



Valutazione delle prestazioni

Prof. Alberto Borghese
Dipartimento di Informatica
alberto.borghese@unimi.it

Università degli Studi di Milano

Riferimento Patterson: 1.6, 1.9, 1.10, 6.10



Sommario

Valutazione delle prestazioni

Benchmark

Il modello Roof-line

Esercizi



Perché valutare le prestazioni?



- Misura/Valutazione quantitativa delle prestazioni (velocità...).
- Fare scelte intelligenti (e.g. installare nuovo hardware o nuovo software).
- Orientarsi nell'acquisto di nuovo Hw.
- Fatturazione delle prestazioni.

Le prestazioni migliorano perché:

- Incrementa il **throughput** (la quantità di lavoro nell'unità di tempo, bandwidth). Quantità di dati elaborati nell'unità di tempo.
- Diminuisce il **tempo di esecuzione** (execution time, response time). Velocità di esecuzione.

Domande:

Un processore più veloce cosa influenza?

Più processori dedicati, cosa modificano?



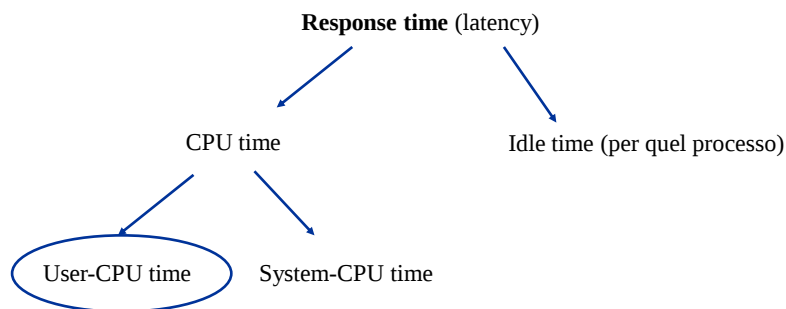
Misura delle prestazioni (Fatturazione)



Compito non semplice per la stretta connessione tra HW, Software di sistema e Software Utente.

Latenza di un'operazione di I/O, di accesso a DRAM viene nascosta facendo altro.
Cosa valutiamo?

Chi valuta? Il progettista? L'utente finale?





Misura delle prestazioni (Progettista)



$$\text{Execution time} = \text{Num_Cicli_Clock} * T_{\text{clock}} = \text{Num_Cicli_Clock} / f_{\text{clock}}$$

$$\text{CPI} = \text{Num_Cicli_Clock} / \text{Num_Istruzioni}$$

IPC – Number of Instructions per Cycle

$$\text{IPC} = 1 / \text{CPI}$$

$$\text{Execution time} = (\text{CPI} * \text{Num_Istruzioni}) * T_{\text{clock}}$$

Equazione delle prestazioni di una CPU basata sul CPI

$$\text{Execution time} = (\text{CPI} * \text{Num_Istruzioni}) * T_{\text{clock}}$$

- Diminuzione del periodo di clock (aumento della frequenza)
- Diminuzione del numero di istruzioni
- Diminuire il numero di cicli di clock medi per ogni istruzione



Esempio - I



Decisione tra due sequenze di istruzioni. Il progettista HW comunica che si identificano 3 cluster di istruzioni, dove ciascun cluster utilizza un numero di cicli di clock diverso:

- Cluster A: 1 ciclo di clock
- Cluster B: 2 cicli di clock
- Cluster C: 3 cicli di clock

Le due sequenze di istruzioni hanno un mix diverso delle istruzioni di tipo A, B e C.

Sequenza 1:

- Istruzioni di tipo A: 2
- Istruzioni di tipo B: 1
- Istruzioni di tipo C: 2

TOTALE: 5 istruzioni

Sequenza 2:

- Istruzioni di tipo A: 4
- Istruzioni di tipo B: 1
- Istruzioni di tipo C: 1

TOTALE: 6 istruzioni

Quale sequenza scegliere?



Esempio - II



Decisione tra due sequenze di istruzioni. Il progettista HW comunica che si identificano 3 cluster di istruzioni, dove ciascun cluster utilizza un numero di cicli di clock diverso:

- Cluster A: 1 ciclo di clock
- Cluster B: 2 cicli di clock
- Cluster C: 3 cicli di clock

Le due sequenze di istruzioni hanno un mix diverso delle istruzioni di tipo A, B e C.

Sequenza 1:

- Istruzioni di tipo A: 2
- Istruzioni di tipo B: 1
- Istruzioni di tipo C: 2

TOTALE: **5 istruzioni**

Sequenza 2:

- Istruzioni di tipo A: 4
- Istruzioni di tipo B: 1
- Istruzioni di tipo C: 1

TOTALE: **6 istruzioni**

Quale sequenza scegliere?

$$T_{CPU_1} = 2*1 + 1*2 + 2*3 = 10 \text{ cicli clock}$$

$$T_{CPU_2} = 4*1 + 1*2 + 1*3 = 9 \text{ cicli clock}$$

$$CPI_1 = 10 \text{ cicli clock} / 5 \text{ istruzioni} = 2$$

$$CPI_2 = 9 \text{ cicli clock} / 6 \text{ istruzioni} = 1,5$$

$$\text{Execution time} = (CPI * \text{Num_Istruzioni}) / f_{\text{clock}}$$

CPI della sequenza 2 è molto minore e compensa l'aumento di un'istruzione a parità di clock.

A.A. 2024-2025

7/55

<http://borghese.di.unimi.it/>

7



Osservazioni



$$\text{Execution time} = (CPI * \text{Num_Istruzioni}) * T_{\text{clock}}$$

$$\text{Execution time} = (CPI * \text{Num_Istruzioni}) / f_{\text{clock}}$$

Modificando l'implementazione dell'ISA si può ottenere un cammino critico inferiore e quindi una frequenza di clock superiore -> un CPI inferiore (istruzioni più veloci).

Modificando l'ISA si può aumentare il numero di istruzioni (e.g. togliendo l'istruzione movs dall'ISA Intel e sostituendola con un ciclo - cf. Core i7)

CPI, Numero di istruzioni e T_{clock} non sono indipendenti.

Occorre un compromesso.

A.A. 2024-2025

8/55

<http://borghese.di.unimi.it/>

8



Recap e osservazioni



$$\text{Execution time} = (\text{CPI} * \text{Num_Istruzioni}) / f_{\text{clock}}$$

Dato un programma Utente, possiamo calcolare il tempo di **risposta** (ma questo conterrà anche il tempo di esecuzione della parte di Sistema e il tempo in cui il processore è idle).

Alternativamente possiamo calcolare:

- f_{clock} -> dato dalle specifiche (in realtà esiste la possibilità di aumentare temporaneamente la frequenza, fino a un certo livello di calore. E.g. Core i-7, la frequenza aumenta del 10% - Turbo mode).
- Num_Istruzioni (in linguaggio macchina) -> dai tool di profilazione.
- CPI -> E' difficile da determinare ed è questionabile.



CPI di un instruction mix



$$\text{Execution time} = (\text{CPI} * \text{Num_Istruzioni}) / f_{\text{clock}}$$

Se tutte le istruzioni fossero dello stesso tipo -> CPI = Num Cicli clock di ogni istruzione

$$CPI_{\text{medio}} = \frac{\sum_{i=1}^n (CPI_i * l_i)}{\sum_{i=1}^n l_i} = \frac{\sum_{i=1}^n (CPI_i * l_i)}{l_{\text{TOT}}} = \sum_{i=1}^n (CPI_i * f_i)$$

$$CPI_1 = 10 \text{ cicli clock} / 5 \text{ istruzioni} = 2$$

$$CPI_2 = 9 \text{ cicli clock} / 6 \text{ istruzioni} = 1,5$$

Operazioni intere più veloci delle operazioni in virgola mobile

Operazioni di salto condizionato più lente

.....

Nelle architetture super-scalari, cluster di istruzioni richiedono tempi diversi.

Quindi. Quale mix di istruzioni?



Esempio - 1



Si consideri un calcolatore in grado di eseguire le istruzioni riportate in tabella (computationally intensive):

Calcolare CPI e il tempo di CPU per eseguire un programma composto da **200 istruzioni** supponendo di usare una frequenza di clock pari a 500 MHz ($T = 2 \text{ ns}$).

	Frequenza	cicli di clock
ALU	43%	1
Load	21%	4
Store	12%	4
Branch	12%	2
Jump	12%	2

$$\text{CPI} = 0,43 * 1 + 0,21 * 4 + 0,12 * 4 + 0,12 * 2 + 0,12 * 2 = 2,23$$

$$T_{\text{CPU}} = 200 \text{ istruzioni} * 2,23 \text{ Clock / istruzione} = 446 \text{ cicli di clock} \Rightarrow 446 * 2 \text{ ns} = 892 \text{ ns}$$

A.A. 2024-2025

11/55

<http://borghese.di.unimi.it/>

11



Esempio - 2



Si consideri un calcolatore in grado di eseguire le istruzioni riportate in tabella (memory intensive):

Calcolare CPI e il tempo di CPU per eseguire un programma composto da **200 istruzioni** supponendo di usare una frequenza di clock pari a 500 MHz ($T = 2 \text{ ns}$).

	Frequenza	cicli di clock
ALU	23%	1
Load	31%	4
Store	22%	4
Branch	12%	2
Jump	12%	2

$$\text{CPI} = 0,23 * 1 + 0,31 * 4 + 0,22 * 4 + 0,12 * 2 + 0,12 * 2 = 2,83$$

$$T_{\text{CPU}} = 200 \text{ istruzioni} * 2,83 \text{ Clock / istruzione} = 566 \text{ cicli di clock} \Rightarrow 566 * 2 \text{ ns} = 1.132 \text{ ns}$$

A.A. 2024-2025

12/55

<http://borghese.di.unimi.it/>

12



Misura del tempo di esecuzione



Tempo di esecuzione -> dipende dal mix di istruzioni

$$CPI_{medio} = \frac{\sum_{i=1}^n (CPI_i * I_i)}{\sum_{i=1}^n I_i} = \frac{\sum_{i=1}^n (CPI_i * I_i)}{I_{TOT}} = \sum_{i=1}^n (CPI_i * f_i)$$

$$t_{medio} = CPI_{medio} * T_{clock}$$

Quale CPI_{medio} scelgo? Su quali istruzioni effettuo la media?

	Frequenza	cicli di clock
ALU	23%	1
Load	31%	4
Store	22%	4
Branch	12%	2
Jump	12%	2

CPI = 2,23

	Frequenza	cicli di clock
ALU	43%	1
Load	21%	4
Store	12%	4
Branch	12%	2
Jump	12%	2

CPI = 2,83

A.A. 2024-2025

13/55

<http://borghese.di.unimi.it/>

13



Gli altri component: numero di istruzioni



$$\text{Execution time} = (CPI * \text{Num_Istruzioni}) / f_{clock}$$

MIPS: milioni di istruzioni al secondo

MFLOPS: milioni di operazioni in virgola mobile al secondo

Problemi con il MIPS:

- Dipende dall'insieme di istruzioni, quindi è difficile confrontare computer con diverse ISA;
- Il tempo totale di esecuzione dipende da diverse caratteristiche: dischi, sottosistema di I/O, sottosistema grafico Per questo motivo occorre menzionare la configurazione del sistema (dipende dall'architettura).
- Varia a seconda del programma considerato;
- Può variare in modo inversamente proporzionale alle prestazioni!
- **Valore di picco**, scelgo il mix di istruzioni per massimizzare il MIPS misurato (fuorviante).

A.A. 2024-2025

14/55

<http://borghese.di.unimi.it/>

14



Esempio di trabocchetto con i MIPS di picco



Macchina con hardware opzionale per virgola mobile (co-processore).

Le istruzioni in virgola mobile **sono più lente**: richiedono **più cicli di clock** rispetto a quelle che lavorano con interi,

- i programmi che usano l'hardware opzionale per la virgola mobile in luogo delle routine software per tali operazioni impiegano **meno tempo** ma hanno un MIPS più **basso** (eseguono meno istruzioni!!).
- L'implementazione software delle istruzioni in virgola mobile esegue semplici istruzioni, con il risultato di avere un elevato MIPS, ma ne esegue talmente tante da avere un tempo di esecuzione **più elevato!!**

A.A. 2024-2025

15/55

<http://borghese.di.unimi.it/>

15



Problemi con MIPS di picco



Intel i860 (1989) dichiarava:

- 2 operazioni VM al secondo
- Clock di 50 Mhz



Prestazioni attese di **100 MFlops**

MIPS R3000 (1989) dichiarava:

- **16 MFlops**
- Clock a 33 Mhz

Su problemi reali l'i860 risultò 12% più lento del MIPS R3000!

Intel i860 dichiarava i MFlops di picco, difficilmente raggiungibili e sostenibili.

Pentium e PentiumPro: a fronte di un raddoppio della frequenza di clock che è passata da 100 a 200 Mhz, si è registrato un aumento delle prestazioni misurate tramite SpecInt di 1,7 volte e di 1,4 volte misurate in SpecFloat. Come mai? Perché alcune istruzioni hanno richiesto più cicli di clock.

A.A. 2024-2025

16/55

<http://borghese.di.unimi.it/>

16



Legge di Amdhal: Come rendere più veloce un elaboratore



$$\text{Execution time} = (\text{CPI} * \text{Num_Istruzioni}) * T_{\text{clock}}$$

Rendere veloce il caso più comune.

Si deve favorire il caso più frequente a discapito del più raro.

Il caso più frequente è spesso il più semplice e può essere quindi reso più veloce del caso infrequente.

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.

Metrica: speed-up: $T_{\text{old}} / T_{\text{new}} = V_{\text{new}} / V_{\text{old}}$



Corollario della legge di Amdhal



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 uno1 meno tale frazione:

$$\text{Speedup}_{\text{globale}} < 1/(1-F_m).$$

1. **Frazione migliorato** ($F_m \leq 1$), ovvero la frazione del tempo di calcolo della macchina originale che può essere modificato per avvantaggiarsi dei miglioramenti. Divido T_{old} in tempo migliorato e tempo non migliorato:

$$T_m = F_m * T_{\text{old}}$$

$$T_{\text{nm}} = (1 - F_m) * T_{\text{old}}$$

$$T_{\text{old}} = T_m + T_{\text{nm}}$$

2. **Speedup migliorato** ($S_m \geq 1$), ovvero il miglioramento ottenuto dal modo di esecuzione più veloce ($T_{\text{old}}/T_{\text{new}}$). Si applica solo al period di tempo T_m :

3. **Tempo migliorato:** $T_{\text{new}} = (F_m * T_{\text{old}}) / S_m + (1 - F_m) * T_{\text{old}}$

$$\begin{aligned} \text{Speedup}_{\text{globale}} &= T_{\text{old}} / T_{\text{new}} = T_{\text{old}} / [(F_m * T_{\text{old}}) / S_m + (1 - F_m) * T_{\text{old}}] = T_{\text{old}} / [(F_m / S_m + (1 - F_m)) * T_{\text{old}}] \\ &= 1 / (F_m / S_m + (1 - F_m)) \end{aligned}$$

$$\text{Per } S_m \rightarrow \infty \quad T_{\text{old}} / T_{\text{new}} = 1 / (1 - F_m)$$



Esempio numerico - 1



Somma di 10 coppie di variabili scalari e somma di una coppia di matrici bidimensionali 12×10 . Ogni somma costa un tempo t .

Supponiamo che **solo la somma di matrici** sia parallelizzabile e che abbiamo a disposizione **40 processori** su cui parallelizzare. Ogni operazione di somma costa un tempo t .

Qual è lo speed-up? **Non è di 40 volte!!**

Il tempo senza parallelizzazione sarà: $10t + (12 \times 10)t = 130t$

Il tempo dopo la parallelizzazione sarà: $10t + ((12 \times 10) / 40)t = 13t$

Lo speed-up sarà quindi: $130t / 13t = 10$

La velocità aumenta di 10 volte e non di 40 come ci si poteva aspettare.

Ci sono delle parti di codice non parallelizzate (somma di scalari) che limitano il guadagno.

Stesso risultato con Amdahl: $\text{Speedup}_{\text{globale}} = 1 / (F_m / S_m + (1 - F_m)) =$
 $1 / (120/130 / 40 + 10/130) = 520 / 5200 = 10$
Frazione di tempo: $F_m = 120/130$ $F_{nm} = 10/130$

A.A. 2024-2025

19/55

<http://borghese.di.unimi.it/>

19



Esempio numerico - 2



Somma di 1000 coppie di variabili scalari e somma di una coppia di matrici bidimensionali 12×10

Supponiamo che solo la somma di matrici sia parallelizzabile e che abbiamo a disposizione 40 processori su cui parallelizzare. Ogni operazione di somma costa un tempo t .

Qual è lo speed-up?

Il tempo senza parallelizzazione sarà: $1000t + (12 \times 10)t = 1120t$

Il tempo dopo la parallelizzazione sarà: $1000t + ((12 \times 10) / 40)t = 1003t$

Lo speed-up sarà quindi: $1120t / 1003t = 1,116$

La velocità aumenta di poco più di 1 volta e non di 40 come ci si poteva aspettare -> non è stata una scelta illuminata.

Ci sono delle parti di codice non parallelizzate (somma di scalari) che limitano il guadagno.

A.A. 2024-2025

20/55

<http://borghese.di.unimi.it/>

20



Sommario



Valutazione delle prestazioni

Benchmark

Il modello Roof-line

Esercizi



I benchmark



MIPS / MFLOPS di poco poco significativi. Sono metriche utilizzate dai progettisti per valutare il tempo di esecuzione.

CPI poco sono poco significativi.

Cosa può utilizzare l'Utente?

Benchmarks = Programmi per valutare le prestazioni.

Benchmarks: Whetstone, 1976; Drystone, 1984.

Kernel benchmark. Loop Livermore, Linpack, 1980. Problema: polarizzazione del risultato. Benchmark con programmi piccoli (10-100 linee, 1980). Problema: mal si adattano alle strutture gerarchiche di memoria. Ma **vengono utilizzati oggi per determinare i calcolatori più veloci.**



Indici SPEC ('89, '92, '95, '00, '06, '16)



<http://www.spec.org/>. The **Standard Performance Evaluation Corporation (SPEC)** is a non-profit corporation formed to establish, maintain and endorse a standardized set of relevant benchmarks that can be applied to the newest generation of high-performance computers. SPEC develops benchmark suites and also reviews and publishes submitted results from our [member organizations](#) and other benchmark licensees.

Insieme di programmi test.

Condizioni diverse: singolo / multiplo processore / time sharing.

Benchmark specifici per valutare SO e I/O.

SPEC'95 -> SPECint, SPECfp, base Sun SPARCstation 10/40.

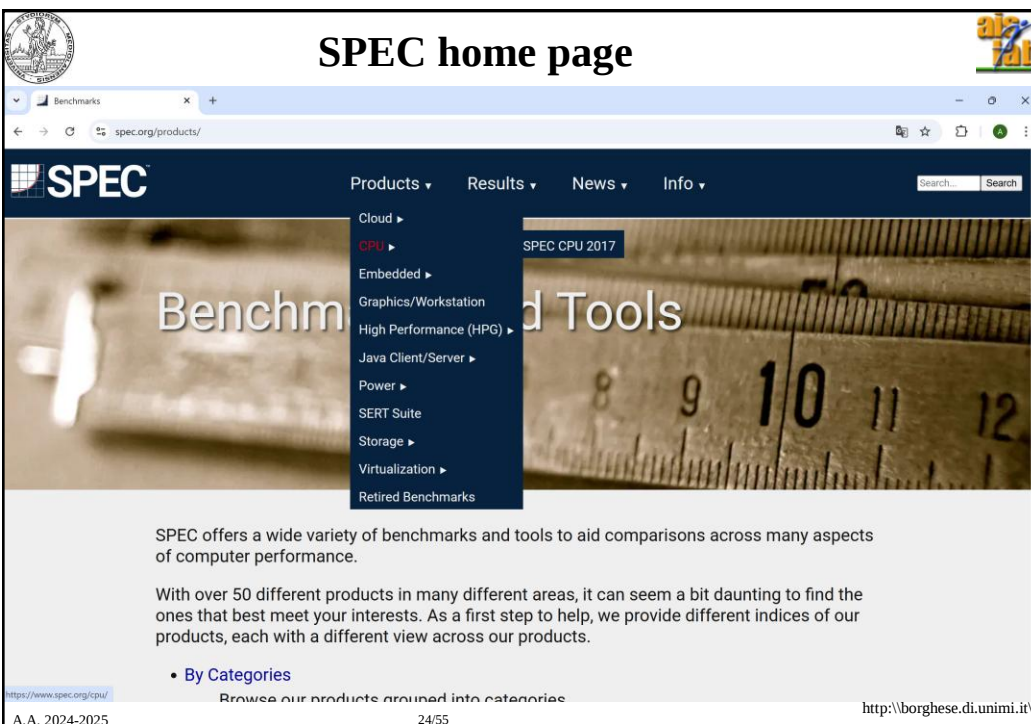
Benchmark particolari:

SDM (Systems Development Multitasking).

SFS (System-level File Server).

SPEC96. Elaborazioni scientifiche ad alto livello.

Orientamento: Benchmark specifici.



The screenshot shows the SPEC home page with a navigation menu and a list of products. The navigation menu includes: Products, Results, News, and Info. The Products menu is expanded, showing: Cloud, CPU, Embedded, Graphics/Workstation, High Performance (HPG), Java Client/Server, Power, SERT Suite, Storage, Virtualization, and Retired Benchmarks. The main content area features a large image of a ruler with the text "Benchmark Tools" and "SPEC CPU 2017". Below the image, there is a paragraph of text: "SPEC offers a wide variety of benchmarks and tools to aid comparisons across many aspects of computer performance. With over 50 different products in many different areas, it can seem a bit daunting to find the ones that best meet your interests. As a first step to help, we provide different indices of our products, each with a different view across our products." At the bottom, there is a section titled "By Categories" with a link to "Browse our products grouped into categories".



SPEC INT 2000



Benchmark	Language	Category	Full Descriptions
164.gzip	C	Compression	HTML Text
175.vpr	C	FPGA Circuit Placement and Routing	HTML Text
176.gcc	C	C Programming Language Compiler	HTML Text
181.mcf	C	Combinatorial Optimization	HTML Text
186.crafty	C	Game Playing: Chess	HTML Text
197.parser	C	Word Processing	HTML Text
252.eon	C++	Computer Visualization	HTML Text
253.perlbmk	C	PERL Programming Language	HTML Text
254.gap	C	Group Theory, Interpreter	HTML Text
255.vortex	C	Object-oriented Database	HTML Text
256.bzip2	C	Compression	HTML Text
300.twolf	C	Place and Route Simulator	HTML Text http://borghese.di.unimi.it/

A.A. 2024-2025

25/55

25



Evaluating Architecture performances



1.8 GHz							
Descrizione	Nome	Numero di istruzioni $\times 10^9$	CPI	Periodo di clock (secondi $\times 10^{-9}$)	Tempo di esecuzione (secondi)	Tempo di riferimento (secondi)	SPECratio
Interprete Perl	perlbench	2684	0,42	0,556	627	1774	2,83
Compilatore GNU C	gcc	2322	0,67	0,556	863	3976	4,61
Ottimizzazione combinatoria	mcf	1786	1,22	0,556	1215	4721	3,89
Libreria di simulazione di eventi discreti	omnetpp	1107	0,82	0,556	507	1630	3,21
Conversione da XML a HTML via XSLT	xalancbmk	1314	0,75	0,556	549	1417	3,21
Compressione video	x264	4488	0,32	0,556	813	1763	2,17
Intelligenza artificiale: ricerca su albero alpha-beta (scacchi)	deepsjeng	2216	0,57	0,556	698	1432	2,05
Intelligenza artificiale: ricerca ad albero Monte Carlo (Go)	leela	2236	0,79	0,556	987	1703	1,73
Intelligenza artificiale: generatore di soluzioni ricorsive (Sudoku)	exchange2	6683	0,46	0,556	1718	2939	1,71
Compressione generale di dati	xz	8533	1,32	0,556	6290	6182	0,98
Media geometrica							2,36

SPEC INT 2017 - integer benchmarks – 2017 su Intel Xeon E5-2650L
SPEC (System Performance Evaluation Cooperative)

A.A. 2024-2025

26/55

<http://borghese.di.unimi.it/>

26



Modalità di incremento delle prestazioni su architetture parallele



Come variano le prestazioni con il numero di nodi di elaborazione?

Weak scaling: Il tempo di esecuzione rimane lo stesso, La dimensione dei dati e programma (working set), cioè (la dimensione del problema, cresce con il numero di nodi di elaborazione

Strong scaling: la dimensione del programma e dei dati è fissa e le prestazioni aumentano linearmente con il numero di processori, senza variare le dimensioni del problema.

Qual è più facile da ottenere?

Che tipo di aumento abbiamo ottenuto nell'esempio della somma degli elementi di un vettore parallelizzata su P procesori?

A.A. 2024-2025

27/55

<http://borghese.di.unimi.it/>

27



Parallel SPEC



Anche valutazione dei cloud:
SPEC Cloud™ IaaS 2016

Anche Java server benchmark:
SPECjbb2015

A.A. 2024-2025

Benchmark	Tipo di scaling	Riprogrammazione	Descrizione
Lintpack	Debole	Si	Algebra lineare con matrici dense (Dongarra, 1979)
SPECrate	Debole	No	Parallelismo di programmi indipendenti (Henning, 2007)
SPLASH 2, Stanford Parallel Applications for Shared Memory [Applicazioni parallele di Stanford per memoria condivisa] (Whoo et al., 1995)	Forte (anche se offre solo due dimensioni di problemi)	No	FFT 1D complessa Decomposizione LU a blocchi Fattorizzazione di Cholesky di matrici sparse a blocchi Ordinamento Radix Sort di interi Barnes-Hut Multiplo veloce adattativo Simulazioni oceaniche Radioattività gerarchica Ray tracing Rendering di volumi Simulazione di acqua con strutture dati spaziali Simulazione di acqua senza strutture dati spaziali
Benchmark paralleli NAS (Bailey et al., 1991)	Debole	Si (solamente C o Fortran)	EP: applicazioni «embarrassingly parallel». ¹ MG: Simplified Multigrid (griglie multiple semplificate) CG: griglie non strutturate per il calcolo del gradiente coniugato FT: soluzione delle equazioni differenziali alle derivate parziali in 3D, utilizzando la FFT. IS: Large integer sort, ordinamento di un vettore di interi di grande dimensione.
Raccolta di benchmark PARSEC (Bienia et al., 2008)	Debole	No	Blackscholes – assegnamento dei prezzi utilizzando le equazioni differenziali alle derivate parziali Black-Scholes Bodytrack – tracking del corpo di una persona Carnegie – versione dell'algoritmo di simulated annealing per l'ottimizzazione dell'instradamento su rete dei pacchetti, che tiene conto della presenza della cache. Dedup – algoritmo di compressione di ultima generazione, basato su deduplicazione dei dati Facesim – Simulazione della mimica di un volto umano Ferret – server di ricerca per similarità di contenuto Fluidanimate – fluidodinamica per l'animazione con il metodo SPH Fragmine – ricerca degli insiemi di elementi frequenti Shvrancluster – clustering online di dati di input in stream Swaptions – assegnamento di un prezzo agli swap di un portafoglio titoli Vips – elaborazione delle immagini X264 – codifica video H.264
Moduli algoritmici di Berkeley (Asanovic et al., 2006)	Forte e debole	Si	Macchine a stati finiti Logica combinatoria Attraversamento di grafi Griglie strutturate Matrici dense Matrici sparse Metodi spettrali (FFT) Programmazione dinamica N-corpi MapReduce Backtrack Branch and bound Inferenza mediante modelli a grafi Griglia non strutturate

di.unimi.it

28



Descrizione dei benchmark paralleli



SPECrate. Weak scaling. Independent job parallelism. Vengono eseguite più copie di uno stesso programma.

Stanford parallel application for shared memory. Strong scaling. Problemi diversi simili agli SPEC CPU: FFT, LU decomposition. Fattorizzazione di matrici sparse. Ordinamento. Ray tracking. Volume rendering....

NAS parallel benchmarking. Weak scaling. Disegnati per la fluido-dinamica. Problemi multi-grid. Large integer sort. Equazioni differenziali parziali 3D risolte con FFT.

PARSEC. Weak scaling. Utilizzano i Pthread (POSIX threads) e OPENMP. Propongono applicazioni di frontiera. Tracking di persone (da video). Routing ottimizzato. Compressione dei dati. Mimica facciale. Ricerca di contenuti simili in un server. Fluido-dinamica. Image processing. Video encoding.

Berkeley design patterns. Weak scaling. Macchine a Stati Finiti. Logica combinatoria. Attraversamento di grafi. Matrici dense e sparse. Metodi di analisi spettrale (FFT). Programmazione dinamica. Problemi N-body. Ottimizzazione.

A.A. 2024-2025

29/55

<http://borghese.di.unimi.it/>

29



Benchmark sui calcolatori più potenti



Linkpack. Programmi di **algebra lineare. Weak scaling.** DGEMM è il nucleo (e quello che costa di più intermini computazionali di questi programmi). Linkpack determina il **più veloce calcolatore al mondo (1.000,000 di miliardi di flops!)**.

<https://www.top500.org/> - 500 GREEN EuroHPC/FZJ in Germany. 72.73 GFlops/Watt

Ultima competizione nel Maggio 2024, 64a edizione. The No. 1 spot is now held by the **El Capitan - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS.** It has **11,039,616 total cores**, sustained rate **1,742.00 PFlop/s**, Peak rate: **2,746.38 PFlop/s**, Power consumption; **29,580.98 kW** with a **power efficiency rating of 58.89 Gigafllops/watt**, **A total memory of 5,437.5 Tera byte. It has an extension of 700 m². It relies on gigabit ethernet for data transfer. Lawrence Livermore Laboratories.**

Fastest in Europe: Eni center in Ferrera, Erbognone, Italy

Summit (2019):

- 2,414,592 core
- 2,801,664 GB di Memoria Principale (>2 PetaByte)
- Potenza: 10,096.00 kW



A.A. 2024-2025

30/55

30



Sommario



Valutazione delle prestazioni

Benchmark

Il modello Roof-line

Esercizi

A.A. 2024-2025

31/55

<http://borghese.di.unimi.it/>

31

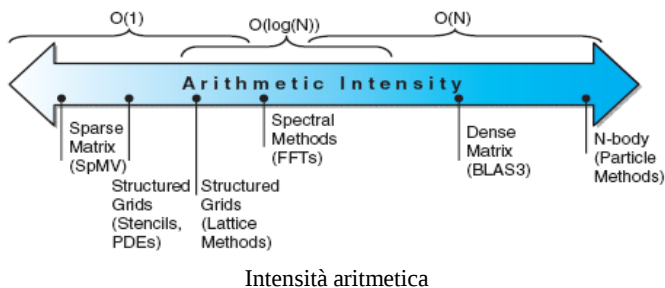


Arithmetic intensity



Quante operazioni facciamo per ogni Byte trasferito dalla memoria?

Obiettivo nella parallelizzazione è riutilizzare il più possibile i dati presenti in cache per evitare le miss penalty (e.g. blocchettizzazione dei dati).



Weak scaling, less memory data requests per byte per task con intensità maggiore

A.A. 2024-2025

32/55

<http://borghese.di.unimi.it/>

32

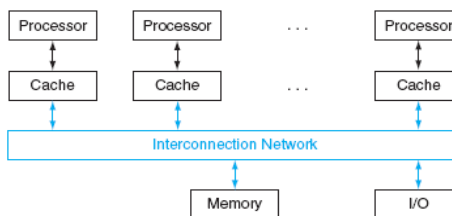


Ruolo dell'intensità aritmetica



Velocità di calcolo di picco: riempio al massimo tutti i core (tutti gli issue)

Velocità di trasferimento dalla gerarchia di memoria: riesco a fornire dati a una velocità sufficiente?



Se effettuiamo $N_{op} \gg 1$ operazioni su ogni byte letto dalla memoria, avremo la velocità di calcolo massima.



Il modello “roofline”



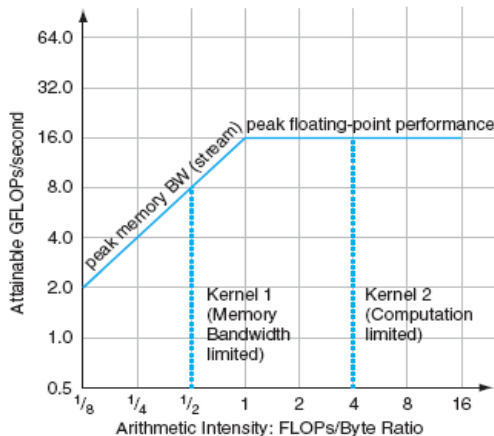
Nessun benchmark può essere contenuto interamente in cache.

2 elements:

- **Computation**
- **Memory transfer**

Per programmi con bassa intensità aritmetica (elevati accessi alla memoria per dato), il limite è offerto dal sistema di memoria.

Per programmi ad alta intensità, il limite è dato dalla capacità di elaborazione della CPU.

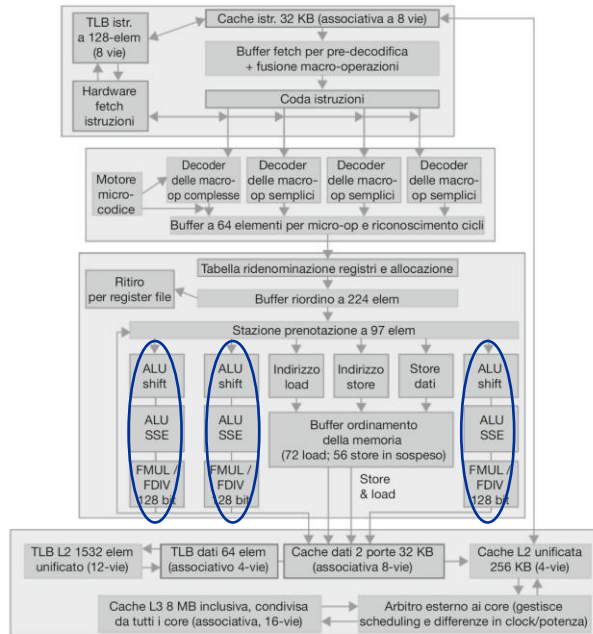


AMD Opteron X2

Memory speed = 16GByte/s



Il modello “roofline”: Computation



Ogni core, a ogni ciclo di clock deve riuscire ad eseguire il massimo numero di istruzioni FP previste, 3 per un Core i7

<http://borghese.di.unimi.it/>

35



Il modello “roofline”: Computation



Se non ci fossero problemi con la memoria le prestazioni sarebbero una linea orizzontale pari alla massima velocità di calcolo V_{calc} :

$$V_{calc} = P * V_{core} \text{ con } P \text{ numero di core}$$

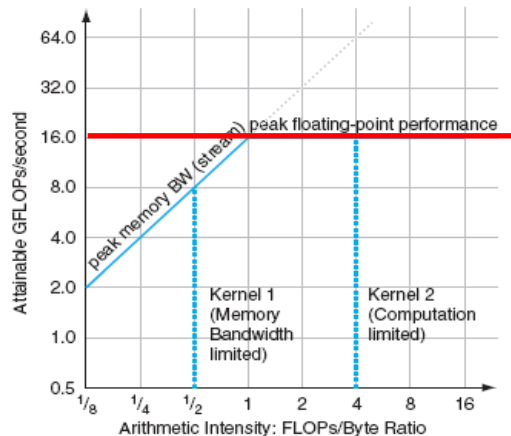
$$V_{calc} = \text{costante} = 16 \text{ Gflops per Opteron X2}$$

Tutto quello che viene letto dalla memoria può essere elaborato senza stalli.

Le prestazioni non dipendono dall'intensità aritmetica. Ma solo da P e V_{core} .

$$V_{core} = N_{cammini_FP} * N_{Dati} / \text{cammino} / \text{sec}$$

Il limite computazionale si può calcolare dalle caratteristiche strutturali della CPU.



AMD Opteron X2

A.A. 2024-2025

36/55

<http://borghese.di.unimi.it/>

36



Il modello “roofline”: Memory transfer



La memoria rifornisce la CPU.

I dati vengono letti dalla memoria con una certa velocità massima: $V_{\text{mem}} = [\text{Byte}]/[\text{s}]$.

Le prestazioni di memoria si valutano con un benchmark particolare: streaming benchmark.

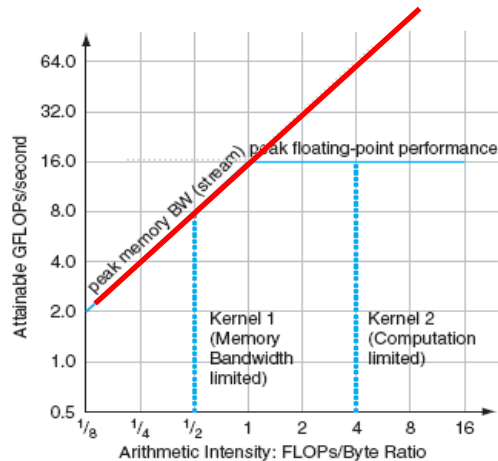
Tutti i dati letti dalla memoria vengono elaborati: per ogni byte effettuato un numero di operazioni N_{flop} definite dall'intensità aritmetica.

Maggiore è l'intensità, maggiore la velocità di calcolo.

Come si inserisce nel grafico questo vincolo?

All'aumentare della velocità del sistema di memoria, la retta trasla verso sx.

Il sottosistema di memoria associato all'AMD Opteron X2 ha una velocità di trasferimento di picco di 16GByte/s.



AMD Opteron X2

A.A. 2024-2025

37/55

<http://borghese.di.unimi.it/>

37



Il modello “roofline”: calcolo della pendenza



Il sottosistema di memoria associato all'AMD Opteron X2 ha una velocità di trasferimento di picco di 16GByte/s.

Intensità aritmetica 1 => 16 Gflops (1 Byte letto dalla memoria -> 1 Flops)

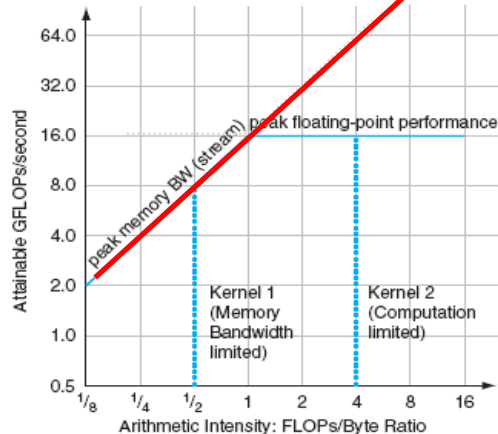
Intensità aritmetica 0.5 => 8 Gflops

Intensità aritmetica 0.25 => 4 Gflops

Intensità aritmetica 2 => 32 GFLOPs

....

Non si può superare la velocità massima offerta di calcolo della CPU



AMD Opteron X2

A.A. 2024-2025

38/55

<http://borghese.di.unimi.it/>

38



Il modello “roofline”: riassunto



Nessun benchmark può essere contenuto interamente in cache.

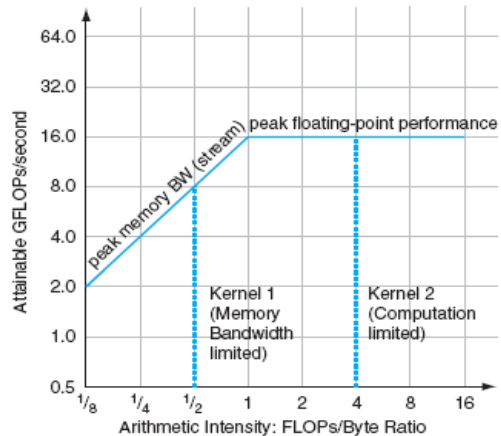
AMD Opteron X2

2 elements:

- Computation
- Memory transfer

Per programmi con bassa intensità aritmetica (elevati accessi alla memoria per dato), il limite è offerto dal sistema di memoria.

Per programmi ad alta intensità, il limite è dato dalla capacità di elaborazione della CPU.



Attainable GFLOPs/sec =

$\text{Min} \{ \text{Peak Memory BW} \times \text{Arithmetic Intensity}, \text{Peak Floating-Point Performance} \}$

Dipende dall'instruction mix, ovverosia dal tipo di kernel.

A.A. 2024-2025

39/55

<http://borghese.di.unimi.it/>

39



Dall'Opteron X2 all'Opteron X4

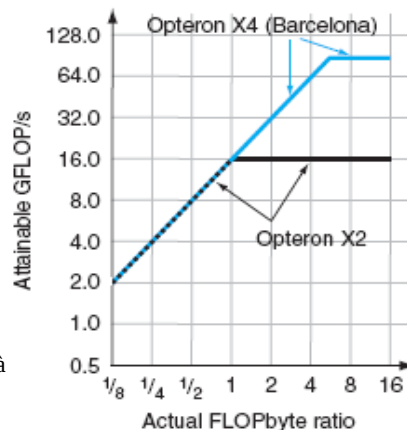


Opteron X4 vs Opteron X2:

- Stesso sistema di memoria (16 Gbyte/s)
- Numero doppio di processori (core)
- Numero quadruplo di operazioni in virgola mobile al secondo
 - Doppia capacità aritmetica della pipeline
 - Doppia capacità di fetch.

La velocità di elaborazione aumenta, ma solo per intensità aritmetiche superiori ad 1 quando si possono sfruttare maggiormente i dati già presenti in cache.

La velocità di calcolo massima di 80 Gflops si raggiunge solo per un'intensità aritmetica pari a 5.



A.A. 2024-2025

40/55

<http://borghese.di.unimi.it/>

40



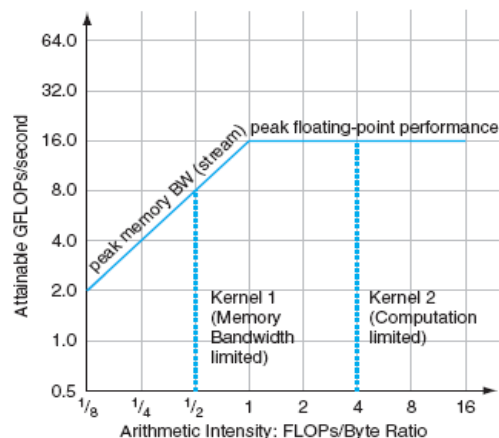
Roof model e ottimizzazioni



Cosa succede se le prestazioni del vostro programma risultano scadenti rispetto alle prestazioni attese?

Riempire meglio la pipeline.

Caricare meglio i dati dalla memoria.



A.A. 2024-2025

41/55

<http://borghese.di.unimi.it/>

41

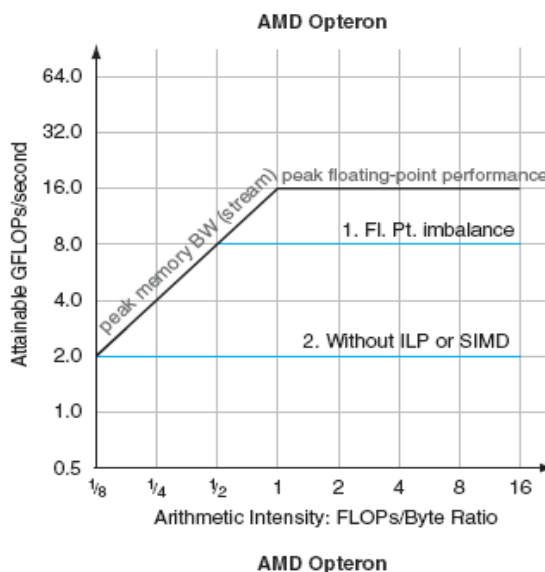


Ottimizzazioni sulla parte di calcolo - 1



Migliorare il mix di operazioni:

- Significativa percentuale di operazioni floating point
- Bilanciamento tra Moltiplicazioni e addizioni (e.g. fused add-multiplications, or equal number of execution paths).
- With imbalance, peak at 8 Gflops.



A.A. 2024-2025

42/55

<http://borghese.di.unimi.it/>

42



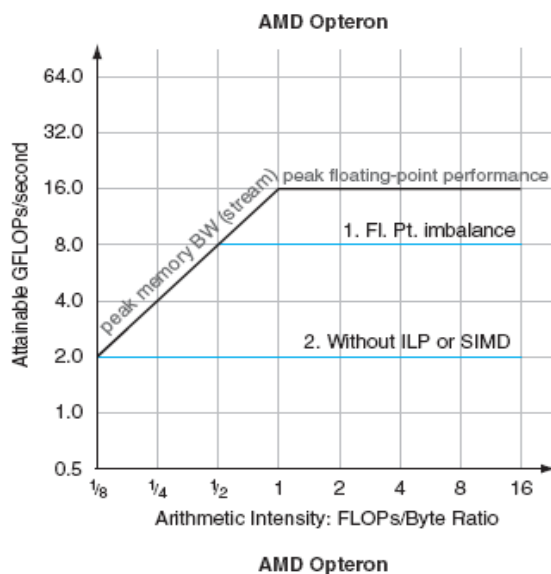
Ottimizzazioni sulla parte di calcolo - 2



Aumentare la parallelizzazione dell'esecuzione (ILP): 3-4 istruzioni per ciclo di clock.

- Srotolamento dei cicli
- Utilizzo delle istruzioni vettoriali (AVX)
- Ottimizzazione del mix delle istruzioni.

Without ILP or SIMD adequately exploited, 2 GFlops.



A.A. 2024-2025

43/55

<http://borghese.di.unimi.it/>

43



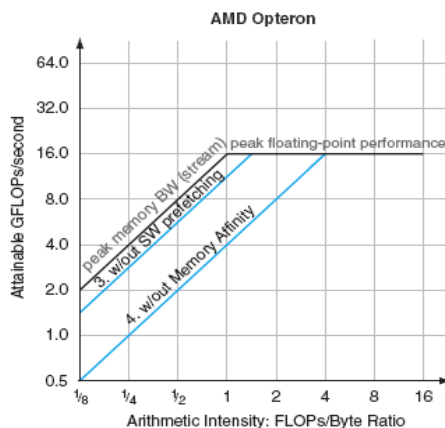
Ottimizzazioni sulla parte di memoria - 3



Software pre-fetching:

- Precaricamento dei dati in cache (speculazione efficace).
- Precaricamento delle istruzioni in cache (predizione dei salti)

Spostamento verso destra della curva della memoria: si reduce a velocità di trasferimento tra Memoria Principale e Cache



A.A. 2024-2025

44/55

<http://borghese.di.unimi.it/>

44



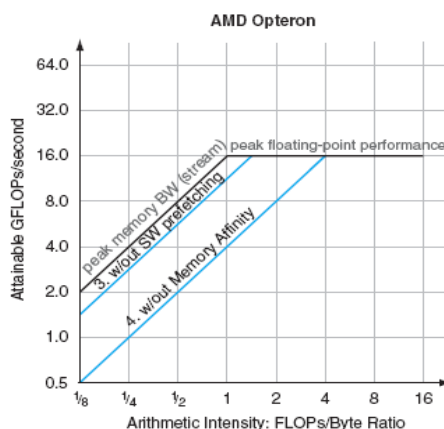
Ottimizzazioni sulla parte di memoria - 4



Affinità della memoria:

- Massimizzare gli hit.
- Separare il codice nei diversi core in modo che gli accessi in memoria siano all'interno della cache associata.
- Minimizzazione degli «invalidate».

Spostamento verso destra della curva della memoria: si reduce a velocità di trasferimento tra Memoria Principale e Cache.



A.A. 2024-2025

45/55

<http://borghese.di.unimi.it/>

45



Quali ottimizzazioni?



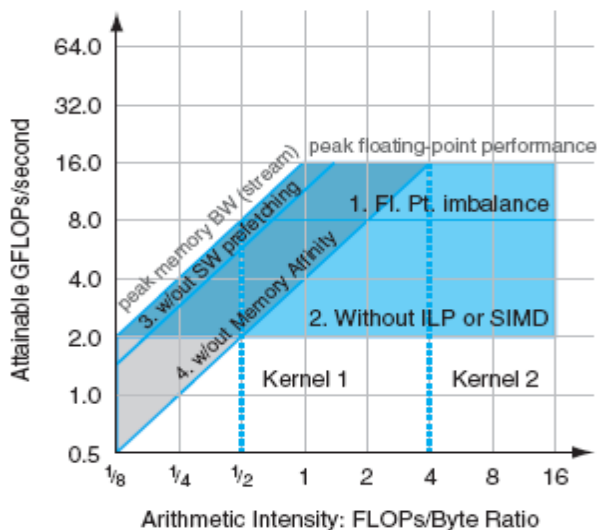
Occorre migliorare il codice perchè venga raggiunto il «tetto». Massima velocità di calcolo raggiungibile.

La distanza dal tetto indica quando si guadagna.

Ottimizzazioni eseguite in sequenza facilitano il compito.

Kernel 2 non ha problemi di memoria -> ottimizzazioni 1 e 2.

Kernel 1 -> ottimizzazioni 1, 2, 3 e 4.



A.A. 2024-2025

46/55

<http://borghese.di.unimi.it/>

46



Sommario



Valutazione delle prestazioni

Benchmark

Il modello Roof-line

Esercizi



Legge di Amdahl: Come rendere più veloci i calcolatori



$$\text{Execution time} = (\text{CPI} * \text{Num_Istruzioni}) * T_{\text{clock}}$$

Rendere veloce il caso più comune.

Si deve favorire il caso più frequente a discapito del più raro.

Il caso più frequente è spesso il più semplice e può essere quindi reso più veloce del caso infrequente.

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.

$$\text{Metrica: speed-up: } T_{\text{old}} / T_{\text{new}} = V_{\text{new}} / V_{\text{old}}$$

$$\text{Speedup}_{\text{globale}} = T_{\text{old}} / T_{\text{new}} = T_{\text{old}} / [(F_m * T_{\text{old}}) / S_m + (1 - F_m) * T_{\text{old}}] =$$

$$T_{\text{old}} / [(F_m / S_m + (1 - F_m)) * T_{\text{old}}] = 1 / (F_m / S_m + (1 - F_m))$$



Corollario della legge di Amdhal



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 uno 1 meno tale frazione:

$$Speedup_{globale} < 1/(1-F_m).$$

1. **Frazione_migliorato** ($F_m \leq 1$), ovvero la frazione del tempo di calcolo della macchina originale che può essere modificato per avvantaggiarsi dei miglioramenti. Divido T_{old} in tempo migliorato e tempo non migliorato:

$$T_m = F_m * T_{old}$$

$$T_{nm} = (1 - F_m) * T_{old}$$

2. **Speedup_migliorato** ($S_m \geq 1$), ovvero il miglioramento ottenuto dal modo di esecuzione più veloce (T_{old}/T_{new}). Si applica solo al periodo di tempo T_m :

$$T_{old} = T_m + T_{nm}$$

3. **Tempo migliorato**: $T_{new} = (F_m * T_{old}) / S_m + (1 - F_m) * T_{old}$

$$Speedup_{globale} = T_{old} / T_{new} = T_{old} / [(F_m * T_{old}) / S_m + (1 - F_m) * T_{old}] = T_{old} / [(F_m / S_m + (1 - F_m)) * T_{old}]$$

$$= 1 / (F_m / S_m + (1 - F_m))$$

$$\text{Per } S_m \rightarrow \infty \quad T_{old} / T_{new} = 1 / (1 - F_m)$$

A.A. 2024-2025

49/55

<http://borghese.di.unimi.it/>

49



Speed-up Esempio - 1



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?

$$Speedup_{globale} = 1 / [1 - F_m + F_m / S_m]$$

$$\text{Frazione}_{migliorato} = 0.4$$

$$\text{Speedup}_{migliorato} = 10$$

$$Speedup_{globale} = 1 / [1 - 0.4 + 0.6 / 10] = 1.56$$

A.A. 2024-2025

50/55

<http://borghese.di.unimi.it/>

50



Speed-up Esempio - 2



Supponiamo di potere aumentare la velocità della CPU della nostra macchina di un fattore 5 (senza influenzare le prestazioni di I/O) con un costo 5 volte superiore.

Assumiamo inoltre che la CPU sia utilizzata per il 50% del tempo ed il rimanente sia destinato ad attesa per operazioni di I/O. Se la CPU è un terzo del costo totale del computer è un buon investimento da un punto di vista costo/prestazioni, aumentare di un fattore cinque la velocità della CPU?

$$\text{Speedup}_{\text{globale}} = 1 / [1 - 0,5 + 0,5 / 5] = 1.67$$
$$\text{Incremento di costo} = 2.33$$

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



Esempio – 3: speed-up dovuto a 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. La *percentuale di vettorializzazione* è la porzione del tempo che può essere spesa usando la modalità vettoriale.

- Disegnare un grafico che riporti lo speedup come percentuale della computazione effettuata in modo vettoriale.
- Quale percentuale di vettorializzazione è necessaria per uno speedup di 2?
- Quale per raggiungere la metà dello speedup massimo?

La percentuale di vettorializzazione misurata è 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.

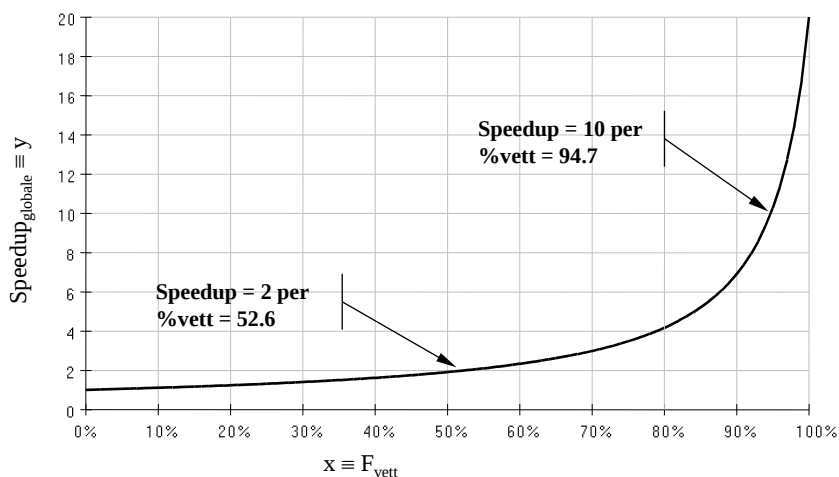
- Quale incremento della percentuale di vettorializzazione sarebbe necessario per ottenere lo stesso guadagno di prestazioni?
- Quale investimento raccomandereste?



Curva di speed-up



$$\text{Speedup}_{\text{globale}} \equiv y = 1 / [1 - x + x / 20] = 20 / (20 - 19x) \quad x \equiv F_{\text{vett}}$$



A.A. 2024-2025

53/55

<http://borghese.di.unimi.it/>

53



Speed-up dovuto a HW



$$\text{Speedup}_{\text{original}} = 1 / [1 - 0.7 + 0.7 / 20] = 1 / (1 - 0.7 * 19 / 20) = 2,9851$$

$$\text{Speedup}_{\text{HW}} = 1 / [1 - 0.7 + 0.7 / 40] = 1 / (1 - 0.7 * 39 / 40) = 3,1496$$

$$\text{Speedup}_{\text{compiler}} = 3,1496 = 1 / [1 - x + x / 20] \rightarrow F_{\text{vettoriale}} = 71,84\%$$

A.A. 2024-2025

54/55

<http://borghese.di.unimi.it/>

54



Sommario



Valutazione delle prestazioni

Benchmark

Il modello roof-line