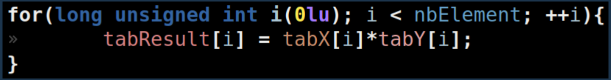
Laboratoire d'Annecy de Physique des Particules

The Hadamard Product : With NaN and Denorm

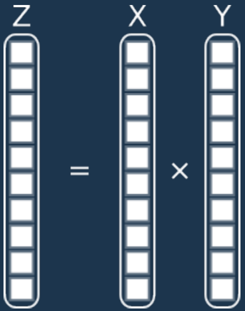
Pierre Aubert

The Hadamard product

$$z_i = x_i \times y_i, \quad \forall i \in 1, N$$



```
for(long unsigned int i(0lu); i < nbElement; ++i){
»       tabResult[i] = tabX[i]*tabY[i];
}
```



Compilation options

`https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html`

Compilation options

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

- **-O0**
  - Try to reduce compilation time, but **-Og** is better for debugging.

Compilation options

`https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html`

- **-O0**
  - Try to reduce compilation time, but **-Og** is better for debugging.
- **-O1**
  - Constant forewarding, remove dead code (never called code)...

Compilation options

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

- **-O0**
  - Try to reduce compilation time, but **-Og** is better for debugging.
- **-O1**
  - Constant forewarding, remove dead code (never called code)...
- **-O2**
  - Partial function inlining, Assume strict aliasing...

Compilation options

`https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html`

- **-O0**
  - Try to reduce compilation time, but **-Og** is better for debugging.
- **-O1**
  - Constant forewarding, remove dead code (never called code)...
- **-O2**
  - Partial function inlining, Assume strict aliasing...
- **-O3**
  - More function inlining, loop unrolling, partial vectorization...

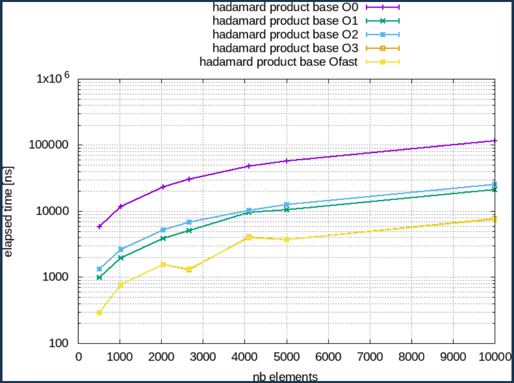
Compilation options

`https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html`

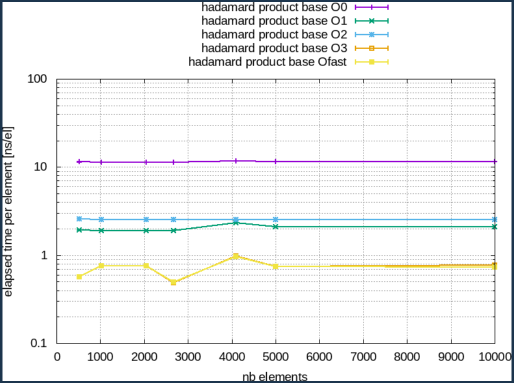
- **-O0**
  - Try to reduce compilation time, but **-Og** is better for debugging.
- **-O1**
  - Constant forewarding, remove dead code (never called code)...
- **-O2**
  - Partial function inlining, Assume strict aliasing...
- **-O3**
  - More function inlining, loop unrolling, partial vectorization...
- **-Ofast**
  - Disregard strict standards compliance. Enable **-ffast-math**, stack size is hardcoded to 32 768 bytes (borrowed from **gfortran**). **Possibily degrades the computation accuracy.**

The Hadamard product : reference Performances

Total Elapsed Time (cy)



Elapsed Time per element (cy/el)

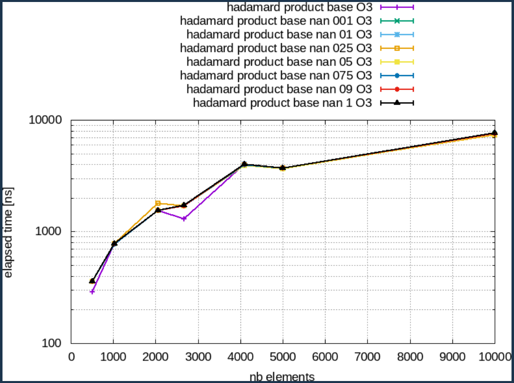


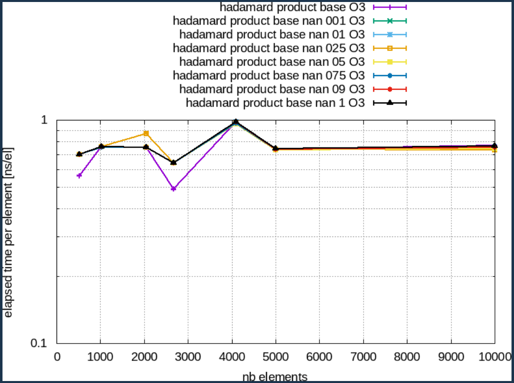
Speed up of **14** between -**O0** and -**O3** or -**Ofast**

The Hadamard product : NaN Performance

Total Elapsed Time (cy)

Elapsed Time per element (cy/el)

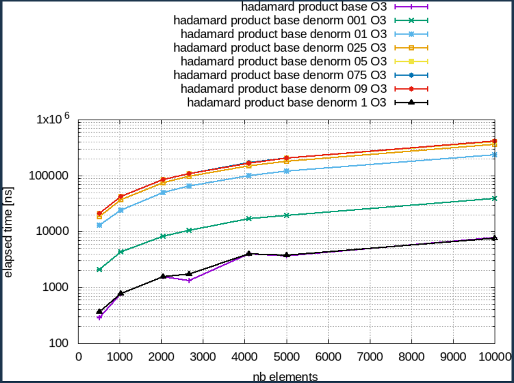




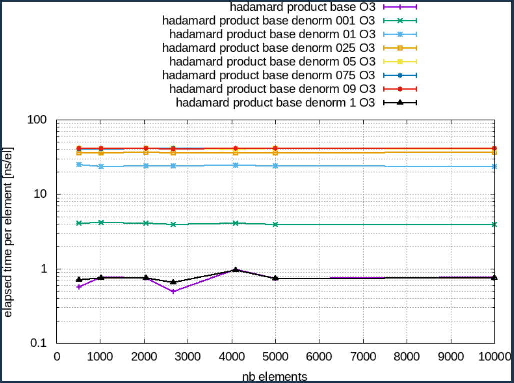
Same performances in **-O3**

The Hadamard product : Denorm Perf

Total Elapsed Time (cy)



Elapsed Time per element (cy/el)

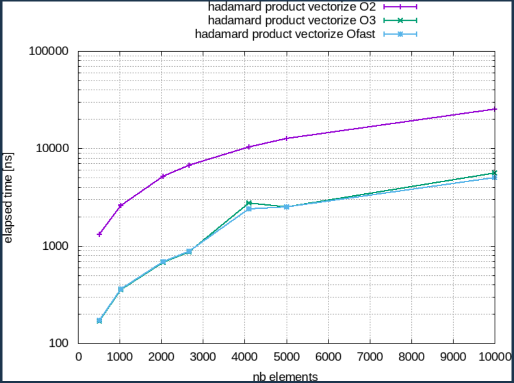


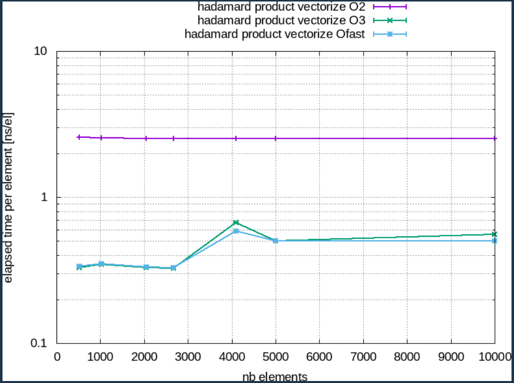
High impact on performances in **-O3**

The Hadamard product : vectorized Performances

Total Elapsed Time (cy)

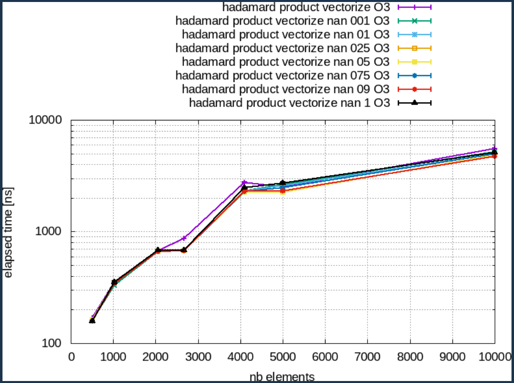
Elapsed Time per element (cy/el)



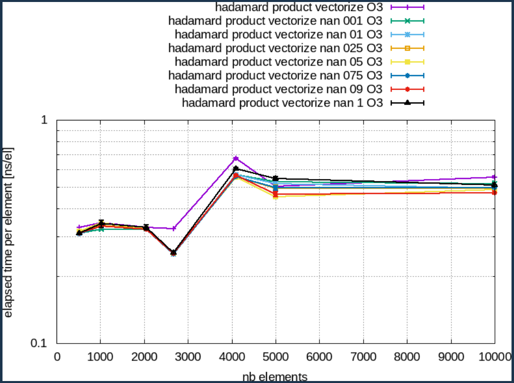


The Hadamard product : vectorized NaN Perf

Total Elapsed Time (cy)



Elapsed Time per element (cy/el)

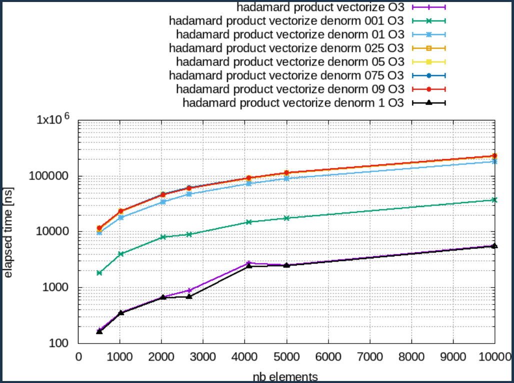


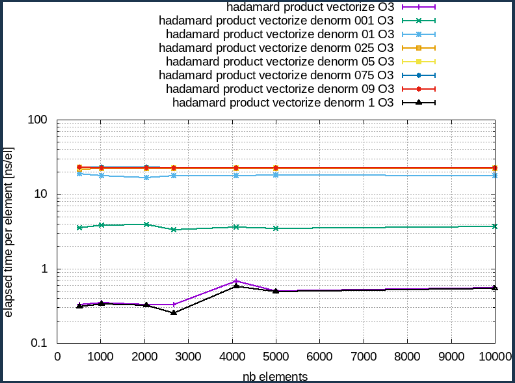
Same performances in **-O3**

The Hadamard product : vectorized Denorm Perf

Total Elapsed Time (cy)

Elapsed Time per element (cy/el)

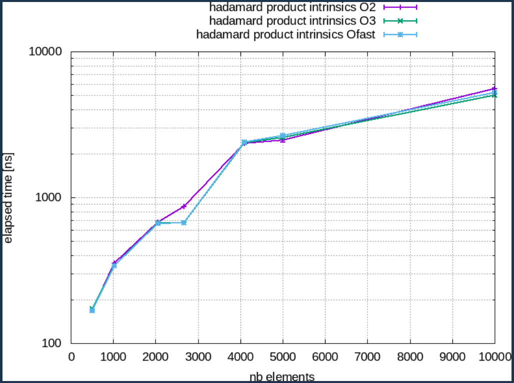




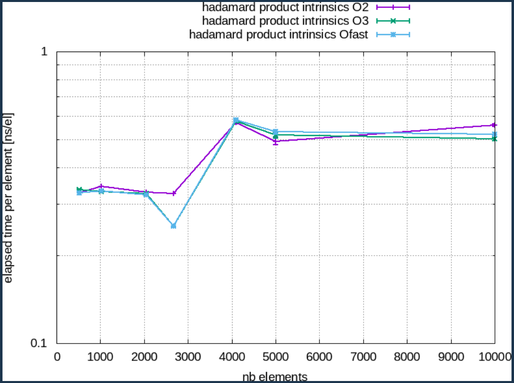
High impact on performances in **-O3**

The Hadamard product : intrinsics Performances

Total Elapsed Time (cy)

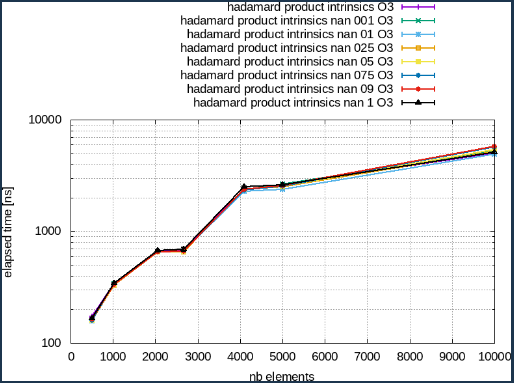


Elapsed Time per element (cy/el)

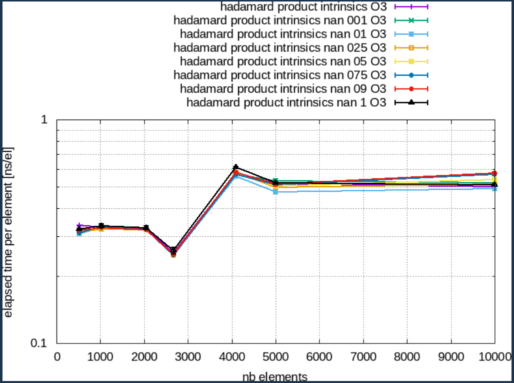


The Hadamard product : intrinsics NaN Perf

Total Elapsed Time (cy)



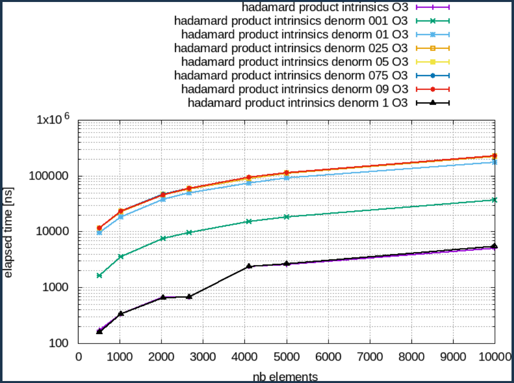
Elapsed Time per element (cy/el)



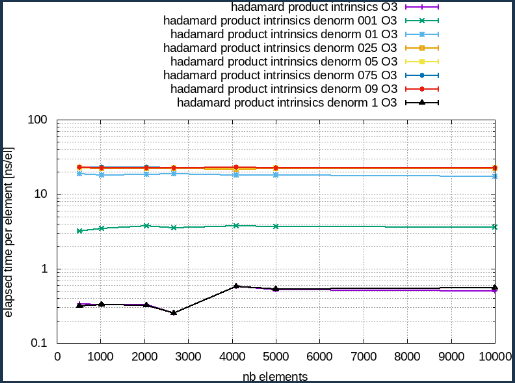
Same performances in -**O3**

The Hadamard product : intrinsics Denorm Perf

Total Elapsed Time (cy)



Elapsed Time per element (cy/el)



High impact on performances in -**O3**

The Hadamard product : Performances summary

NaN values do not slow down the performances

Denormalised values affect a lot the performances :

- 1% : slow down computation by ~ 5.7
- 10% : slow down computation by ~ 31
- $50 - 90\%$: slow down computation by ~ 57
- 100% : same performance as 0%

Denormalised values affect a lot the intrinsics performances :

- 1% : slow down computation by ~ 13
- 10% : slow down computation by ~ 53
- $50 - 90\%$: slow down computation by ~ 22
- 100% : same performance as 0%

But can we solve this problem ?