RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

ONERA
THE FRENCH AEROSPACE LAB

OS 202 (groupe 2)

Jean-Didier Garaud

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

ONERA
THE FRENCH AEROSPACE LAB

TP1 14/01/2025

Produit Matrice-Matrice en OpenMP

Mémoire cache

Premiers pas en MPI

Algèbre linéaire avec Eigen ou Blas

- https://eigen.tuxfamily.org/dox/TopicUsingBlasLapack.html

Functional domain	Code example	BLAS/LAPACK routines
Matrix-matrix oper… EIGEN_USE_BLAS		
Matrix-vector oper… EIGEN_USE_BLAS		
LU decomposition EIGEN_USE_LAPA… EIGEN_USE_LAPA…		
Cholesky decomposition EIGEN_USE_LAPACKE EIGEN_USE_LAPACKE_STRICT	`v1 = m2.selfadjointView<Upper>().llt().solve(v2);`	?potrf
QR decomposition EIGEN_USE_LAPACKE EIGEN_USE_LAPACKE_STRICT	`m1.householderQr();` `m1.colPivHouseholderQr();`	?geqrf ?geqp3
Singular value decomposition EIGEN_USE_LAPACKE	`JacobiSVD<MatrixXd, ComputeThinV> svd;` `svd.compute(m1);`	?gesvd

Eigen-value decompositions EIGEN_USE_LAPACKE EIGEN_USE_LAPACKE_STRICT	`EigenSolver<MatrixXd> es(m1);`
	`ComplexEigenSolver<MatrixXcd> ces(m1);`
	`SelfAdjointEigenSolver<MatrixXd> saes(m1+m1.transpose());`
	`GeneralizedSelfAdjointEigenSolver<MatrixXd>`
	`gsaes(m1+m1.transpose(),m2+m2.transpose());`
		?gees
		?gees
		?syev/?heev
		?syev/?heev,
		?potrf

Schur decomposition EIGEN_USE_LAPACKE EIGEN_USE_LAPACKE_STRICT	`RealSchur<MatrixXd> schurR(m1);` `ComplexSchur<MatrixXcd> schurC(m1);`	?gees

Eigen::Vector<double, size> u(sz);
Eigen::Matrix<double, sizes…> A;
u << … ; A << … ;
f = A.dot(u);

```
#include <mkl.h>
#include <iostream>

int main() {
    // Définition des dimensions des matrices
    int rows_A = 2;
    int cols_A = 3;
    int rows_B = 3;
    int cols_B = 2;

    // Allocation de mémoire pour les matrices
    double* A = (double*)mkl_malloc(rows_A * cols_A * sizeof(double), 64);
    double* B = (double*)mkl_malloc(rows_B * cols_B * sizeof(double), 64);
    double* C = (double*)mkl_malloc(rows_A * cols_B * sizeof(double), 64);

    // Initialisation des matrices
    A[0] = 1.0; A[1] = 2.0; A[2] = 3.0;
    A[3] = 4.0; A[4] = 5.0; A[5] = 6.0;

    B[0] = 7.0; B[1] = 8.0;
    B[2] = 9.0; B[3] = 10.0;
    B[4] = 11.0; B[5] = 12.0;

    // Multiplication de matrices avec MKL
    double alpha = 1.0;
    double beta = 0.0;  // Initialisation de C à zéro

    cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans,
                rows_A, cols_B, cols_A,
                alpha, A, cols_A,
                B, cols_B,

    // Libération de mémoire
    mkl_free(A);
    mkl_free(B);
    mkl_free(C);

    return 0;
}
```

cblas_dgemm(options…, sizes…, pointers…)

g++ mkl_test.cpp -lmkl_rt

« Rappels »

- 2 modes de compilation
  - Debug : **`-g -O0  -fbounds-check -D_GLIBCXX_DEBUG`**
    programme lent, mais facile à debugger (assert actifs, gdb)
  - Optimisé/Release : **`-O3 -march=native -DNDEBUG`**
    programme rapide (au top de votre compilateur et de votre processeur)

  - avec clang ou compilateur Intel, les options ont parfois d'autres noms **`(-Ofast`** pour IntelCXX)

- Avec nos Makefile, c'est assez explicite (`make help`)
- Avec CMAKE : **`-DCMAKE_BUILD_TYPE=Release`** (ou **`Debug`**)

- Ex d’un (gros) calcul de production : 3936 cœurs pendant 30x15h.

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

ONERA
THE FRENCH AEROSPACE LAB

TP2 04/02/2025

Mandelbrot en MPI

Produit Matrice-vecteur en MPI

Pourquoi le 1er TD est à rendre ?

- Pourquoi faire un github

- Compétences secondaires
- GDB
- Git branch, fork,
- Merge request et revue de code
- Rédaction, analyse

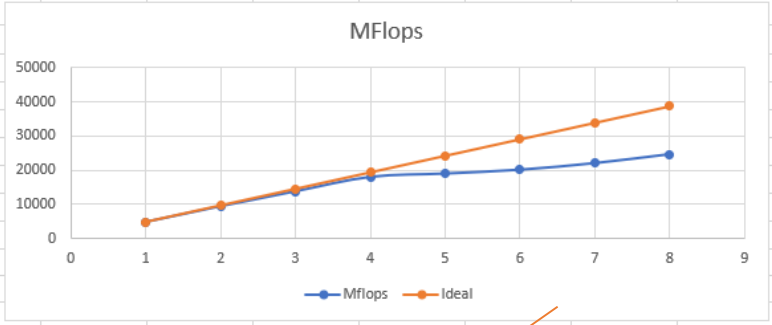
Les points clés du TD1

- Speedup (planche suivante)
- Taille du cache et taille des blocs
  - 3 matrices 512x512 prennent ~6Mo de cache:
    3 (matrices) * 512^2 (nombres) * 8(sizeof double) = 6144k

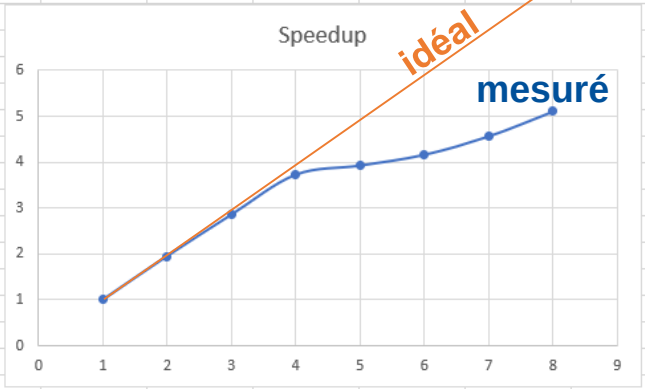
- MPI send / recv
- Les croquis « jeu de rôle » en MPI

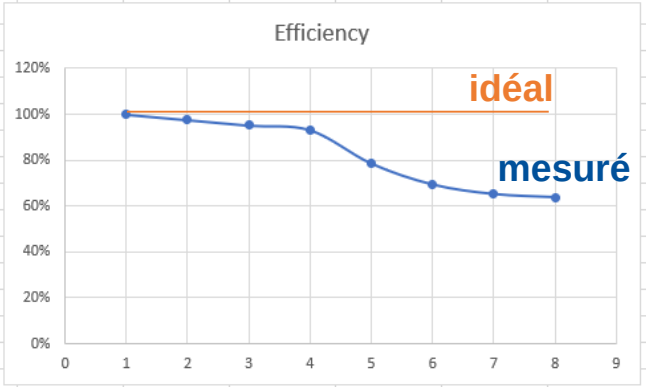
Performances : speedup, efficacité

Nb cpu	Mflops	Ideal	Speedup	Efficiency
1	4840	4840	1	100%
2	9448	9680	1,95206612	98%
3	13823	14520	2,85599174	95%
4	18040	19360	3,72727273	93%
5	19009	24200	3,92747934	79%
6	20141	29040	4,16136364	69%
7	22080	33880	4,56198347	65%
8	24698	38720	5,10289256	64%




```
[jd@ldmas707z ~]$ lscpu
Architecture: x86_64
CPU(s): 8 On-line
Thread(s) per core: 2
Core(s) per socket: 4
Socket(s): 1 NUMA node(s): 1
L1d cache: 32K
L1i cache: 32K
L2 cache: 256K
L3 cache: 8192K
…
```





Misc

- Produit mat – mat : l'an dernier on a donné un nom à ce type d'algo.
- Ecrire un programme `perf_test[.py,.sh,…]` qui prend le tableau précédent et trace les 3 courbes de la planche précédente (vitesse, speedup, efficacité).

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

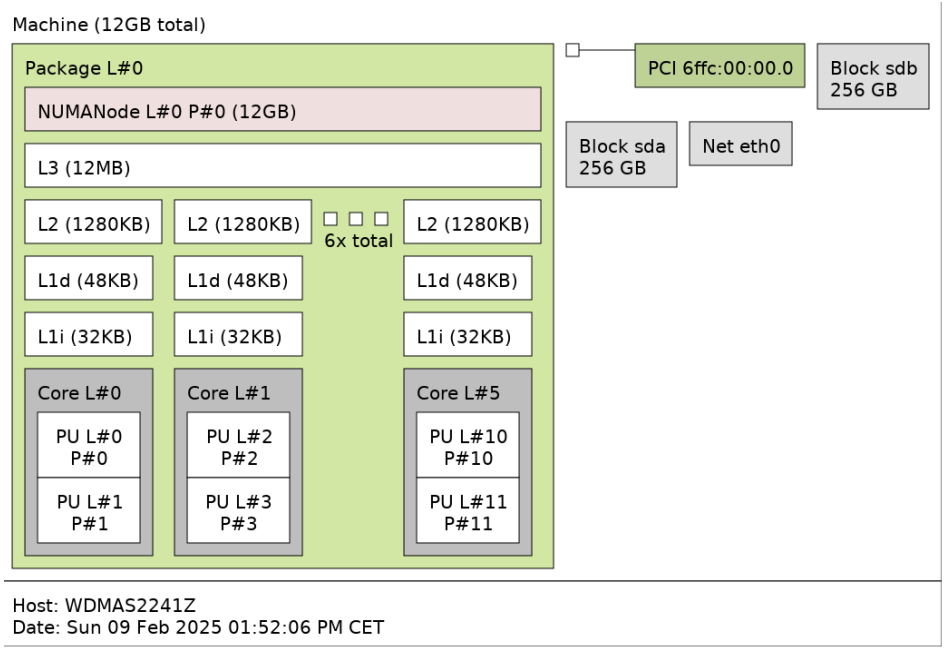
ONERA
THE FRENCH AEROSPACE LAB

TP3

Retour sur TP1

- La mémoire cache a plein de propriétés :
  - Rapide
  - Petite
  - Multi-niveaux
  - Attachée à un CPU
  - Associative
  - Localité des données

`lstopo`



Retour sur TP1 - Quiz

- La mémoire cache a plein de propriétés :
  - Rapide
  - Petite
  - Multi-niveaux
  - Attachée à un CPU
  - Associative
  - Localité des données

- Mflops(1024) << Mflops(1023 ou 1025)
- Permutation des boucles
- dgemm (Blas) est plus rapide que le TP1
- (Si je ne mets pas la stratégie de block product), les Mflops décroissent à partir d'une certaine taille de matrice
- Bug de race condition

Retour sur TP1 - Quiz

```
// ce code donne des réponses absurdes, quel est le problème ?
double x, y;
#pragma omp parallel for private(x,y) reduce (+: nbDarts)
    for (int sample = 0 ; sample < nbSamples ; ++ sample ) {
        nbTotal++;
        double x = distribution(generator);
        double y = distribution(generator);
        if ( x*x+y*y<=1 ) nbDarts ++;
    }
```

Attention au vocabulaire

- Processeur
  - Processeur physique
  - Processeur logique
- Processus
- Thread

TP1 : Analyse a priori, a posteriori

- Produit MatMat
  - N^2 données (*3 matrices)
  - N^3 opérations
  - CPU bound, donc très très très favorable au parallélisme
  - A quoi vont servir les blocs ?
- Calcul de pi
  - … données
  - … opérations

Vérification

- Comment vérifier que compute_pi_mpi.cpp est juste ?

- Calcul de pi
  - J’ai 4 processeurs (8 si on compte l’hyperthread)
  - J’observe un speedup de 3 pour 4 processus MPI

- Py3.8 et type annotations int|float
- Busy wait et speedup
- `mpirun --oversubscribe`

- cache 4 coups d'horloge, ram en 100
  - Pourquoi on observe *6 en temps et pas x25

- Attention aux argumentaires générés par IA :
  ils ne vont jamais à l'essentiel
- Si vous l'utilisez, dites-le clairement (question de déontologie), et expliquez pourquoi l'IA a raison (votre rôle d'humain est de vérifier ce qu'a généré l'IA, exercez votre esprit critique).

- Blas moins rapide que notre TP1 ?

Utiliser github pour les TP et leur correction

- git co -b TP3
- [work…]
- git commit -a -m "my work for TP3"
- git push (--set-upstream si nécessaire)
- Compare & Pull request depuis github
  - Base = votre point de départ
  - Vérifiez vos modifications (→ simplifiez si nécessaire)
  - Ajoutez une description (ex: les points sur lesquels vous voulez que je passe du temps)
  - Reviewers = @JDGaraudEnsta ou un collègue

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

ONERA
THE FRENCH AEROSPACE LAB

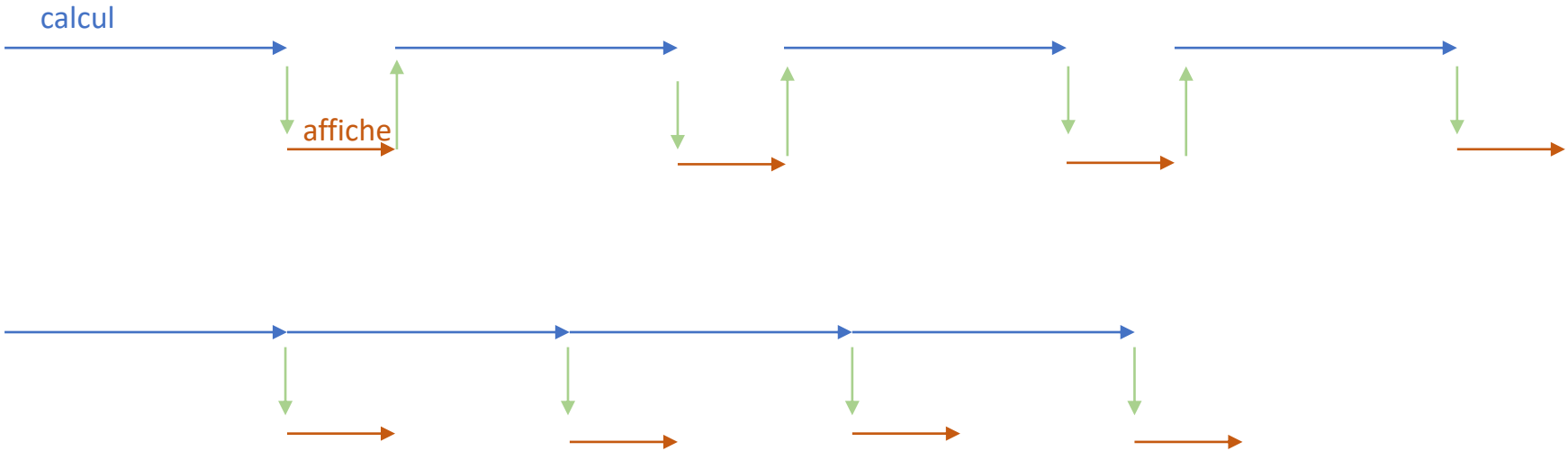
TP5 04/03/2025

Retour sur TP4

- MPI_Send n'est pas obligatoirement bloquant
  - Buffer optimization pour les petits messages
    - « petits » = 2048, ou 64, ou …, selon la config de MPI

```
from mpi4py import MPI
import numpy as np
import time

comm = MPI.COMM_WORLD

imax=17

if comm.rank == 0:
    for i in range(1, imax):
        time.sleep(.5)
        sz = 2**i
        x = np.ones(sz, dtype=np.int32)
        comm.Send(x, dest=1)
        print("I just sent", sz)

else:
    time.sleep(10)
    print("That was a good nap, let's receive...")
    for i in range(1, imax):
        sz = 2**i
        x = np.empty(sz, dtype=np.int32)
        comm.Recv(x, source=0)
        print("I just received", sz)
```

- 11/03 : examen papier (1h), puis TD
- Samedi 15/03 10h00 : rendu projet
  - Mail à jean-didier.garaud@onera.fr , avec votre lien github (y inviter JDGaraudEnsta si vous l'avez mis privé)
  - si binôme : le préciser
  - si IA : préciser ce qu'elle a produit
- 18/03 :
  - examen machine (3h)
  - Mini-soutenance du projet : 5-7 min


calcul
affiche