A Survey of Software Environments for Exploiting Networked Computing Resources

by
Louis H. Turcotte

June 11, 1993

Engineering Research Center for Computational Field Simulation
P.O. Box 6176
Mississippi State, MS 39762

Preface

This report presents a survey of existing and evolving software products which can be used to more effectively exploit networked computing resources.

Funds for this investigation were provided by the Information Technology Laboratory (ITL), US Army Engineer Waterways Experiment Station (WES), Vicksburg, MS. The work was performed by Mississippi State University's Engineering Research Center for Computational Field Simulation[1] in Starkville, MS, under Contract No. DACW39-93-M-0978.

The work was monitored at WES by Dr. N. Radhakrishnan, Director, ITL.

During publication of this report, Dr. Robert W. Whalin was Director of WES. COL Leonard G. Hassell, EN, was Commander and Deputy Director.

[1]Author's address:
Louis H. Turcotte, PhD, PE
USAE Waterways Experiment Station
ATTN: CEWES-IM-MS
3909 Halls Ferry Road
Vicksburg, MS 39180-6199
(601) 634-4421
email: turcotte@wes.army.mil

Abstract

The rapid growth of interconnected high performance workstations has produced a new computing paradigm called clustered workstation computing. This report provides a background of the factors which motivate the implementation of workstation clusters, the characteristics of workstation clusters, and the present software systems which are available to effectively exploit these resources. Software for batch job distribution across both dedicated and enterprise networks and software to support parallelization across a cluster of workstations is surveyed. Several ancillary topics such as distributed computing tools, scientific libraries to support workstation clusters, remote procedure calls, software to support performance monitoring, software for debugging, and data coercion concepts are discussed. A general description of over sixty software systems related to workstation clustering is included.

Contents

List of Figures

List of Tables

1. Metacomputing

1.1. Introduction

During the last decade there has been an exponential growth in networked computing resources. This fact is reflected by the growth in registered systems connected to the Internet. The most recent status report from the Network Information Systems Center [232] summarizes this growth (see Figures 1.1 and 1.2). Over 725,000 hosts have been connected via approximately 17,000 domains in just ten years! The rapid growth of networked computers has been accompanied by an astonishing increase of computational power by these network attached computers. Microprocessors have doubled in performance approximately every eighteen months during the last decade and they continue to increase in performance at a much greater rate than supercomputers (see Figure 1.3) [38]. Cheong [106] summarized the five key technology areas which drive high performance scientific computing: microprocessors, networks, backplane buses, semiconductor main memory, and magnetic fixed disk. Cheong provided the following summation of each technology.

- Since 1985 the performance of CMOS-based microprocessors has quadrupled every threee years, or at the rate of 60% every year. Clock speeds alone have evolved from 200 kHz in 1971 to 50 MHz in 1991.[1]
- Local area networks have improved by a factor of 10 every decade. In 1980 ethernet operated at 10 Mbits/sec. In 1990 FDDI operated at 100 Mbits/sec. Early prototypes indicate that Gbits/sec networks will be commercially available by 2000.
- Computer backplane buses have improved by a factor of 10 every decade. Digital's Unibus operated at 2 Mbits/sec in 1970. Motorola's VME bus operated at 20 Mbits/sec in 1980. In 1990 several buses operated at 200 Mbits/sec.[2]
- Semiconductor memory chips have quadrupled in capacity every three years (annual rate of 60%) since 1972. The chronology on the number of bits per chip follows. 1K (1972), 4K (1975), 16K (1978), 64K (1981), 256K (1984), 1M (1987), 4M (1990).[3]

[1] Late 1992 saw the delivery of 130 MHz and 200 MHz DEC Alpha systems.

[2] IBM systems shipped in late 1992 have buses which operated at 640 Mbits/sec.

[3] Chips having 16M bits/chip should be widely in use by 1994, 64M bits/chip in 1996, and 256M bits/chip in 1998.

- Magnetic disk storage has evolved from a density of 1K bits per square inch (1957) to 1G bits per square inch in 1990 (annual rate of 26% per year, or doubling every three years).

This combined performance growth indicates that significant computational capability is available and interconnected. However, a majority of the computer resources (primarily powerful workstations) included in today's networks may lie idle for as much as 95% of the time [309].

The concept of metacomputing is relatively recent [269] and is reflective of the rapid advances in computer hardware and networks. Many terms have been coined to describe the concept of transparently linking multiple computer systems. For example, the terms "cluster computing", "distributed computing", "ensemble computing", "heterogeneous computing", "hypercomputing", "meta-system", "network-based concurrent computing", "piranha computing", "ultracomputing", and "virtual heterogeneous machine" have all been used to describe similar or closely related concepts of metacomputing.

This report represents a survey of emerging software technologies and philosophies related to workstation clusters. The last couple of years have seen the concept of metacomputing finally reach widespread acceptance as an effective alternative to other, more traditional computing paradigms. Gordon Bell's recent treatise on ultracomputing includes an eloquent presentation of the taxonomy of teracomputing [49]. Bell discusses the candidate computer architectures and their potential to achieve the national goal to produce teraflop systems by 1995. Networked workstation clusters were included in this taxonomy and reflect one multicomputer approach to the teracomputing goal (see Figure 1.4).

Many stable software products are now available to assist with improving/exploiting the utilization of networked computer resources. This report will survey several approaches and summarize the software products which can be employed. The remainder of Chapter 1 will present an overview of workstation clustering. Chapter 2 will discuss the use of dedicated workstation clusters as viable alternates to traditional mainframes and will present techniques which can be employed to more effectively exploit networked computer resources in enterprise networks. The use of workstation clusters for parallel processing is presented in Chapter 3. Chapter 4 presents several ancillary topics which are directly related to the topic of workstation cluster computing. An extensive bibliography has been included and provides numerous references to the large body of work related to the topic of metacomputing. Appendix A contains a glossary of terms which are specifically related to metacomputing. Finally, over sixty individual software products are summarized in Appendix B.

1.2. The NSF Metacenter

In 1992 the National Science Foundation (NSF) accepted a proposal to integrate the four NSF sponsored supercomputer centers into a single metacomputing facility [84, 201, 240, 260, 269]. The NSF report [269] documented the stages in national information infrastructure as:

a) 1985-1990, access to supercomputers and development of expertise,

b) 1991-1995, transition to parallel and distributed computing and the NREN network, and

c) 1996-2000, grand challenge research.

To effectively address the demands of stages (b) and (c) the centers proposed creating a national machine room, or metacenter. The metacenter would provide

- a single national advisory council,
- a single allocation mechanism for the four centers, and
- a coordinated acquisition program.

The central concept of the metacomputing proposal was that problems should be able to migrate to the appropriate computer architecture(s) without regard for where the computers are physically located. Additionally, the specific benefits of various architectures (scalar, vector, SIMD, MIMD, etc) should be leveraged to address complex grand challenge problems by providing the capability to implement distributed solutions in a transparent manner. Several components were identified as necessary to create the metacenter:

- high-speed, intercenter communication links,
- a national file system,
- common user interfaces to systems and information,
- an intercenter videoteleconferencing system, and
- a software support system for distributed applications.

Each of the NSF supercomputer centers has acquired a dedicated workstation cluster as part of their metacomputing initiative. The recognition of workstation clusters as a viable extension to more traditional computing architectures by the NSF supercomputing centers has provided the credibility required for widespread consideration of this computing paradigm.

1.3. Motivation for Workstation Clusters

The utilization of workstation clusters can yield many potential benefits. Some zealous pundits have even implied that workstation clusters are practical replacements for general purpose supercomputers. Nothing in this report is meant to imply that single workstations, or even a cluster of workstations, can or will, replace large $30M supercomputers. However, there are many applications executing on supercomputers which do not absolutely require supercomputing resources. Eugene Brooks [19] has identified two contrasting categories of demand for scientific applications:

capability demand: coping with megaproblems that require all the computational capabilities of any available system including memory as well as CPU. Grand challenge applications tend to fall into this bracket, which has historically been used as the principal justification for adoption of massively parallel processing (MPP).

capacity demand: requiring substantial but far from ultimate performance and making moderate demands upon memory. This class of job is large in total volume and is needed on a regular day-to-day basis. These jobs are good candidates for workstation clusters. (Brooks estimates that approximately 90% of Lawrence Livermore National Laboratory's workload fits this catagory.)

Advocates of workstation clusters [9, 245, 247, 367] have identified numerous benefits of workstation clusters. Some of the motivations for harnessing the computing power of workstation clusters which have been presented are

- the need to provide a significant number of cost effective CPU cycles to serve scientific applications with moderate computational requirements,
- the ever shrinking budgets which are being provided to meet the continuously growing demand for scientific computing resources (especially true of government agencies),
- the desire to exploit the price/performance benefits of RISC processors,
- the need to provide single processor interactive FLOPS during prime hours and also provide abundant batch FLOPS during nonprime hours,
- the need to provide inexpensive configurations to develop and explore parallel applications,
- the desire to utilize abundant workstation cycles which have been estimated to be idle 95% of the time without additional financial investment,
- the desire to rely on commodity, consumer based technology (e.g. workstations and networks) to minimize economic risk and provide an incremental growth path,
- the availability of large memories which are common on workstations,
- the desire to maintain the ease-of-use facilities which already exist in workstation environments,
- the realization that clusters offer the best features of sequential and parallel processing; speed of sequential processing and unlimited growth of computing power in parallel processing, and
- the desire to exploit the benefits of heterogeneous computing environments and thereby produce the most efficient computing solution.

Many of the items listed above require significant changes in the way we operate our computing resources. Additionally, several of these items may still require significant effort to reach mainstream acceptance.

1.4. Goals/Expectations

In the previous section the motivations which have driven the consideration of workstation clusters were summarized. The deployment of workstation clusters has lead to the realization that several goals/expectations are inherently linked to their success [53, 171, 245].

- Provide architecture/processor type transparency.
- Provide network/communication transparency.
- Provide parallelism transparency.
- Provide task location transparency.
- Provide fault transparency, both at the network level and workstation level.
- The use and programming of all resources and types of architectures available must be transparent.
- Must be easy-to-use.
- Must provide support for traditional high level languages.
- Must be capable of delivering increased performance from parallelization.
- Must not only accommodate heterogeneity, but must exploit it.
- Provide portable parallel processing environments.
- Must be possible to run each application on every type of machine in the cluster.
- Performance for parallel applications must scale with the inclusion of more workstations in the cluster.
- Must facilitate closer collaboration among researchers specializing in different parts of the model. In the future, specialists will develop and maintain state-of-the-art algorithms at geographically dispersed locations. New applications will be written to automatically "link" to these libraries. Therefore, an application will be assured of containing the best routines for accomplishing a computational task. High bandwidth networks and system transparency will mean there is no need to frequently download the most recent version of a software library.

Many of the goals/expectations listed above are years away. However, they provide a basis for insight into the environments which will form the backbone of scientific computing in the future.

1.5. Infrastructure Requirements

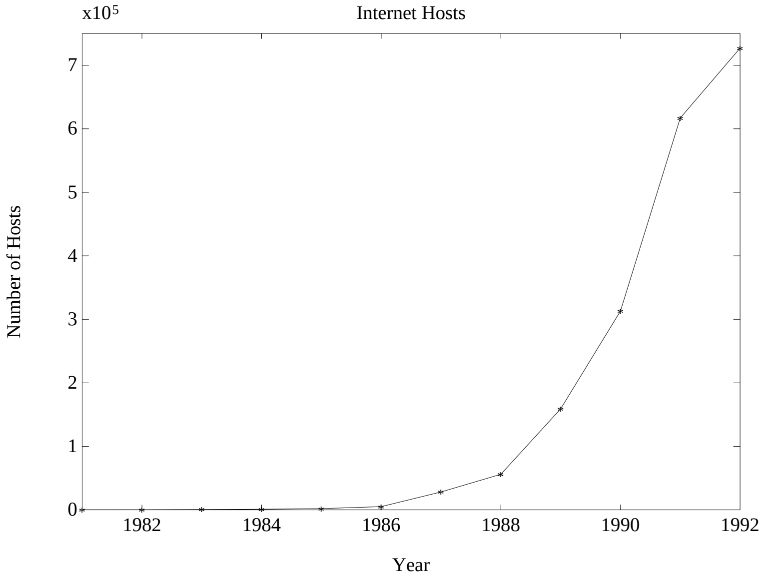
Workstation clusters provide unique infrastructure challenges. They may be geographically dispersed, have multiple owners, include multi-vendor hardware, and have various software differences (vendor enhanced operating systems for example). Some of the infrastructure requirements of workstation clusters are listed below [171, 210, 262].

- High bandwidth networks to support communications requirements (e.g. 100-800 Mbits/sec per host).
- Low latency communications mechanisms (e.g. 100-500 microsecs between hosts).
- Good scaling characteristics (e.g. 10-1000 hosts).
- Support for high-bandwidth multicast communications.
- Capability to automatically recover from network and node failures (e.g. fault tolerant).
- Standard low-level primitives for communications, synchronization, and scheduling across architectures.
- Heterogeneous remote procedure calls that hide architecture, protocol, and system differences.
- Real-time performance monitors.
- Reliable production batch job scheduler.
- Distributed application development tools.
- Support for traditional high level languages for heterogeneous computing.
- Applications which are capable of exploiting workstation clusters.
- New system administration tools to address system management issues for distributed computing resources.
- Development of standards which protect software investments.

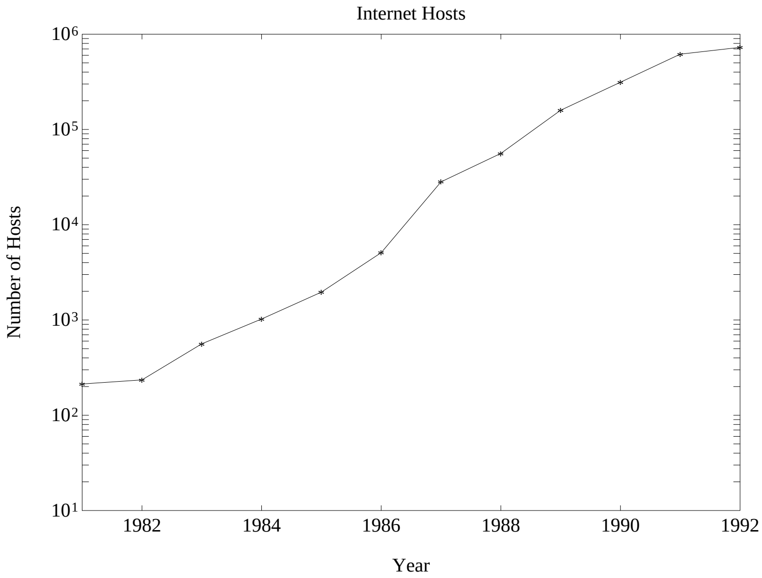
1.6. Barriers/Challenges

The items listed in the previous three sections summarize various characteristics of workstation clusters. There are, however, many impediments which must be addressed to achieve the benefits of clustering. Many of these challenges are being confronted by MPP vendors and are not unique to workstation clusters. Some of the challenges which confront clusters are stated below [171].

- The effort required to achieve speedup via parallelization (Should we simply wait for faster hardware).
- The unknown future of existing tools and their immature stage of development makes selecting platforms risky.
- The lack of standards.
- Open versus proprietary software (can it be addressed in a multi-vendor, heterogeneous environment).
- Heterogeneity (e.g. operating systems, network software, applications, hardware, etc.).
- The ownership and geographical dispersion issues of distributed clusters.
- The ability to ensure security and user authentication in distributed clusters.
- The need to provide accurate resource utilization accounting.
- The need for common, transparent programming tools (e.g. debuggers, performance analyzers, monitors, etc).
- The need for operating system and management software for control of distributed multi-vendor environments (e.g. user management, distributed program loading, load balancing, network management, dynamic reconfiguration, etc.)
- The need to use existing operating systems and software environments.
- The need for unified parallel and distributed programming models for various types of systems (e.g. single processor, shared memory multiprocessor, distributed multiprocessor, etc.).
- The need for software which transparently schedules resources, manages data coercion, communications, and synchronization across different platforms.



(a) Internet Hosts



(b) Internet Hosts (semilog plot)

Figure 1.1: Internet Host Growth in Last Decade

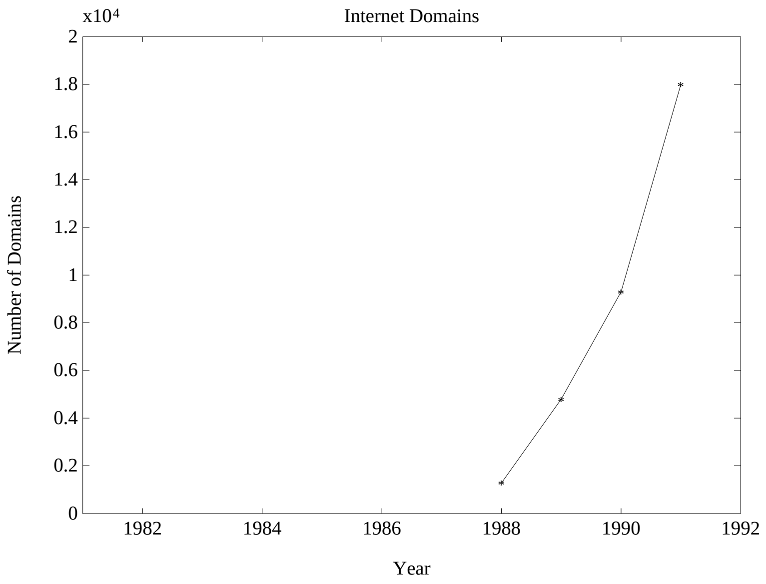
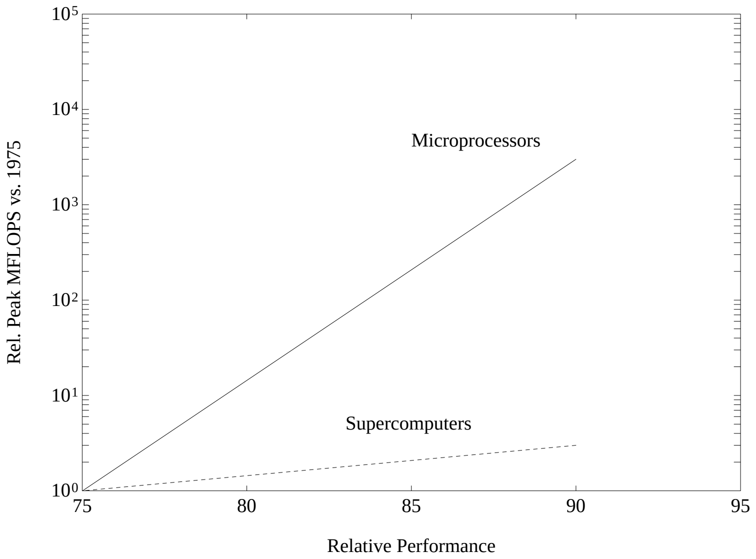
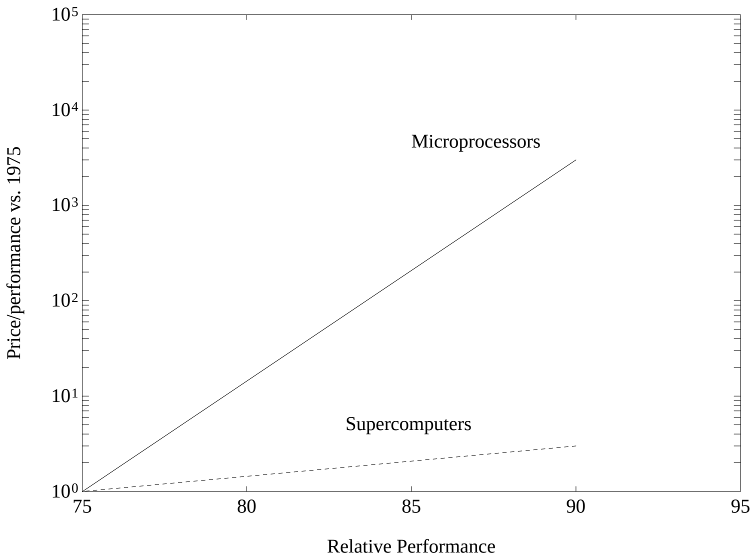


Figure 1.2: Internet Domain Growth in Last Decade



(a)



(b)

Figure 1.3: Improvement of Microprocessors vs. Supercomputers

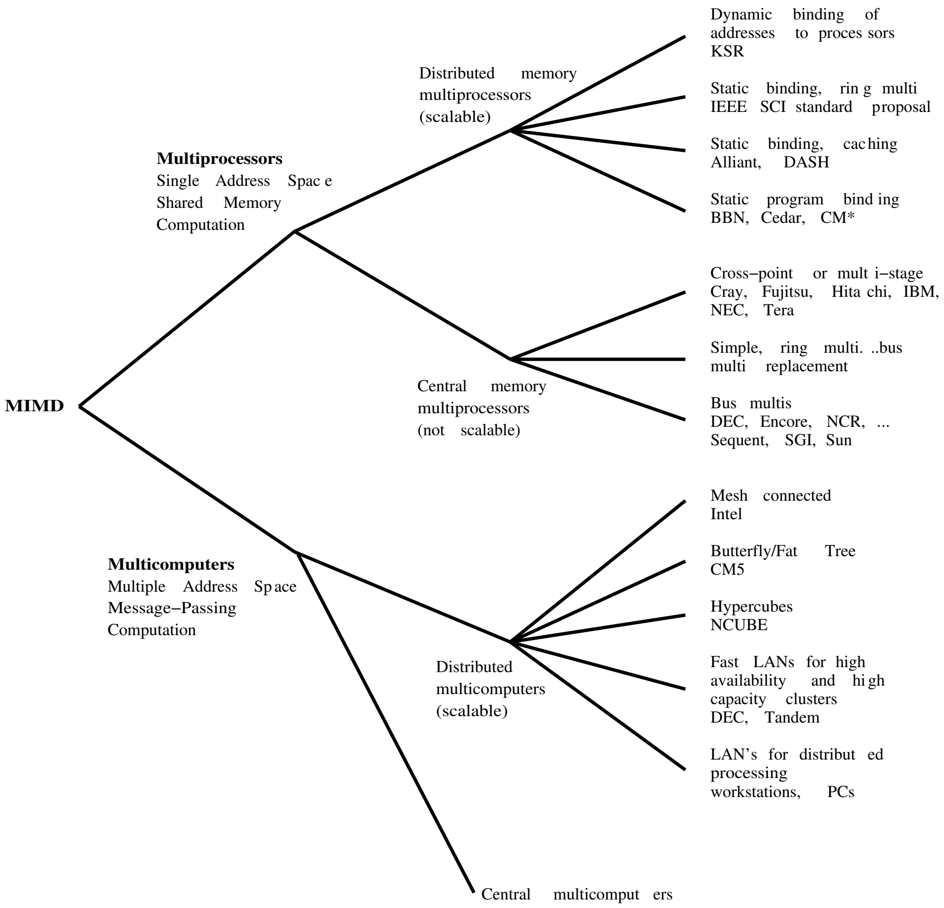


Figure 1.4: Taxonomy of Tera Computing

2. Dedicated and Enterprise Clusters

2.1. Dedicated Workstation Clusters

The price/performance characteristics offered by workstations have enticed a significant number of institutions to consider alternates to the traditional monolithic mainframe solution for interactive and batch jobs. "Dedicated" workstation clusters have been installed at several institutions as substitutes or replacements for traditional computing solutions. Clusters of workstations offer a cost effective method of delivering both interactive and batch CPU cycles [19, 367]. Dedicated workstation clusters may be characterized as follows:

- They are typically installed in 19 inch racks in a central computer room.
- They are homogeneous configurations (within the cluster unit).
- They are managed by a single group which administers the cluster like a central mainframe.
- They are frequently interconnected by networks with better performance than ethernet (e.g. FDDI, IBM's SOCC, etc).
- They are typically accessed only via a gateway, or frontend, system. The gateway system either attaches the user to an available single system or manages the queues which feed batch jobs to the cluster.

2.1.1. Networking

One of the obvious limitations of clusters is created by the relatively slow network interconnection hardware. The interface employed will depend on the bandwidth requirements, latency requirements, distance limitations, and budget constraints. Ethernet is the most commonly implemented network and transmits information at 10 Mbits/sec[1]. Many dedicated clusters are interconnected by more expensive technologies to overcome the limitations induced by the speed of ethernet. The most common alternates to ethernet are FDDI and IBM's Serial Optical Channel Converter (SOCC). Table 2.1 summarizes the speeds of several network technologies.

[1] First generation ethernet cards operating at 100 Mbits/sec are close to being commercially available.

Mohr [253] has summarized the three most typical network topologies commonly employed with workstation clusters in Figure 2.1. The need to maximize the network performance (high bandwidth/low latency) for workstation clusters, particularly for parallel applications, has yielded unique solutions. IBM offers a VME Micro Channel card (manufactured by Bit3 Computer Corporation) as an interconnection for IBM workstation clusters. This hardware, when combined with the Express software from ParaSoft Corporation, achieves both high bandwidth and low latency for applications parallelized with Express.

2.1.2. System Administration

Systems administration is one of the more challenging aspects of workstation clusters. A limited number of tools for performing system administration are available and the logistics of administering large numbers of independent systems can consume significant human resources. Wolbers [367] reports that Fermi National Accelerator Laboratory dedicates two people[2] full time for system administration of their cluster (200+ workstations). An additional 1-2 people are allocated to the task of maintaining their custom cluster software (e.g. batch queuing system, parallelization tools, etc.). System administration tools for workstation clusters are almost nonexistent[3] and remain as one of the areas where significant effort is necessary from the development community.

Dedicated clusters are commonly employed to increase overall batch computing capabilities. They achieve increased throughput by relying on parallelization at the single job level (e.g. individual jobs are submitted to a single workstation in the cluster). Most dedicated clusters have a "server" workstation which manages the job queue, provides extended disk space for I/O, and isolates users from the cluster (users login to the server to perform interactive tasks, submit jobs, etc.). Several software products have been developed to provide the job queuing functionality for clusters. One unique requirement for a cluster queuing system results from the use of workstation clusters as parallel processing environments. Only some of the systems support this capability. A second requirement is imposed by the need to automatically change the number of systems reserved for interactive and batch jobs based on work loads[4].

2.1.3. Representative Dedicated Cluster Sites

Table 2.2 contains a list of several sites which have "dedicated" workstation clusters. Dedicated clusters are commonly located at a central site and managed as a common resource. Typically, they use some form of batch control software to facilitate the execution of queued jobs. These clusters have been installed to provide cost-effective general computing cycles (usually as a replacement/substitute for a single mainframe).

[2]Fermi is not necessarily a good indicator since they have been using workstation clusters for several years and have managed to streamline their operations.

[3]Some vendors have tools with limited scope and capability.

[4]Typically, more systems are allocated for interactive use during the 8am to 5pm period.

2.2. Enterprise Workstation Clusters

One of the greatest attractions of workstation clusters is the ability to utilize idle resources distributed across an enterprise network. Schrage [309] reports that several studies have estimated that approximately 95% of workstation CPU cycles are unused. Enterprise clusters, or decentralized clusters, have the following characteristics.

- Geographically distributed with multiple owners.
- Consist of heterogeneous (multivender) systems.
- Limited control of configuration, system, and cluster participation due to distributed ownership.
- Most often connect via ethernet.
- Attempt to utilize "idle" cycles on existing systems.

The key requirement of enterprise clusters is software which includes the implementation of an "availability policy". An enterprise cluster is configured with workstations which are individually owned. Owners must have a motivation to participate in the cluster. This frequently means that the individual owner believes that they will receive more resources than they contribute. However, individual owners do not want to have their system saturated while they are trying to work. An availability policy allows the individual owner to define how their system will participate in the resource pool and commonly includes items such as:

- system must be inactive for N minutes prior to receiving enterprise jobs,
- a keystroke by the owner suspends all enterprise jobs until the system remains idle for the specified period,
- be able to define configuration parameters (e.g. hardware, software, OS release, etc.), and
- be able to dynamically remove or include their system in the pool.

Two types of software systems exist for operating enterprise clusters;

- dynamic resource sharing and
- automatic resource sharing.

Systems which support dynamic resource sharing have the capability to breakpoint a job (possibly when an owner becomes active) and actually move the job to another system in the pool. Dynamic resource sharing systems typically require the user to link their programs with a runtime library which supports this mechanism. Automatic resource sharing systems will dispatch jobs within the pool but once the job is initiated on a particular machine it is simply suspended when the owner becomes active. Typically, automatic resource sharing systems do not require any modifications to the user's software.

2.3. Batch/Interactive Job Distribution Systems

2.3.1. BALANS

BALANS was developed by VXM Technologies, Inc. and provides both dynamic load balancing and a distributed batch queuing system. Distributed batch processing, distributed RPC-based programs, and network parallel processing can all be performed. BALANS does not require any modification to application code. A unique feature of BALANS is a "fairness" management system. This system prevents individual users from monopolizing the resources, both computers and networks. Additionally, BALANS is a distributed queuing system and therefore does not have a single "master" system. This minimizes the impact of single point failures within the resource pool. BALANS also includes an optional application programming interface (API) for the C language. The API provides complete access to the BALANS system and can be used to develop distributed applications.

2.3.2. Condor and UniJES

Condor [82, 228, 229, 230, 258, 259] was developed by the University of Wisconsin and may be classified as a dynamic resource sharing system since jobs submitted using Condor may be checkpointed and relocated to another system within the pool for continued processing. The major features of Condor are:

- no special programming. Condor is able to run normal Unix applications, only requiring the user to relink their code,
- the local execution environment is preserved for remotely executing processes (e.g. There is no need to move data files to remote workstations before executing programs),
- workstation "owners" have complete priority over their system,
- Condor will checkpoint and relocate jobs if the host system becomes busy,
- Condor prevents any file I/O on the system where a job is executing (except for the owner), and
- Condor automatically locates and allocates idle resources.

Condor does have several documentated limitations:

- only single process jobs are supported (i.e. fork, exec, and similar calls are not implemented),
- signals and signal handlers are not supported,
- interprocess communication calls are not supported, and

- all file operations must be idempotent (e.g. read-only and write-only file accesses work correctly, programs with both read and write to the same file may not).

IBM has commercialized Condor and markets this product under the name UniJES. UniJES includes a significant number of extensions to Condor. These extensions include tools for system administration, tools for job accounting, and tools to improve the security of systems and jobs.

2.3.3. DQS

DQS (Distributed Queuing System) [292] was developed at Florida State University and supports queuing of single machine jobs, multi-node parallel jobs, PVM parallel processing jobs, and interactive sessions across heterogeneous networked workstations. The major features offered by DQS are:

- specificially designed for cluster environments,
- support for batch, interactive, and multi-node jobs,
- PVM is directly supported,
- support for enterprise clusters, and
- automatic load balancing.

2.3.4. DNQS

DNQS (Distributed Network Queuing System) was first developed at Florida State University [167]. Subsequently, DNQS was entirely rewritten by McGill University [212]. DNQS provides distributed queuing services for heterogeneous environments and was specifically developed for workstation clusters. The goals of DNQS/FSU are:

- minimum system administration time for inclusion or removal of an individual workstation,
- not add any security holes,
- the inclusion of an "availability policy" for individual workstation owners, and
- support for heterogeneous environments.

DNQS/McGill was rewritten to address several design limitations of the original DNQS/FSU. The major changes implemented by DNQS/McGill are:

- the need for all machines to have "root" access to a server's disk was removed,
- TCP/IP socket connections were implemented between processes running on different machines in order to allow any operation from any machine on the network,

- the system administrator can configure resource limits for any queue at any time,
- availability of more comprehensive job information on remote processes, and
- output information formats were redesigned.

2.3.5. LSF and Utopia

LSF (Load Sharing Facility) [378] was jointly developed by the University of Toronto and Platform Computing Corporation. Utopia [362, 363, 379] is the name associated with the University implementation. LSF provides resource sharing in heterogeneous networked computing environments and may be classified as an automatic resource sharing system since once jobs are dispatched to a particular computer they must complete their processing there. The system has the following features:

- no restriction on the types of tasks that can be remotely executed (e.g. interactive, batch, sequential, parallel),
- no change to applications or operating systems,
- intelligent job placement based on resource load information and job characteristics,
- a generic interface which supports PVM, p4, and Linda, and
- an application programming interface (API) for developing distributed load sharing applications.

2.3.6. NC Toolset

NC Toolset was developed by The Cummings Group and is a collection of five systems (NCLOGIN, NCSHARE, NCADMIN, NCCACHE, and NCE-API). NCE (Network Computing Executive) provides facilities for locating free resources on a heterogeneous network of Unix based computer systems and acts as the "glue" for the five systems. Additionally, NCE maintains information about the participating system's capacity planning and performance analysis. NCSHARE is a background process distributor and provides checkpoint/restart and process migration. NCLOGIN provides interactive login to the least utilized system. NCADMIN is set of utilities for cluster administration which simplifies performance analysis, performance tuning, configuration, and management of NC Toolset. NCCACHE provides I/O caching facilities for distributed computing environments and works with hierarchical file management systems. NCE-API is an applications programming interface for developing distributed applications. Applications developing using the API can use the NCE global resource manager to schedule processors (particularly useful for parallel cluster applications).

2.3.7. NQS/Exec

NQS/Exec (Network Queuing System/Executive) was jointly developed by The Cummings Group and Sterling Software and provides full NQS batch queuing capability across a heterogeneous networked environment of Unix workstations. NQS/Exec is the result of integrating the original NQS from Sterling Software with NCE from The Cummings Group. NQS/Exec includes the following features:

- automatic load leveling,
- batch/pipe/device queuing,
- queue complexes,
- job priority scheduling,
- accounting information,
- heterogeneous network support, and
- heterogeneous resource support.

Technology	Speed (Mbits/sec)	Comments
Ethernet	10	
	100	Not commercially available
IBM Token Ring	4	
	16	
FDDI	100	
VMEbus	144	IBM with Express software
IBM SOCC	220	Serial Optical Channel Converter
UltraNet	250	
	1000	
HiPPI	800	
	1600	
Fiber Channel Standard	266	
	1000	Not commercially available
Sonet	2400	Not commercially available

Table 2.1: Network Speeds

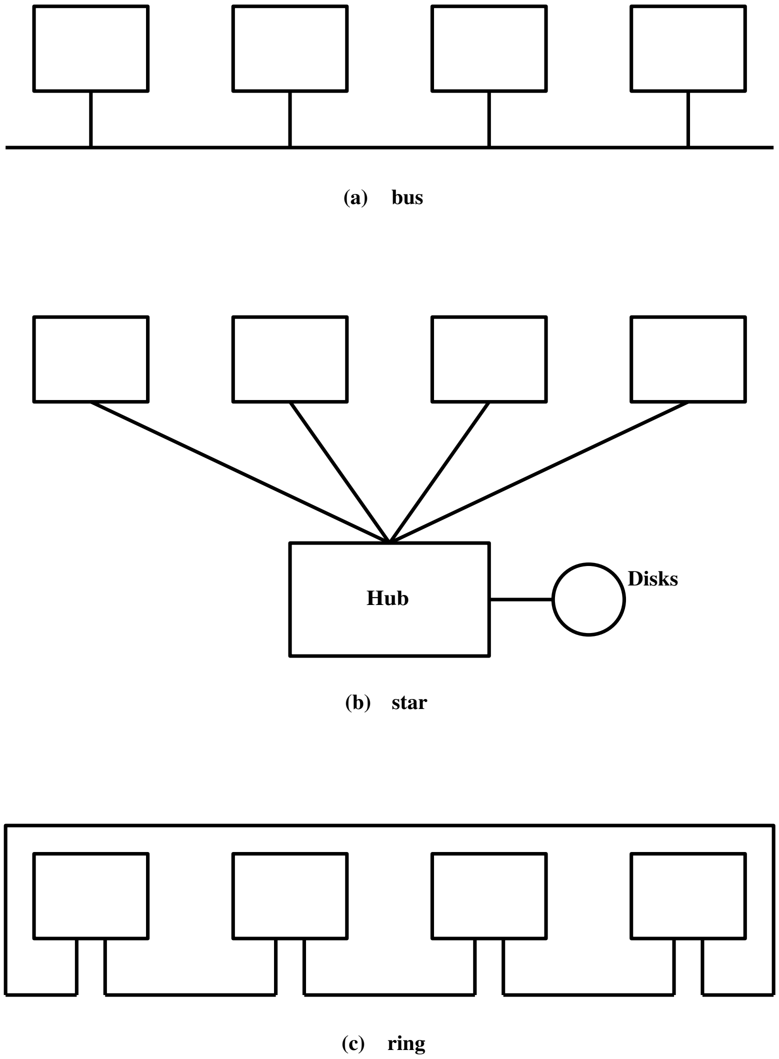


Figure 2.1: Workstation Cluster Topologies

BP Exploration, Alaska	5 IBM RS/6000/550
CERN (Europe)	HP
Colorado School of Mines (Geosciences)	30 IBM RS/6000/320&530 ethernet
Colorado State University (Department of Atmospheric Sciences)	5 IBM RS/6000/550 3 IBM RS/6000/320h 2 IBM RS/6000/320
Cornell University (Supercomputing Technology Center)	8 IBM RS/6000/530 6 IBM RS/6000/520 3 IBM RS/6000/550 IBM fiber optic SLA
Emory University	4 IBM RS/6000/550
Fermi National Accelerator Laboratory	106 IBM RS/6000/320&320h 44 IBM RS/6000/220 6 IBM RS/6000/530 78 SGI/35 80 SGI/Indigos 3 SGI/420 ethernet
Florida State University (Supercomputer Computations Research Center)	30 IBM RS/6000/320h 8 IBM RS6000/530 2 IBM RS6000/550
Indiana University (Computing Services)	4 IBM RS/6000/550
Lawrence Livermore National Laboratory	18 IBM RS/6000/550 FDDI
Lehigh University (Computing Services)	1 IBM RS/6000/950 2 IBM RS/6000/560
NASA Lewis Research Center	32 IBM RS/6000/560 2 ethernet subnets
National Center for Atmospheric Research	5 IBM RS/6000 550 ethernet, fiber optic SLA

continued on next page

continued from previous page	
Naval Research Laboratory	5 IBM RS/6000 320h
Oklahoma University	4 IBM RS/6000 530
Oregon State University	4 IBM RS/6000/550 1 IBM RS/6000/950
Pittsburg Supercomputer Center	8 DEC DS5000/240
Purdue University	4 IBM RS6000/550
University of Georgia	10 IBM RS/6000/550 3 IBM RS/6000/350
University of Illinois (Materials Research Laboratory)	8 IBM RS/6000/320 1 IBM RS/6000/530 2 IBM RS/6000/540
University of Illinois Urbana-Champaign (National Center for Supercomputer Applications)	1 IBM RS/6000/950 6 IBM RS/6000/550
University of Oslo	14 IBM RS/6000/580 2 IBM RS/6000/980
The University of Utah (Chemistry Department)	6 IBM RS/6000/350 15 IBM RS/6000/320
The University of Utah (Utah Supercomputing Institute)	3 IBM RS/6000/520 1 IBM RS/6000/530
Washington University (Center for Molecular Design)	2 IBM RS/6000/550 6 IBM RS/6000/320h

Table 2.2: Representative Dedicated Cluster Sites

3. Parallel Programming Using Clusters

3.1. Introduction

Workstation clusters provide a unique flexibility to prototype and experiment with parallel programming. Previously, parallel programming required investing in computers designed and marketed for this specific purpose. Now any researcher with access to multiple workstations can develop and experiment with parallel programs. Beguelin et. al. [45] have also demonstrated the capability to solve grand challenge problems using a workstation cluster. Two approaches are used to implement parallelization over a workstation cluster:

- extend existing sequential languages such as C and Fortran with parallel constructs to handle communications and synchronization, or
- define new programming languages based on functional, object oriented, or logical paradigms.

3.2. Parallel Programming Systems

Freund [149] has classified parallel programming systems for workstation clusters as "network orchestration tools". These systems are typically constructed around message passing primitives, shared memory primitives, or both.

3.2.1. ADAPTOR

ADAPTOR (Automatic DAta Parallelism TranslatOR) [75, 76, 77, 78, 79] was developed by the German National Research Institute for Computer Science and is a tool that transforms data parallel programs written in Fortran with array extensions, parallel loops, and layout directives to parallel programs with explicit message passing. ADAPTOR supports heterogeneous workstation configurations via the use of PVM as the message passing model.

3.2.2. APPL

APPL (Application Portable Parallel Library) [287] was developed by NASA Lewis Research Center. APPL is a subroutine based communication library and is intended to provide a consistent programmer interface to a variety of MIMD multiprocessors (including networked workstations). APPL has been developed using the PICL and PARMACS libraries as models. The APPL programmer only has to view the application in terms of communicating processes and is freed from dealing with the hardware architectures. APPL provides asynchronous/synchronous send/receive messages and functions for broadcast, probe, and global operations.

3.2.3. Chameleon

Chameleon [174] was developed by Argonne National Laboratory and is a high-level subroutine based communications library. The goal of Chameleon is to provide a hierarchy of models for parallel programming on distributed memory parallel computers. Chameleon can be used with native (vendor) communications libraries, p4, PICL, or PVM and therefore supports many architectures (including networked workstations). A set of scientific functions, Tools, have been implemented using Chameleon. Sparse linear algebra, iterative solvers, and parallel multicolor ILU are included in the Tools library.

3.2.4. DINO

DINO (DIstributed Numerically Oriented language) [121, 297] was developed by the University of Colorado and is a C augmented language for writing parallel programs for distributed memory multiprocessors (including networked workstations). DINO is directed at data parallel algorithms. The most signficant characteristics of DINO are the ability to declare a virtual parallel computer that is best suited to the parallel computation, the ability to map distributed data structures onto this virtual machine, and the ability to define procedures that will run on each processor of the virtual machine concurrently. Process management and interprocessor communications are handled automatically by the compiler.

3.2.5. Express

Express [138, 139, 209] was developed by ParaSoft Corporation and is a set of tools designed with the goal of making parallel and distributed programming efficient and portable. Express includes the following capabilities:

- a hardware independent communication system,
- a high level library of communication facilities which implements most of the commonly used parallel processing strategies,
- load balancing,
- heterogeneous systems capability,
- global data operations (no explicit message passing required)
- an automated decomposition system for distributing programs among the nodes and which supports run-time configuration,
- a parallel I/O and graphics system tailored to parallel applications,

Product	Function
Network Linda	Linda over a network of workstations
Piranha Linda	Dynamic resource utilization over a network of workstations
POSYBL	Public domain implementation of Linda
Tuplex	Alternative commercial version of Linda

Table 3.1: Linda Related Systems

- a set of performance analysis tools,
- an interactive, source level debugger for distributed programs,
- asynchronous debugging tools, and
- a parallelizer that takes conventional, sequential C programs and converts them to parallel programs.

Express offers C, C++, and Fortran bindings and markets several Express-based application specific products (Expray for ray tracing, Dime for irregular problems).

3.2.6. Linda and Related Systems

Linda [159, 160, 161, 162, 163] was developed by Yale University. There have been several commercial implementations of Linda. However, Scientific Computing Associates, Inc. has been the most aggressive marketer of Linda products. Other Linda related products are listed in Table 3.1. Linda is a parallel programming lanaguage created by extending a sequential lanaguage, C or Fortran, with six additional statements. The Linda model is based on "tuples" (a shared memory concept) and data is either written or read from tuple space. The programming sematics of Linda are simple and the programmer is spared the chore of explicit message passing.

Network Linda has recently been released and supports Linda applications over a set of networked workstations. Piranha-Linda is a research project which structures Linda modules as a "cloud of tasks" which computational "Piranhas" attack. The more Piranhas which attack, the faster the job is completed.

3.2.7. p4

p4 (Portable Programs for Parallel Processors) was developed by Argonne National Laboratory [74, 86, 233] and is a library of macros and subroutines for programming a variety of parallel systems (including networked workstations). p4 supports both C and Fortran and includes monitors for shared memory models, message passing for the distributed memory model, and support for combining the two models. The goals [233] of p4 are to provide:

- a portable programming system, dependent only on a small number of computational models (shared memory, distributed memory, cluster)
- an efficient implementation on each architecture supported,
- shared-memory programming at the monitor level (not locks),
- message-passing at an efficient yet portable level,
- support for the cluster model of parallel programming, and
- a foundation for high-level programming systems and libraries.

3.2.8. PVM and Related Systems

PVM (Parallel Virtual Machine) [5, 41, 43, 44, 45, 157, 222, 338, 339, 340, 341, 342] was developed at Oak Ridge National Laboratory and Emory University and is a software package which allows the utilization of a heterogeneous network of parallel and serial computers as a single computational resource. Facilities for spawning, communication, and synchronization are supported. Support for both C and Fortran is included with PVM. PVM version 3.0 will be available in Spring 1993 and will include the following new features and enhancements:

- new naming convention for routines (all functions will begin with pvm_),
- dynamic process groups,
- dynamic configuration,
- super initiate routine (can start up processes under dbx, pass arguments to spawned processes, machine selection based on speed and load),
- signal handling between PVM components,
- daemon-daemon communications remains UDP but has TCP-like retry algorithms,
- more efficient data coercion routines,
- more modular send/receive routines, and
- multiple message buffers.

PVM has been widely accepted by hardware vendors (Cray, Convex, SGI, HP, etc) and therefore has spawned several related development efforts. Table 3.2 summarizes some of the projects related to PVM.

HeNCE (Heterogeneous Network Computing Environment) [40, 42] was also developed at Oak Ridge National Laboratory and is a X-based software program designed to assist with the development of parallel programs. HeNCE programs are described graphically and each node of the graph represents subroutines. The arcs of the graph represent data dependencies. HeNCE is composed of integrated graphical tools for creating, compiling, executing, and analyzing HeNCE progams.

Product	Function
DoPVM	Distributed object PVM
FT-PVM	Fault Tolerant PVM
PVM++	Message passing object oriented PVM
HeNCE	Graphical front-end to PVM
Xab	Run time monitoring of PVM programs

Table 3.2: PVM Related Systems

3.2.9. TCGMSG

TCGMSG (Theoretical Chemistry Group Message passing system) was developed at Argonne National Laboratory and is a toolkit for writing portable parallel programs using a message passing model. TCGMSG supports a variety of architectures (Intel iPSC, Alliant, workstations, networked workstations, supercomputers, etc). TCGMSG is modelled after the p4 system and programs communicate over TCP sockets (shared memory is used when processes reside on the same machine). C and Fortran bindings are included.

3.2.10. TOPSYS

TOPSYS (TOols for Parallel Systems) [48, 50, 52, 53] was developed at the Munich University of Technology and is an integrated tool environment for developing applications for heterogeneous and distributed computing facilities. TOPSYS includes tools for parallel programming, monitoring, debugging, performance analysis, animation, and dynamic load balancing.

3.3. Parallel Programming Languages

Another approach to parallel computing using clusters is to implement programming languages which address the characteristics of distributed applications. Programming languages offer several advantages over other approaches.

- Transparency to programmer of the operating system and hardware idiosyncrasies
- Portability
- Optimization of operations

Some of the disadvantages of using special purpose programming languages for distributed applications are:

- the cost of development,

- the widespread adoption of the language (ensures long term support, availability across systems, etc), and
- the requirement to learn a new language.

Many of the distributed languages are implementated as language extensions to existing languages such as Fortran. Others are completely new grammars which are typically based on object-oriented methodologies.

3.3.1. Fortran M

Fortran M [143] was developed at Argonne National Laboratory and is a language for modular parallel programming. Fortran M provides extensions to Fortran 77 to support message passing and includes the following features:

- Modularity. Programs are constructed using explicitly-declared communication channels to plug together program modules called processes. A process can encapsulate common data, subprocesses, and internal communication.
- Safety. Operations on channels are restricted so as to guarantee deterministic execution.
- Architecture Independence. The mapping of processes to processors can be specified with respect to a virtual computer with size and shape different from that of the target computer.
- Efficiency. Fortran M can be compiled efficiently for uniprocessors, shared-memory computers, distributed-memory computers, and networks of workstations.

3.3.2. Mentat

Mentat [168, 169, 170, 171, 172, 332] was developed by the University of Virginia and was designed to solve the problem of writing software for parallel and distributed systems. Mentat provides high-level abstractions for parallelism that closely match the way developers are accustomed to writing programs. The Mentat Programming Language (MPL) is an object-oriented programming language based on C++. Mentat has been enhanced to support heterogeneous distributed environments under the the project name Mentat MetaSystems Testbed.

3.3.3. Oasis

Oasis (Object and Agent Specification and Implementation System) [106] was developed at the University of Michigan and is a distributed programming language which supports parallel computing. Computational processes are modeled as a distributed collection of autonomous cooperative agents. Oasis supports networked heterogeneous computers.

3.3.4. OOF

OOF (Object Oriented Fortran) [291, 323] was developed at Mississippi State University and is based on a minimum set of extensions to Fortran. OOF provides Fortran extensions to allow for object and operator declaration (including constructor and destructor functions) and invocation of operators. OOF does not implement common object oriented characteristics such as inheritance and strong typing. Parallelism in OOF is implicit since operators for different object instances are all independent and therefore may be operated on concurrently.

3.3.5. PCN

PCN (Program Composition Notation) was developed at Argonne National Laboratory [141, 142] and is a system for developing and executing parallel programs. It comprises a high-level programming language, tools for developing and debugging programs in this language, and interfaces to Fortran and C that allow the reuse of existing code in multilingual parallel programs. PCN programs are portable across many different workstations, networks, and parallel computers. Parallelization over networked workstations is supported.

3.3.6. SR

SR (Synchronizing Resources) [12, 13, 271, 307] was developed at the University of Arizona and is a language for writing distributed and parallel programs. The language supports a wide variety of (reliable) message passing constructs, including shared variables (for processes on the same node), asynchronous message passing, rendezvous, remote procedure call, multicast.

3.3.7. xHPF

xHPF (extended high-performance Fortran) was developed by Applied Parallel Research and Kuck & Associates, Inc and is a preprocessor that generates Fortran 77 code which make calls to a parallel runtime library. The runtime library has interfaces for many of the popular message passing libraries such as PVM, Express, Linda, etc. xHPF accepts the subset of Fortran 90 and HPF directives recommended by the HPFF (High-Performance Fortran Forum). The product supports current MIMD parallel architectures as well as clustered workstations.

4. Ancillary Cluster Topics

4.1. Introduction

The focus of this report was directed at surveying software products which could be employed to exploit the potential of workstation clusters. However, there are numerous ancillary topics which warrant discussion. This chapter includes brief overviews of several topics which are directly related to the main thrust of this report. Many of these topics, such as distributed application development tools, could be the focus of their own indepth survey. The overviews, and software systems, presented in this chapter should be considered an introduction to the various topics. The reader is directed to the extensive bibliography included in this report for more comprehensive presentations of the material in this chapter.

4.2. Collaboration Systems

Collaboration systems were identified in the original Metacenter proposal to the NSF [269] as one of the key technologies required to address grand challenge problems. The National Center for Supercomputer Applications (NCSA) responded to this requirement by establishing the DICE (Distributed Interactive Collaboration Environment) project to develop software tools which promote collaboration via networked computers. The first product (in Beta release) released by the DICE project is a software system named Collage (COLLaborative Analysis and Graphics Environment). Collage has been characterized as a distributed WYSIWIS (what you see is what I see) environment. Collage allows multiple participants, connected via networked computers, to perform data analysis and interpretation as a collaborative endeaver. Collage supports

- real-time collaborative work across systems via a X-based environment,
- image display and analysis,
- color palette editing,
- spreadsheet display of floating point numbers,
- both text display and editing,
- a "whiteboard" for drawing,
- animation, and

- a HDF browser.

All active participants may share common windows and communicate via dialog windows. The goal is to allow geographically distributed research collaborations using evolving high-speed networks and workstations.

4.3. Distributed Application Development Tools

The focus of this report was directed at workstation clusters and software systems which may employed to more effectively utilize them. Batch job distribution systems and parallelization tools for workstation clusters were the primary topics of this report. However, the more common use of distributed client-server software to exploit benefits of heterogeneous systems connected via networks is another approach to exploiting heterogeneous computing environments. In this section several software systems which enable the development of distributed applications are surveyed. Additionally, a brief discussion of Remote Procedure Calls (RPC) is included since most of these systems are based on RPCs.

4.3.1. The Remote Procedure Call

RPCs are the enabling technology which allows the distribution of an application over a network. A client computer (typically the system residing on the user's desk) will connect to a server computer (a system with the desired capabilities) to execute procedures on the server. RPCs provide a high-level interface for interprocess communications across a network and insulate the programmer from the complex details necessary to utilize low-level interfaces such as Berkeley sockets. Additionally, some RPC implementations address the nuances of data coercion (different data formats on different systems).

Figure 4.1 diagrams the basic flow of a RPC-based distributed application. The programmer of an RPC-based application describes the interface between client and server using an Interface Description Language (IDL). The IDL generates two "stub" modules. One module is linked with the client program and the other with the server program. The stub modules contain the necessary software to perform the interconnection between the two components. Commonly, a run-time library is also provided. The run-time library contains all the functions which are required by the IDL to support the various network transport layers, data coercion, error handling, etc.

The two most frequently employed RPC environments are NCS (developed by Hewlett-Packard and adopted by the Open Software Foundation (OSF) as part of their standard) and ONC (developed by Sun). Both NCS [190] and ONC are synchronous RPCs and bind the client process to only a single server process during a procedure call. Asynchronous RPCs are emerging and have the benefit of allowing a single client process to communicate with multiple CPU servers concurrently (called multicasting). Recently the Open Software Foundation has released the DCE (Distributed Computing Environment) definition [272, 296, 317]. Several key components of high performance network computing which OSF DCE supports are RPCs, naming/locator service, and multithreading.

The success of distributed computing environments is inherently linked to the use of "lightweight protocols" to achieve maximum performance of network based applications. These protocols minimize such issues as network processing, memory, and transmission requirements. TCP/IP is the most commonly employed lightweight protocol. However, another lightweight protocol being supported by several hardware vendors is the Express Transfer Protocol (XTP) being developed by Protocol Engines, Inc. XTP uses a fixed frame syntax, reduced data buffering, a reliable multicast technique, and selective retransmission for increased efficiency. Another issue which affects the transparency of RPCs is their ability to address various network transport layers such as TCP/IP, DECnet, ISO/OSI TP4, DDS, etc. which is required for heterogeneous distributed applications.

4.3.2. Cronus

Cronus is an object-oriented distributed computing environment and was developed by BBN Systems and Technologies [54, 55, 56, 359]. Cronus includes

- a complete environment for developing integrated distributed applications,
- support for heterogeneous environments,
- interoperability and coexistance with existing software, operating systems, and applications, and
- asynchronous RPCs to manage multiple, parallel CPU servers (called multicasting).

4.3.3. DAS

The Distributed Application Support package [22] was developed by the University of British Columbia. DAS includes a set of library routines and an ASN.1 [99, 100] compiler for generating the data coercion routines. It provides the remote operations service defined by CCITT standard X.219 [98] using the protocol defined by the CCITT standard X.229. DAS implements the ISO [197, 198] upper layer protocols and supports multi-threading. The DAS remote operations component (ROCP) supports all association classes (1, 2, and 3) and operating classes 1 and 2.

4.3.4. dlib

dlib (Distributed Library) was developed by researchers at NASA Ames Research Center [371, 372, 373]. The primary motivation for dlib was to enable a more efficient development of client-server applications which require the coupling of specialized visualization workstations with a supercomputer. dlib is RPC-based and provides a service which allows for an arbitrary length conversion within a single context between client and server. Additionally, the dlib server process stores state information which persists from call to call and dynamically allocates memory for data storage and manipulation. Like other interconnection systems dlib includes utilities for automatic stub generation and a run-time library. dlib has been used for developing interactive CFD applications and distributed virtual reality applications.

4.3.5. ELROS

ELROS [80, 81] (Embedded Language for Remote Operations Service) was developed by Lawrence Livermore National Laboratory and is an embedded language for programming distributed applications using remote operations (ISO 9072-1 and 9072-2 [197, 198]). ELROS is targeted at large scale scientific distributed applications. Service requests, which may be asynchronous, in ELROS are expressed in the form of a programming language embedded in a traditional lanaguage and are processed by a preprocessor.

4.3.6. HRPC

HRPC [14] was developed at the University of Washington and is targeted at solving the complex issue of network transport layer heterogeneity. HRPC attempts to address two concepts:

- the fact that all systems will not use the same RPC protocol and
- the fact that all the basic components of the RPC, such as network transport, are not always implemented.

Included with HRPC is a run-time system capable of emulating other RPC systems. An Interface Description Language (IDL) is used to define the procedural interface between client and server. A stub compiler generates the necessary stub modules for the application and includes calls to the run-time library.

4.3.7. Isis

Isis [63, 64, 65, 135, 175] was originally developed by Cornell University and has subsequently been productized by Isis Distributed Systems, Inc. Isis is a fault tolerant system for building applications of cooperating, distributed processes. The tools provide high level support for developing distributed applications in C, C++, Fortran, Common Lisp, and Ada. Isis supports groups of cooperating processes, replicated data, and distributed computation. Isis appears to be targeted at financial applications such as brokerage and trading systems and includes several utilities which are summarized in Appendix B.

4.3.8. NetShare

NetShare [370] was developed by Aggregate Computing, Inc. and represents one of the recent commercial entries into the field of distributed computing. The NetShare distributed computing environment is a layer of enhanced services above existing distributed computing tools that adds the capability to intelligently locate applications over the participating network resources. NetShare maintains dynamic status information about each machine participating in the resource pool and dispatches tasks to the most suitable resource. NetShare includes an asynchronous remote execution service targeted at heterogeneous environments and an API (applications programming interface) for developers to access both the remote execution services and the resource management services.

4.3.9. Polylith

Polylith [191, 282, 283, 284, 285, 286] was developed at the University of Maryland and is used to prepare and interconnect mixed-language software components for execution in heterogeneous environments. Polylith is targeted at the following requirements:

- The specification of an applications structure and the implementation of individual components should be independent of each other.
- The application's geometry (or distribution) should be separated from the implementation of individual modules.
- Programmers should be able to specify how components communicate without having to adapt programs at the source level (data coercion is one example).

Polylith includes a Module Interconnection Langauge (MIL) for defining an application's structure and geometry. This language is based on a simple graph model where nodes in the program graph correspond to modules and arcs in the graph represent intermodule bindings. The last requirement is incorporated via a "software toolbus". A software toolbus is any agent that encapsulates, or isolates, all run-time interfacing concerns of an application. Communication in Polylith can be implemented with TCP/IP or XNS protocols in a network, via shared memory between light threads on a tightly-coupled multiprocessor, using custom channels between processors, or using a branch instruction within the same process space.

4.3.10. Schooner

Schooner [192, 193] was developed at the University of Arizona and is a RPC-based interconnection system. Schooner relies on the common "stub" module paradigm to implement transparent distribution and heterogeneity. The system includes three basic services: an external data representation based on the UTS type system [180, 181, 182] for data coercion, a procedure specification language and associated stub compilers, and a run-time system for control flow which includes a heterogeneous remote procedure call facility. Schooner uses a sequential execution model where control passes from procedure to procedure and is typically used for distributed visualization/computation applications (e.g. workstation/supercomputer).

4.4. Distributed Operating Systems

The operating systems used in mainstream computing environments are commonly referred to as "network operating systems". These operating systems do not provide networking functionality as an integral component of their design. Rather, networking primitives have been added to extend the environment so a single computer can communicate with another computer. This method is not transparent due to the numereous differences between heterogeneous configurations. Distributed operating systems were designed to hide the machine characteristics and make collections of machines appear to be a single multi-user time-sharing sytem rather than a pool of autonomous systems. Distributed operating systems share a common kernel on each machine and handle resource management automatically. There are several research efforts presently directed at developing distributed operating systems; Amoeba [256, 257, 346, 347, 355], Charlotte [136], Chorus [298, 299], Clouds [119, 131], and V System [107].

4.5. Distributed Shared Memory

Distributed shared memory (DSM) [252, 266, 333, 376, 377] is an the abstraction for supporting the concept of shared memory in a physically non-shared (distributed) architecture. In a DSM system a process which stores a value does not need to know the existence or location of other processes which may fetch it. DSM systems are difficult to implement because the inherent communication delays are significant and the shared memory paradigm requires all cooperating processes to have an identical image of the shared address space.

4.5.1. ALMS

ALMS (Asynchronous Linked Memory System) [25, 276] was developed at Brookhaven National Laboratory and is a toolkit for constructing parallel applications from independently coded modules. ALMS is a variation of the typical shared memory system because it only requires that all cooperating processes have an identical view of the sequence of successive values stored at a particular address during the computation. This implies that all processes may not see the same member of the sequence at the same time. ALMS is most suitable for asynchronous applications such as graphics/computation, least square minimization of a function over parameter space, and searches for minimum length paths on graphs.

4.5.2. DSM

DSM (Distributed Shared Memory) was developed by Florida State University and is a communications system based on the distributed shared memory paradigm which allows parallel programs to “share” vectors of integers and reals in a simple and intuitive fashion. DSM uses UDP (universal datagram protocol) to provide broadcasting functionality.

4.5.3. Mermera

Mermera [184, 185, 186] was developed by Boston University and is a DSM (Distributed Shared Memory) system which gives the programmer the choice of coherent and non-coherent behavior in the same program. Mermera provides a wide variety of cleanly related memory operations satisfying different ordering constraints thereby providing an opportunity to trade-off programming simplicity for performance.

4.5.4. Mether

Mether [250] was developed by the University of Delaware and provides a shared memory implementation over a group of Sun workstations connected via ethernet. Pages of memory in Mether migrate around the network in a demand-paged fashion and are accessed by the programmer in a fashion indistinguishable from other memory. Mether has been implemented as two separate components; a kernel driver which maintains a set of pages and their associated state and a user-level server driver which facilitates data transmission. The user-level server runs as an event-driven loop where events are UDP messages and Mether page-wanted/page-freed events.

4.6. Heterogeneity and Data Coercion

One of the fundamental concerns with message passing in heterogeneous systems is coping with the various machine-specific data formats for bytes, integers, and floating point numbers (e.g. Cray, VAX, and IEEE floating point representations, and big-endian vs. little-endian byte orders). The most frequently used system for accomplishing this task is the External Data Representation (XDR) developed by Sun [336, 337]. The XDR sender encodes the data into the interchange format while the XDR receiver decodes the data to the target systems format. Another data coercion system is the Universal Type System (UTS) [180, 181, 182] which is internal to the Mixed Language Programming (MLP) system. Basic encoding rules have also been defined in the standard ISO/OSI ASN.1 [197, 198]. Additionally, many of the distributed environment development tools provide their own implementation for data coercion.

4.7. Performance Monitoring and Debugging Systems

A critical component of developing distributed applications and particularly parallel applications is the availability of performance monitoring and debugging tools. Commercial vendors of products like Express and Linda provide excellent tools for aiding the software developer with these activities. However, there are few public domain utilities which serve this requirement.

4.7.1. ParaGraph

Paragraph [183] was developed by the University of Illinois and Oak Ridge National Laboratory. X-based Paragraph is a post processing system which uses PICL format tracefiles generated by an application at run-time. Paragraph provides more than 20 different analysis views and is capable of animating the performance behavior of parallel programs.

4.7.2. Upshot

Upshot [189] was developed at Argonne National Laboratory and is a X-based graphics utility for viewing log files produced by parallel programs. Upshot uses PICL format log files and is similar to ParaGraph. Upshot has fewer functions than ParaGraph but each function has more generality and greater depth than the equivalent function in ParaGraph.

4.7.3. Xab

Xab (X-window Analysis and deBugging) [47] was developed by Carnegie Mellon University and is a tool for run-time monitoring of PVM programs. Xab consists of three components: a user library, a monitoring program, and a X-based interface. The user library included with Xab provides instrumented versions of the PVM calls. Xab also includes a translator which can translate Xab log files to PICL format log files.

4.8. Scientific Libraries

The development of software systems to support distributed applications and parallel applications greatly improves the potential utilization of these paradigms. Additionally, the development of scientific libraries modeled after the BLAS, LINPACK, and LAPACK projects will create a more widespread implementation of applications which can efficiently utilize distributed networked computing resources. This section overviews several ongoing efforts directed at providing scientific libraries.

4.8.1. DLAPACK

DLAPACK [115, 116] is presently being developed by Oak Ridge National Laboratory as an extension to the highly successful LAPACK project. DLAPACK[1] implementation of LAPACK which will be built on top of PVM. DLAPACK will implement all of the LAPACK functions and should produce the first portable library of sophisticated scientific functions for distributed memory architectures. A component of the DLAPACK project is the development of the BLACS (Basic Linear Algebra Communications Subprograms) [10, 125, 126]. The BLACS is a set of functions which will efficiently manage the message passing of matrices of standard structure (e.g. rectangular, trapezoidal, etc).

[1] Dr. Jack Dongarra regularly refers to this effort by the name DLAPACK. However, he admits this may not be the final product name.

4.8.2. DSSLIB

DSSLIB [118] is a commercial product offered by Dakota Scientific Software. DSSLIB is a fault tolerant, optimized version of the LINPACK and BLAS (Levels 1,2,3) libraries which has been parallelized for execution across a network of workstations. DSSLIB requires no modifications to applications which have been developed using the LINPACK and BLAS libraries (they must be relinked with the DSSLIB library). Implementations are available for both Sun and SGI workstations connected via ethernet.

4.8.3. TOOLS

Tools [174] is a set of scientific functions which have been developed on top of the Chameleon system. Functions in the Tools system are portable to any system capable of running p4 or PVM. Tools includes functions for sparse linear algebra, parallel multicolor ILU, and iterative solvers.

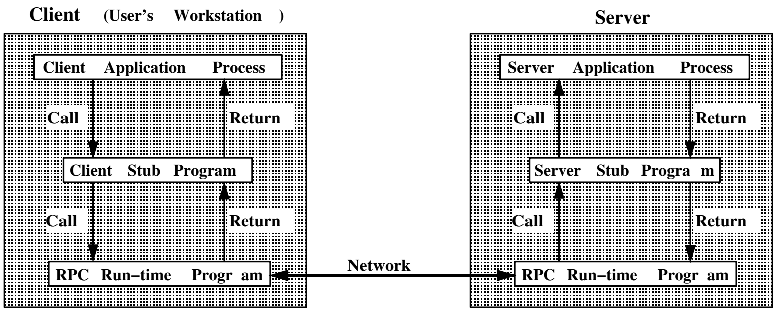


Figure 4.1: Remote Procedure Call

Bibliography

[1] R. Agrawal and A. Ezzat. Location independent remote execution in NEST. *IEEE Transactions on Software Engineering*, 13(8):905–912, August 1987.

[2] R. Agrawal and A. K. Ezzat. Processor sharing in NEST: a network of computer workstations. In *1st International Conference on Computer Workstations*, pages 198–208, November 1985.

[3] S. Ahuja, N. J. Carriero, and D. H. Gelernter. Linda and friends. *IEEE Computer*, 19(8):26–34, August 1986.

[4] S. Ahuja, N. J. Carriero, D. H. Gelernter, and V. Krishnaswamy. Matching language and hardware for parallel computation in the Linda machine. *IEEE Transactions on Computers*, 37(8):921–929, August 1988.

[5] M. Alfano and G. Lo Re. Distributing numerical algorithms: some experiences with network computing system (NCS) and parallel virtual machine (PVM). In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[6] G. T. Almes, A. P. Black, E. D. Lazowska, and J. D. Noe. The Eden system: a technical review. *IEEE Transactions on Software Engineering*, SE-11(1):43–58, January 1985.

[7] A. L. Ananda, B. H. Tay, and E. K. Koh. A survey of asynchronous remote procedure calls. *Operating Systems Review*, 26(2):92–109, April 1992.

[8] Bruce M. Anderson and John P. Flynn. CASES: a system for assessing naval warfighting capability. Technical Report SAIC-90/1508, Science Applications International Corporation, June 1990.

[9] Dan Anderson. The care and feeding of the NCAR cluster. In *Proceedings of the 2nd Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[10] E. Anderson, A. Benzoni, J. Dongarra, S. Moulton, S. Ostrouchov, B. Tourancheau, and R. van de Geijn. Basic linear algebra communications subprograms. In *Sixth Distributed Memory Computing Conference Proceedings*. IEEE Computer Society Press, 1991.

[11] Michael J. Andrescavage. FRAMEWORK: a general purpose, heterogeneous, distributed, concurrent, parallel processing architecture. In H. D. Schwetman, editor, *Proceedings 1991 International Conference on Parallel Processing*, pages II258–II259. CRC Press, Inc., August 12-16, 1991. Volume II: Software.

[12] Gregory R. Andrews and Ronald A. Olsson. *The SR programming language*. The Benjamin/Cummings Publishing Company, 1993.

[13] Gregory R. Andrews, Ronald A. Olsson, Michael A. Coffin, Irving Elshoff, Kelvin Nilsen, Titus Purdin, and Gregg Townsend. An overview of the SR language and implementation. *ACM Transactions on Programming Languages and Systems*, 10(1):51–86, January 1988.

[14] Anonymous. *Heterogeneous Remote Procedure Call - Programmer's Guide*. University of Washington, 1988.

[15] Anonymous. Livermore labs to get IBM machines. *Unix Today*, November 25, 1991.

[16] Anonymous. Clustering to turbocharge RS/6000. *Computerworld*, May 4, 1992.

[17] Anonymous. Custom clusters. *RS/Magazine*, pages 6–8, July 1992.

[18] Anonymous. Eugene brooks: Killer micros settling down in suburbs. *HPCwire (Electronic Newsletter)*, September 1992.

[19] Anonymous. Killer micros settling down in suburbs. *HPCwire (Electronic Newsletter)*, September 1992.

[20] Anonymous. Meta series integrates Convex supers and HP workstations. *HPCwire (Electronic Newsletter)*, October 1992.

[21] Anonymous. NASA Lewis installs RS/6000 cluster for jet engine design. *HPCwire (Electronic Newsletter)*, November 1992.

[22] Anonymous. Distributed application support package (draft). University of British Columbia, Canada, 1990 October.

[23] Apollo. *NCS User's guide*. Apollo Computer Inc., 1988.

[24] M. Arango, D. Berndt, N. Carriero, D. Gelernter, and D. Gilmore. Adventures with network Linda. *Supercomputer Review*, 10(3):42–46, October 1990.

[25] B. Arbeit, G. Campbell, R. Coppinger, R. Peierls, and D. Stampf. The ALMS system: implementation and performance. Brookhaven National Laboratory, Internal Report, (In preparation).

[26] Rob Armstrong and Jane Macfarlane. POET - parallel object-oriented environment and toolkit. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[27] E. H. Baalbergen. Parallel and distributed compilations in loosely-coupled systems: a case study. In *Proceedings of the Workshop on Large Grain Parallelism*, Providence, RI, October 1986.

[28] E. H. Baalbergen. Design and implementation of parallel make. *Computing Systems*, 1:135–158, Spring 1988.

[29] O. Babaoglu, L. Alvisi, A. Amoroso, and R. Davoli. Paralex: An environment for parallel programming in distributed systems. Technical Report UB-LCS-91-01, University of Bologna, Bologna, Italy, February 1991.

[30] H. E. Bal. *Programming Distributed Systems.* Silicon Press, Summit, NJ, 1990.

[31] H. E. Bal, M. F. Kaashoek, and A. S. Tanenbaum. Experience with distributed programming in Orca. In *IEEE CS International Conference on Computer Languages*, pages 79–89, March 1990.

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

[33] H. E. Bal, J. G. Steiner, and A. S. Tanenbaum. Programming languages for distributed computing systems. *ACM Computing Surveys*, 21(3):261–322, September 1989.

[34] H. E. Bal, R. van Renesse, and A. S. Tanenbaum. Implementing distributed algorithms using remote procedure calls. In *Proceedings of the 1987 National Computer Conference*, pages 499–506, Chicago, IL, June 15–18, 1987.

[35] M. Barbacci, D. Doubleday, C. Weinstock, and J. Wing. Developing applications for heterogeneous mechina networks. *USENIX Computing systems*, 2:7–35, 1989.

[36] John Bazik. QuaHog: stealing cycles without getting caught. In *Proceedings of the Heterogeneous Network-Based Concurrent Computing Workshop*, Tallahassee, FL, October 1991. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.91.

[37] John Bazik and Curtis Yarvin. QuaHog: distributed computing for the real world. Computer Science Department, Brown University, October 1991.

[38] BBN. Parallel computing, past, present and future. Technical report, BBN Advanced Computers Inc., Cambridge, MA, November 1990.

[39] J. Becker and L. Dagum. Particle simulation on heterogeneous distributed supercomputers. In *First International Symposium on High Performance Computing*, Syracuse, NY, September 1992.

[40] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, K. Moore, and R. Wade. HeNCE: A user's guide (Draft). Oak Ridge National Laboratory, November 1991.

[41] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, and V. Sunderam. A user's guide to PVM: Parallel virtual machine. Technical Report ORNL/TM-11826, Mathematical Sciences Section, Oak Ridge National Laboratory, September 1991.

[42] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, and V. S. Sunderam. Graphical development tools for network-based concurrent supercomputing. In *Proceedings of Supercomputing 1991*, pages 435–444, Albuquerque, 1991.

[43] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, and V. S. Sunderam. Heterogeneous network supercomputing. *Supercomputing Review*, August 1991.

[44] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, and V. S. Sunderam. Opening the door to heterogeneous network supercomputing. *Supercomputing Review*, pages 44–45, September 1991.

[45] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, and V. S. Sunderam. Solving computational grand challenges using a network of heterogeneous supercomputers. In D. Sorensen, editor, *Proceedings of Fifth SIAM Conference on Parallel Processing*, Philadelphia, PA, 1991. SIAM.

[46] Adam Beguelin and Gary Nutt. Examples in Phred. In Jack Dongarra, Ken Kennedy, Paul Messina, Danny C. Sorensen, and Robert G. Voigt, editors, *Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing*, pages 602–608, Hoston, TX, March 25-27, 1991. SIAM.

[47] Adam Louis Beguelin. Xab: a tool for monitoring PVM programs. School of Computer Science, Carnegie Mellon University, June 5, 1992.

[48] H. Beier, T. Bemmerl, A. Bode, and et. al. TOPSYS - tools for parallel systems. Technical Report SFB-Bericht 342/9/90 A, Technische Universitat Munchen, Munich, Germany, January 1990.

[49] Gordon Bell. Ultracomputers: a teraflop before its time. *Communications of the ACM*, 35(8):27–47, August 1992.

[50] T. Bemmerl and A. Bode. TOPSYS - tools for parallel systems (user's overview and user's manual). Technical Report SFB-Bericht 342/25/90 A, Technische Universitat Munchen, Munich, Germany, December 1990.

[51] T. Bemmerl, A. Bode, T. Ludwig, and S. Tritscher. MMK - multiprocessor multitasking kernel (user's guide and user's reference manual). SFB-Bericht 342/26/90, Technische Universitat Munchen, Munich, Germany, December 1990.

[52] Thomas Bemmerl. An integrated and portable tool environment for parallel computers. In *Proceedings of IEEE International Conference on Parallel Processing*, pages 50–53, 1988.

[53] Thomas Bemmerl and Bernhard Ries. Programming tools for distributed multiprocessor computing environments. In *Proceedings of the Heterogeneous Network-Based Concurrent Computing Workshop*, Tallahassee, FL, October 1991. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.91.

[54] James C. Berets. Building distributed applications: why use objects. In *Proceedings of the Workshop on Object-oriented modelling in distributed systems*, Quebec, Canada, May 1992.

[55] James C. Berets, Ronald A. Mucci, and Richard E. Schantz. Cronus: a testbed for developing distributed systems. In *1985 IEEE Military Communications Conference*, pages 409–417. IEEE Communications Society, October 1985.

[56] James C. Berets and Richard M. Sands. Introduction to Cronus. Technical Report 6986, BBN Systems and Technologies, January 1989.

[57] Donna Bergmark and Caroline Hecht. SPEEDO: a study in supercomputer/workstation cooperative processing. Technical Report CTC89TR16, Cornell Theory Center, Cornell University, June 1989.

[58] G. Bernard and et al. Primitives for distributed computing in a heterogeneous local area network environment. *IEEE Transactions on Software Engineering*, 15(12), December 1989.

[59] B. Bershad, D. Ching, E. Lazowska, J. Sanislo, and M. Schwartz. A remote procedure call facility for interconnecting heterogeneous computer systems. *IEEE Transactions on Software Engineering*, 13(8):880–894, August 1987.

[60] B. N. Bershad and H. M. Levy. Remote computation in a heterogeneous environment. Technical Report 87-06-04, Department of Computer Science, University of Washington, Seattle, Washington, June 1987.

[61] G. Betello, G. Richelli, S. Succi, and F. Ruello. Lattice Boltzmann method on a cluster of IBM RISC System/6000. In *First International Symposium on High Performance Distributed Computing*, Syracuse, NY, September 1992.

[62] Paul A. Bicknell. Software development and configuration management in the Cronus distributed operating system. In *Proceedings of the Conference on Software Maintenance - 1988*, pages 143–147, October 1988.

[63] K. Birman and R. Cooper. The ISIS project: real experience with a fault tolerant programming system. *Operating System Review*, pages 103–107, April 1991.

[64] K. P. Birman, R. Cooper, T. A. Jospeh, K. P. Kane, F. Schmuck, and M. Wood. *ISIS-a distributed programming environment.* Cornell University, Ithaca, NY, June 1990. in User's Guide and Reference Manual.

[65] Kenneth Birman and Keith Marzullo. ISIS and the META project. *Sun Technology*, pages 90–104, Summer 1989.

[66] Roberto Bisiani and Alessandro Forin. Multilanguage parallel programming of heterogeneous machines. *IEEE Transactions on Computers*, 37(8):930–945, August 1988.

[67] R. Bjornson, N. Carriero, D. Gerlernter, T. Mattson, D. Kaminsky, and A. Sherman. Experience with Linda. Technical Report 866, Department of Computer Science, Yale University, August 1991.

[68] Robert Bjornson. Experience with Linda on the iPSC/2. Technical Report 698, Department of Computer Science, Yale University, March 1989.

[69] Robert Bjornson, Nicholas Carriero, David Gelernter, and Jerrold Leichter. Linda, the portable parallel. Technical Report 520, Department of Computer Science, Yale University, February 1987.

[70] Robert Bjornson, Craig Kolb, and Andrew Sherman. Ray tracing with network Linda. *SIAM News*, 24(1), January 1991.

[71] A. P. Black. Supporting distributed applications: experience with Eden. In *10th ACM Symposium on Operating System Principles*, pages 181–193, December 1985.

[72] J. L. Black and C. B. Su. Networked parallel seismic computing. In *Proceedings of the 24th Offshore Technology Conference*, May 1992. OTC-6825.

[73] Barry D. Bowen. Tuplex: a C-Linda implementation for Sun, NeXT, and Mac. *SunWorld*, pages 30–34, June 1992.

[74] J. Boyle, R. Butler, T. Disz, B. Glickfeld, E. Lusk, R. Overbeek, J. Patterson, and R. Stevens. *Portable Programs for Parallel Processors.* Holt, Rinehart, and Winston, 1987.

[75] Thomas Brandes. ADAPTOR: Installation guide. Internal Report Adaptor 1, German National Research Institute for Computer Science, October 92. Draft.

[76] Thomas Brandes. ADAPTOR: Language reference manaul. Internal Report Adaptor 3, German National Research Institute for Computer Science, October 92. Draft.

[77] Thomas Brandes. ADAPTOR: The distributed array library. Internal Report Adaptor 4, German National Research Institute for Computer Science, October 92. Draft.

[78] Thomas Brandes. ADAPTOR: Users guide. Internal Report Adaptor 2, German National Research Institute for Computer Science, October 92. Draft.

[79] Thomas Brandes. Efficient data parallel programming without explicit message passing for distributed memory multiprocessors. Internal Report AHR-92-4, German National Research Institute for Computer Science, September 92. Draft.

[80] M. L. Branstetter, J. A. Guse, and D. M. Nessett. ELROS - an embedded language for remote operations service. Lawrence Livermore National Laboratory, Livermore, CA.

[81] M. L. Branstetter, J. A. Guse, and D. M. Nessett. An ELROS primer. Technical Report UCRL-MA-108543, Lawrence Livermore National Laboratory, Livermore, CA, October 1991.

[82] Allan Bricker, Michael Litzkow, and Miron Livny. Condor technical summary. Computer Sciences Department, University of Wisconsin, Madison, WI.

[83] Mathew Burns. Commentary: Attack of the killer clusters. *HPCwire (Electronic Newsletter)*, May 1992.

[84] Mathew Burns. NSF supercomputer centers form national metacenter. *HPCwire (Electronic Newsletter)*, November 1992.

[85] Lee Busby. A review of the Lasnet batch job system. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[86] Ralph Butler and Ewing Lusk. User's guide to the p4 programming system. Technical Report ANL-92/17, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, October 1992.

[87] Leigh D. Cagan, Nicholas J. Carriero, and Stavros A. Zenios. Pricing mortgage-backed securities with network Linda. Scientific Computing Associates, New Haven, CT, January 1992.

[88] D. Callahan, K. Cooper, R. Hood, K. Kennedy, and L. Torczon. Parascope: a parallel programming environment. *International Journal of Supercomputer Applications*, 2(4), 1988.

[89] John R. Callahan and James M. Purtilo. A packaging system for heterogeneous environments. Technical Report TR-2542, Computer Science Department, University of Maryland, College Park, Maryland, 1990.

[90] Mike Carl. DCE helps light a fire under RPCs. *Open Systems Today*, pages 47,51,54,56, January 4, 1993.

[91] N. Carriero and D. Gelernter. Applications experience with Linda. In *Proceedings ACM/SIGPLAN PPEALS 1988*, pages 173–187, 1988.

[92] N. Carriero and D. Gelernter. How to write parallel programs: a guide to the perplexed. *ACM Computing Surveys*, 21:323–357, September 1989.

[93] N. Carriero and D. Gelernter. Linda in context. *Communications of the ACM*, 32(4):444–458, April 1989.

[94] N. Carriero and D. Gelernter. Tuple analysis and partial evaluation strategies in the Linda precompiler. In *Second Workshop on Languages and Compilers for Parallelism*, August 1989.

[95] N. J. Carriero. *Implementation of tuple space.* PhD thesis, Department of Computer Science, Yale University, December 1987.

[96] Nicholas Carriero and David Gelernter. The S/Net's Linda kernel. *ACM Transactions on Computer Systems*, 4(2):110–129, May 1986.

[97] Thomas A. Casey Jr., Stephen T. Vinter, D. G. Weber, Rammohan Varadarajan, and David Rosenthal. A secure distributed operating system. In *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pages 27–38. IEEE Computer Society, April 1988.

[98] CCITT. Remote operations: model, notation and service definition. Technical report, International Telegraph and Telephone Consultative Committee, 1988. Recommendation X.219.

[99] CCITT. Specification of abstract syntax notation one (ASN.1). Technical report, International Telegraph and Telephone Consultative Committee, 1988. Recommendation X.208.

[100] CCITT. Specification of basic encoding rules for abstract syntax notation one (ASN.1). Technical report, International Telegraph and Telephone Consultative Committee, 1988. Recommendation X.209.

[101] M. Chandy and S. Taylor. *An Introduction to parallel programming.* Jones and Bartlett, 1991.

[102] Craig M. Chase, Alex L. Cheung, Anthony P. Reeves, and Mark R. Smith. PARAGON: a parallel programming environment for scientific applications using communications structures. In H. D. Schwetman, editor, *Proceedings 1991 International Conference on Parallel Processing*, pages II211–II218. CRC Press, Inc., August 12-16, 1991. Volume II: Software.

[103] J. Chase, F. Amador, E. Lazowska, H. Levy, and R. Littlefield. The Amber system: parallel programming on a network of multiprocessors. *ACM Operating Systems Review*, 23(5):147–158, December 1989.

[104] Taitin Chen, Jim Feeney, Geoffrey Fox, Gideon Frieder, Sanjay Ranka, Bill Wilhelm, and Fang-kuo Yu. A prototype switch for scalable high-performance distributed computing. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[105] Doreen Y. Cheng. A survey of parallel programming tools. Technical Report RND-91-005, NASA Ames Research Center, Moffett Field, CA, May 1991.

[106] Fah-Chun Cheong. *OASIS: An agent-oriented programming language for heterogeneous distributed environment.* PhD thesis, The University of Michigan, 1992. School of Computer Science and Engineering.

[107] D. R. Cheriton. The V distributed system. *Communications of the ACM*, 31(3):314–333, March 1988.

[108] A. Cheung. High performance computing on a cluster of workstations. In *First International Symposium on High Performance Distributed Computing*, Syracuse, NY, September 1992.

[109] Chimp. CHIMP concepts. Edinburgh Parallel Computing Center, The University of Edinburgh, UK, June 1991.

[110] Chimp. Prototype CHIMP project report. Edinburgh Parallel Computing Center, The University of Edinburgh, UK, December 1991.

[111] Chimp. CHIMP executive summary: 3. Edinburgh Parallel Computing Center, The University of Edinburgh, UK, September 1992.

[112] Chimp. CHIMP version 1.0 CSN design. Edinburgh Parallel Computing Center, The University of Edinburgh, UK, March 1992.

[113] Chimp. CHIMP version 1.0 distributed Unix design. Edinburgh Parallel Computing Center, The University of Edinburgh, UK, August 1992.

[114] Chimp. CHIMP version 1.0 interface. Edinburgh Parallel Computing Center, The University of Edinburgh, UK, May 1992.

[115] J. Choi, J. J. Dongarra, and D. W. Walker. The design of distributed level 3 BLAS routines, 1992. in preparation.

[116] Jaeyoung Choi, Jack J. Dongarra, Roldan Pozo, and David W. Walker. ScaLAPACK: a scalable linear algebra library for distributed memory concurrent computers. Mathematical Sciences Section, Oak Ridge National Laboratory, 1992.

[117] B. Cmelik. *Concurrent Make: the design and implementation of a distributed program in Concurrent C.* AT&T Bell Laboratories, Murray Hill, NJ, 1986. in Concurrent C Project.

[118] Dakota Scientific Software, 501 E. St. Joseph St., Rapid City, SD 57701-3995, (605) 394-2471. *DSSLIB: a library of parallelized and optimized linear-algebra routines for workstations.*

[119] Partha Dasgupta, Richard LeBlanc, Mustaque Ahamad, and Umakishore Ramachandran. The Clouds distributed operations system. *IEEE Computer*, 24(11):34–44, November 1991.

[120] Jose Mario De Martino and Rolf Kohling. Production rendering on a local area network. *Computers & Graphics*, 16(3):317–324, 1992.

[121] Thomas M. Derby, Elizabeth Eskow, Richard Neves, Matthew Rosing, Robert B. Schnabel, and Robert P. Weaver. DINO 1.0 user's manual. Technical Report CU-CS-501-90, Computer Science Department, University of Colorado at Boulder, April 1990.

[122] A. Deshpand and M. H. Schultz. Solving the shallow water equations using Network Linda. In *Proceedings of Supercomputing 92*, 1992.

[123] H. Dietz, M. O'Keefe, T. Parr, T. Varghese, and P. Woodward. Fortran-P. Presentation for Army High Performance Computing Research Center Research Advisory Committee, University of Minnesota, March 1992.

[124] Harry Dolan, Robert Bjornson, and Leigh Cagan. A parallel CFD study on Unix networks and multiprocessors. United Technologies Research Center.

[125] J. J. Dongarra. Workshop on the BLACS. LAPACK Working Note 34, Technical Report CS-91-134, University of Tennessee, 1991.

[126] Jack J. Dongarra. Lapack working note 34: Workshop on the BLACS. Oak Ridge National Laboratory, February 6, 1992.

[127] Jack J. Dongarra and Robert A. van de Geijn. Lapack working note 37: Two dimensional basic linear algebra communications subprograms. Oak Ridge National Laboratory, Proposed Standard, October 28, 1991.

[128] F. Douglis, M. F. Kaashoek, A. S. Tanenbaum, and J. K. Ousterhout. A comparison of two distributed operating systems. *Computing Systems*, 4(3):353–384, December 1991.

[129] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, 1(1):63–83, March 1989.

[130] Milos D. Ercegovac. Heterogeneity in supercomputer architectures. *Parallel Computing*, 7:367–372, 1988. North-Holland.

[131] P. Dasgupta et al. The design and implementation of the Clouds distributed operating system. *Usenix Computing Systems Journal*, 3(1), 1990.

[132] Michael Factor. Using Linda to build parallel AI applications. Technical Report YALEU/DCS/TR-861, Department of Computer Science, Yale University, June 1991.

[133] J. R. Falcone. A programmable interface language for heterogeneous distributed systems. *ACM Transactions on Computer Systems*, 5(4):330–351, November 1987.

[134] Salvatore Filippone and Maria Loredana Sales. Parallel linear system solvers on IBM RISC/System 6000 clusters. In *Proceedings of the 2nd Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[135] Ralph Finch and Shao-Kong Kao. Coarse-grain parallel computing using ISIS tool kit. *Journal of Computing in Civil Engineering*, 6(2):233–244, April 1992.

[136] R. Finkel and et al. M. Scott. Experience with Charlotte: simplicity and function in a distributed operating system. *IEEE Transactions on Software Engineering*, 15:676–686, June 1989.

[137] Brett D. Fleisch and Gerald J. Popek. Mirage: a coherent distributed shared memory design. *Operating Systems Review*, 23(5):211–223, December 1989.

[138] J. Flower, A. Kolawa, and S. Bharadwaj. The Express way to distributed processing. *Supercomputing Review*, pages 54–55, May 1991.

[139] Jon Flower and Adam Kolawa. A "packet" history of message passing systems. ParaSoft Corporation.

[140] David Forslund. Recent results on high speed networking and distributed computing in the Advanced Computing Laboratory. In *Proceedings of the Heterogeneous Network-Based Concurrent Computing Workshop*, Tallahassee, FL, October 1991. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.91.

[141] Ian Foster, Robert Olson, and Steven Tuecke. Productive parallel programming: the PCN approach. Mathematics and Computer Science Division, Argonne National Laboratory, Argonnne, IL, (To appear in: Scientific Programming).

[142] Ian Foster and Steven Tuecke. Parallel programming with PCN. Technical Report ANL-91/32, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, December 11, 1991.

[143] Ian T. Foster and K. Mani Chandy. Fortran M: A language for modular parallel programming. Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, June 1992.

[144] G. Fox, S. Hiranandani, K. Kennedy, C. Koelbel, U. Kremer, C. Tseng, and M. Wu. Fortran D language specification. Technical Report TR90-141, Department of Computer Science, Rice University, Houston, Texas, 1990.

[145] Avi Freedman. A CASE tool set for constructing distributed heterogeneous applications. In *Proceedings of the 1989 3rd International Workshop on Computer Aided Software Engineering*, Imperial College, London, July 17-21, 1989.

[146] R. F. Freund. Superconcurrent processing: A dynamic approach to heterogeneous parallelism. In *Proceedings of the Parallel/Distributed Computing Networks Seminar*, San Diego, 1990. IEEE.

[147] R. F. Freund. SuperC or distributed heterogeneous HPC. *Computing Systems in Engineering*, 1991. To appear.

[148] R. F. Freund and D. S. Conwell. Superconcurrency, a form of distributed heterogeneous supercomputing. *Supercomputing Review*, pages 47–50, October 1990.

[149] Richard F. Freund. Heterogeneous processing. *HPCwire (Electronic Newsletter)*, 1992.

[150] Richard F. Freund. Heterogeneous processing. *Supercomputing International*, May 1992.

[151] Richard F. Freund and Larry J. Peterson. If the network is the computer, then... *Supercomputing Review*, pages 36–37, October 1991.

[152] Manchi J. Gadbois and Anthony M. Newton. Tri-service distributed technology experiment. Technical Report SAIC-90/1508, Science Applications International Corporation, June 1990.

[153] Carolyn H. Gard. University of Georgia's experience with clustering RS/6000's and ES/9000 720. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[154] Greg Garry. Software links powerful mixed clusters. *Digital News & Review*, pages 7–7, January 4, 1992.

[155] G. A. Geist, M. T. Heath, B. W. Peyton, and P. H. Worley. PICL: A portable instrumented communications library. Technical Report TM-11130, Oak Ridge National Laboratory, Oak Ridge, Tennessee, 1990.

[156] G. A. Geist, M. T. Heath, B. W. Peyton, and P. H. Worley. A user's guide to PICL: a portable instumented communications library. Technical Report ORNL/TM-11616, Oak Ridge National Laboratory, Oak Ridge, Tennessee, January 1992.

[157] G. A. Geist and V. S. Sunderam. Experiences with network based concurrent computing on the PVM system. Technical Report ORNL/TM-11760, Oak Ridge National Laboratory, January 1991.

[158] G. A. Geist and V. S. Sunderam. Network-based concurrent computing on the PVM system. *Concurrency: Practice and Experience*, 4(4):293–311, June 1992.

[159] D. Gelernter. Generative communications in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, January 1985.

[160] D. Gelernter, N. Carriero, S. Chandran, and S. Chang. Parallel programming in Linda. In D. Degroot, editor, *1985 International Conference on Parallel Processing*, pages 255–263, 1985.

[161] David Gelernter. Getting the job done. *BYTE*, November 1988.

[162] David Gelernter. Multiple tuple spaces in Linda. In *PARLE 89*, pages 20–27. Springer-Verlag, June 1989. Volume 366 of Lecture Notes in Computer Sciences.

[163] David Gelernter and James Philbin. Spending your free time. *BYTE*, May 1990.

[164] P. B. Gibbons. A stub generator for multi-language RPC in heterogeneous environments. *IEEE Transactions on Software Engineering*, 13(1):77–87, January 1987.

[165] Brian Gillooly. IBM taps into unused MIPS on networks. *Reseller News*, pages 1,171, May 6, 1991.

[166] Roberto Gomperts and Rosario Caltabiano. ParNet: a simple approach to network parallelism. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL,

December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[167] Thomas P. Green. DNQS, a distributed network queuing system. Draft document, Supercomputer Computations Research Institute, Florida State University, December 1992.

[168] A. S. Grimshaw. The Mentat run-time system: support for medium grain parallel computation. In *Proceedings of the Fifth Distributed Memory Computing Conference*, pages 1064–1073, Charleston, SC, April 9-12, 1990.

[169] A. S. Grimshaw. An introduction to parallel object-oriented programming with Mentat. Technical Report TR-91-07, Department of Computer Science, University of Virgina, April 1991.

[170] A. S. Grimshaw, D. Mack, and T. Strayer. MMPS: portable message passing support for parallel computing. In *Proceedings of the Fifth Distributed Memory Computing Conference*, pages 784–789, Charleston, SC, April 9-12, 1990.

[171] Andrew S. Grimshaw. Meta-systems: an approach combining parallel processing and heterogeneous distributed computing systems. In *Proceedings 1992 Workshop on Heterogeneous Parallel Processing, International Parallel Processing Symposium*, 1992.

[172] Andrew S. Grimshaw, Jon B. Weissman, and Emily A. West. Experiences with the Mentat MetaSystems testbed. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[173] Bill Gropp and Ewing Lusk. A test implementation of the MPI draft message-passing standard. Mathematics and Computer Science Division, Argonne National Laboratory, 1992.

[174] Bill Gropp and Barry Smith. Chameleon parallel programming tools user's manual. Mathematics and Computer Science Division, Argonne National Laboratory, 1992.

[175] ISIS Group. The restructuring of ISIS for modern distributed operating systems, September 1991. Internal Cornell Report.

[176] R. Haber, B. Bliss, and D. Jablonowski. A distributed environment for run-time visualization and application steering in computational mechanics. Technical Report CSRD 1235, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, Urbana, IL, June 1992.

[177] R. Haber, D. McNabb, and R. Ellis. Eliminating distance in scientific computing: an experiment in televisualization. *International Journal of Supercomputing Applications*, 4:71–89, 1990.

[178] Robert Hagmann. Process server: sharing processing power in a workstation environment. In *Proceedings of the 6th International Conference on Distributed Computing Systems*, pages 260–267, Cambridge, MA, May 1986.

[179] Harren Harrison. Tools for multiple-cpu environments. *IEEE Software*, pages 45–45, May 1990.

[180] R. Hayes. *UTS: a type system for facilitating data communications*. PhD thesis, University of Arizona, August 1989. Department of Computer Science.

[181] R. Hayes, S. Manweiler, and R. D. Schlichting. A simple system for constructing distributed, mixed-language programs. *Software - Practice and Experience*, 18(7):641–660, July 1988.

[182] R. Hayes and R. D. Schlichting. Facilitating mixed language programming in distributed systems. *IEEE Transactions on Software Engineering*, 13(12):1254–1264, December 1987.

[183] Michael T. Heath and Jennifer A. Etheridge. ParaGraph: a tool for visualizing performance of parallel programs. University of Illinois and Oak Ridge National Laboratory, January 1992.

[184] Abdelsalam Heddaya and Himanshu Sinha. Coherence, non-coherence and local consistency in distributed shared memory for parallel computing. Technical Report BU-CS-92-004, Computer Science Department, Boston University, Boston, MA, May 1992.

[185] Abdelsalam Heddaya and Himanshu Sinha. An implementation of MERMERA: a shared memory system that mixes coherence with non-coherence. Technical Report BU-CS-92-013, Computer Science Department, Boston University, Boston, MA, October 1992.

[186] Abdelsalam Heddaya and Himanshu Sinha. An overview of MERMERA: a system and formalism for non-coherent distributed parallel memory. Technical Report BU-CS-92-009, Computer Science Department, Boston University, Boston, MA, September 1992.

[187] G. Henry and A. Hoisie. Block factorizations on a cluster of RS/6000s. Technical Report CTC92TR88, Cornell Theory Center, February 1992.

[188] K. Herbst. The dawning of the age of network supercomputing. *Supercomputing Review*, pages 46–51, April 1991.

[189] Virginia Herrarte and Ewing Lusk. Study parallel program behavior with Upshot. Technical Report ANL-91/15, Mathematics and Computer Science Division, Argonne National Laboratory, August 1991.

[190] Hewlett Packard. *Managing NCS Software*, March 1992. Manual No. D-11895-E.

[191] Christine Hofmeister, Joanne Atlee, and James Purtilo. *Writing distributed programs in Polylith.* Computer Science Department, University of Maryland, College Park, Maryland, 1.0 edition, November 1990.

[192] Patrick T. Homer. Adapting AVS to support scientific applications as heterogeneous distributed programs. Technical Report TR 91-23, Department of Computer Science, The University of Arizona, Tucson, Arizona, September 1991.

[193] Patrick T. Homer and Richard D. Schlichting. A software platform for constructing scientific applications from heterogeneous resources. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[194] HP. Cluster computing. Hewlett Packard Workstation Systems Group, 1992.

[195] IBM. Clustering Risc System/6000s: for batch and parallel processing. IBM, 1992. A Technical Computing Perspective.

[196] IBM. Clustering with Risc System/6000 workstations. Technical Computing Systems, IBM, Dallas, TX, August 1992.

[197] ISO. Information processing systems - open systems interconnection - basic reference model. Technical report, International Organization for Standardization and International Electrotechnical Committee, 1984. International Standard 7498-1.

[198] ISO. Information processing systems - open systems interconnection - the directory - authentication framework. Technical report, International Organization for Standardization and International Electrotechnical Committee, December 1989. International Standard 9594-8.

[199] Steven Jeffreys. Uniform access to signal data in a distributed heterogeneous computing environment. In *International Telemetering Conference Proceedings*, pages 707–714. Instrument Society of America, October 1992. Volume XXVIII.

[200] J. Jones, R. Rashid, and M. Thompson. Matchmaker: an interface specification language for distributed processing. In *Proceedings of the 12th ACM Symposium on Principles of Programming Languages*, pages 225–235, January 1985.

[201] Marsha Jovanovic. Metacomputing, high-speed networking, and scientific visualization. in Computational Sciences Advances, San Diego Supercomputer Center, pages 26–29, 1992.

[202] E. Jul, H. Levy, N. Hutchinson, and A. Black. Fine-grained mobility in the Emerald system. *ACM Transactions on Computer Systems*, 6(1):109–133, February 1988.

[203] M. F. Kaashoek, R. van Renesse, H. van Staveren, and A. S. Tanenbaum. FLIP: an internetwork protocol for supporting distributed systems. *ACM Transactions on Computer Systems*, 1993. Accepted for publication.

[204] A. Kamel, P. Squazzero, and V. Zecca. Large scale computing on clustered vector multiprocessors. In *Proceedings of Supercomputing 1990*, pages 418–427, New York, November 1990. IEEE Computer Socity Press.

[205] A. Karp and R. Babb. A comparison of 12 parallel Fortran dialects. *IEEE Software*, 5(5):52–67, 1988.

[206] A. Khokar, V. K. Prasanna, M. Shaaban, and C. L. Wang. Heterogeneous supercomputing: problems and issues. In *Proceedings of the Sixth International Parallel Processing Symposium*, March 1992. Workshop on Heterogeneous Processing.

[207] Alan E. Klietz, Andrei V. Malevsky, and Ken Chin-Purcell. A case study in metacomputing: distributed simulations of mixing in turbulent convection. Preprint 92-121, Army High Performance Computing Research Center, University of Minnesota, Minneapolis, MN, November 1992.

[208] James Kobielus. LANs bid for distributed power. *Network World*, December 3, 1990.

[209] A. Kolawa. Parasoft: a comprehensive approach to parallel and distributed computing. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[210] H. T. Kung, Robert Sansom, Steven Schlick, Peter Steenkiste, Matthieu Arnould, Fracois J. Bitz, Fred Christianson, Eric C. Cooper, Onat Menzilcioglu, Denise Ombres, and Brian Zill. Network-based multicomputers: an emerging parallel architecture. In *Proceedings Supercomputing 91*, pages 664–673, November 1991.

[211] H. T. Kung, Peter A. Steenkiste, Marco Gubitoso, and Manpreet Khaira. Parallelizing a new class of large applications over high-speed networks. In *Proceedings of the Third ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, April 1991.

[212] Martin-Daniel Lacasse. DNQS: A dynamic network queueing system - user manual. Department of Physics, McGill University, Montreal, Canada, April 1992.

[213] Monica S. Lam. Jade: a coarse-grain parallel programming language. In *Proceedings of the Heterogeneous Network-Based Concurrent Computing Workshop*, Tallahassee, FL, October 1991. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.91.

[214] Richard P. LaRowe Jr. and Thomas H. Probert. Heterogeneous by design: an environment for exploiting heterogeneity. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[215] E. D. Lazowska, D. L. Eager, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Transactions on Software Engineering*, 12:662–675, May 1986.

[216] E. D. Lazowska, D. L. Eager, and J. Zahorjan. The limited performance benefits of migrating active processes for load sharing. *ACM Performance Evaluation Review*, 16:63–72, May 1988.

[217] D. B. Leblang and R. P. Chase Jr. Computer-aided software engineering in a distributed network environment. *SIGPLAN Notices*, 19(5):104–112, May 1984.

[218] David B. Leblang and Robert P. Chase Jr. Parallel software configuration management in a network environment. *IEEE Software*, 20(11):28–35, November 1987.

[219] J. Leichter. Shared tuple memories, shared memories, buses and LANs - Linda implementations across the spectrum of connectivity. Technical Report 714, Yale University, July 1989.

[220] Jerry Leichter. *Shared tuple memories, shared memories, buses and LAN's - Linda implementations across the spectrum of connectivity*. PhD thesis, Yale University, July 1989.

[221] W. Leler. Linda meets Unix. *IEEE Computer*, 23(2), February 1990.

[222] Juan Leon, Allan L. Fisher, and Peter Steenkiste. Fail-safe PVM. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[223] Lynda Lester. Clustered workstations to ease the load on shavano. *SCD Computing News*, 13(3):3–7, April 1992. National Center for Atmospheric Research.

[224] K. Li. IVY: A shared virtual memory system for parallel computing. In *1988 International Conference on Parallel Processing*, pages 94–101, August 1988.

[225] David D. H. Lin. *An efficient environment support for heterogeneous distributed programming*. PhD thesis, Department of Computers Science and Engineering, Southern Methodist University, December 1990.

[226] David D. H. Lin, Behrooz Shirazi, and Krishna Kavi. A heterogeneous distributed processing interface specification language. In H. D. Schwetman, editor, *Proceedings 1991 International Conference on Parallel Processing*, pages II274–II275. CRC Press, Inc., August 12-16, 1991. Volume II: Software.

[227] B. H. Liskov. Distributed programming in Argus. *Communications of the ACM*, 31(3):300–313, March 1988.

[228] M. Litzkow. Remote Unix - turning idle workstations into cycle servers. In *Proceedings of the Summer 1987 Usenix Conference*, Phoenix, Arizona, June 1987.

[229] Michael J. Litzkow, Miron Livny, and Matt W. Mutka. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference on Distributed Computing Systems*, pages 104–111, San Jose, CA, June 1988.

[230] Mike Litzkow and Miron Livny. Experience with the Condor distributed batch system. Computer Sciences Department, University of Wisconsin, Madison, WI.

[231] K.-P. Lohr, J. Muller, and L. Nentwig. DAPHNE: support for distributed applications programming in heterogeneous computing networks. In *IEEE 8th International Conference on Distributed Computing Systems*, pages 63–71, 1988.

[232] M. Lottor. Internet growth (1981-1991). Request for Comment 1296, Network Information Systems Center, SRI International, January 1992.

[233] Rusty Lusk and Ralph Butler. Portable parallel programming with p4. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[234] N. B. MacDonald and M. G. Norman. Issues in parallel programming environments. In *BCS Parallel Processing Specialist Group Workshop on Abstract Machine Models for Highly Parallel Computers*, Leeds, England, March 1991.

[235] Messac Makpangou and Kenneth Birman. Designing application software in wide area network settings. Technical Report 90-1165, Department of Computer Science, Cornell University, 1991.

[236] John Markoff. David Gelernter's romance with Linda. *The New York Times*, Sunday, January 19, 1992. Business, Section 3.

[237] Carolyn Duffy Marsan. Livermore lab subs RS/6000s for super. *Federal Computer Week*, pages 10,14, December 16, 1991.

[238] Carolyn Duffy Marsan. Agencies go crazy for clusters. *Federal Computer Week*, pages 1,60, May 11, 1992.

[239] Carolyn Duffy Marsan. IBM, Ancor to build high-speed clusters. *Federal Computer Week*, pages 26,32, June 15, 1992.

[240] Carolyn Duffy Marsan. NSF pursues computing without walls. *Federal Computer Week*, 6(35):1,53, November 30, 1992.

[241] K. Marzullo, R. Cooper, M. Wood, and K. Birman. Tools for distributed applications management. *IEEE Computer*, 24(8):42–51, August 1991.

[242] T. G. Mattson. Is there a crisis in parallel computing software? *Supercomputing Review*, pages 38–40, February 1991.

[243] Timothy G. Mattson, Rob Bjornson, and David Kaminsky. The C-Linda language for networks of workstations. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[244] Richard McCormack. Will clusters soon supplant the traditional supercomputer. *High Performance Computing and Communications Week*, 1(7):1–2, December 16, 1992.

[245] Carlos R. Mechoso, Chung-Chun Ma, John D. Farrara, and Joseph A. Spahr. Distribution of a climate model across high-speed networks. In *Proceedings Supercomputing 91*, pages 253–260, November 1991.

[246] Daniel Menasce and Virgilio Almeida. Cost-performance analysis of heterogeneity in supercomputer applications. In *Proceedings Supercomputing 90*, pages 169–177, November 1990.

[247] Daniel Menasce and Virgilio Almeida. Heterogeneous supercomputing: Is it cost-effective? *Supercomputing Review*, pages 39–41, August 1991.

[248] Paul Messina. Parallel and distributed supercomputing at Caltech. Technical Report CCSF-10-91, Caltech Concurrent Supercomputing Facilities, California Institute of Technology, Pasadena, CA, October 1991.

[249] R. Minnich and D. Pryor. A radiative heat transfer simulation on a SPARCstation farm. In *First International Symposium on High Performance Distributed Computing*, Syracuse, NY, September 1992.

[250] Ronald G. Minnich and David J. Farber. The Mether system: distributed shared memory for SunOS 4.0. *Usenix*, Summer 1989.

[251] R. Mirchandaney, D. Towsley, and J. Stankovic. Adaptive load sharing in heterogeneous distributed systems. *Journal of Parallel and Distributed Computing*, (9):331–346, 1990.

[252] Ajay Mohindra and Umakishore Ramachandran. A survey of distributed shared memory in loosely-coupled systems. Technical Report GIT-CC-91/01, College of Computing, Georgia Institute of Technology, January 1991.

[253] Jeffrey Mohr. Clustered workstations as a parallel processor. Presentation at Supercomputer Technology Conference, April 1992.

[254] H. Moons and P. Verbaeten. Distributed computing in heterogeneous environments. In *EUUG Spring 1990 Conference Proceedings*, pages 15–25, Munich, Germany, April 1990.

[255] R. W. Moore. Distributing applications across wide area networks. In *Proceedings of the Second Gigabit Testbed Workshop*, pages 36–43, Reston, Virginia, February 13-15, 1991. Volume 2.

[256] S. J. Mullender and A. S. Tanenbaum. The design of a capability-based distributed operating system. *The Computer Journal*, 29(4):289–300, March 1986.

[257] S. J. Mullender, G. van Rossum, A. S. Tanenbaum, R. van Renesse, and H. van Staveren. Amoeba: a distributed operating system for the 1990s. *IEEE Computer Magazine*, May 1990.

[258] M. Mutka. *Sharing in a privately owned workstation environment*. PhD thesis, University of Wisconsin, Madison, WI, May 1988.

[259] M. Mutka and M. Livny. Profiling workstations' available capacity for remote execution. In *Proceedings of Performance-87, The 12th IFIP W.G. 7.3 International Symposium on Computer Performance Modeling, Measurement and Evaluation*, Brussels, Belgium, December 1987.

[260] NCSA. NCSA's metacomputer: a special report. *Access*, 5(5), September-December 1991. NCSA, University of Illinois at Urbana-Champaign.

[261] Dan Nessett. Developing coarse-grained scientific distributed applications. In *Proceedings of the Heterogeneous Network-Based Concurrent Computing Workshop*, Tallahassee, FL, October 1991. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.91.

[262] Dan Nessett and Jim Rathkopf. Computing on heterogeneous supercomputer clusters. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[263] John R. Nichol, C. Thomas Wilkes, Richard D. Edmiston, and Joseph C. Fitzgerald. Experiences with accommodating heterogeneity in a large scale telecommunications infrastructure. In *Proceedings of the 3rd Symposium on Experiences with Distributed and Multiprocessor Systems*. Usenix Association, March 1992.

[264] D. A. Nichols. Using idle workstations in a shared computing environment. In *Proceedings of the 11th ACM Symposium on Operating Systems Principles*, pages 5–12, November 1987.

[265] J. R. Nicol. *Operating system design for distributed programming environments*. PhD thesis, University of Lancaster, UK, October 1986.

[266] B. Nitzberg and V. Lo. Distributed shared memory: a survey of issues and algorithms. *IEEE Computer*, 24(8):52–60, August 1991.

[267] D. Notkin, A. Black, and et al. E. Lazowska. Interconnecting heterogeneous computer systems. *Communication of ACM*, 31:258–273, 1988.

[268] D. Notkin, N. Hutchinson, J. Sanislo, and M. Schwartz. Heterogeneous computing environments: report on the ACM SIGOPS workshop on accomadating heterogeneity. *Communications of the ACM*, 30(2):132–140, February 1987.

[269] NSF. The National Science Foundation Metacenter. A report prepared for the program advisory committee to the National Science Foundation Division of Advanced Scientific Computing, February 1992.

[270] Robert W. Numrich and Paul R. Borman. Parallel distributed supercomputing in a heterogeneous environment using high-speed communications links. In Jack Dongarra, Ken Kennedy, Paul Messina, Danny C. Sorensen, and Robert G. Voigt, editors, *Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing*, pages 484–490, Hoston, TX, March 25-27, 1991. SIAM.

[271] Ronald A. Olsson, Gregory R. Andrews, Michael H. Coffin, and Gregg M. Townsend. SR: A language for parallel and distributed programming. Technical Report TR 92-09, Department of Computer Science, The University of Arizona, Tucson, Arizona, March 1992.

[272] OSF. OSF distributed computing environment, release 1.0 developer's kit documentation set (preliminary). Technical report, Open Software Foundation, 11 Cambridge Center, Cambridge, MA, February 1991.

[273] J. Ousterhout, A. Cherenson, F. Douglis, M. Nelson, and B. Welch. The Sprite network operating system. *IEEE Computer*, 21(2):23–36, February 1988.

[274] Joe Panepinto. Unix Linda enables parallel processing. *Digital News*, October 14, 1991.

[275] Lewis I. Patterson, Richard S. Turner, Robert M. Hyatt, and Kevin D. Reilly. Construction of a fault-tolerant distributed tuple-space. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[276] Ronald Peierls and Graham Campbell. ALMS - programming tools for coupling application codes in a network environment. In *Proceedings of the Heterogeneous Network-Based Concurrent Computing Workshop*, Tallahassee, FL, October 1991. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.91.

[277] Robert Pennington. New workstation super cluster available soon. *Pittsburg Supercomputer Center News*, pages 7–7, July/August 1992.

[278] Michael Pernice. Network parallel solution of nonsymmetric partial differential equations. Technical Report USI-37, Utah Supercomputing Institute, The University of Utah, Salt Lake City, Utah, 1992.

[279] J. W. Peterson. Distributed computation for computer animation. In *Fourth USENIX Computer Graphics Workshop*, pages 24–36, 1987.

[280] E. J. Plaskacz, M. R. Ramirez, and Sanjeev Gupta. On distributed processing applications in finite element analysis. *Engineering Mechanics*, pages 107–110, 1992.

[281] Roldan Pozo. A stream-based interface in C++ for programming heterogeneous systems. Department of Computer Science, University of Tennessee, 1992.

[282] J. Purtilo. *A software interconnection technology to support specification of computational environments*. PhD thesis, Department of Computer Science, University of Illinois Urbana-Champaign, 1986. Report UIUC-R-86-1269.

[283] J. Purtilo and P. Jalote. An environment for prototyping distributed applications. In *Proceedings of the Ninth International Conference on Distributed Computing Systems*, pages 588–594, June 1989.

[284] J. Purtilo and P. Jalote. An environment for prototyping distributed applications. *Computer Languages*, 16(3/4):197–207, 1991.

[285] J. Purtilo, D. Reed, and D. Grunwald. Environments for prototyping parallel algorithms. *Journal of Parallel and Distributed Computing*, 5:421–437, 1988.

[286] James M. Purtilo. The Polylith software bus. Technical Report TR-2469, Computer Science Department, University of Maryland, College Park, Maryland, 1990.

[287] A. Quealy, J. Cole, and R. Blech. Portable programming on parallel/networked computers using the application portable parallel library (APPL). NASA Technical Manual, to appear, Fall 1992.

[288] M. Quinn and N. Nedelijkovic. Data-parallel programming on a network of heterogeneous workstations. In *First International Symposium on High Performance Distributed Computing*, pages 28–36, Syracuse, NY, September 1992.

[289] Alan Radding. Attack of the killer micros. *RS/Review*, August 1992. Supplement to LAN Computing magazine.

[290] Jodi Reck-Schrobilgen. HP, Convex team for meta series. *High-performance computing review*, pages 36–37, Winter 1992/1993.

[291] Donna S. Reese and Ed Luke. Object oriented Fortran for development of portable parallel programs. In *Proceedings of the 3rd IEEE Symposium on Parallel Distributed Processing*, pages 608–615, Dallas, TX, December 1991.

[292] Louis S. Revor. DQS users guide. Computing and Telecommunciations Division, Argonne National Laboratory, September 1992.

[293] Martin C. Rinard, Daniel J. Scales, and Monica S. Lam. Heterogeous parallel programming in Jade. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[294] E. S. Roberts and J. R. Ellis. Parmake and Dp: experience with a distributed, parallel implementation of make. In *Proceedings of the 2nd Workshop on Large-Grained Parallelism*, pages 74–76, Pittsburgh, PA, November 1987. Also available as Carnegie-Mellon University Technical Report CMU/SEI-87-SR-5.

[295] J. L. Rogers, K. C. Young, and J. M. Barthelemy. Distributed computer system enhances productivity for SRB joint optimization. Technical Memorandum NASA-TM-89108, NASA Langley Research Center, 1987.

[296] Ward Rosenberry, David Kenney, and Gerry Fisher. *Understanding DCE*. O'Reilly & Associates, 1992.

[297] Matthew Rosing, Robert B. Schnabel, and Robert P. Weaver. The DINO parallel programming language. Technical Report CU-CS-457-90, Computer Science Department, University of Colorado at Boulder, April 1990.

[298] M. Rozier, V. Abrossimov, F. Armand, I. Boule, M. Gien, M. Guillemont, F. Hermann, C. Kaiser, S. Langlois, P. Leonard, and W. Neuhauser. CHORUS distributed operating system. *Computing Systems*, 4(1):299–328, Fall 1988.

[299] M. Rozier and J. L. Martins. *Distributed Operating Systems: Theory and Practice*, chapter The CHORUS distributed operating system: some design issues, pages 262–287. Springer-Verlag, Berlin, 1987.

[300] R. J. Sahulka, E. C. Plachy, L. J. Scarborough, R. G. Scarborough, and S. W. White. Fortran for clusters of IBM ES/3090 multiprocessors. *IBM Systems Journal*, 30:296–311, 1991.

[301] Harjinder S. Sandhu and Songnian Zhou. Cluster-based file replication in large-scale distributed systems. Technical Report CSRI-255, Computer Systems Research Institute, University of Toronto, Toronta, Canada, January 1992.

[302] J. Sanz, K. Pischel, and D. Hubler. The engine design engine. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[303] Alan D. Saunders. *The QBATCH system.* Vita Fibres Ltd., 1991.

[304] L. J. Scarborough, R. G. Scarborough, and S. W. White. Clustering IBM Enterprise System/3090 computers for parallel execution of Fortran programs. *IBM Journal of Research and Development*, 35(5/6):667–679, September/November 1991.

[305] Steve Scearce. Network Linda builds a "virtual supercomputer" on LANs. *RiScWorld*, 1(3), August 1991.

[306] Richard E. Schantz, Robert H. Thomas, and Girome Bono. The architecture of the Cronus distributed operating system. In *Proceedings of the 6th International Conference on Distributed Computing Systems*, pages 250–259. IEEE Computer Society, May 1986.

[307] Richard D. Schlichting and Vicraj T. Thomas. FT-SR: A programming language for constructing fault-tolerant distributed systems. Technical Report TR 92-31, Department of Computer Science, The University of Arizona, Tucson, Arizona, 1992.

[308] G. Schoinas. Issues on the implementation of PrOgramming SYstem for distriButed appLications - a free Linda implementation for Unix networks. Draft.

[309] Michael Schrage. Piranha processing - utilizing your down time. *HPCwire (Electronic Newsletter)*, August 1992.

[310] L. Seidl and J. Wrabetz. NetMake technology summary. In *Sun User Group Tenth Annual Conference Proceedings*, December 1992.

[311] Mark Seymour. Clustered Risc System/6000 workstations. IBM Technical Computing Systems, Dallas, TX, 1991.

[312] A. H. Sherman. *C-Linda Reference Manual.* Scientific Computing Associates, Inc., 1990.

[313] A. H. Sherman. A hybrid approach to parallel compositional reservoir simulation. In *Proceedinngs of the 24th Offshore Technology Conference*, pages 191–197, May 1992.

[314] Yuan Shi and Kostas Blathras. Parallelizing "Scatter-and-Gather" applications using heterogeneous networked workstations. In Jack Dongarra, Ken Kennedy, Paul Messina, Danny C. Sorensen, and Robert G. Voigt, editors, *Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing*, pages 588–595, Hoston, TX, March 25-27, 1991. SIAM.

[315] Mark A. Shifman, Andreas Windemuth, Klaus Schulten, and Perry L. Miller. Molecular dynamics simulation on a network of workstations using a machine-independent parallel programming language. Center for Medical Informatics and Department of Anesthesiology, Yale University School of Medicine.

[316] B. Shirazi and David D. H. Lin. An asynchronous communication protocol for heterogeneous distributed programming. In *Proceedings 1990 International Conference on Parallel Processing*, 1990.

[317] John Shirley. *Guide to writing DCE applications.* O'Reilly & Associates, 1992.

[318] Ellen H. Siegel and Eric C. Cooper. Implementing distributed Linda in Standard ML. Technical Report CMU-CS-91-151, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, October 1991.

[319] G. Singh, F. Westervelt, and S. Abraham. Computing radiosity on a high performance workstation LAN. In *First International Symposium on High Performance Distributed Computing*, Syracuse, NY, September 1992.

[320] R. C. Sinkwitz. Particle motion simulation - a parallel distributed workstation application for real time. In *Proceedings of the Heterogeneous Network-Based Concurrent Computing Workshop*, Tallahassee, FL, October 1991. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.91.

[321] D. B. Skillicorn. Practical concurrent programming for parallel machines. *Computer Journal*, 34:302–310, 1991.

[322] A. Skjellum and A. Leung. Zipcode: a portable multicomputer communications library atop the Reactive Kernel. In *Proceedings of the 5th Distributed Memory Computer Conference*, pages 767–776. IEEE Press, 1990.

[323] Glen Smith. Object-oriented Fortran tutorial. Engineering Research Center, Mississippi State University, 1992.

[324] Norris Parker Smith. IBM supercomputing competition: Prize-winners show cost-effectiveness/performance of distributed systems of networked workstations. *Supercomputing Review*, pages 43–44, July 1991.

[325] Norris Parker Smith. Network Linda achieves near-linear speed-ups on multisystem networks. *Supercomputing Review*, pages 53–54, May 1991.

[326] Norris Parker Smith. How an innocent workstation cluster became notorious. *HPCwire (Electronic Newsletter)*, September 1992.

[327] Norris Parker Smith. Supercomputing '92: Will clusters pass muster. *HPCwire (Electronic Newsletter)*, November 1992.

[328] Norris Parker Smith. Clusters offer tasty, low-budget alternative. *High-Performance Computing Review*, pages 3, 6, February 1993.

[329] R. G. Smith and R. Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):61–70, 1981.

[330] L. Stapleton. Meet the Metacomputer: Where the network is the computer. *Supercomputing Review*, pages 42–44, March 1991.

[331] Lisa Stapleton. Grad student releases public-domain Linda. *Supercomputing Review*, pages 18–18, October 1991.

[332] W. Timothy Strayer. Mentat tutorial. Department of Computer Science, University of Virginia, Charlottesville, VA, December 1992.

[333] M. Stumm and S. Zhou. Algorithms implementing distributed shared memory. *IEEE Computer*, 23(5):54–64, May 1990.

[334] Robert H. Sues, Yuan Jie Lua, and Mark D. Smith. PFRAM's parallel solution can determine fatique life. *High-Performance Computing Review*, pages 14–15,30–31, Winter 1992/1993.

[335] M. Sullivan and D. Anderson. Marionette: a system for parallel distributed programming using a master/slave model. In *IEEE 9th International Conference on Distributed Computing Systems*, pages 181–188, June 1989.

[336] Sun Microsystems. *External Data Representation Reference Manual*. Sun Microsystems, January 1985.

[337] Sun Microsystems. *XDR: External data representation standard*. Sun Microsystems, June 1987. RFC 1014, 20 pages.

[338] B. S. Sunderam. Design issues in heterogeneous network computing. In *Proceedings, Workshop on Heterogeneous Processing*, pages 101–112, Beverly Hills, CA, March 1992.

[339] V. Sunderam. A fast transaction oriented protocol for distributed applications. In *Proceedings of the Winter Usenix Conference*, pages 79–87, February 1988.

[340] V. Sunderam. PVM: A framework for parallel distributed computing. *Concurrency: Practice and Experience*, 2(4):315–339, December 1990.

[341] V. S. Sunderam and G. A. Geist. *Scientific Excellence in Supercomputing*, chapter Supercomputer level concurrent computation on a network of IBM RS/6000 powerstations, pages 779–804. Baldwin Press, March 1992. K.Billingsley, H. Brown, E. Derohanes (eds).

[342] Vaidy Sunderam. Concurrent computing with PVM. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[343] Darryl K. Taft. Energy lab cultivating RISC/6000s. *Government Computer News*, 10(22), October 28, 1991.

[344] Darryl K. Taft. Supercomputer? no, it's a cluster of Risc/6000s. *Government Computer News*, pages 27–27, December 23, 1991.

[345] A. Tanenbaum, M. F. Kaashoek, and H. E. Bal. Parallel programming using shared objects and broadcasting. *IEEE Computer*, 25(8):10–19, August 1992.

[346] A. Tanenbaum, R. van Renesse, H. van Staveren, G. Sharp, S. Mullender, A. Jansen, and G. van Rossum. Experiences with the Amoeba distributed operating system. *Communications of the ACM*, 33(12):46–63, December 1990.

[347] A. S. Tanenbaum and R. van Renesse. Distributed operating systems. *ACM Computing Surveys*, 17(4):419–470, December 1985.

[348] Jeff Terstriep. *DTM: data transfer mechanism.* National Center for Supercomputer Applications, University of Illinois Urbana-Champaign, December 1990. User Documentation.

[349] M. M. Theimer and B. Hayes. Heterogeneous process migration by recompilation. In *Proceedings 11th International Conference on Distributed Computing Systems*, pages 18–25, Arlington, TX, May 1991.

[350] Philip C. Treleaven. Parallel architecture overview. *Parallel Computing*, 8:59–70, 1988.

[351] Ulf Dahlen, Department of Computer and Information Science, Linkoping University, Linkoping, Sweden. *Scheme–Linda*, September 1990. EPCC-TN90-06.

[352] R. van Renesse, A. S. Tanenbaum, and G. J. Sharp. The workstation: computing resource or just a terminal. In *Proceedings of the Workshop on Workstation Operating Systems*, Cambridge, MA, November 1987.

[353] R. van Renesse, A. S. Tanenbaum, H. van Staveren, and J. Hall. Connecting RPC-based distributed systems using wide-area networks. In *Proceedings of the Seventh International Conference on Distributed Systems*, pages 28–34, West Berlin, September 1987. IEEE.

[354] R. van Renesse, H. van Staveren, and A. S. Tanenbaum. Performance of the world's fastest distributed operating system. *Operating Systems Review*, 22:25–34, October 1988.

[355] R. van Renesse, H. van Staveren, and A. S. Tanenbaum. The performance of the Amoeba distributed operating system. *Software-Practice and Experience*, 19:223–234, March 1989.

[356] Robbert van Renesse, Ken Birman, Robert Cooper, Bradford Glade, and Patrick Stephenson. Reliable multicast between microkernels. Computer Science Department, Cornell University.

[357] R. J. Vetter, D. H. C. Du, and A. E. Klietz. Network supercomputing: experiments with a Cray-2 to CM-2 HiPPI connection. In *Proceedings of Workshop on Heterogeneous Processing at 6th International Parallel Processing Symposium*, 1992. To appear.

[358] Stephen T. Vinter. Integrated distributed computing using heterogeneous systems. *SIGNAL*, 43(10):157–162, June 1989.

[359] Stephen T. Vinter, Nilkanth Phadnis, and Richard Floyd. Distributed query processing in Cronus. In *Proceedings of the 9th International Conference on Distributed Computing Systems*, pages 414–422. IEEE Computer Society, June 1989.

[360] Edward F. Walker, Richard Floyd, and Paul Neves. Asynchronous remote operation execution in distributed systems. In *Proceedings of the 10th International Conference on Distributed Computing Systems*, pages 253–259. IEEE Computer Society, May 1990.

[361] S. Wallach. What will the next generation supercomputer look like? *Supercomputing Review*, 1990.

[362] J. Wang, X. Zheng, and S. Zhou. Lstcsh user's manual. Computer Systems Research Institute, University of Toronto, Toronto, Canada, September 1991.

[363] J. Wang, X. Zheng, and S. Zhou. UTOPIA applications interface. Computer Systems Research Institute, University of Toronto, Toronto, Canada, August 1991.

[364] S. White, D. Torney, and C. Whittaker. A parallel computational approach using a cluster of IBM ES/3090 600Js for physical mapping of chromosomes. In *Proceedings of Supercomputing 1990*, pages 112–121, New York, November 1990. IEEE Computer Socity Press.

[365] Robert A. Whiteside and Jerrold S. Leichter. Linda for supercomputing on a local area network. In *Proceedings of Supercomputing 1988*, pages 192–199, Orlando, Florida, October 1988.

[366] Greg Wilson. Linda-like systems and their implementation. Technical Report 91-13, Edinburgh Parallel Computing Centre, June 1991.

[367] Stephen Wolbers. Parallel and clustered computing at Fermilab. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[368] Alexander Wolfe. Parallel tools kick applications into high gear. *Electronic Engineering Times*, Monday, March 30, 1992.

[369] P. H. Worley. A new PICL trace file format. Technical Report TM-12125, Oak Ridge National Laboratory, June 1992.

[370] Joan Wrabetz. Wasting away: does distributed computing maximize network resources. *Workstation News*, 4(2):1,12, February 1993.

[371] M. J. Yamasaki. Distributed library. Technical Report RNR-90-008, NAS Systems Division, NASA Ames Research Center, Moffett Field, CA, April 1990.

[372] M. J. Yamasaki. A distributed library tutorial. NAS Systems Division, NASA Ames Research Center, Moffett Field, CA, January 1990.

[373] M. J. Yamasaki. Cooperative visualization of computational fluid dynamics. NAS Systems Division, NASA Ames Research Center, Moffett Field, CA, May 1991.

[374] S. Yemini and et al. G. Goldszmidt. CONCERT: a high-level language approach to heterogeneous distributed systems. In *Proceedings of the 9th Conference on Distributed Computing Systems*, pages 162–171, June 1989.

[375] Alan G. Yoder and David L. Cohn. SPL: a language for distributed computing. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[376] S. Zhou. Heterogeneous distributed shared memory. Technical Report 244, Computer Systems Research Institute, University of Toronto, September 1990. To appear: IEEE Transactions on Parallel and Distributed Systems.

[377] S. Zhou, M. Stuum, and T. McInerney. Extending distributed shared memory to heterogeneous environments. In *IEEE 10th International Conference on Distributed Computing Systems*, pages 30–37, 1990.

[378] Songnian Zhou. LSF: load sharing in large-scale heterogeneous distributed systems. In *Proceedings of the Workshop on Cluster Computing*, Tallahassee, FL, December 1992. Supercomputing Computations Research Institute, Florida State University. Proceedings available via anonymous ftp from ftp.scri.fsu.edu in directory pub/parallel-workshop.92.

[379] Songnian Zhou, Jingwen Wang, Xiaohu Zheng, and Pierre Delisle. UTOPIA: a load sharing facility for large, heterogeneous distributed computer systems. Technical Report CSRI-257, Computer Systems Research Institute, University of Toronto, Toronto, Canada, April 1992.

A. Glossary

associative memory: memory which can be accessed by content, rather than address. (Tuple space is an associative memory.)

asynchronous: not guaranteed to enforce coincidence in time, e.g., the send and receiver may or may not both be engaged in the operation at the same instant in time.

availability policy: determines how a resource participates in the pool of resources [208].

blocking/nonblocking communication: a communicating process cannot perform any other operations until communication has completed (blocking).

cluster computing: A subset of metacomputing where the resources are located within close proximity of one another.

computational ecology: application processes behave like organisms that spawn and oversee offspring [208].

data coercion: the process of translating data formats between computers with different internal data representations. An essential characteristic of heterogeneous computing.

distributed computer: a computer made up of many smaller, and potentially independent, computers, such as a network of workstations.

distributed memory: an architecture in which each processor has its own memory, not accessible directly by other processors. Communications between nodes is typically accomplished by message passing.

ensemble computing: Synonym of piranha computing [309].

fault tolerance: providing systems able to check for errors and operate in degraded configurations in the presence of faults.

generative communications: a parallel computing paradigm in which processes may be spawned dynamically, turning into data upon completion, and data may be stored in tuples.

granularity: the size of the operations performed by a process between communication events.

heterogeneous: containing components of more than one kind. It is common for different vendors to store information (floating point numbers, etc) in a unique manner which inhibits the direct use by a computer from a different vendor.

hypercomputing: transforms an ordinary LAN into a virtual supercomputer that uses idle processing cycles of LAN-attarched computers [208]. (Synonym of metacomputing).

latency: the time taken to service a request, deliver message, etc., which is independent of the size or nature of the operation.

loosely-coupled: a parallel system which consists of a set of processors, each of which has its own memory and processes information independently from other processors. Processors communicate and synchronize their activities by sending and receiving messages.

metacomputing: a heterogeneous computing environment where parallel/vector supercomputers, workstations, and MPP systems all work seamlessly together to solve complex problems. (Term coined by Smarr of NCSA.) [290]

meta-system: system composed of heterogeneous hosts (both parallel and conventional), possibly controlled by separate organizational entities, and connected by an irregular interconnection network [171].

multicast: to send a message to many, but not necessarily all, possible recipient processes.

multicomputer: couples processors and memories to form cost-effective computer node. Place the same program in all nodes and allocate data across all computers to minimize moving data [49].

network-based concurrent computing: Synonym of metacomputing.

piranha computing: a system which dynamically dispatches tasks to available computer resources on the network, thus, stealing unused cycles [309].

shared memory: an architecture in which there is a single pool of memory to which all processors have access.

stub: a program module specific to an RPC interface that supports remote procedure calls for the interface. Stubs are linked with client and server applications and hide the intricacies of remote procedure calls from the application code.

tuple: an ordered sequence of fixed length of values of arbitrary types.

ultracomputing: Synonym of metacomputing.

undirected programming: a pool of resources and workers are defined and the system determines the optimal mapping for a problem [368].

virtual heterogeneous machine: Synonym of metacomputing.

virtual synchrony: complex distributed events appear to occur one-at-a-time, "in synchrony". (In reality, many events happen concurrently, exploiting parallelism of the distributed computing environment.

workstation cluster: A metacomputer constructed using workstations.

workstation farm: Synonym of workstation cluster.

B. Software Summary

The product descriptions and other information contained in this section were collected from numerous sources. In many cases the product descriptions have been extracted directly from literature provided by the developer. The summaries are intended to be used to briefly describe the product and to identify sources for additional information.

The author realizes that many pertinent products related to the subject matter may have been excluded from this summary. Additionally, there may be entries which need to be updated. The author will gladly consider additions to this section and can be reached at the address provided in the Preface. (Note that each page includes the date of the most recent modification.)

Category	Product(s)
Batch Job Distribution	BALANS Condor DNQS/FSU (Distributed Network Queuing System) DNQS/McGill (Distributed Network Queuing System) DQS (Distributed Queuing System) LSF (Load Sharing Facility) NC Toolset (Network Computing Toolset) NQS/Exec (Network Queuing System/Executive) PBS (Portable Batch System) QBATCH (Queued BATCH processing system) QuaHog Task Broker UniJES (Unix Job Entry System) Utopia
Collaboration	Collage (COLLaborative Analysis and Graphics Environment)
Distributed Applications	NetMake
Distributed Application Development Tools	Cronus DAS (Distributed Application Support package) dlib (distributed library) ELROS (an Embedded Language for Remote Operations Service) HRPC (Heterogeneous Remote Procedure Call) Isis/Cornell Isis/Isis Distributed Systems, Inc. NCS (Network Computing System) NetShare POET (Parallel Object-oriented Environment and Toolkit) Polylith Schooner
Distributed Shared Memory	ALMS (Asynchronous Linked Memory System) DSM (Distributed Shared Memory) Mermera
	continued on next page

continued from previous page

Category	Product(s)
	Mether
Message Passing Utilities	CHIMP (Common High level Interface to Message Passing)
	DTM (Data Transfer Mechanism)
	MPI (Message Passing Initiative)
	PICL (Portable Instrumented Communications Library)
Parallel Programming Languages	DINO (DIstributed Numerically Oriented language)
	Fortran M (Fortran Modular)
	Fortran-P
	Jade
	Mentat
	Oasis (Object and Agent Specification and Implementation System)
	OOF (Object-Oriented Fortran)
	PCN (Program Composition Notation)
	SPL (Spreadsheet Programming Language)
	SR (Synchronizing Resources)
	xHPF
Parallel Programming Systems	ADAPTOR (Automatic DAta Parallelism TranslatOR)
	APPL (Application Portable Parallel Library)
	Chameleon
	CircL
	DoPVM (Distributed object PVM)
	Express
	FT-PVM (Fault Tolerant PVM)
	HeNCE (Heterogeneous Network Computing Environment)
	Linda
	Network Linda
	p4 (Portable Programs for Parallel Processors)
	ParNet
	Piranha Linda
	POSYBL (Programming System for Distributed Applications)
	PowerTools
	PVM (Parallel Virtual Machine)

continued on next page

continued from previous page	
Category	Product(s)
	PVM++ Spawn TCGMSG (Theoretical Chemistry Group MeSsaGe passing system TOPSYS (TOols for Parallel SYStems) Tuplex
Performance Monitoring and Debugging Utilities	ParaGraph Upshot Xab (X-window Analysis and deBugging)
Scientific Libraries	Chameleon Tools DLAPACK (Distributed memory LAPACK) DSSLIB (Dakota Scientific Software LIBrary)

Table B.1: Product Listing by Category

SOFTWARE	Adaptor (Automatic DAta Parallelism TranslatOR)
DEVELOPER	German National Research Institute for Computer Science.
DESCRIPTION	Adaptor is a tool that transforms data parallel programs written in Fortran with array extensions, parallel loops, and layout directives to parallel programs with explicit message passing. Adaptor supports heterogeneous workstation configurations via the use of PVM as the message passing model.
AVAILABILITY	High Performance Computing Center German National Research Institute for Computer Science P.O. Box 1316 D-5205 Sankt Augustin 1 Federal Republic of Germany Contact: Thomas Brandes (`brandes@gmdzi.gmd.de`) Source code/documentation/reference material available via anonymous ftp: `ftp.gmd.de` in directory: `/gmd/adaptor`
COMMENTS	Excellent documentation.
REFERENCES	[75], [76], [77], [78], [79].

SOFTWARE	ALMS (Asynchronous Linked Memory System)
DEVELOPER	Brookhaven National Laboratory.
DESCRIPTION	ALMS is a toolkit for assembling complex parallel applications from independently coded modules. ALMS is based on data sharing in a distributed environment. ALMS is a distributed shared memory model which only requires that all cooperating processes have an identical view of the sequence of successive values stored at a particular address during the course of the computation, without requiring that they all see the same member of the sequence at the same time.
AVAILABILITY	Brookhaven National Laboratory Upton, NY 11973 Contact: Ronald Peierls (`peierls@bnl.gov`)
COMMENTS	None.
REFERENCES	[25], [276].

SOFTWARE	APPL (Application Portable Parallel Library)
DEVELOPER	NASA Lewis Research Center.
DESCRIPTION	APPL is a subroutine-based communication library, intended to provide a consistent programmer interface to a variety of MIMD multiprocessors on the market today. APPL presently supports Intel Delta, Intel iPSC/860, Alliant FX/80, SGI workstations (multiprocessor or networked), IBM RS/6000 workstations (multiprocessor or networked), and networked Sun workstations.
AVAILABILITY	NASA Lewis Research Center Internal Fluid Mechanics Division Cleveland, Ohio 44135 Contact: Angela Quealy (`fsang@kira.lerc.nasa.gov`) APPL is available by completing a licensing agreement.
COMMENTS	Does not presently support heterogeneous network configurations.
REFERENCES	[287].

SOFTWARE BALANS

DEVELOPER VXM Technologies Inc.

DESCRIPTION BALANS is provides both dynamic load balancing and a distributed batch queuing system. Distributed batch processing, distributed RPC-based programs, distributed compiles, and network parallel processing can all be performed using BALANS. BALANS includes an optional application program interface (API) for developing fully distributed applications. Heterogeneous environments are supported by BALANS-IDS (Interoperable Distributed System).

AVAILABILITY VXM Technologies Inc.
30 Gardner Road
Suite 3G
Brookline, MA 02146
(617) 730-8490

Supports: SGI, IBM, Sun, DEC, and HP workstations.

Various configurations and pricing are available.

COMMENTS Marketed as a competitive product to NQS/Exec and HP Task Broker.

REFERENCES

SOFTWARE	Chameleon
DEVELOPER	Argonne National Laboratory.
DESCRIPTION	Chameleon provides a standardized and extremely low overhead interface to message passing software. It provides a uniform interface to program startup, and simplifies the use of clusters of workstations or other computers (including MPP). Code written using Chameleon, in combination with p4 or PVM, will run unchanged on a collection of workstations.
AVAILABILITY	Argonne National Laboratory Mathematics and Computer Science Division 9700 S. Cass Avenue Argonne, IL 60439 Contact: Bill Gropp (`gropp@mcs.anl.gov`) Source code/documentation/reference material available via anonymous ftp: `info.mcs.anl.gov` in directory: `/pub/mpi`
COMMENTS	The Tools library which was developed using Chameleon includes a wide range of functions for sparse linear algebra, iterative solvers, etc. and is available from the same site as Chameleon in the directory (`/pub/pdetools`).
REFERENCES	[174].

SOFTWARE	CHIMP (Common High level Inteface to Message Passing)
DEVELOPER	Edinburgh Parallel Computing Center.
DESCRIPTION	CHIMP provides a basic message passing system for transfer of single messages between end points. CHIMP has both blocking and non-blocking communications, multicasting of messages, and message selection. The interface is available with both Fortran and C language bindings. All of the current systems of CHIMP are built upon the manufacturer's existing message passing software, except for workstation clusters (built on TCP/IP).
AVAILABILITY	Edinburgh Parallel Computing Center The University of Edinburgh United Kingdom Documentation/reference material available via anonymous ftp: `epcc.ed.ac.uk` in directory: `/pub/chimp`
COMMENTS	Supports: SUN, SGI, IBM RS/6000, Meiko T800, Meiko i860, Fujitsu AP1000, CM-5 (scheduled), Intel iPSC (scheduled).
REFERENCES	[109], [110], [111], [112], [114].

SOFTWARE CircL

DEVELOPER Cogent Research.

DESCRIPTION Cogent Research has ceased operations. See comments below.

AVAILABILITY Not available.

COMMENTS A new company has splintered off from this effort and is presently developing a product for parallel computing across networked workstations. The plan for this product, presently called Cleo, is to support C, C++, and Fortran bindings. This product is presently in the development stage. For additional information contact:

Andrew Burke
Paranet Computing, Inc.
2234 SE 35th Place
Portland, Oregon 97214
(503) 231-0489
email: `aburke@paranet.rain.com`

REFERENCES

SOFTWARE	Collage (Collaborative Analysis and Graphics Environment)
DEVELOPER	University of Illinois Urbana-Champaign.
DESCRIPTION	Collage is a collaborative data analysis tool. Multiple people, via their networked workstations, can perform data analysis in an interactive workgroup fashion. Collage supports: data viewing (spreadsheet and image), data analysis (histogram, x-y plots, etc), animation, pallette maniputation, text editing, shared whiteboard, paint operations, HDF browser, and screen capture.
AVAILABILITY	University of Illinois Urbana-Champaign National Center for Supercomputer Applications Project team email: `softdev@ncsa.uiuc.edu` Source code/documentation/reference material available via anonymous ftp: `ftp.ncsa.uiuc.edu` in directory: `/Collab`
COMMENTS	Supports: DEC, SGI, SUN, IBM RS/6000 workstations.
REFERENCES	

SOFTWARE	Condor
DEVELOPER	University of Wisconsin - Madison.
DESCRIPTION	Condor is a software environment which automatically dispatches and relocates jobs within a participating set of networked workstations. Jobs prepared and submitted to Condor will be dispatched to workstations which have idle CPU cycles. Each workstation owner determines the method of participation for his/her workstation.
AVAILABILITY	University of Wisconsin - Madison Computer Sciences Department Contact: Mike Litzkow (608) 262-6122 (`mike@cs.wisc.edu`) Project team email: `condor-request@cs.wisc.edu` Source code/documentation/reference material available via anonymous ftp: `fyvie.cs.wisc.edu` in directory: `/condor`
COMMENTS	Requires relinking application programs with the Condor runtime libraries.
REFERENCES	[82], [228], [229], [230].

SOFTWARE	Cronus
DEVELOPER	BBN Systems and Technologies.
DESCRIPTION	Cronus is an object-oriented distributed computing environment. The goal of Cronus is to provide a coherent and integrated environment in which distributed applications can easily be built. Cronus includes: a complete environment to develop integrated distributed applications, an application development environment which reduces development time, operation in a heterogeneous environment, and interoperability and coexistance with existing systems (e.g. Unix, VMS, etc) and applications.
AVAILABILITY	BBN Systems and Technologies Project team email: `cronus-help@bbn.com` Source code/documentation/reference material available via anonymous ftp: `pineapple.bbn.com`
COMMENTS	Platforms supported: Alliant FX/80, BBN Butterfly GP1000, DEC RISC 3100, DEC VAX, Encore Multimax, Masscomp, SUN 2, SUN 3, SUN 4, SUN 386i.
REFERENCES	[8], [55], [56], [62], [97], [152], [199], [263], [306], [358], [359].

SOFTWARE	DAS (Distributed Application Support package)
DEVELOPER	University of British Columbia
DESCRIPTION	DAS is a software package which provides the remote operations service defined by the CCITT standard X.219 using the protocol defined by the CCITT standard X.229. DAS also provides access to CASN1, an ASN.1 compiler. This compiler enables DAS applications to encode application level data into an external representation suitable for transmission over a heterogeneous computer network.
AVAILABILITY	University of British Columbia Canada Source code/documentation/reference material available via anonymous ftp: `cs.ubc.ca` in directory: `/pub/local/src/das`
COMMENTS	None.
REFERENCES	[22].

SOFTWARE	DINO (DIstributed Numerically Oriented language)
DEVELOPER	University of Colorado.
DESCRIPTION	DINO is a C augmented language for writing parallel programs for distributed memory multiprocessors. The language provides the facility to define a virtual parallel computer that is best suited for a specific computation, the ability to map distributed data structures onto this virtual machine, and the ability to define procedures that will run on each processor of the virtual machine simultaneously. Networked computers are supported.
AVAILABILITY	University of Colorado Contact: Bobby Schnabel (`bobby@lupine.colorado.edu`) Source code/documentation/reference material available via anonymous ftp: `cs.colorado.edu` in directory: `/pub/cs/distribs/dino`
COMMENTS	Does not include a Fortran binding.
REFERENCES	[121], [297].

SOFTWARE	DLAPACK (Distributed Memory LAPACK)
DEVELOPER	Oak Ridge National Laboratory.
DESCRIPTION	DLAPACK is a proposed distributed memory implementation of LAPACK. DLAPACK will use PVM for message passing.
AVAILABILITY	Oak Ridge National Laboratory Mathematical Sciences Division P.O. Box 2008, Bldg 6102 Oak Ridge, TN 37831-6367 Contact: Jack Dongarra Documentation/reference material available via anonymous ftp: `cs.utk.edu` in directory: `/pub`
COMMENTS	DLAPACK is the name presently being used for this effort.
REFERENCES	[115], [116].

SOFTWARE	dlib (Distributed Library)
DEVELOPER	NASA Ames Research Center.
DESCRIPTION	dlib is a tool for developing distributed applications for computational environments which consist of a heterogeneous network of computer systems. dlib consists of a library of functions which handles all the functions of distributing the application.
AVAILABILITY	Numerical Aerodynamic Simulation Systems Division NASA Ames Research Center Moffett Field, CA 94035-1000 Contact: Michael Yamasaki (`yamo@nas.nasa.gov`)
COMMENTS	dlib was developed primarily to create visualization application software which performs computationally intense calculations on a supercomputer and performs the rendering and display on a workstation.
REFERENCES	[371], [372], [373].

SOFTWARE DNQS (Distributed Network Queuing System)

DEVELOPER Florida State University.

DESCRIPTION DNQS is an experimental Unix based network queuing system. DNQS is used to allocate batch jobs in a heterogeneous environment.

AVAILABILITY Supercomputer Computations Research Institute
Florida State University
400 Dirac Science Library B-186
Tallahassee, Florida 32306-4052
(904) 644-0190

Source code/documentation/reference material available via
anonymous ftp: `ftp.scri.fsu.edu`
in directory: `/pub/DNQS`

COMMENTS None.

REFERENCES [167].

SOFTWARE	DNQS (Dynamical Network Queuing System)
DEVELOPER	McGill University, Canada.
DESCRIPTION	This implementation of DNQS is a rewrite of the DNQS available from FSU. This version makes several changes to the basic implementation from FSU; root access from all machines to server is no longer required, socket connections have been implemented, and various formats changed.
AVAILABILITY	Department of Physics McGill University Montreal, Quebec, Canada Contact: Martin-Daniel Lacasse (`isaac@physics.mcgill.ca`) Source code/documentation/reference material available via anonymous ftp: `ftp.physics.mcgill.ca` in directory: `/pub/Dnqs`
COMMENTS	None.
REFERENCES	[212].

SOFTWARE	DoPVM (Distributed object PVM)
DEVELOPER	Emory University.
DESCRIPTION	DoPVM is a shared-object oriented programming extension for PVM. DoPVM is presently under development.
AVAILABILITY	Emory University
	Contact: Vaidy Sunderam (`vss@mathcs.emory.edu`)
COMMENTS	None.
REFERENCES	

SOFTWARE	DQS (Distributed Queuing System)
DEVELOPER	Florida State University.
DESCRIPTION	DQS supports queuing of single machine jobs, multinode parallel jobs, PVM parallel processing jobs, and interactive sessions across a heterogeneous networked cluster of Unix and OSF/1 workstations. DQS will automatically load balance the submitted work across available machines. Also allows workstation owners to enroll and remove their machines from the DQS pool.
AVAILABILITY	Supercomputer Computations Research Institute Florida State University 400 Dirac Science Library B-186 Tallahassee, Florida 32306-4052 (904) 644-0190 Source code/documentation/reference material available via anonymous ftp: `ftp.scri.fsu.edu` in directory: `/pub/DQS`
COMMENTS	None.
REFERENCES	[292].

SOFTWARE	DSM (Distributed Shared Memory)
DEVELOPER	Florida State University.
DESCRIPTION	DSM is a communications system based on the distributed shared memory paradigm which allows parallel programs to "share" vectors of integers and reals in a simple and intuitive fashion. DSM uses UDP (universal datagram protocol) to provide broadcasting functionality.
AVAILABILITY	Supercomputer Computations Research Institute Florida State University 400 Dirac Science Library B-186 Tallahassee, Florida 32306-4052 (904) 644-0190 Contact: Randy Langley (`langley@scri.fsu.edu`) Source code/documentation/reference material available via anonymous ftp: `ftp.scri.fsu.edu` in directory: `/pub/DSM`
COMMENTS	Presently in alpha release.
REFERENCES	

SOFTWARE	DSSLIB (Dakota Scientific Software Libary)
DEVELOPER	Dakota Scientific Software.
DESCRIPTION	DSSLIB is a library of parallelized and optimized linear-algebra routines. LINPACK and levels 1,2, and 3 of the BLAS system are included. Large computations can be run in parallel using networked workstations.
AVAILABILITY	Dakota Scientific Software 501 East St. Joseph Street Rapid City, SD 57701-3995 (605) 394-2471 Contact: Michael L. Boucher (ma.boucher@na-net.ornl.gov) or (mboucher@silver.sdsmt.edu). Ten workstation license starts at $4,995. Supports: Sun Sparc, SGI.
COMMENTS	DSSLIB is fault tolerant.
REFERENCES	[118].

SOFTWARE	DTM (Data Transfer Mechanism)
DEVELOPER	University of Illinois Urbana-Champaign.
DESCRIPTION	DTM is a message passing facility. It is designed to facilitate the creation of sophisticated distributed applications. DTM provides a method to interconnect applications at run-time, a reliable message passing with synchronization and transparent data conversion. DTM has been optimized for large messages containing 100 Kbytes up to several megabytes.
AVAILABILITY	University of Illinois Urbana-Champaign National Center for Supercomputer Applications Source code/documentation/reference material available via anonymous ftp: `ftp.ncsa.uiuc.edu` in directory: `/DTM`
COMMENTS	DTM does not support heterogeneous data formats. The user of DTM is encouraged to use XDR to encode and decode their data prior to using DTM for the transfer.
REFERENCES	[348].

SOFTWARE ELROS (An Embedded Language for Remote Operations Service)

DEVELOPER Lawrence Livermore National Laboratory.

DESCRIPTION ELROS is an embedded language for programming distributed applications using remote operations. ELROS was developed as a tool for scientists and allows them to write complete ISO application protocol software. ELROS relies on directives embedded within the user's C code and avoids the traditional use of stub generators.

AVAILABILITY Distributed Programming Environments Project
Computer Software and Networking Division
Livermore Computer Center
Lawrence Livermore National Laboratory
Livermore, CA 94550

Source code/documentation/reference material available via
anonymous ftp: `framsparc.ocf.llnl.gov`
in directory: `/pub/elros`

COMMENTS None.

REFERENCES [80], [81].

SOFTWARE Express

DEVELOPER ParaSoft Corporation.

DESCRIPTION Express is a set of tools and utilities which assists developers during all phases of parallel application development. Express consists of an automatic parallelizer, an algorithmic visualizer, a high level communications library, a parallel debugger, and a set of performance analysis tools. Express supports heterogeneous networked workstations. Both C and Fortran bindings are supported.

AVAILABILITY ParaSoft Corporation
2500 E. Foothill Bldv.
Suite 205
Pasadena, CA 91107
(818) 792-9941

Contact: Sudy Bharadwaj (`sudy@elephant.parasoft.com`)

Supports: nCube, iPSC/II, iPSC/i860, Cray, IBM/3090, transputers, workstation networks.

Various configurations and pricing are available.
Yearly maintenance: 20% of the original price.

COMMENTS None.

REFERENCES [138], [139], [209].

SOFTWARE	Fortran M (A language for Modular Parallel Programming)
DEVELOPER	Argonne National Laboratory and California Institute of Technology.
DESCRIPTION	Fortran M is a small set of extensions to Fortran 77 which support a modular approach to the design of message-passing programs. The basic paradigm of Fortran M is task parallelism.
AVAILABILITY	Argonne National Laboratory Mathematics and Computer Science Division 9700 S. Cass Avenue Argonne, IL 60439 Project team email: `fortran-m@mcs.anl.gov` Documentation/reference material available via anonymous ftp: `info.mcs.anl.gov`
COMMENTS	Does not presently support heterogeneous network configurations. A heterogeneous network version is planned.
REFERENCES	[143].

SOFTWARE	Fortran-P
DEVELOPER	Purdue University and Army High Performance Computing Research Center at the University of Minnesota.
DESCRIPTION	A restricted dialect of Fortran intended to improve the ability of the compiler to generate highly efficent code for parallel architectures.
AVAILABILITY	H. Dietz School of Electrical Engineering Purdue University West Lafayette, IN 47907
COMMENTS	Present implementations have been directed towards a single application (PPM:Piecewise-Parabolic Method). Target systems: CM-2, CM-5, MasPar MP-1. Plan to develop Fortran-P to PVM translator.
REFERENCES	[123].

SOFTWARE	FT-PVM (Fault Tolerant PVM)
DEVELOPER	Carnegie Mellon University.
DESCRIPTION	FT-PVM is a fault tolerant implementation of PVM. FT-PVM is presently under development.
AVAILABILITY	Carnegie Mellon University School of Computer Science Contact: Juan Leon (`juan.leon@cs.cmu.edu`)
COMMENTS	None.
REFERENCES	[222].

SOFTWARE HeNCE (Heterogeneous Network Computing Environment)

DEVELOPER Oak Ridge National Laboratory.

DESCRIPTION HeNCE is a X-based software program designed to assist scientists in developing parallel programs. HeNCE is built on top of PVM and is based on a parallel programming paradigm where an application program can be described by a graph. HeNCE is composed of integrated graphical tools for creating, compiling, executing, and analyzing HeNCE programs.

AVAILABILITY Oak Ridge National Laboratory
Mathematical Sciences Division
P.O. Box 2008, Bldg. 6012
Oak Ridge, TN 37831-6367

Project team email: `hence@msr.epm.ornl.gov`

Available from: `netlib@ornl.gov`

COMMENTS Requires PVM.

REFERENCES [40], [42].

SOFTWARE	HRPC (Heterogeneous Remote Procedure Call)
DEVELOPER	University of Washington.
DESCRIPTION	HRPC is capable of emulating multiple RPC protocols. The RPC protocol used for a particular sequence of calls is chosen at run-time. HRPC bindings for each system allow the application to use different underlying RPCs in a consistent manner.
AVAILABILITY	University of Washington Department of Computer Science Source code/documentation/reference material available via anonymous ftp: `june.cs.washington.edu` in directory: `/pub`
COMMENTS	None.
REFERENCES	[14].

SOFTWARE	Isis
DEVELOPER	Cornell University.
DESCRIPTION	Isis is a system for building applications consisting of cooperating, distributed processes. The Isis toolkit includes bindings for C, C++, Fortran, Common Lisp, and Ada.
AVAILABILITY	Cornell University Computer Science Department Upson Hall Ithaca, NY 14853 Source code/documentation/reference material available via anonymous ftp: `ftp.cs.cornell.edu` in directory: `/pub`
COMMENTS	Isis based utilities: News: this applications supports a collection of communication topics to which users can subscribe or post messages. Nmgr: this program manages batch-style jobs and performs load sharing in a distributed setting. Deceit: this system provides fault-tolerant NFS compatiable file storage. Meta: this is an extensive system for building fault-tolerant reactive control applications.
REFERENCES	[63], [64], [65], [135], [175].

SOFTWARE	Isis
DEVELOPER	Cornell University and Isis Distributed Systems, Inc.
DESCRIPTION	Isis is a system for building applications consisting of cooperating, distributed processes. The ISIS toolkit includes bindings for C, C++, Fortran, Common Lisp, and Ada. This is a commercial implementation of the Cornell Isis software.
AVAILABILITY	Isis Distributed Systems, Inc. Suite 200 111 Sourth Cayuga St. Ithaca, NY 14850 (607) 272-6327 email: `ids@cs.cornell.edu` Various configurations and pricing are available. Yearly maintenance: 15% of the original price (If less than $100,000).
COMMENTS	ISIS based utilities: Distributed Toolkit: an environment for distributed, fault-tolerant programming. Options include the ability to interconnect VMS systems to UNIX through Isis, and the ability to exploit hardware multicast for high performance networking. Distributed News: this applications supports a collection of communication topics to which users can subscribe or post messages. News is oriented towards banking and brokerage applications. Distributed Resource Manager: this product turns a network of workstations into a fault-tolerant software super-computer, scheduling work onto idle machines and restarting critical services when failures occur. Reliable Network File System: a fault-tolerant network file system and lock manager using Sun's standard NFS protocol. The level of file replication can be varied depending on need.
REFERENCES	[63], [64], [65], [135], [175].

SOFTWARE	Jade
DEVELOPER	Stanford University.
DESCRIPTION	Jade is a parallel programming language. Jade supports message passing machines, shared memory machines, and heterogeneous collections of workstations. C, C++, and Fortran bindings have been developed.
AVAILABILITY	Computer Systems Laboratory Stanford University An alpha implementation is available by request.
COMMENTS	None.
REFERENCES	[213], [293].

SOFTWARE Linda

DEVELOPER Scientific Computing Associates and
Yale University.

DESCRIPTION Linda adds six basic operations to C and Fortran to provide parallelization. Linda maps tasks automatically. Information is exchanged between processors via a mechanism known as "tuple space" (a concept similar to distributed shared memory). Tuples are anonymous and maintain no information regarding ownership and source. Programs modified to use Linda will execute on a variety of parallel architectures, including heterogeneous networked workstations (see Network Linda description).

AVAILABILITY Scientific Computing Associates, Inc.
One Century Tower
265 Church Street
New Haven, CT 06510-0710
(203) 777-7442
email: `software@sca.com`

Supports: Sun 4/Sparcstation, Sun 3, DECstation, SGI Personal Iris, IBM RS/6000, HP/Apollo DN10000, HP/Apollo DN3000 to DN4500, HP/Apollo 9000/700, Sequent, Encore, Cray, Convex, Intel iPSC, nCUBE.

Various configurations and pricing are available.
Yearly maintenance: 18% of the current price.

COMMENTS None.

REFERENCES [3], [4], [68], [69], [70], [87], [91], [92], [93], [94], [95], [96], [132], [160], [161], [162], [163], [219], [220], [221], [236], [242], [275], [325], [351], [366].

SOFTWARE	LSF (Load Sharing Facility)
DEVELOPER	University of Toronto and Platform Computing Corporation.
DESCRIPTION	LSF is a load sharing facility for large, heterogeneous distributed systems. Interactive and batch jobs can automatically be run on lightly loaded hosts. LSF includes LSF API, LSbatch, LStcsh, LSmake, and LSlogin. LSF supports both PVM and Linda applications.
AVAILABILITY	Computer Systems Research Institute University of Toronto Toronto, Ontario, Canada Platform Computing Corporation 203 College Street, Suite 303 Toronto, Ontario, M5T 1P9, Canada (416) 978-0458 email: `info@platform.on.ca` Contact: Songnian Zhou (`zhou@sys.toronto.edu`)
COMMENTS	LSF API: C programming library (application programming interface) for accessing the LSF internals. LSbatch: distributed batch utility for parallel and sequential jobs. LStcsh: load sharing Unix shell that runs interactive commands and applications remotely with full transparency. LSmake: a parallel make that distributes tasks to lightly loaded computers. LSlogin: remote login that starts a user session on a powerful computer with few users.
REFERENCES	[301], [333], [362], [363], [376], [377], [378], [379].

SOFTWARE	Mentat
DEVELOPER	University of Virginia.
DESCRIPTION	Mentat was designed to solve the problem of writing software for parallel and distributed systems. Mentat provides high-level abstractions for parallelism that closely match the way programmers are accustomed to writing programs. The Mentat Programming Language (MPL) is an object-oriented programming language based on C++. Mentat has been enhanced to support heterogeneous distributed environments under the the project name Mentat MetaSystems Testbed.
AVAILABILITY	University of Virginia Department of Computer Science Thornton Hall Charlottesville, Viginia 22903-2442 (804) 982-2200 Project team email: `mentat@virginia.edu` Source code/documentation/reference material available via anonymous ftp: `uvacs.cs.virginia.edu` in directories: `/pub/techreports` and `/pub/mentat`
COMMENTS	Supports: Sun 3, Sun 4, SGI, Intel iPSC/2, iPSC/860 CM-5 (planned), Intel Paragon (planned), IBM RS/6000 (planned).
REFERENCES	[168], [169], [170], [171], [172], [332].

SOFTWARE	Mermera
DEVELOPER	Boston University.
DESCRIPTION	Mermera is a DSM (Distributed Shared Memory) system which gives the programmer the choice of coherent and non-coherent behavior in the same program. Mermera provides a wide variety of cleanly related memory operations satisfying different ordering constraints thereby providing an opportunity to trade-off programming simplicity for performance.
AVAILABILITY	Computer Science Department Boston University Boston, MA 02215 (617) 353-8919 Documentation/reference material available via anonymous ftp: `cs.bu.edu` in directory: `/techreports`
COMMENTS	Currently being implemented on a network of workstations using the Isis toolkit for multicast communications.
REFERENCES	[184], [185], [186].

SOFTWARE	Mether
DEVELOPER	University of Delaware.
DESCRIPTION	Mether is a Distributed Shared Memory system that runs on networked Sun workstations. User programs access Mether address space in a manner indistinguishable from other memory.
AVAILABILITY	Supercomputing Research Center Bowie, Maryland Contact: Ronald Minnich Documentation/reference material available via anonymous ftp: `metropolis.super.org` in directory: `/pub/mether`
COMMENTS	Only runs on Sun workstations. Provides 4 Mbyte of shared memory.
REFERENCES	[249], [250].

SOFTWARE	MPI (Message-Passing Interface)
DEVELOPER	Argonne National Laboratory.
DESCRIPTION	MPI is a test implementation of the draft message-passing standard. This system was developed to allow experimentation and performance evaluations for the proposed operations.
AVAILABILITY	Argonne National Laboratory Mathematics and Computer Science Division 9700 S. Cass Avenue Argonne, IL 60439 Source code/documentation/reference material available via anonymous ftp: `info.mcs.anl.gov` in directory: `/pub/mpi`
COMMENTS	A group has been formed (similar to the HPF, High Performance Fortran, group) to propose a standard message-passing interface, MPI. For additional information regarding this effort contact David Walker (`walker@epm.ornl.gov`)
REFERENCES	[173].

SOFTWARE	NCS (Network Computing System)
DEVELOPER	Hewlett Packard Co.
DESCRIPTION	NCS is a set of tools that support heterogeneous distributed computing. NCS is an implementation for NCA (Network Computing Architecture), an architecture for distributing software applications across heterogeneous collections of coputers, networks, and programming environments. NCA has been accepted by the Open Software Foundation as the architecture for the remote procedure call portion of the Distributed Computing Environment.
AVAILABILITY	Hewlett Packard Co.
COMMENTS	Supports: HP platforms and SUN platforms.
REFERENCES	[190].

SOFTWARE NC Toolset (Network Computing Toolset)

DEVELOPER The Cummings Group.

DESCRIPTION NC Toolset is a collection of products for automatic login to the least active workstation, load sharing/distribution of jobs across the network, and administration of clusters.

AVAILABILITY The Cummings Group
PO Box 3574
Boulder, CO 80307
(303) 499-9543
email: Robert Biro `rpb@tcg.com`

COMMENTS Supports: IBM RS/6000, DEC 3000/5000, SUN, HP, SGI.

NCLOGIN: Provides interactive login to least utilized system.

NCSHARE: Provides a background job distributor and is a good batch system for virtual clusters.

NCADMIN: A cluster administration tool which simplifies performance analysis and tuning.

Plan an application programmers interface to their products in 1Q93. This will allow products like PVM to use the global resource manager to schedule processors in a cluster for parallel programming requirements.

REFERENCES

SOFTWARE	NetMake
DEVELOPER	Aggregate Computing, Inc.
DESCRIPTION	NetMake is a distributed parallel version of the Unix make utility.
AVAILABILITY	Aggregate Computing, Inc. 300 South Highway 169 Suite 400 Minneapolis, MN 55426 (800) 966-1666 (612) 546-5579 email: `info@aggregate.com` Single NetMake floating user license: $2500 Additional licenses: $1250
COMMENTS	Supports only on Sun systems running SunOS 4.1x. Requires X11R4, R5-based windowing system to run the configuration tool.
REFERENCES	[310], [370].

SOFTWARE	NetShare
DEVELOPER	Aggregate Computing, Inc.
DESCRIPTION	The NetShare services were designed to close the gap between distributed tools and distributed applications and to provide location transparency through a unification of remote execution and resource management services. NetShare includes a high performance, asynchronous service remote execution facility that preserves local process semantics across machine boundaries in a heterogeneous network. NetShare is targeted at distributed applications requiring both location transparency and network parallel processing.
AVAILABILITY	Aggregate Computing, Inc. 300 South Highway 169 Suite 400 Minneapolis, MN 55426 (800) 966-1666 (612) 546-5579 email: `info@aggregrate.com`
COMMENTS	None.
REFERENCES	[370].

SOFTWARE	Network Linda
DEVELOPER	Scientific Computing Associates.
DESCRIPTION	A Linda implementation which provides access to parallelism across a heterogeneous network of computers.
AVAILABILITY	Scientific Computing Associates, Inc. One Century Tower 265 Church Street New Haven, CT 06510-0710 (203) 777-7442 email: `software@sca.com` Supports: Sun 4/Sparcstation, Sun 3, DECstation, SGI Personal Iris, IBM RS/6000, HP/Apollo DN10000, HP/Apollo DN3000 to DN4500, HP/Apollo 9000/700. Various configurations and pricing are available. Ten node starter system: $4995. Yearly maintenance: 18% of the current price.
COMMENTS	None.
REFERENCES	[24], [72], [124], [243], [274], [305], [315], [325], [365].

SOFTWARE NQS/Exec

DEVELOPER The Cummings Group and
Sterling Software.

DESCRIPTION NQS/Exec provides automatic load leveling and full NQS (Network Queueing System) batch queueing capability across a heterogeneous networked environment of Unix computers.

AVAILABILITY The Cummings Group
PO Box 3574
Boulder, CO 80307
(303) 499-7572
email: Robert Biro `rpb@tcg.com`

Sterling Software
1121 San Antonio Road
Palo Alto, CA 94303-4311
(415) 964-9900
email: `nqs@ssmac.gsp.sterling.com`

COMMENTS Supports: IBM RS/6000, AIX/370, DEC Ultrix, SUN, HP, FPS Computing, Star Technologies, Solbourne, MIPS, SGI.

REFERENCES

SOFTWARE	Oasis (Object and Agent Specification and Implementation System)
DEVELOPER	The University of Michigan.
DESCRIPTION	Oasis is a distributed programming language. Computational processes are modeled as a distributed collection of autonomous cooperative agents. Oasis supports networked heterogeneous computers.
AVAILABILITY	Department of Electrical Engineering and Computer Science The University of Michigan Ann Arbor, MI 48109-2122 Contact: Fah-Chun Cheong (`fcc@eecs.umich.edu`) Source code/documentation/reference material available via anonymous ftp: `ftp.eecs.umich.edu` in directory: `/oasis`
COMMENTS	Has been demonstrated on IBM RS/6000s.
REFERENCES	[106].

SOFTWARE	OOF (Object-Oriented Fortran)
DEVELOPER	Mississippi State University.
DESCRIPTION	OOF is a portable, parallel language. OOF is based on Fortran with a minimum set of extensions. The extensions provide mechanisms to define and access objects and operators. OOF has been successfully implemented on a network of IBM RS/6000s, SGI, or SUN workstations, the Intel Delta, and the Intel iPSC/860.
AVAILABILITY	NSF Engineering Research Center for Computational Field Simulation Mississippi State University P.O. Box 6176 Mississippi State, MS 39762 (601) 325-2656 Contact: Donna Reese (`dreese@erc.msstate.edu`) Documentation/reference material available via anonymous ftp: `athena.erc.msstate.edu` in directory: `/pub`
COMMENTS	Does not presently support heterogeneous network configurations. However, the next version of OOF will incorporate the OOF kernel on top of PVM.
REFERENCES	[291], [323].

SOFTWARE	p4 (Portable Programs for Parallel Processors)
DEVELOPER	Argonne National Laboratory.
DESCRIPTION	p4 is a library of macros and subroutines for programming a variety of parallel machines in C and Fortran. It can be used to program networks of workstations, advanced parallel supercomputers, and single shared-memory multiprocessors. Support is provided for heterogeneous environments.
AVAILABILITY	Argonne National Laboratory Mathematics and Computer Science Division 9700 South Cass Avenue Argonne, IL 60439-4801 Project team email: `p4@mcs.anl.gov` Source code/documentation/reference material available via anonymous ftp: `info.mcs.anl.gov` in directory: `/pub/p4`
COMMENTS	None.
REFERENCES	[74], [86], [233].

SOFTWARE	ParaGraph
DEVELOPER	University of Illinois and Oak Ridge National Laboratory.
DESCRIPTION	ParaGraph is a graphical display system for visualizing the behavior and performance of parallel programs on message-passing multiprocessor architectures.
AVAILABILITY	Source code/documentation/reference material available via anonymous ftp: `ftp.ncsa.uiuc.edu` in directory: `/misc/ParaGraph`
COMMENTS	ParaGraph is a post processing system. Tracefiles are created by an application at run time and used for subsequent analysis. Tracefiles for ParaGraph are produced by PICL. ParaGraph is X-based and requires only the standard Xlib library.
REFERENCES	[183].

SOFTWARE	ParNet
DEVELOPER	Silicon Graphics Inc.
DESCRIPTION	ParNet is a nonsupported Fortran and C parallelization library which allows programs to execute over a network of SGI computer systems. ParNet uses PVM for the network layer.
AVAILABILITY	Silicon Graphics, Inc. Contact: Roberto Gompert, Applications Consulting Group Contact: Rosario Caltabiano, Eastern Technology Center Project team email: `parnet@boston.sgi.com`. Source code/documentation/reference material available by contacting the project team.
COMMENTS	Only supports SGI systems. Plan to support Linda in the future.
REFERENCES	[166].

SOFTWARE	PBS (Portable Batch System)
DEVELOPER	NASA Ames Research Center.
DESCRIPTION	PBS is the follow-on to NQS and is presently under development.
AVAILABILITY	Numerical Aerodynamic Simulation Systems Division NASA Ames Research Center Moffett Field, CA 94035-1000
COMMENTS	None.
REFERENCES	

SOFTWARE	PCN (Program Composition Notation)
DEVELOPER	Argonne National Laboratory.
DESCRIPTION	PCN is a system for developing and executing parallel programs. It comprises a high-level programming language, tools for developing and debugging programs in this language, and interfaces to Fortran and C that allow the reuse of existing code in multilingual parallel programs. PCN programs are portable across many different workstations, networks, and parallel computers. Parallelization over networked workstations is supported.
AVAILABILITY	Argonne National Laboratory Mathematics and Computer Science Division 9700 S. Cass Avenue Argonne, IL 60439 Source code/documentation/reference material available via anonymous ftp: `info.mcs.anl.gov` in directory: `/pub/pcn`
COMMENTS	At present, PCN only supports accidental heterogeneity (e.g. all systems share common data representations). A heterogeneous implementation is planned.
REFERENCES	[141], [142].

SOFTWARE	PICL (Portable Instrumented Communications Library)
DEVELOPER	Oak Ridge National Laboratory.
DESCRIPTION	PICL is a subroutine library used to develop parallel programs that are portable across distributed-memory architectures. Additionally, PICL provides execution tracing that can be used to monitor performance or aid with debugging.
AVAILABILITY	Engineering Physics and Mathematics Division Mathematical Sciences Section Oak Ridge National Laboratory P.O. Box 2009, Bldg. 9207-A Oak Ridge, TN 37831-8083 Available from: `netlib@ornl.gov`
COMMENTS	Does not presently support heterogeneous network configurations.
REFERENCES	[155], [156], [369].

SOFTWARE	Piranha Linda
DEVELOPER	Yale University and Scientific Computing Associates.
DESCRIPTION	Piranha-Linda is a research project which structures Linda modules as a "cloud of tasks" which computational "Piranhas" attack. The more Piranhas which attack, the faster the job is completed.
AVAILABILITY	Yale University Scientific Computing Associates, Inc. One Century Tower 265 Church Street New Haven, CT 06510-0710 (203) 777-7442 email: `software@sca.com`
COMMENTS	None.
REFERENCES	[67, 236, 243].

SOFTWARE	POET (Parallel Object-oriented Environment and Toolkit)
DEVELOPER	Sandia National Laboratories and Lawrence Berkeley Laboratory.
DESCRIPTION	POET is an object-oriented C++ toolkit for parallel and distributed computing. The toolkit consists entirely on creating and instantiating "pidgets".
AVAILABILITY	Distributed Computing Division Sandia National Laboratories Livermore, CA 94551 Contact: Rob Armstrong (`rob@snll-arpagw.llnl.gov`) Information and Computing Sciences Lawrence Berkeley Laboratory Berkeley, CA 94720 Contact: Jane Macfarlane (`jfmacfarlane@lbl.gov`)
COMMENTS	None.
REFERENCES	[26].

SOFTWARE	Polylith
DEVELOPER	University of Maryland.
DESCRIPTION	Polylith is a software interconnection system. It allows programmers to configure applications from mixed-language components and then execute them in heterogeneous environments. Polylith integrates the creation of stubs, source components and servers into a single system.
AVAILABILITY	Computer Science Department University of Maryland College Park, Maryland 20742 Project team email: `polylith@cs.umd.edu` Source code/documentation/reference material available via anonymous ftp: `flubber.cs.umd.edu` in directories: `/docs` and `/src`
COMMENTS	None.
REFERENCES	[89], [191], [282], [283], [284], [285], [286].

SOFTWARE	POSYBL (Programming System for Distributed Applications)
DEVELOPER	University of Crete in Greece.
DESCRIPTION	POSYBL is a public-domain version of Linda for Unix networks. POSYBL supports heterogeneous environments but has only been implemented on Sun and DEC workstations.
AVAILABILITY	Department of Computer Science University of Crete Greece Contact: Ioannis Schoinas (`sxoinas@csd.uch.gr`) or (`sxoinas@csi.forth.gr`) or (`sxoinas@ariadne.bitnet`) Source code/documentation/reference material available via anonymous ftp: `pythia.csi.forth.gr` in directory: `/posybl`
COMMENTS	POSYBL does not include many of the features and tools of commercial Linda products.
REFERENCES	[308], [331].

SOFTWARE	PowerTools
DEVELOPER	VXM Technologies Inc.
DESCRIPTION	PowerTools consists of a set of interfaces (called PAX-2) which link PVM with Sun's ToolTalk message passing service. ToolTalk is a higher level network message passing service and enables an isolated application to seamlessly integrate its data and services with other network data/applications. Using PAX-2 programmers can create network parallel subroutine objects. Subroutine objects can be accessed and utilized over and over again by a wide variety of applications. PAX-2 also has the ability to replicate PVM-based processes in order to create a given number of parallel modules.
AVAILABILITY	VXM Technologies Inc. 30 Gardner Road Suite 3G Brookline, MA 02146 (617) 730-8490 Supports: SGI, IBM, Sun, DEC, and HP workstations. Various configurations and pricing are available.
COMMENTS	ToolTalk is bundled with Solaris v1.0. ToolTalk is not bundled with Solaris 2.0. ToolTalk may be an optional product for other platforms.
REFERENCES	[154].

SOFTWARE	PVM (Parallel Virtual Machine)
DEVELOPER	Oak Ridge National Laboratory and Emory University.
DESCRIPTION	PVM is a software system that permits the utilization of a heterogeneous network of serial and parallel computers as a unified, general, and flexible concurrent computational resource. PVM is a toolkit for writing portable parallel programs using a message passing model.
AVAILABILITY	Oak Ridge National Laboratory Mathematical Sciences Division P.O. Box 2008, Bldg. 6012 Oak Ridge, TN 37831-6367 Project team email: `pvm@msr.epm.ornl.gov` Available from: `netlib@ornl.gov`
COMMENTS	None.
REFERENCES	[5], [41], [43], [44], [45], [157], [158], [324], [338], [339], [340], [341], [342].

SOFTWARE	PVM++
DEVELOPER	University of Tennessee Knoxville
DESCRIPTION	PVM++ is a message passing object oriented toolkit for PVM. PVM++ is presently under development.
AVAILABILITY	University of Tennessee Knoxville Contact: Roldan Pozo (`pozo@cs.utk.edu`)
COMMENTS	None.
REFERENCES	[281].

SOFTWARE QBATCH (Queued Batch processing system)

DEVELOPER Vita Services Ltd.

DESCRIPTION QBATCH was developed to address the limitations of the Unix "at" and "batch" commands. QBATCH allows several job queues containing jobs to be processed in a specific order. QBATCH provides a portable, consistent method of achieving this goal for most Unix systems.

AVAILABILITY Alan D. Saunders
Vita Services Ltd.
1341 Rochdale Road
Blackley, Manchester, M9 2FG, United Kingdom
email: `alan@tharr.uucp`

Source code/documentation/reference material available via
anonymous ftp: `ftp.uu.net`
in directory: `/usenet/comp.sources.misc/Volume25`

COMMENTS Little or no provision is provided for administering a set of queues running on a network of workstations.

REFERENCES [303].

SOFTWARE	QuaHog
DEVELOPER	Brown University.
DESCRIPTION	QuaHog is a distributed processing system which attempts to dispatch jobs to idle CPUs which are participating in the CPU pool.
AVAILABILITY	Computer Science Department Brown University Providence, RI 02912 Contact: John Bazik (`jsb@cs.brown.edu`) Documentation/reference material available via anonymous ftp: `wilma.cs.brown.edu`.
COMMENTS	None.
REFERENCES	[36], [37].

SOFTWARE	Schooner
DEVELOPER	The University of Arizona.
DESCRIPTION	Schooner is an interconnection system that includes an intermediate data representation, a simple specification language, and a heterogeneous remote procedure call (RPC) facility. Schooner uses a sequential execution model where control passes from procedure to procedure (individual procedures may be on different systems). Schooner is targeted at applications such as visualization/computation (e.g. workstation/supercomputer).
AVAILABILITY	The University of Arizona Department of Computer Science Tucson, Arizona 85721 Contact: Patrick T. Homer (`patrick@cs.arizona.edu`)
COMMENTS	C and Fortran bindings are included.
REFERENCES	[192], [193].

SOFTWARE	Spawn
DEVELOPER	Xerox PARC.
DESCRIPTION	Spawn is a two year old experimental system that explores concepts in “computational ecology”. Application processes in Spawn behave like organisms that spawn and oversee off-spring. Spawn applications have been developed for statistics have achieved near linear speedups in execution time for prallel network applications.
AVAILABILITY	Xerox PARC Dynamics of Computation Group Contact: Bernardo Huberman
COMMENTS	Runs of homogeneous networks of Sun workstations and is built on Sun’s ONC RPC.
REFERENCES	[208].

SOFTWARE	SPL (Spreedsheet Programming Language)
DEVELOPER	University of Notre Dame.
DESCRIPTION	SPL is a spreadsheet based programming language targeted for workstation clusters. Potential application areas are linear systems, simulations, asynchronous algorithms, and database applications.
AVAILABILITY	University of Notre Dame Distributed Computing Research Laboratory 326 Cushing Notre Dame, IN 46556 Contact: Alan Yoder (agy@cse.nd.edu) Contact: David L. Cohn (dlc@cse.nd.edu)
COMMENTS	Still in design phase - 1st implementation to begin in 1st quarter of 1993.
REFERENCES	[375].

SOFTWARE	SR (Synchronizing Resources)
DEVELOPER	University of Arizona.
DESCRIPTION	SR is a language for writing distributed and parallel programs. The language supports a wide variety of (reliable) message passing constructs, including shared variables (for processes on the same node), asynchronous message passing, rendezvous, remote procedure call, multicast.
AVAILABILITY	University of Arizona Department of Computer Science Gould-Simpson Building Tucson, Arizona 85721 Project team email: `sr-project@cs.arizona.edu` Source code/documentation/reference material available via anonymous ftp: `cs.arizona.edu` in directory: `/sr`
COMMENTS	Limited to homogeneous environments.
REFERENCES	[13], [12], [271], [307].

SOFTWARE	Task Broker
DEVELOPER	Hewlett-Packard.
DESCRIPTION	Task Broker configures a network of workstations so that each machine will "bid" for available jobs, with a job going to the highest bidder at a given time.
AVAILABILITY	Hewlett-Packard
COMMENTS	Available only on HP, Apollo, and Sun systems.
REFERENCES	

SOFTWARE TCGMSG (Theoretical Chemistry Group Message passing system)

DEVELOPER Argonne National Laboratory and
Battelle Pacific Northwest Laboratory.

DESCRIPTION TCGMSG is a toolkit for writing portable parallel programs using a message passing model. TCGMSG supports a variety of common Unix workstations, mini-supers, supercomputers, and heterogeneous networked workstations.

AVAILABILITY Argonne National Laboratory
Theoretical Chemistry Group
9700 S. Cass Avenue
Argonne, IL 60439

Battelle Pacific Northwest Laboratory
Mail Stop K1-90
P.O. Box 999
Richland, WA 99352

Contact: Robert J. Harrison (509) 375-2037
(`rj_harrison@pnl.gov`)

Source code/documentation/reference material available via
anonymous ftp: `ftp.tcg.anl.gov`
in directory: `/pub/tcgmsg`

COMMENTS Limited documentation.

REFERENCES

SOFTWARE	TOPSYS (TOols for Parallel SYStems)
DEVELOPER	Munich University of Technology.
DESCRIPTION	TOPSYS is an integrated tool environment for developing applications for heterogeneous and distributed computing facilities. TOPSYS is implemented using MMK (multiprocessor multitasking kernel). TOPSYS includes tools for parallel programming, monitoring, debugging, performance analysis, animation, and dynamic load balancing.
AVAILABILITY	Computer Science Department Munich University of Technology Postfach 20 24 20, Arcisstr. 21 D-W-8000 Munchen 2 Germany Contact: Thomas Bemmerl `(bemmerl@informatik.tu-muenchen.de)` Documentation/reference material available via anonymous ftp: `topsys.informatik.tu-muenchen.de` in directory: `/topsys/public` Source/binary license is available by executing a licensing agreement and making a donation of Deutsch Mark 1.000.
COMMENTS	TOPSYS includes the following tools and software packages: MMK (multi-threaded, message passing programming library), ALLOCCUBE (network-based resource management and accounting package), DETOP (a parallel debugger), PATOP (a performance analysis tool), and VISTOP (a graphical visualization and animation tool). Supports Intel iPSC/2, Intel iPSC/860, and TCP/IP-based networks of SUN SPARCstations.
REFERENCES	[48], [50], [51], [52], [53].

SOFTWARE	Tuplex
DEVELOPER	Torque Systems Inc.
DESCRIPTION	Tuplex is a commercial implementation of Linda.
AVAILABILITY	Torque Systems Inc. 700 High St. Palo Alto, CA 94301 (415) 321-1200 Supports: Sun, NeXT, Mac, i860 computer servers.
COMMENTS	None.
REFERENCES	[73].

SOFTWARE	UniJES (Unix Job Entry System)
DEVELOPER	IBM.
DESCRIPTION	UniJES is IBM's commercialized implementation of Condor. IBM has made numerous enhancements; particularly in the area of administration utilities. IBM is planning to support UniJES implementations on other workstation platforms in the future.
AVAILABILITY	IBM
COMMENTS	None.
REFERENCES	

SOFTWARE	Upshot
DEVELOPER	Argonne National Laboratory.
DESCRIPTION	Upshot is an X-based graphics utility for viewing log files produced by parallel programs. Upshot provides the capability to analyze the performance characteristics for parallel applications.
AVAILABILITY	Argonne National Laboratory Mathematics and Computer Science Division 9700 S. Cass Avenue Argonne, IL 60439 Source code/documentation/reference material available via anonymous ftp: `info.mcs.anl.gov` in directory: `/pub/upshot`
COMMENTS	Upshot requires that applications generate log files in a specific format. Graphics output from Upshot is similar to that generated by ParaGraph.
REFERENCES	[189].

SOFTWARE	Utopia
DEVELOPER	University of Toronto.
DESCRIPTION	Utopia is the University implementation of LSF. See LSF description.
AVAILABILITY	Computer Systems Research Institute University of Toronto Toronto, Ontario, Canada
COMMENTS	Utopia is marketed under the name LSF from Platform Computing Corporation.
REFERENCES	See LSF description.

SOFTWARE	Xab (X-window Analysis and deBugging)
DEVELOPER	Carnegie Mellon University.
DESCRIPTION	Xab is a tool for run time monitoring of PVM programs. Using Xab, PVM programs can easily be instrumented and monitored.
AVAILABILITY	Carnegie Mellon University School of Computer Science 5000 Forbes Avenue Pittsburgh, PA 15213-3890 (412) 268-5295 Contact: Adam L. Beguelin (`adamb@cs.cmu.edu`) Source code/documentation/reference material available by contacting Adam Beguelin.
COMMENTS	Xab uses PVM to monitor PVM programs.
REFERENCES	[47].

SOFTWARE	xHPF
DEVELOPER	Applied Parallel Research and Kuck & Associates, Inc.
DESCRIPTION	xHPF is a preprocessor that generates Fortran 77 code which makes calls to a parallel runtime library. The runtime library has interfaces for many of the popular message passing libraries such as PVM, Express, Linda, etc. xHPF accepts the subset of Fortran 90 and HPF directives recommended by the HPFF (High-Performance Fortran Forum). The product supports current MIMD parallel architectures as well as clustered workstations.
AVAILABILITY	Applied Parallel Research 550 Main Street Suite I Placerville, CA 95667 (916) 621-1600 Kuck & Associates, Inc. 1906 Fox Drive Champaign, IL 61820 (217) 356-2288
COMMENTS	xHPF includes a parallel simulator and a parallel profiler.
REFERENCES	