PROCEEDINGS OF SCIENCE

Vis: Online Analysis Tool for Lattice QCD

Massimo Di Pierro*
School of Computing - DePaul University - Chicago
E-mail: mdipierro@cs.depaul.edu

Yaoqian Zhong
School of Computing - DePaul University - Chicago
E-mail: ati_zhong@hotmail.com

Brian Schinazi
School of Computing - DePaul University - Chicago
E-mail: schinazi@gmail.com

Vis is a system that implements Software as a Service for Lattice QCD computations. At its core, it is a repository of gauge configurations accessed via either a web interface or programmatically via a web service. It gives users the ability to upload data and queue computing jobs for background execution. Jobs can be analysis algorithms and/or visualization algorithms (topological charge, polyakov lines, energy density, etc.). It exposes web services which are accessible from a command line script that allows, for example, to upload all gauge configurations in the current folder, request the server to make a plot of the average plaquette, and generate a movie of the topological charge. It interfaces with VisIt (also running server-side) for 3D visualizations and uses matplotlib for 2D plots. Data and results are automatically posted online with a role based access control mechanism. All major tasks can be executed directly from the web interface.

Lattice 2010

*Speaker.

PoS(Lattice 2010)326

1. Introduction

In this paper we present a tool for storing and organizing Lattice gauge configurations, for scheduling PBS jobs, including visualization jobs, and for viewing the results of those jobs.

The typical Lattice QCD workflow can be summarized in three steps:

- Lattice gauge configurations are generated, one after the other, in a Markov Chain Monte Carlo. Here we will refer to each chain as a stream.
- An algorithm runs on each gauge configuration in a stream and produces a numerical output (typically a correlation function) which is stored in a file.
- The numerical outputs generated in the previous steps are aggregated and averaged in order to compute a quantity of physical interest (for example masses, lifetimes, or wave functions).

This process is today performed by the larger community in an industrial fashion. Physicists have come together to standardize file formats for storing gauge configurations, for their remote storage and distribution, and for job management and submission. Yet today there are still multiple formats (NERSC [7] 2x3, NERSC 3x3, SciDAC [3], IDLG [6], MILC [2], FermiQCD [2]), with each format having the possibility to have big or small endianness or to contain single or double precision values, there are multiple systems for distribution (scp, gridftp, ILDG), and there are multiple hardware architectures to submit jobs to, which can have different configurations.

This multitude of options creates an entry barrier for young scientists who want to be involved in Lattice QCD research and also constitutes a major expense (in terms of time) for the more senior physicists already in this field. This is particularly true when visualization algorithms are involved. These involve additional file formats, third party dependencies, and domain-specific knowledge.

Vis is prototype software that provides an abstraction layer on top of existing systems that perform the tasks described above. It provides a simple to use web interface to store data that is independent of the file format. It supports importing all of the most common file formats for gauge configurations. Streams can be searched and individual files can be uploaded and downloaded via the web interface. Files can also be uploaded and downloaded using batch scripts that provide file integrity, pause and resume functionality, and security. All users are authenticated and all uploads are signed with the user credentials. Streams can be private to the owner or made publicly available.

Once files are uploaded into a stream, they are automatically converted into a common file format and some standard initial computations are performed on the data: endianness and precision are detected and the average (aggregate and moving) of the plaquette is computed. Metadata about stream is then stored, based on the credentials of the user and the computed parameters, and is available for use for search. These steps require no actions from the user's side.

Once a stream has been uploaded and cataloged, it is made available based on the permissions above. Users can them select a stream and run additional algorithms on it, such as an algorithm that computes the topological charge density. This is one of the major applications of Vis, for which it incorporates some domain specific logic to handle VTK files (for visualization purposes) and interoperates with VisIt [5] for 3D volume plots, iso-surfaces and rendering, and with matplotlib [4] for 2D plots.

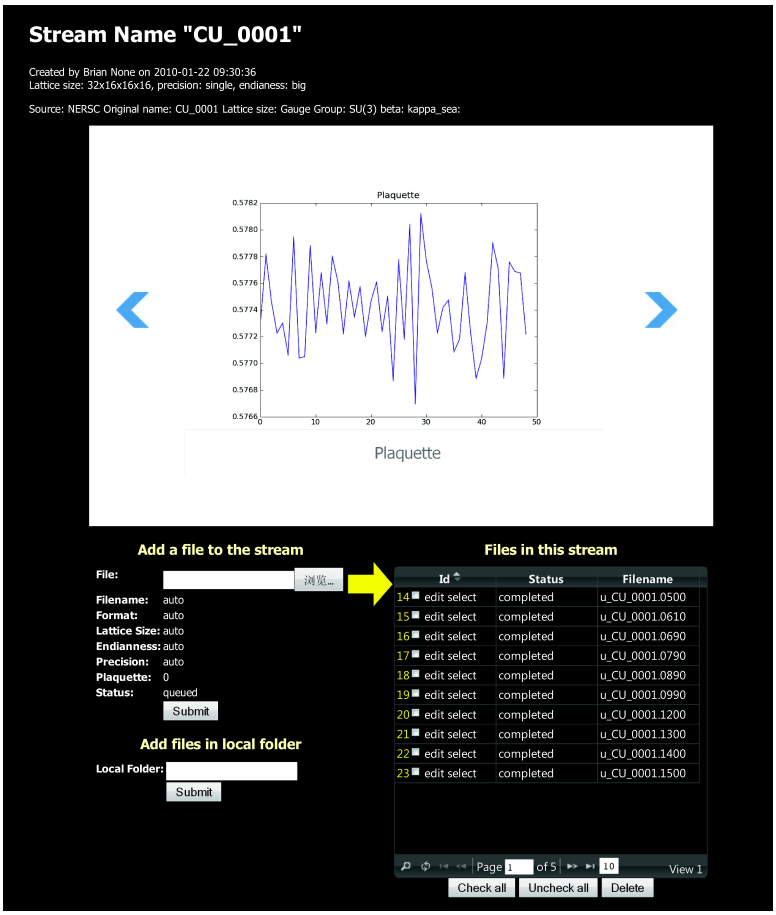

Figure 1: As an example we downloaded a stream of gauge configurations from NERSC [7] into Vis (CU_0001). Vis detects the endianness, precision, computes and plots the average plaquette, as represented in the figure (the plot is generated using matplotlib [4]).

With Vis, users can interact with data and schedule jobs without domain specific knowledge and with minimal previous training. This frees the physicist from some of the most pedantic and repetitive tasks. It also allows tracking of data, tracking of progress, and avoids duplication of effort.

2. Implementation

Vis is implemented in the Python programming language using the web2py [1] web framework. It uses a Database Abstraction Layer to interface with the database. It includes a collection of algorithms written in C++ using FermiQCD, but it is not limited to FermiQCD. Other binary programs can be registered with the system.

It its simplest installation it comes with its own web server and a file based transaction-safe relational database based on SQL. In a more complex installation, it can use other web servers (for example Apache) and other databases (MySQL, PostgreSQL, Oracle, MSSQL, FireBird, DB2, Informix, and Ingres). All data but the gauge configurations themselves are stored in the database.

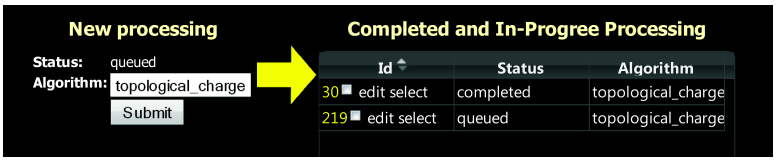

Figure 2: The figure shows the screen that allows the user to select an algorithm and schedule a job. The right hand side lists submitted jobs and completed ones.

The gauge configurations are stored in binary files which are organized into a structure of nested sub-directories, to avoid having too many files under the same folder. This folder structure can and should be hosted on a different filesystem than the primary database. These files are then referenced by individual configfile records, which are referenced by the table containing the record of the files' stream (fig. 1).

Since configfiles are uploaded by users, event though the system requires authentication, the system must account for exposure to directory traversal attacks. One such attack is performed via filenames that contain special characters not allowed by the file system. Vis implements many security features to prevent this vulnerability, as well as others classified by the Open Web Application Security Project (OWASP), including SQL Injection and Cross Site Scripting.

Although streams are created by the web interface, and the files can be uploaded in the same manner, this is not a practical approach and Vis provides an alternative mechanism. Upon creation, each stream is associated with a security token that is a random universally unique identifier (UUID). The owner of the stream can use a provided shell script to easily discover all gauge configuration file from a given local folder and upload them to the server into the stream identified by the UUID. Files have their hashes computed, and are then uploaded one at the time, while showing a progress bar. In case of network error or other upload failure, the script will resume and will use the hash values of the files to determine whether each file has already been successfully uploaded into the stream or whether it has changed locally and needs to be re-uploaded. The script communicates with the server using XML-RPC. An option can be set to have the script also submits jobs to be subsequently run against the data being uploaded.

Progress can be monitored both locally and via the web interface.

Algorithms are submitted via the Portable Batch System (PBS). Vis implements two queues, one that lists algorithms to be submitted per stream and a queue that maps the PBS queue. Vis does not immediately submit all jobs to PBS but monitors the PBS jobs to keep the workload limited and constant (fig. 2).

At the time of writing, only a few algorithms are supported and adding new algorithms requires editing of the Vis source code. Planned improvements include the ability for the user to register any third party program with Vis by providing a startup and configuration script.

Visualization algorithms are somewhat special because they consist of three steps. In the first step, data is analyzed and projected into one or more 3D scalar fields, which are then saved in VTK files. In the second step, the user interacts with the VTK file (this requires a crude but fast visual representation of the data). In the third step the data is visualized at high resolution, for purposes such as printing or high-quality display.

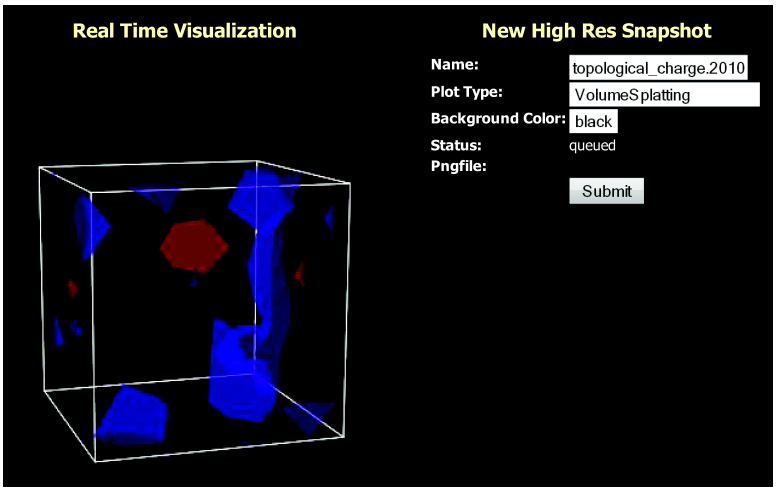

Figure 3: When a job is completed, the user can interact with the output (the VTK files) using the 3D JavaScript Widget shown in the figure. The user can submit a request for high resolution rendering of the data (for example using the volumetric splatting algorithm on black background).

Vis includes workflow and tools that distinguish these steps. Step one consists of a regular Lattice QCD algorithm and it is treated as any other algorithm. Step two is the most complex one to handle via a web interface. For this purpose, we created a 3D JavaScript widget that computes iso-surfaces and allows real time rotation (fig. 3) with the need to install a visualization program on the client-side. Once the user has identified an optimal point of view for visualization of the data, step three can be performed by selecting a high resolution volume or iso-surfaces plot and clicking a button.

Following submission, a VisIt job runs in the background and produces the specified image. A multiple choice menu allows users to pick settings for VisIt (such as colors, axes formatting, etc). There is no need to interact with the visualization engine directly (fig.4).

It is also possible to schedule visualizations of every gauge configuration in a stream and have Vis assemble the resulting images into a video using the ffmpeg open source multimedia tool.

3. Conclusions

Although Vis is still a work in progress, it constitutes one more step toward the ideal goal of the authors to provide a complete QCD Software as a Service platform. Today only the step of generating gauge configurations is performed efficiently in a semi-industrial way. This is primarily because this step is the most computationally expensive and there are few reliably-tested programs available to generate gauge configurations. Yet many other tasks performed by lattice QCD physicists can be automated. This presents many benefits: it reduces the margin of error, provides better optimization of resources, allows better tracking of data and work progress, and, most importantly, it frees the time of scientists for more intellectual activities such as developing new models, new algorithms, and tackling new problems.

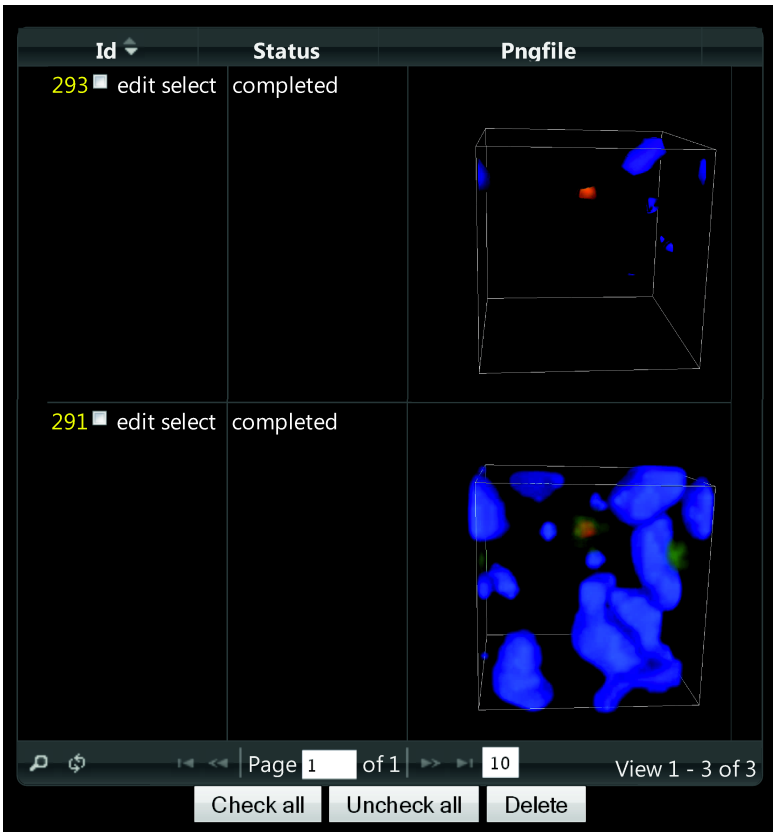

Figure 4: This screen shows a list of visualizations scheduled for the current VTK file. If a visualization if completed, a thumbnail of the image is displayed. The image can be downloaded by clicking on the thumbnail.

At this point Vis is a stand-alone web application but it can be integrated with existing ILDG tools to provide a unified interface to Lattice QCD computations.

In the near future we are planning a better integration with FermiQCD to provide a wider choice of algorithms; a better customization to allow support for algorithms written with different Lattice QCD libraries; a more sophisticated workflow management system to handle complex conditional dependencies; and integration with mc4qcd (a web based plotting and analysis tool also written by the authors).

The hardware currently dedicated to VIS is very limited both in terms of speed and space. Dedication of better hardware to this system will enable us to provide QCD as a service to interested users.

The current version of Vis can be downloaded from:

`https://launchpad.net/qcdvis`

Acknowledgements

This project was funded by the Department of Energy, grant DEFC02-06ER41441.

References

[1] Massimo Di Pierro. web2py website, 2010.

[2] Massimo Di Pierro and Jonthan M. Flynn. Lattice QFT with FermiQCD. *PoS*, LAT2005:104, 2006.

[3] Robert G. Edwards. Nuclear physics using lattice QCD in the SciDAC era, 2009.

[4] John Hunter, Darren Dale, and Michael Droettboom. matplotlib website, 2010.

[5] Lawrence Livermore National Security, LLC. Visit website, 2010.

[6] C. Maynard. International Lattice Data Grid: Turn on, plug in,and download. In *Symposium on Lattice Field Theory*, 2009.

[7] U.S. DOE. Nersc website, 2010.