VALUTAZIONE DELLE PRESTAZIONI

Tempo di risposta, latenza e throughput
Speedup (globale)
Legge di Amdahl
Principio di località
Progettazione e prestazioni
Prestazioni di CPU: Clock Per Instruction
MIPS/MFLOPS
Benckmarks
Esempi

VALUTAZIONE DELLE PRESTAZIONI

Analisi delle prestazioni di un sistema di elaborazione e delle CPU

TEMPO DI RISPOSTA (tempo di esecuzione, latenza): è il ritardo calcolato dall'inizio alla fine di una esecuzione
THROUGHPUT (larghezza di banda)
è la quantità di lavoro fatta nell'unità di tempo

Prestazioni comparate:
x è n% volte più veloce di y
Texec(y)/Texec(x) = 1 + (n/100)

$$n = 100^* \frac{Texec(y) - Texec(x)}{Texec(x)}$$

Esempio Ex. time A = 10, Ex. Time B = 15 A; A è più veloce di B del 50%

$$\text{Speedup (overall)} = \frac{Texec(y)}{Texec(x)}$$

PRESTAZIONI

IN UN SISTEMA DI CALCOLO MIGLIORARE LE PRESTAZIONI SIGNIFICA:

- DIMINUIRE IL TEMPO DI RISPOSTA
- MIGLIORARE IL THROUGHPUT
A volte sono obiettivi in contrasto

Esempio: DALL'87 AL '92 C'E' STATO UN AUMENTO DI PRESTAZIONI ANNUO DEL 54% NEL MERCATO DELLE WORKSTATIONS

Il progetto di una macchina che va 3 volte più veloce di una qualsiasi macchina in commercio (ma il progetto verrà realizzato in 3 anni) avrà successo?

Tx= Ty/(1+n/100)
Tx(1)=Ty/1.54
Tx(2)=Ty/(1.54)^2 = Ty/2,37
Tx(3)=Ty/(1.54)^3 tx(3)=1/3.65
le prestazioni del mercato fra tre anni saranno aumentate di 3.65; non basta

LEGGE DI AMDAHL

SPEEDUP: QUANTO E' MIGLIORATA LA PRESTAZIONE GLOBALE AVENDO MIGLIORATO UNA PARTE DEL SISTEMA?

LEGGE DI AMDAHL:
il miglioramento delle prestazioni dovuto ad un miglioramento di esecuzione è limitato dalla frazione di tempo in cui tale miglioramento si applica

$$Texec(new) = Texec(old) * [(1 - Fenh) + Fenh/Senh]$$

ove **Fenh** è la percentuale di tempo in cui l'incremento di velocità ha luogo e **Senh** è il valore di tale incremento. Da cui:

$$\text{Speedup (overall)} = \frac{1}{(1-Fenh) + \frac{Fenh}{Senh}}$$

ESEMPIO

CPUb è 5 volte più veloce della CPUa ma tale incremento ha effetto solo nel 50% dei casi e il costo della CPUb è 5 volte quello di CPUa.
Si sa inoltre che il costo della CPU è 1/3 del costo globale del computer. Giudichiamo errato l'investimento se il rapporto tra i costi è maggiore dello speedup

$$\text{Speedup} = \frac{1}{(1-0.5) + \frac{0.5}{5}} = 1.67$$

Costo = 2/3 * 1 + 1/3 * 5 = 2.33

L'investimento è errato

PRINCIPIO DI LOCALITA'

Molti programmi esibiscono località temporale e spaziale.

LOCALITA' TEMPORALE: locazioni a cui si è avuto accesso recentemente tendono ad essere riusate nel breve periodo, ovvero avendo acceduto all'indirizzo X all'istante t, esiste una probabilità elevata di accedere ancora a X entro $t+\Delta t$

LOCALITA' SPAZIALE: locazioni con indirizzi vicini tendono ad essere usate nel breve periodo, ovvero avendo acceduto all'indirizzo X all'istante t, esiste una probabilità elevata di accedere a X +/- Δ X entro $t+\Delta t$

Questo vale sia per i dati, sia per le istruzioni:
Ad esempio può accadere che il 10% del codice statico occupa il 90% del tempo di esecuzione

Una architettura, per essere efficiente, deve saper rispondere alle caratteristiche di località della programmazione

PROGETTAZIONE DI CALCOLATORI

studio di un architettura che sopravviva nel tempo ai miglioramenti tecnologici!

IMPATTO DELLA TECNOLOGIA:

1) la densità e la velocità dei chip aumenta del 25% all' anno e, quindi, raddoppia (circa) in 3 anni

Sp(CPU speed)=1.25 Sp(CPU dens)=1.25

2) Densità delle DRAM aumenta del 60% all'anno e, qyuindi, quadruplica in 3 anni, mentre la velocità triplica ogni 10 anni, allora

Sp(DRAM speed)=1.12 Sp(DRAM dens)=1.6

3) I/O: Dischi

Sp(DISK speed)=1.12 Sp(DISK dens)=1.25

IMPATTO DEL SOFTWARE:

necessità di memoria aumenta di 1.5 - 2 all'anno
necessità di bit di indirizzo di 0.5-1 all'anno
abbandono dell'assembler, ma "tenuta" dei linguaggi imperativi (miglioramento dei compilatori)

bilanciamento 1-1-1 MIPS-MBYTE-Mbit/sec

PROGETTAZIONI E PRESTAZIONI

Si ipotizzi che la cache sia 5 volte più veloce della memoria principale e possa essere usata il 90% delle volte……
Conviene usare le cache?

SRAM 20ns, DRAM100ns

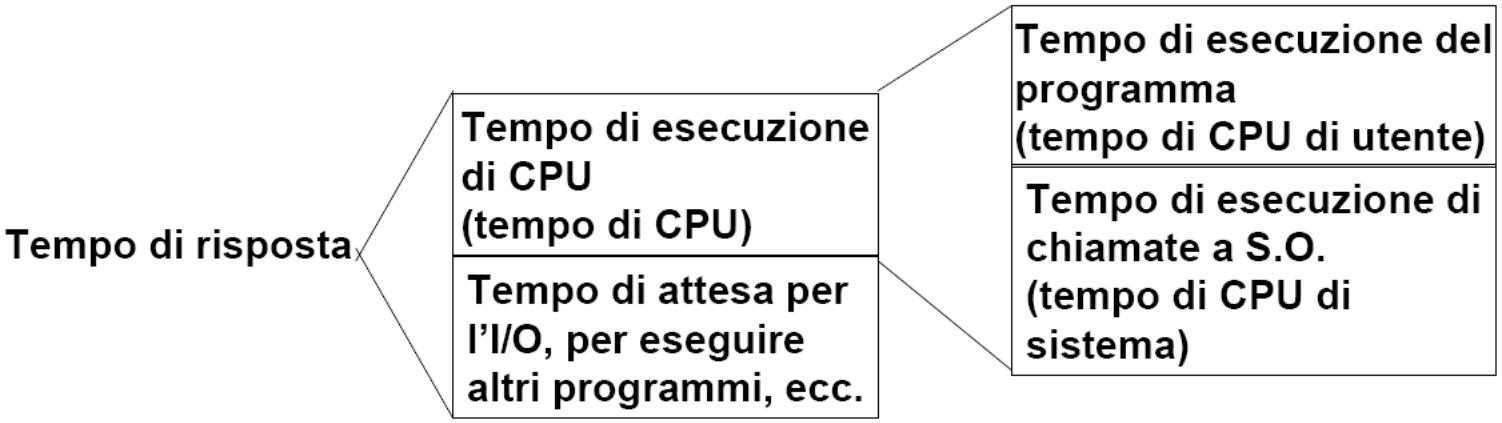
uso di cache al 100% **speedup teorico** di Tnocache/Tcache=5

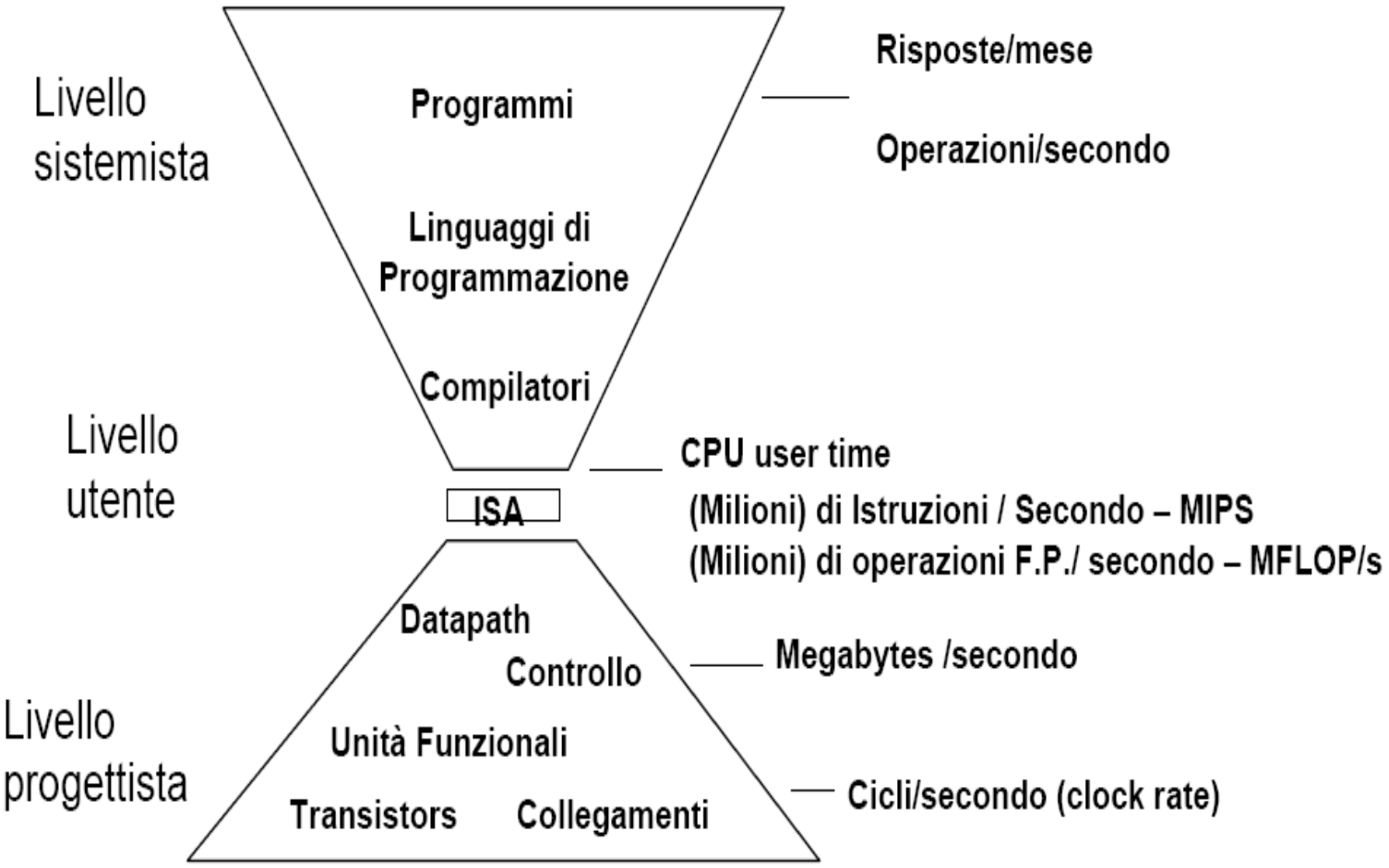
uso le cache al 90%: legge di Amdhal
Sp=1/ ((1-0.9)+0.9/5) = **3.57 Speedup reale**

Supponendo che le DRAM diventino più veloci di 1.12 all'anno (e le SRAM non migliorino)
dopo quanti anni sarebbero diventate competitive?

Sp=3.6 = Told/Tnew= (1 +n/100)^anni (con n=12% quindi n=12)
anni = circa 12 senza contare i miglioramenti architetturali

Quali attività Occupano la CPU





$$\frac{\text{secondi}}{\text{programma}} = \frac{\text{cicli}}{\text{programma}} \times \frac{\text{secondi}}{\text{ciclo}}$$

PRESTAZIONI DI CPU

Tempo Trascorso (ELAPSED TIME):
tempo complessivo speso su un sistema di elaborazione comprendente CPU, I/O, MEM

Esempio *Time* di Unix

Tempo di CPU
Tempo di sistema
Tempo di utente

90.7s | 12.6s | 2:39 64,9%
user | o.s. | elaps di CPU sull'elapsed time

Tcpu= Ncc Tck
Nip= numero istruzioni del programma in exec.
CPI=Ncc/Nip : Clock per instruction

Tcpu = Nip CPI Tck = Sec/Prog
= (Nip/Prog) (Ncc/Nip) (Sec/Ciclo)

Architet. di instr. set **Archit.** **Tecn.**

In che rapporto sono le variabili significative

$$\frac{\text{secondi}}{\text{programma}} = \frac{\text{cicli}}{\text{programma}} \times \frac{\text{secondi}}{\text{ciclo}}$$

Cicli = Cicli i clock = numero di Tclk necessari alla completa esecuzione del programma
Una istruzione quanti cicli è? Dipende dal tipo di istruzione e dalla architettura dell'Instruction Set
Il CPI, è così definito:

$$\text{CPI} = \frac{\text{numero cicli di clock del programma}}{\text{numero istruzioni del programma}}$$

$$\frac{\text{secondi}}{\text{programma}} = \frac{\text{Numero Istruzioni} * \text{CPI}}{\text{Programma}} * \frac{\text{Secondi}}{\text{ciclo}}$$

Numero istruzioni = Nip
Tclk = secondi/ciclo

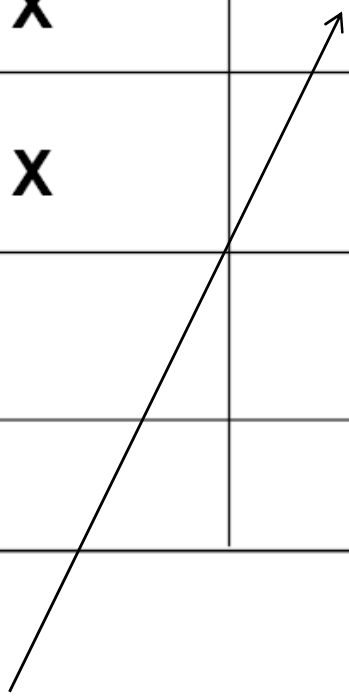
Tcpu = Nip*CPI*Tck = Sec/Prog
= (Nip/Prog)*(Ncc/Nip)*(Sec/Ciclo)

↑ **Architet. di instr. set** ↑ **Archit.** ↑ **Tecn.**

Responsabilità

Su cosa impattano le diverse componenti di un progetto?
ISA = Instruction Set Architecture

	num. istr.	CPI	clock rate
Programma	X		
Compilatore	X	X	
ISA	X	X	X
Organizzaz.		X	X
Tecnologia			X



Dipende dalle ottimizzazioni che esegue nei linguaggi I alto livello, dal C in su.

CPI Clock per instruction

$$\text{CPI} = \frac{\text{Totale dei cicli di clock}}{\text{Numero di istruzioni}}$$

$Ncc = \Sigma (CPIi * Nipi)$ Nipi n. di volte dell'istruzione i-esima

$$CPI = \Sigma \left(CPIi * \frac{Nipi}{N.\ Istruzioni} \right)$$

Esempio:
La CPUa ha due istruzioni separate per if ( )
COMPARE (Ncc=1) e BRANCH (Ncc=2)
CPUb ha soltanto l'istruzione di COMPARE&BRANCH (Ncc=2)
le altre istruzioni richiedono un solo ciclo di clock.
Per la CPUa il 20% sono BRANCH condizionati e il clock della CPUa è del 25% superiore a quello di CPUb. Quale CPU è più veloce ?

CPIa=(0.2*2)+(0.8*1)=1.2
Tcpua= NIPa*1.2 *Tcka
CPUb non esegue compare e il programma è più corto, la CPUb esegue solo l'80% delle istruzioni che esegue CPUa
Nipb=-0.8Nipa
Altra ipotesi: di queste 80 il 25% ha Ncc=2
CPIb=(0.25*2)+(0.75*1)=1.25
Tcpub=Nipb*1,25*Tckb=Nipb*1,25*(1,25*Tcka)
Tcpub= 0.8NIPa*1.25 *(1.25Tcka) = 1.25 NIpa*Tcka
CPUb più lenta
se il clock della CPUa è del 10% superiore a quello di CPUb?

Esempio

Una macchina è caratterizzata dalla seguente distribuzione di CPI su tre classi

Classe	CPI
A	1
B	2
C	3

Una particolare istruzione in un linguaggio di alto livello (HLL) può essere tradotta da un compilatore tramite due sequenze possibili, che usano combinazioni diverse di istruzioni delle tre classi, secondo la tabella seguente:

Sequenza	A	B	C
1	2	1	2
2	4	1	1

Quale delle due è la soluzione migliore in termini di tempo di esecuzione?

CONFRONTO DI PRESTAZIONI

Architettura load/store (es. DLX)

	freq. %	CPI
ALU op	43	1
LOAD letture	21	2
STORE scritt.	12	2
BRANCH	24	2

frequenze calcolate nell'uso del compilatore GCC

il 25% delle ALU op. usano un operando che è stato caricato dalla memoria e non verrà più utilizzato

Architettura reg/mem

si progetta una CPU uguale alla precedente ma con anche l'operazione ALUop' con operando in memoria (reg/mem) che ha un CPI = 2; avendo esteso il set di istruzioni, l'istruzione è più lunga e anche le altre peggiorano: senza aumentare il Tck le istruzioni di branch hanno un CPI=3.
Tale modifica permetterà di usare tutte le volte che serve ALUop' al posto di una ALUop+Load
Questa modifica migliora le prestazioni?

VALUTAZIONI DELLE PRESTAZIONI

Una macchina a 10Mhz con CPI=1 ha 10 MIPS

- MIPS = NI/ Telaps10^6
- MFLOPS = Nopfp/Telaps10^6

MIPS
dipende dal set di istruzioni
dipende dal programma
dipende dai compilatori

MFLOPS
solo operazioni floating point
Spesso vengono indicati gli MFLOPS di picco

	MFLOPS	MEDIO (Bench.)
CRAY X-MP	940	14.8
IBM 3090	800	8.3

BENCHMARKS

Come realizzare dei benchmarks su un calcolatore?

- PROGRAMMI REALI
  (GNU C, SPICE...)
- KERNEL
  (LINPACK, LIVERMORE LOOPS)
- TOY BENCHMARK
  (PUZZLE, QUICKSORT...)
- BENCHMARK SINTETICI
  (WHETSTONE, DHRYSTONE..)

Benchmark SPEC

- SPEC: STANDARD PERFORMANCE EVALUATION CORPORATION
- (1988 Apollo, Sun, MIPS, DEC..) costituito da un mix di applicazioni reali

	SPECint95	SPECfp95
Pentium 133 MHz	3.95	3.53
Pentium II 450 MHz	17.2	12.9
Alpha 833 MHz	49.4	95.6
Pentium III 1 GHz	46.8	32.2

$$\left(\prod_{i=1}^{N} Tref_i / T_i \right)^{1/N}$$

- Media armonica dei rapporti tra i tempi di riferimento e i tempi effettivi di esecuzione di 10 programmi diversi (N=10)
- SPEC CPU95 sostituito da SPEC CPU2000

SPEC 2000

E’ una suite di benchmark studiati per misurare le prestazioni rispetto ad applicazioni di forte peso computazionale. Vengono quindi evidenziati le prestazioni di:

- CPU
- architettura di memoria
- compilatori

Vengono valutate distintamente le prestazioni su operazioni di tipo intero e floating point:

- SPECint2000, SPECfp2000: misurano la velocità di esecuzione
- SPECint_rate2000, SPECfp_rate2000: misurano il throughput.

Benchmark SPEC 2000

SPEC (*System Performance Evaluation Corporation)*

Open Systems Group (OSG):
•SPEC CPU 2000: benchmark per le prestazioni di CPU
•SPEC JBB 2000: benchmark Java server-side.
•SPEC JVM 98: benchmark Java Virtual Machine.
•SPEC MAIL 2001: benchmark mail-server.
•SPEC WEB 99: benchmark per server WWW.
•SPEC SDM 91: benchmark per Multi-user UN*X commands.
•SPEC SFS 97_R1 (3.0): benchmark per File Server.

Graphics Performance Characterization Group (GPC):
•GLperf Results: benchmark per operazioni grafiche OpenGL 2D and 3D. graphics operations (points, lines, triangles, pixels, etc.)
•SPECapc for Pro/ENGINEER Results: Standardized graphics performance tests.

High Performance Group (HPG):
•SPEC HPC96 Results: the SPEC "supercomputer" tests.
•SPEC OMP2001 Results: the SPEC OpenMP/SMP tests.

SPEC int 2000

Nome	*Ref. Time*	*Applicazione*
164.gzip	1400	Data compression utility
175.vpr	1400	FPGA circuit placement and routing
176.gcc	1100	C compiler
181.mcf	1800	Minimum cost network flow solver
186.crafty	1000	Chess program
197.parser	1800	Natural language processing
252.eon	1300	Ray tracing
253.perlbmk	1800	Perl
254.gap	1100	Computational group theory
255.vortex	1900	Object Oriented Database
256.bzip2	1500	Data compression utility
300.twolf	3000	Place and route simulator

SPEC fp 2000

Nome	*Ref. Time*	*Applicazione*
168.wupwise	1600	Quantum chromodynamics
171.Swim	3100	Shallow water modeling
172.mgrid	1800	Multi-grid solver in 3D potential field
173.applu	2100	Parabolic/elliptic partial differential equations
177.mesa	1400	3D Graphics library
178.galgel	2900	Fluid dynamics: analysis of oscillatory instability
179.art	2600	Neural network simulation; adaptive resonance theory
183.equake	1300	Finite element simulation; earthquake modeling
187.facerec	1900	Computer vision: recognizes faces
188.ammp	2200	Computational chemistry
189.lucas	2000	Number theory: primality testing
191.fma3d	2100	Finite element crash simulation
200.sixtrack	1100	Particle accelerator model
301.apsi	2600	Solves problems regarding temperature, wind, velocity and distribution of pollutants