High Performance Linpack (HPL)

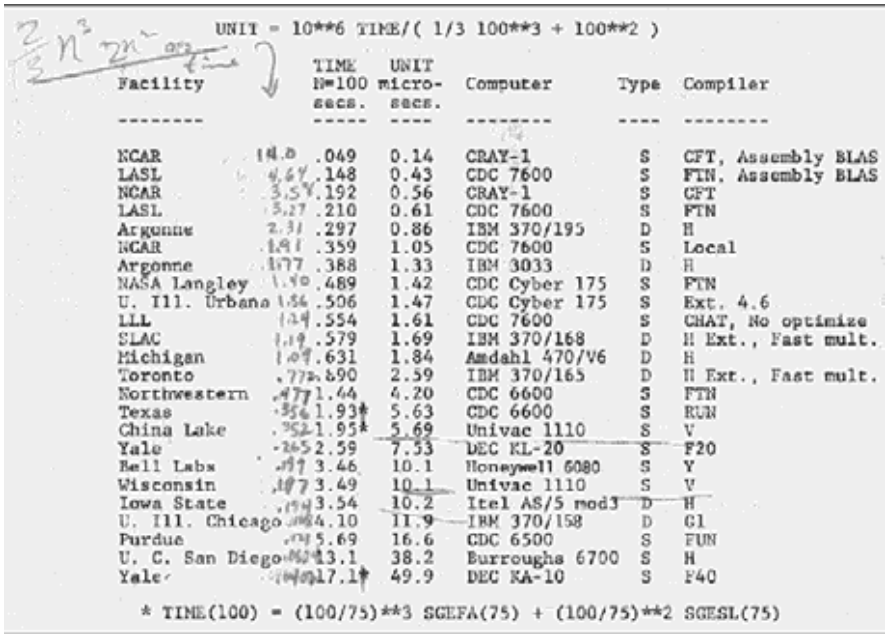
HPL – « High Performance Linpack »

- Source : http://www.netlib.org/benchmark/hpl
- Abusivement appeler linpack
- **LE** benchmark absolu ?
- Benchmark de référence servant à l'établissement du classement du top500 (www.top500.org)

- Résout un système dense d'équation linéaire *Ax+B* de dimension *NxN* sur une architecture à mémoire distribuée, en double précision.
- Le résultat brut est un temps de restitution et une performance pic (quantité absolue)
  - Elle est à ramener à la performance max. du système pour en déduire **une efficacité** (quantité relative)
    - Ce qui n'est pas fait dans le Top500 ...

Linpack

- LINear algebra PACKage (1974)
  - Fortran66
  - Historiquement Linpack une bibliothèque mathématiques pour la résolution de problèmes en algèbre linéaire.
  - Encore un benchmark par accident !
    - Annexe B du manuel de l'utilisateur : permettre aux utilisateurs d'estimer les temps de calculs de résolution de leur problème !
  - 2 benchmarks : linpack-100 (1977) , linpack-1000 (1986)
  - Vient alors HPL



UNIT = 10**6 TIME/(1/3 100**3 + 100**2)

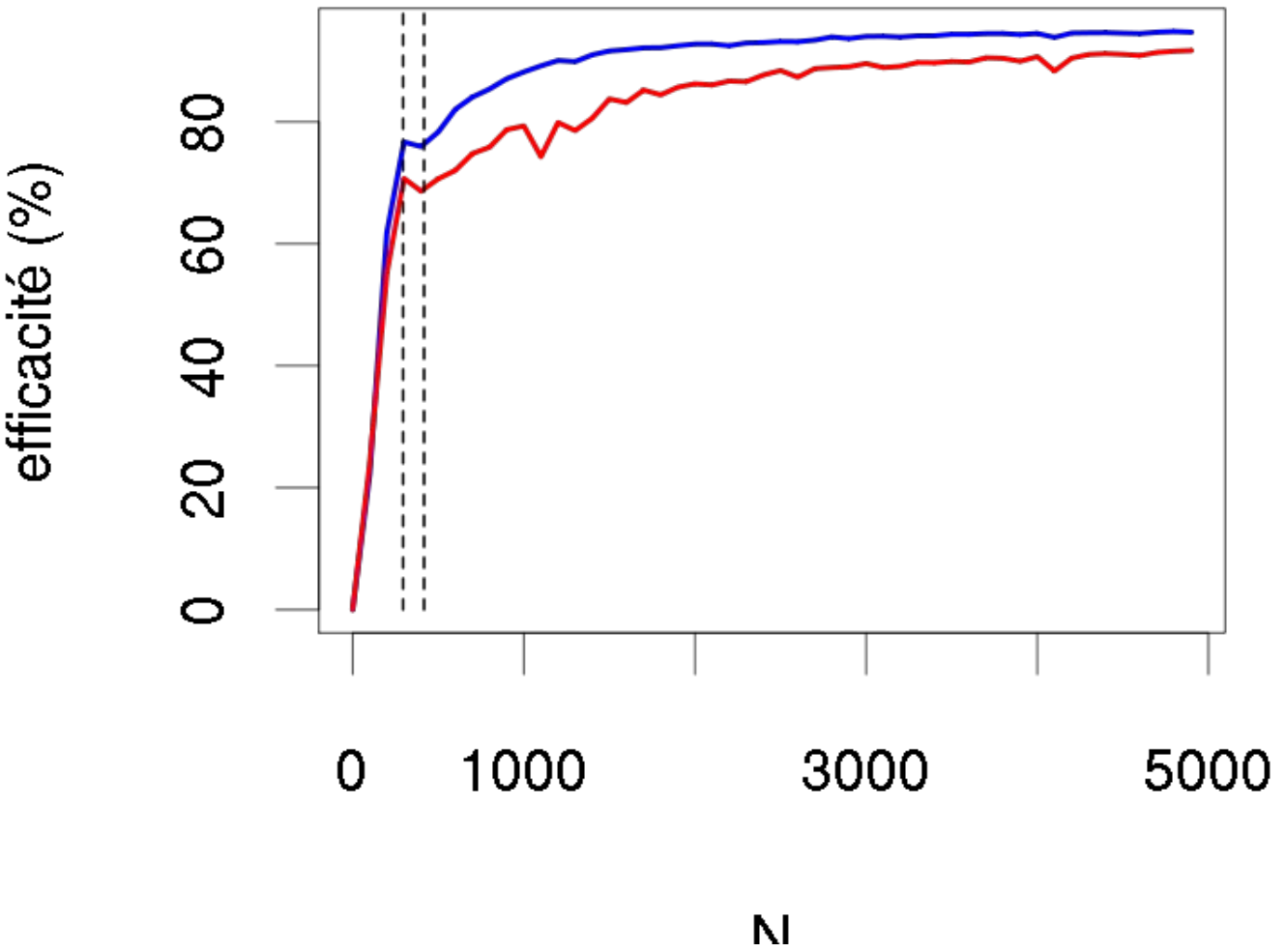
Facility	TIME N=100 secs.	UNIT micro-secs.	Computer	Type	Compiler
NCAR	.049	0.14	CRAY-1	S	CFT, Assembly BLAS
LASL	.148	0.43	CDC 7600	S	FTN, Assembly BLAS
NCAR	.192	0.56	CRAY-1	S	CFT
LASL	.210	0.61	CDC 7600	S	FTN
Argonne	.297	0.86	IBM 370/195	D	H
NCAR	.359	1.05	CDC 7600	S	Local
Argonne	.388	1.33	IBM 3033	D	H
NASA Langley	.489	1.42	CDC Cyber 175	S	FTN
U. Ill. Urbana	.506	1.47	CDC Cyber 175	S	Ext. 4.6
LLL	.554	1.61	CDC 7600	S	CHAT, No optimize
SLAC	.579	1.69	IBM 370/168	D	H Ext., Fast mult.
Michigan	.631	1.84	Amdahl 470/V6	D	H
Toronto	.690	2.59	IBM 370/165	D	H Ext., Fast mult.
Northwestern	1.44	4.20	CDC 6600	S	FTN
Texas	1.93*	5.63	CDC 6600	S	RUN
China Lake	1.95*	5.69	Univac 1110	S	V
Yale	2.59	7.53	DEC KL-20	S	F20
Bell Labs	3.46	10.1	Honeywell 6080	S	Y
Wisconsin	3.49	10.1	Univac 1110	S	V
Iowa State	3.54	10.2	Itel AS/5 mod3	D	H
U. Ill. Chicago	4.10	11.9	IBM 370/158	D	G1
Purdue	5.69	16.6	CDC 6500	S	FUN
U. C. San Diego	13.1	38.2	Burroughs 6700	S	H
Yale	17.1*	49.9	DEC KA-10	S	F40

* TIME(100) = (100/75)**3 SGEFA(75) + (100/75)**2 SGESL(75)

Performances d'un HPL

- De quoi dépendent les performances d'un HPL :
  - L'essentiel des calculs est une DGEMM (BLAS-3). De l'efficacité de la DGEMM dépend l'efficacité du Linpack
    - Les efficacité des DGEMM aujourd'hui sur des nehalem dépasse les 90%
      - → Les données sont spatialement et temporellement locales
    - C'est indirectement un bench de calcul (CPU), à condition d'une DGEMM optimale ; **c'est un benchmark de DGEMM !!!**
  - La taille du problème *N* :
    - L'efficacité d'une DGEMM (et celle comme HPL) dépendent de la taille du problème : plus il est grand, plus efficace est le calcul.
    - Plus dense est la matrice, mieux on peut recouvrir les communications par du calcul.
    - N est fonction du nombre de cœurs et de la quantité de mémoire disponible part cœur
  - De l'interconnect.

Performance d'une DGEMM avec *N*



Côté « pratique » ...

- Portage
  - Implémentation en C
  - Relativement facile
    - Makefile a adapté selon l'architecture
    - Necéssite un bibliothèque de BLAS (netlib, GotoBLAS, Intel MKL,..) ainsi qu'une implémentation MPI-1
- Execution
  - Auto-vérifiant
  - Nécessite un fichier d'entrée : HPL.dat
    - 17 paramètres importants
      - → Taille du système, des blocs, décomposition, « pattern » de communication
    - Un bon point de départ de partir de fichiers publiés pour des architecture comparable sur le site de HPCC dont il fait parti (http://icl.cs.utk.edu/hpcc/index.html)
    - Plusieurs étapes dans l'exécution d'un HPL
      - → Phase de préparation (intégrité des noeuds)
      - → Exécution d'un run moyen
      - → Exécution d'un run long (quelques %)
      - → Optimisation du fichier d'entrée, la chasse < 1%

HPL.dat : un exemple (canonique)

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
4            # of problems sizes (N)
29 30 34 35  Ns
4            # of NBs
1 2 3 4      NBs
0            PMAP process mapping (0=Row-,1=Column-major)
3            # of process grids (P x Q)
2 1 4        Ps
2 4 1        Qs
16.0         threshold
3            # of panel fact
0 1 2        PFACTs (0=left, 1=Crout, 2=Right)
2            # of recursive stopping criterium
2 4          NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
3            # of recursive panel fact.
0 1 2        RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
0            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
0            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

Détermination de N et pxq

- *p x q* : décomposition
  - '*p x q*' est forcément égal au nombre de processeurs (coeurs) utilisés avec
    - *p<q*
    - *p* et *q* proche
- *N* : dimension du système
  - Aussi grand que possible ; mais il faut tenir compte de l'empreinte mémoire MPI (peut-être importante selon les souches).
    - Si *m* est la mémoire disponible par coeur en MiB , *n* le nombre de cœurs (donc *pxq=n*) et e la fraction de mémoire que l'on souhaite utiliser. Alors :

      N = 1024 x racine (e . p . q / 8)
  - Le nombre d'opérations est connu : $F_{ops} = 2/3 \cdot N^3 + 2 \cdot N^2$ FLOPS
  - Connaissant l'efficacité d'une dgemm ε, on peut déduire le temps (en négligeant les communications !) tout au moins le borner.

Estimations ...

- Performance max du système : P_{peak} = p x q x (inst/cycle) x Freq
- Perf max. théorique du système : $P_{max} < \varepsilon P_{peak}$ secondes
- $T = F_{ops} / P_{max}$ secondes
- Ex. Titane (CCRT) – Nehalem 2.93 GHz
  - 1072 noeuds, 24 GiB par noeuds ; perf. Max. théorique : 100.5 TFLOPS/sec
  - Mémoire totale disponible : 25728 GiB
  - Pour 72.7% de mémoire : N=1024 racine(0.727*1072*24*1024)=1584438
  - Pour efficacité de 85% : 30911 secondes soit 8 h 35 min.
  - En réalité : 91.19 TFLOPS/sec soit 90.73% d'efficacité (avec turbomode).

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR01L2L4     1585024   128    67   128           29112.29              9.119e+04
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0003187 ...... PASSED
================================================================================
```

HPC Challenge (HPCC)

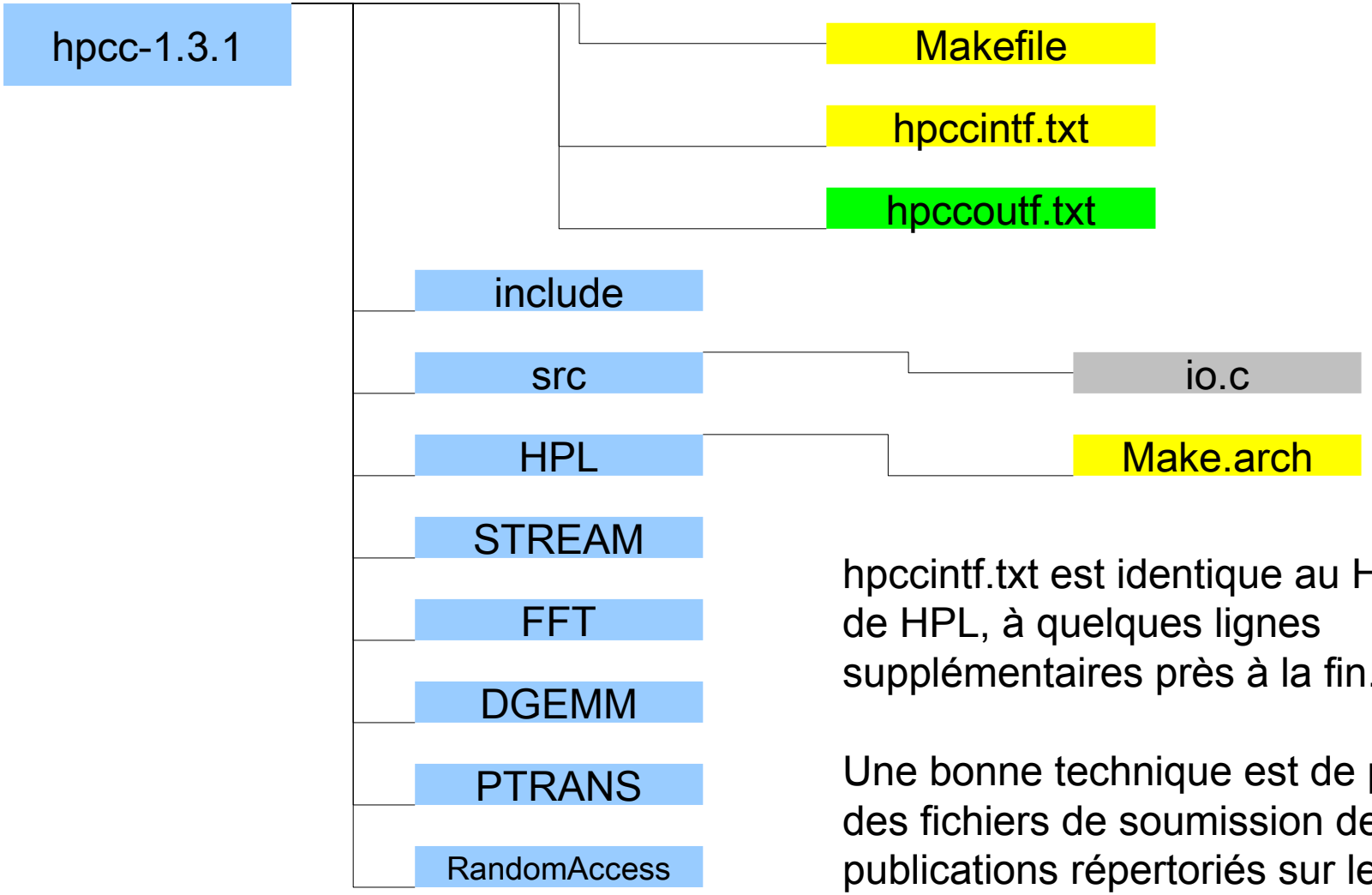
HPCC

- http://icl.cs.utk.edu/hpcc/
- HPC Chalenge
- Suite de microbenchmarks/mid-level range
  - HPL, DGEMM, Stream, PTRANS, RandomAccess, FFT, Beff
- Motivation
  - Pour étudier les performances des architectures HPV en utilisant des « kernels » plus stressants du point de vue des « pattern » d'accès mémoire que peut l'être HPL.
  - Candidat au remplacement de HPL
    - Au CPU ajouter le stress mémoire et de l'interconnect
  - Vers le pétaflopique !

HPCC

- Portage :
  - Écrit en C
  - Relativement simple ; ce base sur HPL
    - Mêmes dépendances donc : une bibliothèque de BLAS, une implémentation MPI. Non obligatoire : une implémentation de fftw (fftw.org ou Intel MKL)
- 2 types de runs
  - Pour des runs publiables :
    - « Baseline » (ou « as-is »)
    - « Optimized »
      - → Sous certaines conditions
  - 128 résultats par runs (là où HPL en fourni un seul !)
  - Les résultats sont auto-vérifiants

HPCC



hpccintf.txt est identique au HPL.dat de HPL, à quelques lignes supplémentaires près à la fin.

Une bonne technique est de partir des fichiers de soumission des publications répertoriés sur le site officiel de HPCC.

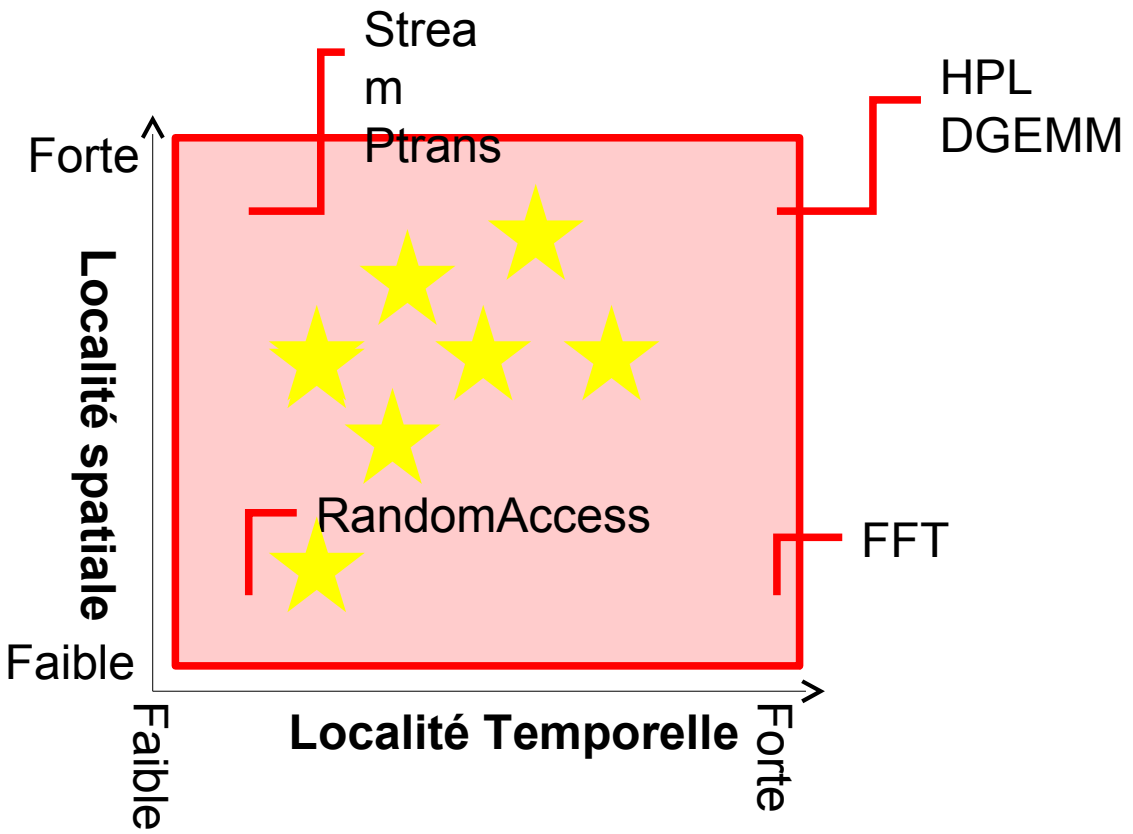
HPCC

- Certains demandent déjà (incorrigibles..) de réduire HPCC à un seul score et faisant une moyenne pondérée des 7 scores principaux :

  *Score = W1*LINPACK + W2*DGEMM + W3*FFT+ W4*Stream + W5*RA + W6*Ptrans + W7*BW/Lat*

- Quel choix pour les poids Wi ?
  - Ce qui se cache derrière c'est bien sûr le « fantasme du benchmarkeur »
    - A chaque type d'application son ensemble de {Wi} ...

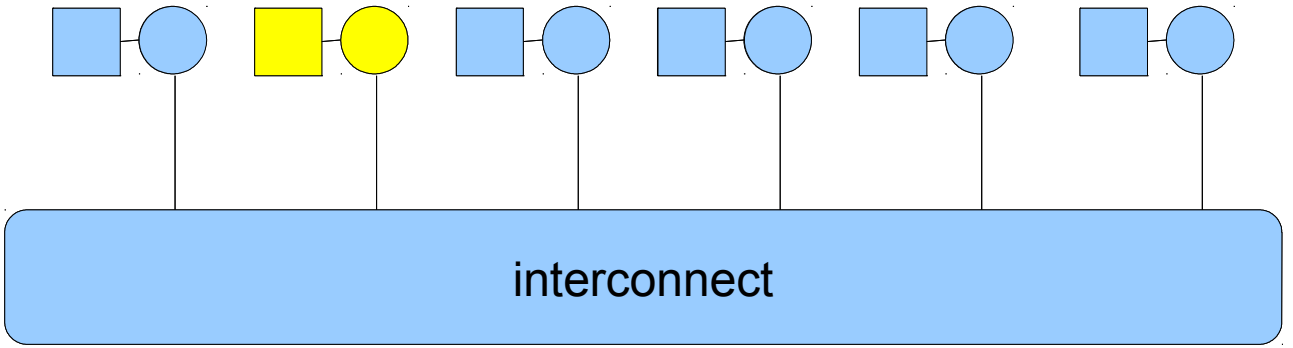
Suite de tests



- HPL
- DGEMM
- Stream
- PTRANS
- RandomAccess
- FFT
- Beff

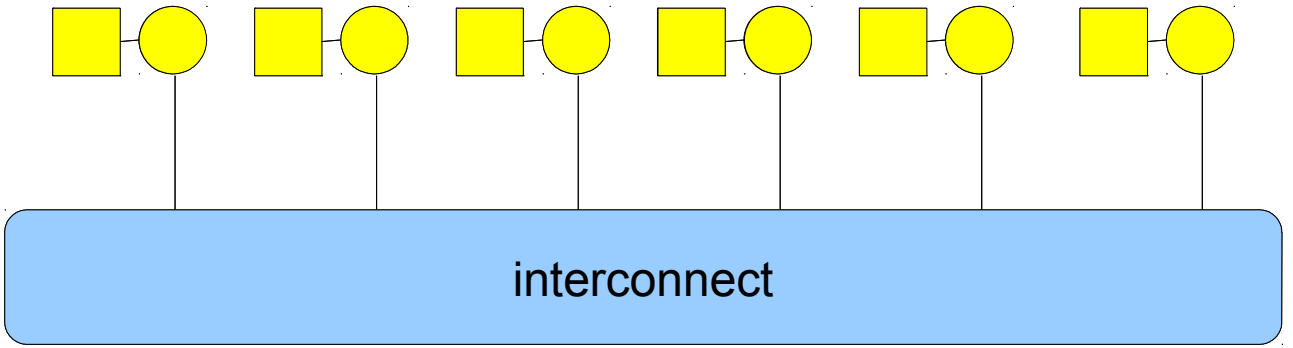
Type de runs

- Chaque code (s'il le permet) effectue 3 tests diffférents :
  - « single »
    - 1 seul processus
  - « embarassingly parallel »
    - Tous les processus indépendamment
  - « global »
    - Tous les processus partagent la charge



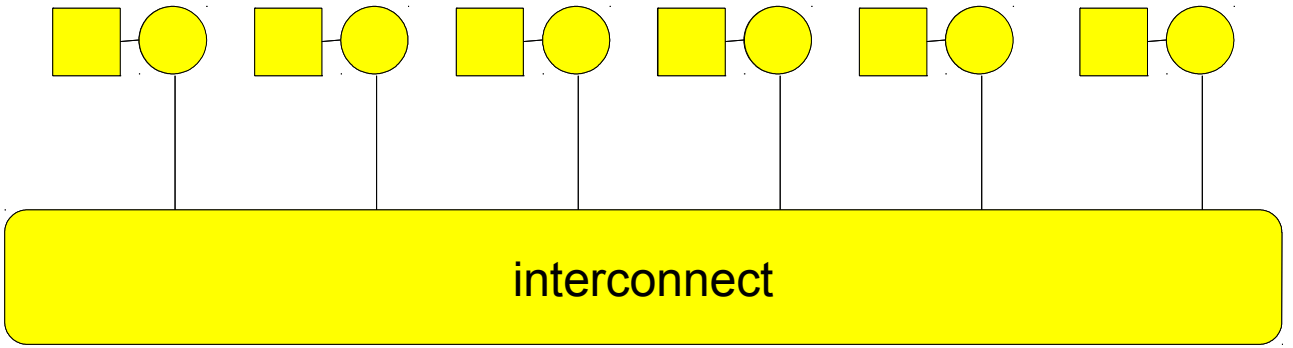
Type de runs

- Chaque code (s'il le permet) effectue 3 tests diffférents :
  - « single »
    - 1 seul processus
  - « embarassingly parallel »
    - Tous les processus indépendamment
  - « global »
    - Tous les processus partagent la charge

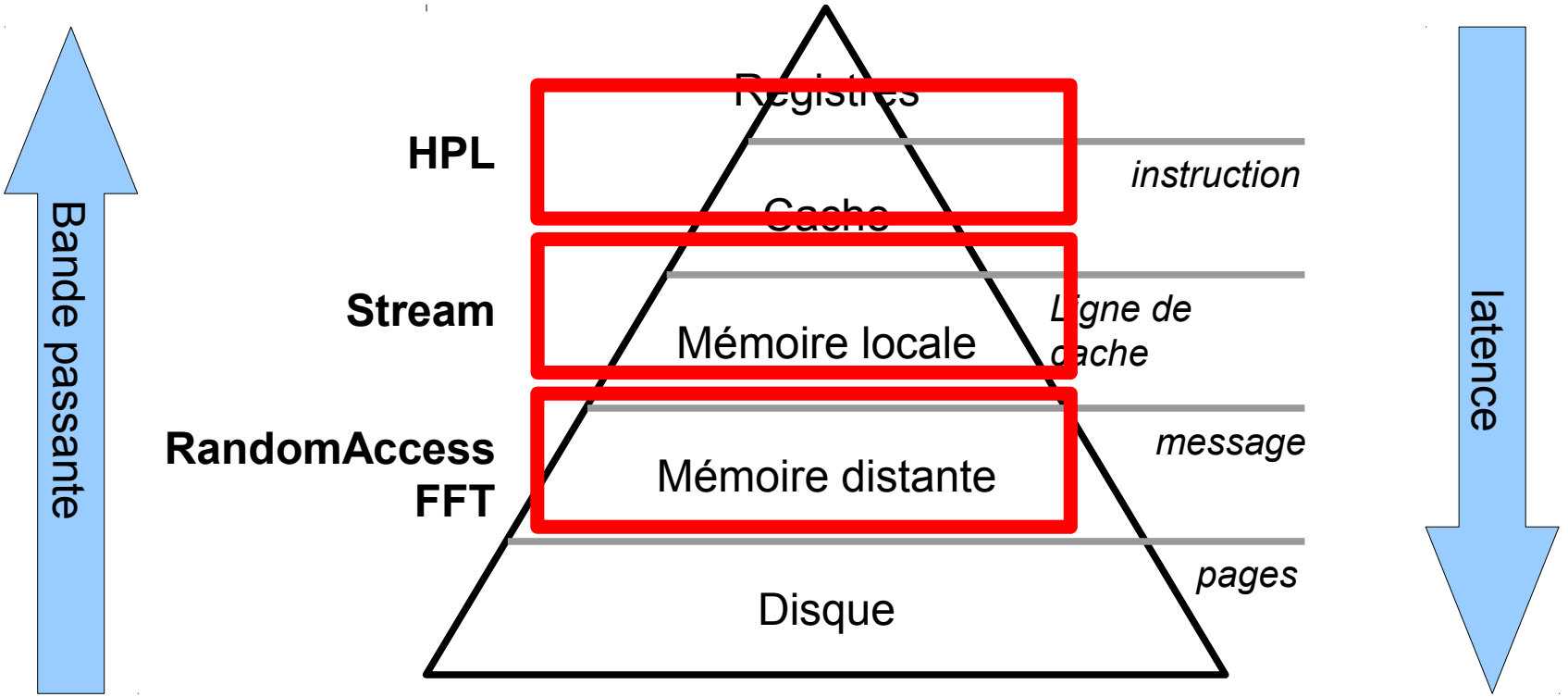


Type de runs

- Chaque code (s'il le permet) effectue 3 tests diffférents :
  - « single »
    - 1 seul processus
  - « embarassingly parallel »
    - Tous les processus indépendamment
  - « global »
    - Tous les processus partagent la charge



Et ça teste quoi ?



NAS Parallel Benchmarks

NAS Parallel benchmarks

- http://www.nas.nasa.gov/Resources/Software/npb.html
- NASA Advanced Supercomputing (NAS) Division
- 8 benchmarks, essentiellement des noyaux couramment utilisés pour la résolution de problèmes en dynamique des fluides
  - Embarrassingly parallel (EP)
  - Multigrid (MG)
  - Conjugate gradient (CG)
  - 3-D FFT PDE (FT)
  - Integer sort (IS)
  - LU solver (LU)
  - Pentadiagonal solver (SP)
  - Block tridiagonal solver (BT)

NAS Parallel benchmarks

- Portage
  - Fortran90 et C
  - Nécessite une implémentation MPI
  - Simple
    - Adapter le fichier ../config/make.def
    - Makefile

      `make <benchmark-name> NPROCS=<number> CLASS=<class>`
- Exécution
  - Simple !
  - Autovérifiant
  - Les sorties, complètes, fournissent un certain nombre de métriques, dont des MFLOPS/sec

SPEC CPU 2006 et HEPSPEC

SPEC et SPEC CPU2006

- Publié par SPEC.org :

  “The Standard Performance Evaluation Corporation (SPEC) is a non-profit corporation formed to establish, maintain and endorse a standardized set of relevant benchmarks that can be applied to the newest generation of high-performance computers. SPEC develops benchmark suites and also reviews and publishes submitted results from our member organizations and other benchmark licensees.”

- **SPEC CPU2006** : “Designed to provide performance measurements that can be used to compare compute-intensive workloads on different computer systems, SPEC CPU2006 contains two benchmark suites:
  - CINT2006 for measuring and comparing compute-intensive integer performance,
  - CFP2006 for measuring and comparing compute-intensive floating point performance.

SPEC CPU2006

- SPEC CPU2006 focuses on compute intensive performance, which means these benchmarks emphasize the performance of:
  - the **computer processor** (CPU),
  - the **memory architecture**, and
  - the **compilers**.

- It is important to remember the contribution of the latter two components. SPEC CPU performance intentionally **depends on more than just the processor**.

SPEC CPU2006 | Les différents type de benchmarks

CINT2006

400.perlbench	C	PERL Programming Language
401.bzip2	C	Compression
403.gcc	C	C Compiler
429.mcf	C	Combinatorial Optimization
445.gobmk	C	Artificial Intelligence: go
456.hmmer	C	Search Gene Sequence
458.sjeng	C	Artificial Intelligence: chess
462.libquantum	C	Physics: Quantum Computing
464.h264ref	C	Video Compression
471.omnetpp	C++	Discrete Event Simulation
473.astar	C++	Path-finding Algorithms
483.xalancbmk	C++	XML Processing

CFP2006

410.bwaves	Fortran	Fluid Dynamics
416.gamess	Fortran	Quantum Chemistry
433.milc	C	Physics: Quantum Chromodynamics
434.zeusmp	Fortran	Physics/CFD
435.gromacs	C/Fortran	Biochemistry/Molecular Dynamics
436.cactusADM	C/Fortran	Physics/General Relativity
437.leslie3d	Fortran	Fluid Dynamics
444.namd	C++	Biology/Molecular Dynamics
447.dealII	C++	Finite Element Analysis
450.soplex	C++	Linear Programming, Optimization
453.povray	C++	Image Ray-tracing
454.calculix	C/Fortran	Structural Mechanics
459.GemsFDTD	Fortran	Computational Electromagnetics
465.tonto	Fortran	Quantum Chemistry
470.lbm	C	Fluid Dynamics
481.wrf	C/Fortran	Weather Prediction
482.sphinx3	C	Speech recognition

SPEC CPU2006

- SPEC CPU2006 contains two suites that focus on two different types of compute intensive performance:
  - The ***CINT2006*** suite measures compute-intensive integer performance, and
  - The ***CFP2006*** suite measures compute-intensive floating point performance.

- Each of the suites can be used to measure along the following vectors as well:
  - **Compilation method:** Consistent compiler options across all programs of a given language (the base metrics) and, optionally, compiler options tuned to each program (the peak metrics).
  - **Speed or throughput:** Time for completion of a set of individual tasks (the SPECspeed metrics) or total throughput for a set of multiple, concurrently executing tasks (the SPECrate metrics).

SPEC CPU2006 | Exemples de Résultats

SPEC CFP2006 Result

Bull SAS

BL265
(Intel Xeon E5504, 2.00 GHz)

SPECfp2006 = 25.0

SPECfp_base2006 = 23.7

CPU2006 license:
Test sponsor:
Tested by:

L3 Cache:
Other Cache:
Memory:

Disk Subsystem:
Other Hardware:

Results Table

Benchmark	Base						Peak					
	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio
410.bwaves	204	66.7	**218**	**62.2**	219	62.0	204	66.6	**219**	**62.1**	220	61.9
416.gamess	**1278**	**15.3**	1279	15.3	1277	15.3	**1188**	**16.5**	1189	16.5	1188	16.5
433.milc	373	24.6	375	24.5	**375**	**24.5**	378	24.3	377	24.4	**378**	**24.3**
434.zeusmp	440	20.7	**445**	**20.5**	445	20.4	440	20.7	**445**	**20.5**	445	20.4
435.gromacs	499	14.3	**499**	**14.3**	499	14.3	490	14.6	492	14.5	**491**	**14.6**
436.cactusADM	104	114	110	109	**105**	**114**	102	117	**103**	**116**	103	116
437.leslie3d	445	21.1	**483**	**19.5**	485	19.4	445	21.1	**483**	**19.5**	485	19.4
444.namd	672	11.9	671	12.0	**672**	**11.9**	677	11.9	**677**	**11.8**	677	11.8
447.dealII	535	21.4	**535**	**21.4**	536	21.4	**504**	**22.7**	503	22.7	504	22.7
450.soplex	**458**	**18.2**	458	18.2	458	18.2	451	18.5	**451**	**18.5**	450	18.5
453.povray	298	17.9	**298**	**17.9**	301	17.7	234	22.8	**234**	**22.7**	235	22.7
454.calculix	**475**	**17.4**	475	17.4	475	17.4	455	18.1	455	18.1	**455**	**18.1**
459.GemsFDTD	**342**	**31.1**	341	31.1	374	28.4	**237**	**44.8**	238	44.6	236	44.9
465.tonto	606	16.2	608	16.2	**607**	**16.2**	543	18.1	**538**	**18.3**	537	18.3
470.lbm	377	36.4	**377**	**36.4**	377	36.5	377	36.4	**377**	**36.4**	377	36.5
481.wrf	**423**	**26.4**	423	26.4	425	26.3	424	26.4	**424**	**26.3**	426	26.3
482.sphinx3	900	21.7	911	21.4	**903**	**21.6**	902	21.6	994	19.6	**905**	**21.5**

Results appear in the order in which they were run. Bold underlined text indicates a median measurement.

SPEC CPU2006 | Exemples de résultats

		Results (peak)	Baseline (base)
CINT2006	Speed (single)	35.8	31.8
CFP2006		40.6	38.0
CINT2006	Rates (throughput)	251 (x7)	234 (x7.35)
CFP2006		196 (x4.82)	189 (x4.97)

- Le produit n'est pas OpenSource
- Mais:
  - Les résultats sont publiés sur le site et leur utilisation est libre de droits
  - La base est conséquente
  - C'est une bonne base de comparaisons des machines, des processeurs, des architectures, …

HEPSpec2006

- HEPSpec est un sous-ensemble de SpecCPU2006
- C'est une initiative du CERN
- Il contient tous les benchmarks C++ de SpecCPU2006:
  - FP : 444.namd, 447.dealII, 450.soplex, 453.povray
  - Entiers : 471.omnetpp, 473.astar and 483.xalancbmk.
- Les codes doivent être compilés avec gnu gcc, en mode 32-bits uniquement. Les fichiers de configuration ne peuvent pas être adaptés (*i.e.* les paramètres à passer au compilateur).
  - Seuls les binaires 32-bits doivent être utilisés, même si l'OS est 64-bits
- Chaque code doit-être exécuté en parallèle, **une copie par cœurs (ou threads)**, le résultat final est la somme de ces résultats individuels. On parle de score "Multiple Speed".