


Valutazione delle prestazioni

Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
alberto.borghese@unimi.it

Università degli Studi di Milano


Sommario

Cosa vuol dire valutare le prestazioni?

Benchmark

Valutazione delle prestazioni multi-core

Legge di Amdhal

Perché valutare le prestazioni?

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

La misura delle prestazioni è il tempo.

$Prestazioni_X > prestazioni_Y \Rightarrow tempo_X < tempo_Y$

$Tempo_Y = (1 + (n / 100)) \times Tempo_X \Rightarrow n = 100 \times (Tempo_Y - Tempo_X)/Tempo_X$

Miglioro passando da Y a X

Le prestazioni migliorano perché:
- Incrementano le prestazioni.
- Diminuisce il tempo di esecuzione.

Criteri (metrica) di valutazione orientati all'utente

Velocità di esecuzione + quantità di informazione elaborata.

Il criterio di valutazione dipende dall'utilizzo del calcolatore!

1) Utilizzo personale -> **tempo di esecuzione**.
2) Utilizzo come server -> **throughput** .

Throughput:

Ammontare di lavori svolti in un dato tempo.
(accessi a banche dati, programmi, transazioni commerciali…).

Domande:

Un processore più veloce cosa influenza?
Più processori dedicati, cosa modificano?

Esempio

$p_A = 2$ $(t_A = 0.5)$ $p_B = 1.5$ $(t_B = 0.666\ldots)$

Valutazione in termini di tempo di prestazioni:
$p_B / p_A = 0.75$ B ha prestazioni pari al 75% di A.
Variazione delle prestazioni percentuale: $(p_B - p_A) / p_A = -0.25\%$
Utilizzando il tempo di esecuzione:
$(1/t_B) / (1/t_A) = t_A / t_B = 75\%$
Variazione delle prestazioni percentuale: $(1/t_B - 1/t_A) /(1/ t_A) = t_A*(1/t_B - 1/t_A) = ½ * (- ½) = -0.25\%$

Valutazione in termini di tempo di esecuzione:
$t_B/t_A = (2/3) / (1/2) = 4/3 = 1.3333\ldots$. B richiede il 133% del tempo di A per eseguire il programma.
Variazione delle prestazioni percentuale: $(t_B - t_A) / t_A = (2/3 - 1/2) / (1/2) = 1/3 \Rightarrow 33.3\ldots\%$
B richiede 33.3…% in più per l'esecuzione del programma.

Unità di misura delle prestazioni (CPI)

Tempo di CPU =
Numero_cicli_clock * Durata_clock =
Numero_cicli_clock / Frequenza_clock.

Determinazione del numero di cicli di clock:

Cicli di clock per istruzione (CPI) =
Cicli_clock_CPU_programma / Numero_istruzioni

Quindi:

$T_{CPU} = CPI * Numero\_Istruzioni * T_{clock}$

Esempio

Tempo di esecuzione del programma: 1.2s
Numero di istruzioni: 400k.
Clock: 1Mhz.

Per l'esecuzione del programma, occorrono: #Cicli_clock = 10^6 * 1.2

CPI = #Cicli_clock / #Istruzioni = 3.

NB Sulle macchine di oggi il CPI è inferiore ad 1.

$T_{medio} = \frac{\sum_i t_i}{\#\text{Istruzioni}} = \frac{T_{tot}}{\#\text{Istruzioni}}$

T_{tot} = CPI * T_{clock} * #Istruzioni

Misura delle prestazioni

Tempo esecuzione singola istruzione, ma:

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

Tempo esecuzione medio (pesato) di un mix di istruzioni:

$$t_{medio} = \frac{\sum_{i=0}^{S} t_i l_i}{\sum_{i=0}^{S} l_i}$$

l_i numero di istruzioni di tipo i

Misura delle prestazioni mediante CPI

$\mathbf{T_{CPU}}$ = CPI * Numero_Istruzioni * T_{clock}

$$t_{medio} = \frac{\sum_{i=0}^{S} l_i t_i}{\sum_{i=0}^{S} l_i} \qquad CPI_{medio} = \frac{\sum_{i=1}^{n}(CPI_i * l_i)}{\sum_{i=1}^{n} l_i} = \frac{\sum_{i=1}^{n}\frac{(CPI_i * l_i)}{l_{TOT}}}{\frac{\sum_{i=1}^{n} l_i}{l_{TOT}}} = \sum_{i=1}^{n}(CPI_i * f_i)$$

- CPI_i numero di cicli di clock per istruzioni di tipi i.
- l_i Numero di volte che l'istruzione i viene eseguita nel programma.
- f_i Frequenza con cui l'istruzione i viene eseguita nel programma.

($\sum_{i=1}^{n} l_i$ rappresenta il numero di istruzioni = l_{TOT})

$$T_{CPU} = \sum_{i=1}^{n}(CPI_i * l_i) * T_{clock}$$

Esempio

Si consideri un calcolatore in grado di eseguire le istruzioni riportate in tabella:

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.

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

CPI = 0,43 * 1 + 0,21 * 4 + 0,12 * 4 + 0,12 * 2 + 0,12 * 2= 2,23

$T_{CPU} = 200 * 2{,}23 * 2_{ns} = 892_{ns}$

MIPS = milioni di istruzioni per secondo

MIPS = (numero_istruzioni / 10^6) / tempo_esecuzione

MIPS = frequenza_clock / (CPI * 10^6) = $1/t_{clock}$ * 1/ (CPI*10^6) = 1 / (t_{medio}*10^6)

t_{clock} * CPI = t_{medio}

Problemi:

· dipende dall'insieme di istruzioni, quindi è difficile confrontare computer con diversi insiemi di istruzioni;

• 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.

• 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).

Esempio: macchina con hardware opzionale per virgola mobile. Le istruzioni in virgola mobile richiedono più cicli di clock rispetto a quelle che lavorano con interi, quindi 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**. 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 più elevato tempo di esecuzione!!

Sommario

Cosa vuol dire valutare le prestazioni?

Benchmark

Valutazione delle prestazioni multi-core

Legge di Amdhal

Misure & Problemi

MIPS relativi = $tempo_{CPU}$ /$tempo_{CPU\_ref}$ * $MIPS_{CPU\_ref}$. La CPU_{ref} è VAX-11/780. Problema: evoluzione dei sistemi.

MFLOPS per i super computer. Problema: misure di picco.

MIPS di picco e sostenuti. Problema: poco significative.

Benchmarks = Programmi per valutare le prestazioni.

Benchmarks: Whetstone, 1976; Drystone, 1984.

Kernel benchmark. Loop Livermore, Linpack, 1980. Problema: polarizzaione del risultato.

Benchmark con programmi piccoli (10-100 linee, 1980). Problema: mal si adattano alle strutture gerarchiche di memoria.

Evaluating Architecture performances

Throughput, Response time, Execution time

Small programs can be incredibly fast (kernel benchmarks)

Description	Name	Instruction Count × 10^9	CPI	Clock cycle time (seconds × 10^9)	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2,118	0.75	0.4	637	9,770	15.3
Block-sorting compression	bzip2	2,389	0.85	0.4	817	9,650	11.8
GNU C compiler	gcc	1,050	1.72	0.4	724	8,050	11.1
Combinatorial optimization	mcf	336	10.00	0.4	1,345	9,120	6.8
Go game (AI)	go	1,658	1.09	0.4	721	10,490	14.6
Search gene sequence	hmmer	2,783	0.80	0.4	890	9,330	10.5
Chess game (AI)	sjeng	2,176	0.96	0.4	837	12,100	14.5
Quantum computer simulation	libquantum	1,623	1.61	0.4	1,047	20,720	19.8
Video compression	h264avc	3,102	0.80	0.4	993	22,130	22.3
Discrete event simulation library	omnetpp	587	2.94	0.4	690	6,250	9.1
Games/path finding	astar	1,082	1.79	0.4	773	7,020	9.1
XML parsing	xalancbmk	1,058	2.70	0.4	1,143	6,900	6.0
Geometric Mean							11.7

SPEC (System Performance Evaluation Cooperative)

Indici SPEC (‘89, ‘92, ‘95)

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 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.

Orientamento: Benchmark specifici.

Esempio becnhmark SPEC95

Elaborazione intera:

1) Go	Intelligenza artificiale
2) m88ksim	Simulatore chip Motorola 88K; esecuzione di un programma.
3) gcc	Compilatore Gnu C che genera codice SPARC.
4) compres	Compressione e decompressione di un file in memoria.
5) li	Interprete lisp
6) ijpeg	Compressione e decompressione di immagini grafiche.
7) perl	Manipolazione di stringhe e numeri primi nel linguaggio di programmazione dedicato Perl.
8) vortex	Programma di gestione di una base di dati.

Esempio becnhmark SPEC95

Elaborazione 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: simuilazione MonteCarlo.
4) Hydro2D	Astrofisica: equazione idrodinamiche di Naiver 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	Risoluzione di problemi di temperatura, velocità del vento e diffusione di agenti inquinanti.
9) Fpppp	Chimica quantistica.
10) Wave5	Fisica dei plasmi: simulazione di particelle elettromagnetiche.

URL: *http://www.spec.org/*

SPEC CPU200 CINT2000

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

Parallel SPEC

Weak scaling: La dimensione dei dati e programma (working set) cresce con il numero di nodi di elaborazione.

Strong scaling: la dimensione del programma e dei dati è fissa e aumentano le prestazioni con l'aumentare del numero dei nodi di elaborazione.

Benchmark	Scaling?	Reprogram?	Description
Linpack	Weak	Yes	Dense matrix linear algebra [Dongarra, 1979]
SPECrate	Weak	No	Independent job parallelism [Henning, 2007]
Stanford Parallel Applications for Shared Memory SPLASH 2 [Woo et al., 1995]	Strong (although offers two problem sizes)	No	Complex 1D FFT Blocked LU Decomposition Blocked Sparse Cholesky Factorization Integer Radix Sort Barnes-Hut Adaptive Fast Multipole Ocean Simulation Hierarchical Radiosity Ray Tracer Volume Renderer Water Simulation with Spatial Data Structure Water Simulation without Spatial Data Structure
NAS Parallel Benchmarks [Bailey et al., 1991]	Weak	Yes (C or Fortran only)	EP: embarrassingly parallel MG: simplified multigrid CG: unstructured grid for a conjugate gradient method FT: 3-D partial differential equation solution using FFTs IS: large integer sort
PARSEC Benchmark Suite [Bienia et al., 2008]	Weak	No	Blackscholes—Option pricing with Black-Scholes PDE Bodytrack—Body tracking of a person Canneal—Simulated cache-aware annealing to optimize routing Dedup—Next-generation compression with data deduplication Facesim—Simulates the motions of a human face Ferret—Content similarity search server Fluidanimate—Fluid dynamics for animation with SPH method Freqmine—Frequent itemset mining Streamcluster—Online clustering of an input stream Swaptions—Pricing of a portfolio of swaptions Vips—Image processing x264—H.264 video encoding
Berkeley Design Patterns [Asanovic et al., 2006]	Strong or Weak	Yes	Finite-State Machine Combinational Logic Graph Traversal Structured Grid Dense Matrix Sparse Matrix Spectral Methods (FFT) Dynamic Programming N-Body MapReduce Backtrack/Branch and Bound Graphical Model Inference Unstructured Grid

Sommario

Cosa vuol dire valutare le prestazioni

Benchmark

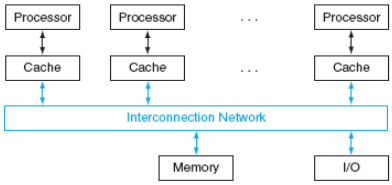
Valutazione delle prestazioni multi-core

Legge di Amdhal

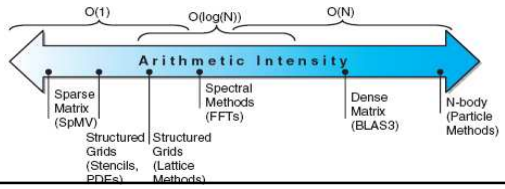
Arithmetic intensity

Velocità di calcolo FLOPS (floating point per second): velocità del singolo core: V_{core}.
In un'achitettura multi-core con P core la velocità di calcolo $V_{calc} = P * V_{core}$

Velocità di trasferimento dalla gerarchia di memoria. Per calcolarla dobbiamo capire quante operazioni devono essere fatte per ciascun byte caricato in cache, N_{op} / Byte (**Arithmetic Intensity**)



Se effettuiamo N_{op} >> 1 operazioni su ogni byte letto dalla memoria, avremo una velocità di calcolo massima pari a: $V_{max} = P * V_{core}$ = [Byte /s].



Weak scaling, less memory request per byte

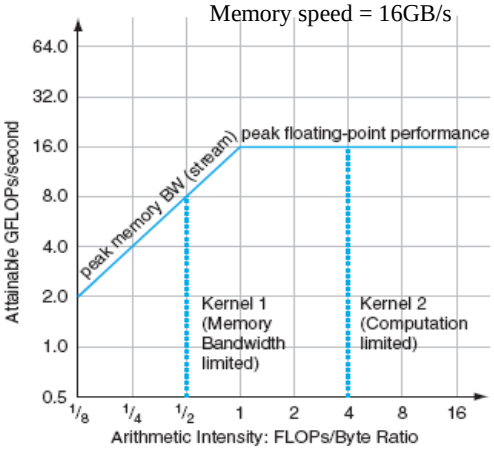
Il modello "roofline"

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

Il modello “roofline”: Computation

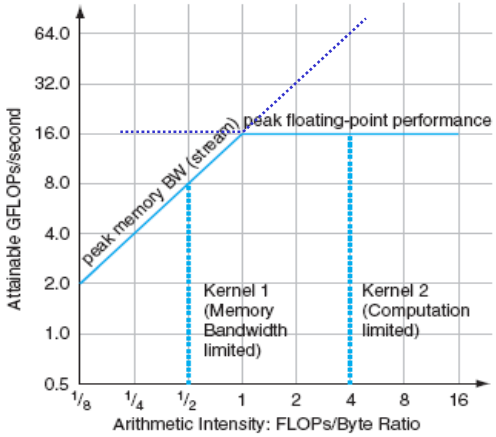
Se non ci fossero problemi con la memoria le prestazioni sarebbero una linea orizzontale pari alla massima capacità di calcolo:

V_{calc} = = P * V_{core}

V_{calc} = cost = 16 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}.



AMD Opteron X2

Il modello “roofline”: Memory transfer

La memoria rifornisce la CPU.

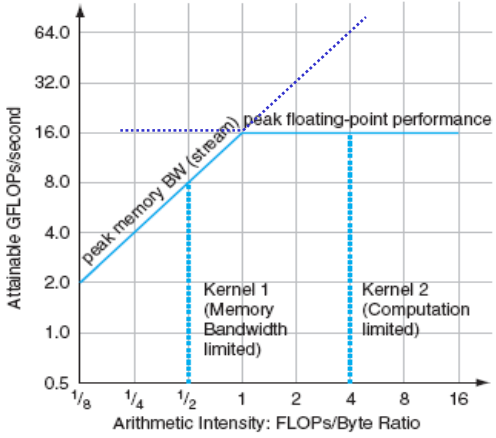
I dati vengono letti dalla memoria con una certa velocità massima: V_{mem} = [Byte]/[s].

La massima velocità di calcolo sarà V_{calc} = V_{mem}

Come si inserisce nel grafico questo vincolo?

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

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



AMD Opteron X2

Il modello “roofline”: riassunto

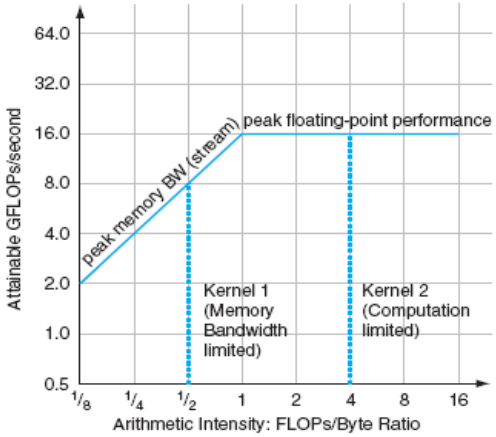
Nessun benchmark può essere contenuto interamente in cache.

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

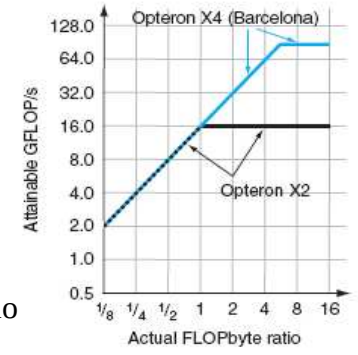


Attainable GFLOPs/sec (V_{calc})=
Min {Peak Memory BW × Arithmetic Intensity, Peak Floating-Point Performance}

Dall’Opteron X2 all’Opteron X4

Opteron X4 vs Opteron X2:

- **Stesso sistema di memoria**
- Numero doppio di processori (core)
- Numero quadruplo di operazionio 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.

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

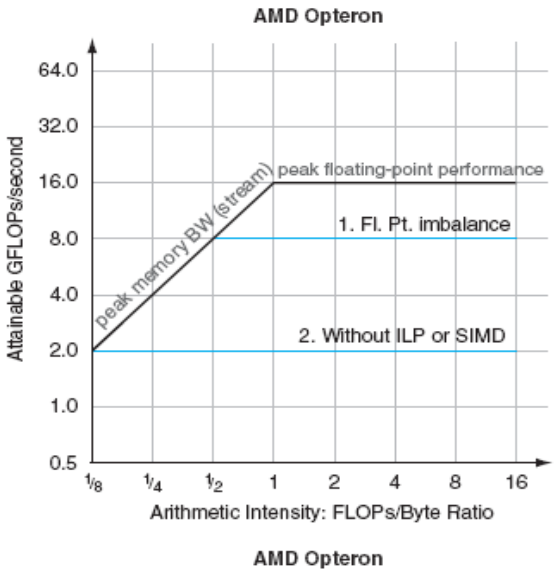
Ottimizzazioni sulla parte di calcolo - 1

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

Riempire meglio le pipeline.

Migliorare il mix di operazioni:

- Significativa percentuale di operazioni floating point
- Bilanciamento tra Moltiplicazioni e addizioni



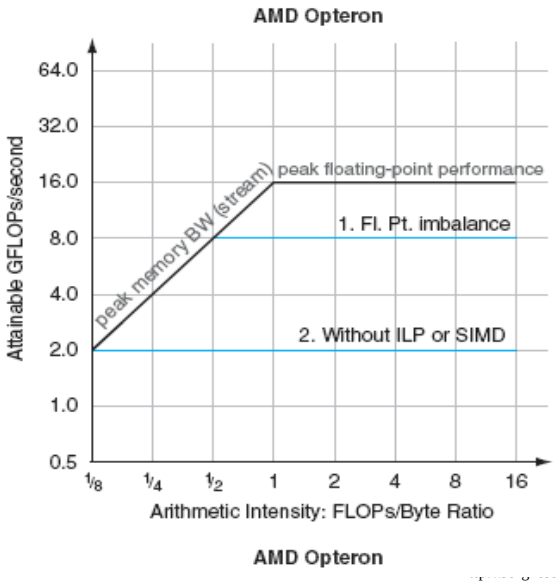
Ottimizzazioni sulla parte di calcolo - 2

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

Riempire meglio le pipeline.

Aumentare la parallelizzazione dell'esecuzione:

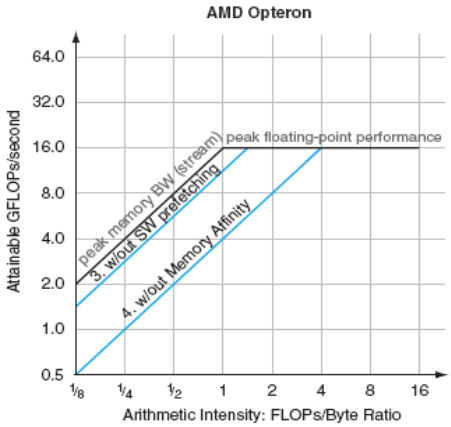
- Srotolamento dei cicli
- Ottimizzazione del mix delle istruzioni.



Ottimizzazioni sulla parte di memoria - 3

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

Caricare meglio i dati in CPU



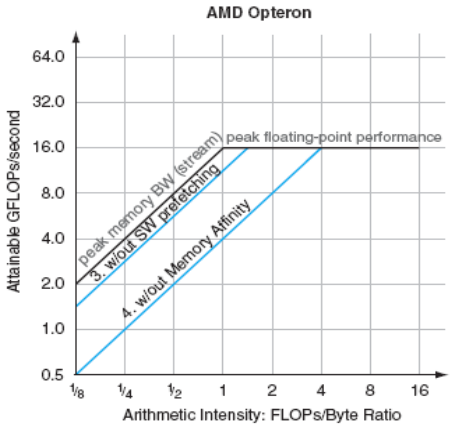
Software pre-fetching:

- Precaricamento dei dati in cache.
- Speculazione sui dati.

Ottimizzazioni sulla parte di memoria - 4

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

Caricare meglio i dati in CPU



Affinità della memoria:

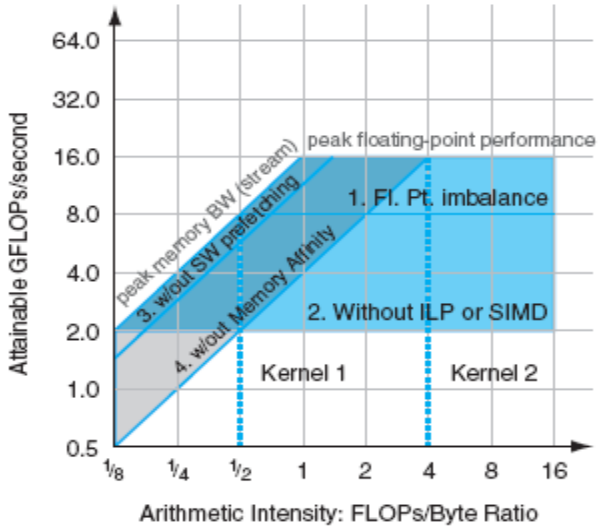
- Massimizzare gli hit.
- Separare il codice nei diversi core in modo che gli accessi in memoria siano all'interno della cache associato.
- Minimizzazione degli «invalidate».

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 da eseguire in sequenza.



Sommario

Cosa vuol dire valutare le prestazioni

Benchmark

Valutazione delle prestazioni multi-core

Legge di Amdhal

Miglioramento delle prestazioni

- Riduzione del numero di cicli di clock.
- Diminuzione del periodo di clock (aumentare la frequenza).

Tempo esecuzione = Numero_Cicli_clock * Durata del clock

CPI rappresenta il tempo di esecuzione medio delle istruzioni.
Miglioramenti dell'architettura per ridurre il CPI.
Miglioramento del compilatore per ridurre il CPI.
Ridurre la durata del clock (aumentarne la frequenza).

Espressione dei risultati

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.

Valutazione delle prestazioni, coerenza

	Calcolatore I	Calcolatore II
Istruzione A (s)	1	10
Istruzione B (s)	1000	100
Istruzione C (s)	10	100

Qual è più veloce? Dipende dal peso dei programmi.
Media pesata, tempo medio di esecuzione di un'istruzione:

$$T = 1/n \sum_{i=0}^{n} n_i t_i$$

Programma 1: 1000 istruzioni A, 1 istruzione B, 10 istruzioni C.

T_I = 1/1011 * (1000*1 + 1*1000+ 10*10) = 2100/1011 ≈ 2

T_{II} = 1/1011 * (1000*10 + 1*100 + 10*100) = 11100/1011 ≈ 100

Programma 2: 100 istruzione A, 10 istruzioni B, 10 istruzioni C.

T_I = 1/120 * (100*1 + 10*1000+ 10*10) = 10200/120 ≈ 100

T_{II} = 1/120 * (100*10 + 10*100 + 10*100) = 3000/120 ≈ 30


Come rendere più veloci i calcolatori

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 era eseguito.

Esempio: 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.


Speed-up

Il miglioramento globale proporzionale al miglioramento di una parte del sistema?

Speed up (accelerazione):

prestazioni_intero_lavoro_con_miglioramento / prestazioni_senza

Oppure

tempo_intero_lavoro_senza_miglioramento / tempo_con_miglioramento.

Speed-up - esempio

Consideriamo un calcolatore (CALC1) con ALU ed una FP_ALU.
Consideriamo un secondo calcolatore (CALC2) in cui la ALU è stata velocizzata (2x).

Consideriamo un'applicazione che prevede un 90% di istruzioni in aritmetica intera. Di quanto è lo speed-up?

	ISTRUZIONI INTERE		ISTRUZIONI TOTALI	
Calcolatore	**T_EXEC**	**Speedup_m**	**T_EXEC**	**Speedup**
CALC1	90	1.0	100	1.0
CALC2	45	2.0	55	1.82

Speed-up = 100/55 = 1.818….

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 meno tale frazione: $Speedup_{globale} < 1/(1-F_m)$.

Definizioni:

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. Nell'esempio precedente la frazione è 0.90.

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

2. **$Speedup_{migliorato}$** ($S_m \geq 1$), ovvero il miglioramento ottenuto dal modo di esecuzione più veloce. Nel precedente esempio questo valore viene fornito nella colonna chiamata Speedup_migliorato (pari a 2).

Dimostrazione

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

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

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

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

$1/ [1-F_m + F_m / S_m)] < 1/ [1-F_m]$ c.v.d. $(S_m \rightarrow \infty)$

Istruzioni non accelerate

Se il tempo di esecuzione delle istruzioni accelerate va all'∞ il tempo di esecuzione diventa il tempo di esecuzione delle istruzioni non accelerate soltanto.

Esempio precedente: $T_{new} = 100 * (1 – 0,9 + 0,9/2) = 55$

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?

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

$Frazione_{migliorato} = 0.4$

$Speedup_{migliorato} = 10$

$Speedup_{globale} = 1.56$

Esempio - 3

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?

$Speedup_{globale} = 1.67$ 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 – speedup 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 vettorizzazione* è la porzione del tempo che può essere spesa usando la modalità vettoriale.

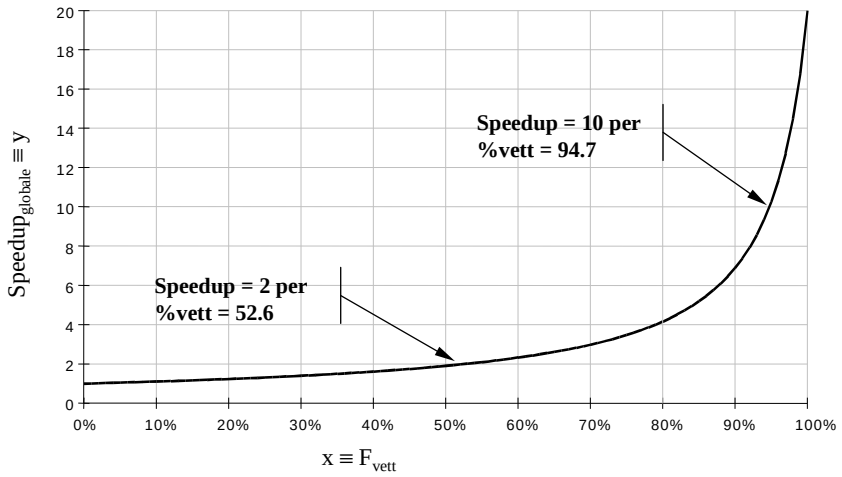
• Disegnare un grafico che riporti lo speedup come percentuale della computazione effettuata in modo vettoriale.
• Quale percentuale di vettorizzazione è necessaria per uno speedup di 2?
• Quale per raggiungere la metà dello speedup massimo?

La percentuale di vettorizzazione 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 vettorizzazione sarebbe necessario per ottenere lo stesso guadagno di prestazioni?
• Quale investimento raccomandereste?

Curva di speed-up

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



Speed-up dovuto a HW

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

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

$Speedup_{compiler} = 3,1496 = 1/ [1- x + x / 20)]$ ➔ $F_{vettoriale} = 71,84\%$

Sommario

Cosa vuol dire valutare le prestazioni

Benchmark

Valutazione delle prestazioni multi-core

Legge di Amdhal