Mesure de la performance

- La performance des ordinateurs s'est améliorée au fil du temps.

- La performance est un terme très vague lorsqu'il est utilisé pour parler des systèmes informatiques en général.

- Le plus souvent ce terme désigne la rapidité avec laquelle un système donné peut exécuter un ou plusieurs programmes. Les systèmes qui exécutent les programmes dans un délai plus court sont dits plus performants que les autres.

- La meilleure mesure de performance qui peut être faite d'un ordinateur consiste à calculer le temps d'exécution du ou des programmes que l'utilisateur souhaite utiliser avec cette machine, mais il n'est généralement pas pratique de tester tous les programmes qui seront lancés sur un système avant de choisir l'ordinateur à acheter.

- Pour résoudre ce problème, les architectes d'ordinateurs ont développé un grand nombre de méthodes de calcul visant à décrire la performance des ordinateurs.

MIPS

- L'un des premiers indicateurs de la performance des ordinateurs a été la vitesse à laquelle une machine donnée pouvait exécuter des instructions.

- On calculait cette vitesse en divisant le nombre d'instructions traitées lors de l'exécution d'un programme par le temps requis pour faire tourner ce programme.

- Le plus souvent, la valeur obtenue était mesurée en millions d'instructions par secondes (MIPS).

- L'unité MIPS est tombée hors d'usage pour la mesure des performances d'ordinateurs, principalement parce qu'elle ne tient pas compte du fait que différents systèmes requièrent souvent un nombre différent d'instructions pour faire tourner un programme.

- La vitesse en MIPS n'indique pas combien d'instructions sont requises pour effectuer une tâche donnée et s'avère donc moins pertinente que d'autres techniques de mesure pour comparer la performance de différents systèmes.

CPI/IPC

- Un autre système de mesure utilisé pour décrire la performance des ordinateurs consiste à évaluer le nombre de cycles d'horloge requis pour exécuter chaque instruction.

- On appelle cet indice de performance le **CPI** (Cycles Par instruction).

- Le CPI d'un programme donné sur un système donné est calculé en divisant le nombre de cycles d'horloge requis pour exécuter le programme par le nombre d'instructions exécutées pour le faire tourner.

- Pour les systèmes qui peuvent exécuter plus d'une instruction par cycle, on fait appel à l'**IPC** (Instructions exécutées Par Cycle) au lieu du CPI.

- L'IPC est calculé en divisant le nombre d'instructions exécutées pour faire tourner le programme par le nombre de cycles d'horloge requis pour son exécution. C'est donc l'inverse du CPI.

- L'IPC et le CPI fournissent la même information et le choix de l'un ou l'autre indice se fait généralement en considérant lequel des deux fournit une valeur supérieur à 1.

- Un indice IPC élevé est généralement significatif d'un haut niveau de performance, tandis qu'un CPI élevé désigne un faible niveau de performance.

- Exemple : un programme donné est constitué d'une boucle de 100 instructions qui s'exécutent 42 fois de suite. S'il faut 16000 cycles pour exécuter le programme sur un système donné, quelles sont les valeurs de CPI et d'IPC pour ce programme ?

- Nombre total d'instructions : 42 x 100 = 4200 instructions

- Nombre de cycle requis pour exécuter le programme : 16000 cycles

- CPI = 16000 / 4200 = 3.81

- IPC = 4200 / 16000 = 0,26

- En général, l'IPC et le CPI sont des mesures encore moins utiles pour évaluer la performance des systèmes actuels que ne l'est le nombre de MIPS, parce que ces indicateurs ne contiennent aucune information concernant la fréquence de l'horloge du système ou le nombre d'instructions que le système requiert pour accomplir une tâche donnée.

- Si vous connaissez le nombre de MIPS d'un système pour un programme donné, vous pouvez le multiplier par le nombre d'instructions exécutées pour faire tourner le programme afin de déterminer le temps d'exécution du programme.

  Temps d'exécution d'un programme = MIPS x nombre d'instructions exécutées

- Si vous connaissez le CPI d'un programme donné, vous pouvez le multiplier par le nombre d'instructions dans le programme afin d'obtenir le nombre de cycles requis pour son exécution.

  Nombre de cycles = CPI x nombre d'instructions

- Le CPI et l'IPC sont rarement utilisés pour comparer les systèmes informatiques actuels

Bancs d'essai (Benchmarks)

- Les MIPS et les CPI/IPC restent des indices de mesure de performance sérieusement limités. Les bancs d'essais constituent une troisième méthode de mesure des performances système. Ils ont été développés pour répondre aux limites des unités MIPS et CPI/IPC.
- Les bancs d'essai regroupent une série de programmes considérés être représentatifs de ceux qui tourneront sur le système testé. La note d'évaluation du système soumis à la suite de tests est calculée en considérant le temps requis par le système pour exécuter tous les programmes du banc d'essai.
- De nombreux bancs d'essais ont été mis au point pour estimer les performances d'un système avec différents types d'applications.
- L'un des bancs d'essais les plus connus est celui de la **SPEC** (Standard Performance Evaluation Corporation).
- Les bancs d'essais offrent un grand nombre d'avantage. Tout d'abord, les résultats de performance reposent sur des temps d'exécution totale et non sur une vitesse d'exécution de l'instruction. Ensuite, ils opèrent une moyenne des performances système sur plusieurs programmes différents afin d'estimer une vitesse moyenne de traitement.

- Les données globales d'un banc d'essai constituent donc un indicateur plus fiable de la performance générale d'un système que ses résultats en MIPS obtenus sur tel ou tel programme.
- En outre, de nombreux bancs d'essais requièrent que les fabricants publient les résultats de leur système pour chacun des programmes inclus dans le banc d'essai, aussi devient-il possible d'établir des comparaisons directes de résultats lorsque vous savez par avance qu'un système sera utilisé pour un type d'application particulier.
- **MFLOPS** (Million Floating point Operations per Second) : est la mesure traditionnelle utilisée pour décrire la vitesse d'un superordinateur. La mesure MFLOPS est basée sur la normalisation du moment de l'exécution d'une instruction de l'ordinateur à une quantité constante de temps lorsque toutes les instructions ne s'exécutent pas en même temps.
- La variation dans le temps d'une instruction peut varier entre 29 et 600 cycles horloge. La mesure MFLOP est une bonne approximation de la vitesse d'un superordinateur tant que les données de référence communes pour l'exécution d'une instruction sont bien publiés ce qui permet une bonne comparaison entre deux superordinateurs concurrents.

- Pour calculer le MFLOPS d'une machine il faut suivre les consignes suivantes :

  - si y (cycles/instruction) est le nombre de cycles nécessaire pour exécuter une instruction ;
  - et x = vitesse d'exécution du processeur en cycles/seconde, cette information peut être obtenue à partir de la documentation fournie avec le processeur ou à l'aide des programmes d'un banc d'essais disponible (par exemple, la vitesse d'exécution d'un processeur Macintosh IIx est de 15.5572 m cycles/seconde).
  - Vitesse du processeur en MFLOPS = x / y.

Accélération

- Les architectes d'ordinateurs utilisent souvent le terme d'accélération pour décrire la manière dont la performance d'une architecture se modifie lorsque différentes améliorations lui sont apportées.

- L'accélération correspond simplement au rapport :

$$\text{Accélération} = \text{Temps d'exécution}_{\text{avant}} / \text{Temps d'exécution}_{\text{après}}$$

La loi d'Amdahl

- La règle la plus importante pour la conception de systèmes informatiques à haute performance consiste à tenter d'accélérer avant tout les tâches les plus courantes.
- Qualitativement, cela veut dire que l'impact d'une amélioration de performance sur la performance générale dépend à la fois de l'efficacité de l'amélioration lorsqu'elle est mise à contribution et de la fréquence avec laquelle cette amélioration peut être exploitée en général.
- Quantitativement, cette règle a été définie par la loi d'Amdahl :

  $\text{Temps d'exécution}_{\text{nouveau}} = \text{Temps d'exécution}_{\text{ancien}} \times [\text{Frac}_{\text{inutilisée}} + (\text{Frac}_{\text{utilisée}} / \text{Accélération}_{\text{utilisée}})]$
- Dans cette équation, $\text{Frac}_{\text{inutilisée}}$ correspond à la fraction de temps (et non d'instructions) durant laquelle l'amélioration n'est pas exploitée. $\text{Frac}_{\text{utilisée}}$ correspond au temps durant lequel l'amélioration est utilisée (il s'agirait de l'accélération générale si l'amélioration était constamment mise à contribution).
- Notez que $\text{Frac}_{\text{utilisée}}$ et $\text{Frac}_{\text{inutilisée}}$ sont calculés en utilisant le temps d'exécution avant que la modification ne soit appliquée. Si ces valeurs étaient calculées en utilisant le temps d'exécution après modification, les résultats obtenus seraient incorrects.
- La loi d'Amdahl peut être réécrite en intégrant la définition de l'accélération :

  $\text{Accélération} = 1/[\text{Frac}_{\text{inutilisée}} + (\text{Frac}_{\text{utilisée}} / \text{Accélération}_{\text{utilisée}})]$

- **Exemple :** Supposons qu'une architecture donnée ne possède pas de support matériel pour la multiplication. Les multiplications doivent donc être effectuées en opérant des additions répétées. S'il faut 200 cycles pour réaliser une multiplication logicielle et 4 cycles pour réaliser la multiplication matérielle, quelle est l'accélération générale offerte par l'implémentation d'un support matériel de la multiplication si le programme testé passe 10 % de son temps à effectuer des multiplications ? Qu'en serait-il si le programme passait 40% de son temps à effectuer des multiplications ?
- Dans les deux cas, l'accélération obtenue grâce à la multiplication matérielle est de 200/4 = 50 (le rapport entre le temps requis pour une multiplication sans le support matériel et celui requis avec le support matériel).
- Si le programme passe 10% de son temps à effectuer des multiplications, $Frac_{inutilisée} = 0.9$ et $Frac_{utilisée} = 0.1$.
- En appliquant ces valeurs à la loi d'Amdahl, on obtient :

  Accélération = 1/ [0.9 + (0,1/50)] = 1.11.
- Si le programme passe 40% de son temps à effectuer des multiplications, l'accélération sera = 1/[0.6 + (0.4 / 50)] = 1.64.

Exercice 1 :

Lors de l'exécution d'un programme donné, l'ordinateur A exécute 100 MIPS contre 75 MIPS pour l'ordinateur B. en revanche, l'ordinateur A prend 60 secondes pour exécuter le programme, tandis que l'ordinateur B n'en requiert que 45.

Comment cela est-il possible ?

Exercice 1 :

Le nombre de MIPS indique la vitesse d'exécution des instructions par le processeur, mais chaque architecture de processeur requiert un nombre différent d'instructions pour réaliser un calcul donné. Si l'ordinateur A doit exécuter un nombre plus important d'instructions que l'ordinateur B pour exécuter le programme, il sera parfaitement possible qu'il prenne plus de temps pour le faire, même s'il parvient à exécuter plus d'instructions par seconde.

Exercice 2 :

Un programme exécuté sur un système donné requiert 1000 000 de cycles. Si le système réalise un CPI de 40, combien d'instructions ont été exécutées pour faire tourner le programme ?

Exercice 2

CPI= NbCycles/NbInstructions.
Donc NbInstrcvutions=NbCycles/CPI.
1000000/40=25000 instructions ont donc été exécutées pour faire tourner le programme

Exercice 3 :

Quel est l'IPC d'un programme qui exécute 35000 instructions et requiert 17000 cycles de temps d'exécution ?

Exercice 3 :

IPC= NbInstructions/NbCycles.
35000/17000=2.06

Exercice 4 :

Soit un programme contenant 200 000 instructions dont 40% sont des opérations flottantes (possédants un CPI de 10), et le reste d'instructions divers (possédants un CPI de 2). Quel sera le temps d'exécution du programme sachant que le temps de cycle est de 10^{-6} secondes ?

Exercice 4 :

Nombre d'instructions flottantes = 40% des 200 000 instructions = 80 000 instructions
Nombre d'instructions diverses = 60% des 200 000 instructions = 120 000 instructions

Nombre de cycles des instructions flottantes = CPI*Nb Inst = 800 000 cycles
Nombre de cycles des instructions diverses = CPI*Nb Inst = 240 000 cycles

Donc nombre total de cycles = 1 040 000 cycles
Temps d'exécution = nombre de cycle * temps de cycle = $1.04*10^{6}*10^{-6}$ = 1.04 secondes

Exercice 5 :

Un ordinateur C obtient une note de 42 auprès d'un banc d'essai (les notes les plus élevées correspondant aux meilleures performances). Un ordinateur D obtient la note 35 pour le même banc d'essai. Lorsque vous exécutez votre programme, vous constatez pourtant que l'ordinateur C prend 20% de temps de plus que l'ordinateur D pour l'exécution du programme. Comment cela est-il possible ?

Exercice 5 :

L’explication la plus plausible est que votre programme dépend fortement d’un aspect du système qui n’est pas mis à l’épreuve par le banc d’essai. Par exemple, votre programme peut avoir à réaliser un très grand nombre de calculs à virgule flottante, alors que le banc d’essai s’intéressait plutôt à la performance sur les entiers, ou vice-versa.

Exercice 6 :

Supposons qu'un ordinateur prenne 90% de son temps à gérer un type particulier de calcul lorsqu'il exécute un programme donné et que son fabricant apporte une modification qui améliore sa performance sur ce type de calcul en la multipliant par 10.

a) Si le programme prenait à l'origine 100 secondes pour s'exécuter, combien de temps lui faudra-t-il après la modification ?
b) Quelle est l'accélération réalisée en passant de l'ancien au nouveau système ?
c) Quelle fraction de son temps d'exécution le nouveau système prend-il pour exécuter le type de calcul qui a été amélioré ?

Exercice 6 :

a) Il s'agit d'une application directe de la loi d'Amdahl :

$$\text{Temps d'exécution}_{nouveau} = \text{Temps d'exécution}_{ancien} \times [\text{Frac}_{inutilisée} + (\text{Frac}_{utilisée} / \text{Accélération}_{utilisée})]$$

$\text{Temps d'exécution}_{ancien} = 100s$ $\quad \text{Frac}_{utilisée} = 0.9$ $\quad \text{Frac}_{inutilisée} = 0.1$ $\quad \text{Accélération}_{utilisée} = 10$

$\text{Temps d'exécution}_{nouveau} = 100 \times [0.1 + 0.9/10)] = 19$ secondes

b) En utilisant la définition de l'accélération, on obtient une accélération de 5,3. On peut également appliquer les valeurs à la loi d'Amdahl pour obtenir le même résultat.

c) La loi d'Amdahl ne nous permet pas de répondre directement à cette question. Le système d'origine passait 90% de son temps à exécuter le type de calcul qui a été amélioré, aussi consacrait-il 90 secondes à cela pour un programme de 100 secondes. La performance de calcul ayant été multipliée par 10, le système amélioré passe 90/10 = 9 secondes à exécuter ce type de calcul. Puisque 9 secondes correspondent à 47% de 19 secondes (le nouveau temps d'exécution du programme), le nouveau système passe à présent 47% de son temps à exécuter le type de calcul qui a été amélioré.

Exercice 7 :

Un ordinateur passe 30% de son temps à accéder à la mémoire, 20% à réaliser des multiplications et 50% à exécuter d'autres instructions. En tant qu'architecte d'ordinateur, vous devez choisir s'il faut améliorer la mémoire, la multiplication matérielle ou l'exécution d'instructions non multiplicatrices. L'espace restant sur la puce ne permet de réaliser qu'une seule amélioration et chaque amélioration se concrétise par un doublement des performances de calcul correspondantes.

a) Sans réaliser le moindre calcul, quelle amélioration vous semble pouvoir apporter la plus importante augmentation de performance et pourquoi ?
b) Quelle accélération offrirait chacune de ces trois améliorations ?

Exercice 7 :

a) L'amélioration de l'exécution des instructions non multiplicatrices doit être la plus avantageuse. Chaque amélioration augmente la performance des capacités concernées à même proportion. Or, le système passe plus de temps à exécuter des instructions non multiplicatrices qu'à traiter l'une ou l'autre des deux autres catégories d'opérations. La loi d'Amdahl stipulant que l'impact global d'une amélioration augmente à mesure qu'augmente la fraction de temps durant laquelle elle est utilisée, on en déduit que l'amélioration du traitement des instructions non multiplicatrices fournira les meilleurs résultats en termes de performance.

b) En appliquant à la loi d'Amdahl les valeurs du pourcentage de temps d'utilisation et de l'amélioration en cours d'utilisation :

$$\text{Accélération} = 1/[\text{Frac}_{\text{inutilisée}} + (\text{Frac}_{\text{utilisée}} / \text{Accélération}_{\text{utilisée}})]$$

$= 1/[0.7 + (0.3 / 2)] = 1.18$ (accès à la mémoire)

$= 1/[0.8 + (0.2 / 2)] = 1.11$ (multiplications)

$= 1/[0.5 + (0.5 / 2)] = 1.33$ (autres instructions)

Ces résultats confirment nos raisonnements précédents.

Exercice 8 :

Quelle amélioration permet de réduire le plus fortement le temps d'exécution : une amélioration utilisée 20% du temps qui multiplie la performance par deux lorsqu'elle est utilisée ou une amélioration 70% du temps qui multiplie la performance par 1.3 lorsqu'elle est utilisée.

Exercice 8 :

En appliquant la loi d'Amdahl, on obtient l'équation suivante pour la première amélioration :

$$\text{Temps d'exécution}_{nouveau} = \text{Temps d'exécution}_{ancien} \times [\text{Frac}_{inutilisée} + (\text{Frac}_{utilisée} / \text{Accélération}_{utilisée})]$$

$$\text{Temps d'exécution}_{nouveau} = \text{Temps d'exécution}_{ancien} \times [0.8 + (0.2 / 2)]$$

➔ $\text{Temps d'exécution}_{nouveau} / \text{Temps d'exécution}_{ancien} = 0.9 = 90\%$.

Ca veut dire que le temps d'exécution avec amélioration correspond à 90% du temps d'exécution sans amélioration.

En appliquant la loi d'Amdahl aux valeurs de la deuxième amélioration,on obtient :

$$\text{Temps d'exécution}_{nouveau} = \text{Temps d'exécution}_{ancien} \times [0.3 + (0.7 / 1.3)]$$

➔ $\text{Temps d'exécution}_{nouveau} / \text{Temps d'exécution}_{ancien} = 0.84 = 84\%$.

En conséquence, c'est la deuxième amélioration qui aura le plus fort impact sur le temps d'exécution global, bien qu'elle représente une amélioration moindre lorsqu'elle est utilisée.

Exercice 9 :

Un architecte d'ordinateur conçoit le système mémoire pour la nouvelle génération d'un processeur. Si dans sa version courante, le processeur passe 40% de son temps à traiter des références mémoires, de combien l'architecte doit-il accélérer le système mémoire pour proposer une accélération générale de 1.2 ?

Qu'en serait-il pour une accélération générale de 1.6 ?

Exercice 9 :

Pour résoudre ce problème, nous appliquons la loi d'Amdahl pour les accélérations locales, la variable $\text{Accélération}_{\text{utilisée}}$ devenant l'inconnue (soit x) au lieu de l'Accélération générale. La valeur de $\text{Frac}_{\text{utilisée}}$ est 0.4 car le système original passe 40% de son temps à gérer des références mémoire. La valeur de $\text{Frac}_{\text{inutilisée}}$ est donc 0.6. Pour une augmentation de 20% des performances générales, on obtient :

$\text{Accélération} = 1.2 = 1/[0.6 + (0.4 / x)] \rightarrow x = 1.71$

Pour obtenir la valeur $\text{Accélération}_{\text{utilisée}}$ requise pour une accélération générale de 1.6, la seule valeur à modifier dans l'équation précédente est celle de l'accélération générale. On obtient ainsi une valeur $\text{Accélération}_{\text{utilisée}} = 16$.

Cet exemple nous montre à quel point les efforts sont peu récompensés lorsqu'on ne fait porter les améliorations que sur un seul aspect des performances du système. Pour faire passer l'accélération générale de 1.2 à 1.6, nous sommes contraints de multiplier pratiquement par 10 le paramètre $\text{Accélération}_{\text{utilisée}}$, car les 60% du temps durant lesquels la mémoire n'est pas utilisée affectent sans cesse plus fortement la performance générale à mesure que nous améliorons la performance de la mémoire.

Exercice 10 :

Considérez une architecture qui possède quatre types d'instruction : additions, multiplications, opérations mémoire et branchements. Le tableau qui suit indique le nombre d'instructions associé à chaque type d'opération pour le programme dont vous vous préoccupez, le nombre de cycles que prend ce programme pour exécuter chaque type d'instruction et l'accélération de l'exécution du type d'instruction offerte par une amélioration proposée (chaque amélioration n'affecte qu'un seul type d'instruction).

Classez les améliorations liées à chaque type d'instruction selon l'ordre de leur impact sur la performance générale.

Type d'instruction	Nombre d'instructions	Temps d'exécution	Accélération pour le type
Addition	10 millions	2 cycles	2.0
Multiplication	30 millions	20 cycles	1.3
Mémoire	35 millions	10 cycles	3.0
Branchement	15 millions	4 cycles	4.0

Exercice 10 :

Pour résoudre ce problème, nous devons en premier lieu calculer le nombre de cycles nécessaires à l'exécution de chaque type d'instruction avant que les améliorations ne soient apportées et la fraction du nombre de cycles passée à exécuter chaque type d'instruction ($Frac_{utilisée}$ pour chaque instruction).

Ceci nous permettra d'utiliser la loi d'Amdahl pour calculer l'accélération générale offerte par chacune des améliorations proposées. En multipliant le nombre d'instructions de chaque type par le temps d'exécution d'une instruction, nous obtiendrons les nombres de cycles passés à exécuter chaque type d'instruction. En additionnant ces valeurs, nous obtiendrons le nombre total de cycles requis pour exécuter le programme. On obtient ainsi les valeurs présentées dans le tableau suivant (le temps d'exécutions total étant de 1 030 millions de cycles) :

Type d'instruction	Nombre d'instructions	Temps d'exécution	Accélération pour le type	Nombre de cycles	Fraction* de cycles
Addition	10 millions	2 cycles	2.0	20 millions	1.94%
Multiplication	30 millions	20 cycles	1.3	600 millions	58.25%
Mémoire	35 millions	10 cycles	3.0	350 millions	33.98%
Branchement	15 millions	4 cycles	4.0	60 millions	5.83%

*Fraction de cycles = (nombre de cycles/ nombre total de cycles) x 100

On peut ensuite appliquer ces valeurs à la loi d'Amdahl en utilisant les fractions de cycles comme variables $Frac_{utilisée}$ afin d'obtenir l'accélération générale pour chacune des améliorations.

On en déduit que l'amélioration des opérations de mémoire fournit la meilleure accélération générale. Ensuite vient l'amélioration des multiplications, puis celle des branchements et enfin celle des additions :

Type d'instruction	Nombre d'instructions	Temps d'exécution	Accélération pour le type	Nombre de cycles	Fraction* de cycles	Accélération générale
Addition	10 millions	2 cycles	2.0	20 millions	1.94%	1.01
Multiplication	30 millions	20 cycles	1.3	600 millions	58.25%	1.15
Mémoire	35 millions	10 cycles	3.0	350 millions	33.98%	1.39
Branchement	15 millions	4 cycles	4.0	60 millions	5.83%	1.05