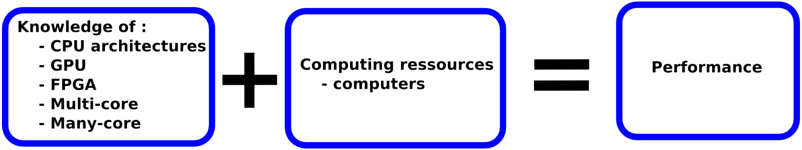
Why use High Performance Computing for analysis ?

Pierre AUBERT, Jean Jacquemier, Gilles MAURIN

Outline

Aim

Use the computer (CPU, GPU, FGPA, multi-core, many-core, ...) as efficent as possible

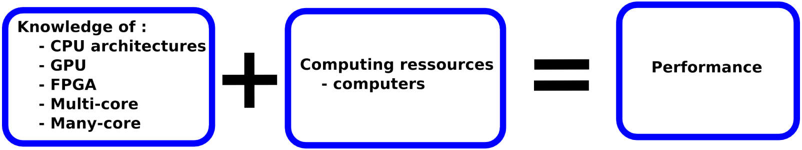


Do we need specific computers ?

NO

Aim

Use the computer (CPU, GPU, FGPA, multi-core, many-core, ...) as efficent as possible



Do we need specific computers ?

NO

Aim

Use the computer (CPU, GPU, FGPA, multi-core, many-core, ...) as efficent as possible

Knowledge of :
- **CPU architectures**
- **GPU**
- **FPGA**
- **Multi-core**
- **Many-core**

+

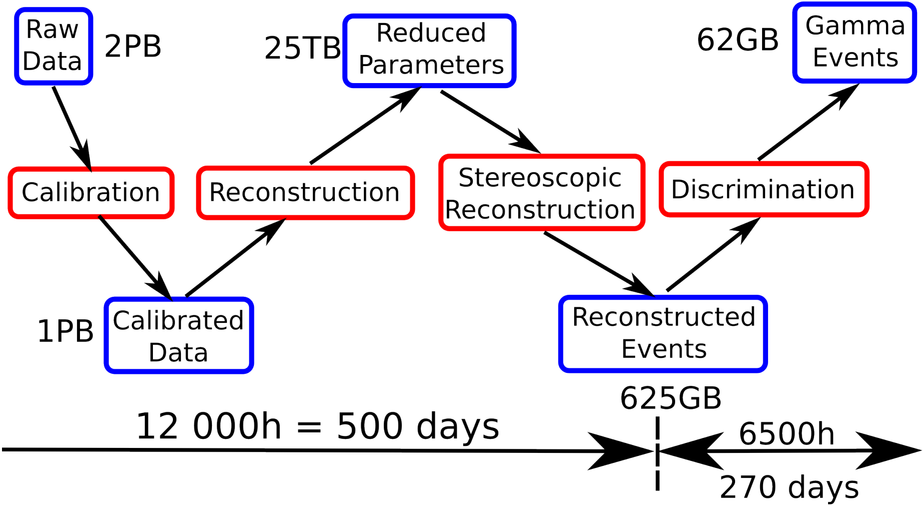
Computing ressources
- **computers**

=

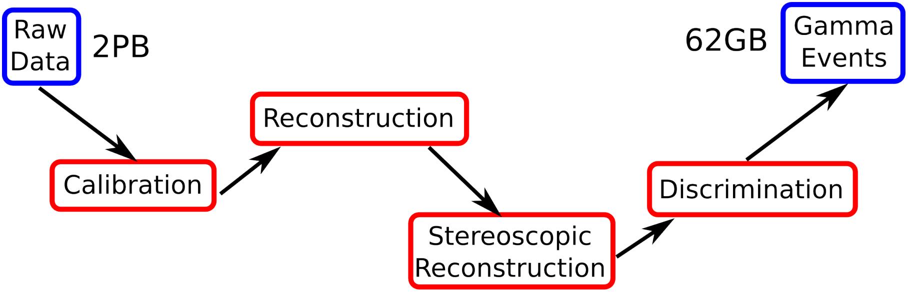
Performance

Do we need specific computers ?

NO



Total Memory : 3.025687 PB



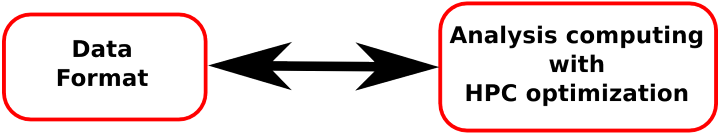
Total Memory : 2.000062 PB

Memory saved : 34 %

- How the Data are used ?
- Why the Data are used ?

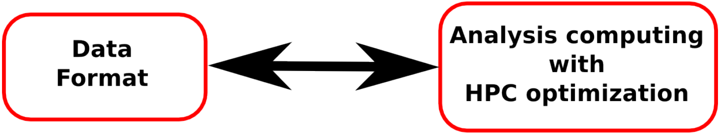
Convertion = Waste of time

- How the Data are used ?
- Why the Data are used ?



Convertion = Waste of time

- How the Data are used ?
- Why the Data are used ?

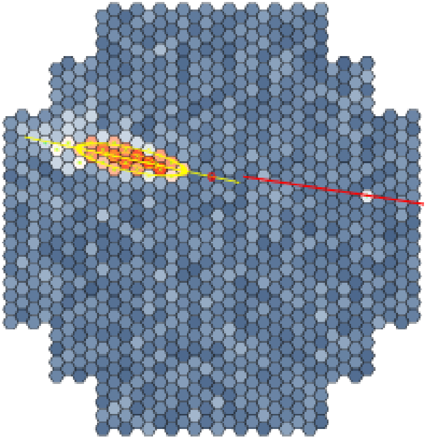


Convertion = Waste of time

Analysis optimization

Hillas parameters calculation

- Reduction : Sum of all vector's elements
- First momentum : Mean of all vector's elements
- Second momentum : Mean of all square vector's elements



Speed up examples

Reduction — Sum of all vector's elements

	Speed (cy/el)
Classical réduction	2.69842

Hillas optimization

	Speed (cy/el)	Speed up
Initial Hillas	2125.5	1
Optimized Hillas good data format	53.1375	40.0

SSE4 architecture

- Compute 4 floats at the same time
- Instruction set : 2006
- CPU : 2007

AVX architecture

- Compute 8 floats at the same time
- Instruction set : 2008
- CPU : 2011

AVX 512 architecture

- Compute 16 floats at the same time
- Instruction set : 2013
- CPU : 2016

Speed up examples

Reduction — Sum of all vector's elements

	Speed (cy/el)	Speed up
Classical réduction	2.69842	1
Vectorized réduction	0.702845	3.839281
Our fast reduction SSE4	0.226675	11.904356
Our fast reduction AVX	0.11379	23.714034

Hillas optimization

	Speed (cy/el)	Speed up
Initial Hillas	2125.5	1
Optimized Hillas good data format	53.1375	40.0
Our fast Hillas SSE4	6.39931	332.14516
Our fast Hillas AVX	2.98499	712.06268

ASTERICS Deliverable

- Source code
  - Optimized reduction (for *GCC*)
  - Optimized barycenter (for *GCC*)
  - Optimized first and second momentum (for *GCC*)
  - *CMake* builder
    - *CMake* macro for finding CPU architecture (SSE4, AVX)
  - Abstract intrinsics functions
- Program
  - Efficent data format generator

Interest

- Efficent use of computers (CPU, GPU, ...)
  - Save Computing time
  - Save development time
  - Save electricity and money
  - Green HPC

Backups

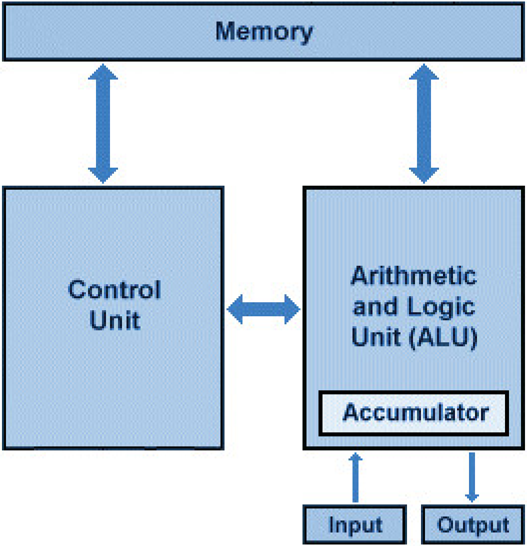
CPU Architecture

Definition

Cycle : basis unit of time in a CPU

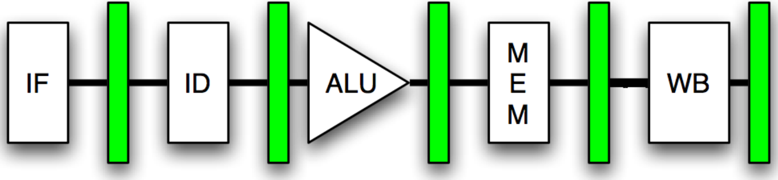
Time

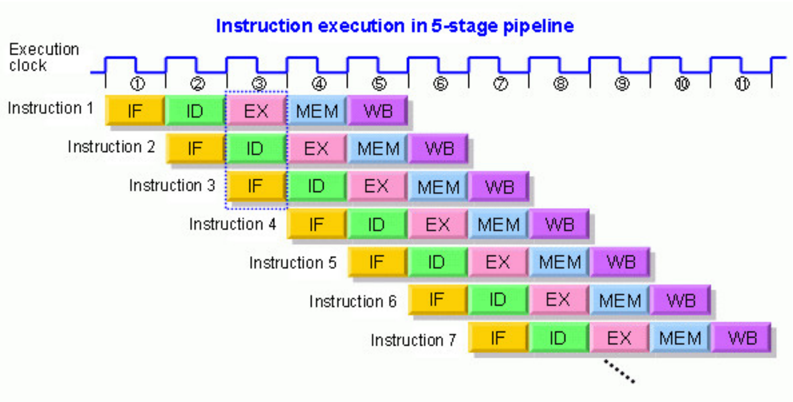
- 1 cycle per elementary operation (load, store, add, ...)
- 6 cycles per whole operation ($c = a + b$)



Pipeline approach

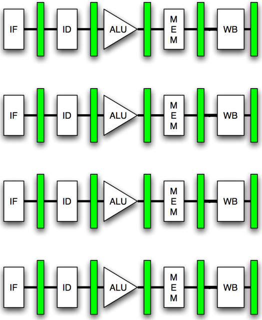
- IF : Instruction Fetch
- ID : Instruction Decode
- EX : Execution
- MEM : Memory
- WB : Write Bytes



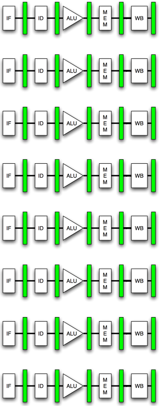

Instruction execution in 5-stage pipeline
Execution clock
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪
Instruction 1
IF ID EX MEM WB
Instruction 2
IF ID EX MEM WB
Instruction 3
IF ID EX MEM WB
Instruction 4
IF ID EX MEM WB
Instruction 5
IF ID EX MEM WB
Instruction 6
IF ID EX MEM WB
Instruction 7
IF ID EX MEM WB

CPU Recent Architectures

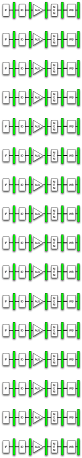
SSE4 4 floats
Instruction set : 2006
CPU : 2007



AVX 8 floats
Instruction set : 2008
CPU : 2011



AVX 512 16 floats
Instruction set : 2013
CPU : 2016



Data format

Efficent only if data are contiguous

Reductions optimization

Speed without optimization — on SSE4

2.69842 cy/el

Speed by vectorizing with *GCC* — -ftree-vectorize, ...

0.702845 cy/el $\Longrightarrow$ acceletation factor 3.839281

- -O3 or -Ofast
- -ftree-vectorize
- -march=native
- -tune=native
- -ffast-math
- -mavx (only for AVX architectures)

Reductions optimization

Speed without optimization — on SSE4

2.69842 cy/el

Speed by vectorizing with *GCC* — -ftree-vectorize, ...

0.702845 cy/el $\Longrightarrow$ acceleration factor 3.839281

With SSE4 vectorizing intrinsics manage by hand

Unrolling	Speed cy/el	Acceleration
1	0.616841	4, 374579
2	0.338947	7.961185
4	0.226990	11.887836
8	0.226675	11.904356

With AVX vectorizing intrinsics manage by hand

Unrolling	Speed cy/el	Acceleration
2	0.172954	15.601951
4	0.115568	23.349197
8	0.11379	23.714034