

Parallélisation et optimisation dans le contexte de la CFD

Application à des codes "stencil"

Philippe Parnaudeau[†],
Ingénieur de Recherche - CNRS

[†] Institut Pprime, CNRS, UPR 3346, Pôle formation du MCIA
e-mail: philippe.parnaudeau@cnrs.pprime.fr - Web page: <https://www.pprime.fr>



- 1 Contexte
- 2 Définitions. Concepts de base
- 3 Métriques et limites
- 4 Méthodologie pour "connaître" votre app.
- 5 Optimisation : règles et exemples
- 6 Outils et règles pour : Analyse, profilage, débogage
- 7 Scientific and math librairies
- 8 Questions?

La simulation numérique

- 3^{ème} pilier de la recherche
- Approche unique envisageable dans certains contextes : **Univers, climat, médecine ...**
- Approche complémentaire moins onéreuse et/ou plus rapide : **Nucléaire, aérodynamique ...**

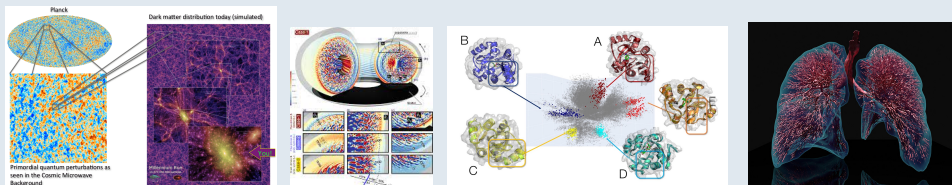
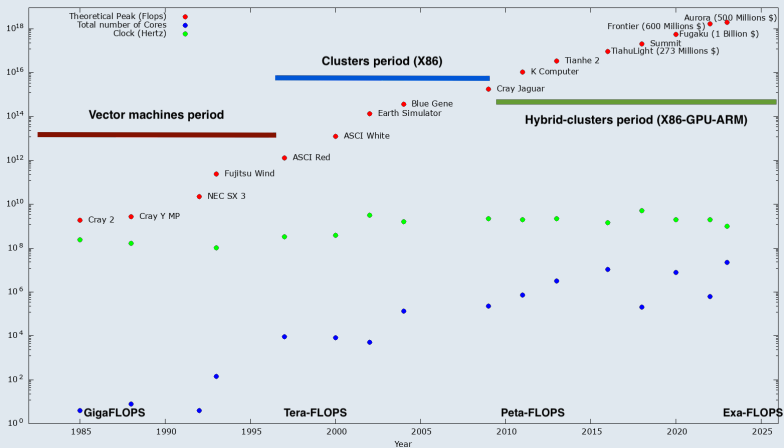


Figure: *Cosmologie-Projet Planck [Leclerc'13]; Fusion plasmas ITER [Dif-Pradalier'22];
Simulations de dynamique moléculaire COVID 19 [nsf'20];
Simulation du système pulmonaire [asme'20].*

”Computer science” : De la théorie au transistor

- 1820 • Concept *machine programmable* **C. Babbage** et **A. Lovelace - Lady Byron** [**Lovelace’1843**]
- 1850 • Algèbre de Boole **G. Boole** [**Boole’1854**]
- 1936 • **A. Turing** [**Turing’36**] : Machine éponyme et premiers algorithmes modernes
- 1940 • **C. Shannon** [**Shannon’40**] : Fondateur de la théorie de l’information
- 1947 • **AT&T Bell laboratory, Lucent, Nokia...** : Premier transistor, l’ère du supercalculateur

70 ans de supercalculateurs : Les 35 dernières années



Top supercomputer: loi de Moore

Architectures : Ordinateur

Ordinateur parallèle $\Leftrightarrow$ **Plusieurs CPU dans un ordinateur**

Une tâche s'exécute sur l'intégralité (ou un sous-ensemble) de la plateforme.

Taxonomie de Flynn : 4 groupes majeurs

- **Single Instruction Single Data: Sequential computer** (Von Neumann)
- **Single Instruction Multiple Data: Vectorial computer**
Packet of datas (Vector) can be addressed in the same CPU cycle (ex: Cray I and II)
- **Multiple Instruction Single Data: Pipeline computer**
Successive operations overlap. Example IMB 360/91
- **Multiple Instruction Multiple Data: Multi-CPU computer**
1 instruction/processor and for different datas
Often, in recent period, same app. is divided in threads which are executed on CPU cores
 - **MIMD shared memory**: Symmetric Shared Memory (SMP) computer, SMP-Numa computer
 - **MIMD distributed memory**: Massively Parallel Processing (MPP) computer, *modern cluster*

app. : application

Architectures : CPU

- **Instruction Set Architecture (ISA)**

Modèle "open" et abstrait décrivant le fct "logiciel" sur le processeur

Quelques standards : X86, RISC, ARM ...

- **Micro-architecture ou computer organization ou μ arch**

L'implémentation d'une ISA, pas toujours "open" (ex : AMD Zen3, Intel Xeon, Apple M2...)

- **Theoretical Peak Performance (Peak_{Flops})**

Nbre Max d'opération en virgule flottante par seconde (**Flop/s**) :

$$\text{Peak}_{\text{Flops}} = \text{Nb}_{\text{cpu}} \times \text{Nb}_{\text{core}} \times \text{Clock} \times 2 \text{ FMA} \times \frac{\text{register_size}}{64}$$

- **Theoretical Memory BandWidth**

Vitesse maximale d'accès cache (L1, L2, L3) et la mémoire (**Byte/s**)

- **Arithmetic Intensity**

Opérations en virgule flottante / mémoire accédée (**Flop/Byte**)

- **Performance / Watt**

Performance d'une app. ou architecture / 1 watt (**F/W**)

Optimisation d'une application*

Déterminer la performance séquentielle :

Performance application (FLOP/S) / Peak_{Flops}

↪ **Réduction et/ou optimisation** partie séquentielle

Déterminer le passage à l'échelle *scalability* :

Identifier la ou les parties parallèles efficaces

↪ **Accélération - speedup** ou **gain de la parallélisation**: $S_p = T_1 / T_{N_{b_p}}$

↪ **VS Amdahl's law speedup/strong scaling**

CPU bound app. ou **App. long tps exec.** Ideal $S_p = N_{b_p}$

↪ **VS Gustafson's law speedup/weak scaling**

Memory bound app. Ideal $S_p = 1$

↪ **Réduction et/ou optimisation** de la partie communication

*Hypothèse : l'application est déjà parallélisée

Computer Fluid Dynamics ?

Des hypothèses physiques

- Stationnaire ou instationnaire
- Fluide newtonien ou non newtonien
- Écoulement compressible ou incompressible
- Laminaire/turbulent
- Single/multiphase flow
- Avec ou sans espèces chimiques
- Interaction fluide-Structure

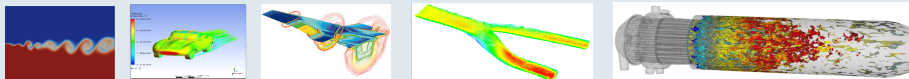


Figure: *Mixing layer, Miata (MX-5a), hyper-X vehicle at Mach 7 [Hyper-X], blood flow, rocket engine [Urbano'16]*

Computer Fluid Dynamics ?

Les principales étapes

- Géométrie du problème (ex: CAO)
- Maillage du problème
- Hypothèses physiques et modèles mathématiques utilisés
- **Algorithmes et implémentation**
- **Calcul**
- Analyse

Remarques

- Les 2 premières parties sont -en production- les plus chronophages (pas notre sujet)
- **Séminaire: jeu d'Eq. maillage cart. pour des problèmes multiphasique et/ou turbulent.**

A retenir

- Mauvais choix structure des données $\implies$ **Ré-écriture du code**
- Performance médiocre (CPU) $\implies$ **Optimisation** CPU - séquentiel
- Performance médiocre (CPU) $\implies$ **Parallelisation** CPU - mémoire partagée
- Performance médiocre (supercalculateur) $\implies$ **Parallelisation** - Optimisation communications
- **Nombreuses bibliothèques HPC** peuvent aider - **Attention au support**
- **Language n'est NI le problème NI la solution ! $\implies$ Le projet dicte le choix du langage!**

Optimisation: CPU - Séquentiel et Vectorielle

Limites et résolutions

- **CPU-bound:** App. limitée fréquence du CPU

Optimisation : CPU-SIMD

- **Cache-bound:** App. limitée taille et/ou fréquence "cache" des CPU

Optimisation : Instructions

- **Memory-bound:** Appl. limitée débit de la RAM

Optimisation : Adapter structure données, améliorer le schémas d'accès à la mémoire

Optimisation: Math libraries

- **I/O-bound:** App. limitée bande passante des E/S (pas abordé)

Optimisation: I/O libraries

Conclusion : Profilage et analyse de l'app. / l'architecture

Arithmetic Intensity (A.I) - Intensité Arithmétique

Avoir à l'esprit pour un programmeur

- Les **opérations de base** ont des latences plus ou moins importantes
- Les codes *stencils* ont, en général, performance médiocre $\Leftrightarrow$ CFD codes $\sim 0.3F/B$
- L'optimisation == investissement **LOURD** dans le choix des algorithmes et mise en œuvre.

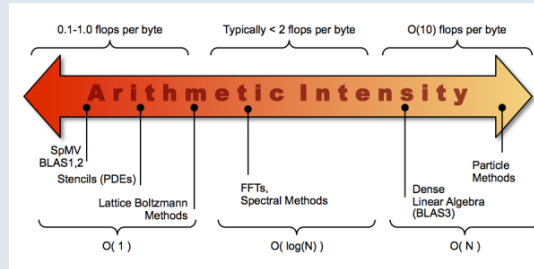


Figure: From Department Of Energy (DOE)

Roofline [Williams'09]

Definition

Evaluer les **performances** de l'app. avec :

Arithmetic Intensity = F(locality, bandwidth, different parallelization paradigms).

Un modèle "roofline" naif :

$$\text{Peak}_{\max} = \min \left(\text{Peak}_{theo}; A. I. \times \text{Mem. Band.} \right)$$

Roofline

- Fournit **Performance de l'app.** vs **Performance la plateforme de calcul**
- **Essentiel pour optimiser la partie séquentielle (CPU) de l'app.**

Roofline [Williams'09]

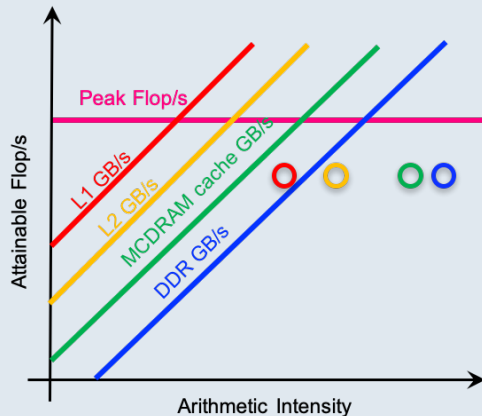
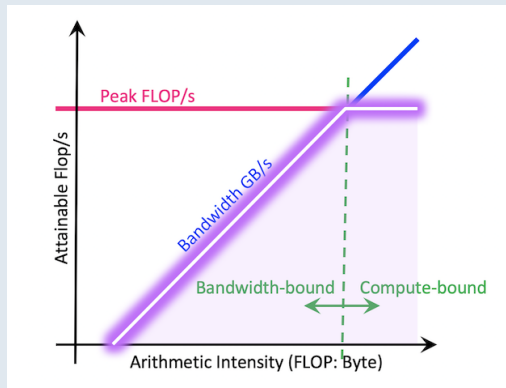


Figure: Roofline: naive view (left), more complex (right). From NERSC

Roofline [Williams'09]

Faire soi-même la "roofline" : Travail **fastidieux et chronophage**, mais **essentiel** !

Conseil : Uniquement pour des courtes parties de l'app.

Quelques outils

- **Intel Advisor**
- Empirial roofline tool
- Roofline Visualizer
- NVIDIA NVProf / NSight (GPU)
- **Papi library** (à la main)

App. maison "CFD" : Performance sur plateforme Intel Xeon 6248

- $I.A \simeq 0.3$ F/B
- $\simeq 7\%$ **Pic performance théorique** (pas si mauvais)
- $\simeq 20\%$ **bande passante théorique de la mémoire** (pas si bon)

Conclusion: Notre app. est limitée par la bande passante mémoire

Parallelisation - "Shared memory" - mémoire partagée

Definitions

- **Process:** Partie exécutable d'un programme, divisée en un ou plusieurs **thread**
- **Thread:** Plus petite partie d'un processus gérée de manière indépendante par l'O.S
- **Core:** Plus petite partie dédiée au calcul d'un processeur moderne

OpenMP

- API standard de l'industrie pour la programmation parallèle à mémoire partagée
- Basé sur des directives pragma à insérer dans le code
- Intégré langages modernes (C, C++, Fortran, Python-PyOMP); Implémenté compilateurs modernes et sur GPU (directives diff.)
- Alternatives: OpenACC, OpenCL, Pthreads

Exemple : Equation de Poisson

Résoudre "Poisson" est un point essentiel en CFD (incompressible)

Problem

Résoudre : $-\nabla u = f$, in Ω

Avec : $f(x, y) = 2(x(x - 1) + y(y - 1))$

CL Dirichlet : $u|_{\partial\Omega} = 0$

Solution exacte : $u(x, y) = xy(x - 1)(y - 1)$

Condition initiale : White noise

Résolution

- Différence finie
- Jacobi

OpenMP : Tactique

Fine Grain (FG) - Grain Fin

- **OpenMP divise la boucle en plusieurs threads**
- (+++) : Simple à utiliser (**pragma**)
- (+++) : Facile de maintenir l'app.
- (—) : Performance médiocre (nbre de "parallel region")
- (—) : Difficile avec code complexe/gros et certain langage. **Dépendance** dans les boucles

```

1 DO j= 1,ny
2 DO i= 1,nx
3   u_new(i,j)= c0 * ( c1*(u(i+1,j)+u(i-1,j)) &
4     + c2*(u(i,j+1)+u(i,j-1)) - f(i,j))
5 ENDDO
6 ENDDO

```

Listing: Jacobi-SEQ

```

1 !$OMP PARALLEL DO PRIVATE(i,j) &
2 !$OMP SHARED (ny,nx,u,u_new,f,c0,c1,c2)
3 DO j= 1,ny
4 DO i= 1,nx
5   u_new(i,j)= c0 * ( c1*(u(i+1,j)+u(i-1,j)) &
6     + c2*(u(i,j+1)+u(i,j-1)) - f(i,j))
7 ENDDO
8 ENDDO
9 !$OMP END PARALLEL DO

```

Listing: Jacobi-OMPFG

```

1 DO j= 1,ny
2 DO i= 1,nx
3   u(i,j)= omega*(c0*(c1*(u(i+1,j)+u(i-1,j)) &
4     +c2*(u(i,j+1)+u(i,j-1)) &
5     -f(i,j)))+(1.-omega)*u(i,j)
6 ENDDO
7 ENDDO
8 DO j= ny,1,-1
9 DO i= nx,1,-1
10  u(i,j)= omega*(c0*(c1*(u(i+1,j)+u(i-1,j)) &
11    +c2*(u(i,j+1)+u(i,j-1)) &
12    -f(i,j)))+(1.-omega)*u(i,j)
13 ENDDO
14 ENDDO

```

Listing: SSOR-loop nest

OpenMP : Tactique

Coarse Grain (CG) - Grain Grossier

- **Décomposition de domaine**
- (+++): Bonne performance
- (—): Gestion des communications
- (—) : Difficile à maintenir

```

1 |
2 | !$OMP PARALLEL PRIVATE (rang,jdeb,jfin)
3 |   rang=OMP_GET_THREAD_NUM()
4 |   nbproc=OMP_GET_NUM_THREADS()
5 |   jdeb=1+(rang*ny)
6 |   jfin=(ny+rang*ny)
7 | !$OMP END PARALLEL

```

Listing: Domain decomposition

```

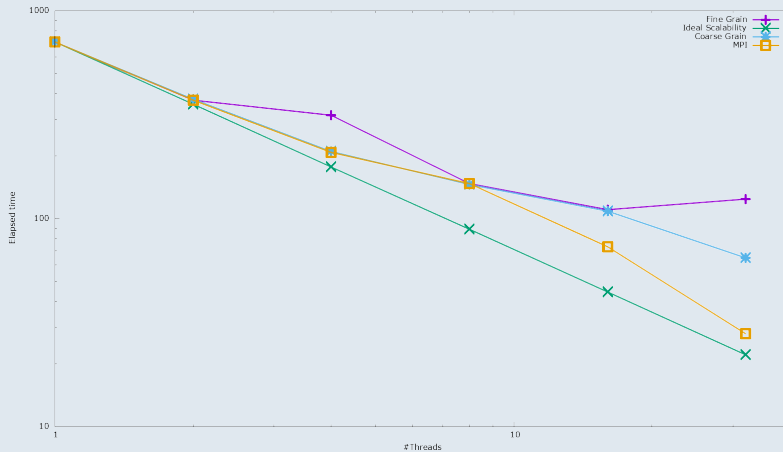
1 | DO j= jdeb,jfin
2 | DO i= 1,nx
3 |   u_new(i,j)=c0*(c1*(u(i+1,j)+u(i-1,j)) &
4 |     +c2*(u(i,j+1)+u(i,j-1)) &
5 |     -f(i,j))
6 | ENDO
7 | ENDO
8 |
9 | DO j= jdeb,jfin
10 | DO i= 1,nx
11 |   u(i,j)=u_new(i,j)
12 | ENDO
13 | ENDO
14 | !$omp barrier
15 | !$omp flush

```

Listing: Jacobi-OMPCG

OpenMP :

”Strong Scaling Test” - Test Passage à l’échelle ”fort”



OpenMP :

Conclusion

- **OpenMP (FG)** : La façon de commencer (rapidement) sur des cas simples : Premiers résultats en quelques heures
- **OpenMP (CG)** : Pour des cas plus complexes et d'excellentes performances
- **OpenMP (TASK)** : Un groupe d'instructions (tâches) est défini et travaille en parallèle. Le nombre de tâches est inconnu à l'avance (non présenté).

Parallelisation - "Distributed memory" - Mémoire distribuée

Definition

- **Decomposition de domaine** : Théorie décomposition de Scharwz (1870).
Séminaire: limité au cas "sans recouvrement"
- **Non-blocking point-to-point (P2P)** communication ("Stencil"-code)
- **Non-blocking collective** communication (FFT-code)

Message Passing Interface (MPI)

- **Thread-safe** : Threads accédant simultanément à la mémoire
- Définit la syntaxe et la sémantique des routines de la bibliothèque
- 2 implementations ppales (open-source) : **OpenMPI** and **MPICH2** or **MVAPICH2**

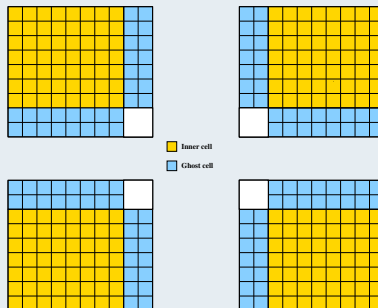
MPI :

Tactique

Non-blocking P2P communication

- **SCB 5 à 9-point "stencil"** / direction
- Ajout **"ghost points"** à chaque sous-domaine
- **Échange de données** entre voisins

- **(+++)** : Communication à faible coût
- **(+++)** : Performance peuvent être excellente
- **(—)** : Problème implicite



(a) Domain communication

MPI :

Tactique

SCB [Dubois'21]: Volume fini

Un des système hyperbolique résolu :

$$\frac{\partial \mathbf{W}}{\partial t} + \nabla \cdot \mathbf{A} + \mathbf{S} \nabla \cdot \mathbf{u} = 0$$

$\mathbf{W} = (\rho, \rho \mathbf{u}, E, \alpha)^\top$: Vecteur état

$\mathbf{A} = (\rho \mathbf{u}, \rho \mathbf{u} \otimes \mathbf{u} + P \mathbf{1}, \alpha \mathbf{u})^\top$: Vecteur flux

$\mathbf{S} = (0, \mathbf{0}, 0, -(K + \alpha))^\top$: Terme source

Sur une grille cartésienne, avec intégration explicite en temps

Les flux sont calculés "cell-vertex" avec diff. sch.

HLLC avec ou sans Muscl-Hancock

WENO-TENO avec ou sans Muscl-Hancock

JST

MPI : Tactique

Non-blocking P2P communication - Communication Point à Point non-bloquante

```

1  DO ndt=1,ndtmax
2  !$OMP PARALLEL IF(ijmax.gt.256) default(none)
3  !$OMP DO SCHEDULE(runtime) PRIVATE(i,j,k) COLLAPSE(2)
4      DO k=kmin,kmax
5          DO j=jmin,jmax
6              DO i=imin,imax
7                  Ri1=w1(i,j,k)-w1(i-1,j,k)
8                  sl=dmax(0.0,dmin(Ri1,1.0))+dmin(0,dmax(1,Ri1))
9                  W1(i,j,k)= W1(i-1,j,k)+1/4*sl*(W1(i-1,j,k)-W1(i-2,j,k))+1/4*sl*(W1(i,j,k)-W1(i-1,j,k))
10             ENDDO
11         ENDDO
12     ENDDO
13 !$OMP END DO
14     CALL BOUNDARY(W1)
15 !$OMP END PARALLEL
16     CALL MPI_SENDRCV(W1, imax*kmax, MPI_DOUBLE_PRECISION,neib_mpi(N),tag, &
17                     W1, imax*kmax, MPI_DOUBLE_PRECISION,neib_mpi(S),tag, &
18                     comm, status, err_mpi)
19 ENDDO

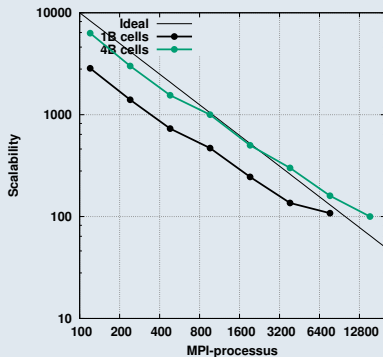
```

- Line 2: Zone parallel unique : Meilleur performance!
- Line 4-5-6: Bonne vectorisation et optimisation du cache
- Line 16: P2P - W1 échange avec élément N - S
- Line 16: Taille du message

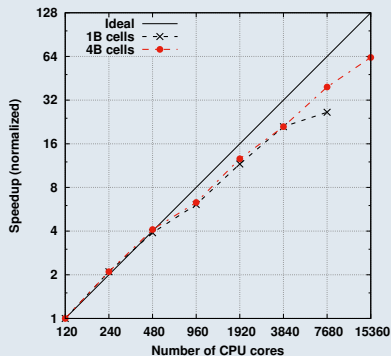
Listing: Hybride MPI-OpenMP implementation

MPI-P2P:

”Strong Scaling Test” - Test Passage à l’échelle ”fort”



Strong-Scaling : Passage à l’échelle



Strong-Scaling : Speedup

MPI: Tactique

Non-blocking collective communication - Communication collective non-bloquante

- **Calcul 1D** d'un op. math. suivant 1 direction
- Transposition via **communication collective**
- **(+++)**: Problème implicite (FFT and co)
- **(—)**: Coût des communications +/- élevé

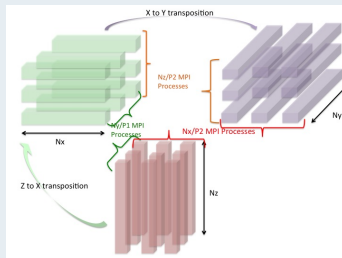


Figure: Pencil MPI communication schemes

MPI: Tactic

GPS [Parnaudeau'15]: FFT code

Soit l' **Equation Gross-Pitaevskii (EGP)** adimensionnée avec un **terme de rotation**, pour un cas **stationnaire**:

$$\mu\phi(\mathbf{x}) = \left(-\frac{1}{2}\Delta + \mathbf{V}(\mathbf{x}) + \beta|\phi(\mathbf{x})|^2 - \Omega L_z \right) \phi(\mathbf{x}) \text{ with } \|\phi\|_0^2 = 1$$

où μ est le **potentiel chimique** du condensat/nuage et

ϕ : fonction d'onde stationnaire

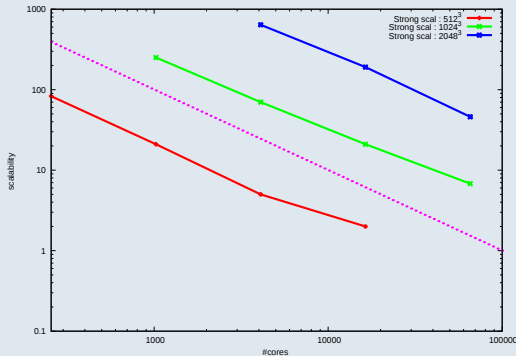
V : Trappe magnétique, qui peut prendre plusieurs formes etc.

β : Interaction entre particules dans le nuage

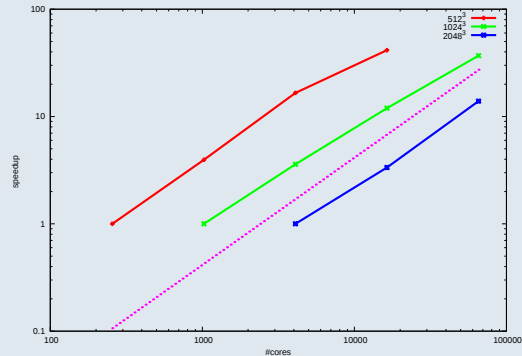
ΩL_z : Moment angulaire

MPI Non-blocking collective:

Strong-Scaling performance test - Test Passage à l'échelle "fort"



Strong-Scaling: Passage à l'échelle



Strong-Scaling: Speedup

MPI:

Conclusion

- **P2P communication** : Commencer par **send/recv** et **step/step non bloquant**.
- **P2P communication**: Début simple, mais difficile à optimiser
- **Collective communication**: A n'utiliser que si impératif (opération de réduction et les limiter)
- **Collective communication**: Aujourd'hui, certaines sont asynchrones et cela importe un gain, pas toujours facile à utiliser
- **Recommendation**: Limiter au maximum les communications, utiliser les threads, nouvelles normes apportent des réponses (one side communication)

Conclusion: optimisation et parallelisation

- Nouvelle génération de supercalculateur est hybride (CPU+GPU)
- Architecture X86-64 est et sera de moins en moins archi-dominant (ex: ARM on Apple (Mx), Fujitsu (A64FX), Nvidia (Grace))
- Nécessité d'utiliser au moins 2 or 3 paradigmes
- Toujours avoir à l'esprit maintenabilité et développement futur
- Flops/Watt est le vrai "mur" pour les développeurs !
Attention le Flops/Watt du constructeur n'a rien à voir avec celui de votre app.

LSCPU: CPU information

```
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 192
On-line CPU(s) list: 0-191
Thread(s) per core: 1
Core(s) per socket: 96
Socket(s): 2
NUMA node(s): 2
Vendor ID: AuthenticAMD
CPU family: 25
Model: 17
Model name: AMD EPYC 9654 96-Core Processor
Stepping: 1
CPU MHz: 3600.695
CPU max MHz: 2400.0000
CPU min MHz: 1500.0000
BogoMIPS: 4792.55
Virtualization: AMD-V
L1d cache: 32K
L1i cache: 32K
L2 cache: 1024K
L3 cache: 32768K
NUMA node0 CPU(s): 0-95
NUMA node1 CPU(s): 96-191
Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov ht lat pse36 clflush mmx
p_good nopl nonstop_tsc cpuid extd_apicid aperfmperf pni pclmulqdq monitor ssse3 fma cx16 pcid sse4_1
pic cr8_legacy x87 sse4a misalignsse 3dnowprefetch osvw ibs skinit wdt tce rdpstore perfctr_core perfctr
p_l3s l3bp stlb vscall fsgsbase bmi1 avx2 smep bmi2 rrm inopid cqm_1024B_avx512f avx512dq rdseed a
xsave xgetbv1 xsavec cqm_llc cqm_occup_llc cqm_rbm_total cqm_rbm_local avx512_vf16 clzero lrpwr xsavec
an_flushbyasid decodeassists pausefilter pfthreshold avic v_vmsave_valldm v_vmsave_valldm v_vspec_ctl avx512vbmi a
2_vppcnddq la57 rdpid overflow_recov succor srca frrn flush_lld
```

Line 1-2-13: Architecture information

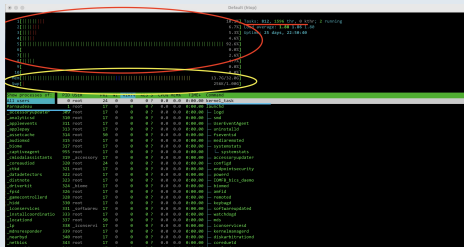
Line 4-5-6-7-8: Number of socket, cores and threads per node

Line 9-24-25: Memory policy: Non Uniform Memory Architecture
AVX512: Vectoriel support (SIMD optimisation)

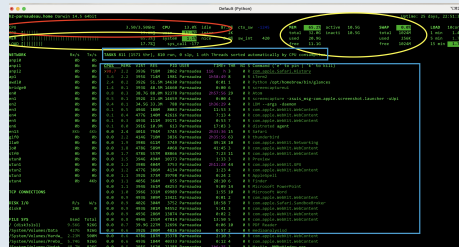
lscpu

HTOP or GLANCES:

System monitoring core usage, memory usage, process information



htop



glances

GNU Debugger (GDB)

- **Compile** the program with options: -g
- **Serial/Sequential/OpenMP**: gdb Binary_name
- **Distributed**: MPIRUN_Command_name xterm -e gdb Binary_name (Nb proc. < 10)
- **Some GUI for GDB**

Useful commands

Command	Argument	Explain
---------	----------	---------

b	file:line	Breakpoint in file at line
n	binary	Execute binary
p	variable	Display variable value
n		Execute next instruction
c		Continue the program instruction
quit		Quit gdb

GNU Profiler (GPROF)

```

[1] flat profile:

Each sample counts as 0.01 seconds.
 % cumulative self      self      total
time seconds  seconds  calls   s/call   s/call   name
39.04    6.04    6.04    63    0.10    0.10  fluxio_2d_4eq
17.04    8.80    2.76    21    0.13    0.48  hllc_hancock_4eq_2d
10.00    10.40    1.64    21    0.08    0.00  visqueux_2d_
 5.30    11.30    0.92 98073120  0.00    0.00  __schema_MOD_slope
 4.65    12.02    0.72    10    0.07    0.10  visua1_gp_
 4.50    12.73    0.71    22    0.03    0.03  inletat_2d_4eq_
 4.07    13.36    0.63    21    0.03    0.04  penal_yemi_impl_2d_
 3.43    13.89    0.53    42    0.01    0.01  pente_2d_
 2.13    14.22    0.33    21    0.02    0.02  compute_turb_stats_
 1.87    14.53    0.29    21    0.01    0.01  pastemp_
 1.49    14.74    0.23    11    0.02    0.02  residu_
 1.23    14.93    0.19    10    0.02    0.02  comput_s_
 1.03    15.00    0.16    21    0.01    0.01  compute_cdrpenal_2d_
 0.58    15.11    0.09    51    0.00    0.00  write_file_dfortran_mpi_
 0.58    15.27    0.09    1    0.09    0.00  __allocation_MOD_allocate
 0.26    15.31    0.04    10    0.00    0.00  dery_
 0.26    15.35    0.04    10    0.00    0.00  dery_
 0.12    15.38    0.03    21    0.00    0.00  comput_ec_
 0.13    15.40    0.02    1    0.02    0.00  init_lcas_4eq_2d_visq_monophas_
 0.06    15.41    0.01    22    0.00    0.00  chlore_4eq_2d_
 0.06    15.42    0.01    21    0.00    0.00  compute_dghost_
 0.05    15.43    0.01    1    0.01    0.00  compute_lsvl_
 0.06    15.44    0.01    1    0.01    0.02  init_ghost_
 0.06    15.45    0.01    1    0.01    0.03  init_penal_2d_
 0.06    15.46    0.01    1    0.01    0.01  init_turb_stats_
 0.00    15.47    0.01    1    0.00    0.00  __schema_MOD_fill
 0.00    15.47    0.00    120    0.00    0.00  __string_mod_MOD_getlowercase
 0.00    15.47    0.00    100    0.00    0.00  sch_print_
 0.00    15.47    0.00    40    0.00    0.00  __allocation_MOD_allocate_error
 0.00    15.47    0.00    30    0.00    0.00  time_measure_
 0.00    15.47    0.00    22    0.00    0.00  chlore_
 0.00    15.47    0.00    22    0.00    0.00  compute_dcdt_
 0.00    15.47    0.00    21    0.00    0.00  compute_sondes_
 0.00    15.47    0.00    21    0.00    0.00  compute_penal_
 0.00    15.47    0.00    21    0.00    0.00  compute_penal_2d_
 0.00    15.47    0.00    21    0.00    0.48  hllc_hancock_
 0.00    15.47    0.00    21    0.00    0.00  output_timeline_

```

Gprof: Flat view

```

[1] Call graph (explanation follows)

granularity: each sample hit covers 2 byte(s) for 0.00% of 15.47 seconds

index % time  self  children  called  name
[1]  99.9  0.00  15.46  1/1  main [2]
      0.00  10.15  21/21  hllc_hancock_ [3]
      0.00  1.64  21/21  visqueux_ [6]
      0.00  0.99  11/11  output_ [8]
      0.00  0.79  21/21  penal_yemi_impl_ [11]
      0.00  0.71  22/22  inletat_ [12]
      0.33  0.00  21/21  compute_turb_stats_ [16]
      0.29  0.00  21/21  pastemp_ [17]
      0.23  0.00  11/11  residu_ [18]
      0.00  0.21  1/1  init_ [21]
      0.00  0.00  55/55  write_file_dfortran_mpi_ [22]
      0.03  0.00  21/21  comput_ec_ [26]
      0.00  0.03  1/1  init_penal_ [27]
      0.00  0.01  22/22  chlore_ [32]
      0.00  0.01  21/21  compute_penal_ [36]
      0.01  0.00  1/1  init_turb_stats_ [38]
      0.00  0.00  10/10  time_measure_ [42]
      0.00  0.00  21/21  comput_dcdt_ [43]
      0.00  0.00  1/1  subdr_ [52]
      0.00  0.00  1/1  init_sondes_ [49]
      0.00  0.00  1/1  init_comput_s_helicity_ [47]
      0.00  0.00  1/1  init_print_ [64]
      0.00  0.00  1/1  close_file_ [66]
      0.00  0.00  1/1  restart_ [51]
      0.00  0.00  1/1  __allocation_MOD_desalloue [108]

-----
[2]  99.9  0.00  15.46  1/1  main [2]
      0.00  15.46  1/1  MAIN__ [1]
      0.00  10.15  21/21  MAIN__ [1]
[3]  65.6  0.00  10.15  21  hllc_hancock_ [3]
      2.76  7.39  21/21  hllc_hancock_4eq_2d_ [4]
      2.76  7.39  21/21  hllc_hancock_ [3]

```

Gprof: Grap view

- **Compile/link** the program with options: `-g -pg`
- **Execute** the program in standard way
- Execution **generates profiling files** in execution directory
- To obtain profiling report generation: **gprof**
Binary_name gmon.out.MPI_Rank
gprof.out.MPI_Rank

Other tools

- **Intel offers a wide and attractive range of tools**

- **Intel Advisor:** Help for design code for efficient vectorization, threading, and offloading to accelerators
- **Intel Inspector:** Locate and debug threading, memory, and persistent memory errors
- **Intel Trace Analyzer and Collector (ITAC):** Help for efficient MPI application
- **Intel VTune™ Profiler:** Analysing and optimizing performance of code for several architecture

- **Cray offers a wide and attractive range of tools**

- **Cray Performance and Analysis Tools:** Help to design code for efficient vectorization, threading, and offloading to accelerators

Scientific and math libraries

- The Netlib math library
 - **BLAS-1-2-3**: (vector and matrix operations) - Fortran
 - **CBLAS** - C
 - **LAPACK**: Solve linear equation systems
 - **ScaLAPACK**: Distributed version of Lapack
- Intel Library: MKL
 - **Netlib, FFTW ...**
- AMD Optimized CPU Libraries: AOCL
 - **Netlib, FFTW ...**
- NVidia GPU Libraries: CUDA-X
 - **Netlib, FFTW ...**
- I/O Libraries
 - **HDF5, Netcdf, Adios2**

[Return to mainpages](#)

Latence de différentes instructions

Operation	coût / cycle CPU	SIMD Optimisation
ADD, OR, SUB, MUL, FMA	2	Excellent
L1-Read	4	Excellent
If, wrong branch	[10;20]	Good
L2-Read	10	Good
DIV, SQRT	[20;40]	Poor
Function callecd (Language and method dependent)	> [30;60]	Poor
L3-Read	[60;70]	Very poor
EXP, LOG, SIN, COS..	> 100	Very poor
RAM-NUMA-Read	[100;500]	No gain!
Allocation/deallocation	[200;500]	No gain!
Kernel call	> 1000	

Table: Cost of instruction latencies:

from A. Fog, *Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD, and VIA CPUs*

[Return to mainpages](#)

hiérarchie de la mémoire

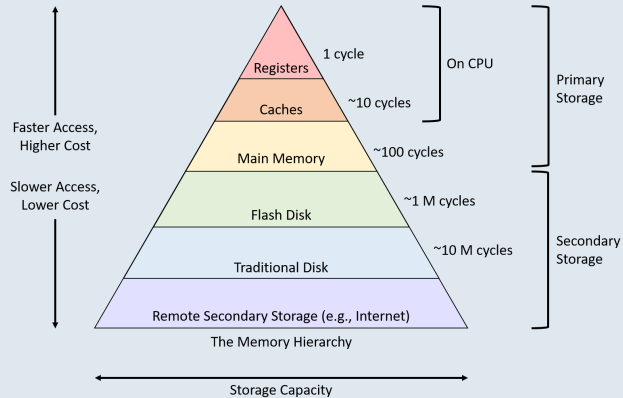


Figure: From Durganshu Mishra's blog

Single Instruction Multiple Data and vectorization pipelining (1/2)

- **Les CPUs-HPC possèdent un jeu d'instructions vectorielles**
ex: Streaming SIMD Extension (SSE) $\in$ Advanced Vector eXtensions AVX $\in$ AVX512
- **Without SIMD:** Instruction/instruction **vs with:** Grouped instructions (vector) in same CPU cycle
- **Use compilers options or using dedicated libraries or using pragma approach (not presented)**
- **Recommendation: *IDRIS SIMD course***

```

1 DO k=kmin,kmax
2 DO j=jmin,jmax
3 DO i=imin,imax
4   W1(i,j,k)= 0.5*(W1(i+1,j,k)+W1(i,j,k))
5   W2(i,j,k)= sqrt(W1(i,j,k)*W1(i,j,k))
6   W2(i,j,k) = 0.5*(W2(i-1,j,k)+W2(i,j,k))
7   if (W2(i,j,k) > Lim) print *, "Value..W2:", w2(i,j,k)
8   W3=user_func(W2(i,j,k))
9 ENDDO
10 ENDDO
11 ENDDO

```

- Line 4: **Vectorisable mais dans une autre boucle**
- Line 5: Ne peut pas être vectorisée: **Fonction transcendante**
- Line 6: Ne peut pas être vectorisée: **Anti dependence**
- Line 7: Ne peut pas être vectorisée: **Test conditionnel**
- Line 8: Ne peut pas être vectorisée: **Appel de fonction**

Listing: Fortran examples

Single Instruction Multiple Data and vectorization pipelining (2/2)

Règles simples

- **Toujours:** spécifier la taille de la boucle
- **Eviter:** E/S ou fonction appelée ou test conditionnel
- **Eviter:** dépendance dans une boucle
- **Eviter:** les pointeurs
- **Eviter:** les boucles internes trop petites
- **Recommandation:** vérifier les options du compilateur et sa documentation

Exemples avec Gfortran

- *-O3* : Maximum optim. (take care) enabled by default
- *-march=native* (AVX1,AVX2, AVX512...): Leave compiler selection to CPU optimization
- *-fopt-info-vec-all*: Vectorization informations
- All compilers (Intel, Cray ...) have an equivalent options: Read the compiler documentation

Libraries:

HPC libraries (BLAS-1,2,3) dedicated to performing basic vector and matrix operations

[Return to mainpages](#)

Optimisation des instructions

Optimisation des cache (1/2)

- Quelle est la différence entre la mémoire cache et la mémoire RAM ? **Memory hierarchy latency**
- Politique de gestion du cache : **Localité spatiale des données**
- Politique de gestion du cache : **Localité temporelle des données**
- **Avoid cache conflicts**

```

1 DO k=kmin,kmax
2 DO j=jmin,jmax
3 DO i=imin,imax
4 W1(i,j,k)=W0(i,j,k)*W3(i,j,k)
5 W2(i,j,k)=0.5*(W1(i,j,k)*W1(i,j,k))
6 W3(i,j,k)=0.5*(W2(i,j,k)+W2(i+1,j,k))
7 W4(i,j,k)= W4(i,j,k)/W0(i,j,k)
8 ENDDO
9 ENDDO
10 ENDDO

```

- Taille du tableau $\propto$ taille du cache : **Des conflits de cache apparaissent**
- **Localité spatiale:** Conserver l'ordre des boucles
- **Localité temporelle:** W3 est ré-utilisé

Listing: Fortran source

Optimisation des instructions

Optimisation des cache (2/2)

Règles simples

- **Mémoire contiguë**: ordre index des boucles (fct du langage) - localité spatiale
- **Réduction de la latence (localité des données)** : Amélioration des conflits de cache
- **Tolérer la latence (prefetching)** : Optimisation localité (proche CPU)
- Point de vue : Optimisation complexe et architecture dépendant

Gfortran Compiler: optimization options

- Compiler optimizations: Prefetching, loop unrolling, cache-aware
- *-fopt-info-note*: Optimization report
- Read the compiler documentation and use the pragma directive optimization carefully

Libraries:

Libraries (Blas, Lapack) dedicated to performing cache optimization

[Return to mainpages](#)

CPU Architecture:

Theoretical Peak Performance

```

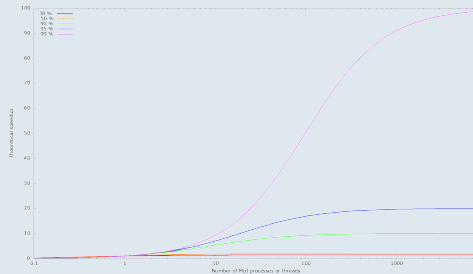
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 192
On-line CPU(s) list: 0-191
Thread(s) per core: 1
Core(s) per socket: 96
Socket(s): 2
NUMA node(s): 2
Vendor ID: AuthenticAMD
CPU Family: 25
Model: 17
Model name: AMD EPYC 9654 96-Core Processor
Stepping: 1
CPU MHz: 3600.095
CPU max MHz: 2400.0000
CPU min MHz: 1500.0000
BogoMIPS: 4792.55
Virtualization: AMD-V
L1d cache: 32K
L1i cache: 32K
L2 cache: 1024K
L3 cache: 32768K
NUMA node0 CPU(s): 0-95
NUMA node1 CPU(s): 96-191
Flags: fpu vme de pse tsc mtr pae mce cx8 apic sep mtrr pge mca cmov npt sse3 clflush w
p_good nopl nonstop_tsc cpuid extd_apicid aperfmperf pni pclmulqdq monitor ssse3 fma cx16 pcid sse4_1
pic cr8_legacy abm sse4a misalignsse 3dnowprefetch osvw ibs skinit wdt tce topoext perfctr_core perfctr
a lbrs lbpb stlb vmcall fsgsbase bmi1 avx2 smep bmi2 erms invpcid cqm_1 avx512f avx512dq rdseed
saver_2negatve_3xsave_4xsave_opt_5l1c_6q_7c_8c_9c_10c_11c_12c_13c_14c_15c_16c_17c_18c_19c_20c_21c_22c_23c_24c_25c_26c_27c_28c_29c_30c_31c_32c_33c_34c_35c_36c_37c_38c_39c_40c_41c_42c_43c_44c_45c_46c_47c_48c_49c_50c_51c_52c_53c_54c_55c_56c_57c_58c_59c_60c_61c_62c_63c_64c_65c_66c_67c_68c_69c_70c_71c_72c_73c_74c_75c_76c_77c_78c_79c_80c_81c_82c_83c_84c_85c_86c_87c_88c_89c_90c_91c_92c_93c_94c_95c_96c_97c_98c_99c_100c_101c_102c_103c_104c_105c_106c_107c_108c_109c_110c_111c_112c_113c_114c_115c_116c_117c_118c_119c_120c_121c_122c_123c_124c_125c_126c_127c_128c_129c_130c_131c_132c_133c_134c_135c_136c_137c_138c_139c_140c_141c_142c_143c_144c_145c_146c_147c_148c_149c_150c_151c_152c_153c_154c_155c_156c_157c_158c_159c_160c_161c_162c_163c_164c_165c_166c_167c_168c_169c_170c_171c_172c_173c_174c_175c_176c_177c_178c_179c_180c_181c_182c_183c_184c_185c_186c_187c_188c_189c_190c_191c_192c_193c_194c_195c_196c_197c_198c_199c_200c_201c_202c_203c_204c_205c_206c_207c_208c_209c_210c_211c_212c_213c_214c_215c_216c_217c_218c_219c_220c_221c_222c_223c_224c_225c_226c_227c_228c_229c_230c_231c_232c_233c_234c_235c_236c_237c_238c_239c_240c_241c_242c_243c_244c_245c_246c_247c_248c_249c_250c_251c_252c_253c_254c_255c_256c_257c_258c_259c_260c_261c_262c_263c_264c_265c_266c_267c_268c_269c_270c_271c_272c_273c_274c_275c_276c_277c_278c_279c_280c_281c_282c_283c_284c_285c_286c_287c_288c_289c_290c_291c_292c_293c_294c_295c_296c_297c_298c_299c_300c_301c_302c_303c_304c_305c_306c_307c_308c_309c_310c_311c_312c_313c_314c_315c_316c_317c_318c_319c_320c_321c_322c_323c_324c_325c_326c_327c_328c_329c_330c_331c_332c_333c_334c_335c_336c_337c_338c_339c_340c_341c_342c_343c_344c_345c_346c_347c_348c_349c_350c_351c_352c_353c_354c_355c_356c_357c_358c_359c_360c_361c_362c_363c_364c_365c_366c_367c_368c_369c_370c_371c_372c_373c_374c_375c_376c_377c_378c_379c_380c_381c_382c_383c_384c_385c_386c_387c_388c_389c_390c_391c_392c_393c_394c_395c_396c_397c_398c_399c_400c_401c_402c_403c_404c_405c_406c_407c_408c_409c_410c_411c_412c_413c_414c_415c_416c_417c_418c_419c_420c_421c_422c_423c_424c_425c_426c_427c_428c_429c_430c_431c_432c_433c_434c_435c_436c_437c_438c_439c_440c_441c_442c_443c_444c_445c_446c_447c_448c_449c_450c_451c_452c_453c_454c_455c_456c_457c_458c_459c_460c_461c_462c_463c_464c_465c_466c_467c_468c_469c_470c_471c_472c_473c_474c_475c_476c_477c_478c_479c_480c_481c_482c_483c_484c_485c_486c_487c_488c_489c_490c_491c_492c_493c_494c_495c_496c_497c_498c_499c_500c_501c_502c_503c_504c_505c_506c_507c_508c_509c_510c_511c_512c_513c_514c_515c_516c_517c_518c_519c_520c_521c_522c_523c_524c_525c_526c_527c_528c_529c_530c_531c_532c_533c_534c_535c_536c_537c_538c_539c_540c_541c_542c_543c_544c_545c_546c_547c_548c_549c_550c_551c_552c_553c_554c_555c_556c_557c_558c_559c_560c_561c_562c_563c_564c_565c_566c_567c_568c_569c_570c_571c_572c_573c_574c_575c_576c_577c_578c_579c_580c_581c_582c_583c_584c_585c_586c_587c_588c_589c_590c_591c_592c_593c_594c_595c_596c_597c_598c_599c_600c_601c_602c_603c_604c_605c_606c_607c_608c_609c_610c_611c_612c_613c_614c_615c_616c_617c_618c_619c_620c_621c_622c_623c_624c_625c_626c_627c_628c_629c_630c_631c_632c_633c_634c_635c_636c_637c_638c_639c_640c_641c_642c_643c_644c_645c_646c_647c_648c_649c_650c_651c_652c_653c_654c_655c_656c_657c_658c_659c_660c_661c_662c_663c_664c_665c_666c_667c_668c_669c_670c_671c_672c_673c_674c_675c_676c_677c_678c_679c_680c_681c_682c_683c_684c_685c_686c_687c_688c_689c_690c_691c_692c_693c_694c_695c_696c_697c_698c_699c_700c_701c_702c_703c_704c_705c_706c_707c_708c_709c_710c_711c_712c_713c_714c_715c_716c_717c_718c_719c_720c_721c_722c_723c_724c_725c_726c_727c_728c_729c_730c_731c_732c_733c_734c_735c_736c_737c_738c_739c_740c_741c_742c_743c_744c_745c_746c_747c_748c_749c_750c_751c_752c_753c_754c_755c_756c_757c_758c_759c_760c_761c_762c_763c_764c_765c_766c_767c_768c_769c_770c_771c_772c_773c_774c_775c_776c_777c_778c_779c_780c_781c_782c_783c_784c_785c_786c_787c_788c_789c_790c_791c_792c_793c_794c_795c_796c_797c_798c_799c_800c_801c_802c_803c_804c_805c_806c_807c_808c_809c_810c_811c_812c_813c_814c_815c_816c_817c_818c_819c_820c_821c_822c_823c_824c_825c_826c_827c_828c_829c_830c_831c_832c_833c_834c_835c_836c_837c_838c_839c_840c_841c_842c_843c_844c_845c_846c_847c_848c_849c_850c_851c_852c_853c_854c_855c_856c_857c_858c_859c_860c_861c_862c_863c_864c_865c_866c_867c_868c_869c_870c_871c_872c_873c_874c_875c_876c_877c_878c_879c_880c_881c_882c_883c_884c_885c_886c_887c_888c_889c_890c_891c_892c_893c_894c_895c_896c_897c_898c_899c_900c_901c_902c_903c_904c_905c_906c_907c_908c_909c_910c_911c_912c_913c_914c_915c_916c_917c_918c_919c_920c_921c_922c_923c_924c_925c_926c_927c_928c_929c_930c_931c_932c_933c_934c_935c_936c_937c_938c_939c_940c_941c_942c_943c_944c_945c_946c_947c_948c_949c_950c_951c_952c_953c_954c_955c_956c_957c_958c_959c_960c_961c_962c_963c_964c_965c_966c_967c_968c_969c_970c_971c_972c_973c_974c_975c_976c_977c_978c_979c_980c_981c_982c_983c_984c_985c_986c_987c_988c_989c_990c_991c_992c_993c_994c_995c_996c_997c_998c_999c_1000c_1001c_1002c_1003c_1004c_1005c_1006c_1007c_1008c_1009c_1010c_1011c_1012c_1013c_1014c_1015c_1016c_1017c_1018c_1019c_1020c_1021c_1022c_1023c_1024c_1025c_1026c_1027c_1028c_1029c_1030c_1031c_1032c_1033c_1034c_1035c_1036c_1037c_1038c_1039c_1040c_1041c_1042c_1043c_1044c_1045c_1046c_1047c_1048c_1049c_1050c_1051c_1052c_1053c_1054c_1055c_1056c_1057c_1058c_1059c_1060c_1061c_1062c_1063c_1064c_1065c_1066c_1067c_1068c_1069c_1070c_1071c_1072c_1073c_1074c_1075c_1076c_1077c_1078c_1079c_1080c_1081c_1082c_1083c_1084c_1085c_1086c_1087c_1088c_1089c_1090c_1091c_1092c_1093c_1094c_1095c_1096c_1097c_1098c_1099c_1100c_1101c_1102c_1103c_1104c_1105c_1106c_1107c_1108c_1109c_1110c_1111c_1112c_1113c_1114c_1115c_1116c_1117c_1118c_1119c_1120c_1121c_1122c_1123c_1124c_1125c_1126c_1127c_1128c_1129c_1130c_1131c_1132c_1133c_1134c_1135c_1136c_1137c_1138c_1139c_1140c_1141c_1142c_1143c_1144c_1145c_1146c_1147c_1148c_1149c_1150c_1151c_1152c_1153c_1154c_1155c_1156c_1157c_1158c_1159c_1160c_1161c_1162c_1163c_1164c_1165c_1166c_1167c_1168c_1169c_1170c_1171c_1172c_1173c_1174c_1175c_1176c_1177c_1178c_1179c_1180c_1181c_1182c_1183c_1184c_1185c_1186c_1187c_1188c_1189c_1190c_1191c_1192c_1193c_1194c_1195c_1196c_1197c_1198c_1199c_1200c_1201c_1202c_1203c_1204c_1205c_1206c_1207c_1208c_1209c_1210c_1211c_1212c_1213c_1214c_1215c_1216c_1217c_1218c_1219c_1220c_1221c_1222c_1223c_1224c_1225c_1226c_1227c_1228c_1229c_1230c_1231c_1232c_1233c_1234c_1235c_1236c_1237c_1238c_1239c_1240c_1241c_1242c_1243c_1244c_1245c_1246c_1247c_1248c_1249c_1250c_1251c_1252c_1253c_1254c_1255c_1256c_1257c_1258c_1259c_1260c_1261c_1262c_1263c_1264c_1265c_1266c_1267c_1268c_1269c_1270c_1271c_1272c_1273c_1274c_1275c_1276c_1277c_1278c_1279c_1280c_1281c_1282c_1283c_1284c_1285c_1286c_1287c_1288c_1289c_1290c_1291c_1292c_1293c_1294c_1295c_1296c_1297c_1298c_1299c_1300c_1301c_1302c_1303c_1304c_1305c_1306c_1307c_1308c_1309c_1310c_1311c_1312c_1313c_1314c_1315c_1316c_1317c_1318c_1319c_1320c_1321c_1322c_1323c_1324c_1325c_1326c_1327c_1328c_1329c_1330c_1331c_1332c_1333c_1334c_1335c_1336c_1337c_1338c_1339c_1340c_1341c_1342c_1343c_1344c_1345c_1346c_1347c_1348c_1349c_1350c_1351c_1352c_1353c_1354c_1355c_1356c_1357c_1358c_1359c_1360c_1361c_1362c_1363c_1364c_1365c_1366c_1367c_1368c_1369c_1370c_1371c_1372c_1373c_1374c_1375c_1376c_1377c_1378c_1379c_1380c_1381c_1382c_1383c_1384c_1385c_1386c_1387c_1388c_1389c_1390c_1391c_1392c_1393c_1394c_1395c_1396c_1397c_1398c_1399c_1400c_1401c_1402c_1403c_1404c_1405c_1406c_1407c_1408c_1409c_1410c_1411c_1412c_1413c_1414c_1415c_1416c_1417c_1418c_1419c_1420c_1421c_1422c_1423c_1424c_1425c_1426c_1427c_1428c_1429c_1430c_1431c_1432c_1433c_1434c_1435c_1436c_1437c_1438c_1439c_1440c_1441c_1442c_1443c_1444c_1445c_1446c_1447c_1448c_1449c_1450c_1451c_1452c_1453c_1454c_1455c_1456c_1457c_1458c_1459c_1460c_1461c_1462c_1463c_1464c_1465c_1466c_1467c_1468c_1469c_1470c_1471c_1472c_1473c_1474c_1475c_1476c_1477c_1478c_1479c_1480c_1481c_1482c_1483c_1484c_1485c_1486c_1487c_1488c_1489c_1490c_1491c_1492c_1493c_1494c_1495c_1496c_1497c_1498c_1499c_1500c_1501c_1502c_1503c_1504c_1505c_1506c_1507c_1508c_1509c_1510c_1511c_1512c_1513c_1514c_1515c_1516c_1517c_1518c_1519c_1520c_1521c_1522c_1523c_1524c_1525c_1526c_1527c_1528c_1529c_1530c_1531c_1532c_1533c_1534c_1535c_1536c_1537c_1538c_1539c_1540c_1541c_1542c_1543c_1544c_1545c_1546c_1547c_1548c_1549c_1550c_1551c_1552c_1553c_1554c_1555c_1556c_1557c_1558c_1559c_1560c_1561c_1562c_1563c_1564c_1565c_1566c_1567c_1568c_1569c_1570c_1571c_1572c_1573c_1574c_1575c_1576c_1577c_1578c_1579c_1580c_1581c_1582c_1583c_1584c_1585c_1586c_1587c_1588c_1589c_1590c_1591c_1592c_1593c_1594c_1595c_1596c_1597c_1598c_1599c_1600c_1601c_1602c_1603c_1604c_1605c_1606c_1607c_1608c_1609c_1610c_1611c_1612c_1613c_1614c_1615c_1616c_1617c_1618c_1619c_1620c_1621c_1622c_1623c_1624c_1625c_1626c_1627c_1628c_1629c_1630c_1631c_1632c_1633c_1634c_1635c_1636c_1637c_1638c_1639c_1640c_1641c_1642c_1643c_1644c_1645c_1646c_1647c_1648c_1649c_1650c_1651c_1652c_1653c_1654c_1655c_1656c_1657c_1658c_1659c_1660c_1661c_1662c_1663c_1664c_1665c_1666c_1667c_1668c_1669c_1670c_1671c_1672c_1673c_1674c_1675c_1676c_1677c_1678c_1679c_1680c_1681c_1682c_1683c_1684c_1685c_1686c_1687c_1688c_1689c_1690c_1691c_1692c_1693c_1694c_1695c_1696c_1697c_1698c_1699c_1700c_1701c_1702c_1703c_1704c_1705c_1706c_1707c_1708c_1709c_1710c_1711c_1712c_1713c_1714c_1715c_1716c_1717c_1718c_1719c_1720c_1721c_1722c_1723c_1724c_1725c_1726c_1727c_1728c_1729c_1730c_1731c_1732c_1733c_1734c_1735c_1736c_1737c_1738c_1739c_1740c_1741c_1742c_1743c_1744c_1745c_1746c_1747c_1748c_1749c_1750c_1751c_1752c_1753c_1754c_1755c_1756c_1757c_1758c_1759c_1760c_1761c_1762c_1763c_1764c_1765c_1766c_1767c_1768c_1769c_1770c_1771c_1772c_1773c_1774c_1775c_1776c_1777c_1778c_1779c_1780c_1781c_1782c_1783c_1784c_1785c_1786c_1787c_1788c_1789c_1790c_1791c_1792c_1793c_1794c_1795c_1796c_1797c_1798c_1799c_1800c_1801c_1802c_1803c_1804c_1805c_1806c_1807c_1808c_1809c_1810c_1811c_1812c_1813c_1814c_1815c_1816c_1817c_1818c_1819c_1820c_1821c_1822c_1823c_1824c_1825c_1826c_1827c_1828c_1829c_1830c_1831c_1832c_1833c_1834c_1835c_1836c_1837c_1838c_1839c_1840c_1841c_1842c_1843c_1844c_1845c_1846c_1847c_1848c_1849c_1850c_1851c_1852c_1853c_1854c_1855c_1856c_1857c_1858c_1859c_1860c_1861c_1862c_1863c_1864c_1865c_1866c_1867c_1868c_1869c_1870c_1871c_1872c_1873c_1874c_1875c_1876c_1877c_1878c_1879c_1880c_1881c_1882c_1883c_1884c_1885c_1886c_1887c_1888c_1889c_1890c_1891c_1892c_1893c_1894c_1895c_1896c_1897c_1898c_1899c_1900c_1901c_1902c_1903c_1904c_1905c_1906c_1907c_1908c_1909c_1910c_1911c_1912c_1913c_1914c_1915c_1916c_1917c_1918c_1919c_1920c_1921c_1922c_1923c_1924c_1925c_1926c_1927c_1928c_1929c_1930c_1931c_1932c_1933c_1934c_1935c_1936c_1937c_1938c_1939c_1940c_1941c_1942c_1943c_1944c_1945c_1946c_1947c_1948c_1949c_1950c_1951c_1952c_1953c_1954c_1955c_1956c_1957c_1958c_1959c_1960c_1961c_1962c_1963c_1964c_1965c_1966c_1967c_1968c_1969c_1970c_1971c_1972c_1973c_1974c_1975c_1976c_1977c_1978c_1979c_1980c_1981c_1982c_1983c_1984c_1985c_1986c_1987c_1988c_1989c_1990c_1991c_1992c_1993c_1994c_1995c_1996c_1997c_1998c_1999c_2000c_2001c_2002c_2003c_2004c_2005c_2006c_2007c_2008c_2009c_2010c_2011c_2012c_2013c_2014c_2015c_2016c_2017c_2018c_2019c_2020c_2021c_2022c_2023c_2024c_2025c_2026c_2027c_2028c_2029c_2030c_2031c_2032c_2033c_2034c_2035c_2036c_2037c_2038c_2039c_2040c_2041c_2042c_2043c_2044c_2045c_2046c_2047c_2048c_2049c_2050c_2051c_2052c_2053c_2054c_2055c_2056c_2057c_2058c_2059c_2060c_2061c_2062c_2063c_2064c_2065c_2066c_2067c_2068c_2069c_2070c_2071c_2072c_2073c_2074c_2075c_2076c_2077c_2078c_2079c_2080c_2081c_2082c_2083c_2084c_2085c_2086c_2087c_2088c_2089c_2090c_2091c_2092c_2093c_2094c_2095c_2096c_2097c_2098c_2099c_2100c_2101c_2102c_2103c_2104c_2105c_2106c_2107c_2108c_2109c_2110c_2111c_2112c_2113c_2114c_2115c_2116c_2117c_2118c_2119c_2120c_2121c_2122c_2123c_2124c_2125c_2126c_2127c_2128c_2129c_2130c_2131c_2132c_2133c_2134c_2135c_2136c_2137c_2138c_2139c_2140c_2141c_2142c_2143c_2144c_2145c_2146c_2147c_2148c_2149c_2150c_2151c_2
```

Amdahl's law - Strong Scaling - Test Passage à l'échelle "fort"

Accélération théorique parallélisation d'une app. pour un problème de taille de cst.

$$S_{pth} = \frac{1}{1 - P_{para} + \frac{P_{para}}{N_{bp}}},$$

$$\lim_{N_{bp} \rightarrow \infty} S_{pth} = \frac{1}{1 - P_{para}} = \frac{1}{P_{seq}}$$



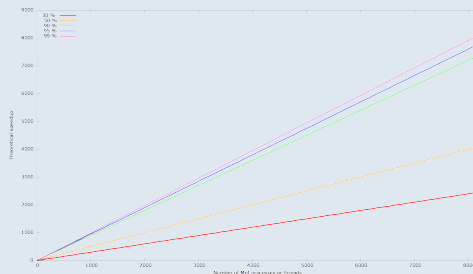
[Return to mainpages](#)

Figure: Amdahl's law, $P_{para} \in [30; 99]\%$

Gustafson-Barsis's law - Weak Scaling- Test Passage à l'échelle "faible"

Accélération théorique parallélisation pour une app. pour un pbme où la taille de chaque sous-domaine est fixe.

$$S_{p_{th}} = 1 - P_{para} + (P_{para})N_{b_p} = 1 + (N_{b_p} - 1)P_{para}$$



[Return to mainpages](#)

Figure: Gustafson's law

References I

Temporary page!

$\text{\LaTeX}$ was unable to guess the total number of pages correctly. As there was some unprocessed content that should have been added to the final page this extra page has been added to receive it. If you rerun the document (without altering it) this surplus page will go away, because $\text{\LaTeX}$ now knows how many pages to expect for this document.