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

Κεφ. 1

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

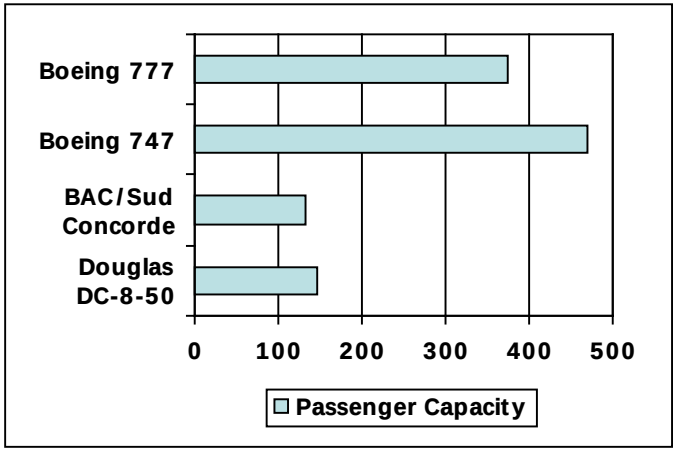
Προκλήσεις

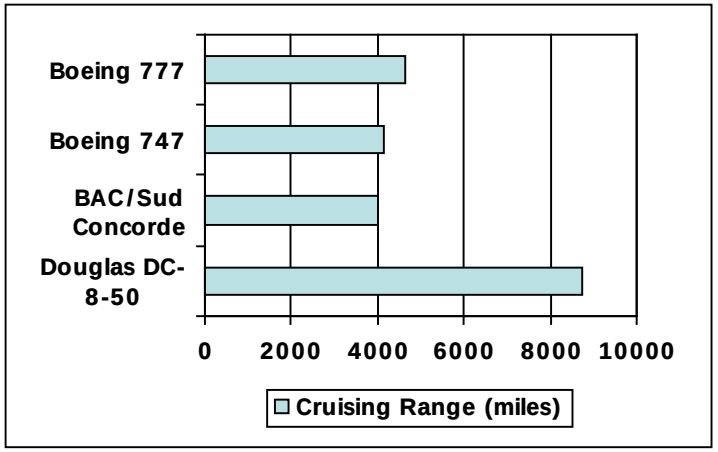
- Πολύ μεγάλη ποικιλία στην αγορά υπολογιστών
- Διαφορετικοί στόχοι και προτεραιότητες
  - Κινητοί (smartphones and tablets)
  - Προσωπικοί (desktops):
  - Εξυπηρετητές (servers)
  - Ενσωματωμένοι (embedded)
  - Data Centers
  - Supercomputers
  - Functional Safety and IOT

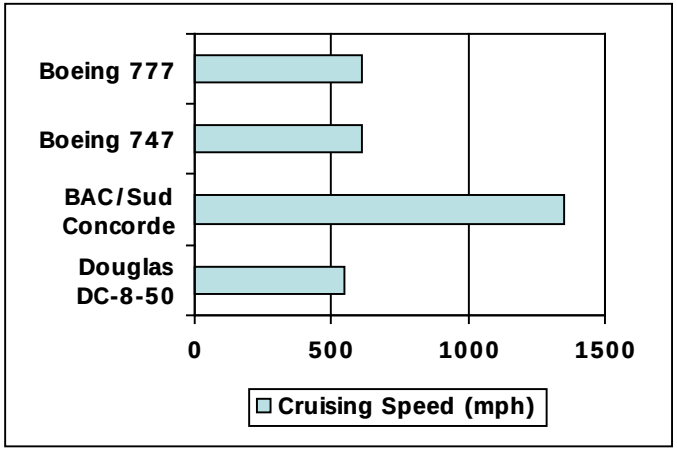
- Τι είναι τα κατάλληλα μετρικά και προγράμματα σε κάθε περίπτωση

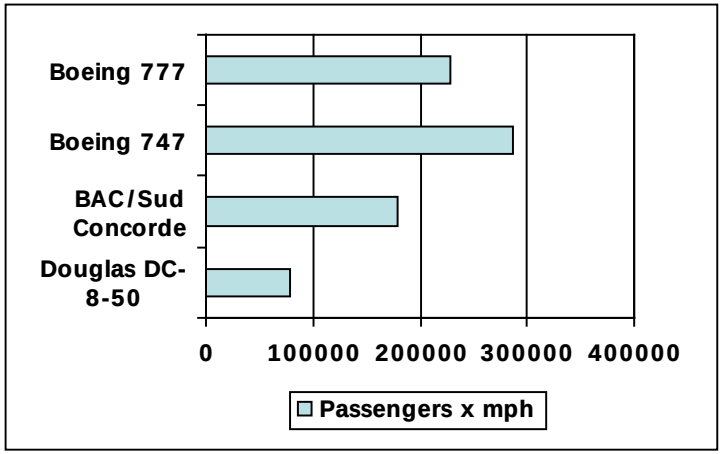
Defining Performance

- Which airplane has the best performance?









Response Time and Throughput

- Response time
  - How long it takes to do a task
- Throughput
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour
- How are response time and throughput affected by
  - A faster processor?
  - More processors?
- We'll focus on response time for now…

Πως συγκρίνουμε;

- Μέτρηση εκτέλεσης προγράμματος σε δύο υπολογιστές (real/simulated)
- Same configuration except (usually) one difference
  - processor, ram, clock, cores, disks, os, compiler, cooling etc
- For example: sensitivity to L1 data cache size
  - 16KB vs 64KB

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
  - Ε/Ε(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

- CPU time 103.6s (Χρόνος ΚΜΕ, CPU time)

Τι επηρεάζει τον χρόνο εκτέλεσης ενός προγράμματος

Χρόνος = Αριθμός Κύκλων x Χρόνος Κύκλου Μηχανής
= Κυκλοι Μηχανης / Συχνοτητα

Χρόνος = Εντολες x CPI x Χρονος Κυκλου Μηχανης

CPU Time = I x CPI x Clock Cycle Time

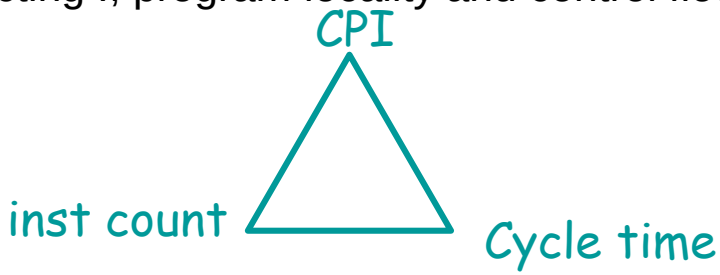
CPU Time = I x CPI / Clock Rate

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

Τι επηρεαζει τις παραμετρους επίδοσης

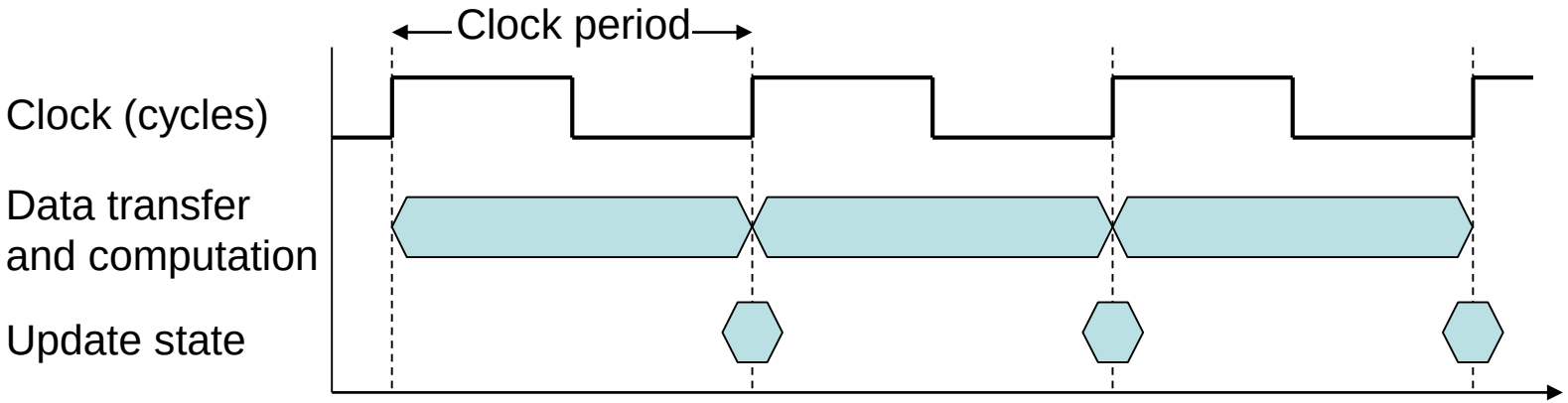
Time = I x CPI x Clock Cycle Time

- I: Instruction Count for a program
  - Determined by algorithm, programming language, compiler, ISA, input data
- CPI: Average cycles per instruction
  - Determined by CPU hardware and program structure
  - Typically different instructions have different CPI
    - Average CPI affected by instruction mix (influenced by parameters affecting I, program locality and control flow predictability)



CPU Clock Cycle Time

- Operation of digital hardware governed by a constant-rate clock



  - Clock period: duration of a clock cycle
    - e.g., 250ps = 0.25ns = 250×10^{-12}s
  - Clock frequency (rate): cycles per second
    - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

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 4GHz. Τι ειναι ο χρονος κυκλου μηχανης: $1/(4GHz)=0.25 \times 10^{-9}s=0.25ns$
- Ποσους κυκλους μηχανης εχει 1s; ____
- Τι καθορίζει το κύκλο μηχανής: οργάνωση, τεχνολογία υλοποίησης, power and thermal constraints

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 /$2x10^9$ cycles/s

Clock Cycles A = $20x10^9$ cycles

CPU Time B = Clock Cycles B / Clock Rate B

6s = 1.2 Clock Cycles A / Clock Rate B

Clock Rate B = 1.2 $20x10^9$ 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;

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

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;
}
```

STATIC:
How many lines of C code? How many assembly instructions?

DYNAMIC:
if n=10. How many instructions 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
```

CPI in More Detail

- If different instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

  - Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\text{Relative frequency}} \right)$$

Example: Calculating CPI

Base Machine

Op	Freq	CPI	WCPI(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

Πως μετρουμε...

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

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 CPU

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

SPEC CPU: 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
- $\text{SpecRatio} = \sqrt[n]{\prod_{i=1}^{n} ER_i}$

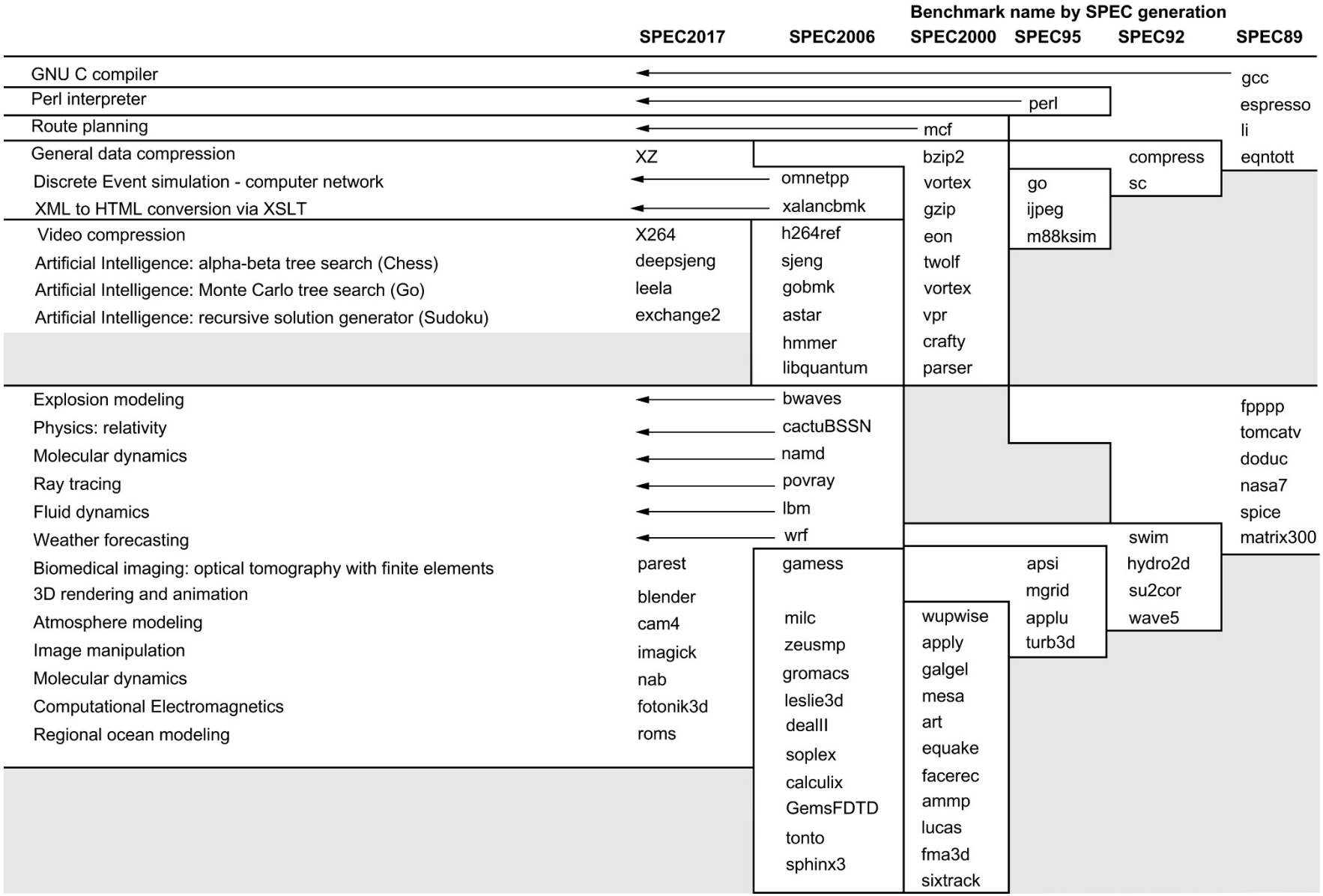
CINT2006 for Intel Core i7 920

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	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

Category	Name	Measures performance of
Cloud	Cloud_IaaS 2016	Cloud using NoSQL database transaction and K-Means clustering using map/reduce
CPU	CPU2017	Compute-intensive integer and floating-point workloads
Graphics and workstation performance	SPECviewperf® 12	3D graphics in systems running OpenGL and Direct X
	SPECwpc V2.0	Workstations running professional apps under the Windows OS
	SPECapcSM for 3ds Max 2015™	3D graphics running the proprietary Autodesk 3ds Max 2015 app
	SPECapcSM for Maya® 2012	3D graphics running the proprietary Autodesk 3ds Max 2012 app
	SPECapcSM for PTC Creo 3.0	3D graphics running the proprietary PTC Creo 3.0 app
	SPECapcSM for Siemens NX 9.0 and 10.0	3D graphics running the proprietary Siemens NX 9.0 or 10.0 app
	SPECapcSM for SolidWorks 2015	3D graphics of systems running the proprietary SolidWorks 2015 CAD/CAM app
High performance computing	ACCEL	Accelerator and host CPU running parallel applications using OpenCL and OpenACC
	MPI2007	MPI-parallel, floating-point, compute-intensive programs running on clusters and SMPs
	OMP2012	Parallel apps running OpenMP
Java client/server	SPECjbb2015	Java servers
Power	SPECpower_ssj2008	Power of volume server class computers running SPECjbb2015
Solution File Server (SFS)	SFS2014	File server throughput and response time
	SPECsfs2008	File servers utilizing the NFSv3 and CIFS protocols
Virtualization	SPECvirt_sc2013	Datacenter servers used in virtualized server consolidation

Active benchmarks from SPEC as of 2017.

SPEC CPU 2017



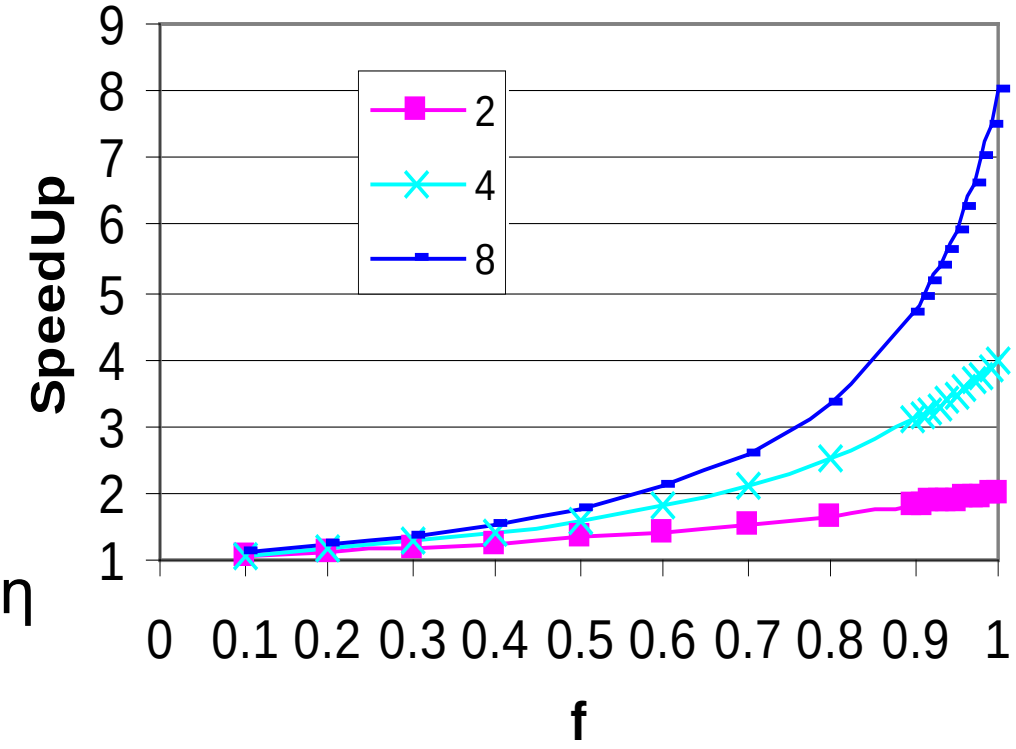
Some tips…

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
- 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

Little's Law

- $Q = \lambda . T$

- Q: how many wait on average
- λ: rate that tasks arrive (on average)
- t: average time it takes to service a task

- E.g. λ = 5tasks/s, t = 0.400ms => Q = 2

Measure and Report 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
  - Narrow deviation means mean is more representative

Αξιοπιστία (Dependability)

- Αναξιόπιστα συστήματα στοιχίζουν
  - Έλλειψη εμπιστοσύνης στην αγορά
  - Δυσαρέσκεια με προϊοντα
- Types of Failures
  - DUE: detected uncorrected errors (lead to crash, unavailability)
  - SDC: silent data corruption (wrong output without knowing about it)
- 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):

$$FailureRate =$$

$$MTTF =$$

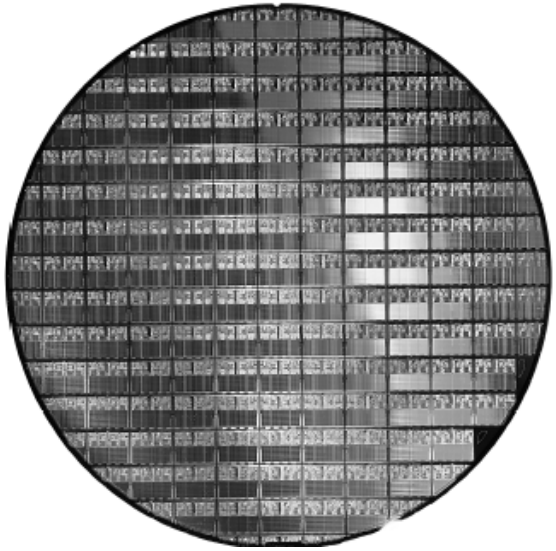
Example calculating reliability

- 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):

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

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



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

Σύνοψη

- 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

Σύνοψη

- 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

Τί συγκρίνουμε;

- 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

Power and Energy

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

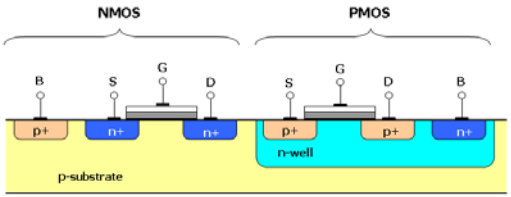
- POWER = ENERGY / TIME
- Units: Watts (W) 1W = 1Joule/1Second
- 1 Joule:
  - The energy required to lift a medium-size tomato (100 g) 1 meter vertically from the surface of the Earth,
  - The typical energy released as heat by a person at rest every 1/60 second
  - Processors: work to move current across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

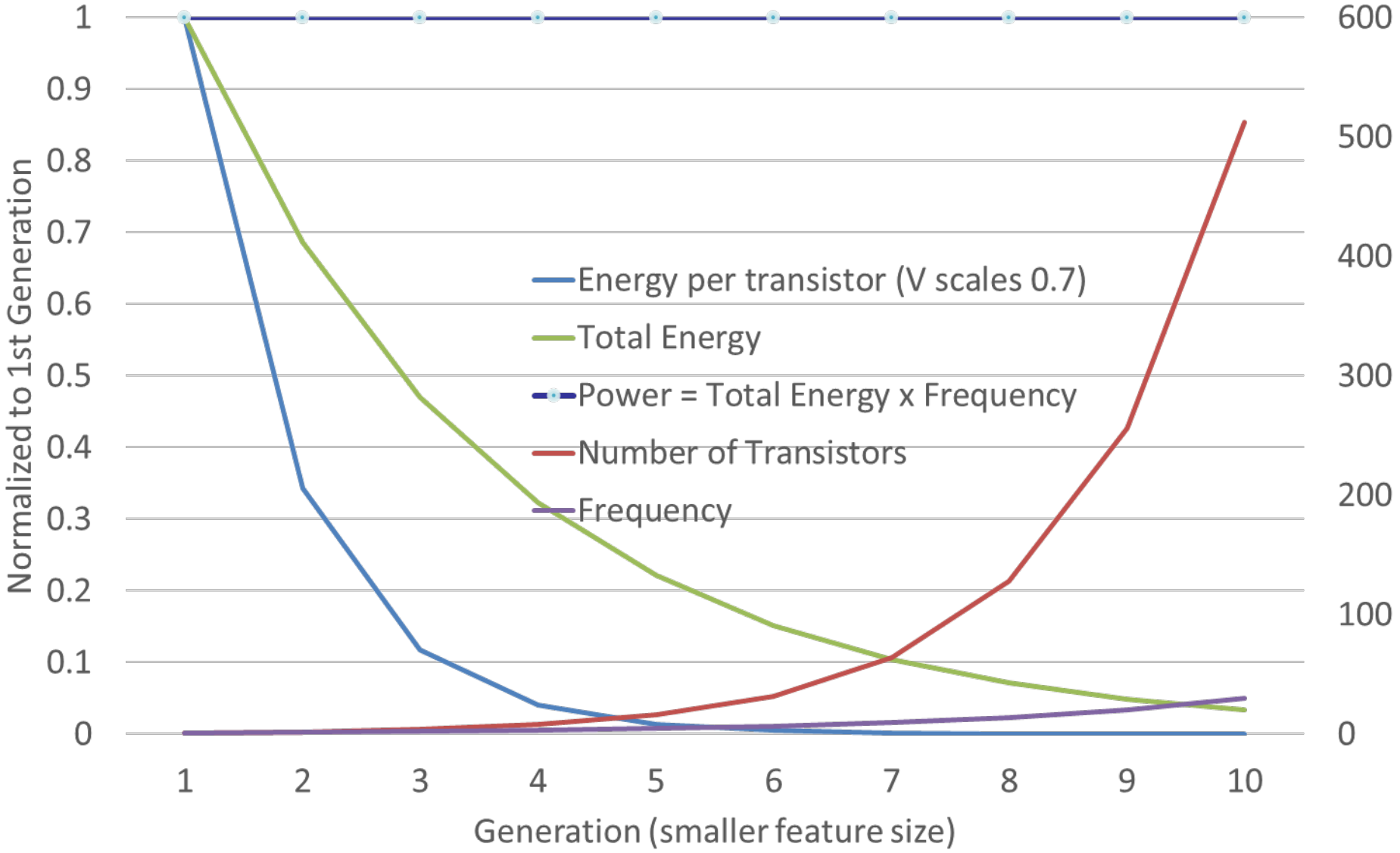


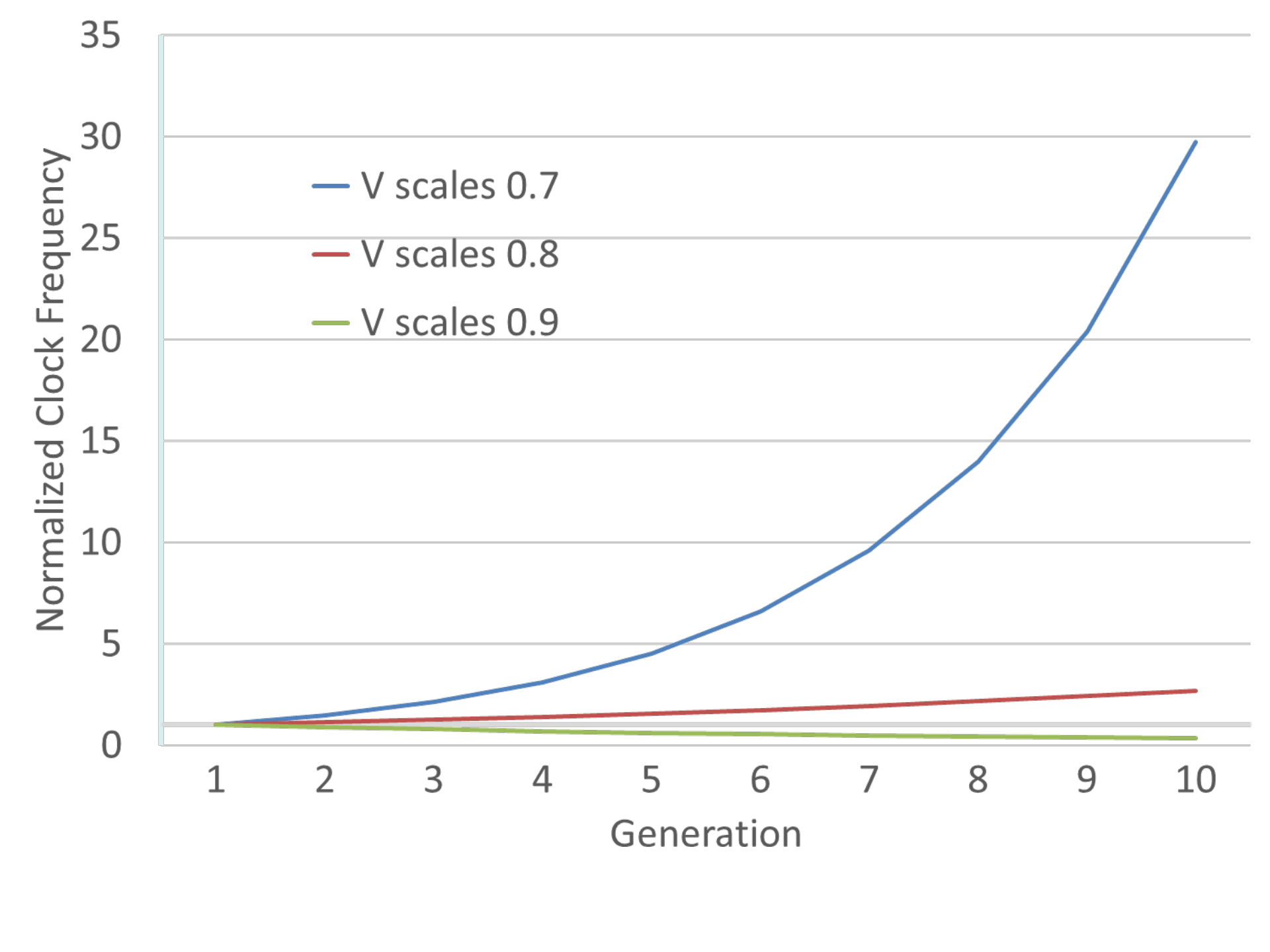
- CMOS Transistors Dynamic energy
  - Transistor switch from 0 -> 1 or 1 -> 0
  - Energy = ½ x Capacitive load x Voltage^2 (½ C x V^2)
  - Capacitive load: depends on number of transistors switching (not all switch!) = α Chip Capacitance
    - α: activity factor
- Dynamic power
  - Power = Energy/Time (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)

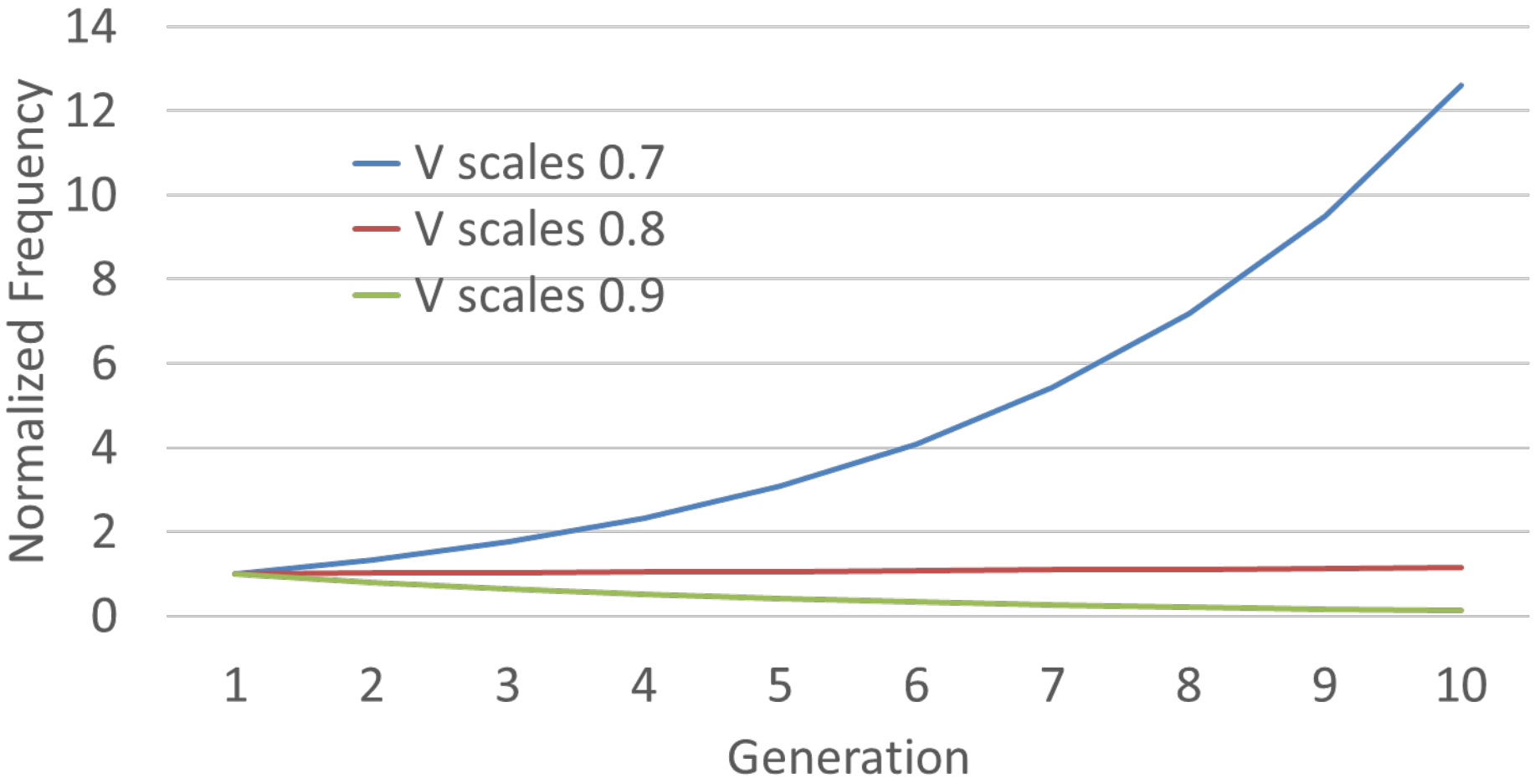
- Number of Active Devices x2
- Capacitance x0.7
- Voltage x0.7

- 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?


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Normalized to 1st Generation
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

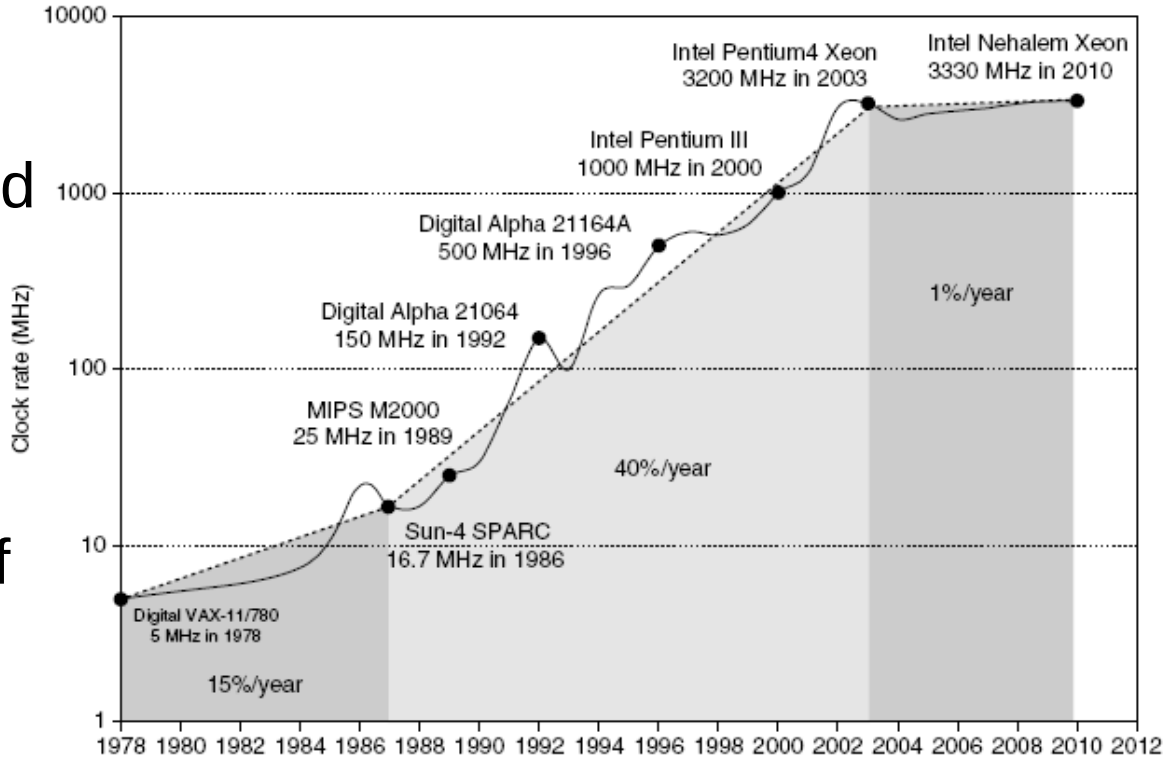

35
30
25
20
15
10
5
0
Normalized Clock Frequency
V scales 0.7
V scales 0.8
V scales 0.9
1
2
3
4
5
6
7
8
9
10
Generation

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
  - $Current_{static}$ 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)

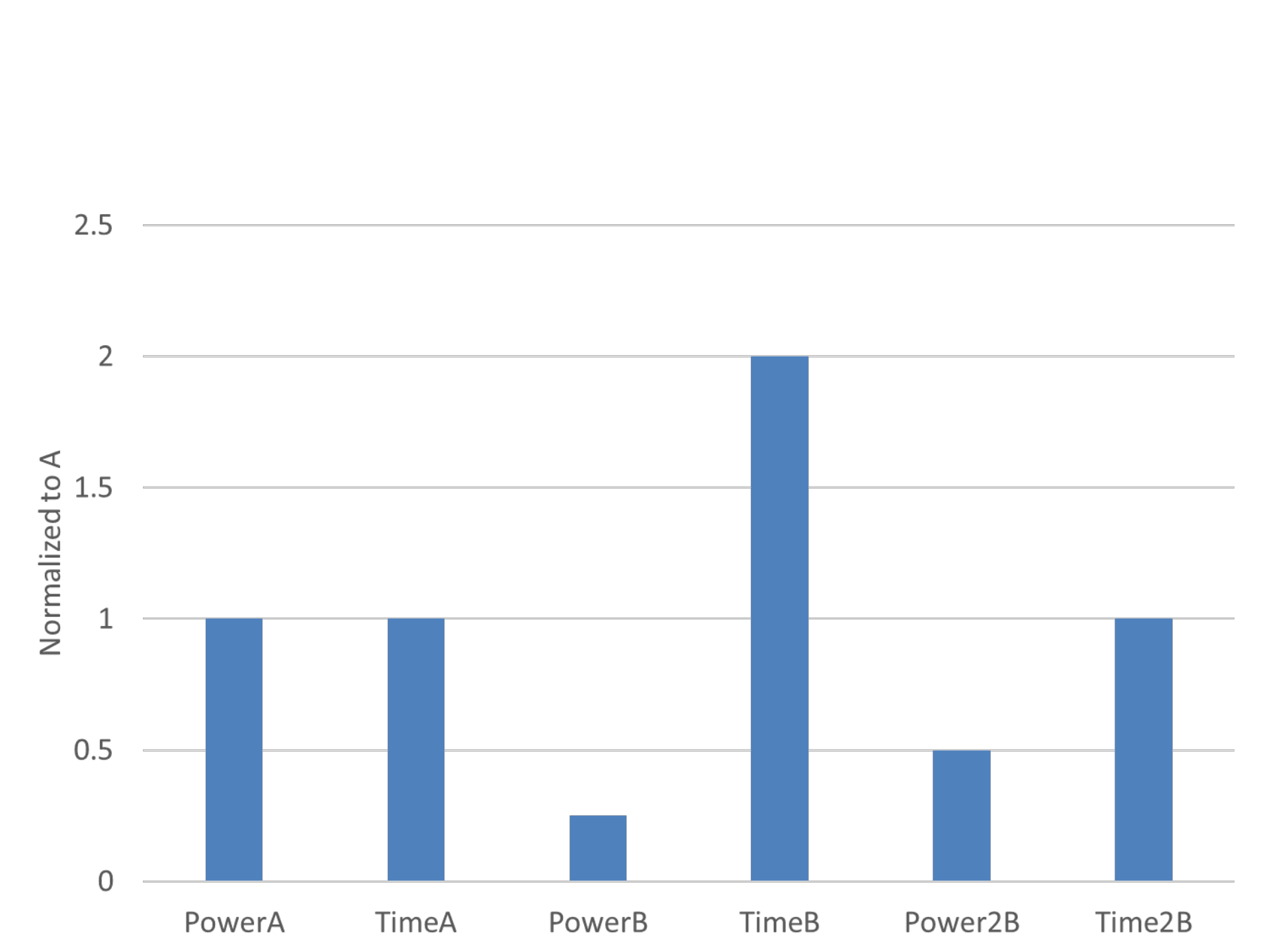
- What happens to the execution time of a program when we lower/increase clock frequency of a processor (DVFS)
  - Execution Time = I . CPI . Cycle Time
- Memory operates at a different clock rate
  - Has its own clock (memory controller)

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

Turbo Mode

- TDP for a CPU
- Nominal frequency below maximum possible frequency
- If there is headroom increase frequency
- else decrease frequency