The Secret Life of Quarks

Final Report, University of North Carolina at Chapel Hill.
DE-FC02-06ER41445, October 2006 – March 2012.

Robert J. Fowler
RENCI, University of North Carolina at Chapel Hill
(rjf@renci.org, 919.445.9670)

I. Overview

This final report summarizes activities and results at the University of North Carolina as part of the the SciDAC-2 Project *The Secret Life of Quarks: National Computational Infrastructure for Lattice Quantum Chromodynamics*.

The overall objective of the project is to construct the software needed to study quantum chromodynamics (QCD), the theory of the strong interactions of subatomic physics, and similar strongly coupled gauge theories anticipated to be of importance in the LHC era. It built upon the successful efforts of the SciDAC-1 project *National Computational Infrastructure for Lattice Gauge Theory*, in which a QCD Applications Programming Interface (QCD API) was developed that enables lattice gauge theorists to make effective use of a wide variety of massively parallel computers. In the SciDAC-2 project, optimized versions of the QCD API were being created for the IBM BlueGene/L (BG/L) and BlueGene/P (BG/P), the Cray XT3/XT4 and its successors, and clusters based on multi-core processors and Infiniband communications networks. The QCD API is being used to enhance the performance of the major QCD community codes and to create new applications. Software libraries of physics tools have been expanded to contain sharable building blocks for inclusion in application codes, performance analysis and visualization tools, and software for automation of physics work flow. New software tools were designed for managing the large data sets generated in lattice QCD simulations, and for sharing them through the International Lattice Data Grid consortium.

This effort was a project of the USQCD Collaboration, which consists of nearly all the high energy and nuclear physicists in the United States engaged in the numerical study of QCD. The Collaboration's web page www.usqcd.org contains information regarding its scientific objectives, membership, and initiatives. All software developed under this grant is publicly available, and can be downloaded from a link on the USQCD web site, or directly from the URL usqcd.jlab.org/usqcd-software. Ten universities and three national laboratories were funded on the project.

As part of the overall project, researchers at UNC were funded through ASCR to work in three general areas. The main thrust has been performance instrumentation and analysis in support of the SciDAC QCD code base as it evolved and as it moved to new computation platforms. In support of the performance activities, performance data was to be collected in a database for the purpose of broader analysis. Third, the UNC work was done at RENCI (Renaissance Computing Institute), which has extensive expertise and facilities for scientific data visualization, so we acted in an ongoing consulting and support role in that area.

This project paid for approximately 0.3 of an FTE staff member per year at UNC over most of its lifetime. Beginning in September 2011 the level of effort and spending rate was progressively reduced to permit continued participation in SciDAC QCD activities through the no-cost extension period.

II. Results and Findings

Throughout the course of the project UNC personnel served on the USQCD Software committee. This participation included active participation in meetings, both weekly telecons as well as semi-annual face-to-face project conferences. These interactions formed the basis for UNC personnel to act in a Computer Science consulting role on a variety of topics and to serve as a liaison between USQCD and the broader Computer Science research community.

During the first year of the of the project, computer scientists at the University of North Caroline focused on designing and developing a web based High Performance Computing (HPC) Performance Database targeted particularly on QCD applications. It was used for performance profiling of the latest releases of MILC code. The HPC Database is a web based infrastructure designed to store the performance data collected by our performance team at Renaissance Computing Institute (RENCI) for QCD applications on various high performance computing systems. It provided web interfaces for users to browse the performance data, perform statistical analysis and conduct performance comparisons. The goal was to create a knowledge base for maintaining and sharing the performance analysis results among the QCD community. The UNC team completed an initial implementation of the system. As of the end of the year, the database contained the performance files collected by running SvPablo instrumented MILC on several HPC systems during the year of 2006. All the performance data can be browsed via the web interface. In addition, two simple web queries are provided for fine grained data search. With partial funding from SciDAC PERI (Performance Engineering Research Institute) project, the HPC Database was merged with PERI Performance Database as part of PERI's Application Engagement effort. The current version of this is the TAU DB supported by the University of Oregon.

The database was used to collect results for a study of the performance scaling properties of MILC. The performance profiling was done using SvPABLO to instrument the FOR_ALL_SITES loops and the contained SU(3) operations in MILC using hardware performance counters to get definitive measurements of the effects of memory bandwidth and latencies of the on-node calculations. This was done on several classes of system, including a BlueGene/L, an SGI shared memory system, and several Linux clusters. The goal was to measure the performance "headroom" and to identify restructuring methods, either manual or compiler-based, to improve on-node performance and to reduce the performance "drop off" as the local domain size increases beyond the size of cache. This methodology separated the performance of phases that are constrained by on-node activity and nearest neighbor communication from from phases dominated by all-to-all communication. The general finding was that the nearest neighbor aspects scale linearly on all systems investigated. The on-node activities were found to be severely constrained by the system memory cache size, thus causing severe performance degradation for local problem sizes that don't fit in level 2 cache. The Krylov method solvers used for matrix inversion exhibit comparatively poor scaling with large numbers of processors because they eventually dominated by all-to-all communication. One conclusion is that even without changing the solver efficiency could be improved by mitigating

the "cache size cliff" to allow larger local sub-problems to be done efficiently. The results were shared with the MILC community. While the qualitative nature of the results were expected, this work helped to quantify the magnitude of the problems and to emphasize trade-offs and areas for improvement. Increased attention was paid to loop ordering, e.g., the Morton ordering used in the MIT Moebius inverter, and to tiling issues. It also helped to quantify the advantages of pursuing new inverter methods through the remainder of this project and its follow-ons.

One goal of this scaling study was to use measurements on multiple systems to quantify the relative importance of CPU and memory performance for QCD applications. In the end, CPU and memory performance, both latency and bandwidth, were all on co-linear curves for the systems available at the time.

The results were written up and circulated in Zhang *et al.* "Performance Characterization for HPC Systems and a QCD Application".

The HPC Performance Database was offered to the physics community to track performance of QCD applications. While it was found to be useful for the scaling study, for a variety of reasons it was not embraced by the physics community. Aspects of the database continue to be used in SciDAC PERI and SciDAC SUPER in the form of the TAU DB.

In the second year of the project (2007-2008), Zhang worked with the MILC collaboration to use the MILC code as one of the major components of the PERI performance database work, which is being used to drive auto-tuning efforts.

A second major effort was performance measurement, analysis, and tuning activities for the QDP++ and Chroma code base in coordination with with SciDAC PERI. This coordination benefited the QCD project by bringing to bear other researchers within PERI. It benefited PERI by presenting a set of challenging research problems not addressed by existing performance tools.

Chroma uses the expressive capabilities of the C++ programming language more aggressively than most other high performance scientific codes. Chroma was thus used to drive the extension of the analysis capabilities of HPCToolkit from Rice University.

QDP++ and Chroma are written using very expressive and powerful modern idioms in C++, specifically template meta-programming and expression templates. This is a very powerful approach that enables complex calculations to be specified in a concise manner, thus improving programmer/scientist productivity. The down side from the performance analysis perspective is that a single source line statement can expand through inlining and template into the equivalent of many hundreds of lines of code and can implicitly invoke C++ methods that recursively expand into many lines of code. Furthermore, after inlining, subsequent compiler optimization passes may reorder object code. This is important for efficiency, but it allows only very coarse grain performance analysis. Inserting instrumentation at the source level impairs performance by inhibiting inlining and other optimizations. Furthermore, instrumentation in inner loops can itself be very expensive. Sampling-based performance profiling tools can avoid the performance penalty, but needs a robust mechanism for attributing costs in optimized object code back to the original source code. Using driving problems from JLab, UNC and Rice addressed the issue of adding the necessary algorithms to the HPCToolkit performance tool to derive such attribution. This work has produced a masters

thesis (Nathan R. Tallent, *Performance Analysis of Optimized Code: Binary Analysis for Performance Insight*) and in two conference papers (Tallent *et al. Binary analysis for measurement and attribution of program performance.*, and Tallent em et al. *Diagnosing performance bottlenecks in emerging petascale applications.*) The work at UNC on this effort was funded through the QCD SciDAC project and the work at Rice was funded through SciDAC PERI.

RENCI personnel (Fowler and Porterfield) worked with JLab on the issue of multi-core performance of LQCD codes. Aspects of the analysis of threading strategies for a DSlash kernel on quad-core AMD chips, such as those used in the ORNL Jaguar (and other Cray XT systems) were reported at SciDAC 2008 in the form of a poster and a journal paper (Fowler *et al.*. *Performance engineering challenges: the view from RENCI.*) This entailed a comparison of three strategies for multi-core parallelism: "MPI everywhere". "MPI and OpenMP 2.5", and "MPI and QMT", where QMT is a threading library written at JLab specifically for QCD computations. Experimental results indicated that the "MPI everywhere" and "MPI and QMT" were competitive, but that "MPI and OpenMP" was considerably worse. We determined that relatively poor performance of OpenMP relative to QMT is due to cores going into and out of a low power mode when they are idle rather than simply going into an active idle loop. Version 3 of the OpenMP standard allows applications to explicitly request "busy waiting", so the comparative advantage of QMT may be diminished when OpenMP v.3 becomes widely available.

The results on synchronization discovered through studying Chroma were further applied under the aegis of SciDAC PERI to improve the performance of Robert Harrison's MADNESS chemistry code at ORNL. UNC and Rice worked on generalizing the analysis methods for diagnosing such problems. See (Tallent, Mellor-Crummey, and Porterfield. *Analyzing lock contention in multithreaded applications.*) for a description of some of the analyses and tool improvements.

Part of UNC's work plan for the project was to exploit RENCI's extensive scientific visualization resources and expertise. During this project year Dreher, Borland, and Fowler from RENCI worked with DiPierro of DePaul to produce prototype visualization capabilities. DiPierro has continued the work on his own with occasional consultation from UNC.

During the 2008-2009 project year, work at the University of North Carolina under the project continued to be focused primarily on performance measurement, analysis, and tuning activities in collaboration with the SciDAC Performance Engineering Research Institute.

During this project year we used UNC internal funding to supplement our SciDAC funding to permit us to continue our collaboration with Massimo DiPierro of DePaul on visualization.

The work on multi-core performance continued during the 2009-2010 project year. Because multi-core, multi-socket performance is often dominated by bottlenecks at resources shared between cores, it is necessary to use measurement and analysis methods that expose contention at those resources. Almost all existing performance tools using hardware performance counters are based on a "first person" perspective in which each process or thread executes performance instrumentation code that accesses virtualized hardware performance counters that are swapped when control is shifted between processes (threads). On most systems the operating system does not support the virtualization of performance counters that monitor off-core resources so these tools cannot provide the necessary information. Some systems do allow virtualized counters to view off-core

resources, but the resulting measurements are nearly impossible to interpret. Since each process observes the shared resource only when it is scheduled to run, no application process captures all of the events at the resource. On the other hand, during time intervals when multiple processes run concurrently on several cores, each event is redundantly counted by all the active processes. The resulting counts are of little use.

To meet these deficiencies, under the SciDAC PERI project UNC began development of an approach we call "Resource Centric Reflection" (RCR) to use a third-person approach to produce node-wide measurements of shared resources, to compute real time analyses of this information to translate raw counts into more useful quantities, and to share this information with system and application programs, including first-person performance tools such as the University of Oregon's TAU and Rice University's HPCToolkit. Based on this approach UNC researchers developed methods for characterizing the degree of memory concurrency supplied by a system (See Mandal *et al.*, 2010.) and under this project used the method to analyze the offered load from LQCD codes. We determined that for the generation of multi-core chips then in use that the QCD codes are memory latency bound, but that if the number of cores were to be increased, they would soon be limited by memory bandwidth unless there were dramatic changes in memory hierarchy design.

In the final year of the original plan (2010-2011) UNC continued to address issues of performance measurement and analysis on multicore and manycore systems in the context of QCD codes.

The UNC group participated in in the discussions regarding GPU algorithms and in the configuration strategies for the hardware that was deployed at JLAB. In support of this, we procured a standalone system with multiple GPUs and began development work on integrating multi-GPU performance monitoring with our framework for monitoring multi-core, multi-socket servers. The tool work done under a non-disclosure agreement with Nvidia that gave UNC access to their CUPTI performance interface to CUDA. Similar work was done at the University of Tennessee Knoxville and Rice University, but the agreements with Nvidia prevented us from sharing information and code among ourselves until CUPTI was made public after the end of this project.

The SciDAC multi-GPU lattice code library QUDA was the main driving application for starting the development of a comprehensive performance measurement environment to integrate Nvidia CUPTI measurements, first-person CPU call-stack profiling using Rice's HPCToolkit, and node-wide bottleneck analysis using UNC's RCRTool. This effort saw mixed success. The effort to get RCRToolkit and RCRTool to interoperate was successful and an analysis of the Chroma code was reported in (Mandal *et al.,* 2012). The versions of CUPTI available during this period, however, were limited to producing only relatively coarse measurements of GPU codes and required that the GPU code run synchronously with the CPU, *i. e.,* the CPU and GPU could not operate simultaneously. This was judged unacceptable for our purposes, so we were forced to abandon this line of development.

Midway through the final year, the participating institutions began to prepare for the end of the SciDAC 2 project and to prepare for SciDAC 3. Anticipating a gap between the programs and under advice from DOE ASCR, the institutions requested six month no-cost extensions and prepared a proposal for a funded extension year. UNC received a six month unfunded extension from March 2011 to Sept 2011. This was extended by another six months to March 2012. While some of the project partners were funded for the full year, UNC did not receive new additional funding.

In response, to keep the collaboration active, beginning in Sept. 2010 we progressively reduced our level of effort, thus stretching the remaining funds to permit continued participation in meetings and technical consultation with the QCD community.

During this period UNC continued to perform specific performance experiments in support of USQCD code developers, particularly with respect to details of memory hierarchy behavior and of the pipeline efficiency of the low level SU(3) operators used in LQCD calculations. The paper (Mandal *et al.,* 2012) uses one aspect of the memory behavior of Chroma as an example.

Discussions and performance experiments conducted during the extension year were also driven by the need to identify long-term strategic problems facing LQCD code development for emerging and future systems. As we move into the exascale era, the number of cores per chip will continue to increase while other aspects of the system such as available power, off-chip memory bandwidth, and interconnect bandwidth will be stagnant or will grow much more slowly. Such architectures will require highly-parallel code that can be vectorized to use pipelined, streaming functional units. The QUDA library for Nvidia GPUs is an example of software that has pioneered the structuring of code to have these properties. The methods used here were recognized by at least part of the USQCD community as being needed on other new system architectures such as the Intel Many Integrated Cores (MIC) and on main stream chip such as the Intel Xeon as well.

In contrast, the QDP++ library can be viewed as a very powerful data parallel, application specific language that has improved scientific productivity by making final application development much easier. Unlike other data parallel languages, such as High Performance Fortran, that were based on cutting edge compiler techniques, QDP++ is implemented using extensive C++ template meta-programming, including the use of expression templates. While this approach can generate relatively good code at the statement level, it cannot apply inter-statement, outer-loop, or global analyses and code generation methods the way a conventional compiler can. In particular, it cannot directly generate the kinds of streaming codes needed to get the best performance for GPUs and other emerging architectures.

Our discussions and groundwork during this period about future approaches to compilation for domain-specific QCD languages influenced the design of the QLUA code being developed at MIT. Through our interactions with USQCD researchers at JLab and Frank Winter of the University of Edinburgh, it has influenced the development of QDP-JIT. It also provided the basis for the strategy for code generation we are taking in the SciDAC-3 Project "Computing Properties Of Hadrons, Nuclei And Nuclear Matter From Quantum Chromodynamics" under funding agreement DE-SC000876.

Publications and Presentations

1. Robert J Fowler, Todd Gamblin, Allan K Porterfield, Patrick Dreher, Song Huang, and Balint Joo. "Performance engineering challenges: the view from RENCI" J. Phys: Conf. Ser, page 5pp, October 2008.
2. Nathan R. Tallent, John M. Mellor-Crummey, Laksono Adhianto, Michael W. Fagan, and Mark Krentel. Diagnosing performance bottlenecks in emerging petascale applications. In SC '09: Proc. of the 2009 ACM/IEEE 2009. ACM. (doi:10.1145/1654059.1654111).

3. Nathan R. Tallent, John Mellor-Crummey, and Michael W. Fagan. Binary analysis for measurement and attribution of program performance. In PLDI '09: Proc. of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation, pages 441-452, New York, NY, USA, 2009. ACM. Distinguished Paper. (doi:10.1145/1542476.1542526).

4. Nathan R. Tallent, John M. Mellor-Crummey, and Allan Porterfield. Analyzing lock contention in multithreaded applications. In PPoPP '10: Proc. of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pages 269-280, New York, NY, USA, 2010. ACM. (doi:10.1145/1693453.1693489).

5. Anirban Mandal, Rob Fowler, and Allan Porterfield. Modeling memory concurrency for multi-socket multi-core systems. In Proceedings of the 2010 IEEE International Symposium on Performance Analysis of Systems and Software, White Plains, NY, March 2010. IEEE.

6. Anirban Mandal, Robert Fowler, and Allan Porterfield. System-wide introspection for accurate attribution of performance bottlenecks. In Workshop on High-performance Infrastructure for Scalable Tools (WHIST), Venice, Italy, 2012.

7. Ying Zhang, Kevin Gamiel, Mark Reed, Morten Somervoll, Jeffery Tilson, and Daniel Reed. Performance Characterization for HPC Systems and a QCD Application, 2007. (Unpublished Report).