Institute of Physics Publishing Journal of Physics: Conference Series **16** (2005) 165–168
doi:10.1088/1742-6596/16/1/021 SciDAC 2005

# National software infrastructure for lattice gauge theory

**Richard C. Brower**

Physics Department, Boston University, 590 Commonwealth Avenue, Boston MA 02215

E-mail: brower@bu.edu

**Abstract.** The current status of the SciDAC software infrastructure project for Lattice Gauge Theory is summarized. This includes the the design of a QCD application programmers interface (API) that allows existing and future codes to be run efficiently on Terascale hardware facilities and to be rapidly ported to new dedicated or commercial platforms. The critical components of the API have been implemented and are in use on the U.S. QCDOC hardware at BNL and on both the switched and mesh architecture Pentium 4 clusters at Fermi National Accelerator Laboratory (FNAL) and Thomas Jefferson National Accelerator Facility (JLab). Future software infrastructure requirements and research directions are also discussed.

## 1. Introduction

The mission of the SciDAC software infrastructure project for Lattice Field Theory is to create a unified programming environment that will enable the U.S. lattice gauge theory community to achieve high efficiency on the computer architectures targeted by this project, the QCDOC, optimized clusters, and commercial supercomputers. It must enable users to adapt quickly to new (specially configured) architectures, to easily develop new applications and to preserve the large investment in existing codes.

The SciDAC QCD API provides a successful example of a well managed application specific code base serving a national research community. It is designed to exploits special features of QCD calculations that make them particularly well suited to massively parallel computers. For example the regular grids or lattices employed in Lattice field theory allow for perfect load balancing and regular and predictable communication patterns so their latency is hidden by overlapping them with local computation.

## 2. QCD Applications Interface

Our solution for the QCD software API takes the form of a 3 level structure:

- Level 1: Machine specific Message Passing and Linear Algebra routines
- Level 2: Data Parallel language to enable rapid code development
- Level 3: Highly optimized, computationally intensive routines.

All the fundamental components have been implemented and are in use on the US QCDOC hardware at BNL and on the Pentium 4 clusters at JLab and FNAL and the source code and documentation can be found at links from usqcd.org/usqcd-software.

 

*2.1. Level 1 Hardware Interface*

Level 1 provides the code that controls communications and the core per processor computation. For efficiency on dedicated Terascale facilities much of this layer maybe have to be written in the native hardware specific assembly language. However it is also implement in C and or C++ using MPI transparent portability of all source code.

**Message Passing:** QMP defines a uniform subset of MPI-like functions with extensions that (1) partition the QCD space–time lattice and maps it into the geometry of the hardware network, providing a convenient abstraction for the Level 2 data parallel API (QDP); (2) contain specialized routines designed to access the full hardware capabilities of the QCDOC network and to aid optimization of low level protocols on networks in use and under development on the clusters. Release 2.0 of the QMP message passing API is published on `usqcd.org/usqcd-software`, along with complete documentation. It includes: (1) a message passing library design and binding for both C and C++, (2) code implementing QMP atop MPI for portability and (3) an implementation atop VIA and gigabit ethernet to support the new gigabit ethernet mesh cluster at JLab. The implementation for the QCDOC has important hardware functionality, such as the ability to start twenty-four different communications with a single CPU instruction, and persistent storage in the communications hardware of the data pattern for repeated communications transfers. There is a basic test suite to verify each implementation.

**Linear Algebra:** The QLA routines can be used in combination with QMP to develop complex data parallel operations in QDP or in existing C or C++ code. The C implementation has on order 24,000 functions generated in Perl, with a full suite of test scripts. The number of functions in C++ implementation of QLA is considerable reduced by making extensive use of the language's class structure and of operator overloading. On the QCDOC assembly language coding for a small set of QLA routines has also led to a substantial boost in performance for MIC code. For the new C++ code base, optimized QLA routines have been generated using an assembler tool called BAGEL written by Peter Boyle from the UKQCD group in Edinburgh. We anticipate a similar strategy for other processors, such as the Altivec instruction set, which is available on the PowerPC(G5) architecture, if they prove to be a cost effective choice.

*2.2. Level 2 Data Parallel Programming Structure*

For easy of programming the vast majority of the high level application routines can be viewed as fully data parallel operations, hiding completely the detailed division of the lattice in to sublattices per processor and the data communication of surface elements between processors. To provide this abstraction to the programmer we provide a QCD specific data parallel "language". Technically it is not a new language since the interface is written entirely in C or C++ using standard compilers to generate all executable.

**Data Parallel Interface:** Level 2 (QDP) contains data parallel operations that are built on QMP and QLA. The C implementation is being used to improve performance for the MILC code and a new application code base Chroma has been written di nuovo on the C++ implementation, QCP++. To appreciate the concise nature of code written in QDP++, an algebraic expression,

$$\psi^i_\alpha(x) = U^{ij}_\mu(x)\chi^j_\alpha(x+\mu) + 2\phi^i_\alpha(x) \qquad \forall\, x \in \text{even and } \forall\, i,\alpha\ , \tag{1}$$

which might be found in code for the inversion of the Dirac operator translated into QDP++ code as,

```
psi[even] = u[mu] * shift(chi,mu) + 2 * phi;
```
(2)

This expression (and more complex variations) allow extensive overlapping of communication and computation in a single line of code. By making use of the QMP and QLA layers, the details of communications buffers, synchronization barriers, vectorization over multiple sites

| Quark Action | Mflop/s per node on $2^4$ | Mflop/s per node on $4^4$ |
|---|---|---|
| Wilson | 32% | 38% |
| Wilson–Clover | 32% | 47.5% |
| Domain Wall | 32% | 42% |
| Asqtad | 19% | 42% |

**Table 1.** Performance of QCDOC assembly code as a percentage of peak at 420 MHz for Dirac CG inverter for the three quark actions that will be used in initial projects on the QCDOC.

on each node, etc are hidden from the user. The [**even**] target label and **shift** communication operator are examples of completely general user defined subsets and permutation maps included in the API. The implementation makes heavy use of standard operator overloading, as well as employing "Expression Templates", a tool called PETE from LANL, that eliminates temporaries and enhances the performance.

*2.3. Level 3 High Performance Subroutines*

The overwhelming fraction of the floating point operations in any lattice QCD calculation are consumed in a few computationally intensive subroutines. Foremost among these are the subroutines for the inversion of the quark Dirac operators, which account for up to 90% of floating point operations in typical computations. Level 3 of the QCD API consists of highly optimized versions of these critical subroutines. They can be called both from QDP and from standard C/C++ code.

The Scientific Program Committee has identified three quark actions as vital to terascale QCD applications: WilsonClover, Domain Wall, and Improved Staggered (Asqtad). Inverters for these three quark actions constitute the first set of Level 3 routines. They are written in assembly language for the QCDOC. The critical part of these routines is the multiplication of a vector by the Dirac operator. Table 1. below shows the performance of this operation as percentage of peak obtained on the initial QCDOC hardware. The differences in performance have to do with the different ratios of floating point operations to data movement for the three actions. The Asqtad action puts a significantly higher demand on the communications system than the other actions, so its performance on such a small local volumes is particularly noteworthy.

Level 3 codes for the 2.8 GHz Pentium 4 processor with a 800 MHz front side bus written in SSE2 instruction exceed 1 Gflop/s per processor.

## 3. Future Directions

The basic QCD API is set, and high performance code built upon it is ready for use on the QCDOC and clusters. To take full advantage of the API additional emphasis must now be placed on integration of these components and the development of the full environment of application codes and a common set of execution and data management tools. In addition more resources should be devoted to research into algorithms and their optimization on current and future hardware platforms.

**Software Integration:** Chroma, the first large body of code written in QDP, must be completed and optimized; additional components of the MILC code should be ported to QDP; and new applications need to be written. A full integration of application codes with common the API and common standards for the level 3 interface and file formats is required to allow for easy interchange of application modules. The development of new formulations of QCD on the lattice has played an important role in advancing our field in recent years, and additional ones will surely be introduced in the near future. Each new formulation will require a new Level 3

inverter. In addition, for some formulations, subroutines other than the inverter use significant numbers of cycles, so Level 3 code may need to be written for them as well. Growing use of the QCD API by members of our community will require additional user support, documentation, and revision control.

As new hardware comes into use, it will be necessary to optimize the QMP and QLA libraries for it, and to develop new Level 3 routines. One example for the near future is the IBM BlueGene/L, which appears to be a very promising platform for the study of QCD. QDP++ and Chroma are already running on it, and we plan to optimize the performance of QMP and write Level 3 routines for this platform. To assist in our optimization efforts, our computer science colleagues in Dan Reed's group at North Carolina have developed a high level performance analysis toolkit for the MILC code. We plan to extend this tool to other major codes.

**Uniform Execution and Data Management:** Dedicated hardware for lattice QCD will be located at BNL, FNAL and JLab. We plan to build a unified user's environment, presenting to the users identical batch environments, identical commands for interacting with disk and tape resources, and identical development environments. As this effort matures, the BNL, FNAL and JLab facilities will be operated as a single meta-facility, including data grid capabilities, and virtual batch queues for job submission. In this effort we will make use of the grid tools developed within the Sciatic Program by groups such as the Particle Physics Data Grid.

A very large fraction of the computing resources used in lattice QCD go into Monte Carlo simulations that generate representative configurations of the QCD ground state. The same configurations can be used to calculate a wide variety of physical quantities. Because of the large resources needed to generate configurations, the U.S. lattice community has agreed to share all of those that are generated with DOE resources. To enable this sharing we have created standards for file formats, and built into the QCD API I/O routines that adhere to them. Thus, all members of our community will be able to access the large data sets we plan to create and archive. We are charter members of the International Lattice Data Grid (BLDG), which seeks to share QCD data internationally. To do so will require the adoption of common standards for data files, agreements on sharing data, and the establishment of an international grid to facilitate transfer of the data. There is significant work yet to be done, but the payoff will be very large, enabling all workers in lattice QCD to greatly enhance their research.

**Algorithmic Research:** If past history is a guide, new algorithms will be as important as faster hardware in advancing research in lattice QCD. Consequently the software infrastructure must be flexible enough to accommodate the evolution of QCD algorithms. For many years the central algorithmic problem faced by our field has been the inversion of the Dirac operator, a very large sparse matrix. We need both improved algorithms for existing applications, and radically new approaches for problems outside the reach of current methods, such as simulations at finite chemical potential. These problems pose fundamental mathematical challenges with strong relations to analogous problems in other areas of science and applied mathematics. Sciatic offers an ideal setting for this type of algorithmic research by encouraging interdisciplinary collaborations.

The use of multigrid methods to accelerate the inversion of the Dirac operator has been explored extensively in the past without significant success. However, members of our group have recently begun working with applied mathematicians from the TOPS multigrid algorithm team on this problem, and have obtained impressive preliminary results. It is very important to continue this work, as even modest gains in performance would have a major impact on the science. In addition, Lüscher has recently introduced a blocking method based on the Schwarz alternating procedure that shows promise. This approach too warrants further exploration.