Empowered by Innovation NEC

Ecole Optimisation

Groupe Calcul et Maison de la Simulation

Benchmarking

Laurent Gatineau
laurent.gatineau@emea.nec.com
Support applicatif
NEC HPC Europe

Maison de la simulation

8 octobre 2013

Plan

- Introduction
  - Définitions
  - Objectifs des benchmarks
  - Les différentes phases d'un benchmark
- Benchmarks synthétiques et éléments d'architecture
  - Processeur
  - Mémoire et cache
  - Réseau rapide
  - Systèmes de disques
  - Accélérateur
- Benchmark applicatif
  - Portage et validation numérique
  - Classification des applications
  - Profiling – Recherche de Hot Spot
  - Projection des performances

Définitions

- Le benchmarking est une méthode qui a été développée au début des années 1980 par la société Xerox pour une prise de décision concernant un investissement lourd destiné à moderniser la gestion des stocks. Xerox s'est intéressé alors aux « meilleures pratiques de la concurrence » mais également aux pratiques dans d'autres secteurs sur le sujet étudié. La comparaison s'est finalement faite avec une firme de vente d'articles de sport par correspondance qui excellait pour la gestion des commandes.

- Benchmark: banc d'essai, référence, point de repère.

- Le « benchmarking » désigne le fait de **dresser une liste de produits ou de services**, de **définir des critères d'évaluation** de la performance et de réaliser l'étude comparative.

Définitions

- Performance crête: performance théorique (peak performance).

- Performance soutenue: performance liée à l'application.

- CPU/GPU: FLOPS: Floating point operations per second.
  - Simple ou double précision (SP / DP).
  - Préfixe décimale: 1 Gflops = 10^9 flops.

- Mémoire: bande passante, latence
  - Préfixe binaire: 1Go/s = 1GB/s = 2^{30} octets par seconde.
  - Notation CEI: 1Gi octet par seconde (GiB/s).
  - Latence en secondes (milli, micro, nano).

Auteur: L. Gatineau

Définitions

Réseau:

- Bande passante:
  - Préfixe binaire: 1Go/s = 1GB/s = 2^{30} octets par seconde.
  - Notation CEI: 1Gi octet par seconde (GiB/s).
  - Simplex / Full duplex.
  - Bits / Bytes / Octets.
- Message rate: Nombre de messages envoyés par seconde.
- Latence: ping / pingpong.

Bus PCI, liens QPI / Hyper Transport:

- Bande passante en transfert par secondes, bits ou octets par secondes.
- Préfixe décimale.
- Simple / Full duplex.

Définitions

Disque, système de fichiers:

- Bande passante: Lecture / écriture / mixte
  - Base binaire: 1Go/s = 1GB/s = 2^{30} octets par seconde.
  - Notation CEI: 1Gi octet par seconde (GiB).

  - Attention on voit apparaître des résultats en notation décimale: 1Go/s = 1GB/s = 10^9 octets par seconde.
  - Notation CEI: 1G octet par seconde (GB/s).
  - 1Go/s = 1000000000 octets par seconde.
  - 1GiB/s = 1073741824 octets par seconde.
- IOPS: I/O par seconde (quelque soit la taille des I/O).
- OPS: Opérations par seconde (création de fichiers, …)

Calculateur: throughput

- Nombre de jobs exécutés sur le calculateur en un temps donné.

Auteur: L. Gatineau

Objectifs des benchmarks

- Comparer la performance de plusieurs calculateurs:
  - Performances théoriques (processeurs, réseau, disque).
  - Temps de restitution d'un ou plusieurs codes de calcul.
  - Throughput des calculateurs.
  - Consommation des calculateurs (pleine charge, …).

- Dimensionner un calculateur:
  - Par rapport à une chaîne applicative (temps de restitution minimum, nombre de jobs à exécuter, …).
  - Par rapport au coût des licences.
  - Par rapport à une enveloppe énergétique.
  - …

Auteur: L. Gatineau

Objectifs des benchmarks

- Connaître / découvrir un calculateur:
  - Connaître les limites du calculateur.
  - Diagnostiquer un problème.

- Connaître / découvrir une application:
  - Permet de définir / dimensionner un besoin.
  - Permet de classifier une application (cpu bound / memory bound, …).
  - Connaître les limites de l'application:
    - En terme de performances (FLOPS).
    - En terme de scalabilité (nombre de nœuds / cœurs, type de réseau).
    - En terme d'entrées / sorties disque.
  - Identifier des voies d'améliorations / optimisations.
  - Projections sur de nouvelles architectures.

Les différentes phases d'un benchmark

- Coté benchmarker / vendeur:
  - Phase de portage (debug, validation numérique).
  - Prise en main de l'application:
    - Fichiers de paramètres.
    - Directives de compilation.
    - Librairies utilisées.
  - Découverte de l'application:
    - Profiling (recherche de hot spot).
    - Classification cpu bound / memory bound / IO bound.
    - Tests de scalabilité (MPI, OpenMP).
  - Définitions des objectifs:
    - Se limiter dans le temps:
      - Tester toutes les architectures (CPU, Accélérateurs, Réseaux, …) ?
      - Tester tous les compilateurs, librairies MPI, … ?
      - …
    - Peut-on modifier le code, jusqu'où ?

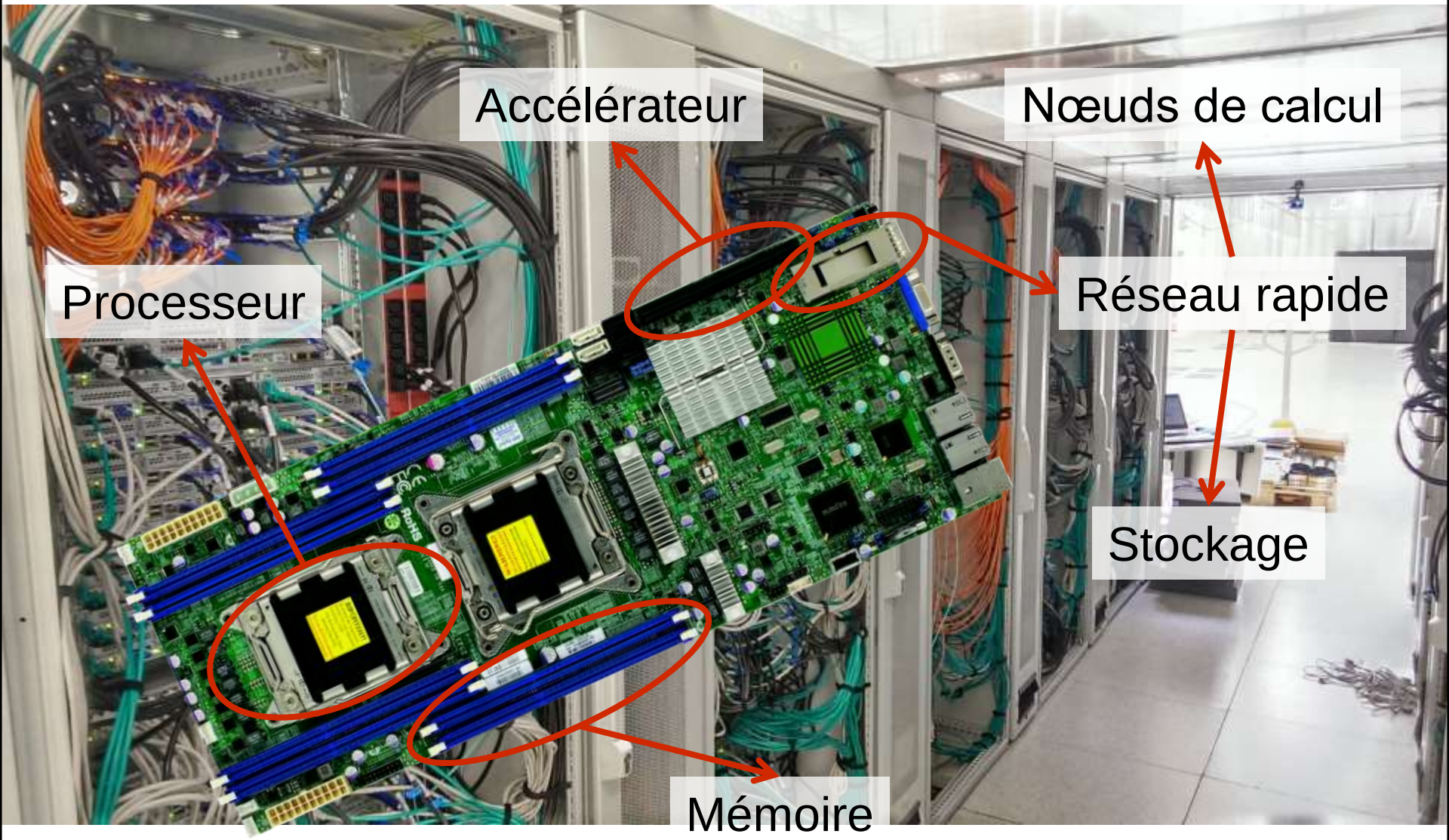
Les différentes phases d'un benchmark

- Coté chef de projet / client:
  - Définition des objectifs.
  - Identifications des benchmarks:
    - Synthétiques
    - Applicatifs
  - Identifications des interlocuteurs:
    - Disponibilités (périodes de vacances).
    - Gestion des licences (éditeurs).
  - Préparation des jeux de données:
    - Petits pour les tests de portages.
    - Dimensionner si facteur d'échelle.
    - Temps de restitution raisonnables…
  - Définir des règles:
    - Mode opératoire (nombre de rang MPI par nœud, …)
    - Autoriser les accélérateurs, les optimisations…
  - Interagir avec les benchmarkers…
  - Dépouiller les résultats…

Plan

- Introduction
  - Définitions
  - Objectifs des benchmarks
  - Les différentes phases d'un benchmark
- Benchmarks synthétiques et éléments d'architecture
  - Processeur
  - Mémoire et cache
  - Réseau rapide
  - Systèmes de disques
  - Accélérateur
- Benchmark applicatif
  - Portage et validation numérique
  - Classification des applications
  - Profiling – Recherche de Hot Spot
  - Projection des performances

Eléments d'architecture



Auteur: L. Gatineau

Processeur

Connaître les limites / caractéristiques:

- Nombre de flops.
- Taille des vecteurs.
- Taille des caches / associativité.
- Nombre d'unités flottantes, entière.
- Coût des opérations (division, nombre dé-normalisé, branchement…)

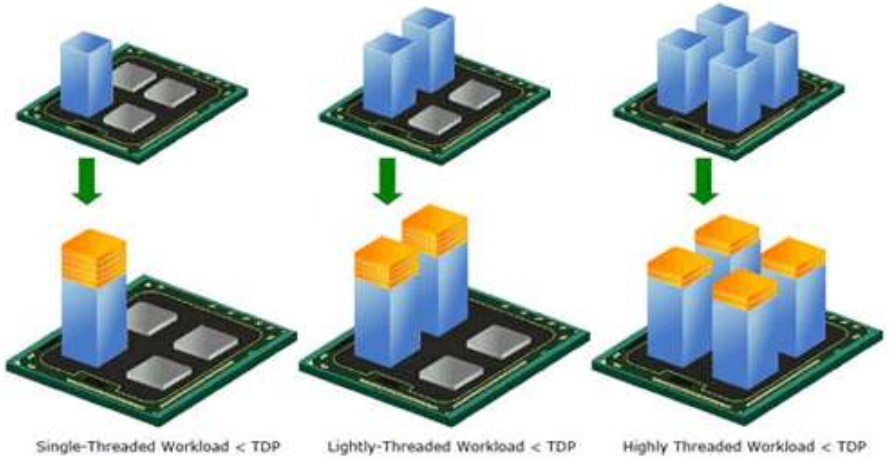
	Intel Sandy Bridge	Intel Ivy Bridge	Intel Haswell	AMD Abu Dhabi
Vecteur	256 bits	256 bits	256 bits	128/256 bits
Flop / cycle / core (DP)	8	8	8 16 (FMA3)	4/8/8 8/16/16 (FMA3)

- FMA (Fused Multiply-Add)
  - d = a + b x c
  - FMA3: d est le même registre que a, b ou c.

Auteur: L. Gatineau

Processeur

- Connaître les limites / caractéristiques: un mot sur le mode Turbo…
  - Augmentation de la fréquence d'un ou plusieurs « bin »:
    - Si l'enveloppe thermique est inférieure au TDP.
    - Possibilité de dépassement du TDP pendant un certain laps de temps.
  - Attention aux mesures avec le mode Turbo:
    - Performances biaisées si tous les cœurs ne sont pas utilisés.
    - Le refroidissement des nœuds calcul peut avoir un impact sur la performance.
- i7z:
  - Mesure temps réel de la fréquence des cœurs.
  - Nécessite accès à /dev/cpu/*/msr



Auteur: L. Gatineau

Processeur

- Performance crête d'un processeur:
  flops par cycle x fréquence x # cœurs.

- Performance d'une application:
  - Compter les opérations flottantes manuellement…
  - Utiliser un outil de profiling.
    - Nombre de cycles par instructions (CPI).
    - Nombre d'opérations flottantes:
      - Opérations x87 (co-processeur)
      - Opérations vectorielles (SIMD, SSE, AVX, …)
      - Simple / double précision
    - Taux de vectorisation.
    - Nombre de branchements ratés.
    - Ratio par rapport au nombre d'instructions exécutées.

Auteur: L. Gatineau

Processeur

Avec PAPI:

- PAPI_FP_OPS: Opérations flottantes (x87)
- PAPI_VEC_SP: Opérations vectorielles (SIMD SP)
- PAPI_VEC_DP: Opérations vectorielles (SIMD DP)

```
#include <papi.h>

int events[NUM_EVENTS] = {PAPI_FP_OPS, PAPI_VEC_SP};
long long values[NUM_EVENTS];

PAPI_library_init(PAPI_VER_CURRENT);
PAPI_start_counters(events, NUM_EVENTS);

<calcul>

PAPI_stop_counters(values, NUM_EVENTS);

printf("PAPI_FP_OPS = %lld\n", values[0]);
printf("PAPI_VEC_SP = %lld\n", values[1]);

PAPI_shutdown();
```

```
$ ./matmul_papi
PAPI_FP_OPS = 0
PAPI_VEC_SP = 16533370328
Elapse=0.650981 s
$
```


Auteur: L. Gatineau

Processeur

- Avec perf:
  - Non intrusif.
  - Evénements prédéfinis (cycle, instruction, branchements, cache…).
  - Evénements ‘raw’
  - Statistiques globales ou profile (avec annotation du code).

```
$ perf stat --event r0111,r2010 -- ./matmul

 Performance counter stats for './matmul':

       2066722610 r0111
                8 r4010

      0.704361737 seconds time elapsed

$
```


Auteur: L. Gatineau

Processeur

```
$ papi_avail -d
[…]
PAPI_VEC_SP     0x80000069      2       |SP Vector/SIMD instr|
 |Single precision vector/SIMD instructions|
 ||
 |DERIVED_POSTFIX|
 |N0|4|*|N1|8|*|+||
 Native Code[0]: 0x4000001e |FP_COMP_OPS_EXE:SSE_PACKED_SINGLE|
 Native Code[1]: 0x4000001f |SIMD_FP_256:PACKED_SINGLE|
[…]

$ perf stat --event r0111,r2010 -- ./matmul
```

Table 19-5. Non-Architectural Performance Events In the Processor Core of 3rd Generation Intel® Core™ i7, i5, i3 Processors (Contd.)

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
10H	40H	FP_COMP_OPS_EXE.SSE_PACKED SINGLE	Counts number of SSE* or AVX-128 single precision FP packed uops executed.	
11H	01H	SIMD_FP_256.PACKED_SINGLE	Counts 256-bit packed single-precision floating-point instructions.	

Intel® 64 and IA-32 Architectures Software Developer's Manual. Volume 3B: System Programming Guide, Part 2

Auteur: L. Gatineau

Processeur

- Matmul: Multiplication de 2 matrices 2000x2000

```
for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            s = 0.0;
            for (k = 0; k < N; k++) {
                s = s + mat1[i][k] * mat2[k][j];
            }
            result[i][j] = s;
        }
    }
```

- Nombre d'opérations: 2 x N x N x N = 16000000000
- PAPI_VEC_SP = 16533370328
- Perf: r0111 = 2066722610
  r4010 = 8

(2066722610+8) x (256 / 32) = 16533780944

Auteur: L. Gatineau

Processeur

Benchmarks synthétiques:

- SPECint / SPECfp:
  - Payant, résultats librement accessibles.
  - Intéressant distinction Entier / Flottant.
  - Plusieurs profils d'application.
  - Difficile à lire, beaucoup de résultats, de configurations:
    - SPEC[Int|fp]: orienté vitesse (généralement sur 1 cœur).
    - SPEC[Int|fp]_rate: orienté throughput (nœud global).
    - SPEC[Int|fp]_rate_base: options de compilations conservatrices.

- HPL:
  - Libre.
  - DGEMM uniquement (Double General Matrix Multiply)…
  - Classement bi-annuel (TOP500)…
  - Bien utile pour mesurer la consommation…
  - Remplacé par HPCG ??? (High Performance Conjugate Gradient)

Processeur

Benchmarks synthétiques:

- NAS Parallel Benchmark (NPB):
  - Libre. Peut-être adapté par les constructeurs.
  - Regroupes plusieurs benchmarks.
  - Pas uniquement CPU (mémoire, réseau, I/O).
  - Plusieurs tailles de problèmes.

- HPC Challenge benchmark (HPCC):
  - Libre.
  - Regroupes plusieurs benchmarks.
  - Pas uniquement CPU (mémoire, réseau, I/O).

- Mubench:
  - Libre.
  - Latence et débit des instructions.
  - Réellement bas niveau.

Auteur: L. Gatineau

Mémoire et cache

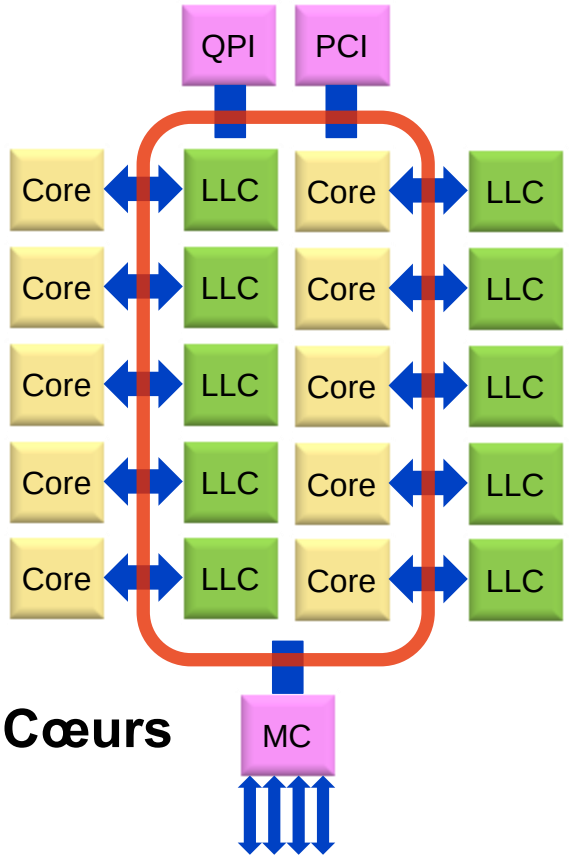
- Connaître les limites / caractéristiques mémoire:
  - Fréquence / débit et latence.
  - Interaction avec le processeur:
    - Fréquence vs nombre de modules par bancs.
    - Topologie mémoire.

Frequency (MHz)	Memory clock rate (MHz)	Bus clock multiplier	Data rate	Bus width (bit)	Peak (GB/s)	8 channels	
						Peak (GB/s)	Stream Triad (GB/s)
1066	133,25	4	2	64	8,5	68,2	53,7
1333	166,625	4	2	64	10,7	85,3	70,6
1600	200	4	2	64	12,8	102,4	79,5
1866	233,25	4	2	64	14,9	119,4	93,9

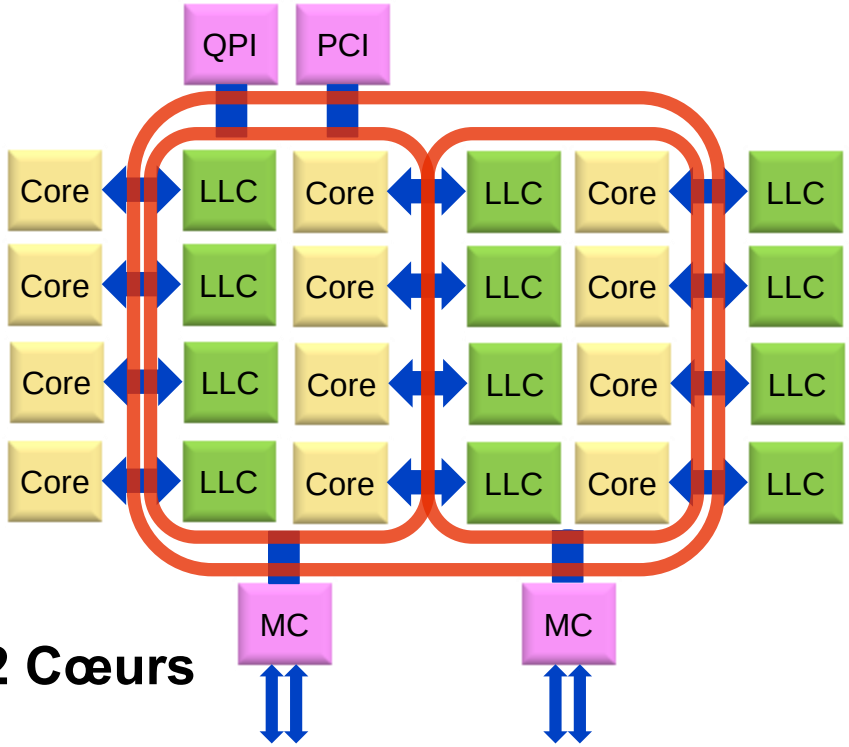
Auteur: L. Gatineau

Mémoire et cache

- Ivy Bridge 10 cœurs: 1 contrôleur mémoire (4 canaux)
- Ivy Bridge 12 cœurs: 2 contrôleurs mémoire (2 canaux)



10 Cœurs



12 Cœurs

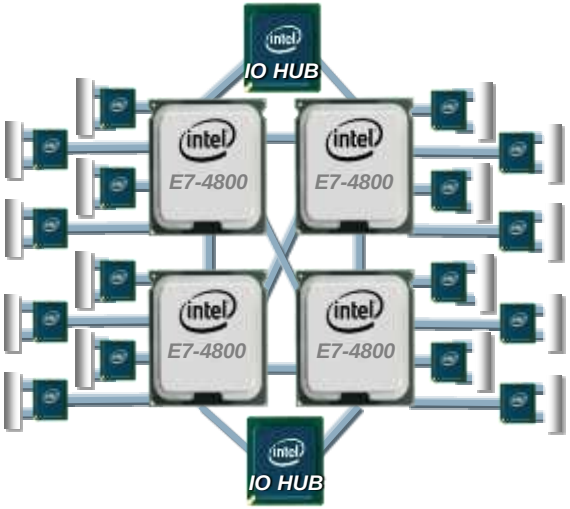
Auteur: L. Gatineau

Mémoire et cache

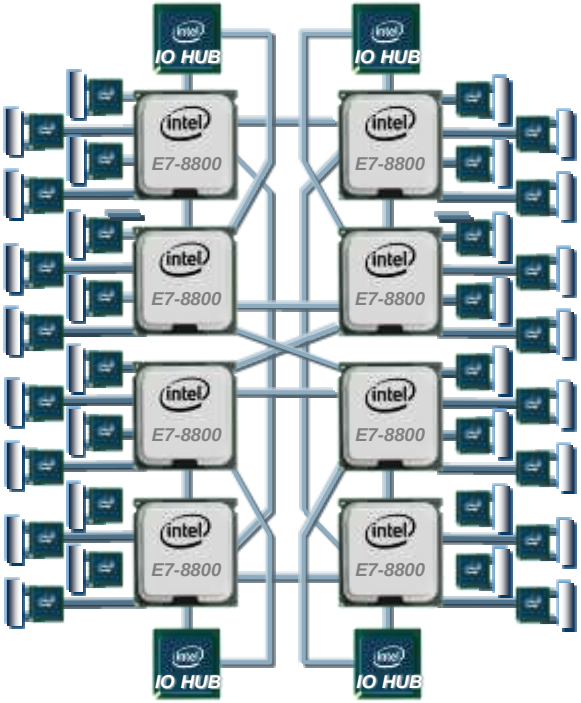
Topologie mémoire NUMA

- Exemple NEC Express5800/A1080a

Configuration 4 CPU



Configuration 8 CPU

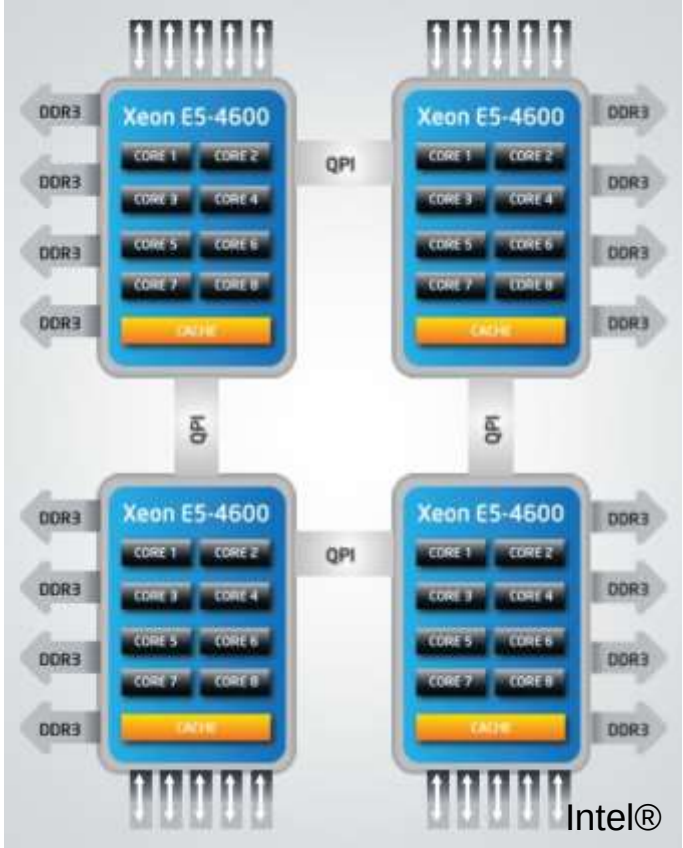


Auteur: L. Gatineau

Mémoire et cache

Topologie mémoire NUMA

- Intel Sandy Bridge / Ivy Bridge:
  - Socket B: 1 lien QPI (8GT/s – 32Go/s bidirectionnel)
  - Socket R: 2 liens QPI (8GT/s – 32Go/s bidirectionnel)



Mémoire et cache

- Connaître les limites / caractéristiques du cache:
  - Taille et nombre de niveau de cache.
  - Type de cache (Instructions, Données, les deux).
  - Partage entre cœurs / processeurs.
  - Ligne de cache, associativité.

```
$ cat /sys/devices/system/cpu/cpu0/cache/index*/type | xargs echo
Data Instruction Unified Unified
$ cat /sys/devices/system/cpu/cpu0/cache/index{0,2,3}/level | xargs echo
1 2 3
$ cat /sys/devices/system/cpu/cpu0/cache/index{0,2,3}/size | xargs echo
32K 256K 20480K
$ cat /sys/devices/system/cpu/cpu0/cache/index{0,2,3}/coherency_line_size | xargs echo
64 64 64
$ cat /sys/devices/system/cpu/cpu0/cache/index{0,2,3}/ways_of_associativity | xargs echo
8 8 20
$ cat /sys/devices/system/cpu/cpu0/cache/index{0,2,3}/number_of_sets | xargs echo
64 512 16384
$
```


Auteur: L. Gatineau

Mémoire et cache

- Connaître les limites / caractéristiques du cache:
  - Partage entre cœurs / processeurs.

```
$ /opt/likwid/3.0/bin/likwid-topology -g
--------------------------------------------------------------
CPU type:       Intel Core SandyBridge EP processor
**************************************************************
Socket 0:
+-------------------------------------------------------------------------------+
| +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ |
| | 0  16 | | 1  17 | | 2  18 | | 3  19 | | 4  20 | | 5  21 | | 6  22 | | 7  23 | |
| +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ |
| +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ |
| |  32kB | |  32kB | |  32kB | |  32kB | |  32kB | |  32kB | |  32kB | |  32kB | |
| +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ |
| +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ |
| | 256kB | | 256kB | | 256kB | | 256kB | | 256kB | | 256kB | | 256kB | | 256kB | |
| +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ +-------+ |
| +-----------------------------------------------------------------------------+ |
| |                                   20MB                                      | |
| +-----------------------------------------------------------------------------+ |
+-------------------------------------------------------------------------------+
```


Auteur: L. Gatineau

Mémoire et cache

Performance d'une application:

- Outils de profiling Perf / PAPI / …
- Cache (Miss, hits, prefetch, load, store, / L1 / L2 / LLC).
- TLB (Miss, hits, prefetch, load, store).

- False sharing: Ratio entre le nombre d'instructions retirées et le nombre de référence mémoire qui accèdent à une ligne de cache modifié sur autre cœur.

- Collision misses: Ratio entre le nombre d'instructions retirées, le nombre d'accès aux caches qui ont raté (cache miss), le nombre de cycles où le cœur/processeur est en attente (resource stall).

Auteur: L. Gatineau

Mémoire et cache

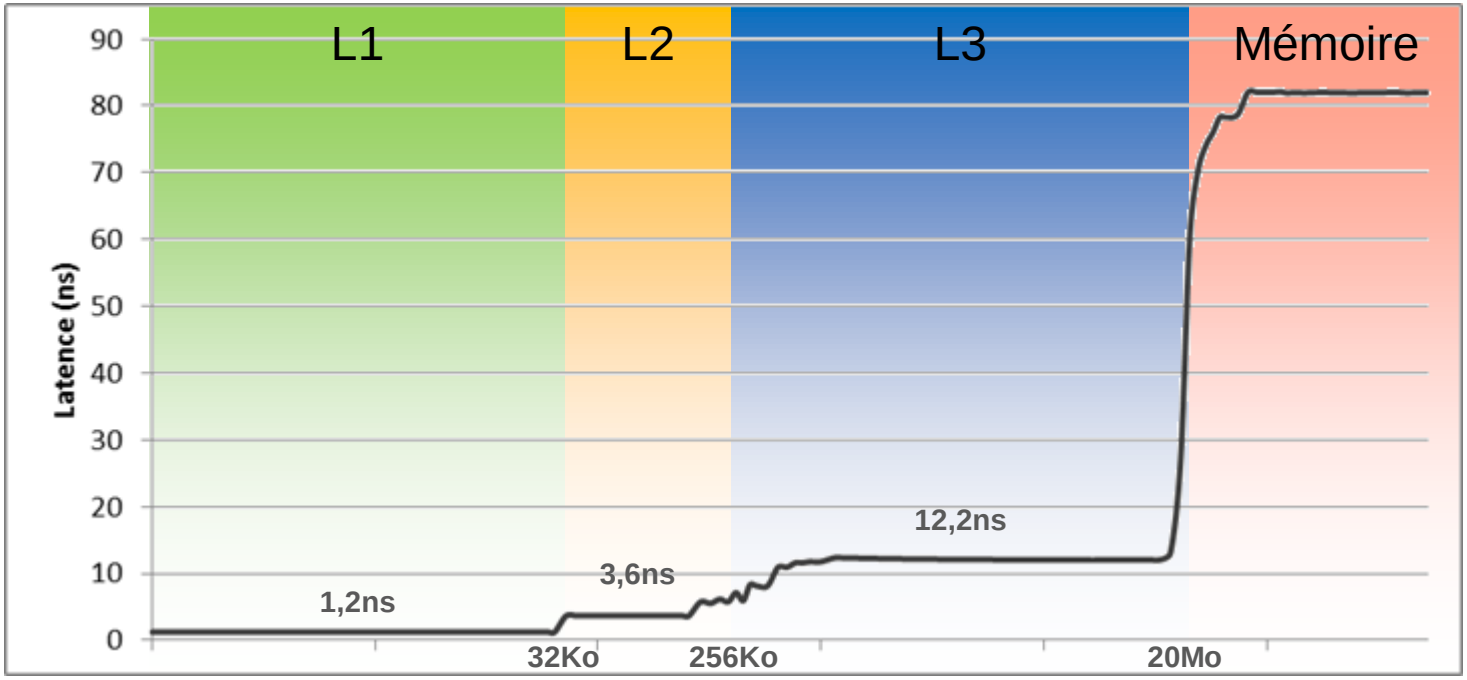
Benchmarks synthétiques:

- Stream (bande passante mémoire).
  - Libre.
  - 4 mesures: Copy, Add, Scale, Triad.
  - A adapter en fonction des processeurs (taille des données, options de compilations, …).
- LMBench (bande passante et latence mémoire).
  - Libre.
  - Plusieurs micro benchmarks.
- numatcl:
  - Permet de gérer le placement mémoire.
  - Combiner avec stream & lmbench, permet de mesurer les débits et latence des liens entre processeurs.

Auteur: L. Gatineau

Mémoire et cache

- Benchmark synthétique:
  - Lmbench: taskset 0x1 ./lat_mem_rd -N 1 -P 1 256M 512
  - Intel Sandy Bridge E5 2670 (8 cœurs @ 2.6GHz)



Auteur: L. Gatineau

Réseau rapide

- Connaître les limites / caractéristiques:
  - Débit, latence, « message rate ».
  - Topologie.
  - Débit des bus PCIe, liens QPI / HT.

	SDR 4x	DDR 4x	QDR 4x	FDR 4x	EDR 4x
Débit signal (Gb/s)	10	20	40	56	104
Encodage	8b/10b			64b/66b	
Débit utile (Gb/s)	8	16	32	54,5	100
Débit utile (GB/s)	1	2	4	6,8	12,5
Latence commutateur 24 ou 36 ports (ns)	200	140	100	170	

- Performance QDR entre PCIe Gen2 et PCIe Gen3: 3.1Go/s vs 3.7Go/s (+16%)

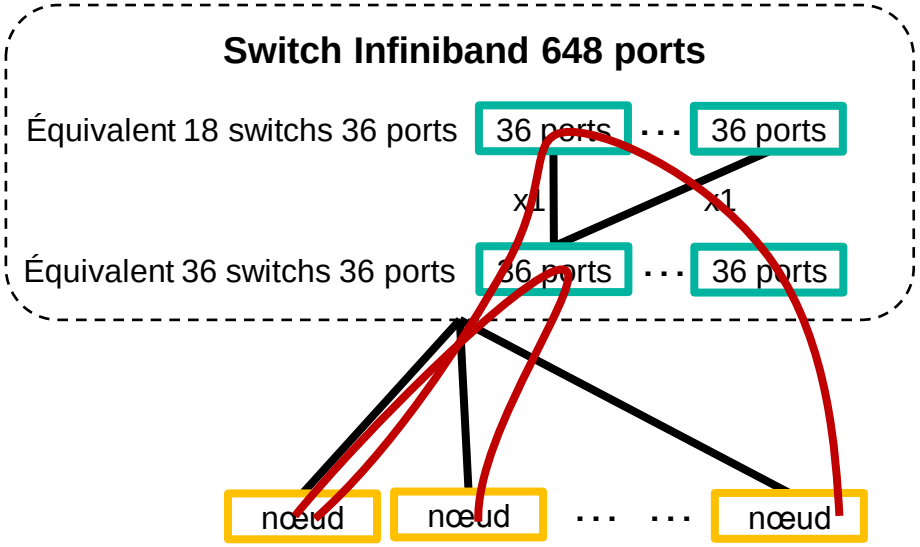
Auteur: L. Gatineau

Réseau rapide

Topologie réseau:

- Fat-tree, hypercube, Tore 3D,…
  - Nombre de switchs traversés (nombre de hop).
  - Single / Dual rail.
  - Câbles cuivre vs fibre: consommation / dissipation…

  - Attention les switchs >36 ports sont composés de plusieurs étages:



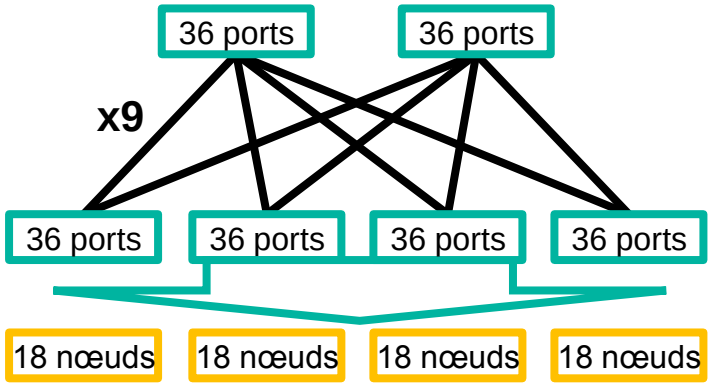
Auteur: L. Gatineau

Réseau rapide

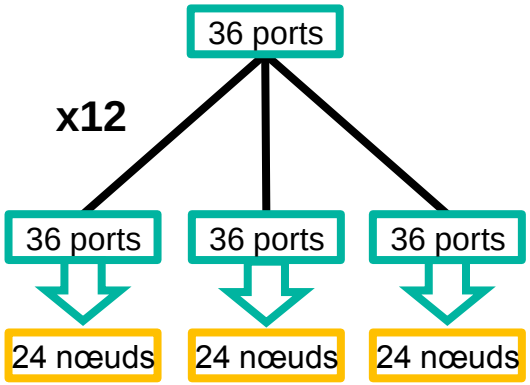
Topologie réseau:

- Facteur de blocage:
  - Permet d'augmenter le nombre de ports en réduisant le nombre de switchs.
  - Diminution de la bissection du réseau.
  - Non bloquant au sein des ilots.
  - Connaissance des ilots à travers le gestionnaire de ressources.

72 nœuds non bloquant



72 nœuds avec facteur de blocage 1:2



Auteur: L. Gatineau

Réseau rapide

Benchmarks synthétiques:

- OFED microbenchmarks:
  - Libre.
  - Test bande passante, latence (point à point).
  - Bas niveau (pas de MPI).
- OSU Micro benchmarks:
  - Libre.
  - Teste bande passante, latence, « message rate » (point à point et nœud à nœud).
  - Teste collectives MPI, one-sided communication (MPI-2).
- IMB (Intel MPI Benchmarks):
  - Libre.
  - Teste bande passante, latence.
  - Teste collectives MPI, one-sided communication (MPI-2), MPI-IO.

Auteur: L. Gatineau

Réseau rapide

Performances d'une application:

- Latence, message rate: applicatif (via un comptage manuel, via des outils de profiling MPI).

- Débit: applicatif ou système.
  - perfquery
    - fourni avec OFED
    - Besoins accès à /dev/infiniband/

  - Lecture des compteur dans /sys:
    - /sys/class/infiniband/mlx4_0/ports/1/counters/port_xmit_data
    - /sys/class/infiniband/mlx4_0/ports/1/counters/port_rcv_data

  - collectl:
    - Libre
    - Option '-s x' pour les informations relatives à Infiniband.

Auteur: L. Gatineau

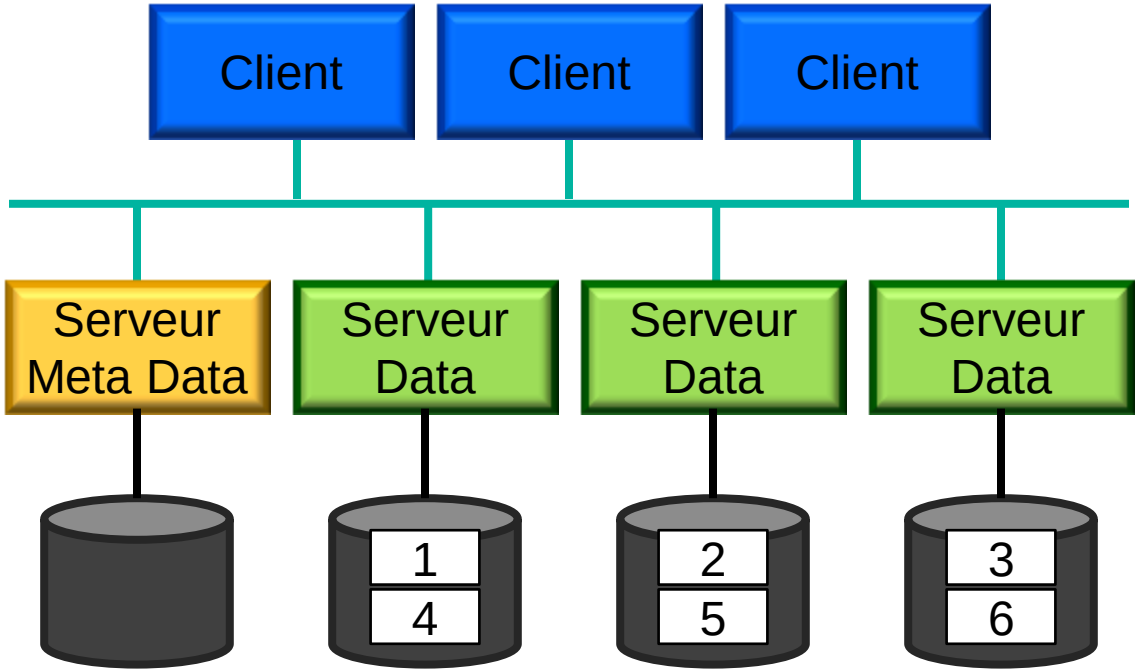
Système de disques

- Connaître les limites / caractéristiques:
  - Infrastructure:
    - Nombre de serveurs d'I/O.
    - Serveurs de méta-data dédié ou non.
    - Réseau utilisé pour l'accès aux données.
    - Type de disques, nombre de disques.
    - Si disques locaux, type de RAID.
    - Taille des caches (contrôleurs, serveurs d'I/O).
  - Système:
    - Type de système de fichiers (parallèle ou non).
    - Gestion des verrous.
    - Connecteur MPI-IO.
    - Taille des I/O.

Auteur: L. Gatineau

Système de disques

- Système de fichiers parallèle, principe de base:



- Placement des fichiers peut dépendre des applications.

Auteur: L. Gatineau

Système de disques

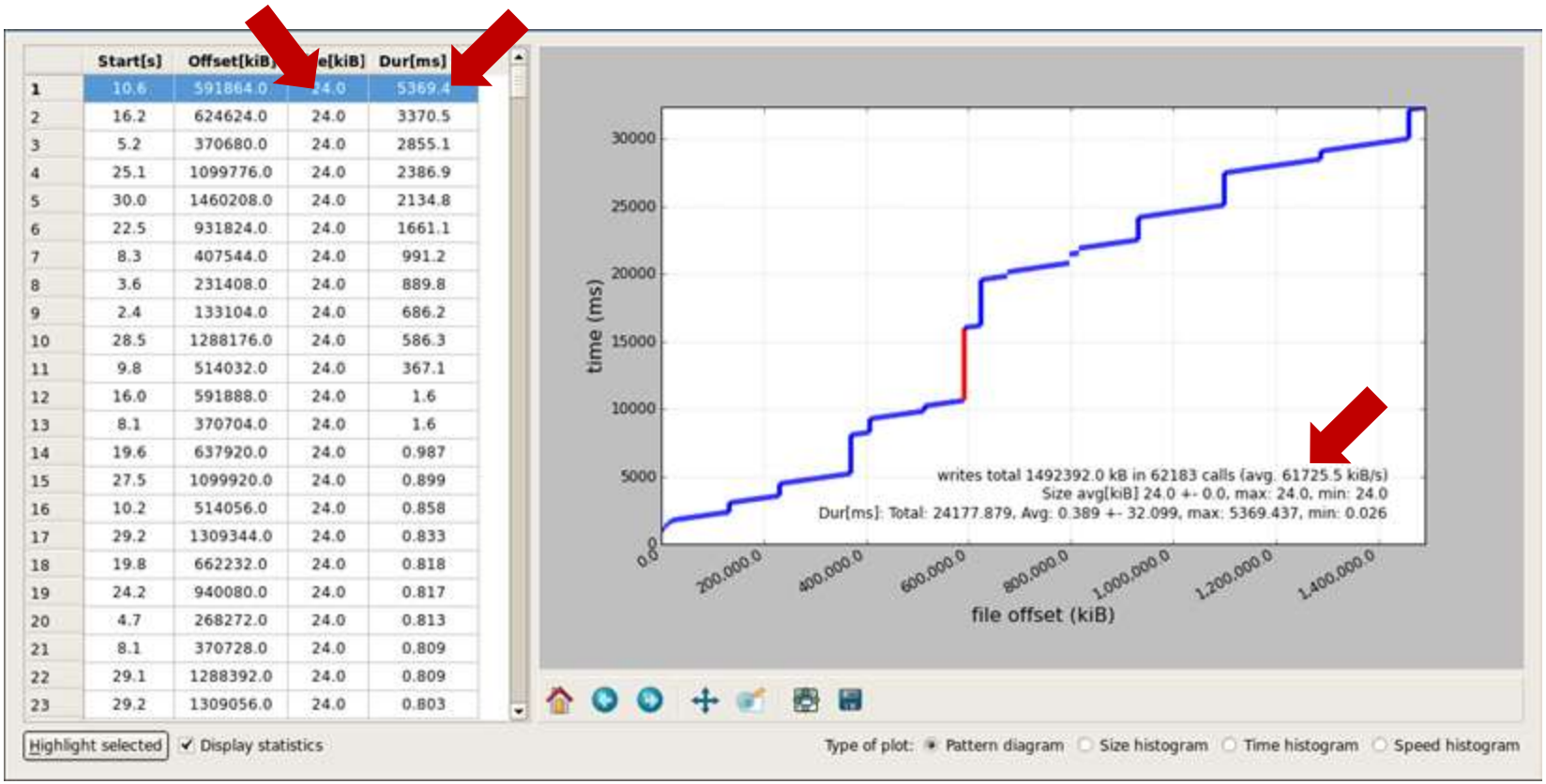
- Performances d'un système de fichiers:
  - Débit/IOPs en lecture, écriture et mode mixte.
  - Opérations de création / ouverture / fermeture de fichiers.
- Performances d'une application:
  - Comptage manuel.
  - Via le système:
    - collectl (débit / IOPS):
      - Option '–s d': disque locaux.
      - Option '-s f': NFS
      - Option '-s l': Lustre
    - strace + ioapps:
      - Trace les appels systèmes (filtre sur les appels systèmes IO).
      - Trace le profile des IOs.

Auteur: L. Gatineau

Système de disques

Exemple de profil:

- IOs continues, mais petites IOs de 24Ko.
- Temps de réponse très perturbé.

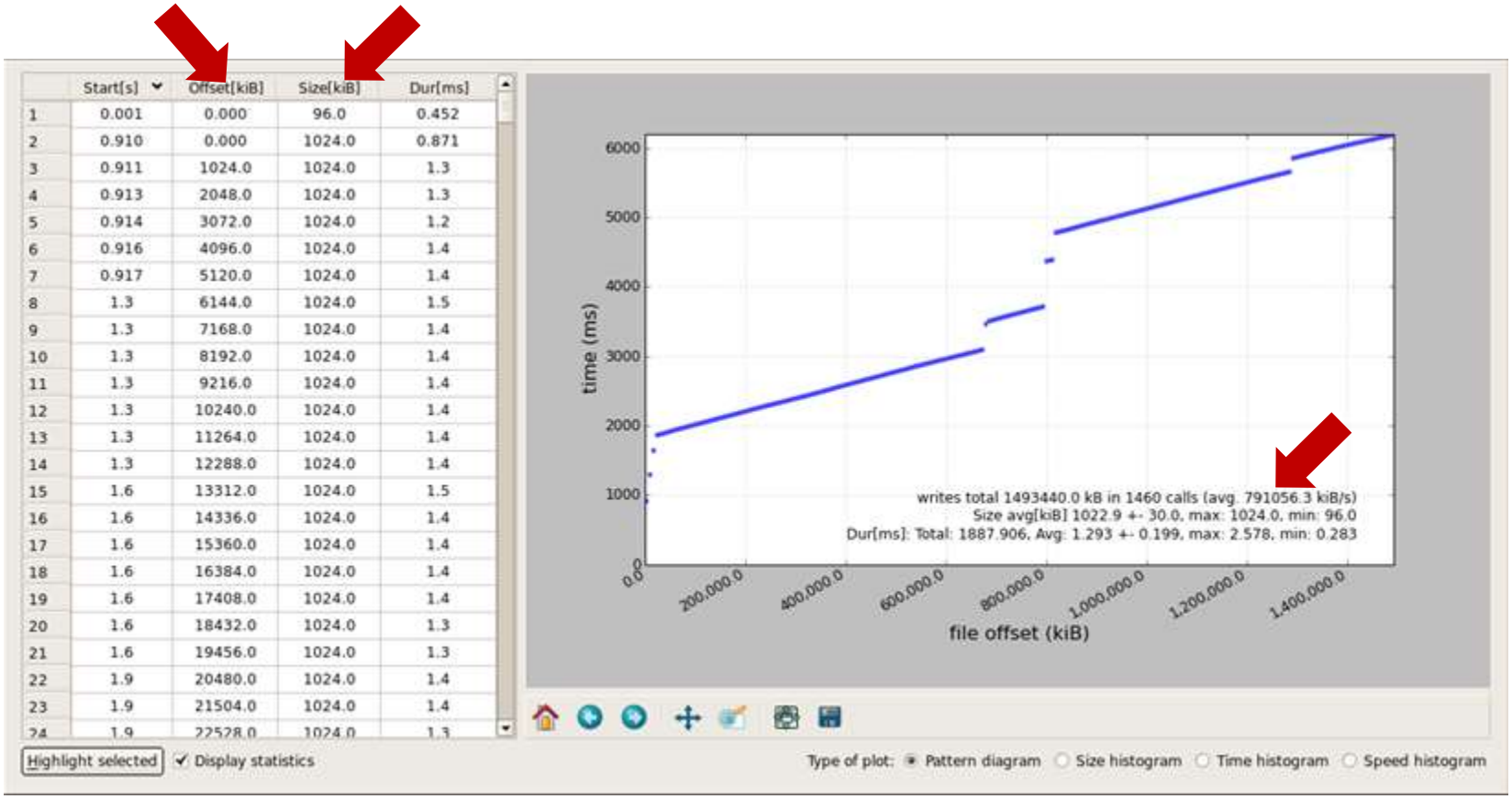


Auteur: L. Gatineau

Système de disques

Exemple de profil:

- Même cas, mais IOs agrégées.
- Meilleur tenue en charge du système de fichiers.



Auteur: L. Gatineau

Système de disques

Benchmarks synthétiques:

- dd:
  - Disponible sur tout les Unix…
- lmbench:
  - lmdd: similaire à dd.
  - lat_fs: performance création / destruction de fichiers.
- bonnie++:
  - Libre
  - Séquentiel uniquement.
  - Lecture / écriture / déplacement dans les fichiers.
  - Création / destruction / attribut de fichiers.

Auteur: L. Gatineau

Système de disques

Benchmarks synthétiques:

- iozone:
  - Libre.
  - Multithreadé / parallèle (rsh/ssh).
  - Lecture / écriture séquentielle.
  - En mode distribué pas de synchronisation des processus !!
- ior:
  - Libre.
  - Parallèle via MPI.
  - Lecture / écriture séquentielle.
  - Support Posix, MPI-IO, HDF5.
- vdbench:
  - Libre.
  - Simulation de « workload ».
  - Multithreadé / parallèle (rsh/ssh).
  - Lecture / Ecriture / Manipulation de fichiers.
  - Permet de tester le mode mixte.

Auteur: L. Gatineau

Système de disques

Benchmarker un système de fichiers:

- Reproductibilité des résultats.
- Différents niveaux de cache:
  - Contrôleurs.
  - Serveurs d'IOs
  - Clients.
- Performances globales.
- Performances d'un serveur d'IOs.
- Performance d'un nœud de calcul.
- Performance d'un stream.

Auteur: L. Gatineau

Accélérateur

- Connaître les limites / caractéristiques:
  - Nombre de threads / cœurs / unités flottantes SP ou DP / …
  - Type d'architecture SIMD ou MIMD.
  - Type de mémoire (bande passante, quantité, niveau de cache, partage entre les cœurs/threads, …).
  - Type de connexion (HT, PCIe, …).
  - Nombre d'accélérateurs par carte.

- Performances théoriques similaire entre accélérateurs.

- Probablement le plus important:
  - Nœuds standard: environ 400GFlops, 64Go de mémoire.
  - Accélérateur: environ 1TFlops, 5 à 8Go de mémoire.
  - => besoin d'échanger les données entre l'accélérateur et la machine hôte.

Accélérateur

Performance d'une application:

- GPU NVIDIA: NVIDIA Visual Profiler (libre).
- GPU ATI/AMD:
  - GPU PerfStudio 2: libre, Windows.
  - GPUPerfAPI: libre
- Intel Xeon Phi:
  - Intel Vtune Amplifier: Soumis à licence.
  - PAPI: nécessite un patch au noyau Linux.
  - Gprof, et autres outils OpenSource.

Auteur: L. Gatineau

Accélérateur

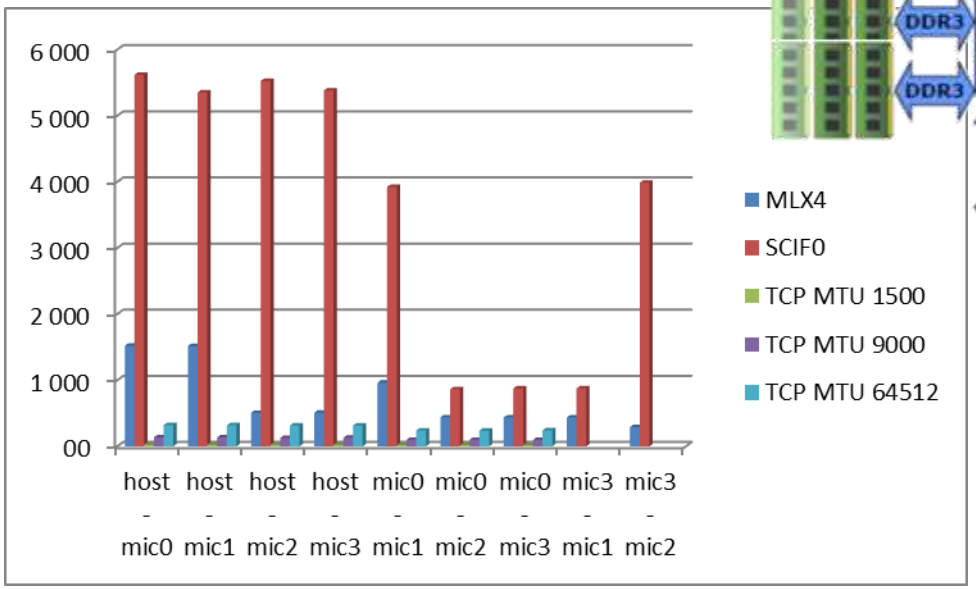
Benchmarks synthétiques:

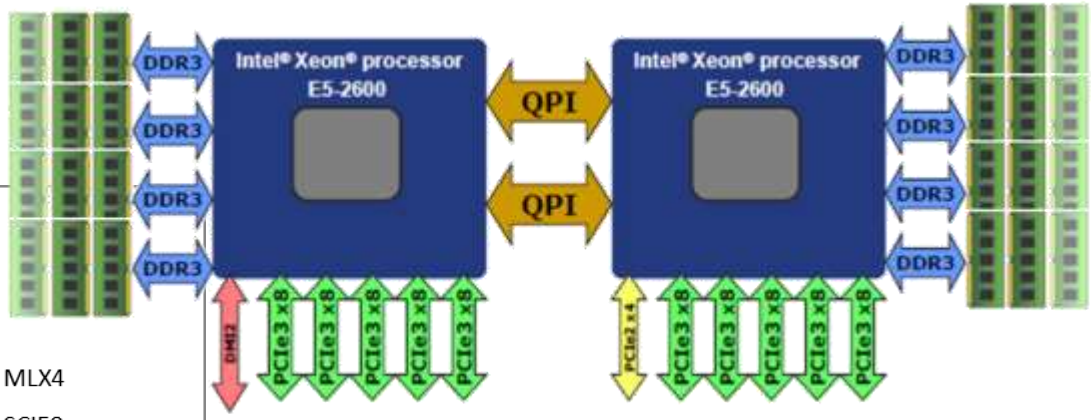
- SHOC (Scalable HeterOgeneous Computing):
  - Libre.
  - OpenCL / CUDA.
  - Ensemble de noyaux de calcul.
  - Tests bande passante mémoire (interne, GPU ⇔ host).
- HPL:
  - Version CPU+GPU (CUDA).
  - Version Intel Xeon Phi (Fourni avec Intel MKL).
- Bande passante mémoire Intel Xeon Phi:
  - Stream.
  - Benchmarks réseau (IMB / OSU).

Auteur: L. Gatineau

Accélérateur

- Importance de l'architecture des bus PCI:
  - Nombre maximum d'accélérateur par processeur.
  - GPUDirect (Peer-to-Peer): doit partager le bus PCI.
  - GPUDirect (GPU-to-Infiniband): doit partager une zone mémoire.





Auteur: L. Gatineau

Plan

- Introduction
  - Définitions
  - Objectifs des benchmarks
  - Les différentes phases d'un benchmark
- Benchmarks synthétiques et éléments d'architecture
  - Processeur
  - Mémoire et cache
  - Réseau rapide
  - Systèmes de disques
  - Accélérateur
- Benchmark applicatif
  - Portage et validation numérique
  - Classification des applications
  - Profiling – Recherche de Hot Spot
  - Projection des performances

Portage et validation numérique

- Portage phase 1: être capable de rejouer les cas tests
  - Compilation de l'application sur le système de benchmark:
    - Changement de compilateurs / options de compilations.
    - Librairies systèmes, mathématiques, gestions de données,
  - Phase de debug...
  - Validation numérique des résultats.

- Portage phase 2: classification de l'application, et premières optimisations.
  - Classification de l'application.
  - Premier profiling CPU (recherche de hot spot).
  - Changement de compilateurs.
  - Options de compilations agressives, adaptation des directives de compilation.
  - Changement de librairies mathématiques, ...
  - Placement des threads / rangs MPI.

Portage et validation numérique

Portage phase 3: optimisation.

- Profiling plus approfondi (taux de vectorisation, accès mémoire, …).
- Profiling MPI.
- Plusieurs niveaux d'optimisation:
  - Vectorisation des boucles, blocage en cache, branchement.
  - Optimisation des entrées / sorties.
  - Optimisation OpenMP / Multithread.
  - Analyse algorithmiques et structures de données.
  - Equilibrage de charge.
  - Changement de solveur.
  - …
- Validation numérique…

Auteur: L. Gatineau

Portage et validation numérique

- Validation numérique…

- La représentation des nombres réels en informatique introduit des erreurs d'arrondis:
  - Par rapport à la façon de les encoder
    - Comment représenter 1/3 ?
    - 1^{E-05} = 0.00000999999974737875163555145263671875
  - Associativité et commutativité n'existent plus:
    - Impact lors des optimisations (changement d'ordre des opérations pour optimiser le pipeline d'instructions).
    - Impact lors de la vectorisation (opération de réduction).
    - Impact lors des décompositions en sous domaine (alignement mémoire).

Auteur: L. Gatineau

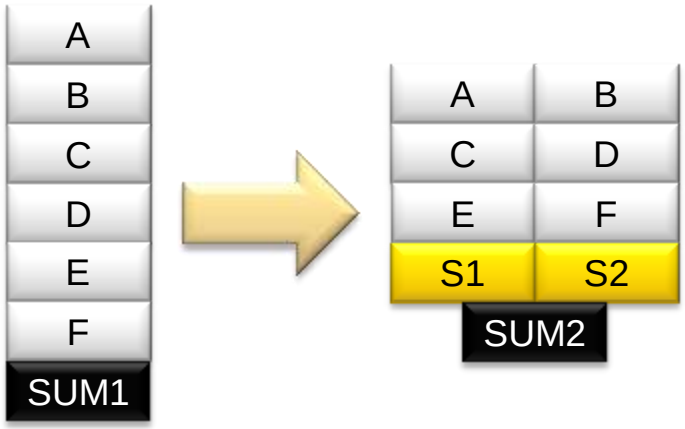
Portage et validation numérique

Optimisations du compilateur:

- Ordre des opérations pour optimiser l'utilisation des unités arithmétiques.
- Ordre des opérations pour optimiser les accès mémoires.

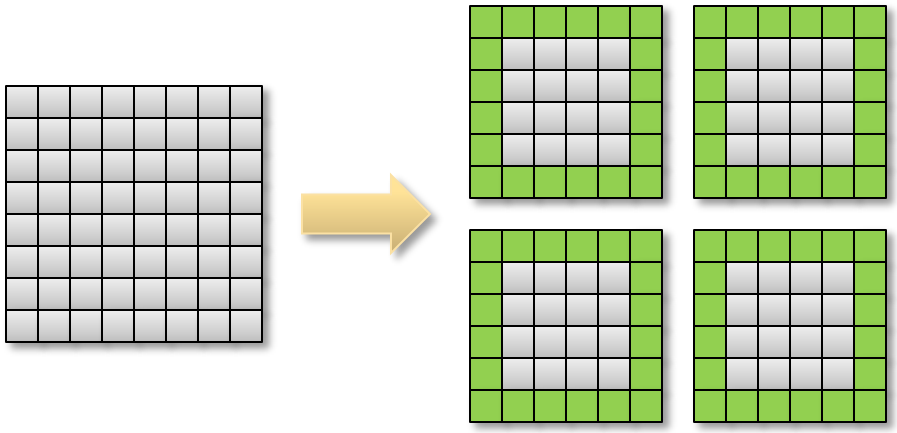
Vectorisation/SIMD (SSE,…):

- Précision 32/64 bits, n'utilise plus les registres 80bits FPE.
- Parallélisme SIMD:



Portage et validation numérique

- Parallélisation (MPI / OpenMP):
  - Opérations de réduction
  - SSE et alignement mémoire dû à la décomposition en sous domaine (ghost zone) en mémoire distribuée:



Auteur: L. Gatineau

Classification des applications (calcul)

- On peut définir 2 grandes classes d'applications:
  - Dominante CPU (cpu bound)
  - Dominante mémoire (memory bound)

- Permet d'orienter un choix d'architecture.

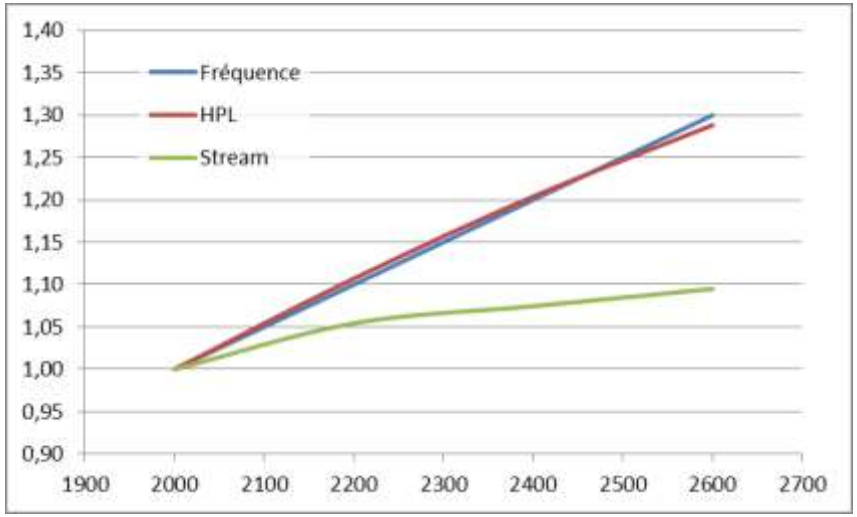
- Peut permettre:
  - De définir un placement optimal des rangs MPI / threads / …
  - Des économies d'énergie: le mode Turbo n'a peut-être pas d'intérêt si l'application est « memory bound »…

- Une application « memory bound » peut ne plus l'être à grande échelle.

Classification des applications (calcul)

- Identifier les applications « cpu bound »
- Exécuter le même cas de calcul en faisant varier la fréquence des processeurs.

Fréquence	2000	2200	2400	2600
Fréquence normalisée	1,00	1,10	1,20	1,30
HPL (Gflops)	226,2	250,4	272,3	291,4
HPL normalisé	1,00	1,11	1,20	1,29
Stream (GB/s)	65843	69413	70750	72086
Stream normalisé	1,00	1,05	1,07	1,09



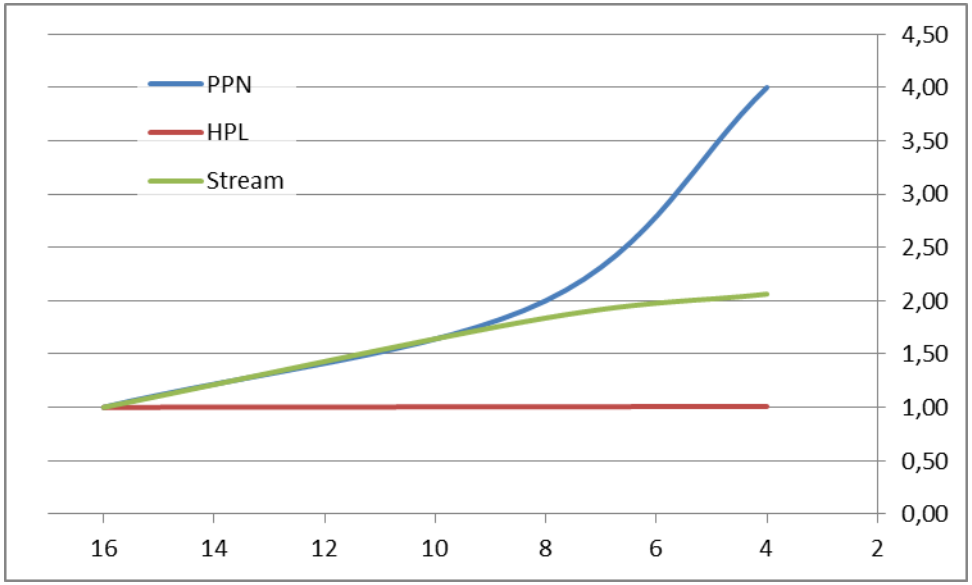
Auteur: L. Gatineau

Classification des applications (calcul)

- Identifier les applications « memory bound »

- Exécuter le même cas de calcul à nombre de cœurs constant, mais en utilisant tous les cœurs d'un nœud, 1 cœur sur 2, 1 sur 3, …

Nombre de cœurs pas nœuds (PPN)	16	8	4
PPN normalisé	1,00	2,00	4,00
HPL (Gflops)	291,4	293,1	293,9
HPL normalisé	1,00	1,01	1,01
Stream (GB/s)	72151	132586	148822
Stream normalisé	1,00	1,84	2,06



Auteur: L. Gatineau

Classification des applications (MPI)

- Identifier les besoins réseaux:
  - Latence: opérations de réduction, synchronisation.
  - Message rate: envoie massif de messages.
  - Bande passante: envoie de messages volumineux (ou grand nombre de cœurs par nœud).
  - Facteur de blocage: communications globales, de voisins à voisins, dimensionnement des calculs, …
- Pas de recette miracle…
  - Tests de scalabilité
  - Profiling
  - Bonne connaissance de l'application…

Auteur: L. Gatineau

Classification des applications (MPI)

Strong scaling:

- La taille du problème est fixé, on augmente le nombre de nœuds.

- Limite de scalabilité peut venir:
  - Du système… Réseau, librairie de passage de message, paramètre système…
  - D'un déséquilibre de la charge de travail.
  - Des entrées / sorties disques qui deviennent pénalisantes.
  - D'une portion de code séquentielle qui devient prépondérante (loi d'Amdahl).

  - De phases de communications (réduction, synchronisation, communication globale).

- Il faut pouvoir mesurer le temps passé dans les différentes phases du code.

Auteur: L. Gatineau

Classification des applications (MPI)

Weak scaling:

- La taille du problème par rang MPI est fixé, on augmente le nombre de nœuds, et donc la taille du problème.

- Limite de scalabilité peut venir:
  - Du système… Réseau, librairie de passage de message, paramètre système…
  - Des entrées / sorties disques qui deviennent pénalisantes.
  - De phases de communications (réduction, synchronisation, communication globale).

- Il faut pouvoir mesurer le temps passé dans les différentes phases du code.

Auteur: L. Gatineau

Classification des applications (MPI)

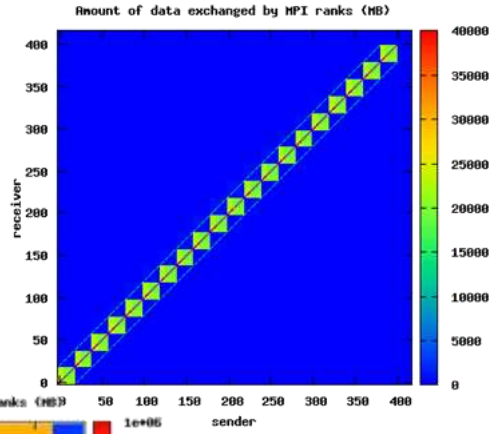
- MPI Profiling: statistiques uniquement.
  - Volume de données et coût d'instrumentation faible.
  - Temps cumulés passés dans les appels MPI.
  - Nombre de messages échangés.
  - Taille des messages échangés.
  - Matrice de communication.

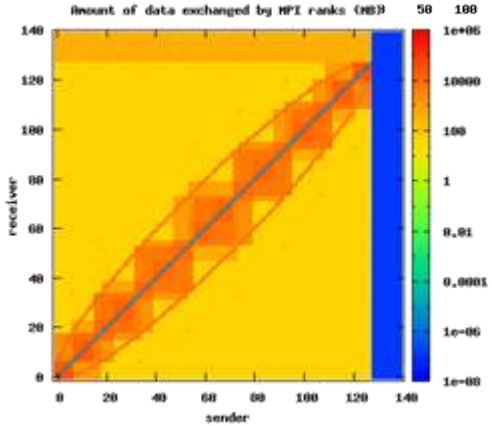
- MPI Tracing: historique des appels MPI.
  - Volume de données et coût d'instrumentation important.
  - Chronologie des appels MPI, éventuellement aussi des autres fonctions.
  - Statistiques MPI.
  - Simulateur de réseau.

Classification des applications (MPI)

MPI Profiling: statistiques uniquement.

- mpiP
  - Libre.
  - Librairie à ajouter lors de l'édition de lien.
  - Nécessite librairie système (binutils, libunwind, …) pour avoir la pile d'appel.
- IPM:
  - Libre.
  - Sans recompilation (LD_PRELOAD).
  - Profile CPU.
- IBM Platform MPI:
  - soumis à licence.
  - Sans recompilation (mpirun –i).
  - Profile CPU.

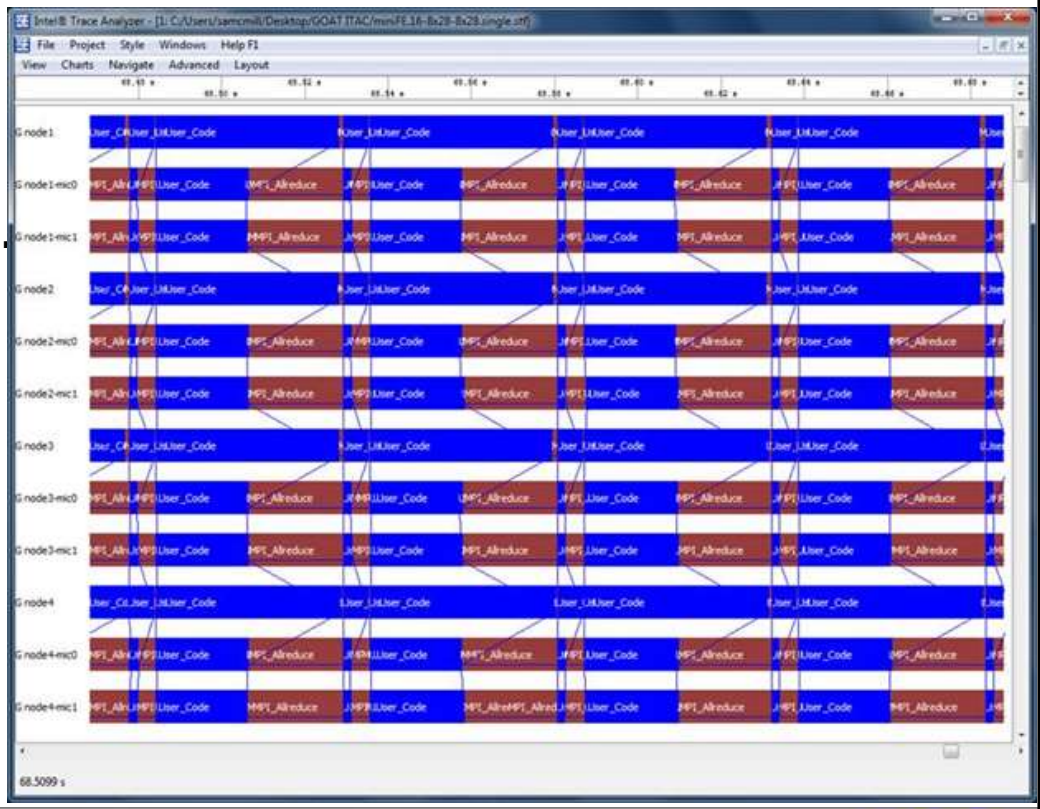




Auteur: L. Gatineau

Classification des applications (MPI)

- MPI Tracing: historique des appels MPI.
  - OpenMPI:
    - Libre.
    - Profile intégré à OpenMPI.
    - Compatible avec VampirTrace (format de fichier OTF).
  - MPE / Jumpshot:
    - Libre.
    - Compatible avec différentes librairies MPI.
  - Intel® Trace Analyzer and Collector:
    - Soumis à licence.
    - Sans recompilation.
    - Anciennement Vampir.
    - Profile CPU.
    - Simulateur.



Auteur: L. Gatineau

Profiling, recherche de Hot Spot

- Première étape, le but est d'identifier les endroits où l'on passe du temps:
  - Dans le code ?
    - Règle 90 / 10 s'applique ?
    - Etaler sur un grand nombre de fonctions ?
  - Dans une librairie ?
    - Librairie MPI.
    - Librairie Mathématique.
    - Librairie système (memcpy, memset, …)
  - Dans le noyau ?
    - Allocation mémoire ?
    - Gestion des pages mémoires ?

Profiling, recherche de Hot Spot

Compilation:

- Activer le mode ‘debug’: -g
  - Intègre des informations de debug dans les objets, dont notamment la correspondance adresse / nom de fonction.
  - Pas d’impact sur la performance.
  - Exécutable plus volumineux.

- Compilateurs Intel: -finstrument-functions
  - Instrumente les entrées / sorties de fonctions.
  - Nécessaire à certains profiler.
  - Peut avoir un impact sur la performance.
  - Désactive le mode ‘inlining’.

- Activer le mode profiling pour gprof: -p
  - Compilateurs GNU, Compilateurs Intel.
  - Open64: -profile.
  - PGI: -pg (pgprof).

Auteur: L. Gatineau

Profiling, recherche de Hot Spot

Exemple HPL / perf:

- Nécessite un petit wrapper pour MPI:

```
$ cat wrapper.sh
#!/bin/sh

RANK=${PMI_RANK:-${OMPI_COMM_WORLD_RANK:-${MPI_RANKID:-0}}}

PERF_OUTPUT=$( printf "perf_%04d.dat" ${RANK} )

perf record -o ${PERF_OUTPUT} -e cpu-cycles "$@"
```

Remplacer: mpirun … <mon binaire> …

Par: mpirun … wrapper.sh <mon binaire> …

Génère un fichier de données par rang MPI.

Auteur: L. Gatineau

Profiling, recherche de Hot Spot

Exemple HPL / perf:

- Vue synthétique programme / librairies / noyau

```
$ perf report --input perf_0000.dat -n --stdio --sort comm,dso
# Events: 155K cycles
#
# Overhead  Samples    Command         Shared Object
# ........ ..........  .......  ....................
#
    89.74%     139255     xhpl  libmkl_avx.so
     5.35%       8309     xhpl  xhpl
     4.08%       6339     xhpl  libmpi_dbg.so.4.1
     0.74%       1146     xhpl  [kernel.kallsyms]
     0.06%         93     xhpl  libc-2.12.so
     0.03%         40     xhpl  libmkl_core.so
     0.00%          6     xhpl  libmkl_intel_lp64.so
     0.00%          3     xhpl  ld-2.12.so
     0.00%          1     xhpl  libmkl_sequential.so
     0.00%          1     xhpl  libpthread-2.12.so
```


Auteur: L. Gatineau

Profiling, recherche de Hot Spot

Exemple HPL / perf:

- Vue par routines programme / librairies / noyau

```
$ perf report --input perf_0000.dat -n -stdio
# Events: 155K cycles
#
# Overhead  Samples    Command         Shared Object                                  Symbol
# ........ ..........  .......  ....................  ......................................
#
    86.67%     134480     xhpl  libmkl_avx.so         [.] mkl_blas_avx_dgemm_kernel_0
     2.31%       3589     xhpl  libmpi_dbg.so.4.1     [.] 0x10154d
     1.80%       2801     xhpl  libmkl_avx.so         [.] mkl_blas_avx_dtrsm_ker_ruu_a4_b8
     1.51%       2351     xhpl  libmpi_dbg.so.4.1     [.] 0x2d91d7
     1.39%       2153     xhpl  xhpl                  [.] HPL_rand
     1.00%       1556     xhpl  xhpl                  [.] HPL_lmul
     0.74%       1146     xhpl  [kernel.kallsyms]     [k] 0xffffffff8103772a
     0.74%       1152     xhpl  xhpl                  [.] HPL_dlaswp06T
     0.65%       1003     xhpl  xhpl                  [.] HPL_setran
     0.52%        804     xhpl  libmkl_avx.so         [.] mkl_blas_avx_dgemm_copybt
     0.47%        732     xhpl  xhpl                  [.] HPL_ladd
     0.33%        510     xhpl  xhpl                  [.] HPL_dlaswp01T
     0.32%        498     xhpl  xhpl                  [.] HPL_dlaswp10N
     0.28%        434     xhpl  libmkl_avx.so         [.] mkl_blas_avx_dgemm_copyan
     0.20%        311     xhpl  libmpi_dbg.so.4.1     [.] MPIDI_CH3I_Progress
     0.20%        310     xhpl  xhpl                  [.] HPL_dlatcpy
```


Auteur: L. Gatineau

Profiling, recherche de Hot Spot

Exemple HPL / perf:

- Vue par routines filtrées sur un objet

```
$ perf report --input perf_0000.dat -n -stdio --dsos=xhpl
# dso: xhpl
# Events: 8K cycles
#
# Overhead  Samples    Command          Symbol
# ........ ..........  .......  ...............
#
    25.97%       2153     xhpl  [.] HPL_rand
    18.78%       1556     xhpl  [.] HPL_lmul
    13.79%       1152     xhpl  [.] HPL_dlaswp06T
    12.09%       1003     xhpl  [.] HPL_setran
     8.83%        732     xhpl  [.] HPL_ladd
     6.12%        510     xhpl  [.] HPL_dlaswp01T
     5.95%        498     xhpl  [.] HPL_dlaswp10N
     3.73%        310     xhpl  [.] HPL_dlatcpy
     2.98%        250     xhpl  [.] HPL_pdlange
     1.24%        103     xhpl  [.] HPL_pdmatgen
     0.13%         11     xhpl  [.] HPL_pipid
     0.11%          9     xhpl  [.] HPL_plindx1
     0.08%          7     xhpl  [.] HPL_dlocswpT
```


Auteur: L. Gatineau

Projection des performances

- Nécessite en premier lieu de connaître:
  - Les caractéristiques de la machine actuelle.
  - Les caractéristiques de la machine cible.

- Si on a une bonne connaissance de l'application:
  - Découpage de l'application en portions élémentaires (p_i):
    - Phases de calcul / d'initialisation.
    - Phases d'entrées / sorties.
    - Phases de communications.
    - Phases parallèles / séquentielles.

  - Projection des performances pour chacune des phases, ou des phases coûtant le plus par rapport au temps de restitution.

$$T = \sum_i a_i p_i$$

Auteur: L. Gatineau

Projection des performances

- Connaissance limitée de l'application:
  - Projections à base d'expériences et de profils…
  - Identifications des phases d'initialisation / calcul / écriture des résultats.
  - Interpolation / extrapolation des temps de calculs:
    - Utilisation d'un tableur.
    - Courbe de tendance (linéaire, polynomiale,…).

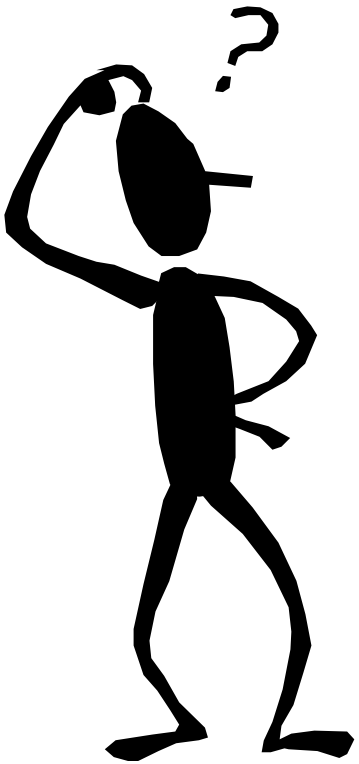
  - Voir l'influence des différents composants d'architecture sur l'application:
    - Processeur: faire varier la fréquence des CPUs.
    - Mémoire: faire varier le nombre de cœurs par processeur.
    - Réseau:
      - Tester différentes configurations.
      - Si trace MPI, utiliser un simulateur.
    - Disque:
      - Tester différentes configurations.
      - Utiliser /dev/shm pour voir le coût des IOs.

Auteur: L. Gatineau

Projection des performances

- Augmentation du nombre de nœuds de calcul:
  - Etude de scalabilité.
  - Si le nombre de nœuds final est dans le même ordre de grandeur que pendant les tests, alors l'extrapolation est généralement fiable.
- Augmentation du nombre de cœurs par processeur:
  - Est-ce que la bande passante mémoire par processeur augmente ?
  - Est-ce que le débit réseau par nœud/processeur augmente ?
  - Est-ce que la quantité de mémoire par cœur est la même ?

QUESTIONS ?



Auteur: L. Gatineau