GPU in High Energy Physics

Pisa, Italy

September 10-12, 2014

Hybrid implementation of the Vegas Monte-Carlo algorithm

G. Grasseau, S. Lisniak and D. Chamont
Leprince-Ringuet Laboratory (LLR),
Ecole polytechnique, Palaiseau
France

Motivations

- Needs from the LLR CMS team
- Cutting edge analyses techniques often imply *multi-dimensional integrations* that are computing intensive. In addition, the samples to analyze are huge (10^6 events).
  - The reconstruction of the mass of a tau-lepton pair (SVFit) requires ~1s/event while the typical time of the rest of the analysis chain is 1ms.
  - matrix-element methods (MEM) can reach 60s/event
- The High Luminosity will increase again the analysis time
- For an analysis team, the situation is difficult if the processing of all samples exceeds 2-3 weeks (elapsed time)
- We focus on the MC integration which is common numerical bottleneck within the LLR CMS group analyses

→ Provide a powerful implementation for Monte Carlo integrations for data analysis

$$I = \int_V d^n \boldsymbol{x} \;\; f(\boldsymbol{x})$$

VEGAS MC Integration

- The ROOT-based MC integration environment is popular with the CMS collaboration
- ROOT MC implementation is a GSL one
- GSL contains 3 MC algorithm : classical, miser, VEGAS

Reducing drastically the restitution time:

- Hardware : aggregate the computing power of nodes and accelerators (GPU, Xeon PhI, …)
- Software : hybrid model using MPI and OpenCL standards to glue the HW

The purpose of this talk is to present all the machinery to compute fast MC integral

VEGAS - Principles

VEGAS algorithm, G.P. Lepage, J. Comput. Phys. 27 (1978) 192

- **MC, M points $x \in V$**

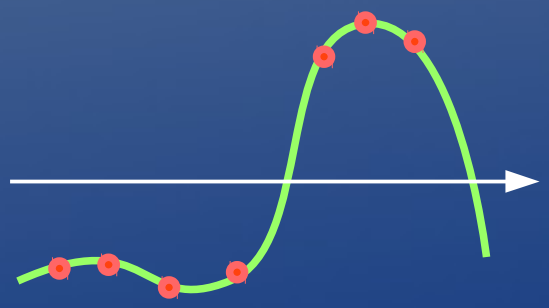
$$I \underset{M \to \infty}{\to} \frac{V}{M} \langle f(x) \rangle_p$$

with *p(x)* probability density

- **Importance sampling**

$$p(x) \propto |f(x)|$$

→ *minimize* σ^2

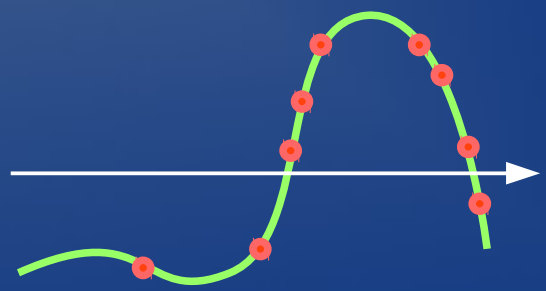


- **Stratified sampling**

V = U V_i *with* $i = 1 .. N$

$$\sigma_i^2 = \frac{\sigma^2}{N}$$

→ *concentrate sampling for high* σ^2 *i.e. p(x)* α *|df(x)/dx|*



Parallelism considerations

- Tried to use the APUs as CPUs → run all the program (like MPI, OpenMP)
  - Easy to do, avoid splitting the program
  - Avoid starting kernel latencies
  - Optimize the computing load
- Turned out to be not possible
  - Limit of the work size of 1024 → performance on NVidia – Intel OK
  - No simultaneous Kernels with NVidia OpenCL
  - Not possible to synchronize WorkGroups (1) - same in CUDA

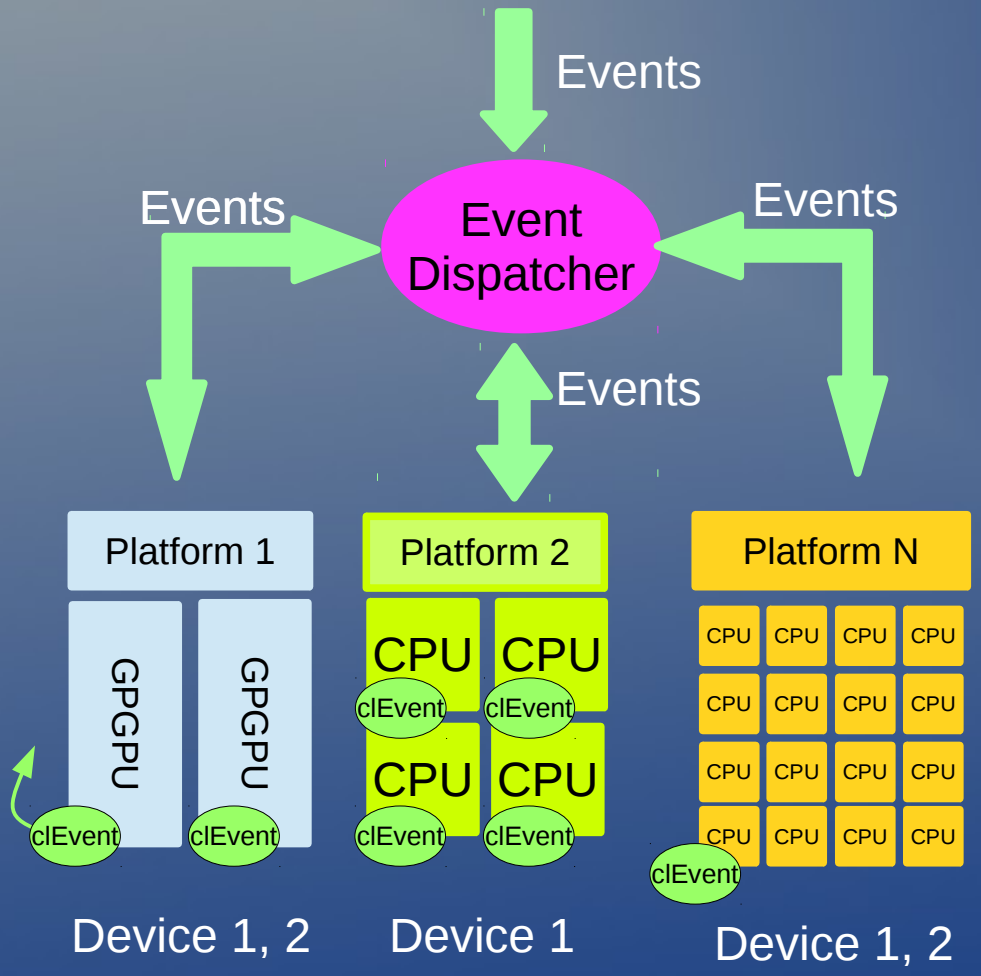
- Simplified numerical scheme

```
Loop <until convergence>
  Loop <internal>
W   Loop <N points x>
      x = rand()
      f(x) = F(x)
      update <f>,var,p(x)
    End Loop
    update I,σ²,χ²
  En Loop
En Loop
```

- Constrained to split !!!

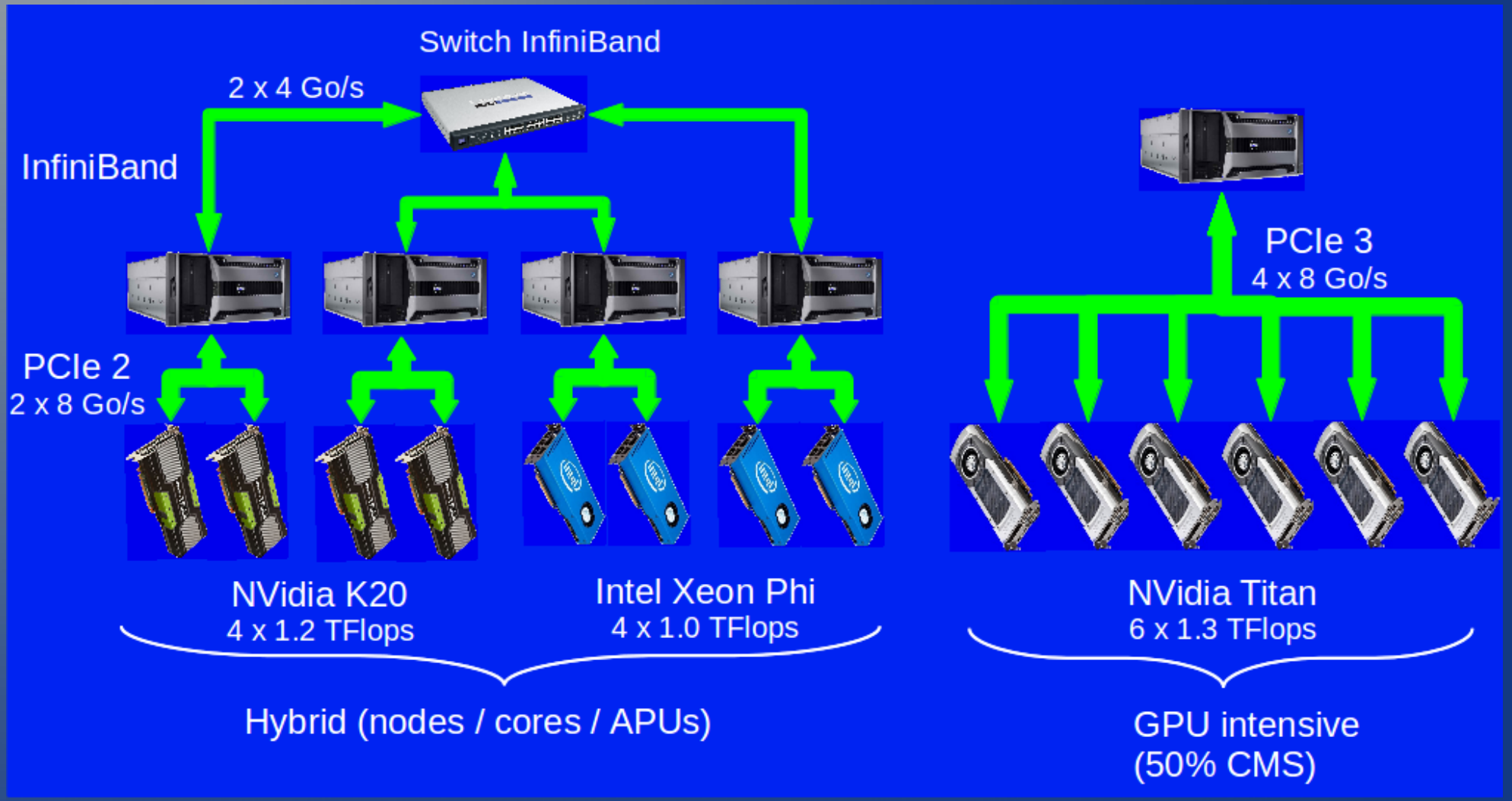
(1) thread blocks in CUDA

Node architecture



- At the node level (OpenCL)
- The platforms are feed by an Event Dispatcher
- Each Event to process (MC) is sent to a Device
- When a Device is idle a new event is sent to be processed
- OpenCL:
  - asynchronous processing write, read, running kernels
  - clEvent

Benchmark description (1)
GridCL Platform



Funded by the P2IO (french) project

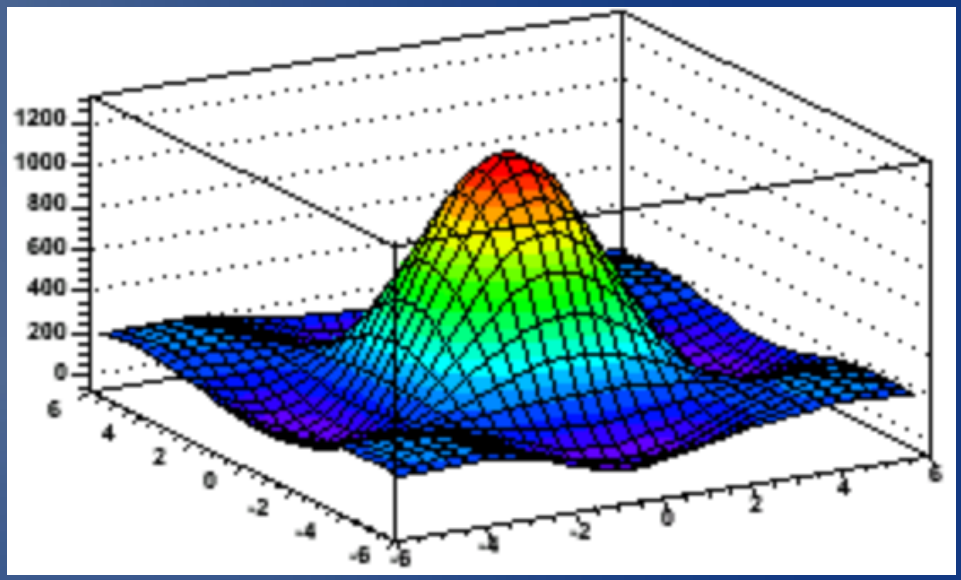
Benchmark description (2) Function to integrate

Function test to integrate

$$I = \int_0^{2\pi} \prod_i \frac{\sin(x_i)}{x_i} \, d^n x$$

- Special function known as sinus integral, n=5
- Number of points cst = 5x5x0.5x10^6
- Only 12 boxes in each direction → 2 points box
- Preliminary performance study → global landscape

Illustration, 2D case



Compilation	OpenCL
mpicc / icc -O3 icc v-13.0.1 OpenMPI 1.6.5	Intel 1.2 NVidia 1.1

http://root.cern.ch/root/.../TGraph2D.html

GG, SL, DC

Single node performances (1)

	GSL	GSL	OCL	OCL	OCL	OCL	OCL + 2xMPI
CPUs	1	32	32			32	
APUs				1	2	2	2
Time/Ev (s)	11.0	0.60	0.22				
Speed up	**1**	**18.5**	**50.0**	56.0	56.0	94.5	110

Platform

- 2 x Intel Xeon E5-2650
- 2 x NVidia Kepler K20M

Good news

- CPUs: benefit of // and vectorization

… but suboptimal

Single node performances (1)

	GSL	GSL	OCL	OCL	OCL	OCL	OCL + 2xMPI
CPUs	1	32	32			32	
APUs				1	2	2	2
Time/Ev (s)	11.0	0.60	0.22	0.20			
Speed up	**1**	**18.5**	**50.0**	**56.0**	**56.0**	**94.5**	**110**

Platform

- 2 x Intel Xeon E5-2650
- 2 x NVidia Kepler K20M

Good news

- CPUs: benefit of // and vectorization

… but suboptimal

- GPGPUs: not enough work for the kernel (cost of the latencies > 25 %, because of the splitting ...)

Single node performances (1)

	GSL	GSL	OCL	OCL	OCL	OCL	OCL + 2xMPI
CPUs	1	32	32			32	
APUs				1	2	2	2
Time/Ev (s)	11.0	0.60	0.22	0.20	0.20	0.15	0.10
Speed up	**1**	**18.5**	**50.0**	**56.0**	**56.0**	**94.5**	**110**

Platform

- 2 x Intel Xeon E5-2650
- 2 x NVidia Kepler K20M

Good news

- CPUs: benefit of // and vectorization
- Good behavior of the OCL Event Dispatcher

… but suboptimal

- GPGPUs: not enough work for the kernel (cost of the latencies > 25 %, because of the splitting ...)
- Nvidia OCL : doesn't allow to feed async. 2 cards → Thanks to MPI

GG, SL, DC

Single node performances (2)

	GSL	GSL	OCL	OCL	OCL	OCL + 2xMPI	OCL
CPUs	1	32	32				All
APUs				1	2	2	2
Time/Ev (s)	11.0	0.60	0.22	1.08	0.55		0.16
Speed up	**1**	**18.5**	**50.0**	**10.1**	**19.9**		**70.3**

Platform

- 2 x Intel Xeon E5-2650
- 2 x Intel Xeon Phi 5110P

Good news

- Same remarks as for K20M platform
- Nice Intel OpenCL implementation : use simultaneously of two cards

… but suboptimal

- Same remarks as for K20M platform
- Xeon Phis: very bad performance ... splitting). VTune to study the issues

GG, SL, DC

Single node performances (3)

	GSL	OCL	OCL	OCL	OCL + 6xMPI	OCL
CPUs	All (1)	All				All
APUs			1	6	6	6
Time/Ev (s)	8.55	0.21	0.15	0.15	0.026	0.089
Speed up	1	39.1	56.0	55.6	328	95.3

Platform

- 2 x Intel Xeon E5-2650
- 6 x NVidia Titan

Good news

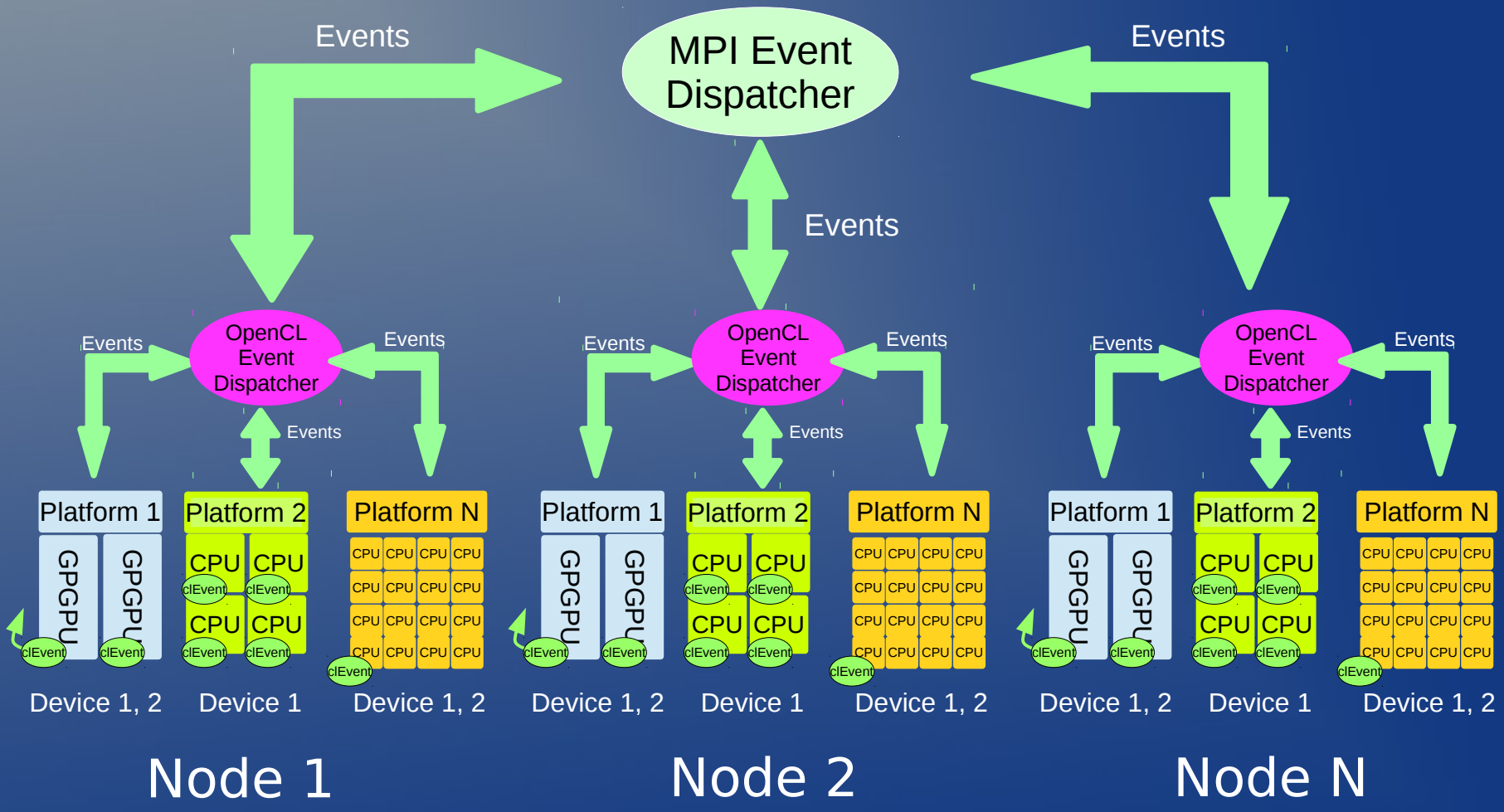
- Same remarks as for K20M platform
- Clock of the processor change

… but suboptimal

- Same remarks as for K20M platform

Multi nodes architecture

- Nodes aggregation with MPI



GG, SL, DC

Multi-nodes performances

	# Nodes	GSL	OCL	OCL
CPUs	1	All (1)	All	All
APUs	1			2
Time/Ev (s)	1	11.0	0.22	0.15
Speed up	1	1	50.0	94.5
Time/Ev (s)	2	11.0	0.127	0.092
Speed up	2	**1**	**85.99**	**119**
Speed up	1/2	**1**	**1.72**	**1.63**

1 node

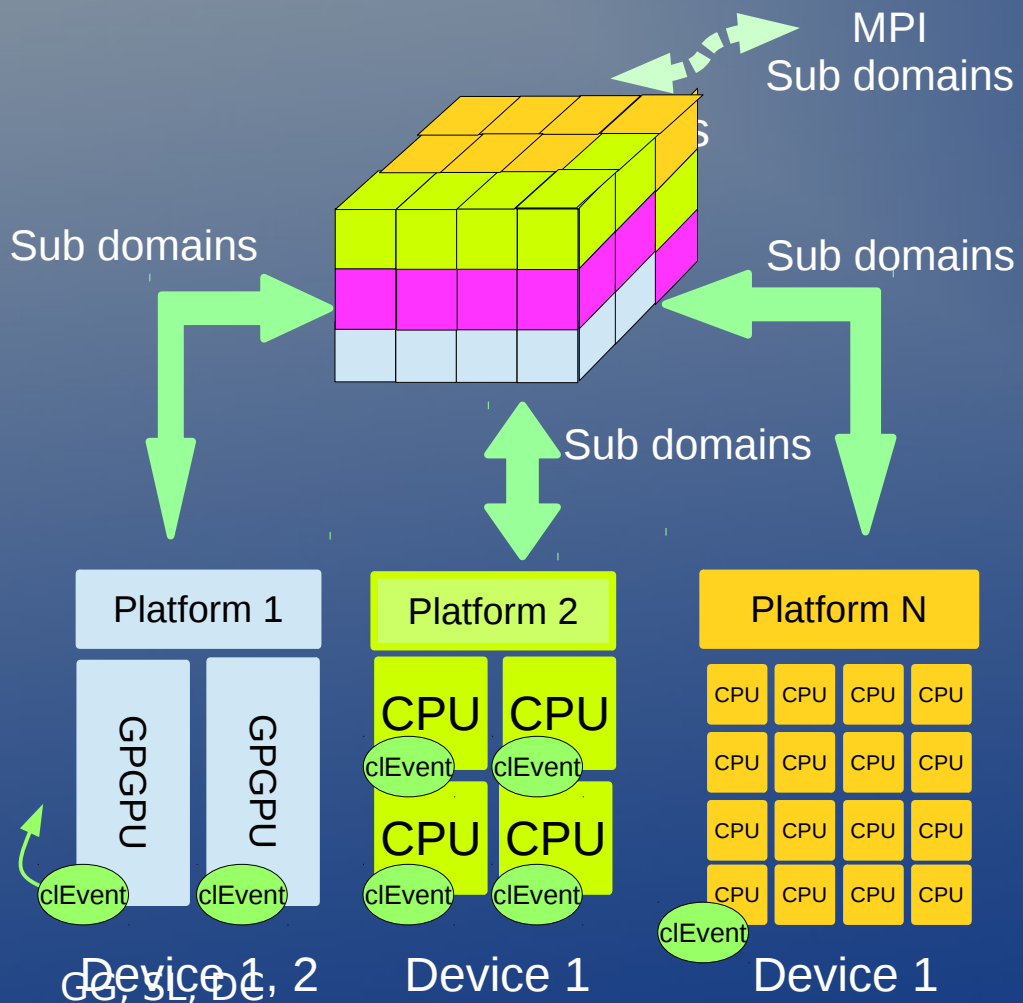
2 nodes

← global speed-up

Platforms

- 2 x Intel Xeon E5-2650
- 2 x NVidia Kepler K20M

Node architecture for high dimensional integral



- For large Integrals with a high dimensionality
- Dispatch sub-domains of the integral
- If needed integration domain can be spread on several nodes thanks to MPI

GG, SL, DC

Conclusion

- Global map of the application. Identified the bottlenecks.
- If we take into account
  - Improving kernel x 2 (workload)
  - Running 2 MPI per node
  - Using 2xK20 nodes (0.9 dispatcher efficiency)
- We target a reasonable global speed up of 450 on K20s, 600 for Titan node

- Remaining work (months):
  - The functions to integrate (LHAPDF, ...) written in Fortran, C++
  - 80 % translated in C99 (kernels)
  - CMSSW integration
- Other method: MCMC

Thanks

- To P2IO funds
- To the organizers of this meeting
- For your attention