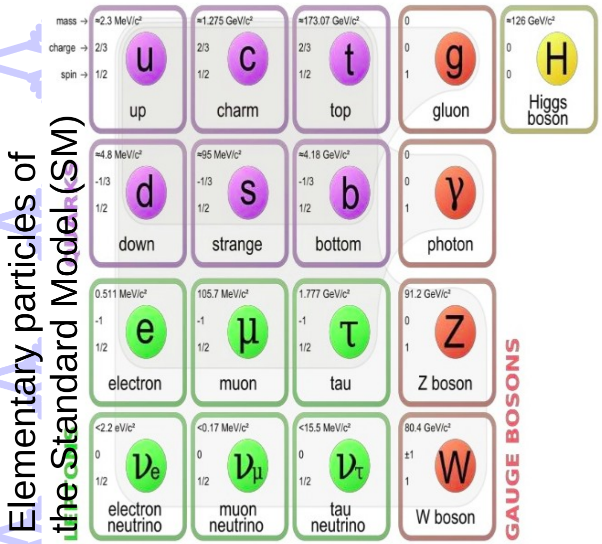
A MPI/OpenCL hybrid implementation of the Matrix Element Method in the context of the Higgs boson property analyses

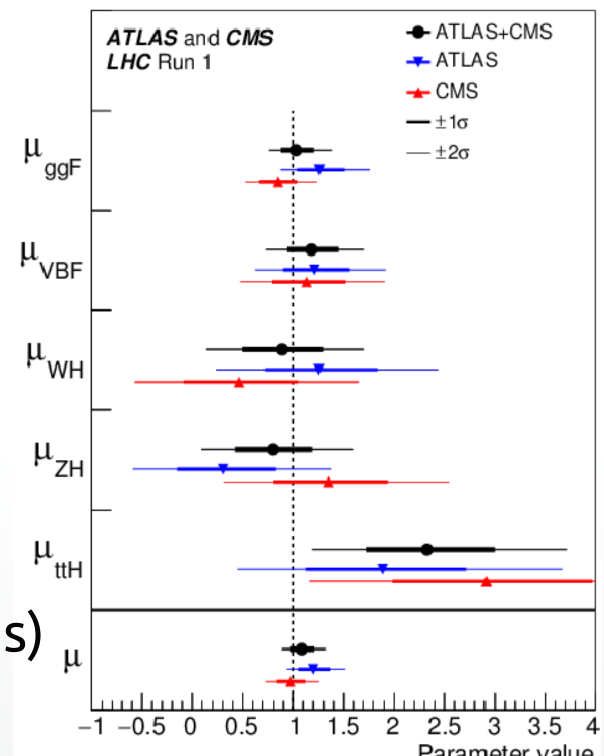
G. Grasseau, T. Strebler, A. Chiron,
P. Paganini and F. Beaudette,

Laboratoire Leprince-Ringuet
CNRS/IN2P3, École Polytechnique

Introduction

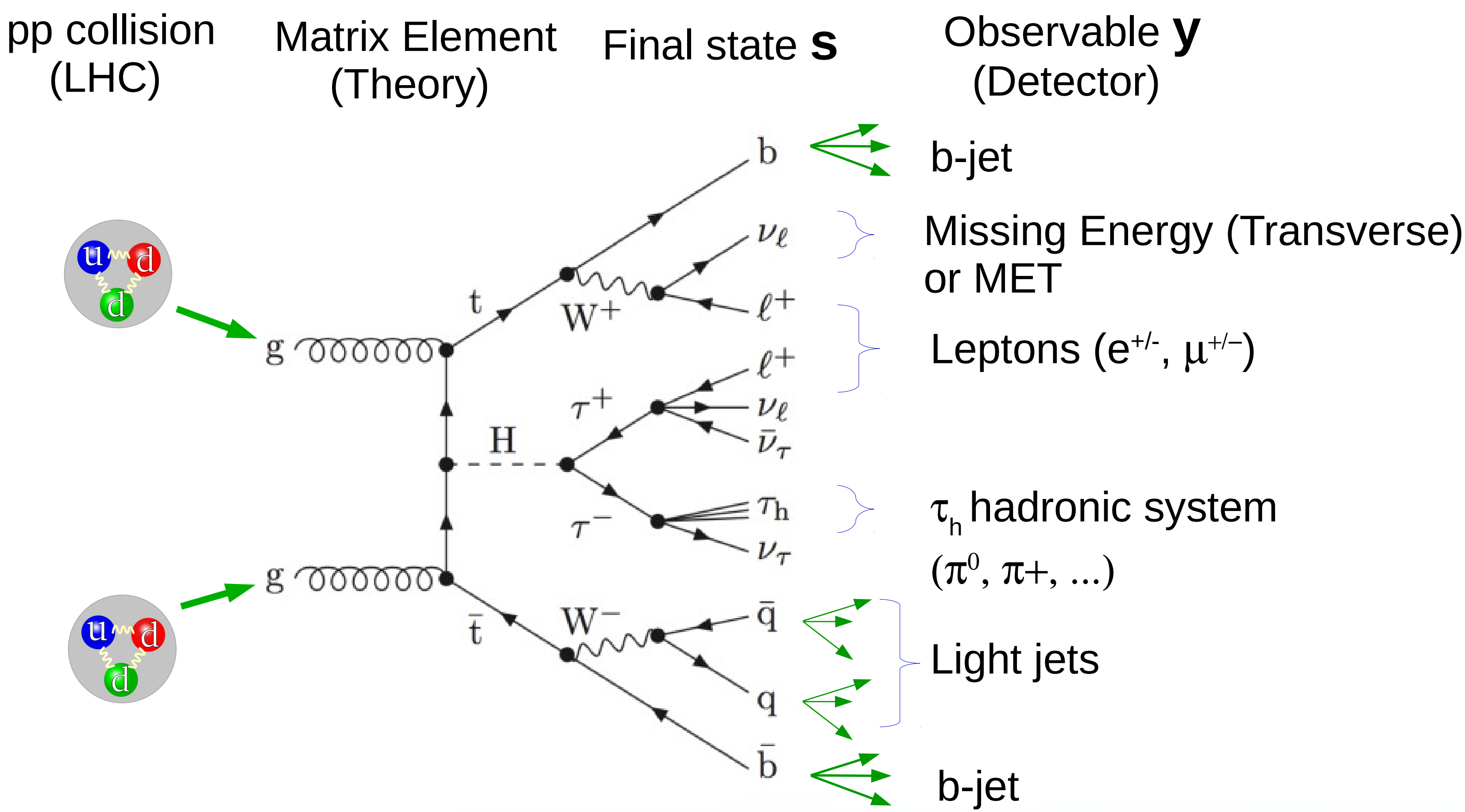


- The recently discovered Higgs boson (2012) can be produced in different ways in pp collisions @LHC
- The combination of LHC experiments shows an excess of events when the H is produced in association with 2 top quarks (ttH channel)



- ttH is an interesting channel to look at Run 2: it allows probing the top-Higgs Yukawa coupling
- In addition, it has several decay channels
- Among all the ttH channels looked by CMS, the H decays in 2 ττ (H->ττ) is one of the most challenging
- LLR team is deeply involved in Matrix Element Method : VBF, ttH channel (T. Strebler PHD thesis)

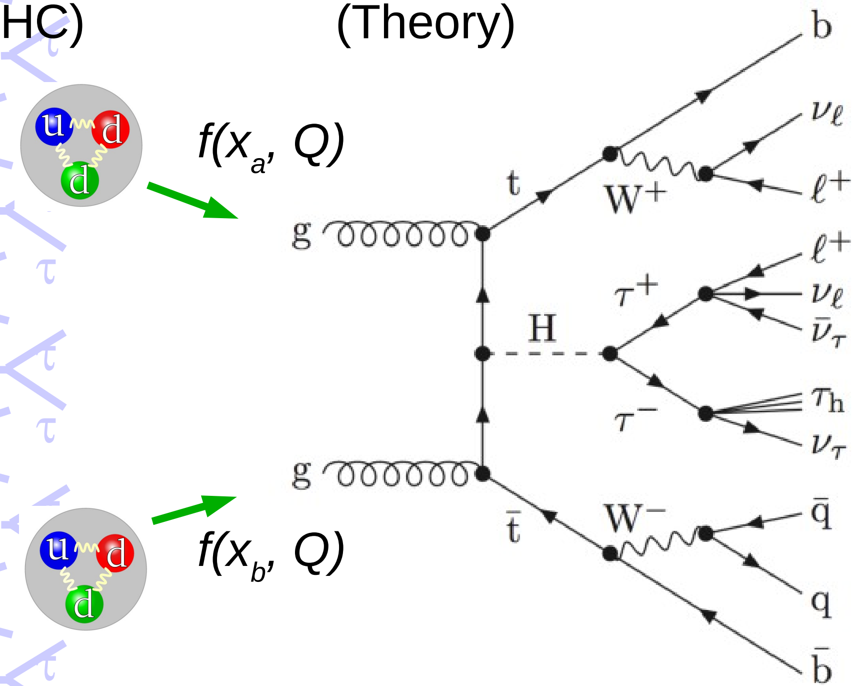
Theory and observables



- Leptons $\ell^{+/-}$, hadronic system τ_h, are precisely reconstructed
- Jet energy reconstructed with a finite resolution
- ν's are unobserved but their global (transverse) momentum can be inferred from the MET

Matrix Element Method (MEM)

pp collision (LHC) | Matrix Element (Theory) | Final states **s**

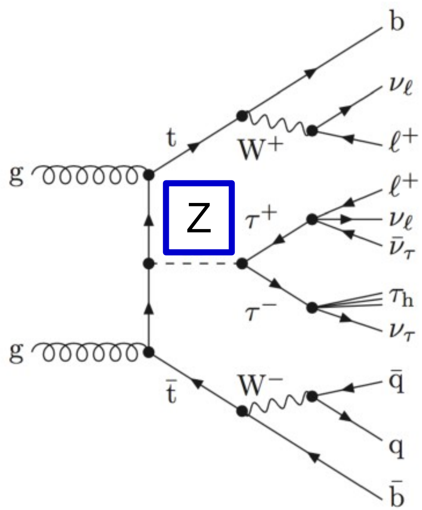


- Studying ttH with the final state ***s*** : *2b, 2q, τ_h, 2 leptons*
- Observable **y** lead to several possible states ***s*** which are of interest (signal) or not (background)
- MEM: explore (give a weight) to all the possible value of **x** which lead to the observable **y**

$$w_i(\mathbf{y}) = \frac{1}{\sigma_i}\sum_p \int d\mathbf{x}\underbrace{dx_a dx_b}_{x_a,\,x_b\ \text{parton impulsion fraction}}\underbrace{\frac{f(x_a,Q)f(x_b,Q)}{x_a x_b s}}_{f\ \text{Parton Density Fraction (PDF)}}\underbrace{\delta^2\left(x_aP_a + x_bP_b - \sum p_k\right)}_{\text{Momentum conservation}}\underbrace{|\mathcal{M}_i(\mathbf{x})|^2}_{\text{ME}}\underbrace{W(\mathbf{y}||\mathbf{x})}_{\text{Transfer Function (TF)}}$$

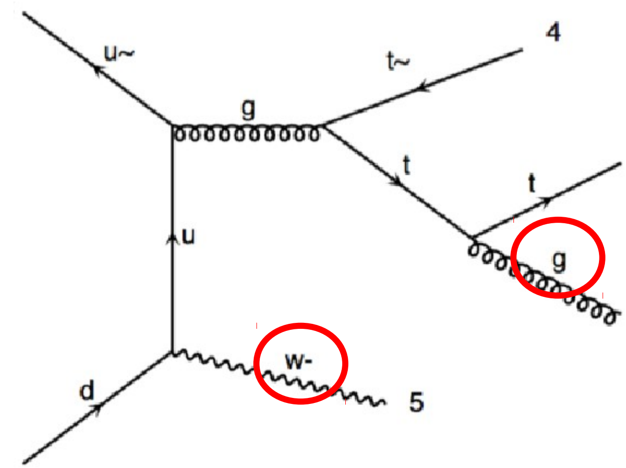
TF: detector response

Final states: backgrounds



ttZ: Z production (decays in $\tau\tau_h$)

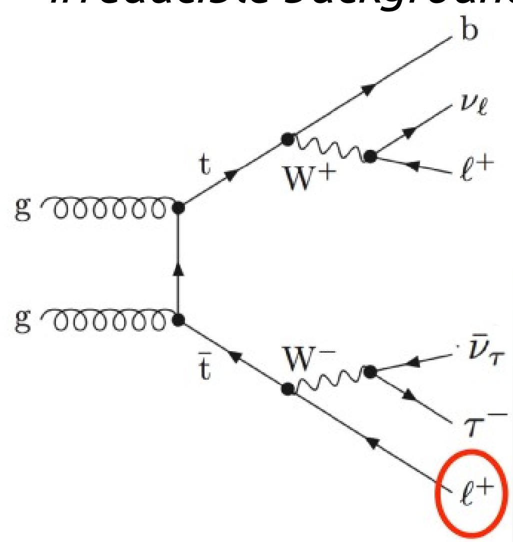
irreducible background



ttW: g misidentified as τ_h

with W → lepton …

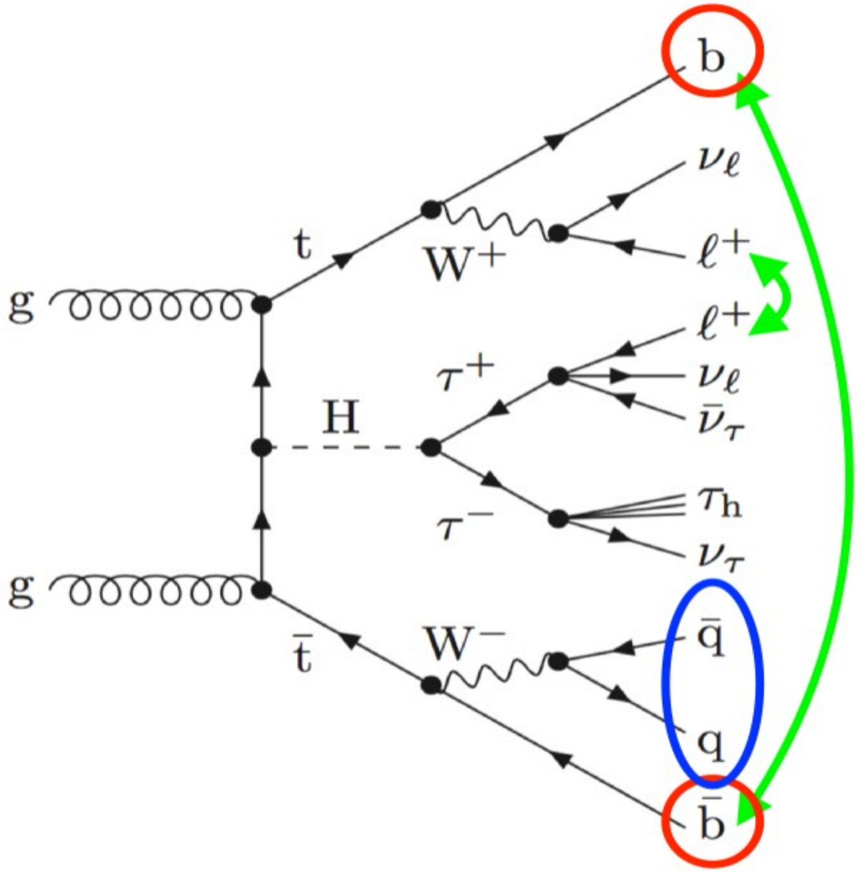
and others possibilities ...



tt+jets: others jets (gluons) can be present in the event (Initial State Radiation)

The definition of the final state drives the S/B

Permutations



- Problem to associate the b-jets measures to the $(b,\bar{b})$ of final state, idem for the 2 leptons
- 4 permutations (green arrows)
- 1 missing q or $\bar{q}$ in the reconstruction:

  → 2 more integration variables (direction)
  → (4 x) permutations on all possible “light jets”

Integration space **dimension**	ttH, H→ ττ	ttZ, Z→ ττ	ttW, W→ lν	tt+jets
no missing jet	**5**	**5**	**6**	**4**
with missing jets	**7**	**7**	**8**	**6**

MEM MPI implementation

- PDF: LHAPDF library
- ME computation: MadGraph5 2.2.1 *code generator* (C++)
- ROOT: I/O, Lorentz/geometric arithmetics
- Integration: VEGAS algorithm (GSL)
- Mean CPU time per event 13 min.
- Parallel version (MPI) to tune the analysis method (T. Strebler)
- One run takes several days on 200-400 physical cores

"Daily used" of MEM-MPI on 400-core platform

OpenCL Implementation

Requirements:

Aggregate all the computing powers of the ≠ nodes (MPI + OCL)

Benefit of all device computing power, including CPUs
→ several OCL queues in a node

VEGAS: keep the computation of the chi-square (GSL)

Features:

- Minimize host/devices communications:
  - 1 event is assigned to a queue/device
  - All the integration part (VEGAS) must be done inside the device (including reductions)
- No blocking calls (kernels, communications) → OCL events
- Minimize the synchronization points (reductions)

OCL Kernels

Main kernel (one Vegas iteration) :

- We developed MadGraph extension to generate the OCL kernel codes
- LHAPDF lib.: Fortran to C-kernel translation
- ROOT tools: Lorentz/geometric arithmetics

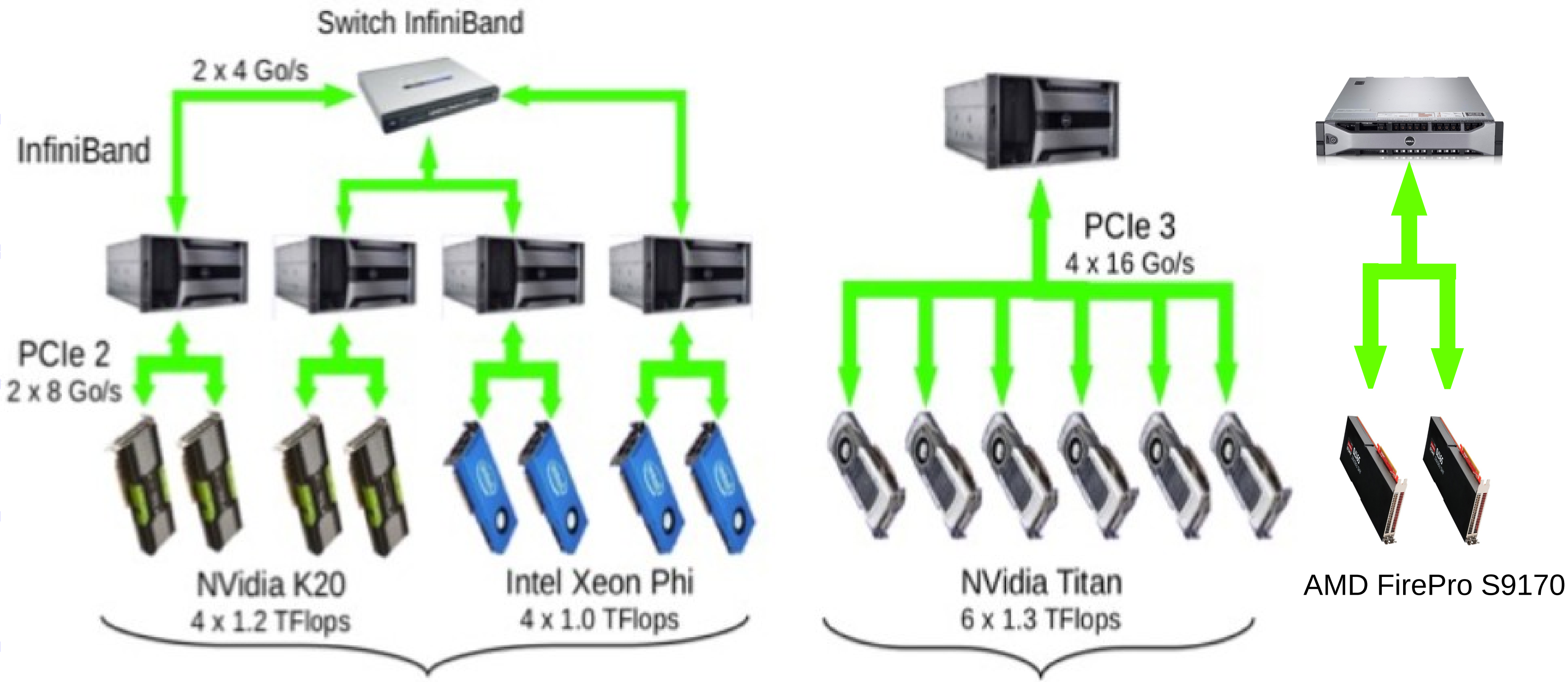
→ big kernels (10-20 x 10^3 lines)

Data/Work flow

```
host→devices( Config )
 Loop on Events
  host→devices( Event )
  Loop on Permutations
   Loop on IntegrationTypes
    host→devices( VegasState )
    kVegasSetUp()
    Loop on χ^2 // for Vegas
     kVegasCompute()
     kVegasFinalize()
    Devices->host(VegasState)
```

- `Config`: LHAPDF, MagGraph (sub) processes, etc.
- `Event`: coming from MPI msg → device/queue
- `IntegrationTypes` loop: asynchronous mode non-blocking calls (cl::Events)

Benchmarking Platform



Each node

2 x Intel E5-2650: 2 GHz,**16** physical cores, with AVX (4 doubles), 64 Go memory

Interconnection

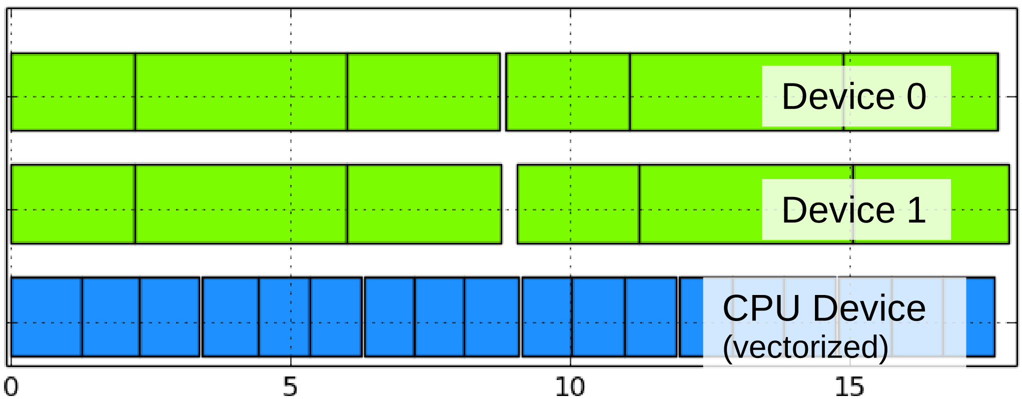
switch InfiniBand

Devices

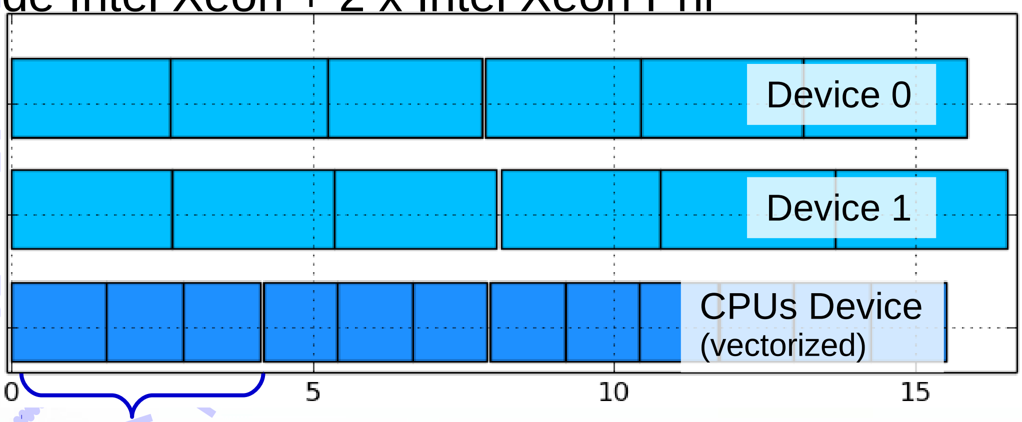
NVidia K20, Titans
Xeon Phi
AMD FirePro S9170

Load-balancing in a single node

Node Intel Xeon + 2 x NVidia K20



Node Intel Xeon + 2 x Intel Xeon Phi



1 event: 3 possible permutations → 3 x ttH integrations

Same event, 3 permutations, ttH (signal hypothesis)

- NVidia specificities:
  - Buffer must be “pinned” in the memory not to block the copy call
- In OCL for NVidia GPUs:

created with CL_MEM_ALLOC_HOST_PTR
allocated enqueueMapBuffer()

The NodeScheduler feeds all the Devices/CPUs inside one node

Preliminary Performance on a single device

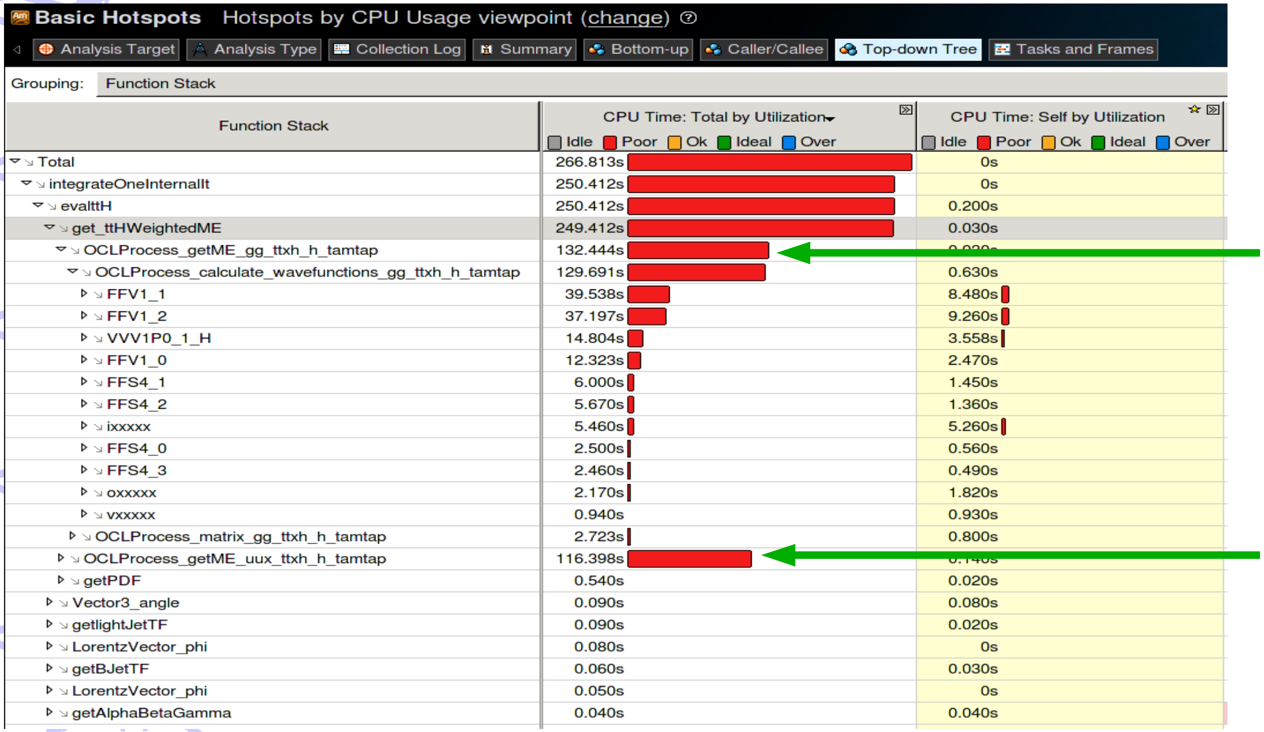
One event, 3 permutations, ttH with 15552 integration points

	C++ -O3	OCL K20	OCL X.Phi	OCL CPUs	OCL AMD
Time (s)	91.6	8.74	6.90	3.16	-
Speedup	1.00	10.74	13.3	29.0	-
Speedup with 16 MPI proc.	1.00	0.66	0.83	1.81	-

We obtained better performance on smaller kernels (simplified ME, speedup > 50 on K20)

How to get performance analysis of kernels with OCL ?

Performance analysis tools



Basic Hotspots Hotspots by CPU Usage viewpoint (change)

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Tasks and Frames

Grouping: Function Stack

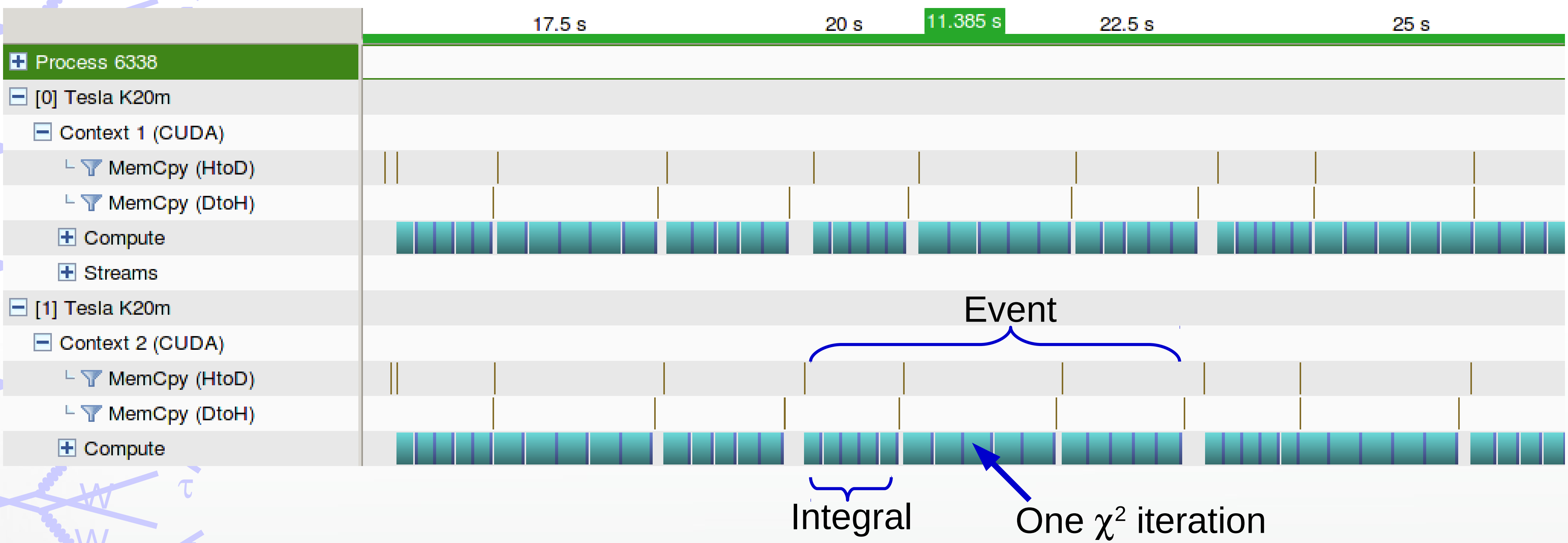
Function Stack	CPU Time: Total by Utilization (Idle, Poor, Ok, Ideal, Over)	CPU Time: Self by Utilization (Idle, Poor, Ok, Ideal, Over)
Total	266.813s	0s
integrateOneInternalIt	250.412s	0s
evalttH	250.412s	0.200s
get_ttHWeightedME	249.412s	0.030s
OCLProcess_getME_gg_ttxh_h_tamtap	132.444s	[illegible]
OCLProcess_calculate_wavefunctions_gg_ttxh_h_tamtap	129.691s	0.630s
FFV1_1	39.538s	8.480s
FFV1_2	37.197s	9.260s
VVV1P0_1_H	14.804s	3.558s
FFV1_0	12.323s	2.470s
FFS4_1	6.000s	1.450s
FFS4_2	5.670s	1.360s
ixxxxx	5.460s	5.260s
FFS4_0	2.500s	0.560s
FFS4_3	2.460s	0.490s
oxxxxx	2.170s	1.820s
vxxxxx	0.940s	0.930s
OCLProcess_matrix_gg_ttxh_h_tamtap	2.723s	0.800s
OCLProcess_getME_uux_ttxh_h_tamtap	116.398s	[illegible]
getPDF	0.540s	0.020s
Vector3_angle	0.090s	0.080s
getlightJetTF	0.090s	0.020s
LorentzVector_phi	0.080s	0s
getBJetTF	0.060s	0.030s
LorentzVector_phi	0.050s	0s
getAlphaBetaGamma	0.040s	0.040s

VTune *analysis on kernels (w/o OCL)*: workload dominated by the ME computation (green arrows)

- CodeXL (AMD) works well for simple kernels (compiler) …
- NVVP (NVidia) not allowed with OCL …
- VTune (Intel) with OCL (CPU) … difficult

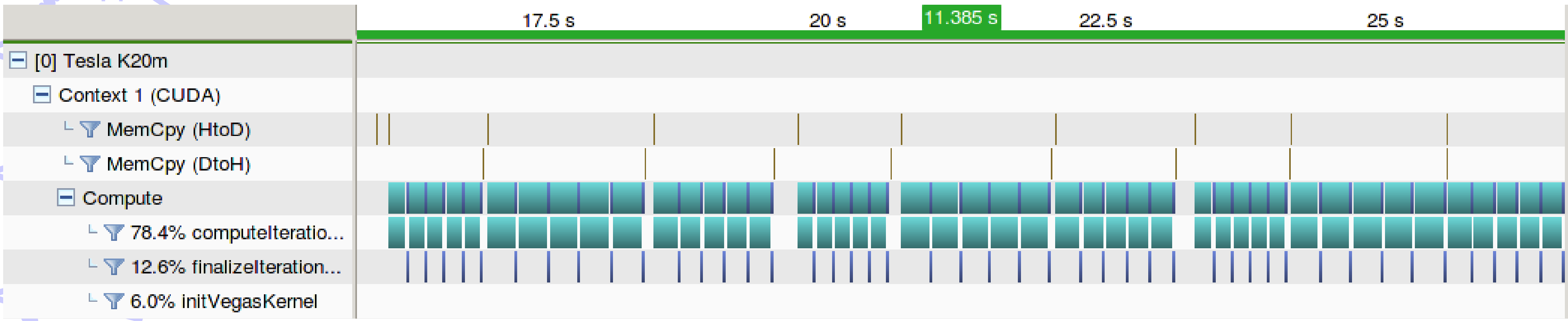
CL-CUDA/cl.hpp

- LLR development, motivations: for debugging, to preserve our OCL developments, ...
- Principles : routes <cl.hpp> calls/methods to CUDA calls. Handle heterogeneous devices
- Change: `#include <CL/cl.hpp>` by `#include <CL-CUDA/cl.hpp>` and `-lcuda`



Kernel performance

- Good device occupancy (asynch. mechanisms)
- Host ↔ Devices copies are negligible
- Kernel performance: ~2 x faster with CL-CUDA

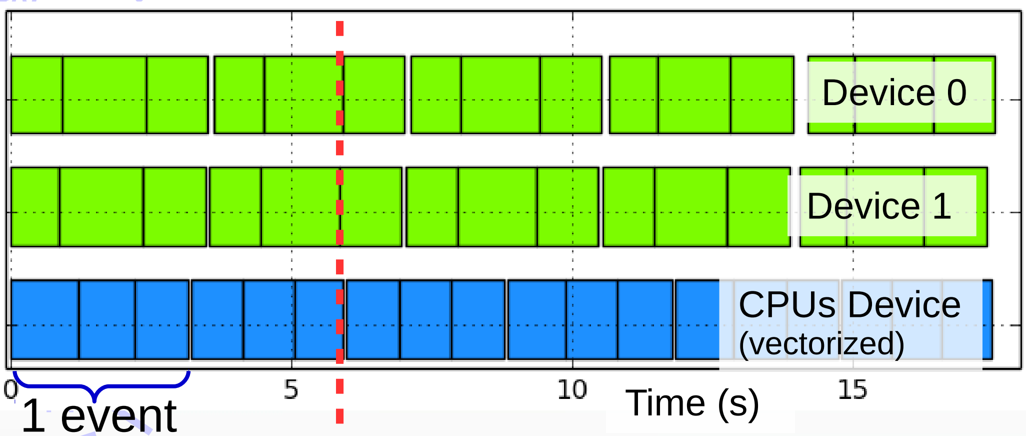


- Kernel performance is limited by the use of 255 registers per threads
- `--maxrregcount` doesn't improve performance
- VTune targets MadGraph expressions
- Better use of __local (__shared__) space memory to avoid register spilling and/or to reduce register use by thread

Conclusion/Perspectives

- CL-CUDA takes advantage of both CUDA/Intel-OCL compilers, speedup ~ 5.5 for one K20's node (speedup ~ 90 compared with a single MPI process)
- Optimization: better use of data locality (__local)

Node Intel Xeon + 2 x NVidia K20



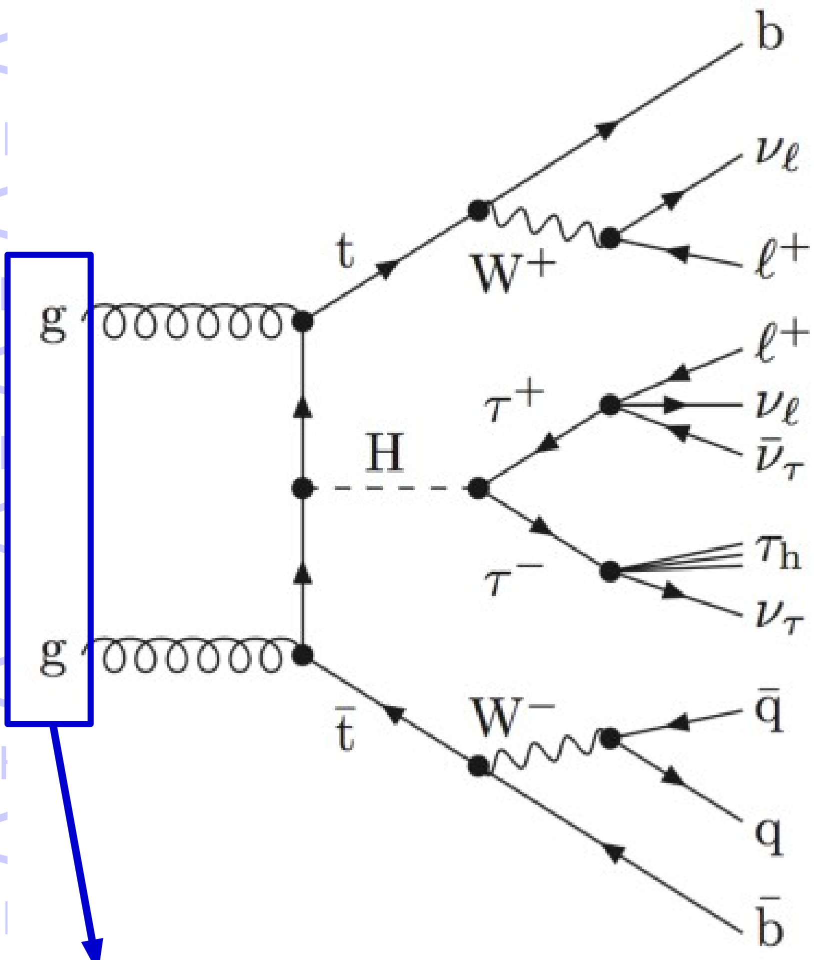
Time per ev. with 16 MPI processes

Next steps:

- Physics: include ttW, tt+jets in the next weeks
- Production on 10 nodes x 2 K80s (CC-IN2P3)
- Allows to compute more accurately integrals (dim. > 5, 15k points)
- Evaluate on recent platforms: NVidia Pascal, Intel KNL (GENCI)
- Evaluate OpenMP 4.x

Backup

Final States: signal

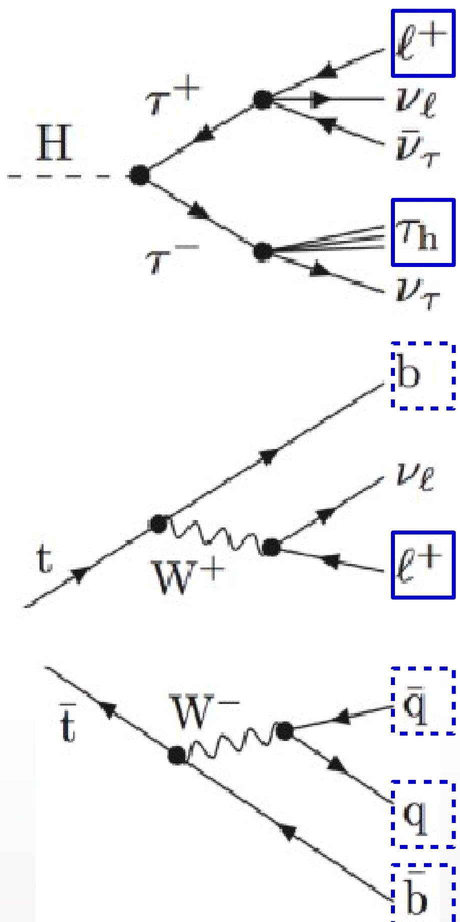


Others initial states ($u\bar{u}$, $d\bar{d}$, $c\bar{c}$, $s\bar{s}$) leading to the same final state (ttH, H$\tau\tau$)

- Final state chosen to optimize the “S/B” ratio:
  - 2 tops production (studied channel)
  - Higgs boson decaying in 2 τ's:
    - one τ decay into hadrons (hadronic system),
    - other τ decays in a lepton
  - Top quarks decays:
    - One decay in a single lepton+b(+neutrino)
    - One decays in quarks $q\bar{q}$ + b
  - And the 2 leptons with same sign

Integration variables for ttH/Z

ttH, H→$\tau\tau$: 3 x 11 variables with the measure constrains, the mass invariant constrains and the momentum conservation → 5 integration variables



- **Higgs/Z decay to $\tau\tau$**

  2 integration var.: $|\vec{\tau}^+|, \cos(\theta_{\tau\tau})$

- **Leptonic top decay**

  Setting ν direction ($\theta_{l\nu}$,$\eta_{l\nu}$) → E_ν → E_b

  2 integration var.: neutrino's direction ($\theta_{l\nu}$,$\eta_{l\nu}$)

- **Hadronic top decay**

  Setting E_q → E_{qbar} → E_{bbar}

  1 integration var.: E_q variable

Example: mass invariant for (W, q, q_{bar}):

$$m_W^2 = E_W^2 - \vec{P}_W^{\,2} = (E_q + E_{qbar})^2 - (\vec{P}_q + \vec{P}_{qbar})^2$$