Benchmarks

Ensemble de codes permettant de tester la fonctionnalité et les performances d'une solution HPC dans son ensemble.

(Merci à Ludovic Saugé)

Françoise BERTHOUD, Francoise.Berthoud@grenoble.cnrs.fr

Les benchmarks : sommaire

Objectifs : Pourquoi et comment mesurer/comparer la performance des serveurs de calcul ?

Plan :

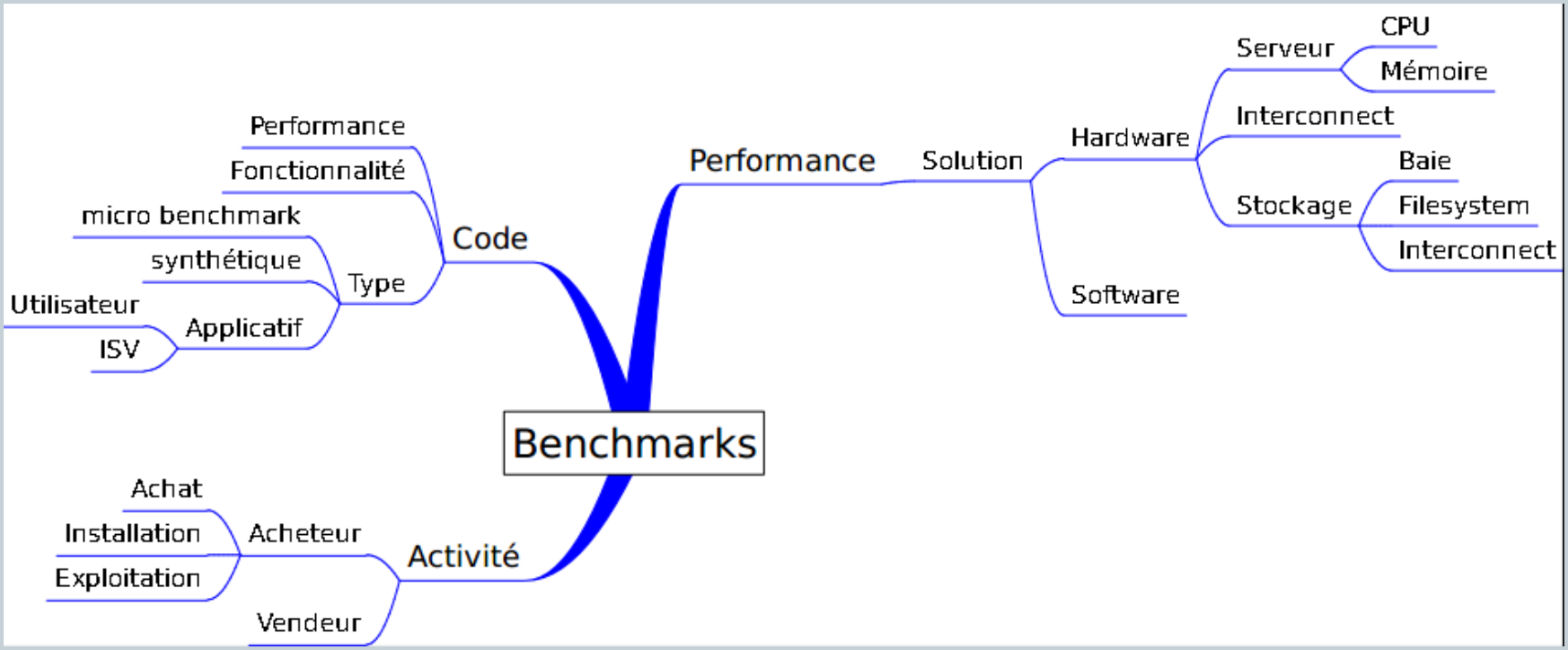
1 – le cadre : qui, quand, pourquoi ?

2 – Comment ?

3 – Mémoire / interconnexion / IO

4 – Quelques considérations au moment de l'achat

Éléments de contexte



Qui, Quand et pourquoi ?

L'acheteur

- **Achat** : Dans la phase de préparation de l'AO
  - Connaître les caractéristiques de la solution existante
  - Préparer la partie benchmark de l'AO
    - → Logistique
    - → Analyse des résultats soumis par les vendeurs
- **Installation** :
  - Phase de recette (« acceptance tests »). Vérification des engagements fonctionnels et de performances.
- **Exploitation** :
  - Suivre tout au long de la durée de vie de sa machine les performances et déceler d'éventuels problèmes ou possibles régressions.

Questionnement ..

Questions aux stagiaires :

- Qu'est-il intéressant de mesurer pour être certain de faire le bon choix ?
- Y'a-t-il un seul indice de performance ?
  - Le temps de restitution d'une application.
  - Le nombre de jobs simultanés
  - La consommation électrique pour un job donné ?
- Existe-t-il le benchmark universel ?
  - Celui qui donne une valeur unique, facile à comparer …
  - Celui qui tourne vite …

  HPL (Top500) ? Spec2006 ? HPCC ?

Quels benchmarks ?

- Les micro-benchmarks et les benchmarks synthétiques fournissent une idée du comportement d'une solution élément à élément. C'est une référence, un point de comparaison entre les technologies

- Les applications sont là pour donner une idée du comportement **GLOBAL** d'une solution
  - La fin des fins d'une machine est d'effectuer des calculs pour les utilisateurs
  - Les applications doivent être réalistes d'une situation de production.

Mais avant tout ..

Il est nécessaire :

- De bien comprendre et définir ce que l'on veut mesurer (pourquoi veut-on le mesurer ?)
- De mettre au point des **outils <u>adaptés</u>**, une expérience (avec ses conditions) pour obtenir la mesure → le **benchmark** ...

  Ex. « capacité d'un nœud de calcul d'effectuer des opérations mémoires aléatoires »

  → On veut donc déterminer la latence mémoire

  → On peut utiliser un bench style 'latmem' ou HPCC s-RandomAccess ...

  - Ces benchs donnent des mesures différentes car ils ne mesurent pas exactement la même chose ...
  - Lequel est-il le plus pertinent pour mon application ?

Et ..

- Comment tel ou tel choix affecte les performances de la solution
  - Ces effets sont-ils mesurables ? Comment les mesurer ?
  - Ex. interconnect
    - Latence, bande passante
    - Performance MPI
    - Code de l'utilisateur …
- Ne pas oublier l'objectif final : la machine est destinée à faire tourner des codes utilisateurs en production et non des benchmarks …

Performances mémoire : facteurs déterminants

- Localisation : cache ? Mémoire centrale ? Mémoire distribuée ? disques ?
- Si les données sont en RAM
- La manière dont elles sont accédées : au travers d'un chipset, directement par le processeur ? Via le processeur voisin ? BP

llcbench :
http://icl.cs.utk.edu/projects/llcbench/
Mesure de la bande passante pour différente taille de tableau (256 bytes à plusieurs dizaines de MBytes) et différents « pattern »
Découverte de la hiérarchie mémoire
Portage et exécution : Immédiat

Lat_mem_rd (LMBench) :
http://www.bitmover.com/lmbench/
Latence d'accès (en nanosecondes) aux données au travers de la hiérarchie de cache
Portage et exécution : simple

Stream *:* http://www.cs.virginia.edu/stream/
« The STREAM benchmark is a simple synthetic benchmark program that measures sustainable memory bandwidth (in MB/s) and the corresponding computation rate for simple vector kernels. »
Benchmark accidentel ! Développé pour comprendre les différences de performance entre deux architectures pour un code météo. Indispensable dans sa trousse-à-outil
Accès mémoires séquentiels
Portage et exécution : Aucune difficulté

Performances réseau

- Le type de réseau (numalink, ethernet, infiniband, etc.), soit la *????* et la *?????* (hard)
- L'architecture du réseau (hard)
- La taille des messages (code)

Attention : sur les SMP / Numa ou gros nœud un test sur l'ensemble des nœuds donne un résultat « moyenné » entre « entre-nœud » (InterConnect) et « inter-nœud » (« shared memory »).

Performances réseau

Performances point à point :

- Indicateurs :
  - Latence
  - Petits messages
- bande passante
  - Gros messages
- taux de messages
  - Permet de comparer les technos entre elles.

- La **latence** est une quantité primordiale pour les petits messages: elle permet aussi de définir le taux de messages (~2 bytes)

- Les tests sur les **opérations collectives** sont intéressantes pour l'applicatif (sensible à l'engorgement) : MPI_**Alltoall**(v), codes à base de FFT, par ex. les codes de chimie (cpmd)

- Le **taux de messages** est une quantité intéressante : il fournit une indication sur la capacité de traitement de la carte.

Performances d'IO

- Toute la chaine d'IO : baie (configuration matérielle, logicielle), type et architecture d'interconnexion, serveur (s), client
- Présence/taille des caches !
- Type de file systèmes et pile logicielle adaptée.
- L'utilisation qu'on en fait (applis !) (Type de pattern : séquentiel ou random , ratio écriture/lecture, nombre, taille)

On ne benchmark pas seulement une baie mais un système global
➔ Système équilibré et performant, qui satisfasse vos contraintes (budget, énergie), pour vos applicatifs !

Performances d'IO : benchmarks

(attention aux propriétés du FS !)

Benchmarks « maison » (attention aux conditions d'exécution) ➔ donneront une idée sur les perf de la solution et le comportement des applications

Bonnie ++ :
http://www.coker.com.au/bonnie++/
Ecriture séquentielle, Lecture séquentielle
Pattern aléatoire : lecture et re-écriture dans 10% des cas.
Portage facile, exécution facile ..

IOZONE : http://www.iozone.org/
« Couteau suisse » du bench d'I/Os
Multiple pattern d'écriture : *write, read, rewrite,reread, random* ...
Multiple taille d'I/Os, multiple volume.
Le bench dans sa globalité peut permettre de déceler les effets de caches.
Portage simple, exécution : nombreuses options

Interleaved or Random (IOR) benchmarks :
http://sourceforge.net/projects/ior-sio/.
Proche des applications : parallèle, adaptés au HPC
Permet de tester les APIs (parallèles) couramment utilisés par les utilisateurs : MPI-IO, parallelNetCDF, pHDF5, posix
Par défaut MPI-IO
Portage : Modéré, requiert une implémentation MPI
Exécution : Simple

Pour ce benchmark, MPI-IO n'est pas adapté à la mesure d'un débit maximale d'une baie. Par contre il parfaitement adapté pour avoir des idées de performances d'applicatifs.

Effective I/O Bandwidth (beff_io) Benchmark :
https://fs.hlrs.de/projects/par/mpi//b_eff_io/
Benchmark orienté MPI-IO, différentes opérations, différents pattern
Synthétique : nombreux tests, nombreuses données : un score !
Portage : Facile, nécessite une implémentation MPI-2
Exécution : Modéré , analyse compliquée ...

En pratique : Comment spécifier les demandes de benchmarks dans les proc d'achat ?

- Choisissez les applicatifs les plus représentatifs ou ceux qui seront exécutés sur le serveur
- Choisissez un jeu de paramètres représentatif (attention aux effets de cache)
- Spécifier ce que vous attendez comme résultat, et l'unité utilisée (conditions d'exécution, temps de restitution en secondes, kwh, ..)
- Spécifier complètement les conditions d'exécution du bench, soit :

En pratique : Comment spécifier les demandes de benchmarks ?

- 1 - Conditions de charge ou de non charge du socket au moment de l'exécution du bench (exemples)
- 2 - Spécifications techniques hardware (par exemple serveur à l'identique à la solution proposée)
- 3 - Si job parallèle, en plus des conditions ci-dessus, spécifier le nombre de cœur en étant très précis sur la répartition sur les sockets/serveurs
- 4 - Fournir un tableau à compléter avec les résultats

Par exemple …

- A construire ensemble ….