




Supélec   

3A-IIC – Calcul parallèle et distribué,  
et Grilles de calculs

## Mesure et analyse de performances

Stéphane Vialle

Stephane.Vialle@supelec.fr  
http://www.metz.supelec.fr/~vialle

Supélec

## Mesure et analyse de performances

- 1 – Mesure des temps d'exécution
- 2 – Expression des performances
- 3 – Sources de perte de performances
- 4 – Loi de Amdahl
- 5 – Loi de Gustafson
- 6 – Liens Amdahl-Gustafson

Supélec

Mesure et analyse de performances

## 1 – Mesure des temps d'exécution

- Différents temps d'exécution
- Les outils de mesure
- Reproductibilité des mesures
- Stratégie de mesure

Supélec

Performances

## Mesure : différents temps mesurables

- En séquentiel :  
 $T_{cpu}, T_{IO}, T_{appels-systèmes}, T_{calculs}, T_{wall-clock}, \dots$
- En parallèle :  
 $T_{cpu}, T_{IO}, T_{appels-systèmes}, T_{calculs}, T_{wall-clock}, \dots$   
+  
 $\frac{T}{i \in [0, P-1]}, \min(T_i), \max(T_i), \max(T_i^{fin}) - \min(T_i^{deb}), \dots$

→ Toujours préciser ce que l'on mesure !

Supélec

Performances

## Mesure : outils disponibles


- Mesures externes :


```
>time mpirun -np 4 a.out
>/usr/bin/time mpirun -np 4 a.out
>times mpirun -np 4 a.out
>timex mpirun -np 4 a.out
.....
```

12.002u user  
0.128s system  
12.150 total

Nom et fonctionnement variables  
selon le système utilisé !!

Fréquemment :  
total > user + system !!

Simple à utiliser   
Pas de modifications des codes sources

Peu précis: ± 0.5s 

Supélec

Performances


## Mesure : outils disponibles


- Mesures internes :

```
time()
clock()
gettimeofday()
```

→ Compte les clicks d'horloge  
→ Appel système le plus précis sous linux

- Toutes ces routines ne sont pas toujours disponibles !
- "gettimeofday" est en général une bonne solution.
- Parfois il existe des outils plus précis pour mesurer de petites durées.

Plus précis que les  
mesures externes 

Besoin de modifier le code source  
Pas toujours totalement portable 

Performances

## Mesure : outils disponibles

• **Précision des outils et des mesures :**

123456789012.1234567890123456

Capacité maximale de l'outil de mesure

Précision théorique (cf. doc)

Précision expérimentale: fluctuation des exécutions

- Ne pas tenir compte de trop de décimales!
- Faire attention à ne pas déborder la capacité de mesure!

Performances

## Mesure : reproductibilité

• **Problème fréquent :**

Test en mode exclusif (mono-user).  
Outil de mesure à 1ms de précision. → Fluctuation de 500ms d'une exécution à l'autre !!

• **Démarche conseillée :**

- Mesurer les fluctuations, ne pas les ignorer
- Ne pas donner que les valeurs moyennes
- Mesurer des temps > 10s si possible

Performances

## Mesure : stratégie

- 1 – **Faire une mesure interne & une externe (& votre montre !)**  
Comparer les résultats, approfondir si différents
- 2 – **Mesurer des temps significatifs :**  
Environ 10s sur P processeurs si possible  
⇒ Éventuellement P×10s sur un processeur  
⊘ Boucler sur le test peut fausser la mesure (effet de cache)
- 3 – **Répéter chaque mesure pour évaluer la reproductibilité :**  
Préciser l'amplitude des fluctuations  
Ne pas donner que des valeurs moyennes  
Si nécessaire tracer des « error-bar »
- 4 – **Faire des séries de mesures homogènes :**  
Toujours les premières exécutions (sans cache favorable)  
Toujours les deuxièmes exécutions  
Toujours la moyenne des 5 premières  
...

Performances

## Mesure : stratégie

- 4 – **Conserver & Indiquer les conditions de mesures :**  
**Date de l'exécution**  
**Auteur(s) du test**  
Outil(s) de mesure utilisé(s)  
Caractéristiques de la machine : RAM, Cache, Processeurs, ...  
OS utilisé (nom et version)  
Compilateur utilisé (nom et version)  
Options de compilation utilisées  
Test en multi-user/mono-user ?  
Présence d'IO dans le test ?  
Configuration du programme de test : taille des données, ...
- 5 – **Archiver les fichiers de tests :**  
Sources des programmes de test  
Mesures obtenues  
Conditions de mesures (ci-dessus)

Mesure et analyse de performances

## 2 – Expression des performances

- Accélération (*Speed up*)
- Efficacité
- Choix de la référence séquentielle
- Graphe des *speed up*

Performances

## Expression des perf. : *Speed up*

• **Définition :**

$$S(P) = \frac{T(1)}{T(P)}$$

→  $\begin{cases} S(P) < 1 : \text{on ralentit !} \\ \text{mauvaise parallélisation} \\ 1 < S(P) < P : \text{"normal"} \\ P < S(P) : \text{hyper-accélération} \\ \text{analyser \& justifier} \end{cases}$

S(P) ↑

hyper-accélération

Accélération normale

ralentissement

1

P

S(P) = P : accélération idéale

Performances

## Expression des perfs. : *Speed up*

• **Hyper-accelération :**

Ce n'est pas *magique*, et ce n'est pas *normal*  
 → On doit analyser le phénomène et l'expliquer  
 → Corriger une erreur ou exploiter une optimisation

**Exemples d'explications :**

- on ne fait plus les bonnes opérations (résultat faux)
- les données tiennent dans le cache total des P processeurs
- on a modifié l'algorithme de départ et on converge plus vite (ex. de l'algorithme génétique optimisé !)
- on cherche une solution dans un arbre et on stoppe le pgm

Performances

## Expression des perfs. : efficacité

• **Définition :**

$$e(P) = \frac{S(P)}{P}$$

Taux d'utilisation des ressources, ou fraction de l'accélération idéale

- $e(P) \in [0,1]$ ,  $\in [0\%;100\%]$
- $e(P) > 100\% \Leftrightarrow$  hyper-accelération

• **Remarque :**  
 L'utilisateur s'intéresse surtout à l'accélération obtenue  
 L'acheteur de la machine s'intéresse beaucoup à l'efficacité  
 Le développeur s'intéresse aux deux

Performances

## Expression des perfs. : Référence seq.

• **Choix de la référence séquentielle :**

A quel programme et exécution séquentielle se comparer ?

- Même programme lancé sur un seul processeur ?  
 Même algorithme implanté en séquentiel ?  
 Meilleur algorithme séquentiel connu ?
- Compilation séquentielle avec le même compilateur ?  
 Compilation avec le meilleur compilateur séquentiel ?
- Optimisations séquentielles autorisées par la parallélisation ?  
 Optimisations séquentielles maximales ?
- Exécution sur un seul processeur de la machine parallèle ?  
 Exécution sur la meilleure machine séquentielle ?

Performances

## Expression des perfs. : Référence seq.

• **Tous les choix sont plausibles :**

Chaque choix de référence séquentielle correspond à :

- un point de vue différent,
- une préoccupation différente,
- un objectif d'analyse différent

L'important est de :

- Faire le choix correspondant à sa problématique
- Énoncer clairement ce choix

• **Exemple de choix :**

- Utilisateur final : SON pgm seq. sur SA machine seq.
- Paralléliseur : même algo sur un proc de la machine parallèle

Performances

## Expression des perfs. : Graphe de SU

• **La référence séquentielle peut être obtenue avec :**

Même algo, même langage, même optim seq., même proc

→

$S(P)$

**Bonnes perfs faciles à obtenir**

Meilleur algo, meilleur langage, meilleur optim seq., meilleur proc

→

$S^{a,o,l,p}(P)$

**Bonnes perfs très difficiles à obtenir**

• **Démarche de développement et d'analyse des perfs :**

- 1 – On commence par chercher de bonnes performances pour  $S(P)$
- 2 – Puis on prend des références séquentielles de plus en plus rapides
- 3 – Et on progresse vers  $S^{a,o,l,p}(P)$

Performances

## Expression des perfs. : Graphe de SU

• **On chemine dans un graphe de *Speed up* :**

Mesure et analyse de performances

### 3 – Sources de perte de performances

- Différentes pertes de performances possibles

Performances

### Sources de perte de performances

- Sous-optimisation séquentielle (Aspects séquentiels)
- Fraction séquentielle (Algorithmique et programmation parallèle)
- False-sharing (en mémoire partagée)
- Surcoût des opérations de gestion du parallélisme
- Déséquilibre de charge
- IO séquentielles/séquentialisées (Environnement de développement)
- Sous-optimisation des outils et langages parallèles

Mesure et analyse de performances

### 4 – Loi de Amdahl

- Motivations
- Définitions
- Impact sur le speed up
- Confrontation à la réalité

Performances

### Loi de Amdahl : Motivations

Fraction séquentielle du programme ...

... quel est son impact sur les performances ?

Performances

### Loi de Amdahl : Définitions

Hypothèses de départ :

$$T(1) = T_{seq} = T_s^a + T_p^a$$

Temps de la partie *parallélisable* du programme (au sens de Amdahl)

Temps de la partie *séquentielle* du programme (au sens de Amdahl)

$$T(P) = T_s^a + T_p^a / P + \text{Overhead}$$

Hyp : machine parallèle idéale

- Synchro sans surcoût !
- Msgs instantanés !
- ...

$$T(P) = T_s^a + T_p^a / P$$

Performances

### Loi de Amdahl : Définitions

Conséquence sur le *speed up* :

Par définition du *Speed up* :  $S^a(P) = \frac{T(1)}{T(P)} = \frac{T_{seq}}{T(P)}$

Def. de Amdahl de T(P) :  $S^a(P) = \frac{T_{seq}}{T_s^a + \frac{T_p^a}{P}}$

Soit :  $S^a(P) = \frac{1}{\frac{T_s^a}{T_{seq}} + \frac{1}{P} \cdot \frac{(T_{seq} - T_s^a)}{T_{seq}}}$

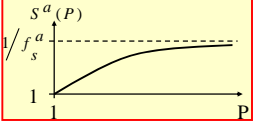
Performances

## Loi de Amdahl : Définitions

- Définition de la fraction séquentielle au sens de Amdahl :
 
$$f_s^a = \frac{T_s^a}{T_{seq}} \in [0;1] \quad f_p^a = \frac{T_p^a}{T_{seq}} \in [0;1] \quad f_s^a + f_p^a = 1$$
- Speed up fonction de la fraction séquentielle :
 
$$S^a(P) = \frac{1}{f_s^a + \frac{1-f_s^a}{P}} = P \cdot \frac{1}{1 + f_s^a \cdot (P-1)}$$

Performances

## Loi de Amdahl : Impact sur Speed up

- Borne supérieure du speed up :
 
$$\lim_{P \rightarrow \infty} S^a(P) = \frac{1}{f_s^a} \quad \text{et} \quad S^a(P) \leq \frac{1}{f_s^a}$$
- Allure du speed up :
 
- Exemple (a.n.) :
 
$$f_s^a = 1\% \Rightarrow \begin{cases} S^a(P) < 100 \\ S^a(100) = 50.3 \end{cases}$$

Petite fraction séquentielle → grosse limitation pour P grand!

Performances

## Amdahl : Confrontation à la réalité

- En réalité les overhead ne sont pas négligeables :
 
$$T(P) = T_s^a + T_p^a / P + \text{Overhead}$$

$\Rightarrow S^{réel}(P) < S^a(P)$

  - ↳ Temps de communication
  - ↳ Temps de synchronisation
  - ↳ Temps d'ordonnancement
  - ↳ ...
- Pourtant on n'observe pas de si mauvais résultats (en général) :
 

“On sait obtenir de bon speed up sur de grosses machines parallèles”

  - ↳ Les fractions séquentielles sont généralement faibles ?
  - ↳ **Le modèle ne correspond pas toujours à la réalité ?**

↳ Loi de Gustafson

Mesure et analyse de performances

## 5 – Loi de Gustafson

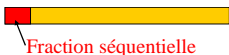
- Motivations
- Définitions
- Impact sur le speed up

Performances

## Loi de Gustafson : Motivations


- 1 – Impact de la fraction séquentielle
 

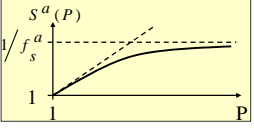
Programme séquentiel



→

Programme parallèle


- 2 – On observe qu'Amdahl n'est pas réaliste (trop pessimiste) :
 



↳ Construire un modèle plus proche de la réalité ...

Performances

## Loi de Gustafson : Définitions

- Idée/Constatation de base :
 

En pratique : on ne diminue pas le temps d'exécution  $T_0$ , mais on augmente la quantité de travail  $W$  traitée en  $T_0$

↓

- On traite  $W_0$  en séquentiel en :  $T(1, W_0)$
  - On traite  $W$  en parallèle en :  $T(P, W)$

Et on choisit P et W tels que :  $T(P, W) = T(1, W_0) = T_0$

Performances

### Loi de Gustafson : Définitions

- Temps que *durerait* le traitement séquentiel de W :

$$T(1, W) = T_s^g(P, W) + P \times T_p^g(P, W) - \text{Overhead}$$

↓ Temps de la partie *parallélisée* sur P processeurs (au sens de Gustafson)  
 ↓ Temps de la partie séquentielle du programme (au sens de Gustafson)

Performances

### Loi de Gustafson : Définitions

- Hypothèse : partie séquentielle constante

$$T_s^g(P_1, W_1) = T_s^g(P_2, W_2) = T_s^g(P, W) = Cte$$

↓ Vrai notamment pour des calculs scientifiques itératifs

Performances

### Loi de Gustafson : Définitions

- Introduction de la fraction séquentielle de Gustafson :

$$f_s^g / T_s^g = f_s^g \times T(P, W), \quad f_s^g \in [0, 1]$$

$$\left. \begin{array}{l} T(P, W) = Cte, \text{ par définition} \\ T_s^g = Cte, \text{ par hypothèse} \end{array} \right\} \rightarrow f_s^g = Cte$$

- Nouvelle expression du temps séquentiel (estimé) :

$$T(1, W) = T_s^g(P, W) + P \times T_p^g(P, W)$$

↓

$$T(1, W) = f_s^g \times T(P, W) + P \times (1 - f_s^g) \times T(P, W)$$

Performances

### Loi de Gustafson : Définitions

- Speed up fonction de la fraction séquentielle :

$$S(P, W) = \frac{T(1, W)}{T(P, W)}$$

$$S^g(P, W) = \frac{f_s^g T(P, W) + P \cdot (1 - f_s^g) T(P, W)}{T(P, W)}$$

$$S^g(P, W) = f_s^g + P \cdot (1 - f_s^g)$$

↓

$$S^g(P) = f_s^g + P \cdot (1 - f_s^g)$$

Speed up au sens de Gustafson

Performances

### Loi de Gustafson : Impact sur speed up

- Speed up non borné :

$$\lim_{P \rightarrow \infty} S^g(P) = \infty$$

- Allure du speed up :

- Bilan de la loi de Gustafson :

- Adopte le **point de vue utilisateur** : augmenter W à T-exec constant
- **Hypothèse optimiste** de partie séquentielle constante ( $T_s^g(P, W) = Cte$ )

Mesure et analyse de performances

## 6 – Liens Amdahl-Gustafson

- Comparaison Amdahl – Gustafson
- Relation entre les fractions séquentielles
- Relation entre les courbes de speed up
- Généralisation des modélisations
- Bilan

Performances

## Amdahl – Gustafson : Comparaison

• Amdahl et Gustafson aboutissent à des conclusions différentes :

Amdahl

Gustafson

Mais :

- Font des hypothèses différentes
- Définissent des fractions séquentielles différentes
- **Ne sont pas incompatibles**

Performances

## Amdahl – Gustafson : Relation entre $f_s$

Amdahl :  $T(P,W) = f_s^a T(1,W) + (1-f_s^a) \cdot \frac{T(1,W)}{P}$

Gustafson :  $T(1,W) = f_s^g T(P,W) + P \cdot (1-f_s^g) T(P,W)$

↓

$$T(1,W) = f_s^g \left( f_s^a T(1,W) + (1-f_s^a) \cdot \frac{T(1,W)}{P} \right) + P \cdot (1-f_s^g) \left( f_s^a T(1,W) + (1-f_s^a) \cdot \frac{T(1,W)}{P} \right)$$

↓

$$1 = f_s^g \left( f_s^a + (1-f_s^a) \cdot \frac{1}{P} \right) + P \cdot (1-f_s^g) \left( f_s^a + (1-f_s^a) \cdot \frac{1}{P} \right)$$

Performances

## Amdahl – Gustafson : Relation entre $f_s$

$$1 = f_s^g \left( f_s^a + (1-f_s^a) \cdot \frac{1}{P} \right) + P \cdot (1-f_s^g) \left( f_s^a + (1-f_s^a) \cdot \frac{1}{P} \right)$$

↓

$$1 = f_s^a \left( f_s^g - \frac{f_s^g}{P} \right) + \frac{f_s^g}{P} + f_s^a \left( P \cdot (1-f_s^g) \left( 1 - \frac{1}{P} \right) \right) + (1-f_s^g)$$

↓

$$0 = f_s^a \left( f_s^g \cdot \left( 1 - \frac{1}{P} \right) + P \cdot (1-f_s^g) \left( 1 - \frac{1}{P} \right) \right) + f_s^g \left( \frac{1}{P} - 1 \right)$$

↓

$$f_s^a \left( f_s^g \cdot \left( 2 - \frac{1}{P} - P \right) + P - 1 \right) = f_s^g \left( 1 - \frac{1}{P} \right)$$

Performances

## Amdahl – Gustafson : Relation entre $f_s$

$$f_s^a \left( f_s^g \cdot \left( 2 - \frac{1}{P} - P \right) + P - 1 \right) = f_s^g \left( 1 - \frac{1}{P} \right)$$

Or :

$$\left( f_s^g \cdot \left( 2 - \frac{1}{P} - P \right) + P - 1 \right) = 0 \Leftrightarrow \begin{cases} P=1 \\ P = -\frac{f_s^g}{1-f_s^g} < 0 !! \end{cases}$$

Donc :  $P \neq 1 \Rightarrow f_s^a = \frac{f_s^g \left( 1 - \frac{1}{P} \right)}{f_s^g \cdot \left( 2 - \frac{1}{P} - P \right) + P - 1}$

$$P > 1 \Rightarrow f_s^a = \frac{f_s^g}{f_s^g \cdot (1-P) + P}$$

Performances

## Amdahl – Gustafson : Relation entre $f_s$

$$P > 1 \Rightarrow f_s^a = \frac{f_s^g}{f_s^g \cdot (1-P) + P}$$

• Pour  $f_s^g$  donné :

$$\lim_{P \rightarrow \infty} f_s^a(f_s^g, P) = 0$$

Tend vers Amdahl idéal !!

• Allure de  $f_s^a$  pour  $f_s^g = 10\%$

$f_s^a(f_s^g, P) : \text{fonction décroissante de } P$

Performances

## Amdahl – Gustafson : Relation entre S

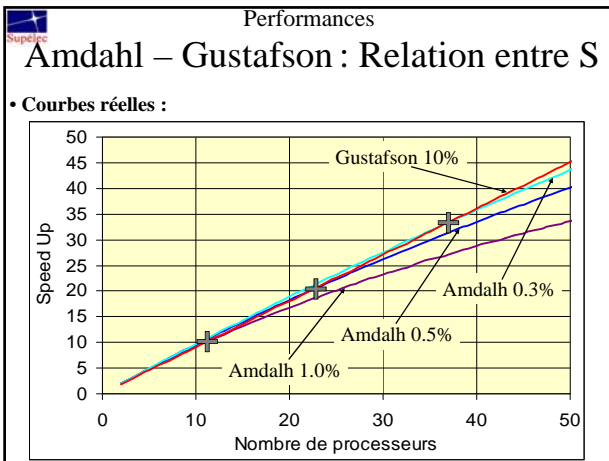
• Expressions des Speed up

$$S^a(P) = 1 / \left( f_s^a + \frac{1-f_s^a}{P} \right)$$

$$S^g(P) = f_s^g + P \cdot (1-f_s^g)$$

• Allures des Speed up

En augmentant P et W (selon Gustafson) à  $f_s^g$  constant :  
 →  $S^g(P)$  intersekte différent  $S^a(P)$ , avec des  $f_s^a$  décroissant



Performances

### Amdahl – Gustafson : Généralisation

• Définitions des parties séquentielles des programmes :

Amdahl :  $T_s(P,W) = f_s^a \cdot T(1,W)$  ..... “Pessimiste” ?

Gustafson :  $T_s(P,W) = Cte$  ..... “Optimiste” ?

• Autres modélisations possibles :

$T_s(P,W) = f_s \cdot \log(T(1,W)) + \dots$

Performances

### Amdahl – Gustafson : Bilan

• Amdahl : W Cte, P ↗, T-exec ↘  
 Réaliste quand on travaille à W Cte  
 Réaliste quand on augmente P pour diminuer T-exec **Informaticien**

• Gustafson : W ↗, P ↗, T-exec Cte  
 Réaliste quand on augmente W sur *certain* pbs  
 Ex : plus de cycles de calculs lors de simulations **Utilisateur**

En général :  
 Valeurs des courbes réelles moins bonnes à cause des *overhead*  
 Allures des courbes réelles entre Amdahl et Gustafson

$Cte < T_s < f_s \cdot T(1,W)$

Mesure et analyse de performances

FIN