TO: DOE Office of Nuclear Physics

DOE AWARD NUMBER: DE-SC0008706

PROJECT TITLE: Computing Properties of Hadrons, Nuclei and Nuclear Matter from Quantum Chromodynamics

PRINCIPAL INVESTIGATOR: Robert Fowler

ADDITIONAL PERSONNEL: Allan Porterfield, Diptorup Deb, Priadarshi Sharma

RECIPIENT ORGANIZATION: The University of North Carolina at Chapel Hill

RECIPIENT ACCOUNT NUMBER: 5-37359
PROJECT/GRANT PERIOD: August 15, 2012 – February 14, 2018

REPORTING PERIOD END DATE: February 14, 2018

REPORT TERM: Final Technical Report.

Abstract

This project served to coordinate the software development effort which the nuclear physics lattice QCD community is developing to ensure that lattice calculations can make optimal use of forthcoming leadership-class and dedicated hardware, including those of the national laboratories, and prepares for the exploitation of future computational resources in the exascale era. The project developed and optimized simulation software for lattice QCD calculations on leadership class computers and on GPU-accelerated heterogeneous architectures. The participating team at multiple institutions improved and extended software libraries (Chroma, CPS, MILC, QLUA, QPhiX, QDP++, QDP-jit) by providing interfaces for heterogeneous computing environments and through the optimization of gauge field evolution algorithms and sparse matrix inverters. The former includes the development of new multi-time scale integrators and the latter focused on the development of new multi-grid and domain decomposition inverters for several QCD fermion discretization schemes. The sub-project at the University of North Carolina worked towards the development a domain specific language for lattice QCD computations with the express purpose of generating code with uncompromised performance.

Group Overview

The stated project plan for the UNC component of the project is for computer scientists at UNC/RENCI to work with lattice field theorists at Jefferson Lab to increase scientific productivity and computational efficiency through the use of SUPER Institute compiler and performance analysis tools to provide efficient implementations of QCD-specific abstract programming languages and libraries. The two goals of this effort are to improve the productivity of lattice theorists while simultaneously achieving an uncompromised level of execution performance that will be portable across the widest variety of platforms including conventional cluster architectures, heterogeneous systems with many-core accelerators, and the power efficient systems that will emerge on the way to extreme scale.

While a large fraction of the USQCD software infrastructure has been extensively hand-tuned to execute efficiently on high-end systems, other parts are directed more at flexibility and modularity for exploring new mathematical methods and physics models. This is important because these new methods can represent orders of magnitude improvements in performance, while further improvement to already well-tuned code faces diminishing returns. Libraries such as Chroma and QDP++, as well as QLUA, support the prototyping of such new methods using a high-level of flexible abstraction for some parts of the program. For some well-studied, core calculations, these frameworks can use extremely well-tuned modules on CPUs as well as GPUs. For new methods, and new physics, however, such modules are not available, resulting in an order of magnitude loss in performance. This has creates what is sometimes called the “ninja programmer gap” between the prototype code and the packaged modules. Our stated goal in this project was to help diminish this gap.

Research Program

Review of Plan

Once the project was funded, two major forces affected our plans. First, the actual funding level was approximately 65% of what was requested in the proposal, requiring us to become far more parsimonious in the level of effort. Second, the physics and software priorities of the Chroma/QDP++ group at JLlab evolved between the writing of the proposal and the start of the project.

The proposal for this project proposed that the funds be primarily used to support a post-doctoral fellow at UNC who would spend approximately 40 percent of his/her time at JLab. The proposed budget had funds to support this travel as well as to pay senior personnel to participate in the project, primarily through the supervision of this postdoc. As funded, the project could support one graduate student and approximately 15 percent of a senior researcher. This resulted in both reduced scope and slower progress than was proposed.

There has been an increasing emphasis in the LQCD community on heterogeneous computing using NVIDIA CUDA GPUs . These devices are a significant part of the systems purchased specifically for LQCD, and several large shared facilities funded by DOE and NSF now feature NVIDIA devices. Furthermore, the QUDA library for utilizing CUDA devices for the solver phases of LQCD applications has matured and become more scalable. Another trend was the emergence of the Xeon Phi architecture from Intel. A number of large-scale systems have been based on Phi and in response Intel and Jefferson Lab coordinated on the development of QPhiX, a package for the optimized solution of the Wilson Dslash and Clover operators and for LQCD solvers based on these operators. A consequence of these developments is that applications written in CHROMA/QDP++ using these packages run much more efficiently, with some examples reporting only 20 to 25 percent of the execution time in the solver. By Amdahl's law, this success means that there will be diminishing returns in these applications from further improvements to the solvers and that attention needs to be focused on improving the other phases of the application. Accelerating those other phases by automatically generating code for NVIDIA devices (also emerging hardware such as the Intel Xeon Phi) from within the CHROMA/QDP++ framework is thus now one of the major goals of the JLab group. Beginning in the Spring of 2013, work at UNC therefore shifted towards aiding JLab in these efforts by doing performance analyses and tuning in the short term while narrowing our goals for applying advanced compiler methods to the problem of efficiency of generated code on accelerator devices.

At about the time the project was funded, JLab began to recruit Frank Winter for a staff position rather than as a post-doc. He joined JLlab in Spring 2013. In the meantime, he began work on QDP-jit, an

offshoot of QDP++ that uses the expression template mechanism to generate code which, when evaluated at run time, instead of evaluating the expression writes code to do an efficient evaluation when compiled or assembled for a GPU. The original version sent the source code to a file to be compiled and later dynamically linked and executed. The current version creates Nvidia PTX abstract assembly code that is assembled on the fly by the Nvidia CUDA driver. QDP-jit uses a software-controlled cache approach to help control the data traffic between the accelerator and the host machine.

QDP-jit shares the same strengths and weaknesses as other expression-template-based approaches. Each individual NVIDIA kernel generated this way can perform very well locally, but there is still no global optimization phase that can do inter-expression optimization. When generating code that runs on a host, this is manifested as increased memory traffic on the host and when generating code for an accelerator it is manifested as memory traffic on the device as well as traffic between host and device. Using mainstream compiler technology to ameliorate this problem through the use of high-level code transforms is therefore our number one priority in the short term.

The original plan was to use the Rose framework from LLNL for our compiler work. While Rose is still an excellent infrastructure for research on domain-specific language development, in 2014 Nvidia has changed its CUDA development environment to use CLANG-LLVM. Other vendors have also moved to LLVM. Our physics partners in the USQCD community have also embraced this trend. Most specifically for this effort, JLab is using CLANG for QDP-jit for GPUs and they are investigating using it for code generation on other important platforms. QDP-jit is also starting to rely on features of C++ version 14, CLANG's support for these features. Also CLANG is now capable of compiling and manipulating expression templates generated in Chroma and QDP++. Thus, to utilize these capabilities and for the sake of compatibility within this project we revisited our previous decision to use Rose and are now using CLANG-LLVM.

Research Results

Using SUPER institute performance tools, in the first 18 months of the project we analyzed the performance tradeoffs between performing lattice operations such as the Wilson DSlash operator in a single large and cumbersome expression versus factoring them into a sequence of more manageable and readable steps. We observed run-to-run variability in nominally identical calculations. We determined that the sources of this variability included both differences in the efficiencies in cooling nodes in an air-cooled cluster as well as CPU die-specific variability that enables nominally identical chips to be run at different voltages, clock rates, and power levels. These studies led to the publication of several papers on the sources and methods to control the variability with researchers from other projects.

For the case of QDP-jit generating Nvidia kernels, we determined that there was very little difference in the total cost of executing the kernels using the different strategies, but there is a significant difference in the cost of data transfer between the GPU and CPU. Specifically, in the multi-expression case intermediate results are copied back from GPU to CPU regardless of whether they will in fact be needed. While the software caching mechanism in QDP-jit avoids redundant copies from CPU to GPU, without

some form of static compiler-based data dependence analysis it must assume that anything computed will possibly be needed on the CPU. The significance of this is that it relieves the programmer from such concerns and allows her to use whatever factoring of expressions is most natural.

In addition to the results on one- versus many-expression experiments in QDP-jit, we performed similar evaluations for QDP++ using traditional expression templates on Intel Sandy and Ivy Bridge systems. These were done for a wide range of problem sizes, including some much larger than are used in production. In contrast with the QDP-jit experiments, the single-expression cases do not dominate. Rather, it was determined that while the single-expression approach minimizes explicit memory traffic, it also creates cache and register pressure problems. This confirms that tradeoffs between "loop-splitting" and "loop-fusing" transformations described in the compiler literature are applicable here. Addressing this is on our agenda for the future.

This study performed the evaluation using four different compiler infrastructures (ICC 14, GCC 4.6.3, GCC 4.8.1, and CLANG-LLVM 3.4) over a wide range of on-node problem sizes, including those much larger than are usually used in production. GCC 4.6.3 was dominated in performance by the others. In all of our experiments, CLANG-LLVM and GCC 4.8.1 yielded more efficient code than ICC, but neither dominated the other. These observations should not be interpreted as being applicable beyond the current set of expression template codes.

The measurement of performance for large local problem sizes was interesting because the conventional wisdom from measurements on older generations of hardware has been that computational efficiency for lattice calculations degrades rapidly when the data does not fit in cache. Our measurements have shown instances in which efficiency degrades much more slowly than expected and other instances in which efficiency increases again for very large problem sizes. This phenomenon has not been consistent across either the examples we used or across the different compilers. We are performing further investigations to identify and quantify the underlying mechanisms.

During Spring and Summer of 2014 we began using the CLANG/LLVM compiler infrastrucure to perform static dependence analyses on the intermediate code generated using QDP++. We also completed an analyses of the efficiency of various manual choices regarding the operator (loop) fusion/splitting) , choice of compiler infrastructure, and problem size.

During this period we added the QPhiX library developed by JLab and Intel to the set of targets for our analyses of architectural dependencies. The main result emerging from these studies is that for both QPhiX and QDP++ both hardware and software prefetching can be very effective at lessening the "performance cliff" that occurs when local problem size exceeds local cache size, but for only a small range of problem sizes and only when using a subset of the available cores. In these ranges, the calculations are constrained more by memory latency than bandwidth. When problems become large and all cores are in use the memory bandwidth becomes a hard constraint. At such levels, injudicious

prefetching generates redundant memory traffic that interferes with the required traffic, thus degrading performance.

We ported the QPhiX library to the BlueGene/Q architecture where it was made available as a “level 3” module available to Chroma. In the course of doing this, optimizations were backported to the Xeon and Xeon Phi versions of the QPhiX and the entire library was modified to be usable with the GNU compilers.

In bandwidth-bound applications such as lattice codes there is an architectural imbalance between CPU speeds and memory speed/throughput. We are used QCD examples to drive SUPER institute research on data center energy management strategies that entail identifying the relatively fast and slow processors in a facility and adjusting their power budgets to balance them. We are also investigating the tradeoffs between energy consumption and throughput. By applying both techniques together, application performance can be enhanced while simultaneously saving energy.

Note that our activities during the period from July 1 2014 through mid-October 2014 were at a reduced pace due to Pi Fowler taking approximately 50% sick leave to care for his wife who was hospitalized, and had multiple surgeries, to treat a life threatening infection in an artificial hip. The pace of work has been accelerated since then.

Beginning in Summer of 2015, we began work on the design and implementation of QUARC, a framework for implementing embedded domain-specific languages (EDSLs) in the Clang/LLVM compilation framework. QUICQ, a statically compiled embeddeded language for LQCD that is similar to the QDP++ library interface, is implemented in the framework. QUARC is the main component of Diptorup Deb’s ovector swizzlengoing Ph.D. project. During the period of performance, two conference papers describing QUARC appeared.

The main goal of the QUARC project is to be able to close the “ninja gap”. In particular, compiler-generated code should be able to match the performance of the best hand-tuned libraries, in particular QPhiX, on the critical LQCD kernels. Furthermore, the framework should be usable to generate code with similar performance goals for other parts of the LQCD applications and for the kernels representing new physics or new mathematical methods. During the performance period we demonstrated performance comparable to that of QPhiX for both the Wilson DSlash kernel and solver used in Chroma and for the Kogut-Susskind Dslash kernel used in MILC. These results were obtained on both Intel Haswell and Intel Knights Landing servers.

Due to the reduced funding and level of effort compared with the proposed project, We were unable to test QUARC on full applications. Work on QUARC is continuing under other funding and we expect to demonstrate the full capabilities of the language in 2018.

Personnel

The following individuals were funded to participate in the project.

Name: Robert Fowler
Project Role: Principal Investigator.
Contribution to the Project: Leads the research.
Collaborated with individual in foreign country: No
Country(ies) of foreign collaborator: N/A
Traveled to foreign country: N/A
Duration of Stay: N/A

Name: Diptorup Deb
Project Role: Ph. D. Candidate in Computer Science, Graduate Research Assistant
Contribution to Project: Researcher.
Collaborated with individual in foreign country: Yes
Country of foreign collaborator: United States person visiting Great Britain
Travelled to foreign country: No
Duration of Stay: N/A

Name: Allan Porterfield
Project Role: Research Scientist.
Contribution to the Project: Collaborated on the early research in this project on the design of optimizers for a domain specific language for LQCD. Collaborated on design and proveded feedback on the design and implementation of QUARC.
Collaborated with individual in foreign country: No
Country(ies) of foreign collaborator: N/A
Travelled to foreign country: N/A
Duration of Stay: N/A

Name: Priadashi Sharma
Project Role: Graduate Research Assistant..
Contribution to the Project: Supported for one semester at the beginning of the project. He left the project before making any significant contribution.
Collaborated with individual in foreign country: No
Country(ies) of foreign collaborator: N/A

Travelled to foreign country: N/A
Duration of Stay: N/A

Personnel Changes

During the first project year, Priadarshi Sharma, a Computer Science graduate student, had been a member of the group and was working on this project. At the end of April 2013, he unexpectedly announced that he was taking a summer internship at a commercial firm. He is no longer on the project or a member of this research group. In August 2013 Diptorup Deb, a second year CS student joined the project. Diptorup's Masters degree project on this work and Ph.D. dissertation are based on this projct.

Allan Porterfied served as a Research Scientist on this project, contributing expertise in advanced compilers. He left UNC and the project in the Summer of 2016 to pursue a position in private industry. Since then he continued in an unfunded role to serve of Diptorup Deb's Ph.D. committee and to provide periodic feedback on the project.

Publications

Diptorup Deb, Robert J. Fowler, and Allan Porterfield. QUARC: an optimized DSL framework using LLVM. In *The Fourth Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC2017)*, Denver, CO, November 2017. ACM SIGHPC.

Sridutt Bhalachandra, Allan Porterfield, Stephen L. Olivier, Jan F. Prins, and Robert J. Fowler. Improving energy efficiency in memory-constrained applications using core-specific power control. In *5th International Workshop on Energy Efficient Supercomputing (E2SC)*, Denver, CO, November 2017. ACM SIGHPC.

Diptorup Deb, Robert Fowler, and Allan Porterfield. *QUARC: An Array Programming Approach to High Performance Computing*, volume 10136 of *Lecture Notes in Computer Science*. Springer International Publishing, Rochester, NY, September 2016. [bib | DOI | Abstract] [7]

Allan Porterfield, Sridutt Bhalachandra, Wei Wang, and Robert Fowler. Variability: A tuning headache. In *Symposium on Parallel and Distributed Processing (IPDPS) - Workshop Proceedings*, Chicago, IL, May 2016. IEEE.

Allan Porterfield, Rob Fowler, Sridutt Bhalachandra, Barry Rountree, Diptorup Deb, and Robert Lewis. Application runtime variability and power optimization for exascale computers. In *International Workshop on Runtime and Operating Systems at Scale (ROSS2015), Portland, OR, June 2015.*