



IC-UNICAMP

MO401

IC/Unicamp

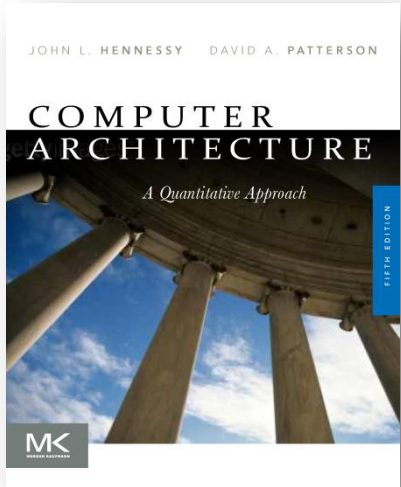
2014s1

Prof Mario Côrtes

Capítulo 1: Fundamentos de Análise e Design Quantitativos

Tópicos

- Classes of Computers
- Computer Architecture (def)
- Trends: technology, power, cost
- Dependability
- Measurements / performance
- Quantitative Principles



Chapter 1

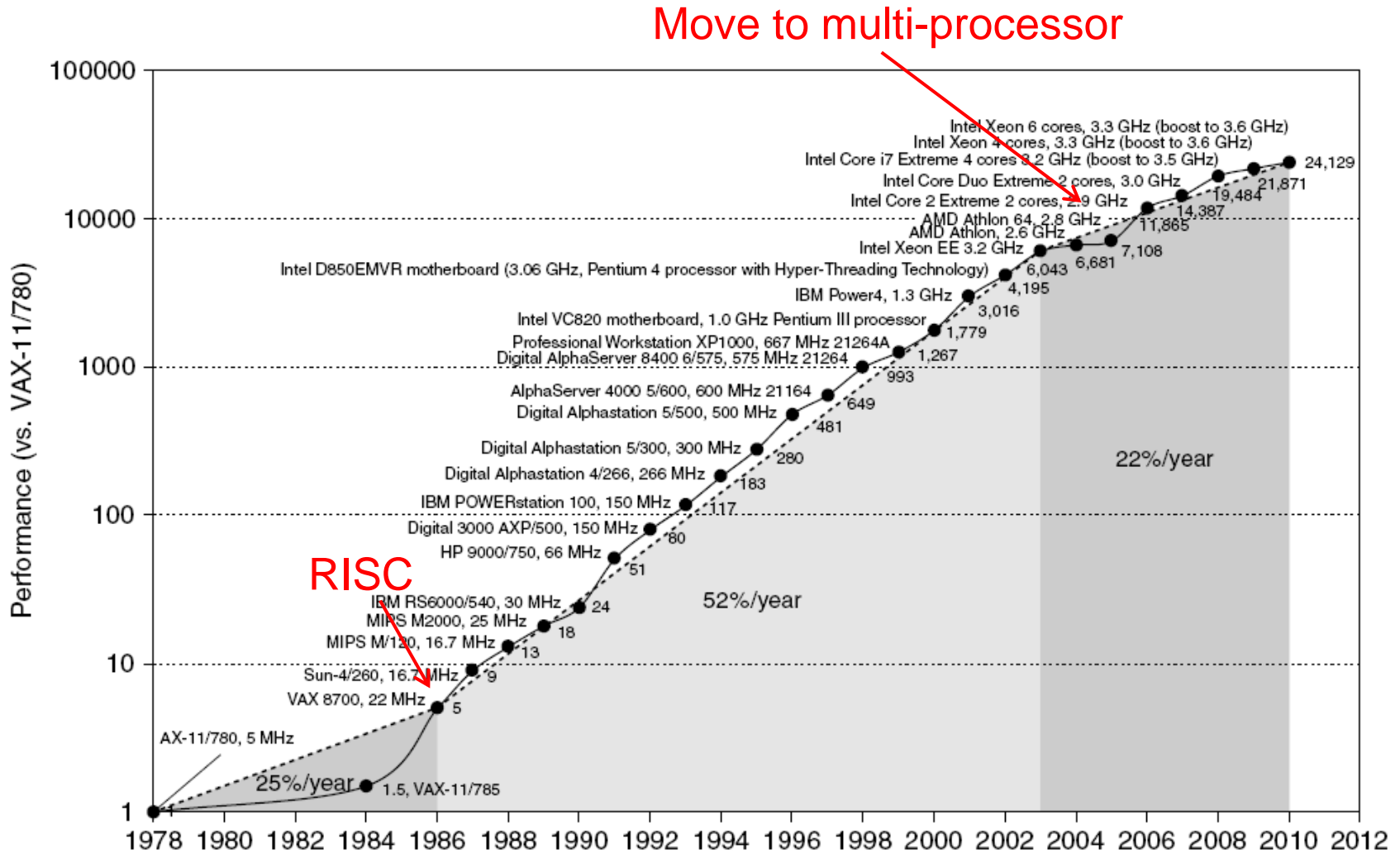
Fundamentals of Quantitative Design and Analysis

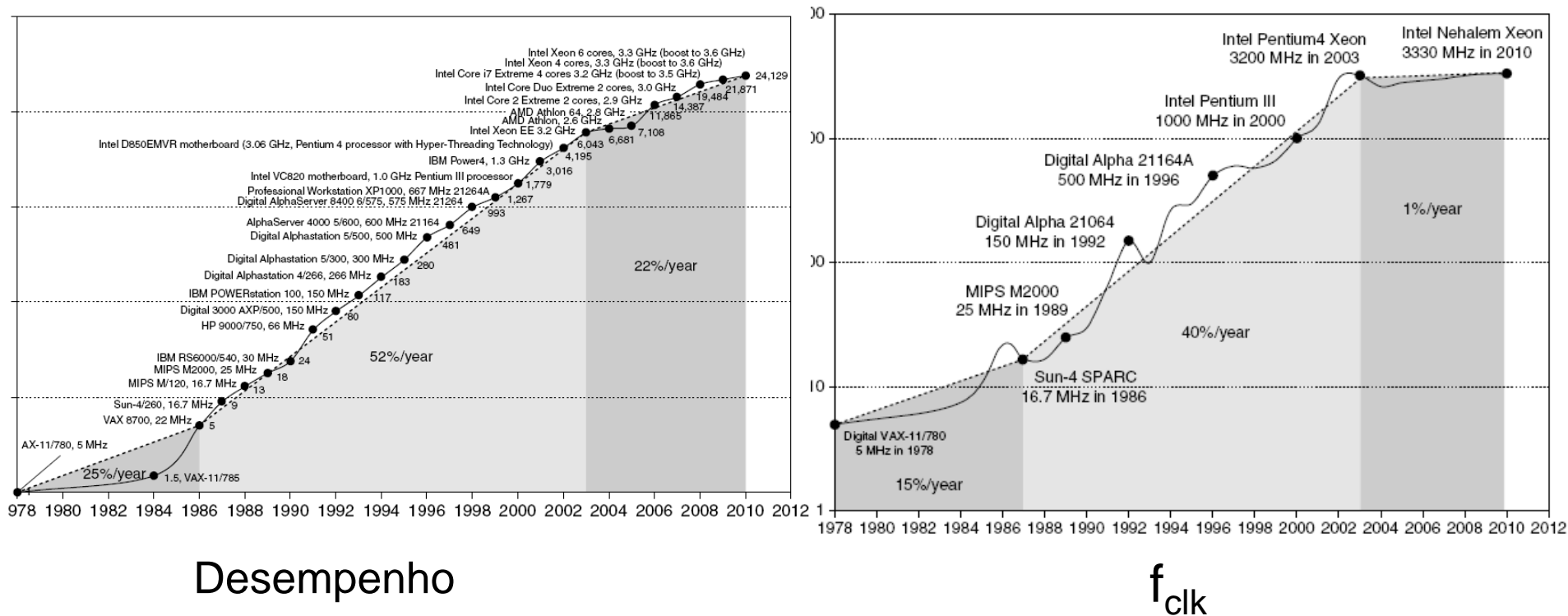
Slides do curso baseados em:

- slides da editora MK
- slides de MO401 preparados pelo Prof. Centoducatte

- Performance improvements:
 - Smartphone 2012 (USD\$500) → mesma capacidade de um computador de \$1M em 1985
 - Improvements in semiconductor technology
 - Feature size, clock speed
 - Improvements in computer architectures
 - Enabled by HLL compilers, UNIX
 - Lead to RISC architectures
 - Together have enabled:
 - Lightweight computers
 - Productivity-based managed/interpreted programming languages

Single Processor Performance





Desempenho

f_{clk}

Figure 1.1 Growth in processor performance since the late 1970s. This chart plots performance relative to the VAX 11/780 as measured by the SPEC benchmarks (see Section 1.8). Prior to the mid-1980s, processor performance growth was largely technology driven and averaged about 25% per year. The increase in growth to about 52% since then is attributable to more advanced architectural and organizational ideas. By 2003, this growth led to a difference in performance of about a factor of 25 versus if we had continued at the 25% rate. Performance for floating-point-oriented calculations has increased even faster. Since 2003, the limits of power and available instruction-level parallelism have slowed uniprocessor performance, to no more than 22% per year, or about 5 times slower than had we continued at 52% per year. (The fastest SPEC performance since 2007 has had automatic parallelization turned on with increasing number of cores per chip each year, so uniprocessor speed is harder to gauge. These results are limited to single-socket systems to reduce the impact of automatic parallelization.) Figure 1.11 on page 24 shows the improvement in clock rates for these same three eras. Since SPEC has changed over the years, performance of newer machines is estimated by a scaling factor that relates the performance for two different versions of SPEC (e.g., SPEC89, SPEC92, SPEC95, SPEC2000, and SPEC2006).



Current Trends in Architecture

- Cannot continue to leverage Instruction-Level parallelism (ILP)
 - Single processor performance improvement ended in 2003
- New models for performance:
 - Data-level parallelism (DLP)
 - Thread-level parallelism (TLP)
 - Request-level parallelism (RLP)
- These require explicit restructuring of the application

1.2 Classes of Computers

- **Personal Mobile Device (PMD)**
 - e.g. smart phones, tablet computers
 - Emphasis on energy efficiency and real-time
- **Desktop Computing**
 - Emphasis on price-performance
- **Servers**
 - Emphasis on availability, scalability, throughput
- **Clusters / Warehouse Scale Computers**
 - Used for “Software as a Service (SaaS)”
 - Emphasis on availability and price-performance
 - Sub-class: Supercomputers, emphasis: floating-point performance and fast internal networks
- **Embedded Computers**
 - Emphasis: price



Fig 1.2: classes de computadores

| Feature | Personal mobile device (PMD) | Desktop | Server | Clusters/warehouse-scale computer | Embedded |
|-------------------------------|---|---|---|---|---|
| Price of system | \$100–\$1000 | \$300–\$2500 | \$5000–\$10,000,000 | \$100,000–\$200,000,000 | \$10–\$100,000 |
| Price of micro-processor | \$10–\$100 | \$50–\$500 | \$200–\$2000 | \$50–\$250 | \$0.01–\$100 |
| Critical system design issues | Cost, energy, media performance, responsiveness | Price-performance, energy, graphics performance | Throughput, availability, scalability, energy | Price-performance, throughput, energy proportionality | Price, energy, application-specific performance |

Figure 1.2 A summary of the five mainstream computing classes and their system characteristics. Sales in 2010 included about 1.8 billion PMDs (90% cell phones), 350 million desktop PCs, and 20 million servers. The total number of embedded processors sold was nearly 19 billion. In total, 6.1 billion ARM-technology based chips were shipped in 2010. Note the wide range in system price for servers and embedded systems, which go from USB keys to network routers. For servers, this range arises from the need for very large-scale multiprocessor systems for high-end transaction processing.

Fig 1.3: custo downtime servidores

| Application | Cost of downtime per hour | Annual losses with downtime of | | |
|-----------------------------|---------------------------|--------------------------------|--------------------|-------------------|
| | | 1% (87.6 hrs/yr) | 0.5% (43.8 hrs/yr) | 0.1% (8.8 hrs/yr) |
| Brokerage operations | \$6,450,000 | \$565,000,000 | \$283,000,000 | \$56,500,000 |
| Credit card authorization | \$2,600,000 | \$228,000,000 | \$114,000,000 | \$22,800,000 |
| Package shipping services | \$150,000 | \$13,000,000 | \$6,600,000 | \$1,300,000 |
| Home shopping channel | \$113,000 | \$9,900,000 | \$4,900,000 | \$1,000,000 |
| Catalog sales center | \$90,000 | \$7,900,000 | \$3,900,000 | \$800,000 |
| Airline reservation center | \$89,000 | \$7,900,000 | \$3,900,000 | \$800,000 |
| Cellular service activation | \$41,000 | \$3,600,000 | \$1,800,000 | \$400,000 |
| Online network fees | \$25,000 | \$2,200,000 | \$1,100,000 | \$200,000 |
| ATM service fees | \$14,000 | \$1,200,000 | \$600,000 | \$100,000 |

Figure 1.3 Costs rounded to nearest \$100,000 of an unavailable system are shown by analyzing the cost of downtime (in terms of immediately lost revenue), assuming three different levels of availability and that downtime is distributed uniformly. These data are from Kembel [2000] and were collected and analyzed by Contingency Planning Research.

Parallelism

- Classes of parallelism in applications:
 - Data-Level Parallelism (DLP): vários dados manipulados simultaneamente
 - Task-Level Parallelism (TLP): tarefas independentes em execução simultânea
- Classes of architectural parallelism:
 - Instruction-Level Parallelism (ILP): μ -arquitetura
 - Vector architectures/Graphic Processor Units (GPUs): dados paralelos
 - Thread-Level Parallelism
 - Request-Level Parallelism: multi-programação

Flynn's Taxonomy

- Single instruction stream, single data stream (SISD)
- Single instruction stream, multiple data streams (SIMD)
 - Vector architectures
 - Multimedia extensions
 - Graphics processor units
- Multiple instruction streams, single data stream (MISD)
 - No commercial implementation
- Multiple instruction streams, multiple data streams (MIMD)
 - Tightly-coupled MIMD
 - Loosely-coupled MIMD

1.3 Defining Computer Architecture

- “Old” view of computer architecture:
 - Instruction Set Architecture (ISA) design
 - i.e. decisions regarding:
 - registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, instruction encoding
- “Real” computer architecture:
 - Specific requirements of the target machine
 - (detalhes à frente)

Organização vs Hardware

- Organização e hardware:
 - Organização (ou micro arquitetura): alto-nível → sistema de memória, interconexão com memória, projeto do processador
 - exemplo: o i7 e AMD Opteron usam mesmo ISA mas tem micro arquiteturas diferentes (pipeline, cache etc)
 - Hardware: detalhes do projeto lógico e empacotamento
 - exemplo: Intel Core i7 e Intel Xeon 7560 são quase idênticos mas têm diferenças no clock e no sistema de memória
 - Neste livro: arquitetura = ISA + μ -arquitetura + hardware

Requisitos de arquitetura

- “Real” computer architecture:
 - Specific requirements of the target machine
 - Design to maximize performance within constraints: cost, power, and availability
 - Includes ISA, microarchitecture, hardware



Fig 1.7: Requisitos funcionais

| Functional requirements | Typical features required or supported |
|--|---|
| <i>Application area</i> | <i>Target of computer</i> |
| Personal mobile device | Real-time performance for a range of tasks, including interactive performance for graphics, video, and audio; energy efficiency (Ch. 2, 3, 4, 5; App. A) |
| General-purpose desktop | Balanced performance for a range of tasks, including interactive performance for graphics, video, and audio (Ch. 2, 3, 4, 5; App. A) |
| Servers | Support for databases and transaction processing; enhancements for reliability and availability; support for scalability (Ch. 2, 5; App. A, D, F) |
| Clusters/warehouse-scale computers | Throughput performance for many independent tasks; error correction for memory; energy proportionality (Ch 2, 6; App. F) |
| Embedded computing | Often requires special support for graphics or video (or other application-specific extension); power limitations and power control may be required; real-time constraints (Ch. 2, 3, 5; App. A, E) |
| <i>Level of software compatibility</i> | <i>Determines amount of existing software for computer</i> |
| At programming language | Most flexible for designer; need new compiler (Ch. 3, 5; App. A) |
| Object code or binary compatible | Instruction set architecture is completely defined—little flexibility—but no investment needed in software or porting programs (App. A) |

Fig 1.7: Requisitos funcionais

| Functional requirements | Typical features required or supported |
|--------------------------------------|---|
| <i>Operating system requirements</i> | <i>Necessary features to support chosen OS (Ch. 2; App. B)</i> |
| Size of address space | Very important feature (Ch. 2); may limit applications |
| Memory management | Required for modern OS; may be paged or segmented (Ch. 2) |
| Protection | Different OS and application needs: page vs. segment; virtual machines (Ch. 2) |
| <i>Standards</i> | <i>Certain standards may be required by marketplace</i> |
| Floating point | Format and arithmetic: IEEE 754 standard (App. J), special arithmetic for graphics or signal processing |
| I/O interfaces | For I/O devices: Serial ATA, Serial Attached SCSI, PCI Express (App. D, F) |
| Operating systems | UNIX, Windows, Linux, CISCO IOS |
| Networks | Support required for different networks: Ethernet, Infiniband (App. F) |
| Programming languages | Languages (ANSI C, C++, Java, Fortran) affect instruction set (App. A) |

Figure 1.7 Summary of some of the most important functional requirements an architect faces. The left-hand column describes the class of requirement, while the right-hand column gives specific examples. The right-hand column also contains references to chapters and appendices that deal with the specific issues.

1.4 Trends in Technology

- Integrated circuit technology
 - Transistor density: 35%/year
 - Die size: 10-20%/year
 - Transistors per die: 40-55%/year
- DRAM capacity: 25-40%/year (slowing)
- Flash capacity: 50-60%/year
 - 15-20X cheaper/bit than DRAM
- Magnetic disk technology: 40%/year
 - 15-25X cheaper/bit than Flash
 - 300-500X cheaper/bit than DRAM



Fig 1.8: evolução de DRAM

| CA:AQA Edition | Year | DRAM growth rate | Characterization of impact on DRAM capacity |
|----------------|------|------------------|---|
| 1 | 1990 | 60%/year | Quadrupling every 3 years |
| 2 | 1996 | 60%/year | Quadrupling every 3 years |
| 3 | 2003 | 40%–60%/year | Quadrupling every 3 to 4 years |
| 4 | 2007 | 40%/year | Doubling every 2 years |
| 5 | 2011 | 25%–40%/year | Doubling every 2 to 3 years |

Figure 1.8 Change in rate of improvement in DRAM capacity over time. The first two editions even called this rate the DRAM Growth Rule of Thumb, since it had been so dependable since 1977 with the 16-kilobit DRAM through 1996 with the 64-megabit DRAM. Today, some question whether DRAM capacity can improve at all in 5 to 7 years, due to difficulties in manufacturing an increasingly three-dimensional DRAM cell [Kim 2005].



Bandwidth and Latency

- Bandwidth or throughput
 - Total work done in a given time
 - 10,000-25,000X improvement for processors
 - 300-1200X improvement for memory and disks
- Latency or response time
 - Time between start and completion of an event
 - 30-80X improvement for processors
 - 6-8X improvement for memory and disks
- rule of thumb:
(improvements BW) = (improvements latency)²

Bandwidth and Latency

Log-log plot of bandwidth and latency milestones

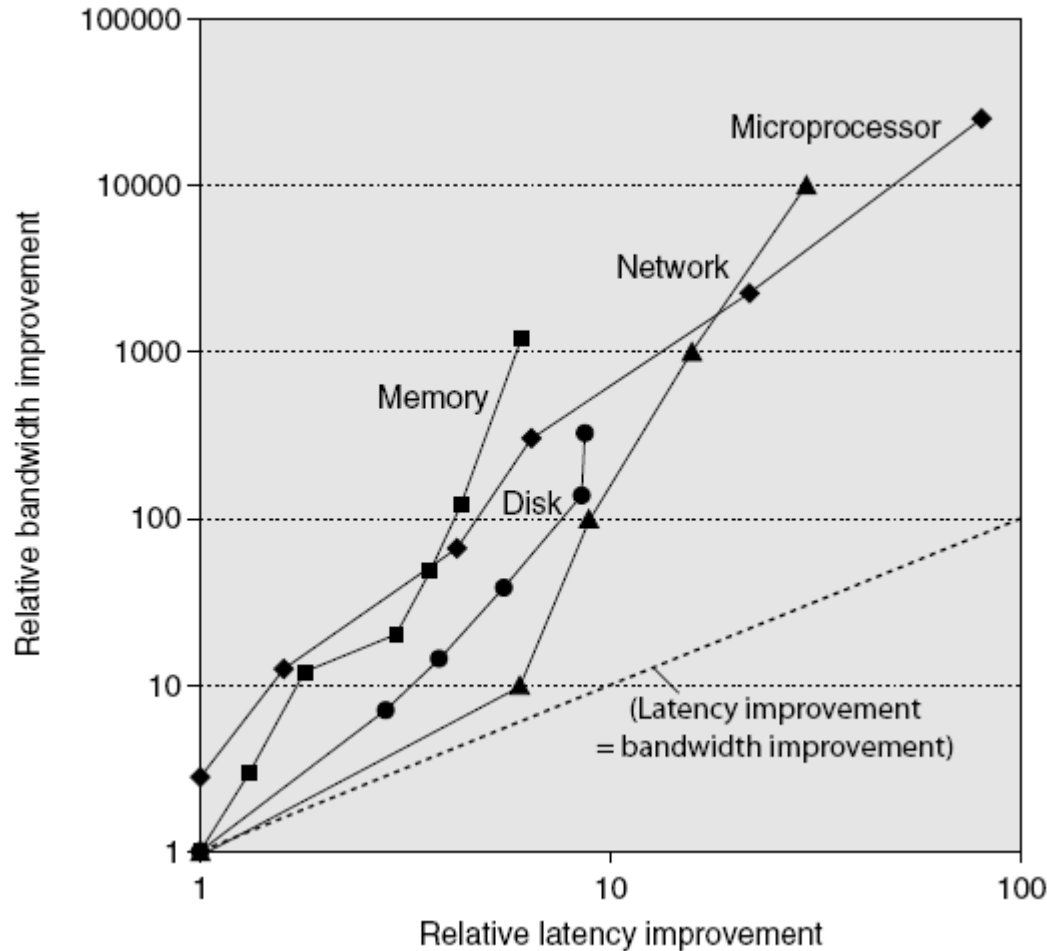


Figure 1.9 Log-log plot of bandwidth and latency milestones from Figure 1.10 relative to the first milestone. Note that latency improved 6X to 80X while bandwidth improved about 300X to 25,000X. Updated from Patterson [2004].



Fig 1.10: performance milestones

| | | | | | | | |
|-----------------------------|---------------------------------------|---------------------------------------|---|----------------------------------|--------------------------------------|--|--|
| Microprocessor | 16-bit address/ bus, microcoded | 32-bit address/ bus, microcoded | 5-stage pipeline, on-chip I & D caches, FPU | 2-way superscalar, 64-bit bus | Out-of-order 3-way superscalar | Out-of-order superpipelined, on-chip L2 cache | Multicore OOO 4-way on chip L3 cache, Turbo |
| Product | Intel 80286 | Intel 80386 | Intel 80486 | Intel Pentium | Intel Pentium Pro | Intel Pentium 4 | Intel Core i7 |
| Year | 1982 | 1985 | 1989 | 1993 | 1997 | 2001 | 2010 |
| Die size (mm ²) | 47 | 43 | 81 | 90 | 308 | 217 | 240 |
| Transistors | 134,000 | 275,000 | 1,200,000 | 3,100,000 | 5,500,000 | 42,000,000 | 1,170,000,000 |
| Processors/chip | 1 | 1 | 1 | 1 | 1 | 1 | 4 |
| Pins | 68 | 132 | 168 | 273 | 387 | 423 | 1366 |
| Latency (clocks) | 6 | 5 | 5 | 5 | 10 | 22 | 14 |
| Bus width (bits) | 16 | 32 | 32 | 64 | 64 | 64 | 196 |
| Clock rate (MHz) | 12.5 | 16 | 25 | 66 | 200 | 1500 | 3333 |
| Bandwidth (MIPS) | 2 | 6 | 25 | 132 | 600 | 4500 | 50,000 |
| Latency (ns) | 320 | 313 | 200 | 76 | 50 | 15 | 4 |
| Memory module | DRAM | Page mode DRAM | Fast page mode DRAM | Fast page mode DRAM | Synchronous DRAM | Double data rate SDRAM | DDR3 SDRAM |
| Module width (bits) | 16 | 16 | 32 | 64 | 64 | 64 | 64 |
| Year | 1980 | 1983 | 1986 | 1993 | 1997 | 2000 | 2010 |
| Mbits/DRAM chip | 0.06 | 0.25 | 1 | 16 | 64 | 256 | 2048 |
| Die size (mm ²) | 35 | 45 | 70 | 130 | 170 | 204 | 50 |
| Pins/DRAM chip | 16 | 16 | 18 | 20 | 54 | 66 | 134 |
| Bandwidth (MBytes/s) | 13 | 40 | 160 | 267 | 640 | 1600 | 16,000 |
| Latency (ns) | 225 | 170 | 125 | 75 | 62 | 52 | 37 |



Fig 1.10: performance milestones

| Local area network | Ethernet | Fast Ethernet | Gigabit Ethernet | 10 Gigabit Ethernet | 100 Gigabit Ethernet | |
|-----------------------|-----------------------|--------------------|--------------------|---------------------|----------------------|----------------------|
| IEEE standard | 802.3 | 803.3u | 802.3ab | 802.3ac | 802.3ba | |
| Year | 1978 | 1995 | 1999 | 2003 | 2010 | |
| Bandwidth (Mbits/sec) | 10 | 100 | 1000 | 10,000 | 100,000 | |
| Latency (μ sec) | 3000 | 500 | 340 | 190 | 100 | |
| Hard disk | 3600 RPM | 5400 RPM | 7200 RPM | 10,000 RPM | 15,000 RPM | 15,000 RPM |
| Product | CDC WrenI 94145-36 | Seagate ST41600 | Seagate ST15150 | Seagate ST39102 | Seagate ST373453 | Seagate ST3600057 |
| Year | 1983 | 1990 | 1994 | 1998 | 2003 | 2010 |
| Capacity (GB) | 0.03 | 1.4 | 4.3 | 9.1 | 73.4 | 600 |
| Disk form factor | 5.25 inch | 5.25 inch | 3.5 inch | 3.5 inch | 3.5 inch | 3.5 inch |
| Media diameter | 5.25 inch | 5.25 inch | 3.5 inch | 3.0 inch | 2.5 inch | 2.5 inch |
| Interface | ST-412 | SCSI | SCSI | SCSI | SCSI | SAS |
| Bandwidth (MBytes/s) | 0.6 | 4 | 9 | 24 | 86 | 204 |
| Latency (ms) | 48.3 | 17.1 | 12.7 | 8.8 | 5.7 | 3.6 |

Figure 1.10 Performance milestones over 25 to 40 years for microprocessors, memory, networks, and disks. The microprocessor milestones are several generations of IA-32 processors, going from a 16-bit bus, microcoded 80286 to a 64-bit bus, multicore, out-of-order execution, superpipelined Core i7. Memory module milestones go from 16-bit-wide, plain DRAM to 64-bit-wide double data rate version 3 synchronous DRAM. Ethernet advanced from 10 Mbits/sec to 100 Gbits/sec. Disk milestones are based on rotation speed, improving from 3600 RPM to 15,000 RPM. Each case is best-case bandwidth, and latency is the time for a simple operation assuming no contention. Updated from Patterson [2004].

Transistors and Wires

- Feature size
 - Minimum size of transistor or wire in x or y dimension
 - From μm to nm
 - 10 μm in 1971 to 32 nm in 2011
 - Transistor performance scales linearly
 - Wire delay does not improve with feature size!
 - Integration density scales quadratically

1.5 Power and Energy

- Problem: Get power in, get power out
- Thermal Design Power (TDP)
 - Characterizes sustained power consumption
 - Used as target for power supply and cooling system
 - Lower than peak power, higher than average power consumption
- Clock rate can be reduced dynamically to limit power consumption
- Energy per task is often a better measurement

Dynamic Energy and Power

- Dynamic energy
 - Transistor switch from 0 -> 1 or 1 -> 0
 - $\frac{1}{2} \times \text{Capacitive load} \times \text{Voltage}^2$
- Dynamic power
 - $\frac{1}{2} \times \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$ switched
- Reducing clock rate reduces power, not energy



Exmpl P23: dynamic energy

Example Some microprocessors today are designed to have adjustable voltage, so a 15% reduction in voltage may result in a 15% reduction in frequency. What would be the impact on dynamic energy and on dynamic power?

Answer Since the capacitance is unchanged, the answer for energy is the ratio of the voltages since the capacitance is unchanged:

$$\frac{\text{Energy}_{\text{new}}}{\text{Energy}_{\text{old}}} = \frac{(\text{Voltage} \times 0.85)^2}{\text{Voltage}^2} = 0.85^2 = 0.72$$

thereby reducing energy to about 72% of the original. For power, we add the ratio of the frequencies

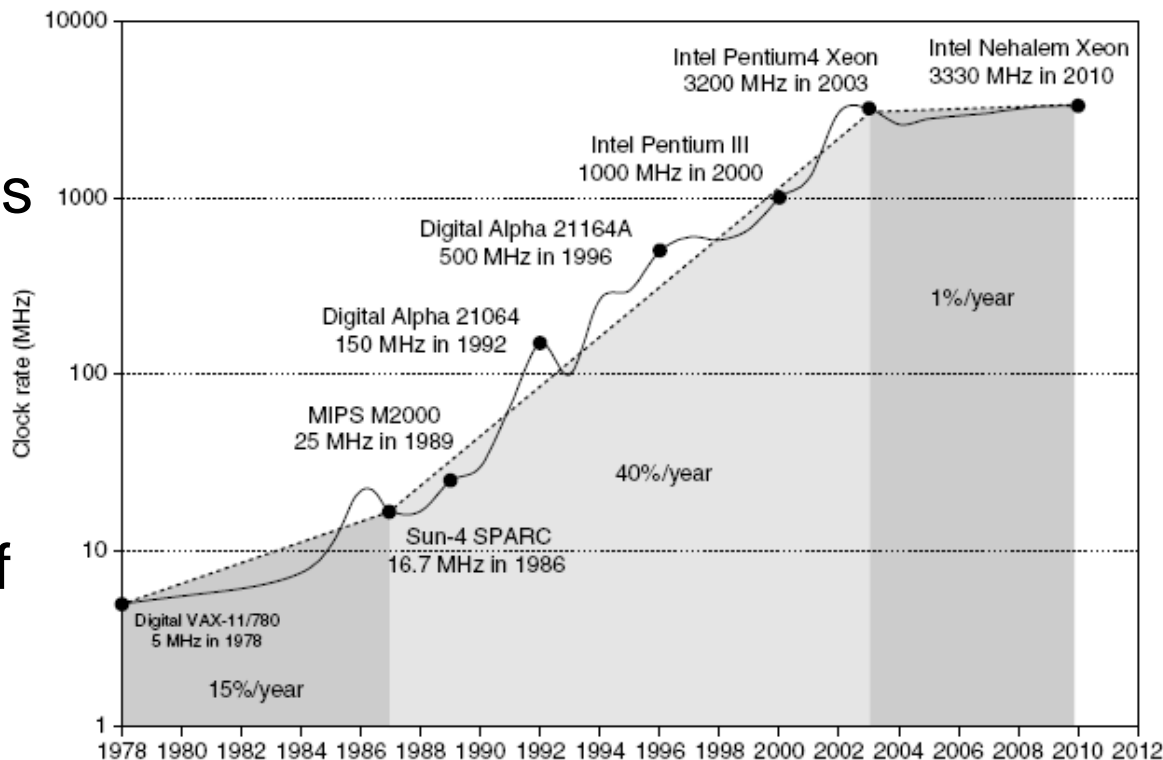
$$\frac{\text{Power}_{\text{new}}}{\text{Power}_{\text{old}}} = 0.72 \times \frac{(\text{Frequency switched} \times 0.85)}{\text{Frequency switched}} = 0.61$$

shrinking power to about 61% of the original.



Power e clock rate

- Intel 80386 consumed ~ 2 W
- 3.3 GHz Intel Core i7 consumes 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



Reducing Power

- Techniques for reducing power:
 - Do nothing well: unidades inativas → power down
 - Dynamic Voltage-Frequency Scaling (DVFS):
alguns valores de V_{dd} e freq disponíveis (fig 1.12)
 - Low power state for DRAM, disks
 - Overclocking, turning off cores: desligar alguns cores e rodar demais em overclocking

Fig 1.12: power savings DVFS

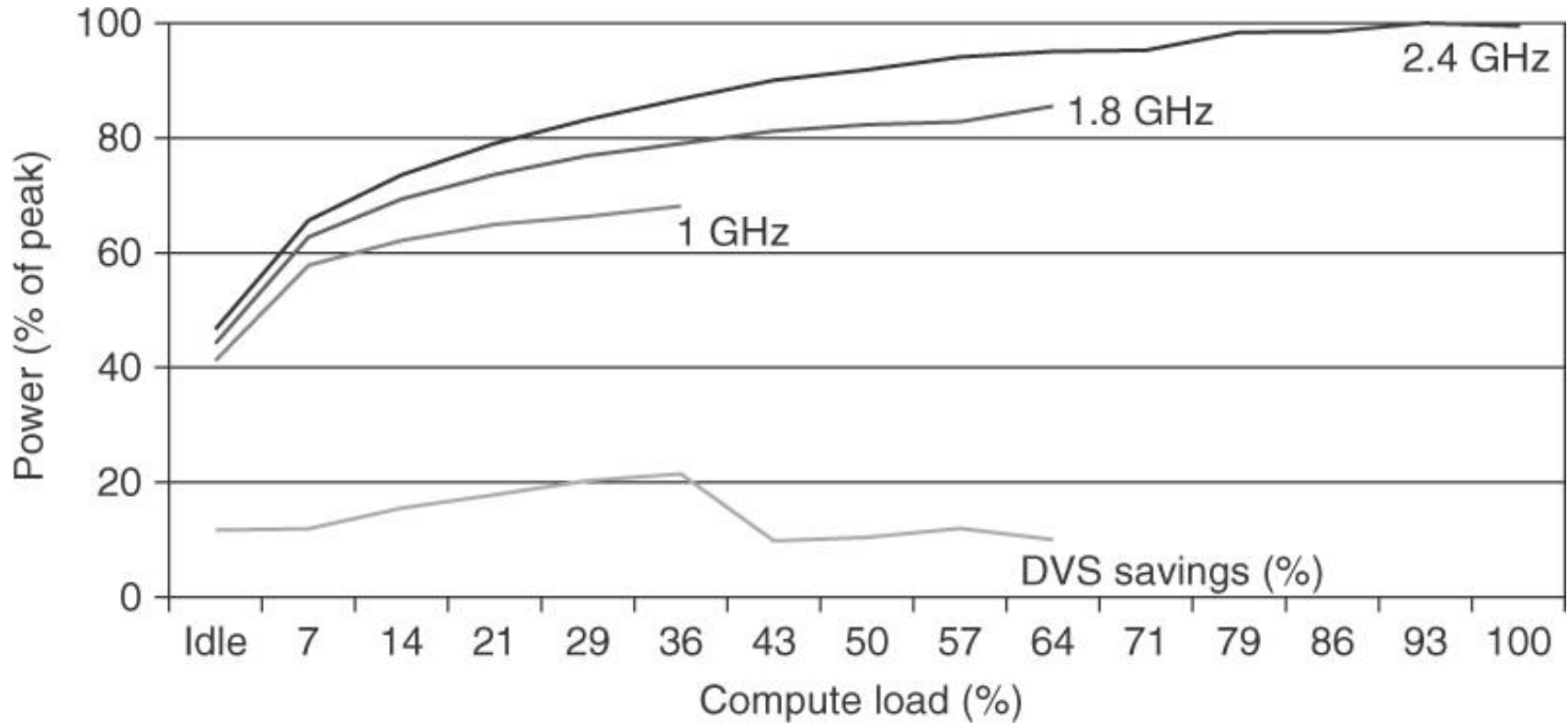


Figure 1.12 Energy savings for a server using an AMD Opteron microprocessor, 8 GB of DRAM, and one ATA disk. At 1.8 GHz, the server can only handle up to two-thirds of the workload without causing service level violations, and, at 1.0 GHz, it can only safely handle one-third of the workload. (Figure 5.11 in Barroso and Hölzle [2009].)

Static Power



IC-UNICAMP

- Static power consumption
 - Leakage current: corrente de fuga (estática)
 - $\text{Current}_{\text{static}} \times \text{Voltage}$
 - Scales with number of transistors
 - To reduce: power gating
 - áreas inativas → desligar alimentação (gating)
 - evita corrente de fuga



1.6 Trends in Cost

- Cost driven down by learning curve
 - Yield
- DRAM: price closely tracks cost
- Microprocessors: price depends on volume
 - 10% less for each doubling of volume

Fig 1.13: die do Intel Core i7

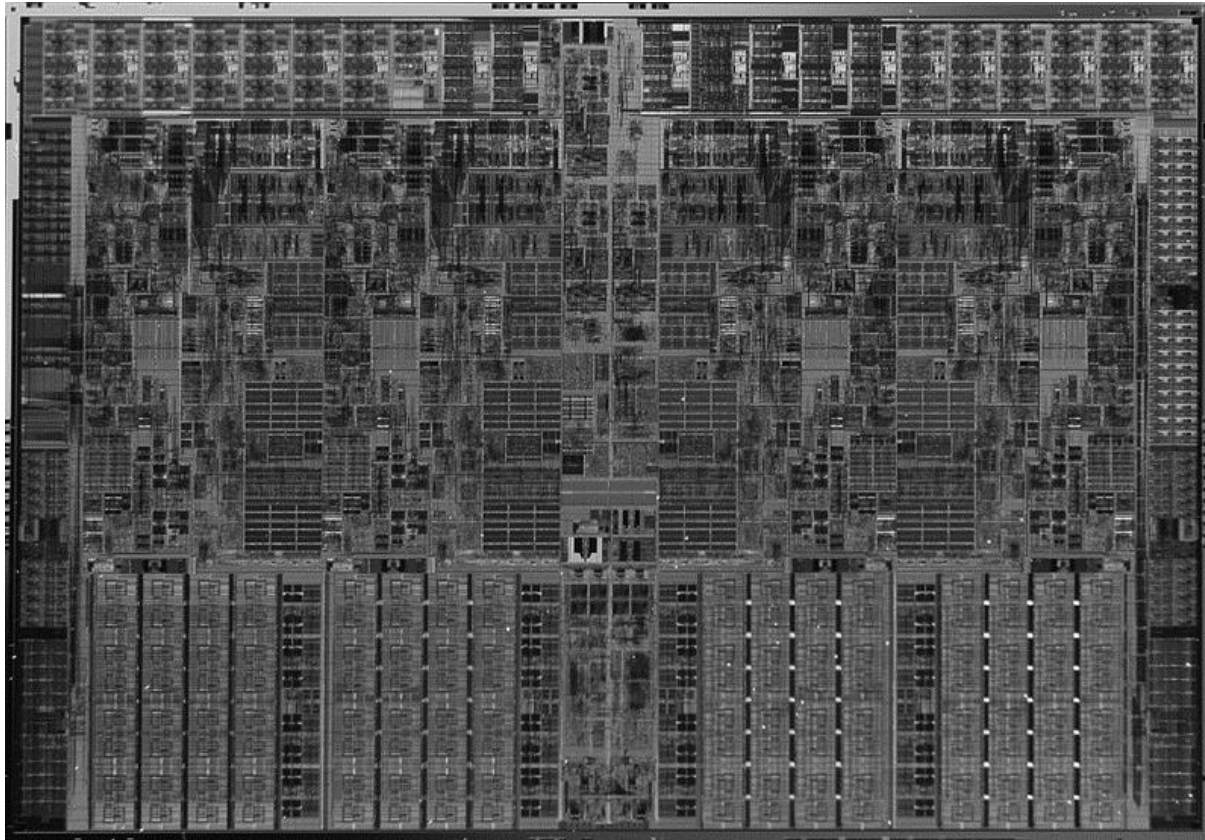


Figure 1.13 Photograph of an Intel Core i7 microprocessor die, which is evaluated in Chapters 2 through 5. The dimensions are 18.9 mm by 13.6 mm (257 mm²) in a 45 nm process. (Courtesy Intel.)

Fig 1.14: Floorplan do Intel Core i7

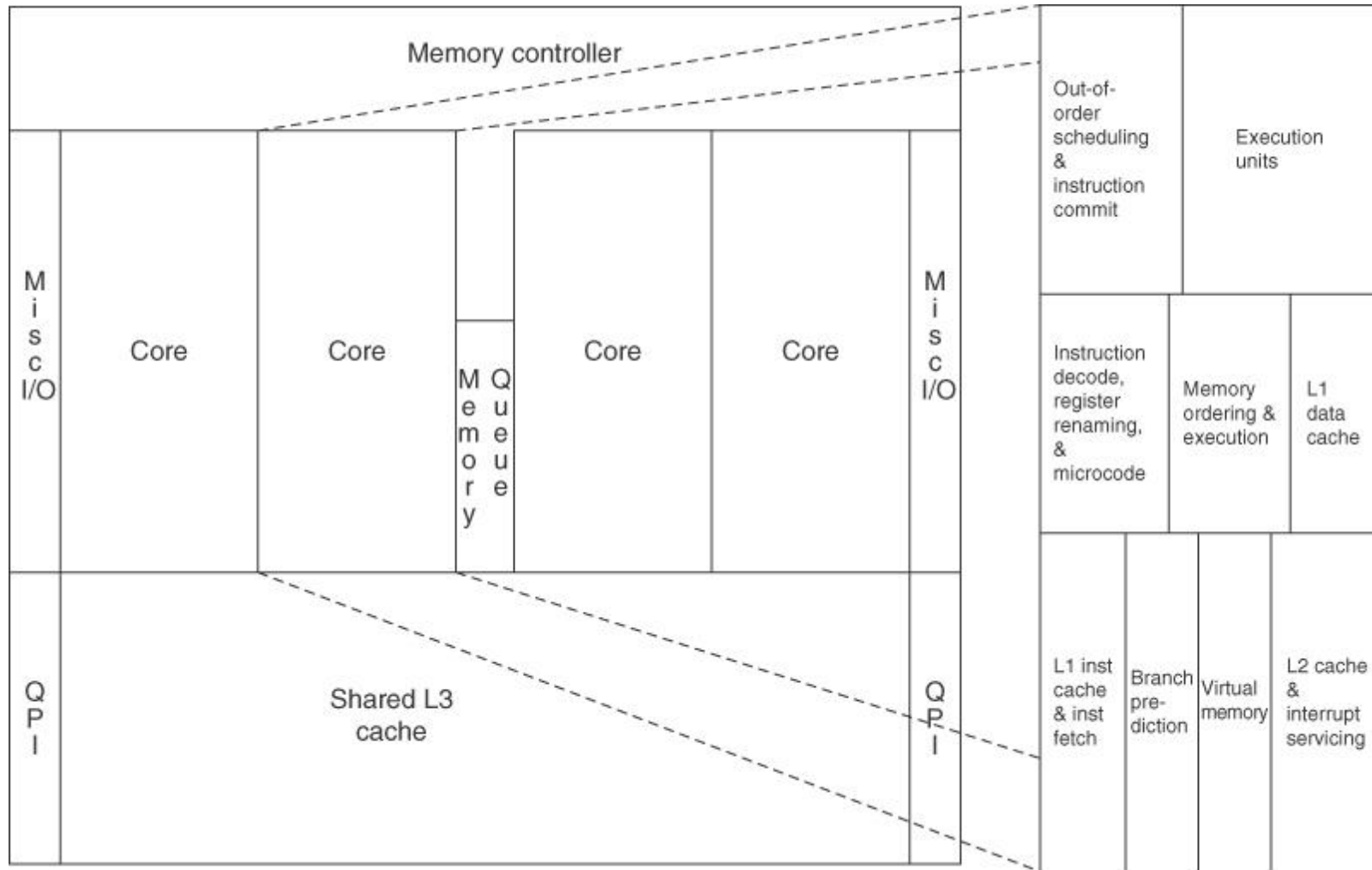


Figure 1.14 Floorplan of Core i7 die in Figure 1.13 on left with close-up of floorplan of second core on right.

Fig 1.15: wafer com 280 dies

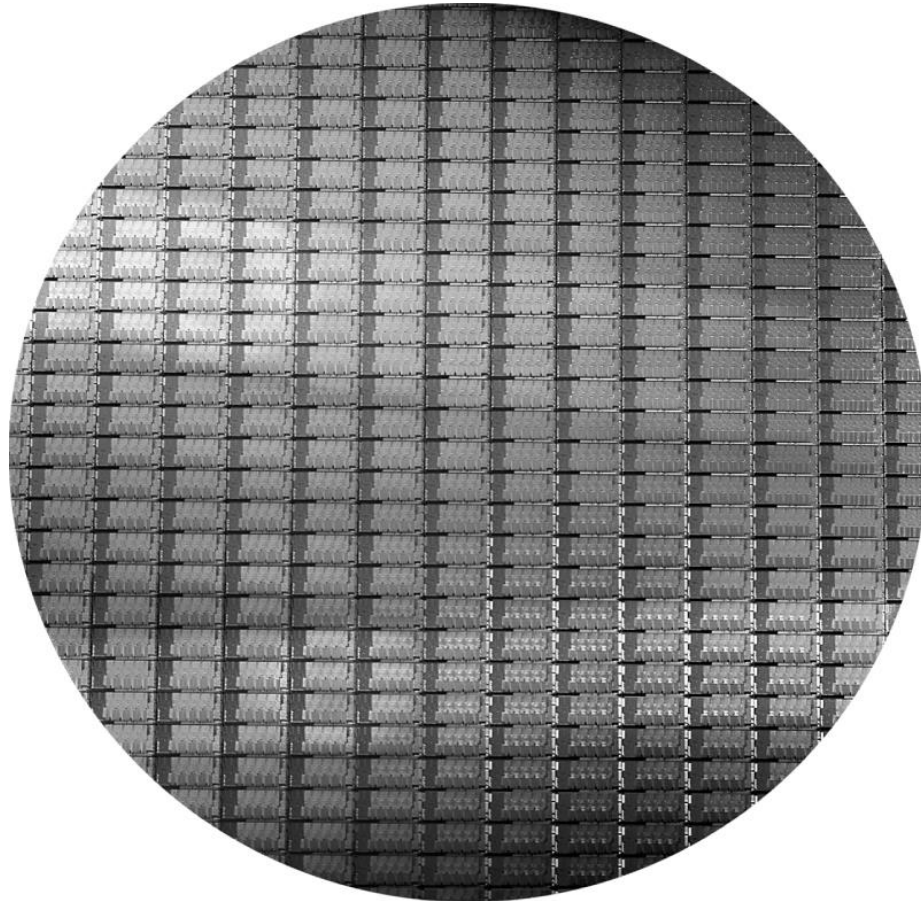


Figure 1.15 This 300 mm wafer contains 280 full Sandy Bridge dies, each 20.7 by 10.5 mm in a 32 nm process. (Sandy Bridge is Intel's successor to Nehalem used in the Core i7.) At 216 mm², the formula for dies per wafer estimates 282. (Courtesy Intel.)

- Integrated circuit

$$\text{Cost of integrated circuit} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging and final test}}{\text{Final test yield}}$$

$$\text{Cost of die} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}}$$

$$\text{Dies per wafer} = \frac{\pi \times (\text{Wafer diameter}/2)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2} \times \text{Die area}}$$

- Bose-Einstein formula:

$$\text{Die yield} = \text{Wafer yield} \times 1 / (1 + \text{Defects per unit area} \times \text{Die area})^N$$

- Defects per unit area = 0.016-0.057 defects per square cm (2010)
- N = process-complexity factor = 11.5-15.5 (40 nm, 2010)
- Wafer yield: wafers completamente ruins, não precisam ser testados



Example Find the number of dies per 300 mm (30 cm) wafer for a die that is 1.5 cm on a side and for a die that is 1.0 cm on a side.

Answer When die area is 2.25 cm²:

$$\text{Dies per wafer} = \frac{\pi \times (30/2)^2}{2.25} - \frac{\pi \times 30}{\sqrt{2} \times 2.25} = \frac{706.9}{2.25} - \frac{94.2}{2.12} = 270$$

Since the area of the larger die is 2.25 times bigger, there are roughly 2.25 as many smaller dies per wafer:

$$\text{Dies per wafer} = \frac{\pi \times (30/2)^2}{1.00} - \frac{\pi \times 30}{\sqrt{2} \times 1.00} = \frac{706.9}{1.00} - \frac{94.2}{1.41} = 640$$

Exmpl

P31:

dies/wafer

However, this formula only gives the maximum number of dies per wafer. The critical question is: What is the fraction of good dies on a wafer, or the *die yield*? A simple model of integrated circuit yield, which assumes that defects are randomly distributed over the wafer and that yield is inversely proportional to the complexity of the fabrication process, leads to the following:

$$\text{Die yield} = \text{Wafer yield} \times 1 / (1 + \text{Defects per unit area} \times \text{Die area})^N$$

This Bose–Einstein formula is an empirical model developed by looking at the yield of many manufacturing lines [Sydow 2006]. *Wafer yield* accounts for wafers that are completely bad and so need not be tested. For simplicity, we'll just assume the wafer yield is 100%. Defects per unit area is a measure of the random manufacturing defects that occur. In 2010, the value was typically 0.1 to 0.3 defects per square inch, or 0.016 to 0.057 defects per square centimeter, for a 40 nm process, as it depends on the maturity of the process (recall the learning curve, mentioned earlier). Finally, N is a parameter called the process-complexity factor, a measure of manufacturing difficulty. For 40 nm processes in 2010, N ranged from 11.5 to 15.5.



Exmpl P31: yield

Example Find the die yield for dies that are 1.5 cm on a side and 1.0 cm on a side, assuming a defect density of 0.031 per cm^2 and N is 13.5.

Answer The total die areas are 2.25 cm^2 and 1.00 cm^2 . For the larger die, the yield is

$$\text{Die yield} = 1 / (1 + 0.031 \times 2.25)^{13.5} = 0.40$$

For the smaller die, the yield is

$$\text{Die yield} = 1 / (1 + 0.031 \times 1.00)^{13.5} = 0.66$$

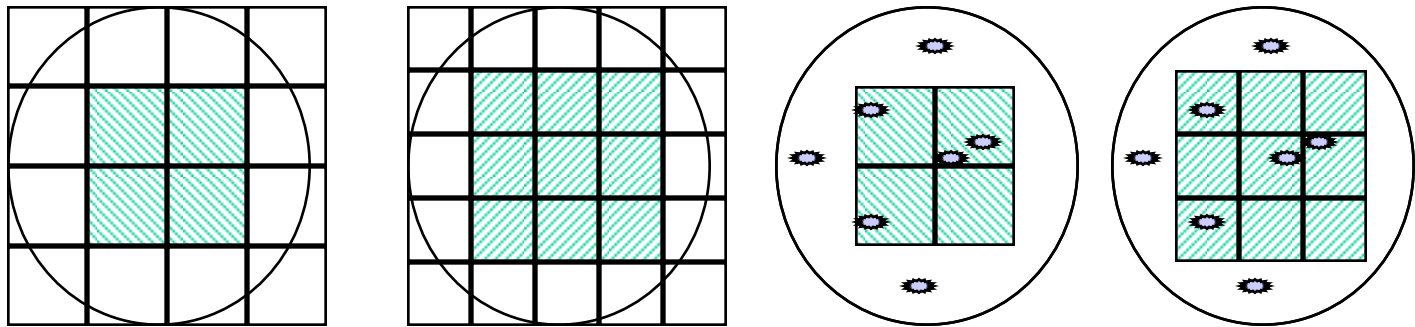
That is, less than half of all the large dies are good but two-thirds of the small dies are good.

Custo de Circuito Integrado (IC)

$$\text{IC cost} = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}$$

$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per Wafer} \times \text{Die yield}}$$

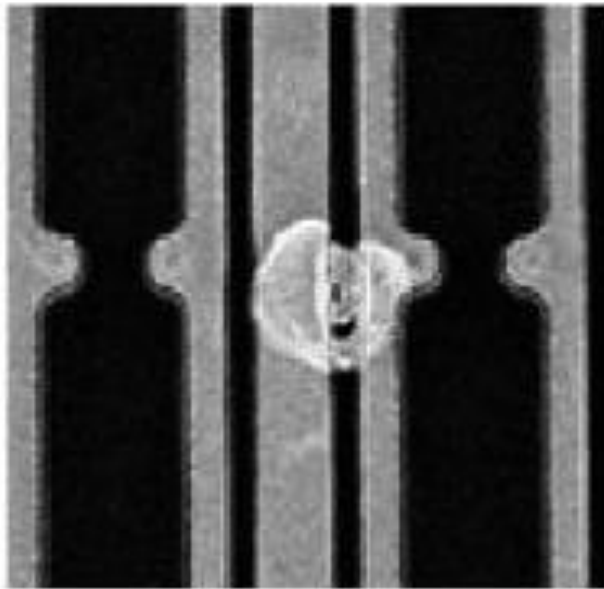
$$\text{Dies per wafer} = \frac{\pi (\text{Wafer_diam}/2)^2}{\text{Die_Area}} - \frac{\pi \times \text{Wafer_diam}}{\sqrt{2} \cdot \text{Die_Area}} - \text{Test_Die}$$



$$\text{Die Yield} = \text{Wafer_yield} \times \left\{ 1 + \left(\frac{\text{Defect_Density} \times \text{Die_area}}{\alpha} \right)^{-\alpha} \right\}$$

Custo do Die é proporcional à (área do die)⁴

Integrated Circuits Yield - Defects





Examplos Reais

| Chip | Metal layers | Line width | Wafer cost | Defect /cm ² | Area mm ² | Dies/wafer | Yield | Die Cost |
|-------------|--------------|------------|------------|-------------------------|----------------------|------------|-------|----------|
| 386DX | 2 | 0.90 | \$900 | 1.0 | 43 | 360 | 71% | \$4 |
| 486DX2 | 3 | 0.80 | \$1200 | 1.0 | 81 | 181 | 54% | \$12 |
| PowerPC 601 | 4 | 0.80 | \$1700 | 1.3 | 121 | 115 | 28% | \$53 |
| HP PA 7100 | 3 | 0.80 | \$1300 | 1.0 | 196 | 66 | 27% | \$73 |
| DEC Alpha | 3 | 0.70 | \$1500 | 1.2 | 234 | 53 | 19% | \$149 |
| SuperSPARC | 3 | 0.70 | \$1700 | 1.6 | 256 | 48 | 13% | \$272 |
| Pentium | 3 | 0.80 | \$1500 | 1.5 | 296 | 40 | 9% | \$417 |

– From "Estimating IC Manufacturing Costs," by Linley Gwennap, *Microprocessor Report*, August 2, 1993, p. 15



1.7 Dependability

- Module reliability
 - Mean time to failure (MTTF)
 - Mean time to repair (MTTR)
 - Mean time between failures (MTBF) = $MTTF + MTTR$
 - Availability = $MTTF / MTBF$
- Taxa de falhas = $1 / MTTF$
 - FIT (failures in time) = taxa de falhas = n^0 de falhas em um bilhão de horas = n^0 falhas / 10^9 h
 - Ex: se $MTTF = 10^6$ h \rightarrow 1000 FIT
- Hipótese comum:
 - distribuição exponencial
 - taxa de falhas constante
 - taxa de falhas de n módulos independentes = Soma (taxa de falhas de cada módulo)

Example Assume a disk subsystem with the following components and MTTF:

- 10 disks, each rated at 1,000,000-hour MTTF
- 1 ATA controller, 500,000-hour MTTF
- 1 power supply, 200,000-hour MTTF
- 1 fan, 200,000-hour MTTF
- 1 ATA cable, 1,000,000-hour MTTF

Using the simplifying assumptions that the lifetimes are exponentially distributed and that failures are independent, compute the MTTF of the system as a whole.

Answer The sum of the failure rates is

$$\begin{aligned}\text{Failure rate}_{\text{system}} &= 10 \times \frac{1}{1,000,000} + \frac{1}{500,000} + \frac{1}{200,000} + \frac{1}{200,000} + \frac{1}{1,000,000} \\ &= \frac{10 + 2 + 5 + 5 + 1}{1,000,000 \text{ hours}} = \frac{23}{1,000,000} = \frac{23,000}{1,000,000,000 \text{ hours}}\end{aligned}$$

or 23,000 FIT. The MTTF for the system is just the inverse of the failure rate:

$$\text{MTTF}_{\text{system}} = \frac{1}{\text{Failure rate}_{\text{system}}} = \frac{1,000,000,000 \text{ hours}}{23,000} = 43,500 \text{ hours}$$

or just under 5 years.

The primary way to cope with failure is redundancy, either in time (repeat the operation to see if it still is erroneous) or in resources (have other components to take over from the one that failed). Once the component is replaced and the system fully repaired, the dependability of the system is assumed to be as good as new. Let's quantify the benefits of redundancy with an example.

Exmpl
P34:
depend



Example Disk subsystems often have redundant power supplies to improve dependability. Using the components and MTTFs from above, calculate the reliability of redundant power supplies. Assume one power supply is sufficient to run the disk subsystem and that we are adding one redundant power supply.

Answer We need a formula to show what to expect when we can tolerate a failure and still provide service. To simplify the calculations, we assume that the lifetimes of the components are exponentially distributed and that there is no dependency between the component failures. MTTF for our redundant power supplies is the mean time until one power supply fails divided by the chance that the other will fail before the first one is replaced. Thus, if the chance of a second failure before repair is small, then the MTTF of the pair is large.

Since we have two power supplies and independent failures, the mean time until one disk fails is $MTTF_{\text{power supply}}/2$. A good approximation of the probability of a second failure is MTTR over the mean time until the other power supply fails. Hence, a reasonable approximation for a redundant pair of power supplies is

$$MTTF_{\text{power supply pair}} = \frac{MTTF_{\text{power supply}}/2}{\frac{MTTR_{\text{power supply}}}{MTTF_{\text{power supply}}}} = \frac{MTTF_{\text{power supply}}^2/2}{MTTR_{\text{power supply}}} = \frac{MTTF_{\text{power supply}}^2}{2 \times MTTR_{\text{power supply}}}$$

Using the MTTF numbers above, if we assume it takes on average 24 hours for a human operator to notice that a power supply has failed and replace it, the reliability of the fault tolerant pair of power supplies is

$$MTTF_{\text{power supply pair}} = \frac{MTTF_{\text{power supply}}^2}{2 \times MTTR_{\text{power supply}}} = \frac{200,000^2}{2 \times 24} \cong 830,000,000$$

making the pair about 4150 times more reliable than a single power supply.

Exmpl P35: Reliability redundant power system

1.8 Measuring Performance

- Typical performance metrics:
 - Response time
 - Throughput
- Speedup of X relative to Y
 - $\text{Execution time}_Y / \text{Execution time}_X$
- Execution time
 - Wall clock time: includes all system overheads
 - CPU time: only computation time
- Benchmarks
 - Kernels (e.g. matrix multiply)
 - Toy programs (e.g. sorting)
 - Synthetic benchmarks (e.g. Dhrystone)
 - Benchmark suites (e.g. SPEC06fp, TPC-C)

Fig 1.16: Evolução SPEC

Benchmark name by SPEC generation

| SPEC2006 benchmark description | SPEC2006 | SPEC2000 | SPEC95 | SPEC92 | SPEC89 |
|--|------------|----------|---------|----------|-----------|
| GNU C compiler | | | | | gcc |
| Interpreted string processing | | | perl | | espresso |
| Combinatorial optimization | | mcf | | | li |
| Block-sorting compression | | bzip2 | | compress | eqntott |
| Go game (AI) | go | vortex | go | sc | |
| Video compression | h264avc | gzip | ijpeg | | |
| Games/path finding | astar | eon | m88ksim | | |
| Search gene sequence | hmmer | twolf | | | |
| Quantum computer simulation | libquantum | vortex | | | |
| Discrete event simulation library | omnetpp | vpr | | | |
| Chess game (AI) | sjeng | crafty | | | |
| XML parsing | xalancbmk | parser | | | |
| CFD/blast waves | bwaves | | | | fpppp |
| Numerical relativity | cactusADM | | | | tomcatv |
| Finite element code | calculix | | | | doduc |
| Differential equation solver framework | dealll | | | | nasa7 |
| Quantum chemistry | gamess | | | | spice |
| EM solver (freq/time domain) | GemsFDTD | | | swim | matrix300 |
| Scalable molecular dynamics (~NAMD) | gromacs | | apsi | hydro2d | |
| Lattice Boltzman method (fluid/air flow) | lbm | | mgrid | su2cor | |
| Large eddie simulation/turbulent CFD | LESlie3d | wupwise | applu | wave5 | |
| Lattice quantum chromodynamics | milc | apply | turb3d | | |
| Molecular dynamics | namd | galgel | | | |
| Image ray tracing | povray | mesa | | | |
| Spare linear algebra | soplex | art | | | |
| Speech recognition | sphinx3 | equake | | | |
| Quantum chemistry/object oriented | tonto | facerec | | | |
| Weather research and forecasting | wrf | ammp | | | |
| Magneto hydrodynamics (astrophysics) | zeusmp | lucas | | | |
| | | fma3d | | | |
| | | sixtrack | | | |

Fig 1.17: SPEC 3 máquinas



| Benchmarks | Ultra 5 time (sec) | Opteron time (sec) | SPECRatio | Itanium 2 time (sec) | SPECRatio | Opteron/Itanium times (sec) | Itanium/Opteron SPECRatios |
|-----------------------|--------------------------|-----------------------|-----------|-------------------------|-----------|--------------------------------|-------------------------------|
| wupwise | 1600 | 51.5 | 31.06 | 56.1 | 28.53 | 0.92 | 0.92 |
| swim | 3100 | 125.0 | 24.73 | 70.7 | 43.85 | 1.77 | 1.77 |
| mgrid | 1800 | 98.0 | 18.37 | 65.8 | 27.36 | 1.49 | 1.49 |
| applu | 2100 | 94.0 | 22.34 | 50.9 | 41.25 | 1.85 | 1.85 |
| mesa | 1400 | 64.6 | 21.69 | 108.0 | 12.99 | 0.60 | 0.60 |
| galgel | 2900 | 86.4 | 33.57 | 40.0 | 72.47 | 2.16 | 2.16 |
| art | 2600 | 92.4 | 28.13 | 21.0 | 123.67 | 4.40 | 4.40 |
| equake | 1300 | 72.6 | 17.92 | 36.3 | 35.78 | 2.00 | 2.00 |
| facerec | 1900 | 73.6 | 25.80 | 86.9 | 21.86 | 0.85 | 0.85 |
| ammp | 2200 | 136.0 | 16.14 | 132.0 | 16.63 | 1.03 | 1.03 |
| lucas | 2000 | 88.8 | 22.52 | 107.0 | 18.76 | 0.83 | 0.83 |
| fma3d | 2100 | 120.0 | 17.48 | 131.0 | 16.09 | 0.92 | 0.92 |
| sixtrack | 1100 | 123.0 | 8.95 | 68.8 | 15.99 | 1.79 | 1.79 |
| apsi | 2600 | 150.0 | 17.36 | 231.0 | 11.27 | 0.65 | 0.65 |
| Geometric mean | | | 20.86 | | 27.12 | 1.30 | 1.30 |

Figure 1.17 SPECfp2000 execution times (in seconds) for the Sun Ultra 5—the reference computer of SPEC2000—and execution times and SPECRatios for the AMD Opteron and Intel Itanium 2. (SPEC2000 multiplies the ratio of execution times by 100 to remove the decimal point from the result, so 20.86 is reported as 2086.) The final two columns show the ratios of execution times and SPECRatios. This figure demonstrates the irrelevance of the reference computer in relative performance. The ratio of the execution times is identical to the ratio of the SPECRatios, and the ratio of the geometric means ($27.12/20.86 = 1.30$) is identical to the geometric mean of the ratios (1.30).



Exmpl P43: geometric mean

Example Show that the ratio of the geometric means is equal to the geometric mean of the performance ratios, and that the reference computer of SPECRatio matters not.

Answer Assume two computers A and B and a set of SPECratios for each.

$$\begin{aligned} \frac{\text{Geometric mean}_A}{\text{Geometric mean}_B} &= \frac{\sqrt[n]{\prod_{i=1}^n \text{SPECRatio } A_i}}{\sqrt[n]{\prod_{i=1}^n \text{SPECRatio } B_i}} = \sqrt[n]{\prod_{i=1}^n \frac{\text{SPECRatio } A_i}{\text{SPECRatio } B_i}} \\ &= \sqrt[n]{\prod_{i=1}^n \frac{\frac{\text{Execution time}_{\text{reference}_i}}{\text{Execution time}_{A_i}}}{\frac{\text{Execution time}_{\text{reference}_i}}{\text{Execution time}_{B_i}}}} = \sqrt[n]{\prod_{i=1}^n \frac{\text{Execution time}_{B_i}}{\text{Execution time}_{A_i}}} = \sqrt[n]{\prod_{i=1}^n \frac{\text{Performance}_{A_i}}{\text{Performance}_{B_i}}} \end{aligned}$$

That is, the ratio of the geometric means of the SPECratios of A and B is the geometric mean of the performance ratios of A to B of all the benchmarks in the suite. Figure 1.17 demonstrates this validity using examples from SPEC.

Medidas

?

- Como descrever em forma numérica o desempenho dos computadores?
- Quais ferramentas (ou qual ferramental) usar para realizar e apresentar as medidas?



Métricas

| Plane | DC to Paris | Speed | Passengers | Throughput (pmp) |
|------------------|-------------|----------|------------|------------------|
| Boeing 747 | 6.5 hours | 610 mph | 470 | 286,700 |
| BAD/Sud Concorde | 3 hours | 1350 mph | 132 | 178,200 |

- **Tempo para executar uma tarefa (ExTime)**
 - **Execution time, response time, latency**
- **Tarefas por dia, hora, semana, segundo, ns, ... (Desempenho)**
 - **Throughput, bandwidth**

Métricas Comparação

"X é n vezes mais rápido que Y" significa:

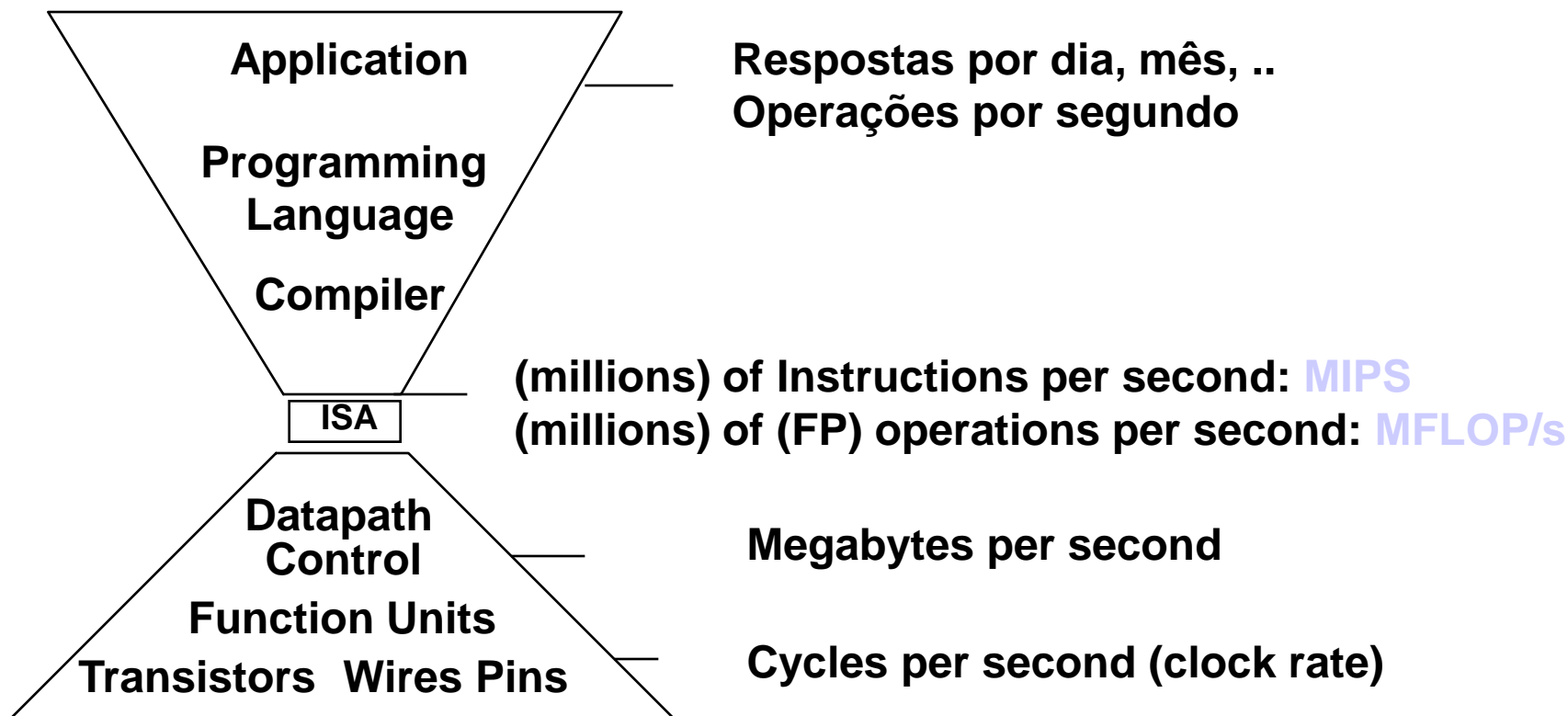
$$\frac{\text{ExTime}(Y)}{\text{ExTime}(X)} = \frac{\text{Performance}(X)}{\text{Performance}(Y)}$$

Velocidade do Concorde vs. Boeing 747

Throughput do Boeing 747 vs. Concorde

Métricas

Throughput



- Benchmarks, Traces, Mixes
- Hardware: custo, delay, área, consumo de energia
- Simulação (vários níveis)
 - ISA, RT, Gate, Circuito
- Teoria das Filas
- Regras Práticas
- “Leis”/Princípios

Benchmarks



IC-UNICAMP

- Aplicações Reais
 - Compiladores, processadores de texto, ...
 - Problema de portabilidade, difícil medir o tempo de execução
- Aplicações Modificadas
 - Melhora a portabilidade, pode ser refinado para medir um certo aspecto de interesse (exp: tempo de cpu)
- Kernels
 - Usados para avaliar características específicas
 - Livermore Loops, Linpack
- Toy Benchmarks
 - 10 a 100 linhas de código, fácil de programar, avaliação inicial
- Benchmarks Sintéticos
 - Semelhantes aos Kernels
 - Whetstone, Dhrystone

Benchmarks



IC-UNICAMP

- Desktop
 - SPEC (<http://www.spec.org>)
- Servidores
 - SPEC
- Sistemas Embarcados
 - EEMBC (Embedded Microprocessor Benchmark Consortium)
(<http://www.eembc.org>)
 - Automotivo
 - Consumidor
 - Rede
 - Automação de Escritório
 - telecomunicações

Benchmarks



IC-UNICAMP

SPEC: Standard Performance Evaluation Corporative

(<http://www.spec.org>)

- Primeira Versão - 1989
 - 10 programas (“SPECmarks”)
- Segunda Versão - 1992
 - SPECInt92 (6 programas)
 - SPECfp92 (14 programas)
 - Compiler Flags: livre
- Terceira Versão - 1995
 - SPECint95 (8 programas)
 - SPECfp95 (10 programas)
 - SPECint_base95, SPECfp_base95
 - “benchmarks útil por 3 anos”
 - Compiler Flags: controladas

Benchmarks



SPEC CPU2000 - CINT2000

| Programa | Linguagem | Finalidade |
|-------------|-----------|------------------------------------|
| 164.gzip | C | Compression |
| 175.vpr | C | FPGA Circuit Placement and Routing |
| 176.gcc | C | C Programming Language Compiler |
| 181.mcf | C | Combinatorial Optimization |
| 186.crafty | C | Game Playing: Chess |
| 197.parser | C | Word Processing |
| 252.eon | C++ | Computer Visualization |
| 253.perlbnk | C | PERL Programming Language |
| 254.gap | C | Group Theory, Interpreter |
| 255.vortex | C | Object-oriented Database |
| 256.bzip2 | C | Compression |
| 300.twolf | C | Place and Route Simulator |

<http://www.spec.org/osg/cpu2000/CINT2000/>

Benchmarks

SPEC CPU2000 - CFP2000



IC-UNICAMP

| Program | Linguagem | Finalidade |
|--------------|------------|---|
| 168.wupwise | Fortran 77 | Physics / Quantum Chromodynamics |
| 171.swim | Fortran 77 | Shallow Water Modeling |
| 172.mgrid | Fortran 77 | Multi-grid Solver: 3D Potential Field |
| 173.applu | Fortran 77 | Parabolic / Elliptic Differential Equations |
| 177.mesa | C | 3-D Graphics Library |
| 178.galgel | Fortran 90 | Computational Fluid Dynamics |
| 179.art | C | Image Recognition / Neural Networks |
| 183.quake | C | Seismic Wave Propagation Simulation |
| 187.facerec | Fortran 90 | Image Processing: Face Recognition |
| 188.ammp | C | Computational Chemistry |
| 189.lucas | Fortran 90 | Number Theory / Primality Testing |
| 191.fma3d | Fortran 90 | Finite-element Crash Simulation |
| 200.sixtrack | Fortran 77 | High Energy Physics Accelerator Design |
| 301.apsi | Fortran 77 | Meteorology: Pollutant Distribution |

<http://www.spec.org/osg/cpu2000/CFP2000/>



Exemplo: SpecINT2000

<http://www.spec.org/osg/cpu2000/results/res2000q3/cpu2000-20000718-00168.asc>

| Benchmarks | Base | Base | Base | Peak | Peak | Peak |
|------------------|----------|----------|-------|----------|----------|-------|
| | Ref Time | Run Time | Ratio | Ref Time | Run Time | Ratio |
| 164.gzip | 1400 | 277 | 505* | 1400 | 270 | 518* |
| 175.vpr | 1400 | 419 | 334* | 1400 | 417 | 336* |
| 176.gcc | 1100 | 275 | 399* | 1100 | 272 | 405* |
| 181.mcf | 1800 | 621 | 290* | 1800 | 619 | 291* |
| 186.crafty | 1000 | 191 | 522* | 1000 | 191 | 523* |
| 197.parser | 1800 | 500 | 360* | 1800 | 499 | 361* |
| 252.eon | 1300 | 267 | 486* | 1300 | 267 | 486* |
| 253.perlbnk | 1800 | 302 | 596* | 1800 | 302 | 596* |
| 254.gap | 1100 | 249 | 442* | 1100 | 248 | 443* |
| 255.vortex | 1900 | 268 | 710* | 1900 | 264 | 719* |
| 256.bzip2 | 1500 | 389 | 386* | 1500 | 375 | 400* |
| 300.twolf | 3000 | 784 | 382* | 3000 | 776 | 387* |
| SPECint_base2000 | | | 438 | | | |
| SPECint2000 | | | | | | 442 |

Intel OR840(1 GHz Pentium III processor)

Benchmarks

Como Apresentar o Desempenho?



Gerentes gostam de números.

Técnicos querem mais:

- Reprodutibilidade – informações que permitam que o experimento seja repetido (reproduzido)
- Consistência nos dados, ie se o experimento é repetido os dados devem ser compatíveis entre si

Como Apresentar os Dados?

| | Computador A | Computador B | Computador C |
|--------------------|--------------|--------------|--------------|
| Programa P1 (secs) | 1 | 10 | 20 |
| Programa P2 (secs) | 1000 | 100 | 20 |
| Total Time (secs) | 1001 | 110 | 40 |



Como Apresentar os Dados

- Média Aritmética (média aritmética ponderada)

$$\Sigma(T_i)/n \text{ or } \Sigma(W_i * T_i)$$

- Média Harmônica (média harmônica ponderada)

$$n/\Sigma(1/R_i) \text{ or } n/\Sigma(W_i/R_i)$$

- Média geométrica $(\Pi T_j / N_j)^{1/n}$

- Tempo de execução normalizado (e.g., X vezes melhor que SPARCstation 10 - [Spec](#))

- Não use média aritmética para tempos de execução normalizado (o resultado, quando comparado n máquinas, depende de qual máquina é usada como referência), use média geométrica



Como Apresentar os Dados

| máquina | A | B |
|------------|-----------|-----------|
| programa 1 | 10 => t1A | 20 => t1B |
| programa 2 | 30 => t2A | 5 => t2B |

Média aritmética normalizada em A:

$$(t_{1A}/t_{1A} + t_{2A}/t_{2A})/2 = 1 < (t_{1B}/t_{1A} + t_{2B}/t_{2A})/2 = 13/12$$

Média aritmética normalizada em B:

$$(t_{1A}/t_{1B} + t_{2A}/t_{2B})/2 = 13/4 > (t_{1B}/t_{1B} + t_{2B}/t_{2B})/2 = 1$$

CONTRADIÇÃO!!!!

Média Geométrica :

$$((t_{1A} * t_{2A}) / (t_{1A} * t_{2A}))^{-.5} = 1 > ((t_{1B} * t_{2B}) / (t_{1A} * t_{2A}))^{-.5} = (1/3)^{-.5} \Rightarrow$$

normalizado em A

$$((t_{1A} * t_{2A}) / (t_{1B} * t_{2B}))^{-.5} = 3^{-.5} > ((t_{1B} * t_{2B}) / (t_{1B} * t_{2B}))^{-.5} = 1 \Rightarrow$$

normalizado em B

1.9 Principles of Computer Design



- Take Advantage of Parallelism
 - e.g. multiple processors, disks, memory banks, pipelining, multiple functional units
- Principle of Locality
 - Reuse of data and instructions
- Focus on the Common Case
 - Amdahl's Law

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times \left((1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Abordagem Quantitativa

- Faça o caso comum ser mais rápido
- Amdahl's Law:
 - Relaciona o speedup total de um sistema com o speedup de uma porção do sistema

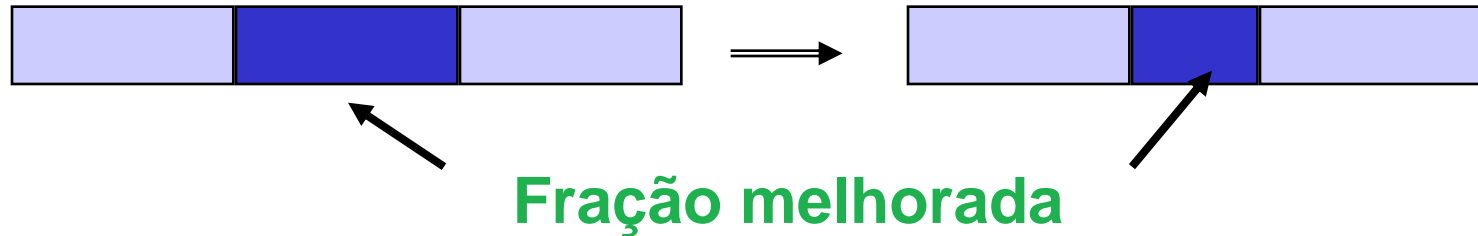
O speedup no desempenho obtido por uma melhoria é limitado pela fração do tempo na qual a melhoria é utilizada

Abordagem Quantitativa

Amdahl's Law

Speedup devido a uma melhoria E:

$$Speedup(E) = \frac{Execution_Time_Without_Enhancement}{Execution_Time_With_Enhancement} = \frac{Performance_With_Enhancement}{Performance_Without_Enhancement}$$



Abordagem Quantitativa

Amdahl's Law

Suponha que a melhoria E acelera a execução de uma fração F da tarefa de um fator S e que o restante da tarefa não é afetado pela melhoria E . Qual o speedup?

$$\begin{array}{l}
 T_{Old} = T_F + T_{nF} \\
 T_{New} = T_F/S + T_{nF}
 \end{array}
 \longrightarrow
 \frac{T_{Old}}{T_{New}} = \frac{T_F + T_{nF}}{\frac{T_F}{S} + T_{nF}} = \frac{T_F + T_{nF}}{\frac{T_F + ST_{nF}}{S}}$$

$$\text{Speedup} = \frac{S(T_F + T_{nF})}{T_F + ST_{nF}}$$

Lim $T_{nF} \rightarrow 0$?

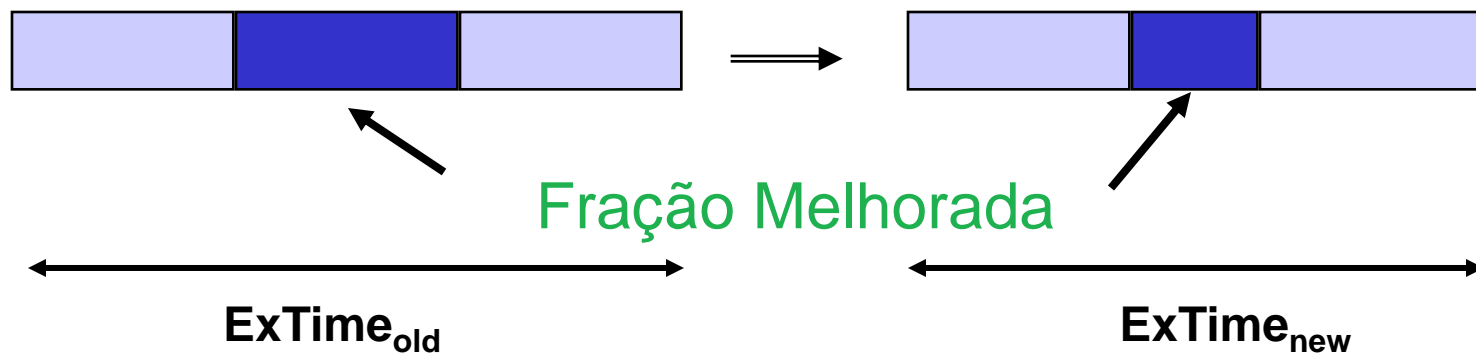
Lim $F \rightarrow 0$?

Abordagem Quantitativa

Amdahl's Law

$$\text{ExTime}_{\text{new}} = \text{ExTime}_{\text{old}} \times \left[(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right]$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{ExTime}_{\text{old}}}{\text{ExTime}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$



Abordagem Quantitativa

Amdahl's Law

- Exemplo: Suponha que as instruções de ponto flutuante foram melhoradas e executam 2 vezes mais rápidas, porém somente 10% das instruções, em um programa, são FP

$$\text{ExTime}_{\text{new}} = \text{ExTime}_{\text{old}} \times (0.9 + 0.1/2) = 0.95 \times \text{ExTime}_{\text{old}}$$

$$\text{Speedup}_{\text{overall}} = \frac{1}{0.95} = 1.053$$

Amdahl's Law



Execução de um programa em N processadores

$\text{Fraction}_{\text{enhanced}}$ = parallelizable part of program

$\text{Speedup}_{\text{enhanced}} = n$

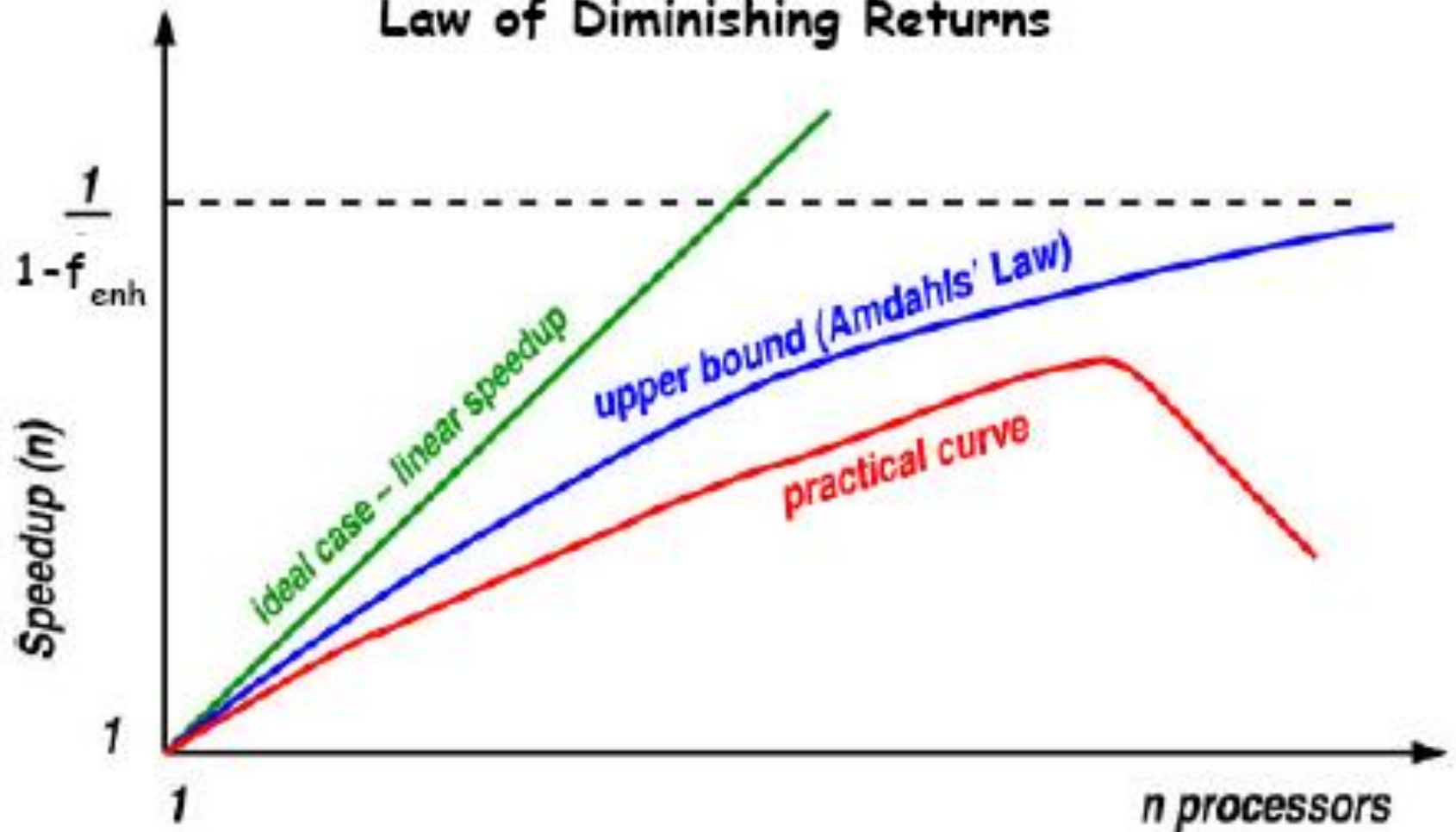
$$\text{ExTime}_{\text{new}} = \text{ExTime}_{\text{old}} (1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{ExTime}_{\text{old}} \times \text{Fraction}_{\text{enhanced}}}{n}$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{ExTime}_{\text{old}}}{\text{ExTime}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$\lim_{n \rightarrow \infty} \text{Speedup}_{\text{overall}} = 1 / (1 - \text{Fraction}_{\text{enhanced}})$$

Amdahl's Law - Graph

Law of Diminishing Returns





Example Suppose that we want to enhance the processor used for Web serving. The new processor is 10 times faster on computation in the Web serving application than the original processor. Assuming that the original processor is busy with computation 40% of the time and is waiting for I/O 60% of the time, what is the overall speedup gained by incorporating the enhancement?

Answer $\text{Fraction}_{\text{enhanced}} = 0.4$; $\text{Speedup}_{\text{enhanced}} = 10$; $\text{Speedup}_{\text{overall}} = \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56$

Exmpl P47: Amdhal

Amdahl's law expresses the law of diminishing returns: The incremental improvement in speedup gained by an improvement of just a portion of the computation diminishes as improvements are added. An important corollary of Amdahl's law is that if an enhancement is only usable for a fraction of a task then we can't speed up the task by more than the reciprocal of 1 minus that fraction.

A common mistake in applying Amdahl's law is to confuse "fraction of time converted to use an enhancement" and "fraction of time after enhancement is in use." If, instead of measuring the time that we *could use* the enhancement in a computation, we measure the time *after* the enhancement is in use, the results will be incorrect!

Amdahl's law can serve as a guide to how much an enhancement will improve performance and how to distribute resources to improve cost-performance. The goal, clearly, is to spend resources proportional to where time is spent. Amdahl's law is particularly useful for comparing the overall system performance of two alternatives, but it can also be applied to compare two processor design alternatives, as the following example shows.



Example A common transformation required in graphics processors is square root. Implementations of floating-point (FP) square root vary significantly in performance, especially among processors designed for graphics. Suppose FP square root (FPSQR) is responsible for 20% of the execution time of a critical graphics benchmark. One proposal is to enhance the FPSQR hardware and speed up this operation by a factor of 10. The other alternative is just to try to make all FP instructions in the graphics processor run faster by a factor of 1.6; FP instructions are responsible for half of the execution time for the application. The design team believes that they can make all FP instructions run 1.6 times faster with the same effort as required for the fast square root. Compare these two design alternatives.

Answer We can compare these two alternatives by comparing the speedups:

$$\text{Speedup}_{\text{FPSQR}} = \frac{1}{(1 - 0.2) + \frac{0.2}{10}} = \frac{1}{0.82} = 1.22$$

$$\text{Speedup}_{\text{FP}} = \frac{1}{(1 - 0.5) + \frac{0.5}{1.6}} = \frac{1}{0.8125} = 1.23$$

Improving the performance of the FP operations overall is slightly better because of the higher frequency.

Exmpl
P47:
Amdhal



Exmpl P48: Amdhal

Amdahl's law is applicable beyond performance. Let's redo the reliability example from page 35 after improving the reliability of the power supply via redundancy from 200,000-hour to 830,000,000-hour MTTF, or 4150X better.

Example The calculation of the failure rates of the disk subsystem was

$$\begin{aligned} \text{Failure rate}_{\text{system}} &= 10 \times \frac{1}{1,000,000} + \frac{1}{500,000} + \frac{1}{200,000} + \frac{1}{200,000} + \frac{1}{1,000,000} \\ &= \frac{10 + 2 + 5 + 5 + 1}{1,000,000 \text{ hours}} = \frac{23}{1,000,000 \text{ hours}} \end{aligned}$$

Therefore, the fraction of the failure rate that could be improved is 5 per million hours out of 23 for the whole system, or 0.22.

Answer The reliability improvement would be

$$\text{Improvement}_{\text{power supply pair}} = \frac{1}{(1 - 0.22) + \frac{0.22}{4150}} = \frac{1}{0.78} = 1.28$$

Despite an impressive 4150X improvement in reliability of one module, from the system's perspective, the change has a measurable but small benefit.

- The Processor Performance Equation

CPU time = CPU clock cycles for a program \times Clock cycle time

$$\text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

$$\text{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}}$$

CPU time = Instruction count \times Cycles per instruction \times Clock cycle time

$$\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}} = \frac{\text{Seconds}}{\text{Program}} = \text{CPU time}$$

- Different instruction types having different CPIs

$$\text{CPU clock cycles} = \sum_{i=1}^n \text{IC}_i \times \text{CPI}_i$$

$$\text{CPU time} = \left(\sum_{i=1}^n \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock cycle time}$$



Exmpl P50: Amdhal (1)

Example Suppose we have made the following measurements:

Frequency of FP operations = 25%

Average CPI of FP operations = 4.0

Average CPI of other instructions = 1.33

Frequency of FPSQR = 2%

CPI of FPSQR = 20

Assume that the two design alternatives are to decrease the CPI of FPSQR to 2 or to decrease the average CPI of all FP operations to 2.5. Compare these two design alternatives using the processor performance equation.

Answer First, observe that only the CPI changes; the clock rate and instruction count remain identical. We start by finding the original CPI with neither enhancement:

$$\begin{aligned} \text{CPI}_{\text{original}} &= \sum_{i=1}^n \text{CPI}_i \times \left(\frac{\text{IC}_i}{\text{Instruction count}} \right) \\ &= (4 \times 25\%) + (1.33 \times 75\%) = 2.0 \end{aligned}$$

We can compute the CPI for the enhanced FPSQR by subtracting the cycles saved from the original CPI:

$$\begin{aligned} \text{CPI}_{\text{with new FPSQR}} &= \text{CPI}_{\text{original}} - 2\% \times (\text{CPI}_{\text{old FPSQR}} - \text{CPI}_{\text{of new FPSQR only}}) \\ &= 2.0 - 2\% \times (20 - 2) = 1.64 \end{aligned}$$

We can compute the CPI for the enhancement of all FP instructions the same way or by summing the FP and non-FP CPIs. Using the latter gives us:

$$\text{CPI}_{\text{new FP}} = (75\% \times 1.33) + (25\% \times 2.5) = 1.625$$

Since the CPI of the overall FP enhancement is slightly lower, its performance will be marginally better. Specifically, the speedup for the overall FP enhancement is

$$\begin{aligned} \text{Speedup}_{\text{new FP}} &= \frac{\text{CPU time}_{\text{original}}}{\text{CPU time}_{\text{new FP}}} = \frac{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{original}}}{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{new FP}}} \\ &= \frac{\text{CPI}_{\text{original}}}{\text{CPI}_{\text{new FP}}} = \frac{2.00}{1.625} = 1.23 \end{aligned}$$

Happily, we obtained this same speedup using Amdahl's law on page 46.

Exmpl P50: Amdhal (2)

1.10 Putting all together: performance, price and power

- Benchmark `ssj_ops`: server side java ops per sec)
 - exercita não só o processador (como o SPEC) mas também: caches, memória, interconexão
- Tabela: comparação de desempenho e desempenho / \$ (winner : maior n^o de cores)
- Figura: duas curvas
 - (`ssj_ops / watt`) vs workload
 - potência média (watts) vs workload
- $\text{ssj_ops} / \text{watt} = \text{ssj_operations} / \text{joule}$
- SPECpower usa
 - $\sum \text{ssj_ops} (p \text{ cada workload}) / \sum \text{power}$
 - somas p cada valor de workload
- Se compararmos (desemp/watt)/ preço, agora o winner é o R710 (resultado inverso do anterior)



Fig. 1.18: Servidores da Dell

System 1

System 2

System 3

| Component | System 1 | | System 2 | | System 3 | |
|-----------------|----------------|---------------|----------------|---------------|----------------|-----------------|
| | | Cost (% Cost) | | Cost (% Cost) | | Cost (% Cost) |
| Base server | PowerEdge R710 | \$653 (7%) | PowerEdge R815 | \$1437 (15%) | PowerEdge R815 | \$1437 (11%) |
| Power supply | 570 W | | 1100 W | | 1100 W | |
| Processor | Xeon X5670 | \$3738 (40%) | Opteron 6174 | \$2679 (29%) | Opteron 6174 | \$5358 (42%) |
| Clock rate | 2.93 GHz | | 2.20 GHz | | 2.20 GHz | |
| Total cores | 12 | | 24 | | 48 | |
| Sockets | 2 | | 2 | | 4 | |
| Cores/socket | 6 | | 12 | | 12 | |
| DRAM | 12 GB | \$484 (5%) | 16 GB | \$693 (7%) | 32 GB | \$1386 (11%) |
| Ethernet Inter. | Dual 1-Gbit | \$199 (2%) | Dual 1-Gbit | \$199 (2%) | Dual 1-Gbit | \$199 (2%) |
| Disk | 50 GB SSD | \$1279 (14%) | 50 GB SSD | \$1279 (14%) | 50 GB SSD | \$1279 (10%) |
| Windows OS | | \$2999 (32%) | | \$2999 (33%) | | \$2999 (24%) |
| Total | | \$9352 (100%) | | \$9286 (100%) | | \$12,658 (100%) |
| Max ssj_ops | 910,978 | | 926,676 | | 1,840,450 | |
| Max ssj_ops/\$ | 97 | | 100 | | 145 | |

Figure 1.18 Three Dell PowerEdge servers being measured and their prices as of August 2010. We calculated the cost of the processors by subtracting the cost of a second processor. Similarly, we calculated the overall cost of memory by seeing what the cost of extra memory was. Hence, the base cost of the server is adjusted by removing the estimated cost of the default processor and memory. Chapter 5 describes how these multi-socket systems are connected together.

Fig. 1.19: Preço/desempenho

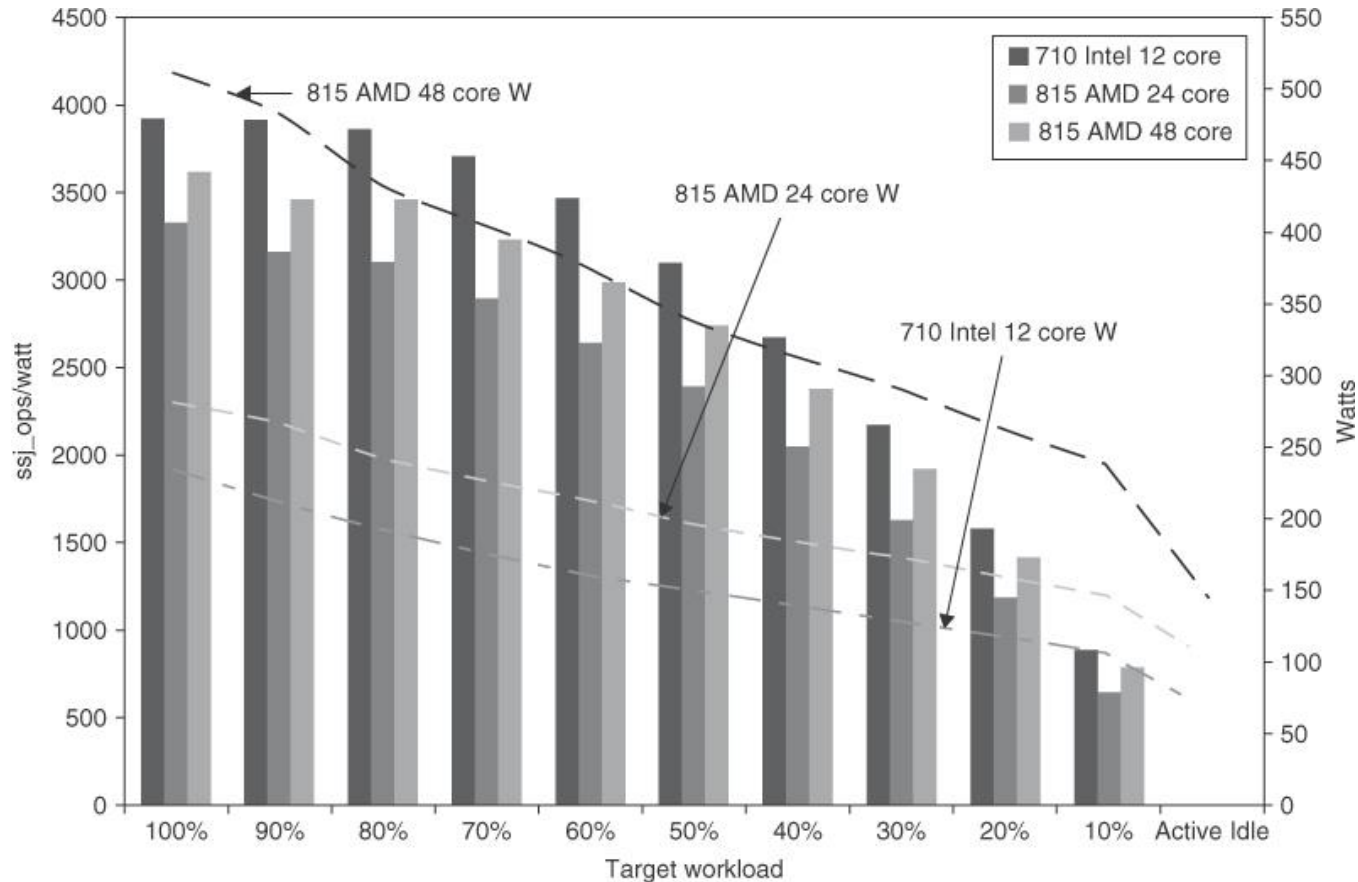


Figure 1.19 Power-performance of the three servers in Figure 1.18. Ssj_ops/watt values are on the left axis, with the three columns associated with it, and watts are on the right axis, with the three lines associated with it. The horizontal axis shows the target workload, as it varies from 100% to Active Idle. The Intel-based R715 has the best ssj_ops/watt at each workload level, and it also consumes the lowest power at each level.