Introduction

Que cherche-t-on à optimiser ?

Matthieu Boileau
Bastien Di Pierro

Définition

Optimiser

> *Donner à un système les meilleures conditions d'utilisation, de fonctionnement, de rendement.*

Exemple du calcul scientifique

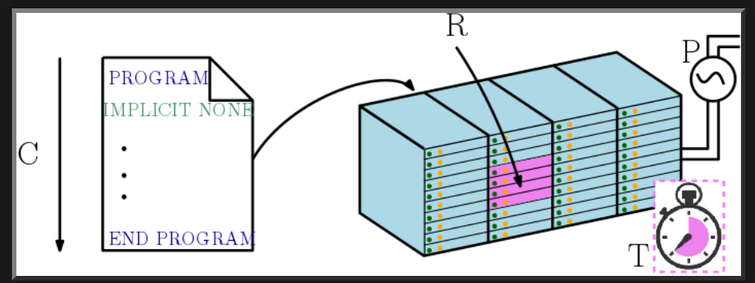
Que cherche-t-on à optimiser ?

Eléments à disposition

- Un code de calcul :
  - Un algorithme
  - Un langage
- Un calculateur :
  - Une architecture matérielle et logicielle
  - Une alimentation électrique

Métriques

Lancement d'un calcul



- C : charge de travail
- R : ressources de calcul
- T : temps total
- P : puissance électrique

Une approche quantitative

$T = \int_0^C \frac{1}{V(l)} dl$ (V : vitesse d'exécution)

$R = \int_0^C \frac{1}{\rho(l)} dl$ (ρ : densité de charge)

$E = \int_0^T P(t) dt$ (E : énergie consommée)

... Tout est lié !

Optimisation

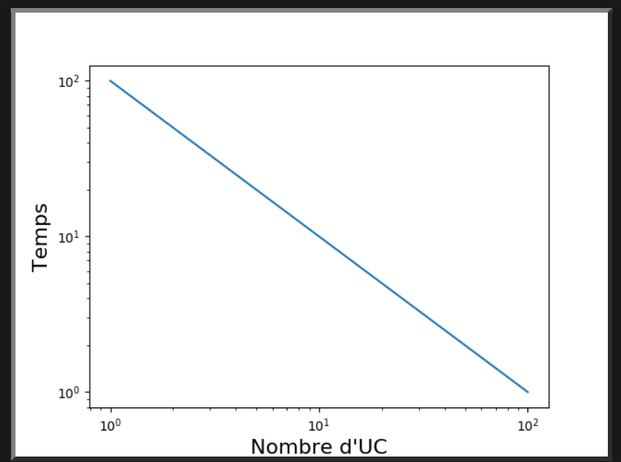
Que cherche-t-on à optimiser ?

- Rapidité d'exécution : $\frac{dT}{d\zeta} = 0$
- Economie de ressources : $\frac{dR}{d\zeta} = 0$
- Performance énergétique : $\frac{dE}{d\zeta} = 0$

Minimisation par rapport à quoi ? $\zeta = ...$?

Un équilibre à trouver !

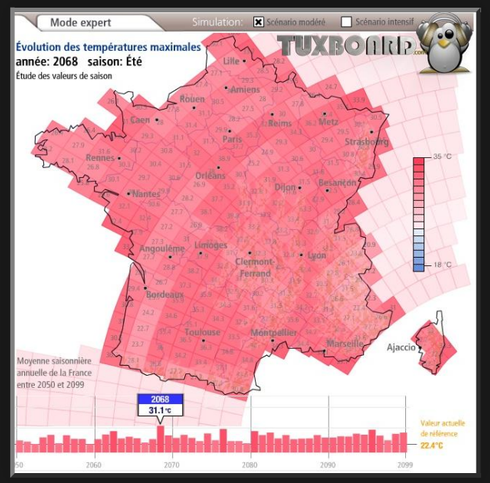
Minimiser une quantité tend à maximiser les autres.



Cas du calcul intensif

On minimise le temps d'exécution par rapport aux ressources :

- Calcul en météorologie
- Réalité virtuelle (temps réel)



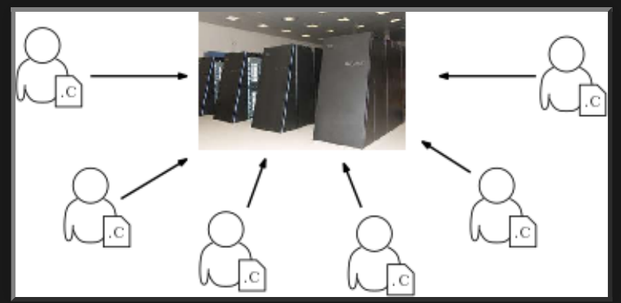
Cas du calcul "écologique"

On minimise la consommation par rapport à la charge :

- Calcul distribué (BOINC)
- Architecture multi-coeurs ($P_{elec} \propto f^3$)

Cas du calcul communautaire

On minimise la ressource par rapport à la charge (calculateurs partagés).



Benchmark

Benchmark :

le point de référence d'une application

Mesure des performances sur une architecture donnée :

- temps
- ressources
- consommation
- ...

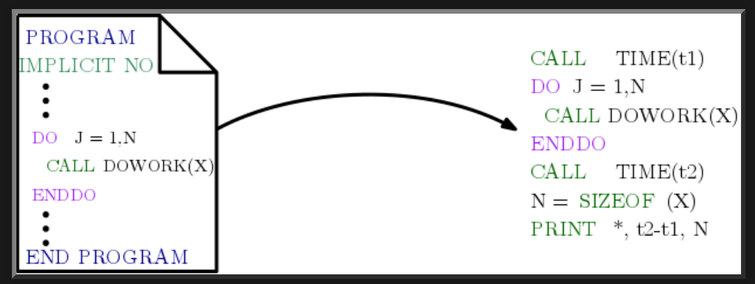
Dépend :

- de la machine
- du système d'exploitation
- du langage
- de l'algorithme
- ...

Micro benchmark

Mesure des performances d'une toute petite portion d'un code :

- $T = O(\mu s-ms)$
- $UC = O(1)$, Mémoire $=O(Ko)$
- indépendant du contexte



Micro benchmark

Un exemple

Résolution d'un problème par éléments finis

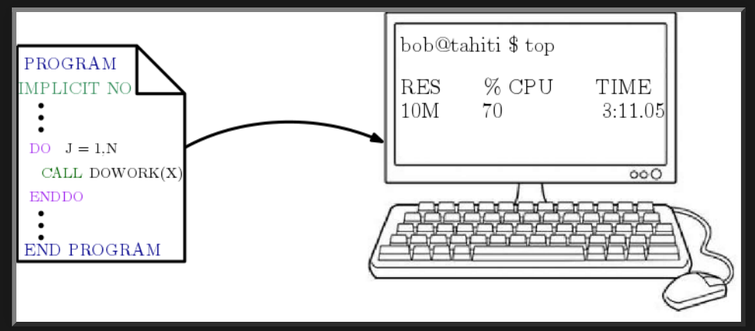
```
Architecture :               Raspberry Pi 3B+
                             Python v3.5

Nombre de degre de liberte :  48
Assemblage du systeme      : 0.026432 s,        18.05 Kbytes
Calcul du second membre    : 0.000778 s,        0.42 Kbytes
Resolution du systeme      : 0.000579 s,        36.11 Kbytes
Calcul des valeurs propres : 0.000872 s,        36.53 Kbytes
```

Macro benchmark

Mesure des performances d'un code entier sur une architecture donnée :

- $T = O(ks - Ms)$
- $UC = 1-10^N$, Mémoire $= O(Go)$
- dépend de tout le contexte (architecture, charge, ecriture disque, ...)



Macro Benchmark

Un exemple

Calcul de turbulence 3D

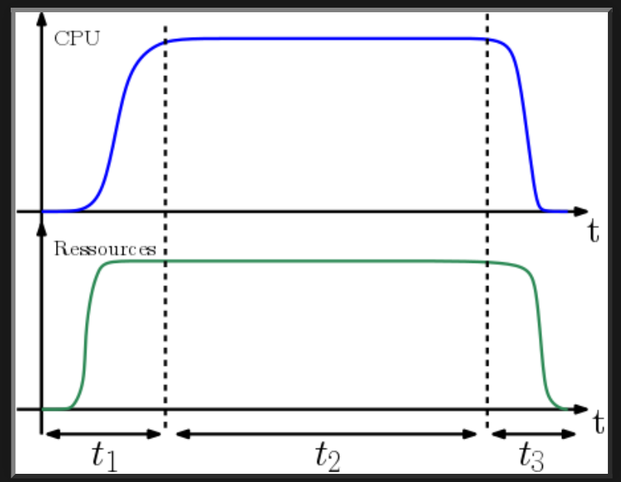
```
Navier-Stokes 3D
Langage : FORTRAN
Algorithme :
 - Decomposition Spectrale (FFT)
 - Semi implicite
 - Resolution des SL : GMRES

Calculateur     : P2CHPD
Parallelisation : MPI

Nombre d UC       : 64 coeurs
temps total       : 33 267 s
Mémoire totale    : 16.8 Go
Espace disque     : 957 Mo
Consommation elec : 15.9 kWh (Estimation)
```

Transitoires et asymptotes

Profil d'exécution d'un code



- t_1 : montée en charge, allocation des ressources (mémoire, CPU, ...)
- t_2 : régime de croisière asymptotique
- t_3 : désallocation mémoire, libération des threads ...

D'autres transitoires

Au sein d'un ensemble de runs d'un même benchmark, d'autres transitoires peuvent se manifester :

- mise en cache du binaire du benchmark, de ses bibliothèques, de ses fichiers de configuration
- téléchargement de données par le réseau

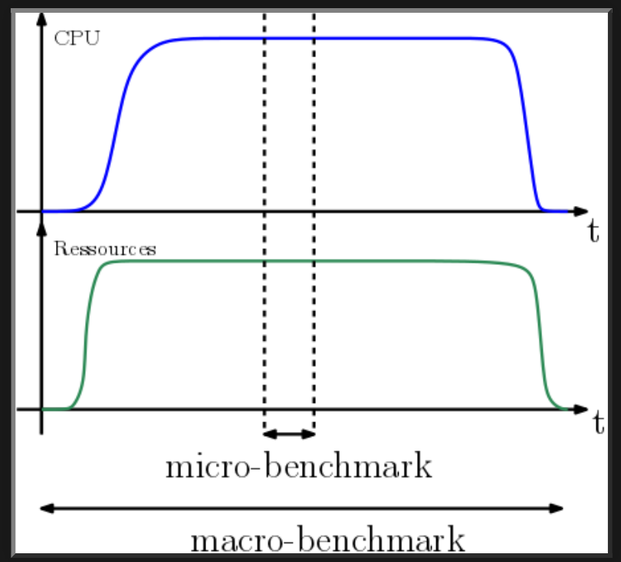
Un benchmark représentatif

En général : $t_1 \approx t_3$

Pour établir un benchmark :

- le régime asymptotique doit être atteint
- $t_2 >> t_3$

Un benchmark représentatif



- t_2 représente une portion importante du temps total
- le microbenchmark se fait dans la région asymptotique

Débit et latence

Débit et latence

Débit

Quantité d'information accessible par unité de temps :

- en (k-M-G) octet/s
- dépend de la (des) architecture(s) "traversée(s)"

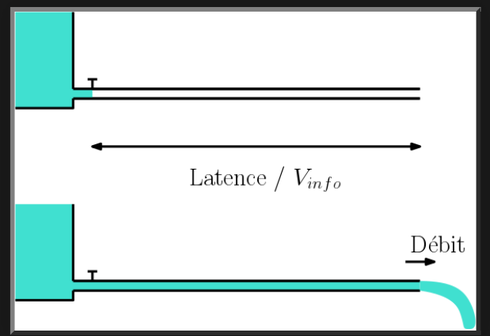
Latence

Temps d'établissement d'une connection :

- en μs ou ms
- dépend de l'architecture et de la distance

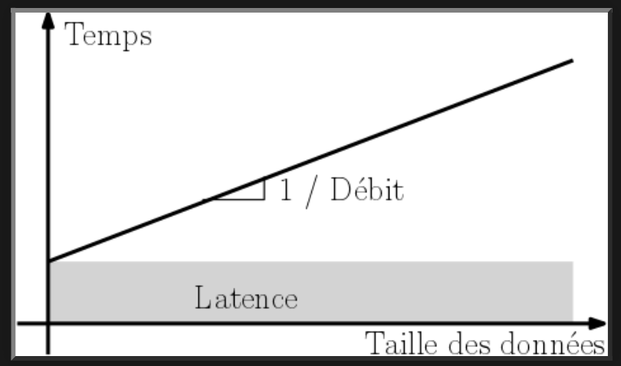
Débit et Latence

Analogie du tuyau d'arrosage



Perception humaine

$T_{total} = Latence + \frac{Volume}{Débit}$



Perception humaine

$T_{total} = Latence + \frac{Volume}{Débit}$

Peut ralentir un algorithme : forte influence sur le benchmark d'un code !

Un exemple

Télécharger 64go sur Marseille/Paris

2 cas limites :

	Pigeon voyageur	Ethernet
Latence	liée à $V_{info} \approx 50 \text{km/h}$	≈ 0
Débit	$\approx \infty$	$\approx 1Mo/s$
T	$\approx 16h$	$\approx 18h$

Conclusion

Définir un bon benchmark

- représentatif du problème
- reproductible sur plusieurs architectures ou plusieurs langages
- facile à implémenter, mesurer, comparer

Exemple

Benchmark LINPACK (J. Dongarra *et al.*)

- résout $A.x = b$
- FORTRAN
- Estime l'écart entre performances théoriques et réelles d'une architecture
- Utiliser dans le classement Top500