FERMILAB-CONF-06-507-CD

National software infrastructure for lattice quantum chromodynamics

RC Brower[1], **CE DeTar**[2†], **RG Edwards**[3], **DJ Holmgren**[4], **RD Mawhinney**[5], **W Watson III**[3] **and Y Zhang**[6]

[1]Department of Physics, Boston University, Boston, MA 02215, USA
[2]Department of Physics, University of Utah, Salt Lake City, UT 85112, USA
[3]Thomas Jefferson National Accelerator Facility, Newport News, VA 23606, USA
[4]Fermi National Accelerator Laboratory, PO Box 500, Batavia, IL 60510, USA
[5]Physics Department, Columbia University, New York, NY 10027, USA
[6]Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599, USA

E-mail: brower@lns.mit.edu, detar@physics.utah.edu, edwards@jlab.org, djholm@fnal.gov, rdm@phys.columbia.edu, watson@jlab.org, zhang8@cs.uiuc.edu

Abstract.

Quantum chromodynamics (QCD) is the widely accepted theory of the strong interactions of quarks and gluons. Only through large scale numerical simulation has it been possible to work out the predictions of this theory for a vast range of phenomena relevant to the US Department of Energy experimental program. Such simulations are essential to support the discovery of new phenomena and more fundamental interactions.

With support from SciDAC the USQCD collaboration has developed software and prototyped custom computer hardware to carry out the required numerical simulations. We have developed a robust, portable data-parallel code suite. It provides a user-friendly basis for writing physics application codes for carrying out the calculations needed to predict the phenomenology of QCD. We are using this efficient and optimized code base to develop new physics application code, to improve the performance of legacy code, and to construct higher level tools, such as QCD-specific sparse matrix solvers.

We give a brief overview of the design of the data parallel API and its various components. We describe performance gains achieved in the past year. Finally, we present plans for further improvements under SciDAC-2.

1. Physics Goals

The goal of our research is to obtain a quantitative understanding of the physical phenomena encompassed by quantum chromodynamics (QCD), the fundamental theory governing the strong interactions of quarks and gluons. Achievement of this goal requires terascale numerical simulations. Such simulations are necessary to solve the fundamental problems in high energy and nuclear physics that are at the heart of the Department of Energy's large experimental efforts in these fields. The SciDAC Program "National Computational Infrastructure for Lattice Gauge Theory" is enabling U.S. theoretical physicists to develop the software and prototype the hardware they need to carry out terascale simulations of QCD.

† Presenter

Under this program our collaboration has developed efficient, core software for carrying out terascale simulations and is using this software in large scale production on US Department of Energy computational resources, including the special-purpose Brookhaven National Laboratory QCDOC and the Fermilab and Jefferson Lab cluster computing facilities.

2. Numerical Methods

Numerical methods for solving QCD have been under development for the past almost thirty years [1]. Over the past decade, algorithmic improvements have been nearly as important as hardware improvements in bringing numerical simulation to the stage that it is now having an impact on the analysis of experimental measurements. We expect that these impacts will grow in the coming decade.

Typical contemporary simulations represent the space-time continuum as regular hypercubic grid of several million or more points. The quark fields are represented on lattice sites as collections of three-dimensional complex vectors, Ψ, and gluon fields occupy the links between the lattice sites as three-by-three complex matrices U. The interactions of the quarks and gluons follow the rules of quantum chromodynamics, transcribed to the lattice. Contact with the real world is achieved in the limit of taking the lattice spacing to zero, while holding the overall physical volume constant.

To measure physically important observables, it is necessary to carry out a massive integration over all of the quark and gluon variables. This is done by Monte Carlo importance sampling methods. Underlying the widely used importance sampling method is a numerical molecular dynamics integration that moves the gluon variables around in the integration volume efficiently.

Each molecular dynamics step takes account of the influence of the quarks on the gluon variables and the influence of the gluon variables on themselves according to the rules of QCD. To calculate the influence of the quarks it is necessary to solve a large sparse matrix problem. Essentially we are solving a first-order matrix differential equation (the Dirac equation), but with a twist. All the finite differences include multiplication by the gluon field variable U, which takes on a different value for every link in the lattice:

$$[D\Psi]^\alpha(x) = \frac{1}{2a}\sum_{\beta,\mu}[U_\mu^{\alpha\beta}(x)\Psi^\beta(x+\mu) - U_\mu^{*\beta\alpha}(x-\mu)\Psi^\beta(x-\mu)] \quad \forall \alpha, x \tag{1}$$

The elementary operation for each site x involves taking the value of the quark field on the neighboring site in the forward μ direction, $\Psi^\beta(x+\mu)$, bringing it to the site x and multiplying it by the gluon matrix for the link between those sites, $U_\mu^{\alpha\beta}(x)$. This is called a parallel transport operation. The operation is repeated for all space-time directions μ forward and backward and the results are combined as shown.

Operations of the type described consume nearly 90% of the computer time in a typical simulation. The numerical operations are extremely uniform and simple. The problem lends itself readily to massively parallel computation. We distribute the space-time volume to the compute nodes in equal, regular subvolumes. The parallel transport operation described above involves passing values on the faces of the subvolumes between "neighboring" compute nodes. Communication can be naturally overlapped with computation.

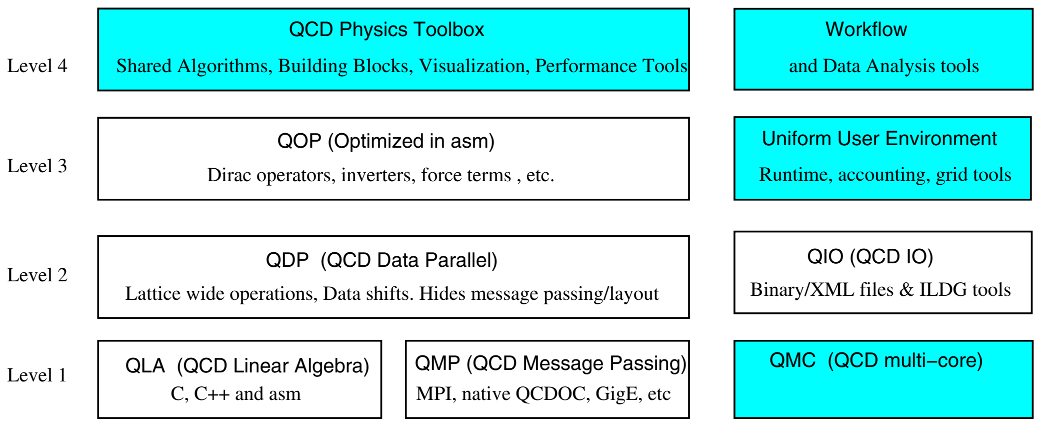
A hardware design with a regular mesh communication fabric suits the problem well. This design was the basis of the QCDOC, the specialized machine developed by Columbia University and IBM and installed at Brookhaven National Laboratory. It is currently one of our major computational resources. The design optimization point for the QCDOC has thousands of inexpensive processors with small local memory and an inexpensive mesh network. It is also the design basis for the IBM BG/L. Standard switch-based commodity cluster architectures also work well for this problem. The optimum design point in this case has faster processors with high processor to memory bandwidth, but still with comparatively modest memory requirements. Future optimum designs involve multicore processors.

The principal objective of software design for this problem is, then, to produce a data parallel API that hides the data locality and movement from the application programmer. Freed from those concerns, the scientist has more time to concentrate on the physics and has more flexibility to experiment and explore.

3. Software Design

The core software developed with SciDAC support is a multi-module three-level API tuned to the needs of lattice QCD, diagrammed in Fig. 1. These modules compartmentalize message passing (QMP), single-processor linear algebra (QLA), data parallel I/O (QIO) and data parallel linear algebra (QDP). They are the general-purpose building blocks for physics application code. The code is immediately portable to any MPP architecture with MPI. We have also written optimized versions of key components for specific MPP architectures, including the QCDOC. The QDP API is available in a standard C version (QDP/C) and in a powerful C++ version (QDP++). Examples of the data parallel API are given below.

The third level consists of widely used, critical utilities, including various sparse matrix solvers of the type described above. They are optimized for specific architectures. Optimization is achieved in the QCDOC implementation through use of hand-coded assembly language components. On clusters and the BG/L, the Level 3 routines have been implemented directly over QDP/C with the key underlying QLA routines being optimized in assembly language.



Level 4
QCD Physics Toolbox
Shared Algorithms, Building Blocks, Visualization, Performance Tools
Workflow
and Data Analysis tools
Level 3
QOP (Optimized in asm)
Dirac operators, inverters, force terms , etc.
Uniform User Environment
Runtime, accounting, grid tools
Level 2
QDP (QCD Data Parallel)
Lattice wide operations, Data shifts. Hides message passing/layout
QIO (QCD IO)
Binary/XML files & ILDG tools
Level 1
QLA (QCD Linear Algebra)
C, C++ and asm
QMP (QCD Message Passing)
MPI, native QCDOC, GigE, etc
QMC (QCD multi–core)

Figure 1. Multilevel API design. The uncolored blocks have been completed, except that new components of the Level 3 code are still being added. The blue-highlighted (or gray) blocks are planned for SciDAC-2.

All components of the software are available together with some documentation at `http://www.usqcd.org`.

4. Software Examples

The key data parallel linear algebra API is available in C (QDP/C) and C++ (QDP++). In both cases we require the ability to carry out a variety of linear algebra operations on a variety of data types. In most cases the operations and data types are QCD-specific. The operations include the parallel transport operation described above. We also needed the ability to work on a subset of the space-time lattice. The key parallel transport operation has two parts: a shift and a multiplication. The data parallel abstraction views the shift as a uniform movement from one lattice site to the next, even though the underlying operation requires communication when the required data is off node. The implementation of QDP and QDP++ involves both internode message passing (QMP) and on-node linear algebra (QLA). Similarly, data parallel I/O (QIO) deals with fields, which have values on every lattice site or link, and "global" data, which are constant over the entire lattice.

Because of the wide variety of possible linear algebra operations and datatypes, the QDP/C and QLA API consists of a library of thousands of procedure calls, only a small subset of which may be needed in a given project. In the QDP++ API this proliferation was avoided through the use of the portable expression template engine (PETE)[2, 3], a clever trick that causes the compiler to construct code for arbitrary combinations of lattice fields. In this way one avoids the common QDP/C need for temporary fields and multiple loops over sites. However, to achieve good performance with QDP++, it is often necessary to improve the underlying implementation with some hand optimization.

As an example of the API, consider the calculation

$$\psi^\alpha(x) = \sum_\beta U_\mu^{\alpha\beta}(x)\chi^\beta(x+\mu) + 2\phi^\alpha(x) \quad \forall\alpha,\, x \text{ even}, \tag{2}$$

similar in spirit to Eq. (1). The QDP/C code for this operation reads

```
QDP_V_eq_M_times_sV(temp, U[mu], chi, dir[mu], QDP_forward, QDP_even);
QDP_V_eq_r_times_V_plus_V(psi, 2, phi, temp, QDP_even);
```

The mnemonic procedure names are patterned after the arithmetic operation. Note that it is necessary to introduce a temporary field to complete the operation. In QDP++ the code reads

```
Psi[even] = U[mu]*shift[chi,mu] + 2*phi.
```

5. Past Year Performance Gains

Further development of the software and high level algorithms has brought huge gains in the productivity of our terascale resources. Here are some highlights of progress in the past year.

RHMC Algorithm An improvement in the molecular dynamics evolution process called the RHMC algorithm, developed recently by Clark and Kennedy [4] reduces the computational cost dramatically. This algorithm has already been implemented by the RBC Collaboration for their domain wall fermion calculation on the QCDOC [see RD Mawhinney, this conference]. Their productivity has increased by a factor of 5.4.

Level 3 Optimization for QCDOC We have continued to develop the QOP (optimized Level 3) API and add a couple more optimized routines. In the past year these routines have improved productivity on the QCDOC for the Asqtad lattice generation project by a factor of 1.9. That brings the cumulative contribution of all SciDAC improvements for this code to a factor of 3.8 for the QCDOC.

Level 3 Optimization for JLab and FNAL clusters and the BG/L As we mentioned, Level 3 optimization for clusters uses the QDP/C interface and assembly-coded optimization of selected underlying QLA routines. This work has led over the past year to an improved performance of a factor of 1.2 and a cumulative SciDAC improvement of 2.3 for a production-scale problem size. Using the same strategies as for clusters, we have achieved 1 TF on a single tower with both the Asqtad and domain wall solvers.

Cluster SSE optimization On a prototype of the soon-to-be-installed Fermilab dual-core AMD Opteron 275 Infiniband cluster with two processors per node, taking advantage of SSE instructions, we have achieved 2 GF per dual-core processor for the domain wall fermion application under production conditions [see Pochinski, this conference].

6. Objectives for SciDAC-2

The following are our major objectives under SciDAC-2

- Porting, optimization for leadership class machines
  We expect to achieve further performance gains on the BG/L and BG/P by writing a version of the message passing QMP specific to that architecture, as we have done for the QCDOC. Similarly, taking advantage of 32-bit SSE in QLA for the Cray XT3 Opteron will help.
- Exploitation of multi-core (SMP)
  Future architectures will rely on multicore processors. We will develop a new low-level API (QMC) to fully exploit their benefits.
- Common runtime environment
  To link the three major USQCD facilities into a practical meta-facility, we will unify file transfers, grid access, system environments, user environment, and bug tracking.
- Tool box (higher level shared algorithms)
  Several higher level utilities are in common use throughout the lattice community. They include the molecular dynamics evolution, eigensolvers, etc. Providing such building blocks will help facilitate rapid prototyping of new algorithms.
- Visualization and performance analysis
  Performance analysis tools will help in finding bottlenecks in software and hardware. Visualization tools will aid in understanding the physics.
- Work flow and data analysis
  A typical project involves moving hundreds of Gigabytes of lattice files to and from archival storage, running a measurement code on those files, and extracting and analyzing results of the measurements. Providing tools for managing the workflow and standard components of data analysis will facilitate the workflow and improve over all productivity.
- Full use of API by entire community
  To facilitate even wider use in the lattice community, we will develop more user-friendly documentation and dissemination of the code.
- Monitoring and control of large systems
  As our clusters continue to grow in size, we need to continue to develop automated fault monitoring and mitigation strategies

7. Cross-fertilization

Components of this project benefit from other large-scale computational science efforts. The NSF TOPS program supports, among other projects, the development of multigrid methods that could help us. The DOE-supported Particle Physics Data Grid project is developing file management tools that we will use. The International Lattice Data Grid project is designed specifically for sharing lattice files. It is using software we have developed and we will be using it to share files internationally.

[1] Thomas DeGrand and Carleton DeTar. *Lattice Methods for Quantum Chromodynamics.* World Scientific Co, Singapore, 2006.

[2] T. Veldhuizen. Expression templates. *C++ Report*, 7:26–31, 1995.

[3] Scott Haney. Is C++ fast enough for scientific computing? *Computers in Physics*, 8:690–694, 1994.

[4] M. A. Clark and A. D. Kennedy. The RHMC algorithm for 2 flavors of dynamical staggered fermions. *Nucl. Phys. Proc. Suppl.*, 129:850–852, 2004.