# Concurrent generator support for CMS computing

**李聪乔 (Congqiao Li** ), *Peking University*

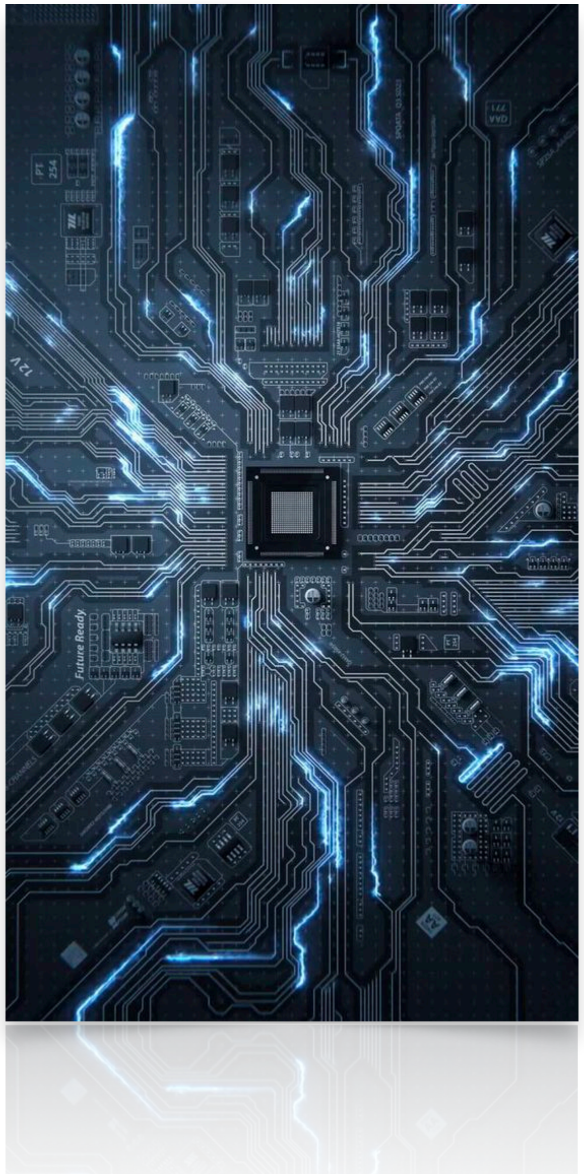
*on behalf of the CMS Collaboration*

**CLHCP 2021, Nanjing, China**

26 November, 2021

# Introduction

- ➔ ***Multithreading has become a tendency in modern computing***
  - ❖ can simplify the program structure and lessen the system resource usage
  - ❖ various threaded C++ libraries enable to streamline the high-throughput program
  - ❖ CMS has been the first LHC experiment to use a multithreading framework (`cmssw`) for event processing *J. Phys.: Conf. Ser. 898 (2017) 042008*
- ➔ ***Event generators*** *FERMILAB-PUB-21-526-QCIO-SCD-T*
  - ❖ Event generation is the earliest step (i.e., GEN step) in the Monte Carlo event processing chain
  - ❖ In GEN step, `cmssw` interfaces with ***external generator C++ libraries*** (Pythia, Herwig, Sherpa…); different physics processes may use different generators
- ➔ ***Multithreading × Event generators?***
  - ❖ Concurrent computing in generators is demanded given the recent multi-threading trend
  - ❖ unlike concurrent implementation of other CMS modules, *concurrent GEN methods may vary, depending on the specific generator type*
- ➔ In this talk…
  - ❖ 🏭 introduce the *recently deployed concurrent method* for different types of generators
    - ‣ focus on general framework supports, specific generator adaptions, etc.
  - ❖ 📈 show the *validation results and the computing efficiency improvements*



Generator

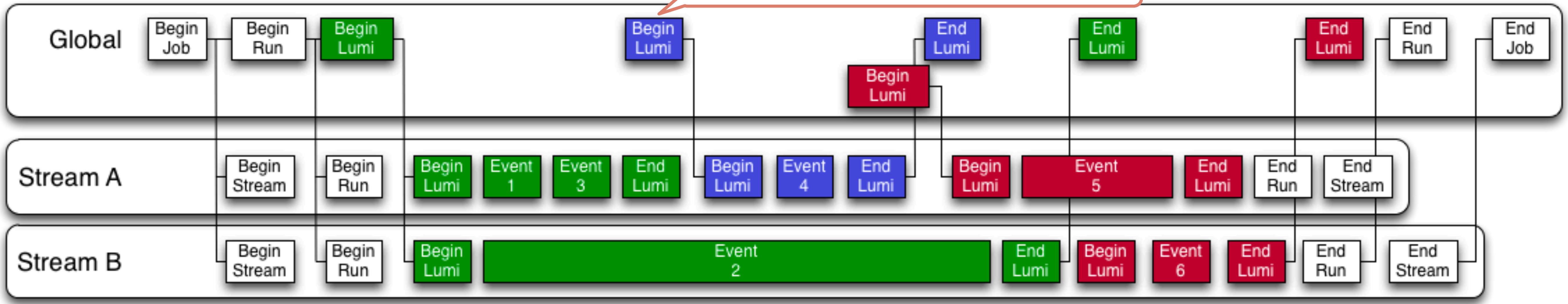
# Multi-threaded framework in CMS

➔ *The concurrent generator implementations are based on the existing multi-threaded `cmssw` framework*

➔ Multi-threaded framework in `cmssw`

- inherit the single-threaded event processing workflow:
  - Begin Run >> Begin Lumi >> event processing loop
- the Global thread controls multiple Stream threads and assigns new event processing job to the idle threads
- the Global and Stream transition regulates the multi-threaded behaviour

*a new lumi can start when another thread does not finish processing the previous lumi*



*events are assigned to idle threads*

*from CMSPublic twiki*

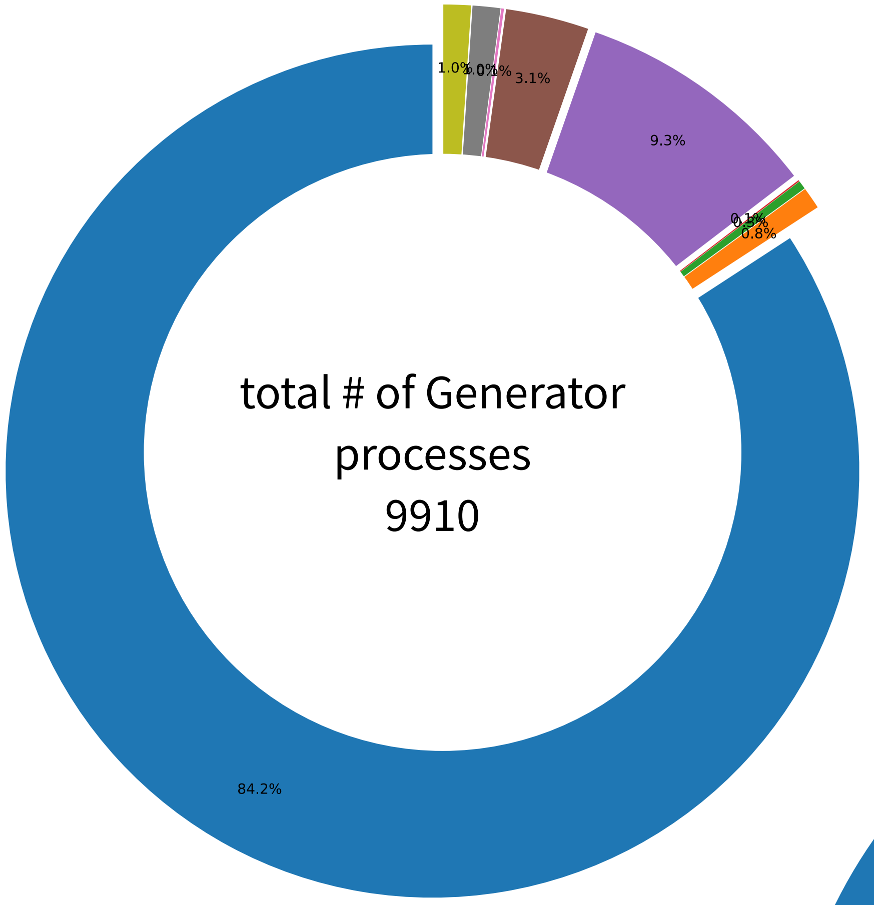
# Generator step in CMS

***Type I:***
*requires the external "gridpack" to first produce LHE events*

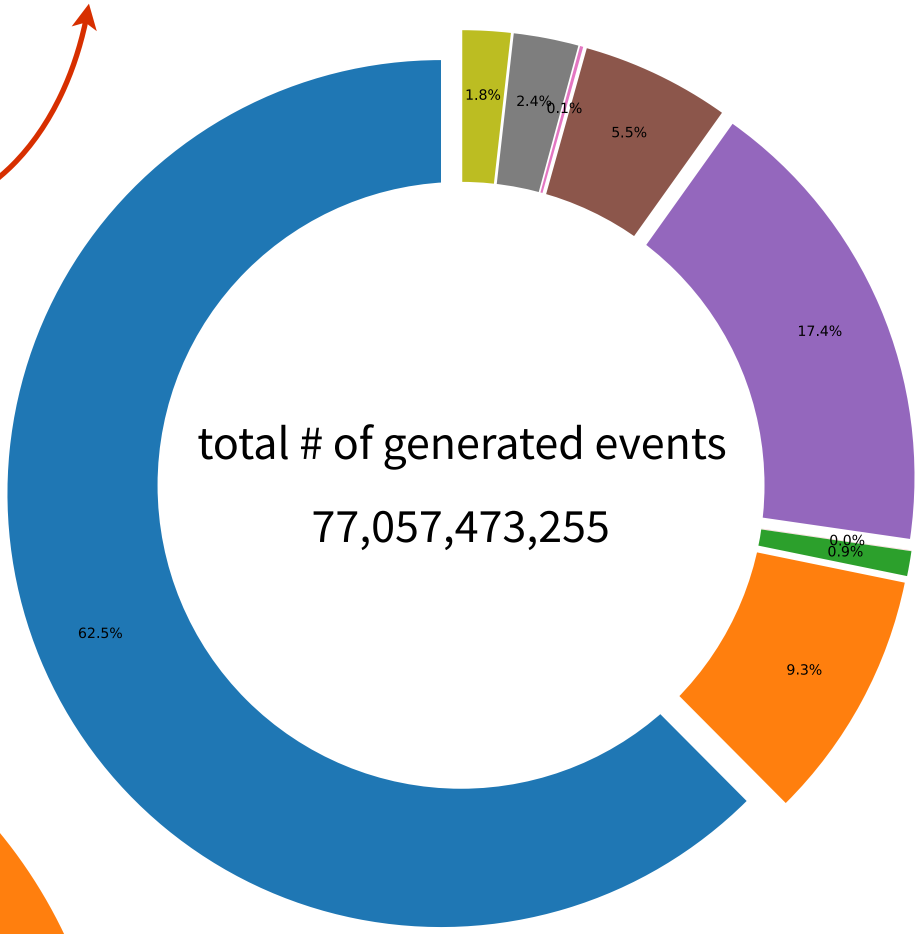
*external decayers: including EvtGen, Tauola, Photon*

- 🟦 LHE gridpack + Pythia8 (no external decayers)
- 🟧 LHE gridpack + Pythia8 (w/ external decayers)
- 🟩 LHE gridpack + Herwig7
- 🟥 LHE gridpack + others
- 🟪 Pythia8 (no external decayers)
- 🟫 Pythia8 (w/ external decayers)
- ▨ Herwig7
- ▨ Sherpa
- ▨ others

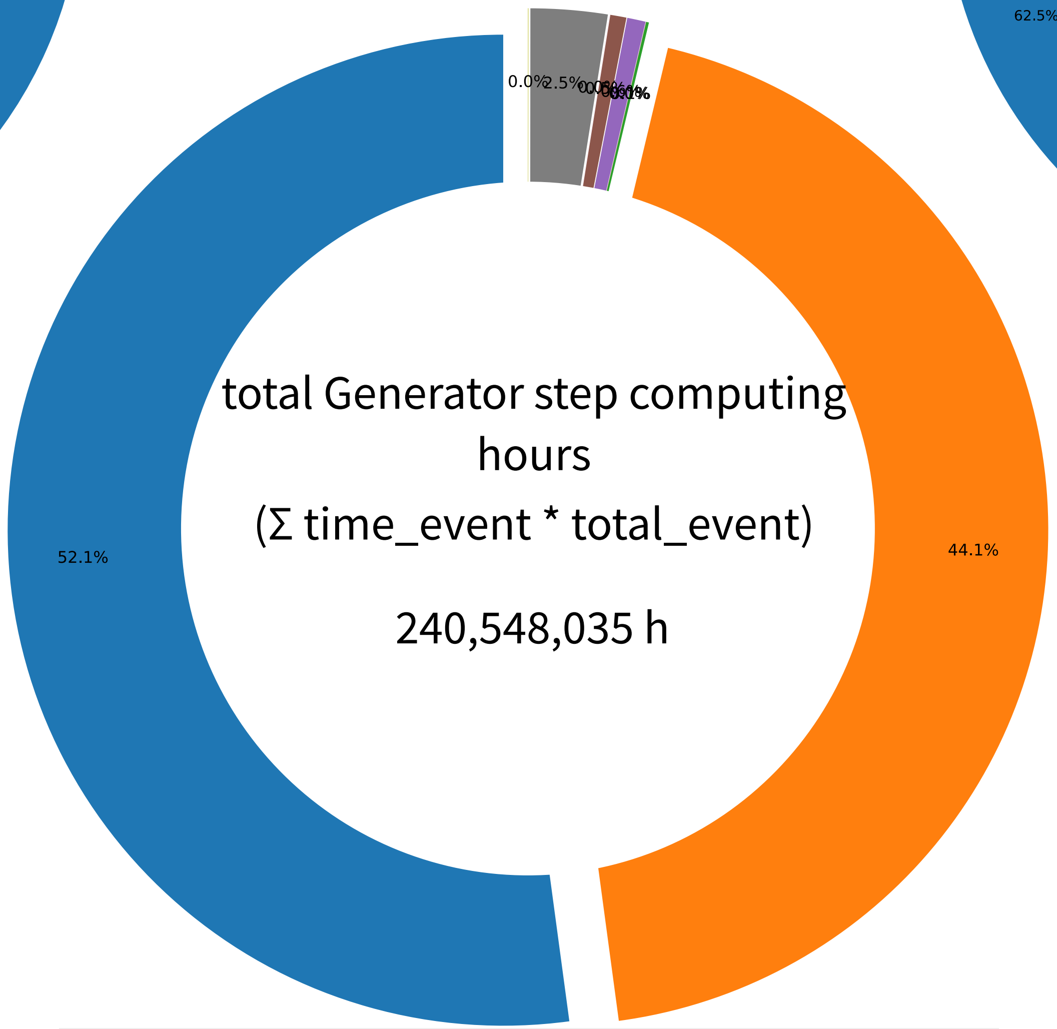
***Type II:** GEN only, no LHE*



total # of Generator processes
9910



total # of generated events
77,057,473,255

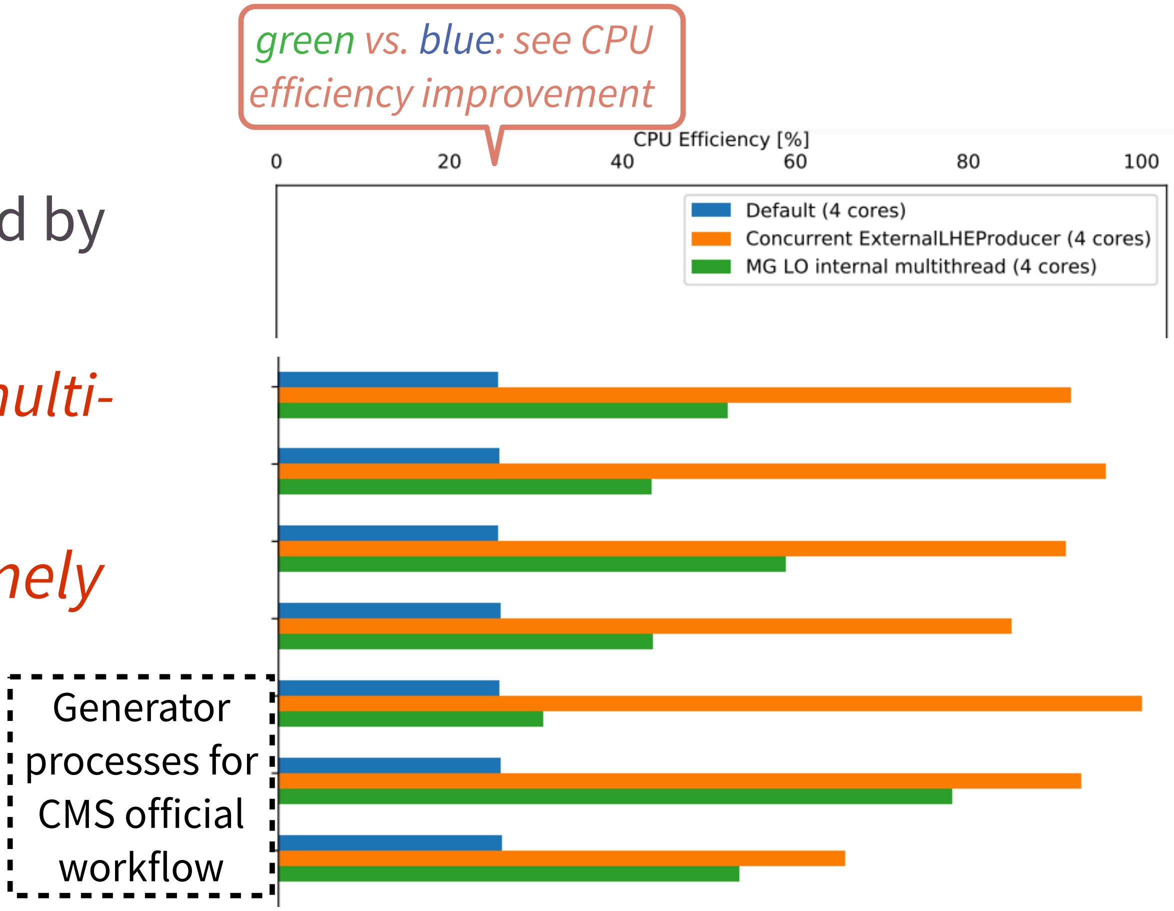


total Generator step computing hours
(Σ time_event * total_event)

240,548,035 h

# MadGraph gridpack in multi-threaded workflow

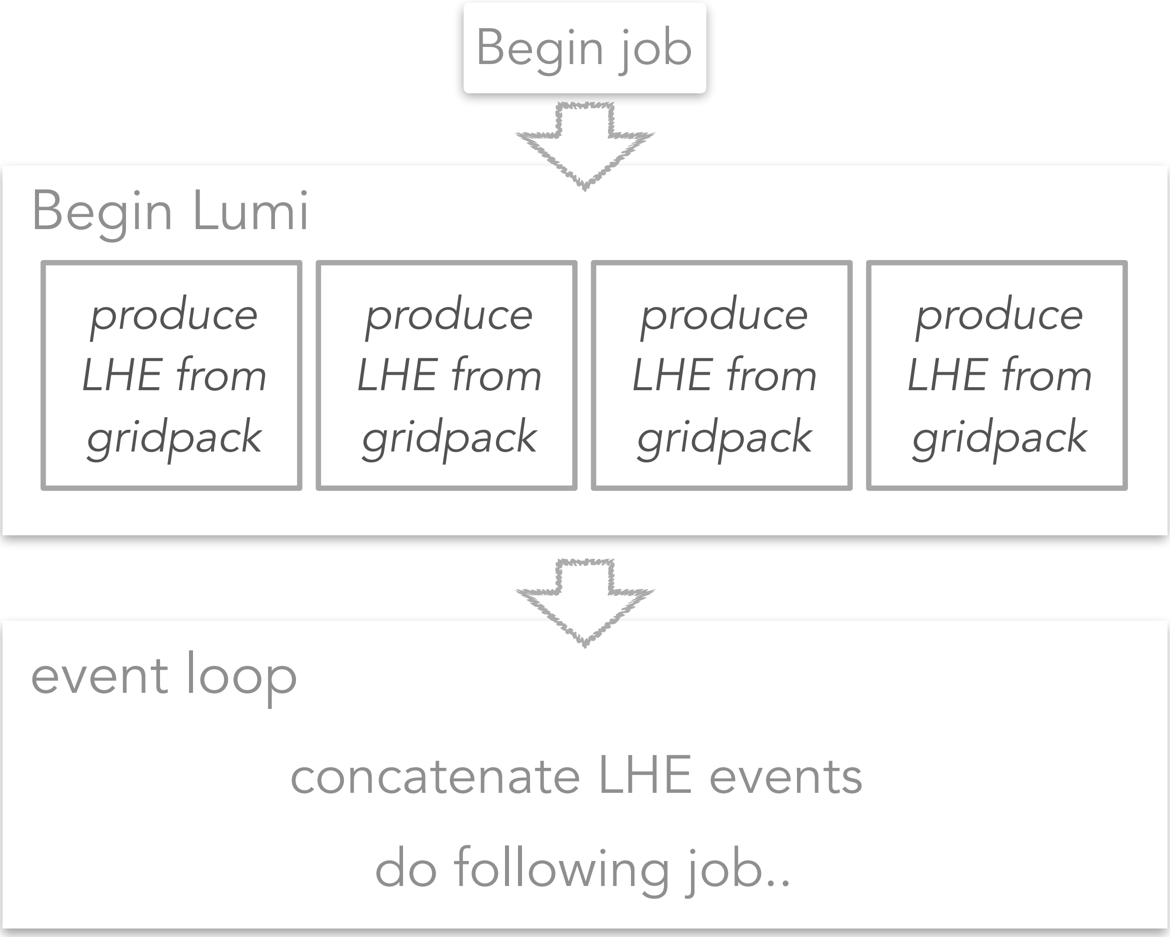
- ➔ CMS uses “gridpack” for LHE event production
  - ❖ gridpacks are sealed generators (MadGraph, Powheg, etc.) stored with phase-space information for a specific physics process → a black box to produce LHEs
  - ❖ use of gridpacks considerably save computing time due to recycling of integration results
- ➔ MadGraph5_aMC@NLO gridpacks produced in “`gridpack = True`” mode
  - ❖ the LO gridpack does not support multi-threaded by nature
  - ❖ we provide an *interface to MadGraph’s internal multi-thread development*
- ➔ A solution to save computing time *for the extremely complex process*, e.g. multi-jet MLM matching process


*green vs. blue: see CPU efficiency improvement*

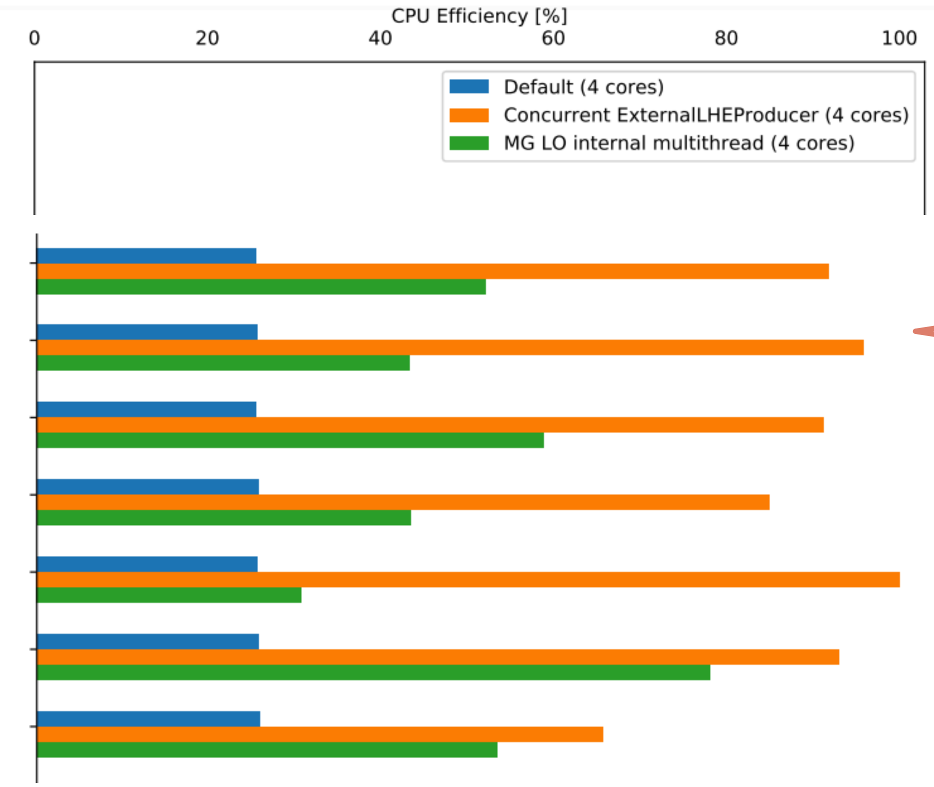
Generator processes for CMS official workflow

# Concurrency for general gridpack

- ➔ A new solution to enable concurrency for *all possible gridpacks*
  - ❖ execute the script for the LHE step in multiple instances concurrently using Intel’s TBB library
  - ❖ process the gridpack in *N* threads at the same time, and **run everything** concurrently


Begin job
Begin Lumi
produce LHE from gridpack
produce LHE from gridpack
produce LHE from gridpack
produce LHE from gridpack
event loop
concatenate LHE events
do following job..

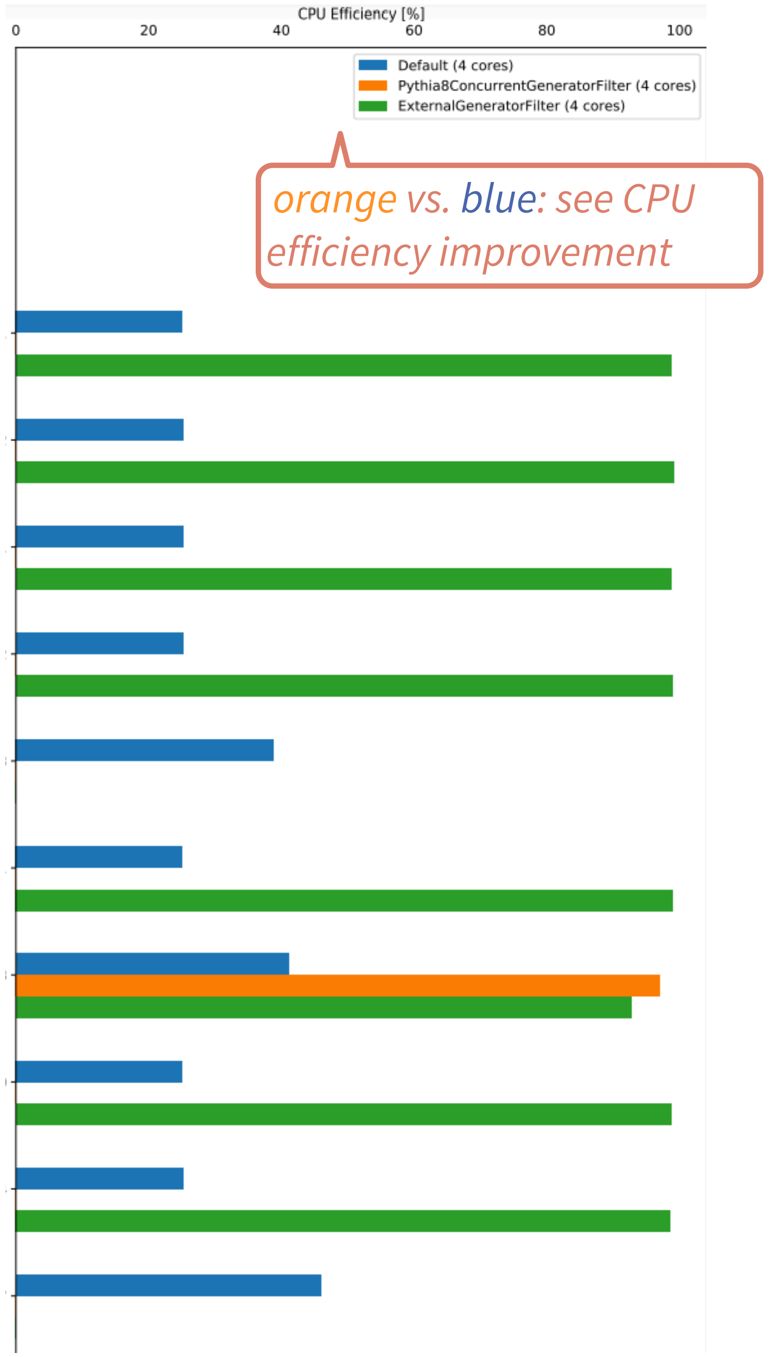
Generator processes for CMS official workflow



*orange vs. blue: see CPU efficiency improvement*

# Concurrent Pythia8 event processing

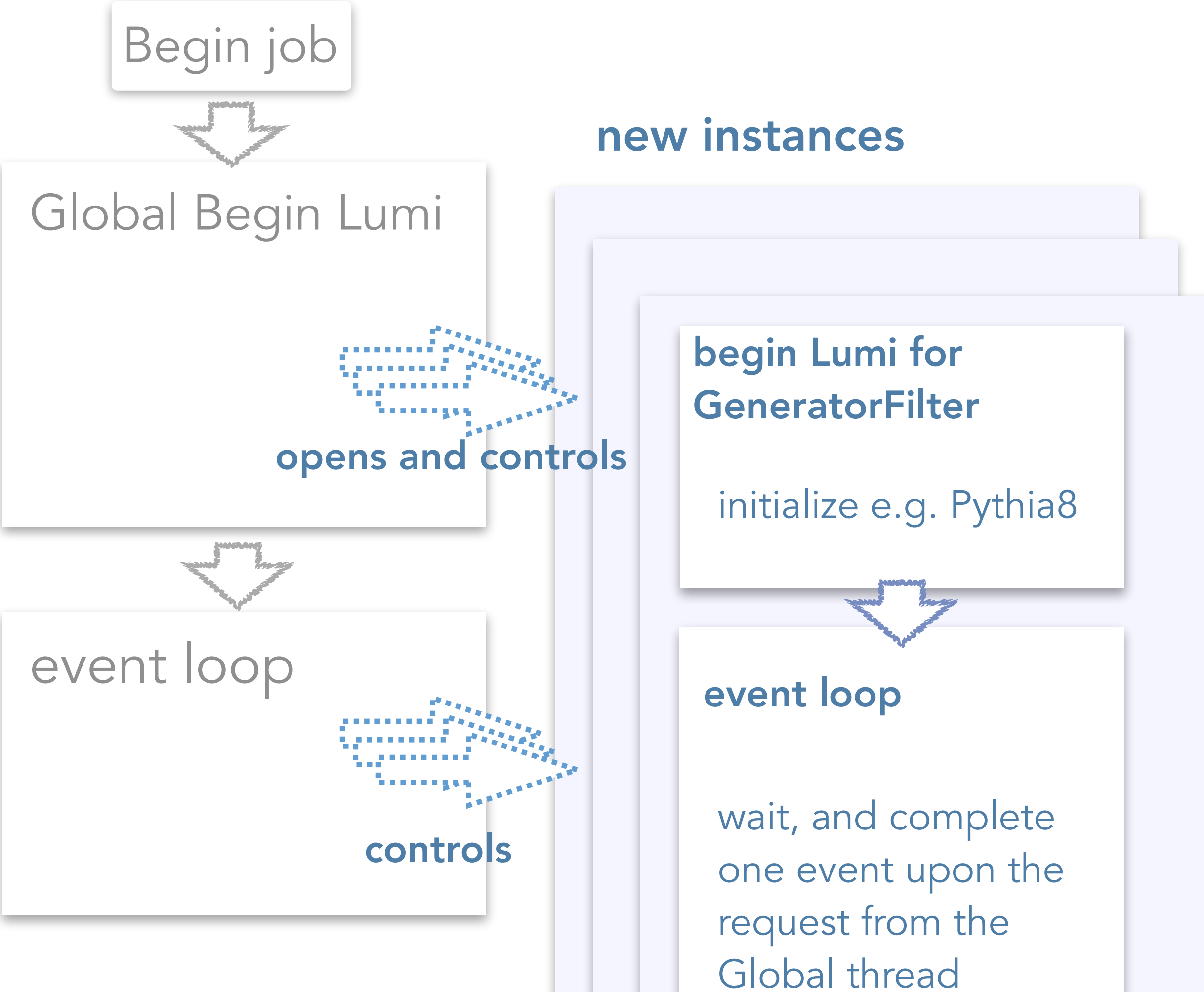
- ➔ Pythia8 is a widely-used MC generator for parton showering and hadronization
- ➔ Module for hadronization to produce HepMC-format output in CMS:
  - ❖ `GeneratorFilter` module: run a hadronized event from scratch with no LHE events as input
  - ❖ `HadronizerFilter` module: read an LHE event then hadronize it
- ➔ Pythia8 in cmssw has both modules
  - ❖ since Pythia8 may also interface to non-thread-safe generators e.g. Tauola, EvtGen; the general Pythia8 `GeneratorFilter/HadronizerFilter` module is not multi-threaded supported
  - ❖ standalone Pythia8 supports multi-threaded instance
  - ❖ new modules designed to only run Pythia8 without interfacing to non-thread-safe modules
    - ▸ named `ConcurrentGeneratorFilter/ConcurrentHadronizerFilter`

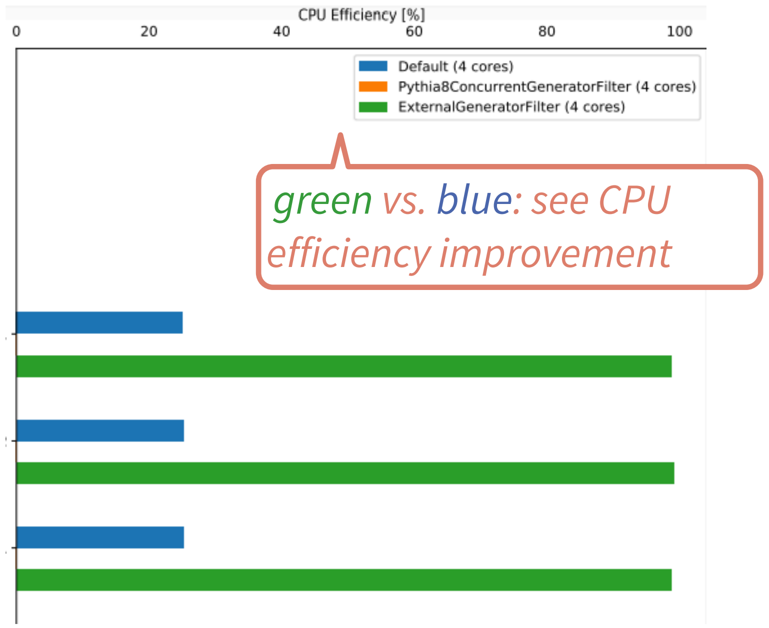


*orange vs. blue: see CPU efficiency improvement*

# External solution to a hadronizer module

- ➔ The module `ExternalGeneratorFilter` is developed to extend the concurrency ability to the classic `GeneratorFilter`
  - ❖ “Once and for all solution” — enable inter-process communication, further beyond a multi-threaded workflow
    - ‣ the global stream ***opens multiple new instances*** and runs a GeneratorFilter on each instance
  - ❖ no thread-safety issue because different instances do not share the memory with each other



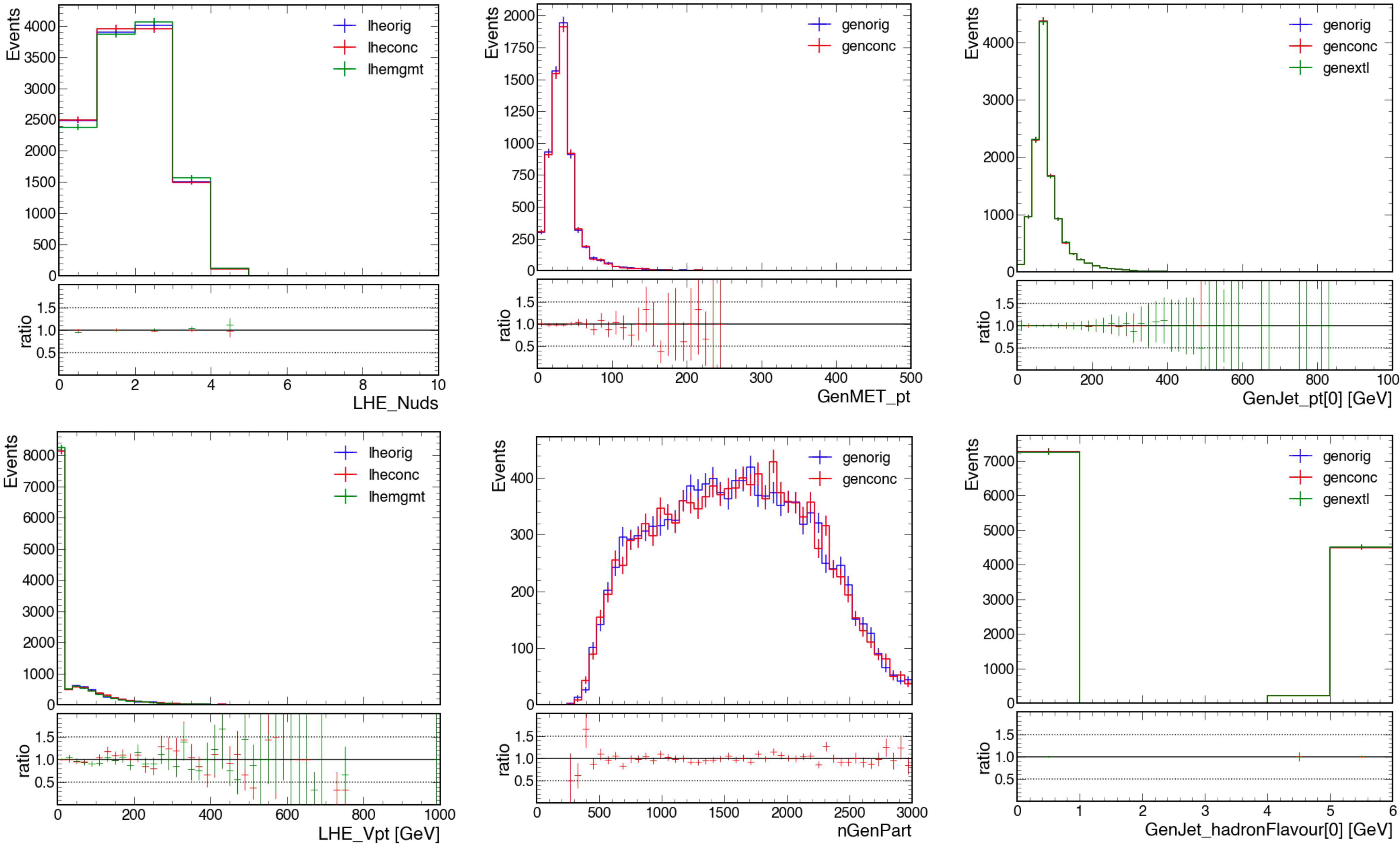


*green vs. blue: see CPU efficiency improvement*

# Physics validation

➔ Validate if the single-threaded and multi-threaded modes give the same physics result

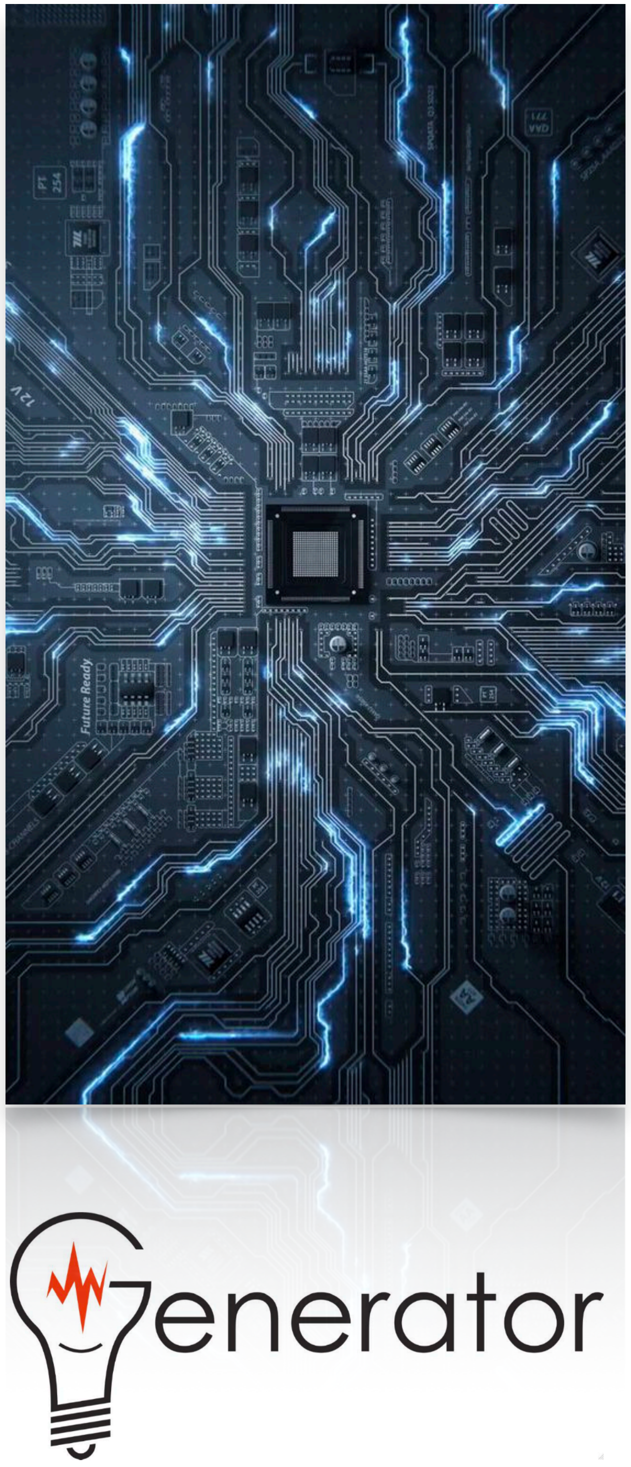
❖ examine the generator-level physics kinematics



see good agreement in general

# Conclusion

- ➔ In CMS, various generator multi-threading utilities developed thanks to the work of many contributors
  - ❖ setup CMSPublic twiki WorkBookGenMultithread
  - ❖ present a thorough view of these utilities from the perspective of the implementation & the generator types
  - ❖ provide direct recipes for CMS users (and Generator contacts) to enable these features
- ➔ All utilities are validated well in physics and show good performance in overall
- ➔ Foresee a better computing environment in the CMS generator step



# Backup