Module : Evaluation de Performances Master 2 Informatique Haute Performance (MIHP)

Soraya Zertal

Li-PaRAD-Université de Versailles

Intervenants

- Soraya Zertal, Laboratoire Li-PaRAD, Université de Versailles

  soraya.zertal@uvsq.fr
- Luigi Brochard, lenovo

  luigi.brochard@lenovo.com
- Jean-Thomas Acquaviva, DDN

  jean-thomas.Acquaviva@gmail.com

Présentation générale du module

- Introduction à l'Evaluation Des Performances : Motivation, problématique et concepts de base, évaluation de performances et le HPC (S. Zertal)
- Méthodes et techniques d'évaluation (S. Zertal)
  - Workloads
  - Modélisation mathématique
  - Simulation
  - Mesure
- Performance/Consommation énergétique (L. Brochard)
- Outils d'évaluation (J-T. Acquaviva)

Plan :

Concepts de base :

1. Motivation et problématique
2. Approche à suivre
3. Checklist pour éviter les erreurs
4. Métriques de base à évaluer
5. Caractéristiques d'un bon métrique
6. Evaluer les performances : méthodes à adopter

Plan :

Evaluation des performances et le HPC :

1. Objectifs
2. Parallélisme et performance
3. Métriques classiques dans le HPC
4. Modèles et lacunes
5. Evaluation

Motivation et problématique

But de tout système : fournir le meilleur rapport coût/performance

⇒ Nécessité d'évaluer cette performance à toutes les étapes du cycle de vie d'un système :

1. Conception : prédiction
2. Réalisation : choix matériel ou logiciel
3. Commercialisation : mise en valeur de métriques
4. Utilisation : tuning selon le contexte
5. Maintenance et Mise à jour : contrôle (monitoring)

Motivations et problématique

Objectifs de l'évaluation de performances :

- Mieux connaitre la faisabilité d'un système/application et l'intérêt de son développement/codage.
- Détecter et mieux comprendre les limitations d'un système.
- Bien cibler les améliorations possibles : ajout de ressources (exp. processeurs), réorganisation des traitements, réorganisation des charges,...etc

Approche à suivre

Les systèmes à analyser sont multiples, de même pour les métriques, les techniques d'évaluation et les types de charge (workloads) associés.
Néanmoins, l'approche générale d'évaluation suit les étapes suivantes:

- Définir le système à analyser et préciser le but à atteindre.
- Etablir la liste des services à fournir par le système analysé.
- Sélectionner les métriques relatifs à la vitesse, la précision et la disponibilité du système analysé.

Approche à suivre

- Etablir la liste des paramètres affectant les performances du système analysé.
  A distinguer les paramètres liés au système de ceux liés aux types de workloads.
- Sélectionner les facteurs (parmi les paramètres) variables pendant l'évaluation ainsi que les valeurs qu'ils peuvent prendre.
- Sélectionner la méthode d'évaluation appropriée parmi la modélisation mathématique, la simulation ou la mesure.

Approche à suivre

- Sélectionner les charges auxquelles sera soumis le système à évaluer : **synthétiques** ou **réelles**.
- Concevoir les jeux de tests pouvant fournir le maximum d'informations pour l'étude du comportement du système.
- Analyser et interpréter les données collectées à l'issue des tests en utilisant des **techniques statistiques**.
- Présenter les résultats et les communiquer sous une forme claire et compréhensible, souvent sous forme graphique, en utilisant une échelle appropriée.

Checklist pour éviter les erreurs (R. Jain)-Annexe

- Is the system correctly defined and the goals clearly stated?
- Are the goals stated in an unbiased manner?
- Have all the steps of the analysis followed systematically?
- Is the problem clearly understood before analyzing it?
- Are the performance metrics relevant for this problem?
- Is the workload correct for this problem?
- Is the evaluation technique appropriate?
- Is the list of parameters that affect performance complete?

Checklist pour éviter les erreurs (R. Jain)

- Have all parameters that affect performance been chosen as factors to be varied?
- Is the experimental design efficient in terms of time and results?
- Is the level of detail proper?
- Is the measured data presented with analysis and interpretation?
- Is the analysis statistically correct?
- Has the sensitivity analysis been done?
- Would errors in the input cause an insignificant change in the results?
- Have the outliers in the input/output been treated properly?

Checklist pour éviter les erreurs (R. Jain)

- Have the future changes in the system and workload been modeled?
- Has the variance of input been taken into account?
- Has the variance of the results been analyzed?
- Is the analysis easy to explain?
- Is the presentation style suitable for its audience?
- Have the results been presented graphically as much as possible?
- Are the assumptions and limitations of the analysis clearly documented?

Introduction aux métriques

Les métriques communs pour l'évaluation des performances :

- Temps de réponse
- Débit
- Accélération
- Efficacité
- Taux d'utilisation
- Fiabilité
- Disponibilité

Temps de réponse

Définition :
Le temps séparant la requête de l'utilisateur de la réponse du système. Le temps de réponse augmente avec la charge du système sous l'effet de l'augmentation du temps d'attente.

Le **facteur d'extension** (stretch factor) est le rapport du temps de réponse pour une certaine charge à celui pour une charge minimale.

Temps de réponse - stretch factor

Ce facteur quantifie l'impact de l'augmentation de la charge sur l'allongement du temps de réponse.

Exemple :
Dans un système à temps partagé, le facteur d'extension (strech factor) est le rapport du temps de réponse avec la multiprogrammation à celui sans multiprogrammation.

Débit

Définition :

Taux de requêtes exécutées en une unité de temps.

Système	Unité de débit
Interactif	requêtes/s
CPU	MIPS ou MFLOPS
Batch	job/s
Réseau	packet-bits/s (pps,bps)
Transactionnel	transactions/s (TPS)

Débit

Remarque :
Le **débit** augmente avec la charge jusqu'à une certaine valeur maximale, une sorte de borne supérieure (exple: bande passante en réseau).

Débit

Débit/capacité nominal est le débit maximal pouvant être délivré par le système soumis à une charge idéale.

Exemple:

Aprés le start-up, une instruction est exécutée par le pipeline à chaque cycle s'il n y a aucune dépendance pénalisante ou branchement.

Débit/capacité d'usage est le débit maximal pouvant être obtenu tout en respectant la limite imposée au temps de réponse pour le maintenir en dèça d'une certaine valeur acceptable par l'utilisateur.

Débit - MIPS et MFLOPS

MIPS: Millions d'instructions par seconde

$$MIPS = \frac{nb\ \ instrs}{10^6 \times tps\ \ execution}$$

MFLOPS: Millions d'instructions flottantes par seconde
Donc, c'est le MIPS restreint aux instructions flottantes.

Remarques : Pour les deux unités, on considère que les instructions sont en mémoire. De même, toutes les deux (surtout le MIPS) ne constituent pas de bons indicateurs de performances.

Accélération

Définition :

Le rapport entre le temps séquentiel et le temps parallèle (amélioré de manière générale)

$$Acc = Speedup = \frac{Tps\ \ sequentiel}{Tps\ \ parallele}$$

Le calcul de l'accélération par La **loi d'Amdahl** simple:

$$Acc = Speedup = \frac{1}{(1-F_a)+\frac{F_a}{P}}$$

F_a: fraction parallélisable (améliorée) du code

P : nombre de processeurs utilisés

Accélération/coût

L'Accélération n'est pas linéaire en fonction du nombre de processeurs. Il y a des coûts supplémentaires liés à la parallélisation (amélioration) tels que : les échanges de messages, la synchronisation ...etc

Le calcul de l'accélération par la **loi d'amdahl enrichie** :

$$Acc = \frac{1}{(1-f_a)+(\frac{f_a}{P})+Ct(f_a,P)}$$

$Ct(f_a, P)$: Coût lié à la parallélisation de la fraction f_a du code sur P processeurs.

Efficacité

Définition :

L'Efficacié d'une accélération obtenue par l'utilisation de P processeurs se calcule par:

$$E = \frac{Acc}{P}$$

ou bien :

$$E = \frac{Tps\ \ sequentiel}{Tps\ \ parallele \times P}$$

Plus l'efficacité est proche de 1, plus l'accélération est optimale.

A contrario, $(1 - E)$ désigne la part de l'inefficacité dûe à la communication/synchronisation liée à la parallélisation.

Taux d'utilisation

Définition :

Fraction du temps pendant laquelle la ressource est utilisée.

Calcul :

$$\frac{temps\ \ d'occupation\ \ de\ \ la\ \ ressource}{le\ \ temps\ \ total\ \ ecoule}$$

Buts :

Maximiser le taux d'utilisation d'une ressource.

Harmoniser au mieux les différentes ressources $\Rightarrow$ l'équilibrage de charge (Load Balancing).

Efficacité et taux d'utilisation

L'efficacité reflète le taux d'utilisation des ressources fonctionnant en parallèle.

Plus elle tend vers 1, plus le taux d'utilisation est meilleur.

Fiabilité (Reliability)

Définition:
C'est l'apptitude d'un système à effectuer ses fonctions, de fournir des résultats non erronés et de maintenir ce fonctionnement en toutes circonstances pendant un temps T.

Data Reliability : le système de stockage -moyennant une redondance intégrale ou partielle- garantit l'accès à des données non eronnées.
Network Reliability : le réseau garantit que les messages sont délivrés aux destinataires

Fiabilité (Reliability)

La fiabilité se **mesure** généralement par la probabilité d'arrivée des erreurs.

Néanmoins, il est d'usage de considérer le temps moyen séparant l'arrivée des erreurs (**Mean Time Between Errors**).

Disponibilité (Availability)

Définition :
C'est la période du temps pendant laquelle le système peut servir les requêtes utilisateur.
C'est la période **uptime** par opposition à celle **downtime** où le système ne répond plus.

La disponibilité se mesure généralement par le temps moyen avant la panne (Mean Time To Failure -**MTTF**-) ou le temps moyen entre les pannes (Mean Time Between Failures -**MTBF**-).

Caractéristique : linéarité

Le métrique doit indiquer une performance **linéairement proportionnelle** à la performance actuelle.

Les utilisateurs sont assez sensibles à cette caractéristique mais pas tous les métriques peuvent l'être. Certains métriques sont logarithmiques.

Caractéristique : fiabilité

Un métrique est considéré fiable, si les valeurs de ce métrique pour les systèmes A et B, indiquent que le système A est plus performant que le système B et
le système A est effectivement **toujours** plus performant que le système B.

Cela parrait évident mais il ne l'est pas toujours.
Exemple : MIPS n'est pas un métrique fiable.

Caractéristique : fiabilité

Exemple de la non fiabilité du MIPS

Le benchmark whetstone est exécuté sur une machine pouvant utiliser un coprocesseur flottant. Une itération de ce benchmark dure 1.08s et génère 1.6 MIPS avec le coprocesseur mais dure 13.6s et génère 2.7 MIPS sans coprocesseur.

L'exécution en utilisant le coprocesseur est alors plus performante (12.6 fois plus rapide) pour un MIPS moindre. Ce dernier ne reflete donc pas cette performance.

Caractéristique : déterminisme

La **même valeur** du métrique est obtenue **à chaque fois** que le test est effectué.

Caractéristique : facilité de mesure

Un métrique difficile à mesurer est par conséquent peu utilisé.

De plus, une mesure difficile à effectuer a des chances d'être erronée.

Caractéristique : indépendance

Un métrique doit être indépendant de l'effet de tout paramètre externe.

Il doit être conçu pour ne favoriser aucun système en particulier s'il y a commercialisation ou autre but profitable.

Méthodes

Trois méthodes pour l'évaluation des performances: la méthode **analytique**, la **simulation** et la **mesure**.

La sélection de la méthode appropriée repose en premier lieu sur l'étape dans laquelle se trouve le système à analyser au moment de l'analyse.

Aussi, le coût temporel et financier de l'évaluation, les outils disponibles ainsi que le degré de précision des résultats.

Méthodes

Opter pour la **mesure** si le **système existe** et est disponible (ou un système s'y approchant puis extrapoler).

Autrement, opter pour la **simulation** ou la méthode analytique.

La **simulation** est plus **consommatrice en temps** mais nécessite moins de **simplifications** que la **méthode analytique**.

En général, on utilise 2 méthodes et on compare leurs résultats pour validation.

Plan :

D'après les transparents J-F. Lemerre (BULL)

- Objectifs
- Parallélisme et performance
- Métriques classiques pour le HPC
- Modèles et lacunes
- Evaluer les performances

Performance en HPC : Objectifs

- Proche du matériel :
  Prédire les performances, mesurer, caractériser, comparer des solutions HW (par modèle), proposer des évolutions, publier,...
- Très orientés performances élémentaires :
  - Activité annexe: stress du matériel grâce aux conditions aux limites
  - Nécessite parfois le recours à l'assembleur pour maîtriser ce qu'on fait
  - Etudes en profondeur, nombre de benchmarks clients

Parallélisme et performance

- Performance HPC
  Loi d'Amdahl, parallélisation(scaleup,speedup), workload, taux de miss, les métriques classiques, définition xxx-bound, latences et débit mémoire, ordres de grandeur
- Evaluer les performances
  - Les erreurs à éviter
  - Bonnes pratiques

Parallélisme et Amdahl

HPC = Parallèle ou massivement parallèle

Que peut on paralléliser ?

$$Temps\ total = temps\ mono + \frac{temps\ \ parallele}{N}$$

Si S proportion sequentielle :

$$R = \frac{1}{S + \frac{(1-S)}{N}}$$

Si N tend vers l'∞

$$R = \frac{1}{S}$$

R est inversement proportionel à la proportion sequentielle quand N est trés grand

Parallélisme explicite/implicite

- Explicite :
  Cela a été programmé, lgges :MPI, OpenMP
- Implicite :
  - Plusieurs jobs en parallèle
  - Tirage Monte Carlo
  - Exploration paramétrique

Scale-UP

- SCALABILITE : capacité à supporter un plus grand nombre de tâches (identiques) en parallèle
- plus d'utilisateurs
- plus de transactions par minute
- plus de calculs (Threads) simultanés
- Typique : nombre CPU $\times$ 2
  $\Rightarrow$ Scalabilité $\times$ 1.7
- Variable de 0.8 à 2

SPEED-UP

- Capacité à exécuter une tâche en un temps plus court
- Si c'est par augmentation du nombre de CPU, le programme doit être parallélisable
- Calculs matriciels avec l'openMP, du MPI, ou des outils comme le fortran parallèle
- Typique : Nombre de CPU $\times$ 2
  $\Rightarrow$ Speed-up $\times$ 1.7
- Variable aussi de 0.8 à 2

Exemple Speed-up

- **Exploration paramétrique :**
  Lancer n*application avec des paramètres d'entrée différents.
  On peut le faire sur n machines en même temps au lieu de le faire en série.
- **Tirage Monte Carlo :**
  1 Milliard d'expériences : durée T sur un coeur
  Pas de dépendances (test sans mémoire)
  Ces tests peuvent être lancés par script shell par exple
  Amélioration de la précision.

Scalabilité d'un code HPC

- **Scalabilité d'un code parallèle :**
  - Etude de l'évolution du temps de calcul d'un problème quand le nombre de processeurs augmente
  - Nombre nœuds et nombre de processeurs/nœud
  - Evolution Ratio temps de comm/temps de calcul
- **Augmenter taille du pb avec nbre de proc?**
  2 axes: on compare les temps de traitement d'un problème de taille N par P processeurs :
  - Temps de traitement d'un problème de taille N sur 2P proc
  - Temps de traitement d'un problème de taille 2N sur 2P proc

Codes et scalabilité

- Certains codes sont très scalables (exple Linpack)
- D'autres le sont beaucoup moins : soit à cause de la nature physique, soit à cause du découpage pas forcément adéquat au problème.
- Les échanges de données et la synchronisation prennent le pas sur les calculs

Workload-cache MISS/HIT

- l'espace occupé par une application (instr+data)="workload"
- Si application petite → tout est dans le cache
- Si application importante, une partie seulement dans le cache
- Miss ratio: nombre de fois sur 100 où la donnée n'est pas dans le cache
- Localité forte ⇒ instuction miss ratio faible
- Les instructions actives sont souvent dans le cache → taux de miss instruction faible
- Les données restent rarement longtemps dans les caches: le tuning permettra de conserver une localité le plus possible.

Latence/Débit/Taux

- **Latence ou temps de réponse :** Durée entre la requête et la réponse de l'objet : latence mémoire, latence du réseau (durée du ping-pong/2), réponse à une requête base de données en transactionnel
- **Débit :** Quantité par unité de temps, débit mémoire, disque ou réseau en GB ou Gb/sec, MIPS, MFLOPS
- **Taux d'occupation ou d'utilisation d'une ressource :** Mémoire, CPU...
  Si saturé $\rightarrow$ Bottleneck ou chemin critique.
  Système en entier (charge variable entre idle et 100%)

Autres métriques...

- **Efficacité :** rapport mesuré (théorique ou crête)
- **Fiabilité :** MTBF
- **Disponibilité :** Downtime/uptime, MTTF ...
- **Rapport performance/prix**
- **Performance/consommation energie** (Flops/watt)

xxx-bound

- **Code "CPU-bound" :** code dont la performance est limitée par la fréquence du CPU
- **Code "memory-bound" :** code limité par le débit mémoire: taux de miss élevé
- **Code "Latency-bound" :** code "memory-bound" pour lequel la latence des accès mémoires limite la performance
- **Code "IO-bound" :** code limité par le débit d'IO

IO-bound

- TRES fréquents mais ne le devrait pas en HPC saufs pour les nœuds d'E/S
- Sur les nœuds d'E/S (avec disques), IOstat donne des infos d'occupation disque.
  Il faut paralléliser, stripper, SSD....ou changer d'architecture

100% CPU

- Memory bound ou CPU bound ?
- Processeurs modernes permettent de faire varier la fréquence (CPU governor)
  - Si performance suit la frequence $\rightarrow$ CPU-bound
  - Si performance varie peu ou pas $\rightarrow$ memory-bound
- Extrapoler : avec un processseur plus rapide, je gagnerai ...
  Attention aux extrapolations : abusives, linéaire ou pas, saturation/effet de seuil...

Considérations memory/CPU-bound

- Un calcul complexe même essentiellement CPU-bound sur un processeur, peut nécessiter un gros travail d'optimisation pour arriver à ce stade.
- La taille des jeux de données peut faire passer un code/calcul d'une catégorie à l'autre (effet cache)
- Un calcul peut être CPU-bound pour une famille de processeurs et memory-bound pour une autre.

Evaluation des métriques classiques

Débit mémoire

- Stream : développé initialement par John D. McCalpin

```
Copy : c[i] = a[i];
Scale : c[i] = scalar*a[i];
Add : c[i] = a[i] + b[i];
Triad : c[i] = a[i] + scalar*b[i];
Read pur : Tot = tot+a[i]
```

- Plusieurs tâches en parallèle pour saturer le débit mémoire
- Write = Read Modify write, donc peut compter pour 2 (depend de l'archi, du jeu d'instr.)

Evaluation des métriques classiques

- FLOPS: oui et encore ....
  Efficacité variable, proportion d'instructions flottantes dans l'appli
- MIPS = Meaningless Instructions per second néanmoins...
  Utilisé pour des architectures tres différentes : RISC, CISC, ...

Modelisation de puissance CPU: exercice

100 instructions à exécuter
Combien instructions en parallèle ?
Sachant qu'il y a des miss L1D et L1 I, mais qu'on les néglige
et donc IPC $\rightarrow$ 1.25
Taux de miss L1 = 5% (dépend du workload)
Taux de miss L2 = 1%
Taux de miss L3 = 0.4%
Négligez le temps d'accès au cache L1 (anticipé)
Temps d'accès au cache L2/L3/la mémoire : 10/40/200 cycles

Modélisation de puissance CPU

Pour exécuter 100 instrs, il faut : IPC=1.25 $\rightarrow$ CPI=0.8
Temps d'accès au cache L2 = 10 cycles
Temps d'accès au cache L3 = 40 cycles
Temps d'accès à la mémoire : 200 cycles
100 $\times$ 0.8 cy (parallélisme) = 80 cy
5 $\times$ 10 cy (Miss L1) = 50 cy
1 $\times$ 40 cy (Miss L2) = 40 cy
0.4 $\times$ 200 cy (Miss L3) = 80 cy
Total = 250 cycles $\Rightarrow$ 1 instr dure 2.5 cycles
CPI=2.5 $\rightarrow$ IPC=0.4

Conclusion : Application réelles : 1/3 CPU, 1/3 Caches, 1/3 mémoire

Obtenir les valeurs

- **Benchmarks ciblés donnent mesures élementaires :**
  Variantes de mesures mémoire, latence des caches et latence mémoire. Instrumentation HW pour le reste
- **Compteurs de performance :**
  Jeu de registres HW enregistrant les événements, Perfmon, Perfctr, PAPI (University Tennessee Knoxville),

Compteurs de performance

- Une quantité astronomique, mais peu en même temps (rejouer la même mesure plusieurs fois)
- Nombre de cycles (temps référence)
- Nombre cycles stalled : car on attend la mémoire ou autre ressource. Optimisation performance = chasse aux stalls
- Nombre instrs, nombre FP instrs.
- Nombre Load/store
- Nombre de branchement pris ou invalides
- Nombre de miss, taux de miss, nombre de miss/nombre instr
- TLB miss ...

Benchmarks et comparaison

- Une mesure isolee n'a pas de sens
  Exemple:
  la terre pèse $6 \times 10^{24} kg$... et alors ?
  Venus = 0.8 fois, Mars = 0.1 fois, Jupiter=315 fois...
- Une référence doit être définie car besoin d'explications, de conséquences ...

Erreurs à éviter -1

SE FIXER DES OBJECTIFS CLAIRS

- **Pas de but** : problème pas assez précisé, modéliser et voir après ...
- **But biaisé** : Démontrer qu'on a raison, faire plaisir
- **Approche pas systématique** : On n'analyse pas tout
- **Analyse sans comprendre le problème** :
  Réponse = le modèle au lieu de poser le problème.
  Un problème bien posé est à moitié résolu.

Erreurs à éviter -2

BIEN CHOISIR METRIQUES ET PARAMETRES

- **Pas le bon métrique**: les MIPS par exple
- **Workload non significatif** : utiliser un bench calcul scientifique pour le transactionnel ...
- **Mauvaise technique d'évaluation** : utiliser une technique connue au lieu de l'utile
- **Paramètres :** en oublier, leur donner des biais trop/peu important
- **Mauvais niveau de détail** : Simuler un processeur moderne en considérant comme un réseau de portes

Erreurs à éviter -3

ANALYSE

- **Collecte de données** (peut être difficile) mais sans interprétation... ne vaut rien.
- **Analyse erronée**
- **Manque d'analyse de sensibilité :** par rapport à la variation d'un paramètre
- **Traitement des valeurs bizarres**
- **Etude de variabilité :** la moyenne n'est pas suffisante

Bonnes pratiques -1

- **Définir les attentes** : exemple: taux d'utilisation d'une machine
- **Choisir les métriques** : débit, efficacité, taux de charge
- **Lister les paramètres** : affectant la performance, de nouveaux peuvent apparaitre en cours d'etude.
- **Choisir les facteurs à etudier et leurs valeurs** parmi les paramètres précédents choisir lesqueles faire varier
- **choisir la technique d'évaluation** faire des mesures ou construire un modèle.

Bonnes pratiques -2

- **Choisir son workload/Bench**
- **Faire un plan d'expériences** : possibilité de regrouper des tests
- **Analyser et interpréter les résultats** : Comprendre, est-ce normal?, est-ce credible?
- **Présenter les résultats**