FIRST BENCHMARK WITH MADGRAPH4GPU FROM CMS

HSF SEMINAR

NOVEMBER 27, 2024

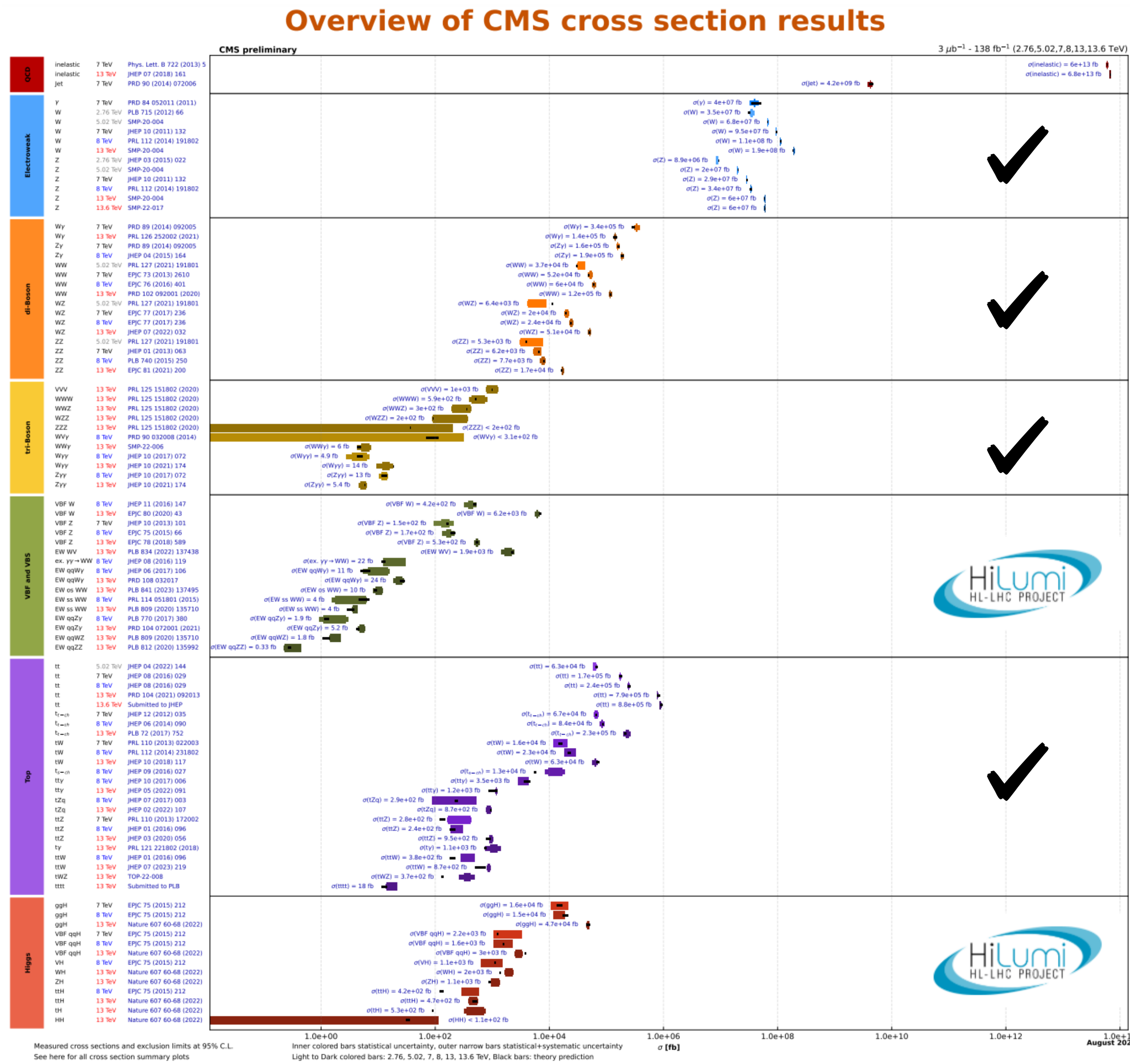
Jin Choi, On behalf of the CMS Collaboration

and special thanks to
Sapta Bhattacharya,Olivier Mattelaer, Andrea Valassi, Stephan Hageboeck,
Stefan Roiser, Robert Schoefbeck, Zenny Wettersten, Daniele Massaro

INTRODUCTION

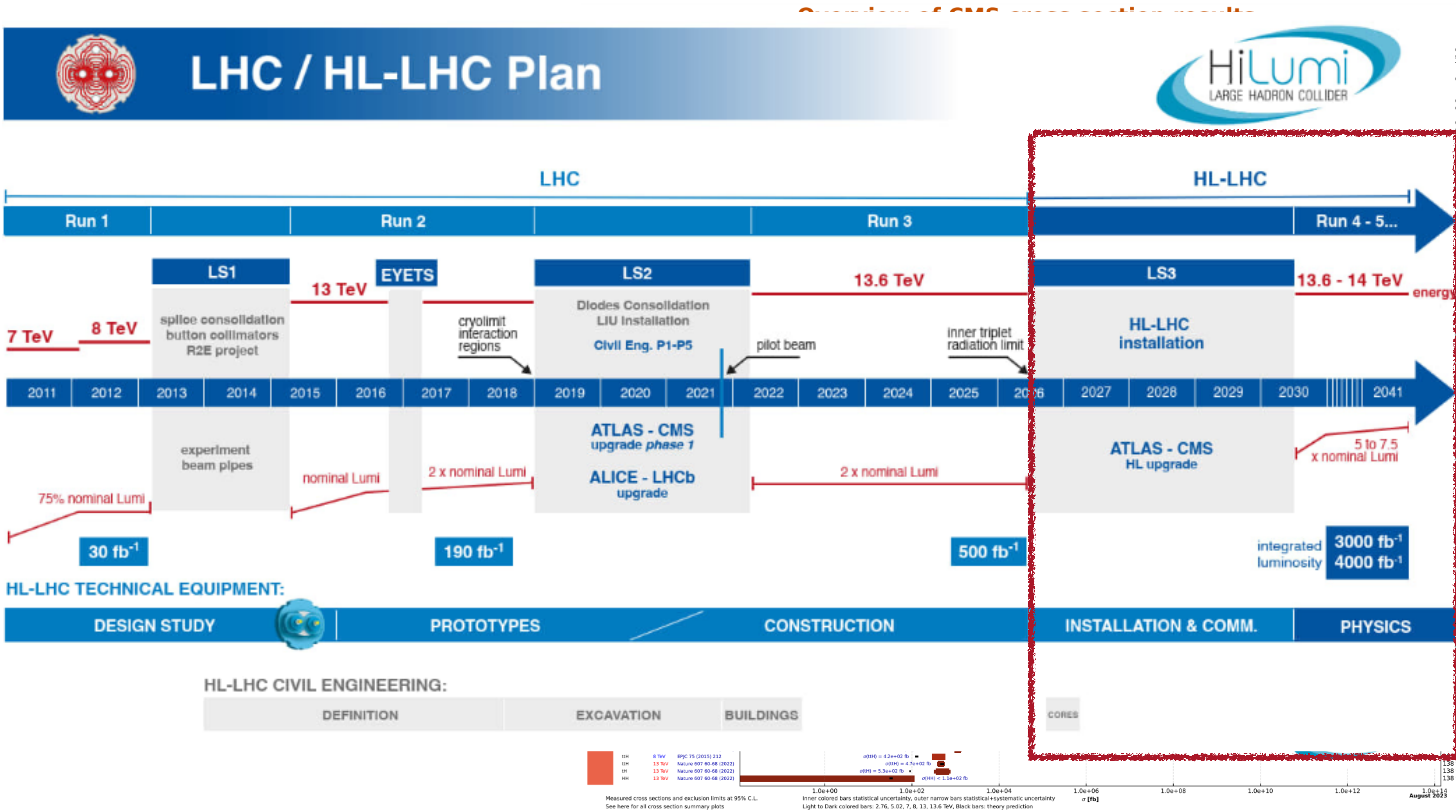
TOWARDS HL-LHC

- **The Rise of the Precision Era**
- Successful measurement of SM properties in Run2 & Run3
- Looking for what we are missing with extra precision
- Measurement of very small cross-section SM processes
- Triple Higgs Coupling?
- O(1) fb- O(100) ab BSM Signals?



[CMS cross section summary]

TOWARDS HL-LHC

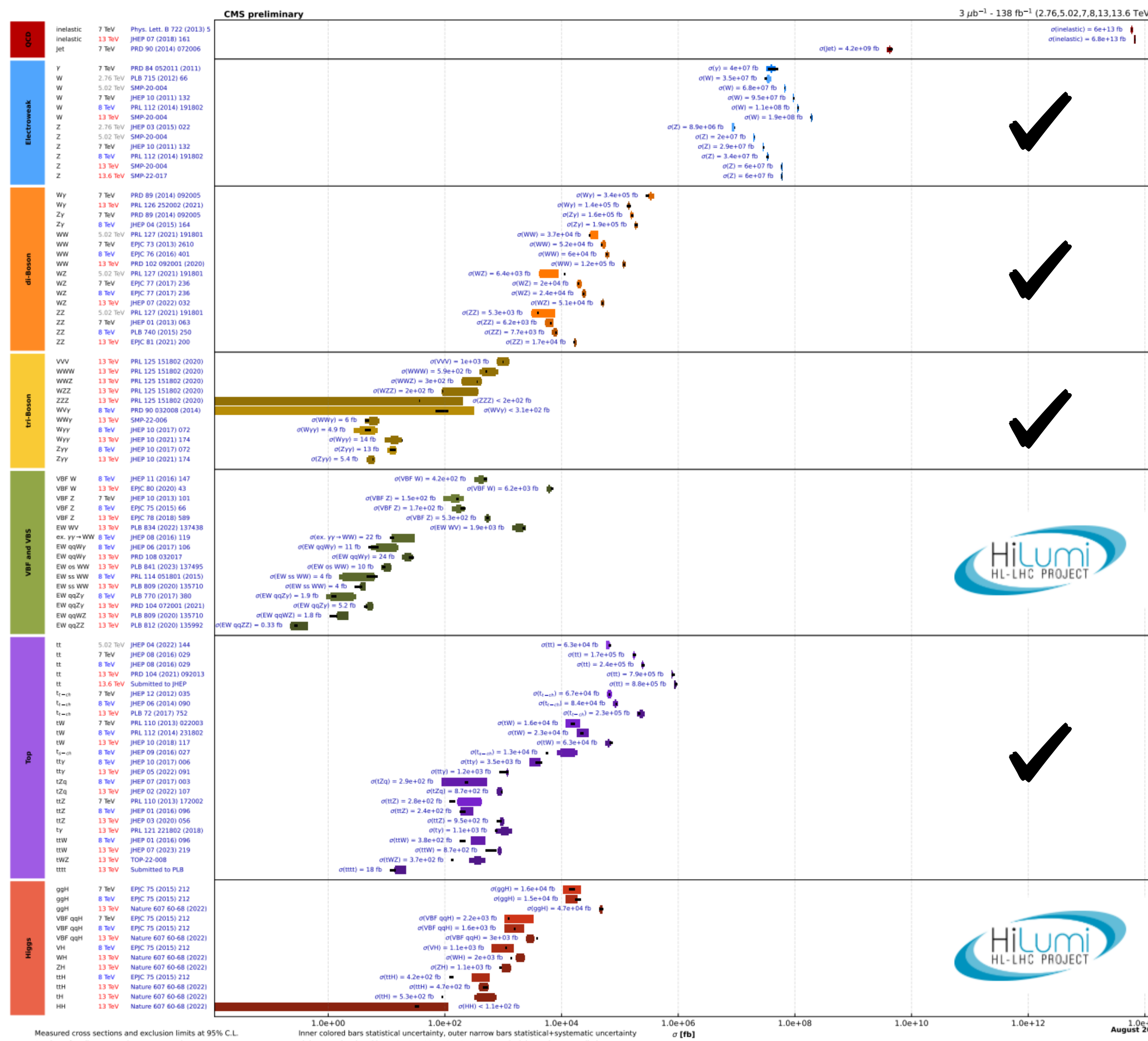


[CMS cross section summary]

TOWARDS HL-LHC

- **The Rise of the Precision Era**
  - Successful measurement of SM properties in Run2 & Run3
  - Looking for what we are missing with extra precision
  - Measurement of very small cross-section SM processes
  - Triple Higgs Coupling?
  - O(1) fb- O(100) ab BSM Signals?
- **HL-LHC**
  - Will collect ~4000 fb^{-1} data
  - with x3-4 increased ins. lumi.
  - Require O(10)-O(100) billion events to be simulated!

Overview of CMS cross section results

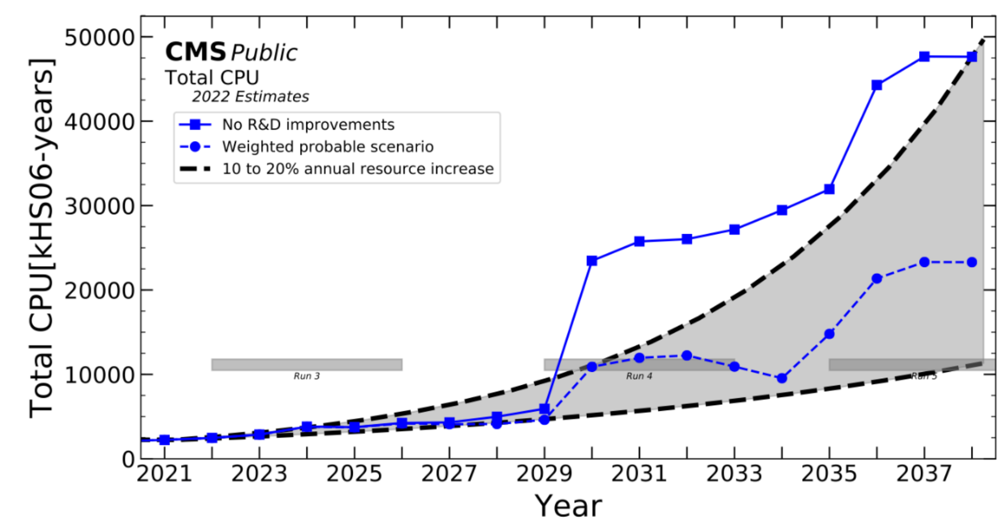


[CMS cross section summary]

COMPUTING ENHANCEMENT

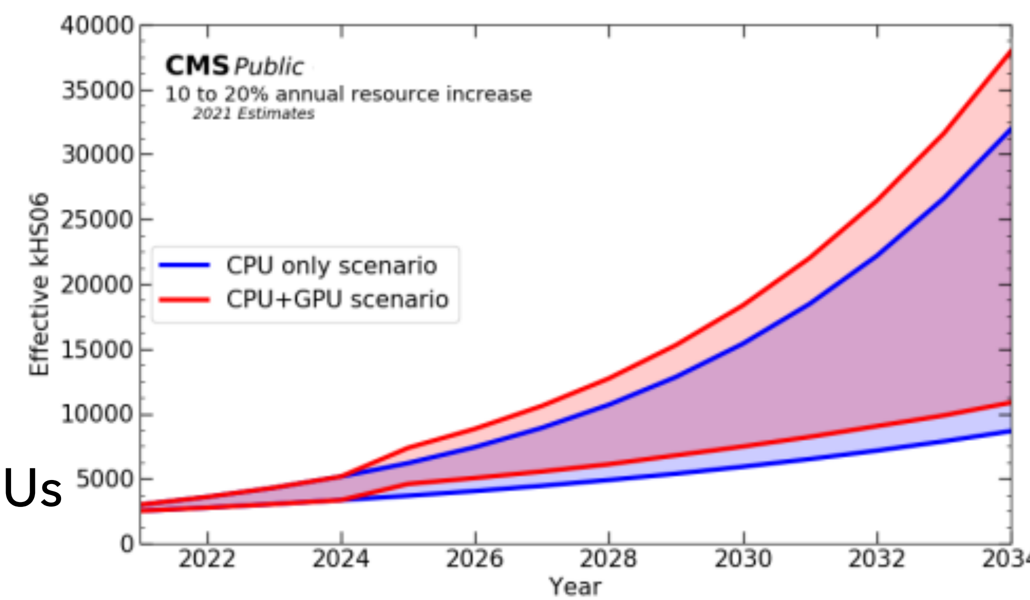
Projected Analysis on Computing Resources

- 10-20% more computing resources will be purchased every year
- Would not match the need with HW only upgrade
- SW acceleration is essential part for the computing in the next phase!



Technology Develops

- Most of the SW for RUN2 and RUN3 are based on SISD - often single core execution, or thread-level parallelism
- Recent advancements in SIMD/SIMT with vectorized CPUs and GPUs
- For vectorized CPUs, approx. 80% of WLCG supports AVX2 vectorization scheme
- Additional powerful source for accelerating SW with GPUs

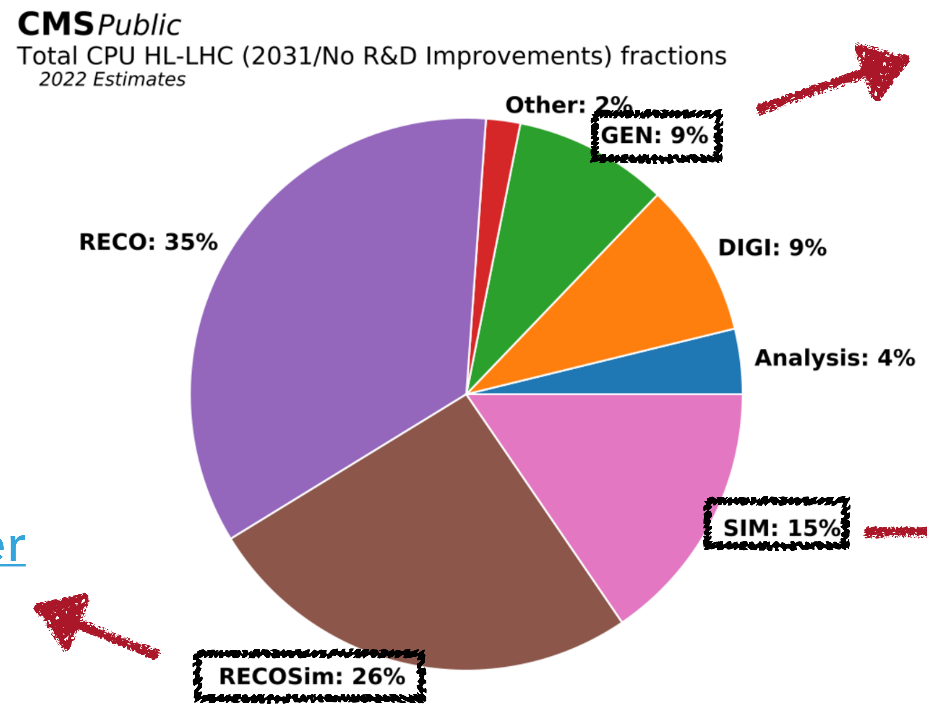


[CMS Phase-2 Computing Model]

ONGOING UPDATES

CMS Monte Carlo Sample Production

- Approx. 50% of computing resources are allocated to MC sample production



[Madgraph4GPU]

[Celeritas]
[ML-based FastSim]

Vectorized KF filter
GPU ported KF filter
ML-based PF

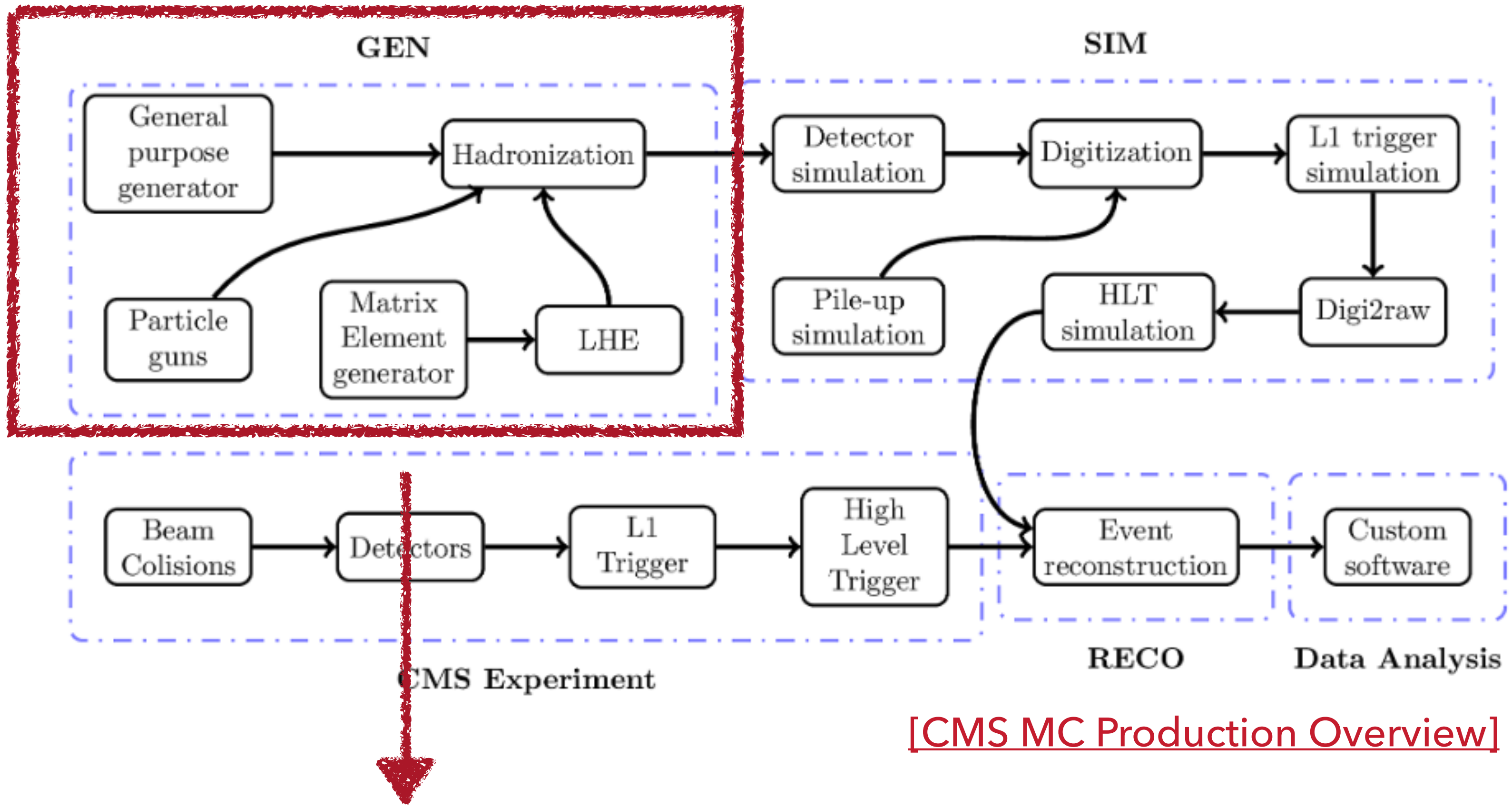
[CMS Phase-2 Computing Model]

End-to-end ML based FastSim - FlashSim

- Replacing FullSim with FastSim or PF reco algorithms often developed within CMS, while updates on Generators / Geant4 should follow the updates from the authors
- Often Generator is the starting point of the sample production - no alternative
- Madgraph4GPU is the first program that can be tested within CMS workflow!

WHY MADGRAPH

- **Generator usage in Run2**
- Madgraph is the generator for MASS PRODUCTION: ~1/3 event simulation done in Run2
- Madgraph is the generator for FIRST SEARCH: ~2/3 samples requested in Run2
- **Madgraph4GPU in CMS workflow**
- Most common backgrounds are generated with Madgraph / aMC@NLO, approx. O(100)M events in Run2 / Run3
- Right time to test it's speed-ups for preparing mass production



[CMS MC Production Overview]

- In CMS, we employ **Gridpacks** to optimize resource utilization and minimize redundant computations
  - package that encapsulates pre-computed matrix elements and all other requisite components.
- Gridpacks are distributed to various computing centers for the generation of LHE events.
- Gridpack production timing is often shorter than event generation
  - but heavy gridpacks with multiple jets can take **weeks** to produce.

e.g. LO DY+4j with MLM
NLO DY+2j with FxFx

EXPERIMENT SETUP

EXPERIMENT SET-UP

- **Physics Process**
  - Testing most common backgrounds in CMS - Drell-Yan(DY) and top quark pair production(TT)
  - DY - No external dependencies, allowing re-use of CMS central cards for production
  - TT - Most speed-up observed by MG team, re-check in CMS sample production setup

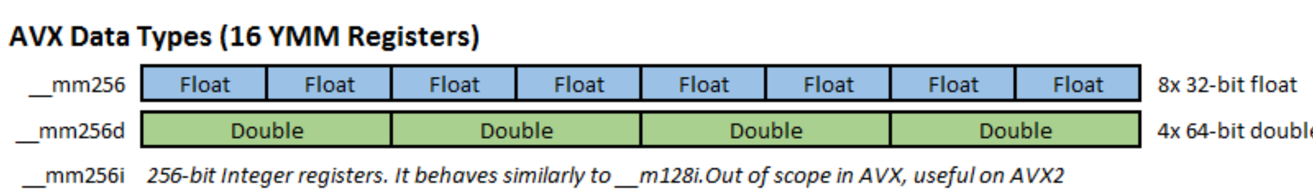
No Madspin support for TT

- **Experiments**
  - Following on the CMS GEN production steps, experiments are divided into **gridpack production** and **event generation**
  - Compared timings by switching backends - FORTRAN, CPP(vectorized C++, AVX2), and CUDA
  - Jet binned study - Allow checking of speed-ups and overheads w.r.t. increasing complexity
  - Inclusive study - Focused on physics of interest

TESTING ENVIRONMENTS

- **HPC Configurations**
- **Lxplus HTCondor Pool**
  - AMD EPYC 7313 16-Core Processor + 1 A100 GPU (40GB memory)
  - Intel(R) Xeon(R) Platinum 8468 (48 CPUs) + 1 H100 GPU (96GB memory)
  - CPU only nodes with Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz
- **SNU HTCondor Pool**
  - Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz (80 CPUs)

- **Software**
- Used **HTcondor** to make test environment isolated:
  CPUs and GPUs are exclusive but memory and I/O bandwidth might be shared
- Used **el8-based singularity image** to synchronize other dependencies
- MadGraph v5.3.6.0 + madgraph4gpu dev. repo
- Integrated to the centralized CMS event generator code with additional modifications.
- Accelerated with:
  - CPUs: AVX2 vectorization scheme [CPP]
  - GPUs: Using nvidia CUDA [CUDA]



Theoretically, x4 speed-up can be achieved for AVX2

INTEGRATION

Workflow

proc_card.dat

```
import model sm-ckm_no_b_mass

generate p p > t t~ @0
add process p p > t t~ j @1
add process p p > t t~ j j @2
add process p p > t t~ j j j @3

output madevent_gpu TT0123j_CKM_LO_5f_FORTRAN -nojpeg
```

backend call for madgraph4gpu plugin

customizecards.dat

```
#put card customizations here (change top and higgs mass for example)
set param_card mass 6 172.5
set param_card mass 25 125.0
set param_card yukawa 6 172.5
set sde_strategy 1
set cudacpp_backend CUDA
```

Backend can be switched by cudacpp_backend parameter in run_card.dat

- Development of generators are managed by authors,
  can be smoothly integrated into the CMS gridpack production workflow
- Used development version of the MG4GPU in the time of testing - last synced on August 19th, 2024.

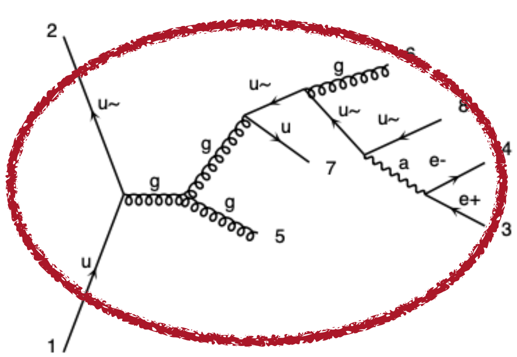
Find the first version of the project!
[cudacpp_for3.6.0_v1.00.00]

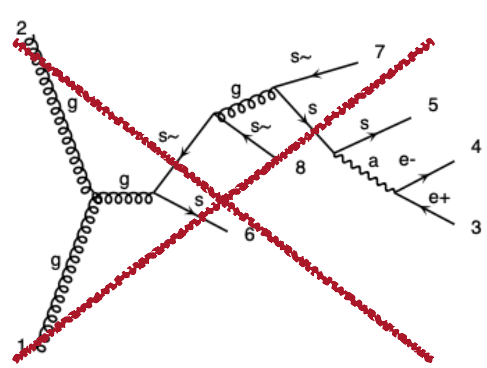
GRIDPACK PRODUCTION

SIMPLIFIED PARTON DEFINITION

Impact of different plugins for many-diagram processes - Parton Grouping

- Madevent is the basis of ME calculation
  - subprocess level parallelism can be achieved by submitting multiple madevent plugins
- To reduce the no. of subprocesses, original MG use parton grouping after generating the diagrams
- Not implemented in madevent_gpu yet:
  madevent - processes with grouped partons are calculated within the same subprocess.
  madevent_gpu - processes with different partons are treated separately
- To compare within the same subprocess level, we **simplified the parton definition** for DY+4j and DY+01234j





Simplified: p = j = u u~ g, ell+ = e+, ell- = e-

```
import model sm-no_b_mass

define p = u u~ g
define j = p
define ell+ = e+
define ell- = e-
define nu = ve vm vt
define nubar = ve~ vm~ vt~

generate p p > ell+ ell- @0
add process p p > ell+ ell- j @1
add process p p > ell+ ell- j j @2
add process p p > ell+ ell- j j j @3
add process p p > ell+ ell- j j j j @4

output madevent_gpu DY01234j_LO_5f_Simplified_CPP -nojpeg
```

- Have large impact in gridpack production, but should not affect in event generation.

Parton Grouping will be implemented in the next version of cudacpp!

JET-BINNED STUDY

Complexity overview

	DY+0j	DY+1j	DY+2j	DY+3j	DY+4j	DY+4j (Simplified)
diagrams	30	180	3120	27600	412560	9856
processes	15	45	285	435	1455	13

Low Complexity ← → High Complexity

	TT+0j	TT+1j	TT+2j	TT+3j
diagrams	8	91	1473	17660
processes	6	16	96	146

With higher multiplicities, the system must handle a greater no. of processes
- other const. overheads can reduce speed-up (e.g. compilation, combining results, etc.)
- this mostly impacts Drell-Yan

JET-BINNED STUDY

DY

Timing normalized to a configuration using 16 madevents in parallel

production time	FORTRAN	CPP - AVX2	CUDA
DY+0j	7m	6m	5m
DY+1j	10m	10m	12m
DY+2j	1h 12m	1h 10m	51m
DY+3j	**22h 40m**	**9h 4m**	**4h 18m**
DY+4j (Simplified)	**440h 46m**	**141h 20m**	**9h 17m**

x3 (FORTRAN → CPP - AVX2), x50 (FORTRAN → CUDA)

TT

production time	FORTRAN	CPP - AVX2	CUDA
TT+0j	6m	7m	5m
TT+1j	11m	11m	7m
TT+2j	1h 15m	38m	22m
TT+3j	**262h 11m**	**79h 19m**	**3h 4m**

x3.5 (FORTRAN → CPP - AVX2), x90 (FORTRAN → CUDA)

- ✓ As the calculation of ME becomes complex in high-multiplicity processes, substantial gains can be realized.
- ✓ CMS utilizes up to 4j (3j) in DY (TT) LO production, an improvement in inclusive samples would be greatly appreciated.

INCLUSIVE STUDY

Production Time

Timing normalized to a configuration using 16 madevents in parallel

	FORTRAN	CPP - AVX2	CUDA (A100)
DY+01234j (Simplified)	424h 36m	133h 38m	9h 32m
TT+0123j	253h 36m	155h 28m	3h 9m

DY+01234j (Simplified): x3 (FORTRAN → CPP - AVX2), x45 (FORTRAN → CUDA)

TT+0123j: x3 (FORTRAN → CPP - AVX2), x85 (FORTRAN → CUDA)

For full DY+01234j, ~2 days is sufficient to produce gridpacks with CUDA

- HPCs with CPU-only setup are often configured with large memory capacities(O(100) GB), allowing submissions with more nb_cores than in CPU+GPU architectures
- In TT, the dominant process is gg_ttxggg, which requires ~ 6GB of memory. Speed-ups are on the order of O(100), resulting in an overall speed-up of ~O(10) for CUDA due to memory restrictions in GPUs

EVENT GENERATION

EVENT GENERATION

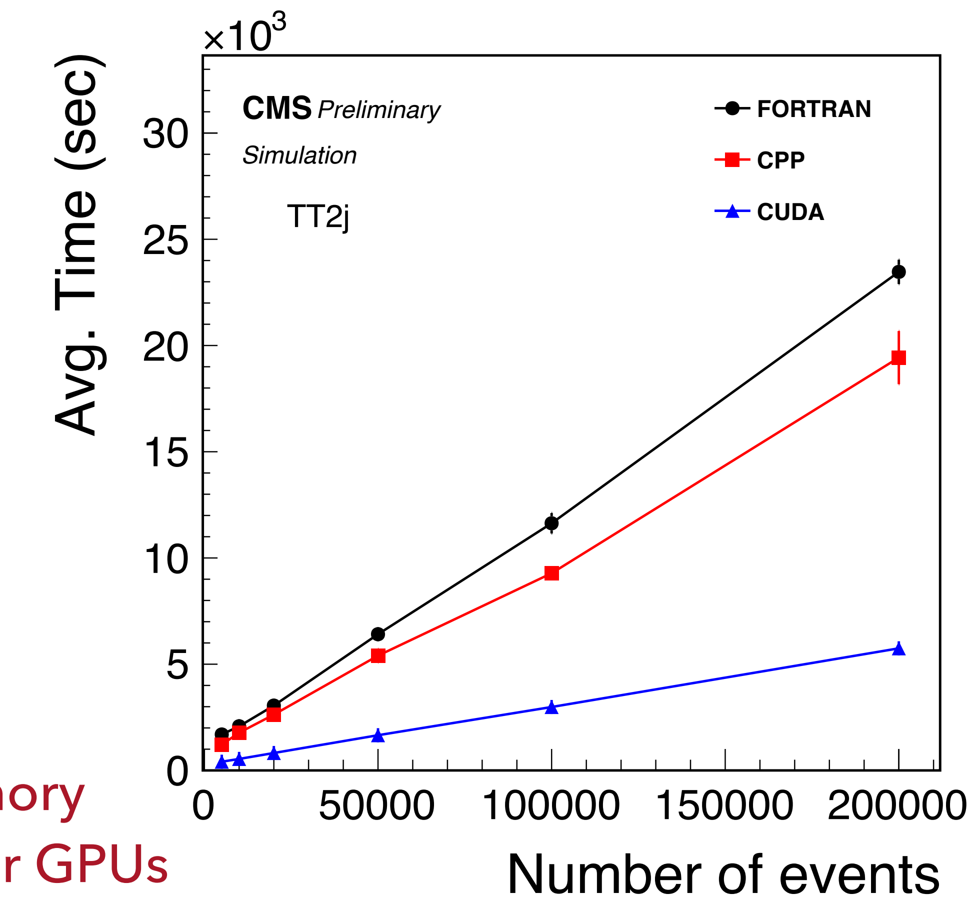
Method

- Each process was tested with 5k, 10k, 20k, 50k, 100k, and 200k event generation using a single core.
- For FORTRAN/CPP: CPU-only nodes in lxplus with Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz
  For CUDA: AMD EPYC 7313 16-Core Processor + 1 A100 GPU (40GB memory)
- Ran each test 8 times in parallel and computed the mean and standard deviation.

TT2j - 20000

	FORTRAN	CPP	CUDA
0	28m9.804s	20m38.510s	6m47.389s
1	29m8.745s	20m2.217s	7m45.196s
2	28m45.357s	19m54.762s	6m53.662s
3	27m32.752s	19m52.637s	7m50.752s
4	28m22.442s	21m0.456s	6m18.169s
5	27m29.009s	19m47.381s	6m43.447s
6	27m21.541s	20m11.514s	6m0.925s
7	28m47.916s	20m31.408s	6m19.430s
avg	28m 12s	20m 14s	6m 50s
err	41s	26s	20s

These numbers are used for the final results

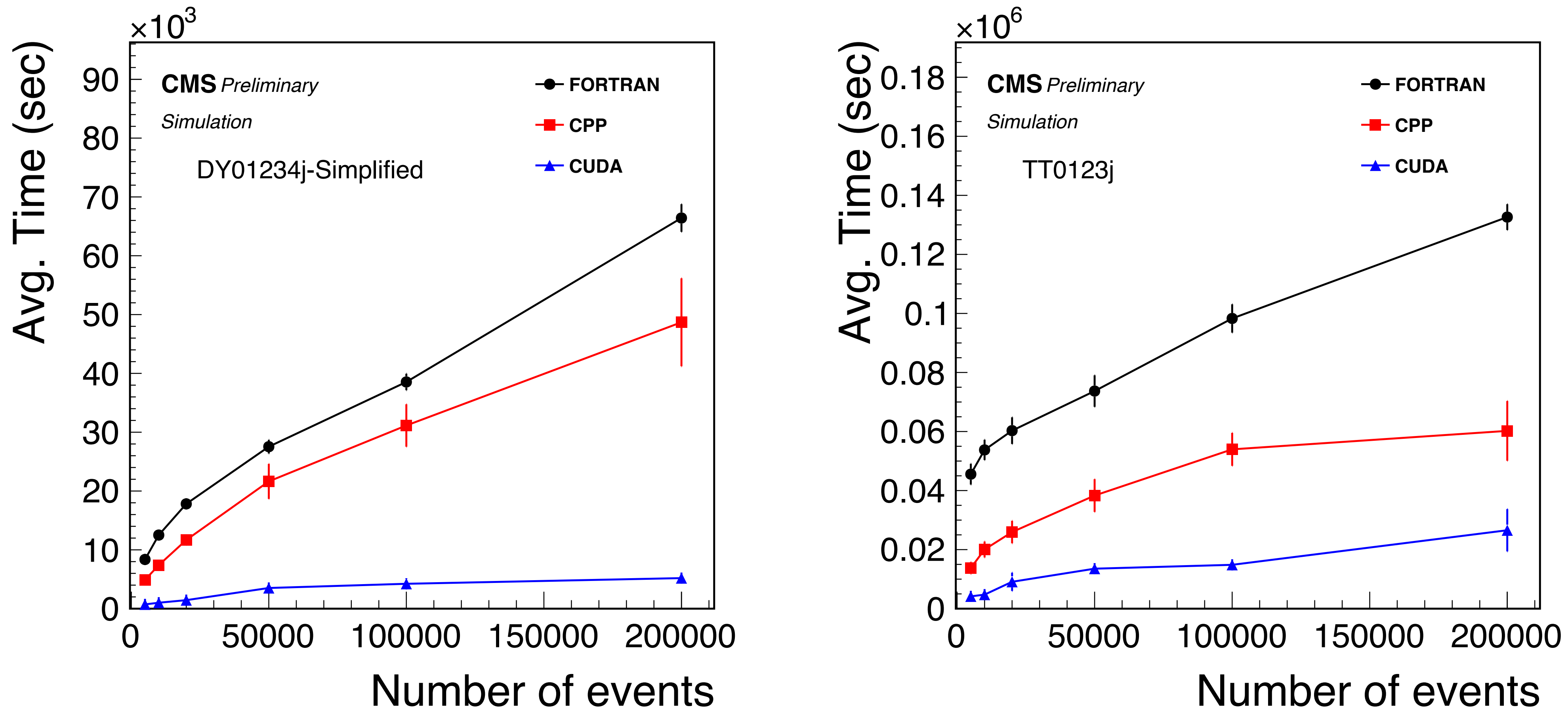


All the numbers used in the plots stored in [here]

※ Again, overhead might be caused due to sharing memory and I/O within the same HT condor node, especially for GPUs

EVENT GENERATION

Inclusive results



- Overall improvement can be achieved for both DY and TT, approximately1.5x(7x) speed-up observed for CPP-AVX2(CUDA) vs. FORTRAN

SUMMARY

The first benchmarks of MadGraph4GPU integrated into CMS workflows

- Precision physics at the HL-LHC will necessitate substantial computational upgrades, with simulation demands requiring **O(10)–O(100)** billion events
- MadGraph is central to CMS MC sample production, generating **~1/3 of events** and **~2/3 of requested samples** during **Run 2**
- Benchmarks reveal significant speed-ups in gridpack production and event generation for common processes like **Drell-Yan (DY)** and **top-quark pair production (TT)**
  - Gridpack production achieved approximately x**3 (x90)** speed-ups for **CPP-AVX2(CUDA)**.
  - **Event generation** observed x**1.5 (x7)** improvements for **CPP-AVX2(CUDA)**, for generating 100k events.

REFERENCE

[1] CMS Collaboration, *Quantifying the computational speedup with madgraph4gpu for CMS workflow*. URL: https://cds.cern.ch/record/2914584/files/DP2024_086.pdf

[2] A. Valassi, T. Childers, L. Field, et al. *Development in Performance and Portability for Madgraph5_aMC@NLO*. PoS, 41st International Conference on High Energy Physics (ICHEP2022), vol. 414, 2022. DOI: 10.22323/1.414.0212.

[3] S. Hageboeck, et al. *Madgraph5_aMC@NLO on GPUs and vector CPUs: experience with the first alpha release,* CHEP 2023 Conference, May 2023. URL: https://indico.jlab.org/event/459/contributions/11829/attachments/9445/13694/23.05.-Madgraph-CHEP-SH.pdf

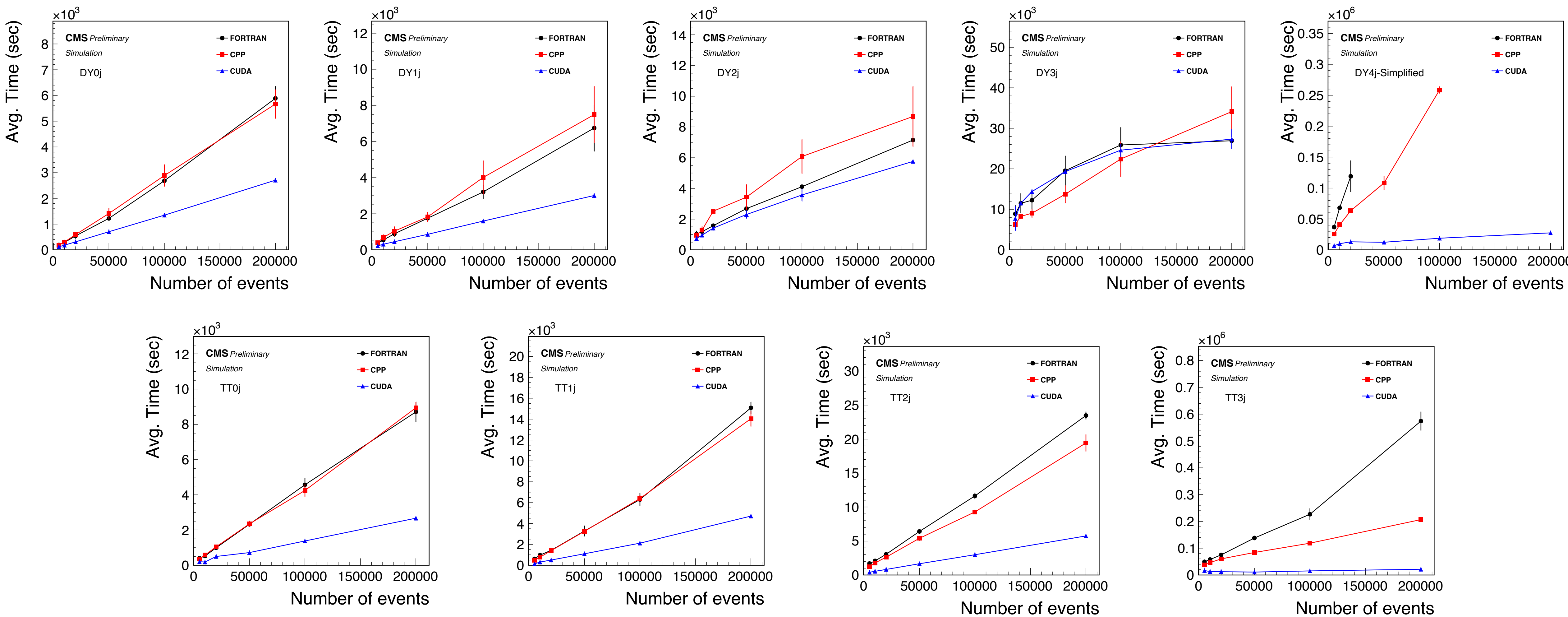
[4] MadGraph4GPU project, GitHub code repository: https://github.com/madgraph5/madgraph4gpu

[5] CMS genproduction, GitHub code repository: https://github.com/cms-sw/genproductions

BACK UP

EVENT GENERATION

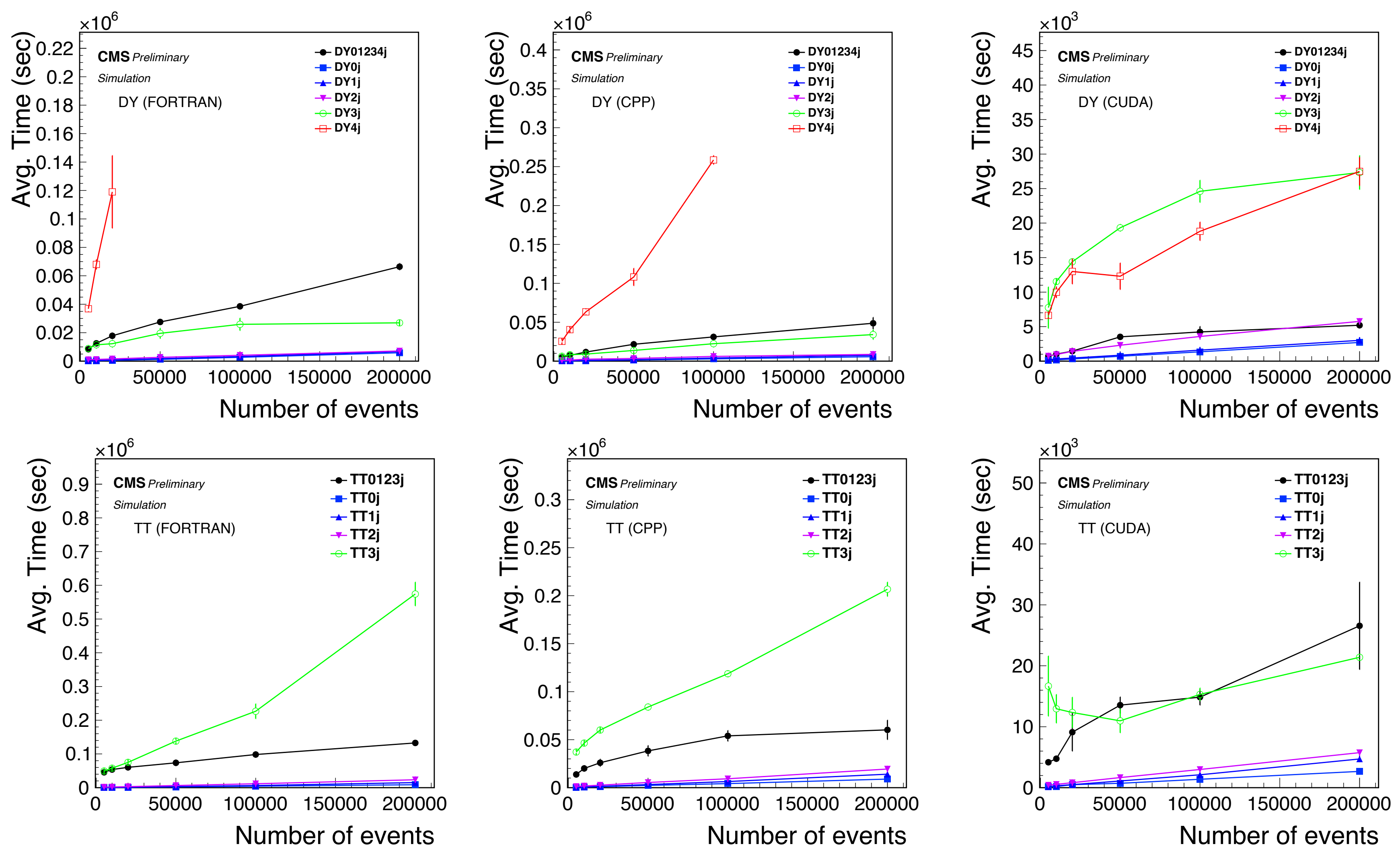
Jet-binned results



- Speed-up starts with DY+4j / TT+2j

EVENT GENERATION

Other Plots - Jet-binned vs. Inclusive Comparison



Caption: Event generation timing, comparing within the same backend. "FORTRAN" refers to the backend used in earlier versions of MadGraph, while "CPP" and "CUDA" represent different backend variations. Each process was tested with 5k, 10k, 20k, 50k, 100k, and 200k events. For the DY4j-Simplified process, the 50k, 100k, and 200k event data points for "FORTRAN" and the 200k data point for "CPP" were omitted due to limited resources. In the DY4j-Simplified CUDA case, event generation often reached full GPU usage within a few madevent executions (~5), causing significant overhead when multiple madevents were run simultaneously on the same GPU. This led to the 20k event generation taking longer than the 50k event generation in some cases.