LAPP
Laboratoire d'Annecy de Physique des Particules

Optimisation : Reduction

Pierre Aubert

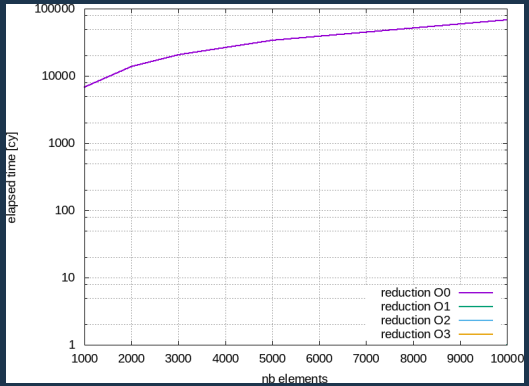
The Reduction (sum)

$$\alpha = \sum_{i=1}^{N} x_i$$

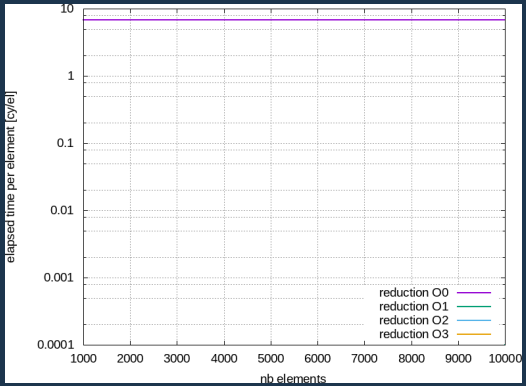

```
float reduction(const float * tabValue, long unsigned int nbElement){
»       float res(0.0f);
»       for(long unsigned int i(0lu); i < nbElement; ++i){
»       »       res += tabValue[i];
»       }
»       return res;
}
```

The reduction : first performances

Total Elapsed Time (cy)



Elapsed Time per element (cy/el)



The Performance : what is the issue ?

- Performances **-O0** : slow but reasonable
- Other performances (**-O1**, **-O2**, **-O3**, **-Ofast**) are too fast (non sence)

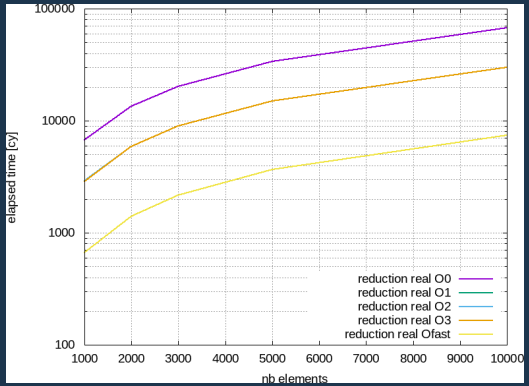
GCC is smart of guileful depending on the points of view.

- GCC noticed you **do not** use the result of the **reduction** function.
- The call to **reduction** is considered as dead code (or never called code).

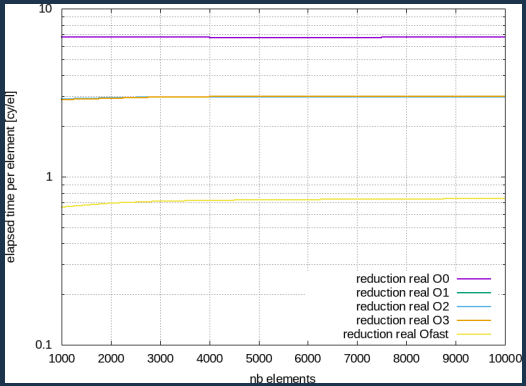
To avoid that, you have to compile the **reduction** function in an other file.

The reduction : real performances

Total Elapsed Time (cy)

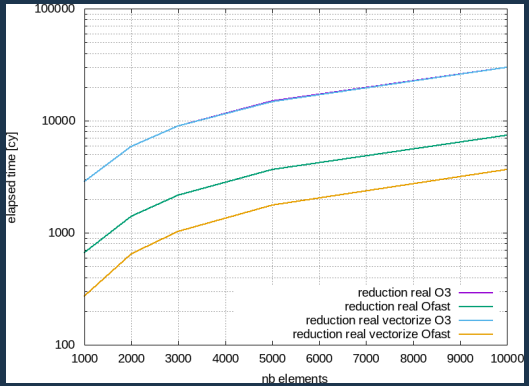


Elapsed Time per element (cy/el)

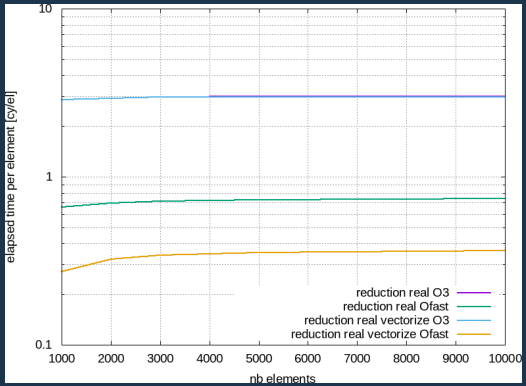


The reduction : vectorize performances

Total Elapsed Time (cy)

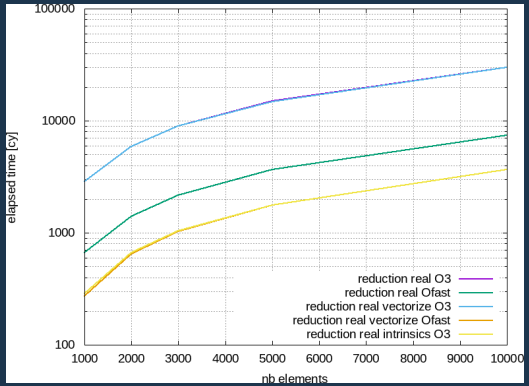


Elapsed Time per element (cy/el)

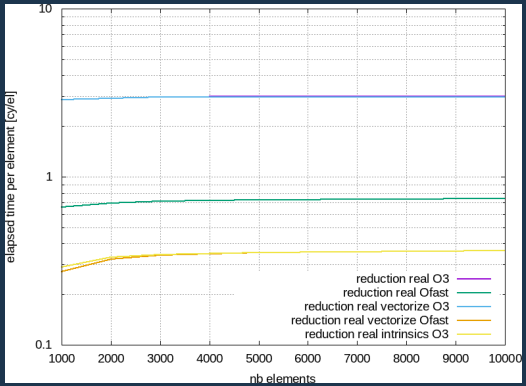


The reduction : intrinsics performances

Total Elapsed Time (cy)

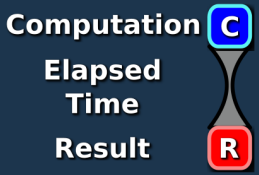


Elapsed Time per element (cy/el)

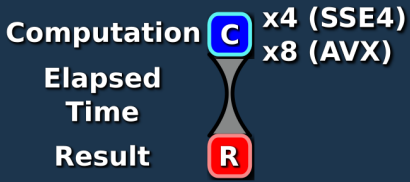


Reduction optimisation

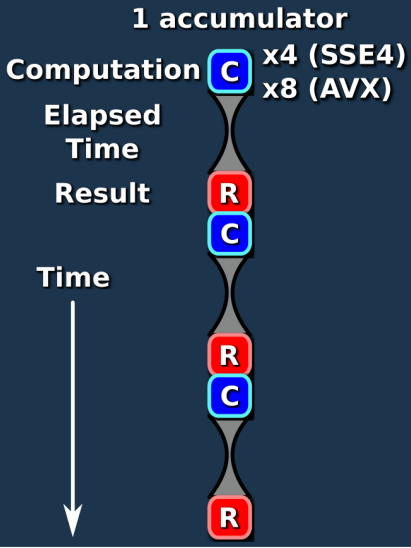
Reduction optimisation



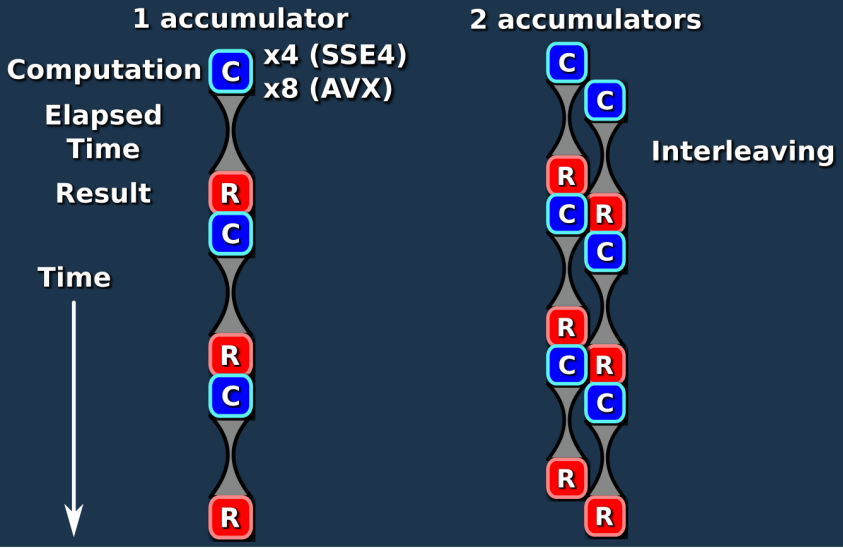
Reduction optimisation



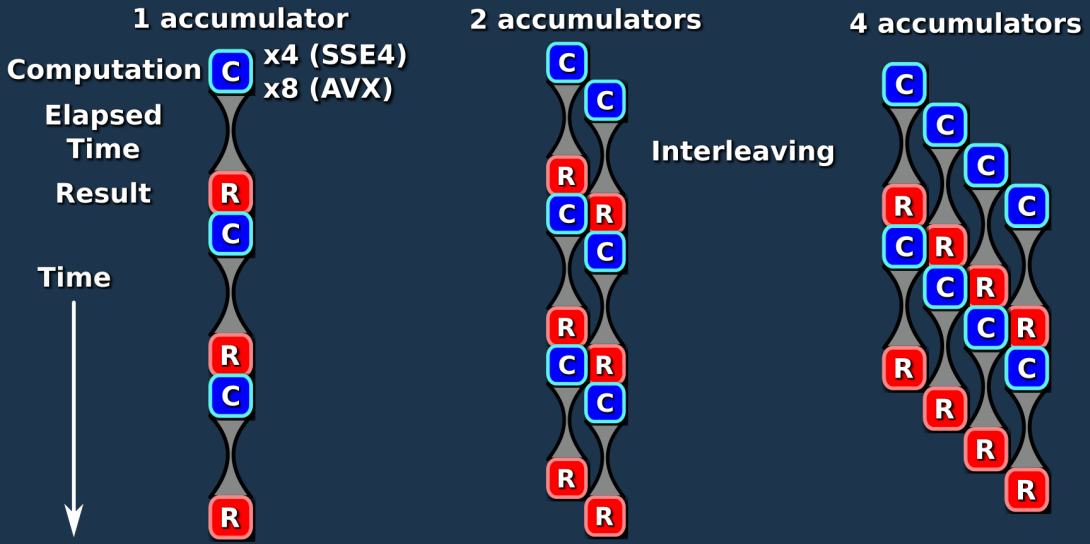
Reduction optimisation



Reduction optimisation

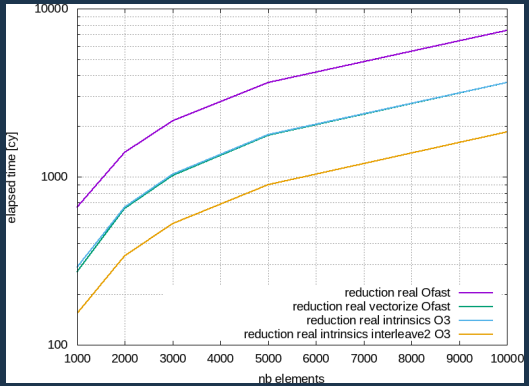


Reduction optimisation

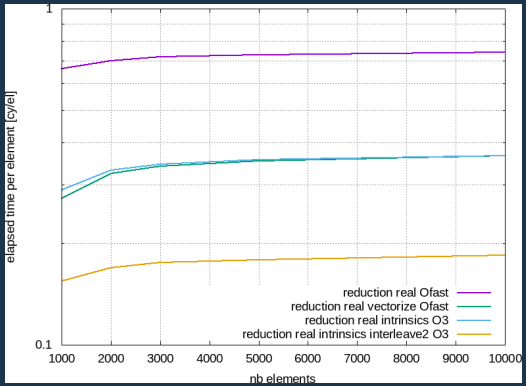


The reduction : intrinsics interleaved 2

Total Elapsed Time (cy)

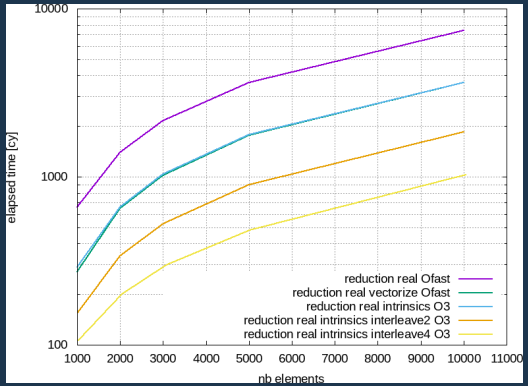


Elapsed Time per element (cy/el)

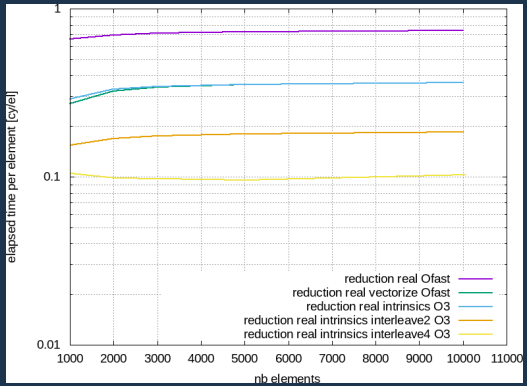


The reduction : intrinsics interleaved 4

Total Elapsed Time (cy)

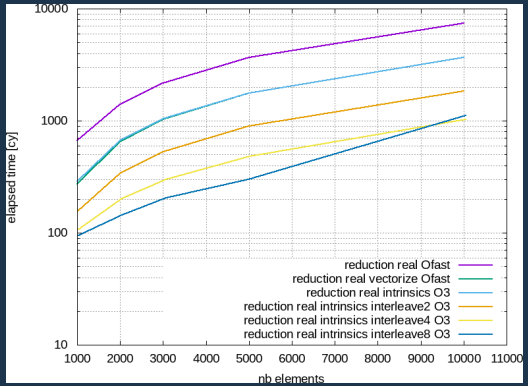


Elapsed Time per element (cy/el)

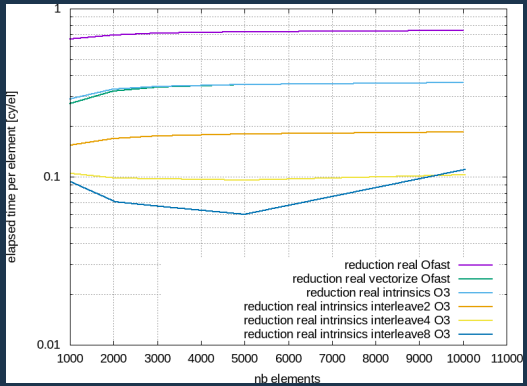


The reduction : intrinsics interleaved 8

Total Elapsed Time (cy)

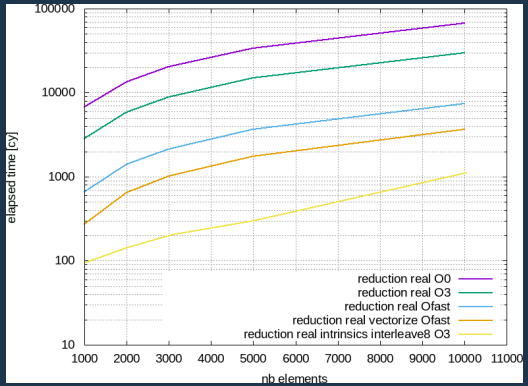


Elapsed Time per element (cy/el)

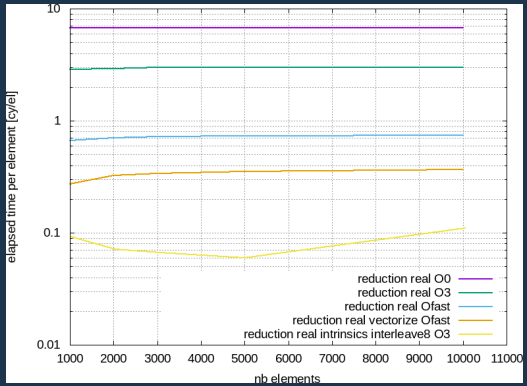


The reduction : summary

Total Elapsed Time (cy)



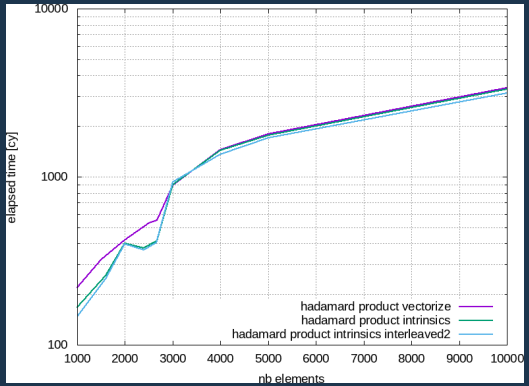
Elapsed Time per element (cy/el)



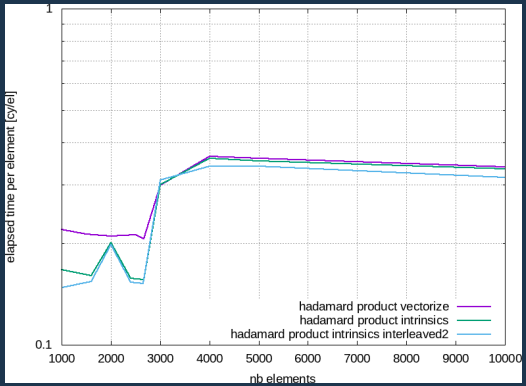
5000 elements, Intrinsics is **166** times faster than -**O0** and 7 times faster than -**Ofast** vectorized

What about the Hadamard product ?

Total Elapsed Time (cy)

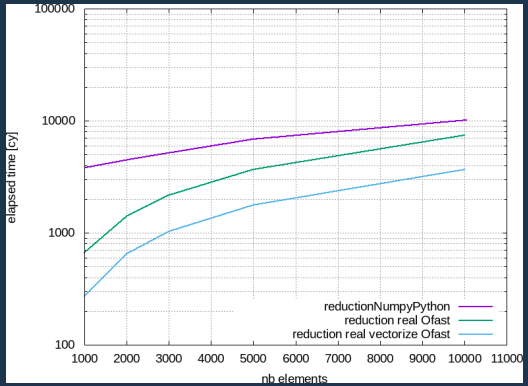


Elapsed Time per element (cy/el)

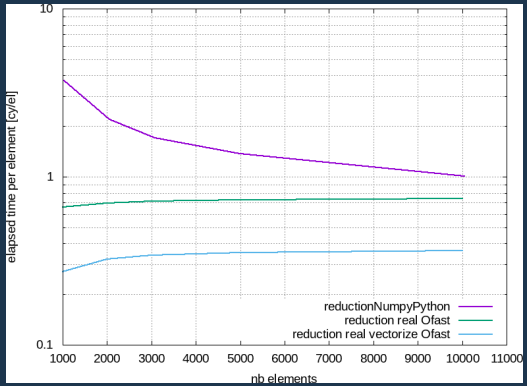


The reduction : Python

Total Elapsed Time (cy)



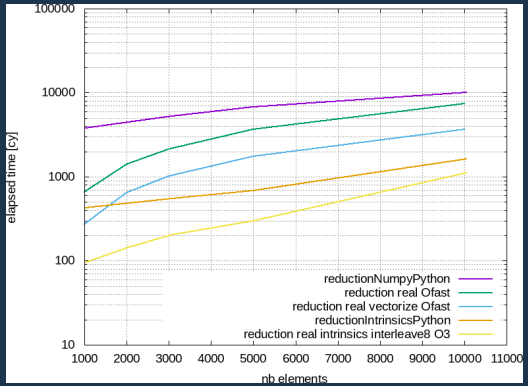
Elapsed Time per element (cy/el)



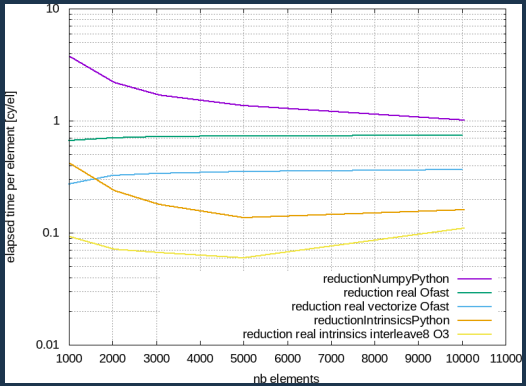
1000 elements, GCC vectorized version is **13** times faster than **numpy** sum

The reduction : Python Summary

Total Elapsed Time (cy)



Elapsed Time per element (cy/el)



1000 elements, our python reduction is **10** times faster than **numpy** sum