Report Number: DOE-UW-10337-1

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

Federal Agency: DOE/Office of Science/Office of Nuclear Physics

DOE Award number: DE-SC0010337

Recipient Organization: University of Washington
Address: Office of Sponsored Programs
4333 Brooklyn Ave NE,
Box 359472,
Seattle,
Washington 98195-9472, USA.

Principal Investigator: Martin J. Savage, Professor of Physics

Postdoctoral Investigator: Emmanuel Chang

Project/Grant Period: 8/15/13 – 8/14/16

Report Term or Frequency: Final

Acknowledgment: "This material is based upon work supported by the Department of Energy under Award Number DE-SC0010337."

Abstract

This project was part of a coordinated software development effort which the nuclear physics lattice QCD community pursues in order to ensure that lattice calculations can make optimal use of present, and 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 UW team improved and extended software libraries used in lattice QCD calculations related to multi-nucleon systems, enhanced production running codes related to load balancing multi-nucleon production on large-scale computing platforms, and developed SQLite (addressable database) interfaces to efficiently archive and analyze multi-nucleon data and developed a Mathematica interface for the SQLite databases.

Project Summary

The UW lattice QCD physics effort is focused on determining the properties and interactions of the lightest nuclei from QCD using the numerical technique of Lattice QCD. This is a multi-faceted operation, relying heavily of USQCD software developed by others particularly for the configuration generation and light-quark propagator production. The calculation of nuclear correlation functions requires algorithms and techniques not used in other calculations, and features a workflow that is inversion and contraction intensive.

A reorganization of the N-body contractions was performed because the propagator plus hadronic building block (sink contraction) part and the source contraction part, which completes the correlation functions, have distinct parallelisms as illustrated in Figure 1. The former consists of multiple MPI jobs, each running on multiple nodes, while the latter consists of serial jobs, each running on a single CPU core. These two regimes must be matched in such a way that all the CPU cores allocated in a batch job are fully utilized throughout during the entire run. This is only possible, and without using an inordinate amount of disk space to temporarily store hadronic blocks, if the calculations of the N-body contractions are distributed evenly among the available CPU cores (so that hadronic blocks are consumed as soon as they are produced with a minimum amount of delay).

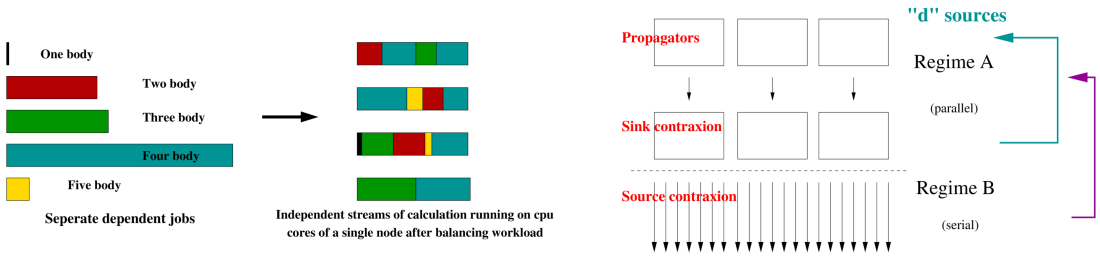

Figure 1. Correlators from different N-body contractions are grouped together to evenly distribute the work across CPU cores. This is shown schematically for a node with 4 CPU cores.

This workflow removes the need for dependent jobs, which were required due to the distinct parallelisms of different parts of the calculation. This simplifies the management of the runs, and reduces the failure

rates and temporary disk space usage. In addition to providing a new Chroma interface to the Multigrid code based on QDP/C, the following improvements were made:

- QDP++/Chroma and QDP/C distribute lattice points of the discretized space-time among the CPU cores differently. This mismatch in geometry showed up as non-convergence issues in propagator solutions. Modifications were made to QMP, the communication layer underlying both QDPs, such that a single command line flag "-qmp-geom" suffices to set correctly the geometry for both QDP++/Chroma and QDP/C.
- Compile times for QLA and QOP both used by Multigrid were improved by a more efficient organization of the functions among files to be compiled. QLA and QOP generate a significant number of source files with one function per file. A large number of small files reduce Make's ability to leverage the performance of a multi-core system. By merging multiple functions into a single compilation unit and combining with a more efficient build program (makepp), which is able to build a single dependency tree between different modules, a more efficient scheduling of the sub-compilations leads to a faster build.
- A new setup process was created specifically to compile and install all the USQCD modules required for building Multigrid with Chroma. Instead of the customary configure/make generated by autoconf for each module and specifying the locations where the dependent modules are installed, the dependency between the different modules are made explicit and the necessary files read by makepp are generated, which then build a single dependency tree between all the modules in the USQCD software ecosystem to better schedule the install and compilation processes. The effect of various C preprocessor defines which affect the compilation is made explicit.

A comparison between the Multigrid inverter and the mixed precision BiCG-Stab inverter was performed for PACS-CS lattices at several light quark masses and a NPLQCD isotropic lattice at a pion mass of 430 MeV, as shown in Figure 2.

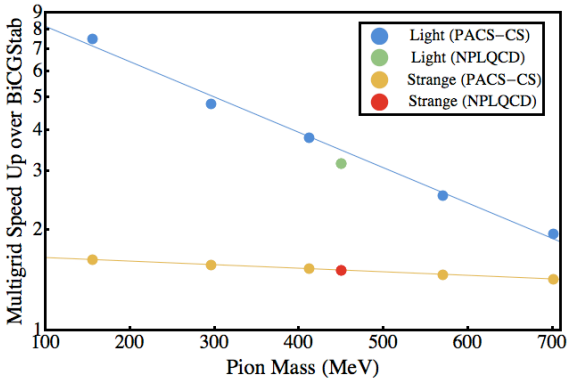

Figure 2. The speed up of the Multigrid inverter compared with the mixed precision BiCG-Stab inverter as a function of the pion mass.

NPLQCD data are produced in XML and SDB format, as outlined in the following table.

Format	Category	Type	
XML	Hadron spectrum	TEXT	Unstructured, mirrors Chroma code

SDB	Multi-baryon Correlators	BINARY	J-Lab in house database

Each XML file contains multiple correlators, and the standard procedure is to extract each correlator into its own file. In both cases, the information is represented as key-value pairs; in the case of XML, the key is the file name of the extracted correlator. The string representation of a key is, to a certain extent, arbitrary but not the information it contains, for example, QCD quantum numbers of the source/sink states of a correlator. Using a database allows more efficient management of this information, for example, one can query a group of correlators with some specific attributes, say, those with strangeness equal to -2.

In addition, one also avoids writing a large number of small files to disk, which are read back to perform averaging. For a typical run, this saves writing 80 ~ 200 million files. As a specific example, the global scratch file system at NERSC can have at most ~ 500 million files and the quota for a typical user is much less than that.

The following software developments have been completed:

- Mathematica notebook client to the new databases for analysis.
- C/C++ code to convert SDB to SQLite3 databases using CppDB (http://cppcms.com/sql/cppdb).
- Perl script to convert XML to SQLite3 databases. Also XML outputs are converted to YAML and compressed to speed up processing and saving disk space by a factor of 10, a schematic of which is shown in Figure 3.
- An ADAT compliant data stripper is also provided for compatibility and for debugging.

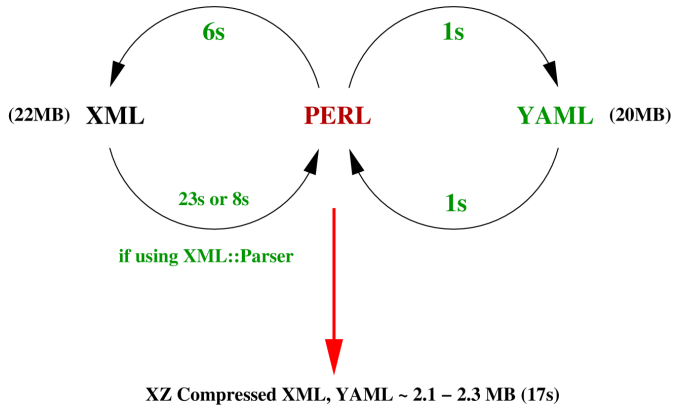

Figure 3. A schematic of the management of correlation functions.

A number of modifications to lattice QCD-related libraries were implemented:

- Updated **makepp** build to use the latest versions of USQCD software from git repositories; previously we had been using versions from 2011/2013. To maintain correctness and track the latest changes upstream, our customizations to **qmp**, **qdp++**, **qdp/c**, **qla** and **chroma** are now in git repositories at **https://github.com/6twirl9**.
- Changes to **qdp++**, **qdp/c**, **qla** and **chroma** are necessary for a successful makepp build but are otherwise compatible with the original build procedure shipped with the software. Change to **qmp** (default value for –qmp-job) is to avoid common error of not specifying the flag due to setup differences by users of **qmp**.
- The **qcdlib** used in our previous build dates back to 2012. Multi-level MG was broken. This was fixed in **qop** in mid-2014. The required changes are minor with a few details to consider, such as

the differences in function prototypes and struct definitions.

- The functions that transfer objects required – DiracFermion, ColourMatrix – between **qdp++** and **qdp/c** have been rewritten to accept all the **qdp++** objects. They have been moved inside **qdp++** with all dependence on **qla** removed. Currently the implementation is simplistic and the separation formal, i.e. it assumes, as is always the case by design, that **qdp/c** and **qdp++** map the lattice sites to the compute nodes in identical fashion.
- The addition of multi gauge-field capability to **wilson-clover-mg** is a natural progression after having added the option of setting up multiple MG for a given gauge field. The procedure is straightforward with slightly more complicated bookkeeping than previously required.
- To give users more flexibility, initialization of **qdp/c** and associated functions, and loading of the gauge-field links, have been separated. We are also looking into sharing of the gauge links by different MG setups to reduce memory usage where possible.

- All data from the NPLQCD collaboration's $m_\pi \sim 450$ MeV isotropic-clover ensembles (with volumes of L^3 where L = 24, 32 and 48) were converted to SQLite3 databases. XML data are converted to YAML first; SDB data are converted to SQLite3 directly using a Perl module interfacing **filedb** via **qdp++**. The SQLite files are kept in three forms corresponding to different stages of processing and serves as starting point for further processing or simply retracing the steps: 1) 1–1 with the original data, 2)Grouped by configuration, and 3) Sourced averaged.
- By examining the schema of a database, one can easily formulate an appropriate query to retrieve the needed data for analysis in a programmatic way. However, analysis usually proceeds by focusing first on a specific, small subset of the data with high impact and are necessary benchmarks then followed by more exotic states with humanely unreadable names. We eased the process of casual browsing by developing a catalogue, organized in a hierarchical fashion – Baryon number, Isospin, Spin, Baryon content –, using Mathematica's custom interface facilities.
- Combined with custom controls, with minimal visual impact (collapsible), which allows easy selection of the required source/sink structures with the desired spin projection. Baryon contents are clearly displayed with their Q-numbers using simply **Grid** with pastel color schemes.

Improvements were then made to the process of converting legacy Chroma XML and SDB files into SQLite database files and the *Mathematica* user interface to the database files. The rationale for this work is to establish a reference point for new ideas and improvement on the current workflow. The impetus is the need to provide a unified, high performance interface to current and past data, which are significant assets to the collaboration and the community.

SQLite is chosen for its simplicity, reliability and the wealth of publicly available and well-documented and supported tools. *Mathematica* is the preferred environment in which many conduct their analysis of the data. It is also well suited for prototyping and experimentation of user interaction. In particular, generalized input allows the construction of sophisticated user interface entirely within *Mathematica*. SQLite is now used to implement a table of *input parameters* and *output data*. This is a very simplistic yet informative way of exploring alternatives for current workflow. The *flow* of the set of input parameters forms the unique thread connecting every stage of a calculation from the start to the end and back. Implementation of each stage then becomes completely independent. This gives the user the option to adopt whichever is the best tool for the specific task from the specification of the tasks to perform, to the calculation proper and to the final analysis of data as one can follow the transformation of a given set of input parameters through all stages.

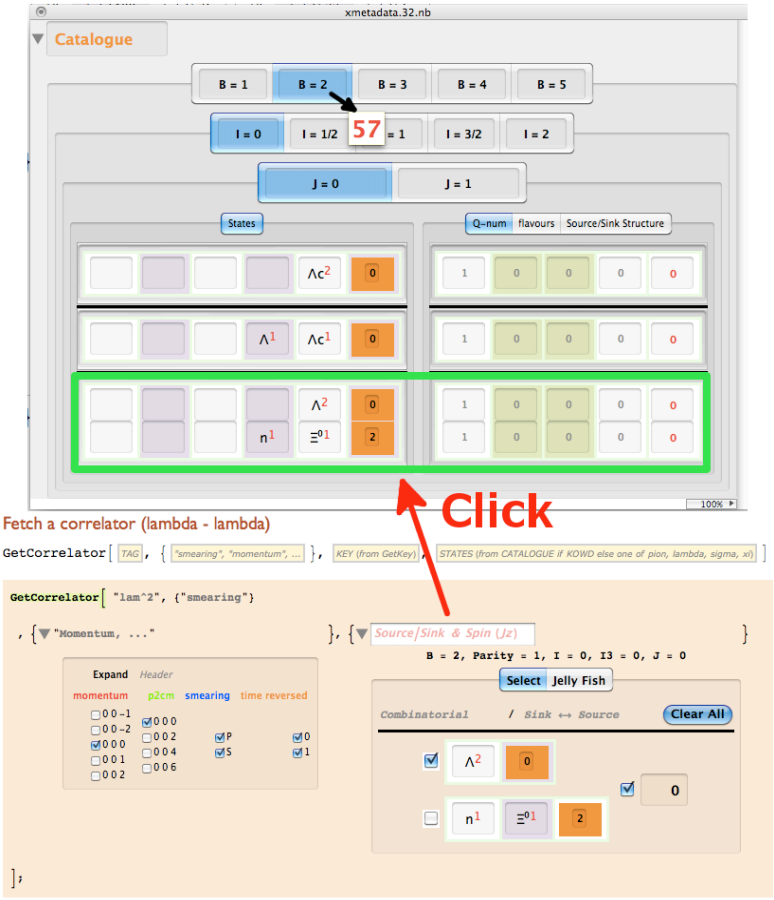

Figure 4. Mathematica notebook interface with SQLite databases containing nuclear correlation functions.

As it stands, one can almost draw a clear line between the calculation and the data. The extraction of specific information from either XML or SDB files depends on the infrastructure used to perform the calculation, which however does not make provision to facilitate the process. SDB makes an attempt yet the implementation is immature and dated. Both suffer from relying on very specific details of the framework on which the numerical codes are built i.e., the input parameters are insufficient to identify the data; additional information such as user defined document format as text or code is required. There is no provision for the long term archival and retrieval of data, which are rather costly to produce in terms of computing resources.

The situation is simply a legacy of past smaller scale production(s) when the framework was constructed and workflow practices established at that time. Our solution is to present a unified view of the data, prepared in forms suitable for archival and retrieval. The conversion process depends slightly on the original framework except where it was necessary to convert SDB key/value pairs to C struct to be passed onto PERL. The process itself serves as explicit descriptions of the format of the data. The XMLs are described using a YAML document augmented with markers to describe lists and extraction of data from specific patterns. SDBs are simple key/values pair. The information required to build up the set of input parameters can be obtained from a disassembly of the keys, which were generated from simpler input. We also ensure that the process is scalable at and above the current size of our complete set of data on two reference systems Edison, Cori at NERSC and Hyak at University of Washington. The whole process

is as automated and transparent as possible on both systems. Users have the option of running non-interactive or interactive if testing and extending the process for new sets of data. TMUX is used to allow one to monitor the job directly on the compute nodes.

The XMLs and SDBs can still be used with care. In fact, this is the presently recommended approach, as a consistent *flow* workflow requires time for users to adopt. Generating the SQLite database files directly from the main code is currently not implemented. To ensure the continuity of the work, the entire set of scripts used for the conversion have been rewritten, reorganized and greatly simplified so that users would find it easier to extend. It also went through much more thorough testing on the two reference systems Edison and Hyak. As far as the user is concerned, there is no difference or extra setup required running on either system. The new organization also makes it suitable as a foundation to build collaboration work upon it. And of course, it will continue to transform and adopt as it attempts to meet the demands of the users. In addition, it serves to pass on the knowledge and experiences accumulated from previous works.

Mathematica is used to perform analysis and as an interface to the SQLite databases. There is nothing that requires specifically *Mathematica* except that it is mightily convenient for those who use it for analysis and its generalized input provides a simple way to build a rather sophisticated interface. A complex interface is not always needed or desired. In this case, one may simply supply an appropriately formulated query to the JDBC SQLite driver and obtain a list of lists in return, which contain the set of necessary and sufficient input parameters together with the correlator data. Formulating the appropriate query is in effect what the *sophisticated* interface does. Often, the information required to index a particular piece of data cannot be described in simple words; a hierarchical and graphical representation of related information makes the process of picking the object of desire more intuitive.

The connection between SQLite and *Mathematica* is not direct. The correlator data, a list of double precision numbers, as a string of bytes does not map directly to a list of real numbers in *Mathematica*. Two approaches can be adopted: a) passing an SQLBinary object (a list of 64 bit integers) to a C function or b) instead of getting an SQLBinary from the JDBC driver, one obtains a HEX string version of the correlator data from SQLite and process that string in a C function returning a list of real numbers. Option b) is not only faster to load but also has a more natural vectorization. In addition, one can adopt a **CSLEGQ** convention (0 – 9, ; < = > ? are consecutive in the ASCII table) for HEX strings, which may work faster. Using the latest version of SQLite (currently version 3.12.2) is important. However, it appears using the *Mathematica* + JDBC combination is not ideal. Calling SQLite directly via LibraryLink for example should provide significant speedup – performing the same query and dump results to a text files formatted as a *Mathematica* package is an order of magnitude faster; loading the package file is relatively slow, however the overall time is still faster. Currently all queries and results are cached in *Mathematica*'s binary format (MX). Reloading the data takes on average less than ½ of a second even for large data set. Work is underway to provide a more direct access to SQLite.

To ensure continuity of the current effort, the entire *Mathematica* framework has been reworked, reorganized and simplified to make it easier for the users to intervene and add new features. In addition, a meta package system is also being written to make extending the current work with additional packages easier. It also provides a flexible basis for the collaboration to contribute and exchange knowledge and analysis methods in a well defined well. It is advisable to factor out complicated *Mathematica* analysis code and place them in package files that can be tracked and version controlled. Notebook files are fragile and cannot be version controlled effectively.