


Valutazione delle prestazioni

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

Università degli Studi di Milano


Sommario

Cosa vuol dire valutare le prestazioni?

Benchmark

Valutazione delle prestazioni del sistema di memoria

Valutazione delle prestazioni multi-core

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: $(t_B - t_A) / t_A = (2/3 – 1/2) / (1/2) = 1/3 => 33.3\ldots\%$
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.

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

$T_{tot} = CPI * T_{clock} * \#\text{Istruzioni}$

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:

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

Misura delle prestazioni mediante CPI

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

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

- 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)

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

Esempio

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

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

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.

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 del sistema di memoria

Valutazione delle prestazioni multi-core

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 x 10^9	CPI	Clock cycle time (seconds x 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 del programma e dei dati è fissa

Strong scaling: la dimensione del programma e dei dati cresce proporzionalmente al numero dei processori.

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 del sistema di memoria

Valutazione delle prestazioni multi-core

Sommario

Cosa vuol dire valutare le prestazioni

Benchmark

Valutazione delle prestazioni del sistema di memoria

Valutazione delle prestazioni multi-core

Principio di località

I programmi riutilizzano dati e istruzioni che hanno usato di recente.

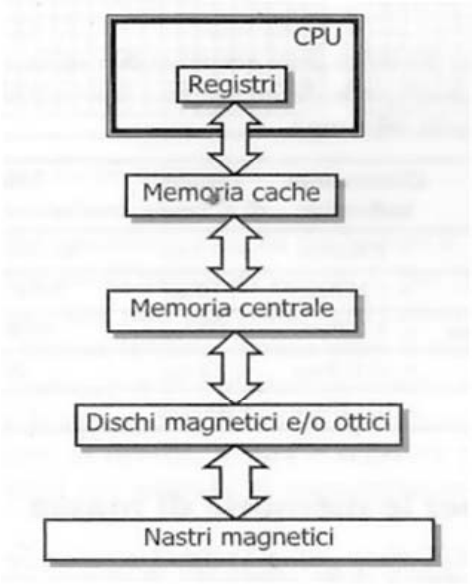
Regola pratica: un programma spende circa il **90%** del suo tempo di esecuzione per solo il **10%** del suo codice.

Basandosi sul passato recente del programma, è possibile predire con ragionevole accuratezza quali dati e istruzioni userà nel prossimo futuro.

località temporale elementi ai quali si è fatto riferimento di recente saranno utilizzati ancora nel prossimo futuro.

località spaziale elementi i cui indirizzi sono vicini, tendono ad essere referenziati in tempi molto ravvicinati.

Gerarchia di memorie



Valutazione prestazioni memoria

Obbiettivo principale della gerarchia di memoria è incrementare le prestazioni => diminuire la velocità di accesso sia in caso di HIT che di MISS.

Cosa succede in caso di MISS?

HIT_TIME Tempo di accesso al livello superiore (che comprende anche il tempo necessario per determinare se l'accesso ha avuto successo oppure fallisce.

MISS_PENALTY è composto da:
TEMPO DI ACCESSO per accedere alla prima parola del blocco dopo che è stato rilevato il fallimento.
TEMPO DI TRASFERIMENTO per trasferire le altre parole del blocco al livello superiore.

MISS_TIME = > HIT_TIME + MISS_PENALTY

Tempo medio di accesso alla memoria

TEMPO DI ACCESSO. E' legato al tempo di accesso del livello inferiore di memoria.

TEMPO DI TRASFERIMENTO. E' legato alla larghezza di banda del canale di comunicazione tra i due livelli di memoria (e.g. bus).

Il tempo medio di accesso alla memoria sarà:

$\mathbf{T_{medio}}$ = HIT_RATE * HIT_TIME + MISS_RATE * MISS_TIME =

HIT_TIME*HIT_RATE+MISS_RATE*(HIT_TIME+MISS_PENALTY) =

HIT_TIME*(HIT_RATE+MISS_RATE)+MISS_RATE*MISS_PENALTY =

HIT_TIME + MISS_RATE * MISS_PENALTY

Impatto di una memoria cache

Il tempo di CPU è composto dal tempo richiesto dalla CPU per eseguire il programma e dal tempo che la CPU trascorre in attesa di risposta dal sottosistema di memoria.

T_{CPU} = (#Cicli della CPU in esecuzione + #Cicli di stallo) * T_{Clock}

Ipotesi:
•Tutti gli stalli di memoria sono dovuti al fallimento di accesso alla cache.
•I cicli di clock utilizzati per un accesso alla cache riuscito (HIT) sono inclusi nei cicli di clock della CPU in esecuzione.


Impatto di una memoria cache

Cicli_clock_stallo = #Accessi_Memoria*MISS_RATE*MISS_PENALTY

$Tempo_{CPU}$ Programma = (#Cicli_clock + #Cicli_clock_stallo)*T_{clock} =
#Istruzioni*CPI_{exec}*T_{clock} + #Cicli_clock_stallo*T_{clock}

CPI_{con_cache} = CPI_{exec} + #Cicli_clock_stallo / #Istruzioni =
CPI_{exec}+(#Accessi_memoria/#Istruzioni)*MISS_RATE*MISS_PENALTY

Caso ideale: (100% HIT, 0%MISS): CPI_{con_cache} = CPI_{exec}

Caso senza cache: (100% MISS): CPI_{senza_cache}=
CPI_{exec} + (#Accessi_memoria/#Istruzioni)* MISS_PENALTY


Esercizio su cache

Si consideri il VAX-11/780. La MISS_PENALTY è di 6 cicli di clock, mentre tutte le istruzioni impiegano 8.5 cicli di clock se si ignorano i MISS (stalli della memoria). Ipotizzando un MISS_RATE dell'11% e che vi siano in media 2 riferimenti alla memoria per ogni istruzione,

⇒ Qual è l'impatto sulle prestazioni quando viene inserita la cache reale rispetto ad una cache ideale?

⇒ Qual è l'impatto sulle prestazioni tra il caso di cache reale e senza inserimento della cache?


Soluzione esercizio su cache

Dati di ingresso: MISS_PENALTY=6 CPI_{exec}=8.5 MISS_RATE=0,11
#Accessi_memoria/#Istruzioni = 2

CPI_{con_cache} = 8,5 + 0,11 * (2 * 6) = 9,82
$CPI_{con\_cache\_ideale}$ = 8,5 + 0 * (2 * 6)
CPI_{senza_cache}= 8,5 + 1 * (2 * 6) = 20,5

Perdita in prestazioni (speed-up): $CPI_{con\_cache\_ideale}$ / CPI_{con_cache} => 8,5 / 9,82 = 0,865

Guadagno in prestazioni (speed-up): CPI_{senza_cache} / CPI_{con_cache} => 20,5 / 9,82 = 2,087


Sommario

Cosa vuol dire valutare le prestazioni

Benchmark

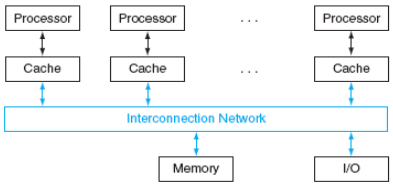
Valutazione delle prestazioni del sistema di memoria

Valutazione delle prestazioni multi-core

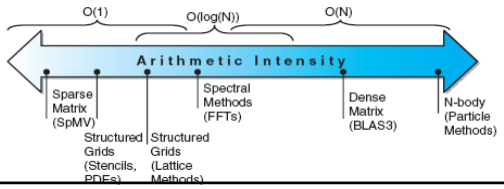
Arithmetic intensity

Velocità di calcolo FLOPS (floating point per second): velocità del singolo core: V_{core}.
In un'achitettura multi-core con N 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} operazioni su ogni byte letto dalla memoria, avremo una velocità di calcolo massima pari a: $V_{max} = V_{mem} * N_{byte}$ = [Byte /s].



Weak scaling, less memory request per byte

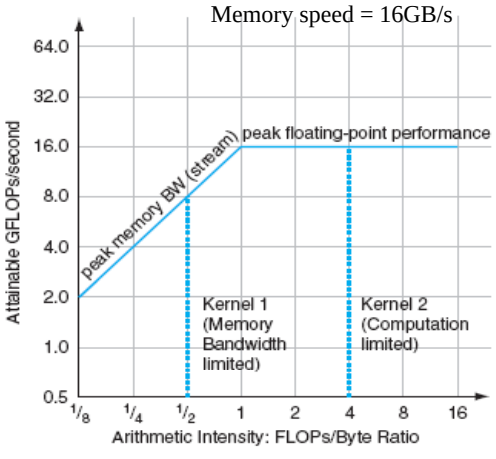
Il modello "roofline"

2 elements:
- Computation
- Memory transfer

Nessun benchmark può essere contenuto interamente in cache.

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

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



AMD Opteron X2

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

Il modello “roofline”: osservazioni

2 elements:
- Computation
- Memory transfer

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

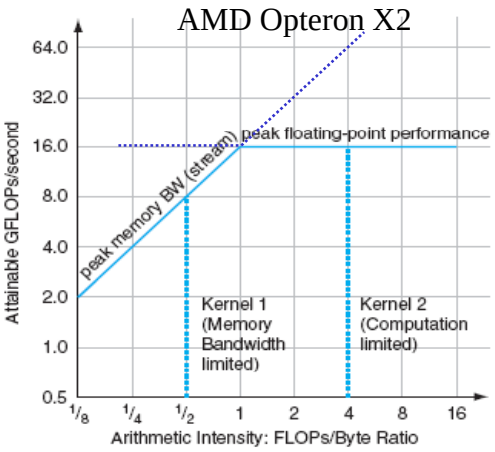
V_{calc} = cost =

16 GFLOPS sull’Opteron X2.

Se tutto quello che viene letto dalla memoria alla massima velocità potesse essere calcolato, si avrebbe una velocità di calcolo pari a:

$V_{calc} = V_{mem} * N_{op} / {}_{Byte}$

V_{mem} = 16GFlop/s

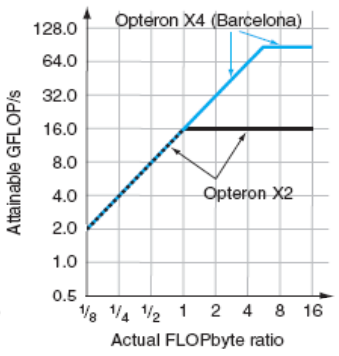


Si disegnano le due curve e le prestazioni sono limitate dalla curva più bassa è sufficiente conoscere l’intensità aritmetica del programma.

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.

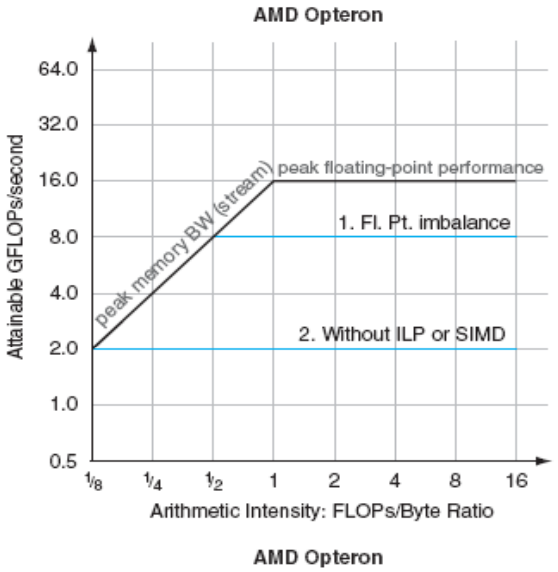
La velcocità di calcolo massima di 128GFLOPS si raggiunge solo per un’intensità aritmetica pari a 5.

Ottimizzazioni sulla parte di calcolo

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

Mix di operazioni:
- Floating point
- Moltiplicazioni e addizioni

Parallelizzazione dell'esecuzione
- Srotolamento dei cicli



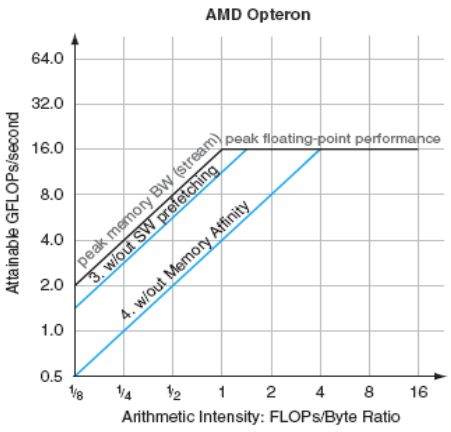
AMD Opteron

Ottimizzazioni sulla parte di memoria

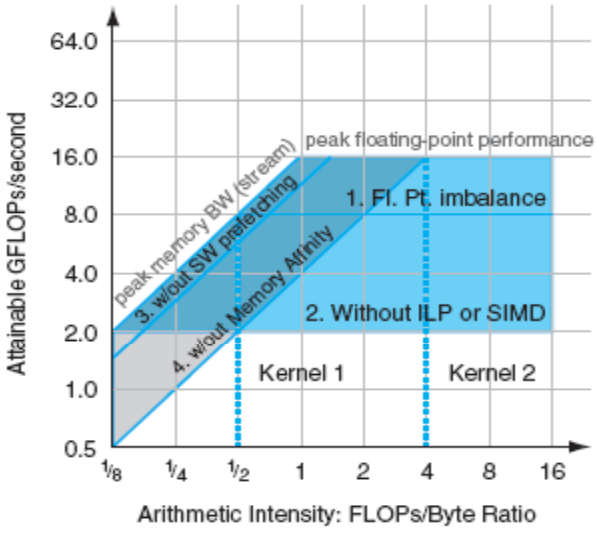
Cosa succede se le prestazioni del vostro programma risultano scadenti rispetto alle prestazioni attese?

Software pre-fetching:
-Precaricamento dei dati in cache

Affinità della memoria:
- Massimizzare gli hit



Quali ottimizzazioni?



Sommario

Cosa vuol dire valutare le prestazioni

Benchmark

Valutazione delle prestazioni del sistema di memoria

Valutazione delle prestazioni multi-core