HEPnOS event selection status

Marc Paterno and Saba Sehrish

14 Jan 2020

Overview of the work

- We have created a client application to exercise the HEPnOS event service.
- We are using the NOvA neutrino interaction candidate selection application, written to use MPI through DIY.
- This application:
  1. Reads some data products for an *event* (containing multiple *slices*) from a `hepnos::Event`.
  2. Transforms the HEPnOS-based data structures into NOvA `StandardRecord` data structures.
  3. Processes each *slice*, to determine whether it contains a candidate neutrino interaction.
- The slice IDs for all slices containing a candidate are written to the output.
- A separate program is used to load the data from HDF5 file(s) into the running HEPnOS store. We do not collect detailing timing information in this program, because it is not our main interest. We do have a summary of how long the loading takes.

Remember that paper?

- For the planned paper, our goal is to demnostrate the *speed* and the *scalability* of the HEPnOS event store.
- We want to study how the performance varies with operational parameters of the system:
  1. size of resources used to serve the data (nodes, ranks per node, . . .)
  2. size of client resources (nodes, ranks, . . .)
- Our goal in this talk is to show what we have learned thus far, so that this afternoon we can make concrete plans for how to proceed toward the completed paper.

What we're currently measuring

We have instrumented the `eventselection` program to record timestamps when certain *steps* take place in each rank of the DIY program:

1. start of program
2. start of processing a block
3. start of processing an event
4. end of reading data from HEPnOS
5. end of creating `StandardRecord` objects for the event
6. end of processing an event
7. end of processing a block
8. finish of program

We use this to measure:

1. whole program execution time
2. block-by-block execution time (for each rank); each block processes N events
3. event-by-event execution time (with several subdivisions, noted later)

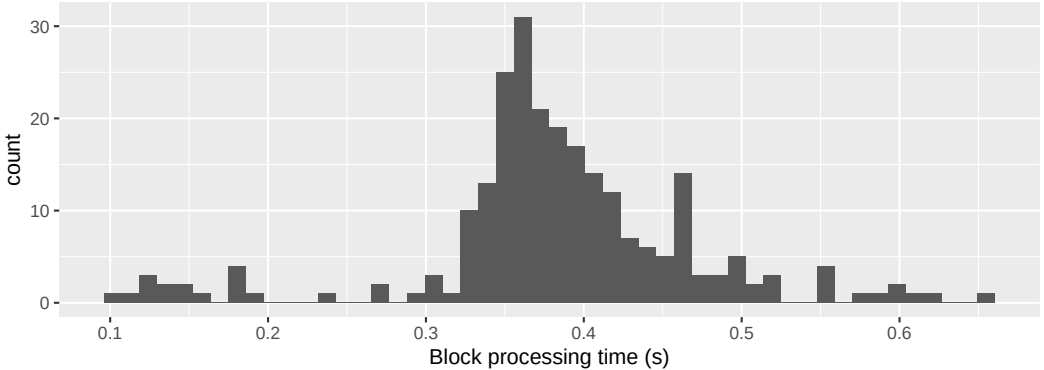
First view of performance: small run on local resources

- To *start* to understand the data, we have run on local FNAL resources.
- Everything run on *grunt3*, a 32-core AMD Opteron 6128 machine with 64 GB of RAM.
- We ran the HEPnOS daemon with 1 rank.
- Client program runs with 30 ranks.
- The program executes 244 blocks.
- There are 2,439 events and 20,673 slices in the data used for these runs.
- The total processing time is 4.98283 seconds.

Block processing

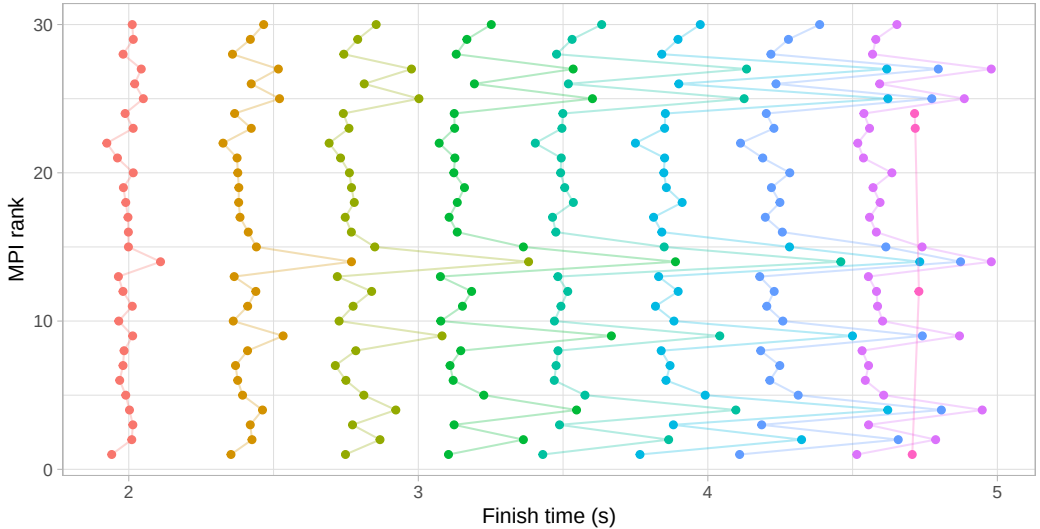
- The main part of the `eventselection` program is processing a series of blocks.
- Each block processes some number of events from a given *subrun*.
- We want to limit the number of events in a block to help load balance the program.

Distribution of block processing times

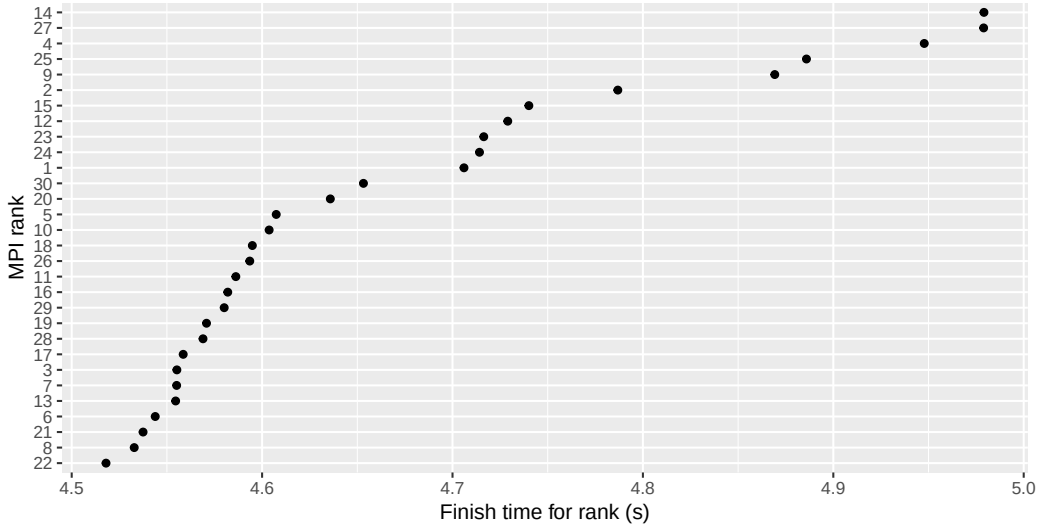


Min	1st Qu.	Median	Mean	S.D.	3rd Qu.	Max
0.106	0.351	0.378	0.383	0.087	0.417	0.659

Load balance: finish time of each “wave” of blocks, by rank



Load balance: time at which each rank has processed its last block

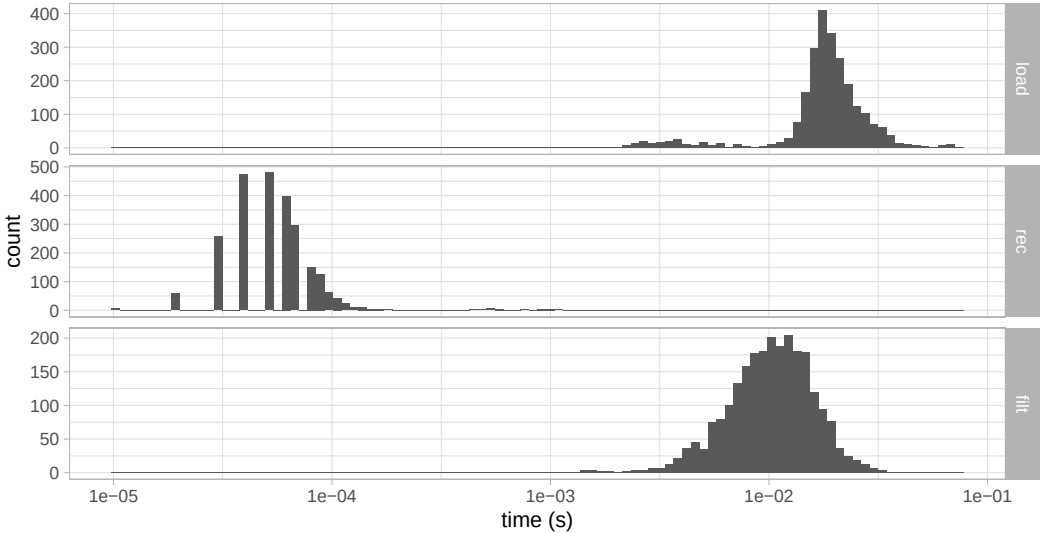


Fermilab

Looking at event-processing times

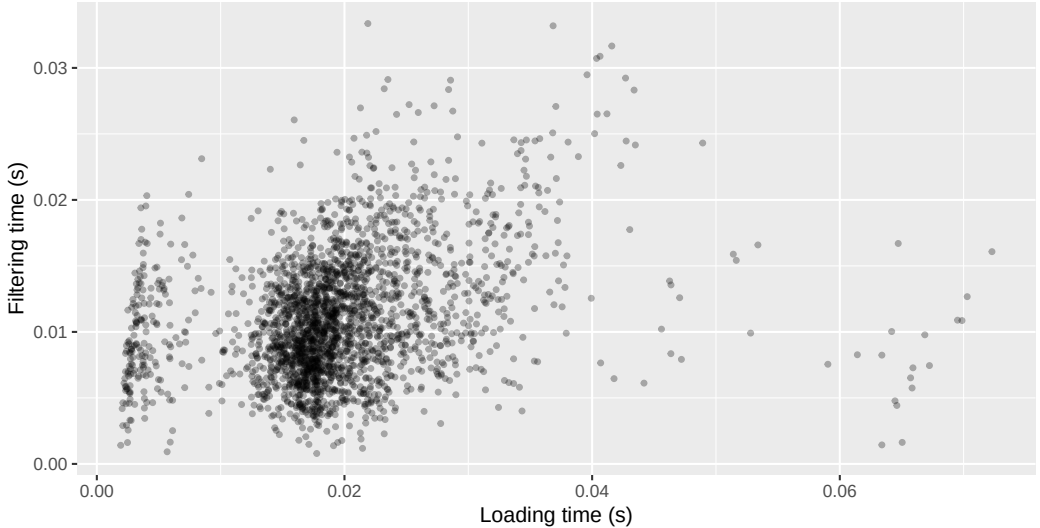
- Recall that for each event we capture 4 timestamps:
  1. before loading data from HEPnOS (start of event)
  2. after loading data from HEPnOS (before starting translation to `StandardRecord` objects)
  3. after translation (before NOvA candidate selection)
  4. after candidate selection is done
- We record the total number of slices in each event.
- We record the total number of bytes read from HEPnOS.
- We do *not* record slice-by-slice information; there is just too much of it.

Time for each task in event processing



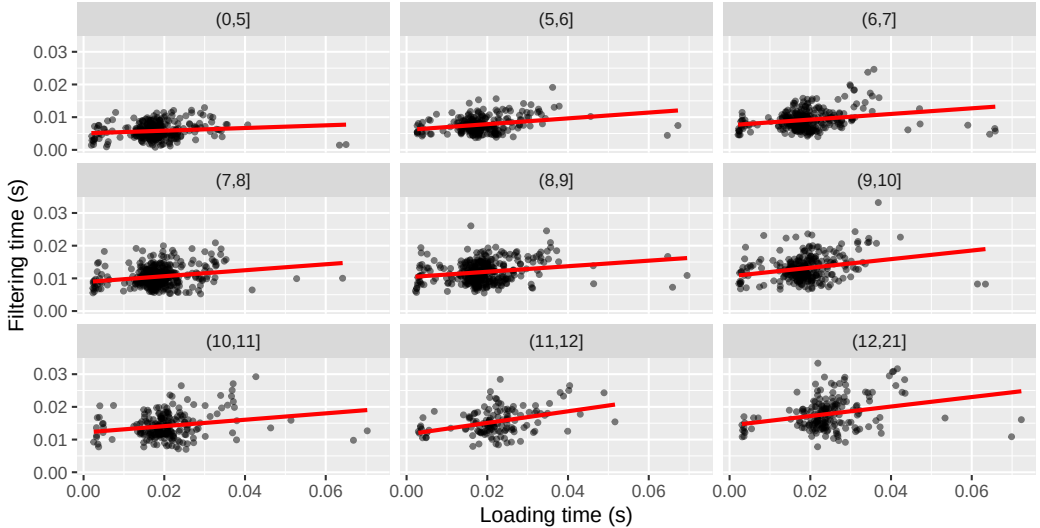
Fermilab

Is there a correlation between slow loading and slow filtering?



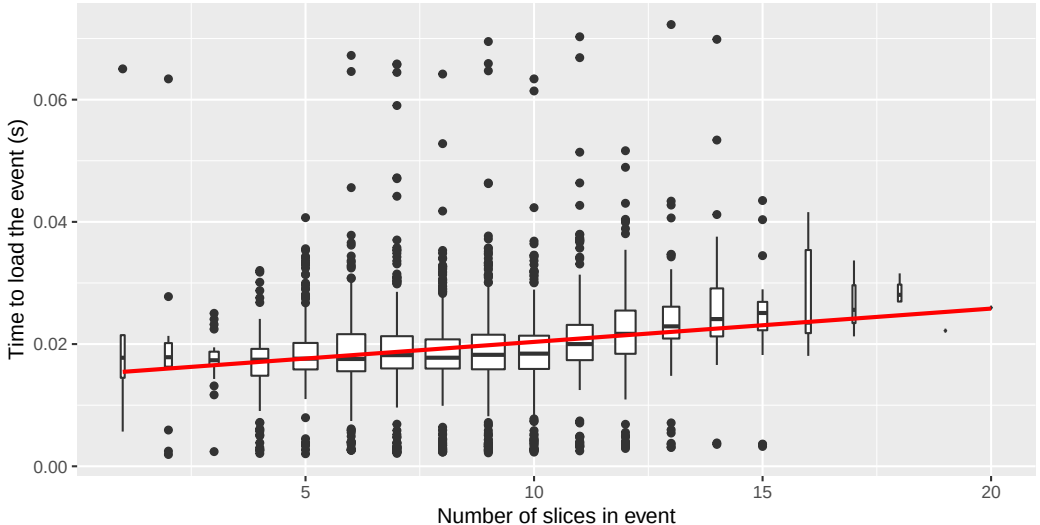
Fermilab

Is there a correlation between slow loading and slow filtering?

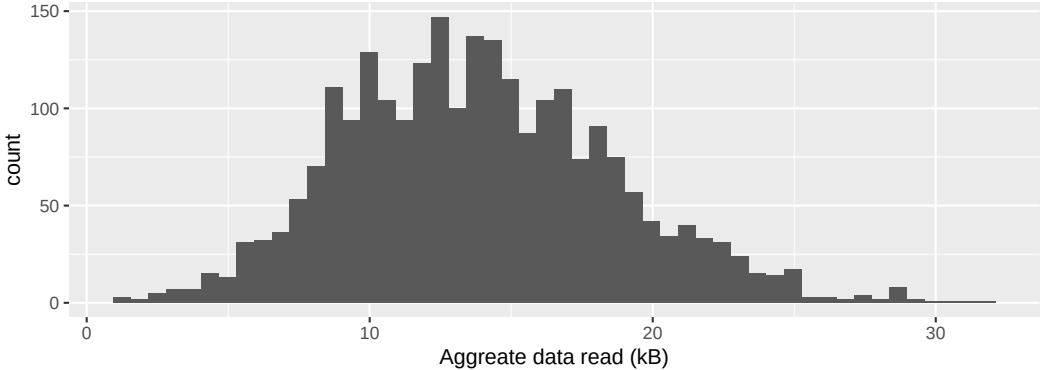


Fermilab

Load time as a function of number of slices

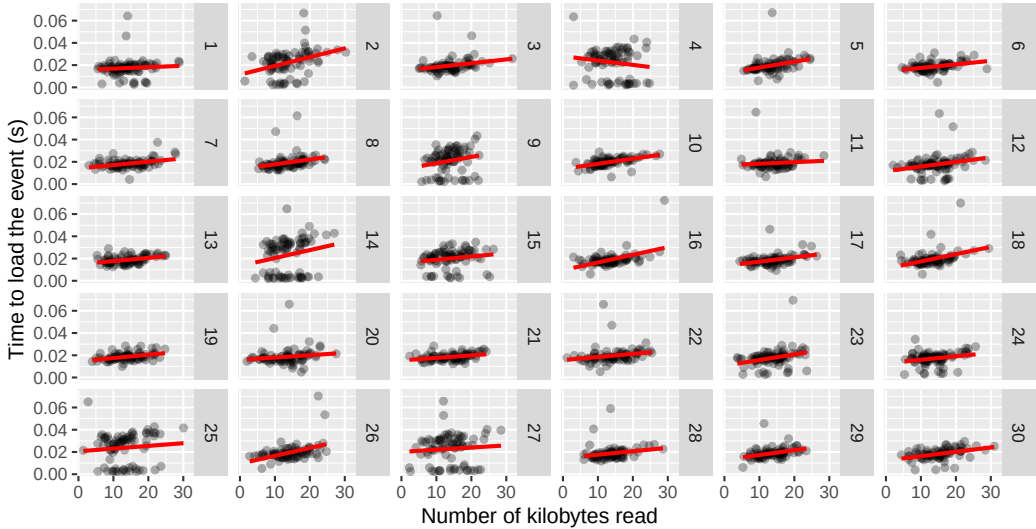


Distribution of aggregate read size (we don't read the whole event)



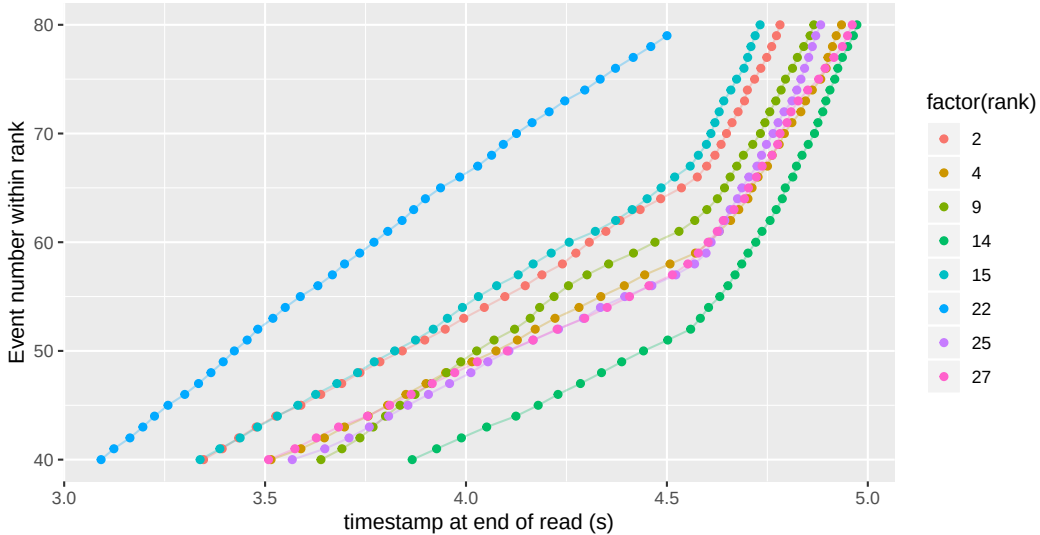
Min	1st Qu.	Median	Mean	S.D.	3rd Qu.	Max
1.16	10.354	13.594	13.86	4.761	16.922	31.711

Load time as a function of aggregate data size



Fermilab

Maybe we're overwhelming the one daemon rank?

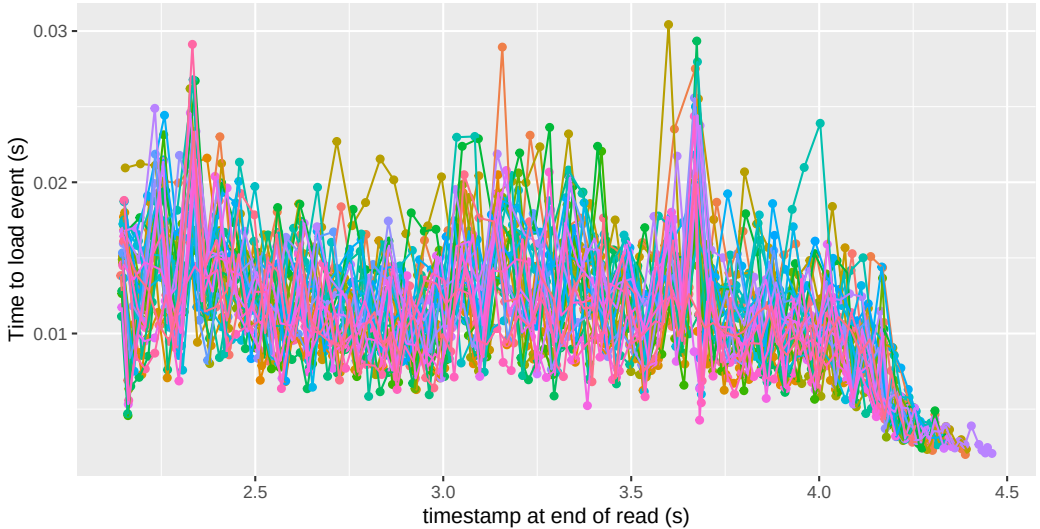


Fermilab

Load time as a function of time

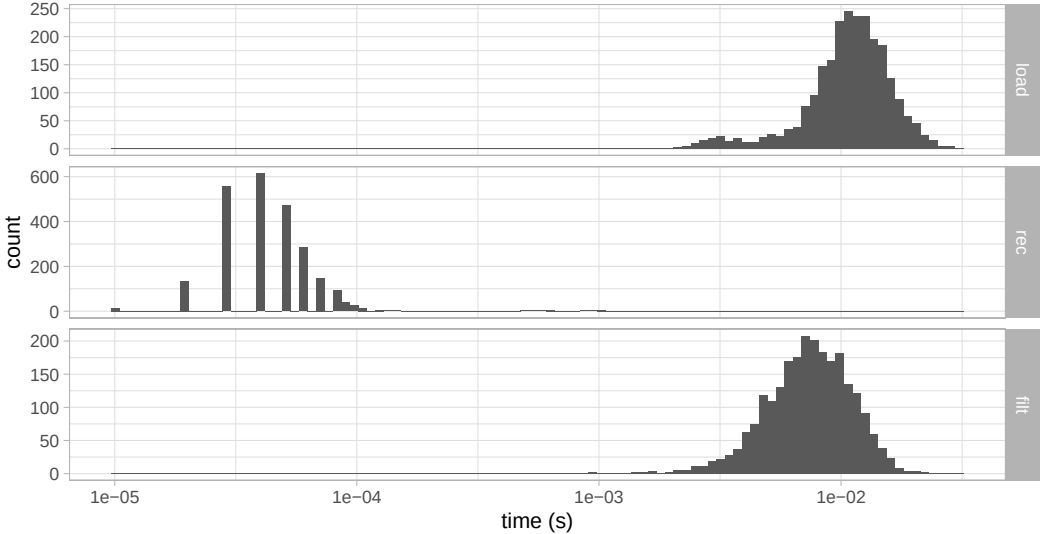
Fermilab

First look at a run with two daemon ranks



Fermilab

Time for each task in event processing with 2 daemon ranks



Fermilab

Comparison of 1- and 2-rank server performance

Min	1st Qu.	Median	Mean	S.D.	3rd Qu.	Max	server ranks
0.002	0.016	0.019	0.020	0.008	0.022	0.072	1
0.002	0.009	0.011	0.011	0.004	0.014	0.030	2

Questions

- Are we collecting the right information to be able to properly characterize the performance we care about?
- Do we need more performance tuning before we start running experiments on Theta?
- What should be the *figure of merit* for HEPnOS?
  - Number of events/second read?
  - Total bytes/second transferred?
  - Throughput (slices/second) of whole `eventselection` program?
  - Something else?
- What should we be varying to characterize scaling?
- Is our data set large enough for the studies we want to do?
  - On Theta, we have files containing ~4.4 million events

Let's talk about this—and make choices—in the afternoon.