Profiling electromagnetic showers with and without Cherenkov Module

Matthieu CARRERE

Thesis - Optimization of HPC code for Gamma Ray experiments

Supervisors : Luisa ARRABITO, David PARELLO, Philippe LANGLOIS and Georges VASILEIADIS
CTA contact : Johan Bregeon

Branch link with commit cf7fcffa based on master branch (commit ac43f9d6) :
https://gitlab.ikp.kit.edu/AirShowerPhysics/corsika/-/tree/275-Cherenkov-module-tmp

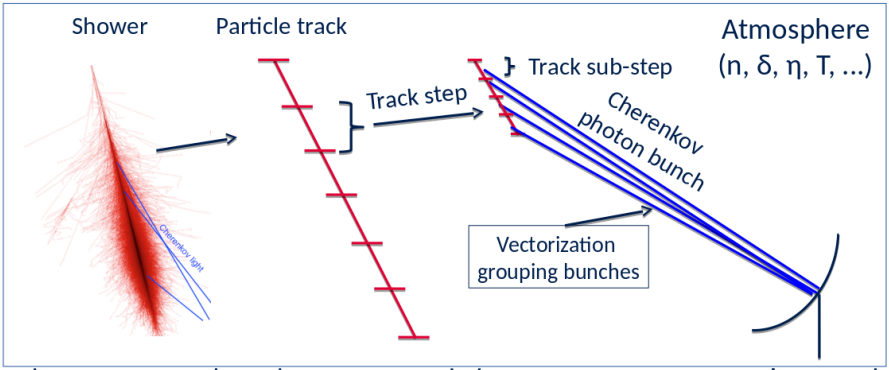
Context

Cherenkov Module in CORSIKA 7

- Important for CTA experiments
- Represents >70% of time elapsed
- Optimization work increased performances with a speedup of 2 in cycles :
  - vectorization
  - memory optimization
  - data layout optimization

Cherenkov Module

Simulation



Cherenkov Module

Roadmap

- Goals :
  - Work step by step
  - Compare physical results and performances
  - Use performance counters
  - Take into account the CORSIKA 7 optimization work
- Build in 2 steps :
  - A version close to CORSIKA 7 algorithms (this version)
  - A version with better C++17 aspect and efficient (to do)

Cherenkov Module

Classes

- Cherenkov : Generation and propagation of photons
- AtmosphereTabulated : Atmosphere created with an input file to propagate photons (takes into account the CORSIKA 8 atmosphere)
- TelescopeGrid : Create a grid of telescopes and save photons by telescope
- Telescope : class telescope with 3 different geometries (daugther classes) which manages bunches
- Bunch : saves some data about photons
- Interpolation : Interpolation calculations like CORSIKA 7 with linear and cubic spline interpolations
- CherenkovToolBox : a Cherenkov version to use CORSIKA 7 entries and to do tests (compare physical results,...)

Strategy

- Profiling -> Optimization -> Validation of results
- Profiler : *Perf Linux* with *stat* and *record* modules
- Performance counters and ratios : cycles, instructions, cache-misses, vectorization ratio τ_{vec},...

$$\tau_{vec} = \frac{\sum 2 * I_{128,dbl} + 4 * I_{256,dbl}}{\sum 2 * I_{128,dbl} + 4 * I_{256,dbl} + \sum I_{scal,dbl}} \quad (1)$$

- CORSIKA's measurements with flags -O3 -fno-fast-math (-mavx2)

CPU	GCC	OS
Intel i7-8665U,1.9Ghz(to 4.8Ghz)	9.3	Ubuntu 20
Intel Bi-Xeon Gold 5122,3.6Ghz	8.4	Centos 7
AMD Ryzen 9 3950x,4.4Ghz	8.4	Centos 7

Strategy

Experiment - Electromagnetic Air Shower : a custom PROPOSAL example in RELEASE mode

- Incident particle : electron
- Total Energy : 50 GeV
- Angles : 0.01 (zenith), 180. (azimutal)
- Seed : 1 (fixed)
- Cut Energies : 20 MeV
- Magnetic Field : none
- Atmosphere's height : 112.8km + Earth's radius
- Observator's height : 2.15km + Earth's radius
- Number of showers : 500
- Build : RELEASE / logging::info

Comparison CORSIKA7/CORSIKA8

With O3 and nofastmath - Xeon CPU

- CORSIKA 7 experiment produced : $1{,}01.10^9$ photons
- CORSIKA 8 experiment should produce : $1{,}39.10^9$ photons

Version	Time	Gcycles	Ginstructions	GfpOperations
C7.7100(avx2)	40s	140	265	99
C7.7100 +38%(estimation)	55s	194	365	137
C8 (without Cherenk/outputs)	149s	549	934	163

- CORSIKA 7 vectorization rate : 57%
- CORSIKA 8 vectorization rate : <1%

Comparison CORSIKA7/CORSIKA8

CORSIKA 7 - With O3, AVX2, c99, nofastmath - Xeon CPU

Version	Function	Cycles Overhead	Categorie	Cycles Overhead
C7	cerenkovopt	16%	Cherenkov	39%
	raybndvec	12%	Math Functions	32%
	vectorasin	8%	Electr	3%
	rmmard	7%		
	telout	7%		
	sinavx	5%		
	ieee754logavx	5%		
	vectorsin	4%		
	thickx	4%		
	cosavx	4%		
	electr	3%		
	vectorcos	3%		
	ieee754asinsse2	3%		

Comparison CORSIKA7/CORSIKA8

CORSIKA 8 Without Cherenkov Module and without outputs - Xeon CPU

Version	Function	Cycles Overhead
C8	pow	33%
	std::Spcountedbase<(gnucxx::Lockpolicy)2>::Mrelease	11%
	PROPOSAL::PhotoTsai::FunctionToIntegral	8%
	PROPOSAL::PhotoPairTsai::DifferentialCrossSection	5%
	log	5%
	exp	4%
	PROPOSAL::Integral::Function	2%
	PROPOSAL::Interpolant::Interpolate	2%
	sin	2%

- Mathematical Functions : 44%
- PROPOSAL : 17%

Comparison CORSIKA7/CORSIKA8

- First proposition : Use a vectorized mathematical library compatible with parallelization (SLEEF ? SIMD vector libm ?)
- Second proposition : Work with arrays to unlock vectorization in some modules when it's possible
- Third proposition : Reduce pow usage

Comparison CORSIKA8/CORSIKA8 with Cherenkov Module

Without output - i7 CPU

- Flag by default (SSE) : -O3 -fno-fast-math

Version	Time	Gcycles	Ginstructions	GfpOperations	VectorRate
NoCherenk	46s	192	367	118	<1%
Cherenk	88s	368	762	207	6%
Cherenk+Grid	188s	786	1740	742	73%

- Flag AVX2 : -O3 -mavx2 -fno-fast-math

Version	Time	Gcycles	Ginstructions	GfpOperations	VectorRate
NoCherenk	46s	192	366	118	<1%
Cherenk	86s	358	729	207	6%
Cherenk+Grid	164s	688	1390	742	73%

Comparison CORSIKA8/CORSIKA8 with Cherenkov Module

Without output - Different CPU

- AVX2 gain (Version / Version AVX2) :

CPU	Version	Time	Gcycles	Ginstructions
i7	NoCherenk	1.0	1.0	1.0
	Cherenk	1.02	1.03	1.05
	Cherenk+Grid	1.15	1.14	1.25
Xeon	NoCherenk	1.0	1.0	1.0
	Cherenk	0.99	1.01	1.02
	Cherenk+Grid	1.06	1.09	0.77
Ryzen	NoCherenk	1.0	1.0	1.0
	Cherenk	1.02	1.01	1.03
	Cherenk+Grid	1.12	1.12	1.25

- Possible gain if algorithm is "auto-vectorization friendly" but can decrease the clock frequency (3.7Ghz->3.5Ghz on the Xeon)

CORSIKA8 - Cherenkov Module

Without Telescope Grid - With AVX2 - i7 CPU

Function	Cycles Overhead	Categorie	Cycles
pow	22%	Math Functions	57%
exp	16%	Cherenkov	11%
log	7%		
Cherenkov::generatePhotons	7%		
sin	6%		
cos	4%		
std::Spcountedbase::Mreleas	4%		
Cherenkov::propagatePhotons	4%		
std::sharedptr	3%		
PROPOSAL::FunctionToIntegral	2%		
PROPOSAL::Interpolate	2%		
asin	2%		

CORSIKA8 - Cherenkov Module

With Telescope Grid - With AVX2 - i7 CPU

Function	Cycles Overhead	Categorie	Cycles
Cherenkov::savePhotonsOnTelescope	28%	Cherenkov	36%
pow	16%	Math Functions	45%
exp	11%		
log	5%		
Cherenkov::generatePhotons	5%		
sin	4%		
cos	3%		
std::Spcountedbase::Mreleas	3%		
Cherenkov::propagatePhotons	3%		
std::sharedptr	2%		
PROPOSAL::FunctionToIntegral	2%		
PROPOSAL::Interpolate	1%		

CORSIKA8 - Cherenkov Module

With AVX2 - Xeon CPU

- Table : cycle usage by categorie, vectorization rate and time

Version	Cherenkov	Math Functions	Vectorization Rate	Time
C7+38%(estimation)	39%	32%	57%	55s
C8	none	44%	1%	149s
Cherenk	11%	57%	5%	215s
Cherenk+Grid	36%	45%	45%	306s

Next Steps

- Reduce pow usage
- Improve telescope grid algorithm
- Improve vectorization (mainly for mathematical functions)
- Fix 2 bugs : zenith angle <0.01 and an experiment bug with output (not the same number of photons are produced)
- Work on a version of Cherenkov module with CORSIKA 8 style
- Add a python script to manage perf counters

Thank you for your attention