CHEP 2018
23RD INTERNATIONAL CONFERENCE ON
COMPUTING IN HIGH ENERGY AND NUCLEAR PHYSICS
9-13 July 2018
National Palace of Culture
Sofia, Bulgaria

Status of parallelized JUNO simulation software

Simon Blyth, Guofu Cao, Ziyan Deng, Xingtao Huang,
Weidong Li, Tao Lin, Zhengyun You, Jiaheng Zou
(on behalf of the JUNO collaboration)

CHEP 2018
9-13 July 2018
National Palace of Culture, Sofia, Bulgaria

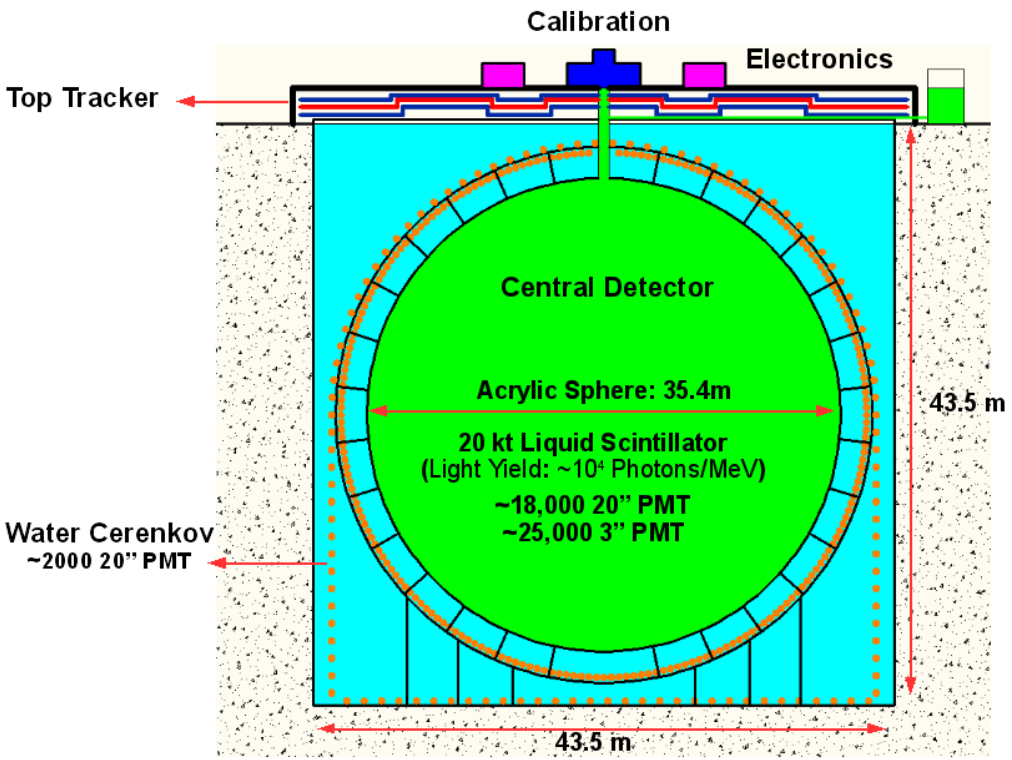
Outline

- JUNO experiment
- JUNO offline software
- Parallelized simulation software
- Summaries and plans

JUNO experiment

- The JUNO (Jiangmen Underground Neutrino Observatory) is designed to determine neutrino mass hierarchy and precisely measure oscillation parameters.
- Located in southern China, 53 km away from Yangjiang and Taishan nuclear power plants.
- Under construction.
- Data taking expected in ~2020.
- JUNO Collaboration
  - 72 institutes
  - ~590 collaborators

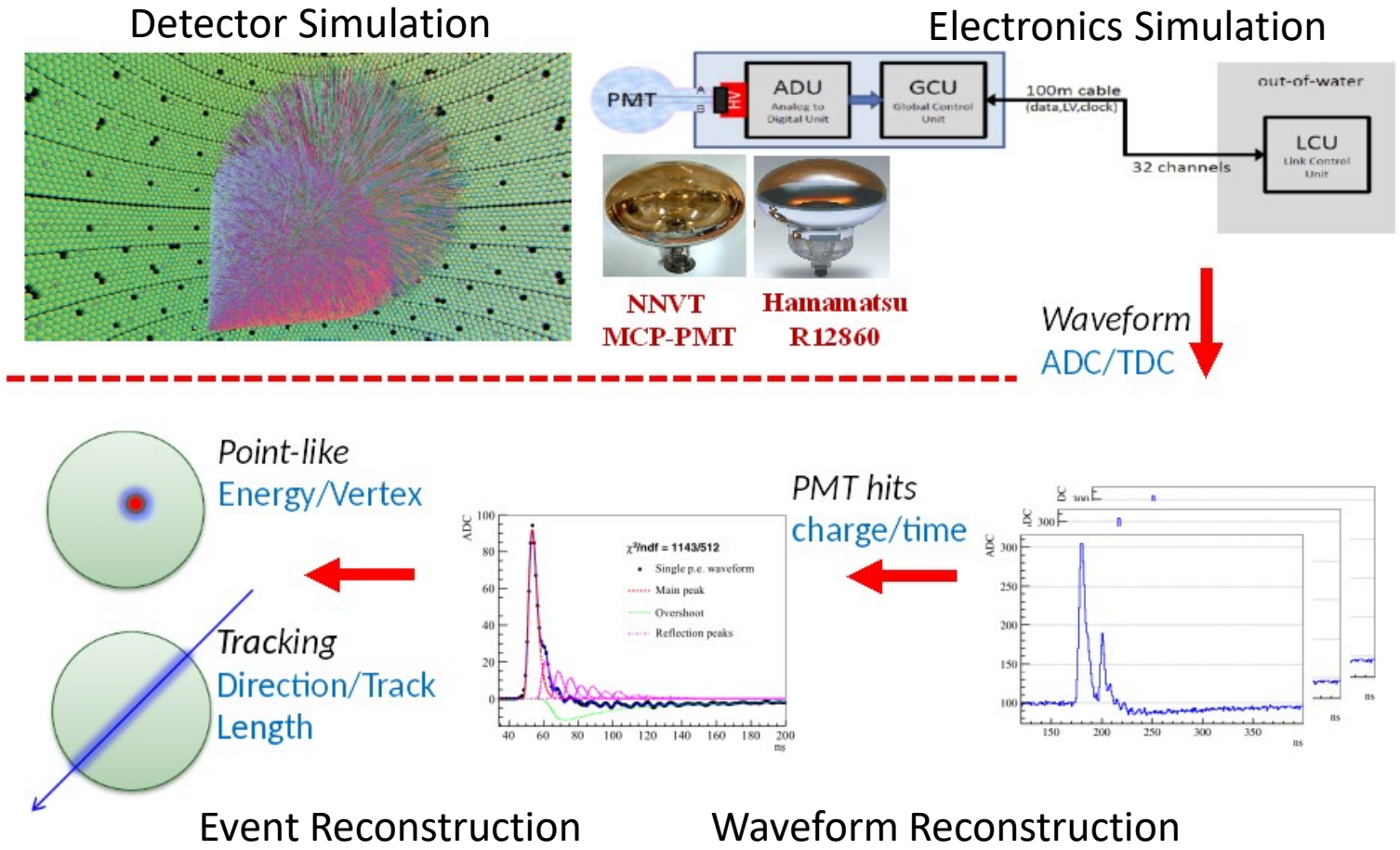
JUNO detectors



- Experiment hall: under 700 m rock
  - Shield cosmic rays
- Neutrino detector: 20 kton liquid scintillator and ~18 k 20 inch PMTs and ~25 k 3 inch PMTs.
  - ~10,000 photons/MeV light yield
  - 3% energy resolution @ 1 MeV
- Veto detectors: Water Cherenkov detector and Top Trackers.
  - Water Cherenkov: shield radioactivity and muon tracking.
  - Top Tracker: independent muon tracking.

JUNO offline data processing

- Detector simulation
  - Neutrino signals
  - Cosmic ray muons
  - Radioactivity
- Electronics simulation
  - Effects of PMTs, electronics
  - FADC @ 1 GHz
- Waveform reconstruction
- Event reconstruction
  - Point-like (CD)
  - Tracking (CD, WP, TT)



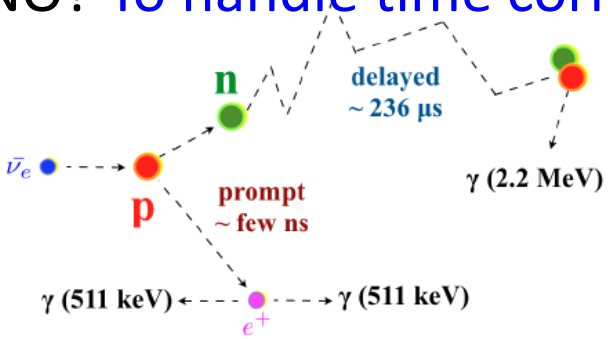
JUNO offline software

- JUNO offline is developed based on SNiPER framework, which consists of
  - framework software, event data management, geometry management, event display, simulation, reconstruction, calibration and analysis.
- SNiPER: (Software for Non-collider Physics ExpeRiment)
  - A general purpose, but lightweight data processing framework.
  - It's already adopted by JUNO (neutrino physics), LHAASO (astroparticle physics), nEXO (neutrinoless beta decay), CSNS (spallation neutron source)
- Why use SNiPER in JUNO? To handle time correlation events.

Inverse Beta Decay (IBD)

- Prompt signals: e+
- delay signals: n capture

One physics event becomes two triggers.



In physics analysis, need to find the pair of prompt and delay signals.

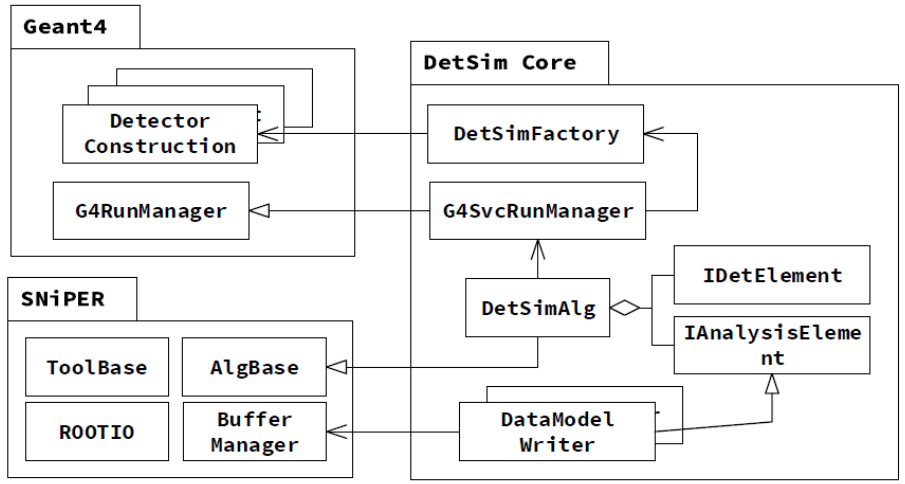
JUNO offline software release

- About 10 official versions have been released from 2013.
  - Two major versions every year.
- In 2018, we are upgrading software stacks to support parallel computing.

	J18v1	J18v2
Language	C++, Python 2	C++ 11, Python 2
OS and compiler	SL 6 and gcc 4.4	SL 6/7 and gcc 4.8+
Framework	SNiPER (prototype)	SNiPER (1.0, LGPL)
Geant4	9.4.p04	10.04.p02
ROOT	5.34.11	6.12.06

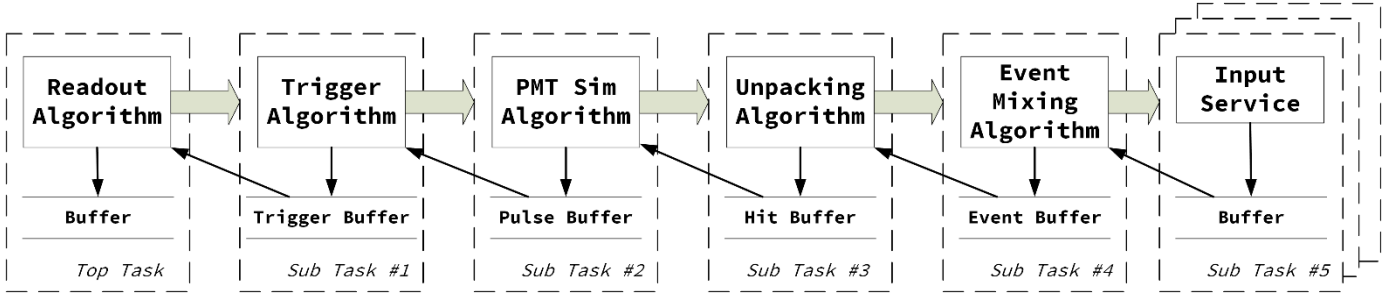
JUNO simulation software

- Detector simulation software is developed based on SNiPER and Geant4 9.4.
- Electronics and digitization simulation adopts “PULL” workflow, which handles event mixing and splitting gracefully.



Detector simulation framework

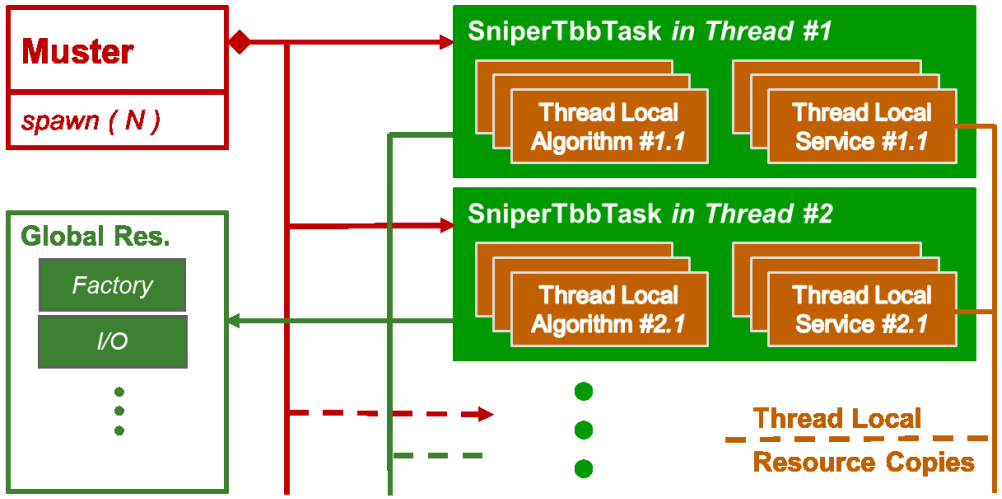
Start from readout algorithm.
Invoke corresponding algorithms on demand.



Each algorithm is associated with a task, which is executed passively.

Evolution of SNiPER framework

- The original SNiPER is serial. SNiPER framework group starts the development of parallelized version in 2017.
- Event level parallelism.
- Developed based on Intel TBB.
  - Muster: Multiple SNiPER Task Scheduler.
  - SniperTbbTask: Binding of a SNiPER Task to a TBB task.

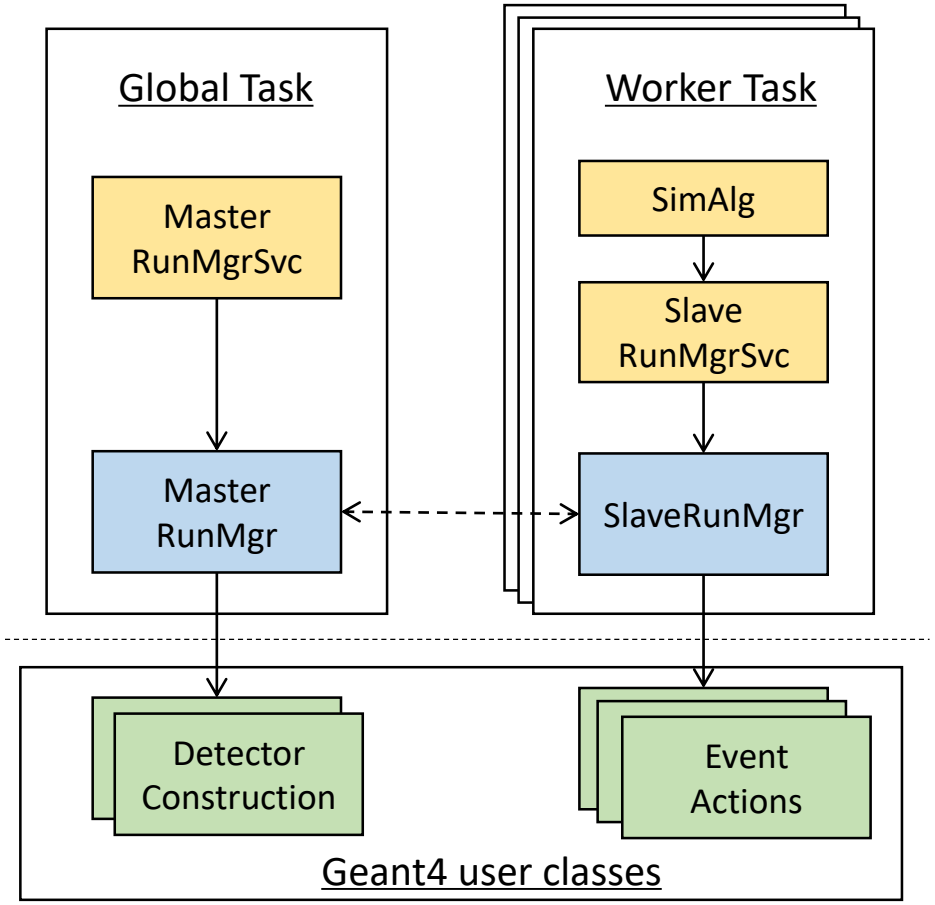


- Global buffer is designed for correlated event management.
  - Purpose: events with correlation need to keep the correct order.
  - Use a global buffer to maintain the order. Two extra threads for I/O.
  - For each worker thread, the local buffer is still used by the algorithms.

See Jiaheng's talk [531].
T5 S7 15:00, Thursday 12 July

Parallelized simulation framework

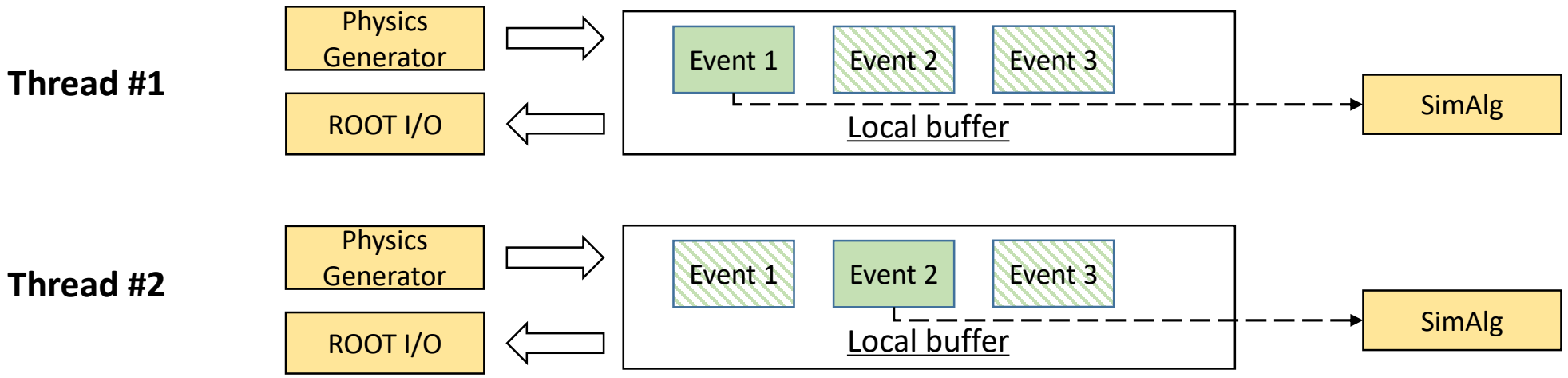
- Parallelized JUNO simulation is necessary.
  - In order to use computing and memory resource efficiently, experiments such as ATLAS and CMS already adopt multi-threading model.
  - Development of parallelized version is started in 2017.
- The framework is updated based on SNiPER Muster.
  - SNiPER Muster is in charge of event loops.
  - Integrate with the Geant4 kernel by extending the existing run managers.
  - A global task initializes geometry and physics lists.
  - Multiple worker tasks manage the simulation of events.



Benefits from Geant4's existing thread safe code, such as random number engines.

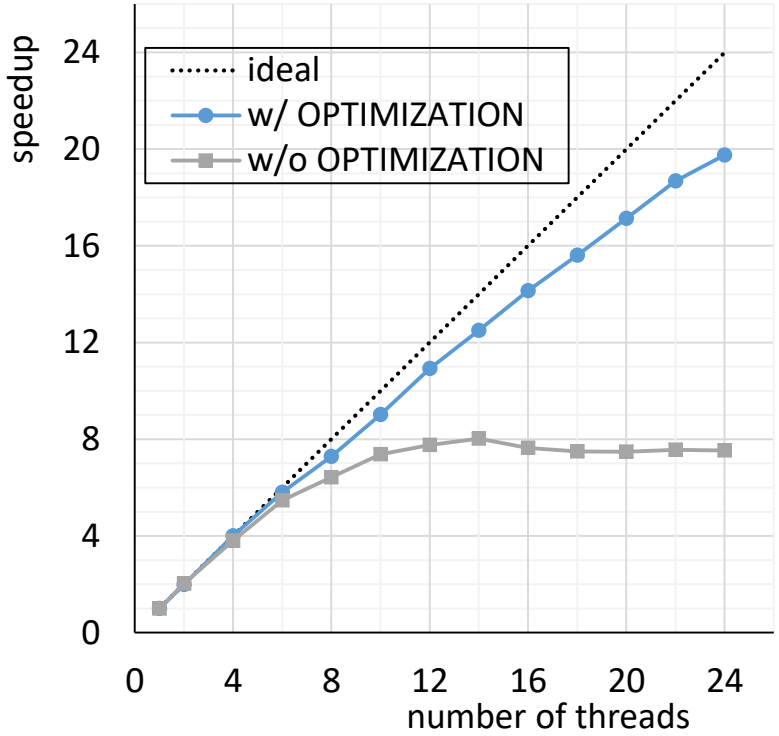
Simulation without global buffer

- If the order is not important, use the thread local algorithms and services.
  - Each thread has its own I/O service, buffer service, generator and simulation algorithms.
- The events in local buffers are independent from others.
- If ROOT I/O is enabled, each thread will save the result into its own file.



Simulation without I/O and global buffer

- In order to eliminate I/O interference, the simulated events are not saved.
- When using Geant4 10.3, a performance issue is found, as shown in right figure.
  - With the help of Intel's VTune Amplifier profiler, we found the hotspot comes from a mutex used in Geant4's material properties table.
  - The problem is that there are a lot of photons in JUNO simulation, so this mutex is accessed frequently to get the velocity.
- This issue is also fixed in the latest Geant4 10.4.
  - Thanks to Soon Yung Jun (Fermilab)



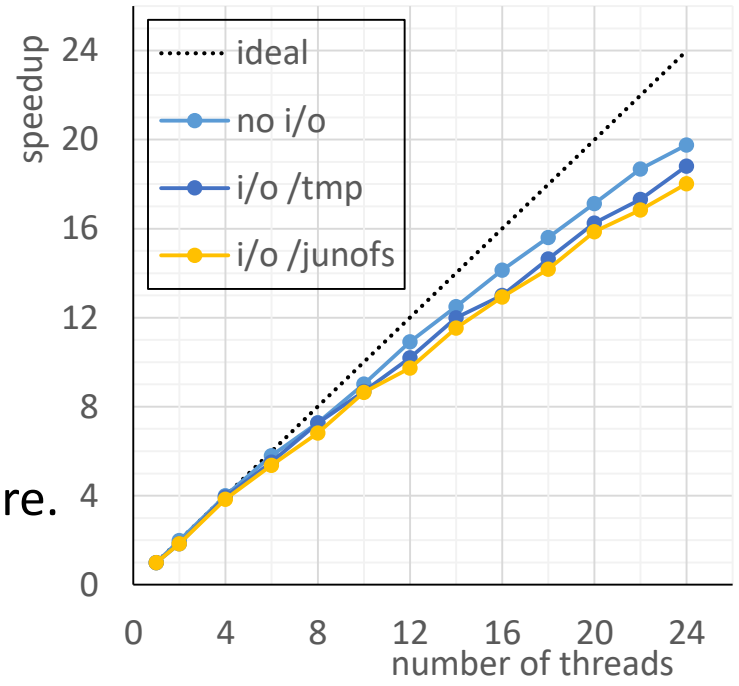
An example (2.2 MeV gamma, 1000 events, 24 threads):

```
OLD: User=283.22s Real=31.25s Sys=430.69s
NEW: User=236.44s Real=10.53s Sys=2.84s
```

The sys time is abnormal before the bug is fixed.

Simulation with I/O (without global buffer)

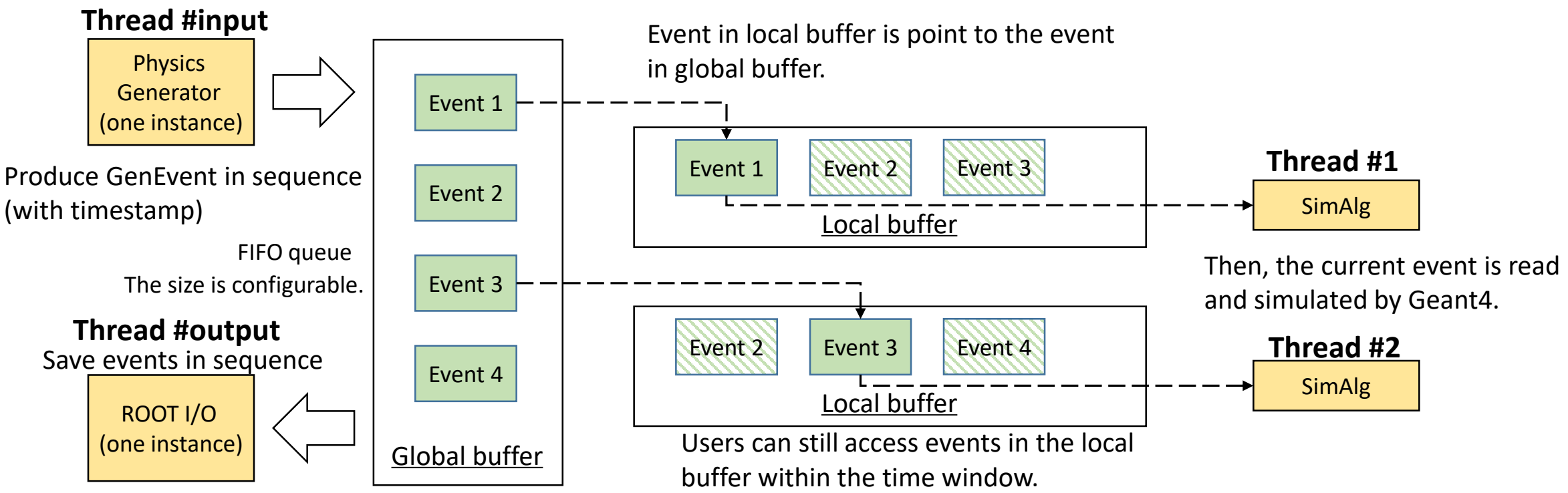
- To get rid of thread safe problem, ROOT 6 is used.
  - In ROOT 5, we need mutex to avoid crash.
  - In ROOT 6, just use ROOT::EnableThreadSafety();
- Each thread will output its own file.
  - Then, use TChain to read these files.
  - ROOT TBufferMerger (experimental) is considered in the future.
- Plain tree is used to save results.
  - Event ID, totalPE (scalar)
  - Hit time, nPE, pmtID (vector, $O(10^3)$)
- We can still get a close to linear speedup.



In the test, we have 3 cases: without i/o, i/o in /tmp directory (local disk), i/o in /junofs (lustre fs).
As we expected, I/O will affect the performance.

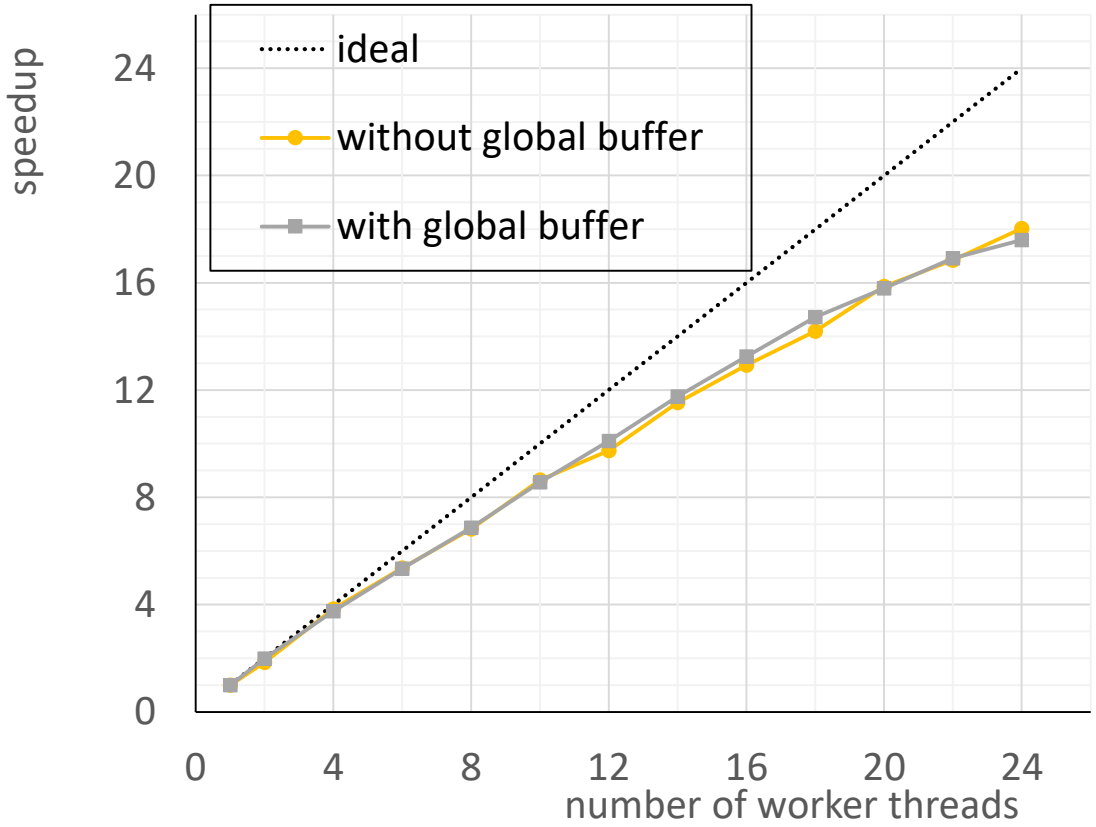
Simulation with global buffer and I/O

- To keep the order of events, SNiPER's global buffer is used in simulation.
  - Start two extra threads for input and output.



Performance (with global buffer+I/O)

- Little performance affect when using global buffer.



Note: $N_{total\ threads} = N_{worker\ threads} + 2$ i/o threads

$N_{worker\ threads}$ is the number of worker tasks.

Summaries and plans

- JUNO offline software is moving to multi-threading with the evolution of SNiPER.
  - Software stacks are updated to c++ 11, Geant4 10, ROOT 6 etc.
  - SNiPER Muster is implemented using Intel TBB library.
  - The data management can support event correlation.
- A parallelized detector simulation framework is developed for JUNO.
  - SNiPER Muster and Geant4 run manager kernel are integrated.
  - Simulation is optimized to achieve a close to linear speedup.
  - Global buffer is used in simulation to keep the correct order.
- Need more effort to release a production version.
  - Electronics simulation, global buffer and ROOTIO.
- SNiPER: https://github.com/SNiPER-Framework

Thank you!