CORSIKA 8 PERFORMANCE

Alice Faure (LUPM) with Luisa Arrabito (LUPM),
David Parello (Univ. Perpignan)

OUTLINE

1. Setup
2. Performance profiling
3. Memory profiling
4. A few questions
5. Suggestions for this workshop

Setup – Use case

A 1 TeV hadronic cascade with 30° inclination:

- N_showers = 10 or 100
- Fixed seed

```
./c8_air_shower --pdg 2212 -E 1e3 -z 30 -N 100 -s 1165816512 -f
      hadron_100n_1e3_30z > std.out
```

Setup – Test platform

- Architecture : Intel Xeon Gold, avx512 support

```
faure@phone10:~$ lscpu
Architecture :                       x86_64
Mode(s) opératoire(s) des processeurs : 32-bit, 64-bit
Boutisme :                           Little Endian
Address sizes:                       46 bits physical, 48 bits virtual
Processeur(s) :                      96
Liste de processeur(s) en ligne :    0-95
Thread(s) par cœur :                 2
Cœur(s) par socket :                 24
Socket(s) :                          2
Nœud(s) NUMA :                       2
Identifiant constructeur :           GenuineIntel
Famille de processeur :              6
Modèle :                             85
Nom de modèle :                      Intel(R) Xeon(R) Gold 5220R CPU @ 2.20GHz
Révision :                           7
Vitesse du processeur en MHz :       1000.004
Vitesse maximale du processeur en MHz : 4000.0000
Vitesse minimale du processeur en MHz : 1000.0000
BogoMIPS :                           4400.00
Virtualisation :                     VT-x
Cache L1d :                          1.5 MiB
Cache L1i :                          1.5 MiB
Cache L2 :                           48 MiB
Cache L3 :                           71.5 MiB
Nœud NUMA 0 de processeur(s) :       0-23,48-71
Nœud NUMA 1 de processeur(s) :       24-47,72-95
Vulnerability Itlb multihit:         KVM: Mitigation: Split huge pages
Vulnerability L1tf:                  Not affected
Vulnerability Mds:                   Not affected
Vulnerability Meltdown:              Not affected
Vulnerability Mmio stale data:       Mitigation; Clear CPU buffers; SMT vulnerable
Vulnerability Retbleed:              Mitigation; Enhanced IBRS
Vulnerability Spec store bypass:     Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Vulnerability Spectre v1:            Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2:            Mitigation; Enhanced IBRS, IBPB conditional, RSB filling, PBRSB-eIBRS SW sequence
Vulnerability Srbds:                 Not affected
Vulnerability Tsx async abort:       Mitigation; TSX disabled
Drapaux :                            fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_
                                     perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1 sse
                                     4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb cat_l3 cdp_l3 invpcid_single intel_ppin ssbd mba ibrs ibpb s
                                     tibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflush
                                     opt clwb intel_pt avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local dtherm ida arat pln pts hwp hwp_act_window hw
                                     p_pkg_req pku ospke avx512_vnni md_clear flush_l1d arch_capabilities
```

Setup – Compilation

- Compilation type : Release with debug
- Compilation flags : `-O3 -g -mavx2 -mavx512f -march=native -mtune=native -mprefer-vector-width=512 –funroll-loops…`
- Compiler : `gcc 9.4.0` (+ `gcc 13.2.0` on my machine)

Performance Profiling – perf

In the Firefox Profiler : https://share.firefox.dev/3T7o5qB

Main takeaways :

- 18% of the time is spent in `M_release()` (from the smart pointers of the geometry module)
- 16% of the time is spent in

`boost::math::interpolators::detail::cardinal_cubic_b_spline()` (called in PROPOSAL)

Performance Profiling – MAQAO

MAQAO profiler

Main takeaways :

- 13% of the time is spent in the libm function `f64xsubf128`
- confirms the profiling obtained with perf for the smart pointers and the interpolation with the cubic splines
- confirms that complete vectorization would have little effect on performance

Performance Profiling – MAQAO

MAQAO profiler

Subtracts a double and a long double + casts the result into a double

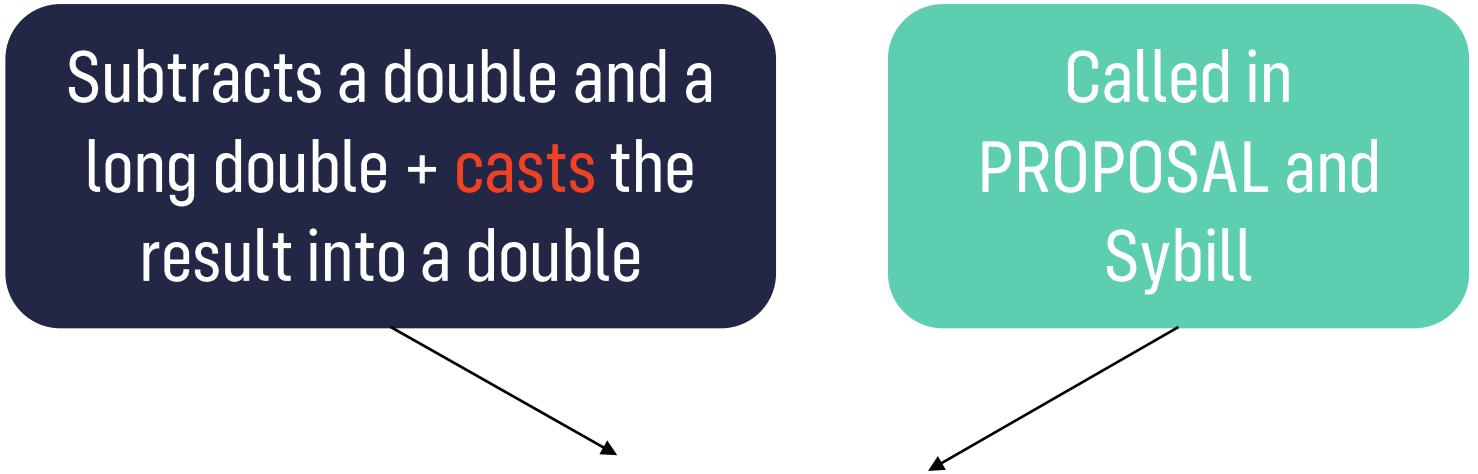
See here and here the source code

Main takeaways :

- 13% of the time is spent in the libm function `f64xsubf128`
- confirms the profiling obtained with perf for the smart pointers and the interpolation with the cubic splines
- confirms that complete vectorization would have little effect on performance

Performance Profiling – MAQAO

MAQAO profiler



Main takeaways :

- 13% of the time is spent in the libm function `f64xsubf128`
- confirms the profiling obtained with perf for the smart pointers and the interpolation with the cubic splines
- confirms that complete vectorization would have little effect on performance

Memory Profiling – perf mem

Main takeaways : Good cache use !

- ✓ L1 cache (= fastest access cache) used in more than 98% of memory operations
- ✓ Very few cache misses
- ✓ No problematic function detected

Memory Profiling – MALT

MALT memory profiler

Main takeaways :

- Lots of global variables, but mostly coming from Fortran common blocks
- ✓ Very few memory leaks (main one in FLUKA), confirms previous results from Valgrind
- ✓ No other problems detected

A few questions

- What are the characteristic dimensions of the simulation ? What are the sizes of the arrays managed by CORSIKA ?

- Are they hardcoded or determined on-the-fly ?

Suggestions for this workshop

1. Replace smart pointers from the geometry module (branch dumb_pointer has a lot of memory leaks, see here)
2. Investigate whether the use of long doubles is necessary
3. Check the use of cubic splines interpolation in PROPOSAL
4. ...Other ideas ?

THANK YOU !