Κεφ. 1: Μετρικά Σύγκρισης Επίδοσης και Χρονοπρογράμματα (Benchmarking)

Κεφ. 1

Εαρινό Εξάμηνο 2017

Metrics

- Computer business is all about metrics and measurements
- What is best: faster, more battery life, better graphics, more reliable, available, cheap, easy to fix
- Metrics…. Performance, Throughput, power, energy, temperature, Availability, MTTF, MTTR, IPC, Cycle Time, Area, TCO, Mispredicts, Misses, Stalls
- Means…. Average, Geometric Mean, Harmonic Mean)
- Probability Distributions… exponential, Weibull, lognormal
- You get the idea ☺: Need to understand what they mean and how to measure them
- Validation key : understand what you measure and report is correct

Αξιολόγηση και Σύγκριση Επίδοσης Υπολογιστικών Συστημάτων

- Χρήσιμη για πολλούς λόγους:
  - Αγοραστή
    - καλή επιλογή
  - Χρήστη
    - βελτίωση συστήματος
  - Προγραμματιστή
    - πιο αποδοτικός κώδικας
  - Κατασκευαστή και Ερευνητή
    - Επίδοση/αξιολόγηση συστήματος
    - Χρησιμότητα καινούργιων ιδεών

Προκλήσεις

- Πολύ μεγάλη ποικιλία στην αγορά υπολογιστών
- Διαφορετικοί στόχοι και προτεραιότητες
  - Κινητοί (smartphones and tablets)
  - Προσωπικοί (desktops):
  - Εξυπηρετητές (servers)
  - Ενσωματωμένοι (embedded)
  - Data Centers
  - Supercomputers
  - Functional Safety and IOT

- Τι είναι τα κατάλληλα μετρικά και προγράμματα σε κάθε περίπτωση

Πως και τί συγκρίνουμε;

- Μέτρηση εκτελεση προγράμματος σε δύο υπολογιστές (real/simulated)
  - configuration (processor, ram, clock, cores, disks, os, compiler, cooling etc)
  - κόστος
- Μέτρο Σύγκρισης
  - Χρόνος εκτέλεσης για ίδια δουλειά - high performance, desktop
  - Διεκπαιρωτικη ικανοτητα - πόσες δουλειές ολοκληρώνονται ανα μονάδα χρόνου (throughput, bandwidth) - servers
  - ενέργεια x latency, ενέργεια x latency^2 – high performance
  - Power – most platforms
  - Θερμοκρασία (peak temperature) – most platforms
  - Battery Life - mobile
  - Αξιοπιστία – πόσο συχνά λάθος, είδος λάθους

Μετρο: Χρονος Εκτέλεσης

- Επίδοση μηχανης Χ στην εκτελεση ενος προγραμματος:

  $\text{Επίδοση}_X = 1 / \text{Χρονος Εκτελεσης}_X$

- Η μηχανη Χ εχει καλυτερη επίδοση απο την Υ στην εκτελεση ενος προγραμματος

$$\text{Επίδοση}_X > \text{Επίδοση}_Y$$

$$\text{Χρονος Εκτελεσης}_X < \text{Χρονος Εκτελεσης}_Y$$

Μετρο: Χρονος Εκτελεσης (συν)

- Η μηχανη Χ ειναι ν φορες γρηγοροτερη απο τη Υ σημαινει:

$$\text{Επίδοση}_X / \text{Επίδοση}_Y = \nu$$

$$\text{Χρονος}_Y / \text{Χρονος}_X = \nu$$

Τι είναι χρονος εκτέλεσης;

- Χρόνος Ανταπόκρισης (response time): ο χρονος που περασε. Αλλα περιλαμβάνει...
  - λειτουργικο συστημα
  - E/E(I/O)
  - συστήματα πολλαπλων χρηστων
- Χρονος ΚΜΕ (CPU time): ο χρονος που ο επεξεργαστης εργαζεται σε ενα προγραμμα
- CPU time = **User time** + System Time

Παραδειγμα

- Στο UNIX χρονος που περασε δινεται απο την εντολη time:

> *time gcc foo.c -o foo*

real 2m39,00

user 1m30,70s

sys 0m12,90s

- Χρονος Χρηστη 90.7s (Χρόνος ΚΜΕ, CPU time)

Συχνότητα, περίοδος, κύκλος μηχανής

- Επεξεργαστες κατεσκευαζονται με ρολοι που τρεχει σε συγκεκριμενη συχνοτητα (frequency cycles/s ή Hz (Hertz))
- Η περιοδος =1/συχνοτητα, ονομαζεται χρονος κυκλου μηχανης (clock cycle time)
- Μια μηχανη με εναν επεξεργαστη που τρεχει στα 2GHz. Τι ειναι ο χρονος κυκλου μηχανης: $1/2GHz=1/2x10^{9}Hz=0.510^{-9}s=0.5ns$
- Ποσους κυκλους μηχανης εχει 1s; ____

Τι επηρεάζει τον χρόνο εκτέλεσης ενός προγράμματος

Χρονος = Εντολες x CPI x Χρονος Κυκλου Μηχανης

CPU Time = I x CPI x Clock Cycle Time

CPU Time = I x CPI / Clock Rate

- Το CPI (ή IPC=1/CPI) επιτρέπει σύγκριση δυο υλοποιήσεων με ίδια αρχιτεκτονική και ρολόι

Static and Dynamic Program Number of Instructions

```
int array_sum(int *a, int n){
        sum = 0;
        for(i=0;i<n;i++)
                sum+=a[i];
        return sum;
}
```

How many lines of C code? How many assembly operations

if n=10. How many operations get executed? How many assembly instructions?

```
// $4 is a, $5 is n
// $? is sum, $? is i
        blez    $5,$L8

        move    $2,$0
        move    $3,$0
$L4:
        lw      $6,0($4)
        addiu   $3,$3,1
        addu    $2,$2,$6
        addiu   $4,$4,4
        bne     $3,$5,$L4

        j       $31     //return
$L8:
        move    $2,$0
        j       $31     // return
```

Τι επηρεαζει την επίδοση (συν)

Κυκλοι Μηχανης (Clock Cycles)

= Αριθμος εντολων που εκτελουνται x

Μεσος Αριθμος Κυκλων Μηχανης ανα Εντολη

= Εντολες x Κυκλοι Ανα Εντολη

= Instructions x Cycles Per Instruction = I x CPI

Τι επηρεαζει την επίδοση

Χρονος ΚΜΕ= Κυκλοι Μηχανης ΚΜΕ για ενα Προγραμμα x
Χρονος Κυκλου Μηχανης

= CPU Clock Cycles x Clock Cycle Time

= **Κυκλοι Μηχανης x Χρονος Κυκλου Μηχανης**

= **Κυκλοι Μηχανης / Συχνοτητα**

Παράδειγμα

Ενα προγραμμα χρειαζεται 10s για να τρεξει στον υπολογιστη Α με συχνοτητα 2GHz.

Ενας σχεδιαστης προτεινει μια καινουργια μηχανη Β, οι οποια θα χρειαστει 1.2 περισσοτερους κυκλους απο την Α αλλα με σημαντικη αυξηση στην συχνοτητα του ρολογιου.

Σε ποια συχνοτητα πρεπει να τρεχει η μηχανη Β για να εχει χρονο εκτελεσης 6s;

Παραδειγμα (συν)

- CPU Time A = Clock Cycles A/ Clock Rate A

  10s = Clock Cycles A /2×10^9 cycles/s

Clock Cycles A = 20×10^9 cycles

CPU Time B = Clock Cycles B / Clock Rate B

6s = 1.2 Clock Cycles A / Clock Rate B

Clock Rate B = 1.2 20×10^9 cycles/6 s = 4 GHz

Παραδειγμα

Δυο υλοποιησεις της ιδιας αρχιτεκτονικης και compiler εχουν για καποιο προγραμμα

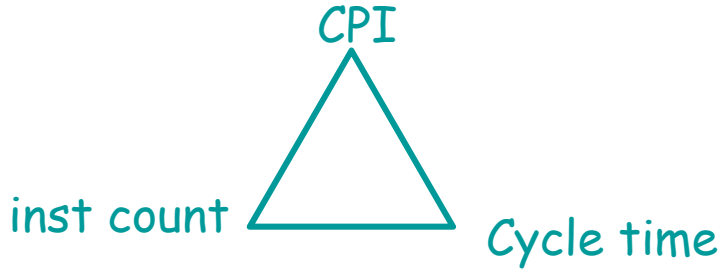
	Κυκλο Ρολογιου	CPI
A	1ns	2
B	2ns	1.2

Ποια μηχανη εχει την καλυτερη επίδοση και ποσο καλυτερη ειναι;

Τι επηρεαζει τις παραμετρους επίδοσης

Time = I x CPI x Clock Cycle Time

- I: ΑΣΕ, μεταγλωτιστης, προγράμμα, data
- CPI: οργανωση/μικροαρχιτεκτονική, ΑΣΕ
- Clock Cycle Time: τεχνολογια, οργάνωση
- Αλληλοεπιδρασεις μεταξυ I, CPI και Clock Cycle Time (συγκρουομενοι στοχοι)



Πως μετρουμε...

- Clock Cycle Time: κατασκευαστής
- Χρονος ΚΜΕ: σύστημα
- CPI: προσομοίωση, μετρητές υλικού
- I: με instrumentation, προσομοίωση, μετρητες υλικου

Example: Calculating CPI bottom up

Base Machine

Op	Freq	Cycles	CPI(i)	(% Time)
ALU	50%	1	.5	(33%)
Load	20%	2	.4	(27%)
Store	10%	2	.2	(13%)
Branch	20%	2	.4	(27%)
			1.5	

Typical Mix of instruction types in program

Παρατηρήσεις

- Χρονος ειναι το μονο αξιοπιστο μετρο συγκρισεως για επίδοση

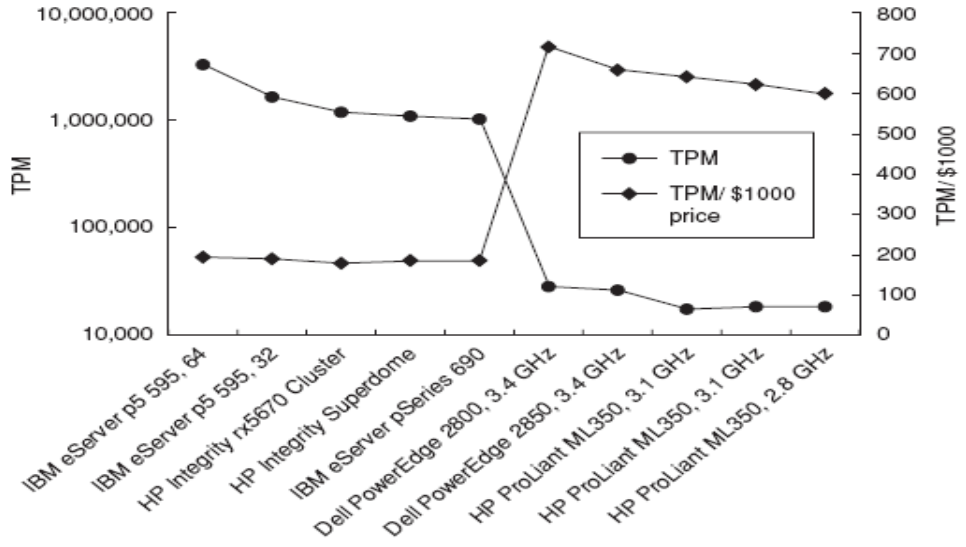
- Όταν συγκρίνουμε εξετάζουμε όλες τις μεταβλητές που επηρεάζονται, πχ
  - Πχ όταν ο χρονος κυκλου μηχανης ειναι ιδιος και έχουμε περισοτερες εντολες αλλα μικροτερο CPI μπορεί να έχουμε καλύτερη επίδοση

Αλλα Μετρα Συγκρισης: BIPS, BFLOPS,GIPS,GFLOPS

- Billion Instructions Per Second (GIPS)
- Προβληματα με το xIPS και τα xFLOPS
  - δεν σχετιζεται με την αρχιτεκτονικη ή οργανωση (δεν μπορει να συγκρινεις διαφορετικες αρχιτεκτονικες ή οργανωσεις)
  - δυνατον να διαφερει αντιστροφως αναλογα με την επίδοση (αν συγκρίνουμε διαφορετικές αρχιτεκτονικές)
- Μέγιστη Επίδοση (σπάνια εφικτή)

Throughput

- Important for Servers
- For Database, Websearch like applications metric Transactions per Minute (TPM)
- Benchmarks TPC, Websearch



Παραδείγματα Servers for TP

Vendor and system	Processors	Memory	Storage	Database/OS	Price
IBM eServer p5 595	64 IBM POWER 5 @1.9 GHz, 36 MB L3	64 cards, 2048 GB	6548 disks 243,236 GB	IBM DB2 UDB 8.2/ IBM AIX 5L V5.3	$16,669,230
IBM eServer p5 595	32 IBM POWER 5 @1.9 GHz, 36 MB L3	32 cards, 1024 GB	3298 disks 112,885 GB	Orcale 10g EE/ IBM AIX 5L V5.3	$8,428,470
HP Integrity rx5670 Cluster	64 Intel Itanium 2 @ 1.5 GHz, 6 MB L3	768 dimms, 768 GB	2195 disks, 93,184 GB	Orcale 10g EE/ Red Hat E Linux AS 3	$6,541,770
HP Integrity Superdome	64 Intel Itanium 2 @ 1.6 GHz, 9 MB L3	512 dimms, 1024 GB	1740 disks, 53,743 GB	MS SQL Server 2005 EE/MS Windows DE 64b	$5,820,285
IBM eServer pSeries 690	32 IBM POWER4+ @ 1.9 GHz, 128 MB L3	4 cards, 1024 GB	1995 disks, 74,098 GB	IBM DB2 UDB 8.1/ IBM AIX 5L V5.2	$5,571,349
Dell PowerEdge 2800	1 Intel Xeon @ 3.4 GHz, 2MB L2	2 dimms, 2.5 GB	76 disks, 2585 GB	MS SQL Server 2000 WE/ MS Windows 2003	$39,340
Dell PowerEdge 2850	1 Intel Xeon @ 3.4 GHz, 1MB L2	2 dimms, 2.5 GB	76 disks, 1400 GB	MS SQL Server 2000 SE/ MS Windows 2003	$40,170
HP ProLiant ML350	1 Intel Xeon @ 3.1 GHz, 0.5MB L2	3 dimms, 2.5 GB	34 disks, 696 GB	MS SQL Server 2000 SE/ MS Windows 2003 SE	$27,827
HP ProLiant ML350	1 Intel Xeon @ 3.1 GHz, 0.5MB L2	4 dimms, 4 GB	35 disks, 692 GB	IBM DB2 UDB EE V8.1/ SUSE Linux ES 9	$29,990
HP ProLiant ML350	1 Intel Xeon @ 2.8 GHz, 0.5MB L2	4 dimms, 3.25 GB	35 disks, 692 GB	IBM DB2 UDB EE V8.1/ MS Windows 2003 SE	$30,600

Trade-off: Throughput vs Response Time

- Some services/applications have tight QoS requirements
  - Response time 99.9% of queries within 300ms
  - 90^{th}: the maximum latency of 90% of the queries when sorted in ascending order
  - 99^{th}: the maximum latency of 99% of the queries when sorted in ascending order
- Increasing throughput on a machine may use idle resources but contention on shared resources hurt response time
- Challenge: configure system for QoS that maximizes efficiency

Power and Energy

- Power one of the main design constraint nowadays
- Most compute platforms have fixed power envelopes
  - Thermal and power delivery reasons

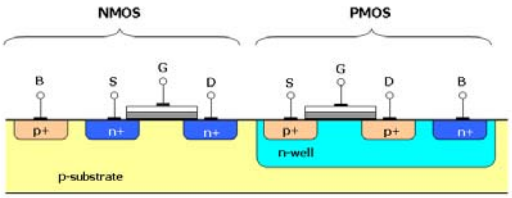
- POWER = ENERGY / TIME
- Units: Watts (W) 1W = 1Joule/1Second
- 1 Joule:
  - The energy required to lift a medium-size tomato (100 g) 1 meter vertically from the surface of the Earth,
  - The typical energy released as heat by a person at rest every 1/60 second
  - Processors: work to move current across the chip

- Can burn same amount of energy fast or slow (higher or lower power)

- Higher power means higher temperature and ability to deliver required power

- Problem: Get power in, get power out from processors and without burning them (or needing exotic/expensive cooling)

Dynamic Energy and

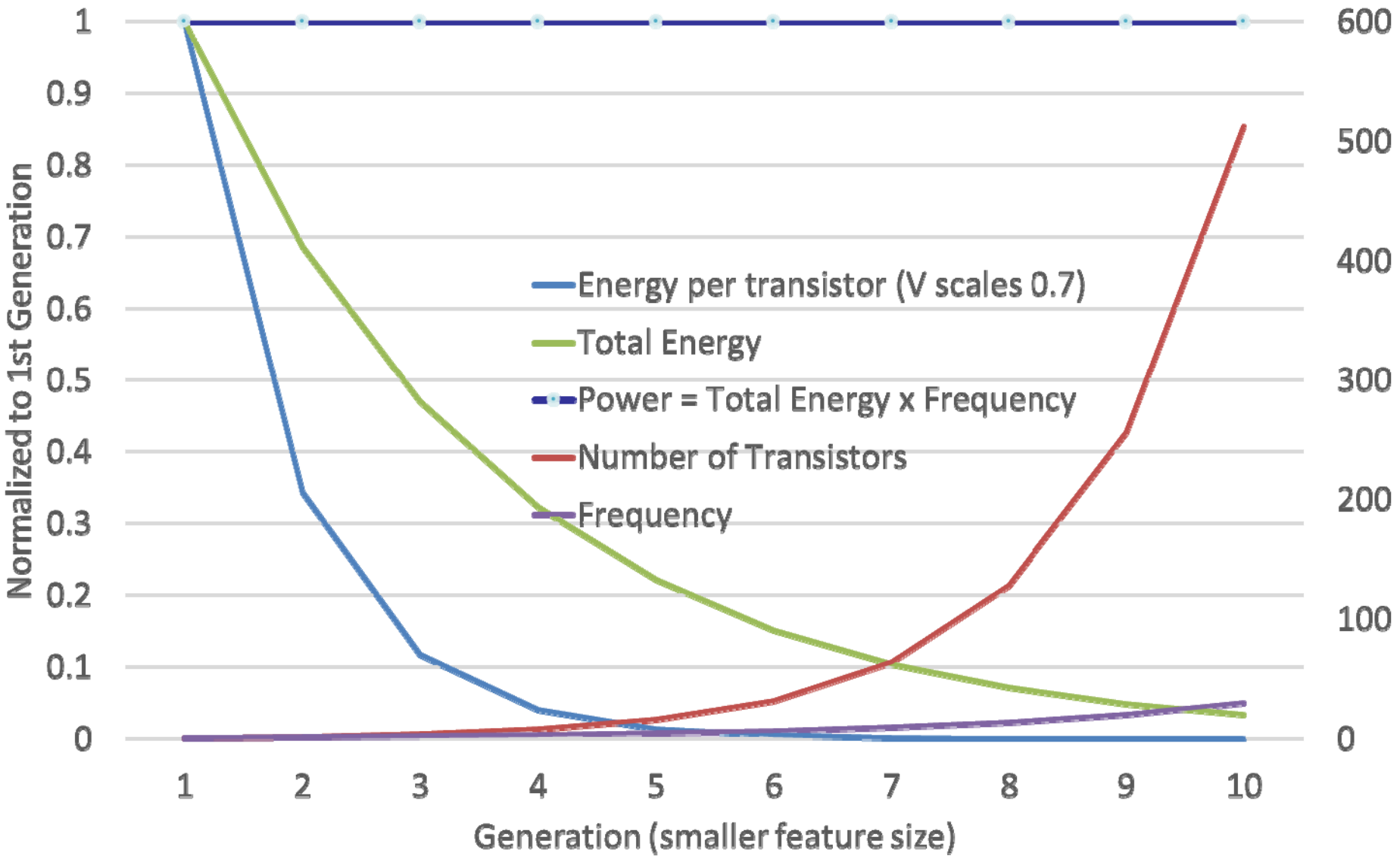


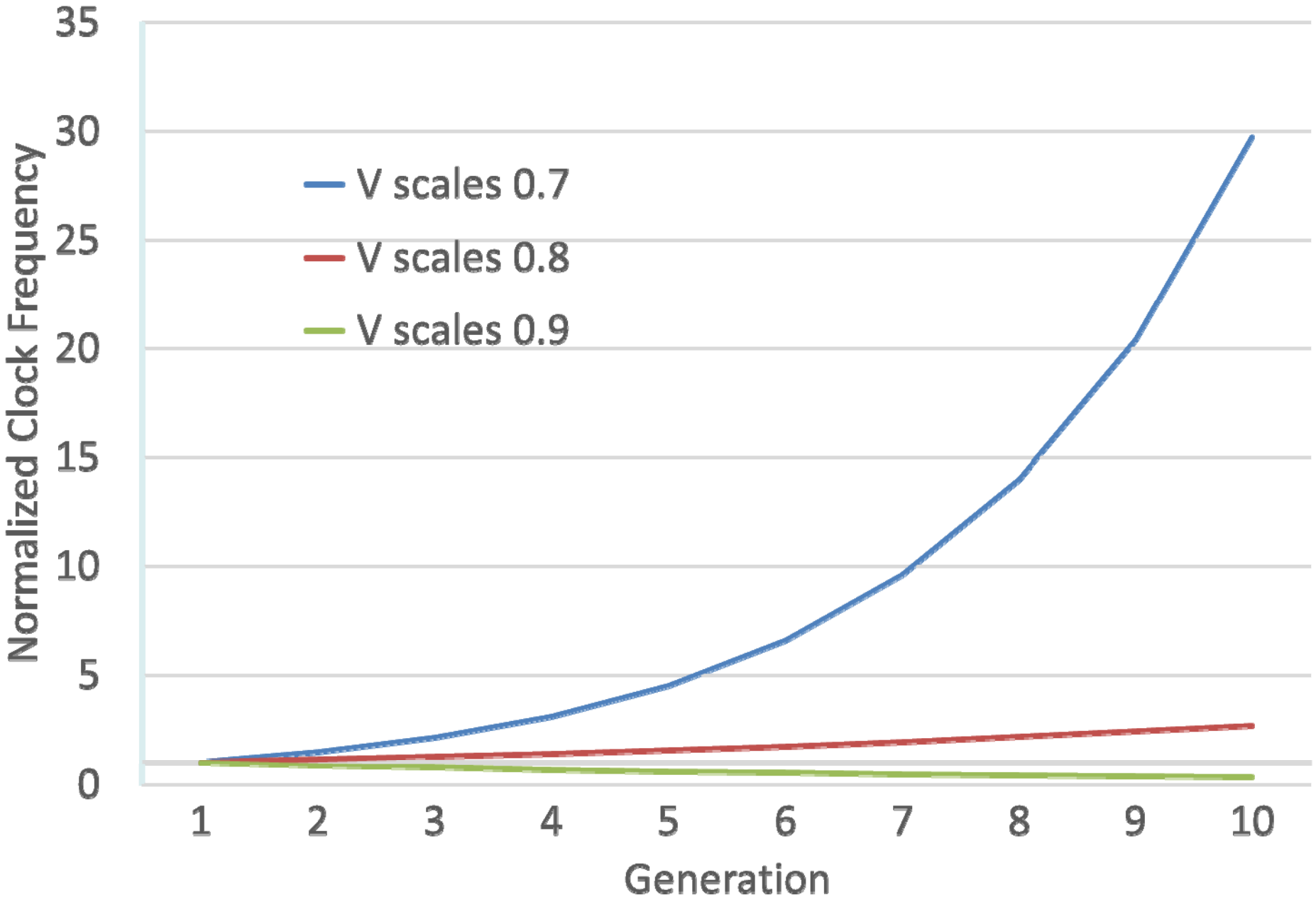
- CMOS Transistors Dynamic energy
  - Transistor switch from 0 -> 1 or 1 -> 0
  - Energy = ½ x Capacitive load x Voltage^2 (½ C x V^2)
  - Capacitive load: depends on number of transistors switching (not all switch!) = α Chip Capacitance
    - α: activity factor
- Dynamic power
  - Power = Energy/Time (rate of energy consumption)
  - ½ x Capacitive load x Voltage^2 / Clock Period
  - ½ x Capacitive load x Voltage^2 x Frequency
- Reducing clock rate reduces power, not energy (the same works gets done but slower)

Technology Scaling (simplified model)

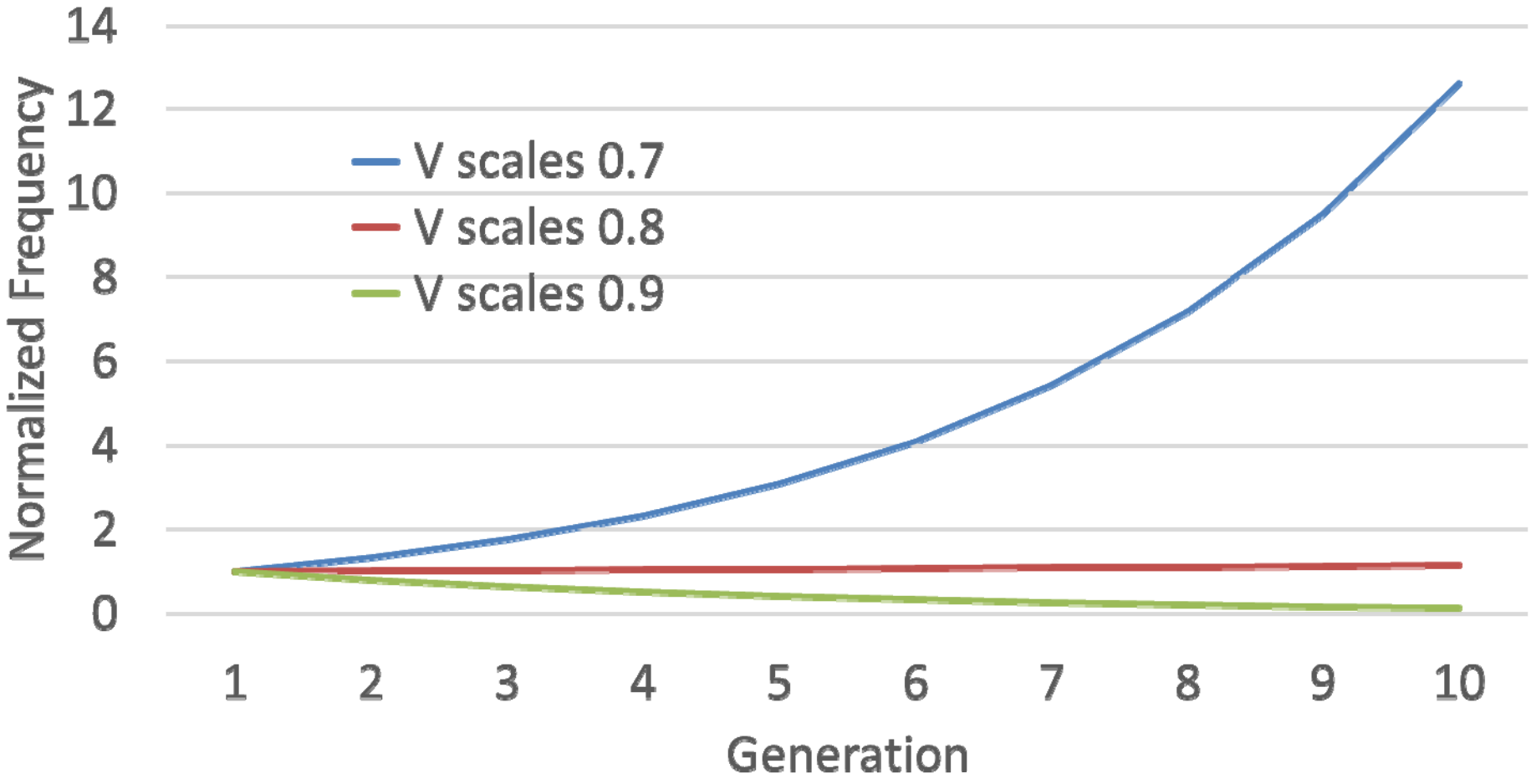
- Number of Active Devices x2
- Capacitance x0.7
- Voltage x0.7

- Device Energy x0.35
  - $0.7 \times (0.7)^2 = 0.35$
- Chip Energy x0.7
  - $2 \times 0.7 \times (0.7)^2 = 0.7$
  - Energy Decreases
  - For same power room options:
    - increase Frequency by 1/0.7=1.4
    - Include more transistors (functionality)
- Unfortunately V stop scaling by 0.7x. Implications?


Normalized to 1st Generation
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
600
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
10
Generation (smaller feature size)
Energy per transistor (V scales 0.7)
Total Energy
Power = Total Energy x Frequency
Number of Transistors
Frequency

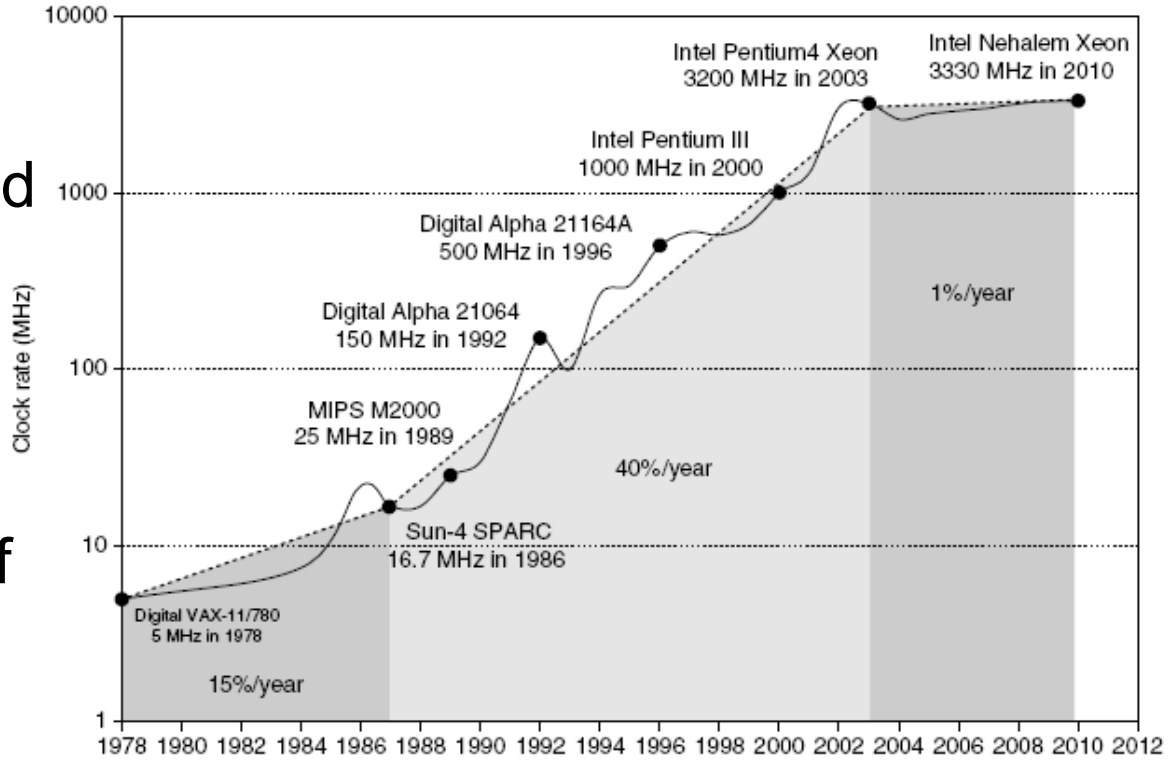

35
30
25
20
15
10
5
0
Normalized Clock Frequency
V scales 0.7
V scales 0.8
V scales 0.9
1
2
3
4
5
6
7
8
9
10
Generation

Die size? Increase by 20 per generation



Power Wall

- Intel 80386 consumed ~ 2 W
- 3.3 GHz Intel Core i7 consumed 130 W
- Heat must be dissipated from 1.5 x 1.5 cm chip
- This is the limit of what can be cooled by air



Static Power

- Power = Dynamic + Static
  - 20-30% Static
- Static is losses due to imperfections
  - also called leakage
- Static power consumption
  - $\text{Current}_{\text{static}}$ x Voltage
  - Scales with number of transistors
- Reduce Static Power
  - Higher supply Voltage ☹ - breaks energy scaling
  - Power gating (no current for inactive parts)

Reducing Power

- Techniques for reducing power:
  - Dynamic Voltage-Frequency Scaling
  - Low power state for DRAM, disks
  - Turning off cores (power gating)
  - Clock Gating
  - Do nothing well
  - Better transistors
  - Choose between power hungry with less area transistors over power efficient but larger area transistors

Performance(Clock Frequency)

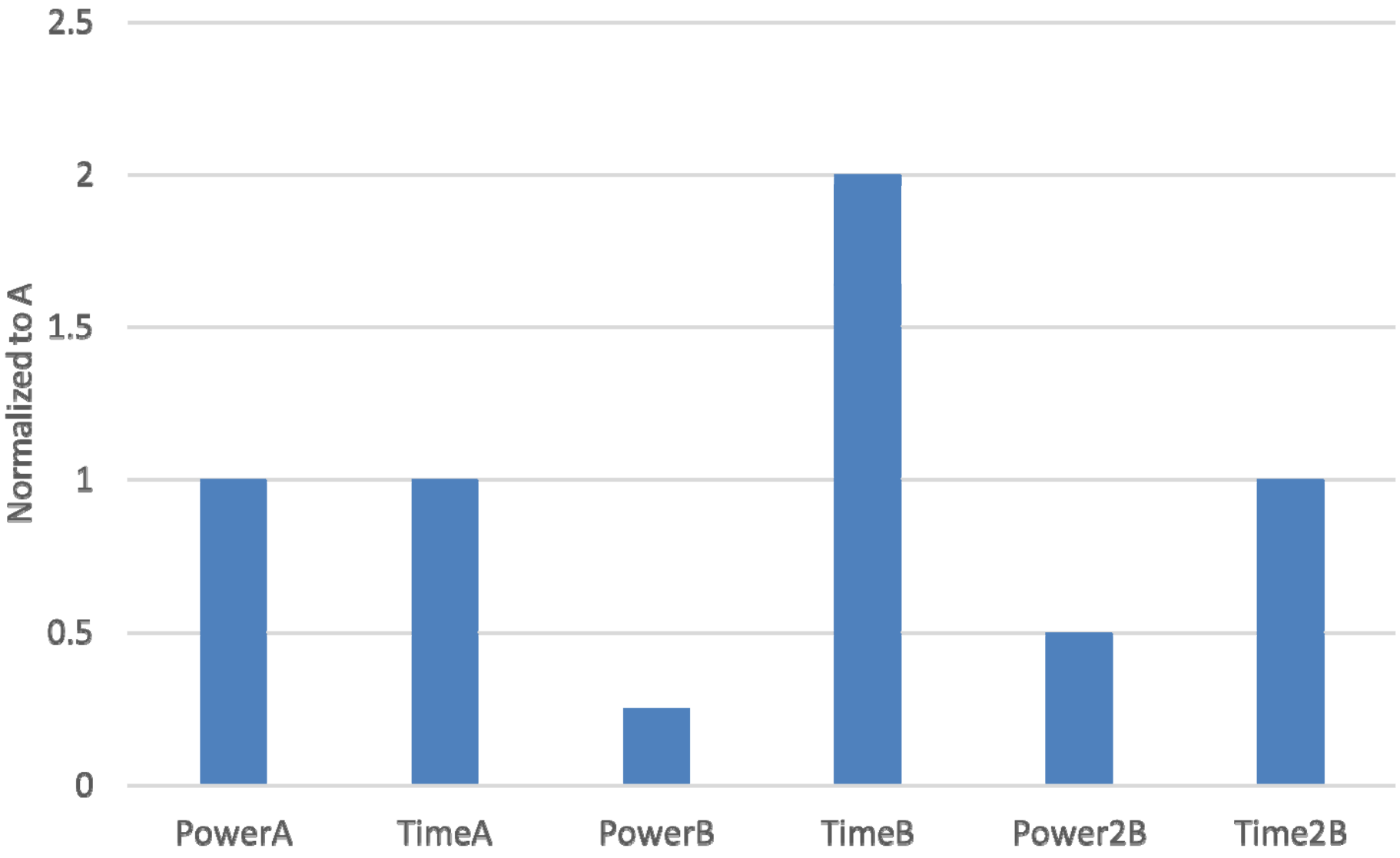
- What happens to the execution time of a program when we lower/increase clock frequency of a processor (DVFS)
  - Execution Time = I . CPI . Cycle Time
- Memory operates at a different clock rate
  - Has its own clock (memory controller)

Performance(Clock Frequency)

- What happens to the execution time of a program when we lower/increase clock frequency of a processor
  - Execution Time =I . CPI . Cycle Time
- Memory operates at a different clock rate
  - Has its own clock (memory controller)
- Tref = $T_{CPU\_REF} + T_{MEM\_REF}$
- $T_{CPU} = T_{CPU\_REF} * F_{new}/F_{ref}$
- $T_{MEM} = T_{MEM\_REF}$

Reducing Power

- **Parallelism:** get same performance with less power.
- Provided program is parallelizable
- Assume F proportional to V (simplified example)
  1. $P = C V^2 F$
  2. $P = 2 (C (V/2)^2 F/2) = C V^2 F /4$
     - use twice real estate
- What is best choice 1 or 2?
- Answer of processor industry 1.x


2.5
2
1.5
1
0.5
0
Normalized to A
PowerA
TimeA
PowerB
TimeB
Power2B
Time2B

Πως μετρούμε Power, Energy, Temperature

- Πραγματικές Μηχανές
  - Μετρητές υλικού παρέχουν δυνατότητα ρύθμισης/παρατήρησης energy, voltage, frequency και της θερμοκρασίας ενός επεξεργαστή (DRAM)
- Υπό μελέτη-κατασκευή
  - με εργαλεία προσομοίωσης (wattch, cacti, hotspot, atmi, mcpat, cad tools)

Μετρικά Απόδοσης Ισχύος (Power Efficiency Metrics)

- Energy – more emphasis on energy ignores performance
  - If you voltage scale then low energy BUT very slow!
  - Energy useful metric if a system does not use voltage scaling
- Energy.Delay –considers performance
  - Ίδιο βάρος σε ενέργεια και χρόνο
- Energy.Delay^2 - more emphasis on performance
  - Περισσότερη έμφαση στην επίδοση
  - If you voltage scale you will pay square on performance
- Energy/Instruction (energy efficiency)

- Παράδειγμα για το ίδιο έργο A: 1W, 1s, B: 2W, 0.75s

	A	B
E(J)	1	1.5
ED(Js)	1	1.125
ED^2(Js^2)	1	0.85

Αξιοπιστία (Dependability)

- Αναξιόπιστα συστήματα στοιχίζουν
  - Έλλειψη εμπιστοσύνης στην αγορά
  - Δυσαρέσκεια με προϊοντα
- Μετρικά
  - MTTF (mean time to failure) in hours
  - FIT (failure in time) = 10^9hrs/MTTF
  - MTTR (mean time to repair)
  - Availability = MTTF/(MTTF+MTTR)
    - 99.999 high target (large banks)
- E.g. server companies build highly dependable systems
  - On error detection retry
  - Error detection correct
  - If permanent error use spare cores, spare memory
  - …

Example calculating reliability

- If modules have *exponentially distributed lifetimes* (age of module does not affect probability of failure) then overall failure rate is the sum of failure rates of the modules
  - **Πιθανότητα λάθους ανά πάσα στιγμή ανεξάρτητή του χρόνου**
  - **Total FIT = Σ FITi**

- Calculate FIT and MTTF for 10 disks (1M hour MTTF per disk), 1 disk controller (0.5M hour MTTF), and 1 power supply (0.2M hour MTTF):

$$FailureRat\ \ e =$$

$$MTTF =$$

Example calculating reliability

- If modules have *exponentially distributed lifetimes* (age of module does not affect probability of failure), overall failure rate is the sum of failure rates of the modules
- Calculate FIT and MTTF for 10 disks (1M hour MTTF per disk), 1 disk controller (0.5M hour MTTF), and 1 power supply (0.2M hour MTTF):

$$
\begin{aligned}
FailureRate &= 10 \times (1/1{,}000{,}000) + 1/500{,}000 + 1/200{,}000 \\
&= (10 + 2 + 5)/1{,}000{,}000 \\
&= 17/1{,}000{,}000 ==> 17\ failures\_in\_1e6hours \\
&= 17{,}000 FIT \\
MTTF &= 1{,}000{,}000{,}000/17{,}000 \\
&\approx 59{,}000 hours \\
59000/(24x365) &=\sim 7\ years
\end{aligned}
$$

Συνοψη και Συγκριση Επίδοσης

- Ενας αριθμος για την περιγραφη της Επίδοσης σε πολλα προγραμματα
- Προταθηκαν και χρησιμοποιουνται διαφοροι μεσοι οροι:
  - *αριθμητικός,*
  - Σταθμισμένος αριθμητικός (weighted arithmetic)
  - γεωμετρικός
  - αρμονικός

Σύνοψη

	Υπολογιστης Α	Υπολογιστης Β
Προγρ. 1(s)	1	10
Προγρ. 2(s)	1000	100
Συν. Χρονος(s)	1001	110
Αριθ. Μεσος	500.5	55

Για το 1 η μηχανη Α ειναι 10 φορες πιο γρηγορη
Για το 2 η μηχανη Β ειναι 10 φορες πιο γρηγορη
Συνολικα η μηχανη Β ειναι 9.1 φορες πιο γρηγορη

Σύνοψη

- Αριθμητικος Μεσος Ορος (Arithmetic Mean) συνοψιζει τον συνολικο χρονο εκτελεσης n προγραμματων

$$AM = \Sigma\ Xronos_i / n$$

- Τι γινεται εαν το προγραμμα 1 ειναι πιο ‘‘σημαντικο’’ απο το 2; Χρηση βαρων για το καθε προγραμμα. Weighted Arithmetic Mean,

$$WAM = \Sigma\ w_i Xronos_i / n$$

Πραγματικα Δεδομενα: SPEC

- Οργανισμός αξιολόγησης με μέλη διάφορες εταιρείες
- Μέτρα:
  - Execution Time Ratio
  - Γεωμετρικος Μεσος (Geometric Mean)
  - Spec Ratio

SPEC: Γεωμετρικος Μεσος

- Για καθε προγραμμα i υπολογισε το execution ratio ER_i=*(χρονος μηχανής αναφοράς / χρονος μηχανη μετρησης)*
- Συνοψισε τα Execution Ratio n προγραμματων με Γεωμετρικο Μεσο
- $\text{SpecRatio} = \sqrt[n]{\prod_{i=1}^{n} ER_i}$

Measure Deviation

- Does a single mean well summarize performance of programs in benchmark suite?
- Can decide if mean a good predictor by characterizing variability of distribution using standard deviation

Measure Deviation

- Does a single mean well summarize performance of programs in benchmark suite?
- Can decide if mean a good predictor by characterizing variability of distribution using standard deviation
- Like geometric mean, geometric standard deviation is multiplicative rather than arithmetic
- Can simply take the logarithm of SPECRatios, compute the standard mean and standard deviation, and then take the exponent to convert back:

$$GeometricMean = \exp\left(\frac{1}{n} \times \sum_{i=1}^{n} \ln\left(SPECRatio_i\right)\right)$$

$$GeometricStDev = \exp\left(StDev\left(\ln\left(SPECRatio_i\right)\right)\right)$$

How Summarize Suite Performance

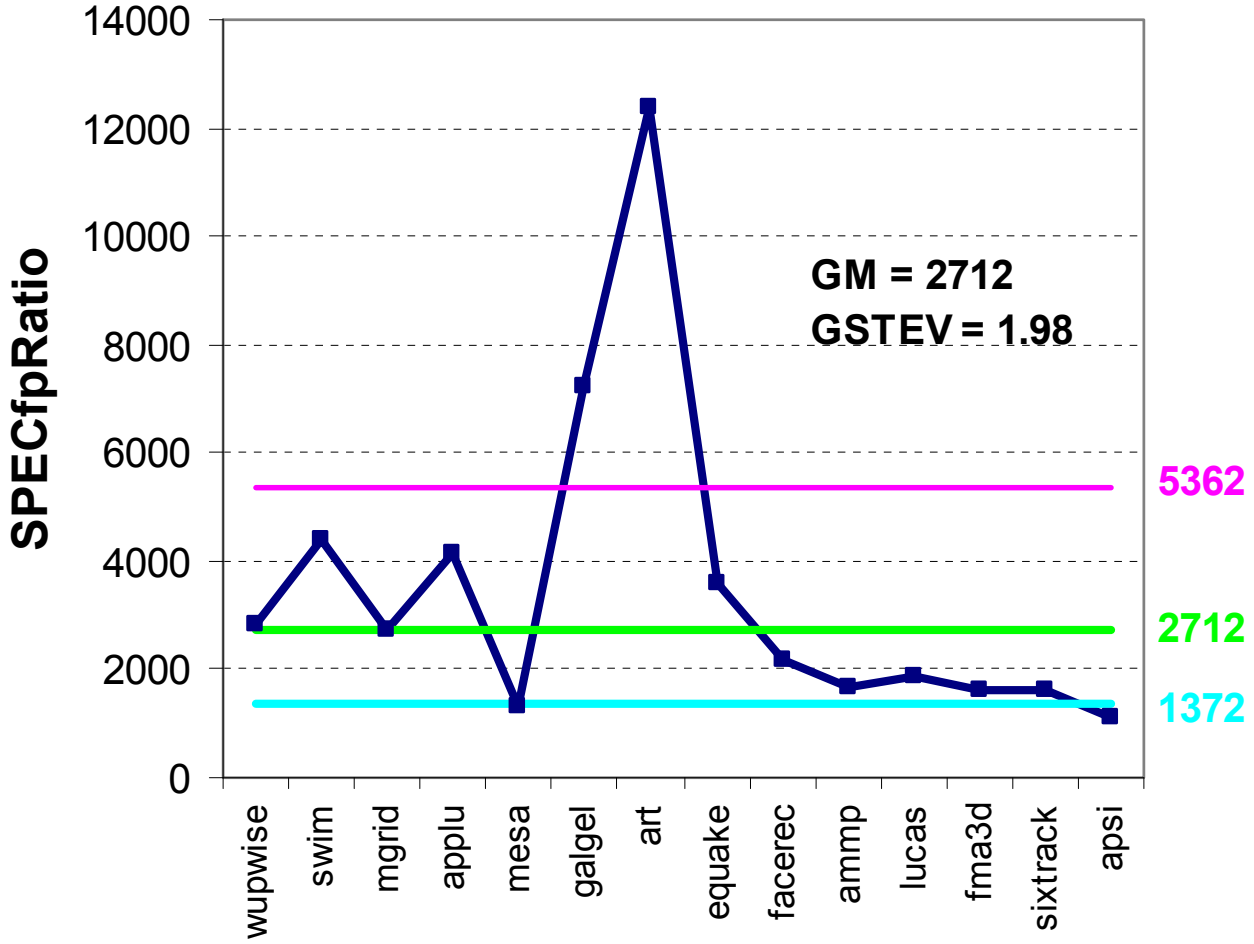
- Standard deviation is more informative if know distribution has a standard form
  - *bell-shaped normal distribution*, whose data are symmetric around mean
  - *lognormal distribution*, where logarithms of data--not data itself--are normally distributed (symmetric) on a logarithmic scale
- For a lognormal distribution, we expect that

68% of samples fall in range $[mean / gstdev, mean \times gstdev]$

95% of samples fall in range $[mean / gstdev^2, mean \times gstdev^2]$

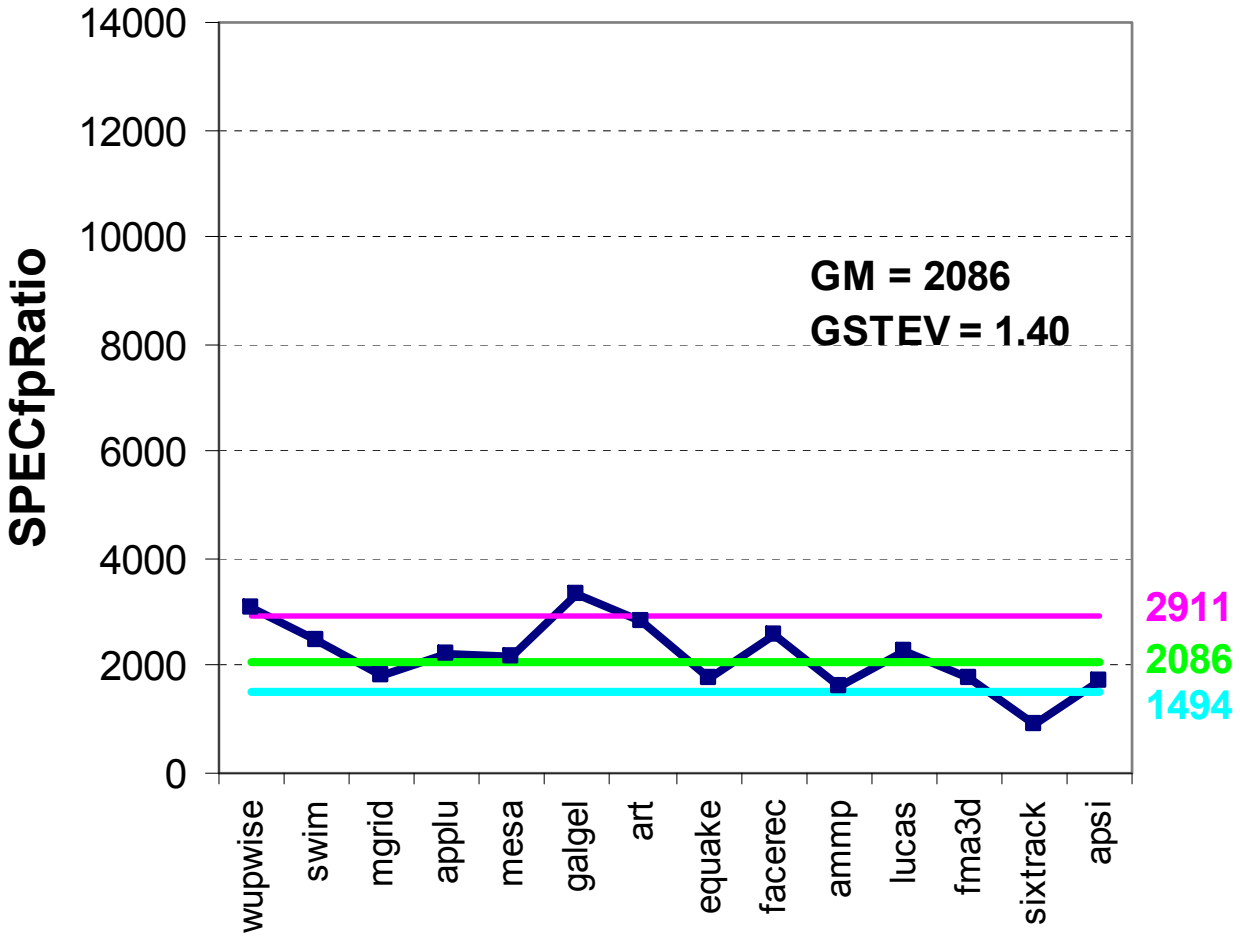
Example Standard Deviation (1/2)

- GM and multiplicative StDev of SPECfp2000 for Itanium 2



Example Standard Deviation (2/2)

- GM and multiplicative StDev of SPECfp2000 for AMD Athlon



Comments on Itanium 2 and Athlon

- Standard deviation of 1.98 for Itanium 2 is much higher-- vs. 1.40--so results will differ more widely from the mean, and therefore are likely less predictable
- Falling within one standard deviation:
  - 10 of 14 benchmarks (71%) for Itanium 2
  - 11 of 14 benchmarks (78%) for Athlon
- Thus, the results are compatible with a lognormal distribution (expect 68%)

Benchmarking

- SPEC οργανισμός για παροχή χρονοπρογραμμάτων σύγκρισης επίδοσης
  - Usually used for desktop ad servers
  - www.spec.org
- EEMBC
  - For embedded processors
  - www.eembc.org

Different Benchmarks

- SPECCPU: popular desktop benchmark suite
  - CPU only, split between integer and floating point programs
  - SPECint2006 has 12 integer, SPECfp2006 has 17 integer prog
  - SPECSFS (NFS file server) and SPECWeb (WebServer) added as server benchmarks
- Transaction Processing Council measures server performance and cost-performance for databases
  - TPC-C Complex query for Online Transaction Processing
  - TPC-H models ad hoc decision support
  - TPC-W a transactional web benchmark
  - TPC-App application server and web services benchmark

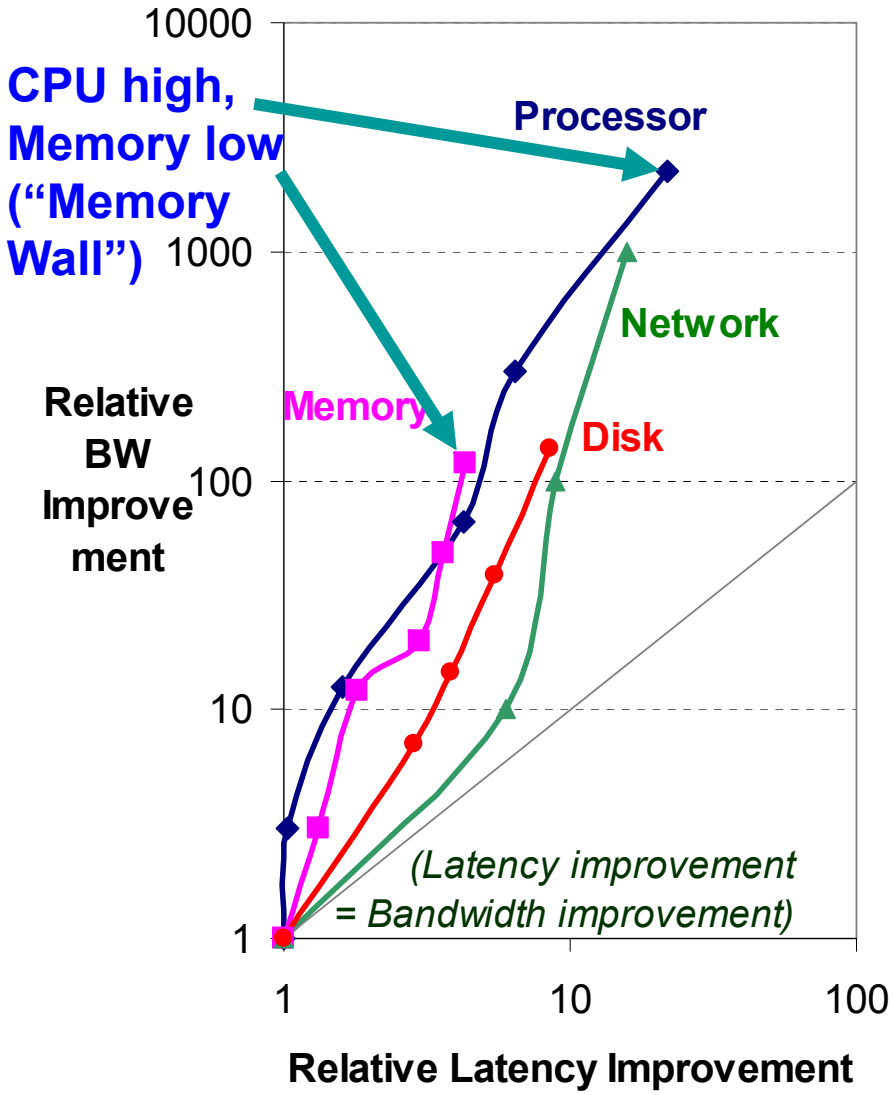
SPEC2006

GNU C compiler ←
Interpreted string processing ←
Combinatorial optimization ←
Block-sorting compression ←
Go game (AI)
Video compression
Games/path finding
Search gene sequence
Quantum computer simulation
Discrete event simulation library
Chess game (AI)
XML parsing

CFD/blast waves
Numerical relativity
Finite element code
Differential equation solver framework
Quantum chemistry
EM solver (freq/time domain)
Scalable molecular dynamics (~NAMD)
Lattice Boltzman method (fluid/air flow)
Large eddie simulation/turbulent CFD
Lattice quantum chromodynamics
Molecular dynamics
Image ray tracing
Spare linear algebra
Speech recognition
Quantum chemistry/object oriented
Weather research and forecasting
Magneto hydrodynamics (astrophysics)

…και κάποιες σημαντικές αρχές/τάσεις

Latency Lags Bandwidth



Rule of Thumb for Latency Lagging BW

- In the time that bandwidth doubles, latency improves by no more than a factor of 1.2 to 1.4

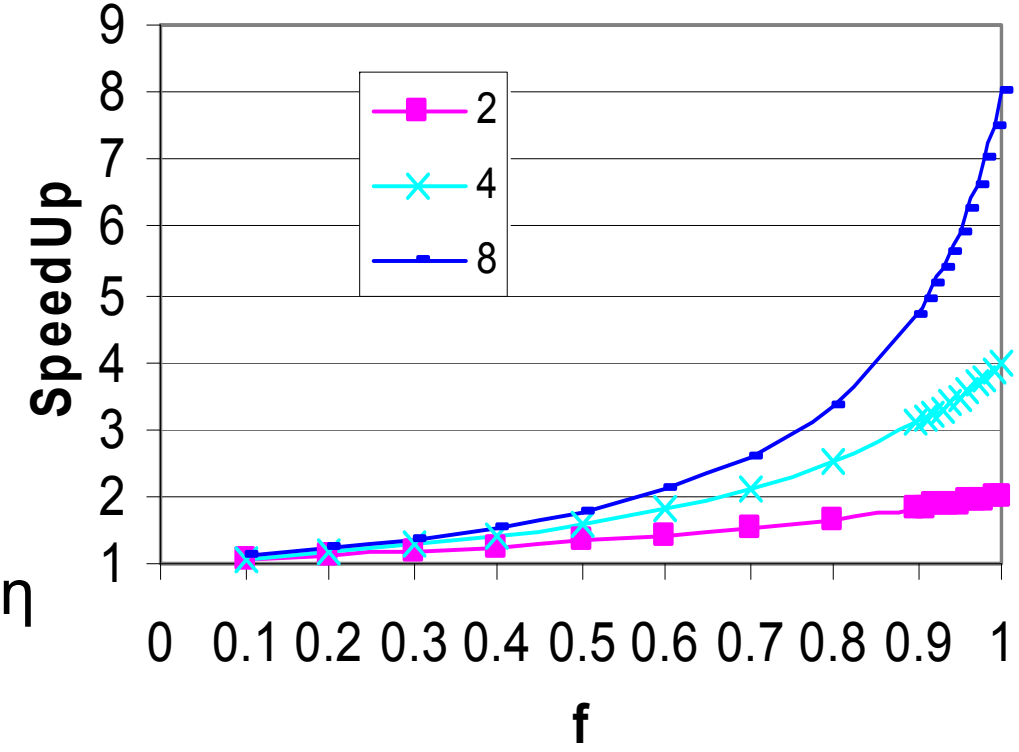
- Bandwidth improves by more than the square of the improvement in Latency

Focus on the Common Case

- Common sense guides computer design
  - E.g., Instruction fetch and decode unit used more frequently than multiplier, so optimize it 1st
  - E.g., If database server has 50 disks / processor, storage dependability dominates system dependability, so optimize it 1st
- Frequent case is often simpler and can be done faster than the infrequent case
  - E.g., overflow is rare when adding 2 numbers, so improve performance by optimizing more common case of no overflow
  - May slow down overflow, but overall performance improved by optimizing for the normal case
- What is frequent case and how much performance improved by making case faster => Amdahl's Law

Amdhal's Law

- SpeedUp = 1/(1-f + f/P)
- f: fraction of program time that is improved
- P: factor of improvement



- Μην βελτιστοποιήσετε ένα μηχανισμό αν δεν είναι χρήσιμος συχνά
- Κάνετε την συχνή περίπτωση γρήγορη

Κόστος Επεξεργαστή

- Το κόστος ενός επεξεργαστή επηρεάζεται από το yield
  - Το ποσοστό των chips που κατασκευάζονται και δεν έχουν κατασκευστικά λάθη
  - Επηρεάζεται από το μέγεθος του chip και του wafer
  - Δες τε βιβλίο/HW