A Survey on Parallel Architecture and Parallel Programming Languages and Tools

C.Namrata Mahender
Dept. of CS and IT,
Dr.Babasaheb Ambedkar Marathwada University,
Aurangabad,India
e-mail: nam.mah@gmail.com

***Abstract*—** In this paper, we have presented a brief review on the evolution of parallel computing to multi-core architecture. The survey briefs more than 45 languages, libraries and tools used till date to increase performance through parallel programming. We have given emphasis more on the architecture of parallel system in the survey.

Keywords- *Parallel Computing, Parallelization, Classification, shared memory, distributed memory architecture*

*****

I. INTRODUCTION

In 1945 Von Neumann suggested the stored-program model of computing, the architecture stated that a program is a sequence of instructions stored sequentially in the computer's memory and are executed one after the other in a linear, single-threaded fashion. As time went on advancement in mainframe technology expanded the idea of proposed by Von Neumann, the 1960 saw the advent of time sharing OS, run on large mainframe computer. Then the standalone PC became the buzz but due to large computation required and cost of Pc, need of concurrency came into picture, thus parallel computing became the need of time.[1,2]

Parallel computing came in the late 1950's ,with advancements surfacing in the form of supercomputers throughout the 60's and 70's. These were shared memory multiprocessors ,with multiple processors working side-by-side on shared data. In the mid 1980's a new kind of parallel computing was launched when the Caltech Concurrent computation project built a supercomputer for scientific applications from 64 Intel 8086/8087 processors. This system showed that extreme performance could be achieved with mass market, off the shelf microprocessors. These massively parallel processor came to dominate the top end of computing, with ASCI Red supercomputer in 1997 breaking the barrier of one trillion floating point operations per seconds. Science then MPPS has continued to grow in size and power.

Staring in the late 80's, cluster came to compete and eventually displace MPPs for many applications. A cluster is a type of parallel computer built from large numbers of off-the-shelf computers connected by an off-the-shelf network. Today, clusters are the workhorse of scientific computing and are the dominant architecture in the data centers that power the modern information age.

Today, parallel computing is becoming mainstream based on multicore processors most desktop and laptop systems now ship with multiple cores. The reason is that increasing performance through parallel processing can be far more energy-efficient than increasing microprocessor clock frequency.[2,3]

A. Era of computing

The most prominent two eras of computing are: sequential and parallel era. In the past decade parallel machines have become significant competitors to vector machines in the quest of high performance computing. A century wide view of development of computing eras is shown in figure 1. The computing era starts with development in hardware architecture, followed by system software particularly in the era of compilers and operating system, applications and reaching its saturation point with its growth in problem solving environment s. every element of computing undergoes three phases: R&D, Commercialization and commodity.[4]

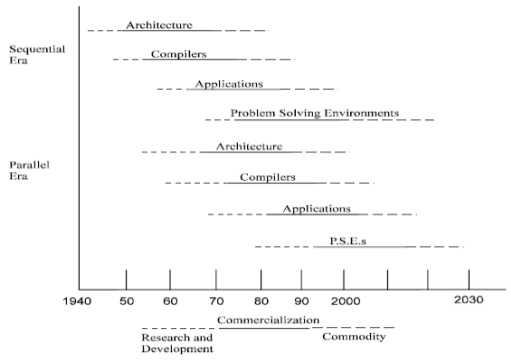

Figure 1. Two Eras of Computing

II. PARALLEL COMPUTING CLASSIFICATION

Parallel computing can be defined based on architecture or the way programs are implemented, or the way the algorithm or program is decomposed.

A. *Parallel computer based on Architecture*

1) Flynn's taxonomy

Flynn's taxonomy distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of Instruction and Data. Each of these dimensions can have only one of two possible states: Single or Multiple.The system has four classes SISD,SIMD,MISD,MIMD

a) Single Instruction, Single Data (SISD):

- A serial (non-parallel) computer
- Single Instruction: Only one instruction stream is being acted on by the CPU during any one clock cycle
- Single Data: Only one data stream is being used as input during any one clock cycle
- Deterministic execution
- This is the oldest and even today, the most common type of computer
- Examples: older generation mainframes, minicomputers and workstations; most modern day PCs.

b) Single Instruction, Multiple Data (SIMD):

- A type of parallel computer
- Single Instruction: All processing units execute the same instruction at any given clock cycle
- Multiple Data: Each processing unit can operate on a different data element
- Best suited for specialized problems characterized by a high degree of regularity, such as graphics/image processing.
- Synchronous (lockstep) and deterministic execution
- Two varieties: Processor Arrays and Vector Pipelines

c) Multiple Instruction, *Single Data (MISD):*

- A type of parallel computer
- Multiple Instruction: Each processing unit operates on the data independently via separate instruction streams.
- Single Data: A single data stream is fed into multiple processing units.
- Few actual examples of this class of parallel computer have ever existed. One is the experimental Carnegie-Mellon C.mmp computer (1971).
- Some conceivable uses might be: multiple frequency filters operating on a single signal stream, multiple cryptography algorithms attempting to crack a single coded message.

d) Multiple Instruction, Multiple Data (MIMD):

- A type of parallel computer
- Multiple Instruction: Every processor may be executing a different instruction stream
- Multiple Data: Every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic
- Currently, the most common type of parallel computer - most modern supercomputers fall into this category.
- Examples: most current supercomputers, networked parallel computer clusters and "grids", multi-processor SMP computers, multi-core PCs. [4]

2) Feng's Classification

Tse-yun Feng suggested the use of degree of parallelism to classify various computer architectures. Serial Versus Parallel Processing The maximum number of binary digits that can be processed within a unit time by a computer system is called the maximum parallelism degree P.A bit slice is a string of bits one from each of the words at the same vertical position. There are 4 types of methods under above classification

- Word Serial and Bit Serial (WSBS)
- Word Parallel and Bit Serial (WPBS)
- Word Serial and Bit Parallel(WSBP)
- Word Parallel and Bit Parallel (WPBP)

WSBS has been called bit parallel processing because one bit is processed at a time.WPBS has been called bit slice processing because m-bit slice is processes at a time.WSBP is found in most existing computers and has been called as Word Slice processing because one word of n bit processed at a time. WPBP is known as fully parallel processing in which an array on n x m bits is processes at one time.[4]

Table 1: Feng's Computer Classification

Mode	Computer Model	Degree of Parallelism
WSPS N=1,M=1	The 'MINIMA'	(1,1)
WPBS N=1,M>1	STARAN, MPP, DAP	(1,256), (1,16384), (1,4096)
WSBP n>1,m=1(Word Slice Processing)	IBM 370/168 UP CDC6600 Burrough 7700 VAX 11/780	(64,1) (60,1) (48,1) (16/32,1)
WPBP n>1,m>1(fully parallel processing)	ILLiav IV	(64,64)

3) Handler's Classification

Wolfgang Handler has proposed a classification scheme for identifying the parallelism degree and pipelining degree built into the hardware structure of a computer system. He considers at three subsystem levels:

- Processor Control Unit (PCU)
- Arithmetic Logic Unit (ALU)
- Bit Level Circuit (BLC)

Each PCU corresponds to one processor or one CPU. The ALU is equivalent to Processor Element (PE). The BLC corresponds to combinational logic circuitry needed to perform 1 bit operations in the ALU.

A computer system C can be characterized by a triple containing six independent entities

T(C) = <K x K', D x D', W x W' >

Where K = the number of processors (PCUs) within the computer,

D = the number of ALUs under the control of one CPU,

W = the word length of an ALU or of an PE,
W' = The number of pipeline stages in all ALUs or in a PE,
D' = the number of ALUs that can be pipelined,
K' = the number of PCUs that can be pipelined.

B. Parallel computer based on memory architecture

Generally system for parallel computer can be classified by using memory in two types shared memory and distributed memory

1) Shared Memory Architecture

Shared memory machines can be divided into two main classes based upon memory access times: UMA and NUMA.

a) Uniform Memory Access (UMA):

- Most commonly represented today by Symmetric Multiprocessor (SMP) machines
- Identical processors
- Equal access and access times to memory
- Sometimes called CC-UMA - Cache Coherent UMA.Cache coherent means if one processor updates a location in shared memory, all the other processors know about the update. Cache coherency is accomplished at the hardware level.

b) Non-Uniform Memory Access (NUMA):

- Often made by physically linking two or more SMPs
- One SMP can directly access memory of another SMP
- Not all processors have equal access time to all memories
- Memory access across link is slower
- If cache coherency is maintained, then may also be called CC-NUMA - Cache Coherent NUMA

2) Distributed memory architecture

Distributed memory systems require a communication network to connect inter-processor memory.

- Processors have their own local memory. Memory addresses in one processor do not map to another processor, so there is no concept of global address space across all processors.
- Because each processor has its own local memory, it operates independently. Changes it makes to its local memory have no effect on the memory of other processors. Hence, the concept of cache coherency does not apply.
- When a processor needs access to data in another processor, it is usually the task of the programmer to explicitly define how and when data is communicated. Synchronization between tasks is likewise the programmer's responsibility.
- The network "fabric" used for data transfer varies widely, though it can can be as simple as Ethernet.

3) Hybrid -Distributed memory architecture

The largest and fastest computers in the world today employ both shared and distributed memory architectures.

- The shared memory component can be a cache coherent SMP machine and/or graphics processing units (GPU).
- The distributed memory component is the networking of multiple SMP/GPU machines, which know only about their own memory - not the memory on another machine. Therefore, network communications are required to move data from one SMP/GPU to another.
- Parallel Processing based on data Parallelism
- There are many ways program is parallelized, majorly it is done in three ways instruction, task and data . In data parallel processing, full jobs are assigned for processing . it is efficient with coarse grained tasks and quasi scheduling , the figure 2 shows the different methods of data processing .[3,5]

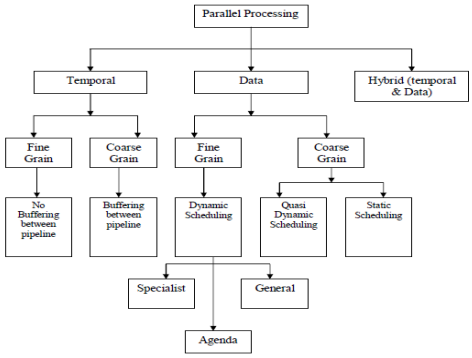

Figure 2. Shows various Methods of parallel processing

III. PARALLEL PROGRAMMING LANGUAGE AND TOOLS

Numerous programming frameworks have been developed so far to support parallel execution in parallel computing system. The frameworks function at various levels of abstraction, represents different formal models for parallelism, exploit dedicated or general-purpose languages and vary from local area network based systems to geographically distributed systems. Each framework is specialized for a particular type of applications. Here we have given brief information about more than 45 languages and tools used to perform parallel programming.

1) Parallel Programming

Here we are just briefing the majorly used languages for parallel programming with for which they extension, support

used for executing on platform and does has GUI interface. Table 2 gives the brief about parallel languages.[6-62]

Table 1: Types of Parallel Programming [6-62]

Sr. No	Language	Language supported	GUI Support	Type
1	Adaptor(Automatic Data Parallelism TranslatOR)	CM,Fortran,Subset of Fortran 90	Athena Widget,X-Windows system	Language
2	Caper(Concurrent Application Programming Environment)	C,C++,Concurrent C	X11 R3-R5	Language + Debugger+ Performance Tool
3	CC++(Compositional C++)	CC++	None	Language
4	Charm	C,C++	X Motif	Language+ Performance Tool
5	Code2.0	Code2.0,C	X11R4 or X11R5	Language+ Performance Tool
6	COOL(Concurrent Object Oriented Language)	COOL	None	Language+ Performance Tool
7	Dino	C	None	Language
8	Fortran90	Fortran90	None	Language
9	Fortran D	Fortran77	X11R4	Language
10	Fortran M	FortranM	None	Language
11	Grids	Fortran77	X11R5	Language
12	HPF(High Performance Fortran)	HPF	None	Language
13	Hypertool	C	None	Language+ Performance Tool
14	Jade	C	None	Language
15	Linda	C, Fortran77	X11R4	Language+ Debugger + Performance Tool
16	MeldC	MeldC	None	Language+ Debugger
17	Mentat	MPL-an Extended C++	None	Language
18	Modula-2*	Modula-2*	None	Language
19	O-OFortran(Object oriented Fortran)	Fortran77,C++	None	Language
20	P-D Linda (Prolog-DLinda)	SICStus Prolog 0.7 and 2.1	None	Language
21	P-Languages	PC,P Fortran	None	Language
22	Parallelax	Parallelax	Macintosh	Language+ Performance Tool
23	Parallelaxis	Parallelaxis	X11R5	Language+ Debugger+ Performance Tool
24	PC++(Parallel C++)	pC++	None	Language
25	PCN(Program Compostion Notation)	PCN,C, Fortran	X11R5	Language+ Debugger+ Performance Tool
26	PCP/PFP(Parallel C/Fortran Preprocessor)	C, Fortran77	None	Language
27	PDDP(The Parallel Data Distribution Preprocessor)	Fortran77	None	Language
28	SR(Synchronizing Resources)	SR	X Windows	Language
29	Strand 88	Strand88, Fortran,C	X11	Language+ Debugger+ Performance Tool
30	TOPSys(Tools for Parallel Systems)	Fortran77,C	X Windows	Language+ Debugger+ Performance Tool
31	Vienna Fortran	Vienna Fortran, Fortran77	X11R5,OSF/Motif	Language
32	Visage(VIsual Attributed Graph Enviornment)	C	GL	Language+ Performance Tool
33	X3H5	Fortran,C	None	Language

2) Parallel Libraries and tools

The fifteen tools described in this part try to achieve portability by providing libraries. Most of them support parallelism within an application, Application-oriented high-level abstractions are provided in Canopy, whereas the others support programming languages such as C and Fortran with parallel extensions. A few support programming in both shared-memory paradigm and message-passing paradigm. Several of them extend the support for distributed memory machines to a network of computers. Table 3 shows the different libraries used till date.[62-89]

Table 2: Types of Parallel Programming Libraries [62-89]

Sr.No	Libraries	Language Supported	GUI Support
1	APPL	Fortran,C	None
2	Canopy	Fortran,C	None
3	CM(Communication Manager)	C	None
4	CPS(Cooperative Preocesses Software)	Fortran,C	None
5	Express	Fortran 77,C	X-indows, Sunview,postscript
6	GenMP(GENeric MultiProcessor)	Fortran 77	None

7	LMPS(The Livermore Message Passing System)	Fortran 77,C	None
8	Mtask(MultiTasking Package)	C	None
9	P4	Fortran 77,C	None
10	Parmacs	Fortran77,C, PARMACS 6.0	X11R3
11	Parti(Parallel Automated Runtime Toolkit at ICASE)	Fortran,C	None
12	PICL(A portable Instrumented Communication Library for Intel)	Fortran 77,C	None
13	PVM(Parallel Virtual Machine)	Fortran 77,C	None
14	SPPL(The Stuttgart Parallel Processing Library)	C	None
15	TCGMSG (Theoretical Chemistry Group Message Passing Toolkit)	Fortran,C	None
16	OpenMp	Fortran,C/C++	X Windows
17	Pthreads	C/C++	Any

IV. CONCLUSION

In this paper we have discussed all major parallel architecture and its characteristics. We have discussed all major aspects of parallel computing, staring from its evolution to the present multi-core era. We covered almost all languages and libraries and tools which are used to implement parallel programs, improve performance and computing speed respectively.

REFERENCES

[1] Shameem Akhter,J Roberts,"Multi-Core Programming Increaseing Performance through Software Multi-threading",Intel Press,2006.

[2] Herb Sutter,March,, "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software", Dr. Dobb's Journa,2005.

[3] Hesham El-Rewini M. Abd-El-Barr, "Advanced Computer Architecture and Parallel Processing",John Wiley & Sons, Inc,2005.

[4] Kai Hwang , Faye A. Briggs , "Computer Architecture and Parallel Processing" , McGraw-Hill Education,1986

[5] V. Rajaraman and C.Siva Ram Murthy, "Parallel Computer Architecture and programming",PHI,2000

[6] 1. Glenn Kubena, Kenneth Liao, and Larry Roberts, "White Paper on Massively Parallel,Programming Languages", IBM, Dec. 3,1992

[7] Thomas Sterling, Paul Messina, Marina Chen, Frederica Darema, Geoffrey Fox,Michael Heath, Ken Kennedy, Robert Knighten, Reagon Moore, Sanjay Ranka, JoelSaltz, Lew Tucker, and paul Woodard, ''System Software and Tools for High Performance Computing Environments,'' A Report on the Findings of the JPL PasadenaWorkshop, April, 1992.

[8] ISO/IDE, ''Information technology -- Programming languages -- Fortran,'' InternationalStandard, Reference number ISO/IEC 1539 : 1991 (E),.

[9] High Performance Fortran Forum, High Performance Fortran Language Specification,Version 1.0 Draft, Jan. 25, 1993.

[10] ANSI Technical Committee X3H5, "Parallel Processing Model for High Level ProgrammingLanguage"s, June 1992.

[11] ANSI Technical Committee X3H5, "Parallel Fortran Standard", 1992.

[12] ANSI Technical Committee X3H5, "Fortran Binding -- Data Model Section", 1992.

[13] T. Brandes, "Efficient Data Parallel Program without Explicit Message Passing for Distributed Memory Multiprocessors" , GMD Technical Report, TR92-4, 1992

[14] L. Ridgway Scott, ''Pfortran: a parallel dialect of Fortran,'' Fortran Forum 11, vol.No. 3, pp. 20-31,10. Harry Jordan, Jan. 1987 ''The Force,'' ECE Tech. Report 87-1-1,.

[15] Harry F. Jordan, Muhammad S. Benten, Norbert S. Arenstorf, and Aruna V.Ramanan, March 1989,Force User's Manual.

[16] Donna Reese, ''Object-Oriented Fortran for Portable, Parallel Programs,'' The ThirdIEEE Symposium on Parallel and Distributed Processing, pp. 608-615, December 1991

[17] B.Chapman, P.Mehrotra, and H.P.Zima,,''Vienna FORTRAN - A Fortran Language Extension for Distributed Memory Multiprocessors,'' in Compilers and Runtime Software for Scalable Multiprocessors, ed. J. Saltz and P. Mehrotra, Elsevier, Amsterdam, , 1991

[18] P.Brezany, B.Chapman, and H.Zima, , "Automatic Parallelization for GENESIS",Austrian Center for Parallel Computation, Technical Report ACPC/TR 92--16, November 1992

[19] B.M.Chapman, P.Mehrotra, and H.Zima, ''Programming in Vienna Fortran",ScientificProgramming, vol. Vol.1, No.1. 1992.

[20] Seema Hiranandani, Ken Kennedy, and Chau-Wen Tseng, ''Compiling Fortran D for MIMD Distributed-Memory Machines,'' Communications of the ACM, vol. 35(8),pp. 66-80, August 1992.

[21] Geoffrey Fox, Seema Hiranandani, Ken Kennedy, Charles Koelbel, Ulrich Kremer, Chau-Wen Tseng, and Min-You Wu, ''Fortran D Language Specification,'' Dept. of Computer Science Technical Report TR90-141, Rice University, December 1990 .

[22] I. Foster and K. M. Chandy, ‘‘Fortran M: A Language for Modular Parallel Programming,’’Preprint MCS-P327-0992, Argonne National Laboratory, Argonne, Ill, 1992.

[23] P. Newton and J. C. Browne, ‘‘The CODE 2.0 Parallel Programming Language,’’,Proc. ACM International Conf. on Supercomputing. July 1992.

[24] A. Reuter, U. Geuder, M. Haerdtner, B. Woerner, and R. Zink, ‘‘The GRIDS Project,’’,Technical Report, University of Stuttgart, 1992.

[25] Eugene Brooks, Brent Gorda, and Karen Warren, ‘‘The Parallel C Preprocessor,’’ in Scientific Programming, vol. vol 1, Number 1, John Wiley & Sons, Inc., New York.

[26] Brent Gorda, Karen Warren, and Eugene D. Brooks III ‘‘Programming in PCP,’’Technical Report, Lawrence Livermore National Laboratory, UCRL-MA-107029, , April 1991.

[27] Brent Gorda, ‘‘Data Parallel Programming,’’ Spring Proceedings, 1992 Cray User Group.

[28] Eugene D. Brooks , ‘‘The 1992 MPCI Yearly Report: Harnessing the Killer Micros,’’ Lawrence Livermore National Laboratory, CA94550,pp. 1-6, August 1992.

[29] M. C. Rinard, D. S. Scales, and M. S. Lam, ‘‘Heterogeneous Parallel Programming in Jade,’’ Proceedings of Supercomputing ’92, pp. 245-256, Nov. 1992.

[30] M. S. Lam and M. C. Rinard, ‘‘Coarse-Grain Parallel Programming in Jade,’’ Proceedingsof the Third ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 94-105,. April, 1991.

[31] M. C. Rinard and M. S. Lam, ‘‘Semantic Foundations of Jade,’’ Record of the Nineteenth Annual ACM Symposium on Principles of Programming Languages, pp.105-118, Jan., 1992.

[32] L.V. Kale, ‘‘The Chare Kernel Parallel Programming Language and System,’’ in:Proc. of the International Conference on Parallel Processing, Aug. 1990.

[33] L.V. Kale, ‘‘A Tutorial Introduction to Charm,’’ Parallel Programming Laboratory Internal report, 1992.

[34] Matthew Rosing, Robert B. Schnabel, and Robert P. Weaver, ‘‘The DINO Parallel Programming Language,’’ Tech. Report CU-CS-457-90, CS Dept. Univ. of Colorado at Boulder, April 1990.

[35] Thomas M. Derby, Elizabeth Eskow, Richard Neves, Matthew Rosing, Robert B.Schnabel, and Robert P. Weaver, ‘‘DINO 1.0 User’s Manual,’’ Tech Report CUCS- 501-90, CS Dept. Univ. of Colorado at Boulder, April 1990.

[36] Min-You Wu and Daniel D. Gajski, ‘‘Hypertool: A Programming Aid for Message- Passing Systems,’’ IEEE Trans. on Parallel and Distributed Systems, vol. 1 No. 3,pp. 330-343, July 1990.

[37] Francois Bodin, Peter Beckman, Dennis Gannon, Srinivas Narayana, and Shelby Yang, ‘‘Distributed pC++: Basic Ideas for an Object Parallel Language,’’ Proceedings of Supercomputing 91 (Albuquerque, Nov. 1991), IEEE Computer Sciety and ACM SIGARCH, pp. 273-282,1991.

[38] A. S. Grimshaw, ‘‘Easy to Use Object-Oriented Parallel Programming with Mentat,’’ IEEE Computer, May 1993.

[39] A. S. Grimshaw, W. Timothy Strayer, and Padmini Narayan, ‘‘The Good News About Dynamic Object-Oriented Parallel Processing,’’ University of Virginia, Computer Science Report TR-92-41,, 1992.

[40] Rohit Chandra, Anoop Gupta, and John L. Hennesy, ‘‘Integrating Concurrency and Data Abstraction in the COOL Parallel Programming Language,’’ Technical Report CSL-TR-92-511, Computer Systems Lab, Stanford University,, February 1992.

[41] Rohit Chandra, Anoop Gupta, and John L. Hennesy, ‘‘Data Locality and Load Balancing in COOL,’’ To Appear in the Symposium on Principles and Practices of Parallel Programming (PPoPP), May 1993.

[42] Steven S. Popovich, Shyhtsun F. Wu, and Gail E. Kaiser, ‘‘An Object-Based Approach to Implementing Distributed Concurrency Control,’’ 11th International Conference on Distributed Computing Systems, Arlington TX, pp. 65-72, May, 1991.

[43] Wenwey Hseush, James C. Lee, and Gail E. Kaiser, ‘‘MeldC Threads: Supporting Large-Scale Dynamic Parallelism,’’ Technical Report CUCS-010-92, Columbia University,March, 1992.

[44] Thomas Braunl, ‘‘Structured SIMD Programming in Parallaxis,’’ Structured Programming Structured Programming,Journal, vol. 10/3, 1998.

[45] Thomas Braunl, ‘‘ Parallel Programming,’’ in An Introduction Textbook, Prentice-Hall, Summer 1993.

[46] Walter F. Tichy and Christian G. Herter, ‘‘Modula-2*: An Extension of Modula-2 for Highly Parallel, Portable Programs,’’ Technical Report KA-INF, No. 4/90, Jan.,1990.

[47] Michael Philippsen and Walter F. Tichy, ‘‘Modula-2* and its Compilation,’’ First International Conference of the Austrian Center for Parallel Computation, pp. 169-183, Springer Verlag, September 30 - October 2, 1991.

[48] Ian Foster and Stephen Taylor, “Strand New Concepts in Parallel Programming”, Prentice Hall, 1989.

[49] D. Cann, ‘‘Retire Fortran? A Debate Rekindled,’’ Communications of the ACM, vol.Vol 35, Number 8, August, 1992.

[50] 47. J. T. Feo, D.C. Cann, and R. R. Oldehoeft, ‘‘A Report on the SISAL Language Project,’’Journal of Parallel and Distributed Computing, vol. Vol 12 No. 10, pp.349-366, December 1990.

[51] J. R. McGraw and et. al., ‘‘Sisal: Streams and iterations in a single-assignment language,Language Reference Manual, Version 1.2,’’ Lawrence Livermore National Laboratory Manual M-146 (Rev. 1), March 1985.

[52] Lawrence Livermore National Laboratory, ‘‘Proceedings of the Second Annual Sisal Users Conference,’’ Lawrence

Livermore National Laboratory Technical Report,UCRL JC112593, October 1992.

[53] I. Foster, R. Olson, and S. Tuecke, ‘‘Productive parallel programming: The PCN approach,’’ Scientific Programming, vol. 1(1), pp. 51-66, 1992.

[54] K. M. Chandy and S. Taylor, “An Introduction to Parallel Programming”, Jones and Bartlett, 1991.

[55] Alan H. Karp, ‘‘Some Experience with Network Linda,’’ The International Journal for High Speed Computing (to appear), 1993.

[56] Doug Kimeleman and Dror Zernik, ‘‘On-the-Fly Topological Sorting for Interactive Debugging and Live Visualization of Parallel Programs,’’ To appear in the Third ACMONR Workshop on Parallel and Distributed Debugging, may, 1993.

[57] Leonid Gluhovsky and Dror Zernik, ‘‘ILGA - A Little Language For Processing Graphs,’’ Technical report No. 872. Electrical Engineering Faculty, Technion.

[58] Gregory R. Andrews and Ronald A. Olsson, “The SR Programming Language”: Concurrency in Practice, ISBN 0-8053-0088-0, Benjamin/Cummings Publishing Company,1993.

[59] G. Sutcliffe and J. Pinakis, ‘‘Prolog-D-Linda: An Embedding of Linda in SICStus Prolog,’’ Technical Report 91/7, Department of Computer Science, The University of Western Australia, Perth, Australia.

[60] H. El-Rewini and T. Lewis, ‘‘Scheduling and Performance Evaluation Tool for Parallel Computing,’’ Proceedings of 4th Annual Symposium on Parallel Processing,Fullerton, CA., p. 60, April 1990.

[61] H. El-Rewini and T. Lewis, ‘‘Scheduling Parallel Program Tasks onto Arbitrary Target Machines,’’ Journal of Parallel and Distributed Computing, vol. Vol 9, pp.138-153, June 1990.

[62] Parasoft Co., Express C User’s Guide, Version 3.0, 1990.

[63] J. J. Dongarra, G. A. Geist, R. Manchek, and V. S. Sunderam, ‘‘A Users’ Guide to PVM,’’ Technical Report No. ORNL/TM-11826, Oak Ridge National Laboratory,Oak Ridge, Tennessee 37831-6367, July 1991.

[64] V. S. Sunderam, ‘‘PVM : A Framework for Parallel Distributed Computing,’’ Concurrency:Practice and Experience, vol. Vol. 2 No. 4, pp. 315--339, Dec. 1990.

[65] J. J. Dongarra, G. A. Geist, R. Manchek, J. Plank, and V. Sunderam, ‘‘HeNCE: A User’s Guide Version 1.2,’’ Technical Report, Computer Science Department, CS-92-157, February 1992.

[66] J. J. Dongarra, G. A. Geist, R. Manchek, K. Moore, R. Wade, and V. S. Sunderam, ‘‘HeNCE: Graphical Development Tools for Network-Based Concurrent Computers,’’Proceedings of the Scalable High Performance Computing Conference IEEE Computer Society Press, pp. 129-136, April 1992, Williamsburg.

[67] Ralph Butler Ewing Lusk, ‘‘User’s Guide to the p4 Parallel Programming System,’’Technical Report, Argonne National Laboratory, Mathematics and Computer Science Division, ANL-92/17, Oct. 1992.

[68] Timothy G. Mattson, Craig C. Douglas, and Martin H. Schultz, ‘‘A Comparison of CPS, LINDA, P4, POSYBL, PVM, and TCGMSG: Two Node Communication Times,’’ Yale University Technical Report YALEU/DCS/TR-975, 1993.

[69] Robert J. Harrison, ‘‘Moving Beyond Message Passing: Experiments With A Distributed- Data Model,’’ Technical Report, Argonne National Laboratory, 1991.

[70] M. Fischler, G. Hockney, and P. Mackenzie, ‘‘Canopy 5.0 Manual,’’ Technical Report, Fermilab.

[71] M. Fischler, ‘‘The ACPMAPS System - A Detailed Overview,’’ Fermilab publication FERMILAB-TM-1780.

[72] Fausey, Rinald, Wolbers, Potter, Yeager, Ullfig, ‘‘CPS & CPS Batch Reference Guide,’’ Fermi Computing Division #GA0008.

[73] Fausey, Rinaldo, Wolbers, Potter, Yeager, Ullfig, ‘‘CPS User’s Guide,’’ Fermi Computing Division #GA0009.

[74] Frank Rinaldo and Stephan Wolbers, ‘‘Loosely-Coupled Parallel Processing at Fermilab,’’To be published in ’Computers in Physics’, March-April 1993.

[75] Matthew R. Fausey, ‘‘CPS and the Fermilab Farms,’’ FERMILAB-Conf-92/163, June 1992.

[76] Tom Nash, ‘‘High Performance Parallel Local Memory Computing at Fermilab,’’ Proceedings of WHP92 on Heterogeneous Processing, Beverly Hills, Calif, March 23, 1992.

[77] L. Bomans, R. Hempel, and D. Roose, ‘‘The Argonne/GMD macros in Fortran for portable parallel programming and theirimplementation on the Intel iPSC/2,’’ Parallel Computing, North-Holland, vol. Vol. 15, pp. 119-132, 1990.

[78] R. Hempel, ‘‘The ANL/GMD Macros (PARMACS) in Fortran for Portable Parallel Programming using the Message Passing Programming Model,’’ User’s Guide and Reference Manual, Version 5.1, Nov. 1991.

[79] R. Hempel, H.-C. Hoppe, and A. Supalov, ‘‘PARMACS 6.0 Library Interface Specification,’’GMD internal report, Dec. 1992.

[80] R. Hempel and H. Ritzdorf, ‘‘The GMD Communications Library for Grid-oriented Problems,’’ GMD Arbeitspapier No. 589.

[81] Diane T. Rover and Joan M. Francioni, ‘‘A Survey of PICL and Paragraph Users (1992),’’ Technical Report, Department of Electrical Engineering, Michigan State University (TR-MSU-EE-SCSL-01193), Feb. 1993.

[82] G. A. Geist, M. T. Heath, B. W. Peyton, and P. H. Worley, ‘‘PICL, A Portable Instrumented Communication Library, C Reference Manual,’’ ORNL/TM-11130, July, 1990.

[83] A. Quealy, G. L. Gole, and R. A. Blech, ‘‘Portable Programming on Parallel/Networked Computers Using the

Application Portable Parallel Library (APPL),''to be published as a NASA TM, 1993.

[84] R. Ponnusamy, R. Das, J. Saltz, and D. Mavriplis, ''The Dybbuk Runtime System,'' Compcon, San Francisco, February 1993.

[85] C. Chase, K. Crowley, J. Saltz, and A. Reeves, ''Parallelization of Irregularly Coupled Regular Meshes,'' Sixth International Conference on Supercomputing, Washington DC, June 1992.

[86] R. Das, D. J. Mavriplis, J. Saltz, S. Gupta, and R. Ponnusamy, ''The Design and Implementation of a Parallel Unstructured Euler Solver Using Software Primitives (AIAA-92-0562),,'' Proceedings of the 30th Aerospace Sciences Meeting, Reno NV, 1992.

[87] A. Sussman, J. Saltz, R. Das, S. Gupta, D. Mavriplis, and R. Ponnusamy, ''PARTI Primitives for Unstructured and Block Structured Problems,''Computing Systems in Engineering (Proceedings of Noor's Flight Systems conference), pp. 73-86, 1992, 3, 1.

[88] R. Kannan, et. al., ''Software Environment for Network Computing,'' Workshop on"Heterogeneous Network-Based Concurrent Computing" SCRI/Florida State University, October 16-18, 1991 also appears in the newsletter for the IEEE Technical Committee on Operating Systems and Application Environments, Vol. 6, No. 1,1992. Also available as a technical report (CERC-TR-RN-91-007) from CERC,WVU, Morgantown, WV 26505.

[89] R. Kannan, C.L.Chen, Michael Packer, and Hawa Singh, ''Directory Service for Group Work,'' CSCW 92 Tools & Technologies Workshop, Toronto Canada. Also available as a technical report (CERC-TR_RN-91-004) from CERC, WVU, Morgantown,WV 26505.