NEC'2019
Accelerating personal computations with HTCondor: large number events generation with GENIE

Nikita Balashov [1] Igor Kakorin [2] Kostantin Kuzmin [2,3,4]
Vadim Naumov [2]

[1]Laboratory of Information Technologies, Joint Institute for Nuclear Research, RU-141980 Dubna, Russia

[2]Bogoliubov Laboratory of Theoretical Physics, Joint Institute for Nuclear Research, RU-141980 Dubna, Russia

[3]Institute for Theoretical and Experimental Physics, RU-117259 Moscow, Russia

[4]Kurchatov Institute, RU-123182 Moscow, Russia

01th of October 2019

What is GENIE (http://www.genie-mc.org/)

Genie Event Generator & Global Analysis of Neutrino Scattering Data

Home

Mission statement
GENIE collaboration
Policy documents
Copyright notices
Citing GENIE
Logos

Public releases
Global fits & physics tunes
Naming conventions
Associated data releases

User forum
Project incubator

Physics & user manual
Document database

Slack workspace

User mailing list
Developer mailing list

GitHub organization page

Get started
GENIE course

News

GENIE is an **international collaboration** of scientists that plays the leading role in the development of **comprehensive physics models** for the **simulation of neutrino interactions**, and performs a highly-developed **global analysis of neutrino scattering data**.

The GENIE collaboration **maintains a popular suite of software products** (including the ***Generator***, ***Comparisons***, ***Tuning***, and ***Reweight*** products) for the experimental neutrino community.

The well-known *Generator* implements a modern framework for Monte Carlo event generators and includes state-of-the-art physics modules. The **GENIE physics model is universal and comprehensive**: It handles all neutrinos and targets, and all processes relevant from MeV to PeV energy scales. The *Generator* includes several tools (flux drivers, detector geometry navigation drivers, and specialized event generation apps) to **simulate complex experimental setups in full detail**. The GENIE *Generator* is **used by nearly all modern neutrino experiments** and its predictions serve as standard reference points for the neutrino community.

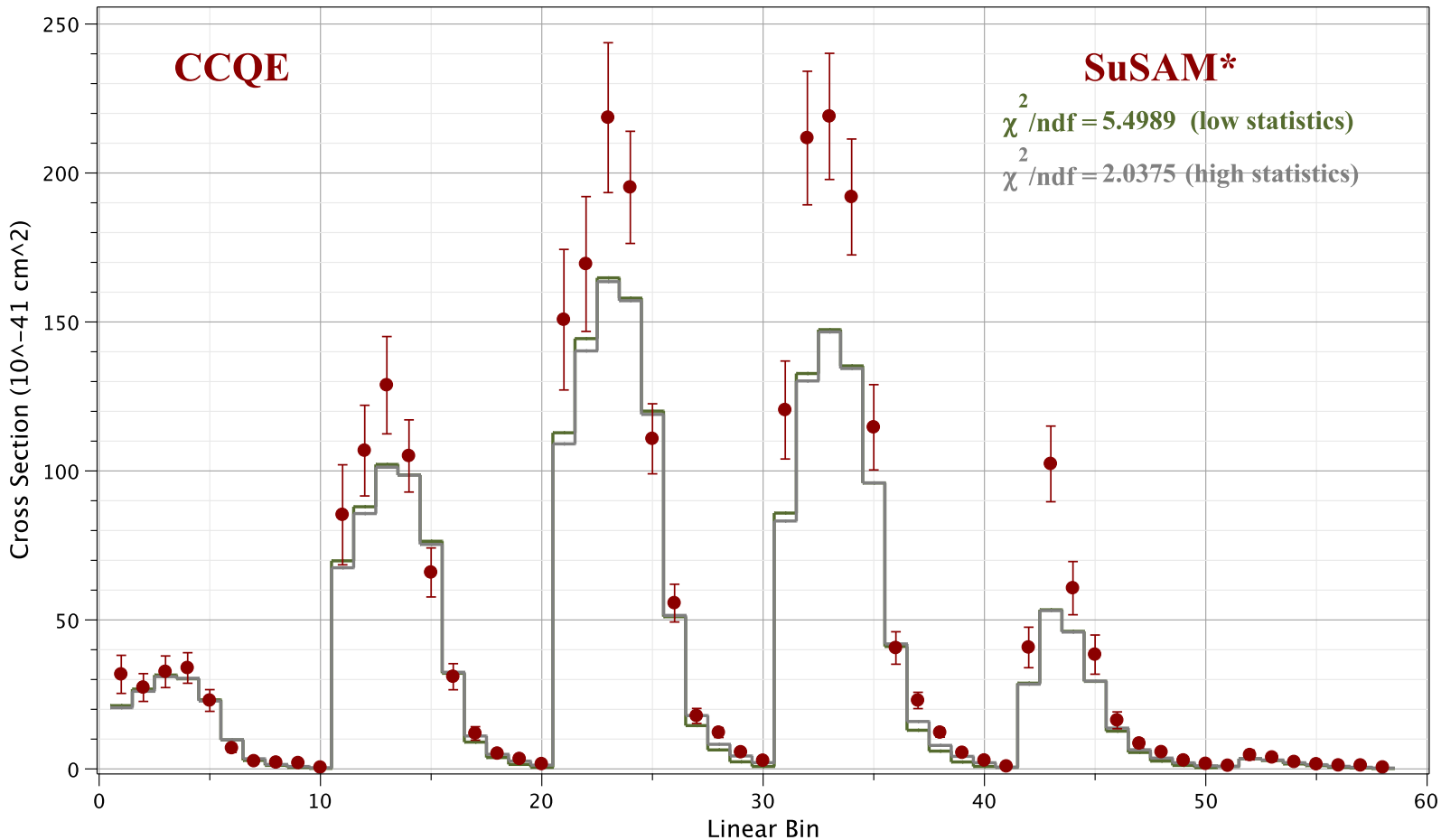
The *Comparisons* product includes very **extensive curated archives of neutrino, charged-lepton and hadron scattering data**, as well as highly-developed software to produce a **comprehensive set of data/MC comparisons**. It includes embedded **interfaces to the Professor tuning tool** which "*reduces the exponentially expensive process of brute-force tuning to a scaling closer to a power law in the number of parameters and allows for massive parallelisation*". The *Comparisons* product plays a key role in comprehensive model characterization in GENIE, it underpins the GENIE global analysis, and it enabled the production of several new tunes.

The *Tuning* product implements the **powerfull new GENIE global analysis of neutrino scattering data**. The GENIE global analysis produces **physics tunes which are fully integrated in the *Generator* product**.

Finally, the *Reweight* product includes a selection of tools to **propagate model uncertainties and to support generator-related analysis tasks**. The reweighting procedure has inherent limitations. Important modelling systematics are not reweightable in principle and they have no corresponding weight calculator in the *Reweight* product. Indeed, the GENIE tuning procedure itself makes no use of the Reweight product but it relies on response functions constructed from brute-force parameter scans made with the aid of the Professor tool. Currently, the *Reweight* product it does not provide the full systematic error for any GENIE tune. However, we have **medium-term plans to overhaul this product** and use it for **public release of the detailed Professor/YODA response functions** constructed from our brute-force systematic parameter scans, as well as to release all covariance matrices from the GENIE global fits of neutrino scattering data. The upgraded *Reweight* product will support all public GENIE physics tunes!

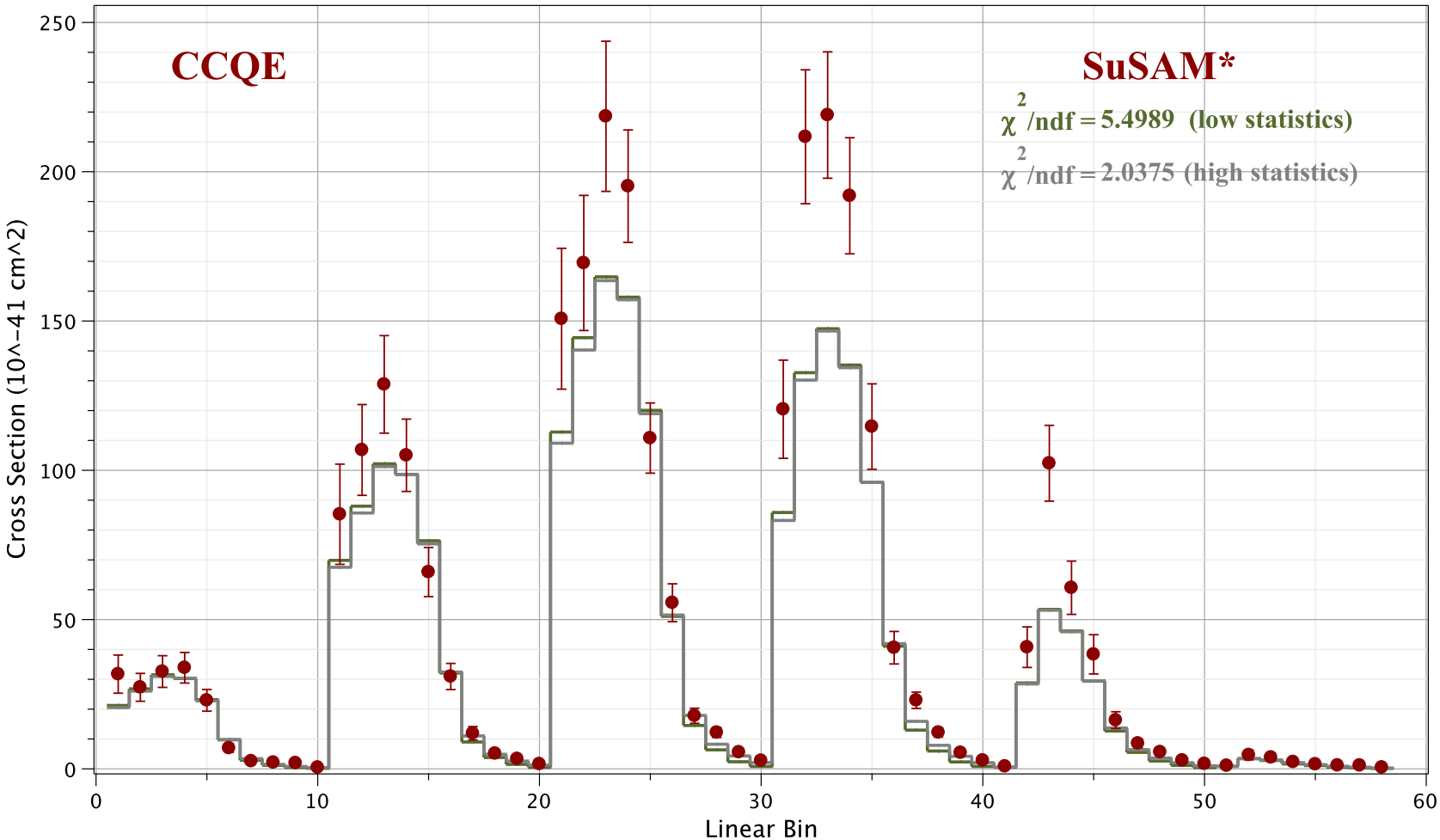
Why generation of large number of events is important?

The comparison of experimental data and **GENIE** predictions generated with high/low statistics. The points with bars shows MINERνA data ($\bar{\nu}_\mu$-scattering on hydrocarbon) with systematics errors. The histograms shows predictions of the super-scaling model with relativistic effective mass (Phys.Rev. D97 (2018) no.11, 116006) calculated with GENIE.



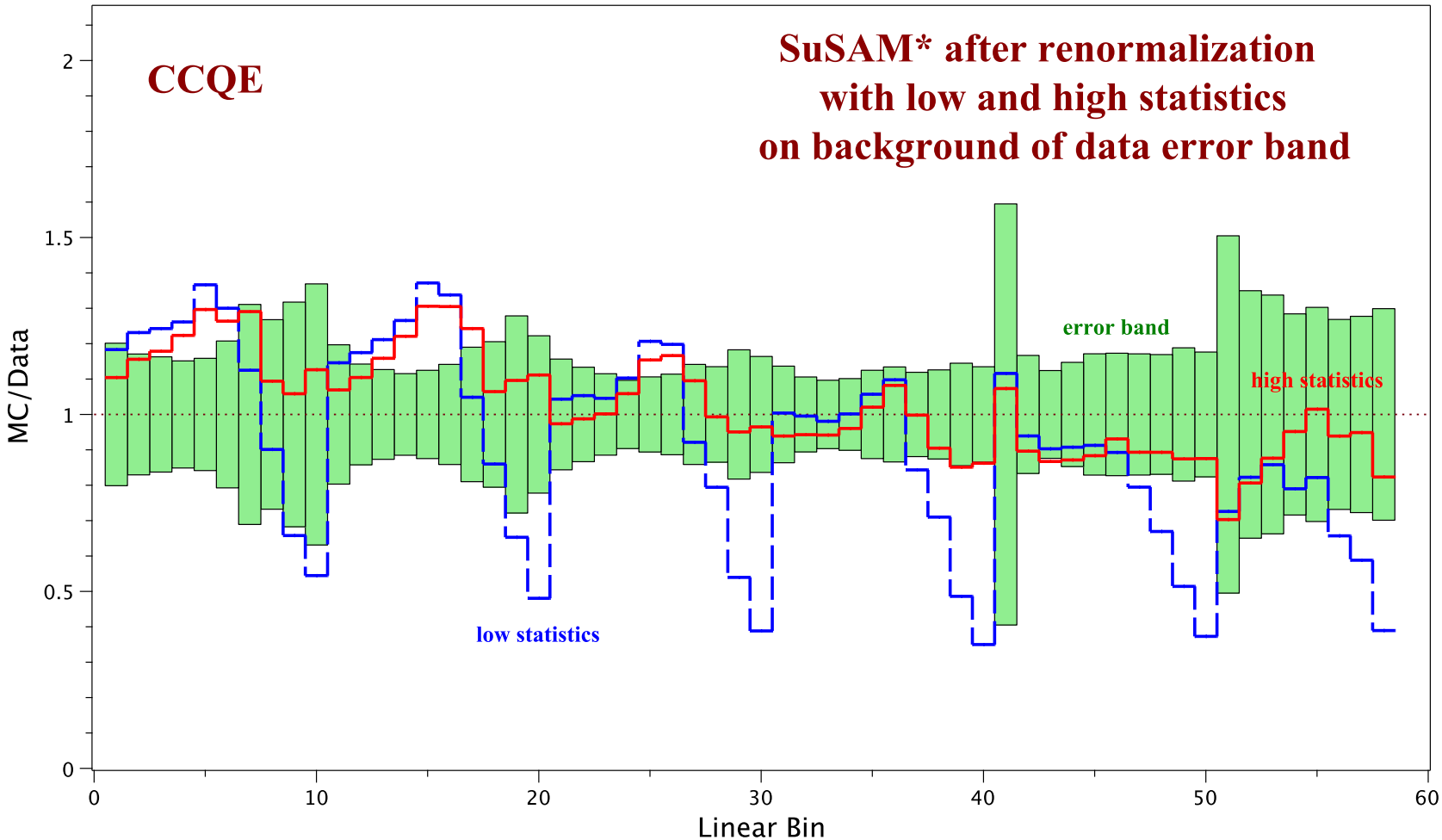
Why generation of large number of events is important?

$\frac{\text{Number of events for high statistics histogram}}{\text{Number of events for low statistics histogram}} \approx 5$ and difference between histograms is barely visible by eye, but χ^2-s differ significantly.



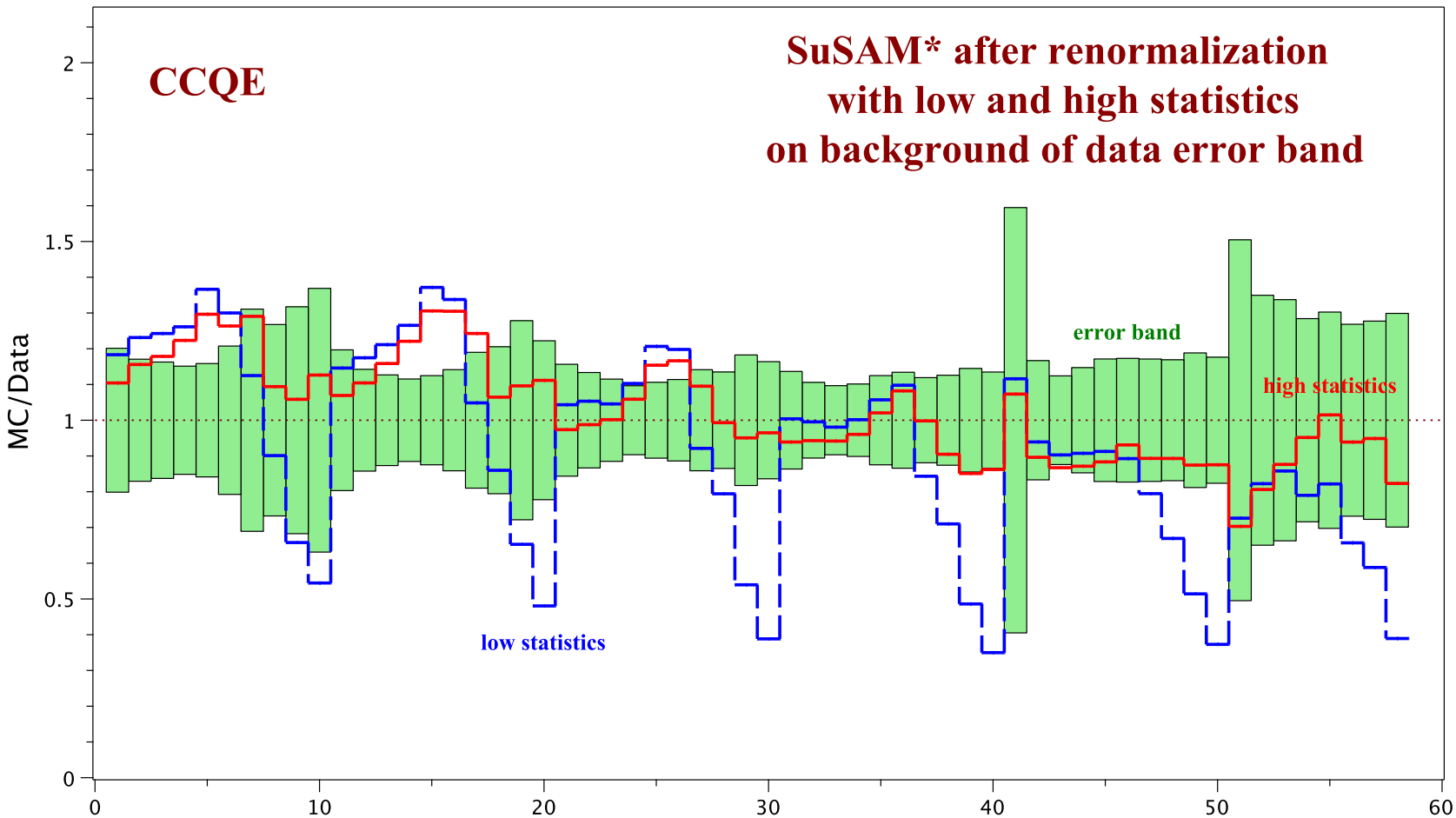
Why generation of large number of events is important?

Why there is such a difference become clear from this representation of the same data. Here we renormalized histograms and divided them by data. The error band represents the relative systematics errors.



Why generation of large number of events is important?

The dips on the histogram are due to the cross section for the corresponding bins are small, but the weight of an event in these bins are large: if the number of generated events is small there is almost no chance for an event to get in dips, while data contains such events. That is why the χ^2 for low statistic histogram is so large.

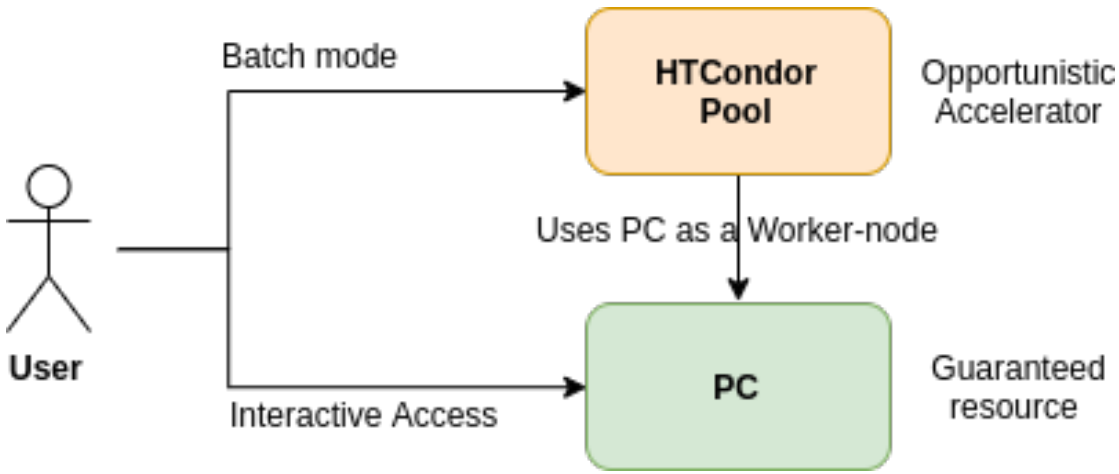


Times of events generation

It takes about 576/115 CPU hours to generate high/low statistics histogram. There is a need to generate tens of such histogram to compare theoretical prediction with only one experiment. It takes not less than **10 days** to generate all histograms with sufficient statistics at a workstation with **24 CPU** (if there are no errors in the calculation), it is **unreasonably long**. If one use **HTCondor** with **576 CPU** then it will take only **10 hours**.

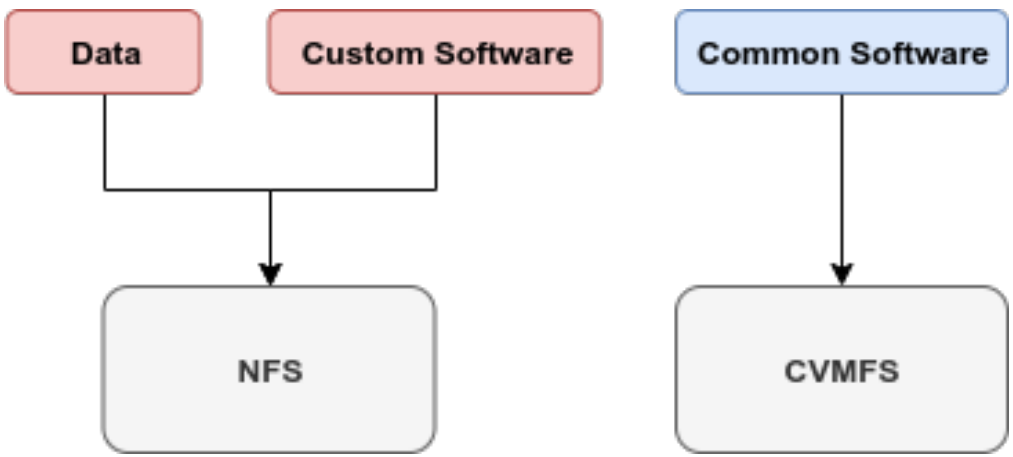
Combine HTCondor with a PC

- A guaranteed maximum execution time (limited by a user's personal machine)
- Possible decrease of the execution time depending on the availability of the batch-cluster resources
- A unified submission system - no need to start the workload on a personal computer differently than on the cluster
- Major drawback: more complicated software distribution



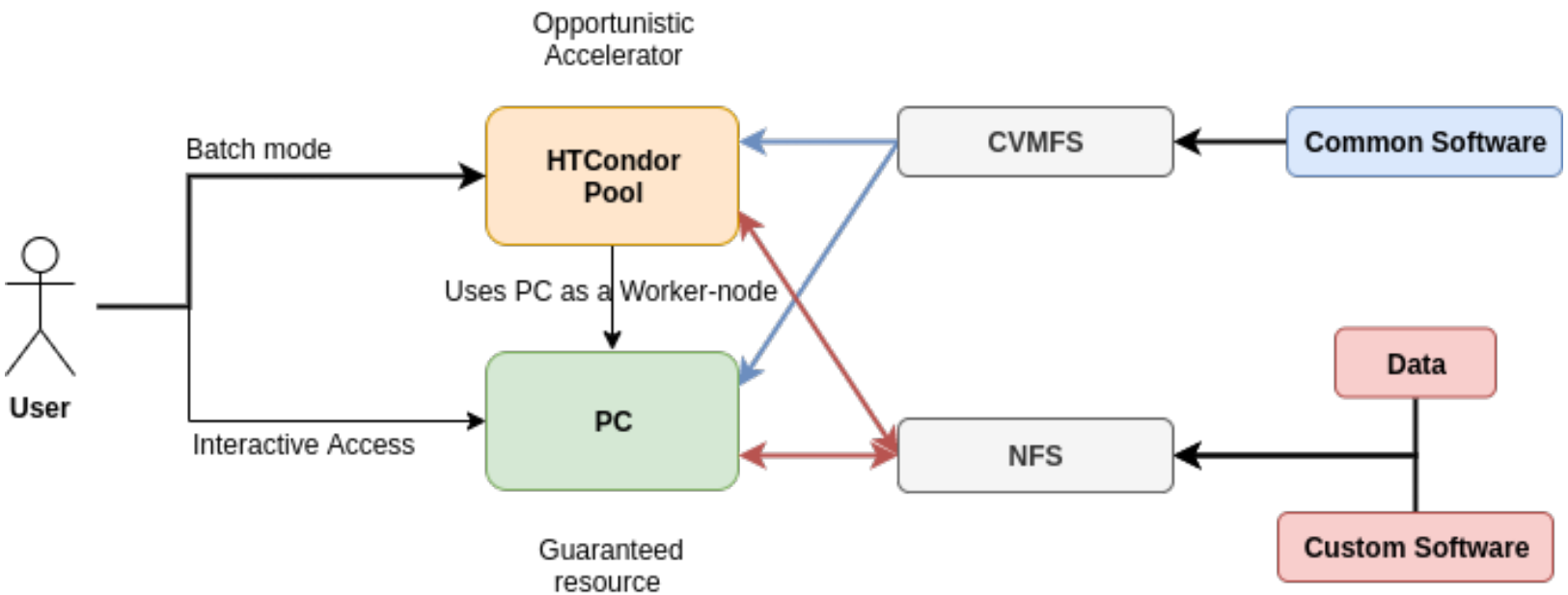
Software distribution

- Users want their software be ready to use as soon as possible after being built
- PC operating system most likely to differ from the cluster's
- We came up with using two systems:
  - NFS share for some data and software that changes often
  - CVMFS for common software (slow release cycle)



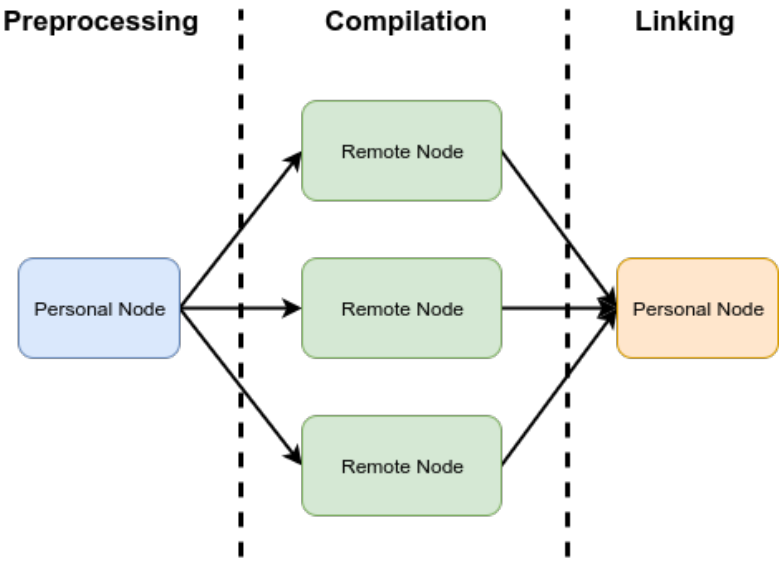
Test Environment

- A shared SSL certificate is used to authenticate user node against the central manager to join the HTCondor cluster
- The PC is a cloud virtual machine running Debian 8
- The whole HTCondor cluster runs Scientific Linux 6
- NFS and CVMFS are mounted on all the worker-nodes and the user's machine

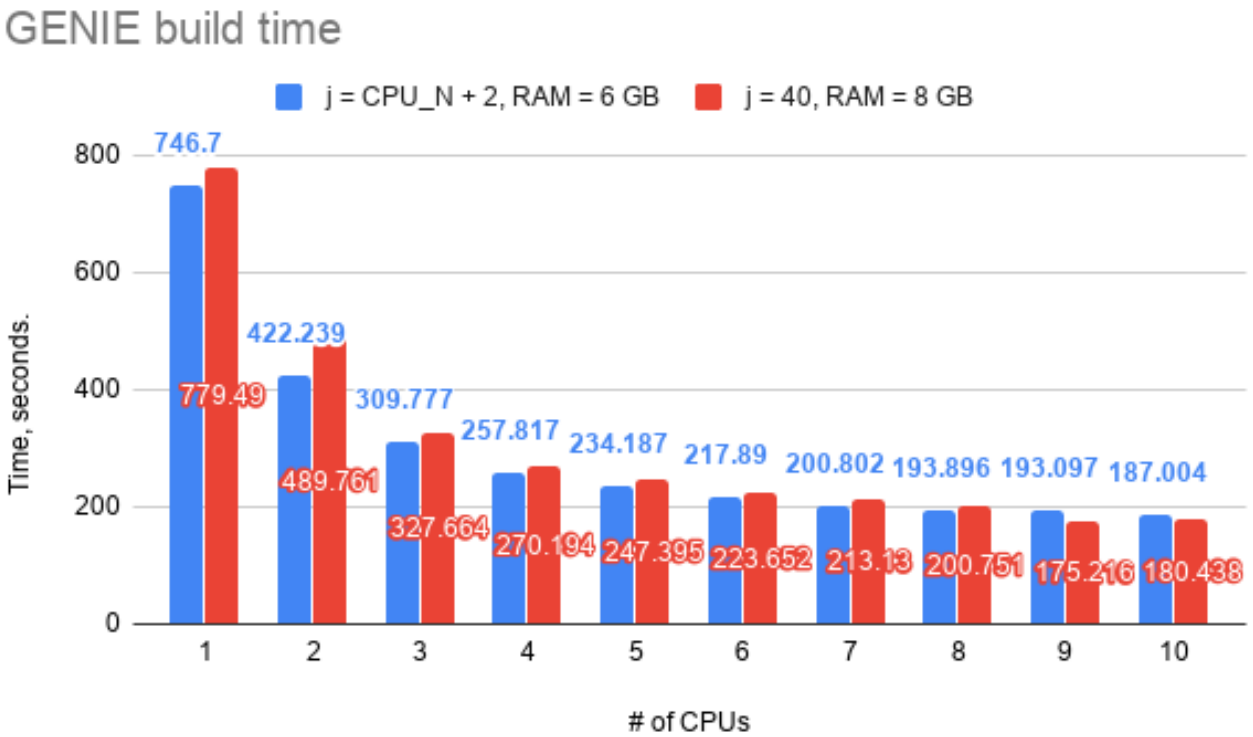


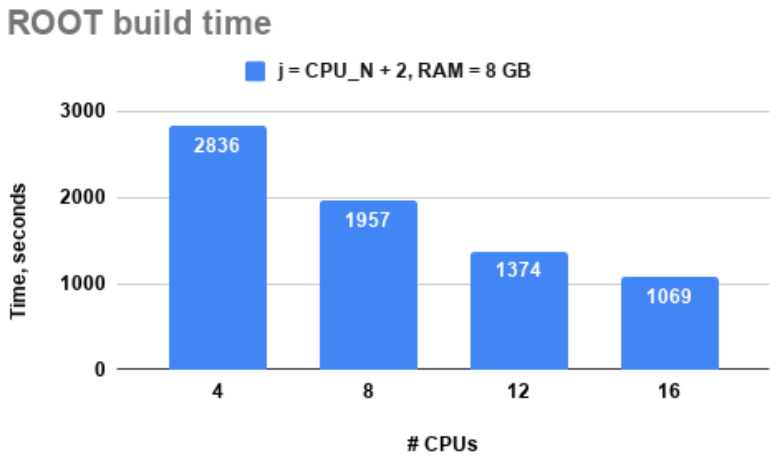
Going more exotic: distributed compilation

- Building the software can also be a resource-intensive operation and can be distributed (at least some stages of it)
- There are several compilers that can do it, the most popular are:
  - DistCC
  - Icecream
  - DMUCS
- Can we leverage HTCondor?



Software builds scalability





DistCC with HTCondor

- There's no easy way to run multiple DistCC daemons on a single node, so we need to submit multicore jobs
- Limit DistCC daemons to the number of slots acquired
- Its good to know the number of available CPUs beforehand to chose the optimal "-j" for the *make*
- When compiling is finished the DistCC daemons need to be stopped manually
- **Ideally**, client environment should match the cluster environment
- A bunch of wrapper scripts can automate most of these tasks, **but...**
- All this makes it a bit **too** complicated for the ordinary user to make use of it

Possible DistCC-HTCondor workflow

- Check the number of free HTCondor nodes and the number of slots/cores available on each of the node and make a list
- Submit multiple times jobs of different slot sizes
- Wait some "reasonable" time for the DistCC daemons to start
- Make the ip-address/number_of_slots list and put it into the env of the DistCC client
- Start the build
- When the build has finished, remove all of the condor jobs

TODOs

- Consider other options for software distribution:
  - Substitute NFS with EOS
  - Fine-tune CVMFS for higher publication rates
  - Use Docker Universe
  - Establish the CA to issue certificates on per-node basis
- Facilitating software builds with HTCondor feels to complex normal users, but looks promising for the automated system to facilitate Continuous Integration

Thanks!