve or by-pass the traditional operating system communication structure.

Thanks to its superior intuitiveness message passing has been the most used communication paradigm for parallel machines. For this reason a number of applications have been *casted* in this paradigm and many parallel machine vendors implemented their own variants. Unfortunately the presence of a number of variants made it difficult to port applications between different architectures. Two major de-facto standard have been consolidated to address this shortcoming: PVM and MPI. Both (PVM right from the beginning, MPI with its more recent extensions) address also re-configuration and fault-tolerance issues.

Object-oriented programming languages and libraries are used with success to develop programs for NOW architectures and, on the whole, for parallel machines [66]. Message passing is the natural communication paradigm for the interaction between objects, therefore, message passing libraries (such as, e.g., MPI) can be encapsulated in objects to support their interactions.

2.1 Interconnection technology for NOW/cluster

The main trend we may notice is the gradual moving from the expensive, special purpose interconnection structures typical of today commercial MMP architectures to the off-the-shelf standard solutions. It must be pointed out that a quite similar switch of emphasis from custom to off-the-shelf technology has already taken place a few years ago in the case of the CPU's. Such shift of design strategy is mainly motivated by scale economy and market sizes.

Very different solutions are possible today in order to interconnect the processing nodes of a NOW according to the needs of the different applications, to the performance required, and to the current cost as well as the projected cost in the near future.

The main parameters that must be considered are:

Bandwidth This parameter allows to evaluate the ability of the system to transfer large amount of data in a reasonably short amount of time.

Latency Reductions in communication latency (i.e. on the time needed for a receiver to notice the arrival of a message after the sender sent it) allow to reduce the grain of exploited parallelism. It is a key parameter to obtain high performance and reasonable scalability.

Cost Due to the always decreasing cost of PC nodes, the cost to connect a processing node to the system (network) must be at least comparable if not negligible to the cost of the node itself. The cost of off-the-shelf standard hardware is one order of magnitude lower than the cost of dedicated solutions. Moreover, the projected trend of the cost for off-the-shelf technology in the near future tends to follow the one of PCs due to expanding market shares. The cost of special purpose hardware can be in the range of the cost of a single PC node, and is not expected to benefit of market scale effects, at least not to the same extent as off-the-shelf technology.

Switching/Broadcast Switching of point-to-point communications as compared to broadcast or multicast "collective" communications.

2.1.1 Standard LAN technology

In the following we try to briefly discuss the various available and/or proposed solutions. It must be considered however that the solutions that are available are continuously changing and that this section had to be updated very frequently.

ATM ATM network protocol is intended to be used over long distance interconnections to manage efficiently multimedia streams. In the wide area environment ATM will become the optimal solution to transport voice and video with various, individually negotiated, Quality of Service (QoS). In cluster application the basic ATM traffic assumptions may be invalidated. In clusters the flow of data is much more bursty and the number of node is not large enough to produce a continuous data stream.

Max bandwidth 155 Mb/s to 2.4 Gb/s (on fiber), 25 Mb/s to 51 Mb/s (on UTP).

Minimum latency : $< 40\mu$s.

Cost 3.000 $ per node.

FDDI FDDI is a token-passing ring system providing a total 100 Mb/s system bandwidth that is shared among all nodes. The best application for FDDI is as backbone for 10 Mb/s networking. As Fast Ethernet gains prevalence and Gigabit links are used for connecting Fast Ethernet switches, FDDI will not play any role in high performance NOW clusters.

Ethernet (standard) 10 Mbits/s Ethernet has been during the last ten years the standard network technology. The standard configuration (hub) can serve up to 100 machines with minimal cost but is not appropriate for fine grain parallel processing requiring intensive data exchange. The use of multiple network connections with load sharing is a cost-effective way to increase bandwidth

Max bandwidth 10 Mb/s

Minimum latency : 60 μs.

Cost 100 $ per node.

Fast Ethernet This solution is the evolution of the standard 10 Mb/s Ethernet and in the last few years appeared to be the best choice for any low cost PC or workstation cluster application. Moreover the cost of the hardware is dropping every day. Switched hubs can provide full speed for each node simultaneously and are less expensive than ATM switches taking advantage of the expected lower switching frequency. [36].

Max bandwidth 100 Mb/s

Minimum latency : $< 10\mu$s.

Cost 200 $ per node.

Gigabit Ethernet This solution is the evolution of the 100 Mb/s Ethernet and it is intended to be cheap, mass-market, computer network support for IP. The main role of Gigabit connection is now the implementation of backbone structure connecting FastEthernet subnetwork. A significant number of products are available.

Max bandwidth 1000 Mb/s

Minimum latency : 10μs.

Cost 2.500 $ per node.:

The costs of this technology are expected to drop substantially once the new copper-based standard will be approved and consolidated. In a couple of years Gigabit Ethernet is assumed to be able to take over the role that is currently plaied by FastEthernet.

2.1.2 High performance interconnect technology

Several research projects carried out in the last few years yielded new technologies (wormhole routing, etc.) that was found more appropriate than standard LAN technology for high-speed interconnects. We review some of the most interesting products currently available that implement such new concepts.

Myrinet Myrinet is a Gigabit-per-second network based on the technology used for packet communication and switching within concurrent and parallel supercomputers. Characteristics that distinguish Myrinet from other networks include:

- Full-duplex 1.28+1.28 Gigabit/second links, switch ports, and interface ports.
- Flow control and error control on every link.
- Low-latency, cut-through switches.
- Host interfaces that execute a control program to handle direct interaction with host processes; sending, receiving, and buffering packets; and network mapping and monitoring.

Currently Myrinet is offered by a single vendor

Max bandwidth 1280 Mb/s (full duplex)

Minimum latency : $< 5\mu$s.

Cost 1.800 $ per node.

SCI SCI is the equivalent of a Processor-Memory-I/O bus and a Local Area Network, combined and made parallel that allows building large systems out of many inexpensive mass-produced building blocks. SCI originated as an offshoot from the IEEE Standard Futurebus+ project in 1988, when it became clear that coming microprocessors were soon going to be too fast for any bus. As a result backplane buses have been replaced by unidirectional point to point links [32]. Each node has an input and output link connected to the SCI ring or crossbar.

Max bandwidth 8000 Mb/s

Minimum latency : $< 10\mu$s.

VIA - Virtual Interface Architecture The VIA specification standardizes an interface for high performance network technology allowing to cluster servers. The standard has been promoted by Compaq, Intel and Microsoft. and the first specification can be found at URL: `http://www.viarch.org/` The specification of VIA essentially derives from the experience of efficient implementation of the U-Net prototype at Cornell University. So far, no hardware device conforming to the VI Architecture standard is available, however a software emolation for 100 Mb/s and 1000 Mb/s Ethernet NICs is currently under development (M-VIA, `http://www.nersc.gov/research/FTG/via/`) for the Linux environment.

2.2 Low-level architecture of NOW/cluster message passing systems

Traditional operating system support for communications in local area networks is usually considered not appropriate for NOW/cluster computing, due to extremely poor user-level performance as compared to the potentials of hardware technology.

Various attempts to address the inefficiency of standard inter-process low-level communication mechanisms in NOW platforms were carried out in the last few years. In this chapter we examine the low-level architecture of a number of NOW message passing systems. Our discussion is mainly concerned with architectural as well as implementation choices addressing performance issues. In some cases the low-level programming interfaces are briefly outlined.

Given a message of size S, it is easy to compute the *communication delay* $D(S)$ as half the message round-trip time measured by some form of the "ping-pong" microbenchmark. One of the most complete performance measurement of a messaging system is the *throughput curve*, namely the complete graph of the throughput $S/D(S)$ as a function of S. Often only a partial view of the throughput curve is given, by reporting the so called *one-way latency*, which we define as $D(S_m)$ where S_m is the smallest message allowed by the communication system, and the *maximum throughput*. If the maximum throughput is achieved with the longest message allowed by the communication system, the term *asymptotic bandwidth* is often used as a synonym of "maximum throughput".

2.2.1 Common message passing performance optimizations

Most NOW messaging systems are committed to efficiency, at least to a certain degree. Although following different architectural approaches, most of them share a number of common performance-oriented features which we briefly outline below.

Multiple commodity network devices exploited in parallel. This is the most straighforward way to increase the aggregate communication bandwidth of the platform.

Simplified LAN-wide host naming. Indeed addressing conventions in LAN might be much simpler than in WAN.

Simplified communication protocols. Long protocol functions are time-consuming, and their low degree of locality in the access to huge data structures generates a large number of cache misses and pollutes the working sets of user processes, thus leading to a poor exploitation of cache with degradation of overall system performance.

Minimal number of intermediate copies of messages. Making temporary intermediate copies of information along the communication path during the execution of complex protocols is a time-consuming task and pollutes the working sets of user processes as well. A frequent (partial) answer to this challenge is remapping the kernel-level intermediate buffers into user memory space, so that user processes no longer have to issue memory transfers from kernel to user space.

Avoid system calls for communication. Invoking an OS system call is often considered too much a time-consuming task. Therefore one challenge is to minimize the need of system calls in the communication path. Remapping the kernel-level intermediate communication buffers into user memory space so that user processes no longer have to invoke system calls to read/write from/to such buffers is a frequent solution which raises protection issues, as we will see.

Light-weight system calls for communication. As an alternative to eliminate the need of system calls, so called light-weight system calls are often implemented, which save only a subset of CPU registers and do not invoke the scheduler upon return.

Fast interrupt path. The code path to the interrupt handler of the network device driver is optimized in order to reduce interrupt latency in interrupt-driven receives.

Polling the network device. Polling the network for incoming messages allows to save the overhead of interrupt launch and service. This is possible in case of synchronous, explicit message receive, and implies a proper structure of the computational model behind the distributed application.
From a software-oriented standpoint, existing high performance message passing systems running on NOW platforms can be grouped into three main families, namely:

- the *standard interface* approach;
- the *user-level* approach;
- the *efficient OS support* approach.

A relatively small number of message passing systems have been designed and developed to run on either specialized or custom fast interconnects. They will be discussed in a separate section.

2.2.2 The *standard interface* approach

In the *standard interface* approach the main goal besides efficiency is to provide applications with an industry standard interface to the communication layer. Retaining an industry standard communication interface allows portability and reuse of existing applications and libraries developed for that interface.

Usually this approach does not force any major modifications to the existing OS; rather, the new communication system is simply added as an extension of the OS itself (e.g. a custom device driver, a Linux kernel module, or a user-level programming library).

A drawback of the standard interface approach is that some optimizations in the underlying communication layer could be hampered by the choice of an industry standard, like BSD sockets for instance, which was not originally conceived to address any performance issues.

Fast Sockets. Fast Sockets [53] is an implementation of TCP sockets atop an Active Message communication layer. The underlying Active Message layer provides good performance, whereas the upper socket interface, implemented entirely as a user-level library, provides the messaging system with industry standard characteristics. The protocol reverts to plain TCP/IP when the LAN boundary is crossed, e.g. through a gateway. In case of forking processes Fast Sockets do not support the plain semantics of sockets, according to which socket descriptors which are open at fork time are shared with child processes. Fast Sockets have been reported not to have an efficient connection phase. Measurements carried out on a pair of UltraSPARC 1's interconnected by Myrinet and running the Solaris OS have shown poor performance (57.8 μs latency, 32.9 MByte/s asymptotyc bandwidth) compared to Myrinet raw communication performance.

Parma Parallel Machine ($PARMA^2$). The $PARMA^2$ project [40] is aimed at reducing communication overhead in a cluster of PCs connected by Fast Ethernet and running the Linux OS, by eliminating flow control and packet acknowledge from Linux TCP/IP. The resulting communication protocol, called PaRma Protocol (PRP), is presented to applications through the standard BSD socket interface. The MPICH [28] version of MPI was then able to run on PRP with only minor modifications. At the lowest level, PRP preserves the Linux interface to the NIC driver. No optimization was carried out at the level of NIC driver. PRP exhibits 74 μs one-way latency and 6.6 MByte/s asymptotic bandwidth, that is 37% improvement over Linux TCP/IP latency with same maximum throughput. However the absolute communication performance of PRP is still far from the Fast Ethernet raw performance.

Beowulf. The Beowulf project [57] runs on a Linux-based cluster of PCs. It follows a conservative approach, namely retaining the standard Unix protocol suites. It improves communication performance by exploiting two or more LANs in parallel, a technique called "channel bonding" which Beowulf supports at the level of Linux modified NIC driver. Of course much better results would have been obtained if the communication software has been optimized, in addition to using more LANs. Besides the standard bus topology, a 2-D mesh topology has been investigated [22]. Each node in the mesh is connected to two distinct Ethernet LANs, namely the "horizontal" and the "vertical". Therefore each node is adjacent to every node in the same row as well as column. Moreover, each node acts as a software router in order to allow non-adjacent nodes to communicate. This way a node A can reach a non-adjacent node B by two distinct disjoint paths. The "channel bonding" technique is then applied to allow node A to communicate with node B using both such two independent network paths in parallel, with two intermediate nodes engaged in parallel for routing. However a parallel disk I/O test conducted on such a 2-D mesh topology has shown throughput improvement over bus topology only in case of small workloads.

2.2.3 The *user-level* approach

The *user-level* approach to fast communication in a LAN environment aims at improving performance by minimizing the OS involvement in the communication path in order to obtain a closer integration between the application and the communication device. More precisely, access to the communication buffers of the network interface is granted without invoking any system calls. Any communication layer as well as standard interface is implemented as a user-level programming library. This approach is rooted in the idea of F-Bufs [20], which was followed also in the Exokernel OS architecture [21]. Often (but not always) a primary concern besides efficiency is to provide such a "direct access" to the communication device without violating the OS protection model.

In all the existing instances of such approach, a non-standard programming interface is presented to applications. The choice of the programming interface itself is often determined by performance considerations.

In order for such approach not to compromise protection in the access to the communication devices, either a single-user network access or strict gang scheduling are usually required. A third alternative is to leverage programmable communication devices which can run the necessary support for device multiplexing in place of the OS kernel. All three alternatives have their own drawbacks:

- single-user network access appears to be an unreasonable restriction in a modern processing environment;
- gang scheduling appears inefficient and requires major interventions at the level of OS scheduler;
- moving multiplexing support from the OS kernel to the device is unfea-

sible with commodity low-cost components which usually are not programmable.

In the user-level approach, modifications to the OS kernel may range from simple addition of custom device drivers to deep interventions at the scheduler level, depending on the degree of protection to be preserved and the extent to which multi-user access to the communication device is to be allowed.

Illinois Fast Messages (FM). FM [51] is an Active Message-like messaging systems originally developed for a Myrinet-connected cluster of SPARCStations and for the Cray T3D, and recently re-implemented on a Myrinet network of PPro PCs running the Linux OS as well as Microsoft Windows NT. Currently, FM allows only single-user access to the network (a multi-user release running on NT has not been documented yet), despite the Myrinet adapter coming equipped with the LANai co-processor which could have been programmed to multiplex the device among many user processes. FM provides reliable delivery with flow control and retransmission. The FM communication performance is quite far from the raw performance of the Myrinet interconnect, especially in terms of asymptotic bandwidth. According to [33], on SPARCStations 5 FM delivers only 17.5 MByte/s asymptotic bandwidth, due to the slow SPARCStation SBus, with one-way user-to-user latency as high as 21.5 μs. FM delivers much better performance on PPro clusters (12.5 μs latency, 49 MByte/s asymptotic bandwidth), thanks to the much faster PCI bus.

With FM, a temporary receive buffer is preliminarly allocated in kernel memory space. Network access is granted to one single user process per workstation by issuing an initialization function which re-maps such kernel temporary receive buffer into user memory space via the `mmap()` Unix system call. This way the user process can directly access the kernel receive buffer by actually accessing the "user space image" of the buffer itself. Therefore no system call must be issued to receive incoming messages.

To send a message, the user process must simply issue the `FM_send(dest, handler,buf, buf_size)` function, which fragment and copies the message body from the source buffer `buf` directly into the on-board send queue of the Myrinet adapter by Programmed I/O without invoking any system call. The outgoing data are detected by the LANai processor and immediately transmitted on the network.

On message arrival, the LANai processor of the receiver side moves incoming data to the kernel temporary receive buffer via DMA. The destination user process can receive incoming messages by issuing the `FM_extract()` function, which only triggers the receiver handler associated with each complete message. Transferring incoming messages from the "user space image" of the kernel receive buffer to the final destination buffer does not require any system call, and is performed by the receiver handler associated with the message itself.

U-Net on ATM. The original prototype of U-Net [61] ran on an ATM network of SPARCstations with SunOS 4.1.3. Actually such first implementation was a kernel-level emulation of the U-Net user-level principle, given that the old Fore Systems SBA-100 ATM adapter had no programmable on-board co-processor. With the second U-Net implementation, a 25 MHz Intel i960 processor located on the newer Fore Systems SBA-200 ATM adapter executes proper firmware in order to multiplex a pre-defined number of virtual "endpoints" over the NIC itself, with no OS involvement. Basically an endpoint is a kernel memory buffer plus a send queue and a receive queue for host-NIC synchronization. Different of FM, the U-Net endpoints are used as intermediate storage also in the send phase. A user processes can get attached to an endpoint through a dedicated system call, which re-maps the endpoint into the user process' memory space. After a user process is attached to one or more endpoints, the process may exchange data and synchronize with the NIC through the "user-level images" of its own endpoints.

To send a message, the sender process copies the message into an endpoint buffer and pushes a send descriptor into the endpoint send queue. The on-board co-processor of the SBA-200 adapter continuously polls endpoint send queues for new messages to be transmitted, and performs message fragmentation if necessary. On each packet arrival on the other side, the receiver co-processor interpretes the header and moves the packet to the correct destination endpoint, then pushes a receive descriptors in the corresponding receive queue. The destination process can detect packet arrivals by inspecting the receive queues of its own endpoints. Message reassembly is performed by the receiver process itself, if needed.

Therefore, with the U-Net architecture the OS support to device multiplexing runs on the communication co-processor and no system call is needed to actually perform send/receive operations. The U-Net programming interface is very low level compared to other user-level communication systems. Indeed U-Net should not be considered a true messaging system, but rather a low-overhead approach to protected multiplexing of network devices in a multitasking environment with protection boundaries.

U-Net saturates the available 15 MByte/s ATM bandwidth. However the 44.5 μs endpoint-to-endpoint one-way latency [65] is not impressive, despite the very low abstraction level of the programming interface.

Emulation of Active Messages [62] as well as TCP interfaces were implemented in U-Net as user-level programming libraries. A kernel-level emulation of U-Net for Fast Ethernet has been developed [65].

Other projects.

HPAM [44] is one of the first user-level messaging systems. It is an Active Message-like system running on a FDDI-connected network of HP 9000/735 workstations. Connecting the FDDI interface to the high-speed graphic bus of the HP 9000/735 workstation instead of using the I/O bus leads to low latency and good (12 MByte/s) asymptotic bandwidth. Protected direct access to the network is granted to each process in mutual exclusion: an OS daemon schedules the processes to the network, and other major modifications to the OS allow the delivery of messages to non-scheduled processes. This way gang scheduling is avoided, but the OS plays a non-negligible role in the communication path,

at least in the general case of non-scheduled destination processes. Like FM, HPAM provides reliable delivery with flow control and retransmission.
MU-Net [47] yields protected user-level access to Myrinet interconnection networks under Solaris 2.5. Its main feature is that multiple user processes in the same workstation are allowed to access the network simultaneously and in a protected way, thanks to efficient device multiplexing code run by the LANai processor on the Myrinet adapter. Similar to U-Net, MU-Net uses kernel-allocated communication endpoints re-mapped into user memory space of owning processes. On SUN Ultra 1 workstations a one-way latency time of 20 μs for a 8 byte message with as low as 18 MByte/s maximum throughput (probably due to the slow SBus) is reported.
The Basic Interface for Parallelism (BIP) [52] is a simple programming interface with send and receive operations which has been implemented on a Myrinet cluster of PPro 200 MHz PCs running Linux. It has been developed to the purpose of efficiently supporting higher-level parallel programming interfaces like MPI. Both protection and multi-user network access have been sacrificed for the sake of maximum performance. Indeed BIP exhibits very low one-way user-to-user latency (less than 5 μs) and very high asymptotic bandwidth (more than 125 MByte/s, 98% of the maximum Myrinet throughput). Porting the TCP/IP stack atop BIP provides an impressive demonstration of the poor efficiency of industry-standard protocol layers on fast interconnects (one-way latency 70 μs, asymptotic bandwidth 35 MByte/s).

2.2.4 The *efficient OS support* approach

In the *efficient OS support* approach, the messaging system is supported by the OS kernel with a small set of flexible and efficient low-level communication mechanisms and by a simplified communication protocols carefully designed according to a performance-oriented approach. Here challenges like choosing the right mechanisms and degree of virtualization as well as implementing such mechanisms efficiently are more important than choosing the level at which the communication layer is to be placed in the overall OS architecture.

The efficient OS support approach fits nicely into the architecture of modern OSs, providing protected access to the communication devices with no need to limit or modify its multi-users, multitasking features. With appropriate programming techniques it is even possible not to compromise at all the match between the industry standard and the modified OS kernel.

U-Net on Fast Ethernet. In the Fast Ethernet version of U-Net [65], designed for a 100base-T network of Pentium PCs running Linux, the low-cost commodity NIC has no programmable on-board processor. The host CPU itself multiplexes the endpoints over the NIC by means of proper OS kernel support. For instance, the send operation requires invoking a light-weight system call to warn the NIC driver about the presence of a fresh outgoing message in an endpoint. For this reason we do not consider such a communication system as a user-level one, but only as a kernel-level emulation of the original U-Net concept [61] yielding still good communication performance (30 μs one-way latency, 12.1

MByte/s asymptotic bandwidth) but with the drawback of presenting the very raw U-Net programming interface.

Genoa Active Message MAchine (GAMMA). Following the efficient OS support principle, with GAMMA [13, 15] the Linux kernel has been enhanced with a native Active Message-like communication layer implemented as a small set of additional light-weight system calls and a custom NIC driver with a fast interrupt path. Most of the communication layer is thus embedded in the Linux kernel, the remaining part being placed in a user-level programming library. All the functionalities of the original Linux kernel were left unmodified. All the communication software of GAMMA has been carefully developed according to a performance-oriented approach. The adoption of the Active Message paradigm allowed a "zero copy" protocol, with no intermediate copies of messages along the whole user-to-user communication path. The GAMMA custom device driver allows the NIC to start transmitting a new frame as soon as the first few bytes of a new packet have been uploaded to the NIC's send FIFO queue, and is able to start DMA download of a packet whose physical arrival is still in progress. This forces the communication path to work in a pipelined way yet with very short messages. GAMMA yields very low one-way user-to-user latency (12.7 μs) and high asymptotic bandwidth (12.2 MByte/s, corresponding to 98% of the raw 100base-T bandwidth). Differently from most high performance communication layers running on NOWs, GAMMA yields high communication performance while still offering a fairly flexible and easy-to-use virtualization of the communication hardware in a multi-user, multitasking UNIX environment.

Other projects.
SSAM [60] is one of the first instances of efficient OS support messaging systems working on a NOW platform, besides being the first Active Message communication system running on a NOW, more precisely a 140 Mbit/s ATM network of SPARCstation-20 workstations. Communication performance of such an early prototype was not particularly brilliant.
Another instance is the implementation of Sender-Based protocols described in [58]. With this messaging system, the sender process specifies the location where the message will be stored in the memory space of the destination process. Sender-Based protocols have been reported to exploit about 82% of the raw communication bandwidth and latency below 24 μs on a FDDI-connected cluster of HP730 workstations. In [58] an excellent decrease (44%) in latency obtained with the fast interrupt path optimization is shown.
Pupa [59] is a messaging system providing reliable delivery and flow control. It is implemented as an extension of the FreeBSD OS kernel. The main features of Pupa are a smart buffer management policy and a light-weight communication protocol. However the Pupa communication performance on Fast Ethernet is comparable than Linux TCP/IP sockets.
Net* [54] is a communication system for Fast Ethernet based upon a reliable protocol implemented at kernel level. A user process can be granted network

access by initially allocating kernel-space send and receive buffers and queues which are re-mapped into user-space like with U-Net. Only one user process per node can be granted network access. To send a message, a process copies the message itself into the "user-level image" of a send buffer, then invokes the kernel-level transmission protocol function through a dedicated system call. Such system call is not necessary if another send operation was submitted a short enough time before, since the IRQ launched at the end of each physical transmission is serviced by a routine which tests send queues for new messages to transmit. To receive a message, a process may perform either a busy or a sleeping waiting on the receive queue. Net* appears to share architectural limitations of both the user-level access and the efficient OS support approaches. Indeed no kernel-operated network multiplexing is performed despite the kernel being involved in the protocol execution, and user processes have to explicitly fragment and re-assemble messages longer than the Ethernet MTU. Net* exhibits quite low (30.1 μs) one-way latency, but is comparable to Linux TCP/IP sockets as for asymptotic bandwidth.

2.2.5 Message passing on custom communication devices

A number of projects [19, 30, 38, 7, 41] use either special purpose or proprietary communication devices in order to achieve low latency and high bandwidth communication. The performance characteristics as well as the technological level of some custom/expensive devices are often much better than off-the-shelf ones. The performance superiority is often achieved thanks to a closer integration between the communication device and the host CPU: many such communication devices could be plugged directly on the memory bus and tightly integrated in the memory hierarchy of the workstation. Indeed most of such novel interconnects are targeted to distributed shared memory systems rather than message passing systems. We argue that in the long term such attempts are not likely to spread in the user community, as non-commodity interconnection devices are usually too expensive and do not enjoy the large R&D efforts that only commodity hardware vendors can sustain in the long term. Indeed many vendors of traditional massively parallel platforms have run into enormous difficulties in keeping their product up-to-date with technology, due to the limited market segment that they could meet. Conversely systems exploiting only commodity components are guarenteed to keep up with the fast pace of the evolving hardware technology, as the market segment which they belong to is much broader.

ParaStation. ParaStation [64, 63] is a message passing system based on a custom packet switching interconnect with 2-D toroidal mesh topology reconfigurable to a ring topology. Flow control is provided at link level. Maximum link throughput is 16 MBytes/s. The network interface is PCI-compatible and provides virtual cut-through routing with 250 ns delay. The ParaStation hardware provides good raw communication performance (3.4 μs one-way latency, 15.6 MByte/s maximum throughput). The ParaStation programming interface provides a suite of various low-level abstractions including UNIX sockets [6], implemented at user level and compatible with a multi-user, multi-tasking environment. The current implementation suffers from a serious lack of protection, due to the ParaStation network interface lacking of an on-board processor to run protected network multiplexing code. On Pentium 166 MHz PCs, ParaStation sockets achieve as low as 13.9 μs one-way latency with 11.9 MByte/s maximum throughput. At the lowest level of the ParaStation programming interface, called "rawdata", communication performance is even better (6.4 μs one-way latency, 14.8 MByte/s maximum throughput).

Telegraphos. Telegraphos [41] is a communication hardware system originally designed to support shared memory on workstation clusters by providing efficient primitives like remote memory read, point-to-point as well as multicast remote memory write, and remote atomic operations. It consists of a switch and a custom network interface developed for the TurboChannel I/O bus of the DEC Alpha 3000/300 workstation. All the operations may be invoked by user level with no OS involvement. issuing user-level DMA transfers with no risk of race conditions among multiple user processes required developing an original interaction protocol between the user processes and the network interface [42]. The importance of efficiently supporting message passing model in addition to shared memory has been recently recognized by the Telegraphos team, which added a so called "remote enqueue operation" to the set of hardware-supported Telegraphos remote atomic operations [43]. By issuing a remote enqueue operation, a user process is allowed to atomically push an item into a queue owned by a remote process. This may be used to notify an event like the completion of a remote memory write to the remote process itself. This way a "message send" operation can be easily and efficiently implemented as a remote memory write for the actual message delivery, followed by a remote enqueue to notify the message arrival to the receiver process.

2.3 High-level message passing programming interfaces

For the benefit of parallel program portability a few high-level message passing programmin interfaces have been standardized. These include Parallel Virtual Machine (PVM), Bulk Synchronous Programming Library (BSPLib) and Message Passing Interface (MPI).

2.3.1 The *Parallel Virtual Machine* PVM

PVM is a software package that allows a heterogeneous collection of workstations and supercomputers to function as a single high-performance parallel machine, namely as a single *Virtual Machine* [25]. The PVM project began in 1989 at Oak Ridge National Laboratory. The current version is 3.4 and subsequent releases are expected. PVM is available across a wide range of Unix based computers, as well as VMS, Windows NT, Windows 95, and massive parallel supercomputers such as CCM-5, CS-2 and iPSC/860, Cray T3D, SP2, ...

Central to the design of PVM is that a set (*pool*) of heterogeneous hosts connected by a network appears to the user as a single large parallel computer, namely:

- PVM combines the power of a number of computers;
- PVM may be used to create a virtual computer from multiple supercomputers enabling the solution of previously unsolvable *grand challenge* problems;
- PVM may be used as an educational tool to create a parallel computer from low cost workstations.

The main PVM Principles are summarized:

- The user can completely define and control the host pool:
  - thanks to a user interface (the *PVM console*) the user can select the machines to be used for a given run of the PVM program;
  - the host pool may be dynamically changed by adding/deleting machines/processes during runs;
- Translucent access to hardware:
  - the hardware environment can be viewed as a mere collection of virtual processing elements;
  - at the same time the capabilities of specific machines can be exploited by accordingly selecting the process distribution among the host pool;
- Process-based computation:
  - the unit of parallelism in PVM is a process that is in charged of both computation and communication tasks;
- Explicit message-passing model:
  - tasks co-operate by explicitly sending and receiving messages to and from one another;
  - message size is limited only by the amount of available memory;
- Heterogeneity support:
  - PVM can handle heterogeneous pools of machines and network;
  - different data type are transparently handled by the PVM communication layer, namely messages can contain one or more data types to be exchanged amongst machines having different data representations;
- Multiprocessor support:
  - PVM uses the native message-passing facilities on multi-processors to take advantage of the underlying hardware;
- PVM currently supports C, Fortran, and even C++ languages.

The PVM system is mainly composed of two parts. The first part is a daemon (*pvmd*) that resides on all the machines of the pool. When a user wishes to run a PVM application, he first creates a virtual machine by starting up PVM. The PVM application can then be started on any of the hosts. Multiple users can configure overlapping virtual machines, and each user can execute several PVM applications simultaneously.

All PVM tasks are identified by an integer task identifier (TID). Messages are sent to and received from TIDs. Since TIDs must be unique across the entire virtual machine they are supplied by the local pvmd and are not user chosen. Nevertheless, since there are applications where it is natural to think of a group of tasks, PVM includes the concept of user named groups, allowing to group the processes. In addition any PVM task can join or leave any group at any time.

A very important feature of PVM is fault-tolerance handling. Under the control of the user tasks can register with PVM to be *notified* when the status of the virtual machine changes or when a tasks fail. In this case when a task dies (i.e. for system crashes) receiving tasks will be notified and thus will have the possibility to cope with the problem.

2.3.2 MPI

In 1992 the major vendors of parallel computers as well as researcher of private and public institutions defined the syntax and the semantics of a standard programmer interface devoted to enable the programmers to write fully portable *parallel* code in C or Fortran 77 [23]. This interface is known as *Message Passing Interface* (MPI) and represents now the most used standard for writing portable message passing programs. It is not a new programming language; rather it is a library of subprograms that can be called from C and Fortran 77 programs. It has rapidly received widespread acceptance because it was carefully designed to maximize the performance on a wide variety of systems [31, 50].

The main target of the MPI is to establish an efficient, portable, flexible, and practical standard for message passing. In this standardization effort it is of paramount importance to understand the constraints under which the standardization committee operated:

- define an interface easily portable on the largest range of vendors' platforms;
- allow efficient communications;
- allow the use in heterogeneous environments;
- assume reliable communications;
- C and Fortran 77 bindings;
- the interface should be not too different from similar environments (such as PVM);
- provide a number of extensions to increase the flexibility.

In addition the committee decided to specify only the message passing portion of the standard and to postpone other aspects of parallel programming such as process control to a future standard.

The MPI standard specification rules many aspects of the concurrent programming, such as:

- Point to point communications: communications between two processes;
- Collective communications: communications (or signals) amongst group of processes;
- Processes grouping: how to create specific classes of processes;
- Topologies: a mechanism that allows to establish specific geometries of communications.

Conversely other aspects (partially faced by the outcoming new standard, MPI2 [46]) are not included in the standard, namely those operations that were identified as too dependent on specific hardware or operating systems, such as:

- shared memory operations;
- active messages;
- interrupt driven messages;
- process spawn or management;
- debug;
- thread support;
- parallel I/O;
- real-time support.

Point to Point Communications The simplest communication mechanisms defined by the MPI standard are communications between a pair of processes, namely Point to Point (PtP) communications. MPI defines a set of send and receive primitives that allow to transmit *typed* data with a specific *tag*. In fact, since MPI is expected to face also heterogeneous systems, all communications primitives requires a type parameter that is used to perform data conversions as data are sent from one architecture to another. In addition a tag parameter can be used by the receiver to select the message. MPI PtP communications can be *blocking* or *nonblocking* operations. Blocking send blocks until the send buffer can be safely overwritten. Likewise blocking receive does not return until the receive buffer contains the full message. Conversely nonblocking primitives start the requested operation and return the control to the program. Specific primitives must be used for testing the completion of nonblocking operations. Moreover MPI provides a set of *modes* for PtP communications. The basic idea is to enable the programmer to specify the communication semantics of the send operation in order to optimize a program for a specific architecture or for a specific communication protocol, without sacrificing the portability. The allowed PtP communication modes are:

Standard: when using this mode the completion of a blocking send does not imply that the matching receive has started.

Buffered: this mode decouples send and receive operations, the blocking send returns as soon as the outcoming message is copied in a buffer used for the transmission. The user must define and allocate such a buffer.

Synchronous: in this case a send would not complete until the matching receive has begun to receive the data.

Ready: this mode is based on the assumption that a matching receive has been already started. On many systems this postulate could improve the performance of transmission.

Collective Communications Collective communications involve all the processes belonging to a *communicator*. A communicator is a group of processes and specifies the scope of a communication. The MPI standard defines the following collective communications functions that are all blocking operations:

Broadcast: a single process sends the same data to all the processes of a communicator;

Gather: data distributed among all the processes of a communicator are collected by a single process;

Scatter: data belonging to a single process are distributed across the communicator;

Allgather: like a gather followed by a broadcast of the gather output;

Alltoall: like a set of gather in which each process in a communicator receives distinct data;

Reduction: a binary operation that involves data distributed among all the processes of a communicator;

Barrier: a synchronization mechanism across all the processes.

2.3.3 Comparison between PVM and MPI

PVM is centered around the concept of *virtual machine* as a dynamic collection of (potentially heterogeneous) computational resources managed as a single parallel computer. Hence PVM has been developed to be very general and transparent to the user, even sacrificing communication performance for the sake of providing powerful process control functions, dynamic resource management and fault-tolerance mechanisms.

Conversely MPI has been focused on message-passing and tuned for producing high communication performance. Thus MPI provides a larger set of communication functions than PVM [26].

Recently the University of Tennessee and Oak Ridge Laboratory started to work on the PVMPI project. PVMPI is a project to combine the virtual machine features of PVM with the message passing features of MPI. PVMPI allows interoperability of vendor's optimized MPI implementations, i.e., it allows MPI applications to span heterogeneous clusters. PVMPI also provides advanced resource management to MPI applications.

2.4 OOP languages and libraries

In the following, a brief introduction of some of the most interesting OOP languages and libraries for NOW architectures is presented. A large part of the languages and libraries presented are based on the C++ language, moreover, they can be divided in two classes: i) the ones using PVM or MPI to support object interaction; ii) the ones using their own specific set of primitives to support object interaction. Both classes have some advantages and some disadvantages. The use of PVM and MPI allows an easy extension of a language/library for different platform, because it is the PVM/MPI layer that manages heterogeneity, and the possibility of an integration with PVM/MPI programs. However, the introduction of a layer on the top of MPI/PVM causes a loss of performance. The use of a specific set of primitives to support object interaction may complicate the extension of the language/library for different architectures; however it may yield similar performance as compared to PVM/MPI, with the advantage of offering OOP features for the development of programs. The use of C++ or an object-oriented extension of Fortran as the basis for the parallel language/library appears to be a good solution both to promote the re-use of C, C++, and Fortran code and to exploit the good compilers that are available for such languages. The use of Java for a parallel language/library currently does not seem to be appropriate because of severe performance limits. However, expected evolutions of the language and/or new compilers could hopefully improve performances and render this choice more interesting in the near future.

Amelia. The Amelia Vector Template Library (AVTL) [55] provides a data-parallel programming model for C++. It is part of the Excalibur project at The Research Institute for Advanced Computer Science. The library borrows ideas from the Standard Template Library (STL) of C++, providing generic collection classes, algorithms, and combining functions. HPC++ uses a similar approach to data parallelism. AVTL is written in MPI and has been used on various distributed-memory machines.

C++//. C++// C++ parallel [10] is part of the Sloop project at the University of Nice. The Sloop project is also responsible for Eiffel//. C++// is designed within the framework of a MIMD model to provide the benefits of software reusability while maintaining high performance. There are no extensions to the C++ syntax; all functionality is through classes. The classes implement active objects, each with their own thread of control, and passive objects, which may be shared. The C++// system is built on top of PVM and has been ported to a number of Unix-based operating systems.

C4. The Canonical Classes for Concurrency Control (C4) [24] are a set of C++ classes developed at the University of Texas. The classes provide various objects for synchronization and data transmission, implemented on top of Intel's NX and MPI. Not surprisingly, C4 implements a message-passing model. C4 relies heavily on the Data Structures Library in C++ (DS++), developed by the same group.

Charm++. Charm++ [37] is work of the Parallel Programming Laboratory at the University of Illinois. It is designed as an object-oriented machine-independent parallel programming system. Charm++ provides a small number of extensions to C++ that differentiate between parallel objects (chares) and sequential objects. The programming model is "message driven", an idea similar to the use of active objects. Performance results for regular and irregular parallel computations appear promising.

EC++. EUROPA Parallel C++ (EC++) [49] is the work of the EUROPA Working Group on Parallel C++, and it is designed to be the standard framework for parallel C++ programming environments. The goal is to make parallel C++ systems portable across hardware architectures and compiler implementations. The standard does not define extensions to C++, but instead defines a standard structure allowing the programming of various parallel programming models within C++.

Eiffel//. Eiffel Parallel [9] is part of the Sloop project at the University of Nice. The Sloop project is also responsible for C++//. Eiffel// is an extension to the Eiffel programming language designed to permit reusability of parallel and sequential code. The interest of this approach is to formally introduce parallelism within the framework of the Eiffel language, and more generally in object-oriented programming, leading towards formal transformations and parallelization for object-oriented languages.

HPC++. The High-Performance-C++ Consortium (HPC++) [29], lead by the Extreme! Computing group at Indiana University, has the ambitious goal of defining a standard model for parallel computing, based on C++. Parallel computers are modeled as distributed nodes of shared-memory multiprocessors. The HPC++ design has two levels: (1) the parallel library and (2) language extensions. As of this writing, the emphasis is on (1). The effort includes the definition of directives, parallel versions of various sequential libraries, and a set of analysis tools. The parallel libraries extend the standard template library and include multi-dimensional array classes and distributed active objects.

High-Performance Java. High-Performance Java [5] is a project of the Extreme! Computing group at Indiana University, the same group responsible for pC++/Sage++ and HPC++. The Java work includes the development of an automatically parallelizing byte-code compiler. The compiler transforms sequential loops to parallel loops; it takes sequential Java code and creates threaded code. The threaded code may run in parallel on virtual machines that support concurrent execution.

ICC++. Illinois Concert C++ (ICC++) [12] is part of the Concert project at the University of Illinois. ICC++ is a dialect of C++ designed to support efficient sequential and parallel execution; the programmer indicates the potential for concurrency, and the system optimizes parallel granularity for the underlying machine. Both task-parallelism and data-parallelsim are supported.

KeLP. The Kernel Lattice Parallelism (KeLP) programming system [35] is the successor of LPARX, both developed at the University of California, San Diego.

KeLP is a software tool for efficiently implementing multi-level and adaptive finite-difference methods on distributed-memory MIMD architectures. It is a domain-specific C++ class library that supports dynamic irregular data structures organized into blocks and levels. KeLP's abstractions represent data decompositions and communication patterns as first-class objects. Applications are portable across a diversity of architectures.

Mentat. The Mentat system [27], developed at the University of Virginia, includes the Mentat Programming Language (MPL) and the Mentat run-time system. The MPL is based on C++, and parallelism is modeled by special classes that provide the granule of parallel computation. The programmer controls granularity and partitioning, while data and control dependences are automatically determined by the compiler and run-time system. The run-time system supports a data-driven message-passing model. The Mentat research covers a wide range of parallel-programming topics, including combining data and task parallelism.

MPC++. Massively Parallel, Message Passing, Metalevel Processing C++ (MPC++) [34] is a project of the Real-World Computing Partnership's Massively Parallel Software Laboratory in Japan. OCore comes from the same lab. Version 2 of MPC++ is designed in two levels; level 0 is a template library that implements concurrent threads, and level 1 specifies a "metalevel architecture" and application-specific extensions. The metalevel architecture is a compromise between language extensions and libraries; the metalevel language can be extended in metalevel object libraries (MOL's), and the metalevel language can be used for programming. The resulting code can be optimized, not merely linked to a library.

Object-Oriented Fortran. Object-Oriented Fortran [56] is a dialect of FORTRAN 77 developed at the Engineering Research Center at Mississippi State University. Parallelism is modeled through distributed objects, and objects communicate through asynchronous method invocations. The objects define the granularity of parallelism.

OOMPI. The Object-Oriented Message-Passing Interface (OOMPI) [39], developed at the University of Notre Dame, is a class-library specification intended to encapsulate the functionality of MPI. The full capabilities of MPI-1 are implemented in communicator, message, and data objects.

Orca. Orca [3] is a unique programming language developed at the Vrije Universiteit in the Netherlands. It is based on a shared-data-object model; objects represent communication, via distributed shared memory, instead of representing computation, as in active objects. To implement shared objects efficiently on a distributed-memory machine, the Orca compiler generates regular expressions describing how shared objects are accessed. The run-time system uses this information together with run-time statistics to decide which objects to replicate and where to store nonreplicated objects. Orca's shared objects effectively implement task-parallelism. Current work has included adding data-parallel extensions to the Orca language.

Para++. Para++ [18] is a C++ binding developed at l'Institut National de Recherche en Informatique et en Automatique. The aim of Para++ is to provide C++ bindings to use any message passing library. With it, the use of Message Passing libraries is simplified and more attractive, without significant performances loss. To ensure portability, Para++ is fully implemented with PVM and MPI, so that you can run the same Para++ program on both message-passing libraries.

pC++/Sage++. pC++/Sage++ [8] is a predecessor of HPC++ at Indiana University. pC++ is a set of extensions to C++ that allow data-parallel programming similar to HPF. Sage++ is an object-oriented compiler preprocessor toolkit. A preprocessor written using Sage++ translates pC++ to C++ for execution on a number of parallel machines. The system includes Tuning and Analysis Utilities (TAU), a visual programming and performance-analysis environment, also written using Sage++.

PETSc. The Portable, Extensible Toolkit for Scientific computation (PETSc) [4] is a toolbox for the parallel, numerical solution of elliptic and related partial differential equations using finite element, finite difference or related discretization techniques. PETSc has been developed at Argonne National Lab, and the parallel portions run atop MPI. The developers have made the unique choice of ANSI C for the implementation of their thoroughly object-oriented design. PETSc 2.0 is coded in ANSI C using data encapsulation (data hiding), data bstraction, polymorphism, and some inheritance in order to capture the benefits of object-oriented programming. C was chosen as the library programming language, rather than C++, to ensure maximum code portability, maintainability, and simplicity.

UC++. UC++ [48] is a project at University College London to implement active objects in C++. The first attempt involved extensions to C++, but the current system, UC++-2, avoids language extensions. Instead, it uses the "level 0" interface of EC++, implemented in PVM, MPI, or pthreads.

Chapter 3

Survey of European Activities

The DISCO project started during the conclusion of the Fourth Framework and the detailed definition of the Fifth Framework. An analysis of the goals, the methodologies adopted and the results obtained in the particular field of Cluster Computing as a result of actions supported within the Fourth Framework and an assessment of the possible continuation during the Fifth Framework are therefore quite appropriate in this context to start our survey. In the rest of the chapter we will present s short overview of the hardware and software technology that is currently under development in research and industrial contexts. Finally, we shall try to summarize some of the projects supported by the EU and assess their effectiveness in terms of acquisition of Cluster Technology by end users.

3.1 The VI and V Frameworks

Browsing through the documentation of ESPRIT HPCN IV Framework programme we found the following declarations of intent.

From `http://www.cordis.lu/esprit/src/hpcnhome.htm`

> Statement of intent.
>
> The objective of the High-Performance Computing and Networking (HPCN) domain is to help all sectors of industry exploit the opportunities offered by advanced computing and networking systems to add higher levels of intelligence, reach larger throughputs or ensure shorter response times in their products, processes or services. Detailed objectives in the ESPRIT HPCN workprogramme
>
> Goals for SME.
>
> Preparatory, Support and Transfer Activities: The objectives are to complement R&D projects to enhance the take-up of HPCN solutions in industry and in particular in SME's. Activities include awareness creation, assessment of solutions, demonstration of successful applications and establishment of best practice industry. The activities are coordinated by the HPCN TTN Network started in

March 1997. It consists of 21 Technology Transfer Nodes across Europe with 181 funded activities.

From `http://www.cordis.lu/esprit/src/domsix.htm`

> ...
>
> All activities must be driven by user requirements and oriented towards development of concrete applications and later marketing of added-value in products or services. This applies to all levels of development - from the applications themselves, through systems and sub-systems, to architectures and platforms.
>
> An integral part of the work is the dovetailing of high performance computing and networking into traditional technologies and the transparent integration of new applications within existing infrastructures, working practices and mindsets of potential adopters.
>
> All aspects of industrial activity, in both the goods and services sectors, are concerned; however, new types of applications and the involvement of non-traditional and emerging markets are encouraged. A priority is to establish the necessary vertical relationships in the value-added chain between user and supplier.
>
> The domain addresses "HPCN at large". Applications may exploit parallel systems and distributed and heterogeneous infrastructures, including high performance workstation or PC clusters and heterogeneous architectures with multiple processors. The use of standard, "off-the-shelf" building blocks enables application builders to concentrate their efforts on delivering added-value to the user.
>
> Advanced networking services have become an integral part of these systems and infrastructures, enabling, not only applications which allow sharing and provide interactive use of remote resources, but also applications which support interaction between concurrent activities in geographically dispersed locations.

We can notice that above there is a slight mention to PC cluster but there is a total absence on the subject related to the market potential of "freeware." The concept does not even seem to be considered, although DISCO claimed expressely to address specifically that solution. Therefore we could infere that DISCO is already a step forward in trying to explore this very promising field.

There are also interesting statements made in Area 5, among which task 6.32 quoted below:

> Development and Execution Environments - area 5
>
> ...
>
> 6.32 Deployment and Execution Environments. This covers work on resource management including performance analysis and monitoring, as well as software for communication layers. Priority will be

> given to techniques for the deployment of commercial or technical applications on clusters and distributed heterogeneous systems.

Moreover, under the heading "Preparatory, support and transfer activities" we find the chapter "Demonstration and Best Practice Actions" containing the following statement concerning activity 2:

> Objectives
>
> To demonstrate the use of HPCN technologies and advanced networking services. The target is to bridge the gap between suppliers and potential users and to lower the risk of early adopters.
>
> ...

This last objective seems to fit pretty well our conclusions concerning the need to develop administration software tools to facilitate the scalability of clusters and reduce administration and overheads "of early adopters."

As far as the upcoming Fifth Framework is concerned, we can read the following statements in the official documentation.

From `http://www.cordis.lu/fifth/src/prop2.htm`

> ...
>
> Creating a user-friendly Information Society:
>
> - Systems and services for the citizen
> - New methods of work and electronic commerce
> - Multimedia content and tools
> - Essential technologies and infrastructures

Related to the IV chapter "Essential technologies and infrastructures" we can read from `http://www.cordis.lu/fifth/src/305b-e-6.htm#key4`:

> Objectives and RTD priorities
>
> The aim of this work is to promote excellence in the technologies which are crucial to the information society, to accelerate their take-up and broaden their fields of application. The work will address the convergence of information processing, communications and networking technologies and infrastructures. The focus will be on technologies and infrastructures common to several applications, while those specific to one application only would be addressed in the context of that application in other parts of the Framework Programme.
>
> ...

And finally :

> RTD priorities: software and systems engineering: work will focus on dependable, survivable and scaleable systems and address the reduction of the development cycle and costs; the use and further development, as required, of reliable methods and tools will be a central issue; the integration of custom and off-the-shelf components into systems will be a key aspect; service engineering will address the integration of heterogeneous platforms and networks and the increasing complexity and sophistication of the new services and their creation and provision - the objective will be to develop technologies and tools for rapid, customised and cost-effective service creation, deployment, provision and management that provide for an open information and communications service infrastructure, with the necessary reliability, security and quality of service; software technologies work will foster knowledge-based methods and tools, which increase the usability as well as the capability of systems and the intelligence in the network, this includes the timely collection, production, dissemination and communication of high-quality information (including statistical and management information); take-up: best practice and other take-up measures will be a priority; work will be complemented by the validation and assessment of technologies, systems and services in field trials, and by concertation measures to contribute to standards.

We could not agree more on such ambitious objectives, especially if the term "off-the-shelf components" refer also to software components as well as hardware components. It is needless to say that off-the-shelf software can only be integrated with custom components or extensions if it is reliable, widely available, conforming to accepted standards, and free from patent and or license restrictions. Cluster Computing can fit very well into this general framework and provide the basis for a tangible presence of European research and industrial entities in the global market that is currently developing arond this technology. We find it particularly appropriate that the focus of the incoming Fifth Framework will return on basic "technologies and infrastructures common to several applications" after (in our opinion unfortunate) emphasis given by the Fourth Framework to the development of individual applications, often of limited general interest besides the one of the partners involved in the supported project.

3.2 Status of technology in Europe

It is no mistery that Europe is far behind North America and Japan in general with respect to the development and adoption of advanced computer technology, and Cluster Computing is no exception. If this gap might be justified in terms of scale economy and labour costs considerations as far as hardware is concerned, the same justifications apply to a much lesser degree in case of software. Apart from sporadic exceptions (that are mainly due to the intelligence and ability of individual researchers and/or organizations rather than a real support from the environment) most of the hardware and software components currently used in Cluster Computing are imported from the USA, Canada, and the Far East.

3.2.1 Basic research projects

Academic research is probably the part of European scientific and technological development infrastructure that is best keeping the pace of international developments in the field of Cluster Computing. Research groups working in most countries of the EU are currently undertaking software as well as (interconnection) hardware oriented efforts. Their scientific production is internationally recognized by means of publications in the proceedings of international conferences and journals that are quite comparable to analogous work carried out in North America. Among the numerous projects aiming at producing improved software support for message passing interconnections (that is one of the main research topics in the field) we shall briefly outline two projects developed in France and Italy during the last two or three years, and that achieved better results than analogous efforts originated in the most prestigious US Universities (the Berkeley NOW, the Illinois Fast Messages, and Cornell's U-Net).

BIP

The Basic Interface for Parallelism (BIP) project was undertaken by French researchers of the Ecole Normale Superieur de Lyon and of INRIA (ReMap project). It defines a new, simplified programming interface for message passing in a Cluster environment. The hardware interconnect is 1.2Gb/s Myrinet (from Myricom, USA), a whormhole routing network characterized by extremely low switching latency and potentially very high bandwidth. The careful optimization of the software device driver allowed this group to obtain much lower overall message latency and much higher actual bandwidth than prototypes developed at UC Berkely and at University of Illinois at Urbana Champain, based on exactly the same hardware platform. The MPICH standard programming environment has been ported over BIP, allowing user applications to take full advantage of the improved messaging systems without any need for application source code modifications. Several "Beowulf class" clusters deployed in North American Universities in addition to European research centers are currently preferring the BIP/MPI software to Illinois' FM to efficiently exploit Myrinet based clusters. The current limitations of BIP that reduce its diffusion are due to the Myrinet hardware technology (relatively high cost for the interconnection as compared to the computational nodes, limited length of the wires, saturation of the 32bit 33MHz PCI bus, etc.) rather than to software limitations.

GAMMA

Differently from BIP, the Genoa Active Message MAchine (GAMMA) project focused on the adoption of lowest cost 100Mb/s Fast Ethernet. This choice was forced by circumstances, as the project was developed exploiting already available hardware and without special funding for its development. This same characteristics, however, makes the prototype ideal for use in cases in which no additional hardware cost is desired besides the processing nodes themselves.

GAMMA implements a "port-oriented" version of Active Messages with the use of per-process bidirectional *communication ports*, in order to support MIMD as well as SIMD parallel programming. A GAMMA "port" is a number in the range from zero to 255 meant to denote a bi-directional communication channel. Let P denote a generic GAMMA port. As an output port, P can be bound to a destination, which is a port of a single GAMMA process located on a different processing node in the LAN. Alternatively, a broadcast destination may be specified. As an input port, P can be bound to an application-defined receive buffer (that is an application data structure which is the final destination of messages incoming through port P) and an application-defined receiver handler. The receive buffer must span a contiguous region of virtual memory (typically is is an array), and must be pinned down in RAM. Different of FM, the GAMMA receiver handler are not responsible for message delivery into final receive buffers, as this is performed by the GAMMA driver. An *error handler* may be specified in addition, in order to manage possible communication errors (e.g. missing packets due to overflow). After a port has been bound to a destination and a receive buffer and handler, the port can be used to exchange data with the destination process(es).

To send a message, a GAMMA process must invoke the `gamma_send( out_port, buf,size)` function, which traps to the kernel-level send function of the GAMMA network driver. The GAMMA driver writes a 12 byte header to the NIC's output queue, then copies the message from the user-level buffer `buf` to the NIC's output queue directly (either by Programmed I/O or by DMA). Messages longer than the Ethernet MTU are transferred chunk by chunk with the same technique. The header of each packet contains information useful to re-assemble the whole message in the user-level buffer bound to the destination port of the destination process.

On the receive side, for each incoming packet the GAMMA driver checks for error conditions, reads the destination process and port from the header, computes the address of the user-level buffer bound to the destination port, and moves the payload from the NIC's receive queue to the destination buffer directly by DMA bypassing any temporary kernel buffer. This bypassing is possible since the GAMMA driver has been notified the address of the final destination buffer (which is pinned down in RAM) at the time of port binding.

As soon as the last packet of a message has been delivered, the GAMMA driver itself launches the receiver handler bound to the destination port, regardless of the schedule state of the destination process.

The GAMMA driver receive activity may be triggered asynchronously by the 100base-T interface raising an IRQ, as well as synchronously by a GAMMA process invoking the `gamma_wait( input_port ,n)` function which polls the NIC receive queue for incoming packets until `n` complete messages have been delivered through port `input_port`. Message delivery to port `input_port` is notified to the destination process by the receiver handler issuing the `gamma_signal( input_port)` function.

Besides point-to-point and broadcast basic communication functions, the

GAMMA programming library provides an efficient barrier synchronization primitive implemented in terms of the basic GAMMA communication functions [14]. Work is in progress as for implementing other collective communication primitives and porting the MPICH message passing environment.

As in the case of BIP, GAMMA achieved much better performance results than analogous prototypes developed in the USA (such as Beowulf, U-Net, and M-VIA) based on comparable hardware components. It thus succeded in showing the importance of good, performance-oriented design for critical software components to enable efficient Cluster Computing based on off-the-shelf hardware.

3.2.2 Hardware and software technology production

European industry is not visibly present in the field of hardware technology development for relevant components that can be used in low-cost Cluster Computing. Some research centers of large multi-national companies are located in Europe (IBM, 3Com), but product strategies and economic benefits of the relevant researches carried out there are controlled from the Corporates outside Europe.

Concerning software technology, apart from few exceptions (such as the NAG Group for optimized numerical libraries, Genias Software GmbH for distributed programming and monitoring environments, or Simulog S.A. for discrete event simulation software), the main world-wide recognized development efforts are in the area of Free Software components development. In particular, US Government restrictions on exports of cryptographic software indeed favours European based developers of Free Software security and authentication components. Other significant Free Software European efforts are in the areas of multimedia communications, hypertext, virtual reality and computer graphics.

3.3 Spawning of industrially relevant applications

3.3.1 Some experiences with the LSF applications

Load Sharing Facility (LSF) can be considered as a first step towards the exploitation of clusters in real-life computing environments. Originillay developed as a research prototype at the University of Toronto, Canada, LSF is now a commercial product distributed worldwide. It facilitates the execution of significant applications in a distributed fashion exploiting already existing workstations or PC clusters connected by local area networks. The software is sold together consultancy to teach the final user how to use it in its own environment. We collected some of the "success stories" of European industries that adopted LSF for various applications.

At Pratt & Whitney, a United Technologies company working in the aerospace field, LSF has been adopted for load sharing, batch scheduling, and parallel computing, helping the company use its workstations 24 hours a day. By running computational fluid dynamics (CFD) and other critical design applications in parallel, Pratt& Whitney is able to take large jobs, break them down into a series of small components, and then run each part simultaneously. LSF ensures that the right computing resources are found to meet the demands of each CFD process. This enables the company to run large jobs faster and more efficiently than before, with parallel CFD jobs running nightly on more than 1,500 workstations that would otherwise be idle. While resource utilization is one benefit of using LSF, speed is another. In the design Pratt& Whitney's airfoils, the company uses LSF in tandem with its grid-generation applications. While a typical grid-generation job would take more than an hour to run in batch on an individual supercomputer, LSF enables engineers to break up the job and run it interactively over a parallel computing cluster in less than five minutes. By implementing LSF and clustered computing, Pratt& Whitney has dramatically reduced the cost of engine design and significantly improved product efficiency and quality.

In the automotive industry, Audi uses LSF to manage and prioritize application workloads in a distributed heterogeneous computing environment. The company runs about 35 applications - for structural analysis, flow-mechanics calculations, vehicle dynamics, vibrations of complex structures, as well as for demanding deep-drawing and crash simulations - on a wide range of workstations and servers. Depending on circumstances and conditions, some types of jobs are more important from a business perspective than others. Audi's challenge was to manage its distributed heterogeneous workload so that job priorities could be established according to business objectives.

BP Exploration is part of the British Petroleum Company, one of the world's largest petroleum and petrochemical companies. The primary heavy compute users are the reservoir simulation packages which generate 3D grids that simulate hydrocarbons over time. A cluster of IBM's (580s, 590s, and 3CTs) is provided for reservoir simulation jobs. The users don't log into these machines directly. They submit their jobs from their workstations using a submission script. This script asks for resource requirements such as: how much memory is required for their job and what version of the software is needed. The script then reads their data file and determines what licensed features will be used by their job. Moreover one of the reservoir simulation packages has been improved in order to enable parallel processing using PVM.

Headquartered in Stavanger, Norway, Statoil is a multinational company with branch offices throughout the world. The company is the world's second largest exporter of crude oil. Statoil exploits LSF to make the most of more than 50 powerful CPU servers and 50 high-end workstations spread throughout the company's offices in Norway. The LSF cluster at Statoil consists of IBM SP2s, SGI Origin and HP 8000 servers as a pool of CPU power besides a number of UltraSPARC based workstations and servers supporting the geologist and engineers for interactive usage. The workstations are only available for running simulations on GeoQuest's Eclipse when they are idle or over night. Statoils provides also support for running parallel reservoir simulations. The parallel software uses PVM for parallel processes, so LSF is primarily used to select and reserve hosts/processors for the parallel jobs.

3.3.2 Some NOW applications

Commercial or industrial experiences exploiting network of workstations frequently involve the collaboration between industrial partners and scientific research centers. We will briefly summarize three major European experiences carried out in different domains.

ANSE project The ANSE project by Edinburg Parallel Computing Center (EPCC) and Rolls-Royce has the purpose of extending the functionality of the parallel version of the ANSE software. ANSE is a Computational Fluid Dynamics application developed by Rolls-Royce for use on vector supercomputers. In this and previous projects, EPCC has been responsible for migrating ANSE to distributed memory parallel computers. The underlying message-passing system used in parallel ANSE is PVM. The parallel version of ANSE can be run on networks of workstations, thus increasing the aggregate memory available for solving large problems in-core. The application of parallel high-performance computing to the ANSE software will enable Rolls-Royce to perform much more detailed simulations in shorter times. The ability to run these simulations over an installed network of workstations is also a major benefit in terms of more efficient and cost-effective use of existing resources.

PAEPR Project The PAEPR Project by PAC, Manchester Medical Informatics Group, Centre for Novel Computing and Medical Products Group of Hewlett-Packard aims to support smart semantic access to a large persistent subsumption-based knowledge store, capable of representing asserted and implied intensional concepts. Such an environment is needed for applications such as medicine, which require data models capable of supporting large, variable and complex schemas. The main PAEPR task was to significantly improve the performance of the classification, sanctioning and consistency checking of new implied descriptive concepts by the application of coarse-grained parallelism and intelligent semantic indexing. Two industrial-strength pilots (one based on a specialist parallel architecture and the other using collections of workstations), have been developed. The NOW version is based upon HP workstations.

MASCOT Project The MASCOT Project was set up by Edinburg Parallel Computing Center and AEA Technology Harwell Laboratory. The purpose is the production of a parallel version of the MASCOT probabilistic safety assessment code, targeted at workstation clusters. A client-server based parallel version of MASCOT has been fully implemented. A key feature of MASCOT is the calculation of many independent safety consequences from a given set of initial cases. Throughput is thus extremely important, and applying a parallel client-server based task farm proved to be an excellent solution in achieving this. Performance results measured on clusters of Silicon Graphics and Sun workstations indicated a very significant price/performance improvement. Running parallel MASCOT over four workstations reduced the runtime for 100 test cases from 70 minutes on the Cray Y/MP to 50 minutes. The application of parallelism to MASCOT meant that AEA were able to exchange expensive time on a Cray Y/MP with much more cost-effective time on CAD workstations, improving the price-performance ratio as well as the overall throughput.

3.3.3 Some EU supported projects

There are a number of European projects developed in the last few years having the development of distributed applications based on workstation or PC clusters as their main goal. We will be able to easily recognize throughout the list that the number of projects working on PC clusters is much higher in this research context than when treating purely commercial applications in the first chapter.

This is of course what these projects aim to boost. The purpose of EU funding, in fact, is to deeply expand the knowledge of low-cost, high-performance cluster computing in general and of PC-based networks in particular, especially in industrial domains. The question we are trying to answer is whether the outcome of these projects is actually achieving the desired goals in terms of increased (or, at least, non-decreased) competitiveness of the European industry as compared to the other developed countries.

A survey of European NOW-related projects

The application fields of the technology involved by these projects are the most different. We will produce an exhausting overview of the projects distinguishing them on the basis of the technology adopted, namely:

1. clusters of workstations;
2. clusters of PCs with standard LAN interconnection;
3. clusters equipped with dedicated high speed networking support.

We are describing four projects falling in the category of clusters of workstations based on commodity networks.

PROCLIPSE The PROCLIPSE project is part of the EUROPORT 2 initiative funded by the European Union. The consortium consists of Schlumberger GeoQuest, owners of the ECLIPSE 100 code, Edinburg Parallel Computing Center and the end-users AGIP SpA and Statoil, who are supplying data sets with which to benchmark the code. The objectives of the project are the production of a commercial quality portable and scalable parallel version of the ECLIPSE 100 black oil reservoir simulation code and the evaluation of the code on two classes of HPC systems: clusters of workstations and massively parallel systems. Reservoir simulation is used to design optimum production schemes for oil fields. If an improved production scheme results in a 1% increase in production from a billion-barrel field, then at current prices this will yield an increase in revenue of $160,000,000. A high performance version of ECLIPSE would provide major benefits to designers of production plans: larger, more detailed models of reservoirs can be simulated more quickly, thus enabling better depletion schemes to be developed in a shorter timescale. As an early-adopter of high performance computing technology in its reservoir simulator, GeoQuest would gain an important competitive advantage. A parallel prototype of ECLIPSE 100 using the new solver has been ported to the PVM message passing system, and tested on a Meiko CS-2, an IBM SP2, and on a DEC Alpha workstation cluster.

HP-RETAIL The HP-RETAIL (High Performance Parallel Computing For Medium Sized Retailers) project by Bull SA, Informationsmanagement GmbH Zur Beratung und Realisierung Betrieblicher Informations Systemanwendungen, Angewandten Forschung, Unigro Nederland BV, Julius Meinl AG and Allkauf Sb-Warenhaus is funded within ESPRIT 4. Food and non-food retailing are extremely competitive markets. Large retail groups concentrate their efforts on the exploitation of new consumer groups. They have developed information systems which deliver the necessary marketing information and also shorten the time delay between consumer demand and stock replenishment. High performance parallel computing is ultimately required for data warehouses. Present installations at the large retailing chains are based on massively parallel systems from US companies. Medium sized retailing companies are concerned that, although high performance computing is considered extremely beneficial for efficient consumer response tracking, these large systems are prohibitively expensive and not scalable in cost to fit the needs of a medium sized company. By clustering several symmetric multiprocessor nodes, a high performance data warehouse computer can be built at a reasonable cost. The result is a fairly scalable architecture and a set of pre-configured decision support tools that can help to improve the average trade margins not only overall but also for individual sales outlets, allow benchmarking between different outlets and help to optimize stock replenishment.

SEEDS The SEEDS project (Simulation Environment For The Evaluation Of Distributed Traffic Control Systems) by Alenia S.p.A., Rigel Engineering SA, Sogitec Industries, Sistemi Innovativi Per Il Controllo Del Traffico Aereo, is funded within ESPRIT 4. Purpose of this project is to design and test a distributed interactive Simulation Environement (SE) for the analysis and evaluation of distributed traffic control systems. The SE would provide a general purpose tool kit for the simulation of a wide range of industrial applications of strategic relevance, such as air and maritime traffic control systems. The SE is based on a network of commercial high performance workstations playing the roles of the different actors (traffic controllers, aircraft pilots, vehicle drivers, external world) and of the different sensors (radar, camera, GPS, eye) used in the Advanced Surface Movement Guidance and Control System (A-SMGCS).

COP-214 The COP-214 project (Development Of Computational Methods For Enhanced Analysis And Design Of Industrial Casting Processes In Parallel Computers) by Centro Internacional De Metodos Numericos En Ingenieria, Technische Hochschule of Darmstadt, Institute of Fundamental Technological, University of Constantza, Meconst SA, Biuro Uslug Komputerowych, Ursus Metallurgical Factory, Metalogenia S.A, is funded within INCO. Casting is a very complex process of paramount importance for many industrial sectors such as the automotive and the nautical industries. Although a great deal of progress has been made in the computer modelling of casting processes, there is still a need for a better understand of the physical phenomena occurring in casting problems. This requires investigation in different areas, including numerical modelling of mould filing, prediction of thermal stresses in a solidifying material, micro and macro modelling in order to predict material microstructures, experimental identification of material properties and defect formation. The high computing time required for the solution of 3D industrial casting processes using advanced computational methods such as the finite element method involves days of computation on a fast workstation. The use of parallel computing techniques could allow a cost effective solution of the problem by means of clusters or "farms" of workstations using advanced message passing systems, such as PVM. The resulting parallel codes can be validated in the analysis of solidification in industrial casting problems, for which experimental results will be available from the industrial partners.

We also encountered several projects based on the exploitation of clusters of PC's with commodity networks. These projects are the most similar ones compared to the objectives of the DISCO Tender, as they aim at exploiting extremely low-cost, off-the-shelf technology, without major investments in hardware that are becoming less and less affordable by SMEs as well as Universities needing entry-level access to high-performance computing.

WINPAR The WINPAR project (Windows Based Parallel Computing) by Genias Software GmbH, Simulog SA, Kfki Research Institute For Measurement and Computing Techiques, Gmd-Forschungszentrum Informationstechnik Mbh, Faculdade de Ciencias E Tecnologia Da Universidade De Coimbra, Institute of Applied Computer Science and Information Systems University of Vienna, University of Miskolc, is funded within ESPRIT 4. Within the WINPAR project an integrated software development environment for parallel computing is developed, targeting a new class of parallel computers: personal computers interconnected by local area networks running Windows NT. The technical objectives of WINPAR are to provide a message passing layer including MPI and PVM, to provide a set of tools for code development, simulation, performance prediction, graphical high-level debugging, monitoring and visualisation of parallel applications. The commercial objectives are to offer an affordable parallel development environment for training and education at universities, research organisations, and industries.

FLASH The FLASH project (Hpcn Tools For Enhanced Hydrodynamic Design Of Fast Ships On Parallel Computing Platforms) by Empresa Nacional Bazan de Construcciones Navales Militares, Centro Internacional de Metodos Numericos En Ingenieria, Estaleiros Navais de Viana Do Castelo Sa, Nautatec Ingenieria Naval Diseno de Yates, Structural Polymer Technologies, Bassin D'Essai des Carenes, Inesc/Instituto de Engenharia de Sistemas e Computadores, is funded within ESPRIT 4. The overall goals of the project are:

- to provide the fast ship construction community with a new HPCN system for efficient exploitation in the hydrodynamic design and optimisation of hull shapes and structures using heterogeneous parallel computing platforms composed of clusters of workstations and PCs;
- to demonstrate and assess the developed HPCN system in selected large-scale, multi-physics computations, typical in ship hull design involving coupling between hydrodynamics and structural mechanics problems;
- to promote the HPCN system and disseminate the results of the demonstration actions among naval architecture industries by means of product marketing and organisation of workshops;
- to raise interest for HPCN technology in the European ship construction industry by providing a user friendly parallel application software for both design and optimisation of fast and conventional ferries for passenger and cargo transportation which meets industrial needs.

More specifically, the project focuses on development and integration in a parallel computing environment of existing finite elements based simulation software for CFD (Computational Fluid Dynamics) and CSM (Computational Structural Mechanics) to be used as efficient tools for solving relevant problems of the naval architecture industry.

TRAWE The TRAWE project by Sysdeco Innovation, Sysdeco Profdoc, Formula Opensoft AS, Advanced Travel Technology, Merkantildata Application A/S, is funded within ESPRIT 4. The trial application project belongs to the distributed systems and database technology domains. It is the intention of the partners to demonstrate competitive solutions for each of their market segments, including integrated solutions that cross application borders. This will be achieved by using the following technology incorporated into a workbench environment (offered by Sysdeco as QBE Vision) to prototype some parts of their existing applications:

- interoperability features of a workbench environment for distributed database oriented applications;
- extensible and reusable user interface components;
- object oriented language mechanisms;
- multimedia data types.

The project is expected to result in application prototypes (a travel application, a business administrative application, and a medical application) as well as an improved toolset. The technical objectives are:

- to use object oriented technology to build an application which adapts easily to different set-ups, including connectivity to other applications, without complicated and time consuming modifications;

- to have a distributed application that supports multimedia data types with reusable user interface;
- to have interoperability between PCs with Windows and UNIX computers, using a toolset interfaced to middleware like Orbix-OLE from the Irish company Iona Technologies.

PACAN-D PACAN-D (Parallel Acoustic Analysis Demonstrator) by Parallel Applications Centre, Motor Industry Research Agency (MIRA), Celestion, is funded within ESPRIT. The PACAN-D project aims to show how an existing serial code may be parallelised and applied on a cluster of PCs so as to show real commercial benefits to both a large organisation and an SME. Car manufacturers and loud speaker manufacturers have a common requirement to reduce unwanted acoustic affects from their equipment. In the case of car manufacturers the requirement is to reduce total noise, hence noise pollution. In the case of loud speaker manufacturers it is to improve the quality of the reproduced sound. The use of coupled finite element, boundary element acoustic models would be of benefit to both. However, although commercial software packages exist to solve these models, the solution time and local storage requirements prohibit problems of a realistic size from being run on low cost machines. A small-medium sized enterprise simply does not have the resources to purchase hardware suitable for the task. Even in a larger organisation these resources are often shared with other departments and may not be available for sufficient time to solve the problem. This project aims to show the commercial benefits to both a smaller and a larger organisation of a single parallelised code. As a result of the project, Celestion hopes to have a parallelised code which will run on a network of PCs. These PCs are standard, low cost, Intel based machines which will not be dedicated full-time to analysis work.

Some other EU funded projects are based on NOW clusters exploiting specialized network components instead of commodity ones. Trying to achieve very small communication latencies and high transfer bandwidths is the main purpose of using such specialized hardware components.

SISCI The SISCI project (Standard Software Infrastructure For SCI-Based Parallel Systems) by Dolphin Interconnect Solutions, Technische Universitaet Muenchen, CERN, Kongsberg Informasjonskontroll AS, AEA Technology GmbH, Scali AS, Emu Unterwasserpumpen GmbH, Council for the Central Laboratory of The Research Councils, University of Bergen, is funded within ESPRIT 4. The IEEE/ANSI/ISO standard 1596-1992, Scalable Coherent Interface (SCI), defines a high-speed, scalable, bus-like interconnect to hook-up nodes: processors, memories, peripheral devices, commodity WorkStations (WS) or PCs in a high-performance parallel system. An SCI interconnect enables a standard WS or PC cluster to be operated and conveniently programmed as a shared-memory parallel machine. Given proper application programming environments, SCI is claimed to provide unprecedented flexibility and performance for parallel processing on "commercial off-the-shelf" WS or PC clusters. The SISCI project wants to provide programming environments for SCI-based parallel systems.

The project develops and tests on industrial applications the following formal or de-facto standard software environments: the MPI communication library, the PVM programming library, and a POSIX-compliant thread package that operates across machine boundaries.

DOMITIUS The DOMITIUS project (Demonstrate OMI Technology In Computing Clusters And Storage Systems) by Multistream Systems GmbH, Matra Mhs, Cetia, Grau Storage Systems GmbH & Co, University of Patras, Giga A/S, is funded within ESPRIT 4. DOMITIUS is a consortium of European storage and networking industry leaders combined with leading integrated circuit suppliers that aims to develop, demonstrate, and market products for the emerging high speed Asynchronous Transfer Mode (ATM) networks that will form the core of tomorrow's information superhighway. The partners plan to combine their resources to bring to market all the components of an ATM Adapter card operating at 622 megabits per second constituting a complete ATM endpoint system, including the chip-sets, an adapter board, a device driver, and application program interfaces. The ATM networks systems can easily be integrated with the legacy networks and allows direct attachment of various intelligent peripherals, resulting in low latency and high performance. Low-cost ATM networks wants to be a method to deploy tomorrow's multimedia and audio-video applications. The resulting ATM adapter will also appeal to the increasing desire of computer vendors and end users to link computers into clusters. Clusters of computers, interconnected with low latency, high-speed networks, are becoming an attractive and low-cost alternative to supercomputers and symmetric multi processors. Clustering of high end RISC based workstations, although attractive, has some times been impractical due to lack of low latency network interfaces capable of supporting very high data rates required by the process migration. ATM technology can efficiently create high speed, switched networking infrastructure.

A survey on EU projects outcomes and lessons learnt

In order to evaluate the real impact of some of the EU funded projects, we have submitted a questionnaire to several of the industrial partners involved, after the termination of the projects in which they were involved. From the answers to our questions we can draw some common situations and derive lessons from the presented experiences about NOW applications.

All experiences considered in our overview started from existing applications developed for traditional sequential computer architectures. Only some of them considered other approaches alternatives to NOW clusters, such as using shared memory multiprocessors. The main motivation for adopting the NOW cluster approach was, for all the projects, to increase run time productivity at a lower cost. In the case of the ANSE project however the other motivation was to exploit idle workstation time, for example during overnight hours, for production.

Since in this context we are dealing with projects developed mainly by the scientific community and only adopted for production by the industry end partners, only in few cases problems with administration of NOW clusters have been noticed. In this case the experience of the researchers acquired in this field plays the most important role. On the contrary, that experience is typically not present in an (non IT) industry environment.

For what concerns the level of achieved results, not all the researchers show complete satisfaction. While some experiences achieved excellent results, both in saving costs and for the performance improvements, some of the people that answered are only sufficiently satisfied of their NOW-based experience.

Typically, some people observe that it would be better to optimize the algorithmic aspects of the sequential applications rather than to port to a distributed environment the existing ones. They notice in this case the significant amount of time that was necessary to complete the porting of the sequential applications to a distributed environment. This fact explains why many companies, outside the scientific community, do not start new distributed approaches for their projects: the time needed for the development, implementation and optimisation of distributed applications with traditional instruments on NOW clusters is very high and the costs higher than running for longer time traditional applications developed with lesser human resources. This consideration for a typical company translates in cost savings.

The other aspect to notice is the importance of exploiting user-friendly development frameworks and RAD tools based on distributed environments. RAD technology that nowadays starts to appear typically allows very fast development times for tools and applications. Companies prefer to spend in buying expensive development environments allowing them fast develpment and deployment of applications rather than to spend in exploiting relevant human resources. This produces faster development times and therefore cost savings.

Not all the projects produced applications that the end user (the non IT industry partner) actually exploits in its final production environment. This lack of exploitation was due to two main reasons. Both because the administration of the platform to use was not easy enough and because in some cases the performance achieved was not completely satisfactory. On the other hand, the majority of the projects ended with applications adopted by the industry partner and used for production.

For the research community to start projects that impact also the industry partners, a crucial aspect is to concentrate on products that allow for an easy administration and development of applications. This requirement seems to be even more compelling than application performance for most non IT industrial partners. For many companies the adoption of good distributed development environments and RAD tools, allowing much faster development and deployment times is maybe the most relevant factor that determines the final choice.

Chapter 4

Cost and Performance Analysis

In order to quantify the benefit/cost ratio of the adoption of cluster computing technology we shall now analyze in some more details the performance in terms of communication bottleneck and processing power as well as the hardware and management costs.

4.1 A performance comparison among message passing platforms

Table 4.1 reports a latency-bandwidth characterization of messaging systems running on a number of NOW prototypes, including the standard Linux TCP/IP stack. A few communication layers running on commercial MPPs are reported as well, in order to provide a rough comparison with more traditional expensive parallel computers.

Table4.1 is divided into four sections, respectively for standard interface, user-level and efficient OS NOW message passing architectures, and messaging systems running on traditional MPPs.

The most performant NOW communication systems appear indeed to be competitive w.r.t. MPPs, although comparing non-standard, low-level NOW programming interfaces with higher-level, standard message passing on MPPs is not completely fair. Of course NOW platforms cannot compete with MPPs as for asymptotic bandwidth, and lacks one of the main MPP feature, namely scalability with respect to the number of processing nodes. On the other hand, the current cost of the scalability characteristics of an MPP is hardly justified by a large number of applications where parallel processing would, in principle, make sense. Indeed most of the MPP installations are "small" machines, equipped with only tenths of processors. A "small" MPP configuration is clearly outperformed in terms of price/performance and even absolute performance by an efficient modern NOW.

It is apparent from Table 4.1 that the standard interface architectural approach delivers quite poor exploitation of the interconnection hardware. This is evident with low-cost 100base-T (PARMA2, Linux TCP/IP sockets) as well as a

Platform	Latency (μs)	Bandwidth (MByte/s)
Linux 2.0.29 TCP/IP sockets (3COM 3c595 100base-T)	117.0	6.6
Fast Sockets (Myrinet,UltraSPARC)	57.8	32.9
PARMA^2 PRP sockets (3COM 3c595 100base-T)	74.0	6.6
FM (Myrinet,PPro) [33]	12.5	49.0
U-Net (ATM) [65]	44.5	15.0
HPAM (FDDI) [44]	14.5	12.0
MU-Net (Myrinet, UltraSPARC) [47]	20.0	18.0
BIP (Myrinet,PPro) [52]	4.3	125.0
GAMMA (100base-T) [15]	12.7	12.2
U-Net (100base-T) [65]	30.0	12.1
SSAM (ATM) [60]	30.0	5.6
Sender-based protocols (FDDI) [58]	23.5	10.2
Pupa (100base-T) [59]	198.0	7.8
Net* (100base-T) [54]	30.1	6.2
CM-5 CMAML ports [11]	10 - 15	8.5
SP2 MPL [40, Table 2]	44.8	34.9
T3D PVMFAST [40, Table2]	30.0	25.1

Table 4.1: “Ping-pong” comparison of message passing systems.

much more expensive and performant interconnect like Myrinet (Fast Sockets). This is the price to pay for retaining low-level industry-standard programming interfaces which hamper many possible optimizations.

It is apparent as well that the user-level approach has been adopted only with higher performance interconnect (Myrinet, FDDI, ATM). This is due to two reasons, namely: higher-end network devices have on-board programmable processors which could run multiplexing/protection code, and an OS kernel mediation between user processes and network hardware is believed to become a communication bottleneck with fast enough LAN hardware. However it could be argued that user-level access to network devices be not of primary concern as for communication performance, given the loose integration between high-speed commodity network adapters and the host memory hierarchy. Porting some efficient OS supported communication architectures on high-end LAN interconnects would provide more insights about the actual impact of kernel-mediated message multiplexing on communication performance with fast commodity LANs. Up to our knowledge, currently no research project is engaged in such a porting activity. From the reported performance numbers it is clear that user-level systems range from very efficent (BIP) to very poor (MU-Net). The choice of a workstation equipped with a slow I/O bus (like SUN SBus) is really a bad one in this respect. More and more NOW projects, although using high-speed interconnects, are leveraging high-end PCs instead of workstations due to unbeatable price/performance and the good capabilities of the PCI bus.

Efficient, OS supported messaging systems almost exclusively run on low-cost commodity interconnected NOWs (with the exception of the early SSAM on ATM). This kind of platform, leveraging high-end PCs and Fast Ethernet LAN, emerges as a low-cost alternative to fast NOWs, offering better price/performance ratio. The poorer network performance poses even harder efficiency challenges. Nevertheless no plain user-level messaging architecture (maybe with no protected device multiplexing) has been ported on low-cost NOWs yet. However GAMMA shows that excellent results on low-cost NOWs can be obtained even with OS kernel mediation and even with a fairly high-level and flexible programming interface (GAMMA delivers half the latency of the raw U-Net on Fast Ethernet), provided that communication protocol and device multiplexing support be carefully developed according to a performance-oriented approach.

4.2 Hardware, software and assembly/maintenance costs

Recent advances in hardware technology have brought the lowest cost, off-the-shelf PC components to levels of performance quite acceptable for Cluster Computing. Intel x86 compatible processors with 400MHz or more clock, 100 MHz memory bus, 128 MByte SDRAM, AGP, 33MHz SuperDMA, etc., are now able to deliver performance that only one year ago were achievable only by high-end, much more expensive workstations. Disk storage technology is also experiencing a tremendous increase in performance and capacity, with 7200RPM EIDE disks capable of storing tenths of GBytes at a cost of less than 35 Euros per

GByte. Fast Ethernet interconnection technology can provide 100Mb/s transfer rates for less than 10% of the total cost of a computing node. In summary, today (December 1998) it is possible to assemble a cluster of PC for a total hardware cost of approximately 1,000 Euros per node, including medium/high speed interconnect, processing power of more than 200MFLOPS per node, and significant disk space capabilities. A cluster of 100 computing nodes could be easily assembled at a total cost of about 100,000 Euros, delivering 20GFLOPS, with 12GByte aggregated RAM and 600GByte aggregated disk space for number crunching and/or data mining applications.

The software cost could be significant for a large cluster installation if proprietary, licensed software components would be chosen. However Free Software components developed at various University or research labs are readily available that can solve the problem at no cost in a quite satisfactory way.

The main costs involved in the assembly, set-up, management and maintenance of a large cluster are probably the cost of skilled man-power that is currently needed to perform such activities. Only few research-oriented institutions and IT industries currently already have the skilled man-power to devote to such support activities. Non IT industries that would be the large majority of potential users of the technology would rather pay for consultancy or, even better, for the license of a turn-key software solution than hire skilled people that can develop and maintain the system. However, currently there is no such product to buy the license of as a turn-key cluster that works properly.

4.3 Processing power: benchmarks and real applications

A characterization of the Processing Power of a computing platforms is usually derived running computationally intensive *benchmarks* and is usually measured in Million (double precision) FLOating-point instructions executed Per Second (MFLOPS). A well established benchmark for both sequential and parallel platforms is the Linpack routine, an optimized Fortran implementation of resolution of square matrix linear systems by LU factorization with partial pivoting and back-substitution. Linpack MFLOPS are used to classify the "top 500 Supercomputer list" [2]. According to that list, the most powerful machine that is currently available is the ASCI Option Red assenbled at Sandia National Lab out of 9,000 Pentium PRO processors. A specially optimized version of the Basic Linear Algebra Subroutines (BLAS) has been developed for Pentium PRO processors in the framework of such project and is freely available over the Internet.

Figure 4.1 reports the execution speed curves of the MPI version of Linpack obtained using the MPICH implementation of MPI [1] on a cluster of 4 PentiumII processors running at 300MHz clock rate and disposing of 64MByte 10ns. SDRAM. Of course with larger and larger matrices the communication overhead (which grows quadratically with matrix size) becomes less and less important compared to the computation time (which grows cubically with matrix size). Therefore good performance results can be measured with large matrices

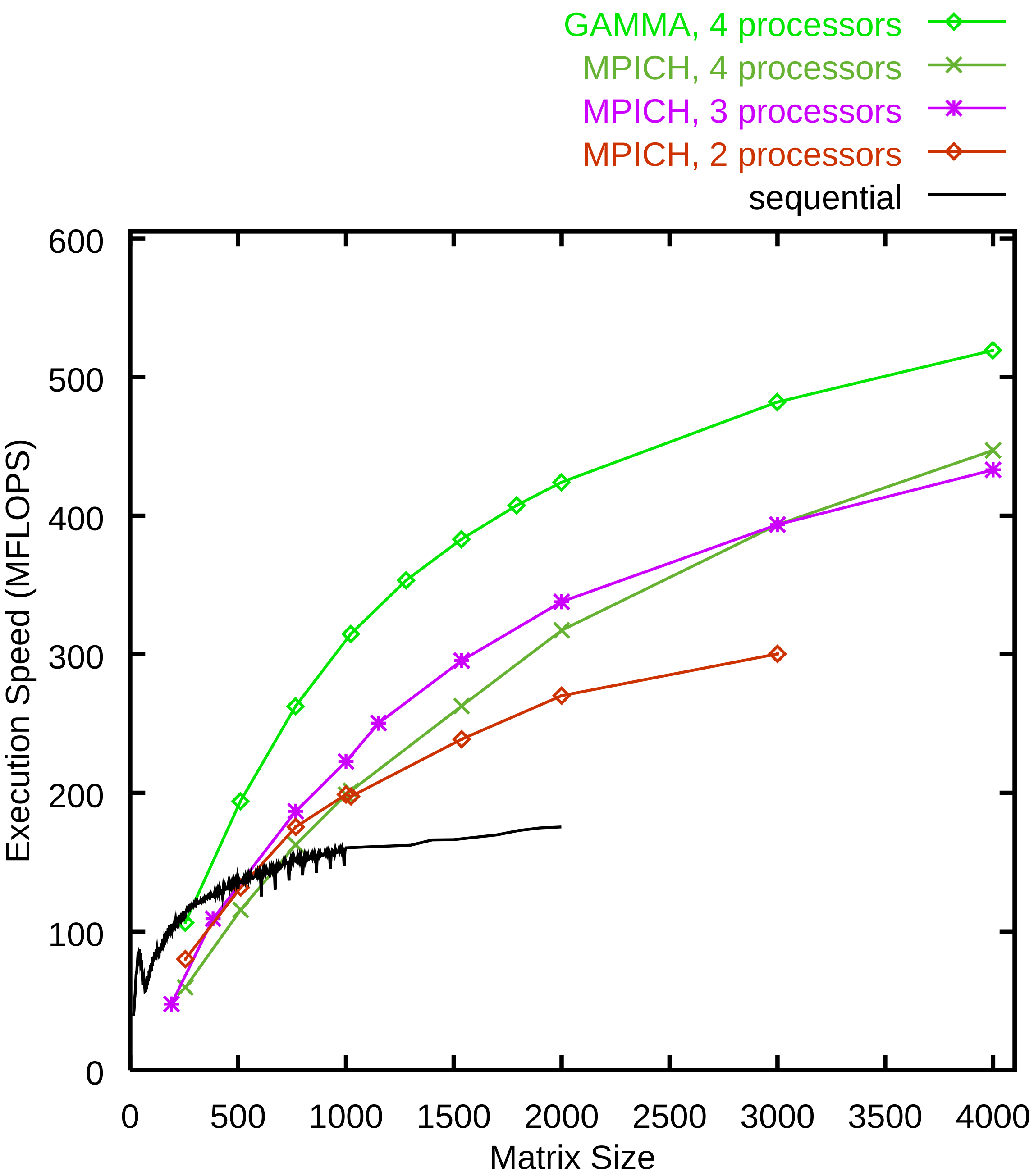

Figure 4.1: Double precision Linpack performance using MPICH atop Linux TCP/IP on Pentium II 300MHz cluster. Performance from sequential LAPACK and from GAMMA 4 processors cluster are also included for comparison.

more or less independently of the efficiency of the underlying message passing libraries. On the other hand, if we want to use this benchmark to compare different messaging systems on the same processing nodes, results are significant only with rather small problem sizes.

The performance of a four processors GAMMA cluster indeed appears much more satisfactory than standard MPI for relatively small matrices, with speedup greater than 1 obtained with matrices as small as 300×300 instead of 800×800. More than 500 MFLOPS (519 to be precise) are achieved with four processors yet with matrices as small as 4000×4000. Performance achieved by the original LAPACK sequential version of the benchmark with the same BLAS and processor is reported as a reference in the performance graph. The theoretical peak floating point speed of Pentium II 300 MHz is 300 MFLOPS, since the pipelined FPU of the Pentium architecture is able to start one floating point operation per clock cycle. Of course this is only an upper bound of what a Pentium CPU can actually do. Assuming 180 MFLOPS as an upper bound on the sequential performance, the speed-up achieved by a four processors GAMMA cluster with 4000×4000 matrices would be 2.88. The MPICH implementation based on TCP/IP reaches in this case 450MFLOPS, namely a speed-up of 2.5.

Two "real" simulation applications originally written in Fortran 77 and already parallized under PVM have recently been ported to GAMMA as well. The first one was a molecular dynamic simulation code [16]. The second one was a flame front propagation simulation code [17]. Both applications required some re-structuring of the communication patterns in order to adapt to the characteristics of the GAMMA platform and run efficiently on it. The programming effort for the porting was a few man-weeks in each case, in order to debug and tune the code for good performance. In both cases a collision-free communication pattern was almost immediately identified and implemented, taking advantage both of efficient multicast communication and of low latency for token passing and barrier synchronizations. Both codes exhibited almost linear speed-up on Pentium 133 platforms comprising up to 16 processors. In both cases, the same collision-free communication pattern could not be reproduced in PVM efficiently, thus proving the substantial advantage of the GAMMA programming interface in case of exploitation of shared Fast Ethernet. The GAMMA version on Pentium 133 achieved good absolute performance compared to the PVM version running on an IBM SP-2 platform, while the lack of sufficient processing power on single nodes did not allow the GAMMA version to compete in absolute terms with DEC cluster implementations of the PVM version. The transition from Pentium 133MHz to PII (400MHz or higher) is however expected to make a GAMMA implementation competitive with state-of-the-art SMP platforms at least in terms of performance/cost ratio if not in absolute MFLOPS terms.

The experience with the porting on GAMMA of all those benchmarks and simulation applications showed that a good matching between the efficient communication primitives offered by the system and the "communication needs" of a benchmark or a real application may have a greater impact on overall performance than mere bandwidth, latency, and even throughput curves. In particular, several applications can naturally be programmed exploiting serialized (i.e., collision-free, over shared Fast Ethernet channels), multicast communications. Low-latency plus direct broadcast communications offer more flexibility at the programming level to implement collision-free collective communication patterns. Similar collision-free patterns, which usually require an explicit serialization of send operations, are not exploitable by high-latency messaging systems like TCP/IP based implementations of MPI and PVM because of their poor emulations of broadcast communications and their high communication overhead. Based on efficient messaging systems such as, e.g., GAMMA, significant numerical applications such as Linpack, Matrix multiplication, etc., can currently be run on several nodes delivering performance/cost ratios of the order of 0.1 or 0.2 MFLOPS/Euro, up to several GFLOPS.

The availability of direct hardware broadcast makes Ethernet very attractive as a cluster interconnect for parallel processing, provided that this feature is directly exposed to the application or library designer. Therefore the forthcoming Gigabit Ethernet technology (the cheapest one, based on Copper rather than Fiber Optic wiring) appears to be a great promise in this field, but such a promise needs additional efforts from low-level communication software developers to be fulfilled. As a final goal, a performance-oriented version of MPI especially tailored for clusters of PCs should be implemented. Indeed the latest version of MPICH provides an optional (although neither stable nor yet complete) support to more efficient collective communications. Efficient direct broadcast communications would allow the exploration of novel data distribution patterns in data-parallel numerical applications, whereas efficient collective communication functions would encourage the development of more modular and portable parallel code.

4.4 Reliability of and support for low-cost clusters

Hardware reliability is certainly not a major issue in the context of Cluster Computing due to the very low cost of spare parts and they high degree of standardization. The substitution of a failed disk, NIC, CPU or memory bank is easy, cheap, and requires very little equipment and technical skill besides a screw-driver and the ability to use it. Good backup strategies must however be organized in order to restore data, software installations and customization in case of disk failures.

On the other hand, software reliability and fault-tolerance is certainly an issue for several applications that justify the adoption of Cluster Computing technology because of the large amount of data to be processed and/or because of their long computation times. Good OS and system software choices may reduce the risk but avoiding it altogether is beyond current technology. For instance, a study conducted in 1995 [45] showed that basic Unix application abort or hang-up when subject to random input due to software bugs in the range of 18% to 23% for major commercial distributions (with the best robustness shown by GNU/Linux applications, in the range of 7% to 9%). Substantial scientific and technological research efforts are still needed in the field in order to obtain truly satisfactory platforms. However the problems are not specific to

Cluster Computing. Similar problems may be found in all the other computing platforms and applications subject to similar requirements and constraints. We could say therefore that overall ICT is not mature enough to deal with such challenging requirements and constraints rather than restrict the criticism to Cluster Computing.

As an example of a rather trivial technology that is currently lacking in the domain of low cost platforms and that would greatly improve reliability and usability not only of PC Clusters but also of individual low-cost workstations we could mention RAID. RAID technology is currently available for relatively expensive SCSI disks, but not for much cheaper and cost-effective EIDE disks. With recent advancements in magnetic storage technology, commodity, lowest-end EIDE disk are now raising capacity (10-14 Gbytes for 3"$\frac{1}{2}$ disks at 7200RPM) and performance (average seek times less than 9 ms.) that only one year ago could be achieved only by highest end SCSI disks at the ridicolous cost of about 25 Euros per GByte. The only condition hampering the use of such cost-effective devices for large database applications seems to be the lack of appropriate software to operate RAID on them, thus guaranteeing an easy backup for sensitive information in the (unlikely but possible) case of hardware failure.

4.5 Evaluation of technology transfer to industry

The main requirement at the basis of the majority of european projects sponsored by the UE, especially within the Fourth Framework, was the identification of an industrial end user partner interested to consume products and results achieved during the project. In the case of software product developments, a strong emphasis was also put on the "market" perspectives and strategies for the software developers, in terms of their capacity to sell the same or derived software to other users of the same type. These requirements, when falling in the context of projects with potentially high scientific relevance, as in the case of cluster computing applications, can have a significantly negative impact on the general validity of the produced results.

The presented overview on UE sponsored projects shows that in all cases results of the project are software products ready to use for an industrial partner, porting of an existing application on workstation or PC clusters, the aim to exploit existing hardware (networks of PCs) belonging to a company to allow batch computing. This customization of software prototypes inevitably reduces the potential users population, so that the development of radically new software components is not justified in terms of the software provider market strategies. Therefore, many times this approach does not lead to new technologies, or to significant scientific improvements. European projects often limit themselves to fund the application to existing industrial needs of technoligies already studied and developed in scientific domains (with little engineering effort to convert them into real products) or already developed and commercially exploited in the US. Many funded projects are proposed by a high number of partners with very different experience domains. Often, many of them propose very ambitious projects, but starting from inadequate experiences to be successfully performed.

The double emphasis always reserved to the end user and their needs for very specific results on one hand and the emphasis on the profitability of the market for the software provider on the other hand (that of course results in the strong preference for the use of consolidated market standards rather than the investment in development of new and therefore of dubious customer acceptance technologies) prevents these projects to obtain results having widespread interest or of new products of general interest to be distributed in public domain.

Recently, the trend towards the production of public domain software is evident. In particular, the most important project has been the development of the public domain Linux OS, one of the most significant experiences carried out in the last years, mainly in Europe but without any support from the EU. The WWW technology is another example of the lack of exploitation capabilities for European enterprises of advanced technology originally developed in Europe (by CERN). This kind of approach toward very innovative and/or public domain products is completely prevented by following that particular strategy. And, by the way, also the EU itself does not show clear signs of interest in the support for the dissemination of the new and/or public domain software technology as demonstrated by the fact that even the Cordis database for the access to information on European technology research and dissemination is built upon proprietary software products and data formats owned by US based corporates instead of much more appriate public domain, platform independent standards such as HTML (originated in Europe) or Java.

Chapter 5

Consolidating Lesson Learned

At the end of this survey it might be worth to try to consolidate some general principles emerging from the survey itself. The most relevant question by far is whether the technology appears mature enough to be proposed to SMI in a way that they could integrate, within their production process, without incurring in out-of-scale costs. The second question that naturally raises is whether this technology meets performance and cost expectations.

The technology, in itself, is inherently cheap but it seems that the status of the support software is not quite adequate yet. This bear as a consequence that, from the candidate user a level of technical competence is required that is not generally and readily available.

The main point here revolves around the administration software of a cluster of workstations in general and of PCs in the specific. Still today a cluster does not present an enough unified image as a single resource partaking in the solution of large problems. From the administration point of view, in all of the case examined (we can't rule out that some other relevant experience might have escaped our survey) a distributed computing resource is still seen as a set of independent computing platforms, each one with its own administration burden. This generates a landslide of consequences, probably the most crucial of which is an inherently non scalable environment. Each software upgrade has to be performed on each individual node, and sooner or later this will cause a large system to be out of synch.

It might be worth, for future actions, to take into consideration the promotion of software activities aimed at developing such a distributed management system. We believe that all the needed pieces of software are available within the public domain software, what is required is a focussed integration effort.

Concerning the problem of meeting performance expectations, the answer is basically "yes." Cluster of workstations and in particular cluster of PC's are a reliable and effective computing platform that can carry enormous benefits. The statement although must be qualified in the sense that expectations must be "reasonable." We are facing, for this issue as well, a choice between specialisation and special purpose. We believe that the most beneficial sector is in application areas not strictly concerned with supercomputing, but rather what we could call "cheap higher performance" (as opposed to individual, ever increasingly powerful, PCs). For real supercomputing we are again hampered by scalability issues of technology built upon off-the-shelf components. This involves both the administration issues mentioned above but even more technical questions like the limitations of commercial, low cost, interconnection technology as well as problems related to mean life time between hardware failures (in case of very large clusters) and the need for automated tools for auto-reconfiguration and on-line fault tolerance.

It is our opinion that in the technological field of interconnection, just waiting for commercial technology to improve will suffice to obtain reasonably good solutions. On the other hand, it might be worth promoting actions aimed at developing hardware redundant systems with (possibly software) dynamical capabilities of replacing failing parts. This issue is all the more relevant if considered within the contest of commercial data management applications that is, in our view, possibly the most promising segment for low-cost distributed computing.

Bibliography

[1] MPICH - A Portable MPI Implementation, `http://www. mcs. anl. gov /mpi /mpich/`, 1998.

[2] Top 500 Supercomputer Sites, `http://www. netlib. org /benchmark /top500.html`, 1998.

[3] H.E. Bal, M.F. Kaashoek, and A.S. Tanenbaum. Orca: A language for parallel programming of distributed systems. *IEEE Trans. on Software Engineering*, 18(3):190–205, 1992.

[4] S. Balay, W.D. Gropp, L.C. McInnes, and B.F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhauser Press, 1997.

[5] A.J.C. Bik and D.B. Gannon. A note on native level 1 blas in java. Available at http://www.extreme.indiana.edu/hpjava/papers/-native/native.html.

[6] J. M. Blum, T. M. Warschko, and W. F. Tichy. PULC: Parastation User-Level Communication. Design and Overview. In *Proc. Workshop PC-NOW, IPPS/SPDP'98*, number 1388 in Lecture Notes in Computer Science, pages 498–509, Orlando, Florida, April 1998. Springer.

[7] M. A. Blumrich, K. Li, R. Alpert, C. Dubnicki, E. W. Felten, and J. Sandberg. Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer. In *Proc. of the 21th Annual Int'l Symp. on Computer Architecture (ISCA'94)*, pages 142–153, April 1994.

[8] F. Bodin, P. Beckman, D. Gannon, S. Narayana, and S.X. Yang. Distributed pC++: Basic ideas for an object parallel language. *Scientific Programming*, 2(3), 1993.

[9] D. Caromel. Toward a method of object-oriented programming. *Com of ACM*, 36(9):90–102, 1993.

[10] D. Caromel, F. Belloncle, and Y. Roudier. The C++// system. In G. Wilson and P. Lu editors, editors, *Parallel Programming Using C++*. MIT Press, 1996.

[11] Center of Computational Science of the U.S. Naval Research Laboratory. On-line documentation,`http://www .nrl.navy.mil /CCS /help /cm5 /products /cmmd.html`.

[12] A.A. Chien, U.S. Reddy, J. Plevyak, and J. Dolby. ICC++: A C++ dialect for high- performance parallel computing. In *Proc. 2nd Int. Symp. on Object Technologies for Advanced Software*, Kanazawa, Japan, 1996.

[13] G. Chiola and G. Ciaccio. Implementing a Low Cost, Low Latency Parallel Platform. *Parallel Computing*, (22):1703–1717, 1997.

[14] G. Chiola and G. Ciaccio. Fast Barrier Synchronization on Shared Fast Ethernet. In *Proc. of the 2nd International Workshop on Communication and Architectural Support for Network-Based Parallel Computing (CANPC'98)*, number 1362 in Lecture Notes in Computer Science, pages 132–143. Springer, February 1998.

[15] G. Ciaccio. Optimal Communication Performance on Fast Ethernet with GAMMA. In *Proc. Workshop PC-NOW, IPPS/SPDP'98*, number 1388 in Lecture Notes in Computer Science, pages 534–548, Orlando, Florida, April 1998. Springer.

[16] G. Ciaccio and V. Di Martino. Porting a Molecular Dynamics Application on a Low-cost Cluster of Personal Computers running GAMMA. In *Proc. Workshop PC-NOW, IPPS/SPDP'98*, number 1388 in Lecture Notes in Computer Science, pages 524–533, Orlando, Florida, April 1998. Springer.

[17] G. Ciaccio, V. Di Martino, and P. Lanucara. Porting the Flame Front Propagation Problem on GAMMA. In *Proc. International Conference and Exhibition on High-Performance Computing and Networking 1998 (HPCN Europe '98)*, number 1401 in Lecture Notes in Computer Science, Amsterdam, The Netherlands, April 1998. Springer.

[18] O. Couland and H. Dillon. PARA++: C++ bindings for message passing libraries: User guide. Technical Report 0174, INRIA, Villers Les Nancy, France, 1995.

[19] A. Davis, M. Swanson, and M. Parker. Efficient Communication Mechanisms for Cluster Based Parallel Computing. In *Proc. of the 1st International Workshop on Communication and Architectural Support for Network-Based Parallel Computing (CANPC'97)*, number 1199 in Lecture Notes in Computer Science. Springer, February 1997.

[20] P. Druschel and L. Peterson. Fbufs: A High-bandwidth Cross-domain Transfer Facility. In *Proc. of the 14th ACM Symp. on Operating Systems Principles (SOSP'93)*, December 1993.

[21] D. Engler, M. Kaashoek, and J. O'Toole. Exokernel, an Operating System Architecture for Application-level Resorce Management. In *Proc. of the 15th ACM Symp. on Operating Systems Principles (SOSP'95)*, December 1995.

[22] C. Reschke et al. A Design Study of Alternative Topologies for the Beowulf Parallel Workstation. In *Proc. of the Fifth IEEE Int'l Symp. on High Performance Distributed Computing (HPDC-5)*, August 1996.

[23] MPI Forum. MPI: A Message-Passing Interface MPI Forum. Technical Report CS/E 94-013, Department of Computer Science, Oregon Graduate Institute, March 1994.

[24] G. Furnish. C4 – canonical classes for concurrency control. Available at http://dino.ph.utexas.edu:80/ furnish/c4/.

[25] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. *PVM: Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing.* MIT Press, Cambridge, MA 02142, USA, 1994. Also available at `http://www.netlib.org/pvm3/book/pvm-book.html`.

[26] G. A. Geist, J. A. Kohl, and P. M. Papadopolous. PVM and MPI: A Comparison of Features. *Calculateurs Paralleles*, 8(2), 1996.

[27] A.S. Grimshaw. Easy to use object-oriented parallel programming. *IEEE Computer*, 26(5):39–51, 1993.

[28] W. Gropp and E. Lusk. User's Guide for MPICH, a Portable Implementation of MPI. Technical Report MCS-TM-ANL-96/6, Argonne National Lab., University of Chicago, 1996.

[29] The HPC++ Working Group. HPC++ white papers. Technical Report TR 95633, Center for Research on Parallel Computation, 1995.

[30] M. Heinrich, J. Kuskin, D. Ofelt, J. Heinlein, J. Baxter, J. P. Singh, R. Simoni, K. Gharachorloo, D. Nakahira, M. Horowitz, A. Gupta, M. Rosenblum, and J. L. Hennessy. The Performance Impact of Flexibility in the Stanford FLASH Multiprocessor. In *Proc. of the 6th Symp. on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VI)*, pages 274–285, October 1994.

[31] R. Hempel. The MPI Standard for Message Passing. *Lecture Notes in Computer Science*, 797:247–252, 1994.

[32] Kay Hwang. *Advanced Computer Architecture.* Mc Graw-Hill, Inc., 1997.

[33] G. Iannello, M. Lauria, and S. Mercolino. Cross-Platform Analysis of Fast Messages for Myrinet. In *Proc. of the 2nd International Workshop on Communication and Architectural Support for Network-Based Parallel Computing (CANPC'98)*, number 1362 in Lecture Notes in Computer Science, pages 217–231. Springer, February 1998.

[34] Y. Ishikawa, A. Hori, H. Tezuka, M. Matsuda, H. Konaka, M. Maeda, T. Tomokiyo, and J. Nolte. MPC++. In G.V. Wilson and P. Lu, editors, *Parallel Programming Using C++*, pages 429–464. MIT Press, 1996.

[35] S. J. Fink J. H. Merlin, S. B. Baden and B. M. Chapman. Multiple data parallelism with HPF and KeLP. In *Proc. HPCN '98*, Amsterdam, 1998.

[36] Howard W Johnson. *FastEthernet: Down of a new network*. Prentice Hall, Inc., Upper Saddle River, New Jersey, USA, first edition, 1996.

[37] L.V. Kale and S. Krishnan. Charm++ A portable concurrent object oriented system based on C++. In *Proc. OOPSLA '93*, pages 91–109, Washington DC, 1993.

[38] J. Kuskin, D. Ofelt, M. Heinrich, J. Heinlein, R. Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum, and J. L. Hennessy. The Stanford FLASH Multiprocessor. In *Proc. of the 21th Annual Int'l Symp. on Computer Architecture (ISCA'94)*, pages 302–313, April 1994.

[39] A. Lumsdaine. Object oriented MPI: A class library for the Message Passing Interface. In *Proc of Parallel Object-Oriented Methods and Applications '96*, Santa Fe, New Mexico, 1996.

[40] P. Marenzoni, G. Rimassa, M. Vignali, M. Bertozzi, G. Conte, and P. Rossi. An Operating System Support to Low-Overhead Communications in NOW Clusters. In *Proc. of the 1st International Workshop on Communication and Architectural Support for Network-Based Parallel Computing (CANPC'97)*, number 1199 in Lecture Notes in Computer Science, pages 130–143. Springer, February 1997.

[41] E. P. Markatos and M. G. H. Katevenis. Telegraphos: High-Performance Networking for Parallel Processing on Workstation Clusters. In *Proc. of the 2nd IEEE Symp. on High-Performance Computer Architecture (HPCA-2)*, February 1996.

[42] E. P. Markatos and M. G. H. Katevenis. User-Level DMA without Operating System Kernel Modification. In *Proc. of the 3th IEEE Symp. on High-Performance Computer Architecture (HPCA-3)*, February 1997.

[43] E. P. Markatos, M. G. H. Katevenis, and P. Vatsolaki. The Remote Enqueue Operation on Networks of Workstations. In *Proc. of the 2nd International Workshop on Communication and Architectural Support for Network-Based Parallel Computing (CANPC'98)*, number 1362 in Lecture Notes in Computer Science, pages 1–14. Springer, February 1998.

[44] R. P. Martin. HPAM: An Active Message layer for a Network of HP Workstations. In *Proc. of Hot Interconnect II*, August 1994.

[45] B.P. Miller, D. Koski, C.P. Lee, V. Maganty, R. Murthy, A. Natarajan, and J. Steidl. Fuzz revisited: a re-examination of the reliability of UNIX utilities and services. Technical Report CS-TR-95-1268, Computer Science Department, University of Wisconsin, April 1995.

[46] MPI Forum. Special Issue: MPI2: A Message-Passing Interface Standard. *The International Journal of Supercomputer Applications and High Performance Computing*, 12(1–2):1–299, Spring–Summer 1998.

[47] S. Nagar, D. Seed, and A. Sivasubramaniam. Implementing Protected Multi-User Communication for Myrinet. In *Proc. of the 2nd International Workshop on Communication and Architectural Support for Network-Based Parallel Computing (CANPC'98)*, number 1362 in Lecture Notes in Computer Science, pages 30–44. Springer, February 1998.

[48] T. O'Brien, G. Roberts, M. Wei, and R. Winder. Uc++ v1.0: Language definition and semantics. Available at http://www.dcs.kcl.ac.uk/UC++/ucpp_1_0/ucpp.html.

[49] EUROPA Working Group on Parallel C++. Europa parallel c++ specification. Available at http://www.dcs.kcl.ac.uk/EUROPA/Final/.

[50] Peter S. Pacheco. *Parallel Programming with MPI.* Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 1997.

[51] S. Pakin, M. Lauria, and A. Chien. High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet Computation. In *Proc. Supercomputing '95*, San Diego, California, 1995. ACM Press.

[52] L. Prylli and B. Tourancheau. BIP: a new protocol designed for high performance networking on Myrinet. In *Proc. Workshop PC-NOW, IPPS/SPDP'98*, number 1388 in Lecture Notes in Computer Science, pages 472–485, Orlando, Florida, April 1998. Springer.

[53] S. Rodrigues, T. Anderson, and D. Culler. High-performance Local-area Communication Using Fast Sockets. In *Proc. USENIX'97*, 1997.

[54] R. D. Russel and P. J. Hatcher. Efficient Kernel Support for Reliable Communication. In *Proc. 1998 ACM Symp. on Applied Computing (SAC'98)*, Atlanta, Georgia, February 1998. ACM Press.

[55] T.J. Sheffler. A portable MPI-based parallel vector template library. Technical Report 95.04, RIACS, 1995.

[56] Glen D. Smith. Object-Oriented Fortran Tutorial. Technical Report MSSU-EIRS-93-17, Mississippi State University, July 1993.

[57] T. Sterling, D.J. Becker, D. Savarese, J.E. Dorband, U.A. Ranawake, and C.V. Packer. BEOWULF: A Parallel Workstation for Scientific Computation. In *Proc. 24th Int. Conf. on Parallel Processing*, Oconomowoc, Wisconsin, August 1995.

[58] M. R. Swanson and L. B. Stoller. Low Latency Workstation Cluster Communications Using Sender-Based Protocols. Technical Report UUCS-96-001, Dept. of Computer Science, University of Utah, January 1996.

[59] M. Verma and T. Chiueh. Pupa: A Low-Latency Communication System for Fast Ethernet. In *Proc. Workshop PC-NOW, IPPS/SPDP'98*, number 1388 in Lecture Notes in Computer Science, Orlando, Florida, April 1998. Springer.

[60] T. von Eicken, V. Avula, A. Basu, and V. Buch. Low-latency Communication Over ATM Networks Using Active Messages. *IEEE Micro*, 15(1):46–64, February 1995.

[61] T. von Eicken, A. Basu, V. Buch, and W. Vogels. U-Net: A User-Level Network Interface for Parallel and Distributed Computing. In *Proc. of the 15th ACM Symp. on Operating Systems Principles (SOSP'95)*, Copper Mountain, Colorado, December 1995. ACM Press.

[62] T. von Eicken, D.E. Culler, S.C. Goldstein, and K.E. Schauser. Active Messages: A Mechanism for Integrated Communication and Computation. In *Proc. of the 19th Annual Int'l Symp. on Computer Architecture (ISCA'92)*, Gold Coast, Australia, May 1992. ACM Press.

[63] T. M. Warschko, J. M. Blum, and W. F. Tichy. The ParaStation Project: Using Workstations as Building Blocks for Parallel Computing. In *Proceedings of the International Conference on Parallel and Distributed Processing, Techniques and Applications (PDPTA'96)*, volume 1, pages 375–386, Sunnyvale, California, August 1996.

[64] T. M. Warschko, W. F. Tichy, and C. H. Herter. Efficient Parallel Computing on Workstation Clusters. Technical report, `http://wwwipd.ira.uka.de /~warschko /parapc /sc95.html`, Karlsruhe, Germany, 1995.

[65] M. Welsh, A. Basu, and T. von Eicken. Low-latency Communication over Fast Ethernet. In *Proc. Euro-Par'96*, Lyon, France, August 1996.

[66] B.B. Wyatt, K. Kavi, and S. Hufnagel. Parallelism in object-oriented languages: a survey. *IEEE Software*, 9(6):56–66, 1992.