Tema 1. Rendiment i consum

Estructura de Computadors (EC)

Rubèn Tous

rtous@ac.upc.edu
Computer Architecture Department
Universitat Politecnica de Catalunya

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Índex

Índex

(Aclariment)

Aclariment

El Tema 1 inclou la introducció a l'assignatura (secció 1.1) i també aspectes de rendiment i consum (seccions 1.2, 1.3 i 1.4). Per motius de calendari les parts de rendiment i consum s'expliquen ara, després del Tema 4.

1.2.1 Conceptes i mètriques

- **Temps d'execució**: temps transcorregut entre l'inici i el final d'una **única** tasca.
  - Exemple: temps de resposta d'una app.
- **Productivitat (throughput)**: nombre de tasques completades per unitat de temps.
  - Exemple: jobs per hour en un centre de càlcul.
- Aquí ens centrarem en el temps d'execució.

1.2.1 Conceptes i mètriques

- Temps d'execució (t_{exe}): temps de CPU (t_{cpu})+ temps d'E/S + temps altres programes.
- Per simplificar suposarem que $t_{exe} = t_{cpu}$

1.2.1 Conceptes i mètriques

- Rendiment relatiu o **guany** = $s = \frac{t_{original}}{t_{millorat}}$
- Exemple: després d'haver introduït una millora un programa que trigava 3 segons ara en triga 2. **guany** = $s = \frac{3}{2} = 1.5$ vegades més ràpid.
- No té unitats.

1.2.2 Càlcul del temps d'execució (de CPU)

- Temps d'execució d'un programa (t_{exe}).
- $t_{exe} = n_{cicles} * t_c$
- Número total de cicles de CPU = n_{cicles}.
- Temps de cicle = $t_c = \frac{1}{f_{clock}}$
- Per exemples si $f_{clock} = 2GHz = 2 * 10^9$ aleshores $t_c = \frac{1}{2*10^9} = 0.5 * 10-9 = 0.5$ ns.

1.2.2 Càlcul del temps d'execució (de CPU)

- Com determinar el n_{cicles}?
- El programa té I instruccions.
- Si suposem que, de mitjana, cada instrucció triga CPI cicles (Cicles Per instrucció): $n_{cicles} = I * CPI$.
- Però cada tipus d'instrucció triga un nombre de cicles diferent (CPI_i).
- $n_{cicles} = I_A * CPI_A + I_B * CPI_B + ...$

1.2.2 Càlcul del temps d'execució (de CPU)

- Per exemple:

Tipus instrucció	I	CPI_i
A	$2 * 10^9$	1
B	$1 * 10^9$	2
C	$2 * 10^9$	3

Exemple dades d'un programa.

- $n_{cicles} = I_A * CPI_A + I_B * CPI_B + I_C * CPI_C = (2 * 1 + 1 * 2 + 2 * 3) * 10^9 = 10 * 10^9 cicles$

1.2.2 Càlcul del temps d'execució (de CPU)

- Si suposem que $f_{clock} = 2GHz$ i per tant $t_c = 0.5 * 10^{-9}s$.
- $t_{exe} = n_{cicles} * t_c = 10 * 10^9 * 0.5 * 10^{-9} = 5s$

1.2.2 Càlcul del temps d'execució (de CPU)

- De vegades, en comptes del CPI_i de cada tipus d'instrucció ens poden donar el CPI mitjà de qualsevol instrucció dins un programa.
- O ens el poden fer calcular.
- A l'exemple seria: $CPI = \frac{n_{cicles}}{I} = \frac{I_A * CPI_A + I_B * CPI_B + I_C * CPI_C}{I_A + I_B + I_C} = \frac{(2*1+1*2+2*3)*10^9}{(2+1+2)*10^9} = \frac{10*10^9}{5*10^9} = 2cicles$.

1.2.2 Càlcul del temps d'execució (de CPU)

- Si disposem del CPI global tenim que:
- $n_{cicles} = I * CPI$
- I per tant podem calcular t_{exe} així:
  $t_{exe} = I * CPI * t_c = 5 * 10^9 * 2 * 0.5 * 10^{-9} = 5s$

1.2.2 Càlcul del temps d'execució (de CPU)

- Aprofitem l'exemple per fer un càlcul de guany (speedup).
- Quin seria el guany si reduïm el CPI_C del programa a 2?
  $n_{cicles} = (2 * 1 + 1 * 2 + 2 * 2) * 10^9 = 8 * 10^9 cicles$
  $t_{exe} = n_{cicles} * t_c = 8 * 10^9 * 0.5 * 10^{-9} = 4s$
  $s = \frac{t_{original}}{t_{millorat}} = 5/4 = 1.25$

1.2.3 Núm de cicles vs freqüència de rellotge (temps de cicle)

- Les operacions hardware necessiten un temps mínim (e.g. llegir de memòria).
- Per exemple, imaginem unes instruccions de tipus A que necessiten 0.8ns

```
|----------------| 0.8ns
```

- Una instrucció triga un nombre **enter** de cicles.
- Per exemple, si $t_c = 1ns$ les instruccions de tipus A trigaran 1 cicle ($CPI_A = 1$ cicle):

```
|----------------| 1 cicle = 1ns
```

1.2.3 Núm de cicles vs freqüència de rellotge (temps de cicle)

- Reduir el temps de cicle implica incrementar el CPI dels tipus d'instruccions les operacions hardware de les quals estaven al límit del temps de cicle antic.
- Això pot resultar contraproduent (dependrà de la proporció d'instruccions de cada tipus).
- Per exemple, si reduïm el temps de cicle a 0.6ns les nostres instruccions de tipus A trigaran més ja que necessitaran 2 cicles:

```
|--------------|--------------| 2 cicles = 1.2ns
```

Índex

1.3 Llei d'Amdahl

- Suposem una tasca que triga $t_{original}$ segons (e.g. un programa).
- Suposem que hi ha una **una part** d'aquesta tasca (e.g. el CPI_i d'un tipus d'instruccions), que ara triga t_x.
- Anomenarem P_x a la fracció que representa aquesta part ($P_x = \frac{t_x}{t_{original}}$).
- Quin és el màxim guany (speedup) que es pot aconseguir si només millorem aquesta part que ocupa una fracció P_x del temps?

1.3 Llei d'Amdahl

- Si millorem S_x vegades una part P_x aleshores $t_{millorat} = \frac{t_{original} * P_x}{Sx} + t_{original} * (1 - P_x)$.
- $s = \frac{t_{original}}{t_{millorat}} = \frac{t_{original}}{\frac{t_{original} * P_x}{Sx} + t_{original} * (1 - P_x)} = \frac{1}{\frac{P_x}{S_x} + (1 - P_x)}$

Llei d'Amdahl 1/2

Si millorem S_x vegades una part P_x aleshores el guany global és $s = \frac{1}{\frac{P_x}{S_x} + (1 - P_x)}$

1.3 Llei d'Amdahl

- Exemple: Un programa 90% paral·lelitzable en 3 ordinadors: $s = \frac{1}{\frac{0.9}{3}+(1-0.9)} = \frac{1}{0.3+0.1} = \frac{1}{0.4} = 2.5$.

1.3 Llei d'Amdahl

- I si millorem al màxim ($S_x \to \infty$)?
- $\lim_{S_x \to \infty} \frac{1}{\frac{P_x}{S_x} + (1 - P_x)} = \frac{1}{1 - P_x}$

Llei d'Amdahl 2/2

El màxim que podem millorar una tasca si només millorem una fracció P_x és $s = \frac{1}{1 - P_x}$

1.3 Llei d'Amdahl

- Exemple: El programa 90% paral·lelitzable només pot millorar: $\frac{1}{1-0.9}\,\textit{vegades} = \frac{1}{0.1} = 10$ vegades.

1.3 Llei d'Amdahl

- Conclusió: Cal concentrar els esforços d'optimització en aquelles parts del programa (o circuit) que comporten el major cost en temps.

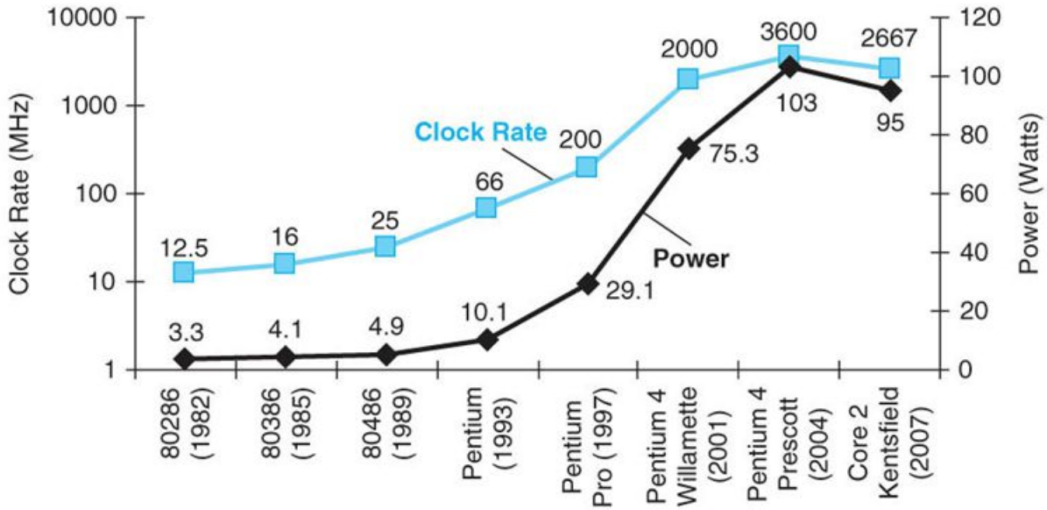
Índex

1.4 Mesures de consum

- El 1965 Gordon Moore (co-fundador d'Intel) va predir que la densitat de transistors es podria duplicar cada 2 anys (a la pràctica, cada 18 mesos).
- Grandària$\Downarrow \rightarrow$ temps de conmutació dels transistors$\Downarrow \rightarrow$ podem freq. de rellotge$\Uparrow$.

1.4 Mesures de consum

- Veurem que freq. de rellotge$\Uparrow \rightarrow$ consum$\Uparrow$.
- L'increment del consum imposa un límit a la millora del rendiment d'un únic CPU (Power Wall).



1.4.1 Càlcul del consum dinàmic

- P = Potència = consum d'energia per unitat de temps, en watts.
- El consum d'energia dinàmic d'un circuit és l'originat pels corrents de càrrega i descàrrega de les capacitats equivalents dels transistors que tenen lloc en les commutacions.
- No tots els transistors d'un processador commuten en cada flanc de rellotge. $f_{commut} = f_{clock} * \alpha$, on α és el factor d'activitat.

1.4.1 Càlcul del consum dinàmic

P (potència)

El consum d'energia dinàmic d'un circuit construït amb transistors de la familia CMOS és $P = C * V^2 * f_{commut}$

- P (potència) = watts = joules per segon.
- C (capacitància agregada de tots els transistors que commuten) = farads (F).
- V (voltatge) = volts (V).

1.4.1 Càlcul del consum dinàmic

- D'on surt aquesta fòrmula?
- El corrent generat per les commutacions és $I = C * V * f_{commut}$.
- $P = I * V \rightarrow P = C * V^2 * f_{commut}$

1.4.1 Càlcul del consum dinàmic

Exemple (examen parcial 2016/2017 Q2):

- El processador d'un telèfon mòbil disposa de 4 tipus d'instruccions (A, B, C, D), amb diferents CPI.
- El processador dissipa una potència dinàmica de 16W i la potència estàtica es considera nul·la (0W).
- La seva bateria està plena i pot emmagatzemar 500J.

1.4.1 Càlcul del consum dinàmic

- Executem un programa de test que triga 30 segons:

Tipus d'instrucció	nombre d'instruccions	CPI
A	$6 * 10^9$	1
B	$2 * 10^9$	2
C	$4 * 10^9$	5
D	$3 * 10^9$	4

1.4.1 Càlcul del consum dinàmic

- Quina és la freqüència de rellotge del processador?
  $n_{cicles} = (6 * 1 + 2 * 2 + 4 * 5 + 3 * 4) * 10^9 =$
  $(6 + 4 + 20 + 12) * 10^9 = 42 * 10^9$
  $t_{exe} = 30s = 42 * 10^9 * \frac{1}{f}$
  $f = \frac{42}{30} * 10^9 = 1,4GHz$
- Durant quant de temps més podrem tornar a executar el programa de test amb l'energia que queda a la bateria?
  $E = P * t \rightarrow E = 16 * 30 = 480J \rightarrow$ queden
  $500 - 480 = 20J$
  $20 = P * t \rightarrow 20 = 16 * t \rightarrow t = 20/16 = 1,25s$

1.4.1 Càlcul del consum dinàmic

- Si augmentem la freqüència del processador a 2 GHz però les instruccions de tipus B tenen un CPI=6 calcula:
- El temps d'execució en segons.
  $n_{cicles} = (6 + 12 + 20 + 12) * 10^9 = 50 * 10^9$
  $t_{exe} = 50 * 10^9 * 0,5 * 10^{-9} = 25s$
- El guany de rendiment.
  speedup $= \frac{30}{25} = 1,2$
- La potència dissipada en Watts.
  $P_{old} = C * V^2 * f \rightarrow 16 = C * V^2 * 1,4 * 10^9$
  $C * V^2 = \frac{16}{1,4} * 10^{-9} = 11,43 * 10^{-9}$
  $P = 11,43 * 10^{-9} * 2 * 10^9 = 22,86W$

1.4.1 Càlcul del consum dinàmic

Tendències històriques derivades de la miniaturització:

- Transistors petits $\rightarrow$ $C \downarrow$. Però però n'hi ha molts més: la capacitat agregada C no ha variat significativament.
- Temps de commutació $\downarrow \rightarrow$ $f \uparrow \rightarrow$ $P \uparrow$:-(.
- Per compensar l'augment de consum: $V \downarrow$ (de 5 a 1V).
- $V \downarrow \rightarrow$ V_t (tensió umbral del transistor) $\downarrow$ (de 0,7 a 0,4V). Això implica un problema de **consum estàtic**.

1.4.2 Consum estàtic

- Pels transistors en circuit obert hi circula de manera constant un petit corrent de fuga (I_{leak}).
- I_{leak} no depèn de l'activitat del processador, i augmenta exponencialment a mesura que disminueix la tensió umbral (V_t).
- Per a $V_t = 0{,}2$V el consum estàtic podria arribar a ser el 50% del total!

1.4.2 Consum estàtic

Tècniques de reducció del consum (estàtic i dinàmic):

- clock gating: Inhabilitar la conmutació del senyal de rellotge de determinats circuits.
- dynamic voltage and frequency scaling (DVFS): Reduir V i f.
- power gating (e.g. nuclis del processador, bancs de la cache, etc.): Reduir el consum dinàmic i estàtic.
- high-k materials: Reduir I_{leak} usant aïllants amb millor constant dielèctrica.