SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 1(1), 1–4 (JANUARY 1988)

Cluster Computing: The Commodity Supercomputing

MARK BAKER
School of Computer Science, University of Portsmouth, Milton Campus, Southsea, Hants., United Kingdom, PO4 8JF (`Mark.Baker@port.ac.uk`)

RAJKUMAR BUYYA
School of Computing Science, Queensland University of Technology 825a, PLAS Lab, S-Block Gardens Point Campus, Brisbane, Australia. (`rajkumar@fit.qut.edu.au`)

SUMMARY

The availability of high-speed networks and increasingly powerful commodity microprocessors is making the usage of clusters, or networks, of computers an appealing vehicle for cost effective parallel computing. Clusters, built using commodity-off-the-shelf (COTS) hardware components as well as free, or commonly used, software, are playing a major role in redefining the concept of supercomputing. In this paper we discuss the reasons why COTS-based clusters are becoming popular environments for running supercomputing applications. We describe the current enabling technologies and present four state-of-the-art cluster-based projects. Finally, we summarise our findings and draw a number of conclusions relating to the usefulness and likely future of cluster computing.

KEY WORDS   Commodity Components   Clusters   Message-Passing   Supercomputing   Parallel Computing

INTRODUCTION

Very often, applications need more computing power than a sequential computer can provide. One way of overcoming this limitation is to improve the operating speed of processors so that they can offer the power required by computationally intensive applications. Even though this is currently possible, future improvements are constrained by factors such as: the speed of light, certain thermodynamic laws and the high financial costs for processor fabrication. A viable and cost-effective alternative solution is to connect multiple processors together and coordinate their computational efforts. The resulting systems are popularly known as parallel computers and they allow the sharing of a computational task among multiple processors.

As Pfister[1] points out there are three ways of improving performance:

- *Work harder*,
- *Work smarter*, and
- *Get help*.

In terms of computing technologies the analogy to this mantra is that working harder is like using faster hardware (high performance processors, memory, or peripheral devices). Working smarter concerns doing things more efficiently and this revolves around the algorithms and techniques used to solve computational tasks. Finally, getting help refers to using multiple computers to solve a particular task.

CCC 0038–0644/88/010001–04

Received 6 October 1998
Revised 4 January 1999

The usage of parallel processing as a means of providing high-performance computational facilities for large-scale and grand-challenge applications has been investigated widely for many years. However, until fairly recently, the benefits of this research were confined to those individuals who had access to large (and normally expensive) parallel computing platforms. Today the situation is changing. Many organisations are moving away from using large supercomputing facilities and turning towards clusters of workstations.

This move is primarily due to the recent advances in high speed networks and improved microprocessor performance. These advances mean that clusters are becoming an appealing and cost effective vehicle for parallel computing. Clusters*, built using commodity hardware and software components, are now playing a major role in redefining the concept of supercomputing and the situation now exists where clusters can be considered today's *commodity supercomputers*.

An important factor that has made the usage of workstations a practical proposition is the standardisation of many of the tools and utilities used by parallel applications. Examples of these standards are the message passing library MPI[2] and data-parallel language HPF[3]. In this context, standardisation enables applications to be developed, tested and even run on NOW and then at a later stage to be ported, with little modification, onto dedicated parallel platforms where CPU-time is accounted and charged for. The following list highlights some of the reasons why workstation clusters are preferred over specialised parallel computers[4,5,6]:

- Individual workstations are becoming increasingly powerful.
- The communications bandwidth between workstations is increasing as new networking technologies and protocols are implemented.
- Workstation clusters are easier to integrate into existing networks than specialised parallel computers.
- Typical low user utilisation of *personal workstations*.
- The development tools for workstations are more mature than the contrasting proprietary solutions for parallel computers - mainly due to the non-standard nature of many parallel systems.
- Workstation clusters are a cheap and readily available alternative to specialised high performance computing platforms.
- Clusters can be enlarged and individual node capabilities can be easily extended, for example additional memory or processors can be installed.

At a basic level a cluster is a collection of workstations or PCs that are interconnected via some network technology. A more likely scenario is that the computers will be state-of-the-art and high-performance and the network will be one with a high bandwidth and low latency. Such a cluster can provide fast and reliable services to computationally intensive applications.

A Cluster is capable of providing similar or better performance and reliability than traditional mainframes or supercomputers. Also, if designed correctly, better fault-tolerance at a much lower hardware cost can be achieved. Many applications have successfully used clusters including computationally intensive ones such as quantum chemistry and computational fluid dynamics. In this paper, we focus on the components, tools, techniques, and methodologies involved in using clusters for high-performance or parallel computing.

Towards Low Cost Parallel Computing

In the 1980s, it was believed that computer performance was best improved by creating faster and more efficient processors. This idea was challenged by parallel processing, which in essence means

* Networks of Workstations (NOW), Clusters of Workstations (COW) and Workstation Clusters are synonymous

linking together two or more computers to jointly solve some problem. Since the early 1990's there has been an increasing trend to move away from expensive and specialised propriety parallel supercomputers towards networks of workstations. Among the driving forces that have enabled this transition has been the rapid improvement and availability of commodity high-performance components for workstations and networks. These technologies are making networks of computers (PCs or workstations) an appealing vehicle for parallel processing and this is consequently leading to low-cost *commodity supercomputing*.

Clusters can also be classified as[7]:

- Dedicated Clusters
- Non-dedicated Clusters

The distinction between these two cases is based on the ownership of the workstations in a cluster. In the case of dedicated clusters, a particular individual does not own a workstation and the resources are shared so that parallel computing can be performed across the entire cluster[8]. The alternative non-dedicated case is where individuals' own workstations and in this case applications, are executed by stealing idle CPU cycles[9]. The rationale for this is based on the fact that most workstation CPU cycles are unused, even during peak hours[10]. Parallel computing on a dynamically changing set of non-dedicated workstations is called *adaptive parallel computing*.

Where non-dedicated workstations are used, a tension exists between the workstation owners and remote users who need the workstations to run their application. The former expects a fast interactive response from their workstation, whilst the latter is only concerned with fast application turnaround by utilising any spare CPU cycles. This emphasis on sharing the processing resources removes the concept of node ownership and introduces the need for complexities such as process migration and load balancing strategies. Such strategies allow clusters to deliver adequate interactive performance as well as providing shared resources to demanding sequential and parallel applications.

Clearly, the workstation environment is better suited to applications that are not communication intensive - typically, one would see high message start-up latencies and low bandwidths. If an application requires higher communication performance, the existing LAN architectures, such as Ethernet, are not capable of providing it.

Traditionally in science and industry a workstation referred to some sort of UNIX platform and the dominant function of PC-based machines was for administrative work and word processing. There has, however, been a rapid convergence in processor performance and kernel-level functionality of UNIX workstations and PC-based machines in the last three years - this can be associated with the introduction of high-performance Pentium-based machines and the Window NT operating system. This convergence has led to an increased level of interest in utilising PC-based systems as some form of computational resource for parallel computing. This factor coupled with the comparatively low cost of PCs and their widespread availability in both academia and industry has helped initiate a number of software projects whose primary aim is to harness these resources in some collaborative way.

Commodity Components for Clusters

The continuous improvement of workstation and networks performance and availability of standardised APIs are helping pave the way for cluster-based parallel computing. In this section, we discuss some of the hardware and software components commonly used.

Processors and Memory

Processors

Over the past two decades, rapid advances have taken place in the design of microprocessor architectures, for example, RISC, CISC, VLIW, and Vector. Today, a single-chip CPUs is almost as powerful as processors used in supercomputers of the recent past. Indeed, recently researchers have been attempting to integrate combinations of processor, memory and network interfaces into a single chip. For example the Berkeley Intelligent RAM[11] project is exploring the entire spectrum of issues involved in designing general-purpose computer systems that integrate a processor and DRAM onto a single chip - from circuits, VLSI design and architectures to compilers and operating systems. Digital, in their Alpha 21364 processor, are trying to integrate processing, memory controller and network interface into a single chip.

Intel processors are most commonly used in PC-based computers. The current generation Intel x86 processor family includes the Pentium Pro and II. These processors, whilst not in the high band of performance, match the performance of medium level workstation processors[12]. In the high performance band, the Pentium Pro has good integer performance, beating Sun's UltraSPARC at the same clock speed, however the floating-point performance is much lower. The Pentium II Xeon[13], like the newer Pentium II's, uses a 100 MHz memory bus. It is available with a choice of 512KB to 2MB of L2 cache, and the cache is clocked at the same speed as the CPU, overcoming the L2 cache size and performance issues of the plain Pentium II. The accompanying 450NX chipset for the Xeon has a 64-bit PCI bus which can support *Gbps* interconnects.

Other popular processors include x86 variants (AMD x86, Cyrix x86), Digital Alpha, IBM PowerPC, SUN SPARC, SGI MIPS, and HP PA. Computer systems based on these processors have also been used as clusters; for example, Berkeley NOW uses Sun's SPARC family of processors in their cluster nodes. Further information about the performance of commodity microprocessors can be found at the VLSI Microprocessors Guide[14].

Memory

The amount of memory needed for the cluster is likely to be determined by the cluster target applications. Programs that are parallelised should be distributed such that the memory, as well as the processing, is distributed between processors for scalability. Thus, it is not necessary to install enough RAM to hold the entire problem in memory on each system (in-core), but there should be enough to avoid frequent swapping of memory blocks (page-miss) to disk, since disk access has a huge impact on performance.

Access to DRAM is extremely slow compared to the speed of the processor and can take orders of magnitude more time than a single CPU clock cycle. Caches are used to overcome the DRAM bottleneck. Recently used blocks of memory are kept in cache for fast CPU access if another reference to the memory is made. However, the very fast memory used for cache (Static RAM) is expensive and cache control circuitry becomes more complex as the size of the cache increases. Because of these limitations, the total size of a cache is usually in the range of 8KB to 2MB.

Within Pentium based machines it is not uncommon to have a 64-bit wide memory bus as well as a chip sets that supports 2 Mbytes of external cache. These improvements were necessary to exploit the full power of the Pentiums, and make the memory architecture very similar to that of UNIX workstations.

Disk IO

There are two disk interfaces commonly used with PCs. The most common is IDE, which on early generations of the PC was usually found on a daughter card, but is now often built into Pentium motherboards in the Enhanced IDE (EIDE) form[15]. This is a 16-bit wide interface with a peak transfer rate of 5 MBytes/s. The largest IDE drive commonly available is 10 Gbytes with a price of about $35 per GByte. An IDE interface allows two devices to be connected, and there are typically only one or two interfaces within a PC. CD-ROMs and tape drives are also available with IDE interfaces.

The other disk interface found on PCs is SCSI[16]. The Fast and Wide SCSI interface is capable of a peak rate of 10 MBytes/s in asynchronous and 20 MBytes/s in synchronous modes. A SCSI interface may contain up to 8 devices (this includes the host). Although most commonly used for disks, other devices such as CD-ROMs, printers and scanners are available with SCSI interfaces.

For similar sizes, SCSI disks tend to outperform IDE drives, however IDE has a significant price advantage (as much as two to one). SCSI disks are however available with much larger capacities: up to about 18 GBytes is common, retailing at about $55 per GByte.

Improving I/O Performance

Improvements in disk access time have not kept pace with microprocessor performance, which has been improving by 50% or more per year. Although magnetic-media densities have increased, reducing disk transfer times by approximately 60-80% per annum[17], overall improvement in disk access times, which rely upon advances in mechanical systems, has been less than 10% per year.

Parallel/Grand challenging applications need to process large amounts of data and data sets. *Amdahl's law implies that the speedup obtained from faster processors is limited by the system's slowest components*; therefore, it is necessary to improve I/O performance so that it is balanced with the CPU's performance. One way of improving I/O performance is to carry out I/O operations concurrently with the support of a parallel file system. One such parallel file system, known as software RAID, can be constructed by using the the disks associated with each workstation in the cluster.

Cluster interconnects

Individual nodes in a cluster are usually connected with a low-latency and high-bandwidth network. The nodes communicate over high-speed networks using a standard networking protocol such as TCP/IP or a low-level protocol such as Active[18] or Fast Messages[19]. A number of high performance network technologies are available in the marketplace. In this section, we discuss several of these technologies, including Fast Ethernet, ATM, and Myrinet as the means of interconnecting clusters. Further performance figures using the NetPIPE analysis tool can be found on the Interconnect Performance Web site[20].

Requirements

In most facilities where there is more than one workstation it is still likely that the interconnect will be via standard Ethernet[22]. In terms of performance, (latency and bandwidth) this technology is showing its age, but it is a cheap and easy way to provide file and printer sharing. A single Ethernet connection cannot be used seriously as the basis for cluster-based computing: its bandwidth and latency are not balanced compared to the computational power of he workstations now available. Typically one would expect the cluster interconnect bandwidth to exceed 10 MBytes/s and have message latencies of less than 100 μs.

A number of interconnect options now exist and were originally intended for use in WANs, but have been adapted to be used as the basis of high performance LANs, such as ATM. Other interconnects have been designed primarily for use in LANs, for example Myrinet.

The System Bus

The initial PC bus (AT, or now known as ISA bus) was clocked at 5 MHz and was 8-bits wide. When first introduced its abilities were well matched to the rest of the system. PCs are modular systems and until fairly recently only the processor and memory were located on the motherboard, other components being typically found on daughter cards, connected via a system bus. The performance of PCs has increased significantly since the ISA bus was first used and it has consequently became a bottleneck which has limited the machines throughput abilities. The ISA bus was extended to be 16-bits wide and was clocked in excess of 13 MHz. This, however, is still not sufficient to meet the demands of the latest CPUs, disk interfaces and other periperals.

Although a group of PC manufacturers introduced the VESA local bus, a 32-bit bus which matched the systems clock speed, this has largely been supplanted by the Intel created PCI[21] bus, which allows 133 Mbytes/s transfers and is used inside Pentium based PCs. PCI has also being adopted for use in non-Intel based platforms such as the Digital AlphaServer range. This has further blurred the distinction between PCs and workstations as the I/O sub-system of a workstation may be built from commodity interface and interconnect cards.

Ethernet and Fast Ethernet

Standard Ethernet[22] has become almost synonymous with workstation networking. This technology is in widespread usage, both in the academic and commercial sectors, however, its 10 Mbps bandwidth is no longer sufficient for use in environments where users are transferring large data quantities or there are high traffic densities. An improved version, commonly known as Fast Ethernet, provides 100 Mbps bandwidth and has been designed to provide an upgrade path for existing Ethernet installations. Standard and Fast Ethernet cannot co-exist on a particular cable, but each uses the same cable type. When an installation is hub-based and uses twisted-pair it is possible to upgrade the hub to one, which supports both standards, and replace the Ethernet cards in only those machines where it is believed to be necessary.

Asynchronous Transfer Mode (ATM)

ATM[23] is a switched virtual-circuit technology and it was originally developed for the telecommunications industry. It is embodied within a set of protocols and standards defined by the International Telecommunications Union (ITU). The international ATM Forum, a non-profit organisation continues this work. Unlike some other networking technologies, ATM is intended to be a used for both LAN and WAN, presenting a unified approach to both. ATM is based around small fixed sized data packets called *cells*. It is designed to allow cells to be transferred using a number of different media such as both copper wire and fibre optic. This hardware variety also results in a number of different interconnect performance levels.

When it was first introduced ATM used optical fibre as the link technology. However, in desktop environments this is undesirable, for example, twisted pair cables may have been used to interconnect a networked environment and moving to fibre-based ATM would mean an expensive upgrade. The two most common cabling technologies found in a desktop environment are telephone style cables, termed Category 3, and a better quality cable termed Category 5. The former, CAT-3, is suitable for Ethernet, the latter, CAT-5 is more appropriate for Fast Ethernet and is suitable for use with ATM. CAT-5 can be used with ATM at speeds of 15.5 MBytes/s, allowing upgrades of existing networks without replacing cabling.

Scalable Coherent Interface (SCI)

SCI[24,25], is an IEEE 1596 standard aimed at providing low-latency distributed shared memory access across a network. SCI is the modern equivalent of a Processor-Memory-I/O bus and LAN, combined. It is designed to support distributed multiprocessing with high bandwidth and low latency. It provides a scalable architecture that allows large systems to be built out of many inexpensive mass produced components[26].

SCI is a point-to-point architecture with directory-based cache coherence. It can reduce the delay of interprocessor communications even when compared to the newest and best technologies currently available, such as FibreChannel and ATM. SCI achieves this by eliminating the need for run-time layers of software protocol-paradigm translation. A remote communication in SCI takes place as just a part of a simple load or store process in a processor. Typically, a remote address results in a cache miss. This in turn causes the cache controller to address remote memory via SCI to get the data. The data is fetched to cache with a delay in the order of a few μs and then the processor continues execution.

Dolphin currently produces SCI cards for the SPARC SBus and PCI-based systems. Dolphin provide a version of MPI for their SCI cards. Currently MPI is available for Sun SPARC platforms, where it achieved a less than 12 μs zero message-length latency, they intend to port MPI to Windows NT in the near future. The Portland Group Inc. provide a SCI version of High Performance Fortran (HPF). Further information relating SCI-based cluster computing activities can be found at the University of Florida's HCS Lab.[27]

Although SCI is favoured in terms of fast distributed shared memory support, it has not been taken up widely because its scalability is constrained by the current generation of switches and its components are relatively expensive.

Myrinet

Myrinet is a 1.28 Gbps full duplex LAN supplied by Myricom[28]. It is a proprietary, high performance interconnect used in many of the more expensive clusters. Myrinet uses low latency cut-through routing switches, which offers fault tolerance by automatic mapping of the network configuration and simplifies the setting up of a network. There are Myrinet drivers for many operating systems, including Linux, Solaris and Windows NT. In addition to TCP/IP support, the MPICH implementation of MPI[29] is provided as well as a number of other customer developed packages, which offer sub 10 μs latencies.

Myrinet is rather expensive when compared to Fast Ethernet, but has the advantage of a very low-latency (5 μs, one-way point-to-point), high throughput (1.28 Gbps), and a programmable on-board processor that allows for greater flexibility. Myrinet can saturate the effective bandwidth of a PCI bus at almost 120 Mbytes/s with 4 Kbytes packets[30].

One of the main disadvantages of Myrinet is, as mentioned, its price compared to Fast Ethernet. The cost of Myrinet-LAN components for connecting high-performance workstations or PCs, including the cables and switches, is in the range of $1,600 to $1,800 per host. Also, switches with more than 16 ports are not available, so scaling can be messy, although switch chaining can be used to construct larger Myrinet clusters.

Operating Systems

In this section we focus on some of the popular operating systems available for workstation platforms. Operating system technology has matured and today an OS can be easily extended and new subsystems added without modification of underlying structure. Modern operating systems support multithreading at kernel level, and high-performance user level multithreading systems can be built without their kernel intervention. Most PC operating system have become stable and support multitasking, multithreading, and networking and they are smart enough to operate on each cluster node. The most popular and widely used cluster operating systems are Solaris, Linux and Windows NT.

Linux

Linux[31] is a UNIX-like operating system, which was initially developed by Linus Torvalds, a Finnish undergraduate student in 1991-92. The original releases of Linux relied heavily on the Minix OS, however, the efforts of a number of collaborating programmers resulted in development and implementation of a robust and reliable, POSIX compliant, operating system. Although initially developed by a single author, there are now a large number of authors involved in the development of Linux. One major advantage of this distributed development has been that there is a wide range of software tools, libraries and utilities available. This is due to the fact that any capable programmer has access to the OS source and can implement the feature that they wish. Obviously the features that are implemented vary in functionality and quality but the release strategy of Linux, that kernel releases are still controlled at a single point, and availability via the Internet, which leads to fast feedback about bugs and other problems has led to Linux providing a very rich and stable environment. The following are some of the reasons why Linux is so popular:

- Linux runs on cheap x86 platforms, yet offers the power and flexibility of UNIX.
- Linux is available from the Internet and can be downloaded without cost.
- It is easy to fix bugs and improve system performance.
- Users can develop or fine-tune hardware drivers and these can easily be made available to other users.

Linux provides the features typically found in UNIX implementations such as:

- Pre-emptive multi-tasking.
- Demand-paged virtual memory.
- Multi-user support.
- Multi-processor support.

Linux provides a POSIX compatible UNIX environment. Most applications written for UNIX will require little more than a recompile. In addition to the Linux kernel, a large amount of application and systems software is freely available. This includes the GNU software such as Bash, Emacs and the C compiler as well as XFree86, a public domain X server.

Microsoft Windows NT

Microsoft Corp.[32] is the dominant provider of software in the personal computing market place. Microsoft provides two basic operating systems: Windows 95/98 and Windows NT 4 (soon to become Windows NT 5/2000)[33]. NT and Windows 95 had approximately 66% of desktop operating systems market share in 1996 - IBM OS/2, UNIX, Mac OS and DOS comprise the remainder of the market share.

Windows NT (both Workstation and Server versions) is a 32-bit pre-emptive, multi-tasking amd multi-user operating system[35,36]. NT is fault tolerant - each 32-bit application operates in its own virtual memory address space. Unlike earlier versions of Windows (such as Windows for Workgroups and Windows 95/98), NT is a complete operating system and not an addition to DOS. NT supports most CPU architectures, including Intel x86, IBM PowerPC, MIPS and DEC Alpha. NT also support multiprocessor machines through the usage of threads. NT has object-based security model and its own special file system (NTFS) that allows permissions to be set on a file and directory basis. A schematic of the NT architecture is show in Figure 1. Windows NT has the network protocols and services integrated with the base operating system.

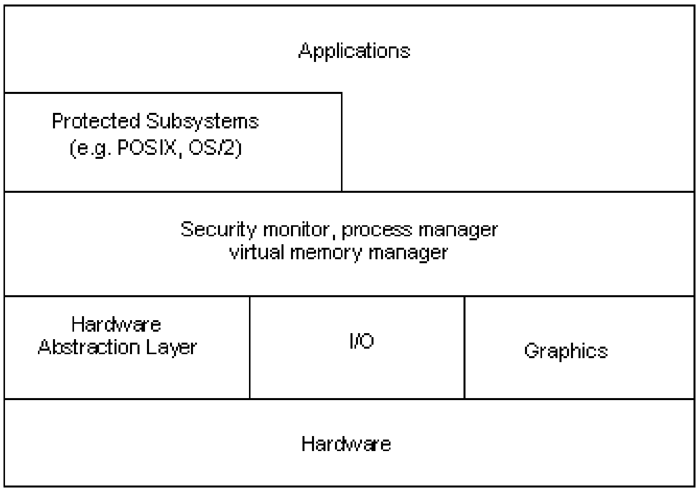

Figure 1. Windows NT Architecture

Solaris

The Solaris operating system from SunSoft, is a UNIX based multithreaded and multiuser operating system. It supports Intel x86 and SPARC based platforms. Its networking support includes a TCP/IP protocol stack and layered features such as Remote Procedure Calls (RPC), and the Network File System (NFS). The Solaris programming environment includes ANSI-compliant C and C++ compilers, as well as tools to profile and debug multithreaded programs.

The Solaris kernel supports multithreading, multiprocessing and has real-time scheduling features that are critical for multimedia applications. Solaris supports two kinds of threads: *Light Weight Processes (LWPs)* and user level threads. The threads are intended to be sufficiently lightweight so that there can be thousands present and that synchronisation and context switching can be accomplished rapidly without entering the kernel.

Solaris, in addition to the BSD file system, also supports several types of non-BSD file systems to increase performance and ease of use. For performance there are three new file system types: `CacheFS`, `AutoClient` and `TmpFS`. The `CacheFS` caching file system allows a local disk to be used as an operating system managed cache of either remote NFS disk or CD-ROM file systems. With

`AutoClient` and `CacheFS`, an entire local disk can be used as cache. The `TmpFS` temporary file system uses main memory to contain a file system. In addition there are other file systems like the `Proc` file system and *Volume* file system to improve system usability.

Solaris supports distributed computing and is able to store and retrieve distributed information to describe the system and users through the Network Information Service (NIS) and database. The Solaris GUI, OpenWindows, is a combination of X11R5 and the Adobe Postscript system, which allows applications to be run on remote systems with the display shown along with local applications.

Windows of Opportunity

The resources available in the average NOW, such as processors, network interfaces, memory and disks, offer a number of research opportunities, such as:

- *Parallel Processing*: Use the multiple processors to build MPP/DSM like system for parallel computing.
- *Network RAM*: Use the memory associated with each workstation as aggregate DRAM cache: this can dramatically improve virtual memory and file system performance.
- *Software RAID*: Use the arrays of workstation disks to provide cheap, highly available, and scalable file storage by using redundant arrays of workstation disks with the LAN as I/O backplane. In addition, it is possible to provide parallel I/O support to applications through middleware such as MPI-IO
- *Multi-path Communication*: Use the multiple networks for parallel data transfer between nodes.

Scalable parallel applications require good floating-point performance, low latency and high bandwidth communications, scalable network bandwidth, and faster access to files. Cluster software can meet these requirements by using resources associated with clusters. A file system supporting parallel I/O can be built using disks associated with each workstation instead of using expensive hardware RAID. Virtual memory performance can be drastically improved by using Network RAM as a backing-store instead of hard disk. In a way, parallel file systems and Network RAM reduces the widening performance gap between processors and disks.

It is common to connect cluster nodes using the standard Ethernet and a specialised high performance networks such as Myrinet. These multiple networks can be utilised for transferring data simultaneously across cluster nodes. The multi-path communication software can demultiplex data at the transmitting end to multiple networks and multiplex data at the receiving end. Thus all available networks can be utilised for faster communication of data between cluster nodes.

PROGRAMMING TOOLS FOR HPC ON CLUSTERS

Message Passing Systems (PVM/MPI)

Message passing libraries allow efficient parallel programs to be written for distributed memory systems. These libraries provide routines to initiate and configure the messaging environment as well as sending and receiving packets of data. Currently, the two most popular high-level message-passing systems for scientific and engineering application are the PVM[38] (Parallel Virtual Machine) from Oak Ridge National Laboratory and MPI (Message Passing Interface) defined by MPI Forum[37].

PVM is both an environment and a message-passing library, which can be used to run parallel applications on systems ranging from high-end supercomputers through to clusters of workstations. Whereas MPI is a specification for message passing, designed to be standard for distributed memory parallel computing using explicit message passing. This interface attempts to establish a practical, portable, efficient, and flexible standard for message passing. MPI is available on most of the HPC systems including SMP machines.

The MPI standard[39] is the amalgamation of what were considered the best aspects of the most popular message-passing systems at the time of its conception. It is the result of work undertaken by the MPI Forum, a committee composed of vendors and users formed at the Supercomputing Conference in 1992 with the aim of defining a message passing standard. The goals of the MPI design were portability, efficiency and functionality. The standard only defines a message passing library and leaves, amongst other things, the initialisation and control of processes to individual developers to define. Like PVM, MPI is available on a wide range of platforms from tightly coupled, massively parallel machines, through to NOWs. The choice of whether to use PVM or MPI to develop a parallel application is beyond the scope of this paper, but generally application developers choose MPI as it is fast becoming the de facto standard for message passing. MPI and PVM libraries are available for Fortran 77, Fortran 90, ANSI C and C++. There also exist interfaces to other languages - one such example is Java[40].

MPICH[41], developed by Argonne National Laboratory and Mississippi State, is probably the most popular of the current, free, implementations of MPI. MPICH is a version of MPI built on top of Chameleon[43]. The portability of MPICH derives from being built on top of a restricted number of hardware-independent low-level functions, collectively forming an Abstract Device Interface (ADI). The ADI contains approximately 25 functions and the rest of MPI approximately 125 functions. Implementing the ADI functions is all that is required to run MPICH on a new platform.

The ADI encapsulates the details and complexity of the underlying communication hardware into a separate module. By restricting the services provided to basic point-to-point message passing, it offers the minimum required to build a complete MPI implementation as well as a very general and portable interface. On top of the ADI, the remaining MPICH code implements the rest of the MPI standard, including the management of communicators, derived data types, and collective operations. MPICH has been ported onto most computing platforms[44], including Windows NT[45].

Distributed Shared Memory (DSM) Systems

The most efficient and widely used programming paradigm on distributed memory systems is message passing. A problem with this paradigm is its complexity and difficulty of efficient programming as compared to having a single image system, such as a uni-processor or shared-memory system. Shared memory systems offer a simple and general programming model, but they suffer from scalability. An alternate cost effective solution is provided by Distributed Shared Memory (DSM) systems. These offer a shared memory programming paradigm as well as the scalability of a distributed memory system. A DSM system offers a physically distributed and logically shared memory, which is an attractive solution for large scale high-performance computing.

DSM systems can be implemented by using software or hardware solutions[50]. The characteristics of software implemented DSM systems are: they are usually built as a separate layer on top of the message passing interface; they take full advantage of the application characteristics; virtual memory pages, objects, and language types are units of sharing. The implementation can be achieved by: compiler implementation; user-level runtime package; and operating system level (inside or outside the kernel). That is, a shared memory-programming model for a distributed memory machine can be implemented either solely by runtime methods[51], by compile time methods[52], or by combined compile time and runtime approach[53]. A few representative software DSM systems are Munin[54], TreadMarks[55], Linda[56], and Clouds[57].

The characteristics of hardware implemented DSM systems are:

- Better performance (much faster than software DSM approaches).

- No burden on user and software layers (full transparency).
- Finer granularity of sharing (cache block); extensions of the cache coherence schemes (snoopy or directory).
- Increased hardware complexity (not unreasonable).

Typical classes of hardware DSM systems are CC-NUMA, for example DASH[58], COMA, for example the KSR1[59], and reflective memory systems, such as Merlin[60].

Parallel Debuggers and Profilers

To develop correct and efficient high performance applications it is highly desirable to have some form of easy-to-use parallel debugger and performance profiling tools. Most vendors of HPC systems provide some form of debugger and performance analyser for their platforms. Ideally, these tools should be able to work in a heterogeneous environment. Thus making it possible to develop and implement a parallel application on, say, a NOW, and then actually do production runs on a dedicated HPC platform, such as the SGI/Cray T3E.

Debuggers

The number of parallel debuggers that are capable of being used in a cross-platform, heterogeneous, development environment is very limited[61]. For this reason an effort was begun in 1996 to define a cross-platform parallel debugging standard that defined the features and interface users wanted. The High Performance Debugging Forum (HPDF) was formed as a Parallel Tools Consortium[62] (PTools) project. The HPDF members include vendors, academics and government researchers, as well as HPC users. The forum meets at regular intervals and its efforts have culminated in the HPD Version 1 specification. This standard defines the functionality, semantics, and syntax for a command-line parallel debugger. Currently reference implementations for the IBM SP2 and the SGI/Cray Origin 2000 are underway.

Ideally, a parallel debugger should be capable of at least:

- Managing multiple processes and multiple threads within a process.
- Displaying each process in its own window.
- Displaying source code, stack trace and stack frame for one or more processes.
- Diving into objects, subroutines and functions.
- Setting both source-level and machine-level breakpoints.
- Sharing breakpoints between groups of processes.
- Defining watch and evaluation points.
- Displaying arrays and array slices.
- Manipulation of code variables and constants.

TotalView

TotalView is a commercial product from Dolphin interconnect Solutions[63]. It is currently the only widely available parallel debugger that supports multiple HPC platforms. TotalView supports most commonly used scientific languages (C, C++, F77, F90 and HPF), Message Passing libraries (PVM/MPI) and operating systems (SunOS/Solaris, IBM AIX, Digital UNIX and SGI IRIX 6). Even though TotalView can run on multiple platforms, it can only be used in an homogeneous environments. Namely, where each processes of the parallel application being debugged must be running under the same version of the OS. TotalView is GUI driven and has no command-line interface.

Performance Analysis Tools

The basic purpose of performance analysis tools is to help a programmer understand the performance characteristics of a particular application. In particular analyse and locate parts of an application that exhibit poor performance and create program bottlenecks. Such tools are useful for understanding the behaviour of normal sequential applications and can be enormously helpful when trying to analyse the performance characteristics of parallel applications.

To create performance information, most tools produce performance data during program execution and then provide a post-mortem analysis and display of the performance information. Some tools do both steps in an integrated manner and a few tools have the capability for run-time analysis, either in addition to or instead of post-mortem analysis.

Most performance monitoring tools consist of the some or all of the following components:

- A means of inserting instrumentation calls to the performance monitoring routines into the user's application.
- A runtime performance library that consists of a set of monitoring routines that measure and record various aspects of a program's performance.
- A set of tools that process and display the performance data.

In summary, a post-mortem performance analysing tool works by:

1. Adding instrumentation calls into the source code
2. Compiling and linking the application with a performance analysis runtime library.
3. Running the application to generate a tracefile.
4. Processing and viewing the tracefile.

A particular issue with performance monitoring tools is the intrusiveness of the tracing calls and their impact on the applications performance. It is very important that instrumentation does affect the performance characteristics of the parallel application and thus provide a false view of its performance behaviour. A lesser, but still important, secondary issue is the format of the trace-file. The files format is important for two reasons: firstly, it must contain detailed and useful execution information but not be huge in size, and secondly, it should conform to some 'standard' format so that it possible to use various GUI interfaces to visualise the performance data. Table I lists the most commonly used tools for performance analysis and visualisation on message passing systems.

Cluster Monitoring - System Administration Tools

Monitoring clusters is a challenging task that can be eased by tools that allow entire clusters to be observed at different levels using a GUI. Good management software is crucial for exploiting a cluster as a high performance computing platform.

There are many projects investigating system administration of clusters that support parallel computing, including:

- The Berkeley NOW[18] system administration tool gathers and stores data in a relational database. It uses a Java applet to allow users to monitor a system from their browser[6].
- The SMILE (Scalable Multicomputer Implementation using Low-cost Equipment) administration tool is called *K-CAP*. Its environment consists of *compute nodes* - these execute the compute-intensive tasks, a *management node* - a file server and cluster manager as well as a management console - a *client* that can control and monitor the cluster. K-CAP uses a Java applet to connect to the management node through a predefined URL address in the cluster.

Table I. Performance Analysis and Visualisation Tools

Tool	Supports	URL
AIMS	instrumentation, monitoring library, analysis	`http://science.nas.nasa.gov/Software/AIMS/`
MPE	logging library and snapshot performance visualisation	`http://www.mcs.anl.gov/mpi/mpich/`
Pablo	monitoring library and analysis	`http://www-pablo.cs.uiuc.edu/Projects/Pablo/`
Paradyn	dynamic instrumentation runtime analysis	`http://www.cs.wisc.edu/paradyn/`
SvPablo	integrated instrumentor, monitoring library and analysis	`http://www-pablo.cs.uiuc.edu/Projects/Pablo/`
Vampir	monitoring library performance visualisation	`http://www.pallas.de/pages/vampir.htm`
VT	monitoring library performance analysis visualisation tool for IBM SP machine	`http://www.ibm.com/`
Dimemas	performance prediction for message passing programs	`http://www.pallas.com/pages/dimemas.htm`
Paraver	program visualisation and analysis	`http://www.cepba.upc.es/paraver/`

- Solstice *SyMon* from Sun Microsystems allows standalone workstations to be monitored. *SyMon* uses client-server technology for monitoring such a workstation[66]. The Node Status Reporter (NSR) provides a standard mechanism for measurement and access to status information of clusters[67]. Parallel applications/tools can access NSR through the NSR Interface.
- PARMON[68] is a comprehensive environment for monitoring large clusters. It uses client-server technology to provide transparent access to all nodes to be monitored. The two major components of PARMON are the *parmon-server* - system resource activities and utilisation information provider and the *parmon-client* - a Java applet capable of gathering and visualising realtime cluster information.

REPRESENTATIVE CLUSTER SYSTEMS

There are many projects[69,70], investigating the development of supercomputing class machines using commodity off-the-shelf components. They include;

- The Networks of Workstations (NOW) project at University of California, Berkeley.
- The High Performance Virtual Machine (HPVM) project at University of Illinois at Urbana-Champaign.
- The Beowulf Project at the Goddard Space Flight Center, NASA;
- The Solaris-MC project at Sun Labs, Sun Microsystems, Inc., Palo Alto, CA.

Networks Of Workstations (NOW)

The Berkeley Network of Workstations[18] (NOW) project demonstrates how a successful large-scale parallel computing systems can be put together with volume produced commercial workstations and the latest commodity switch-based network components. To attain the goal of combining distributed workstations into a single system, the NOW project included research and development into network interface hardware, fast communication protocols, distributed file systems, distributed scheduling and job control. The Berkeley NOW system consists of the following five packages:

Inter-processor Communications

Active Messages (AM) are the basic communications primitives in Berkeley NOW. This interface generalises previous AM interfaces to support a broader spectrum of applications such as client/server programs, file systems, operating systems, as well as continuing support for parallel programs. The AM communication is essentially a simplified remote procedure call that can be implemented efficiently on a wide range of hardware.

Berkeley NOW includes a collection of low-latency, parallel communication primitives. These include extensions of current ones, including Berkeley Sockets, Fast Sockets, shared address space parallel C (Split-C), MPI and a version of HPF. These communications layers provide the link between the large existing base of MPP programs and the fast, but primitive, communication layers of Berkeley NOW.

Process Management

GLUNIX (Global Layer UNIX) is an operating system layer for Berkley NOW. GLUNIX is designed to provide transparent remote execution, support for interactive parallel and sequential jobs, load balancing, and backward compatibility for existing application binaries. GLUNIX is a multi-user system implemented at the user-level so that it can be easily ported to a number of different platforms. GLUNIX aims to provide:

A cluster-wide namespace, GLUNIX uses Network PIDs (NPIDs) and Virtual Node Numbers (VNNs). NPIDs are globally unique process identifiers for both sequential and parallel programs throughout the system. VNNs are used to facilitate communications among processes of a parallel program. A suite of user tools for interacting and manipulating NPIDs and VNNs, equivalent to UNIX run, kill, make, tcsh, up and stat. A programming API through the GLUNIX runtime library, which allows interaction with NPIDs and VNNs.

Virtual Memory

On a fast network with AM, fetching a page from a remote machine's main memory can be more than an order of magnitude faster than getting the same amount of data from the local disk. The Berkeley NOW system is able to utilise the memory on idle machines as a paging device for busy machines. The designed system is serverless, and any machine can be a server when it is idle, or a client when it needs more memory than physically available. Two prototype global virtual memory systems have been developed within the Berkeley NOW project to allow sequential processes to page to memory of remote idle nodes. One of these uses custom Solaris segment drivers to implement an external user-level pager which exchanges pages with remote page daemons. The other provides similar operations on similarly mapped regions using signals.

File System

xFS is a serverless, distributed file system, which attempts to have low latency, high bandwidth access to file system data by distributing the functionality of the server among the clients. The typical duties of a server include maintaining cache coherence, locating data, and servicing disk requests. The function of locating data in xFS is distributed by having each client responsible for servicing requests on a subset of the files. File data is striped across multiple clients to provide high bandwidth.

The High Performance Virtual Machine (HPVM)

The goal of the HPVM project[19,71], is to deliver supercomputer performance on low-cost COTS systems. HPVM also aims to hide the complexities of a distributed system behind a clean interface. The HPVM project, being undertaken by the Concurrent System Architecture Group (CSAG) in the Dept. of Computer Science at the University of Illinois at Urbana-Champaign, provides software that enables high-performance computing on clusters of PCs and workstations. The HPVM architecture is shown in Figure 2 and it consists of a number of software components with high-level APIs, such as MPI, SHMEM and Global Arrays, that allows HPVM clusters to be competitive with dedicated MPP systems.

The HPVM project aims to address the following challenges:

- Delivering high-performance communication to standard, high-level APIs
- Co-ordinating scheduling and resource management
- Managing heterogeneity

A critical part of HPVM was the development of a high-bandwidth and low-latency communications protocol known as Illinois Fast Messages (FM). The FM interface is based on Berkeley Active Messages. Unlike other messaging layers, FM is not the surface API, but the underlying semantics. FM contains functions for sending long and short messages and for extracting messages from the network. The services provided by FM guarantees and controls the memory hierarchy that FM provides to software built with FM. FM also guarantees reliable and ordered packet delivery as well as control over the scheduling of communication work.

The FM interface was originally developed on a Cray T3D and a cluster of SPARCstations connected by Myrinet hardware. Myricom's Myrinet hardware is a programmable network interface card capable of providing 160 MBytes/s links and with switch latencies of under a μs. FM has a low-level software interface that delivers hardware communication performance; however, higher-level layers interface offer greater functionality, application portability and ease of use.

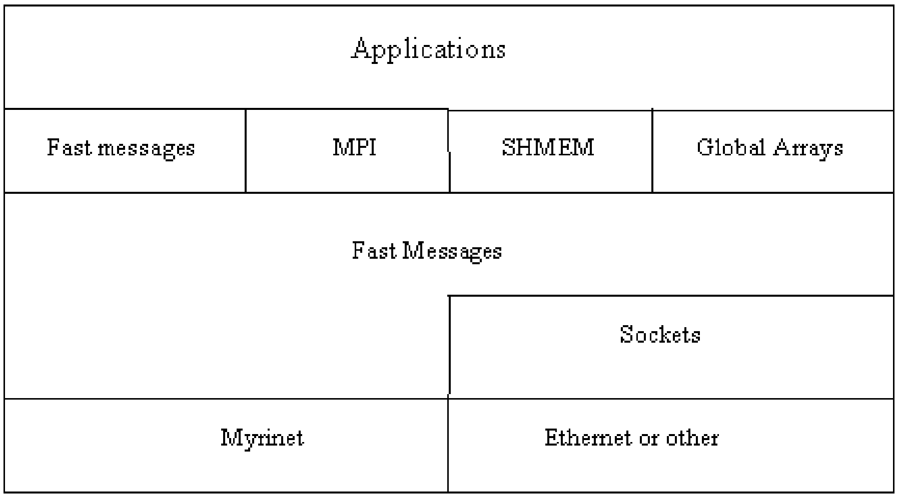

Figure 2. The HPVM Layered Architecture

The Beowulf Project

The Beowulf project[72] was initiated in the summer of 1994 under the sponsorship of the US NASA HPCC Earth and Space Sciences (ESS) project. The aim of the project was to investigate the potential of PC clusters for performing computational tasks.

Beowulf refers to a Pile-of-PCs (PoPC) to describe a loose ensemble or cluster of PCs which is similar to Cluster or Network Of Workstations (NOW). An emphasis of PoPC is the use of mass-market commodity components, dedicated processors (rather than stealing cycles from idle workstations), and the usage of a private communications network. An overall goal of Beowulf is to achieve the 'best' overall system cost/performance ratio for the cluster.

In the taxonomy of parallel computers, Beowulf clusters fall somewhere between tightly coupled systems like the Cray T3E and cluster architectures such as Berkeley NOW. Programmers using a Beowulf still need to worry about locality, load balancing, granularity, and communication overheads in order to obtain the best performance.

Beowulf adds to the PoPC model by emphasising a number of other factors into the PoPC equation:

- *No custom components* - Beowulf exploits the use of commodity components and industry standards which have been developed under competitive market conditions and are in mass production. A major advantage of this approach is that no single vendor owns the right to the product as the essentially identical subsystem components such as motherboards, peripheral controllers and I/O devices can be multi-sourced. These subsystems provide accepted, standard interfaces such as PCI bus, IDE and SCSI interfaces, and Ethernet communications.
- *Incremental growth and technology tracking* - As new PC technologies become available. The commodity approach to Beowulf also means that its systems administrator has total control over configuration of the cluster and so they may choose which one may 'best' suit their applications needs, rather than being restricted to vendor-based configurations.
- *Usage of readily available and free, software components* - Beowulf uses the Linux OS for

performance, availability of source code, device support, and wide user acceptance. Linux is distributed with X windows, most popular shells, and standard compilers for the most popular programming languages. Both PVM and MPI are available for Linux as well as a variety of distributed-memory programming tools.

Grendel Software Architecture

The collection of software tools being developed and evolving within the Beowulf project is known as *Grendel*. These tools are for resource management and to support distributed applications. The Beowulf distribution includes several programming environments and development libraries as separate packages. These include PVM, MPI, and BSP, as well as, SYSV-style IPC and `pthreads`.

Key to the success of the Beowulf environment is inter-processor communications bandwidth and system support for parallel IO. The communication between processors on a Beowulf cluster is achieved through standard TCP/IP protocols over the Ethernet internal to cluster. The performance of inter-processor communications is, therefore, limited by the performance characteristics of the Ethernet and the system software managing for message passing. Beowulf has been used to explore the feasibility of employing multiple Ethernet networks in parallel to satisfy the internal data transfer bandwidths required. Each Beowulf workstation has user-transparent access to multiple parallel Ethernet networks. This architecture was achieved by 'channel bonding' techniques implemented as a number of enhancements to the Linux kernel. The Beowulf project has shown that up to three networks could be ganged together to obtain significant throughput, thus validating the usage of channel bonding technique. New commodity network technologies, such as Fast Ethernet, will ensure that even better inter-processor communications performance will be achieved in the future.

In the Beowulf scheme, as is common in NOW clusters, every node is responsible for running its own copy of the kernel. However, in the interests of presenting a uniform system image to both users and applications, Beowulf has extended the Linux kernel to allow a loose ensemble of nodes to participate in a number of global namespaces. Normal UNIX processes 'belong' to the kernel running them and have a unique identifier process ID (PID). In a distributed scheme it is often convenient for processes to have a PID that is unique across an entire cluster, spanning several kernels.

Beowulf implements two Global Process ID (GPID) schemes. The first is independent of external libraries. The second, GPID-PVM, is designed to be compatible with PVM Task ID format and use PVM as its signal transport. The traditional UNIX calls `kill()` and `getpid()` work transparently with both schemes.

While the GPID extension is sufficient for cluster-wide control and signaling of processes, it is of little use without a global view of the processes. To this end, the Beowulf project is developing a mechanism that allows unmodified versions of standard UNIX process utilities (`ps` and `top`) to work across a cluster.

Programming Models

Beowulf supports several distributed programming paradigms. The most commonly used are the message passing environments, PVM, MPI and BSP. In the near future, a distributed shared memory package will be available also. Beowulf systems can take advantage of a number of libraries written to provide parallel filesystem interfaces to NOWs. MPI-IO is expected to become core software for new applications.

Solaris MC: A High Performance Operating System for Clusters

Solaris MC[73] (Multi-Computer) is a distributed operating system for a multi-computer, a cluster of computing nodes connected by a high-speed interconnect . It provides a single system image, making the cluster appear like a single machine to the user, to applications, and to the network. Solaris MC is built as a globalisation layer on top of the existing Solaris kernel, as shown in Figure 3. It extends operating system abstractions across the cluster and preserves the existing Solaris ABI/API, and hence runs existing Solaris 2.x applications and device drivers without modifications. Solaris MC consists of several modules: C++ and support for object framework, globalised processes and file system, and networking .

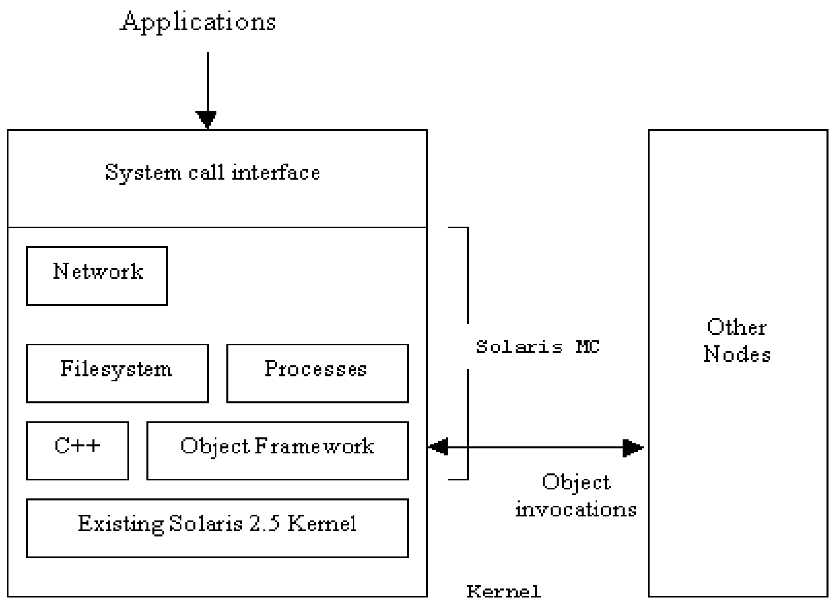

Figure 3. The Solaris MC Architecture

The interesting features of Solaris MC include the following:

- Extends existing Solaris operating system
- Preserves the existing Solaris ABI/API compliance
- Provides support for high availability
- Uses C++, IDL, CORBA in the kernel
- Leverages Spring technology

Solaris MC uses object-oriented framework for communication between nodes. The object-oriented framework is based on CORBA and provides remote object method invocations. It looks like a standard C++ method invocation to the programmers. The framework also provides object reference counting: notification to object server when there are no more references (local/remote) to the object. Another feature of Solaris MC object framework is that it supports multiple object handlers.

A key component in proving a single system image in Solaris MC is a global file system. It provides consistent access from multiple nodes to files and file attributes. It uses caching for high performance and uses a new distributed file system called ProXy File System (PXFS), which provides globalised file system without the need for modifying the existing file system.

The second important component of Solaris MC supporting a single system image is its globalised process management. It globalises process operations such as signals. It also globalises the /proc file system providing access to process state for commands such as ps and for the debuggers. It supports remote execution, which allows it to start up new processes on any node in the system.

Solaris MC also globalises its support for networking and I/O. It allows more than one network connection and provides support to multiplex between arbitary network links.

A Comparison of the Four Cluster Environments

The cluster projects described in the above sections share a common goal of attempting to provide a unified resource out of interconnected PCs or workstations. Each system claims that it is capable of providing supercomputing resources from common-off-the-shelf components. Each project provides these resources in different ways, both in terms of how the hardware is connected together and the way the system software and tools provide the services for parallel applications.

Table II shows the key hardware and software components that each system uses. Beowulf and HPVM are capable of using any PC, whereas Berkeley NOW and Solaris MC function on platforms where Solaris is available - currently PCs, Sun workstations and various clones systems. Berkeley NOW and HPVM use Myrinet with a fast, low-level communications protocol (Active and Fast Messages). Beowulf uses multiple standard Ethernet and Solaris MC uses NICs which are supported by Solaris - these range from Ethernet to ATM and SCI.

Table II. Cluster Systems Comparison

Project	**Platform**	**Communications**	**OS**	**Other**
`Beowulf`	PCs	Multiple Ethernet with TCP/IP	Linux and Grendel	MPI/PVM, Sockets and HPF
`Berkeley NOW`	Solaris-based PCS and workstations	Myrinet and Active Messages	Solaris + GLUUNIX + XFs	AM, PVM/MPI, HPF and Split-C
`HPVM`	PCs	Myrinet with Fast Messages	NT or Linux connection and global resource manager + LSF	Java-frontend, FM, Sockets, Global Arrays, SHMEM and MPI
`Solaris MC`	Solaris-based PCs and workstations	Solaris-supported	Solaris + Globalisation layer	C++ and CORBA

Each system consists of some middleware interfaced into the OS kernel, which is used to provide a globalisation layer, or unified view of the distributed cluster resources. Berkeley NOW and Solaris MC use the Solaris OS. Whereas Beowulf uses Linux with a modified kernel and HPVM is available for both Linux and Windows NT.

All four systems provide a rich variety of tools and utilities commonly used to develop, test and run parallel applications. These include various high-level APIs for message passing and shared-memory programming.

Summary and Conclusions

In this paper we have briefly discussed the different hardware and software components that are commonly used in the current generation cluster-based systems. We have also described four state-of-the-art projects that are using subtly different approaches ranging from an all COTS approach through to a mixture of technologies.

It was never an objective of this paper to try and determine the ‘best’ technologies, software or cluster computing projects. Rather, the goal of this paper was to lead the reader through the range of hardware and software technologies available and then briefly discuss a number of cluster projects that are using these technologies. The reader should be able to draw their own conclusions about which particular cluster project would suit their needs.

Hardware and Software Trends

In the last five years several important advances have taken place and prominent among these are:

- A network performance increase of 10 fold (10x) with 100BaseT Ethernet with full duplex support.
- The availability of switched network circuits, including full crossbar switches for proprietary network technologies such as Myrinet.
- Workstation performance has significantly improved and now appears like yesterday’s supercomputers.
- Improvements in microprocessor performance has led to the availability of desktop PCs with the performance of low-end workstations, but at a significantly lower cost.
- The availability of a powerful and stable UNIX-like operating systems (Linux) for x86 class PCs with source code access.

Culler and Singh[76] quantify a number of hardware trends in their book *Parallel Computer Architecture: A Hardware/Software Approach*. Foremost of these is the design and manufacture of microprocessors. A basic advance is the decrease in feature size which enables circuits to become either faster or lower in power consumption. In conjunction with this is the growing die size that can be manufactured. These factors mean that:

- The average number of transistors on a chip is growing by about 40% per annum
- The clock frequency growth rate is about 30% per annum

It is anticipated that by early 2000 AD there will be 700 MHz processors with about 100 million transistors.

There is a similar story for storage but the divergence between memory capacity and speed is more pronounced. Memory capacity increased by three orders of magnitude (1000x) between 1980 and 1995, yet its speed has only doubled (2x). It is anticipated that Gigabit DRAM (128 MByte) will be available in early 2000, but the gap to processor speed is getting greater all the time.

The problem is that memories are getting larger whilst processors are getting faster. So getting access to data in memory is becoming a bottleneck. One method of overcoming this bottleneck is to configure the DRAM in banks and then transfer data from these banks in parallel. In addition, multi-level memory hierarchies organised as caches make memory access more effective, but their effective and efficient design is complicated. The access bottleneck also applies to disk access, which can also take advantage of parallel disks and caches.

The performance of network interconnects is increasing day-by-day with the reduction of costs. The use of networks such as ATM, SCI and Myrinet in clustering for parallel processing appears promising. This has been demonstrated by many commercial and academic projects such as Berkeley's NOW and Beowulf. But no single network interconnect has emerged as a clear winner. Myrinet is not a commodity product and costs a lot more than Ethernet, but has real advantages over it: very low-latency, high bandwidth, and a programmable on-board processor allowing for greater flexibility. SCI network has been used to build distributed shared memory system, but lacks scalability. ATM is used in clusters that are mainly used for multimedia applications.

Two of the most popular operating systems of the 90s are Linux and NT. Linux has become a popular alternative to a commercial operating system due to its free availability and superior performance compared to other desktop operating systems such as NT. Linux has more than 7millions users currently worldwide and it is has become the researcher's choice of operating system. Linux is also available for multiprocessor machines with up to eight processors.

NT has large installed base and it has almost become a ubiquitous operating system. NT 5 will have a thinner and faster TCP/IP stack, which supports faster communication of messages, yet using standard communication technology. NT systems for parallel computing are in a similar situation to UNIX workstations 5/7 years ago and it is only a matter of time before NT catches up. NT developers benefit from the time and money invested in research by the UNIX community as it seems that they are borrowing many of their technologies.

Cluster Technology Trends

We have discussed a number of cluster projects within this paper. These range from those which are proprietary-based (Solaris) through to a totally commodity-based system (Beowulf). HPVM can be considered a hybrid-system using commodity computers and specialised network interfaces. It should be noted that the projects detailed in this paper are a few of the most popular and well known, rather than an exhaustive list of all those available.

All the projects discussed claim to consist of commodity components. Although this is true, one could argue that true commodity technologies would be those that are pervasive at most academic or industrial sites. If this were the case, then true commodity would mean PCs running Windows 95/98 with standard 10 Mbps Ethernet. However, when considering parallel applications with demanding computational and network needs, then this type of low-end cluster would be incapable of providing the resources needed.

Each of the projects discussed tries to overcome the bottlenecks associated with using cluster-based systems for demanding parallel applications in a slightly different way. Without fail, however, the main bottleneck is not the computational resource (be it a PC or UNIX workstation), rather it is the provision of a low-latency, high-bandwidth interconnect and an efficient low-level communications protocol to provide high-level APIs.

The Beowulf project explores the usage of multiple standard Ethernet cards to overcome the communications bottleneck, whereas Berkeley NOW and HPVN use programmable Myrinet NICs and AM/FM communications protocols. Solaris MC uses standard high-performance NICs and TCP/IP. The choice of what is the best solution cannot just be based on performance, the cost per node to provide the NIC should also be considered. For example, a standard Ethernet card costs less than \$100, whereas Myrinet cards cost in excess of \$1000 each. Another factor that must also be considered in this equation is the availability of Fast Ethernet and the advent of Gigabit Ethernet. It seems that Ethernet technologies are more likely to be main stream, mass produced and consequently cheaper than specialised network interfaces. As an aside, all the projects that have been discussed are in the vanguard of the cluster computing revolution and their research is helping the following army determine which are the best techniques and technologies to adopt.

Some Predictions about the Future

Emerging hardware technologies along with maturing software resources mean that cluster-based systems are rapidly closing the performance gap with dedicated parallel computing platforms. Without doubt the gap will continue to close.

Cluster systems that scavenge idle cycles from PCs and workstations will continue to use whatever hardware and software components that are available on public workstations. Clusters dedicated to high performance applications will continue to evolve as new and more powerful computers and network interfaces become available in the market place.

It is likely that individual cluster nodes will comprise of multiple processors. Currently two and four processor PCs and UNIX workstations are becoming fairly common. Software that allows SMP nodes to be efficiently and effectively used by parallel applications will no doubt be developed and added to the OS kernel in the near future. It is likely that there will be widespread usage of Fast and Gigabit Ethernet and as such they will become the *de facto* networking technology for clusters. To reduce message passing latencies cluster software systems will by-pass the OS kernel, thus avoiding the need for expensive system calls, and exploit the usage of intelligent network cards. The OS used on future clusters will be determined by its ability to provide a rich set of development tools and utilities as well as the provision of robust and reliable services. UNIX-based OSs are likely to be most popular, but the steady improvement and acceptance of Windows NT will means that it will be not far behind.

Final Thoughts

Our need for computational resources in all fields of science, engineering and commerce far outstrip our ability to fulfil these needs. The usage of clusters of computers is, perhaps, one of the most promising means by which we can bridge the gap between our needs and the available resources. The usage of a COTS-based cluster system has a number of advantages including:

- Price/performance when compared to a dedicated parallel supercomputer.
- Incremental growth that often matches yearly funding patterns.
- The provision of a multi-purpose system: one that could, for example, be used for secretarial purposes during the day and as a *commodity parallel supercomputer* at night.

These and other advantages will fuel the evolution of cluster computing and its acceptance as a means of providing commodity supercomputing facilities.

ACKNOWLEDGEMENTS

We would like to thank Rose Rayner, John Rosbottom, Toni Cortes, and Lars Rzymianowicz for their efforts proof reading this paper.

REFERENCES

1. G. Pfister, *In Search of Clusters*, Prentice Hall PTR, 2nd Edition, 1998, ISBN 0-13-899709-8
2. M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra *MPI: The Complete Reference (Vol. 1) - 2nd Edition* The MIT Press, September 1998, ISBN 0-262-69215-5
3. C. Koelbel, D. Loveman, R. Schreiber, G. Steele Jr., and M. Zosel, *The High Performance Fortran Handbook*, The MIT Press, 1994.

4. M. Baker, G. Fox and H. Yau, *Review of Cluster Management Software*, NHSE Review, May 1996, - http://nhse.cs.rice.edu/NHSEreview/CMS/
5. L. Turcotte, *A Survey of Software Environments for Exploiting Networked Computing Resources*, Engineering Research Center for Computational Field Simulation, Mississippi State, 1993
6. T. Anderson, D. Culler, and D. Patterson, *A Case for NOW (Network of Workstations)*. IEEE Micro, 15(1):54-64, February 1995.
7. R. Buyya (Editor), *High Performance Cluster Computing: Systems and Architectures*, Volume 1, Prentice Hall PTR, NJ, 1999.
8. Jazznet: A dedicated cluster of Linux PCs - http://math.nist.gov/jazznet/
9. Gardens Project - http://www.fit.qut.edu.au/˜szypersk/Gardens/
10. M. Mutka and M. Living, *Profiling Workstations' Available Capacity for Remote Execution*, Performance'87, Procs. of the 12th IFTP WG7.3 Symposium on Computer Performance, Brussels, December, 1987.
11. The Berkeley Intelligent RAM Project. http://iram.cs.berkeley.edu/
12. The Standard Performance Evaluation Corporation (SPEC), http://open.specbench.org
13. Intel's Pentium II Xeon Processor, http://www.tomshardware.com/xeon.html
14. Russian Academy of Sciences, VLSI Microprocessors: A Guide to High Performance Microprocessors, http://www.microprocessor.sscc.ru/
15. *The Enhanced IDE FAQ*, J. Wehman and P. Herweijer.
16. *SCSI-1: Doc # X3.131-1986*, ANSI, 1430 Broadway, NY, USA
17. C. Ruemmler and J. Wilkes, J. Modelling Disks. Tech. Rep. HPL9368, Hewlett Packard Labs., July 1993
18. Berkeley NOW - http://now.cs.berkeley.edu/
19. HPVM - http://www-csag.cs.uiuc.edu/projects/clusters.html
20. Interconnect Performance page - http://www.scl.ameslab.gov/Projects/ClusterCookbook/icperf.html
21. *PCI BIOS Specification*, Revision 2.1, August 1994, PCI SIG, Hillsboro, Oregon.
22. *CSMA/CD (Ethernet)*, ISO/IEC 8802-3:1993 (ANSI 802.3), 1993
23. *ATM User-Network Interface Specification*, ATM Forum/Prentice Hall, September 1993, ISBN 0-01-225863-3.
24. *Scalable Coherent Interface (SCI)*, IEEE 1596-1992, August 1993
25. R. Clark and K. Alnes, *An SCI Interconnect Chipset and Adapter*, In Symposium Record, Hot Interconnects IV, pages 221-235, August 1996.
26. SCI Association - http://www.SCIzzL.com/
27. HCS Lab. University of Florida - http://www.hcs.ufl.edu/sci.html
28. Myricom, Inc. - http://www.myri.com/
29. MPI-FM: MPI for Fast Messages - http://www-csag.cs.uiuc.edu/projects/comm/mpi-fm.html
30. N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seisovic, and W. Su, *Myrinet - A Gigabit-per-Second Local-Area Network*, IEEE Micro, Vol. 15, No.1, February 1995.
31. Linux Meta-FAQ - http://sunsite.unc.edu/mdw/linux.html
32. Microsoft Corporation - http://www.microsoft.com
33. A. Watts, *High Command*, PC Direct, December 97
34. The Microsoft Market Share - http://newsport.sfsu.edu/ms/markets.html
35. H. Custer, *Inside Windows NT* , Microsoft Press, 1993, ISBN 1-55615-481-X
36. Windows NT Server - http://www.microsoft.com/ntserver/
37. MPI Forum - http://www.mpi-forum.org/docs/docs.html
38. A. Beguelin, J Dongarra, G. Geist, R. Manchek, and V. Sunderam, *The PVM project*, Technical Report, Oak Ridge National Laboratory, February 1993.
39. *Message Passing Interface Forum*, MPI: A Message-Passing Interface Standard, May 5, 1994, University of Tennessee, Knoxville, Report No. CS-94-230
40. mpiJava - http://www.npac.syr.edu/projects/prpc/mpiJava/, August 1998.
41. MPICH - http://www.mcs.anl.gov/mpi/mpich/

42. W. Gropp, E. Lusk, N. Doss, and A. Skjellum, *A high-performance, portable implementation of the MPI message passing interface standard* - `http://www.mcs.anl.gov/mpi/mpicharticle/paper.html`
43. W. Gropp and B. Smith, *Chameleon parallel programming tools users manual*. Technical Report ANL-93/23, Argonne National Laboratory, March 1993.
44. MPI Implementations - `http://www.mpi.nd.edu/MPI/`
45. M.A.Baker, *MPI on NT: The Current Status and Performance of the Available Environments*, EuroPVM/MPI98, LNCS Vol 1497, Springer Verlag, pp 63 - 75, September 1998, ISBN 3-540-65041-5
46. R. Butler and E. Lusk, *User's Guide to the p4 Parallel Programming System*. ANL-92/17, Argonne National Laboratory, October 1992.
47. R. Butler and E. Lusk, *Monitors, messages, and clusters: The p4 parallel programming system*. Parallel Computing, 20:547-564, April 1994.
48. S. Parkin, V. Karamcheti and A. Chein, *Fast-Messages (FM): Efficient, Portable Communication for Workstation Clusters and Massively-Parallel Processors*, IEEE Microprocessor Operating Systems, April - June, 1997, pp 60 -73.
49. T. von Eicken, D. Culler, S. Goldstein and K. Shauser, *Active Messges: a mechanism for integrated communications and computation*, Procs of International Symposium on Computer Architectures, 1992.
50. J. Protic, M. Tomasevic, and V. Milutinovic. *Distributed Shared Memory: Concepts and Systems*, Tutorial delivered at the International Conference on HPC, Seoul, Korea, April 28 - May 2, 1997.
51. K Li and P Hudak. *Memory coherence in Shared Virtual Memory systems*. ACM TTOCS, November 1989.
52. S Hiranandani, K Kennedy, and C. Tseng, *Compiling Fortran D for MIMD distributed memory machines* CACM, August 1992.
53. S. Dwarkadas, A. Cox, and W. Zwaenepoel. *An Integrated Compile-Time/Run-Time Software Distributed Shared Memory System*, Operating System Review, December 1996.
54. W Zwaenepoel, J. Bennett and J. Carter, *Munin: DSM Using Multi-Protocol Release Consistency*, Operating Systems of the 90s and Beyond, Eds.A.Karshmer and J.Nehmer, Springer-Verlag LNCS 563, pp. 56-60, 1991.
55. TreadMarks - `http://www.cs.rice.edu/~willy/TreadMarks/overview.html`
56. N. Carriero and D. Gelernter, *Linda in context*, Communications of the ACM, 32(4): 444-458, April 1989.
57. P. Dasgupta and R. C. Chen, et. al., *The Design and Implementation of the Clouds Distributed Operating System*, Technical Report,(Computing Systems Journal, Vol 3, USENIX, Winter 1990), `ftp://helios.cc.gatech.edu/pub/papers/design.ps.Z`
58. Lenoski, D. Laudon J, et al., *The Stanford DASH Multiprocessor*, IEEE Computer, Vol. 25, No. 3, March 1992.
59. S. Franks, Burkhardt III, J. Rothnie, *The KSR1: Bridging the Gap Between Shared Memory and MPPs*, COMPCON93. February 1993.
60. C. Mapples and L. Wittie, *Merlin: A Superglue for Multiprocessor Systems*, CAMPCON'90, March 1990.
61. S. Browne, *Cross-Platform Parallel Debugging and performance Analysis Tools*, Proceeding of the EuroPVM/MPI98 Workshop, Liverpool University, September 1998.
62. Parallel Tools Consortium project - `http://www.ptools.org/`
63. Dolphin Interconnect Solutions - `http://www.dolphinics.no/`
64. E. Anderson and D. Patterson, *Extensible, Scalable Monitoring for Clusters of Computers*, Proceedings of the 11th Systems Administration Conference (LISA '97), October 26-31, 1997, San Diego, California, USA.

65. P. Uthayopas, C. Jaikaew, and T. Srinak, *Interactive Management of Workstation Clusters Using World Wide Web*, Cluster Computing Conference-CCC 1997, Proceedings, http://www.mathcs.emory.edu/~ccc97/
66. Sun Microsystems, *Solstice SyMON 1.1 User's Guide*, Palo Alto, CA 1996.
67. C. Roder, T. Ludwig, and A. Bode, *Flexible Status Measurement in Heterogeneous Environment*, Proceedings of the 5^{th} International Conference on Parallel and Distributed Processing, Techniques and Applications (PDPTA'98), CSREA Publishers, Las Vegas, USA, 1998.
68. Rajkumar, Krishnamohan and Bindu, *PARMON: A Comprehensive Cluster Monitoring System*, The Australian Users Group for UNIX and Open Systems Conference and Exhibition, AUUG'98 - Open Systems: The Common Thread, Sydney, Australia, 1998
69. Cluster Computing links - http://www.tu-chemnitz.de/informatik/RA/cchp/
70. Computer Architecture links - http://www.cs.wisc.edu/~arch/www/
71. S. Parkin, M. Lauria, A. Chien, et. al, *High Performance Virtual Machines (HPVM): Clusters with Supercomputing APIs and Performance*, 8^{th} SIAM Conference on Parallel Processing for Scientific Computing (PP97); March, 1997.
72. The Beowulf Project - http://www.beowulf.org/
73. Solaris MC - http://www.sunlabs.com/research/solaris-mc/
74. Y. Khalidi, J. Bernabeu, V. Matena, K. Shirriff and M Thadani, *Solaris MC: A Multi-Computer OS*, 1996 USENIX Conference, January 1996.
75. K. Shirriff, *Next Generation Distributed Computing: The Solaris MC Operating System*, Japan Talk,1996, http://www.sunlabs.com/research/solaris-mc/doc/japantalk.ps
76. D. Culler and J. Singh, *Parallel Computer Architecture: A Hardware/Software Approach*, - http://www.cs.berkeley.edu/~culler/book.alpha/