High performance with MG5aMC

Olivier Mattelaer
University Catholique de Louvain

Plan of my talk

plan

- Trick for (B)SM generation
- Speed up of the code at 0 cost.
- MPI
- GPU

Th uncertainty @NLO

- generate p p > t t~ [QCD]
- output
- launch
  - set store_rwgt_info T

- systematics run_01 (**OFFLINE**)

```
INFO: # events generated with PDF: NNPDF23_nlo_as_0119_qed (244800)
INFO: #Will Compute 144 weights per event.
INFO: #***********************************************************************
#
# original cross-section: 704.418156719
#     scale variation: +9.75% -10.7%
#     central scheme variation: + 0% -28.2%
# PDF variation: +1.55% -1.55%
```

- Allow to reweight sample with **FUTURE** pdf keeping the NLO acccuracy
  - **Trade** of speed with disk space

Re-Weighting

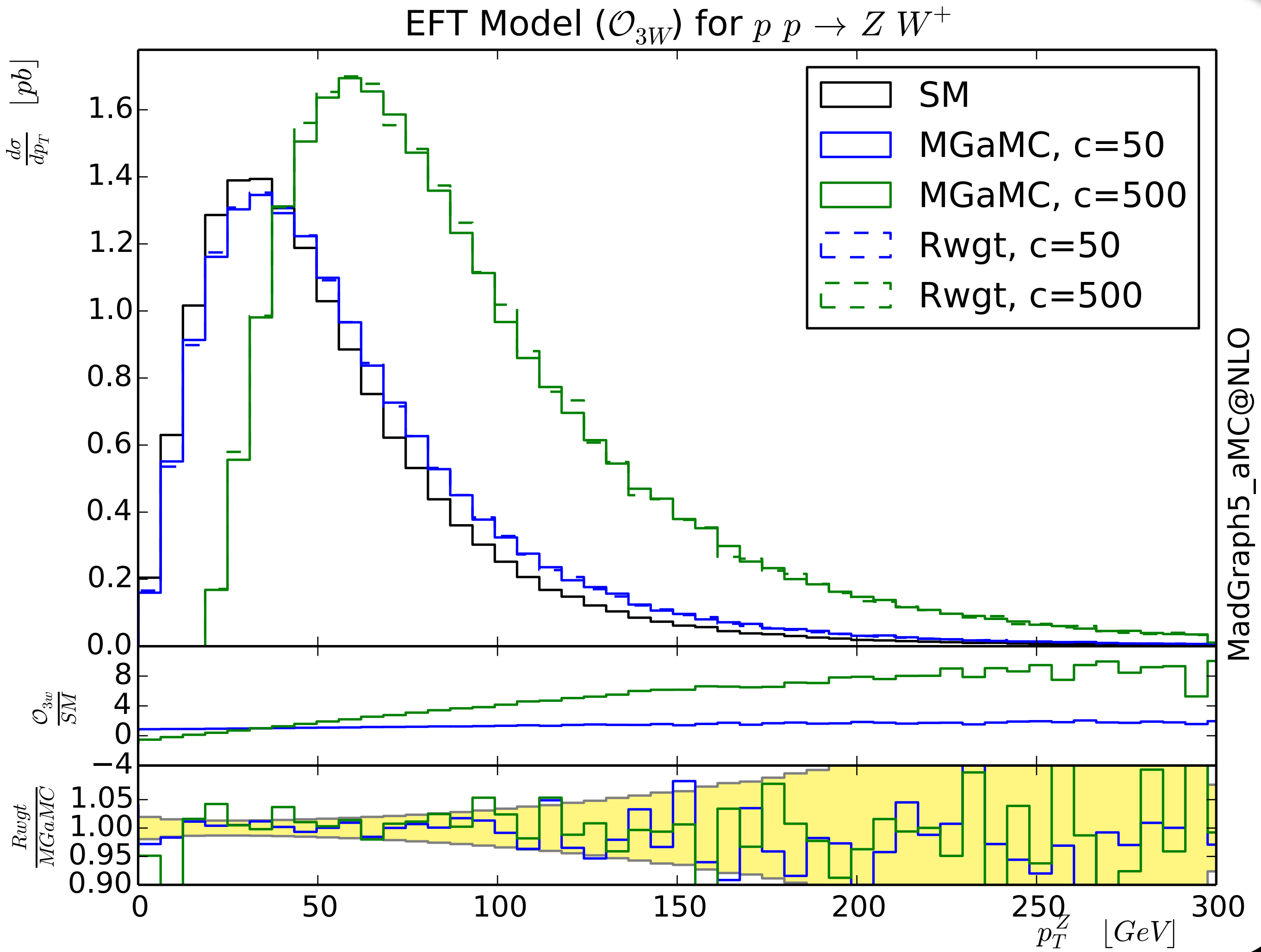
1404.7129
1607.00763

- Change the weight of the events

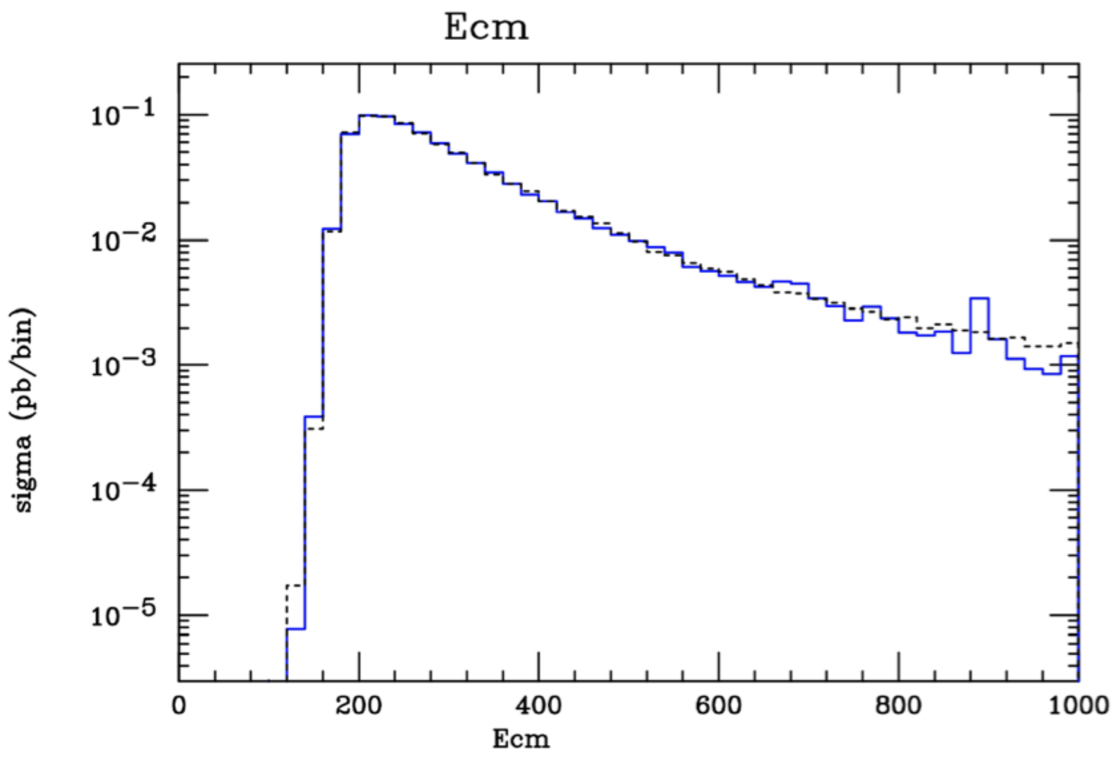
$$W_{new} = \frac{|M_{new}|^2}{|M_{old}|^2} * W_{old}$$

EFT Case

$$\mathcal{O}_{3W} = Tr\left[W_{\mu\nu}W^{\nu\rho}W_{\rho}{}^{\mu}\right]$$



Re-Weighting Limitation



$$\Delta\sigma_{new} = \frac{\sigma_{new}}{\sigma_{old}} \Delta\sigma_{new} + \frac{Var_{wgt}}{\sqrt{N}} \sigma_{old}$$

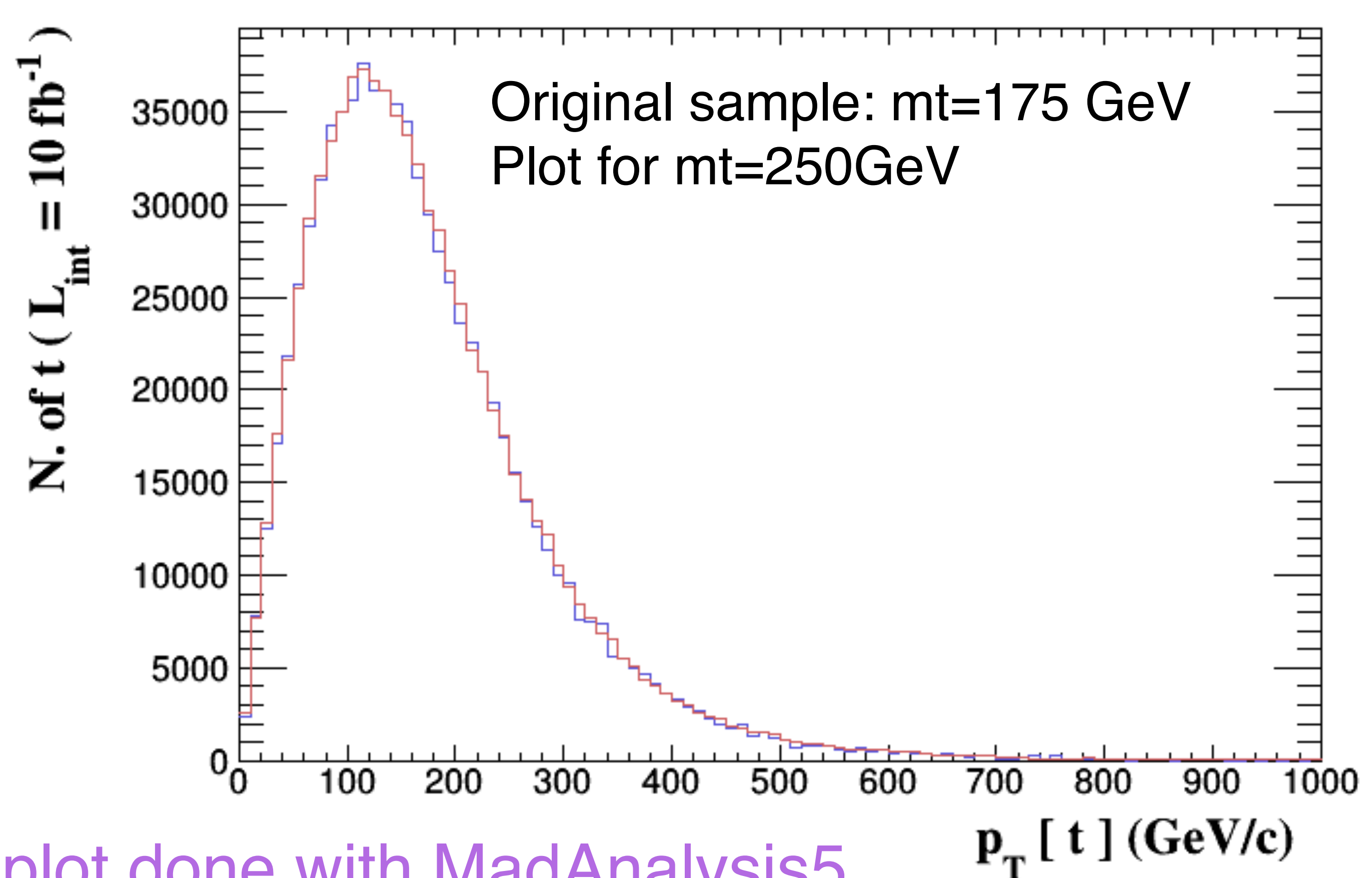
- statistical uncertainty can be enhanced by the re-weighting

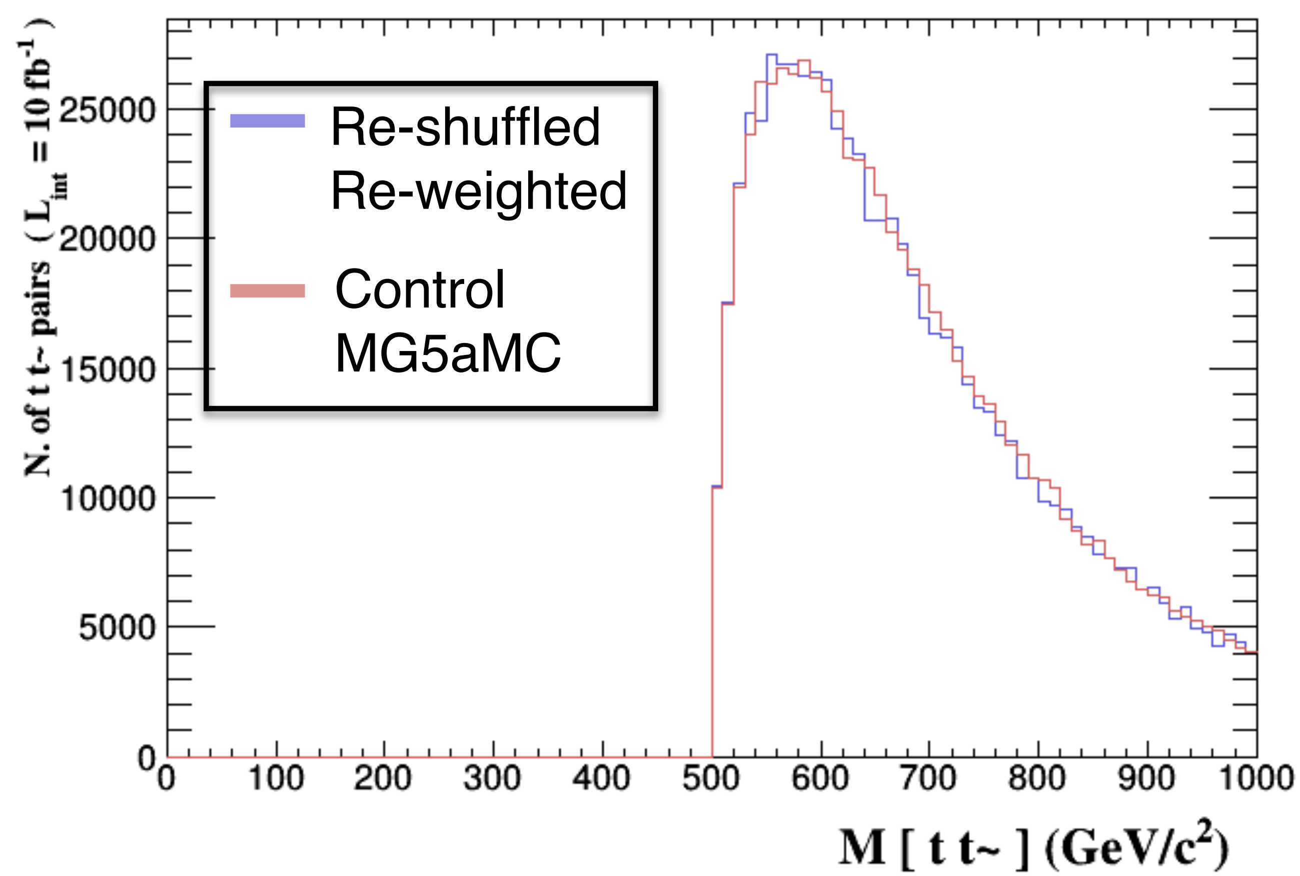
Limitation

- You need to have the same phase-space (more exactly a subset)
  - Mass scan are possible only in special case

Re-Weighting for mass scan

- Use Rambo mass re-shuffling method for generating kinematics at new mass point [CERN-TH.4299]
- Use standard Re-weighting approach to get correct weight.
  - Therefore you can also change spin (stop pair. production form tt~ sample)





plot done with MadAnalysis5

NLO Re-Weighting

NLO method

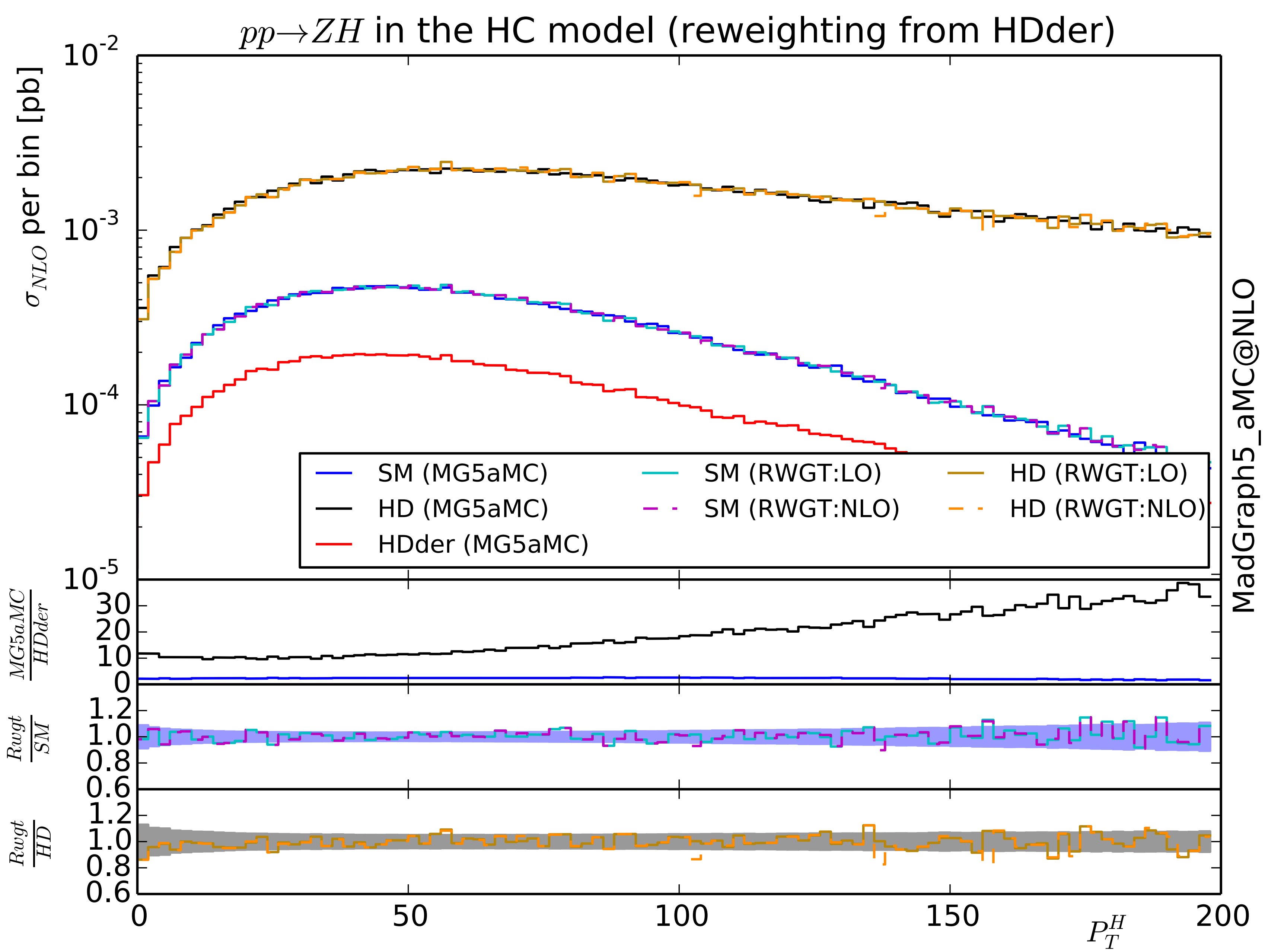
- tracks the dependencies in the various matrix-elements (born, virtual, real)

$$d\sigma^\alpha = f_1(x_1,\mu_F) f_2(x_2,\mu_F) \Big[\mathcal{W}_0^\alpha + \mathcal{W}_F^\alpha \log\left(\mu_F/Q\right)^2 + \mathcal{W}_R^\alpha \log\left(\mu_R/Q\right)^2 \Big] \, d\chi^\alpha ,$$

$$\mathcal{W}_\beta^\alpha = \mathcal{B} * \mathcal{C}_{\beta,B}^\alpha + \mathcal{B}_{CC} * \mathcal{C}_{\beta,B_{CC}}^\alpha + \mathcal{V} * \mathcal{C}_{\beta,V}^\alpha + \mathcal{R} * \mathcal{C}_{\beta,R}^\alpha$$

- re-weight each part according to the associated matrix-element
  - need the same information as for systematics

NLO example



$pp \rightarrow ZH$ in the HC model (reweighting from HDder)
σ_{NLO} per bin [pb]
10^{-2}
10^{-3}
10^{-4}
10^{-5}
SM (MG5aMC)
HD (MG5aMC)
HDder (MG5aMC)
SM (RWGT:LO)
SM (RWGT:NLO)
HD (RWGT:LO)
HD (RWGT:NLO)
MadGraph5_aMC@NLO
$\frac{MG5aMC}{HDder}$
30
20
10
0
$\frac{Rwgt}{SM}$
1.2
1.0
0.8
0.6
$\frac{Rwgt}{HD}$
1.2
1.0
0.8
0.6
0
50
100
150
200
P_T^H

Plan of my talk

plan

- Trick for (B)SM generation
- Speed up of the code at 0 cost.
- MPI
- GPU

GCC/Intel

Compiler option

- MadGraph is conservative on compiler flag option (-O1)

Aggressive flag

- Using -Ofast
  - ➡ Code 30% faster at LO/NLO (tested on tt~jjj @LO and tt~j @NLO)
- ➡ Flag breaking standard (-> need validation)
  - ➡ Validation needed but worth

Profile based compilation

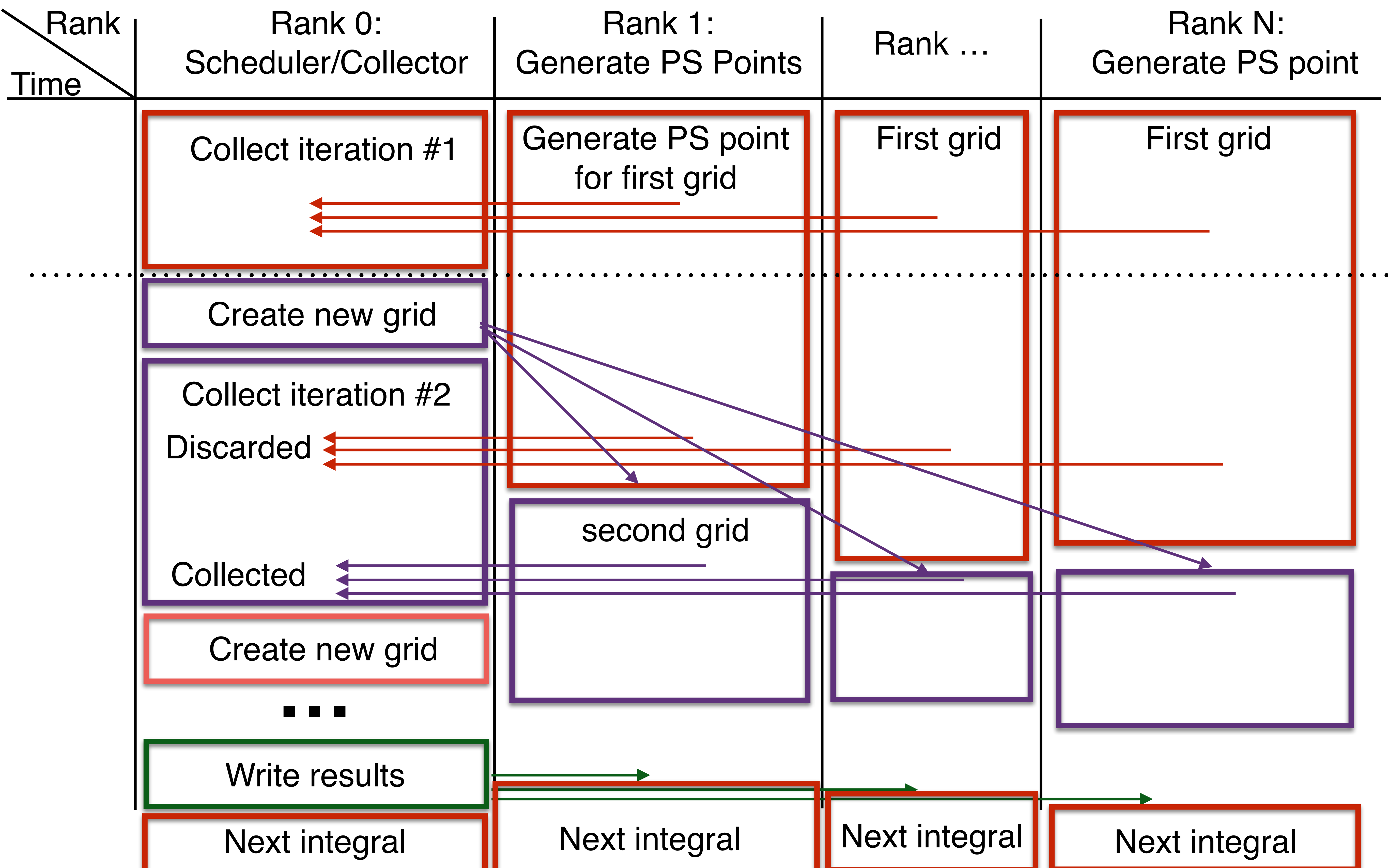
- Marginal gain (1%) and very long setup (LO)

Plan of my talk

plan

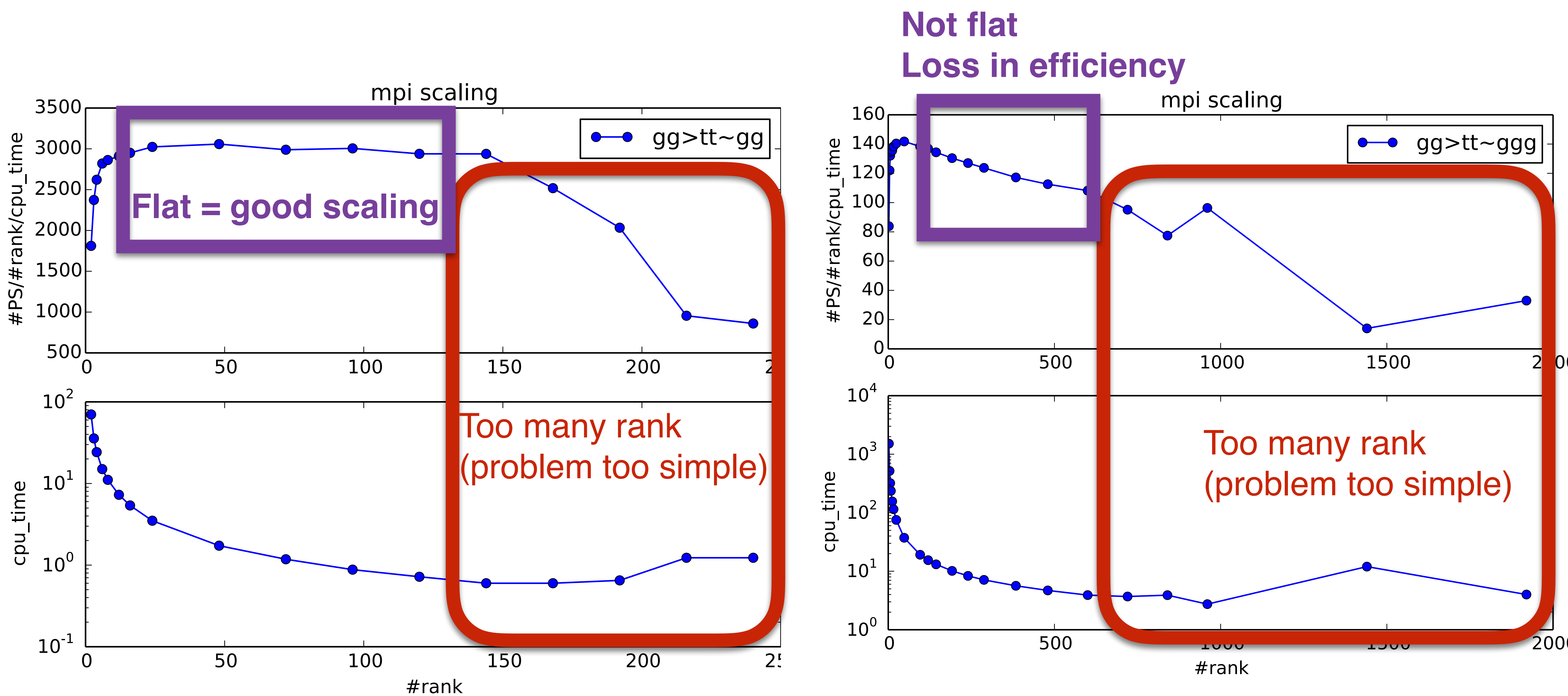
- Trick for (B)SM generation
- Speed up of the code at 0 cost.
- MPI
- GPU

MPI Strategy



Timing

- One Single integral timing (gridpack creation)



Integration time: No initialisation and submission time
-> We need to group the channel to be slow enough!

Situation

LO Strategy situation

- Do not scale higher than 500-2000 rank
- Assume that all PS point takes the same time to compute
  - ➡ If this is not the case, this method can induce bias
- Discarding events is at the end as bad as waiting doing nothing
- This method can run with **slow communication** and with **different arch** in the pool (good for **Tier2**)

NLO situation

- All phase-space point do not take the same amount of cpu time (variation by two order of magnitude
- Need other strategy for having the scaling

HTC vs HPC

	HTC cluster	HPC/MPI
Total waiting time		
Total cpu time		
Job granularity	faster on queue	
Infrastructure cost		+ 30% due to infiniband/OPA
GCC flag		-march=native

WINNER: The Turtle!

Plan of my talk

plan

- Trick for (B)SM generation
- Speed up of the code at 0 cost.
- MPI
- GPU

Bibliography

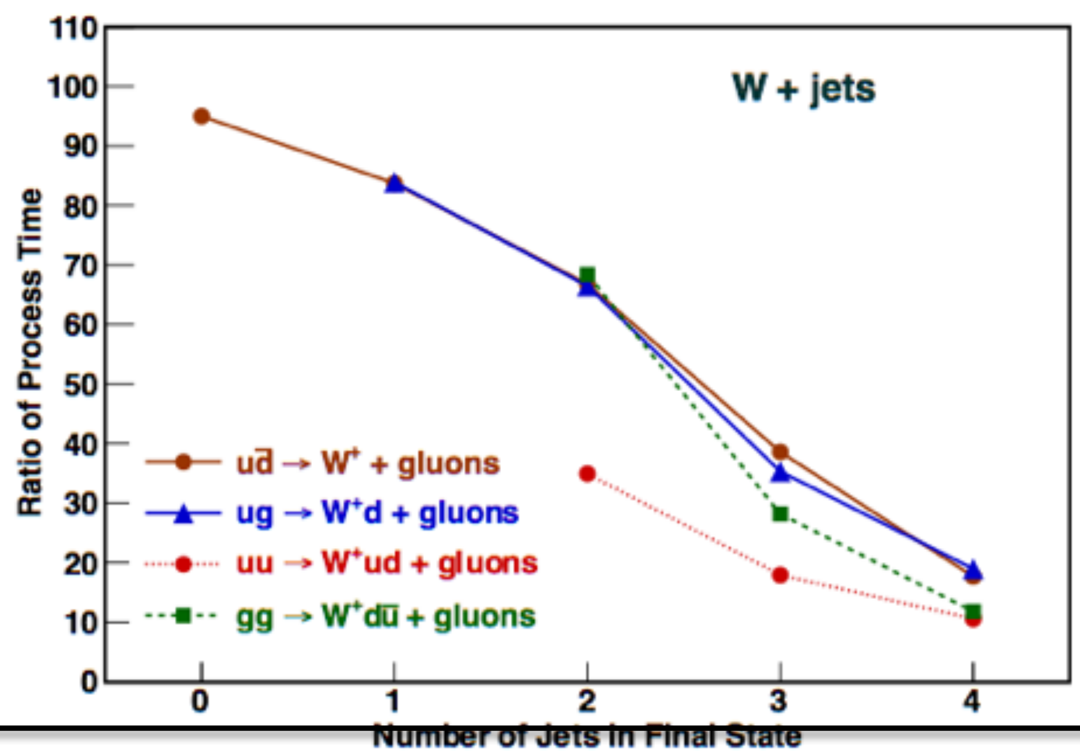
- QED: K. Hagiwara, J. Kanzaki, N. Okamura, D. Rainwater and T. Stelzer, Eur. Phys. J. C66 (2010) 477, e-print arXiv:0908.4403.
- QCD: K. Hagiwara, J. Kanzaki, N. Okamura, D. Rainwater and T. Stelzer, Eur. Phys. J. C70 (2010) 513, e-print arXiv:0909.5257.
- MC integration (VEGAS & BASES): J. Kanzaki, Eur. Phys. J. C71 (2011) 1559, e-print arXiv:1010.2107.
- SM: K. Hagiwara, J. Kanzaki, Q. Li, N. Okamura, T. Stelzer, Eur.Phys.J. C73 (2013) 2608 (2013), e-print arXiv:1305.0708v2.

Status

- All SM processes tested in 2013
- Efficiency and/or more jet should be possible with latest GPU

processes:

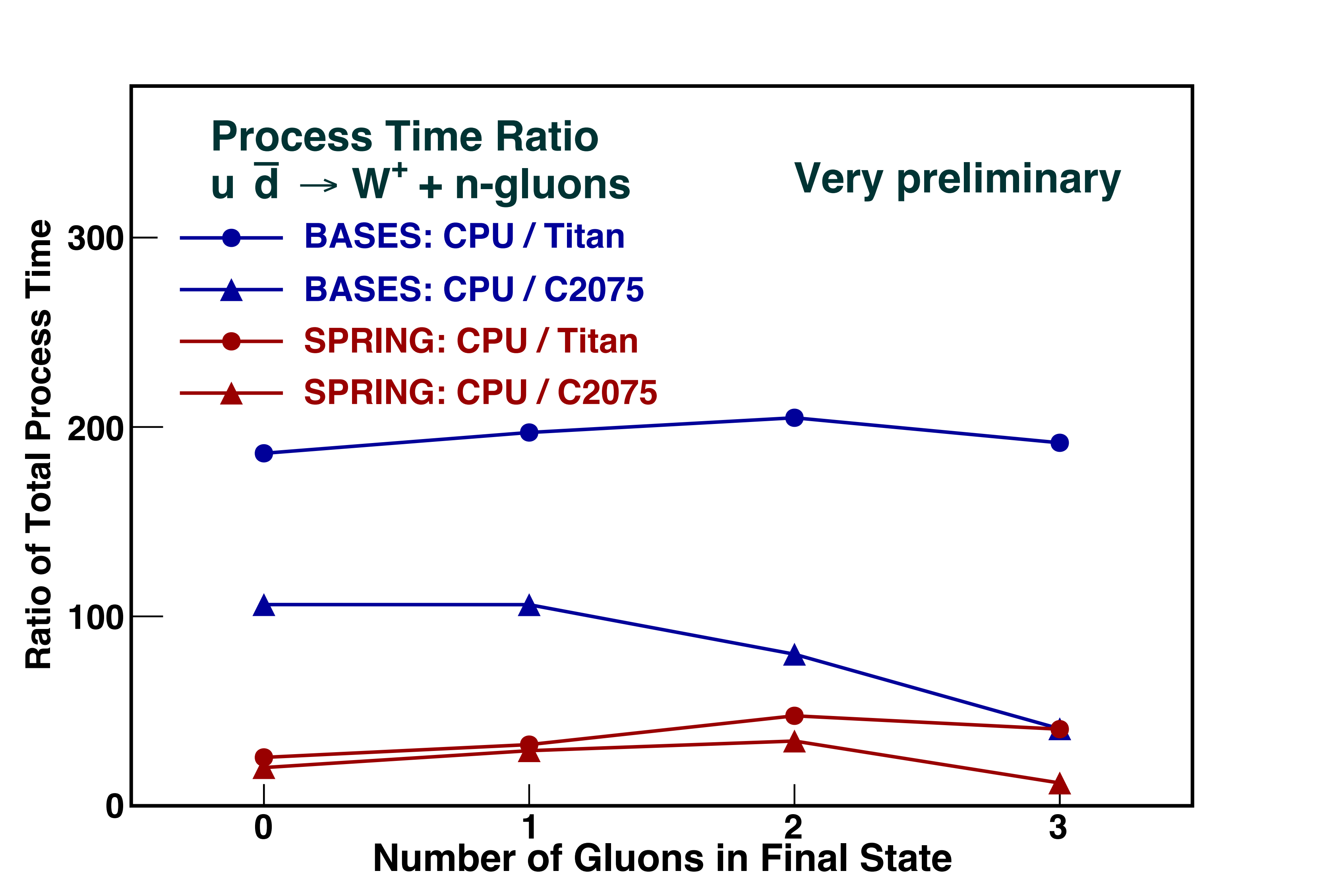
$W/Z + n$-jets	$(n \leq 4)$,
$WW/WZ/ZZ + n$-jets	$(n \leq 3)$,
$t\bar{t} + n$-jets	$(n \leq 3)$,
$HP/HZ + n$-jets	$(n \leq 3)$,
$Ht\bar{t} + n$-jets	$(n \leq 2)$,
$H^k + (n-k)$-jets via WBF	$(k \leq 3, n \leq 5)$.



- GPU/CPU
- GPU (C2085) [2011]
- CPU: i7 (2.7Ghz) [2011]
- High gain (especially at low multiplicities

Ratio of process time (C2075 & Titan)

- Preliminary results on C2075 and Titan.



CPU vs GPU

	CPU	GPU
Efficiency		
Cost		
Efficiency/cost	???	???
Code development		CUDA/MEM
Multiplicities		

NO
clear winner
Likely GPU

Conclusion

- HTC
  - Store more to compute less
- HPC/MPI
  - Working but we should not push in that direction
    - I'm happy to help to deploy it on existing HPC farm
- GPU
  - Promising result but seem to suffer from a lack of interest