An Application Perspective on High-Performance Computing and Communications

Geoffrey C. Fox
Syracuse University
Northeast Parallel Architectures Center
111 College Place
Syracuse, New York 13244
gcf@npac.syr.edu
http://www.npac.syr.edu

Abstract

We review possible and probable industrial applications of HPCC focusing on the software and hardware issues. Thirty-three separate categories are illustrated by detailed descriptions of five areas—computational chemistry; Monte Carlo methods from physics to economics; manufacturing, and computational fluid dynamics; command and control, or crisis management; and multimedia services to client computers and settop boxes. The hardware varies from tightly-coupled parallel supercomputers to heterogeneous distributed systems. The software models span HPF and data parallelism, to distributed information systems and object/dataflow parallelism on the Web.

We find that in each case, it is reasonably clear that "HPCC works in principle," and postulate that this knowledge can be used in a new generation of software infrastructure based on the WebWindows approach, and discussed in an accompanying paper.

1 Introduction

This paper could be viewed as a feasibility study for the success and viability of HPCC. We will analyze the majority of large-scale real-world problems, and find that it is relatively clear that all "can" perform well on large-scale parallel and distributed systems. We put "can" in quotes because most applications "will not," in fact, "run" well today, and it is quite hard to find the necessary return on investment for significant industrial investment in the use of HPCC systems.

We believe that this is not a failure of the concept or the work on HPCC up to now, but rather that now we know what to do, we must build a robust HPCC software and systems infrastructure. In an accompanying paper [Fox:96c], we have suggested that the key to HPCC is implementation of the essential technologies, concepts and capabilities on top of a pervasive technology, and application base. In contrast, today, HPCC is "top-down" with a set of beautiful "niche" technologies that we can not afford to build, maintain, and integrate into a robust infrastructure. Our proposed bottom-up approach is illustrated in Figure 1, which shows both "high-end" applications—typically, the so called Grand Challenges [HPCC:96a]—and the

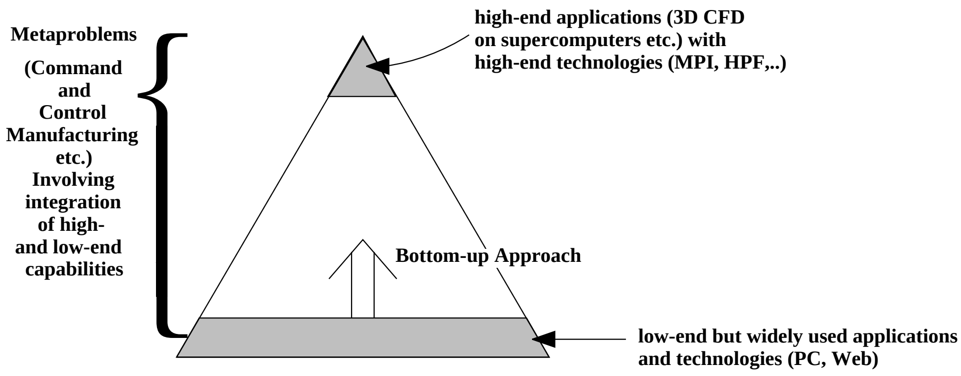


Figure 1: Integration of Grand Challenges and Pervasive Technologies

"integrated" metaproblems—or national challenges. In the analysis that follows, we group possible industrial uses of parallel computers into 33 broad classes, which include both Grand and National Challenges. In the final section, we describe five particular applications to illustrate the analysis of the relevance of HPCC in their solution. These include Grand Challenges, such as Monte Carlo simulation, computational fluid dynamics, and molecular dynamics as well as National Challenges, such as multimedia (Web) information systems, manufacturing, and command and control. The latter three areas are "metaproblems" (defined precisely in Section 2), which integrate several distributed applications including component grand challenges, such as vehicle and process simulation in manufacturing, and weather prediction in command and control.

The bottom-up approach of Figure 1 is proposed so that one can build HPCC applications and software on a commercially viable base [Fox:95k]. There are two such natural technology springboards—firstly, shared memory multiprocessors, and secondly, Web or distributed computing. The first choice leads to the interesting distributed shared memory environments, whereas the second is naturally a message passing environment. We expect that both these "viable bases" should and will be explored. One important feature of the broader distributed computing base is that it "by definition" includes "everything," and so one can build complete metaproblems in terms of a single technology framework.

From this point of view, this paper can be considered as a summary of results and requirements for "top of the pyramid" software, algorithms, and applications that need to be used in designing and building the bottom-up HPCC technology.

Section 2 reviews our general study of the structure of problems, as this is helpful in understanding the appropriate hardware and software system in each case.

In Section 3, 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 2, we point that our old classification of problems omitted *metaproblems*—problems built up from several components—each of which could have its own parallelization issues.

Note that our discussion is far more complete in the classic HPCC (parallel processing, MPP) areas, as we have far more examples, and a clearer understanding of the programming paradigms have than for the National (NII) challenges. However, we indicate where NII (Web) concepts, such as Java or VRML might fit as our environments evolve to include these rapidly developing technologies.

In Section 4, we combine the previous sections and describe informally the problem architecture, and possible software and hardware needs for a selection of "real-world" applications. We have grouped our discussion into five broad case studies; Monte Carlo methods; computational chemistry; manufacturing and computational fluid dynamics; command and control; *InfoVISiON*, or the delivery of multimedia information on the "digital" superhighway. These cover a range of software issues including, as we discussed, both the grand and national challenges, and spanning both the bottom and top of the pyramid. We conclude with a glossary of some terms used here, and in the accompanying paper [Fox:96c].

The applications, and their classification come from a survey of New York State industry [Fox:92e], [Fox:94a], [Fox:94b], [Fox:94c], [Fox:94h], [Fox:94i], [Mills:93a]. Tables 1–4 summarizes the industrial opportunities for parallel computing in the form we will use them. Some 80 different applications used in the survey have been broken up into 33 distinct areas. This is certainly somewhat arbitrary, and there are many overlaps (and omissions). 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 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", in InfoMall, 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. There is also a pointer to Section 4, if appropriate. 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.

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 MPPs and Grand Challenges 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. World Wide Web, and online services are "prototype"
  • 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 HPC++, 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 Java, 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, but harder to express and implement
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 or object parallel
Embarrassingly parallel: Essentially independent execution of disconnected components (can involve reductions, e.g., global sums to accumulate information).	Possible in most software
Metaproblems Asynchronous collection of (loosely) synchronous components where these programs themselves can be parallelized	Coarse grain parallelism—each component data parallel different components typically integrated by object, dataflow, or task parallel systems.

Table 4: Industrial HPCC Applications 1 to 5: SIMULATION

Item	Application Area and Examples	Problem Comments	Machine and Software
1 Sec. 4.3	**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 Sec. 4.3	**Structural Dynamics**	• PDE, FEM • Dominated by Vendor Codes such as NASTRAN	• MIMD as complex geometry • HPF(+)
3 Sec. 4.2, 4.3	**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 Sec. 4.1, 4.3	**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 Sec. 4.1	**Environmental Phenomenology** —Complex Systems, (Lead Concentration in blood, acid rain in the Adirondacks)	• Empirical Models • Monte Carlo and Histograms	• Some SIMD but • MIMD more natural • HPF
7 Sec. 4.1, 4.2	**Basic Chemistry** • Chemical Potentials • Elemental Reaction Dynamics	• Calculate Matrix Elements • Matrix Eigenvalue determination, Inversion, Multiplication	• Probably MIMD with perhaps SIMD possible • HPF
8 Sec. 4.2	**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 or data parallel C++ for fast multipole
9 Sec. 4.1	**Economic Modelling** • Real Time Optimization • Mortgaged backed Securities • Option Pricing	• Single financial instrument by Monte Carlo	SIMD, HPF [Makivic:95a]
		• 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 Sec. 4.1	Particle Transport Problems	Monte Carlo Methods as in neutron transport for (nuclear) explosion simulations	MIMD HPF
12	**Graphics** (rendering) • Hollywood • Virtual Reality (VRML needs parallelism?)	• 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, DSI, 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 Sec. 4.4	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 Sec. 4.1	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 Sec. 4.1	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 (Sec. 4.5)

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 Sec. 4.5	**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 Sec. 4.5	**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 (Sec. 4.5)

Item	**Application Area**	**Comments**	**Problem Structure**	**Machine & Software**
23 **Sec. 4.5**	**Imagery on Demand** • Kodak Global Image On Demand • "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 **Sec. 4.1, 4.4, 4.5**	**Simulation on Demand** • Education, Tourism, City planning, • Defense mission planning • VRML displays in the home from weather simulations • Financial modeling on server or client to aid investment	• 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.

- Many need collaboration and "computational steering" technology corresponding to integration of computers, people, and instruments "in the loop."

- Sometimes Called System of Systems

- **25: Military and Civilian Command and Control** (C^2, C^3, C^4I ...) (Sec. 5.4)

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

- **26 to 28: Applications of InfoVision Services** (Sec. 5.5)

  • Generalize Compuserve, Prodigy, America Online, Dialog and Other Information Services, implemented with Web Server, Digital Library, Internet, and Netscape/Java/VRML/.. technology.

  • **26: Decision Support Information and Entertainment for Society**
    • Settop Box/PC based Information Systems
    • 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, Authentication, Electronic Cash, etc.
- **31: Electronic Shopping**
- **32: Agile Manufacturing—Multidisciplinary Design and Concurrent Engineering** (Sec. 4.4)
  - Combines CAD with Applications 1 to 3
  - Requires major changes to Manufacturing Infrastructure and Approach
- **33: Education** (Sec. 4.5)
  - Many commonalities with Application 25
  - InfoMall Living Schoolbook—6 Schools on ATM network linked to HPCC InfoVision Servers at NPAC [Mills:95a]

This paper is not intended to advocate a particular parallel software environment or language. Rather, we want to describe the broad capabilities of, and give examples of 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, but 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++, or even Java, environments could well be more attractive, and can be substituted in the table.

2 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

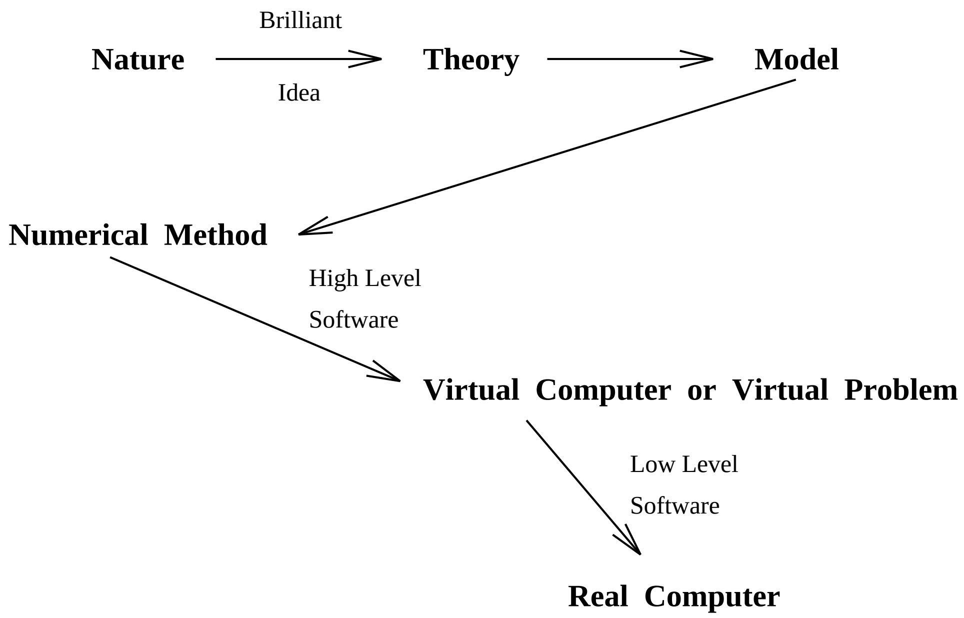


Figure 2: Computation and Simulation as a Series of Maps

the Butterfly, transputers, the SIMD Connection Machine, and DAP. We originally identified three temporal structures and one especially important (as it was so simple) spatial structure, which are the first four entries in Table 3. 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 2) 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 3, 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 4a describing the problem at a given instant of simulation time [Fox:88tt]. This is important in determining 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 3, we only single out one special spatial structure, "embarrassingly parallel," where there is little or no connection between the individual parallel program components. For embarrassingly parallel problems, illustrated in Figure 5, 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 funda-

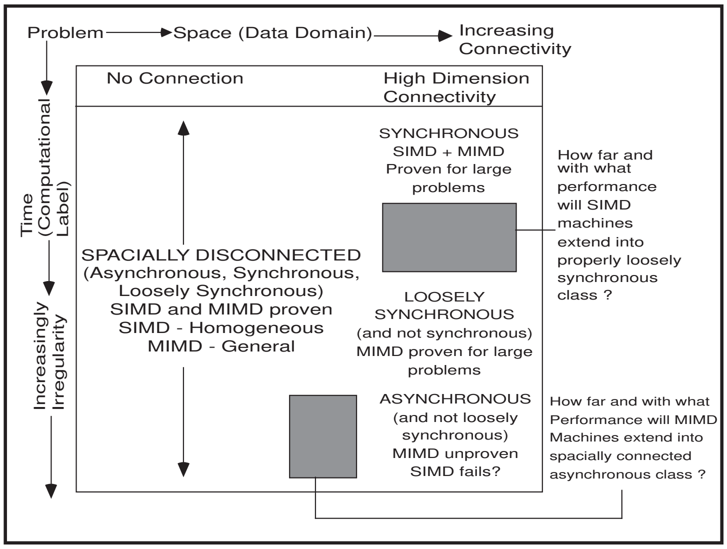


Figure 3: Issues Affecting Relation of Machine, Problem, and Software Architecture

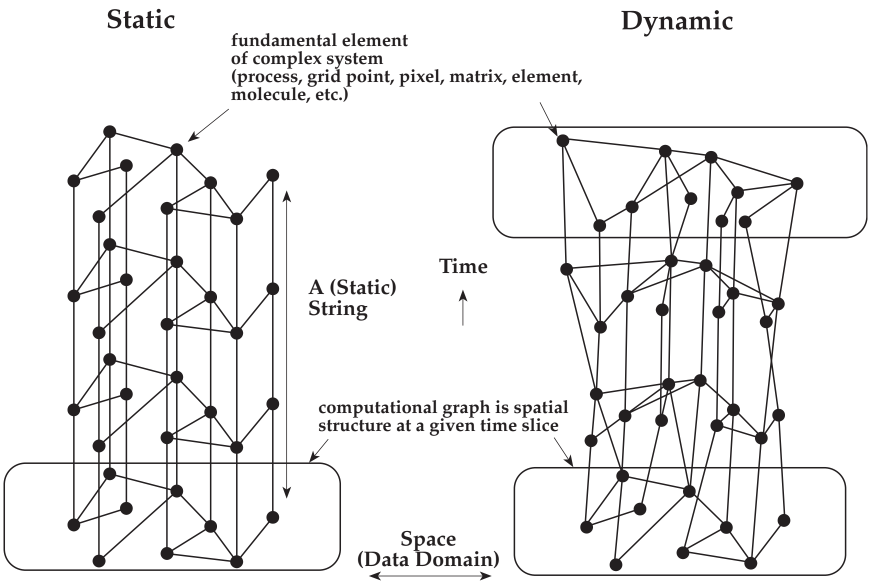
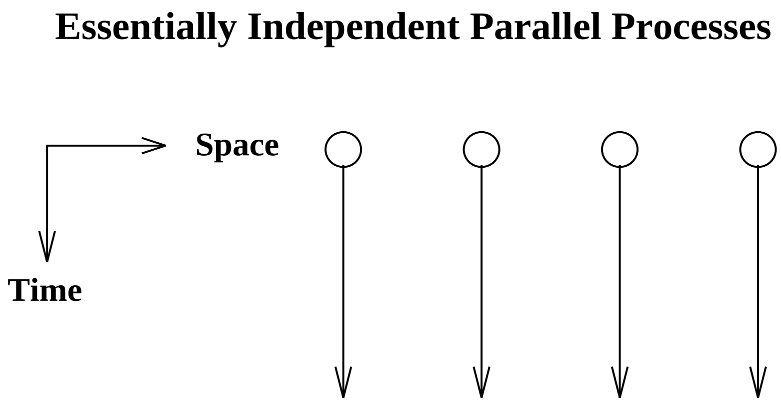


Figure 4: (a) Synchronous, Loosely Synchronous (Static), and (b) Asynchronous (Dynamic) Complex Systems with their Space-Time Structure



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

Figure 5: Embarrassingly Parallel Problem Class

mental units. This is illustrated in Figure 6 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 whenever one describes the world macroscopically in terms of the interactions between irregular inhomogeneous objects evolved in a time synchronized fashion. Typical examples, as in Figure 7, are computer or biological circuit simulations where different components or neurons are linked irregularly and modeled 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.

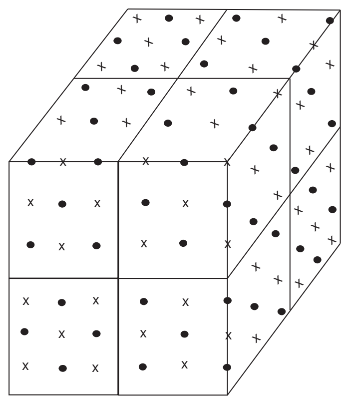
Loosely synchronous problems are spatially irregular, but temporally regular. The final *asynchronous* class is irregular in space and time, as in Figure 4b. A good example is an event driven simulation, illustrated in Figure 8, that can be used to describe the irregular circuits we discussed above, 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

EXAMPLE OF SYNCHRONOUS PROBLEMS

For example:

Microscopic Description of Fundamental Interactions
— In Particular, QCD

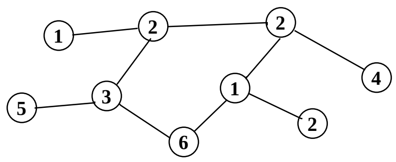


- Computational structure (almost) identical for all elements in the data domain
- Parallelize by regular partition of data domain
- Run well on SIMD machines
- Message Passing or High Performance Fortran implementation on MIMD machines

Figure 6: The Synchronous Problem Class

Loosely Synchronous Problems -
Example from Biology and General Structure

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 networks



Parallelize by irregular partition of data domain

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

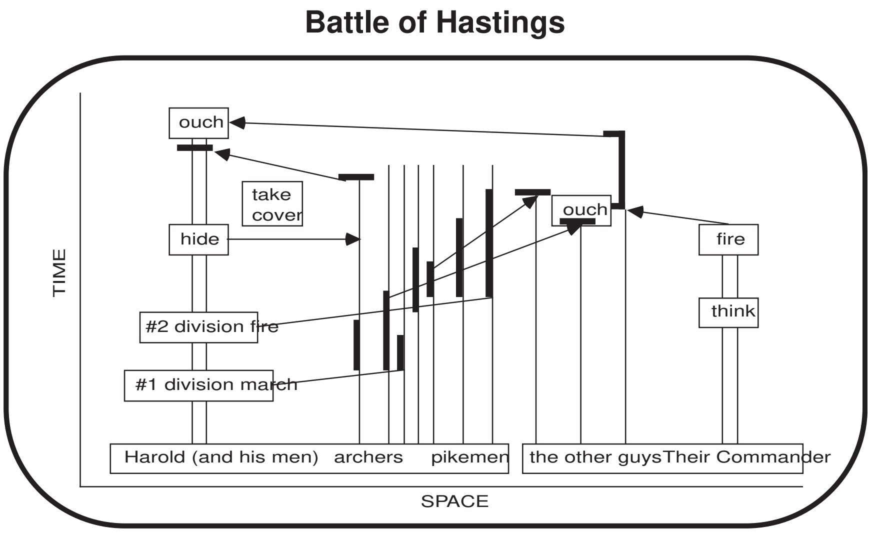
Software:
Initial version of High Performance Fortran cannot describe.

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

Figure 7: The Loosely Synchronous Problem Class

ASYNCHRONOUS PROBLEMS

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
- See SIMNET or DSI with very loosely coupled distributed event-driven simulation

Figure 8: The Asynchronous Problem Class

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 [Fox:94a], [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 9 illustrates another example where we outline an approach to designing a new airframe that involves aerodynamics, structures, radar signature, and the optimization discussed later in Section 4.3. This figure also points out the interesting analogy between heterogeneous metaproblems, and a heterogeneous computer network. Section 4.3 and Figure 12 gives a more concrete example of such a metaproblem, which as usual, involves both database (I/O), and computational subproblems.

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 asynchronous 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.

The World Wide Web is an enormous metasystem with many task parallel subproblems (Web servers handling many connections). Examples include data or task parallel "applets" handling computationally intense client computing for financial modeling, or a VRML rendering job. Data parallelism occurs in large data mining sub-applications on servers with links to Java or VRML clients, that just handle visualization and interpretation modules.

There were a few examples of metaproblems in our original survey, but a major part of Table 4, from our New York State activity, is the Information Integration classification. This class includes manufacturing and the applications 25–33, all examples of metaproblems. As stated boldly in Table 1, this class is the most important long-term area for HPCC, and is discussed in Sections 4.3 and 4.4. 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 to be very important.

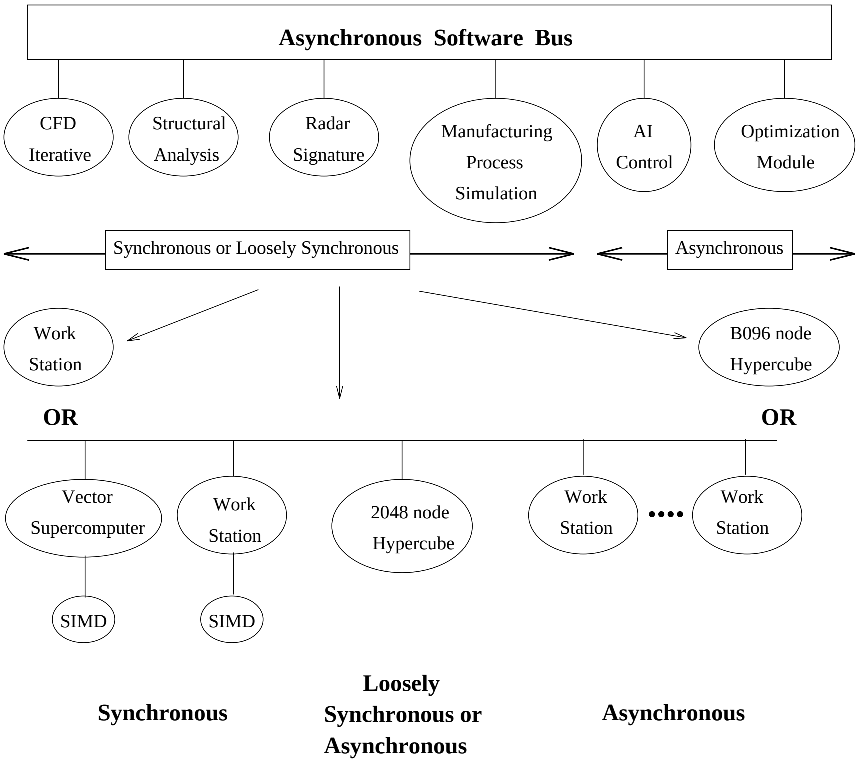


Figure 9: The Mapping of Heterogeneous Metaproblems onto Heterogeneous Metacomputer Systems

3 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.

We have described those issues at length in [Fox:90p;91g;94a], and here we present only 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 10 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 11, 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 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 HPC++ [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], including Java based distributed simulation

- **Compound Metaproblems**: AVS [Mills:92a;92b], [Cheng:93a]; PCN, Linda (or Trellis built on Linda); Webwork (CGI, Java,and Javascript Integration) [Fox:96c]; 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]

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
  • Support rollback, as in time warp
  • Java thread based systems very promising

- **INTEGRATION Runtime and Programming Environments**
  • Metaproblems
  • AVS works well, but such integration capability also can be integrated into languages such as HPC++, Fortran-M
  • Web Technology very important—JavaScript integrates frames in a Netscape Metawindow

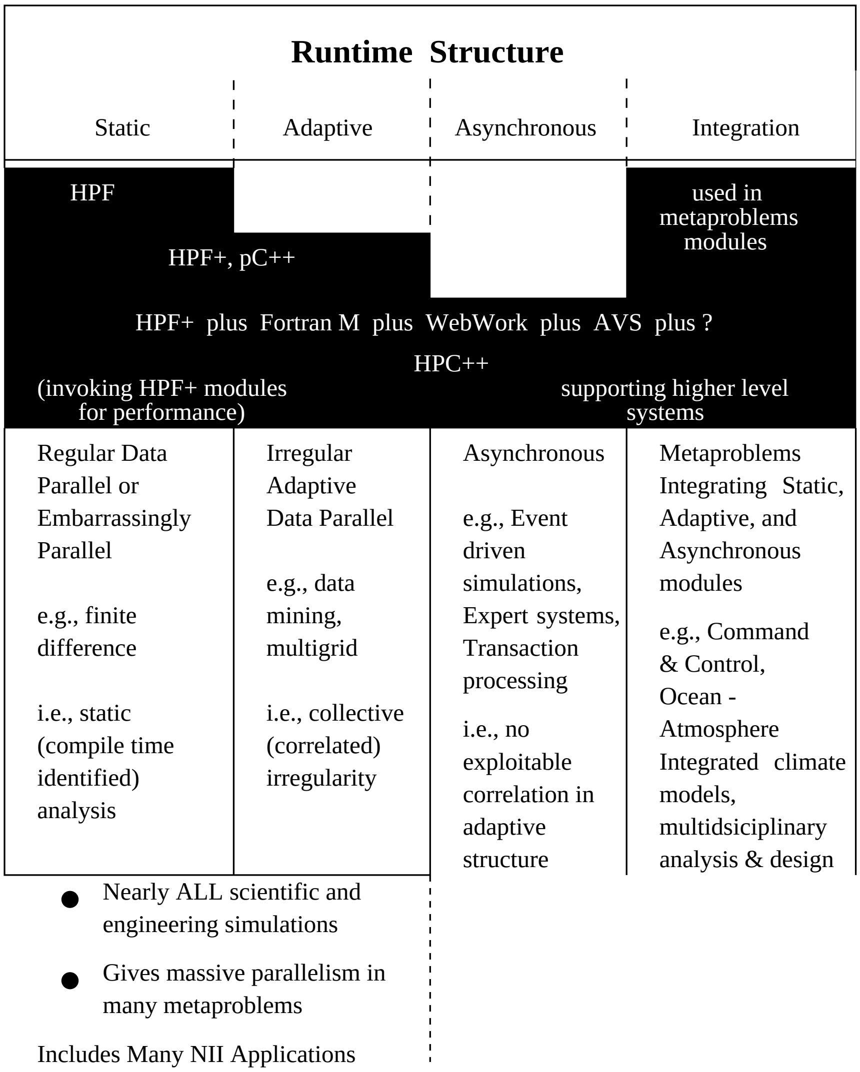


Figure 10: General Applicability of HPF, HPF+, HPC++ Classified by Problem Architecture and type of Runtime Support needed

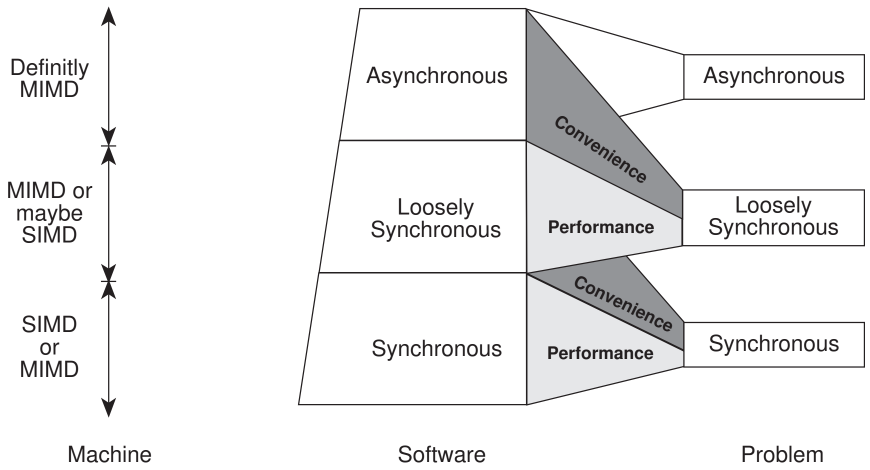


Figure 11: Mapping of Asynchronous, Loosely Synchronous, and Synchronous Levels or Components of Machine, Software and Problem. Each is pictured hierarchically with the asynchronous level at the top and synchronous components at lowest level. Any one of the components may be absent.

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

4 General HPCC Hardware and Software Issues for Five Case Studies

We now illustrate the different machine, problem, and software issues with five 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. We will move from compute dominated to communication/data intensive cases.

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

We have already mentioned in Section 2, 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 4.3 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 asynchronous 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 shown 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.

Makivic [Makivic:95a] has described the implementation of sophisticated path integral Monte Carlos for economic modeling, in particular for option pricing. He describes an HPF implementation, but one would probably use simple parallel C++ (or even Java) in a production implementation, as parallelization is rather straightforward with an array holding the different Monte Carlo paths linked by global reduction operations. This field shows a generalization to metaproblems when pricing a full portfolio consisting of many separate financial instruments.

4.2 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 usually 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].

4.3 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 4(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 HPF provides for arrays.

We now move from CFD to manufacturing, or rather design and manufacturing, which metaproblem includes CFD as a subarea. Manufacturing is particularly interesting, as it needs all aspects of the HPCC initiative from powerful computational engines, huge databases to pervasive secure high-performance networks. Now we use the word "manufacturing" in its broadest sense to include design, the actual processes that build the product, and the full life cycle maintenance. HPCC has a natural high profile role in implementing the popular concept of agile manufacturing, which supposes the model where virtual corporations generate "products-on-demand." The NII is used to link collaborating organizations. HPCC is needed to support instant design (or more accurately redesign or customization) and sophisticated visualization and virtual reality "test drives" for the customer. At the corporate infrastructure level, concurrent engineering involves integration of the different component disciplines—such as design, manufacturing, and product life cycle support—involved in engineering. These general ideas are tested severely when they are applied to the design and manufacturing of complex systems such as automobiles, aircraft, and space vehicles such as shuttles. Both the complexity of these products, and in some sense the maturity of their design, places special constraints and challenges on HPCC.

High-performance computing is important in all aspects of the design of a new aircraft. However, it is worth noting that it has been estimated that less than 5% of the initial costs of the

Boeing 777 aircraft were incurred in computational fluid dynamics (CFD) airflow simulations—the "classic" Grand Challenge in this field described above. On the other hand, over 50% of these sunk costs could be attributed to overall systems issues. Thus, it is useful but not sufficient to study parallel computing for large scale CFD. This is "Amdahl's law for practical HPCC." If only 5% of a problem is parallelized, one can at best speed up and impact one's goals—affordability, time to market—by this small amount. HPCC, thus, must be fully integrated into the entire engineering enterprise to be effective. Very roughly, we can view the ratios of 5% to 50% as a measure of ratio of 1:10 of the relevance of parallel and distributed computing in this case, or alternatively as the ratio of "old style" to "new style" HPCC.

The maturity of the field is illustrated by the design criterion used today. In the past, much effort has been spent on improving performance—more speed, range, altitude, size. These are still critical under extreme conditions, but basically these just form a given design framework that suffices to buy you a place at the table (on the short-list). Rather, the key design criteria is competitiveness, including time to market, and total affordability. Although the design phase is not itself a major cost item, decisions made at this stage lock in most of the full life cycle cost of an aircraft with perhaps 80% of total cost split roughly equally between maintenance and manufacturing. Thus, it certainly would be important to apply HPCC at the design phase to both shorten the design cycle (time to market) and lower the later ongoing costs of manufacturing and maintenance.

We take as an example the design of a future military aircraft—perhaps 10 years from now. This analysis is taken from a set of NASA sponsored activities centered on a study of ASOP—Affordable Systems Optimization Process. This involved an industrial team, including Rockwell International, Northrop Grumman, McDonnell Douglas, General Electric, and General Motors. ASOP is one of several possible approaches to multidisciplinary analysis and design (MAD) and the results of the study should be generally valid to these other MAD systems. The hypothetical aircraft design and construction project could involve six major companies and 20,000 smaller subcontractors. This impressive virtual corporation would be very geographically dispersed on both a national and probably international scale. This project could involve some 50 engineers at the first conceptual design phase. The later preliminary and detailed design stages could involve 200 and 2,000 engineers, respectively. The design would be fully electronic and demand major computing, information systems, and networking resources. For instance, some 10,000 separate programs would be involved in the design. These would range from a parallel CFD airflow simulation around the plane to an expert system to plan location of an inspection port to optimize maintainability. There is a corresponding wide range of computing platforms from PCs to MPPs and a range of languages from spreadsheets to high-performance Fortran. The integrated multidisciplinary optimization does not involve blindly linking all these programs together, but rather a large number of suboptimizations involving at one time a small cluster of these base programs. Here we see clearly, an essential role of HPCC to implement these optimizations, that could well need linking of geographically separated compute and information systems. An aircraft is, of course, a very precise system, which must work essentially flawlessly. This requirement implies a very strict coordination and control of the many different components of the aircraft design. Typically, there will be a master systems database to which all activities are synchronized at regular intervals–perhaps every month. The clustered suboptimizations represent a set of limited excursions from this base design that are managed in a loosely synchronous fashion on a monthly basis. The configuration management and database system are both critical and represent a major difference between manufacturing

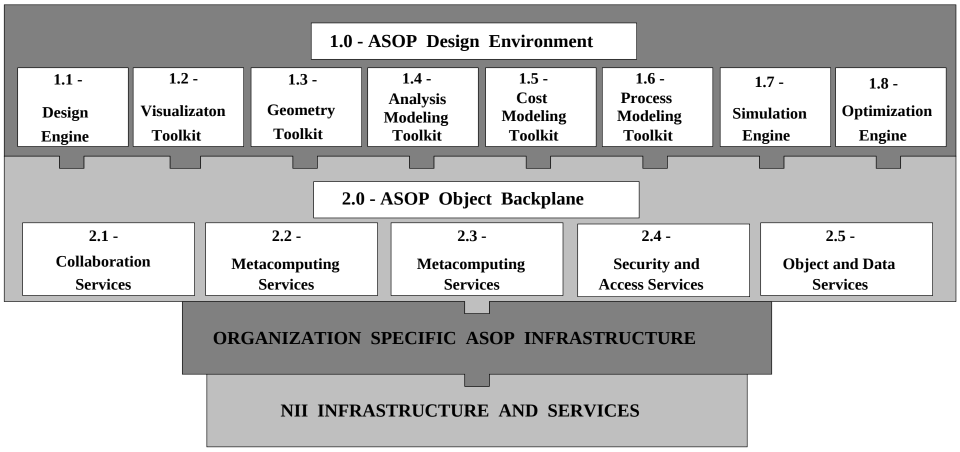


Figure 12: Affordable Systems Optimization Process (ASOP) Implemented on the NII for Aeronautics Systems

and command and control, where in the latter case, real time "as good as you can do" response, is more important than a set of precisely controlled activities. These issues are characteristic of the linkage of HPCC, and the NII where, although loosely coupled, the computers on our global network are linked to "solve a single problem."

ASOP is designed as a software backplane (the NII) linking eight major services or modules shown in Figure 12. These are design (process controller) engine, visualization, optimization engine, simulation engine, process (manufacturing, productibility, supportability) modeling toolkit, costing toolkit, analytic modeling toolkit, and geometry toolkit. These are linked to a set of databases defining both the product and also the component properties. Parallel computing is important in many of the base services, but the linkage of NII, and HPCC technologies is seen in the full system. Using compute enhanced Web Technology (such as, WebWork [Fox:95a] in such systems is natural because one has the mix of geographically distributed people/computers needing both data and simulation services.

4.4 Command and Control (Crisis Management)—Application 25

Command Control (sometimes adding in Computing, Communications, Intelligence Surveillance, and Battle Management with abbreviations lumped together as BMC^{4}IS) is the task of managing and planning a military operation. It is very similar to the civilian area of Crisis management, where the operations involve combating effects of hurricanes, earthquakes, chemical spills, forest fires, etc. Both the military and civilian cases have computational "nuggets" where parallel computing is relevant. These include processing sensor data (signal and image processing—Application 15) and simulations of such things as expected weather patterns and chemical plumes. In this way, many of the components (Applications 1 to 12 in Table 4) are linked as part of this large metaproblem. One also needs large-scale multimedia databases with HPCC issues related to those described for InfoVISiON in Section 4.5.

High-performance communication is needed to link military planners and decision makers, crisis managers, experts at so-called anchor desks, workers (warriors) in the field, information sources such as cable news feeds, and large-scale database and simulation engines.

A key characteristic of the required HPCC support is adaptivity. Crises and battles can occur anywhere and destroy an arbitrary fraction of the existing infrastructure. Adaptivity means making the best use of the remaining links, but also deploying and integrating well mobile enhancements. The information infrastructure must exhibit security and reliability or at least excellent fault tolerance (adaptivity). Network management must deal with the unexpected capacity demands and real time constraints. Priority schemes must allow when needed critical information (such as the chemical plume monitoring and military sensor data) precedence over less time critical information, such as background network video footage.

Needed computing resources will vary from portable handheld systems to large backend MPPs. As there will be unpredictable battery (power) and bandwidth constraints, it is important that uniform user interfaces and similar services be available on all platforms with, of course, the fidelity and quality of a service reflecting the intrinsic power of a given computer. As with the communications infrastructure, we must cope with unexpected capacity demands. As long as the NII is deployed nationally, computational capacity can be exploited in remote sites. The Department of Defense envisages using the basic NII (GII) infrastructure for command and control, augmented by "theater extensions" to bring needed communications into critical areas. The "take it as it is" characteristic of command and control requires that operating systems and programming models support a general adaptive mix (metacomputer) of coordinated geographically distributed but networked computers. This network will adaptively link available people (using perhaps personal digital assistants) to large-scale computation on MPPs and other platforms. There are large computational requirements when forecasting in real-time physical phenomena, such as the weather effects on a projected military action, forest fires, hurricanes, and the structure of damaged buildings. On a longer time scale, simulation can be used for contingency planning and capability assessment. Training with simulated virtual worlds supporting televirtuality, requires major computational resources. In the information arena, applications include datamining to detect anomalous entries (outliers) in large federated multimedia databases. Data fusion including sensor input and processing, geographical information systems (with perhaps three-dimensional terrain rendering), and stereo reconstruction from multiple video streams are examples of compute intensive image processing forming part of the needed HPCC environment.

A critical need for information management involves the best possible high-level extraction of knowledge from databanks—the crisis manager must make judgments in unexpected urgent situations—we cannot carefully tailor and massage data ahead of time. Rather, we need to search a disparate set of multimedia databases. As well as knowledge extraction from particular information sources, the systematic use of metadata allowing fast coarse grain searching is very important. This is a specific example of the importance of standards in expediting access to "unexpected" databases. One requires access to databases specific to crisis region or battlefield, and widespread availability of such geographic and community information in electronic form is essential. There are very difficult policy and security issues, for many of these databases need to be made instantly available in a hassle-free fashion to the military commander or crisis manager—this could run counter to proprietary and security classification constraints. The information system should allow both network news and warriors in the field to deposit in near real-time, digital versions of their crisis and battlefield videos and images.

As mentioned, we expect that human and computer expertise to be available in "anchor desks" to support instant decisions in the heat of the battle. These have been used in a set

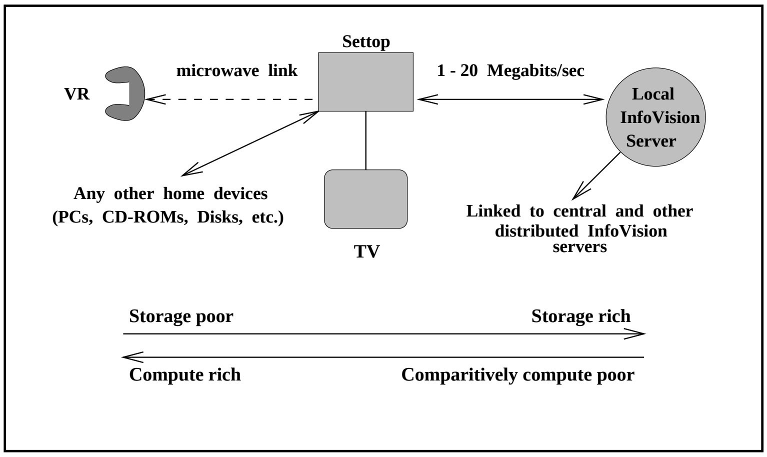


Figure 13: The basic *InfoVISiON* scenario as seen by a home in the year 2000 with an intelligent settop box interfacing the digital home to a hierarchical network of InfoVISiON servers

of military exercises called JWID (Joint Warrior Interoperability Demonstrations). We note that this information scenario is a real-time version of that described in the next section as InfoVISiON to support the society of the Information Age.

Command and Control has historically used distributed computing as the relevant computer and communication resources, are naturally distributed, and not centralized into a single MPP. We see such a HPCC model growing into the standard information support environment for all the nation's enterprises, including business, education, and society. We now explore this in the following section.

4.5 InfoVISiON in Society (Applications 19–24, 26–28, 33)

High-performance distributed computers solve problems in science and engineering. We think of these problems as simulations of airflow, galaxies, bridges, and such things. However, presumably entertaining, informing, and educating society is an equally valid problem. Computers and networks of the NII will be able (see Figure 13) to deliver information at many megabits/second to "every" home, business (office), and school "desk." This network can be considered as a distributed HPCC system because one expects the information to be stored in a set of distributed multimedia services that could vary from PCs to large MPPs and be delivered to a larger set of clients. As shown in Figure 14, one can estimate that the total compute capability in these servers and clients will be some hundred times greater than that of the entire set of supercomputers in the nation.

The computational issues in this application are somewhat different than those for the previous cases we considered. Classic data parallelism and languages, such as High Performance Fortran, are not important. Large-scale distributed databases are the heart of this application, which are accessed through the exploding set of Web technologies. Presumably, these will need to migrate from today's clients (PC/workstations) to more user friendly, and at least initially less flexible settop box implementations controlling home entertainment systems. We will find the same need for data locality as in large scale simulations. As shown in Figure 15, when the latest Hollywood movie is released on the NII, one will not have half the nation directly

NII Compute & Communications Capability in Year 2005 - 2020

100 Supercomputers at a Teraflop each	10^{14} (Fl)ops/sec at 100% Duty Cycle
100 Million NII Offramps or Connections at realtime video speeds	10^{14} bits to words/sec at about 10% to 30% Duty Cycle
100 Million home PCs, Videogames or Settop Boxes at 100-1000 Mega(Fl)ops each	10^{16} to 10^{17} (Fl)ops/sec at about 10% to 30% Duty Cycle
1,000 to 10,000 High Performance Multimedia (parallel) servers each with some 1,000 to 10,000 nodes	10^{15} to 10^{16} ops/sec at 100% Duty Cycle

Each of three components (network connections, clients, servers) has capital value of order \$10 to \$100 billion

Figure 14: An estimate of the communication bandwidth and compute capability contained in the NII and supercomputer industries

connected to Hollywood. Rather, data is automatically replicated or cached on local servers so that one will only need to communicate such "hot" information over a distance of a few miles. As in simulation examples, communication bandwidth will be limited and such steps are needed to reduce demand.

InfoVISiON will require simulation, but it will be more loosely coupled than for say large-scale CFD, and consist of very many smaller problems. Interactive videogaming with multiple players sharing a virtual world is one clear need for simulation on the NII, and for this the three-dimensional database VRML has been introduced. However, another example that can use the same technology is remote viewing and exploration of consumer products, such as cars, furniture, and large appliances. Simulation will support virtual reality like exploration and the automatic customization of such products for particular customers.

InfoVISiON at its "simplest" is "just" access to multimedia database. But even here the "media" can, as in Application 24 include simulation. Further, InfoVISiON underlies decision support in business, government, and military areas (Applications 25, 27, and 28), as well as education (Application 33). Thus, this HPCC application is of great importance, and can be expected to a major business opportunity for the vendors, as shown in Figure 14.

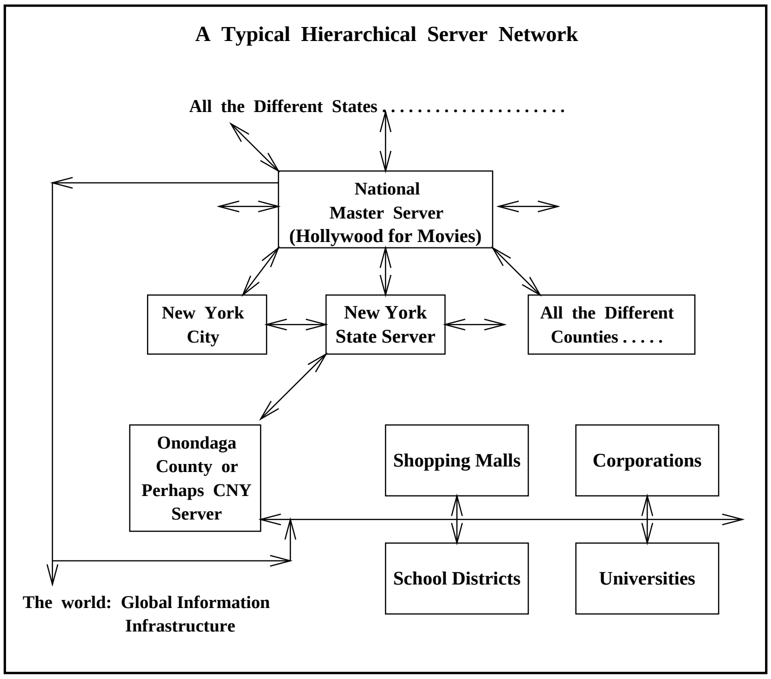


Figure 15: A typical hierarchical server network depicted for a master system in Hollywood cascading down with a fragment of node systems shown for central New York

Glossary

ACTION (Advanced Computing Technology is an Innovative Opportunity Now) The original form of InfoMall focused on large-scale simulations.

Adaptive See Table 2.

APL See Table 6.

Applets An application interface where referencing (perhaps by a mouse click) a remote application as a hyperlink to a *server* causes it to be downloaded and run on the *client*.

ASOP (Affordable Systems Optimization Process) refers to a process using multidisciplinary optimization to produce more affordable systems.

Asynchronous See Table 3.

Asynchronous Transfer Mode (ATM) ATM is expected to be the primary networking technology for the *NII* to support multimedia communications. ATM has fixed length 53 byte messages (cells) and can run over any media with the cells asynchronously transmitted. Typically, ATM is associated with Synchronous Optical Network (SONET) optical fiber digital networks running at rates of OC-1 (51.84 megabits/ sec), OC-3 (155.52 megabits/sec) to OC-48 (2,488.32 megabits/sec).

Asyncsoft See Table 2.

AVS See Table 6.

Back Substitution The stage in *LU decomposition* where an upper triangular matrix $U\underline{x} = L^{-1}b$ is solved "backwards" from the end.

Bandwidth The communications capacity (measured in bits per second) of a transmission line or of a specific path through the network.

C* See Table 6.

C++ See Table 6.

CC++ See Table 6.

CFD (Computational Fluid Dynamics) refers to computational solutions of differential equations, such as the Navier Stokes set, describing fluid motion.

Clustered Computing A commonly found computing environment consists of many workstations connected together by a local area network. The workstations, which have become increasingly powerful over the years, can together, be viewed as a significant computing resource. This resource is commonly known as cluster of workstations, and can be generalized to a heterogeneous collection of machines with arbitrary architecture.

Collective A communication or processing activity in which many messages or compute primitives are linked together.

Command and Control This refers to the computer support decision making environment used by military commanders and intelligence officers. It is described in Section 4.4.

Computational Graph The computational or spatial structure of a problem that is expressed as a dependency graph among linked elements in the computation.

Conjugate Gradient Method A technique for solving systems of linear algebraci equations, which proceeds by minimizing a quadratic residual error function. The method is iterative, but quite powerful: in the absence of roundoff error, it will converge exactly in M steps, where M is the order of the system in question.

Crystal See Table 6.

Dataflow A model of parallel computing in which linked programs are represented by a graph, and execution of a node (of graph) is triggered by arrival of one or more pieces of data "flowing" from previously executed nodes.

Data Fusion A common *command and control* approach where the disparate sources of information available to a military or civilian commander or planner, are integrated (or fused) together. Often, a *GIS* is used as the underlying environment.

Data Locality and Caching A key to sequential parallel and distributed computing is data locality. This concept involves minimizing "distance" between processor and data. In sequential computing, this implies "caching" data in fast memory and arranging computation to minimize access to data not in cache. In parallel and distributed computing, one uses migration and replication to minimize time a given node spends accessing data stored on another node.

Data Mining This describes the search and extraction of unexpected information from large databases. In a database of credit card transactions, conventional database search will generate monthly statements for each customer. Data mining will discover using ingenious algorithms, a linked set of records corresponding to fraudulent activity.

Data Parallelism A model of parallel or distributed computing in which a single operation can be applied to cell elements of a data structure simultaneously. Often, these structures are arrays.

Decision Support Use of computers to supply and process information needed to make decisions. Term is typically applied to business applications.

Distributed Computing The use of networked heterogeneous computers to solve a single problem. The nodes (individual computers) are typically loosely coupled.

Distributed Memory A computer architecture in which the memory of the nodes is distributed as separate units. Distributed memory hardware can support either a distributed memory programming model, such as message passing or a shared memory programming model.

Embarrassingly Parallel A class of problems that can be broken up into parts, which can be executed essentially independently on a parallel or distributed computer (see Table 3).

Event Driven Simulation See Table 2.

Express See Table 6.

Finite Difference Method A *direct method* for the approximate solution of partial differential equations on a discrete grid, by approximating derivatives of the unknown quantities on the grid by linear differences. SIMD machines typically lend themselves very well to the efficient implementation to this sort of applications. See also *finite element method.*

Finite Element Method An approximate method for solving partial differential equations by replacing continuous functions by piecewise approximations defined on polygons, which are referred to as elements. Usually, polynomial approximations are used. The finite element method reduces the problem of finding the solution at the vertices of the polygons to that of solving a set of linear equations. This task may then be accomplished by a number of methods, including *Gaussian elimination*, the *conjugate gradient method*, and the *multigrid method*. See also *finite difference method.*

Fortran-M See Table 6.

Gaussian Elimination A method for solving sets of simultaneous linear equations by eliminating variables from the successive equations. The original equation in the form $Ax = b$ (A is a matrix, b the vector of known values, and x the unknown solution vector) is reduced to $A = LU$ and $Ux = c$, where U is an upper triangular matrix. The solution vector x can then be found by *back substitution*. This method is usually formulated as *LU decomposition*.

Geographical Information System (GIS) A user interface where information is displayed at locations on a digital map. Typically, this involves several possible overlays with different types of information. Functions, such as image processing and planning (such as shortest path) can be invoked.

Gigabit A measure of *network* performance—one Gigabit/sec is a bandwidth of 10^9 bits per second.

Gigaflop A measure of computer performance—one Gigaflop is 10^9 floating point operations per second.

Global Information Infrastructure (GII) The GII is the natural world-wide extension of the *NII* with comparable exciting vision and uncertain vague definition.

Grand Challenges Computation-intensive fundamental problems in science and engineering, with broad economic and scientific impact, whose solution can be advanced by applying HPCC technologies and resources.

HPC++ See Table 6.

High-Performance Computing and Communications (HPCC) Refers generically to the federal initiatives, and associated projects and technologies that encompass *parallel computing*, *HPDC*, and the *NII*.

High-Performance Distributed Computing (HPDC) The use of distributed networked computers to achieve high performance on a single problem, i.e., the computers are coordinated and synchronized to achieve a common goal.

High-Performance Fortran (HPF) A language specification published in 1993 by experts in compiler writing and parallel computation, the aim of which is to define a set of directives which will allow a Fortran 90 program to run efficiently on a distributed memory machine. At the time of writing, many hardware vendors have expressed interests, a few have preliminary compilers (Digital, IBM), and a few independent compiler producers (APR, Portland Group) also have early releases. If successful, HPF would mean data parallel programs can be written portably for various multiprocessor platforms.

HPF+ See Table 2.

Hyperlink The user level mechanism (remote address specified in a *HTML* or *VRML* object) by which remote services are accessed by *Web Clients* or *Servers*.

Hypertext Markup Language (HTML) A syntax for describing documents to be displayed on the *World Wide Web*.

Hypertext Transport Protocol (HTTP) The *protocol* used in the communication *Web Servers* and *clients*.

InfoMall (http://www.infomall.syr.edu) A virtual corporation of some 50 organizations, funded by New York State and led by NPAC at Syracuse University, aimed at building practical HPCC systems and applications.

InfoVISiON Information, Video, Imagery, and Simulation ON demand is scenario described in Section 4.5 where *multimedia servers* deliver multimedia information to clients on demand—at the click of the user's mouse.

Integrated Service Data Network (ISDN) A digital multimedia service standard with a performance of typically 128 kilobits/sec, but with possibility of higher performance. ISDN can be implemented using existing telephone (*POTS*) wiring, but does not have the necessary performance of 1–20 megabits/second needed for full screen TV display at either VHS or high definition TV (HDTV) resolution. Digital video can be usefully sent with ISDN by using quarter screen resolution and/or lower (than 30 per second) frame rate.

Integration Software See Table 2.

Internet A complex set of interlinked national and global networks using the IP messaging protocol, and transferring data, electronic mail, and *World Wide Web*. In 1995, some 20 million people could access Internet—typically by *POTS*. The Internet has some high-speed links, but the majority of transmissions achieve (1995) bandwidths of at best 100 kilobytes/sec. the Internet could be used as the network to support a *metacomputer*, but the limited *bandwidth* indicates that *HPDC* could only be achieved for *embarrassingly parallel* problems.

Internet Protocol (IP) The network-layer communication protocol used in the DARPA Internet. IP is responsible for host-to-host addressing and routing, packet forwarding, and packet fragmentation and reassembly.

Irregular Problems Problems whose computational graph has an irregular structure as seen in finite element approaches to simulation of complex systems.

Java A distributed computing language (*Web Technology*) developed by Sun, which is based on C++ but supports *Applets*.

JavaScript A fully interpreted language based loosely on Java, but aimed at integration of client capabilities of Netscape Web clients (browsers or navigators).

Latency The time taken to service a request or deliver a message which is independent of the size or nature of the operation. The latency of a *message passing* system is the minimum time to deliver a message, even one of zero length that does not have to leave the source processor. The latency of a file system is the time required to decode and execute a null operation.

Linda See Table 6.

Living School Book A collaboration involving NYNEX, Syracuse University School of Education, Columbia Teachers College, and NPAC developing InfoVISiON applications for six schools on a high-speed ATM network.

Loose and Tight Coupling Here, coupling refers to linking of computers in a network. Tight refers to low latency, high bandwidth; loose to high latency and/or low bandwidths. There is no clear dividing line between "loose" or "tight."

Loosely Synchronous See Table 3.

LU Decomposition A technique where a matrix A is represented as the product of a lower triangular matrix, L, and an upper triangular matrix U. This decomposition can be made unique either by stipulating that the diagonal elements of L be unity, or that the diagonal elements of L and U be correspondingly identical.

MAD (Multidisciplinary Analysis and Design, or Multidisciplinary Optimization) refers to the coupling of several areas, such as structural dynamics and fluid flow in a combined tradeoff to produce higher capability vehicles.

Massively Parallel Processing Processor (MPP) The strict definition of MPP is a machine with many interconnected processors, where 'many' is dependent on the state of the art. Currently, the majority of high-end machines have fewer than 256 processors. A more practical definition of an MPP is a machine whose architecture is capable of having many processors—that is, it is scalable. In particular, machines with a distributed memory design (in comparison with shared memory designs) are usually synonymous with MPPs since they are not limited to a certain number of processors. In this sense, "many" is a number larger than the current largest number of processors in a shared-memory machine.

Megabit A measure of network performance—one Megabit/sec is a bandwidth of 10^6 bits per second. Note eight bits represent one character—called a byte.

Message Passing A style of inter-process communication in which processes send discrete messages to one another. Some computer architectures are called message passing architectures because they support this model in hardware, although message passing has often been used to construct operating systems and network software for sequential processors, shared memory, and distributed computers.

Message Passing Interface (MPI) The parallel programming community recently organized an effort to standardize the communication subroutine libraries used for programming on massively parallel computers such as IBM SP2, Intel's Paragon, Cray's T3D, as well as networks of workstations. MPI not only unifies within a common framework programs written in a variety of exiting (and currently incompatible) parallel languages but allows for future portability of programs between machines.

Metacomputer This term describes a collection of heterogeneous computers networked by a high-speed wide area network. Such an environment would recognize the strengths of each machine in the Metacomputer, and use it accordingly to efficiently solve so-called *Metaproblems*. The *World Wide Web* has the potential to be a physical realization of a Metacomputer.

Metaproblem This term describes a class of problem which is outside the scope of a single computer architectures, but is instead best run on a Metacomputer with many disparate designs. These problems consist of many constituent subproblems. An example is the design and manufacture of a modern aircraft, which presents problems in geometry grid generation, fluid flow, acoustics, structural analysis, operational research, visualization, and database management. The *Metacomputer* for such a Metaproblem would be networked workstations, array processors, vector supercomputers, massively parallel processors, and visualization engines.

Metropolis A distinguished Los Alamos scientist whose name is associated with a particular importance sampling (optimized choice of points) approach to *Monte Carlo* methods aimed at statistical physics problems.

Monte Carlo An approach in which many independent trials are used to evaluate integrals, with trial parameters chosen randomly.

Message Passing Fortran (MPF) See Table 2.

Multimedia Server or Client Multimedia refers to information (digital data) with different modalities, including text, images, video, and computer generated simulation. Servers dispense this data, and clients receive it. Some form of browsing, or searching, establishes which data is to be transferred. See also *InfoVISiON*.

Multiple-Instruction/Multiple-Data (MIMD) A *parallel computer* architecture where the nodes have separate instruction streams that can address separate memory locations on each clock cycle. All *HPDC* systems of interest are *MIMD* when viewed as a *metacomputer*, although the nodes of this metacomputer could have *SIMD* architectures.

Multipurpose Internet Mail Extension (MIME) The format used in sending multimedia messages between *Web Clients* and *Servers* that is borrowed from that defined for electronic mail.

National Challenges Information-intensive fundamental applications that have broad and direct impact on the Nation's competetitivess and the well-being of its citizens, and that can benefit from the application of *HPCC* technologies and resources.

National Information Infrastructure (NII) The collection of *ATM*, cable, *ISDN*, *POTS*, satellite, and wireless networks connecting the collection of 10^8–10^9 computers that will be deployed across the U.S.A. as set-top boxes, PCs, workstations, and MPPs in the future.

The NII can be viewed as just the network infrastructure or the full collection of networks, computers, and overlayed software services. The *Internet* and *World Wide Web* are a prototype of the NII.

Network A physical communication medium. A network may consist of one or more buses, a switch, or the links joining processors in a multicomputer.

Node A parallel or distributed system is made of a bunch of nodes or fundamental computing units—typically fully fledged computers in the *MIMD* architecture.

NPAC (http://www.npac.syr.edu) The Northeast Parallel Architectures Center at Syracuse University is an HPCC research and development organization focusing on industrial relevance.

N(UMA) UMA—Uniform Memory Access—refers to shared memory in which all locations have the same access characteristics, including the same access time. NUMA (Non-Uniform Memory Access) refers to the opposite scenario.

Parallel Computer A computer in which several functional units are executing independently. The architecture can vary from *SMP* to *MPP* and the *nodes* (functional units) are *tightly coupled*.

Partial Differential Equation (PDE) The most common mathematical description of physical systems.

PCN See Table 6.

POTS The conventional twisted pair based Plain Old Telephone Service.

Protocol A set of conventions and implementation methodologies defining the communication between nodes on a network. There is a famous seven layer OSI standard model going from physical link (optical fiber to satellite) to application layer (such as Fortran subroutine calls). Any given system, such as PVM or the Web implements a particular set of protocols.

PVM PVM was developed at Emory and Tennessee Universities, and Oak Ridge National Laboratory. It supports the *message passing* programming model on a network of heterogeneous computers (http://www. epm.ornl.gov/pvm/).

Quantum Chromodynamics (QCD) A model of the behavior of matter on sub-nuclear scales, the simulation of which is very hungry of computing power.

Regular Problems Problems whose computational graph has a regular structure as seen in Quantum Chromodynamics simulations or many finite different problems.

Runtime The part of systems support software used when a program is executed. This should be contrasted with compile-time preparation of program.

Settop Box The interface between the home entertainment system (now TV) and outside information (now cable). Future settop boxes will need personal computer-type functionality, and link to Web Technology.

Shared Memory Memory that appears to the user to be contained in a single address space that can be accessed by any process or any node (functional unit) of the computer. Shared memory may have *UMA* or *NUMA* structure. *Distributed computers* can have a shared memory model implemented in either hardware or software—this would always be NUMA. Shared memory parallel computers can be either *NUMA* or *UMA*.

Virtual or Distributed Shared Memory is (the illusion of) a shared memory built with physically distributed memory.

Single-Instruction/Multiple-Data (SIMD) A *parallel computer* architecture in which every node runs in lockstep accessing a single global instruction stream, but with different memory locations addressed by each node. Such synchronous operation is very unnatural for the nodes of a *HPDC* system, or the NII.

Supercomputer the most powerful computer that is available at any given time. As performance is roughly proportional to cost, this is not very well defined for a scalable parallel computer. Traditionally, computers costing some \$10 M–\$30 M are termed supercomputers.

Symmetric Multiprocessor (SMP) A Symmetric Multiprocessor supports a shared memory programming model—typically with a *UMA* memory system, and a collection of up to 32 nodes connected with a bus.

Synchronous See Table 3.

Televirtual The ultimate computer illusion where the user is fully integrated into a simulated environment and so can interact naturally with fellow users distributed around the globe.

Teraflop A measure of computer performance—one Teraflop is 10^{12} floating point operations per second.

Terrain Rendering The graphics processing of a three-dimensional scene, typically with a digital map as a 3D backdrop.

Timewarp See Table 6.

Time Stepped Simulation (TS) See Table 2.

Video-on-Demand A component of *InfoVISiON* where digital video (movies, news, etc.) is supplied at the request of clients.

Vienna Fortran See Table 6.

Virtual Reality (VR) See Table 2.

Virtual Reality Modeling Language (VRML) A "three-dimensional" HTML that can be used to give a universal description of three-dimensional objects that supports *hyperlinks* to additional information. The latest VRML 2.0 is aimed at supporting virtual environments implementing televirtuality.

Web Clients and Servers A distributed set of clients (requesters and receivers of services) and servers (receiving and satisfying requests from clients) using *Web Technologies*.

WebTop refers to the implementation of a set of standard desk top and personal computer tools, which are essential in any computing environment.

WebWindows The operating environment created on the World Wide Web to manage a distributed set of networked computers. WebWindows is built from *Web clients* and *Web servers*.

WebWork (Fox:95a) An environment proposed by Boston University, Cooperating Systems Corporation, and Syracuse University, which integrates computing and information services to support a rich distributed programming environment.

World Wide Web and Web Technologies A very important software model for accessing information on the Internet based on hyperlinks supported by *Web technologies*, such as *HTTP*, *HTML*, *MIME*, *Java*, *Applets*, and *VRML*.

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," in *Proceedings of Supercomputing '93*, page 351. IEEE Computer Society Press, November 1993. Syracuse University, NPAC Technical Report SCCS-498.

[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.

[Fox:95k] Fox, G., Hawick, K., and White, A. "Characteristics of HPC scientific and engineering applications." Technical Report SCCS-741, Syracuse University, NPAC, Syracuse, NY, 1995. Report of Working Group 2 of the *Second Pasadena Workshop on System Software and Tools for High Performance Computing Environments.*

[Fox:96c] Fox, G. C., and Furmanski, W. "SNAP, Crackle, WebWindows!." Technical Report SCCS-758, Syracuse University, NPAC, Syracuse, NY, April 1996.

[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.

[HPCC:96a] National Science and Technology Council, "High performance computing and communications," 1996. 1996 Federal Blue Book. A report by the Committee on Information and Communications. Web address http://www.hpcc.gov/blue96/.

[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.

[Makivic:95a] Makivic, M. "Path integral Monte Carlo method for valuation of derivative securities: Algorithm and parallel implementation," *Neural Network World*, 5(4):503–524, 1995.

[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." Technical Report SCCS-731, Syracuse University, NPAC, Syracuse, NY, July 1995. 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.