Mieux calculer avec un ordinateur

Présentation au Congrès Pluridisciplinaire des Doctorants de l'UPVD

Amine Najahi

Thèse encadrée par Matthieu Martel et Guillaume Revy

au sein de l'équipe-projet DALI

7 juin 2012

Sommaire

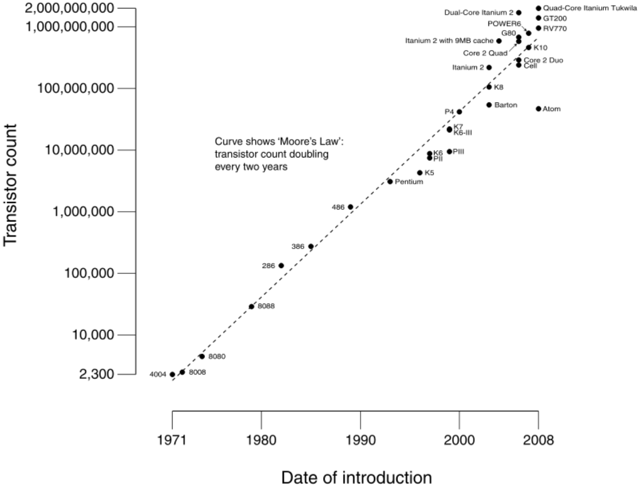
Sommaire

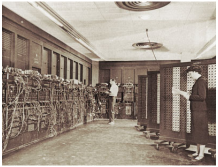
Du tube à vide au transistor

- 1834 La machine analytique de Babbage : calculateur à vapeur
- 1941 Zuse Z3 (1941) : relais électromécanique
- 1943 Colossus (1943) : tubes à vide
- 1944 Mark I (1944) : relais électromécanique
- 1946 ENIAC (1946) : tubes à vide
- 1947 Invention du transistor
- 1958 Invention du circuit intégré

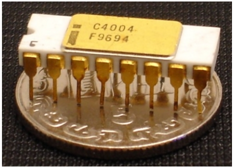
tube à vide

Du tube à vide au transistor





ENIAC (1946)



Intel 4004 (1971)

À quoi servent les ordinateurs ?

Au début de l'ère informatique :

- conçus comme un outil scientifique :
  - Colossus $\mapsto$ Déchiffrement des codes allemands.
  - Mark I $\mapsto$ Calcul des tables de trajectoire pour l'artillerie.
  - ENIAC $\mapsto$ Simulations d'essais nucléaires.
- leurs avantages :
  - plus rapides que les humains
  - se trompent plus rarement que les humains (en général)
  - ne se plaignent pas des taches répétitives

De nos jours :

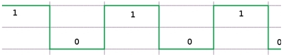
- Très utilisés comme outils scientifiques et d'ingénierie.
- Utilisés pour l'organisation et le stockage des données.
- Utilisés comme un outil de divertissement (Jeux, Vidéo, Son ...).
- Partage des données grâce aux réseaux (Réseaux sociaux, Échange de données ...).

Sommaire

Les unités d'information

L'unité fondamentale : le bit

Bit : contraction de « binary digit ». Ne prend que deux états possibles : 0 ou 1.



suite de 6 bits

0	0
0	1
1	0
1	1

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Avec n bits, on peut coder 2^n états différents.

Comment représente-t-on les entiers ?

- Avec 16 bits : $2^{16} = 65536$ valeurs, de 0 à 65535.
- Avec 32 bits : $2^{32} = 4294967296$ valeurs, de 0 à 4294967295.
- Avec 64 bits : $2^{64} = 18446744073709551616$ valeurs, de 0 à 18446744073709551615.

Représentation des réels

Qu'en est-il pour des valeurs telles que :

$$\frac{1}{2} = 0,5 \quad ✓$$

$$\frac{1}{3} = 0,3333333333333333333333333333333333333333333333\ldots \quad ✗$$

$$\sqrt{2} = 1,4142135623730950488016887242096980785696718753\ldots \quad ✗$$

$$\pi = 3,1415926535897932384626433832795028841971693993\ldots \quad ✗$$

$$e = 2,7182818284590452353602874713526624977572470936\ldots \quad ✗$$

Représentation des réels

- Sur certaines machines Cray, on avait :

$$\text{pour certaines entrées } x, \quad 1 \times x \quad \rightsquigarrow \quad \text{overflow (dépassement de capacité)}$$
$$x + y \neq y + x$$
$$0.5 \times x \neq x/2.0$$

- Sur IBM 370 (1970), en Fortran, on avait :

$$\sqrt{-4} = 2$$

Représentation des réels

La norme IEEE-754 (1985)

Les nombres réels sont codés par les nombres flottants qui contiennent :

- un bit s donne le signe (0 pour positif et 1 pour négatif).
- une mantisse m de précision p ($p = 24$ bits).
- un exposant e de n chiffres ($n = 8$ bits).

Exemple en décimal, avec $p = 6$ et $n = 2$:

- $s = 1$
- $m = 4,58745$ avec une précision $p = 6$
- $e = 2$

$$\text{est } -4,58745 \cdot 10^2 = -458,745$$

L'arrondi

Pendant les calculs, des résultats intermédiaires peuvent ne pas rentrer dans le format flottant. Il faut alors arrondir. Exemple avec $4,587458$:

- $\circ(4,587458) = 4,58746$ pour l'arrondi au plus proche.
- $\circ(4,587458) = 4,58745$ pour l'arrondi vers 0 (troncature).

Missile Patriote (25 février 1991)

- Compteur dans la batterie du missile : ajout de $1/10$ tous les dixièmes de seconde
  - $\hookrightarrow$ $1/10 \rightsquigarrow$ non représentable exactement en machine
  - $\hookrightarrow$ $1/10 \approx (0.0001100110011001100110011001100110011001\ldots)_2$
  - $\hookrightarrow$ erreur (24 bits) $\approx 9.5 \times 10^{-8}$ par ajout de $1/10$
  - $\hookrightarrow$ au bout de 100h : erreur ≈ 0.34 secondes

- Vitesse du missile Scud : 1676 m/s
  - $\hookrightarrow$ au bout de 100h : erreur ≈ 568 m
- Échec lors de l'interception d'un Scud (Dharan, Arabie Saoudite)
  - $\hookrightarrow$ bilan : 28 morts / 100 blessés
- Source du bug : accumulation d'erreurs dues à la troncature de $1/10$.

Premier vol d'Ariane 5 (4 juin 1996)


- Après 39 sec. de vol : autodestruction de la fusée
  - ↪ coût de la fusée / du cargot ≈ 500 millions $
  - ↪ coût du développement ≈ 7 milliard $
- Système de Référence Inertielle (SRI) : calcule la position, la vitesse et l'inclinaison de la fusée, en fonction de mesures d'accélération et de rotation
  - ↪ identique à celui d'Ariane 4
  - ↪ accélération 5 fois plus élevée
- Forte accélération de la fusée ⇝ dépassement de capacité lors du calcul des position et vitesse
  - ↪ dû à la conversion d'un nombre virgule flottante 64 bits (`double`) en nombre entier de 16 bits dans un logiciel en Ada
- Source du bug : l'emplacement mémoire alloué n'était pas suffisant

Sommaire

Peut-on faire confiance aux flottants ?

Les propriétés « classiques » des réels sont-elles vraies dans le monde des flottants ?

- L'associativité ?
- La distributivité ?
- Zéro unique ?

$$x = 1,1111 \ , \ y = 1,0004 \cdot 10^{-1} = 0,10004 \text{ et } z = 1,0002 \cdot 10^{-1} = 0,10002.$$

	1,1111	x
+	0,10004	y
	1,21114	$(x+y)$
arrondi à	1,2111	$\circ(x+y)$
+	0,10002	z
	1,31112	$((x+y)+z)$
arrondi à	1,3111	$\circ((x+y)+z)$

	0,10004	y
+	0,10002	z
	0,20006	$(y+z)$
+	1,1111	x
	1,31116	$((y+z)+x)$
arrondi à	1,3112	$\circ((y+z)+x)$

Peut-on faire confiance aux flottants ?

Les propriétés « classiques » des réels sont-elles vraies dans le monde des flottants ?

- ~~L'associativité ?~~
- La distributivité ?
- Zéro unique ?

$x = 9,9999 \cdots 10^{99}$ le plus grand flottant normal représentable et $y = 1,0000 \cdot 10^{1}$.

$$\begin{array}{llr} & 9,9999 \cdots 10^{99} & x \\ - & 9,9999 \cdots 10^{99} & x \\ \hline & 0 & (x-x) \\ \times & 10 & y \\ \hline & 0 & (x-x) \cdot y \end{array}$$

$$\begin{array}{llr} & 9,9999 \cdots 10^{99} & x \\ \times & 10 & y \\ \hline & +\infty & (x \cdot y) \end{array}$$

$$\begin{array}{llr} & +\infty & (x \cdot y) \\ - & +\infty & (x \cdot y) \\ \hline & NaN & (x \cdot y) - (x \cdot y) \end{array}$$

Peut-on faire confiance aux flottants ?

Les propriétés « classiques » des réels sont-elles vraies dans le monde des flottants ?

- ~~L'associativité ?~~
- ~~La distributivité ?~~
- Zéro unique ?

$x = 1,2345$ et $y = 2,0000 \cdot 10^{-5} = 0,00002$.

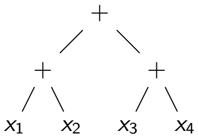
$$\begin{array}{lll} & 1,2345 & x \\ + & 0,00002 & y \\ \hline & 1,23452 & (x+y) \\ \text{arrondi à} & 1,2345 & \circ(x+y) = x \end{array}$$

Peut-on faire confiance aux flottants ?

Les propriétés « classiques » des réels sont-elles vraies dans le monde des flottants ?

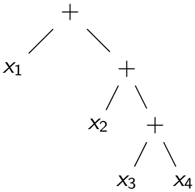
- ~~L'associativité ?~~
- ~~La distributivité ?~~
- ~~Zéro unique ?~~

L'expression $(x_1 + x_2 + x_3 + x_4)$ peut être évaluée de plusieurs façons.



$((x_1 + x_2) + (x_3 + x_4))$

- Schéma d'évaluation parallèle (rapide).
- En général moins précis.



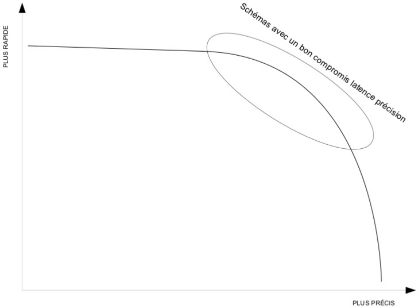
$(x_1 + (x_2 + (x_3 + x_4)))$

- Schéma d'évaluation séquentiel (lent).
- Précis en général.

Thévenoux, Langlois, Martel (2011)

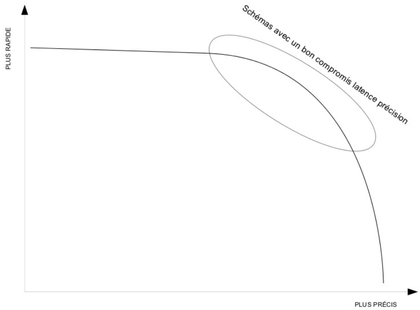
- Le schéma le plus rapide est en général moins précis.
- Le schéma le plus précis est en général plus lent.
- Il faut un compromis entre vitesse et précision.
  - Certains schémas presque optimaux (coté latence) ont une précision raisonnable.

CGPE



Compromis entre latence et précision

CGPE

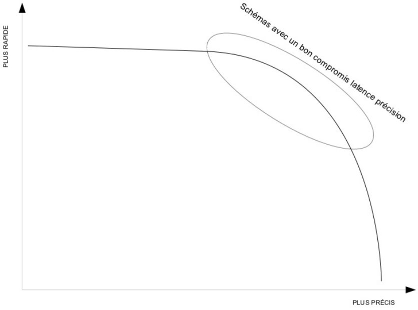


Compromis entre latence et précision

C'est quoi CGPE ?

- Un logiciel qui prend des expressions arithmétiques (des polynômes) et cherche les meilleurs schémas (latence/précision).
- A été écrit par mon encadrant de thèse : Guillaume Revy.
- Nous y travaillons encore avec Christophe Mouilleron (ATER à DALI).

CGPE



Compromis entre latence et précision

Nos objectifs ?

- Étendre les fonctionnalités de CGPE.
- Faire de CGPE un générateur efficace de code pour des architectures différentes.

Merci de votre attention

Des questions ?