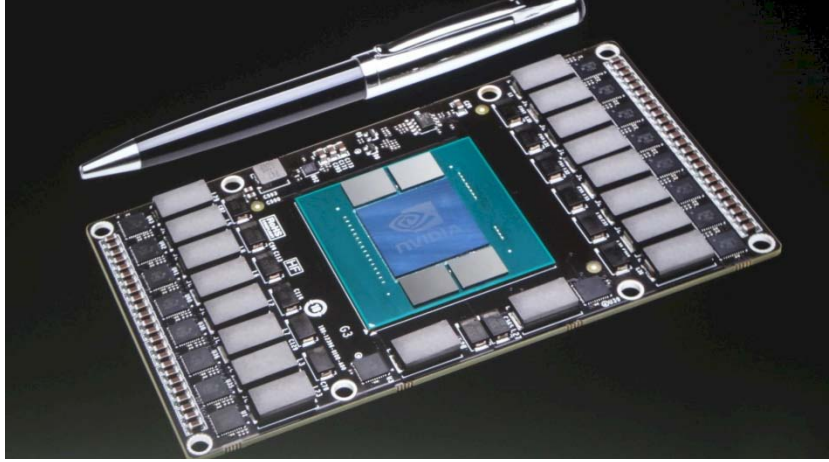
Implementing NNLO into MCFM

Downloadable from **mcfm.fnal.gov**

- A Multi-Threaded Version of MCFM, J.M. Campbell, R.K. Ellis, W. Giele, 2015
- Higgs boson production in association with a jet at NNLO using jettiness subtractions, R. Boughezal, C. Focke, W. Giele, X. Liu, F. Petriello, 2015
- Color singlet production at NNLO in MCFM, R. Boughezal, J.M. Campbell, R.K. Ellis, C. Focke, W. Giele, X. Liu, F. Petriello, C. Williams, 2016

The publicly available code MCFM 8.0 provides predictions for color singlet production at Next-to-Next-to Leading Order. The code runs in parallel using a hybrid openMP/MPI version of Vegas enabling MCFM to get accurate distributions in a few hours of runtime on moderate clusters of a few hundred computing cores. Included processes in this version are *W, Z, H, WH, ZH* and *di-photon* production.

Impact of technology on tools



- Used to programming in a non-parallel manner with a glut of memory favoring storing over on the fly recalculation.
- Technology is moving rapidly to “many integrated core” architecture.
- To take advantage of these new developments, parallel programming is needed (openMP, CUDA, openACC,…).
- Optimal methods to do Monte Carlo programs can shift in non-intuitive manners due to shifts in technology:
  - Limiting off-chip memory → PDF grid vs PDF evolution
  - Preference for brute force simple methods → slicing vs subtraction
- With careful programming one can ultimately run the algorithm on GPU’s

MCFM@NNLO

- Requirements for NNLO Monte Carlo's:
  - **Fast** and **accurate** NLO Monte Carlo predictions extended into the soft/collinear regions in phase space. These are the bremsstrahlung events for NNLO.
    MCFM provides validated and precise predictions for this using analytic matrix elements developed for almost 2 decades.
  - MCFM has hundreds of processes already build in at NLO, reusing these to upgrade the Monte Carlo to NNLO will be a real time saver.

MCFM for the Tevatron and the LHC, J.M. Campbell and R.K. Ellis, 2010

MCFM@NNLO

- Implementation:
  - Need of subtraction scheme to regularize the soft/collinear divergences. We use “non-local jettiness subtraction”. This is a slicing method with the advantage that all analytic slicing functions have been calculated already.
    - W-boson production in association with a jet at next-to-next-to-leading order in perturbative QCD, R. Boughezal, C. Focke, X. Liu and F. Petriello, 2015
    - N-jettiness Subtractions for NNLO QCD Calculations, J. Gaunt, M. Stahlhofen, F.J. Tackmann and J.R. Walsh, 2015
  - This is similar in methodology to q_t-subtraction, however it also works for jet final states.
    - An NNLO subtraction formalism in hadron colliders and its applications to Higgs boson production at the LHC, S. Catani and M. Grazzini, 2007

MCFM@NNLO

The jettiness factorization method tells us how to combine all the functions.

Hard scattering factorization from effective field theory,
C. Bauer, S. Fleming, D. Pirjol, Z. Rothstein and I. Stewart, 2002

N-jettiness: An Inclusive Event Shape to Veto Jets,
I.W. Steward, F.J. Tackmann and W.J. Waalewijn, 2010

Shown are 2 slicing contributions

- Towards a NNLO … Two-loop results for the jet function,
  T. Becher and M. Neubert, 2006
- The gluon jet function at two-loop order,
  T. Becher and G. Bell, 2011
- The two-loop hemisphere soft function,
  R. Kelley, M.D. Schwartz, R.M. Schabinger, H.X. Zhu, 2011
- The Quark Beam Function at Two Loops,
  J.R. Gaunt, M. Stahlhofen and F.J. Tackmann, 2014
- The Gluon Beam Function at Two Loops,
  J.R. Gaunt, M. Stahlhofen and F.J. Tackmann, 2014
- N-jettiness soft function at next-to-next-to-leading order,
  R. Boughezal, X. Liu and F. Petriello, 2015

$$\int_0^{\mathcal{T}_0^{cut}} \mathrm{d}\mathcal{T}_0\; S^{(2)} \otimes \mathcal{I}_{ai}^{(0)} \otimes \mathcal{I}_{bj}^{(0)} = \delta_{ai}\,\delta_{bj} \left\{ s_{-1}^{(2)} + \sum_{n=0}^{3} \frac{1}{n+1}\, s_n^{(2)}\, L^{n+1} \right\} ,$$

$$\int_0^{\mathcal{T}_0^{cut}} \mathrm{d}\mathcal{T}_0\; S^{(1)} \otimes \mathcal{I}_{ai}^{(1)} \otimes \mathcal{I}_{bj}^{(0)} =$$

$$\delta_{bj} \left\{ s_{-1}^{(1)} i_{-1,ai}^{(1)}(z) + s_{-1}^{(1)} \sum_{n=0}^{1} \frac{1}{n+1}\, i_{n,ai}^{(1)}(z)\, L^{n+1} + i_{-1,ai}^{(1)}(z) \sum_{n=0}^{1} \frac{1}{n+1}\, s_n^{(1)}\, L^{n+1} \right.$$

$$\left. + \sum_{m,n=0}^{1} s_m^{(1)} i_{n,ai}^{(1)}(z) \Gamma_{m,n} \right\} , \qquad (2.13)$$

where

$$L = \ln\left(\frac{\mathcal{T}_0^{cut}}{\mu}\right) . \qquad (2.14)$$

and

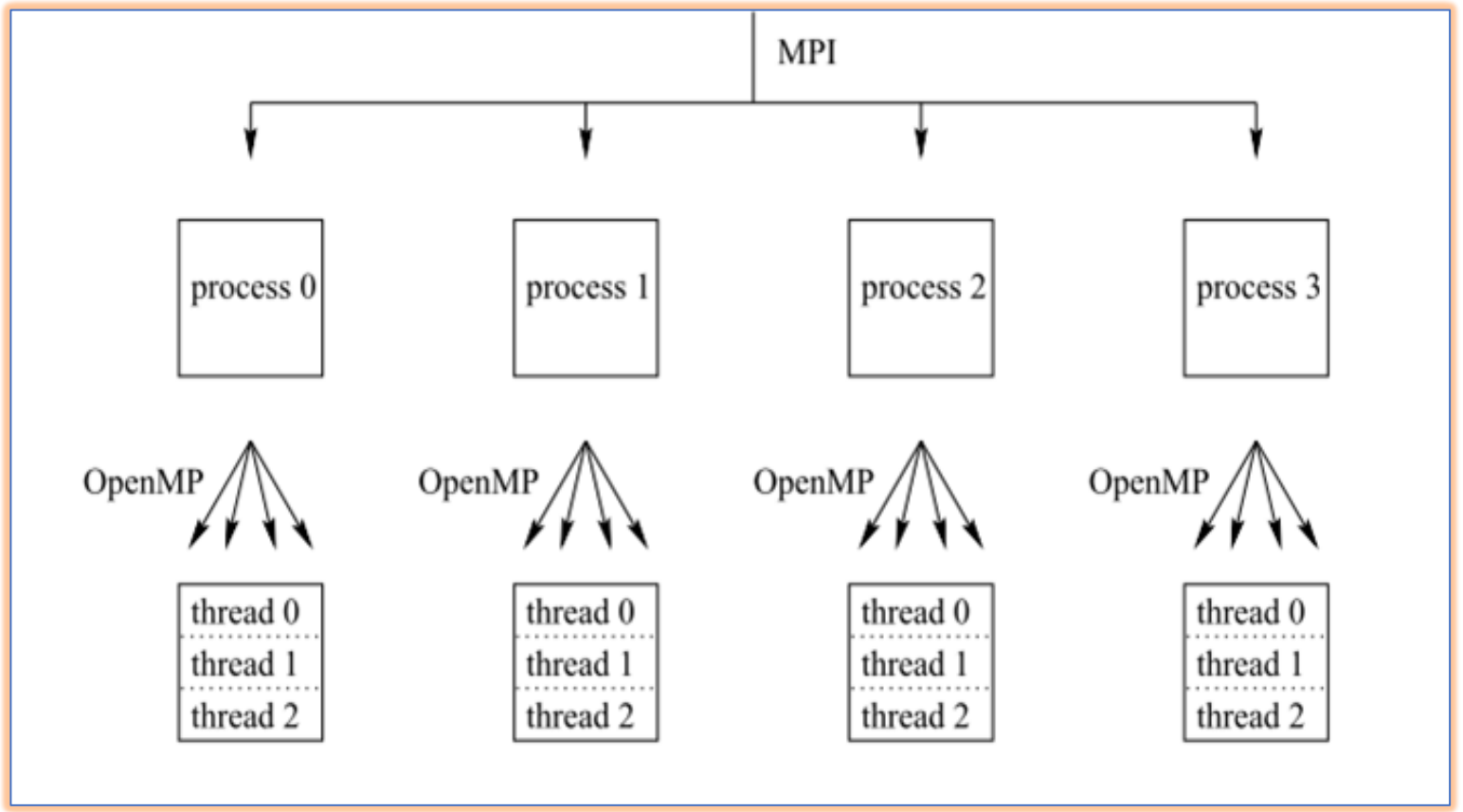
$$\begin{aligned} \Gamma_{0,0} &= L^2 - \zeta_2 \\ \Gamma_{1,0} &= \Gamma_{0,1} = \frac{L^3}{2} - \zeta_2 L + \zeta_3 \\ \Gamma_{1,1} &= \frac{L^4}{4} - \zeta_2 L^2 + 2\zeta_3 L - \frac{\zeta_2^2}{10} . \end{aligned} \qquad (2.15)$$

MCFM@NNLO

- Combining the jettiness approach with MCFM to produce NNLO predictions works well.
- Example: A sketch of making NNLO ***PP → V + 1 jet***
  - Use MCFM NLO ***PP→V+ 2 jets*** prediction and modify it so we produce 1 jet and regulate the resulting soft/collinear singularity with the jettiness slicing cut
  - Add the already calculated analytically integrated part below the slicing cut (the beam, jet and soft functions) to the already calculated 2-loop ***PP → V+1 jet***
  - Combine the two and the slicing cut cancels, giving us the NNLO prediction
    - W-boson production in association with a jet at NNLO in pQCD, R. Boughezal, C. Focke, X. Liu, F. Petriello, 2015
    - Higgs boson production in association with a jet at NNLO using jettiness subtractions, R. Boughezal, C. Focke, W. Giele, X. Liu, F. Petriello, 2015
    - Z-boson production is association with a jet at NNLO in pQCD, R. Boughezal, J.M. Campbell, R.K. Ellis, C. Focke, W. Giele, X. Liu, F. Petriello, 2015

Parallelism in MCFM: hybrid openMP/MPI

- Two levels of parallelism:
  - OpenMP: For use on single processor to use the multiple cores on the processor. Uses a unified memory. Time consuming to develop, easy to use
  - MPI: Connects the different processors. Communication through messaging over e.g. infiniband. Quick to develop, harder to use.

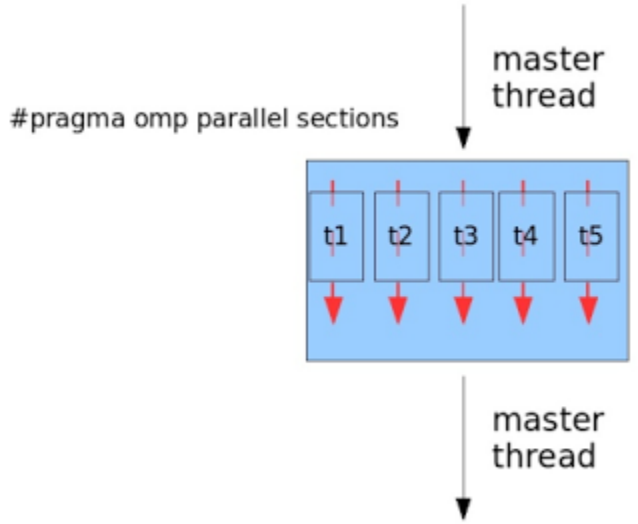


With the advance of MIC architecture with unified memory openMP/openACC will become more and more important in the future.

MCFM 7: openMP

A Multi-Threaded Version of MCFM, J.M. Campbell, R.K. Ellis, W. Giele, 2015

- openMP can be used to parallelize a program using the available cores on a single motherboard.
- All cores see the same memory and cache (per processor).
- Shared memory leads to great speedups but also makes development/debugging harder.
- openMP is part of Intel and GNU compilers. It uses available cores without user interference. It is actively pushed by Intel for their MIC architecture (synergy between hardware and openMP)
- Simple pragma's are added to existing code (which are comments for non-openMP compilations)



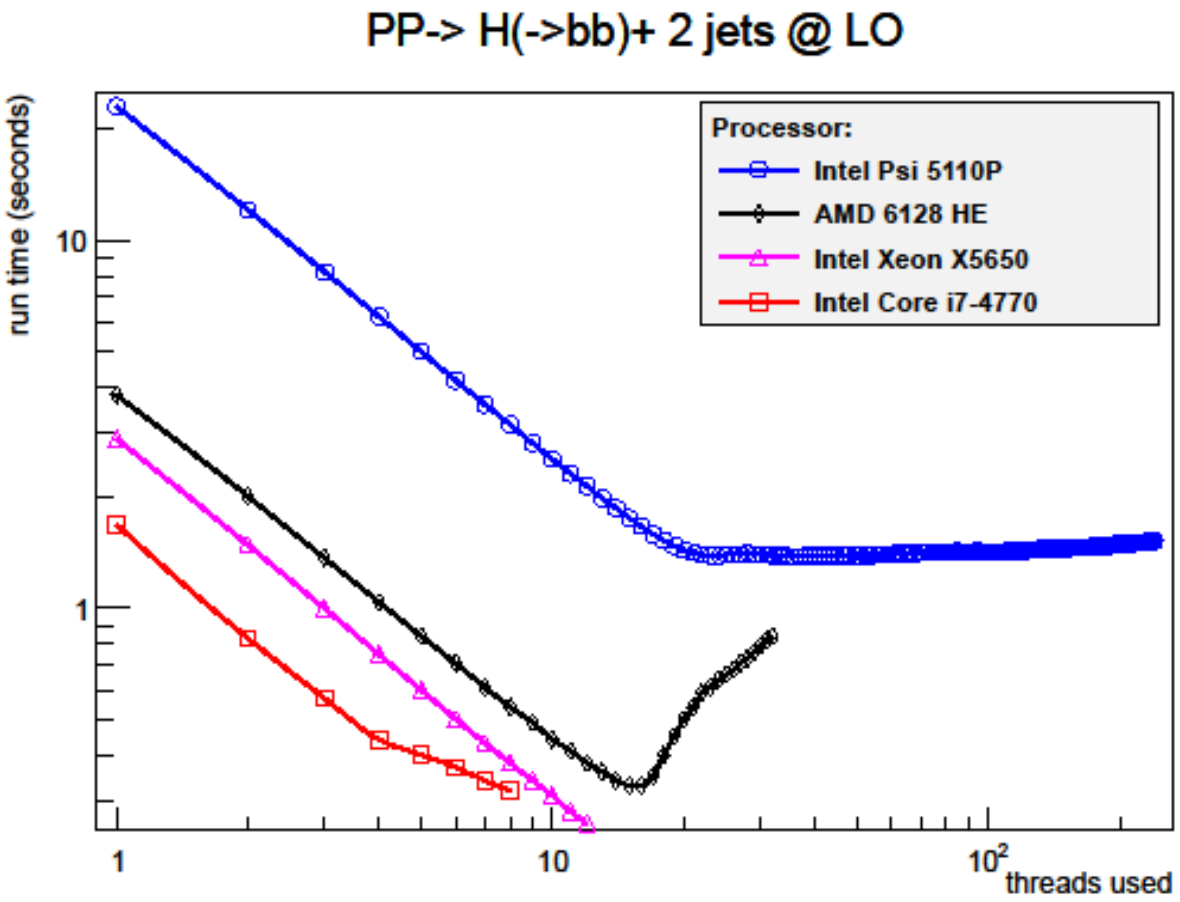
MCFM 7: NLO using openMP

(The NNLO bremsstrahlung part)

- We use 4 different configurations to calculate ***pp →H+2 jets*** at NLO:
  - Desktop using an Intel core I7-4770 (4 cores, 3.4Ghz, 8Mb cache)
  - Two Intel x5650 processors (2x6 cores, 2.66Ghz,12Mb cache)
  - Quadruple AMD 6128 HE opteron (4x8 cores, 2Ghz, 12Mb cache)
  - Xeon Phi co-processor (4x64 threads, 1.1Ghz, 28.5 Mb cache)
    New Xeon Phi stand-alone: 18 cores, 2.3-3.6 GhZ, 45Mb cache.
    → MPI a disaster, openMP a must.

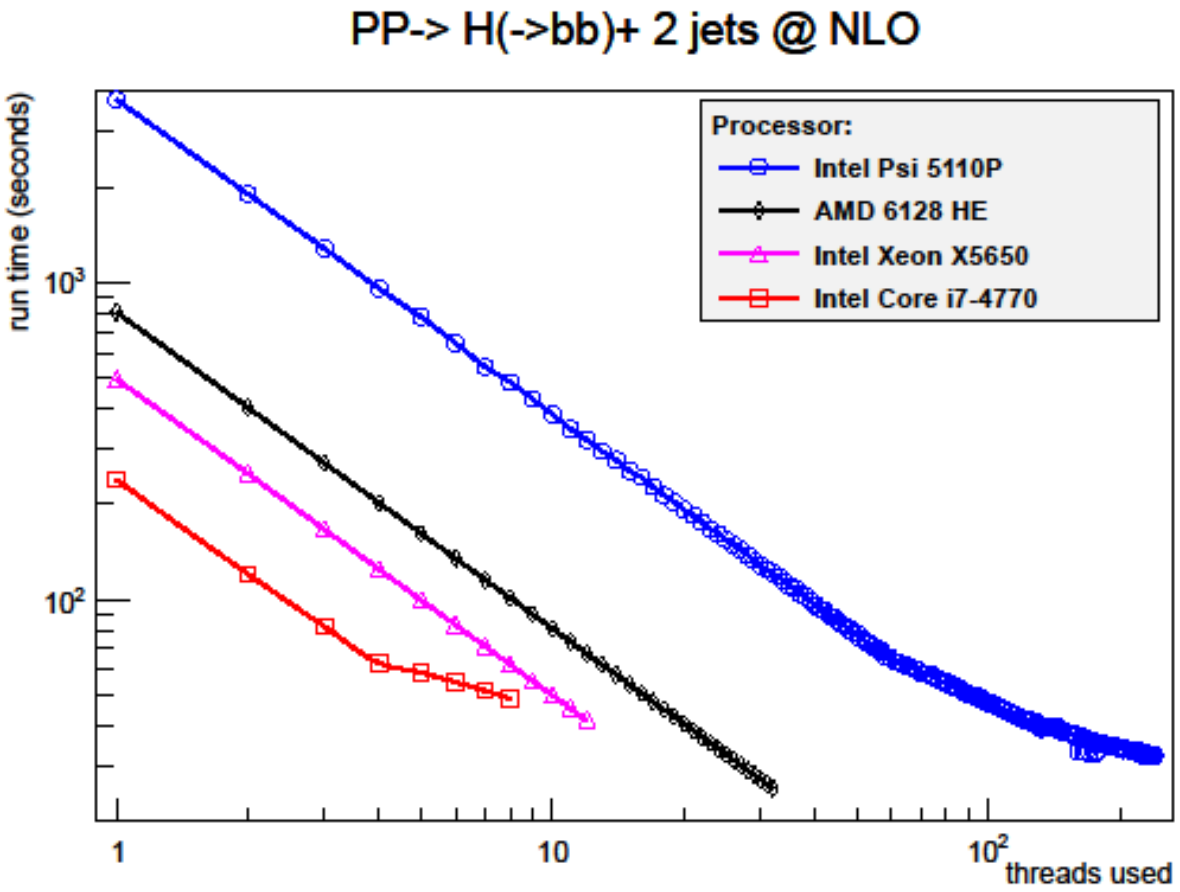
MCFM 7:openMP at LO

- Used *4x1,000+10x10,000* VEGAS events.
- Under perfect circumstances one would expect the run time to go as $t_n = t_1/n$.
- Deviations from that due to:
  - Hyper-threading
  - Memory bound limits because we do not calculate enough (memory transfer time dominates over computational time)
  - Traffic jams: We have excessive memory transfer between cache and main memory
  - Starvation: Number of events per thread low



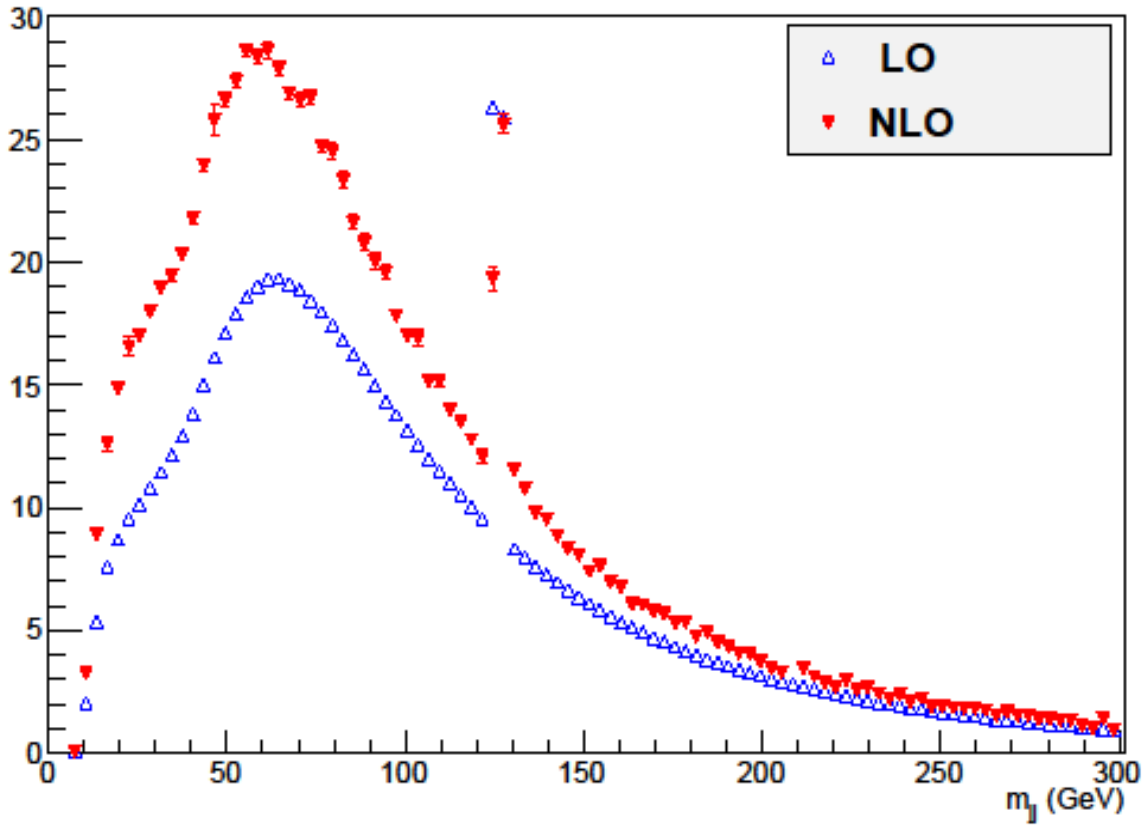
MCFM 7:openMP at NLO

- Used *4x1,000+10x10,000* VEGAS events
- Because the calculations are more complicated all memory bound issues have disappeared.
- openMP allows for doing the NLO phenomenology in on a workstation. No cluster required.
- Except for hyper threading the scaling is nearly perfect.



MCFM 7: NLO on a workstation

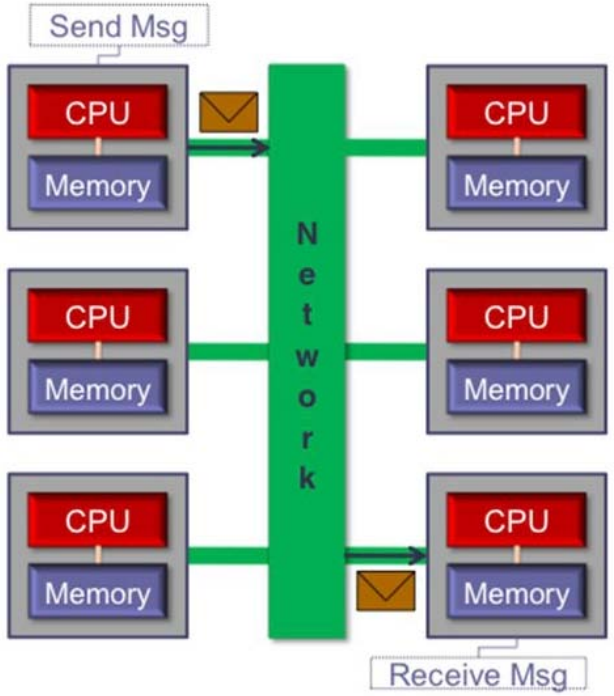
- The dijet invariant mass distribution (5 GeV bins) for *PP → H(→bb) + 2 jets* at NLO
- Used *4x1,500,000+10x15,000,000* VEGAS events
- LO: 12 min on *2x6* core dual Intel Xeon X5650
- NLO: 22 hours on the *4x8* core quad AMD Opteron system



MCFM 8: NNLO using hybrid openMP/MPI

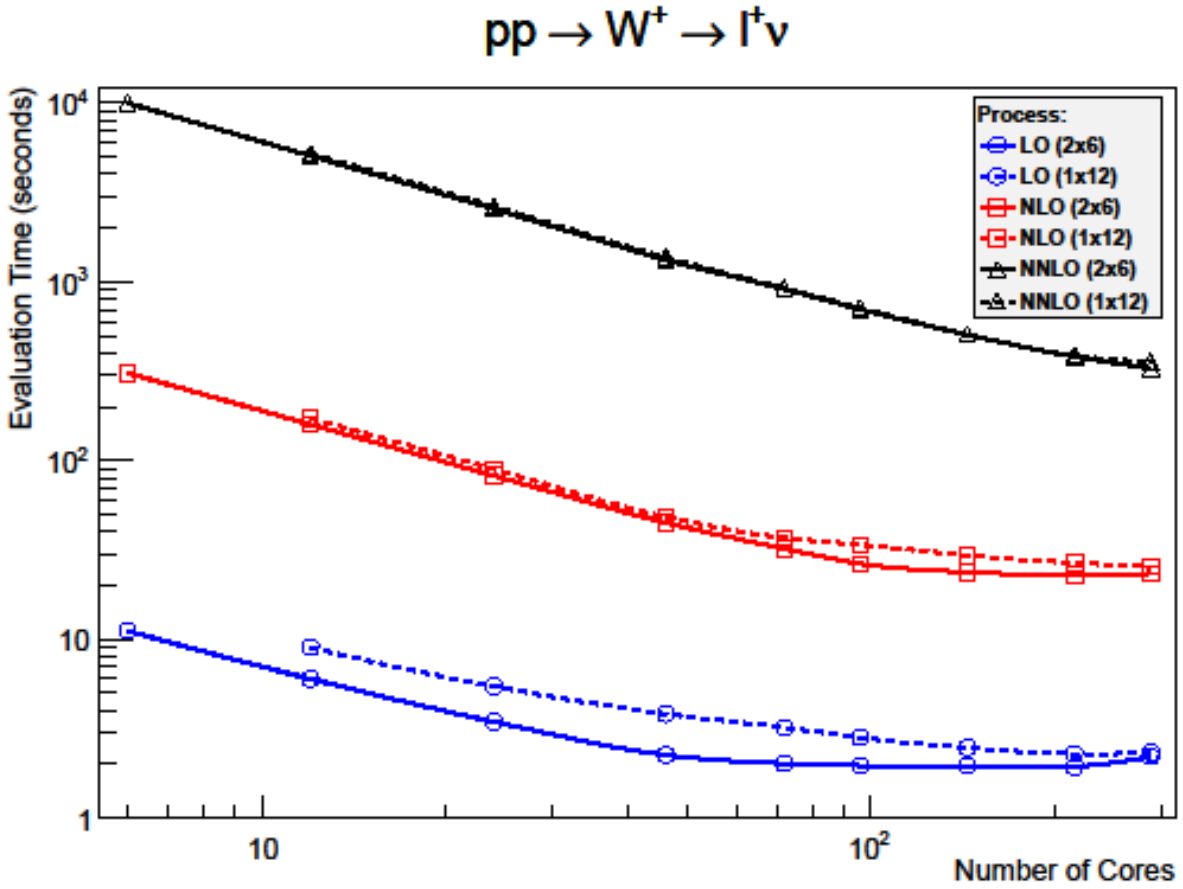
- Higgs boson production in association with a jet at NNLO using jettiness subtractions,
  R. Boughezal, C. Focke, W. Giele, X. Liu, F. Petriello, 2015
- Color singlet production at NNLO in MCFM,
  R. Boughezal, J.M. Campbell, R.K. Ellis, C. Focke, W. Giele, X. Liu, F. Petriello, C. Williams, 2016

- MPI can be used to parallelize the program over multiple processors, each with its own memory/cache.
- Communication between the different processors is slow and should be minimized.
- Easy to program, though one has to add explicit program statements. So program will not compile/run outside MPI.
- Harder to use, need some software to submit jobs in queues. No MPI standard.



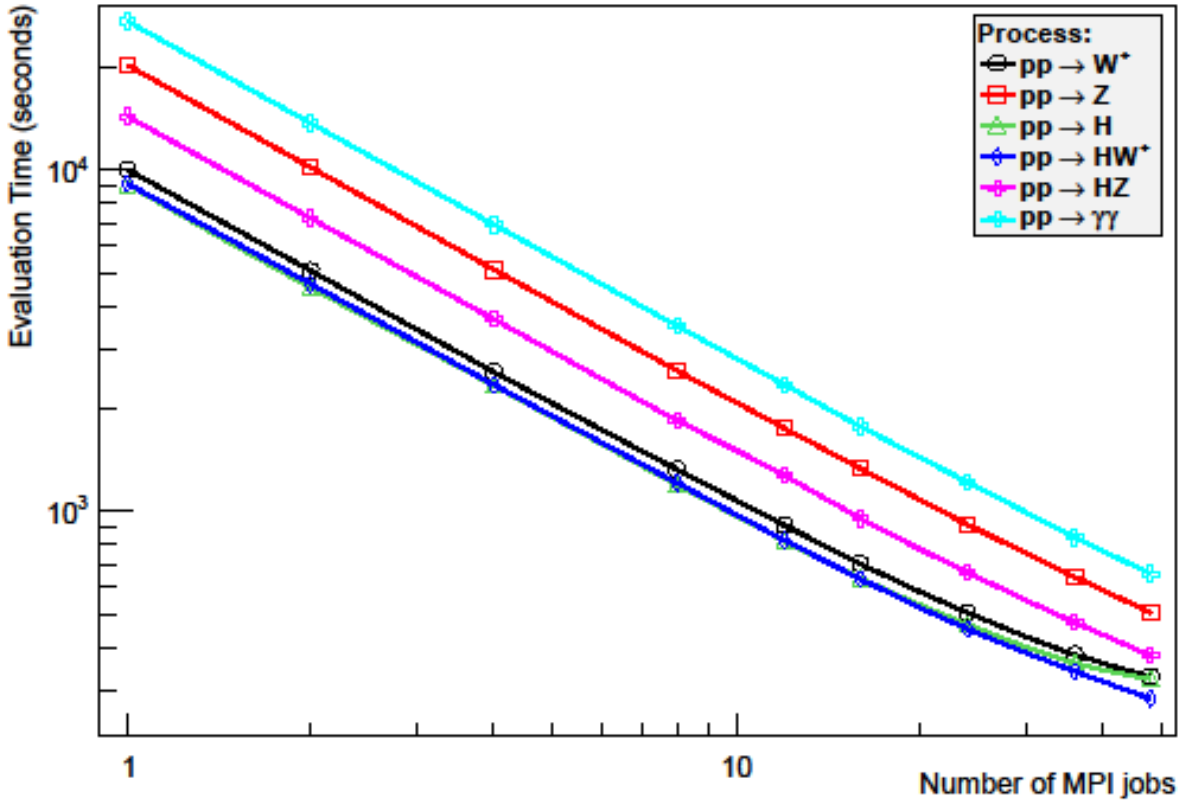
MCFM 8: Thread scaling (1)

- Runtime of *pp →W⁺* for LO/NLO/NNLO from 1 up to 288 cores
- The cluster consists of 24 nodes, each containing 2 processors of 6 cores
- Two running modes:
  - 1 MPI job per node: 1x12 (divided cache)
  - 2 MPI jobs per node, i.e. 1 MPI job per processor: 2x6
- Used *4x100,000+10x1,000,000* Vegas events
- LO/NLO stopped scaling above 50/100 cores → Memory dominated regime.
- 1x12 runs slower than 2x6 because openMP does not have to sync cache between the 2 processors in the 2x6 case.



MCFM 8: Thread scaling (2)

- The NNLO scaling for all singlet processes included in MCFM 8.0 as a function of the number of MPI jobs
- Used *4x100,000+10x1,000,000* Vegas events
- Each MPI job is one processor with 6 cores
- Only the *PP→H* shows the onset of non-scaling at 48 MPI jobs.
- All other processes can be speed up efficiently using a larger cluster



MCFM 8: Timing

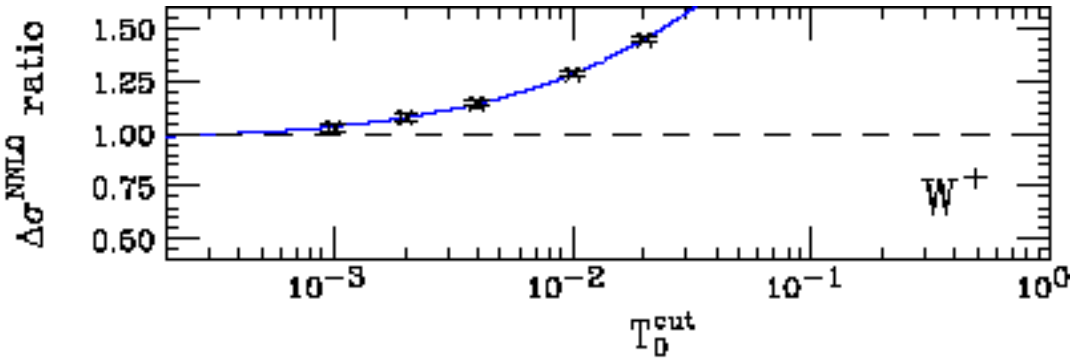
$\mathcal{T}_0^{cut}$	W^+	Z	H	HW^+	HZ	$\gamma\gamma$
0.001	2% (1397)	0.9% (2770)	0.05% (1256)	10% (1263)	6% (1939)	0.4% (3706)
0.005	0.7% (1358)	0.4% (2701)	0.04% (1234)	3% (1238)	2% (1906)	0.2% (3661)
0.01	0.5% (1356)	0.2% (2677)	0.04% (1214)	2% (1222)	1% (1847)	0.15% (3585)
0.05	0.2% (1315)	0.08% (2572)	0.04% (1197)	0.6% (1206)	0.4% (1841)	0.09% (3492)
0.1	0.09% (1307)	0.05% (2526)	0.04% (1186)	0.3% (1186)	0.2% (1847)	0.08% (3427)
0.5	0.04% (1266)	0.04% (2356)	0.04% (1176)	0.1% (1150)	0.09% (1768)	0.07% (3376)

Table 10. The relative statistical precision (in percentages) on the $pp \to W^+ \to l^+\nu$, $pp \to Z \to l^+l^-$, $pp \to H \to \gamma\gamma$, $pp \to H + W^+ \to \gamma\gamma + l^+\nu$ $pp \to H + Z \to \gamma\gamma + l^+l^-$ and $pp \to \gamma\gamma$ cross sections at NNLO as a function of $\mathcal{T}_0^{cut}$ (in GeV) using $4 \times 2 \times 6$ cores. Also given in brackets is the evaluation time (in seconds).

- Note the runtime is 48 cores only using *4x100,000+10x1,000,000* Vegas events.
- As can be seen the value of the slicing cut greatly affects the achieved statistical precision.
- The choice of the value of the jettiness slicing cut is important
  - Large: the power correction contributes (incomplete cancelation)
  - Small: the statistical error is large

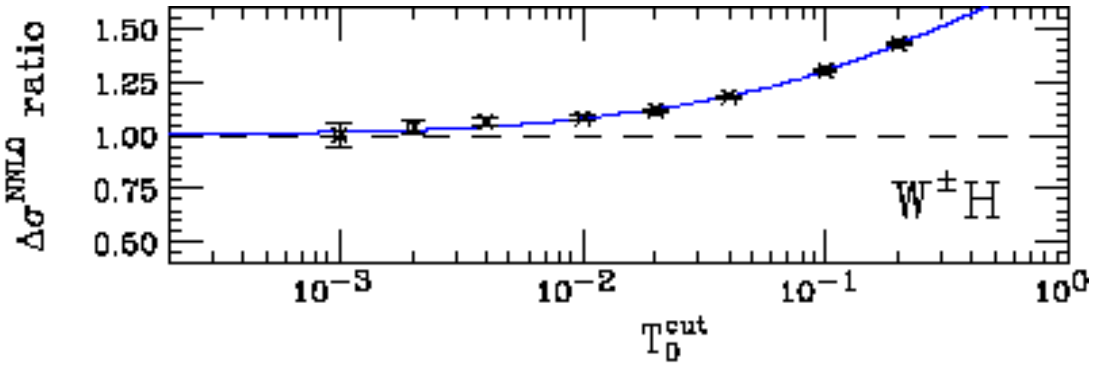
MCFM 8: What to do with the slicing cut T^{cut}

- MCFM calculates the NNLO cross section for a given value of the slicing cut.
- The physical cross section is given at value zero for the slicing cut.
- This zero value extrapolation can be done in different ways.
- This requires some considerations to avoid pitfalls



Compared to ZWMS

Campbell, Ellis, Williams, 1601.00658



Compared to vh@nnlo

MCFM 8: choosing a small value $\mathcal{T}^{cut}$

- I general you want to choose slicing cut such that the statistical MC integration uncertainty is larger.
- MCFM has two preset values for the slicing cut: The 1.0% and 0.2% power correction uncertainty.
- As an example what runtime we need for the statistical error to be smaller than this we look at *pp →W* using ~20 min of run time on 48 cores using *4x100,000+10x1,000,000* Vegas events
  - 1.0%: cut=0.005 → 0.7% stat error
  - 0.2%: cut=0.001 → 2.0% stat error
- In the 0.2% case we need a factor of 10x more statistics or 100x more events by increasing number of cores and/or events to match the systematic uncertainty

Process		σ^{NNLO}	
		1% accuracy	0.2% accuracy
$gg \to H$	inclusive	0.03	0.002
Z	inclusive	0.01	0.002
	lep. cuts	0.07	0.005
W^+	inclusive	0.005	0.001
	lep. cuts	0.03	0.003
ZH	inclusive	0.3	0.02
	lep. cuts	0.8	0.04
$W^{\pm}H$	inclusive	0.2	0.01
	lep. cuts	0.8	0.08
$\gamma\gamma$	cuts [25]	0.01	0.001

Table 11. Values of $\mathcal{T}_0^{cut}$ (in GeV) required to perform the NNLO jettiness calculation to a given accuracy, for the processes studied in this paper. At larger $\mathcal{T}_0^{cut}$ the accuracy deteriorates because of increased power corrections.

[25] Campbell, Ellis, Li, Williams, 1603.0663

MCFM 8: curve fitting

- The power corrections have a know functional form

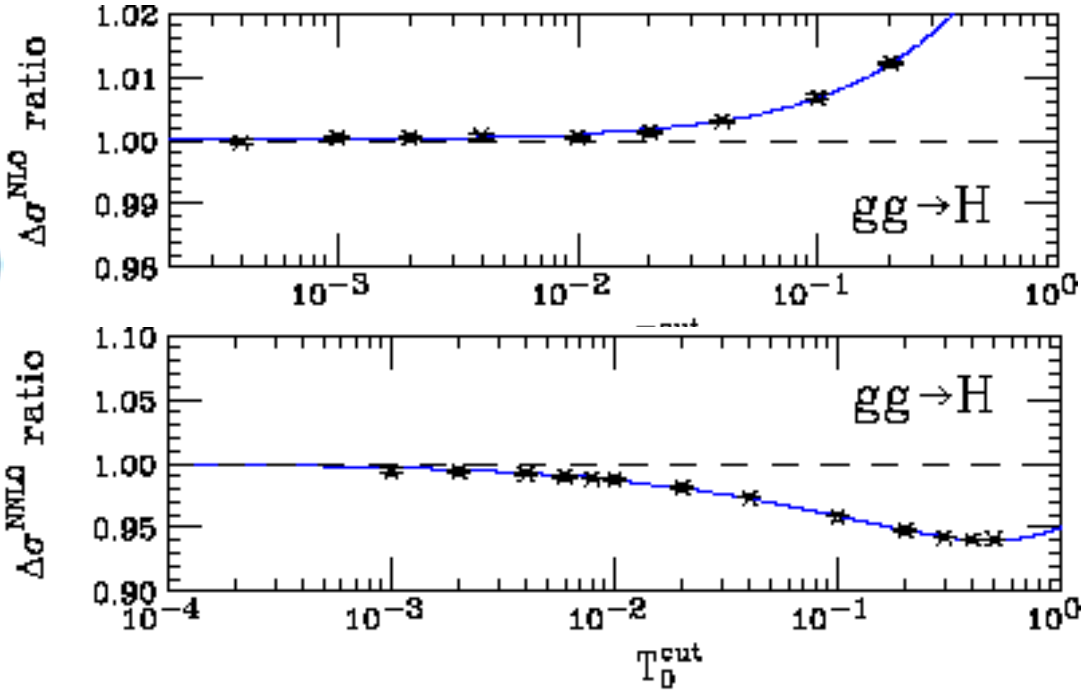
$$\Delta\sigma^{NLO}_{\text{jettiness}}(\mathcal{T}_0^{cut}) = \Delta\sigma^{NLO} + c \times \left(\frac{\mathcal{T}_0^{cut}}{Q}\right) \times \log\left(\frac{\mathcal{T}_0^{cut}}{Q}\right) \quad \text{(+ polynomial)}$$

$$\Delta\sigma^{NNLO}_{\text{jettiness}}(\mathcal{T}_0^{cut}) = \Delta\sigma^{NNLO} + c_3 \times \left(\frac{\mathcal{T}_0^{cut}}{Q}\right) \times \log^3\left(\frac{\mathcal{T}_0^{cut}}{Q}\right) + c_2 \times \left(\frac{\mathcal{T}_0^{cut}}{Q}\right) \times \log^2\left(\frac{\mathcal{T}_0^{cut}}{Q}\right)$$

N-jettiness Subtractions for NNLO QCD Calculations,
J. Gaunt, M. Stahlhofen, F.J. Tackmann and J.R. Walsh, 2015

- MCFM is designed to produce the (differential) cross section for a given slicing cut
- One can run for a handful of cut values and fit the parameter to find the extrapolation to zero.
- This can be fully automated in e.g. root or Mathematica with a per bin fit including the statistical error
- The fit does not only give the parameters but also goodness of fit, indicating if more terms in the expansion are required.

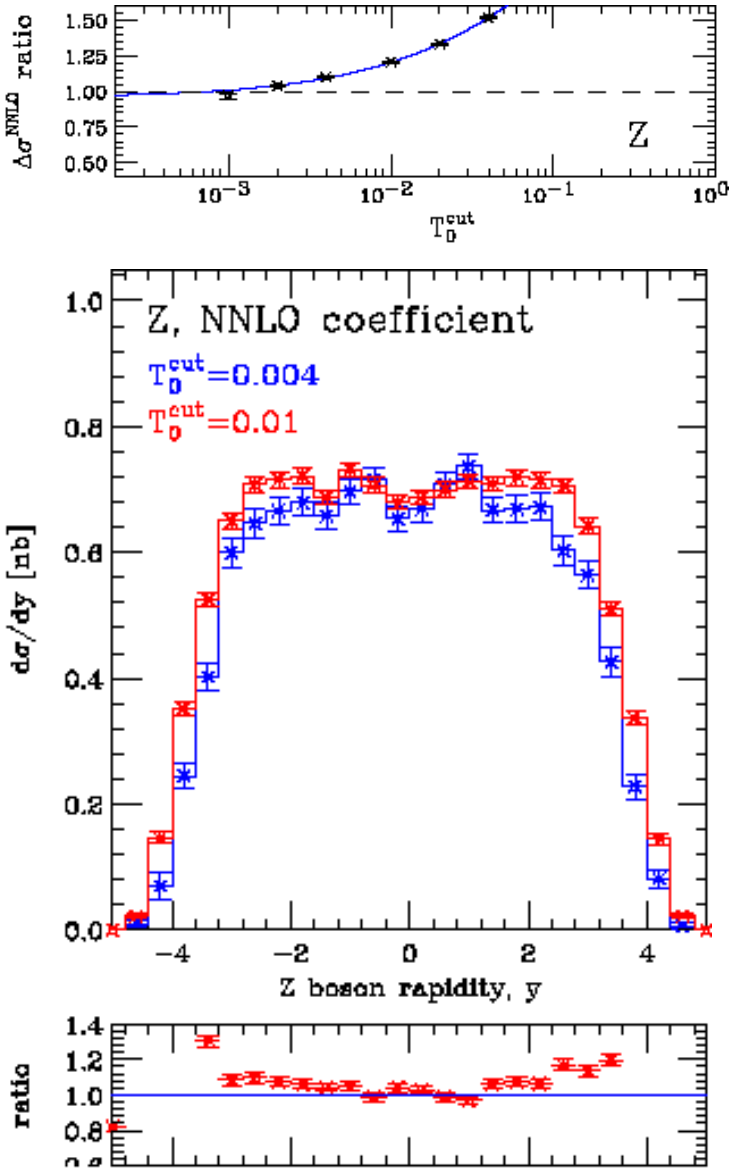
Compared to ggh@nnlo



NLO : $c_0 = 1.000$; $c_1 = -1.17$
NNLO: $c_0 = 0.998$; $c_3 = 0.324$; $c_2 = 1.30$

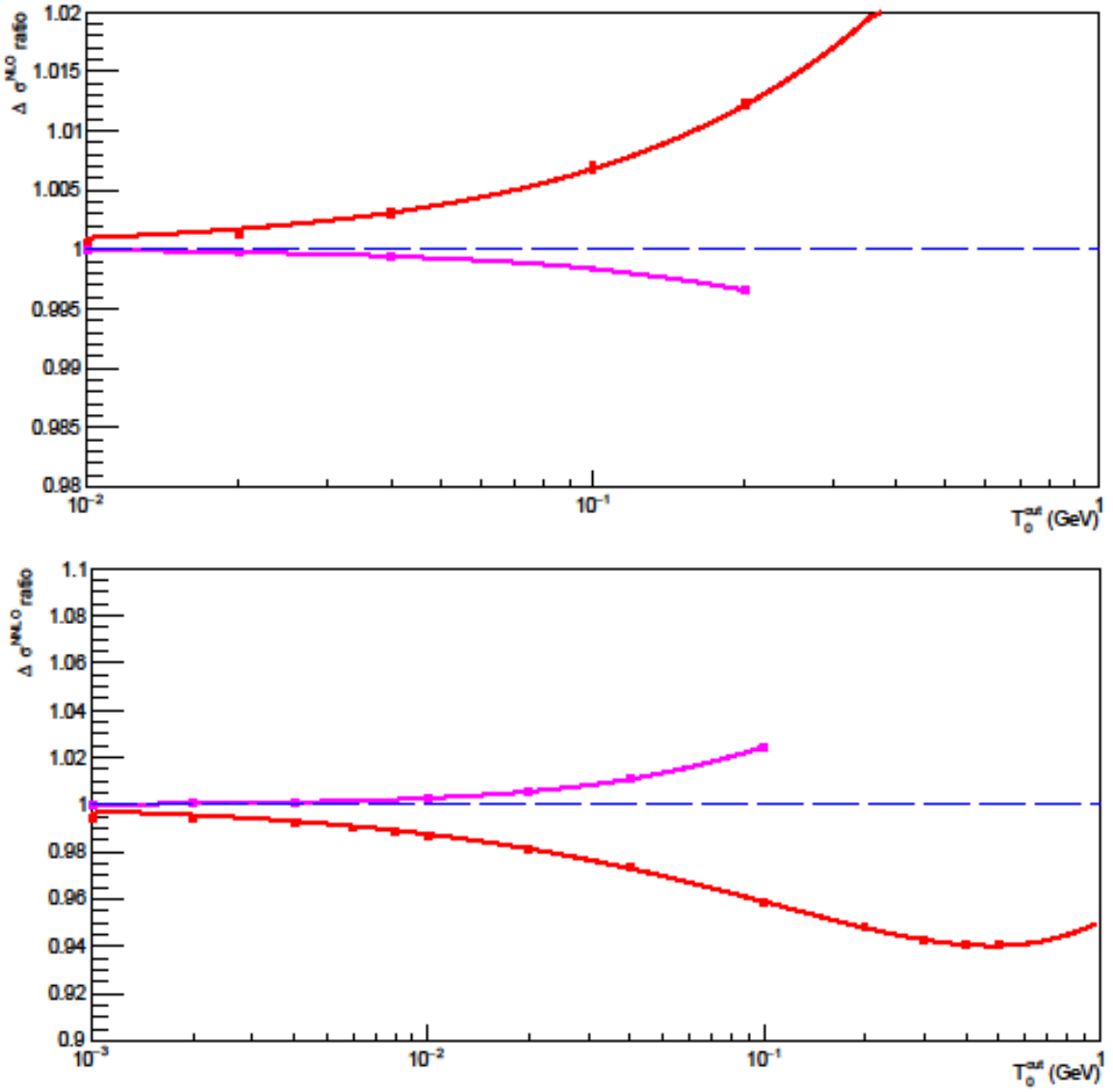
MCFM 8: Example distribution

- A $T^{cut} = 0.004\ GeV$ should be enough to have the statistical uncertainty dominate over the systematic uncertainty
- Though one might have doubts in the forward region.
- Better would be do add to more runs at e.g. $T^{cut} = 0.05\ GeV$ and $0.1\ GeV$ and do a 2-parameter (or 3-parameter depending on goodness of fit) fit using the 4 results bin-by bin. This will give the value at $T^{cut} = 0\ GeV$.



Improving slicing

- The leading power correction can be calculated and added to MCFM
  R. Boughezal, X. Liu, F. Petriello, 2016
- A preliminary study has been performed for *pp* →*H*.
- This will be included in a future release of MCFM, allowing one to choose much larger values of the cut.
- This holds great promise, especially if one can calculate more and more of the coefficients.



MCFM 8: Back to the Desktop

Thread-Scalable Evaluation of Multi-Jet Observables, W. Giele, G. Stavenga, J. Winter, 2010

- Use a desktop with a multi-core processor and a Nvidia GPU.
- The most time consuming part on NNLO is the double bremsstrahlung tree level evaluation
- One can program a GPU to do tree level recursion relations!
- The speedup times in the table are on a GPU several generations out
- Expect an order of magnitude more gain on a modern GPU (from 0.66 Teraflops → 5.5 Teraflops for DP).

n	$T_n^{\rm GPU}$ (seconds)	$P_n(3)$	$T_n^{\rm CPU}$ (seconds)	$P_n(4)$	G_n
4	2.975×10^{-8}		8.753×10^{-6}		294
5	4.438×10^{-8}	0.91	1.247×10^{-5}	0.87	281
6	8.551×10^{-8}	1.03	1.966×10^{-5}	0.93	230
7	2.304×10^{-7}	1.19	3.047×10^{-5}	0.96	132
8	3.546×10^{-7}	1.01	4.736×10^{-5}	0.98	133
9	4.274×10^{-7}	0.94	7.263×10^{-5}	0.99	170
10	6.817×10^{-7}	1.05	1.044×10^{-4}	0.99	153
11	9.750×10^{-7}	1.02	1.529×10^{-4}	1.00	157
12	1.356×10^{-6}	1.02	2.129×10^{-4}	1.00	158

Table 2: The GPU and CPU evaluation times per event, $T_n^{\rm GPU}$ and $T_n^{\rm CPU}$, given as a function of the number n of gluons for $gg \to (n-2)\,g$ processes. The polynomial scaling measures are also shown, for the GPU, $P_n(3)$, and for the CPU, $P_n(4)$. The $P_n(m)$ are defined as $P_n(m) = [(n-1)/n] \times \sqrt[m]{T_n/T_{n-1}}$. The rightmost column finally displays the gain $G = T_n^{\rm CPU}/T_n^{\rm GPU}$.

MCFM 8.0: Here to stay

GPU's are here to stay:
Virtual reality, Deep learning & self-driving cars

NVIDIA DRIVE PX 2

12 CPU cores | Pascal GPU | 8 TFLOPS | 24 DL TOPS | 16nm FF | 250W | Liquid Cooled

World's First AI Supercomputer for Self-Driving Cars

	"Fermi"	"Fermi"	"Kepler"	"Kepler"	"Maxwell"	"Pascal"
Tesla GPU	**GF100**	**GF104**	**GK104**	**GK110**	**GM200**	**GP100**
Compute Capability	2.0	2.1	3.0	3.5	5.3	6.0
Streaming Multiprocessors (SMs)	16	16	8	15	24	56
FP32 CUDA Cores / SM	32	32	192	192	128	64
FP32 CUDA Cores	512	512	1536	2880	3072	3584
FP64 Units	-	-	512	960	96	1792
Threads / Warp	32	32	32	32	32	32
Max Warps / Multiprocessor	48	48	64	64	64	64
Max Threads / Multiprocessor	1536	1536	2048	2048	2048	2048
Max Thread Blocks / Multiprocessor	8	8	16	16	32	32
32-bit Registers / Multiprocessor	32768	32768	65536	65536	65536	65536
Max Registers / Thread	63	63	63	255	255	255
Max Threads / Thread Block	1024	1024	1024	1024	1024	1024
Shared Memory Size Configurations						
			32 KB	32 KB		
			48 KB	48 KB		
Hyper-Q	No	No	No	Yes	Yes	Yes
Dynamic Parallelism	No	No	No	Yes	Yes	Yes
Unified Memory	No	No	No	No	No	Yes
Pre-Emption	No	No	No	No	No	Yes

Conclusions

- We extended MCFM to provide a public code for NNLO processes using jettiness slicing (happy hacking…)
- Using a hybrid openMP/MPI version we made MCFM fast enough to run on a small to moderate cluster
- Currently included *H, W, Z, WH, ZH* and *di-photon* processes
- Work is/will/should be going on to extend MCFM 8.0
  - More color singlet processes will be added at NNLO
  - Include the leading power corrections into the jettiness slicing functions.
  - Novel phase space methods to improve phase space integration
  - Inclusion of color singlet + jet processes (H, Z, W +jet) at NNLO
  - Matching MCFM to showers at both NLO and NNLO.
  - Using GPU's to achieve NNLO on desktops.

LOOPFEST XV

August 15-17 2016
Buffalo NY

Radiative corrections for the LHC and Future Colliders

Organizing Committee

Sally Dawson — Lance Dixon

Frank Petriello — Doreen Wackeroth

Local Organizing Committee

Tobias Neumann — Simone Marzani

Vincent Theeuwes — Ciaran Williams

https://indico.fnal.gov/LoopFestXV

Support provided by the UB Physics Department
and the College of Arts and Sciences

University at Buffalo
The State University of New York