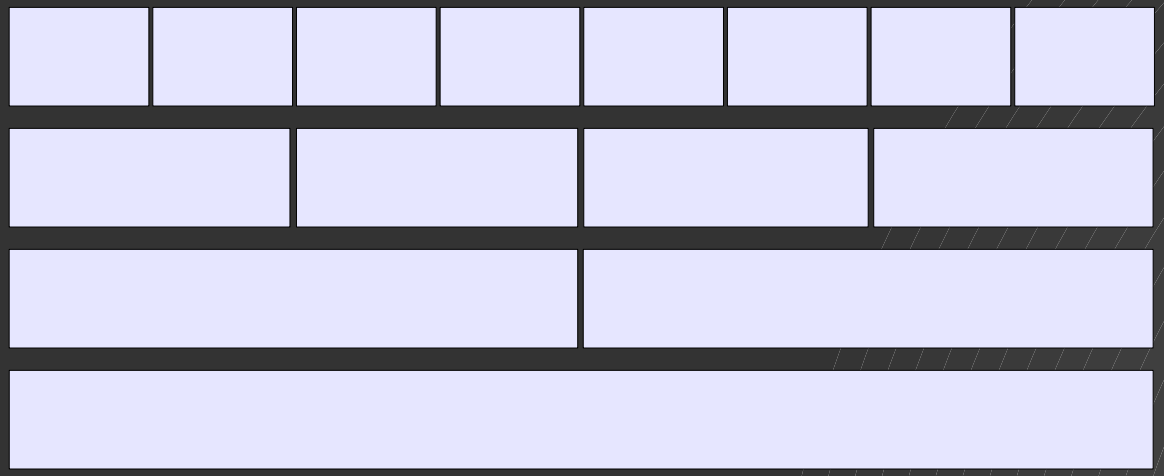
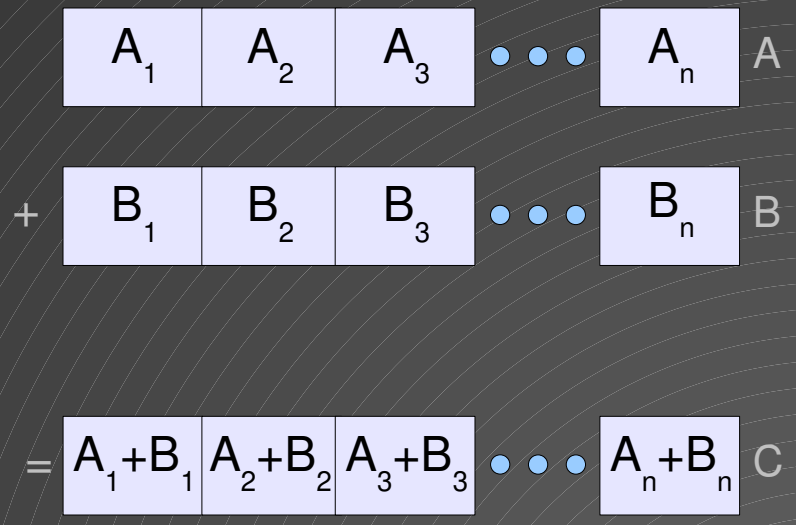
Implémentation sur CPU

- Implémentation sur CPU
  - Utilisation du parallélisme
  - Utilisation des instructions SIMD
    - Auto-vectorisation
    - Intrinsics SIMD
    - Boost::simd
  - ...

CPU : SIMD

- SIMD
  - Single instruction multiple data
    - MMX, SSE, AVX, NEON...
    - Registres sous forme de vecteurs





  - Bien adapté à l'image.

CPU : SIMD

- SIMD : un peu d'histoire
  - 1997 jeu d’instruction MMX sur P166 (intel) *L’ordinateur multimedia* - Registres 64bits partagés avec le FPU.
  - 1997 jeu d’instruction - 3DNow ! (AMD)
  - 1999 jeu d’instruction SSE - Registres 128bits Registres
  - SSE2, SSE3, SSE4 - Registres 128bits
  - AVX - Registres 256bits
  - AVX512 - Registres 512bits
  - Neon sur ARM

CPU : SIMD

- SIMD Usage
  - cat /proc/cpuinfo
    - flags : 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 arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx rdseed adx smap clflushopt
  - Par le compilateur :
    - Activé sour GCC avec l'option -ftree-vectorize (par default activé avec -O3)
    - Renseigner l'option -march= (corei7, native...)
    - On peut avoir plus d'info avec les options -fopt-info-vec-* (-fopt-info-vec-optimized -fopt-info-vec-missed)
  - Sorties :
    - knn.cpp:229: note: LOOP VECTORIZED.
      (Attention : plusieurs passes.)

CPU : SIMD

- Auto-vectorisation

```
for (std::size_t x=0 ; x<l ; ++x) {
        output_image[x] = input_image1[x] + input_image2[x];
} // en complet désaccord avec votre coding style.
```

- Entrées :
  - -Wall -O3 -g -Wextra -Werror -m64 -march=native *-ftree-vectorize* -std=c++11 -fopt-info-vec-optimized #-fopt-info-vec-missed
- Sorties :

```
simd_add.cpp:35:5: note: loop vectorized
simd_add.cpp:35:5: note: loop versioned for vectorization because of possible aliasing
__restrict__
simd_add.cpp:35:5: note: loop peeled for vectorization to enhance alignment
typedef unsigned char auchar __attribute__ ((__aligned__(16)));
```

CPU : SIMD

- Par le compilateur (auto-vectorisation)
  - pas toujours facile
  - s’assurer que les données soient alignées (quasi Impossible en cpp :-( )
    - __attribute__((aligned(TL_IMAGE_ALIGNMENT)))
    - std : :align
    - Alignas(.)
    - aligned_alloc
  - __restrict__
  - assume dans ICC

CPU : SIMD

- Par le compilateur (auto-vectorisation), bonnes pratiques :
  - Utiliser des indices plutôt que des pointeurs
  - Array of Structures vs Strucure of Arrays
  - Ne pas interrompre une boucle

```
for( k=0 ; k < size_vect ; k++) {
        double t = v_example [ k ] – data[k_data ++];
        res += t  *  t ;
        // if ( res >tresh ) {
        //      break ;
        // }
}
res  * = –g ;
return exp ( res );
```

CPU : SIMD

- SIMD - intrinsics
  - Possible de les utiliser en c/c++
- Exemple :

```
  for (std::size_t x=0 ; x<l ; x+=16) {
      __m128i v_input_image1 = _mm_loadu_si128((const __m128i*)(input_image1 +
x));
      __m128i v_input_image2 = _mm_loadu_si128((const __m128i*)(input_image2 +
x));
      __m128i v_output1 = _mm_add_epi8(v_input_image1, v_input_image2);
      _mm_storeu_si128((__m128i*)(output_image+x), v_output1);
  }
```

  - Problème d'alignement des données
  - Difficilement portable

CPU : SIMD

- SIMD - intrinsics
  - Possible de les utiliser en c/c++
- Exemple :

```
for (std::size_t x=0 ; x<l ; x+=16) {
    __m128i v_input_image1 = _mm_load_si128((const __m128i*)(input_image1 + x));
    __m128i v_input_image2 = _mm_load_si128((const __m128i*)(input_image2 + x));
    __m128i v_output1 = _mm_add_epi8(v_input_image1, v_input_image2);
    _mm_store_si128((__m128i*)(output_image+x), v_output1);
}
```

  - Problème d'alignement des données

    aligned_alloc

  - Difficilement portable

Implémentation sur CPU

- SIMD

	Min	Mean	Max
no_vect	5.913e-06	0.0120603	0.052094
unaligned_simd	4.83e-07	0.00363181	0.0142756
aligned_simd	4.99e-07	0.00364205	0.0145116

Implémentation sur CPU

- Auto-vectorisation

	Min	Mean	Max
no_vect	5.913e-06	0.0120603	0.052094
unaligned_simd	4.83e-07	0.00363181	0.0142756
aligned_simd	4.99e-07	0.00364205	0.0145116
auto_vect	4.83e-07	0.00364648	0.014371

CPU : SIMD

- Function Multiversioning (GCC 4.8) - https ://gcc.gnu.org/wiki/FunctionMultiVersioning

```
__attribute__ ((target ("default")))
int foo ()
{
    // The default version of foo.
}
__attribute__ ((target ("sse4.2")))
int foo ()
{
    // foo version for SSE4.2
}
__attribute__ ((target ("arch=atom")))
int foo ()
{
    // foo version for the Intel ATOM processor
}
```

Implémentation sur CPU

- Boost::simd

Implémentation sur GPU

Implémentation sur GPU

- Implémentation sur GPU
  - Cuda
  - OpenCL
  - Compute Shaders
  - ...

Implémentation sur GPU

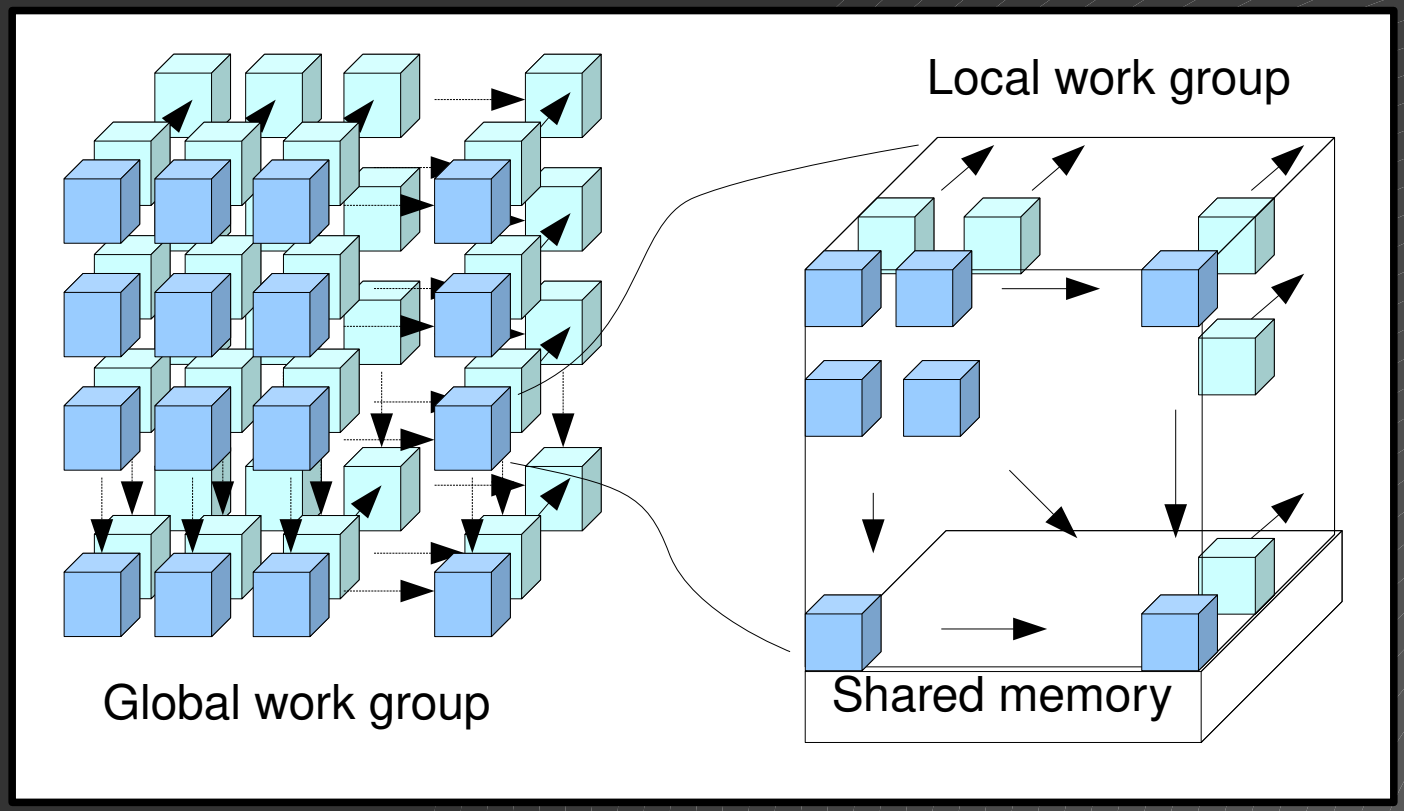
- Cuda

Implémentation sur GPU

- OpenCL

Implémentation sur GPU

- Compute Shaders
  - Glsl « portable » ou autre



Conclusion

Cours 06

Fin