Simulating Quarks and Gluons on MIMD Parallel Computers

T. DeGrand (for the MILC Collaboration), Physics Department,
Univeristy of Colorado, Boulder, CO

Physicists believe that Quantum Chromodynamics (QCD) describes the strong interactions of high energy physics. This fundamental theory explains how quarks and gluons interact to form the subatomic particles observed in nature. Calculations with this theory require large scale numerical simulations which tax the capabilities of the largest supercomputers presently available. The objectives of these calculations will be briefly described, but emphasis will be placed upon the use of massively parallel computers, such as the Cray T3D, to carry out the simulations.

1. INTRODUCTION

The particles that make up the atomic nucleus, protons and neutrons, are not fundamental, but are themselves bound states of quarks. The force that binds quarks together is called "quantum chromodynamics" or QCD. Some of the properties of QCD can be calculated analytically, but when quarks are separated by distances on the order of a nuclear diameter (10^{-13} cm) their interactions become strong. At present the only known approach for performing such calculations from first principles is through large scale numerical simulations within the framework of lattice gauge theory.

There are many good reviews and introductions to lattice gauge theory and its use in QCD [1]. The lattice community has a large annual meeting and the proceedings of those meetings (Lattice 'XX, published so far by North Holland) are the best places to find the most recent results. However, as in any large community with its own set of problems, most of the papers in those proceedings tend to talk to each other in a language which is rather opaque to nonmembers. My goal is an impressionistic overview of the field as it presently exists, which might be useful to an outsider.

The bottom line is that for the past several years there have been a lot of lattice calculations of masses and matrix elements which agree with experiment at the ten to fifteen per cent level.

I will begin with a very superficial overview of how lattice calculations are performed. I will briefly describe the algorithms we use, and then discuss how the research collaboration I am representing (the MILC [2] collaboration) brought these calculations to MIMD machines like the T3D. Then I will discuss my own experiences on the T3D. Finally I will turn to a set of case studies: spectroscopy of light hadrons, the decay constants of D- and B-mesons, and a few words about QCD thermodynamics.

2. HOW LATTICE CALCULATIONS ARE CARRIED OUT

Lattice calculations are performed using the Euclidean path integral formulation of quantum field theory. If we have some field theory with field variables ϕ (ϕ could be quarks, gluons (the particles in the theory which mediate interactions in analogy with the photon), ...) and a Lagrange density $\mathcal{L}(\phi)$, we define an analog of the partition function in statistical mechanics

$$Z = \int [d\phi(x,t)] \exp(-\int d^4x \mathcal{L}(\phi)) \qquad (1)$$

(here $x_\mu = (x, it)$). The expectation value of any observable $O(\phi)$ is given by

$$\langle O \rangle = \frac{1}{Z} \int [d\phi(x,t)] O(\phi) \exp(-\int d^4x \mathcal{L}(\phi)). \qquad (2)$$

To be able to perform calculations in any quantum field theory one must introduce a short distance cutoff which regulates the ultraviolet divergences. We do that by replacing continuous space time by a lattice of grid points $x = ax_i$ where a is the lattice spacing, and defining the field on those grid points $\phi(x) \to \phi_i = \phi(x_i)$. Then the functional integrals Eqns. (1) and (2) become ordinary integrals of very high dimensionality. One evaluates Eqn. (2) using importance sampling: somehow one creates an album of snapshots of the field variables ϕ_i where the probability that a particular configuration is present in the album is $P(\phi_j) = \exp(-\sum_x \mathcal{L}(\phi_j)$ and then

$$\langle O \rangle = \frac{1}{N} \sum_{j=1}^{N} O(\phi_j) + O(\frac{1}{\sqrt{N}}). \qquad (3)$$

The generation of the album is done using Monte Carlo techniques.

Lattice calculations are hard for several reasons: The lattice spacing should be small–small enough that physics

on a size scale less than a lattice spacing can be described using perturbation theory. The size of the simulation volume L^4 should be greater than the physical size of the hadrons. This point is in conflict with the first item. The number of grid points is $n = (L/a)^4$. A gluon field is a three by three complex matrix per each direction on each lattice site, or 72 real numbers per lattice site. Fermions have four spins and three colors or 24 real numbers per site. Typical simulations have lattice spacings around 1/10 fermi (within a factor of two) and a number of mesh points ranging from $16^3 \times 32$ to $24^3 \times 40$ to $32^3 \times 64$: the end is not yet in sight! One needs a lot of statistics–tens to hundreds of uncorrelated lattice measurements. It is very hard to compute with light (u,d) quark masses at their physical values. On the lattice calculating a quark propagator $G_q(x, x')$ involves inverting the matrix problem $(\not{D} - m_q)G_q(x, x') = \delta^4(x - x')$. The matrix becomes singular as $m_q \to 0$. One typically performs a calculation at an unphysically heavy value of the light quark mass and then tries to extrapolate to $m_q = 0$. Sea quarks (virtual quark-antiquark pairs) are a problem because of Fermi statistics, which effectively introduces long range interactions among the quarks. There are techniques for dealing with this problem[3,4] but they make QCD with dynamical fermions orders of magnitude more difficult than if the sea quarks were not there (and the difficulty scales inversely as a power of the quark mass). A rather drastic approximation called the quenched approximation neglects this problem simply by throwing away all the sea quarks. This is an uncontrolled approximation which people do mainly because the alternative (keeping light sea quarks) is too time consuming for the computer.

All these constraints add up to a very hard numerical problem. We use the fastest supercomputers available. After about 1990 old-style Cray's were too slow. Some groups have built their own computers. One of the projects I belong to used half of a Connection Machine CM-2 (at a speed of about 3 1/2 Gflops) for about two years. This is not considered an excessive amount of resources.

Our numerical tools include sparse matrix inverters (to construct quark propagators) and various Monte Carlo or molecular dynamics evolution schemes. The only robust sparse matrix inversion algorithm which works for us is conjugate gradient or its relatives. The matrices we invert are not diagonal in Fourier space, and the matrices are so big ($N > 16^4$) that the only practical preconditioning is the use of a "reduced basis" (incomplete Cholesky decomposition on a red-black checkerboard). Nobody has invented a working multigrid algorithm for production running. Molecular dynamics updating is almost always staggered leapfrog (we need reversability). We also have not been able to invent practical implicit algorithms which satisfy our physics constraints and for which improvement beats the extra computational overhead.

In some ways lattice gauge theory is easier than "standard Cray projects" like weather forecasting. Our grids are regular. Load balancing is almost automatic. We also do not have to deal with interactions on wildly different length scales; for us the aspect ratio is set by the ratio of the length of the box to the lattice spacing. (Physics on distances shorter than the lattice spacing must be handled by perturbation theory, which is a nontrivial problem in itself.) These reasons are probably why lattice QCD groups are among the first "friendly users" on a new machine.

The major problem facing lattice calculations these days are systematics: Quenching, is a small enough, is L big enough, is the quark mass small enough? All these systematics are entangled in any one calculation. Lattice calculations produce as output not a hadron mass m_H but the combination am_H. One finds a by dividing am_H by a measured m_H (in MeV). The problem is, which mass to use? Most lattice calculations only reproduce mass ratios at the ten or fifteen per cent level, so the lattice spacing is uncertain at that level. This uncertainty propagates into essentially all interesting calculations.

3. OUR MIMD CODES

The MILC Collaboration has developed a family of codes for the study of QCD. The codes have compilation options which allow one to choose between different algorithms. In all lattice gauge theory calculations, a very significant fraction of the computer time is spent inverting the lattice Dirac operator. Compiler options allow one to chose between the conjugate gradient and conjugate residual algorithms for this purpose. Our code runs on a wide variety of scalable parallel computers including the Paragon, CM-5, T3D and SP2. It also runs on single processor workstations, which we use for much of our code development. The codes are available on the Web: look at `http://heplibw3.slac.stanford.edu/FIND/FREEHEP/NAME/MILC_QCD/FULL`.

In our programs all of the physical variables associated with a lattice point are stored together in a single structure. This organization of the data is advantageous for a processor with a data cache, such as the i860. It is very different from the optimal organization for vector machines, where, for example, all the real parts of the first row first column of the x-direction matrices would be stored as a single vector, followed by another vector for the imaginary parts, etc. Pointers to the structures at neighboring sites and the pointers set by the communications routines, are stored in separate lists. This turns out to optimize the cache hits since these lists are usually scanned sequentially.

Since the variables "live on" a four dimensional lattice, it is natural to divide the computation among processors by assigning each processor a piece of the four dimensional space-time. These pieces can be four dimensional hyper-

cubes, or they can be two or three dimensional slices of the four dimensional lattice. For example, if we are dividing an $8 \times 8 \times 8 \times 8$ lattice among 16 processors, we might assign each processor a $4 \times 4 \times 4 \times 4$ piece of the lattice. Alternatively, we might assign each processor an $8 \times 8 \times 4 \times 1$ part of the lattice. The division into hypercubes has the advantage that when the amount of lattice on each processor is large the number of sites "on the surface" is minimized. On the other hand, for small lattices this may be outweighed by the fact that in the "slice" distributions there are some directions in which every site has its nearest neighbor on the same processor. The key point is that the lattice remains regular throughout our computations, and approximately the same amount of computation is required at each lattice site. This contrasts with many applications for which the load on the processors can become imbalanced as the computation proceeds unless some of the computation originally assigned to one processor is given to another.

In thinking about the computation, we imagine that computing is done "at a site", by whatever processor is in charge of that lattice site. In evaluating the updated value of a field at a site, we need to know the variables at other lattice sites. In thinking about the physics we do not want to worry about whether these other lattice sites are on the same processor as the variable we are evaluating. As always, the solution is to hide the details of accessing variables. What is needed is a set of routines for accessing fields at other sites which work whether the other site is on the same processor or a different one. Once these are written, we need think only about whether a variable is stored at the same lattice site at which we are computing, and not worry about what processor it is on. This is a very general need, and we have only addressed it for simulations which share some simplifying features.

There are three important simplifying features of the accesses to variables at other lattice sites in QCD simulations: they are homogeneous, mostly local and predictable. The accesses are homogeneous in the sense that when a computation at one lattice site reads a variable from a neighboring lattice site, the computations at all lattice sites will access the same variable at their neighboring sites. Actually, things are a little more complicated. Most of the applications involve different computations on the red or black sites of a four dimensional checkerboard. If the computations on red and black sites are different, homogeneity means that when a computation at one red site accesses data at a neighbor, all red sites will access data at their respective neighbors. In QCD simulations these other sites at which variables are accessed are almost always neighbors, usually nearest neighbors, of the site at which the new variable is being computed. In this sense the accesses are "local". (Among the exceptions are the butterfly and bit reverse in the FFT. These are homogeneous and predictable, but long range in the four dimensional lattice.) Finally, the accesses are predictable. We take advantage of this predictability by making tables in the startup part of our program which list all of the sites on each processor that have their nearest neighbors on another processor, with separate tables for each of the eight directions (forward in four directions and backward in four directions). Fortunately, most of these lists are empty! For reasonable distributions of lattice sites among processors, fetching data from the nearest neighbor in one direction of every site on the processor involves communications with only one or two of the other nodes. On a machine which allows synchronous communication and computation like the iPSC/860 we can start the accesses, do some other piece of computation while waiting for any internode communication that may be required, and then use the data when it arrives.

Most of the accesses to fields at other sites can be done by a sequence of three routines. These routines work by setting a pointer at each lattice site on the desired checkerboard. If the neighboring site is on the same node, the pointer will just be set to the address of the field on the neighboring site, but if the neighboring site is on another node, the pointer will be set to an address in a temporary buffer used to receive a message. The first routine, "`start_gather`", checks to see if any messages need to be sent or received, allocates buffers for messages to be received, sends any needed messages, and sets pointers on all sites where the neighboring site is on the same node — all hidden from the user. Following the call to this routine, other computation may be done, or gathers from other directions or of other variables may be started. This happens concurrently with any communication required by the gather. Before the gathered data is accessed a second routine, "`wait_gather`", must be called. This routine makes sure that all required messages have arrived and sets the pointers on all sites whose neighbors are off-node. Finally, after we are done with the data a third routine, "`cleanup_gather`", frees all the temporary buffers that were allocated. To set up the lists of neighbors used by these routines, the high level code calls a routine "`make_gather`", one of whose arguments is a function which returns the coordinates of the neighbor site. It is not necessary that the "neighbor" actually be close on the four dimensional lattice. The same algorithms can be used for any orderly interchange of data among lattice sites. For example, in our analysis code we do Fast Fourier transforms using these same gather routines for the butterfly and bit reverse interchanges.

All of the routines which involve internode communications, as well as various "housekeeping" routines such as the function that returns the node number of a processor, are isolated in one file. A version of this file exists for each machine on which our code runs. To move from

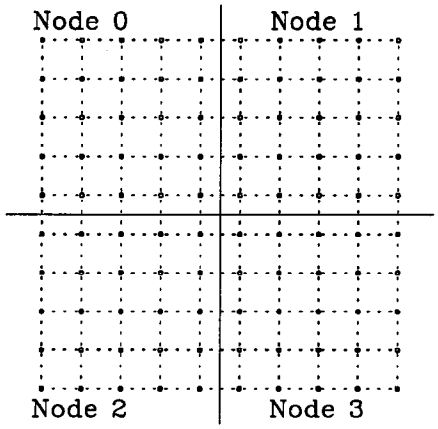

Figure 1. A cartoon of a lattice spread among several processors.

one machine to another we simply link the appropriate file. Similarly, the functions which determine the layout of the lattice sites among the processors are isolated in another file, which contains routines that return the processor number on which any lattice site lives and the address on its processor of any lattice site. Thus to change the way in which lattice sites are distributed among processors we just plug in another file. It turns out that there are not great differences in the performance with different layout algorithms. (The exception is a random distribution of sites among processors, which we used in debugging the code. This slows the code by a large factor, since in a gather every node must send and receive messages from every other node.)

The vanilla version of our code is written in C, and is highly portable. However, in order to optimize performance, it is necessary to take into account the specific architecture of the processors, and to do some assembler language programming. For the Paragon we have written approximately 25 computationally intensive subroutines in assembler language. This has led to a speedup of slightly more than a factor of three in the performance on a single node. For the CM-5 the entire conjugate gradient routine, which uses approximately 90% of the floating point operations in a typical lattice gauge theory calculation, has been written in assembler language. This routine runs at 40 Mflops per node. However, the CM-5 C compiler does not support access to the vector units, so the remainder of the code runs on the SPARC chips at a speed of approximately 3 Mflops per node. For the T3D we have rearranged some of the assembler code produced by the Cray compiler, and written a few subroutines in assembler language. This work has led to a factor of two improvement in the performance over the vanilla C code. We obtained early access to parallel operations on the CTC SP2 in December, 1994, and ported our SP1 MPL code it. The benchmarks presented above are from this first port. As we start tuning the code, we expect a substantial improvement. We are currently developing machine language versions of some of the small, but time-critical routines, such as the SU(3) matrix times vector operation.

Table I above shows the current performance for our code on various MIMD machines. For the Paragon, T3D and SP2 we present benchmarks for the conjugate gradient inversion routine for staggered quarks. The full code runs at a slightly faster rate. For the CM-5 we present benchmarks for the conjugate gradient routine for staggered quarks, and the overall performance of the code for a typical production run on a $24^3 \times 12$ lattice with a quark mass $am_q = 0.008$.

4. MY LIFE ON THE T3D

Essentially all my recent parallel computing has been done on Intel machines. Although my colleagues in the MILC Group have been making extensive use of T3Ds at the Pittsburgh Supercomputer Center and at NERSC, before February of this year the only thing I had ever done on a Cray was to change my password. When I was asked to give this talk, I thought I should see what the Cray was actually like. I also had to compute a few numbers for a work station sized project I am involved in. I figured the T3D could produce the numbers in a few evenings' running, which my Indigo work stations would need a couple of weeks to duplicate.

Others in the MILC group had ported their particular projects. I copied a pre-existing Makefile and edited it to produce my executable. The (borrowed) communication routine did not compile so I borrowed its object file. The program compiled but crashed. A few mail messages to my friends revealed that Cray had been fooling around with the system software (and one of the header files redefined "complex" in a way that conflicted with our definitions of complex numbers.) One of my friends had patched a private copy the offending header file and with it I got a clean compile and a running code. Total elapsed time: a couple of hours spread over a couple of days.

Batch jobs were easy; I just copied a T3D NQS file and edited it. Some of the calls are different from Paragon dialect, but this was not a big deal. Two or three days later (at one two hour job per day) I was done. I would say the machine is slightly more user-friendly that the Paragon (where I have most of my experience), but it is not a Sun work station, or even an Indigo.

My friends report the machine is very stable. The only jobs that failed were ones that exceeded the time limit. I have no experience with mass storage. I know that our Cray binary lattices are incompatible with ones from other

Table 1: MIMD QCD code performance in Megaflops (MF) per node on current machines.

Machine	Nodes	Lattice Size	MF per node
Paragon	1	8^4	26
OSF/1	16	16^4	24
	64	$16^2 \times 32^2$	23
	128	16×32^3	23
	256	32^4	22
CM-5 (Conjugate gradient)	128	$24^3 \times 12$	40
CM-5 (Full code)	128	$24^3 \times 12$	23
T3D	1	8^4	27
	16	16^4	22
	64	$16^2 \times 32^2$	22
	128	16×32^3	22
SP2	1	$12^3 \times 6$	41
(Not Tuned)	4	$12^3 \times 24$	39
	8	$12^2 \times 24^2$	34
	16	12×24^3	31
	32	24^4	26

machines; this is annoying because we compute on a lot of platforms and like to be able to move projects around. This problem presumably arises because DEC does not believe in IEEE standards for binary words.

5. SPECTROSCOPY

The calculation of the masses of the strongly interacting particles is one of the major objectives of lattice gauge theory. Besides being an important test of quantum chromodynamics in its own right, a controlled calculation of the mass spectrum would also demonstrate that we are in a position to carry out reliable calculations of nonperturbative effects in QCD that are not as well determined experimentally as the masses. A computation of the low lying spectrum in the continuum limit for physical values of the quark masses is a prerequisite for calculations of strong and weak interaction matrix elements.

In order to measure the mass of a hadron which has some set of quantum numbers, invent an operator J which has the same set of quantum numbers and compute

$$C(t) = \langle 0|J(t)J(0)|0\rangle. \tag{4}$$

A little fiddling rewrites this as

$$C(t) = \sum_n |\langle 0|J|n\rangle|^2 \exp(-E_n t) \tag{5}$$

where $|n\rangle$ is an energy eigenstate with energy E_n, which at big t goes over to

$$C(t) \simeq |\langle 0|J|1\rangle|^2 \exp(-E_1 t) \tag{6}$$

where E_1 is the lightest state with the quantum numbers of J. The exponential falloff of the correlator gives us the mass, while its intercept gives us a matrix element $\langle 0|J|1\rangle$.

For bound states of quarks the operator $C(t)$ is basically the Feynman graph shown in Fig. 1: it is made of the appropriate number of quark and antiquark quarks propagating in the background of gluon fields in your album of configurations.

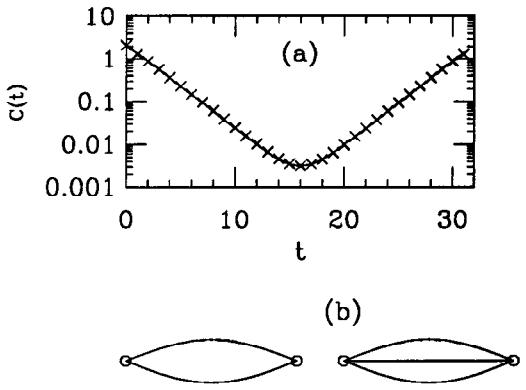

Figure 2. (a) A typical correlator showing good exponential falloff (the correlator has periodic boundary conditions in the time direction). (b) Feynman diagrams for meson and baryon correlators.

Generally in lattice calculations people try to deal with dimensionless quantities as much as possible, since they are independent of the precise value of the lattice spacing. In spectroscopy, people present their data on so-called "Ed-

inburgh plots," M_N/M_ρ vs. M_π/M_ρ. (The name is after the collaboration which invented the plot).

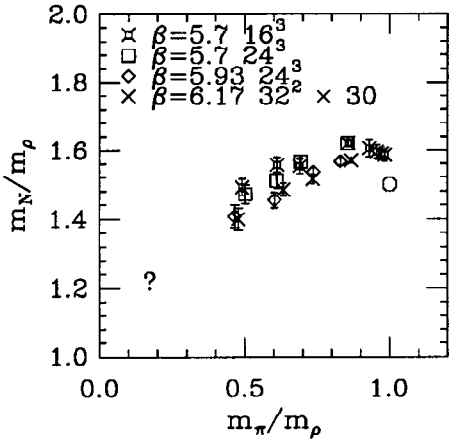

Figure 3. Edinburgh plot prepared by me from the data of Ref. 5. showing ratios at several values of the lattice spacing (different β's). The octagon shows the expected result at infinite quark mass, and the question mark is the real world value.

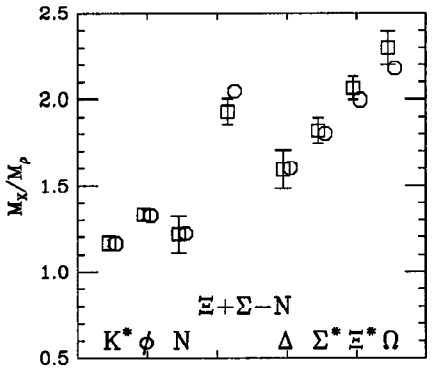

Figure 4. Ratio of lattice masses to the rho mass, after extrapolations to infinite simulation volume and zero lattice spacing, from Ref. 5. Circles are real world data, squares from simulations.

An interesting recent calculation is by a group from IBM which built its own computer to do QCD [5]. Fig. 3 shows their data plotted by me on an Edinburgh plot. There appears to be a small change between the data at larger lattice spacing ($\beta = 5.7$, about 0.14 fm) to the smaller lattice spacing ($\beta = 5.93$, 6.17, a down to about 0.07 fm). The authors of Ref. [5] have extrapolated their masses in a and L and present the limits in Fig. 4, as a plot of mass divided by M_ρ at $M_\pi = 0$. The agreement with observation is spectacular.

Steve Gottlieb [6] in our collaboration has recently finished a related set of calculations with a different lattice formulation of quarks. It is significant that calculations on essentially the same lattice volumes were done a few years earlier by another collaoration, but the data sets were small and the analysis was not done carefully enough. Gottlieb's thorough analysis was as necessary as the big computer was.

6. A CASE STUDY–HEAVY MESON DECAY CONSTANTS

The decay constant f_M of a pseudoscalar meson M is defined as

$$\langle 0|\bar{\psi}\gamma_0\gamma_5\psi|M\rangle = f_M m_M. \tag{7}$$

Decay constants are interesting because some of them (π and K) are measured and provide a benchmark for lattice calculations, while some of them are not measured and allow predictions (D, D_s, and B). They probe very simple properties of the wave function: in the nonrelativistic quark model

$$f_M = \frac{\psi(0)}{\sqrt{m_M}} \tag{8}$$

where $\psi(0)$ is the $\bar{q}q$ wave function at the origin.

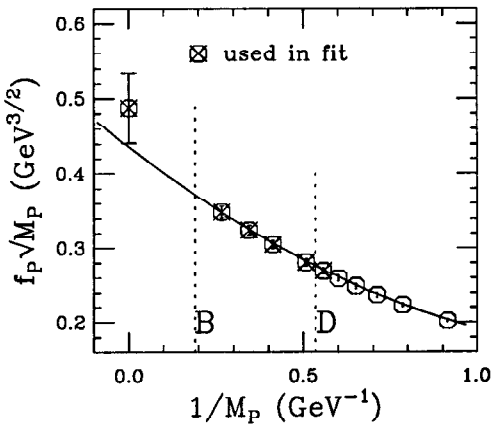

Figure 5. Pseudoscalar decay constant from the lattice.

One of my colleagues, Claude Bernard, has been doing the definitive f_D calculation at the Oak Ridge National Laboratory's Center for Computational Sciences (lattice volume $24^3 \times 80$), and at Indiana University (lattice volumes up to $16^3 \times 48$). Some of his results [7] are shown in shown in Fig. 5.

With the current data, he is able both to extrapolate in lattice spacing to the continuum limit and to control finite volume effects. In this way, all the important systematic errors in the quenched approximation can be controlled. A report of the preliminary results from this project was presented at Lattice-94. He found (in MeV):

$$\begin{aligned} f_B &= 147(6)(23); \quad f_D = 181(4)(18); \\ f_{B_s} &= 164(5)(20); \quad f_{D_s} = 195(3)(16); \\ \frac{f_{B_s}}{f_B} &= 1.13(2)(8); \quad \frac{f_{D_s}}{f_D} = 1.09(1)(4) \; . \end{aligned} \tag{9}$$

There are several other lattice predictions of these numbers. They differ in detail, but all give numbers in their range. There are two experimental measurements of f_{D_s}. They are $232 \pm 45 \pm 20 \pm 48$ MeV [8] or $344 \pm 37 \pm 52 \pm 42$ MeV [9]. The experimental error bars are so large that the lattice calculation is a prediction for experiment to verify or disprove.

7. HIGH TEMPERATURE QCD

Under normal laboratory conditions one does not directly observe the fundamental entities of QCD, the quarks and gluons. Instead one observes their bound states, the strongly interacting particles: the proton, the neutron and the scores of short lived particles produced by high energy accelerators. However, at very high temperatures or densities one expects to see a crossover or phase transition to a new state of matter, the quark–gluon plasma. Nuclear physicists hope to observe the plasma in heavy-ion collisions being planned for facilities such as RHIC and LHC. However, to make such an observation it is necessary to understand the nature of the transition and the properties of the high temperature state. The quark–gluon plasma may exist today in the cores of neutron stars, and it played a role in the evolution of the early universe prior to nucleosynthesis. Studies of high temperature QCD have an important bearing on other lattice gauge theory calculations because they provide a straightforward means of determining the lattice spacing at which one can safely extract continuum results.

The study of high temperature QCD in the vicinity of the crossover between the low and high temperature regimes is inherently a strong coupling problem, and at present it can be addressed from first principles only by lattice methods. The questions to be answered are the nature of the transition or crossover, the temperature at which it occurs, and the properties of the high temperature regime. We have been actively involved in addressing all of these questions.

During the past year one of our major MetaCenter projects has been the study of high temperature QCD with two flavors of staggered quarks on $24^3 \times 12$ lattices. We have performed simulations with quark masses 0.008 and 0.016 in lattice units (about 14 MeV and 28 MeV in physical units). These parameters correspond to a lattice spacing approximately 33% smaller than has been used in previous studies of full QCD at high temperatures. Thus, we believe that this work is a significant step in pushing the study of QCD towards the continuum limit. This project will yield an improved estimate of the crossover temperature between the low temperature regime of ordinary hadronic matter and the high temperature quark-gluon plasma. We also expect it to provide insight into the nature of the crossover and the properties of the plasma. In Fig. 6 we show a plot of the deconfinement temperature of QCD from our simulations.

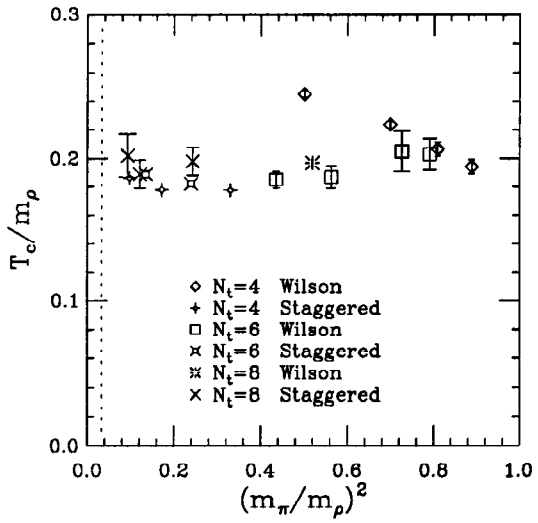

Figure 6. Deconfinement transition temperature (in units of the rho mass, 770 MeV) as a function of the pi/rho mass ratio, from lattice simulations with various formulations of quarks (Wilson or Staggered) and for various levels of discretization (the lattice spacing is $1/(N_t T_c)$). The real world is along the dotted line.

As a second project in high temperature QCD, we are doing a nonperturbative calculation of the equation of state for two flavors of staggered quarks. A determination of the energy and pressure as a function of temperature is important for understanding the formation of the quark-gluon plasma. It is also needed for phenomenological models of strongly interacting matter, and to calculate the hydrodynamic evolution of matter in the aftermath of relativistic heavy ion collisions.

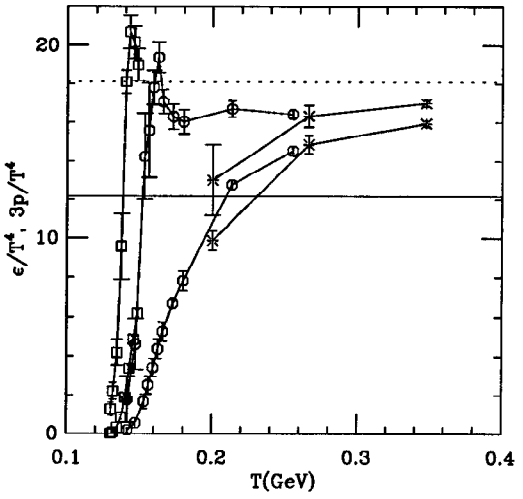

Figure 7. The equation of state for two flavor QCD as a function of temperature. The temperature is calculated from the rho mass. The energy density is obtained from the interaction measure and the pressure. The lower curves are three times the pressure. The octagons are for $am_q = 0.025$ and the squares are for $am_q = 0.1$. The errors contain the uncertainty in the β function. The bursts are extrapolations of the data to $am_q = 0$. At high temperature they approach the Stephen-Boltzmann law for noninteracting particles on a finite lattice (dashed line, the solid line gives the continuum value).

During the past year we carried out the first measurement of the equation of state of QCD with two flavors of quarks.[10] We found the pressure by numerically integrating derivatives of the partition function with respect to the bare parameters, $6/g^2$ and am_q. (g is the gauge coupling constant, m_q the quark mass and a the lattice spacing). These derivatives were measured directly on the lattice.

We began by performing a feasibility study at a rather large lattice spacing, $a = 1/(4T)$, that is at $N_t = 4$. (T is the temperature and N_t the number of Euclidean time slices of the lattice). Although this work already produced interesting results, it is clearly necessary to push these calculations to smaller lattice spacings in order to make contact with continuum physics. We have therefore begun a new series of simulations at lattice spacing $a = 1/(6T)$, that is $N_t = 6$. We are using a spatial volume of $N_s = 12$, and studying quark masses $am_q = 0.025$ and 0.0125. The strategy, as in the $N_t = 4$ calculation, is to perform simulations along lines of constant bare coupling at a few quark masses and extrapolate to $am_q = 0$. The phase diagram will then be filled in by running simulations along lines of constant am_q. Histogram reweighting can be used to smoothly cover the entire region.

8. SUMMARY

Present day lattice calculations are able to produce ten to fifteen per cent numbers for a wide variety of physical observables. Most of the uncertainties are systematics limited (at the cost of large amounts of computing to beat down statistics). The major systematic is the lattice spacing. It is just not understood how small the lattice spacing should be so that lattice calculations are insensitive to it (or more precisely, so that all physics on scales less than a are perturbative).

New supercomputers help in two ways. The obvious way is that they give us the resources to increase the lattice size, decrease the lattice spacing, and reduce the quark mass, in short, to more accurately model the real world.

However, in my opinion, our problems will not be solved simply with more powerful machines. The T3D (and the Paragon and anything else of the same power) are just not going to take us all the way. The most interesting new ideas to me, which might lead to improved calculations on smaller computers, are concerned with the question: Can one find a more complicated discretization which allows one to work at bigger lattice spacings? Doubling the number of terms in the lattice action roughly doubles the amount of work, while halving the lattice spacing at fixed simulation volume increases the work by a factor of 16. This subject is under active study [11]. New algorithms are tested on small systems, and most of my work has been done on (fast) work stations. I use supercomputers for turnaround, so that I can find out reasonably quickly whether my latest great idea is a failure or not. For that mode of research our code family is extremely convenient and our paradigm for parallel computing might serve as a model for other users.

I would like to thank my friends on the MILC Collaboration (who have actually done all the work listed here) for their insights and advice. This work was supported by the U. S. Department of Energy and by the National Science Foundation. Simulations were performed on the NERSC T3D and the San Diego and Indiana Paragons.

REFERENCES

1. M. Creutz, "Quarks, Strings, and Lattices," Cambridge, 1983. M. Creutz, ed., "Quantum Fields on the Computer," World, 1992.
2. The members of the MILC collaboration are: C. Bernard, Washington University, T. Blum, University of Arizona, T. DeGrand, University of Colorado, C. DeTar, University of Utah, S. Gottlieb, Indiana University, J. Hetrick, University of Arizona, U. Heller, Florida State University, N. Ishizuka, Washington University, L. Kärkkäinnen, Nordita, K. Rummukainen, Indiana University, R. Sugar, UC Santa Barbara, D. Toussaint, University of Arizona, M. Wingate, Uni-

versity of Colorado.
3. S. Duane, Nucl. Phys. B257, 652, 1985; S. Duane and J. Kogut, Phys. Rev. Lett 55, 2774, 1985; S. Gottlieb, W. Liu, D. Toussaint, R. Renken and R. Sugar, Phys. Rev. D 35, 2531, 1987.
4. S. Duane, A. Kennedy, B. Pendleton, and D. Roweth, Phys. Lett. 194B, 271, 1987.
5. F. Butler, et. al., Phys. Rev. Lett., 70, 2849, 1993.
6. S. Gottlieb, hep-lat/9412021, to appear in the Proceedings of Lattice '94.
7. MILC collaboration (C. Bernard, T. Blum, A. De, T. DeGrand, C. DeTar, S. Gottlieb, U.M. Heller, N. Ishizuka, L. Kärkkäinen, J. Labrenz, A. Soni, R. Sugar, and D. Toussaint) talk presented at the International Symposium, *Lattice '94*, Bielefeld, Germany, Sep. 27–Oct. 1, 1994, to be published in Nucl. Phys. **B** (Proc. Suppl.), Wash. U. HEP/94-37, hep-lat 9411080.
8. S. Aoki, et. al., CERN preprint CERN-PPE/92-157 (1992).
9. D. Acosta, et. al., Cornell preprint CLNS 93/1238 CLEO 93-14 (1993).
10. T. Blum, S. Gottlieb, L. Karkkainen and D. Toussaint, to be published in Nucl. Phys. B. (Proc. Suppl.).
11. Compare K. Symanzik, Nucl. Phys., B226, 187, 1983; P. Hasenfratz and F. Niedermayer, Nucl. Phys. B414 (1994) 785; P. Hasenfratz, Nucl. Phys. B (Proc. Suppl) 34 (1994) 3; F. Niedermayer, *ibid.*, 513. T. DeGrand, A. Hasenfratz, P. Hasenfratz, F. Niedermayer, U. Weise, hep-lat/9412058, talk presented at the International Symposium, *Lattice '94*, Bielefeld, Germany, Sep. 27–Oct. 1, 1994, to be published in Nucl. Phys. **B** (Proc. Suppl.).