Software and Hardware Requirements for Some Applications of Parallel Computing to Industrial Problems

Geoffrey C. Fox
gcf@npac.syr.edu
http://www.npac.syr.edu

Northeast Parallel Architectures Center
111 College Place
Syracuse University
Syracuse, New York 13244-4100

1 Introduction

This paper combines a recent survey of industrial applications with a problem classification developed for largely academic applications. We show how this allows one to isolate the parallel computing hardware and software characteristics needed for each problem. The industrial applications come from a survey undertaken of New York State industry in 1991 and 1992. Further details of the survey will be found in [Fox:92e], [Fox:94b], [Fox:94h], [Fox:94c], [Fox:94i], [Mills:93a]. Here, we summarize relevant features of it in Section 2. Section 3 reviews and extends a classification of problem architectures originally developed in 1988 from a rather complete survey of parallel applications at the time [Fox:88b], [Fox:88tt], [Fox:91g], [Fox:94a].

In Section 4, we show how the different problem categories or architectures are addressed by parallel software systems with different capabilities. We give illustrative examples, but not an exhaustive list of existing software systems with these characteristics. We consider High Performance Fortran and its extensions as a data parallel language; message passing systems, such as those supplied with commercial multicomputers; as well as approaches to software integration. In Section 3, we point that our old classification of problems omitted what we can call *metaproblems*—problems built up from several components—each of which could have its own parallelization issues.

In Section 5, we combine the previous three sections and describe informally the problem architecture, and possible software and hardware needs for a selection of industrial applications. We have grouped our discussion into three broad case studies; computational chemistry; computational fluid dynamics and manufacturing; Monte Carlo methods. Each case study covers several of the 33 industrial areas identified in Section 2. Further, in each case study, we find good examples of the different problem architectures and show that each application area requires a rich software and hardware architecture support.

2 Industrial Applications

In 1991–92, New York State funded a survey of the industrial opportunities for parallel computing. This was part of a project, ACTION (Advanced Computing Technology is an Innovative Opportunity Now), at Syracuse University. ACTION's goal was to accelerate the use of parallel computing into the industry of New York State. The purpose of the survey was to isolate some initial projects and, more generally, to provide information on which to develop a long-term plan for ACTION. Relevant information included the basic parallel computing software and algorithm technologies needed for particular industry sectors. In this paper, we concentrate on the software capabilities needed for these applications, and also the appropriate parallel machine architectures. Further discussion of the survey technique, particular companies interviewed, and the detailed nature of the applications will be found in [Fox:92e], [Fox:94a], and papers cited earlier. The survey has inevitable limitations. There are many important applications—oil exploration is a good example—that are not well represented in New York State. Further, the survey was not complete within its limited geographic region, and is still being extended.

Tables 1–4 summarizes the industrial opportunities for parallel computing in the form we will use them. Some 50 different applications used in the survey have been broken up into 17 distinct areas. This is certainly somewhat arbitrary, and there are many overlaps (and omissions as discussed). The importance, difficulty of implementation, and degree of risk also differ from case to case. However, these issues will not be discussed here.

Table 1: Guidelines used in Developing Categories of Industrial and Government Applications of HPCC shown in Tables 3–4

- Define information generally to include both CNN headline news and the insights on QCD gotten from lattice gauge theories. There are four broad categories.

- **Information Production** (e.g., Simulation)

  • Major concentration of MPP and HPCC at present

- **Information Analysis** (e.g., Extraction of location of oil from seismic data, Extraction of customer preferences from purchase data)

  • Growing area of importance and Short term major MPP opportunity in decision support combined with parallel databases

- **Information Access and Dissemination**—*InfoVision* (e.g., Transaction Processing, Video-On-Demand)

  • Enabled by National Information Infrastructure

  • Very promising medium term market for MPP but need the NII to be reasonably pervasive before area "takes off"

  • MPPs used as high performance, high capacity, multi-media servers

- **Information Integration**

  • Integrates Information Production, Analysis and Access, e.g.,
    - Decision support in business
    - Command and Control for Military
    - Concurrent Engineering and Agile Manufacturing

  • Largest Long Term Market for MPP

Table 2: Abbreviations used in Tables 3–4 of Industrial Applications of HPCC

Adaptive	Software for Irregular Loosely Synchronous Problems handled by pC++, HPF extensions, Message Passing (Table 6)
Asyncsoft	Parallel Software System for (particular) class of asynchronous problems (Table 6)
CFD	Computational Fluid Dynamics
ED	Event Driven Simulation
FD	Finite Difference Method
FEM	Finite Element Method
HPF	High Performance Fortran [HPF:93a], [HPFF:95a]
HPF+	Natural Extensions of HPF [Choudhary:92d], [HPF:94a], [HPFapp:95a]
Integration	Software to integrate components of metaproblems (Table 6)
MPF	Fortran plus message passing for loosely synchronous programming support
PDE	Partial Differential Equation
TS	Time Stepped Simulation
VR	Virtual Reality

Note on Language: HPF, MPF use Fortran for illustration, one can use parallel C, C++ or any similar extensions of data parallel or message passing languages

Table 3: Five Categories of Problems

Problem Architecture	**Overall Software Issue**
Synchronous: Data Parallel Tightly coupled and software needs to exploit features of problem structure to get good performance. Comparatively easy as different data elements are essentially identical.	Data Parallel
Loosely Synchronous: Data Parallel As above, but data elements, and/or their linkage, are not identical. Still parallelizes due to macroscopic time synchronization.	Data Parallel
Asynchronous: Functional (or data) parallelism that is irregular in space and time. Often loosely coupled and so need not worry about optimal decompositions to minimize communication. Hard to parallelize (massively) unless ...	Difficult Task parallel
Embarrassingly parallel: Essentially independent execution of disconnected components (can involve reductions).	Possible in most software
Metaproblems Asynchronous collection of (loosely) synchronous components where these programs themselves can be parallelized	Coarse grain task parallelism—each component data parallel

Table 4: Industrial HPCC Applications 1 to 5: SIMULATION

Item	Application Area and Examples	Problem Comments	Machine and Software
1	**Computational Fluid Dynamics** • Aerospace • Military, Civilian Vehicles • Propulsion	• PDE, FEM • Turbulence • Mesh Generation	• SIMD, MIMD for irregular adaptive • HPF(+) but • Unclear for adaptive irregular mesh
2	**Structural Dynamics**	• PDE, FEM • Dominated by Vendor Codes such as NASTRAN	• MIMD as complex geometry • HPF(+)
3	**Electromagnetic Simulation** • Antenna Design • Stealth Vehicles • Noise in high frequency circuits • Mobile Phones	• PDE solved by moment method • Matrix solve dominates	SIMD HPF
		• Newer FEM and FD Methods? • Also fast multipole	SIMD, MIMD HPF(+)
4	**Scheduling** • Manufacturing • Transportation (Dairy delivery to Military deployment) • University Classes • Airline Scheduling of crews, planes in static or dynamic (Syracuse snow storm) cases	Expert Systems and/or	MIMD (unclear Speedup) AsyncSoft
		Neural Networks Simulated annealing	SIMD HPF
		Linear Programming (hard sparse matrix)	MIMD HPF+?
5	**Environmental Modeling**— Earth/Ocean/Atmospheric Simulation	• PDE, FD, FEM • Sensitivity to Data	• SIMD but • MIMD for irregular adaptive mesh • HPF(+) except this unclear for adaptive irregular mesh

Table 4: Industrial HPCC Applications 6 to 10: SIMULATION

Item	Application Area and Examples	Problem Comments	Machine and Software
6	**Environmental Phenomenology** —Complex Systems, (Lead Concentration in blood)	• Empirical Models • Monte Carlo and Histograms	• Some SIMD but • MIMD more natural • HPF
7	**Basic Chemistry** • Chemical Potentials • Elemental Reaction Dynamics	• Calculate Matrix Elements • Matrix Eigenvalue determination, Inversion, Multiplication	• Probably SIMD with perhaps SIMD possible • HPF
8	**Molecular Dynamics** in Physics & Chemistry • Biochemistry • Discrete Simulation Monte Carlo for CFD (DSMC) • Particle in the Cell (PIC)	• Particle Dynamics with irregular cutoff forces • Fast Multipole Methods • Mix of PDE and Particle methods in PIC and DSMC	• HPF(+) except • need MPF for fast multipole
9	**Economic Modelling** • Real Time Optimization • Mortgaged backed Securities • Option Pricing	Single financial instrument by Monte Carlo	SIMD, HPF
		Full Simulations of complete portfolios	MIMD or SIMD with Integration Software
10	**Network Simulations** • Electrical Circuit • Microwave and VLSI • Biological (neural) Circuit	• Sparse matrices • Zero structure defined by connectivity	MIMD HPF for matrix elements MPF or library for matrix solve

Table 4: Industrial HPCC Applications 11 to 13: SIMULATION

Item	Application Area and Examples	Problem Comments	Machine and Software
11	Particle Transport Problems	Monte Carlo Methods as in neutron transport for (nuclear) explosion simulations	MIMD HPF
12	**Graphics** (rendering) • Hollywood • Virtual Reality	• Several Operational Parallel Ray tracers • Distributed model hard	HPF for simple ray tracing but MPF for best algorithms
			MIMD & Asyncsoft for distributed database
13	**Integrated Complex System Simulations** • Defense (SIMNET, Flight Simulators) • Education (SIMCITY) • Multimedia/VR in Entertainment • Multiuser Virtual Worlds • Chemical and Nuclear Plants	• Event Driven (ED) and • Time stepped (TS) simulations • Virtual Reality Interfaces • Database backends • Interactive	Timewarp or other Event Driven Simulation needs Appropriate Asyncsoft
			Integration Software Database
			HPF+ for TS Simulation

Table 4: Industrial HPCC Applications 14 to 18: Information Analysis—"DataMining"

Item	Application Area and Examples	Problem Comments	Machine and Software
14	Seismic and Environmental Data Analysis	• Parallel Computers already important but • No oil in New York State	• SIMD useful but MIMD might be necessary • HPF
15	Image Processing • Medical Instruments • EOS (mission to Planet Earth) • Defense Surveillance • Computer Vision	• Many commercial Applications of Defense Technology • Component of many "metaproblems" (Information Integration category) • e.g., Computer Vision in Robotics	Metacomputer Low Level Vision is SIMD and HPF Medium/High Level Vision is MIMD and HPF(+) Software Integration needs Asyncsoft and Database
16	Statistical Analysis Packages and Libraries	• Optimization • Histograms • See application area 4	HPF+ and especially C++ analogues is excellent for many libraries
17	Healthcare Fraud • Inefficiencies • Securities Fraud • Credit Card Fraud	Linkage Analysis of Data records for correlations and outlier detection	• SIMD or MIMD • Parallel Relational Database access plus application area 16
18	Market Segmentation • Mail Order • Retail • Banking	Sort and classify records to determine customer preference by region from city to even individual home	• Some cases are SIMD • Parallel Database plus application area 16

Table 4: Industrial HPCC Applications 19 to 22 for Information Access *InfoVision*—Information, Video, Imagery and Simulation on Demand

Item	Application Area	Comments	Problem Structure	Machine & Software
19	**Transaction Processing** • ATM (automatic teller machine)	Database-most transactions short. As add "value" this becomes Information integration	Embarrassingly Parallel	MIMD Database
20	**Collaboration** • Telemedicine • Collaboratory for Research • Education • Business	Research Center or doctor(s)—patient interaction without regard to physical location	Asynchronous	High Speed Network
21	**Text on Demand** • Digital (existing) libraries • ERIC Education database, • United Nations-Worldwide newspapers	Multimedia database (see areas 22, 23) Full text search	Embarrassingly Parallel	MIMD Database
22	**Video on Demand** • Movies, News (CNN Newsource & Newsroom), • Current cable, • United Nations-Policy Support	Multimedia Database Interactive VCR, Video Browsing, Link of video and text database	Embarrassingly Parallel for multiple Users Interesting parallel compression	MIMD Database Video Editing Software SIMD compression

Table 4: Industrial HPCC Applications 23 to 24 for Information Access *InfoVision*—**Information, Video, Imagery and Simulation on Demand**

Item	Application Area	Comments	Problem Structure	Machine & Software
23	**Imagery on Demand** • Kodak GIODE • "clip art" on demand • Medical images • Satellite images	Multimedia database Image Understanding for Content searching and (terrain) medical feature identification	Metaproblem Embarrassingly Parallel plus Loosely Synchronous Image Understanding	MIMD but much SIMD image analysis
24	**Simulation on Demand** • Education, Tourism, City planning, • Defense mission planning	Multimedia map database Generalized flight simulator Geographical Information System	Synchronous terrain rendering with Asynchronous Hypermedia	SIMD terrain engine (parallel rendering) MIMD database **Integration** software

Table 4: Information Integration Applications 25 to 28

- These involve combinations of Information Production, analysis, Access and Dissemination and thus need the Integration of the various Software and Machines Architecture Issues discussed under previous application areas.

- Sometimes Called System of Systems

- **25: Military and Civilian Command and Control** (C^2, C^3, C^4I ...)

  • Battle Management, Command, Control, Communication, Intelligence and Surveillance (BMC^3IS)
  • Military Decision Support
  • Crisis Management—Police and other Government Operations
  • SIMNET simulates this and with people and computers in the loop has many of same issues

- **26 to 28: Applications of InfoVision Services**

  • Generalize Compuserve, Prodigy, America Online, Dialog and Other Information Services

  •**26: Decision Support for Society**

  • Community Information Systems

  • Travel and Generalized Yellow Page Services

  •**27: Business Decision Support**—One example is:

  • Health Care with Image and Video databases supporting telemedicine

  •**28: Public Administration and Political Decision Support**

  • Government Information Systems

  • Maxwell School at Syracuse University teaches use of realtime video to aid world wide decisions (United Nations)

Table 4: **Information Integration Applications 29 to 33**

- **29: Real-Time Control Systems**
  - Robotics uses Imagery to make decisions (control vehicles)
  - Energy management controls power use and generation
- **30: Electronic Banking**
  - Requires Security, Privacy, Electronic Cash, etc.
- **31: Electronic Shopping**
- **32: Agile Manufacturing—Multidisciplinary Design and Concurrent Engineering**
  - Combines CAD with Applications 1 to 3
  - Requires major changes to Manufacturing Infrastructure and Approach
- **33: Education**
  - InfoMall Living Textbook—6 Schools on ATM network linked to HPCC InfoVision Servers at NPAC [Mills:95a]

Table 1 describes the general guidelines used in organizing Table 4. Note that we did not directly cover academic areas, and a more complete list (which included our industrial table) was produced by the Petaflops meeting [Peta:94a]. Notice that Tables 1–4 are organized around the concept of "information." This corresponded to an early realization from the survey that the major industrial opportunities for HPCC in New York State were information related. Thus, for instance, simulation is subtitled "Information Production" with say, computational fluid dynamics simulations providing information to be used in either manufacturing (application 32) or education (application 33). It is not directly relevant to this paper, but the results of this survey caused the ACTION program to refocus its efforts and evolve into InfoMall [Fox:93c], [Fox:94f], [Fox:94h], [Fox:95b], [Infourl:95a], [Mills:94a]. Here, "Info" refers to the information based application focus and "Mall" to the use of a virtual corporation (groups of "storeholders") to produce the complex integrated applications enabled by HPCC.

The first column of Table 4 contains the area label and some sample applications. Algorithmic and other comments are in column two. The third and fourth columns describe, respectively, the problem architecture and an estimate of appropriate parallel software approach. The background for these two columns is described in the following two sections.

This paper is not intended to advocate a particular parallel software environment or language. Rather, we want to describe the broad capabilities in the parallel programming paradigm needed for the applications of Table 4. We believe that the programming functionality needed by a particular application is broadly determined by the problem architecture described in the following section. In discussing software needs, we do not discuss all the components of the parallel software environment and just those relevant for expressing problems.

For this reason, we use broad software classifications using, for instance, MPF (Fortran plus message passing) as typical of all similar explicit messaging systems—one could substitute here C plus message passing, or Fortran M programming environments. Again, PVM, MPI, or any such message passing system could be used without changing the significance of the tables. High Performance Fortran is used as a typical data parallel language, although this has an evolving definition and similar C++ environments could well be more attractive, and can be substituted in the table.

3 Problem Architectures

We have described our classification of problem architectures several times before, but here we just summarize it.

This classification [Angus:90a], [Denning:90a], [Fox:88b;90p;91g;94a], was deduced from our experience at Caltech combined with a literature survey that was reasonably complete up to the middle of 1989. At Caltech, we developed some 50 applications on parallel machines, 25 of which led to publications in the scientific literature, describing the results of simulations performed on our parallel computers [Fox:87d] [Fox:88a], [Fox:88oo], [Fox:89n]. Our Caltech work was mainly on the hypercube, but the total of 300 references used in original classification covered work on the Butterfly, transputers, the SIMD Connection Machine, and DAP. We originally identified three temporal structures and one especially important (as it was so

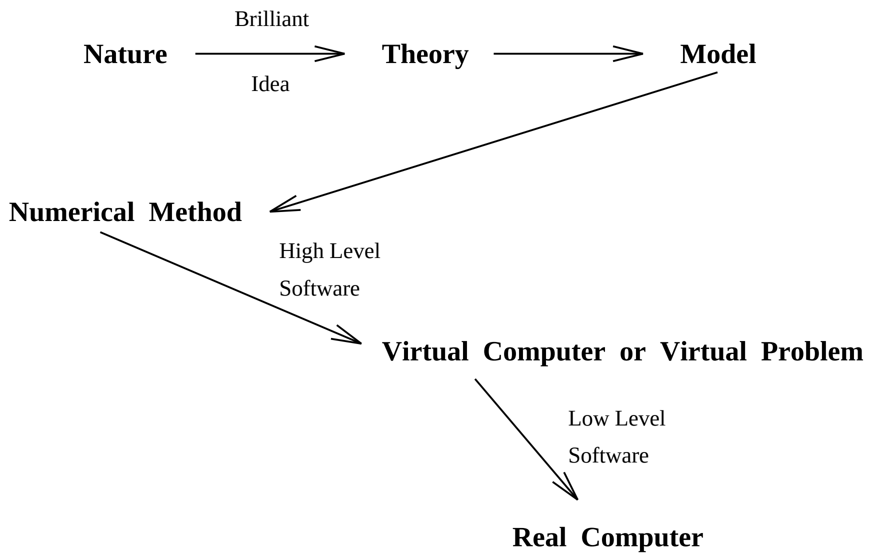


Figure 1: Computation and Simulation as a Series of Maps

simple) spatial structure, which are the first four entries in Table 4. Chapter 3 of [Fox:94a] describes a "complex systems" approach to computation and introduces the spatial and temporal structure of problems and computers. We studied software as a mapping (Figure 1) of problems to computers with the software structure determined by the structure (architecture) of both the individual complex systems—computers and problems—and their interrelation. In Figure 2, we summarize issues in the spatial-temporal plane. "Space" here refers to data (problem) or nodes and their linkage (computer). "Time" is iteration number and simulation time (problem) or counts clock cycles (computer).

The three general temporal structures are called synchronous, loosely synchronous, and asynchronous. The temporal structure of a problem is analogous to the hardware classification into SIMD and MIMD. Further detail is contained in the spatial structure or computational graph of Figure 3a describing the problem at a given instant of simulation time [Fox:88tt]. This is important in determing the performance, as shown in Chapter 3 of [Fox:94a] of an implementation, but it does not affect the broad software issues discussed here. In Table 4, we only single out one special spatial structure, "embarrassingly parallel," where there is little or no connection between the

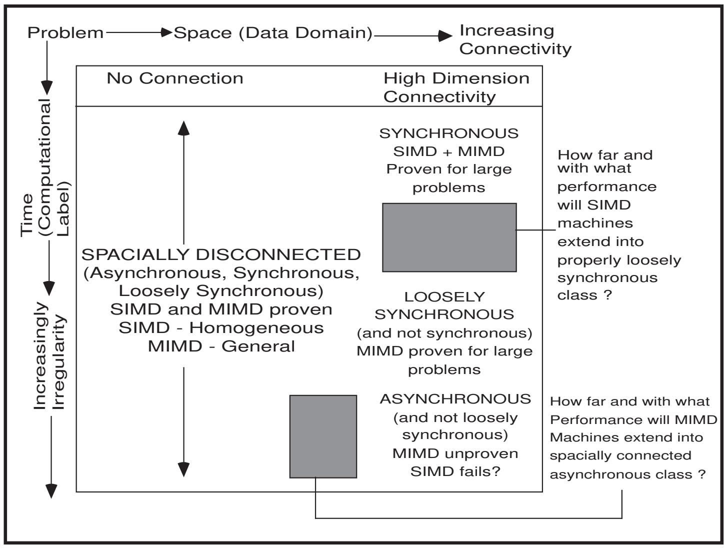


Figure 2: Issues Affecting Relation of Machine, Problem, and Software Architecture

individual parallel program components. For embarrassingly parallel problems, illustrated in Figure 4, the synchronization (both software and hardware) issues are greatly simplified.

Synchronous problems are data parallel in the language of Hillis [Hillis:87a] with the restriction that the time dependence of each data point is governed by the same algorithm. Both algorithmically and in the natural SIMD implementation, the problem is synchronized microscopically at each computer clock cycle. Such problems are particularly common in academic applications as they naturally arise in any description of some world in terms of identical fundamental units. This is illustrated in Figure 5 by quantum chromodynamics (QCD) simulations of the fundamental elementary particles that involve a set of gluon and quark fields on a regular four-dimensional lattice. These computations form one of the largest use of supercomputer time in academic computing.

Loosely synchronous problems are also typically data parallel, but now we allow different data points to be evolved with distinct algorithms. Such problems appear whever one describes the world macroscopically in terms of the interactions between irregular inhomogeneous objects evolved in a time synchronized fashion. Typical

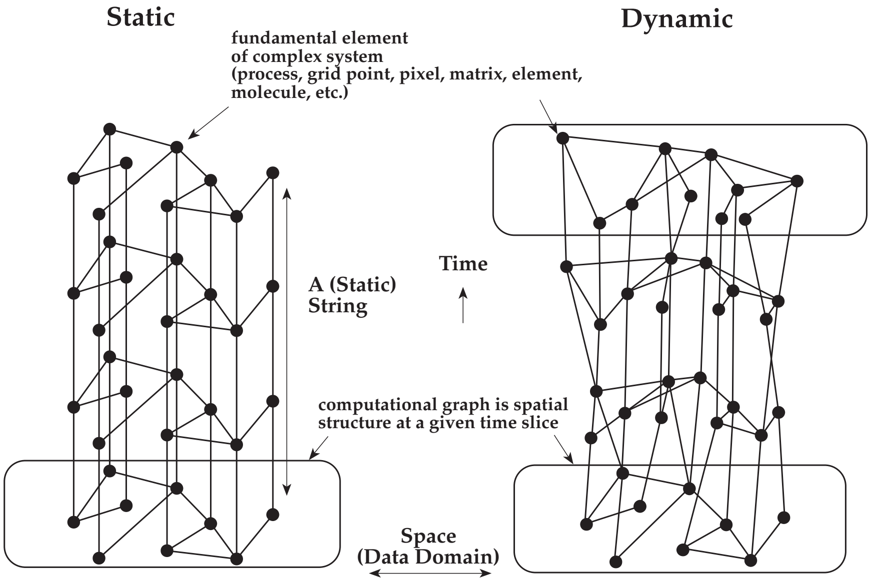
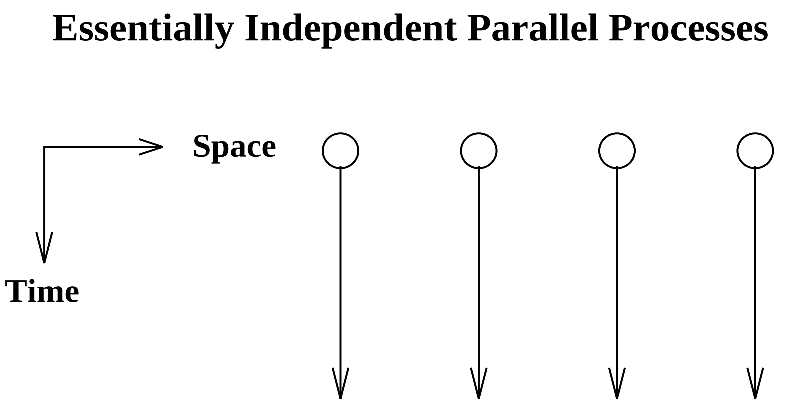


Figure 3: (a) Synchronous, Loosely Synchronous (Static), and (b) Asynchronous (Dynamic) Complex Systems with their Space-Time Structure



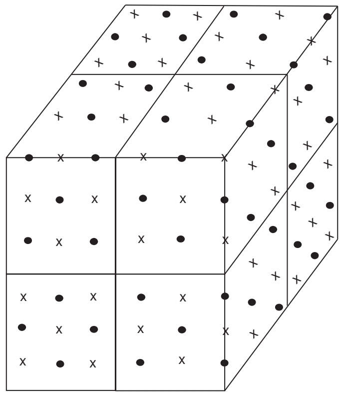
Example: Divide large database among processors and independently search each portion of database to answer query.

Figure 4: Embarrassingly Parallel Problem Class

SYNCHRONOUS PROBLEMS

For example:

Microscopic Description of Fundamental Interactions
In Particular, QCD

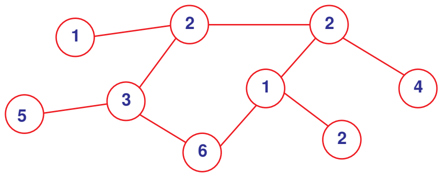


- Computational structure (almost) identical for all elements in the data domain
- Parallelize by regular partition of data domain
- Run well on SIMD machines
- Message Passing or High Performance Fortran implementation on MIMD machines

Figure 5: The Synchronous Problem Class

Loosely Synchronous Problems

For example: Macroscopic description of physical system in terms of interactions between irregular inhomogeneous objects evolved as a time synchronized simulation. In particular - biological neural network



Parallelize by irregular partition of data domain

Hardware:
In general will not run well on SIMD machine.

Software:
Initial version of High Performance Fortran cannot describe.

Message passing or extensions of High Performance Fortran on MIMD machines will describe.

Figure 6: The Loosely Synchronous Problem Class

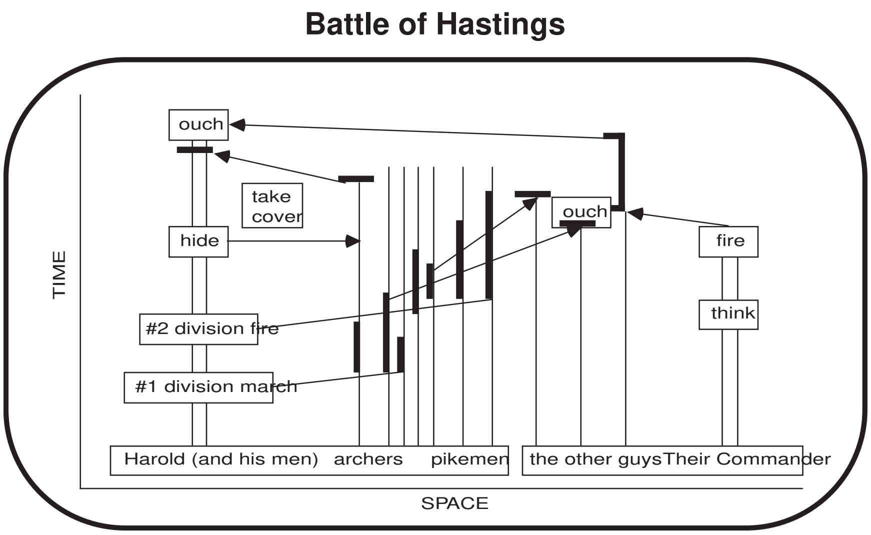
examples, as in Figure 6, are computer or biological circuit simulations where different components or neurons are linked irregularly and modelled differently. Time driven simulations and iterative procedures are not synchronized at each microscopic computer clock cycle, but rather only macroscopically "every now and then" at the end of an iteration or a simulation time step.

Loosely synchronous problems are spatially irregular, but temporally regular. The final *asynchronous* class is irregular in space and time, as in Figure 3b. A good example is an event driven simulation, illustrated in Figure 7, that can be used to describe the irregular circuits we discussed above, but now the event paradigm replaces the regular time stepped simulation. Other examples include computer chess [Felten:88i] and transaction analysis. Asynchronous problems are hard to parallelize and some may not run well on massively parallel machines. They require sophisticated software and hardware support to properly synchronize the nodes of the parallel machine, as is illustrated by time warp mechanism [Wieland:89a].

Both synchronous and loosely synchronous problems parallelize on systems with many nodes. The algorithm naturally synchronizes the parallel components of the problem without any of the complex software or hardware synchronization mentioned above for event driven simulations. In the original survey, 90% of the surveyed applications fell into the classes that parallelize well. This includes 14% from the embarrassingly parallel classes, and roughly equal (38% each) amounts from synchronous or loosely synchronous class. It is interesting that massively parallel distributed memory MIMD machines that have an asynchronous hardware architecture are perhaps most relevant for loosely synchronous scientific problems.

We have looked at many more applications since the detailed survey in [Fox:88b], and the general picture described above remains valid. Industrial applications have less synchronous and more loosely synchronous problems than academic problems. We have recently recognized that many complicated problems are mixtures of the basic classifications. The first major example with which I was involved was a battle management simulation implemented by my collaborators at JPL [Meier:89a]. This is formally asynchronous with temporally and spatially irregular interconnections between various modules, such as sensors for control platforms and input/output tasks. However, each module uses a loosely synchronous algorithm, such as the multi-target Kalman filter [Gottschalk:90b] or the target-weapon pairing system. Thus, the whole metaproblem consists of a few ($\sim$ 10–50) large grain asynchronous objects, each of which is a data parallel synchronous or loosely synchronous algorithm. This type of asynchronous problem can be implemented in a scaling fashion on massively parallel machines. We call this a metaproblem or asynchronous combination of several synchronous or loosely synchronous problems. A similar example of this asynchronous or embarrassingly synchronous problem class is machine vision and signal processing, where one finds an asynchronous collection of data parallel modules to perform various image processing tasks, such as stereo matching and edge detection. Figure 8 illustrates another example where we outline an approach to designing a new airframe that involves

For example:
The world looked at macroscopically in terms of interactions between irregular inhomogeneous objects evolved as an event-driven simulation



- Parallelize by "data parallelism" over space of events but no automatic algorithmic synchronization
- Need sophisticated software built on top of message passing between events to ensure synchronization
- Speedup very problem-dependent
- MIMD architectures essential

Figure 7: The Asynchronous Problem Class

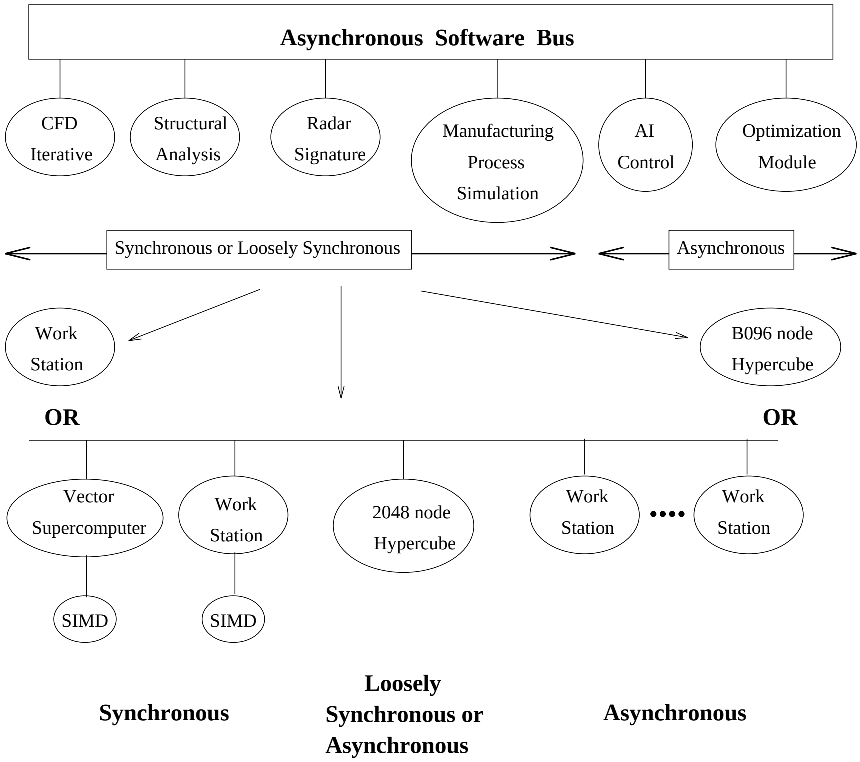


Figure 8: The Mapping of Heterogeneous Metaproblems onto Heterogeneous Metacomputer Systems

aerodynamics, structures, radar signature, and the optimization discussed later in Section 5.2. This figure also points out the interesting analogy between heterogeneous metaproblems of class and a heterogeneous computer network.

In the above cases, the asynchronous components of the problems were large grain modules with modest parallelism. This can be contrasted with Otto and Felten's MIMD computer chess algorithm, where the asynchornous evaluation of the pruned tree is "massively parallel" [Felten:88i]. Here, one can break the problem up into many loosely coupled but asynchronous parallel components, which give excellent and scalable parallel performance. Each asynchronous task is now a synchronous or loosely synchronous modestly parallel evaluation of a given chess position.

There were a few examples mentioned above of metaproblems in our original survey, but a major part of Table 4, from our New York State activity, is the Information Integration classification, including manufacturing and the applications 25–33 are essentially all metaproblems. As stated boldly in Table 1, this class is the most important long-term area for HPCC. Further, as in battle management case, many problems that formerly appear asynchronous and were classified in this way in our original survey, are in fact metaproblems. Thus, the parallelism does not come from the difficult (impossible?) asynchronous structure, but the synchronous or loosely synchronous components buried inside the asynchronous shell. Thus, we believe metaproblems and their software support very important.

4 Some Software and Machine Issues

Naturally parallel implementations work "best" if the machine architecture is "similar" to that of the problem. This is summarized in Table 5 where to be precise, success requires that the machine architecture "contains" (is a superset of) the problem architecture. Thus, both SIMD and MIMD machines express synchronous problems, but SIMD machines are typically unsuitable for loosely synchronous problems.

Table 5: What is the "Correct" Machine Architecture for each Problem Class?

Problem Class	*Machine*
Synchronous	SIMD, MIMD
Loosely Synchronous	MIMD, maybe SIMD
Asynchronous	MIMD, but may not perform well without special hardware features
Compound (Metaproblems)	Heterogeneous network (including World Wide Web)
Embarrassingly Parallel	Network of workstations MIMD, World Wide Web, sometimes SIMD

Software systems need to be designed so that they can express problems well, and be targeted to relevant machines. Software should not be designed for a particular machine model—it expresses problem and not machine characteristics.

Table 6: Candidate Software Paradigms for Each Problem Architectures

- **Synchronous**: High Performance Fortran (HPF) [Foster:95a], [HPFCSep:95a], [Koelbel:94a]; Fortran 77D [Bozkus:93a], [Fox:91e], [Hiranandani:92c]; Vienna Fortran [Chapman:92b]; C* [Hatcher:91a;91b]; Crystal [Chen:88b]; APL; Fortran for SIMD parallel computers

- **Loosely Synchronous**: Extensions of the above, especially HPF [Chapman:94b], [Choudhary:92d], [HPF:94a]; and parallel C++ [Bodin:91a], [Chandy:93a], [Grimshaw:93b], [Lemke:92a]; Fortran or C plus message passing [Fox:91m], [McBryan:94a]

- **Asynchronous**: Linda [Factor:90a;90b], [Gelertner:89a]; CC++ [Chandy:93a]; Time Warp [Wieland:89a]; PCN [Chandy:90a]; WebWork [Fox:95a]

- **Compound Metaproblems**: AVS [Mills:92a;92b], [Cheng:93a]; PCN, Linda (or Trellis built on Linda); Webwork; Fortran-M [Foster:95a]. Generally, extensions of ADA, Fortran, C, or C++ controlling modules written in synchronous or loosely synchronous approach

- **Embarrassingly Parallel**: Several approaches work?

  • PCN, Linda, WebWork, PVM [Sunderam:90a], Network Express [Parasoft:88a], ISIS [Birman:87a;87b;91a]

We have described those issues at length in [Fox:90p;91g;94a], and here we will just present a simple table (Table 6) mapping the five problem architectures into possible software environments. This is presented in a different fashion for HPF and HPC++ in Figure 9 and

Table 7, which also points out the distinct runtime support needed for each problem class. One always has a tradeoff between performance and flexibility. Systems listed under "asynchronous" in Table 6 can typically also be used for synchronous and loosely synchronous problems. As shown in Figure 10, the "asynchronous" software used on loosely synchronous problems will probably provide greater flexibility, but lower performance than software systems explicitly designed for this problem class.

Table 7: Imprecise Mapping of Problem Classes into Runtime and Language Terms

- **STATIC Runtime**
  - Synchronous and Embarrassingly Parallel Problems—current HPF
- **ADAPTIVE Runtime**
  - Loosely Synchronous but not Synchronous—future capabilities of High Performance Fortran (HPF+) but can be supported well in message passing
- **ASYNCHRONOUS Runtime**
  - Asynchronous Problems
- **INTEGRATION Runtime and Programming Environments**
  - Metaproblems
  - AVS works well but also can be integrated into languages such as HPC++, Fortran-M

Loosely synchronous problems are in some sense the hardest as they have difficult irregularities which must be expressed with high efficiency by the underlying compiler and runtime systems. We, and others, have discussed this at length, both in general [Choudhary:92d;92e], [Fox:90p], [Goil:94a;95a], , and in case of High Performance Fortran [Bogucz:94a], [Chapman:94b], [Cheng:94e], [Choudhary:92g;94c], [Fox:94g], [Hawick:95a;95c], [HPF:94a], [HPFapp:95a], [Joubert:95a], [Muller:95a], [Robinson:95a], [Sturler:95a].

Note that Figure 9 refers to "HPF+"—this is some extension, called officially HPF2 (and later 3 perhaps) of HPF [HPF:93a], [HPFF:95a]

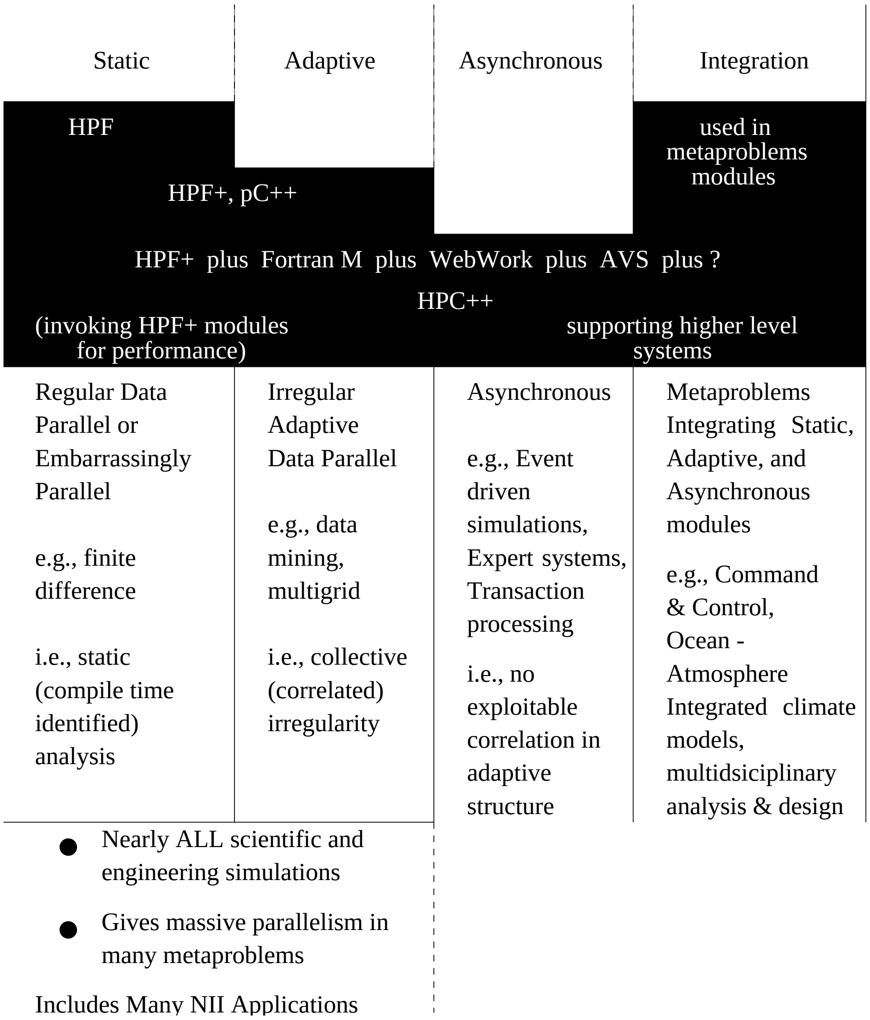


Figure 9: General Applicability of HPF, HPF+, HPC++ Classified by Problem Architecture and type of Runtime Support needed

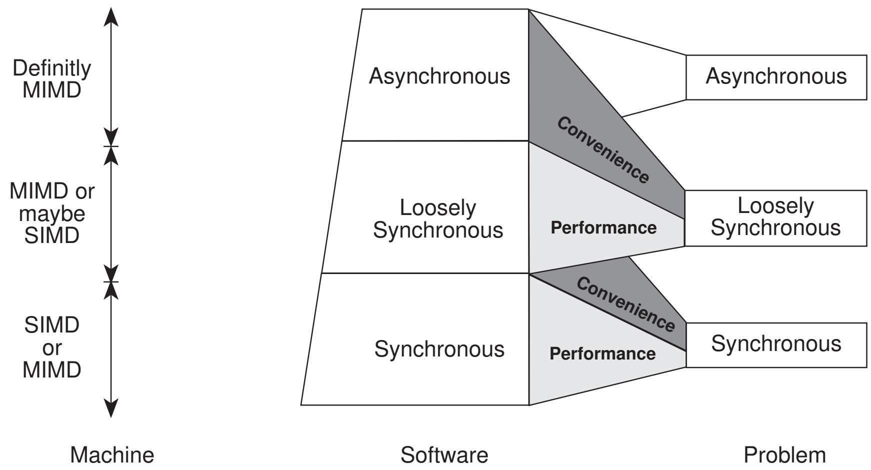


Figure 10: Mapping of Asynchronous, Loosely Synchronous, and Synchronous Levels or Components of Machine, Software and Problem. Each is pictured hierarchically with the asynchronous level at the top and synchronous components at lowest level. Any one of the components may be absent.

to fill gaps in the original language. The current HPF1 handles most synchronous and embarrassingly applications, but requires extension to handle the adaptive irregular data structures typical of loosely synchronous problems.

We now quantify these remarks with three case studies, which will link the material of Sections 2,3, and 4.

5 Machine and Problem Architectures and Appropriate Programming Paradigms in Three Case Studies

We now illustrate the different machine, problem, and software issues with three case studies. These are each broad application areas where there is no one approach. Rather, several very distinct application subclasses are present in each case for which different programming paradigms and machine architectures are appropriate.

5.1 Computational Chemistry and Electromagnetics (Applications 3, 7, and 8)

Many chemistry problems are formulated in terms of states of a chemical system, which can be labelled by an index corresponding to species, choice of wave function, or internal excitation (see Chapter 8 of [Fox:94a]). The calculation of energy levels, potential or transition probability can often be related to a matrix M_{ij} whose rows and columns are just the possible system states. M is often an approximation to the Hamiltonian of the system or it could represent overlap between the states. There are two key stages in such problems

a) firstly, calculate the matrix elements M_{ij}

b) secondly, perform one or more of a set of matrix operations

- Matrix Multiplication as in change of basis
- Matrix Eigenvalue determination as in energy level computations
- Matrix Equation solution as in solving multichannel scattering problems

This structure has been elegantly exploited within the "Global Array" programming model built at Pacific Northwest Laboratory [Nicplocha:94a] with a set of tools (libraries) designed for this class of computational chemistry problem.

These two steps have very different characteristics. The matrix element computations a), is of the embarrassingly parallel case as each M_{ij} can essentially be calculated independently even though subexpressions may be shared between two or more distinct M_{ij}. Each M_{ij} is a multi-dimensional integral with the computation depending on the details of the states i and j. Thus, this computation is very time consuming and is not suited for SIMD machines. The natural parallel algorithm associates sets of (i, j) with each node of a parallel computer. There are some relatively straightforward load balancing issues and essentially no internode communication. Thus, a MIMD cluster of workstations with modest networking is sufficient for this step a). The final matrix manipulations have quite a different character. These synchronous problem components are suitable for SIMD machine and often required substantial communication so that a workstation cluster will not be effective. Matrix multiplication could be exception as it is insensitive to latency and communication bandwidth for large matrices and so suitable for workstation clusters.

One of the standard approaches to computational electromagnetics (CEM) is the method of moments [Harrington:61a;67a;68a], [Jordon:69a]. This is a spectral method, which rather than solving the underlying partial differential equation (Maxwell's), expands the desired solution in a set of "moments". This leads to a similar situation to that described above for computational chemistry where i and j label moments for CEM and not the chemical state [Cheng:94a;94c]. Note that in both cases, the matrix M is treated as full [Cheng:94c], and is quite different from the familiar sparse matrices gotten from discretizing a partial differential equation. We note in passing that such spatial discretization is a quite viable approach to CEM and leads to a totally different computational problem architecture from the spectral moment formulation.

HPF can handle both stages of the matrix based CEM or chemistry problems [Robinson:95a]. The matrix solution stage exploits fully the Fortran 90 array manipulation and clearly requires good compiler support for matrix and vector manipulation primitives. NPAC's experience with a production CEM code PARAMOM from the Syracuse Research Corporation is illuminating [Cheng:94c]. Both stages could be implemented on IBM SP-2 with specialized Fortran code for the matrix element generation joined to SCALAPACK based matrix solution [Choi:92c]. However, the CM-5 implementation was not so simple. The CMSSL library provided exceptional matrix solution with good use being made of the CM-5's vector nodes. However, the matrix element computation was not so straightforward. Performance on the CM-5 nodes was poor and required conversion of the original Fortran 77 to Fortran 90 to both exploit the vector nodes and link to CMSSL. However, whereas the Fortran 90 notation was very suitable for matrix manipulation, it is quite irrelevant for the matrix element generation stage—as already explained, this exploits the `INDEPENDENT DO` and not the array notation for explicit parallelism. Thus, we split the PARAMOM code into a metaproblem with two sub-problems corresponding to the two stages discussed above. Now we implemented each stage on the most appropriate architecture. The "embarrassingly parallel" Fortran 77 matrix element generation stage was run on a network of workstations, the equation solution stage used the optimized libraries on the CM-5 or SP-2. The linkage of these stages used AVS, but one could alternatively use many other coordination software approaches. We expect to test our use of World Wide Web technology *WebWork* [Fox:95a] on this example.

This simple example illustrates three problem classes: embarrassingly parallel, synchronous and metaproblems, and associated machine and software architecture. There is an interesting software engineering issue. Typically, one would develop a single Fortran program for such a computational chemistry or electromagnetics problem. However, better is separate modules—in this case, one for each of two stages—for each part of problem needing different parallel computer treatment. In this way, we see the breakup of metaproblems into components, and use of systems such as AVS as helpful software engineering strategies [Cheng:92a;94d]. We have successfully used such an approach to produce an effective parallel version of the public domain molecular orbital chemistry code MOPAC [MOPAC:95a].

Not all chemistry computations have this structure. For instance, there is a set of applications such as AMBER and CHARMM that are based on molecular dynamics simulations, as described in Chapter 16 of [Fox:94a], [Ranka:92a]. These are typically loosely synchronous problems with each particle linked to a dynamic set of "nearest neighbors" combined with long-range nonbonded force computations. The latter can either use the synchronous $O(N^2_{\mathrm{particle}})$ algorithm or the faster, but complex loosely synchronous fast multiple $O(N_{\mathrm{particle}})$ or $O(N_{\mathrm{particle}} \log N_{\mathrm{particle}})$ approaches [Barnes:86a], [Edelsohn:91b], [Goil:94a], [Goil:95a], [Greengard:87b], [Salmon:90a], [Singh:93a], [Sunderam:93a], [Warren:92b], [Warren:93a].

5.2 Computational Fluid Dynamics and Manufacturing (Applications 1, 2, 3, 4, and 32)

CFD (Computational Fluid Dynamics) has been a major motivator for much algorithm and software work in HPCC, and indeed extensions of HPF have largely been based on CFD (or similar partial differential equation based applications) and molecular dynamics [Bogucz:94a], [Choudhary:92d;94c], [Dincer:95b], [Goil:94a;95a], [Hawick:95a;95b], , [HPF:94a]. Partial differential equations can be quite straightforward on parallel machines if one uses regular grids, such as those coming from the simplest finite difference equations. However, modern numerical methods use either finite elements or a refinement strategy for finite elements, which gives rise to irregular meshes. Approaches, such as domain decomposition and multigrid, also give use to complex data structures. From a Fortran programmer's point of view, simple finite differences can be well described by Fortran array data structures. Corresponding parallelization of such applications is well suited to the current HPF language, which is centered in decomposing arrays. All the more advanced partial differential equation schemes naturally need somewhat more sophisticated (than simple arrays) data structures, including arrays of pointers, linked lists, nested arrays, and complex trees. The latter are also seen in fast multipole particle dynamics problems, as well as fully adaptive PDE's [Edelsohn:91b]. Some excellent methods, such as the Berger-Oliger adaptive mesh refinement [Berger:84a] require modest HPF extensions as we have shown in our Grand Challenge work on colliding black holes [Haupt:95a]. However, as Saltz's group has shown in a set of pioneering projects [HPF:94a], many important PDE methods require nontrivial HPF language extensions, as well as sophisticated runtime support, such as the PARTI [Saltz:91b] and CHAOS systems [Edjali:95a], [Hwang:94a], [Ponnusamy:93c;94b]. The needed language support can be thought of as expressing the problem architecture (computational graph as in Figure 3(a), which is only implicitly defined by the standard (Fortran) code. Correctly written, this vanilla Fortran implies all needed information for efficient parallelism. However, this information is realized in terms of the values of pointers and cannot be recognized at compile time for either static or compiler generated dynamic runtime parallelism. This fundamental problem is of course why Fortran is a more successful parallel language than C as latter naturally uses pointer constructs that obscure the problem architecture even more. The runtime support for PDE's must cope with irregular and hierarchical meshes and provide the dynamic alignment decomposition and communications optimization that HPF1 provides for arrays.

Now, let us consider manufacturing as a major industrial application of PDE's. Here, HPCC offers an important opportunity to build futuristic manufacturing systems allowing customizable products with integrated design (conceptual and detailed), manufacturing process, sales and support. This scenario—sometimes called agile manufacturing—implies other major thrusts including concurrent engineering and multidisciplinary analysis and design. Here, we can use an example where NPAC is working under NASA sponsorship with the MADIC (Multidisciplinary Analysis and Design Industry Consortium) collaboration involving Rockwell, Northrop Grumman Vought, McDonnell Douglas, General Electric, General Motors and Georgia Tech. We are in particular involved in establishing for this NASA project, the NII requirements for a future concurrent engineering concept called ASOP (Affordable Systems Optimization Process). Aircraft are now built by multi-company collaborations with international scope which virtual corporation needs collaborative and networked software engineering and workflow support. Configuration management is a critical need. ASOP links a range of disciplines (from manufacturing process simulation, electromagnetic signature, aeronautics and propulsion computation linked to CAD databases and virtual reality visualization) using MDO—multidisciplinary optimization—techniques. The integration of conceptual (initial) and detailed design with the manufacturing and life cycle support phases naturally requires the integration of information and computing in the support system. WebWork [Fox:95a] has been designed for this. Further, we see this problem is a heterogeneous metaproblem with perhaps up to 10,000 (Fortran) programs linked together in the full optimization process. This is basically an embarrassingly parallel meta-architecture with only a few of the programs linked together at each stage. The program complexity varies from a full PDE simulation to an expert system to optimize location of an inspection port to minimize support costs. So efficient parallel solution of PDEs is part, but not all of the support needed for manufacturing. HPCC will only have major impact on manufacturing when it can support such heterogeneous metaproblems, including large scale database integration.

5.3 Monte Carlo Methods (Applications 4, 6, 7, 9, 11)

We have already mentioned in Section 3, Quantum Chromodynamics Simulations as a classic example of large scale Monte Carlo simulations suitable for parallel machines. As described in Chapter 4 of [Fox:94a], this application is straightforward to parallelize and very suitable for HPF as the basic data structure is an array. The array represents a regular structure in space time as seen in the simplest finite different problems. The Monte Carlo occurs at each grid point and is typically local (nearest neighbor) so that the overall problem architecture is just like that of a PDE. This specific computation is from an academic field, but is typical in structure of some practical material science problems. Further, just as many PDEs have irregular data structures, the same is true of many Monte Carlos. QCD is typical of simulations of crystalline substances with a regular array of atoms. However, many substances—in particular gases and liquids—have irregular particle distributions and many of issues discussed briefly in Section 5.2 for finite element methods. As described in Chapter 14 of [Fox:94a], there is a subtle point that distinguishes Monte Carlo and PDE algorithms as one cannot simultaneously update in Monte Carlo, sites with overlapping neighbors. This complicates the loosely synchronous structure and can make problem architecture look like that of a synchronous event driven simulations—here events are individual Monte Carlo updates. "Detailed balance" requires that such events be sequentially (if arbitrarily) ordered. In the example of [Johnson:86c] described in [Fox:94a], a clever implementation gave good parallel performance.

Monte Carlo methods can be implemented quite differently—above we decomposed the underlying physical data. One can also use "data parallelism" on the random number set used in the simulation. This is not possible for QCD for two reasons. Firstly, the physical dataset is so large it would not fit in the memory of a single node—we need to decompose the physical dataset just to get enough total memory. More importantly, one can run QCD with several different starting points. However, all Monte Carlos—using importance sampling of the Metropolis type employed by QCD—have a "thermalization stage" where one must get "into equilibrium" before the sampling is useful. Thermalization is very time consuming for QCD and makes multiple starting points of limited value. However, there are many cases where this is not true, and as show in Chapter 7 of [Fox:94a], one can get an embarrassing parallel architecture for Monte Carlo problems. Each instance of the problem has the full physical dataset, but can be run independently with different random number streams. Like many such embarrassingly parallel cases, the different instances do need to accumulate their data—in this case, Monte Carlo averages. One important examples of this class of application is Quantum Monte Carlo used in many ab initio chemistry problems [Kalos:85a].

Yet, a different set of issues comes with a class of Monte Carlo problems which are termed "clustered." In most physical system Monte Carlos, one updates a single "entity" (grid point or particle) at a time. This is very ineffective when there is substantial correlation between neighboring points. A simple example comes from ferromagnetic materials where domains form where spins are locked in the same direction over large regions., Clustering algorithms are quite hard to find for sequential systems, and their parallelization is challenging and very different from the earlier examples. As discussed in Section 12.6 of [Fox:94a], the algorithm is similar to that used in region finding in image processing [Copty:93a;94a;95a]. Parallelism requires consideration (as in domain decomposition for PDEs) of inter and intra region issues.

5.4 Summary

Each of three case studies illustrates how different applications and different numerical approaches to a given problem, lead to very different problem architectures and correspondingly the needed software support. Although our discussion is not complete, we do think that it is quite typical, and that a similar situation is seen in the other applications of Table 4, and summarized in the last two columns.

References

[Angus:90a] Angus, I. G., Fox, G. C., Kim, J. S., and Walker, D. W. *Solving Problems on Concurrent Processors: Software for Concurrent Processors*, volume 2. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1990.

[Barnes:86a] Barnes, J., and Hut, P. "A hierarchical $O(N \log N)$ force calculation algorithm," *Nature*, 324:446–449, 1986.

[Berger:84a] Berger, M. J., and Oliger, J. "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of Computational Physics*, 53:484, 1984.

[Birman:87a] Birman, K. P., and Joseph, T. "Reliable communication in the presence of failures," *ACM Trans. on Computer Systems*, 5:47–76, February 1987.

[Birman:87b] Birman, K. P., and Joseph, T. "Exploiting virtual synchrony in distributed systems," in *Proceedings of the Eleventh Symposium on Operating Systems Principles*, pages 123–138. ACM, November 1987.

[Birman:91a] Birman, K., and Cooper, R. "The ISIS project: Real experience with a fault tolerant programming system," *Operating Systems Review*, pages 103–107, April 1991. ACM/SIGOPS European Workshop on Fault-Tolerance Techniques in Operating Systems, held in Bologna, Italy (1990).

[Bodin:91a] Bodin, F., Beckman, P., Gannon, D., Narayana, S., and Shelby, Y. "Distributed pC++: Basic ideas for an object parallel language," in *Proceedings of Supercomputing '91*, pages 273–282. (IEEE) Computer Society and (ACM) (SIGARCH), November 1991.

[Bogucz:94a] Bogucz, E., Fox, G., Haupt, T., Hawick, K., and Ranka, S. "Preliminary evaluation of high-performance Fortran as a language for computational fluid dynamics." Technical Report SCCS-625, Syracuse University, NPAC, Syracuse, NY, June 1994. Proc. AIAA 25th Computational Fluid Dynamics Conference, Colorado Springs, AIAA 94-2262.

[Bozkus:93a] Bozkus, Z., Choudhary, A., Fox, G. C., Haupt, T., and Ranka, S. "Fortran 90D/HPF compiler for distributed memory MIMD computers: Design, implementation, and performance results." Technical Report SCCS-498, Syracuse University, NPAC, Syracuse, NY, 1993. Proceedings of Supercomputing '93, Portland, OR, November 1993.

[Chandy:90a] Chandy, K., and Taylor, S. "A primer for program composition notation." Technical Report CRPC-TR90056, California Institute of Technology, Pasadena, CA, June 1990.

[Chandy:93a] Chandy, K. M., and Kesselman, C. *CC++: A Declarative Concurrent Object-Oriented Programming Notation.* Research Directions in Concurrent Object-Oriented Programming. MIT Press, 1993.

[Chapman:92b] Chapman, B., Mehrotra, P., and Zima, H. "Programming in Vienna Fortran," *Scientific Programming*, 1(1):31–50, 1992.

[Chapman:94b] Chapman, B., Mehrotra, P., and Zima, H. "Extending HPF for advanced data-parallel applications," *IEEE Parallel and Distributed Technology*, 2(3):15–27, 1994.

[Chen:88b] Chen, M., Li, J., and Choo, Y. "Compiling parallel programs by optimizing performance," *Journal of Supercomputing*, 2:171–207, 1988.

[Cheng:92a] Cheng, G., Faigle, C., Fox, G. C., Furmanski, W., Li, B., and Mills, K. "Exploring AVS for HPDC software integration: Case studies towards parallel support for GIS." Technical Report SCCS-473, Syracuse University, NPAC, Syracuse, NY, March 1992. Paper presented at the 2nd Annual International AVS Conference *The Magic of Science: AVS '93*, Lake Buena Vista, Florida, May 24–26, 1993.

[Cheng:93a] Cheng, G., Lu, Y., Fox, G. C., Mills, K., and Haupt, T. "An interactive remote visualization environment for an electromagnetic scattering simulation on a high performance computing system." Technical Report SCCS-467, Syracuse University, NPAC, Syracuse, NY, March 1993. Proceedings of Supercomputing '93, Portland, Oregon, November 15–19.

[Cheng:94a] Cheng, G., Fox, G., Mills, K., and Podgorny, M. "Developing interactive PVM-based parallel programs on distributed computing systems within AVS framework." Technical Report SCCS-611, Syracuse University, NPAC, Syracuse, NY, January 1994. Proceedings of the 3rd Annual International AVS Conference, JOIN THE REVOLUTION: AVS'94, Boston, MA, May 2–4.

[Cheng:94c] Cheng, G., Hawick, K., Mortensen, G., and Fox, G. "Distributed computational electromagnetics systems." Technical Report SCCS-635, Syracuse University, NPAC, Syracuse, NY, August 1994. Proceedings of the 7th SIAM Conference on Parallel Processing for Scientific Computing, February 15–17, 1995.

[Cheng:94d] Cheng, G., Fox, G., and Mills, K. "Integrating multiple programming paradigms on Connection Machine CM5 in a dataflow-based software environment (draft)." Technical Report SCCS-548, Syracuse University, NPAC, Syracuse, NY, October 1994.

[Cheng:94e] Cheng, G., Fox, G. C., and Hawick, K. *A Scalable Parallel Paradigm for Effectively-Dense Matrix Formulated Applications*, volume 797 of *Lecture Notes in Computer Science*, pages 202–210. Springer-Verlag, April 1994. Proceedings of the European Conference and Exhibition on High-Performance Computing and Networking (HPCN Europe) 1994, Munich, Germany; Syracuse University Technical Report SCCS-580.

[Choi:92c] Choi, J., Dongarra, J. J., Pozo, R., and Walker, D. W. "Scalapack: A scalable linear algebra library for distributed memory concurrent computers," in *Proceedings of the Fourth Symposium on the Frontiers of Massively Parallel Computation*, pages 120–127. IEEE Computer Society Press, 1992.

[Choudhary:92d] Choudhary, A., Fox, G., Hiranandani, S., Kennedy, K., Koelbel, C., Ranka, S., and Saltz, J. "A classification of irregular loosely synchronous problems and their support in scalable parallel software systems," in *DARPA Software Technology Conference 1992 Proceedings*, pages 138–149, April 1992. Syracuse Technical Report SCCS-255.

[Choudhary:92e] Choudhary, A., Fox, G., Ranka, S., Hiranandani, S., Kennedy, K., Koelbel, C., and Saltz, J. "Software support for irregular and loosely synchronous problems," *Computing Systems in Engineering*, 3(1–4):43–52, 1992. CSE-MS 118, CRPC-TR92258.

[Choudhary:92g] Choudhary, A., Fox, G., Haupt, T., and Ranka, S. "Which applications can use high performance Fortran and FortranD—industry standard data parallel languages?," in *Proceedings of Fifth Australian Supercomputing Conference*, December 1992. CRPC-TR92264.

[Choudhary:94c] Choudhary, A., Dincer, K., Fox, G., and Hawick, K. "Conjugate gradient algorithms implemented in high performance Fortran." Technical Report SCCS-639, Syracuse University, NPAC, Syracuse, NY, October 1994.

[Copty:93a] Copty, N., Ranka, S., Fox, G., and Shankar, R. "Solving the region growing problem on the Connection Machine," in *Proceedings of the 22nd International Conference on Parallel Processing*, volume 3, pages 102–105, 1993. Syracuse University, NPAC Technical Report SCCS-397b.

[Copty:94a] Copty, N., ranka, S., Fox, G., and Shankar, R. "A data parallel algorithm for solving the region growing problem on the Connection Machine," *Journal of Parallel and Distributed Computing*, 21(1), 1994. Syracuse University, NPAC Technical Report SCCS-596.

[Copty:95a] Copty, N. *Language and Runtime Support for the Execution of Clustering Applications on Distributed Memory Machines.* PhD thesis, Syracuse University, 1995.

[Denning:90a] Denning, P. J., and Tichy, W. F. "Highly parallel computation," *Science*, 250:1217–1222, 1990.

[Dincer:95b] Dincer, K., Hawick, K., Choudhary, A., and Fox, G. "High performance Fortran and possible extensions to support conjugate gradient algorithms." Technical Report SCCS-703, Syracuse University, NPAC, Syracuse, NY, March 1995. To appear in Proc. Supercomputing '95, December 1995.

[Edelsohn:91b] Edelsohn, D., and Fox, G. C. "Hierarchical tree-structures as adaptive meshes." Technical Report SCCS-193, Syracuse University, NPAC, Syracuse, NY, November 1991. Published in the International Journal of Modern Physics C, Vol. 4, No. 5, pp. 909–917; CRPC-TR91186.

[Edjali:95a] Edjali, G., Agrawal, G., Sussman, A., and Saltz, J. "Data parallel programming in an adaptive environment," in *IPPS '95*, pages 827–832, 1995. An extended version also available as University of Maryland Technical Report CS-TR-3350 and UMIACS-TR-94-109.

[Factor:90a] Factor, M. "The process Trellis architecture for real_time monitors," in *Proceedings of the Second ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPOP)*, March 1990. Held in Seattle, Washington.

[Factor:90b] Factor, M., and Gelertner, D. G. "Experience with Trellis architecture." Technical Report YALEU/DCS/RR-818, Yale University, New Haven, CT, August 1990.

[Felten:88i] Felten, E. W., and Otto, S. W. "A highly parallel chess program," in *Proceedings of International Conference on Fifth Generation Computer Systems 1988*, pages 1001–1009. ICOT, November 1988. Tokyo, Japan, November 28 – December 2. Caltech Report C3P-579c.

[Foster:95a] Foster, I. *Designing and Building Parallel Programs.* Addison-Wesley, 1995. http://www.mcs.acl.gov/dbpp/.

[Fox:87d] Fox, G. C. "Questions and unexpected answers in concurrent computation," in J. J. Dongarra, editor, *Experimental Parallel Computing Architectures*, pages 97–121. Elsevier Science Publishers B.V., North-Holland, Amsterdam, 1987. Caltech Report C3P-288.

[Fox:88a] Fox, G. C., Johnson, M. A., Lyzenga, G. A., Otto, S. W., Salmon, J. K., and Walker, D. W. *Solving Problems on Concurrent Processors*, volume 1. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1988.

[Fox:88b] Fox, G. C. "What have we learnt from using real parallel machines to solve real problems?," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 897–955. ACM Press, New York, January 1988. Caltech Report C3P-522.

[Fox:88oo] Fox, G. C. "The hypercube and the Caltech Concurrent Computation Program: A microcosm of parallel computing," in B. J. Alder, editor, *Special Purpose Computers*, pages 1–40. Academic Press, Inc., Boston, 1988. Caltech Report C3P-422.

[Fox:88tt] Fox, G. C., and Furmanski, W. "The physical structure of concurrent problems and concurrent computers," *Phil. Trans. R. Soc. Lond. A*, 326:411–444, 1988. Caltech Report C3P-493.

[Fox:89n] Fox, G. C. "Parallel computing comes of age: Supercomputer level parallel computations at Caltech," *Concurrency: Practice and Experience*, 1(1):63–103, September 1989. Caltech Report C3P-795.

[Fox:90p] Fox, G. C. "Hardware and software architectures for irregular problem architectures," in P. Mehrotra, J. Saltz, and R. Voigt, editors, *Unstructured Scientific Computation on Scalable Microprocessors*, pages 125–160. The MIT Press, Cambridge, MA, 1992. Scientific and Engineering Computation Series. Held by ICASE in Nags Head, North Carolina. SCCS-111; CRPC-TR91164.

[Fox:91e] Fox, G. C., Hiranandani, S., Kennedy, K., Koelbel, C., Kremer, U., Tseng, C.-W., and Wu, M.-Y. "Fortran D language specification." Technical Report SCCS-42c, Syracuse University, Syracuse, NY, April 1991. Rice Center for Research in Parallel Computation; CRPC-TR90079.

[Fox:91g] Fox, G. C. "The architecture of problems and portable parallel software systems." Technical Report SCCS-134, Syracuse University, NPAC, Syracuse, NY, July 1991. Revised SCCS-78b.

[Fox:91m] Fox, G. C. "Lessons from massively parallel architectures on message passing computers," in *The 37th Annual IEEE International Computer Conference, COMPCON '92*. IEEE Computer Society Press, Los Alamitos, CA, December 1991. Held February 24–28, 1992 San Francisco, California. CRPC-TR91192; SCCS-214.

[Fox:92e] Fox, G. C. "Parallel computing in industry—an initial survey," in *Proceedings of Fifth Australian Supercomputing Conference (supplement)*, pages 1–10. Communications Services, Melbourne, December 1992. Held at World Congress Centre, Melbourne, Australia. Syracuse University Technical Report SCCS-302b. CRPC-TR92219.

[Fox:93c] Fox, G., Bogucz, E., Jones, D., Mills, K., and Podgorny, M. "InfoMall: a scalable organization for the development of HPCC software and systems." Technical Report SCCS-531, Syracuse University, NPAC, Syracuse, NY, September 1993. Unpublished.

[Fox:94a] Fox, G. C., Messina, P. C., and Williams, R. D., editors. *Parallel Computing Works!* Morgan Kaufmann Publishers, San Francisco, CA, 1994. http://www.infomall.org/npac/pcw/.

[Fox:94b] Fox, G., and Mills, K. *Information Processing and Opportunities for HPCN Use in Industry*, pages 1–14. Number 796 in Lecture Notes in Computer Science. Springer-Verlag, New York, April 1994. Proceedings of HPCN Europe 1994, "High Performance Computing and Networking.

[Fox:94c] Fox, G., and Mills, K. "Information processing and HPCC applications in industry," in *Proceedings of Annual 1994 Dual-use Conference*, Utica, NY, May 1994. IEEE Mohawk Valley.

[Fox:94f] Fox, G., Furmanski, W., Hawick, K., and Leskiw, D. "Exploration of the InfoMall concept." Technical Report SCCS-634, Syracuse University, NPAC, Syracuse, NY, August 1994.

[Fox:94g] Fox, G., and Hawick, K. "An applications perspective on high performance Fortran." Technical Report SCCS-641, Syracuse University, NPAC, Syracuse, NY, November 1994.

[Fox:94h] Fox, G., Hawick, K., Podgorny, M., and Mills, K. *The Electronic InfoMall—HPCN Enabling Industry and Commerce*, volume 919 of *Lecture Notes in Computer Science*, pages 360–365. Springer-Verlag, November 1994. Syracuse University Technical Report SCCS-665.

[Fox:94i] Fox, G. C. "Involvement of industry in the national high performance computing and communication enterprise." Technical Report SCCS-716, Syracuse University, NPAC, Syracuse, NY, May 1994. *Developing a Computer Science Agenda for High Performance Computing*, edited by U. Vishkin, ACM Press.

[Fox:95a] Fox, G. C., Furmanski, W., Chen, M., Rebbi, C., and Cowie, J. H. "WebWork: integrated programming environment tools for national and grand challenges." Technical Report SCCS-715, Syracuse University, NPAC, Syracuse, NY, June 1995. Joint Boston-CSC-NPAC Project Plan to Develop WebWork.

[Fox:95b] Fox, G. C., Furmanski, W., Hawick, K., and Leskiw, D. "Exploration of the InfoMall concept—building on the electronic InfoMall." Technical Report SCCS-711, Syracuse University, NPAC, Syracuse, NY, May 1995.

[Gelertner:89a] Gelertner, D. *Multiple Tuple Spaces in Linda*, volume 366 of *Lecture Notes in Computer Science, Proceedings of Parallel Architectures and Languages, Europe, Volume 2*, pages 20–27. Springer-Verlag, Berlin/New York, June 1989.

[Goil:94a] Goil, S. "Primitives for problems using hierarchical algorithms on distributed memory machines." Technical Report SCCS-687, Syracuse University, NPAC, Syracuse, NY, December 1994. Proceedings of the First International Workshop in Parallel Processing, Bangalore, India.

[Goil:95a] Goil, S., and Ranka, S. "Software support for parallelization of hierarchically structured applications on distributed memory machines." Technical Report SCCS-688, Syracuse University, NPAC, Syracuse, NY, February 1995.

[Gottschalk:90b] Gottschalk, T. D. "Concurrent multi-target tracking," in D. W. Walker and Q. F. Stout, editors, *The Fifth Distributed Memory Computing Conference, Volume I*, pages 85–88. IEEE Computer Society Press, Los Alamitos, CA, 1990. Held April 9–12, Charleston, SC. Caltech Report C3P-908.

[Greengard:87b] Greengard, L., and Rokhlin, V. "A fast algorithm for particle simulations," *Journal of Computational Physics*, 73:325–348, 1987. Yale University Computer Science Research Report YALEU/DCS/RR-459.

[Grimshaw:93b] Grimshaw, A. S. "Easy to use object-oriented parallel programming with Mentat," *IEEE Computer*, pages 39–51, May 1993.

[Harrington:61a] Harrington, R. F. *Time-Harmonic Electromagnetic Fields.* McGraw Hill Book Company, New York, 1961.

[Harrington:67a] Harrington, R. F. "Matrix methods for field problems," in *Proc. IEEE*, volume 55(2), pages 136–149, February 1967.

[Harrington:68a] Harrington, R. F. *Field Computation by Moment Methods.* The Macmillan Company, New York, 1968. Reprinted by Krieger Publishing Co., Malabar, FT. (1982).

[Hatcher:91a] Hatcher, P. J., and Quinn, M. J. *Data-Parallel Programming on MIMD Computers.* MIT Press, Cambridge, Massachusetts, 1991.

[Hatcher:91b] Hatcher, P., Lapadula, A., Jones, R., Quinn, M., and Anderson, R. "A production-quality C* compiler for hypercube multicomputers," in *Third ACM SIGPLAN Symposium on PPOPP*, volume 26, pages 73–82, July 1991.

[Haupt:95a] Haupt, T. http://www.npac.syr.edu/projects/bbh describes use of High Performance Fortran for solving Einstein's equations for the collision of two black holes.

[Hawick:95a] Hawick, K., Dincer, K., Robinson, G., and Fox, G. "Conjugate gradient algorithms in Fortran 90 and high performance Fortran." Technical Report SCCS-691, Syracuse University, NPAC, Syracuse, NY, February 1995.

[Hawick:95b] Hawick, K., and Fox, G. *Exploiting High Performance Fortran for Computational Fluid Dynamics*, volume 919 of *Lecture Notes in Computer Science*, pages 413–419. Springer-Verlag, May 1995. International Conference on High Performance Computing and Networking, HPCN Europe 1995, Milan; Syracuse University Technical Report SCCS-661.

[Hawick:95c] Hawick, K., Bogucz, E. A., Degani, A. T., Fox, G. C., and Robinson, G. "Computational fluid dynamics algorithms in high performance Fortran," in *Proc. AIAA 26th Computational Fluid Dynamics Conference*, June 1995.

[Hillis:87a] Hillis, W. D. "The Connection Machine," *Scientific American*, 256:108–115, June 1987.

[Hiranandani:92c] Hiranandani, S., Kennedy, K., and Tseng, C. "Compiling Fortran D for MIMD distributed-memory machines," *Comm. ACM*, 35(8):66–80, 1992.

[HPF:93a] High Performance Fortran Forum. "High performance Fortran language specification." Technical Report CRPC-TR92225, Center for Research on Parallel Computation, Rice University, Houston, Texas, 1993.

[HPF:94a] *HPF-2 Scope of Activities and Motivating Applications*. November 1994. ftp://hpsl.cs.umd.edu/pub/hpfbench/index.html.

[HPFapp:95a] http//www.npac.syr.edu/hpfa/algorithms.html. A collection of applications designed to test HPF, which is online at NPAC.

[HPFCSep:95a] "Fortran 90 and computational science." Online Computational Science Educational Project; http://csep1.phys.ornl/csep.html.

[HPFF:95a] High Performance Fortran Forum. http://www.erc.msstate.edu/hpff/home.html.

[Hwang:94a] Hwang, Y.-S., Moon, B., Sharma, S., Das, R., and Saltz, J. "Runtime support to parallelize adaptive irregular programs," in *Proceedings of the Workshop on Environments and Tools for Parallel Scientific Computing*, 1994.

[Infourl:95a] The InfoMall Home Page http://www.infomall.org.

[Johnson:86c] Johnson, M. A. "The specification of CrOS III." Technical Report C3P-253, California Institute of Technology, Pasadena, CA, February 1986.

[Jordon:69a] Jordon, E. C., and Balmain. *Electromagnetic Waves and Radiating Systems.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969. Second Edition.

[Joubert:95a] Joubert, A. "Financial applications and HPF." Technical report, The London Parallel Applications Centre, London, UK, 1995.

[Kalos:85a] Kalos, M. *The Basics of Monte Carlo Methods.* John Wiley and Sons, 1985.

[Koelbel:94a] Koelbel, C., Loveman, D., Schreiber, R., Steele, G., and Zosel, M. *The High Performance Fortran Handbook.* MIT Press, 1994.

[Lemke:92a] Lemke, M., and Quinland, D. "P++, a parallel C++ array class library for architecture-independent development of structured grid applications," in *Proc. Workshop on Languages, Compilers, and Runtime Environments for Distributed Memory Computers.* ACM, 1992.

[McBryan:94a] McBryan, O. "An overview of message passing environments," *Parallel Computing*, 20(4):417–444, 1994.

[Meier:89a] Meier, D. L., Cloud, K. C., Horvath, J. C., Allan, L. D., Hammond, W. H., and Maxfield, H. A. "A general framework for complex time-driven simulations on hypercubes." Technical Report C3P-761, California Institute of Technology, Pasadena, CA, March 1989. Paper presented at the Fourth Conference on Hypercubes, Concurrent Computers and Applications.

[Mills:92a] Mills, K., Vinson, M., Cheng, G., and Thomas, F. "A large scale comparison of option pricing models with historical market data," in *Proceedings of The 4th Symposium on the Frontiers of Massively Parallel Computing.* IEEE Computer Society Press, October 1992. Held in McLean, VA. SCCS-260.

[Mills:92b] Mills, K., Cheng, G., Vinson, M., Ranka, S., and Fox, G. "Software issues and performance of a parallel model for stock option pricing," in *Proceedings of the Fifth Australian Supercomputing Conference*, pages 125–134, December 1992. Held in Melbourne, Australia. SCCS-273b.

[Mills:93a] Mills, K., and Fox, G. C. "HPCC applications development and technology transfer to industry," in I. D. Scherson, editor, *The New Frontiers: A Workshop on Future Directions of Massively Parallel Processing*, pages 58–65, Los Alamitos, CA, October 1993. IEEE Computer Society Press.

[Mills:94a] Mills, K., and Fox, G. "InfoMall: an innovative strategy for high-performance computing and communications applications development," *Internet Research*, 4:31–45, 1994.

[Mills:95a] Mills, K., Fox, G., Coddington, P., Mihalas, B., Podgorny, M., Shelly, B., and Bossert, S., "The living textbook and the K–12 classroom of the future." http://www.npac.syr.edu/projects/ltb/SC95/index.html.

[MOPAC:95a]
See electronic description of NPAC's activities in this area at http://www.npac.syr.edu/projects/mopac/mopac.html.

[Muller:95a] Müller, A., and Rühl, R. "Extending high performance Fortran for the support of unstructured computations," in *International Conference on Supercomputing*, July 1995. Barcelona, Spain.

[Nicplocha:94a] Nicplocha, J., Harrison, R. J., and Littlefield, R. J. "Global Arrays: a portable 'shared-memory' programming model for distributed memory computers," in *Supercomputing '94*, 1994. Pacific Northwest Laboratory, http://www.emsl.pnl.gov.2080.

[Parasoft:88a] ParaSoft. *EXPRESS: A Communication Environment for Parallel Computers.* ParaSoft, Inc., Pasadena, CA, 1988.

[Peta:94a] http://www.npac.syr.edu/roadmap/petaapps.html is HTML version of application table. The full published proceedings is T. Sterling, P. Messina, and P. H. Smith, *Enabling Technologies for Petaflops Computing*, MIT press, 1995.

[Ponnusamy:93c] Ponnusamy, R., Saltz, J., Choudhary, A., Hwang, Y.-S., and Fox, G. "Runtime support and compilation methods for user-specified data distributions." Technical Report CS-TR-3194 and UMIACS-TR-93-135, University of Maryland, Department of Computer Science, 1993. To appear in IEEE Transaction on Parallel and Distributed Memory Systems.

[Ponnusamy:94b] Ponnusamy, R., Hwang, Y.-S., Saltz, J., Choudhary, A., and Fox, G. "Supporting irregular distributions in FORTRAN 90D/HPF compilers." Technical Report CR-TR-3268 and UMIACS-TR-94-57, University of Maryland, Department of Computer Science, 1994. Also available in IEEE Parallel and Distributed Technology, Spring 1995.

[Ranka:92a] Ranka, S., Fox, G. C., Saltz, J., and Das, R. "Parallelization of CHARMM molecular dynamics code on multicomputers." Technical Report SCCS-236, Syracuse University, NPAC, Syracuse, NY, January 1992.

[Robinson:95a] Robinson, G., Hawick, K. A., and Fox, G. C. "Fortran 90 and high performance Fortran for dense matrix-formulated applications." Technical Report SCCS-709, Syracuse University, NPAC, Syracuse, NY, May 1995.

[Salmon:90a] Salmon, J. *Parallel Hierarchical N-Body Methods.* PhD thesis, California Institute of Technology, December 1990. SCCS-52, CRPC-TR90115. Caltech Report C3P-966.

[Saltz:91b] Saltz, J., Berryman, H., and Wu, J. "Multiprocessor and runtime compilation," *Concurrency: Practice and Experience*, 3(6):573–592, December 1991. Special Issue: Practical Parallel Computing: Status and Prospects. Guest Editors: Paul Messina and Almerico Murli.

[Singh:93a] Singh, J. P. *Parallel Hierarchical N-body Methods and Their Implications for Multiprocessors.* PhD thesis, Stanford University, 1993.

[Sturler:95a] De Sturler, E., and Strumpen, V. "First experiences with high performance Fortran on the Intel Paragon." Technical Report 95-10, Interdisciplinary Project Center for Supercomputing, Swiss Federal Institute of Technology Zurich, 1995.

[Sunderam:90a] Sunderam, V. S. "PVM: a framework for parallel distributed computing," *Concurrency: Practice and Experience*, 2(4):315–340, 1990.

[Sunderam:93a] Sunderam, S. *Fast Algorithms for N-body Simulations.* PhD thesis, Cornell University, 1993.

[Warren:92b] Warren, M. S., and Salmon, J. K. "Astrophysical N-Body simulations using hierarchical tree data structures," in *Supercomputing '92.* IEEE Comp. Soc., Los Alamitos, CA, 1992.

[Warren:93a] Warren, M. S., and Salmon, J. K. "A parallel hashed oct-tree N-Body algorithm," in *Supercomputing '93.* IEEE Comp. Soc., Los Alamitos, CA, 1993.

[Wieland:89a] Wieland, F., Hawley, L., Feinberg, A., DiLoreto, M., Blume, L., Ruffles, J., Reiher, P., Beckman, B., Hontalas, P., Bellenot, S., and Jefferson, D. "The performance of a distributed combat simulation with the time warp operating system," *Concurrency: Practice and Experience*, 1(1):35–50, 1989. Caltech Report C3P-798.