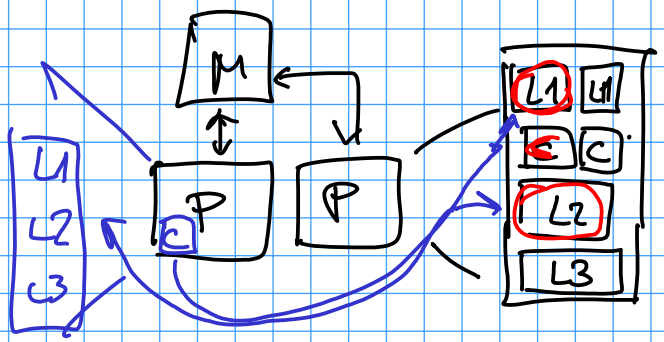
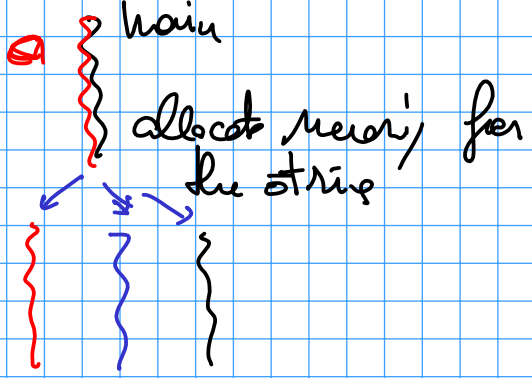
NUMA non uniform memory access (SMP)

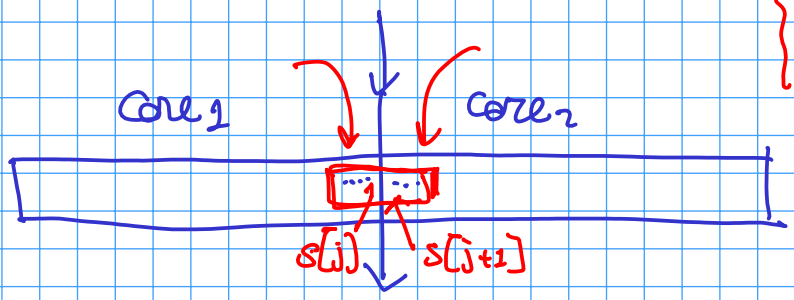


IBM CELL

main()

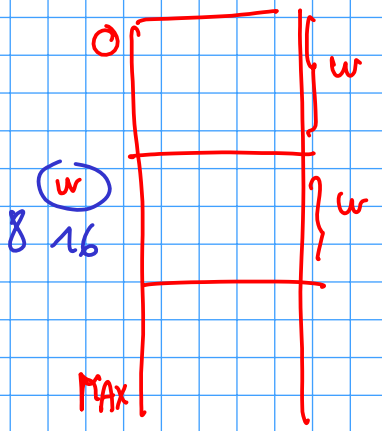
string text = ...

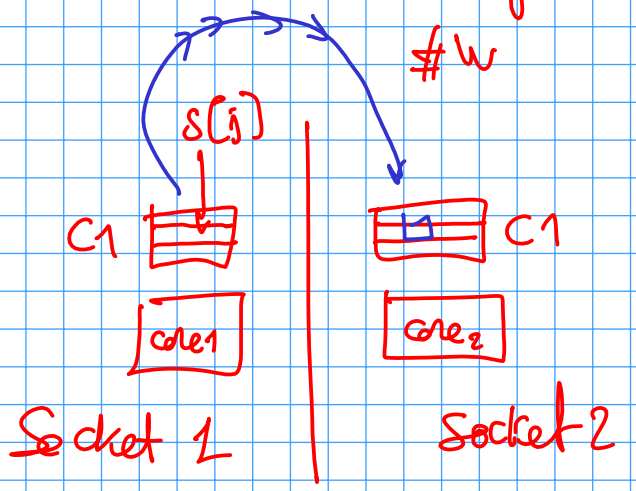




line of cache

#w



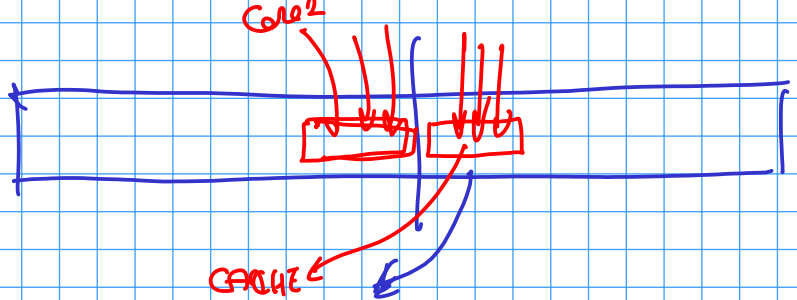


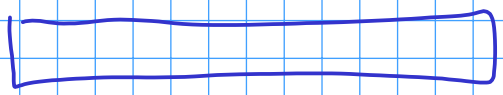
false sharing

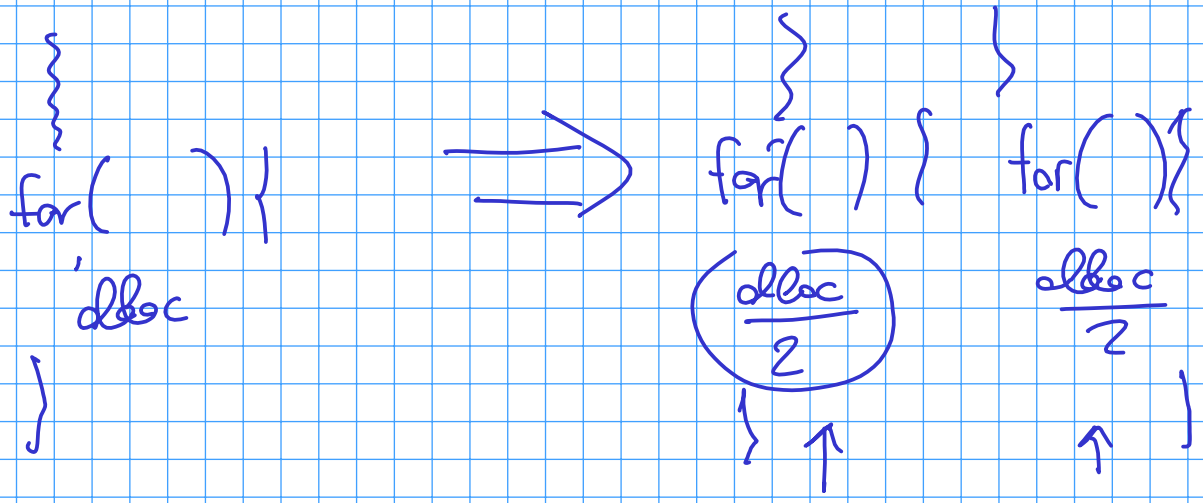
NUMA

align directive

__align(x, -)

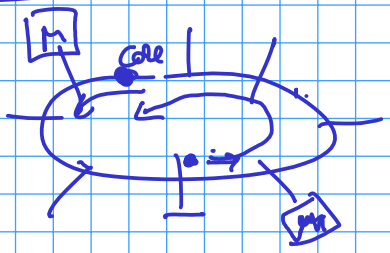




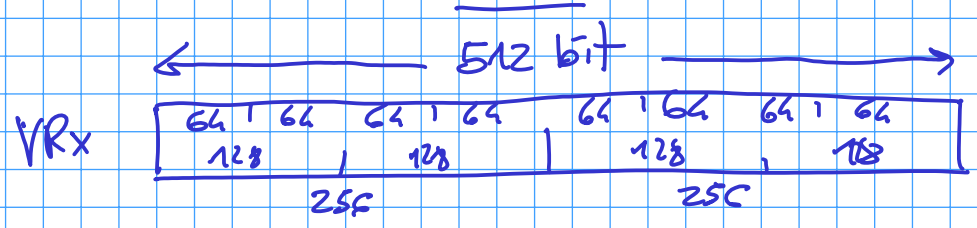


PHI

8 interfaces ≡ 8 bonds



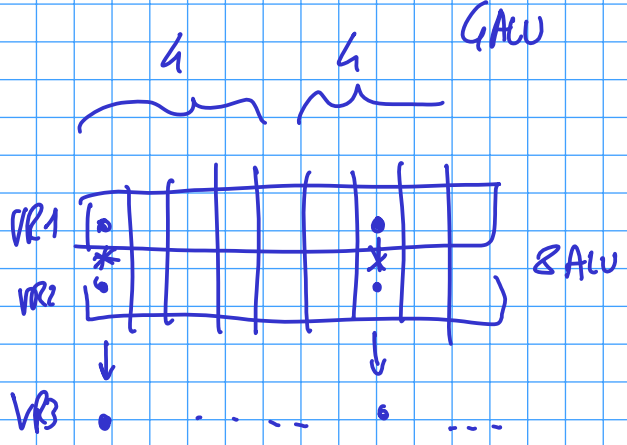
Vector unit VFPU



8 x 64bit registers

LOAD ..... , VR_5

MUL64 VR_1, VR_2, VR_3



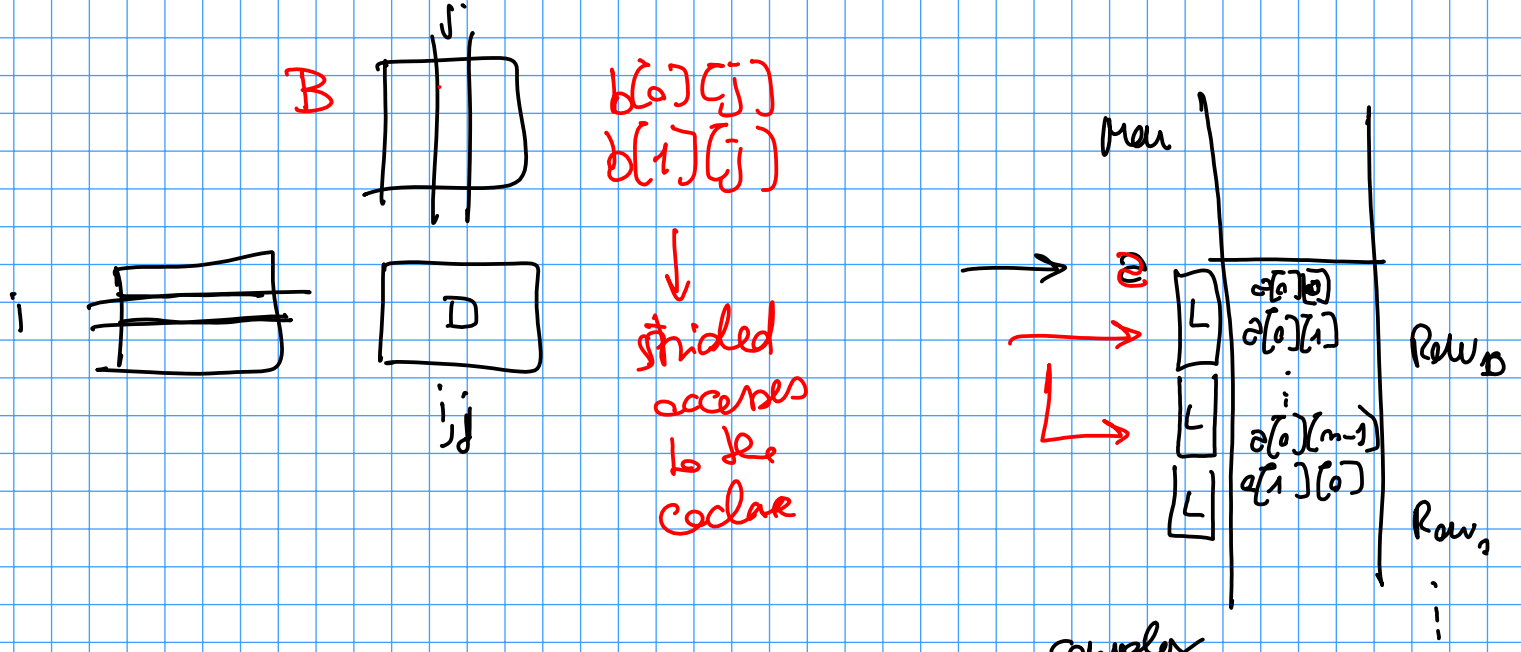
x86

SSE | SSE 2 | SSE4

AVX

duce
- VLOAD
- VLOAD
- VMUL
- VSTOR

8 diverse
- LD ...
- LD ...
- MUL ...
- ST
- INC ..
- IF ...



complex

Vectorize

avoid to use expression

```
for(int i=0; i<N; i++) {
    x[i]
}
```

i = f()

No dependencies

#pragma ivdep
for( ) { }

avoid usage of library calls

x[i] = f(x[i-1])

lib ? g(x, y) = ...

x[i] = g(x[i], i)

inline float f(...) { ... }

main

```
for( ) {
    x[i] = f(x[i])
}
```

Jacobi iterative

N×N N N

A b x $Ax = b$

$$x_i = \frac{\left(b_i - \sum_{\substack{k=0 \\ k \neq i}}^{n} x_i b_{ik}\right)}{a_{ii}}$$

if $(j != i)$ acc = acc − (....)

main

generate n×n A matrix random

n b vector "

n x vector random

a single iteration of the Jacobi

$$(x_i) = f(A, b, x_{(i-1)})$$

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.$$

```
for( i=0 ; i< N , i++)
    x[i]

for ( auto &t : x ) {
    t
```

source /home/spm1501/intel18/compiler_and_libraries/linux/bin/compilervars.sh intel64

Vtune amplifier

source /home/spm1501/intel18/vtune_amplifier/amplxe-vars.sh

amplxe-cl — command line

amplxe-gui — grafic interface