Managing Diffraction Beam Data with an Offline Event Builder

Ian Laurent van Roque[1], Dr. Monarin Uervirojnangkoorn[2+], Dr. Paul Christopher O'Grady [2++]

[1]Intern.

[2]Linac Coherent Light Source, SLAC National Accelerator Laboratory, 2575 Sand Hill Road, Menlo Park, CA 94025, USA.

[+] Contact: monarin@.stanford.edu [++]Contact: cpo@slac.stanford.edu

Overview

We developed an Offline Event Builder to process diffraction data in support of major upgrades to LCLS. The LCLS-II project intends to increase pulse frequency by 8000 fold, allowing beam diffraction information to pour in at 20GB of data every second. We evaluate the effectiveness of numerous analysis scripts in order to determine the optimal performance of a data-reduction pipeline intended to filter through "interesting" beam events.

Keywords: Python, MPI, core threading, LCLS-II

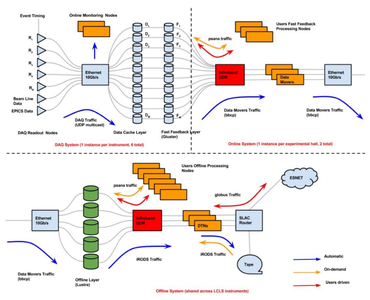

Fig 1. Flowchart depicting LCLS data flow. The top half of the image illustrates the "online" portion of the process in which data is handled as it is received. The bottom half portrays an "offline" portion of analysis where event data has already been saved to memory. Our scripts are to be implemented in the bottom half

Methods

In our analysis, variations of three principle scripts are timed in order to test their effectiveness in auditing large quantities of data. Scripts are required to inspect h5 files across a multi-core threading system using MPI. Data is distributed to cores in user-defined "batches." It is expected that scripts will become more efficient as batch number increases toward a maximum-efficiency batch after which analysis time plateaus off.

First Principle Script: The "mpiscript"

Analyzes single h5 file.
Timestamps distributed to cores according to batch.
Cores fetch event data for analysis

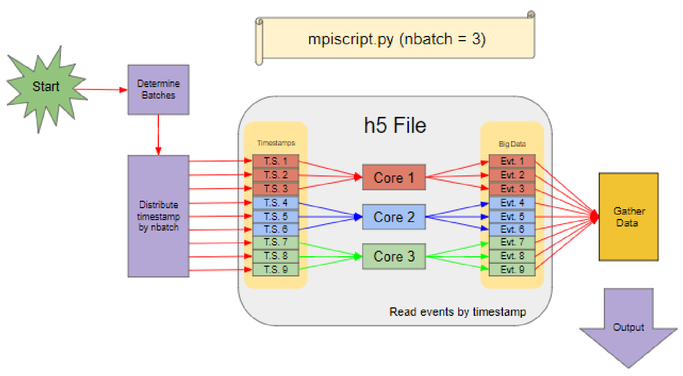

Fig 2. Flowchart of the first principle script. Cores gather corresponding event data from the timestamp indices. The output for this script is the time it takes for all the events to be paired with their timestamps.

Second Principle Script: The "superscript"

Pairs timestamp data over multiple h5 files.
Event data with matching timestamps are gathered.
np.searchsorted pairs variable length data.

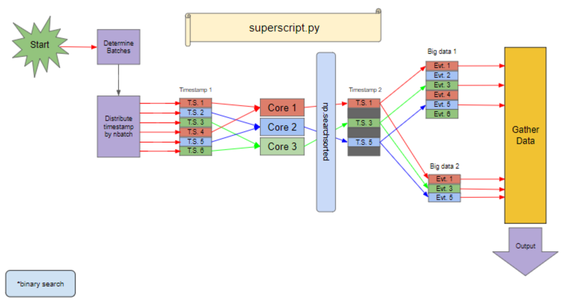

Fig 3. Illustration of the "superscript." The event data that has been matched across timestamps are gathered together as in "mpiscript." The output for this script is the time for all identical timestamps to be paired to their corresponding event data.

Final principle Script: The "superdealer"

Utilizes master-client relationship between cores.
Indices with matching timestamps are gathered.
Client cores request batches of data from the master.
Clients pair event data with their timestamps.

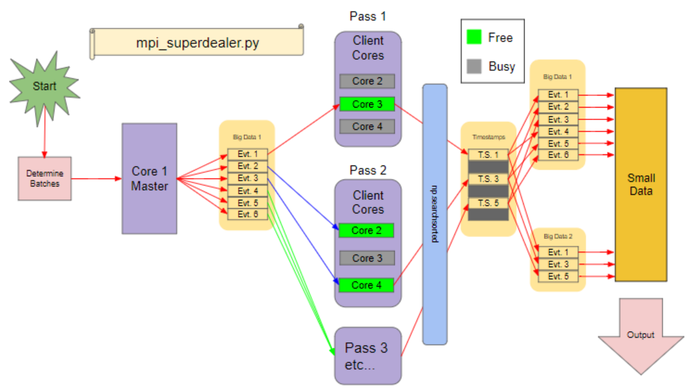

Fig 4. Flowchart of the "superdealer" principle script. The master core matches timestamps for all of the events which are then given to the client cores. Client cores gather event data over the ten h5 files.

Analysis

Each script is tested over multiple batch sizes. From the data we can see that the primitive "mpiscript" can filter though 1GB of data on minute-length timescales, too slow for our ambitious analysis requirements. As expected, analysis time is seen to decrease with both batch size and number of cores.

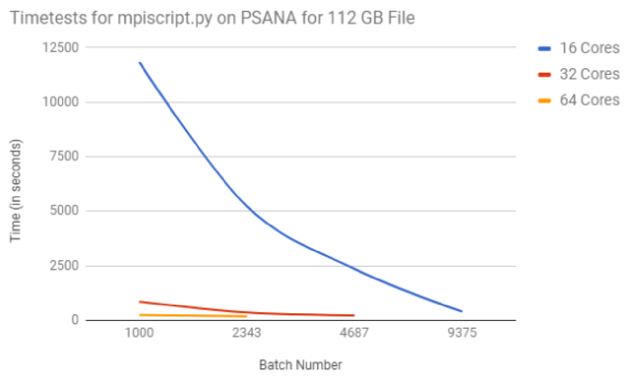

Fig 5. Plot for the "mpiscript." As seen here, analysis times of about a gigabyte per minute have been recorded. As expected the algorithm is faster on more cores and approaches a minimum analysis time as batch increases.

The "superscript" is shown to fare better in the volley of event data it encounters, and is capable of pairing data at about a gigabyte every second. Further tests show analysis time approaching a maximum efficiency, agreeing with our assumptions.

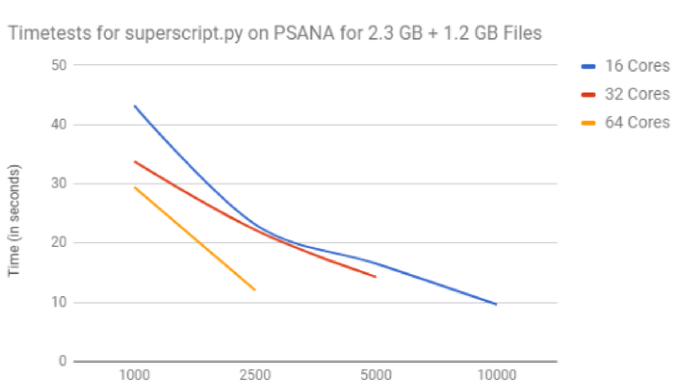

Fig 6. Data for the "superscript." The "superscript" filters through data faster than the original principle script. The same pattern as the previous script is seen confirming our hypothesis that the analysis time is dependent on batch and the number of cores.

Results

Unfortunately for our "superdealer," time tests have displayed asymptotic behavior towards a minimum analysis time of nine gigabytes a second. At this point, increasing the batch size of the distributed data will not increase efficiency. In fact, increasing batch size much further is expected to reduce performance as cores are overwhelmed by prodigious volumes of data. As seen in the graphs, the inspection times of 32 cores is similar to the 64 core analysis. While adding more cores would speed up survey length, it is not expected to bolster the process by more than a few gigabytes a second at most. (Best time for 256 cores is 17.6 GB per second)

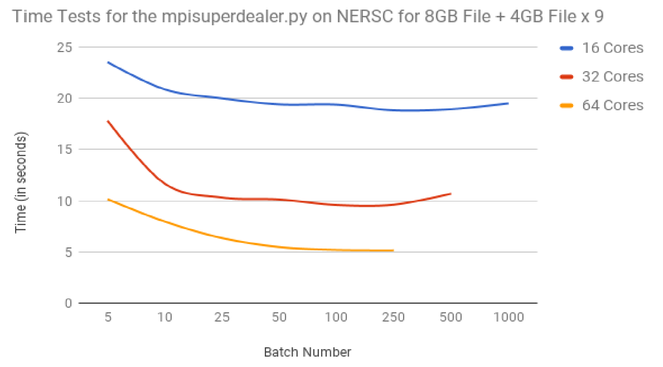

Fig 7. Plot for the "superdealer" script at NERSC. As seen here analysis times of 8.8 gigabytes per second have been recorded. As surmised the analysis time wavers around a maximum efficiency as batch size increases. Analysis time actually increases for very large batches as client cores begin to be inundated with data.

Future

While the ultimate goal of managing 20 gigabytes of data has not yet been reached, many improvements to the scripts can still be made to increase the effectiveness of the algorithms. The script is intended to be optimized in C to maximize efficiency between the code and the client cores. Refactoring of the script may increase speed as well as more sophisticated use of while loops to quickly run through event data. The script will also need to be updated in order to accommodate different or changing numbers of files, especially large event arrays much heftier than 10GB per file. Tests on better file systems such as the Burst Buffer at NERSC are being prepared and are expected to produce better results. With these improvements, the master-client method of analysis may become more agile as it sifts through event data, allowing for the possibility of a 20GB per second data reduction pipeline.

Acknowledgments

Use of the Linac Coherent Light Source (LCLS), SLAC National Accelerator Laboratory, is supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences under Contract No. DE-AC02-76SF00515.

Date: 08/28/2017