Valutazione delle prestazioni di calcolo

A. Borghese, F. Pedersini

Dipartimento di Informatica
Università degli Studi di Milano

Perché valutare le prestazioni?

PERCHÉ?

- Per poter fare misure **quantitative** delle prestazioni di un sistema di elaborazione
  - Misura di velocità / throughput
  - Fatturazione delle prestazioni
- Per poter fare scelte "intelligenti"
  - *È più vantaggioso acquistare nuovo hardware o modificare il software?*
  - *Analisi costi/benefici relativi all'acquisto di nuovo hardware*

La misura di prestazioni riguarda di solito il **tempo di calcolo**

$Prestazioni_X > Prestazioni_Y \rightarrow T_X < T_Y \rightarrow 1/T_X > 1/T_Y$

*"**X** ha prestazioni maggiori di **Y di un fattore** T_Y / T_X"*

Misura relativa (percentuale): *"X ha prestazioni **maggiori di** Y di **N %**"*

$Prestazioni(X) = (1 + N/100) \cdot Prestazioni(Y)$

$T_Y = (1 + N/100) \cdot T_X \rightarrow N = 100 \cdot (T_Y - T_X) / T_X$

Criteri (metriche) di valutazione orientati all'utente

Parametri di prestazione tipici:

- *Velocità di esecuzione*
- *Quantità di informazione elaborata*

- Il criterio di valutazione dipende dall'utilizzo del calcolatore:
  - Utilizzo personale → **tempo di esecuzione**
  - Utilizzo come server → **throughput**

  THROUGHPUT: quantità di lavoro svolto in un dato tempo (accessi pagine web, a banche dati, ad applicazioni, transazioni commerciali…)

- In casi particolari, si utilizzano anche altri criteri:
  - **Efficienza energetica:** **Joule** / operazione (es: *sonda spaziale*)
  - **Ingombro in volume:** (operazioni / sec) / $\mathbf{m^3}$ (es: *data center*)

Criteri di valutazione orientati alla macchina

Misure di prestazione ad alto livello (livello utente):

- **Tempo di risposta**: tempo totale per completare un lavoro
  - includendo accessi a disco, accessi a memoria, attività di I/O…

- **Tempo di CPU**: rappresenta il tempo speso dalla CPU per eseguire il programma dato
  - Non include i tempi di attesa per I/O o esecuzione di altri programmi
  - Include:
    - **tempo di CPU utente:** tempo speso dalla CPU per eseguire le linee di codice che stanno nel nostro programma
    - **tempo di CPU di sistema:** tempo speso dal sistema operativo per eseguire i compiti richiesti dal programma

- Comando `time` (UNIX):

  Produce tutti i tempi relativi al lavoro indicato

  Sintassi: `time <job>` → `90.7u 12.9s 2:39 65%`

Criteri di valutazione orientati alla macchina

Misure di prestazione a basso livello (livello istruzioni macchina)

Tempo di CPU (T_{CPU}):

$$T_{CPU} = N\_cicli\_clock \times T_{CLOCK} = N\_cicli\_clock / f_{CLOCK}$$

CPI: Clock **P**er **I**nstruction

- È il numero **medio** di **cicli di clock / istruzione**:

  $$CPI = N\_cicli\_clock(programma) / N\_istruzioni(programma)$$

- T_{CPU} e **CPI** sono quindi grandezze legate dall'espressione:

  $$T_{CPU} = CPI \times N\_Istruzioni \times T_{clock}$$

- Esempio: *Calcolare il CPI di un programma di 400.000 istruzioni, che necessita per la sua esecuzione (T_{CPU}) 1,2 sec, su un elaboratore con f_{CLOCK} = 1 MHz*

  Per l'esecuzione del programma, occorrono:

  - N_cicli_clock = 10^6 cicli/sec × 1,2 sec = $1{,}2 \cdot 10^6$ cicli
  - **CPI** = N_cicli / N_istruzioni = $1{,}2 \cdot 10^6$ cicli / $4 \cdot 10^5$ istr. = **3 cicli/istruzione**

Sulle macchine di oggi il **CPI è inferiore a 1** (arch. **superscalari**)

Misura delle prestazioni : $T_{medio/istruzione}$

Misure di prestazione a basso livello *(livello istruzioni macchina)*

$T_{medio/istruzione}$ $(= CPI \times T_{Clock})$

- in genere, istruzioni di tipo diverso richiedono quantità diverse di tempo
  - la moltiplicazione richiede più tempo dell'addizione
  - l'accesso alla memoria richiede più tempo dell'accesso ai registri

$T_{medio/istruzione}$ è il tempo medio di esecuzione del set di istruzioni di un'architettura, pesato secondo la frequenza di utilizzo di ogni istruzione.

Sommatoria delle durate dei diversi tipi di istruzione

$$T_{medio/istruzione} = \frac{T_{tot}}{\#\,Istruzioni} = \frac{\sum_{i=0}^{S} n_i t_i}{\sum_{i=0}^{S} n_i}$$

n_i : numero di volte che l'istruzione i viene eseguita nel programma

t_i : tempo di esecuzione dell'istruzione i

Misura delle prestazioni: CPI_{medio}, t_{medio}

CPI_i numero di cicli di clock necessari all'istruzione di tipo i

n_i numero di volte che l'istruzione i viene eseguita nel programma.

f_i frequenza con cui l'istruzione i viene eseguita nel programma.

$$T_{medio/istr.} = \frac{\sum_{i=0}^{S} n_i t_i}{\sum_{i=0}^{S} n_i} = \sum_{i=1}^{n} (t_i \cdot f_i) \qquad CPI_{medio} = \frac{\sum_{i=1}^{n} (CPI_i * n_i)}{\sum_{i=1}^{n} n_i} = \sum_{i=1}^{n} (CPI_i \cdot f_i)$$

($\sum_{i=1}^{n} n_i$ numero totale di istruzioni del programma)

$$T_{CPU} = t_{medio} \times N\_istruzioni = CPI_{medio} \times T_{clock} \times N\_istruzioni$$

$$T_{CPU} = \sum_{i=1}^{n} (t_i \cdot n_i) = t_{medio} \sum_{i=1}^{n} n_i = T_{clock} \sum_{i=1}^{n} (CPI_i \cdot n_i) = T_{clock} \cdot CPI_{medio} \sum_{i=1}^{n} n_i = T_{clock} \cdot CPI_{medio} \cdot N_{istruzioni}$$

Esempio: calcolo CPI_m, T_{CPU}

Esempio: *si consideri un calcolatore in grado di eseguire le istruzioni riportate in tabella.*

Calcolare:

CPI medio *e il* ***tempo di CPU*** *per eseguire un programma composto da* ***200 istruzioni****, assumendo una frequenza di clock pari a* ***500 MHz****.*

istruzione	frequenza	CPI
`ALU`	43%	1
`load`	21%	4
`store`	12%	4
`branch`	12%	2
`jump`	12%	2

$CPI_{medio} = 0{,}43 \times 1 + 0{,}21 \times 4 + 0{,}12 \times 4 + 0{,}12 \times 2 + 0{,}12 \times 2 =$
$= \mathbf{2{,}23}$

$T_{CPU} = N_{istruzioni} \times CPI_{medio} \times T_{clock} =$
$= 200 \times 2{,}23 \times 2 \text{ nsec} = \mathbf{892 \text{ nsec}}$

$t_{clock} * CPI = t_{medio}$

Indici di prestazione

- ***MIPS: Million Istructions Per Second***

  MIPS = (N_istruzioni / 10^6) / T_{CPU}

  MIPS = f_{CLOCK} [espressa in MHz] **/ CPI**

Problemi di misura con MIPS:

- dipende dall'insieme di istruzioni, quindi è difficile confrontare architetture con diversi set di istruzioni;
- varia a seconda del programma considerato;
- può variare in modo inversamente proporzionale alle prestazioni!

MIPS relativo = $T_{CPU} / T_{CPU\_REF} \times MIPS_{CPU\_REF}$

CPUref : VAX–11/780 di **Digital**

Problema – macchine con hardware di calcolo in virgola mobile:

- Le istruzioni in virgola mobile richiedono più cicli di clock di quelle su interi
- Hardware dedicato per la virgola mobile (in luogo delle routine software) impiegano meno tempo pur avendo un MIPS più basso

MFLOPS: Million FLoating-point Operations Per Second
introdotto per i **supercomputer**

- Problema: misure di picco.
- MIPS di picco e sostenuti. Problema: poco significative.

BENCHMARKS = Programmi per valutare le prestazioni

- Es: Whetstone, 1976; Drystone, 1984. Kernel benchmark. Loop Livermore, Linpack, 1980.
- Problema: polarizzazione del risultato.
- Benchmark con programmi piccoli (10-100 linee, 1980) ➔ non adatti a misurare le prestazioni delle strutture gerarchiche di memoria.

Benchmarks: indici SPEC

PRINCIPIO di MISURA: **insieme di programmi di test**

Misura in più condizioni diverse (singolo / multiplo processore / time sharing...)

Benchmark specifici per valutare anche S.O. e I/O.

- **SPEC '95 → SPECint, SPECfp**
  - base: Sun SPARCstation 10/40

Benchmark particolari:

- **SDM** (Systems Development Multitasking).
- **SFS** (System-level File Server).
- **SPEChpc96** Elaborazioni scientifiche ad alto livello

SPEC '95 – Test prestazioni elaborazione intera:

1)	`Go`	Intelligenza artificiale
2)	`m88ksim`	Simulatore chip Motorola 88K
3)	`gcc`	Compilatore Gnu C che genera codice SPARC.
4)	`compress`	Compressione e decompressione di un file in memoria.
5)	`li`	Interprete LISP
6)	`ijpeg`	Compressione e decompressione immagini
7)	`perl`	Manipolazione di stringhe e numeri primi (in PERL)
8)	`vortex`	Gestione di una base di dati.

SPEC '95 – Test prestazioni elaborazione in virgola mobile

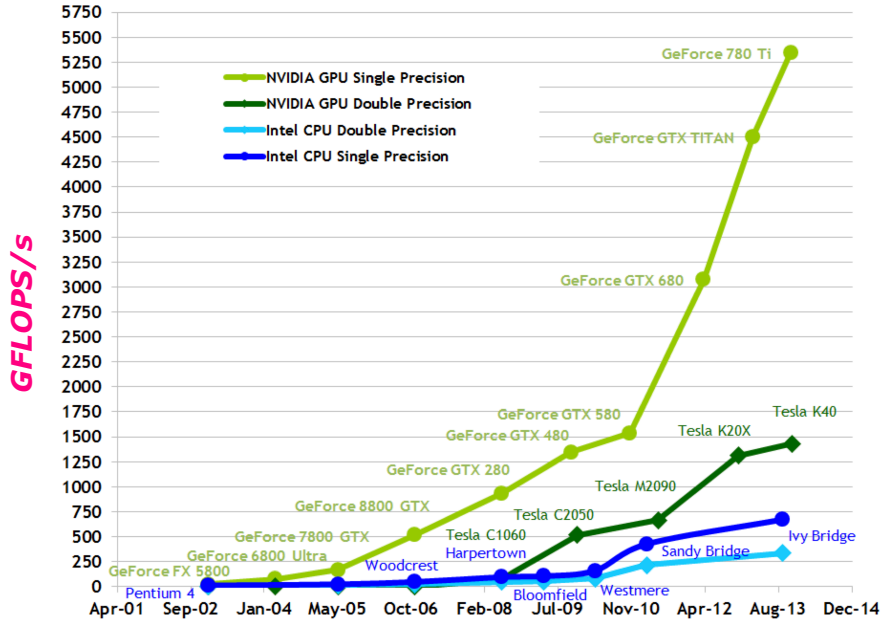
1)	`Tomcatv`	Programma per generazione di griglie
2)	`Swim`	Modello per acqua poco profonda con griglia 513 x 513
3)	`Su2cor`	Fisica quantistica: simulazione MonteCarlo
4)	`Hydro2D`	Simulazione sistemi fluidodinamici con equazioni di Navier-Stokes
5)	`Mgrid`	Risolutore multi-griglia in campo di potenziale 3D
6)	`Applu`	Equazioni alle differenze parziali paraboliche/ellittiche
7)	`Turb3D`	Simulazione di turbolenza isotropica ed omogenea in un cubo
8)	`Apsi`	Simulatore meteorologico, per il calcolo della diffusione di agenti inquinanti
9)	`Fpppp`	Chimica quantistica
10)	`Wave5`	Fisica dei plasmi: simulazione di particelle in campi elettromagnetici

SPEC 2000: programmi simili, di utilizzo comune (**`gzip`**, compilaz. C, ...)

Prestazioni **CPU** vs. **GPU**:

calcolo
floating-point

(da: NVIDIA, 2014)



- Fare scelte "intelligenti"
  - Installare nuovo hardware o nuovo software?
  - Analisi costi/benefici relativi all'acquisto di nuovo hardware

Valutazione quantitativa delle prestazioni:

legge di Amdahl

Considero 2 macchine con prestazioni differenti:

	Calcolatore I	**Calcolatore II**
Durata istruzione A	*1*	*10*
Durata istruzione B	*1000*	*100*
Durata istruzione C	*10*	*100*

$$t_{medio} = \frac{\sum_{i=1}^{n} n_i \cdot t_i}{\sum_{i=1}^{n} n_i}$$

Quale delle due macchine è più veloce?

Dipende da quanto spesso uso le diverse istruzioni.

Tempo **medio** di esecuzione di un'istruzione:

- **Programma 1: 1000 istruzioni A, 1 istruzione B, 10 istruzioni C.**
  $t_I = 1/1011 * (1000*1 + 1*1000 + 10*10) = 2100/1011 \approx 2$
  $t_{II} = 1/1011 * (1000*10 + 1*100 + 10*100) = 11100/1011 \approx 11$

- **Programma 2: 100 istruzioni A, 10 istruzioni B, 10 istruzioni C.**
  $t_I = 1/120 * (100*1 + 10*1000 + 10*10) = 10200/120 \approx 100$
  $t_{II} = 1/120 * (100*10 + 10*100 + 10*100) = 3000/120 \approx 30$

Criteri di ottimizzazione di calcolatori

PRINCIPIO: **rendere veloce il caso più frequente**.

- Favorendo (velocizzando) il caso più frequente (a discapito del più raro), la velocità media aumenta.

Definizioni:

f_m : *frazione*$_{miglioramento}$ $(0 \le f_m \le 1)$

- la frazione del tempo di calcolo della macchina originale che può essere modificato per avvantaggiarsi dei miglioramenti.

s_m : *speed-up*$_{miglioramento}$ $(s_m \ge 1)$

- il fattore di aumento di velocità dovuto al miglioramento, rispetto alla velocità originale.

Speed-up – esempio 1

Consideriamo un calcolatore (**Comp1**) …

…e un secondo calcolatore (**Comp2**) identico a Comp1, dove però la **ALU è stata velocizzata di un fattore 2x**

Consideriamo un'applicazione che prevede **90%** di istruzioni in aritmetica intera.

Di quanto è lo **speed-up globale di Comp2 rispetto a Comp1**?

Speed-up attività modificata: $s_M = 2$

Frazione di tempo attività modificata: $f_M = 0.9$ (90%)

$T_{Computer1}$: $T_1 = T_{ALU} + T_{RESTO}$; $T_{ALU} = 0.9\ T_1$, $T_{RESTO} = 0.1\ T_1$

$T_{Computer2}$: $T_2 = T_{ALU}/2 + T_{RESTO} = (0.9/2)\ T_1 + 0.1\ T_1 = 0.55\ T_1$

Speed-up globale: $S = T_1/T_2 = 1 / 0.55 = 1.82$

In generale:

$T_{Computer1}$: $T_1 = T_M + T_R$; $\boldsymbol{T_M = f_M \cdot T_1}$, $\boldsymbol{T_R = (1-f_M)\cdot T_1}$

$T_{Computer2}$: $\boldsymbol{T_2} = T_M/s_M + T_R = f_M/s_M \cdot T_1 + (1-f_M)\cdot T_1 = \boldsymbol{T_1\,[f_M/s_M + (1-f_M)]}$

Speed-up: $\boldsymbol{S = T_1/T_2 = 1 / [f_M/s_M + (1-f_M)]}$

Corollario della legge di Amdhal

- **Legge di Amdahl:**
  Il miglioramento delle prestazioni globali ottenuto con un miglioramento particolare (e.g. un'istruzione), dipende dalla frazione di tempo in cui il miglioramento viene eseguito.

  $$\boldsymbol{Speedup = 1 / [\,1 - f_m + f_m / s_m)\,]}$$

$$S_{globale} = \frac{1}{1 - f_M + \frac{f_M}{s_M}}$$

In generale, per N fattori di speed – up:

$$S_{globale} = \frac{1}{1 - \sum_{i=1}^{N} f_{M,i} + \sum_{i=1}^{N} \frac{f_{M,i}}{s_{M,i}}}$$

- **Corollario:**
  - *Se un miglioramento è utilizzabile solo per una frazione del tempo di esecuzione complessivo (f_m), allora non è possibile accelerare l'esecuzione più del **reciproco di** $(1 - f_m)$:*

    $$\boldsymbol{Speedup \le 1 / (1 - f_m)} \quad (\text{caso limite: } s_M \to \infty)$$

Dimostrazione

$$T_{new} = T_{nm} + T_m = T_{old} * (1 - f_m) + (T_{old} / s_m) * f_m$$

0.1 — 0.9 / 2

$$T_{new} = T_{old} * (1 - f_m + f_m / s_m) = T_{old} * [1 - f_m * (1 - 1/s_m)]$$

Istruzioni non accelerate — Istruzioni accelerate

$$Speedup_{globale} = T_{old}/T_{new} = T_{old} / T_{old} * [1 - f_m * (1 - 1 / s_m)] =$$

$$1/ [1 - f_m + f_m / s_m] < 1 / [1 - f_m] \quad \text{c.v.d.} \quad (s_m \to \infty)$$

Istruzioni non accelerate

Corollario:
Se il tempo di esecuzione delle istruzioni accelerate aumentasse all'infinito, il tempo di esecuzione T_{NEW} coinciderebbe con il tempo di esecuzione delle istruzioni non accelerate: $T_{OLD}(1-f_m)$

Quindi: $S_{max} = T_{OLD} / T_{NEW} = 1 / (1-f_m)$

Esempio 2

Esempio:

- Si consideri un miglioramento che consente un funzionamento **10** volte più veloce rispetto alla macchina originaria, ma che sia utilizzabile solo per il **40%** del tempo.
- Qual è il guadagno complessivo che si ottiene incorporando detto miglioramento?

$Speedup_{globale} = 1 / (1 - f_m + f_m/S_m)$

- ***Frazione**$_{miglioramento}$* → $f_m = 0.4$
- ***Speedup**$_{miglioramento}$* → $S_m = 10$

Speedup$_{globale}$ = 1 / (1 – 0.4 + 0.4/10) = 1 / (1–0.4+0.04)
= 1 / 0.64 = **1.5625** (**56%** più veloce)

Speed-up dovuto a cache memory

*Esempio: velocizzazione dovuta alla **memoria cache***

Supponiamo che una cache sia **5** volte più veloce della memoria principale e che venga usata per il **90%** del tempo
(miss rate = 0.1)

- Qual'è il guadagno in velocità dovuto all'uso della cache?
  - $f_M = 0.9$
  - $s_M = 5$
- $Speedup_{globale} = 1/ (1 - f_{tempo\_cache} + f_{tempo\_cache}/ S_{cache}) =$
  $= 1 / (1 - 0.9 + 0.9/5) \approx 3.6$
- Velocità **3.6** volte superiore usando la cache

Esempio 3

Esempio: valutazione rapporto prestazioni/costo

- Supponiamo di dover scegliere tra un server con una CPU standard ed uno con una CPU 5 volte più veloce (senza influenzare le prestazioni di I/O), ad un costo (della CPU) 5 volte superiore.
- Da misurazioni si sa che che la CPU viene utilizzata per il 50% del tempo (il tempo rimanente è dedicato ad operazioni di I/O).
- Se la CPU è un terzo del costo totale del server, quale CPU rappresenta il miglior investimento, in termini di rapporto prestazioni/prezzo?

Incremento globale di prestazioni:

- $Speedup_{globale} = 1 / (1 - 0.5 + 0.5/5) = 1 / 0.6 = 1.67$

Incremento di costo del server = $2/3 + (1/3)*5 = 7/3 = 2.33$

- L'incremento di costo è quindi "più grande" del miglioramento di prestazioni
  - ➔ la modifica ***non*** migliora il rapporto costo/prestazioni

Esempio – speedup dovuto a vettorializzazione

Esempio: speed-up mediante vettorializzazione

- Si deve valutare un miglioramento di una macchina per l'aggiunta di una modalità vettoriale. La computazione vettoriale è 20 volte più veloce di quella normale.
  Si definisce *percentuale di vettorizzazione* la porzione di tempo di calcolo che può sfruttare la modalità vettoriale.

1. Disegnare un grafico che riporti lo speed-up in funzione della percentuale della computazione effettuata in modo vettoriale
  - Quale percentuale di vettorizzazione è necessaria per uno speed-up di 2?
  - Quale per raggiungere la metà dello speed-up massimo?

Si supponga ora che la percentuale di vettorizzazione misurata sia del 70%

- I progettisti hardware affermano di potere raddoppiare la velocità della parte vettoriale se vengono effettuati significativi investimenti.
- Il gruppo che si occupa dei compilatori può incrementare la percentuale d'uso della modalità vettoriale

2. Quale incremento della percentuale di vettorizzazione sarebbe necessario per ottenere lo stesso guadagno di prestazioni?

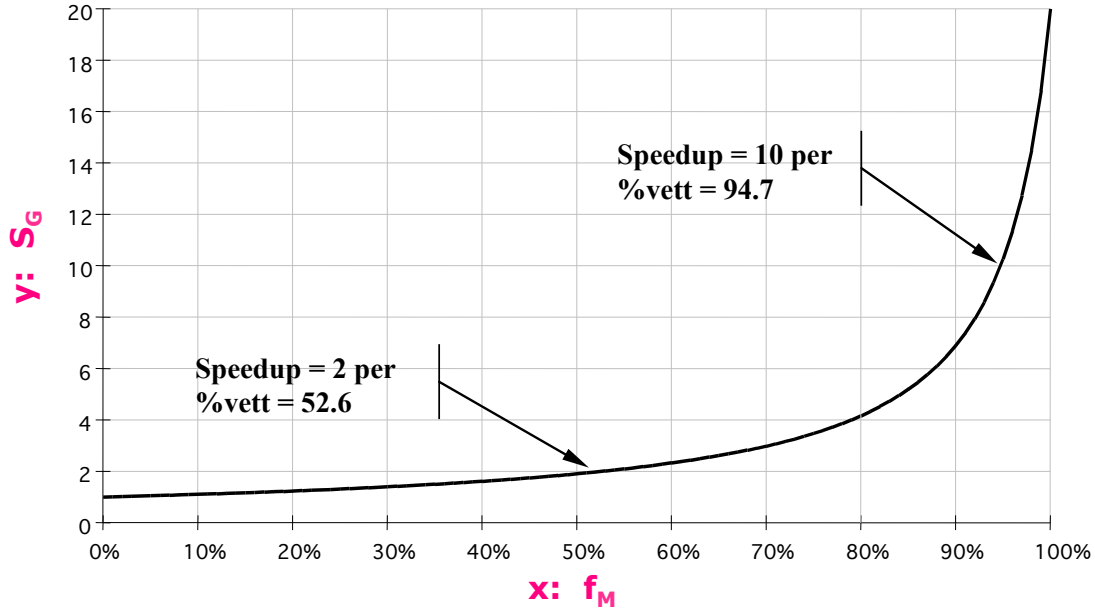
3. Quale investimento raccomandereste?

Esempio: curva di speed-up

Speed-up come funzione della frazione di vettorizzazione:

$S_M = 20;\quad f_M = x;$

$S_G \equiv S_G(f_M) = y(x) = 1/\,[1 - x + x/20] = 20/(20\text{-}19x)$; $0 \le x \le 1$



Esempio: speed-up dovuto a HW

- L'incremento di velocità con f_{VETT} = **70 %** …
  - $\text{Speedup}_G = 1/ [1 - 0.7 + 0.7 / 20)] =$
    $= 1/(1 - 0.7*19/20) =$ **2,9851**
- HW: raddoppiando la velocità dell'unità vettoriale:
  - $\text{Speedup}_{HW} = 1/ [1 - 0.7 + 0.7 / 40)] =$
    $= 1/(1 - 0.7*39/40) =$ **3,1496**
- SW: aumentando la percentuale di vettorizzazione, f_{VETT} …
  - $\text{Speedup}_{SW} = 3{,}1496 = 1/ [1 - x + x / 20)]$
    ➔ $x = f_{VETT}$ **= 71,84%**

➔ La soluzione SW è ragionevolmente più vantaggiosa!