SG6: High Performance Computing

Performance métrics, measurement and analysis

Stéphane Vialle

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

Métriques, mesure et analyse de performances

---

1 – Métriques de performance (*T*, *S*, *e*)

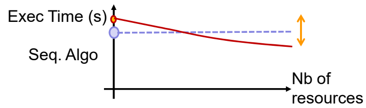
- Observation de la courbe du T_{exec}
- Métrique d'accélération (*Speedup*)
- Métrique d'efficacité (*Efficiency*)
- Choix de la référence séquentielle
- Sources de perte de performances

Observation de la courbe du T_{exec}

Difficultés pour accélérer une application (de taille fixée)

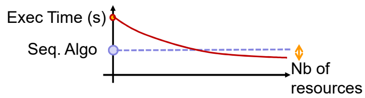
Quand les surcoûts de gestion du parallélisme (*Tsynchro*, *Tcomm*…) sont grands…

…l'accélération est faible !



Quand l'algorithme parallèle a une complexité beaucoup plus grande que le meilleur algorithme séquentiel…

…le gain final reste faible !

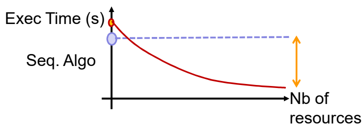


Observation de la courbe du T_{exec}

Courbe de temps prometteuse

Quand l'algorithme parallèle a :
- un temps d'exécution qui décroit significativement
- un surcoût limité sur une seule ressource

…le gain final sera important !



…et on **poursuit** l'analyse

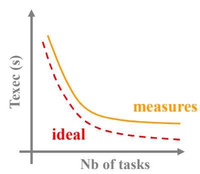
Texec *vs* Texec ideal
Speedup
Efficiency
Size up
Scalability…

Observation de la courbe du T_{exec}

Quelle representation adopter pour un temps d'exécution ?

Remarque Générale :
- Il est préférable de connaître de l'allure de la courbe attendue…
- …et de choisir une représentation qui permet de visualiser *des droites* ou des *formes géométriques simples*

Cas d'un temps d'exécution parallèle :
- courbe idéale : T(P) = T(1)/P
  → une hyperbole
- l'hyperbole est mal identifiée par l'œil…
  …on la confond facilement avec une courbe d'une *autre loi !*



Observation de la courbe du T_{exec}

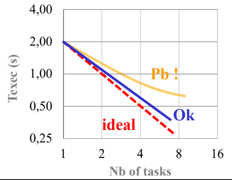
Quelle representation adopter pour un temps d'exécution ?

Remarque Générale :

- Il est préférable de connaître de l'allure de la courbe attendue…
- …et de choisir une représentation qui permet de visualiser *des droites* ou des *formes géométriques simples*

Cas d'un temps d'exécution parallèle :

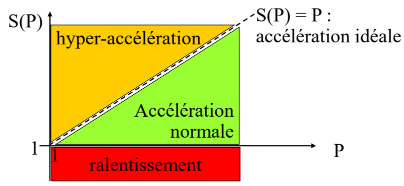
- courbe idéale :
  log(T(P)) = log(T(1)) – log(P)
- en échelle log une hyperbole est une droite de pente -1
- on détecte facilement :
  - un écart complet à la théorie
  - un écart dû à une loi proche



Métrique d'accélération

Speedup :

$$S(P) = \frac{T(1)}{T(P)} \longrightarrow \begin{cases} S(P) < 1 : \text{on ralentit ! mauvaise parallélisation} \\ 1 < S(P) < P : \text{“normal”} \\ P < S(P) : \text{hyper-accélération, analyser \& justifier} \end{cases}$$

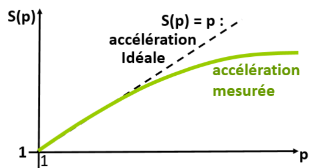


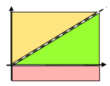
Métrique d'accélération

Speedup :

$$S(P) = \frac{T(1)}{T(P)} \longrightarrow \begin{cases} S(P) < 1 : \text{on ralentit ! mauvaise parallélisation} \\ 1 < S(P) < P : \text{“normal”} \\ P < S(P) : \text{hyper-accélération, analyser \& justifier} \end{cases}$$

Cas standard :

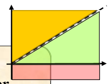




Il y a toujours des sources de perte de performance…

Métrique d'accélération

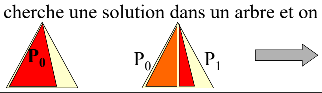
Cas d'hyper-accélération :



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



S(2) = 3

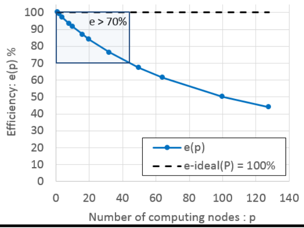
Métrique d'efficacité

Efficacité :

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

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

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



L'utilisateur s'intéresse à l'accélération obtenue

L'acheteur de la machine s'intéresse à l'efficacité des applications exécutées

Le **développeur** s'intéresse aux **deux**

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 ?

Choix de la référence séquentielle

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

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

Exemples :

Utilisateur final →
SON pgm séquentiel sur SA machine séquentielle

Développeur de code parallèle →
SON pgm parallèle sur UN proc de SA machine parallèle

Choix de la référence séquentielle

La référence séquentielle peut être obtenue avec :

Même algo, même langage, même optim seq., même proc → $S_1(P)$ — Bonnes perfs faciles à obtenir

Meilleur algo, meilleur langage, meilleur optim seq., meilleur proc → $S_2(P)$ — **Bonnes perfs très difficiles à obtenir**

Sources de perte de performances

- Sous-optimisation séquentielle
- Impossibilité de vectoriser les boucles

⟺ Sous-utilisation de chaque coeur

- Fraction séquentielle
- Surcoût des synchronisations
- Surcoût des communications
- Déséquilibre de charge entre tâches

⟺ Sous-utilisation des noeuds de calcul

- IO séquentielles/re-séquentiallisées
- Réseau d'interconnexion trop faible

⟺ Plate-forme trop faible

Rmq : certains algorithmes nécessitent des plates-formes très performantes (d'autres non…)

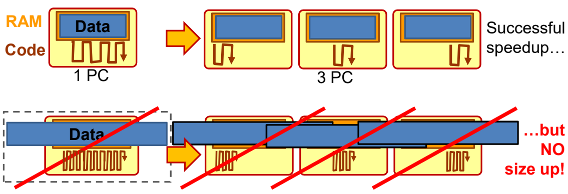
2 – Métriques de *size up*

- Conception d'un code apte au *size up*
- Métrique de *size up* en temps d'exécution
- Métrique de *size up* en temps et ressources
- Bilan : *Size Up* en 3 objectifs successifs
- Exemple expérimental

Conception d'un code apte au *size up*

1er objectif : pouvoir traiter des problèmes plus gros sur plus de rsrcs

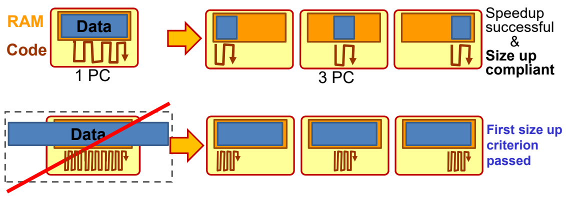
Un code qui **réplique** la plupart de ses données sur toutes les machines sera toujours limité par la taille mémoire d'**une** machine…
…et ne sera **pas apte au *size up***



Conception d'un code apte au *size up*

1er objectif : pouvoir traiter des problèmes plus gros sur plus de rsrcs

Un code qui **répartit** la plupart de ses données sur toutes les machines pourra stocker plus de données sur plus de machines…
… et sera **apte au *size up***

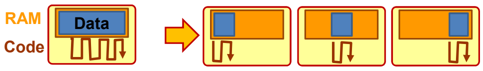


Métriques de *size up*

Conception d'un code apte au *size up*

1[er] objectif : pouvoir traiter des problèmes plus gros sur plus de rsrcs

→ Conception d'une répartition initiale des données, et d'un schéma de communication à volume minimal



- **Besoin de faire circuler les données initiales entre les nœuds** : pour qu'un nœud puisse poursuivre ses calculs sur d'autres données que les siennes
- **Besoin de faire circuler les résultats intermédiaires entre les nœuds** : pour qu'un nœud puisse poursuivre les calculs d'un autre, avec ses propres données

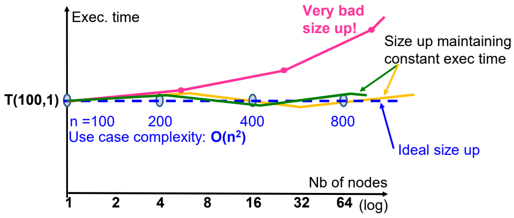
Métriques de *size up*

Métrique de *size up* en temps d'exec.

2[ème] objectif : maintenir constant T_{exec}

$T(1\times n_1,p_1) = T(2\times n_1,p_2) = T(k\times n_1,p_k) = C^{te}$

avec : T(taille pb, nb rsrc)



→ Objectif non atteint !

Métriques de *size up*

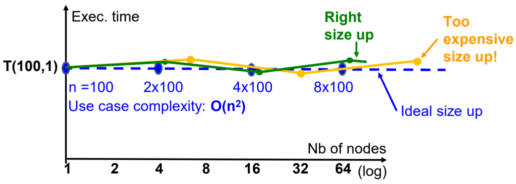
Métrique de *size up* en temps et rsrc.

3[ème] objectif : maintenir constant T_{exec} avec le nombre min de rsrc

$T(1\times n_1,p_1) = T(2\times n_1,p_2) = T(k\times n_1,p_k) = C^{te}$

avec : T(taille pb, nb rsrc)

avec : $O(p_k) = O(calcul(k\times n_1))$



→ Objectif non atteint ! **→ Objectif atteint**

Métriques de *size up*

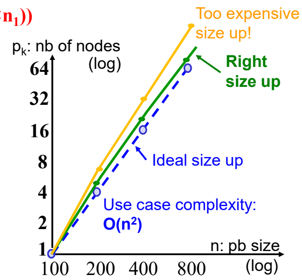
Métrique de *size up* en temps et rsrc.

3[ème] objectif : maintenir constant T_{exec} avec le nombre min de rsrc

$T(1\times n_1,p_1) = T(2\times n_1,p_2) = T(k\times n_1,p_k) = C^{te}$

avec : T(taille pb, nb rsrc)

avec : $O(p_k) = O(calcul(k\times n_1))$



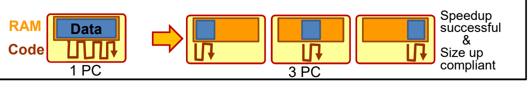
→ Objectif non atteint !

→ Objectif atteint

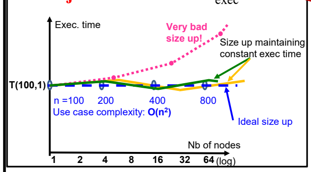
Métriques de *size up*

Bilan : *size up* en 3 objectifs successifs

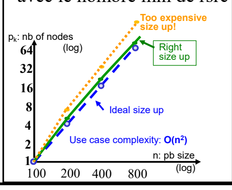
1[er] objectif : pouvoir traiter des problèmes plus gros sur plus de rsrcs



2[ème] objectif : maintenir $T_{exec} = C^{te}$



3[ème] objectif : $T_{exec} = C^{te}$ avec le nombre min de rsrc



Métriques de *size up*

RISEGrid

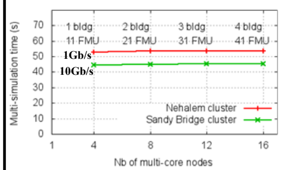
Exemple expérimental

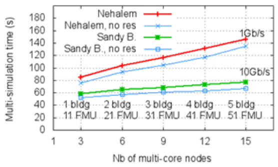
Co-simulation d'échanges thermiques au sein de buildings *faiblement couplés* *(calculs quasi-indépendants)*

Objectif : *size up* à temps constant avec nombre minimal de rsrc

$T(1\times 1building,p_1) = T(2\times 1bldg,p_2) = T(k\times 1bldg,p_k) = C^{te}$

avec : $O(p_k) = O(calcul(k\times 1bldg)) \approx O(k)$





Version 1 : **bcp de calculs** et peu de comm. → **Size up dès 1Gb/s**

Version 2 : **peu de calculs** et peu de comm.→ **Size up avec 10Gb/s**

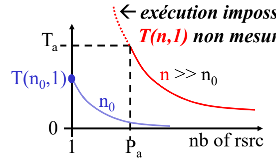
3 – Critères de passage à l'échelle

- Une métrique basée sur le T-exec
- Démarche de *Size Up* et de *Speedup*
- Graphique de passage à l'echelle

Une métrique basée sur le T_{exec}

Quand la taille du pb croît on ne peut plus faire d'exéc séquentielle :

- pas assez de RAM, pas assez de disque…
- exécution trop longue, rsrc non monopolisable…



→ on ne peut pas mesurer de *référence séquentielle !*

→ on ne peut pas calculer de *Speedup !!*

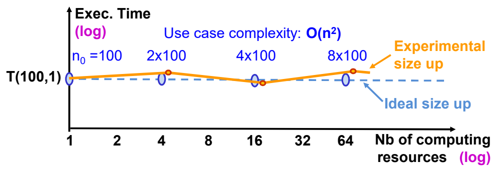
→ Bâtir une analyse seulement sur des mesures du T_{exec}

Démarche de *Size Up* & de *Speedup*

Définition et critères d'un *passage à l'échelle « simple »* :

- *Permettre un **Size Up** pour traiter de plus gros pb sur plus de rsrc*
- *Et de manière économiquement supportable*

→ *Size Up* avec temps d'exéc. constant et nbr minimal de rsrcs



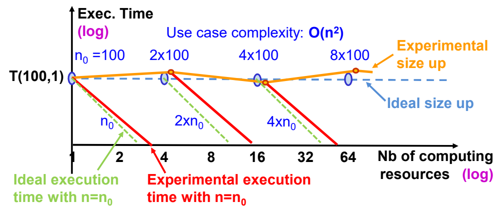
Démarche de *size up* & de *speedup*

Définition et critères d'un *passage à l'échelle* « complet » :

- *Permettre un **Size Up** pour traiter de plus gros pb sur plus de rsrc*
- *Et de manière économiquement supportable*

&

- *Permettre un **Speedup** pour toutes tailles de pb*
- *Avec toujours le même profil de décroissance du temps d'exéc.*

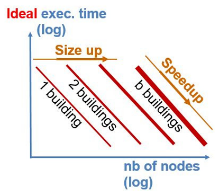


Graphique de *passage à l'échelle*

Constitution et exploitation d'un graphique de passage à l'échelle :

- on mesure T(n,p) pour différentes tailles de pb (n) et nbr de rsrc (p)
- on trace les courbes T(n,p) en échelle log

Cas idéal : pour chaque taille de pb on obtient une droite parallèle aux autres, simplement translatée vers la droite!



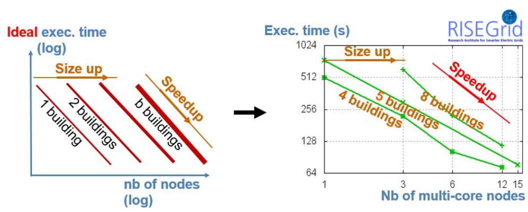
« Pouvoir traiter des problèmes sans limite de taille dans le futur »

Graphique de *passage à l'échelle*

Constitution et exploitation d'un graphique de passage à l'échelle :

- on mesure T(n,p) pour différentes tailles de pb (n) et nbr de rsrc (p)
- on trace les courbes T(n,p) en échelle log

Validation du passage à l'échelle : par comparaison des mesures aux courbes attendues (pentes et positions)

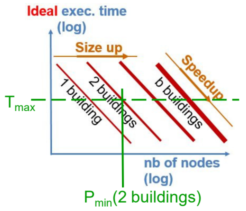


Graphique de *passage à l'échelle*

Constitution et exploitation d'un graphique de passage à l'échelle :

- on mesure T(n,p) pour différentes tailles de pb (n) et nbr de rsrc (p)
- on trace les courbes T(n,p) en échelle log

Utilisation en « abaque » : Pour une taille de pb donnée on peut identifier le nb de rsrcs à utiliser pour respecter un T-exec maximum



Une solution qui *passe à l'échelle complètement* permet de :
- répondre aux besoins de calculs
- nécessiter le minimum de rsrc
- quantifier les rsrcs nécessaires
- planifier les dépenses associées

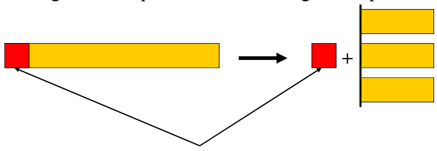
4 – Modélisations sur machines idéales

4.1 - Loi d'Amdahl

Loi d'Amdahl : Motivations

Programme séquentiel

Programme parallèle



Fraction séquentielle du programme …

… quel est son impact sur les performances ?

Loi d'Amdahl : Définitions

Hypothèses de départ :

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

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

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

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

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

Hyp : machine parallèle idéale
- Synchro sans surcoût !
- Msgs instantannés !
- …

Loi d'Amdahl : Définitions

Définition de la fraction séquentielle au sens de Amdahl :

$$\boxed{f_s^a = \frac{T_s^a}{T(1)} = \frac{T_s^a}{T_s^a + T_p^a} \in [0;1]}$$

Et la fraction parallèle: $f_p^a = \frac{T_p^a}{T(1)} = \frac{T_p^a}{T_s^a + T_p^a} \in [0;1]$

D'où : $f_s^a + f_p^a = 1$

Réécriture du temps d'exécution:

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

$$T(P) = f_s^a . T(1) + (1 - f_s^a) . T(1) / P$$

Loi d'Amdahl : Définitions

Conséquence sur le *speedup* :

Par définition du *Speedup* : $S^a(P) = \frac{T(1)}{T(P)}$

Donc : $S^a(P) = \frac{T(1)}{f_s^a . T(1) + (1 - f_s^a) . T(1) / P}$

Soit : $$\boxed{S^a(P) = \frac{1}{f_s^a + \frac{1 - f_s^a}{P}}}$$

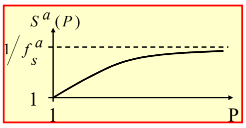
$$\lim_{P \to \infty} S^a(P) = \frac{1}{f_s^a} \qquad S^a(P) \le \frac{1}{f_s^a}$$

$$\boxed{S^a(P) = P . \frac{1}{1 + f_s^a . (P-1)}} \qquad S^a(P) = S^{ideal}(P) . \frac{1}{1 + f_s^a . (P-1)}$$

Modélisations sur machines idéales

Loi d'Amdahl : Impact sur le *speedup*

Allure du *speedup* :



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!

Modélisations sur machines idéales

Loi d'Amdahl : Confrontation à la réalité

En réalité les overhead ne sont pas négligeables :

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

- Temps de communication
- Temps de synchronisation
- Temps d'ordonnancement
- …

$$\boxed{\Rightarrow S^{réel}(P) < S^a(P)}$$

Pourtant on n'observe pas de si mauvais résultats (en général) !

"On sait obtenir de bon *speedup* sur de grosses machines parallèles"

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

Loi de Gustafson

---

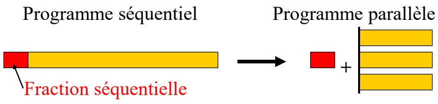
4 – Modélisations sur machines idéales

4.2 - Loi de Gustafson

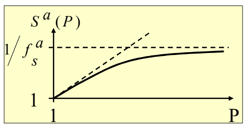
Modélisations sur machines idéales

Loi de Gustafson : Motivation

1 – Impact de la fraction séquentielle (selon Amdahl) ?



2 – Mais on observe qu'Amdahl n'est pas réaliste (trop pessimiste) :



Construire un modèle plus proche de la réalité …

Modélisations sur machines idéales

Loi de Gustafson : Définitions

Idée/Constatation de base :

En pratique : on ne veut pas diminuer le temps d'exécution T_0, mais augmenter 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 : $T(P,W) = T(1,W_0) = T_0$

Modélisations sur machines idéales

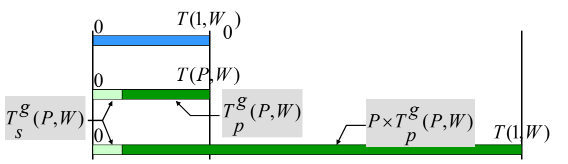
Loi de Gustafson : Définitions

Temps que *durerait* le traitement séquentiel de W :

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

Temps de la partie séquentielle du programme (au sens de Gustafson)

Temps de la partie *parallélisée* sur P processeurs (au sens de Gustafson)

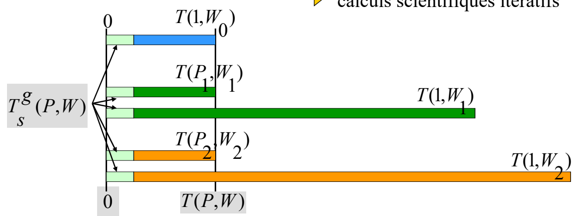


Loi de Gustafson : Définitions

Hypothèse : partie séquentielle constante

$$\boxed{T_s^g(P_1,W_1) = T_s^g(P_2,W_2) = T_s^g(P,W) = Cte}$$

Vrai notamment pour des calculs scientifiques itératifs



Loi de Gustafson : Définitions

Introduction de la fraction séquentielle de Gustafson :

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

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

Nouvelle expression du temps séquentiel (estimé) :

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

$$\Downarrow$$

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

Loi de Gustafson : Définitions

***Speedup* fonction de la fraction séquentielle :**

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

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

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

$$\Downarrow$$

$$\boxed{S^g(P,W) = f_s^g + P.(1-f_s^g)}$$ *Speedup* au sens de Gustafson

Mais la modélisation de Gustafson considère uniquement des couples (P_i,W_i) tels que $T(P_i,W_i) = T(1,W_0)$

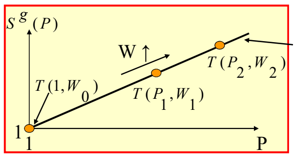
$$\boxed{S^g(P(W)) = f_s^g + P(W).(1-f_s^g)}$$

Loi de Gustafson : Impact sur le *speedup*

***Speedup* non borné :**

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

Allure du *speedup* :



Pente en $(1-f_s^g)$

Bilan de la loi de Gustafson :

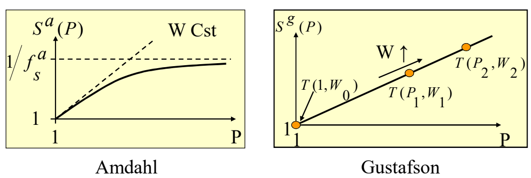
- Adopte le **point de vue utilisateur** : augmenter W à T-exec constant
- **Hypothèse optimiste** de partie séquentielle constante $\left(T_s^g(P,W) = Cte\right)$

4 – Modélisations sur machines idéales

4.3 - Lien Amdahl-Gustafson

Amdahl – Gustafson : Comparaison

Amdahl et Gustafson aboutissent à des conclusions différentes :



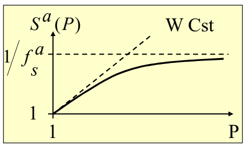
Amdahl — Gustafson

Etudie **l'extensibilité forte** (*strong scalability*) : capacité à décroitre T-exec qd seule la taille de la machine croit.

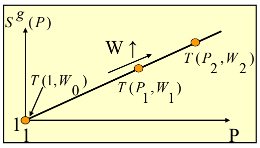
Etudie **l'extensibilité faible** (*weak scalability*) : capacité à maintenir T-exec qd les tailles de la machine et du pb croissent

Amdahl – Gustafson : Comparaison

Amdahl et Gustafson aboutissent à des conclusions différentes :



Amdahl

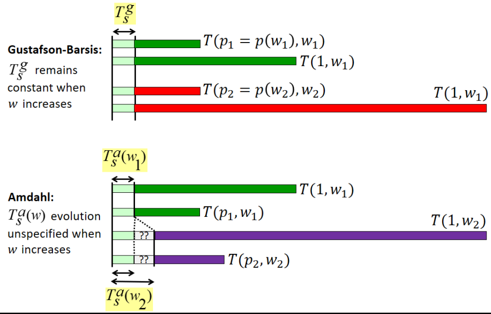


Gustafson

En fait Amdahl et Gustafson :
- Font des hypothèses différentes
- Définissent des fractions séquentielles différentes
- **Ne sont pas incompatibles**

Amdahl – Gustafson : Comparaison

Différence de modélisation de la fraction séquentielle :



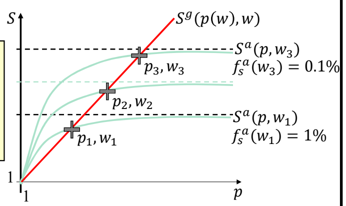
Amdahl – Gustafson : Relation entre *SU*

Expressions completes des *speedup*

$$S^a(P,w) = \frac{1}{f_s^a(w) + \frac{1 - f_s^a(w)}{P}}$$

$$S^g\left(P_g(w)\right) = f_s^g + P_g(w).(1\text{-} f_s^g)$$

Allures des *speedup*



En augmentant P **et** W (selon Gustafson) à f_s^g constant :

→ $S^g(P)$ intersecte différent $S^a(P)$, avec des $f_s^a(w)$ décroissant

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 *certains* 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 Amdalh et Gustafson

5 – Méthodologie de mesure du temps d'exécution

Méthodologie de mesures

Mesures externes :

```
>time myAppli
>/usr/bin/time myAppli
>times myAppli
>timex myAppli
......
```

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

Méthodologie de mesures

Mesures internes :

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

time(), clock() → Compte les clicks d'horloge

gettimeofday(), omp_get_wtime() → Compte le temps écoulé en s

- 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

Méthodologie de mesures

Précision des outils et des mesures :

123456789012.1234567890123456

Capacité maximale de l'outil de mesure

Précision théorique (cf. doc)

Précision expérimentale, vu la fluctuation des exécutions

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

Méthodologie de mesures

Problème fréquent :

Test en mode exclusif (mono-user).
Outil de mesure à 1ms de précision.

⇒ Fluctuation de 500ms d'une exécution à l'autre !!

Et plus encore avec la montée en fréquence automatique des procs. (effet de "chauffe")

Démarche conseillée :

- Mesurer les fluctuations, ne pas les ignorer
  (le *warm up* des processeurs peut perturber les premières)
- Ne pas donner que les valeurs moyennes
- Mesurer des temps > 10s si possible

Méthodologie de mesures

Stocker des meta-données sur les conditions de mesure :

- **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, …

On oublie souvent (et rapidement) à quel benchmark se réfère une série de mesures !

On manque souvent de détail sur les conditions de réalisation d'une série de mesures !

Performance métrics, measurement and analysis

Questions ?