LAPP
Laboratoire d'Annecy de Physique des Particules

Small Matrix Multiply Optimisation (2x2)

Pierre Aubert

Asterics
Astronomy ESFRI & Research Infrastructure Cluster

cta
cherenkov telescope array

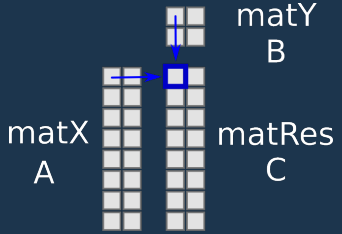
LISTIC

cnrs

UNIVERSITÉ SAVOIE MONT BLANC

The 2x2 SGEMM

$$C_{i,j} = \sum_{k=1}^{2} A_{i,k} \cdot B_{k,j}, \quad \forall i \in 1, N, \quad j \in [1,2]$$




```
void sgemm(float* matOut, const float * matX, const float* matY,
           long unsigned int nbRowOutX, long unsigned int nbColOutY, long unsigned int nbColXRowY)
{
	for(long unsigned int i(0lu); i < nbRowOutX; ++i){
		for(long unsigned int j(0lu); j < nbColOutY; ++j){
			float res(0.0f);
			for(long unsigned int k(0lu); k < nbColXRowY; ++k){
				res += matX[i*nbColXRowY + k]*matY[k*nbColOutY + j];
			}
			matOut[i*nbColOutY + j] = res;
		}
	}
}
```

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...

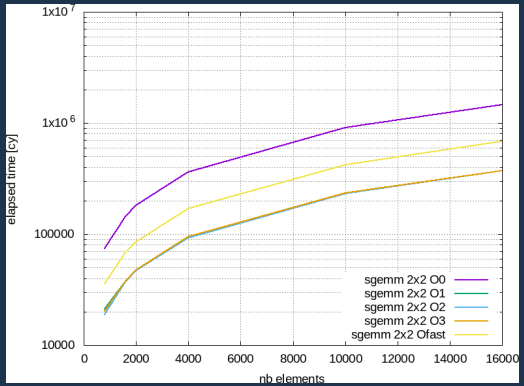
Compilation options

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

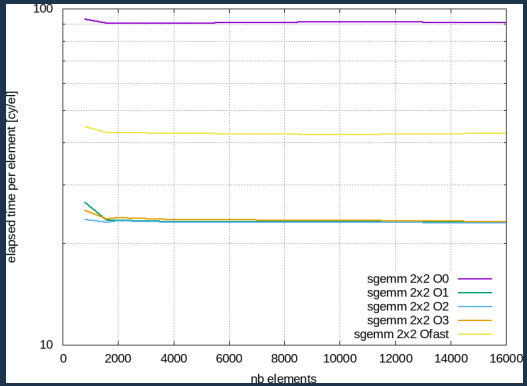
- **-O0**
  - Try to reduce compilation time, but **-Og** is better for debugging.
- **-O1**
  - Constant forewarding, remove dead code (never called code)...
- **-O2**
  - Partial function inlining, Assume strict aliasing...
- **-O3**
  - More function inlining, loop unrolling, partial vectorization...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

The SGEMM : Performance

Total Elapsed Time (cy)



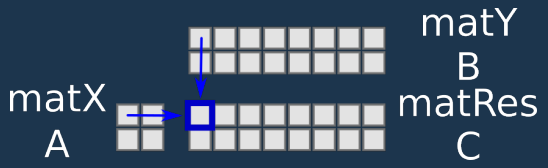
Elapsed Time per element (cy/el)



Speed up of > 3 between **-O0** and **-O3**

The 2x2 SGEMM Transposed

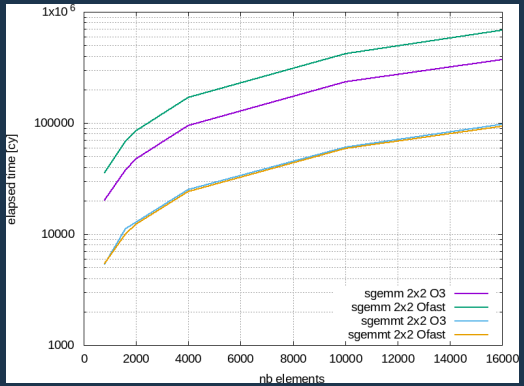
$$C_{i,j} = \sum_{k=1}^{2} A_{i,k} \cdot B_{k,j}, \quad \forall i \in [1,2], \quad j \in 1,N$$



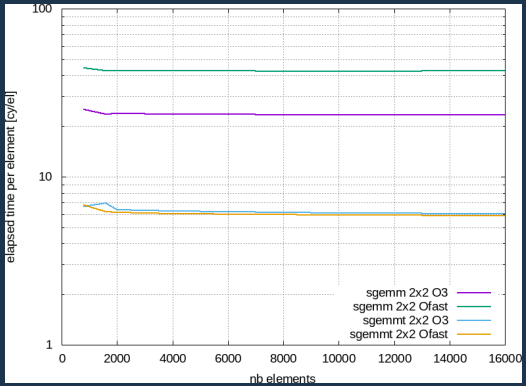

```
void sgemm(float* matOut, const float * matX, const float* matY,
           long unsigned int nbRowOutX, long unsigned int nbColOutY, long unsigned int nbColXRowY)
{
        for(long unsigned int i(0lu); i < nbRowOutX; ++i){
                for(long unsigned int j(0lu); j < nbColOutY; ++j){
                        float res(0.0f);
                        for(long unsigned int k(0lu); k < nbColXRowY; ++k){
                                res += matX[i*nbColXRowY + k]*matY[k*nbColOutY + j];
                        }
                        matOut[i*nbColOutY + j] = res;
                }
        }
}
```

The SGEMM Transposed : Performance

Total Elapsed Time (cy)



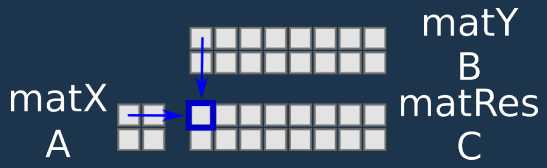
Elapsed Time per element (cy/el)



Speed up of $> \mathbf{3}$ between **-O3** naive and transposed.

The 2x2 SGEMM Expanded

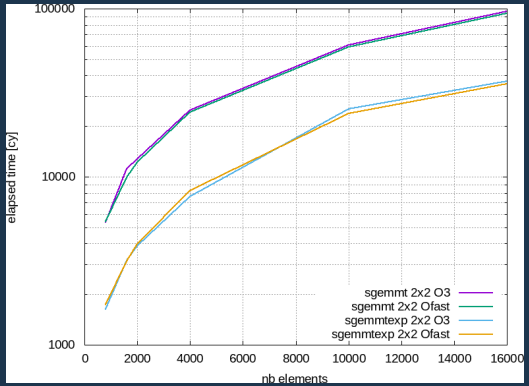
$$C_{i,j} = \sum_{k=1}^{2} A_{i,k} \cdot B_{k,j}, \quad \forall i \in [1,2], \quad j \in 1,N$$



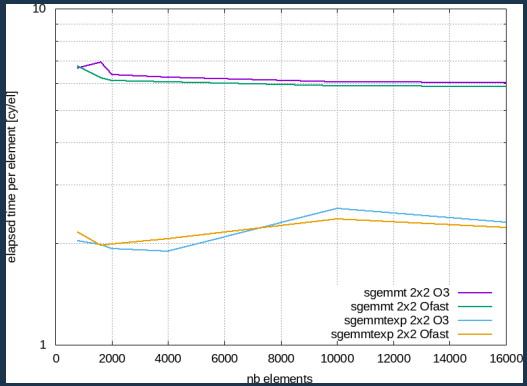

```
void sgemm2x2(float* matOut, const float * mat2x2, const float* matY, long unsigned int nbColOutY){
        const float* matY2 = matY + nbColOutY;
        float* matOut2 = matOut + nbColOutY;
        for(long unsigned int i(0lu); i < nbColOutY; ++i){
                matOut[i] = mat2x2[0]*matY[i] + mat2x2[1]*matY2[i];
                matOut2[i] = mat2x2[2]*matY[i] + mat2x2[3]*matY2[i];
        }
}
```

The SGEMM Expanded : Performance

Total Elapsed Time (cy)



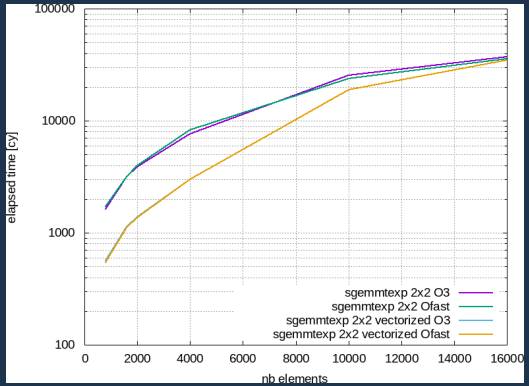
Elapsed Time per element (cy/el)



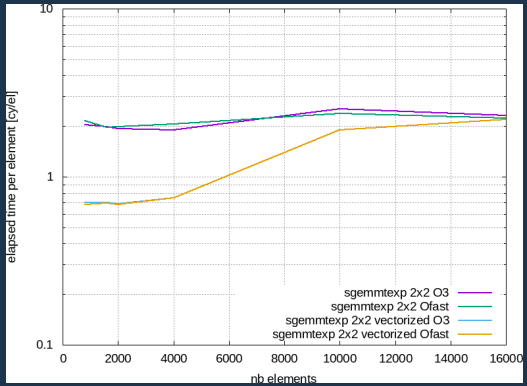
At 2000 elements : speed up of **10** between transposed and expanded.

The SGEMM Vectorized : Performance

Total Elapsed Time (cy)

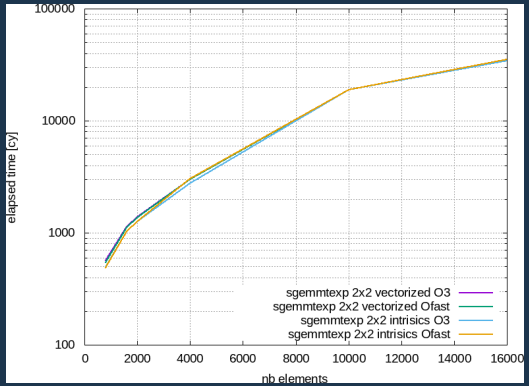


Elapsed Time per element (cy/el)

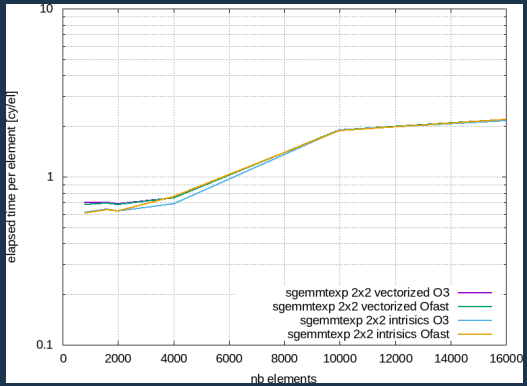


The SGEMM Intrinsics : Performance

Total Elapsed Time (cy)

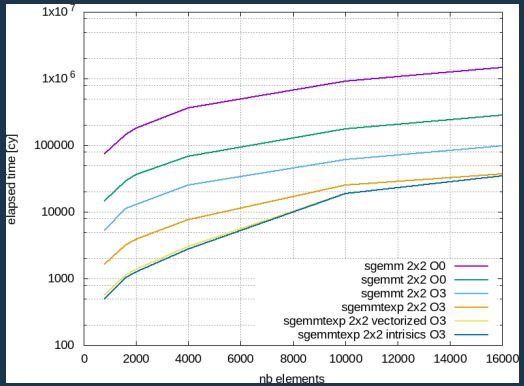
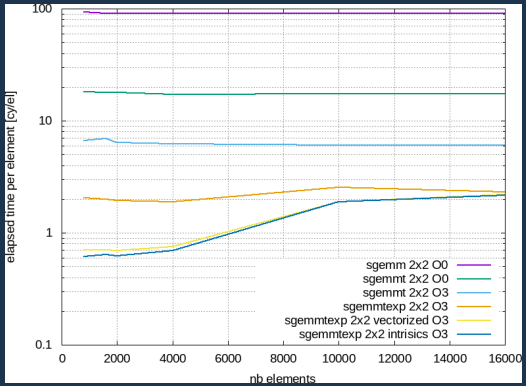


Elapsed Time per element (cy/el)



The SGEMM Performance summary

Total Elapsed Time (cy)



Elapsed Time per element (cy/el)