ACCELERATING PERSONAL COMPUTATIONS WITH HTCONDOR: GENERATING LARGE NUMBERS OF EVENTS WITH GENIE

N. Balashov[1,a], I. Kakorin[2], V. Naumov[2]

[1] *Laboratory of Information Technologies, Joint Institute for Nuclear Research, 6 Jolio-Curie st., Dubna, 141980, Russia*

[2] *Bogoliubov Laboratory of Theoretical Physics, Joint Institute for Nuclear Research, 6 Jolio-Curie st., Dubna, 141980, Russia*

E-mail: [a] balashov@jinr.ru

GENIE is one of the most popular MC neutrino event generators, widely used in many modern neutrino experiments (e.g. NOvA, MINERvA, MicroBooNE, KM3NeT, IceCube). The tasks related to the development and optimization of the generator itself require creating a large number of events in the shortest possible time in order to reduce the overall development time. The usage of large-scale distributed computing infrastructures, such as Grid, does not guarantee the minimal execution time due to possibly long queue times. At the same time, the power of a modern PC is not capable of performing such computations in a reasonable amount of time. In this work we give an example of a hybrid approach: accelerating computations by using a personal computing device in conjunction with a general-purpose batch system based on HTCondor.

Keywords: distributed computing, HTCondor, GENIE

Nikita Balashov, Igor Kakorin, Vadim Naumov

1. Introduction

Modern research in neutrino physics requires a large amount of computing power to carry out data analysis in a reasonable time. Traditionally research computing centers provide the majority of computing capacities through batch systems (e.g. HTCondor [1]), which form a queue of user jobs and handle their distribution across multiple computing nodes. Of course, not any computing job can be easily distributed and processed on a batch system; one of the tasks that can be distributed is to generate a large number of neutrino events using the GENIE generator [2].

GENIE (Neutrino Event Generator & Global Analysis of Scattering Data), as one of the main components, is used in the detector simulation software chain of the FNAL accelerator of the neutrino oscillation experiment NOvA (NuMI Off-axis νe Appearance) [3]. To date there is no official publication of results for (anti)neutrino interaction cross sections in the near detector (ND), though preliminary data are available [4]. The GENIE package is also actively used in another NOvA-related Fermilab experiment MINERvA (Main Injector Experiment for ν-A) [5] using the same beam NuMI at FNAL, but operating at different energies (MINERvA is on-axis, while NOvA is the off-axis experiment). A careful comparison of GENIE simulation with MINERvA measurements is necessary to test the models of interaction for (anti) neutrinos with nuclei used in NOvA. Figure 1 shows an example of such a comparison of MINERvA experimental data [6] with GENIE simulation based on the phenomenological model SuSAM* (a super-scaling model operating with the so-called nucleon effective mass, which accounts for non-trivial nuclear effects in lepton-nucleus interactions) [7]. The number of events generated for the high statistics histogram is 5 times greater than the corresponding number for low statistics: the difference between the histograms is barely noticeable, while the corresponding $\chi 2$ values (calculated with the full covariance matrix) differ significantly.

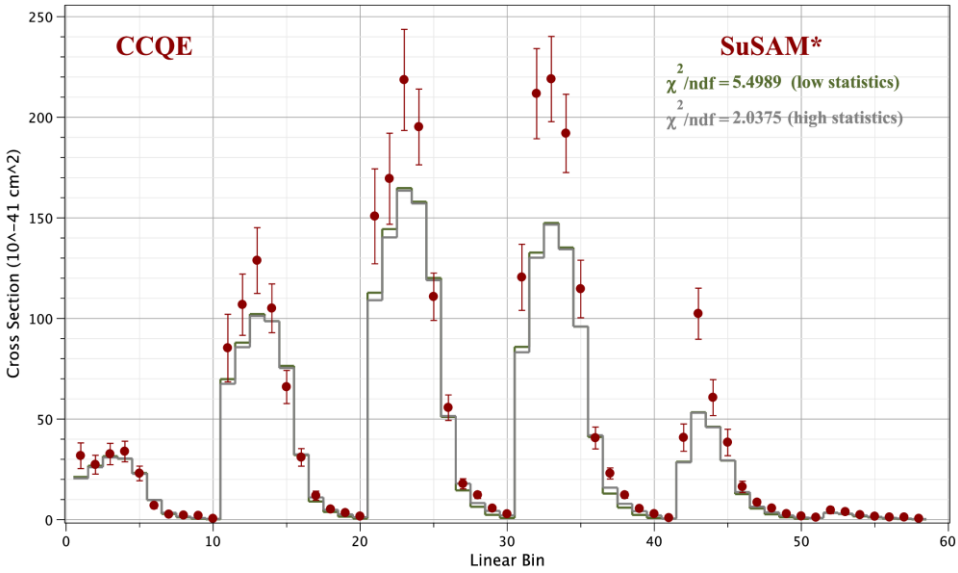

Figure 1. Comparison of experimental data with GENIE predictions. The points with bars show MINERvA data (a double differential cross section for the muon antineutrino scattering on hydrocarbon vs linear bins) with full (statistical and systematical) errors. The histograms show predictions of the SuSAM* model [7] incorporated into the GENIE MC generator. Simulations with GENIE were performed with low and high statistics

Figure 2 illustrates the reason for the observed difference. It uses a different representation of the same data but as ratios of Monte Carlo simulation (after the best-fit re-normalization allowed due to the uncertainty in the NuMI antineutrino flux) to MINERvA data; the experimental error band is shown on the background. The dips on the low statistics histogram are due to the fact that the cross section values in the corresponding bins are very small in comparison with the ones in other bins (see Fig.1), but the weight of contributions from these bins into the $\chi 2$ sum is rather large: if the number of generated events is not sufficiently large, there is almost no chance for an event to get into the dips, while the experimental data contain such events. That is why $\chi 2$ for the low statistics histogram is so large.

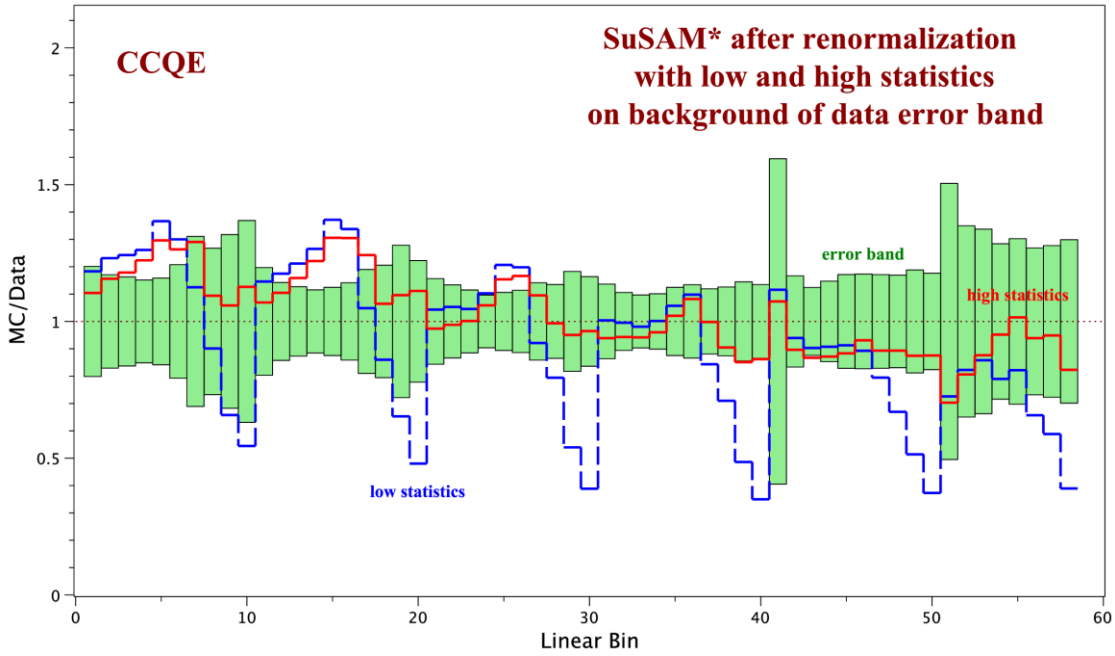

Figure 2. Re-normalized high and low statistics histograms

It takes about 576 CPU hours to generate the high statistics histogram and 115 CPU hours for the low statistics one correspondingly. To compare a theoretical prediction with only one experiment, one needs to generate dozens of such histograms, which will take about 10 days to produce results on a workstation with 24 CPU cores and that is unreasonably long. Using a dedicated batch cluster with 576 CPU cores can reduce this time to just 10 hours, but since research clusters are normally multi-user with a fair-share usage of resources, the overall waiting time may still be significant when the cluster is busy. In this paper, we show an example of the combined workflow: accelerating a personal computing resource using a general-purpose batch cluster.

2. Combining a batch system with a PC

As shown in the introductory part, batch clusters can significantly speed up computations even if a user has a powerful personal computer, but this is not guaranteed. The major factor is the availability of nodes on the cluster: the resources of the cluster are normally co-shared by many users and their jobs may stay in a job queue for days waiting for an execution. By connecting a user personal computer to the batch cluster as a worker node we achieve the following goals:

- The personal computer is still dedicated to the user, thus guaranteeing the maximum execution time limited by the personal computer power.
- The execution time can be speeded up depending on the availability of computing nodes of the batch system.

- The user gets a unified job execution system: the entire workload gets started via a submission mechanism of the batch system, so there is no need to start it differently on the personal computer.

  The major drawback of this approach is a much more complicated software distribution.

2.1 Testing environment

The suggested approach was tested in the following environment:

- HTCondor batch system with a total of 418 CPU cores.
- The memory in the cluster ranges from 2 GB to 5 GB per CPU core.
- All general-purpose nodes run Scientific Linux 6.10.
- One personal computer with 24 CPU cores and 48 GB RAM running Debian 8.

The HTCondor system allows one to connect new worker nodes without restarting any services (daemons), so it is possible to dynamically add or remove computing nodes, and it does not affect the functioning of the rest of the cluster. It also has a daemon-to-daemon authentication system making it possible to restrict access to specific nodes in order to join the cluster. In our test setup we used the shared x509 certificate for the daemon-to-daemon authentication, but it is possible to use per-node certificates for better control over the system.

2.2 Software distribution

As was mentioned above, the main issue to consider in such an environment is how to organize the storage system. In our particular use case we were able to distinguish two types of software:

- Software that does not change much and does not need to be frequently delivered to worker nodes
- Software that changes frequently

The standard approach for delivering software coming from the field of High Energy Physics (HEP) is the use of CVMFS [8]. Unfortunately, the deployment of the CVMFS service at JINR is not well suited for storing software and data that change a lot and need to be delivered to worker nodes: it takes at least 1 hour to synchronize changes between the CVMFS servers. However, it works well for the software that does not change much.

For frequently changing software and data, we used the NFS share: it provides access to the latest changes, but introduces a much higher network load, as it is a classic network-attached storage.

In our case the cluster operating system (OS) is different from the user node OS, so it is also necessary to build all the software for all the operating systems available on the cluster.

2.3 Resulting system and future modifications

As a result, we came up with the system shown in Figure 3.

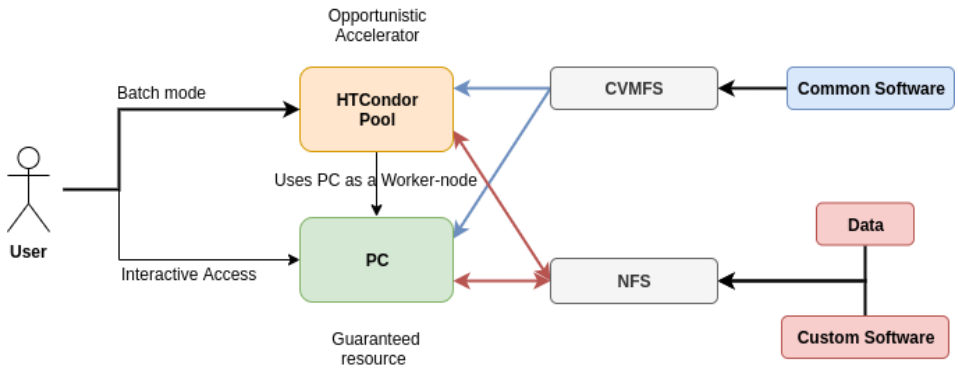

Figure 3. Scheme of the final system

The presented scheme works quite well for our users, but we consider some future improvements:

1. The Laboratory of Information Technologies at JINR plans to make a common file system for generic use, which is based on EOS [9]. We plan to test it as a possible alternative to the NFS share in the current solution.
2. CVMFS can probably be configured to increase the speed of publishing software. This will allow us to transfer some of the software and data from NFS.
3. Establish a CA to issue certificates on a per-node basis.
4. HTCondor has a Docker universe that allows one to run jobs in docker containers – setting it up will eliminate the need to build software for different operating systems, as nodes will simply run user-defined containers.

3. Distributed compilation

Modern scientific software is complex, and building it can also be a resource-intensive operation, e.g., building a ROOT framework [10] on a machine with 4 CPU cores takes about 45 minutes on average. Speeding up software builds through distributing the compilation process can also improve the user experience.

Not all software builds can be efficiently parallelized, and a user needs to decide whether it is worth it. In some cases, it is easy to decide by simply looking at the contents of the "Makefile", but in some other more complicated cases, it may require to profile the builds.

Figure 4 shows the build time of GENIE on one machine with a different number of CPUs: it is obvious that it does not parallelize well, and running it on a distributed system can make it even worse due to the network and distribution management overhead.

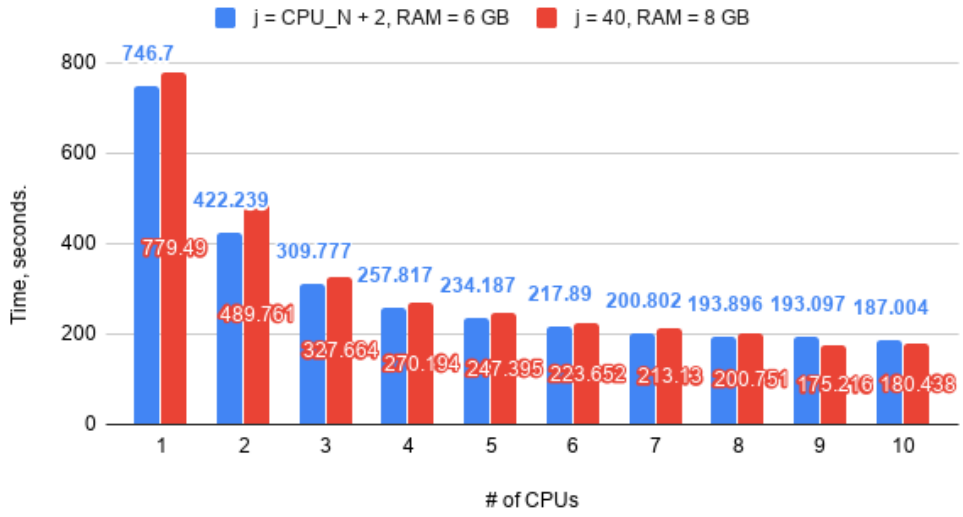

Figure 4. Parallel compilation of GENIE on one machine. The j-parameter is the number of parallel processes

On the contrary, Figure 5 shows an example of software compilation that scales well. The ROOT framework in this example is the base for many other scientific software packages, and compiling it can be efficiently distributed.

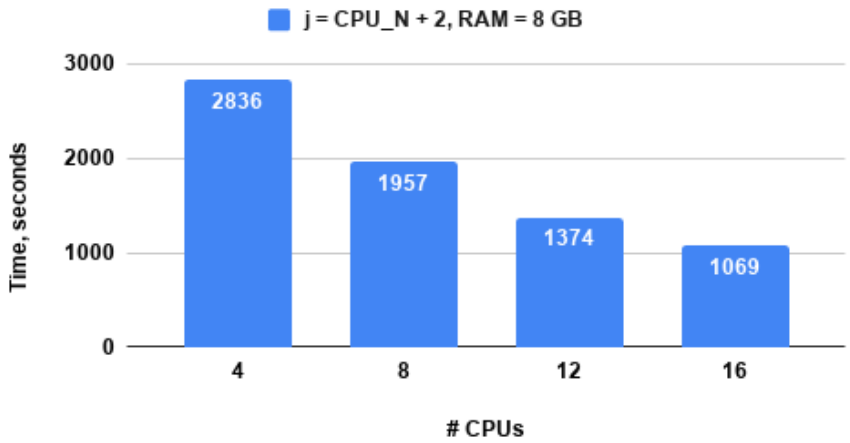

Figure 5. Distributed compilation of the ROOT framework. The j-parameter is the number of parallel processes

There are several compilers that allow distributed compilation, in our experiments we used DistCC [11], an open-source distributed compiler. We tried to use it with HTCondor directly, without any other auxiliary system; a possible workflow consists of several stages:

1. Check the number of available HTCondor nodes and the number of slots/cores available on each node to make a list.
2. Submit DistCC jobs multiple times to fit different slot sizes — there should be only one DistCC daemon on each running node.
3. Wait for some "reasonable" time for the DistCC daemons to start.
4. Make a list of IP-addresses/the number of slots and put it into the environment of the DistCC client to tell it how many processes each daemon is able to run.
5. Start the build as usual.
6. When the build is complete, remove all DistCC jobs from HTCondor to stop wasting resources.

Using DistCC and HTCondor, it is possible to distribute the compilation stage of the build process, while pre-processing and linking should be handled by the same dedicated machine, in our case, a PC of the submitting user.

This workflow can be seen as a proof of concept, but without any automated auxiliary system, the workflow is very complicated for the average user, as it requires a deeper understanding of how DistCC and HTCondor systems work.

4. Conclusions

In this paper, we described the idea of using a batch system as a flexible computing resource, gave an example of a typical workload and showed that such a system could be built of well-known components, such as HTCondor, CVMFS and NFS. We also described how software builds could be treated as a distributed workload, i.e. technically possible, but too complicated for ordinary users without better integration of a distributed compiler with a batch system. To further develop this idea, it is worth noting that using batch systems for distributed software compilation also looks as a promising technology to complement Continuous Integration systems.

5. Acknowledgments

The research is supported by the grant of the Russian Science Foundation (project № 18-12-00271).

References

[1] Computing with HTCondor. Available at: https://research.cs.wisc.edu/htcondor/ (accessed 28.10.2019).

[2] C. Andreopoulos et al. [GENIE Collaboration]. The GENIE Neutrino Monte Carlo Generator // Nucl. Instrum. Meth. A. 2010. V. 614. P. 87; C. Andreopoulos et al. [GENIE Collaboration]. The GENIE Neutrino Monte Carlo Generator: Physics and User Manual // e-Print: arXiv:1510.05494.

[3] Ph. Adamson et al. [NOvA Collaboration]. First measurement of muon-neutrino disappearance in NOvA // Phys. Rev. D. 2016. V. 93. P. 051104

[4] M. Betancourt. Study of Quasi-Elastic Scattering in the NOvA Detector Prototype // PhD thesis. 2013. Minnesota University; E. Arrieta Díaz. Observation of Muon Neutrino Charged Current Events in an Off-axis Horn-focused Neutrino Beam Using the NOvA Prototype Detector // PhD thesis. 2014. Michigan State University; B. Behera. Measurement of the Double Differential Inclusive Muon Neutrino Charged-current Cross-section in the NOvA Near Detector // PhD thesis. 2018. Indian Inst. of Technology Hyderabad.

[5] L. Aliaga Soplin, et al. [MINERvA Collaboration]. Design, Calibration, and Performance of the MINERvA Detector // Nucl. Inst. and Meth. A. 2014. V. 743. P. 130

[6] Ch. E. Patrick et al. [MINERvA Collaboration]. Measurement of the Muon Antineutrino Double-Differential Cross Section for Quasielastic-like Scattering on Hydrocarbon at Eν~3.5 GeV // Phys. Rev. D. 2018. V. 97. P. 052002

[7] I. Ruiz Simo, et al. Quasielastic charged-current neutrino scattering in the scaling model with relativistic effective mass // Phys. Rev. D. 2018. V. 97. P. 116006

[8] CERNVM File System (CernVM-FS). Available at: https://cernvm.cern.ch/portal/filesystem (accessed 28.10.2019)

[9] EOS – Open Storage Documentation. Available at: http://eos-docs.web.cern.ch/eos-docs/ (accessed 28.10.2019)

[10] ROOT, an Object-Oriented Data Analysis Framework. Available at: http://root.cern.ch (accessed 28.10.2019)

[11] Distcc – a free distributed C/C++ compiler system. Available at: https://github.com/distcc/distcc (accessed 28.10.2019)