A C++ Cherenkov photons simulation in CORSIKA 8

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

05/18/21

Context

CTA (Cherenkov Telescope Array)

- Next generation ground-based observatory for gamma-ray astronomy at very-high energies
  - 2 sites in the two hemispheres and counts more than tens of telescopes of three different sizes
- Simulation represents 80% of CPU time
  - Consumes almost 100 million HS06 CPU hours/year
  - Uses CORSIKA 7 with the Cherenkov module
  - Runs on the EGI grid

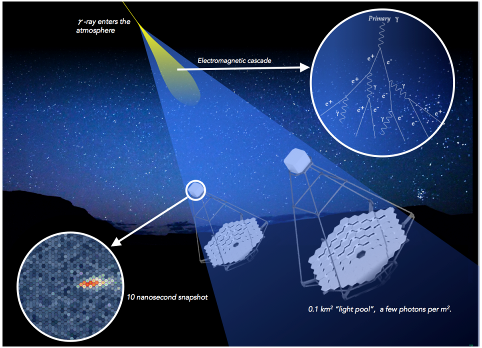

γ-ray enters the atmosphere
Primary γ
Electromagnetic cascade
10 nanosecond snapshot
0.1 km² "light pool", a few photons per m².

Figure 1: Representation of electromagnetic cascade with production and reception on telescopes of Cherenkov photons

Context

CORSIKA (COsmic Ray SImulations for KAscade)

- A Monte Carlo code to simulate detailed extensive air showers
- CORSIKA 7 : in production (optimization work is presented in first part)
  - Developed 30 years ago
  - Mainly in Fortran 77 with some C modules
  - Extremely complex to maintain and modify
- CORSIKA 8[1] : under development (first Cherenkov module results are presented in second part)
  - Rewriting in modern C++

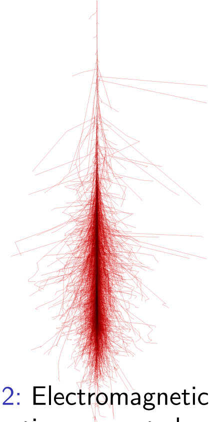


Figure 2: Electromagnetic air shower with xz projection generated with CORSIKA - 100GeV - Photon - Zenith angle: 0 deg

[1] https://gitlab.ikp.kit.edu/AirShowerPhysics/corsika

Strategy

- Profiling -> Optimization -> Validation of results
- Profilers : *Perf Linux* (with *stat* and *record* modules) and *PAPI*
- Performance counters and ratios : cycles, instructions, cache-misses, vectorization ratio τ_{vec} (1),...

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

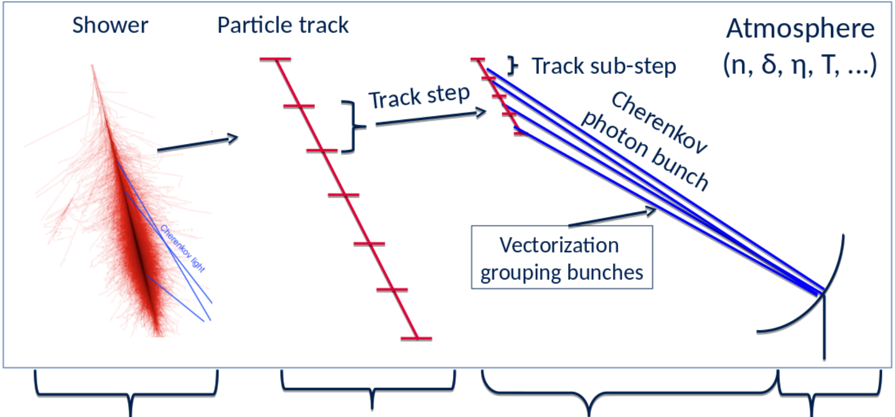
- Hardware :

CPU	GCC	OS
Intel i7-8665U,1.9Ghz(to 4.8Ghz)	8.3	Ubuntu 20
Intel Bi-Xeon Gold 5122,3.6Ghz	8.4	Centos 7

- Compilation flags : -O3 -mavx2 (-std=c99 for CORSIKA 7)

CORSIKA 7 - Optimization work

Main steps of the Cherenkov simulation



EGS4/SHOWER/ELECTR

- Shower development
- Transport of electron/positron

cerenkovopt

- Nb of emitted Cherenkov photons
- Grouped in bunches

raybnd_vec

- Deviation of photon bunches by atmosphere refraction

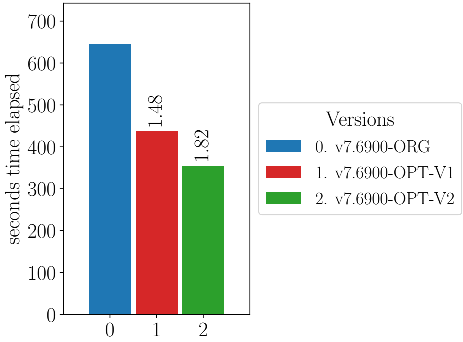
telout

- Record coordinates of photon bunches hitting the telescopes

CORSIKA 7 - Optimization work

Xeon CPU - 5000 showers - Energy spectrum E^{-2} from 3 GeV to 330 TeV

- CORSIKA 7.6900-OPT-V1 :
  - Vectorized rewriting for the Cherenkov module
  - Usage of the vectorized mathematical library *vlibm*[2]
- CORSIKA 7.6900-OPT-V2 :
  - Memory optimization : cache misses decrease by x20
  - Data layout optimization
- General speedup in time : 1.82
- Produces equivalent results
- Used in the last large scale CTA production
  - Economized almost 20 million HS06 CPU hours since July 2020



[2] vector-libm: https://gitlab.com/cquirin/vector-libm

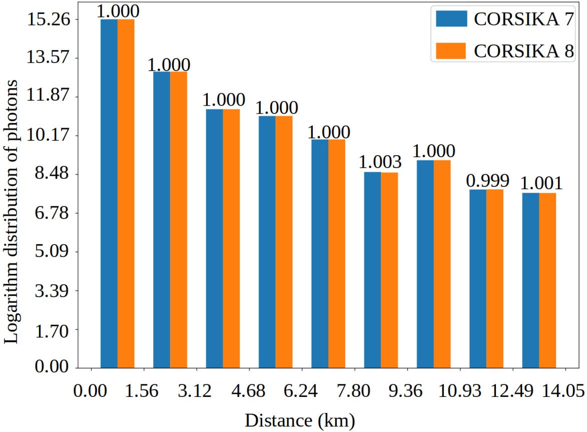
CORSIKA 8 - Development of Cherenkov module

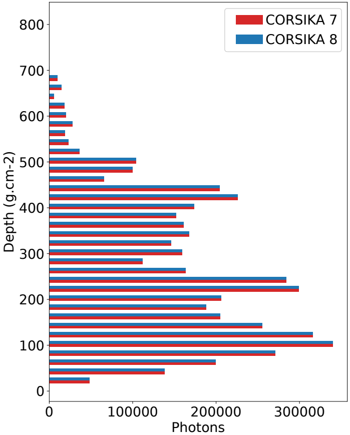
CORSIKA 8 : a framework in modern C++

- Modularity : each physical process has its module and offers the possibility to create custom process sequences, ...
- Flexibility : custom geometries, custom types
- Reliability : continuous integration, unit testing
- Efficiency : static polymorphism, parquet format for outputs, vectorization,...

Physics results comparison for Cherenkov Module C8/C7

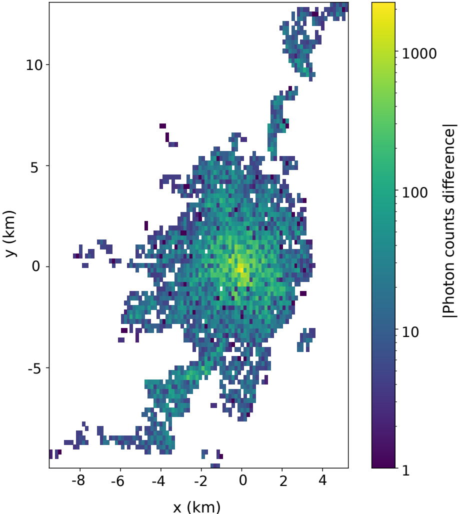
I7 CPU - 10 showers - (Left) Lateral distribution of photon density with respect to the shower axis (Right) Longitudinal distribution





Physics results comparison for Cherenkov Module C8/C7

I7 CPU - 10 showers - 2D map of the photon counts difference on the ground - 4,9.10^{6} produced



	Pearson	Kolmogorov
NbPhotons	1.000	1.000
PosZ	1.000	1.000
DistXY	0.996	0.994
DirX	0.982	0.963
DirY	0.986	0.939
Time	0.991	0.972

Performance comparison for Cherenkov Module C8/C7

I7 CPU - with *vlibm* - 10 showers - Energy spectrum E^{-2} from 3 GeV to 330 TeV

Version	Instructions	Cycles	Wall clock time
C8 / C7	1.02	1.57	1.39

Version	Scalars[3] (Mi)	Vectors (Mi)	Vectorization Ratio
C7	108	53	66.4%
C8	219	29	34.8%

- Number of instructions is almost the same
- Vectorization rate decreased
- An increase of number of cycles

[3] Scalars : scalars double precision floating-point instructions, Vectors : vectorized double precision floating-point instructions

Performance comparison for Cherenkov Module C8/C7

I7 CPU - with *vlibm* - 10 showers - Energy spectrum E^{-2} from 3 GeV to 330 TeV

Version	BM/B[4]	CM/CR	LLCM/LLC
C7	3.0%	6.0%	9.8%
C8	3.3%	32.7%	41.8%

- Similarly BM/B = same algorithms
- Cache misses and L3 load misses increase
- We need more investigations

[4] BM: branch misses, B: branch, CM: cache misses, CR: cache reference, LLCM: last level cache misses, LLC: last level cache load

Next Steps

- Optimize the Cherenkov Module
  - Test different mathematical libraries appropriate for CORSIKA 8
  - Develop our own vectorized operators thanks to C++
  - Evaluate the single precision to improve vectorization efficiency
- Build in 2 steps :
  - A version close to CORSIKA 7 algorithms to obtain same results
  - A version with better C++17 features and more efficient

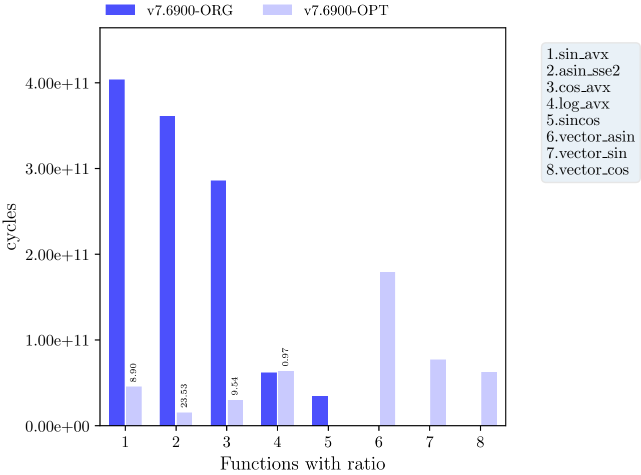
Thank you for your attention

This work was conducted in the context of the CTA Consortium

Backups

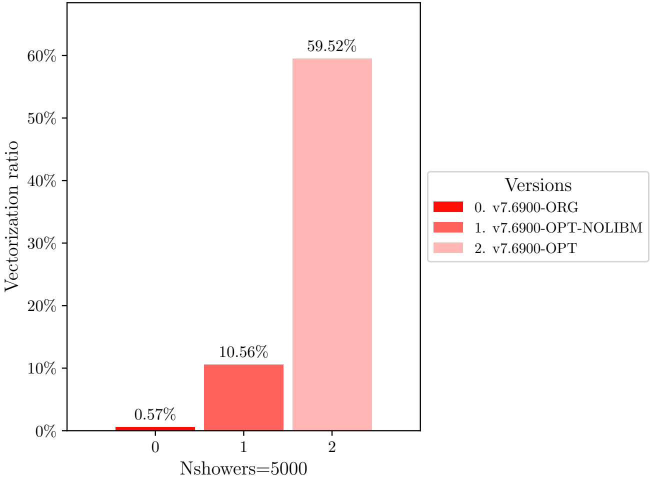
Previous Optimizations

Mathematical functions



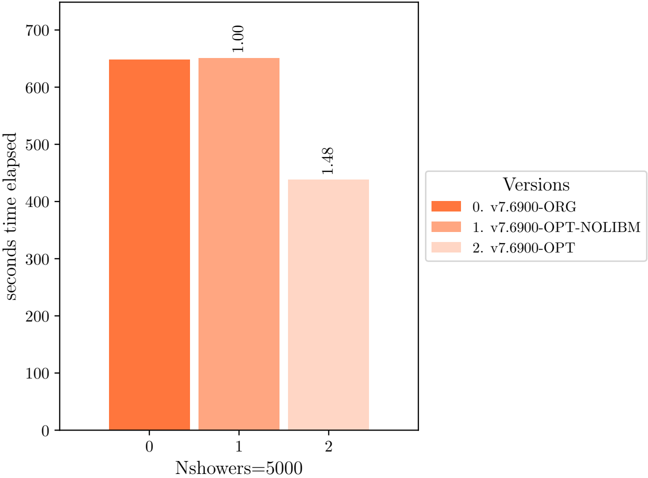
Previous Optimizations

NoLibm - Vectorization



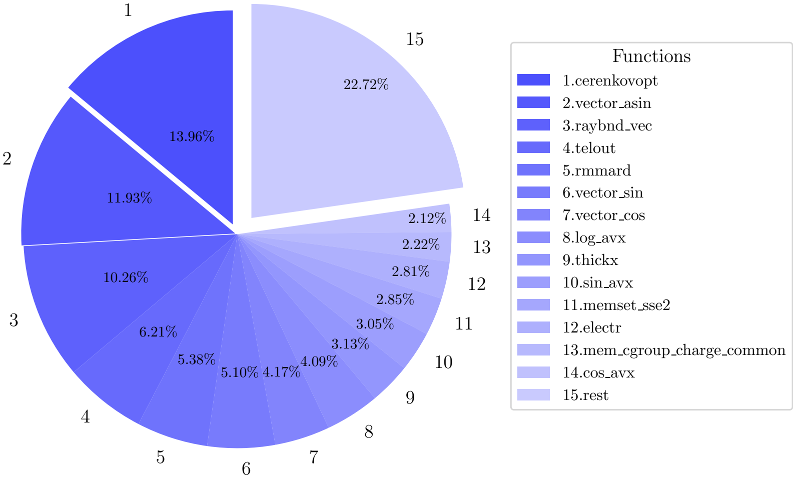
Previous Optimizations

NoLibm - Time



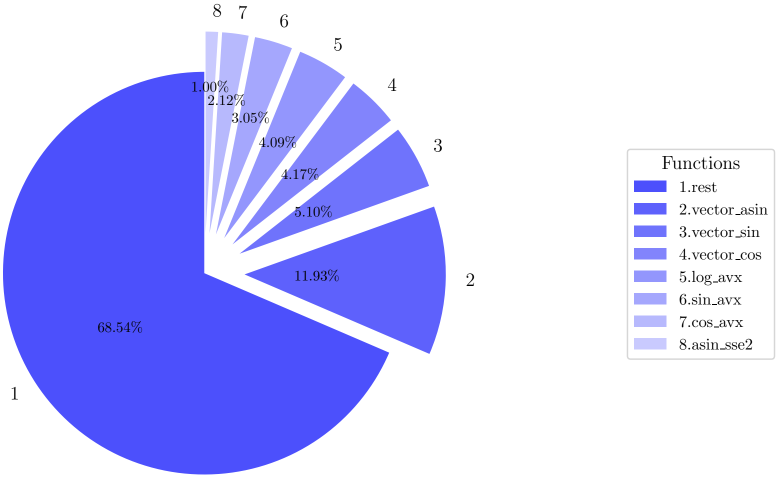
Optimization work

Cycles Distribution - Complete



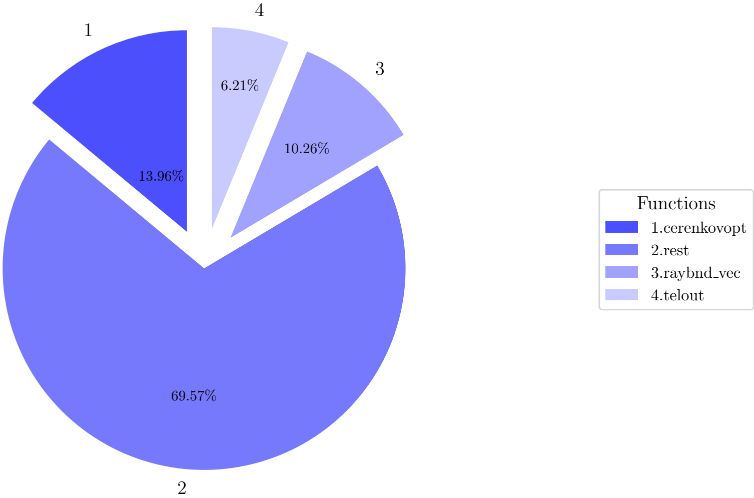
Optimization work

Cycles Distribution - Maths



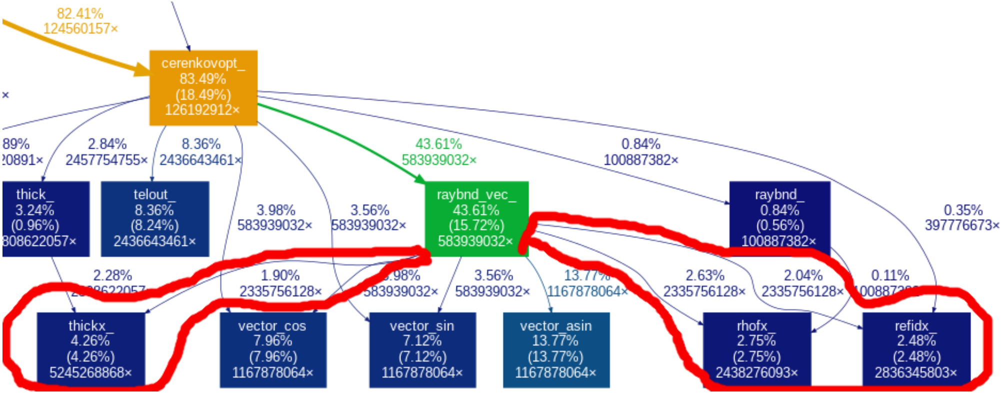
Optimization work

Cycles Distribution - Cerenk functions



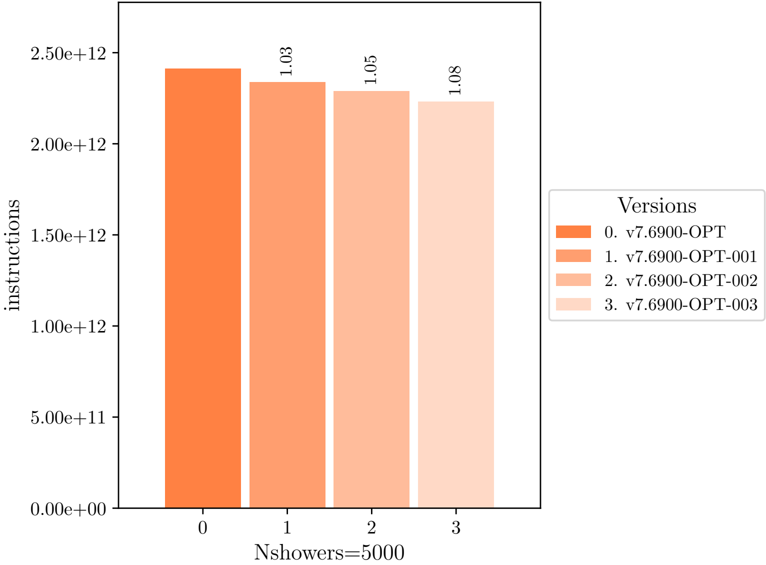
Optimizations work

II.Reducing function calls and instructions - Profiling with gprof



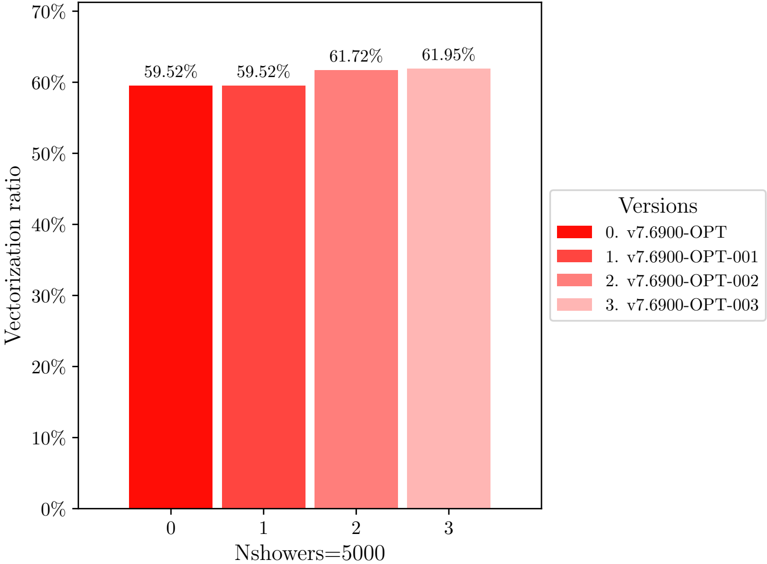
Optimization work

II.Reducing function calls and instructions - Total instructions



Optimization work

II.Reducing function calls and instructions - Vectorization rate



Optimization work

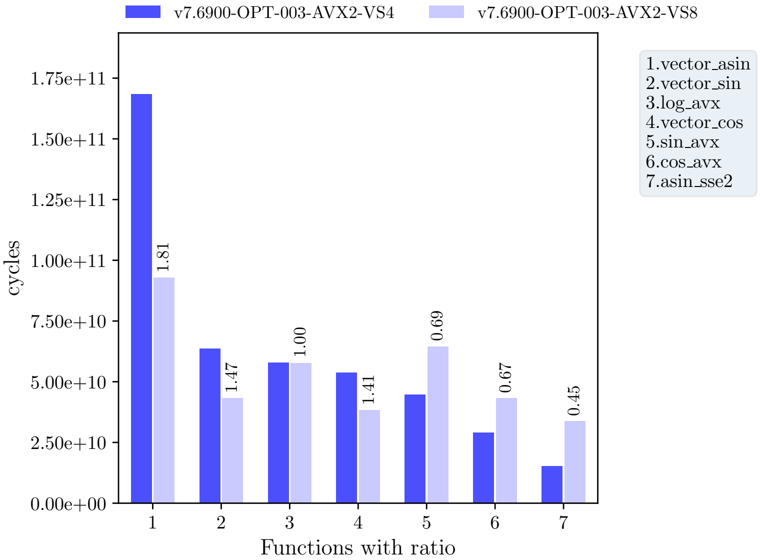
III.Data layout optimization - Cerenkovopt Function

```
// Vector version
for( istc= 0; istc < (mstepc/VECTORSIZE); istc++){
// ...
   vector_cos(cosphi_vec, phicer_vec);
// ...
   raybndvec(zemvec, uemisvec, vemisvec, wemisvec, xcervec, ...);
// ...
}

// Scalar version
for( istc= 0; istc < mstepc % VECTORSIZE; istc++){
// ...
   raybnd(&zem, &(crceren2.uemis), &(crceren2.vemis), ...);
// ...
}
```

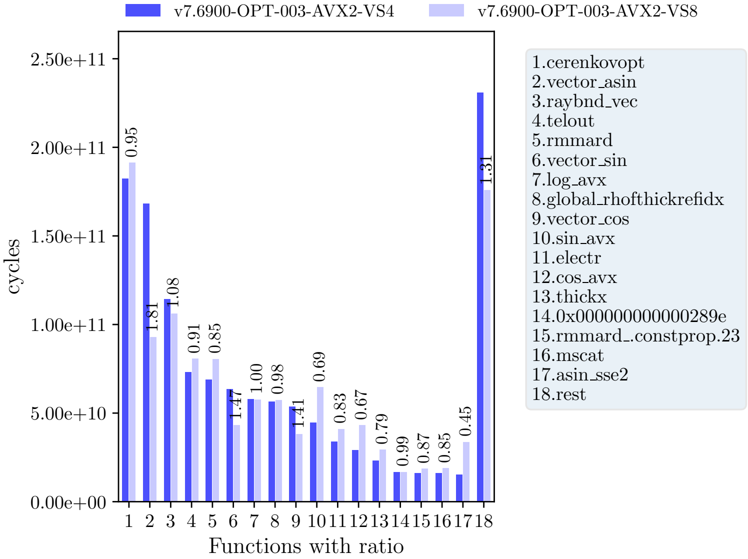
Optimization work

III.Data layout optimization - Cycles by maths functions



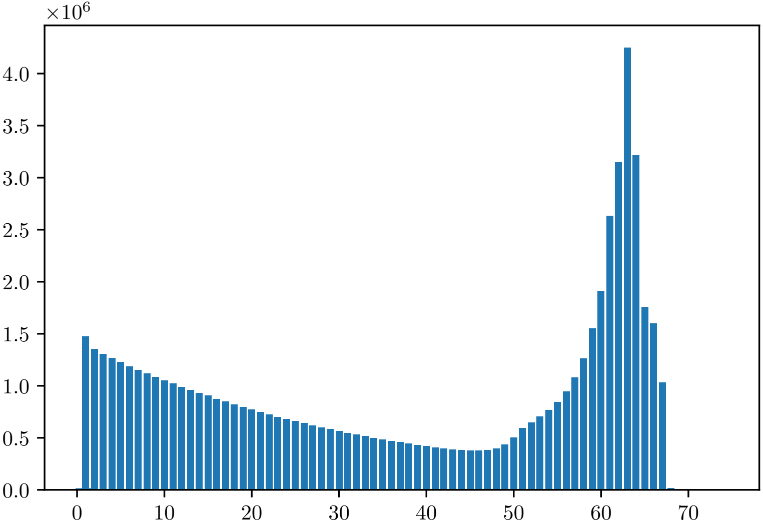
Optimization work

III.Data layout optimization - Cycles by functions



Optimization work

III.Data layout optimization - Substep distribution



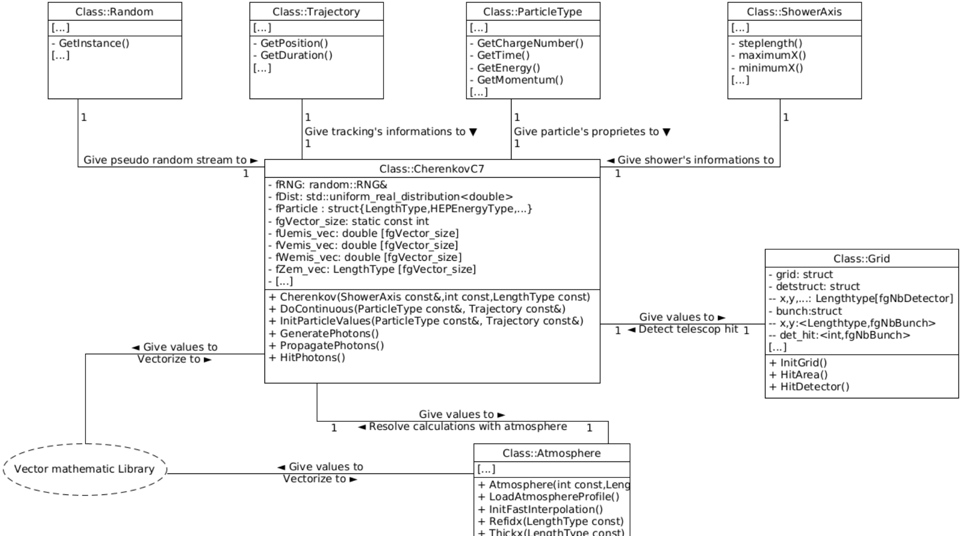
Conclusion

Details

Counter	Ratio(ORG/OPT-004)	Ratio(OPT/OPT-004)
Time	1.82	1.26
Cycles	1.90	1.25
Instructions	1.77	1.07
Cache References	1.61	1.61
Cache Misses	21.11	20.86
Branches	1.90	0.97
Branch Misses	1.90	0.97

Corsika 8

Class diagram



Class::Random
[...]
- GetInstance()
[...]

Class::Trajectory
[...]
- GetPosition()
- GetDuration()
[...]

Class::ParticleType
[...]
- GetChargeNumber()
- GetTime()
- GetEnergy()
- GetMomentum()
[...]

Class::ShowerAxis
[...]
- steplength()
- maximumX()
- minimumX()
[...]

Class::CherenkovC7
- fRNG: random::RNG&
- fDist: std::uniform_real_distribution<double>
- fParticle : struct{LengthType,HEPEnergyType,...}
- fgVector_size: static const int
- fUemis_vec: double [fgVector_size]
- fVemis_vec: double [fgVector_size]
- fWemis_vec: double [fgVector_size]
- fZem_vec: LengthType [fgVector_size]
- [...]
+ Cherenkov(ShowerAxis const&,int const,LengthType const)
+ DoContinuous(ParticleType const&, Trajectory const&)
+ InitParticleValues(ParticleType const&, Trajectory const&)
+ GeneratePhotons()
+ PropagatePhotons()
+ HitPhotons()

Class::Grid
- grid: struct
- detstruct: struct
-- x,y,...: Lengthtype[fgNbDetector]
- bunch:struct
-- x,y:<Lengthtype,fgNbBunch>
-- det_hit:<int,fgNbBunch>
[...]
+ InitGrid()
+ HitArea()
+ HitDetector()

Class::Atmosphere
[...]
+ Atmosphere(int const,Leng
+ LoadAtmosphereProfile()
+ InitFastInterpolation()
+ Refidx(LengthType const)
+ Thickx(LengthType const)

Vector mathematic Library

1
Give pseudo random stream to ►
1
1
Give tracking's informations to ▼
1
1
Give particle's proprietes to ▼
1
1
◄ Give shower's informations to
1
Give values to ►
1 ◄ Detect telescop hit 1
◄ Give values to
Vectorize to ►
Give values to ►
1 ◄ Resolve calculations with atmosphere 1
◄ Give values to
Vectorize to ►

CORSIKA 8 - Development of 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,...)

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 category, 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

Strategy

Experiment choose for CORSIKA 7 and CORSIKA 8 in slides

- Incident particle : gamma
- Total Energy : 3GeV-330TeV (power law in energy)
- Angles : 20. (zenith), 180. (azimuthal)
- Seed : fixed
- Cut Energies : 20 MeV
- Magnetic Field : 21 muT
- Atmosphere's height : 112.8km
- Observator's height : 2.15km