Workfest Announcement

Optimization of Monte Carlo production for physics analysis

Monte Carlo simulation for physics analysis can be very **expensive** in terms of **computing resources** and **wall time**.

When planning a large Monte Carlo production campaign, it's important to know **how long** it may take and, if necessary, leverage **smart techniques** to reach the required statistical precision.

Several options are available, from **event weighting** to populate low-efficiency and low-cross-section regions, to **fast Monte Carlo** based on an AI model of the detector acceptance.

During this Workfest session, we will explore these ideas with the help of collaborators who have tackled these kinds of problem before. We invite everyone who feels Monte Carlo simulation for their analysis may be too expensive to bring their use case and explore with us how these alternative options could be applied.

How long is going to take?

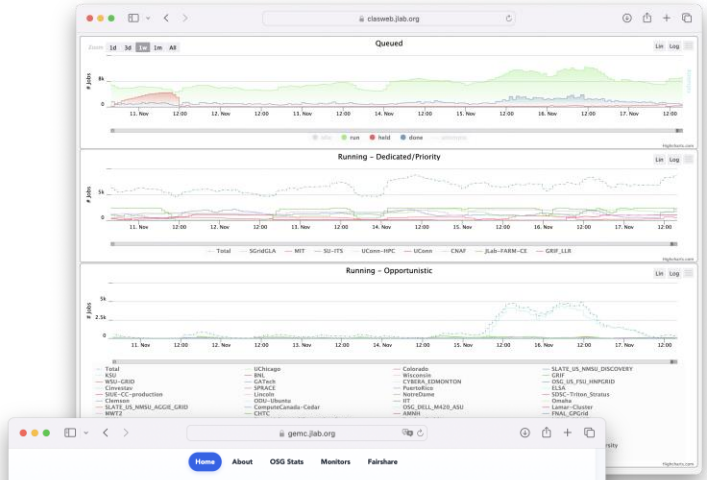
"I need to simulate 1B events…"

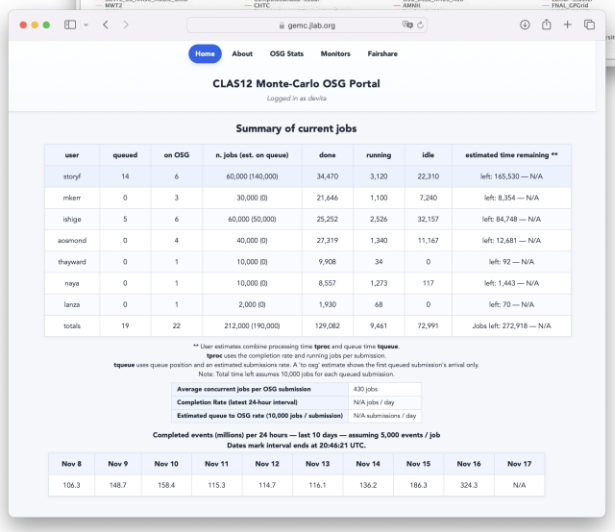
- Full GEANT4 simulation (gemc) + reconstruction (coatjava) of 1 event can take 1-2 s of CPU time, depending on the topology and other factors:
  - 1B events / 1 event/s ~ 0.3M Core hours ~11000 Core days

"But I can run multiple jobs in parallel"

- On OSG, we typically run several thousand jobs in parallel and produce ~200M events per day shared among the users submitting jobs. For 40 users:
  - 1B events / 200M events/day x 40 = 200 days

https://clasweb.jlab.org/clas12offline/osg





https://gemc.jlab.org/web_interface/index.php

How long is going to take?

"I need to simulate 1B events…"

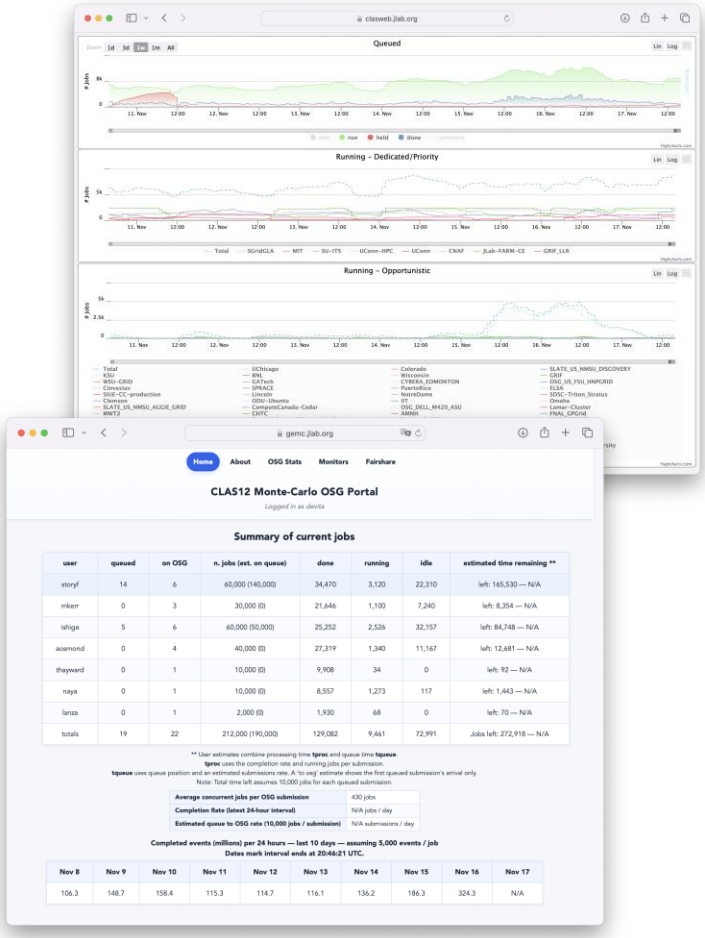
- Full GEANT4 simulation (gemc) + reconstruction (coatjava) of 1 event can take 1-2 s of CPU time, depending on the topology and other factors:
  - 1B events / 1 event/s ~ 0.3M Core hours ~11000 Core days

"But I can run multiple jobs in parallel"

- On OSG, we typically run several thousand jobs in parallel and produce ~200M events per day shared among the users submitting jobs. For 40 users:
  - 1B events / 200M events/day x 40 = 200 days

Do you really need 1B events?
Do you need full GEANT4 simulations?

https://clasweb.jlab.org/clas12offline/osg



https://gemc.jlab.org/web_interface/index.php

Reducing the number of events

- The MC statistics is often chosen to have the number of reconstructed events $N_{rec} >> N_{data}$, to make the MC statistical uncertainty negligible compared to the one from data
  - Great, if possible, but making the MC statistical uncertainty negligible with respect to the total expected uncertainty (including systematics) be sufficient

Reducing the number of events

- The MC statistics is often chosen to have the number of reconstructed events $N_{rec}>>N_{data}$, to make the MC statistical uncertainty negligible compared to the one from data
  - Great, if possible, but making the MC statistical uncertainty negligible with respect to the total expected uncertainty (including systematics) be sufficient
- Generated events can be filtered to exclude the ones that are well outside the CLAS12 acceptance
  - E.g. drop events where the electron is below 5-6 degrees

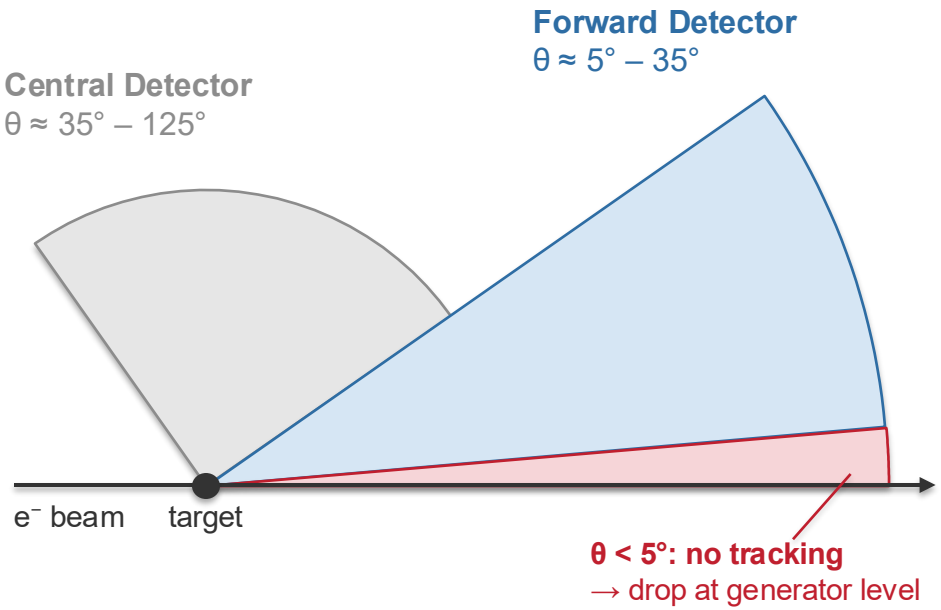
Reducing the number of events

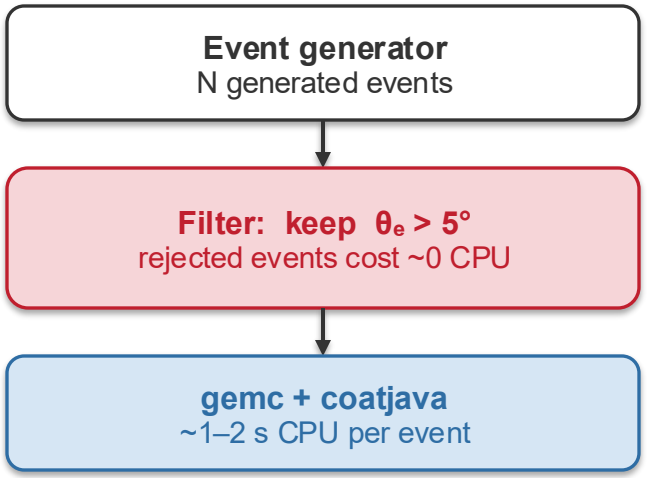
- The MC statistics is often chosen to have the number of reconstructed events $N_{rec} >> N_{data}$, to make the MC statistical uncertainty negligible compared to the one from data
  - Great, if possible, but making the MC statistical uncertainty negligible with respect to the total expected uncertainty (including systematics) be sufficient
- Generated events can be filtered to exclude the ones that are well outside the CLAS12 acceptance
  - E.g. drop events where the electron is below 5-6 degrees
- When the event generator cross section has strong variations over the phase space of interest, e.g. Q^2 dependence, populating low cross-section regions results in simulating most of the events in the high cross-section regions
  - Event weighting can be used to “flatten” the cross section and generate more events where necessary

Reducing the number of events - 1

- Generated events can be filtered to exclude the ones that are well outside the CLAS12 acceptance
  - E.g. drop events where the electron is below 5-6 degrees

CLAS12 polar-angle coverage (side view, schematic)





CPU saved ∝ fraction filtered out
with no loss in reconstructed statistics

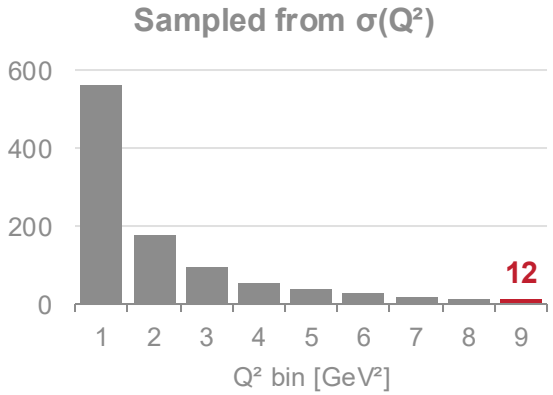
Keep small-angle electrons if the analysis uses the Forward Tagger (θ ≈ 2.5°–4.5°)

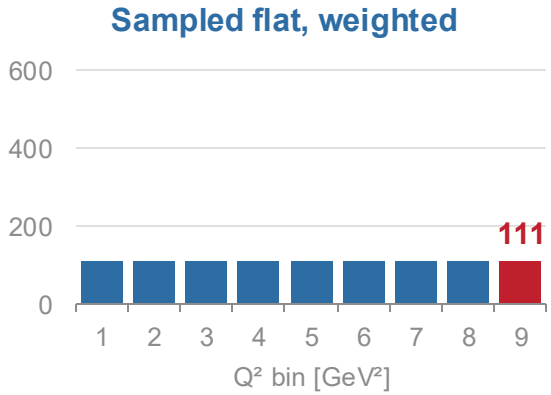
Diagram and flowchart prepared with Claude (Anthropic AI)

Reducing the number of events - 2

- When the event generator cross section has strong variations over the phase space of interest, e.g. Q^2 dependence, populating low cross-section regions results in simulating most of the events in the high cross-section regions
  - Event weighting can be used to “flatten” the cross section and generate more events where necessary

Same 1000 generated events, two sampling strategies (schematic, $\sigma \propto 1/Q^4$)





Event weight

$w = \sigma(Q^2) / g(Q^2)$

g = sampling density (flat)

Fill histograms with w → physical distribution recovered

Stat. error: $\sqrt{\Sigma w^2}$

Highest-Q^2 bin: 12 → 111 events (≈9×)

Charts and weight box prepared with Claude (Anthropic AI)

Reference Numbers

- These are all ballpark/average numbers
  - the truth depends on luminosity and generator
- Hall B's entire (pro) fairshare at JLab is currently 3.5k (2.5k) CPUs
  - ~500 million reconstructed real CLAS12 data events per day
- CLAS12 OSG dedicated resources are in the same ballpark
  - ~200 million fully-simulated CLAS12 events per day
- Event sizes in the CLAS12 simulation pipeline
  - LUND (EVIO) ⇨ HIPO-raw ⇨ HIPO-full* ⇨ HIPO-dst (*unnecessary temporary footprint)
  - 1kB (30kB) ⇨ 10 kB ⇨ 100 kB ⇨ 1 kB
  - Note, intermediates not shown (and EVIO is not in simulation)
    - background merging, old-school decoding & denoising
  - *Assuming you only keep the DSTs:*
    - 1 GB per million events, 1 TB per billion events
    - *Keeping all intermediates would explode*

farm25	2025	30	30	dual 32-core AMD EPYC 9354 (Genoa)	576 GB	dual 1.7 TB NVMe	EDR IB
farm23	2023	25	25	dual 64 core AMD EPYC 7763 (Milan)	512 GB	dual 3.5 TB SSDs	EDR IB
farm19	2019-2021	107	99	dual 32 core AMD EPYC 7502 (Rome)	256 GB	1 TB SSD	FDR IB
farm18	2018	88	34	dual 20 core 2.4 GHz Xeon (Skylake)	96 GB	480 GB SSD	FDR IB

Coatjava Fast MC – Side Note

- Coatjava has a conventional Fast MC
  - Field swimming plus detector geometry intersections
  - But only implemented for a few detectors (and no resolution smearing)
  - This could be resurrected
- Coatjava later got "trajectories" for real data, also based on the same swimming and detector intersections
  - This could easily be extended to new trajectories for MC particles, and then extended to a Fast MC with common fiducial cuts on the new trajectory bank
  - This would probably be pretty easy