Fermi National Accelerator Laboratory

FERMILAB-Conf-89/220

Designing Machines for Lattice Physics and Algorithm Investigation*

M. Fischler, R. Atac, A. Cook, J. Deppe, E. Eichten, I. Gaines, M. Gao,
G. Hockney, D. Husby, A. Kronfeld, P. Mackenzie, T. Nash, T. Pham, and T. Zmuda
Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

October 1989

*Presented by M. Fischler at the International Workshop "LATTICE 89," Isola di Capri, Italy, September 18-21, 1989. Proceedings to be published in Nucl. Phys. B, Proceedings Supplement Section.

Operated by Universities Research Association Inc. under contract with the United States Department of Energy

DESIGNING MACHINES FOR LATTICE PHYSICS AND ALGORITHM INVESTIGATION

M. Fischler, R. Atac, A. Cook, J. Deppe, E. Eichten, I. Gaines, M. Gao, G. Hockney, D. Husby,
A. Kronfeld, P. Mackenzie, T. Nash, T. Pham and T. Zmuda

Fermi National Accelerator Laboratory, P. O. Box 500, Batavia, IL 60510, USA*

Special-purpose computers are appropriate tools for the study of lattice gauge theory. While these machines deliver considerable processing power, it is also important to be able to program complex physics ideas and investigate algorithms on them. We examine features that facilitate coding of physics problems, and flexibility in algorithms. Appropriate balances among power, memory, communications and I/O capabilities are presented.

1. INTRODUCTION

To be useful as a platform for the investigation of algorithms and for doing a broad spectrum of physics analyses, a system must be powerful and large enough to do the problems of interest. Beyond that, however, are issues of machine architecture, software tools, and paradigms provided.

Systems range from those designed for development to those intended to run a single program over extended periods. Machines at either end of this spectrum have diminished utility. Production systems need the flexibility to do complex physics analysis based on the large field configurations generated, and must not allow machine restrictions to determine what physics is done. Development systems need sufficient power to investigate the physically interesting regions of large lattices near critical points. In either case, the benefits of natural and well structured coding are considerable.

Based on experience of physicists using CANOPY[1] grid-oriented software tools on the ACPMAPS[2] system at Fermilab, we note hardware and software features needed to facilitate physics coding and algorithm development.

2. MACHINE REQUIREMENTS

Inevitably, tradeoffs occur among power, flexibility and ease of programming. An algorithm development system will resolve these emphasizing programming issues, rather than peak efficiency per unit cost. Such a system should not be designed around a particular algorithm in mind — flexibility should be the major goal. These considerations cannot be taken to extremes. For example, it is appropriate to design an architecture for site-oriented problems, rather than a completely general-purpose system.

Features which facilitate algorithm development include large memory, transparent and fast global communication, large-scale I/O capabilities, modularity, and multiple instruction stream architecture. A quantitative feel for these factors can be obtained by doing physics on a flexible system and noting where the development process is affected.

Memory should be large enough that it is not the primary factor in choosing algorithms or lattice sizes (this is somewhat problem dependant). Interprocessor communication must be fast enough, in terms of overhead and bandwidth, that desired algorithms are not communica-

* Fermilab is operated by Universities Research Association, Inc. under contract # DE-AC02-76CH03000 with the U.S. Department of Energy.

tions bound. In order to investigate non-local algorithms, global communications are needed.

The I/O system should be large and fast enough for temporary data storage, and must include archiving capabilities, since generated data will often require multiple passes of physics analysis. The system should be expandable to whatever levels of CPU and I/O power are appropriate.

Multiple instruction, multiple data (MIMD) architecture can handle problems which are awkward or impossible for single instruction, multiple data (SIMD) architectures, such as heat bath and incomplete LU decomposition algorithms[3] and random lattice problems. Even for algorithms which can be cast into SIMD form, MIMD architecture simplifies the implementation of software tools to facilitate coding. It also makes it easy to decouple lattice sizes and shapes from the hardware details.

There is no simple dichotomy between SIMD and MIMD. The extreme case of multiple instruction streams would be each CPU proceeding whenever data is ready, without centralized control — this may be awkward for tightly coupled problems. For lattice physics, it is natural to consider algorithms to be done in steps — the processors act independently during each "task", but re-synchronize at the end of each step. The ACPMAPS system is used in this way. Less flexible are systems which are MIMD in processing, but require lockstep communication (e.g. the Columbia machine[4]; these are often called SIMD systems, because for tightly coupled problems their MIMD potential is rarely important. Uniprocessor scalar machines act as MIMD systems; vector machines behave like SIMD systems. Here too, the "SIMD" machine may be more efficient, but is harder to program and less flexible.

3. SOFTWARE DESIGN

A package of tools to be used by applications programs should provide an "underpinning" for physics code. The goals should be to minimize the programming efforts needed, to provide a natural coding environment, and to maximize the chances that programs will be correct. Absent from these primary goals is efficiency. However, many ubiquitous operations (such as accessing data from neighboring sites) are done by the software underpinning. Those operations can be optimized once and for all. A user sensitive to efficiency can concentrate on routines particular to one problem. For these reasons, the cost of using of a convenient set of underpinning routines can be low. For example, the heat-bath link update time per link is about the same (relative to peak power) for ACPMAPS as for the less flexible APE[5], GF11[6], QCDPAX[7], and Columbia 256-node machines.

Since the user wants to concentrate on the physics or algorithm being coded, the tools provided should handle grid related and machine dependant issues. The multiprocessing nature of the system should be transparent; the parallelism over sites automated, and the grid connectivity structure should be implemented in a natural and automatic way. The environment must allow easy coding, and encourage well organized programs, which are easier for other physicists understand and build upon. The tools should be implemented on a variety of systems, so programs are portable.

On ACPMAPS, this software underpinning is provided by CANOPY, a set of routines expressing a natural "paradigm" of how lattice problems look. We assume that the problem can be expressed in terms of grids, consisting of sites, with data at each site; the algorithm consists of a sequence of tasks, each done for some set of sites. This is the natural paradigm for lattice gauge algorithms, and for a large variety of other scientific problems.

A program using CANOPY has two parts: the control program, and task routines. The control program manages global matters such as defining lattices and fields and starting lattice-wide operations on "fields". The per-site data structures are fields—the data associated with each site is kept on the node responsible for executing tasks for that site. Instead of constructing loops over sites, the CANOPY user calls do_task to execute the task routine for an entire set of sites—the multinode nature of the system is transparent. Access to field data is via CANOPY routines, which compute the location of the data and perform any internode communication necessary—the distribution of data among nodes is also hidden.

Fundamental CANOPY concepts include grids, sites on a grid, fields on each site, link fields, directions along the grid, paths, and maps between grids. These geometric concepts automate the "routine" (but error-prone) portions of the algorithm.

CANOPY is written in C and is easily portable to single-CPU or MIMD systems which support UNIX calls. Thus, programs can originate on a workstation, and migrate to production machines without changing any code. The software has been implemented on the ACPMAPS system, and several single-CPU and a smaller multiprocessing system. CANOPY applications tend to be structured and modular. This leads to more readable code, easier modification and debugging, and the ability to confidently optimize time-critical routines.

The natural language for working with CANOPY concepts is C, although Fortran subroutines are also supported. UNIX system calls are supported, and the system is networked with other computers, for ease in handling programs, inputs, and result summary data.

The ability to split a machine into several partitions, with one user running on each part, is very important. The alternative would be to provide several development machines of varying sizes; aside from inefficiency, that has the drawback of having to guess the "right" sizes. An intelligent scheduler allocates resources to each user, balancing the needs of production running with the rapid turnaround desired for development.

4. THE ACPMAPS HARDWARE

The ACPMAPS machine satisfies the requirements for algorithm development and production running. The system shares some features with most other special-purpose lattice gauge engines: it is powered by many processors, each with its own local memory and some means of accessing other processors' data. However, it is a MIMD system, with transparent global communication. The current ACPMAPS system uses 256 processing nodes, providing 5 Gflops of peak power.

The processor nodes for ACPMAPS are based on the Weitek XL-8032 chip set, which provides an integrated integer/floating point computer with support for C and Fortran. 8 Mbytes of data and 2 Mbytes of instruction memory are on each 20 Mflop processor[8]. Each node works asynchronously and, in general, on different numbers of sites for each task. Processors employing the XL-3164 FPU for double precision are also available.

The key flexibility feature of the ACPMAPS hardware is fast, asynchronous, transparent global communication. The nodes plug into a network of switch crates[9], whose backplanes implement sixteen port crossbar switching at bandwidths of 20 Mbytes/second per channel. These bus switch backplane crates are interconnected by 20 Mbytes/sec cables. Each crate contains a PROM which provides routing information. The communications system is logically equivalent to a telephone switching network — a channel is opened by specifying the destination, and data is transmitted independent of other channels opened. The current switch crate hardware supports systems of up to 2048 devices.

The topology chosen for the 256-node ACPMAPS system is that of a 2^5 hypercube of crates, each containing eight processors. Switching times are .5 μsec per connection; processor overhead for establishing the communications channel is 3-4 μsec. Because each node needs only one communications interface, the expense for communications may be less than that for systems with hardwired neighbor grids.

The I/O subsystem design includes 32 SCSI-based disk drives (allowing for a high bandwidth to 20 Gbytes of distributed disk) and 32 Exabyte tape drives for archiving fields. A tape system alone is not sufficient for production or development work. The disks provide the ability to: •eliminate tape mount and seek overheads; •share data files; •rearrange data before writing it to tape (e.g. propagators in time slice order); •swap out long jobs, making possible reasonable turnaround for development; •Store temporary large data fields, under user control, to soften memory limitations. CANOPY routines handle the distributed I/O; the user reads or writes fields, and is not concerned with how many nodes or disks are involved.

The system is hosted by a UNIX computer, capable of accessing the nodes and the I/O devices, and networked to other lab computers.

5. QUANTITATIVE REQUIREMENTS

The following observations are based on several physicists' use of ACPMAPS for a few months, doing quenched configuration and propagator generation and physics analysis[10]. Nobody has a surplus of power; therefore, all other requirements are "per effective megaflop" rather than absolute. The effective power of each node can be derived from the time taken (.6 msec) per link update; this works out to about 8 Mflops.

8 Mbytes of data memory per node seems barely sufficient. There are instances of "stuffing in as big a lattice as you can", and the consensus is that "any less would be awkward". Thus, **memory size of 1-2 Mbytes/Mflop** may be a good balance. (Dynamical fermions require more power, but algorithm improvements tend to decrease power needs.) The bandwidth to memory in ACPMAPS is 40 Mbytes/sec; we can experimentally halve that, costing 25% in performance. Thus **memory bandwidth ≥ 2 - 3 Mbytes/sec/Mflop** is needed.

Communications needs scale with power and surface/volume ratio (except in lockstep systems where communication rates affect bandwidth to local memory). Thus a relevant number is bandwidth/power*L (L is the geometric mean size in each dimension of the chunk of lattice in each node). Since our intercrate bandwidth is the earliest bottleneck, our "processing unit" is the 8-node crate (L=14). On that basis, BL/P is 4.4, and our bandwidth effects are small. For optimized conjugate gradient-like algorithms, **communications bandwidths ≥ 2/L Mbytes/Mflop** are acceptable. Communications overhead (O_c) of 4.5 µsec * 8 flops/ µsec in a node (L=8) affects performance at the 5% level. Roughly, this cost is **.01 * (O_c* power/L).**

For staging and shuffling of data, the bandwidth to disk must be capable of moving the entire **memory to disk in ≤ 300 sec** (100 sec for effective swapping). The total **disk system size needed is ≥ 10 times memory size.** This necessitates a distributed I/O system.

6. SUMMARY

Special-purpose systems for lattice gauge can be flexible and programmable enough to be useful for algorithm development and complicated physics applications, yet cost effective for production running. Results of work done on ACPMAPS on heavy—light mesons, mass spectroscopy, and full QCD thermodynamics, are reported in this volume[3]. A detailed study of various conjugate gradient and minimal residual propagator inversion algorithms, with various LU decomposition and FFT preconditioners, is also described[10]. Methods were found which converge more rapidly than the naive ones; they are less effective in terms of raw flops, yet several times better in terms of net speed. These improved algorithms, as well as the CANOPY software that facilitates investigations, would be difficult or impossible to implement efficiently on SIMD hardware.

REFERENCES

1. CANOPY Manual, G. Hockey, P. Mackenzie, and M. Fischler, Fermilab document, available through the authors.
2. T. Nash *et al.* , A Site Oriented Supercomputer for Theoretical Physics, FERMILAB-Conf-89/58, to be published in Proceedings of the 4th Hypercubes Concurrent Computers and Applications Conference.
3. G. Hockney, Comparison of Algorithms, this volume.
4. F. Butler, Proceedings of the 1988 Symposium on Lattice Field Theory (North Holland), 557-561. See also N. Christ, Status of the Columbia 256-node Parallel Supercomputer, this volume.
5. The APE collaboration, Proceedings of 1988 Symposium on Lattice Field Theory (North Holland), 562-565.
6. J. Sexton, ibid. See also D. Weingarten, Progress Report on the GF11 Project, this volume.
7. Y. Iwasaki, Status of QCDPAX, this volume.
8. D. Husby *et al.* , IEEE Transactions on Nuclear Science, NS-36, No. 1, 734-737 (1989)
9. R. Atac *et al.*, ibid.
10. E. Eichten, B Meson Weak Interactions, this volume. A. Kronfeld, Improved Operators for Hadrons, this volume. D. Toussaint, Simulating QCD with a Chemical Potential, this volume.