

# Trends and evaluation

## Arquitectura de Computadores

J. Daniel García Sánchez (coordinador)  
David Expósito Singh  
Javier García Blas  
J. Manuel Pérez Lobato

Grupo ARCOS  
Departamento de Informática  
Universidad Carlos III de Madrid

- 1 Technology trends
- 2 Power and energy trends
- 3 Trends in cost
- 4 Performance evaluation
- 5 Conclusion

# Technology impact

- Technology changes have impact on **ISA** implementation mechanisms.
  
- **Technologies:**
  - Integrated circuit logic.
  - DRAM.
  - Flash.
  - Magnetic disks.
  - Networks.

# Trends

## ■ Integrated circuits technologies.

- Transistors density:  $\uparrow$  35% per year.
- Die size:  $\uparrow$  10%-20% per year.
- Combined effect:  $\uparrow$  40%-55% per year (Moore's Law).
  - **No longer true!**

## ■ DRAM Capacity.

- $\uparrow$  25%-40% per year (going down).

## ■ Flash Capacity.

- $\uparrow$  50%-60% per year.
- 8-10 times cheaper per bit than DRAM.

## ■ Magnetic disks capacity.

- $\uparrow$  5% per year.
- 8-10 times cheaper per bit than Flash.
- 200-300 times cheaper than DRAM.

# Bandwidth and latency

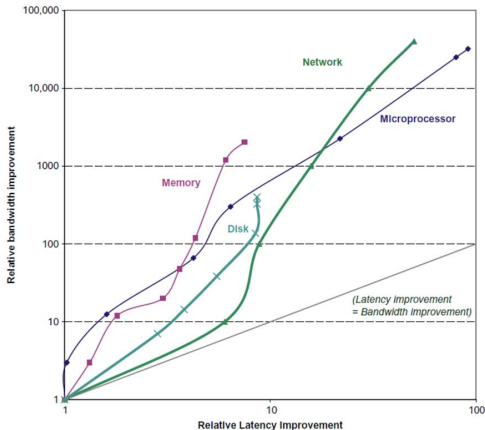
## ■ **Bandwidth** or **throughput**.

- Amount of work performed per unit of time.
- **Processors**: Increase between 32,000 and 40,000.
- **Memory and disks**: Increase between 300 and 1,200.

## ■ **Latency** and **response time**.

- Time between event start and end.
- **Processors**: Increase between 50 and 90.
- **Memories and disks**: Increase between 6 and 8.

# Bandwidth versus latency



Fuente: Computer Architecture: A Quantitative Approach. 6 Ed  
Hennessy and Patterson. Morgan Kaufmann. 2017.

- 1 Technology trends
- 2 Power and energy trends
- 3 Trends in cost
- 4 Performance evaluation
- 5 Conclusion

# Power and Energy: Example

- Two different systems (**A** y **B**).
  - **A** consumes 20% more power than **B**.
  - **A** runs a task in 70% of **B** time.
  - Which has a lower cost?
- The adequate metric for comparison is **Energy**.
  - $E(B) = P(B) \cdot t(B)$
  - $E(A) = 1.2 \cdot P(B) \cdot 0.7 \cdot t(B) = 0.84 \cdot E(B)$
  - System **A** uses 84% of **B** energy.



# Energy and power in microprocessors

- In CMOS technology, **energy consumption** is derived from **transistors switching**.
- **Dynamic energy:**
  - Amount of energy needed to switch.
    - $0 \rightarrow 1$  or  $1 \rightarrow 0$ .
    - $E_d \approx \frac{1}{2} \cdot X_c \cdot V^2$
- **Dynamic power:**
  - Depends on switching frequency.
    - $P_d \approx \frac{1}{2} \cdot X_c \cdot V^2 \cdot f$

## Note

$X_c$ : Capacitive load

$V$ : Voltage

$f$ : Frequency

# Example

- If a 15% voltage reduction implies a 15% frequency reduction:
  - Which is the effect on dynamic power?

## Solution

$$\frac{P_{new}}{P_{old}} = \frac{(V \cdot 0.85)^2 \cdot (f \cdot 0.85)}{V^2 \cdot f} = 0.85^3 = 0.61$$

# Consequences

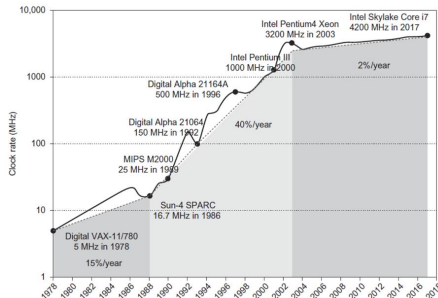
## ■ Reduction:

- Power and dynamic energy get reduced when voltage is reduced.
  - In 20 years voltage got reduced from 5V to 1V.
- Capacitive load depends on transistors fan-out.
  - Mechanism to control power and energy.

# Evolution

- Evolution dominated by number of transistors increase and frequency increase.
  - Power and energy increase.

- Intel 80386 → 2 W
- Intel Core i9-10900K 5.3GHz → 95 W.
  - Chip:  $1.5 \times 1.5$  cm.
  - Limit of cooling by ventilation.



Fuente: Computer Architecture: A Quantitative Approach. 6 Ed  
Hennessy and Patterson. Morgan Kaufmann. 2017.

# Energy efficiency

## ■ Techniques:

- Turn off clock for inactive modules.
- Dynamic Voltage-Frequency Scaling (DVFS).
- Low power modes for memory and disks.
  - Requires reactivation mechanism.
- Automatic overclocking.
  - Enabled when it is safe.
  - Example: Core i7 3.3 GHz may run bursts at 3.6 GHz.

- 1 Technology trends
- 2 Power and energy trends
- 3 Trends in cost
- 4 Performance evaluation
- 5 Conclusion

# Cost

- Manufacturing cost for a computer decreases over time.
  - Learning curve principle.
    - Measured by yield of manufacturing process (percentage of devices surviving manufacturing)
    - When yield is doubled, cost is reduced to half.
    - **DRAM**: Average yearly decrease around 40% in cost and price (except when there is shortage or oversupply).
  - Volume:
    - 10% decrease in cost when volume is doubled.
    - Reduction of cost amortized per unit.
    - Increase of manufacturing process efficiency.
  - Multiple vendors selling the same product (*commodities*):
    - Highly competitive market.

# Cost of integrated circuits

- Manufacturing process.
  - Wafer → Dies.

## Cost

$$Cost_{IC} = \frac{Cost_{die} + Cost_{testing} + Cost_{packing}}{yield}$$

$$Cost_{die} = \frac{Cost_{wafer}}{Dies_{wafer} \times yield}$$

$$Dies_{wafer} = \frac{\pi \times \left(\frac{diameter}{2}\right)^2}{area} - \frac{\pi \times diameter}{\sqrt{2} \times area}$$



# Example

- Wafer with 30 cm. diameter.
  - Dies of 1.5 cm.
    - **Dies per wafer:** 270.
  - Dies of 1 cm.
    - **Dies per wafer:** 640.

- 1 Technology trends
- 2 Power and energy trends
- 3 Trends in cost
- 4 Performance evaluation
- 5 Conclusion

## 4 Performance evaluation

- Performance metrics
- Benchmarks
- Amdahl's Law
- Processor performance

# Execution speed

- What does it mean that computer **A** is faster than computer **B**?
  - **Desktop.**
    - My program runs in less time.
    - I want to decrease execution time.
  - **Website admin.**
    - I can process more transactions per hour.
    - I want to increase throughput.

# Performance and execution time

- Performance  $P(x)$  is a metric, inverse to execution time  $T(x)$ .

## Performance

$$P(x) = \frac{1}{T(x)}$$

- High Performance  $\rightarrow$  Low execution time.

- $x$  runs  $n$  times faster than  $Y$ .

## Speedup

$$n = \frac{T(x)}{T(y)} = \frac{\frac{1}{P(x)}}{\frac{1}{P(y)}} = \frac{P(y)}{P(x)}$$

# Metrics

- The **only** reliable metric for comparing computer performance is the execution of **real programs**.
  - Any other metric is error-prone.
  - Any alternative other than real programs is error-prone.
  
- **Execution time.**
  - **Response time:** Total elapsed time.
  - **Perceived by the user:** CPU time: Time the CPU has been busy.



## 4 Performance evaluation

- Performance metrics
- **Benchmarks**
- Amdahl's Law
- Processor performance

# Workload

- Computer **performance** depends on the evaluated **workload**.
  
- Computers adapted to specific workloads:
  - Web servers.
  - Database servers.
  - File servers.
  - Personal computers.
  - Multiprocessors.
  - Multicomputers.
  - ...



# Benchmarks

- Application or set of applications used to evaluate performance.
- **Approaches:**
  - **Kernels:** Small parts of real applications.
    - *Example:* FFT.
  - **Toy programs:** Short programs.
    - *Example:* Quicksort.
  - **Synthetic benchmarks:** Invented to represent real applications.
    - *Example:* Dhrystone.
- All are bad approaches:
  - **Architect and compiler might cheat!**

# Benchmarks

## ■ Embedded:

- Dhrystone (arguable relevance).
- EEMBC (kernels).

## ■ Desktop:

- SPEC2017 (mix of integer and floating point programs).

## ■ Servers:

- SPECWeb, SPECSFS, SPECjbb, SPECvirt\_Sc2010.
- TPC

# Example: SPEC2006

- **SPECrate/SPECspeed 2017 Integer:** Integer arithmetic.
  - 20 programs: C (10), C++ (8), Fortran (2).
  - Several application domains:
    - Languages and compilers, route planning, event simulation, video compression, artificial intelligence, ...
  
- **SPECrate 2017, SPECspeed 2017 Floating point:** Floating point
  - 23 programs: Fortran (6), C (6), Fortran/C (5) C++ (2), C++/C (2), C++/C/Fortran (2).
  - Several application domains:
    - Physics, Chemistry, Biology, Algebra, image *rendering*, ...

## 4 Performance evaluation

- Performance metrics
- Benchmarks
- **Amdahl's Law**
- Processor performance

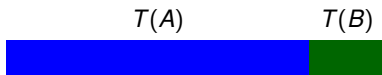
# Amdahl's Law

- Performance increase obtained using a faster execution mode is limited by the fraction of time that the mode can be used.
- **Speedup:**
  - Ratio between improved performance ( $P(I)$ ) and original performance ( $P(O)$ ).

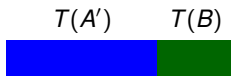
$$S = \frac{P(I)}{P(O)}$$

$$S = \frac{T(O)}{T(I)}$$

# Execution time



$$F = \frac{T(A)}{T(A) + T(B)}$$



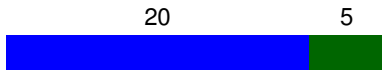
$$S(i) = \frac{T(A)}{T(A')}$$

$$T' = T(A') + T(B) = \frac{T(A)}{S(i)} + (1 - F) \times T$$

$$T' = \frac{F \times T}{S(i)} + (1 - F) \times T$$

$$T' = T \times \left[ (1 - F) + \frac{F}{S(i)} \right]$$

# Example



$$F = \frac{20}{20 + 5} = 0.8$$

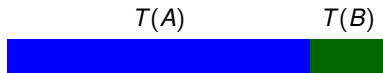


$$S(i) = \frac{20}{10} = 2$$

$$T' = T \times \left[ (1 - F) + \frac{F}{S(i)} \right] = 25 \times \left[ (1 - 0.8) + \frac{0.8}{2} \right] = 15$$

■ We already knew this!

# Amdahl's Law



$$T' = T \times \left[ (1 - F) + \frac{F}{S(i)} \right]$$

$$S = \frac{T}{T'} = \frac{T}{T \times \left[ (1 - F) + \frac{F}{S(i)} \right]} = \frac{1}{(1 - F) + \frac{F}{S(i)}}$$

- **Speedup** depends **exclusively** on the **improvement fraction** and the **speedup of the improvement**.



# Case 1

- A Web server distributes its time between:
  - **Computing**: 40
  - **I/O**: 60
- When replaced by another machine that can perform computing 10 times faster, which is the global *speedup*?

## Solution

$$S = \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} = 1.5625$$

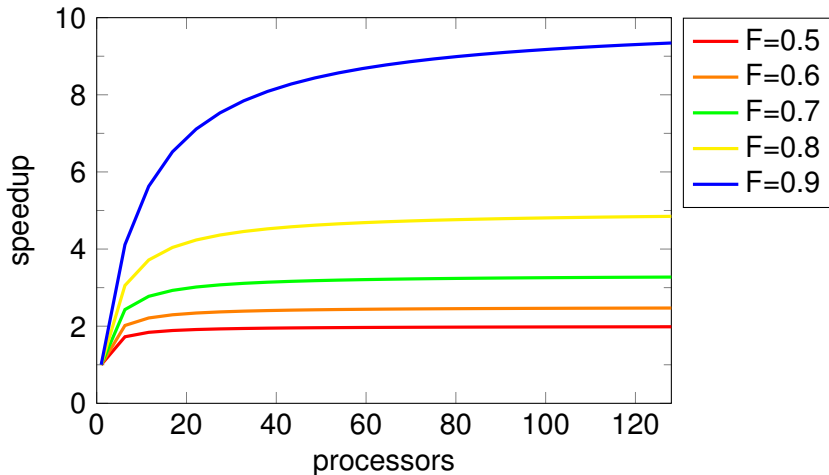
## Case 2

- An application has a parallelizable part that takes 50% of the execution time.
  - Assuming that this part can be fully parallelized with 32 processors, which is the maximum speedup?

### Solution

$$S = \frac{1}{0.5 + \frac{0.5}{32}} = \frac{1}{0.515625} = 1.9393$$

# Speedup evolution



# Amdahl's Law consequences

- The greater the fraction of improvement ( $F$ ), the more effective the improvement is.
- To improve a complex system you must optimized the elements that are used most of the time (most common case).
- Optimization application:
  - **Within the processor**: in the data path.
  - **In the instruction set**: the execution of most frequent instructions.
  - **In the design of memory hierarchy, programming and compilation**: exploiting reference locality.
    - 10% of code is executed for 90% of time.



## 4 Performance evaluation

- Performance metrics
- Benchmarks
- Amdahl's Law
- Processor performance



# Execution time

- A processor executes each instruction during several clock cycles.

## Time consumed by CPU

$$time_{CPU} = \frac{cycles_{CPU}}{\text{clock frequency}}$$

# CPI: Cycles per instruction

- Average speed may be expressed as cycles per instruction (CPI) using:
  - Total number of consumed cycles, and
  - number of executed instructions or instruction count (IC).

## CPI

$$CPI = \frac{cycles_{CPU}}{IC}$$

# Factors in execution time

## CPI and CPU time

$$CPI = \frac{cycles_{CPU}}{IC}$$

$$time_{CPU} = \frac{cycles_{CPU}}{f} = \frac{CPI \times IC}{f} = CPI \times IC \times T$$

- If any of the 3 factors is reduced by 10% the total execution time is reduced by 10%.
  - But the 3 factors are related.



# Instructions classes

- Different instruction classes have different IC and CPI.

## Global CPI

$$cycles_{CPU} = \sum_{i=1}^n IC_i \times CPI$$

$$time_{CPU} = \left( \sum_{i=1}^n IC_i \times CPI_i \right) \times T$$

$$CPI_{global} = \frac{\sum_{i=1}^n IC_i \times CPI_i}{IC} = \sum_{i=1}^n \frac{IC_i}{IC} \times CPI_i$$

Impact of  
instructions  
relative frequency  
in program  
execution.

# Example

- In a program's execution we have observed that:
  - **Floating point operation**: 25% (4.0 CPI on average).
  - **Operation FPSQR** (*square root*): 2% (20.0 CPI).
    - Included in floating point.
  - **Rest of instructions**: 1.33 CPI.
  
- Choose among design alternatives:
  - a Decrease FPSQR CPI to 2.
  - b Decrease all floating point instructions CPI to 2.5.

# Solution

$$CPI = 0.25 \times 4 + 1.33 \times 0.75 = \mathbf{1.9975}$$

$$0.25 \times CPI_{FP} = 0.23 \times CPI_{otherFP} + 0.02 \times CPI_{FPSQR}$$

$$0.25 \times 4 = 0.23 \times CPI_{otherFP} + 0.02 \times 20$$

$$CPI_{otherFP} = \frac{0.24 \times 4 - 0.02 \times 20}{0.23} = 2.6087$$

$$CPI_{nuevoFPSQR} = 0.23 \times 2.6087 + 0.02 \times \mathbf{2} + 0.75 \times 1.33 = \mathbf{1.6375}$$

$$CPI_{newFP} = 0.25 \times \mathbf{2.5} + 0.75 \times 1.33 = \mathbf{1.6225}$$

- 1 Technology trends
- 2 Power and energy trends
- 3 Trends in cost
- 4 Performance evaluation
- 5 Conclusion

# Summary

- Bandwidth has improved much more than latency during the last 20 years.
- The increasing used power limits the clock frequency.
- Decrease in manufacturing cost over time.
- The only reliable metric to compare computer performance is the execution of real programs.
- Amdahl's Law sets a limit on performance improvement with multiple applications.
- Relative instruction frequency has a high impact on program execution speed.

# References

- **Computer Architecture. A Quantitative Approach**  
5th Ed.  
Hennessy and Patterson.  
Sections 1.4 to 1.9.

# Trends and evaluation

## Arquitectura de Computadores

J. Daniel García Sánchez (coordinador)  
David Expósito Singh  
Javier García Blas  
J. Manuel Pérez Lobato

Grupo ARCOS  
Departamento de Informática  
Universidad Carlos III de Madrid