Topic 01: Introduction to Quantitative Analysis

1. Trends in computing systems
2. Quantifying performance
3. Quantifying power
Moore’s law

- Started with 10 um feature size in 1970. We are now at 14 nm feature size.
- Number of transistors per unit area doubles every 2-3 area.
- More transistors in die → more capable cores and/or more cores, GPU, accelerating functional units as in SoCs
- Transistors can switch faster with new technology
1. Process technology → more design per unit area → build more powerful cores (more ILP) and/or add more cores (TLP)
2. Process technology → faster transistors → higher clock rate
3. Deeper pipelines → higher clock
4. Better circuit design techniques and better CAD tools
Evolution of clock rate

- Dynamic power proportional to $fV^2$
- Increasing the frequency with less than ideal voltage scaling leads to increased power density
Power wall

Economical Heat removal mechanisms (e.g., air and liquid) limit the maximum amount of power consumption
Lack of parallelism wall

- Programming parallel applications is hard
- Sometimes applications do not scale; synchronization could be a bottleneck
- Example: speedup of PARSEC benchmarks on a 8-core Xeon server

- Circumventing the parallelism wall: incorporate heterogeneous functional units on die (e.g., GPU, accelerators)
Memory wall

- Unlike processor performance, DRAM performance improved by 7% a year → a memory wall.

- Density followed Moore’s law

- Memory wall = memory_cycle/processor_cycle

- In 1990, it was about 4 (25 MHz, 150 ns). Grew to 200 until 2002 but has tapered off since then.

- Although still a big problem, the memory latency wall stopped growing around 2002.

- With the advent of multicore microarchitectures the memory problem has shifted from latency to bandwidth
Summary of current and future challenges to computing

- **Memory wall**
  - Increasing number of cores requires increased memory bandwidth; otherwise, starvation and stalling occurs

- **Parallelism wall**
  - Some applications lack enough ILP or TLP → not much benefit from aggressive superscalar or many-core designs

- **Power wall**
  - Limits on heat removal imposes a limit on power density and frequency of operation

- **Power and parallelism wall** → dark silicon (only a small portion of the chip is operational at any moment of time)
Performance metrics

1. **execution time, latency, response time**: time to complete a task

2. **throughput**: number of tasks (e.g., instructions, queries, frames rendered) completed per unit time.

- Is throughput = 1/av. response time?
  - Only if NO overlap
  - Otherwise, throughput > 1/av. response time
Which benchmarks?

1. Real programs (mpeg encoding)
2. Synthetic benchmarks (e.g., measuring I/O storage bandwidth)
3. Kernels
4. Toy benchmarks (e.g., quicksort)
5. Benchmark suites:
   • SPEC: Standard Performance Evaluation Corporation
     – SPEC CPU Integer point
     – SPEC CPU Floating point
     – SPEC POWERSSJ transactional
     – SPEC Viewperf for GPU performance
   • PARSEC for multi-threaded applications
   • Rodinia for GPGPU performance
   • NASA Parallel benchmarks (NPB) for clusters (e.g., FFT)
   • HPC Challenge benchmarks (HPCC) for clusters (e.g., linear solver)
Examples of benchmark results

Runtime and average processor power for SPEC CPU2006 benchmarks using AMD Phenom II X4 965 Black edition at 3.4 GHz and 4GB DRAM running Linux 2.6.10.8
Reporting performance for a set of programs

• **Arithmetic mean:** \( \frac{\sum_{i=1}^{N} T_i}{N} \) or \( \sum_{i=1}^{N} \omega_i T_i \)

  problem is that programs with longest execution delays can dominate the result

• **Reporting speedups (Why?):**
  – Speed-up measures the advantage of a machine over reference machine \( R \) for program \( i \): \( S_i = \frac{T_{R,i}}{T_i} \)
  
  – Arithmetic mean of speedups: \( \sum_{i} S_i/N \)

  – Geometric mean of speedups: \( \sqrt[N]{\prod_{i} S_i} \) What is the advantage?

  – Harmonic mean: \( \frac{N}{\sum_{i} 1/S_i} \)
### Example

Which is better machine 1 or machine 2?

<table>
<thead>
<tr>
<th></th>
<th>Program A</th>
<th>Program B</th>
<th>Arithmetic Mean</th>
<th>Ratio of means (ref X)</th>
<th>Ratio of means (ref Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine 1</td>
<td>10 sec</td>
<td>100 sec</td>
<td>55 sec</td>
<td>91.8</td>
<td>10</td>
</tr>
<tr>
<td>Machine 2</td>
<td>1 sec</td>
<td>200 sec</td>
<td>100.5 sec</td>
<td>50.2</td>
<td>5.5</td>
</tr>
<tr>
<td>Reference X</td>
<td>100 sec</td>
<td>10000 sec</td>
<td>5050 sec</td>
<td>50.2</td>
<td>5.5</td>
</tr>
<tr>
<td>Reference Y</td>
<td>100 sec</td>
<td>1000 sec</td>
<td>550 sec</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Program A</th>
<th>Program B</th>
<th>Arithmetic</th>
<th>Harmonic</th>
<th>Geometric</th>
</tr>
</thead>
<tbody>
<tr>
<td>speedup wrt Reference X</td>
<td>Machine 1</td>
<td>10</td>
<td>100</td>
<td>55</td>
<td>18.2</td>
</tr>
<tr>
<td></td>
<td>Machine 2</td>
<td>100</td>
<td>50</td>
<td>75</td>
<td>66.7</td>
</tr>
<tr>
<td>speedup wrt Reference Y</td>
<td>Machine 1</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Machine 2</td>
<td>100</td>
<td>5</td>
<td>52.5</td>
<td>9.5</td>
</tr>
</tbody>
</table>
When to use the harmonic mean?

- Consider a processor that executed instructions for the first 10 billion instructions at a rate of 1 BIPS (billion instructions per second) and then for the second 10 billion instructions at a rate of 2 BIPS, what is the average instruction rate?

- Average BIPS = (1+2)/2 = 1.5 WRONG
- Average BIPS = (10 + 10)/(10/1 + 10/2) = 20/15 = 1.33

- Harmonic mean of rates =

\[
\frac{n}{\sum_{i=1}^{n} \frac{1}{\text{rate}(i)}}
\]

- Use HM if forced to start and end with rates (e.g. reporting CPI or miss rates or branch misprediction rates)
Performance metrics for clusters

• **Supercomputers:**
  – Execution time
  – FLOPS (FLOP/s): theoretical peak or using a standard benchmark (e.g., LINPACK is used for Top-500 supercomputer ranking)

• **Warehouse scale:**
  – Latency is important metric because it is seen by users
  – Bing study: users will use search less as response time increases
  – Service Level Objectives (SLOs)/Service Level Agreements (SLAs). E.g. 99% of requests be below 100 ms
Amdahl’s law

- Enhancement $E$ accelerates a fraction $F$ of a task by a factor of $S$
  \[
  \text{speedup} = \frac{T_{exe}(\text{without } E)}{T_{exe}(\text{with } E)} = \frac{1}{(1 - F) + \frac{F}{S}}
  \]

- Enhancement is limited by the fraction of execution time that can’t be enhanced → law of diminishing returns

- Amdahl’s law → optimize the common case
Physical reasons for power consumption

- If transistor input voltage above $V_t$ → transistor is ON (short circuit); otherwise, it is off (open circuit)
- **Dynamic power** is consumed when transistors switch status.
- **Static or leakage power** is consumed when there is no switching (historically negligible but growing in significance with nanoscale CMOS)
1. Static (leakage) power

\[ P_{static} = VI_{sub} \propto Ve^{-\frac{KV_t}{T}} \]

- When input voltage is less than \( V_t \), transistor should be off; however, some electrons are still able to go through because of reductions in threshold voltage (\( V_t \)) of recent technology \( \rightarrow \) static or leakage power consumption.
- Leakage current is exponentially dependent on \( V_t \) as well as the operating temperature (\( T \)).
- As \( V_t \) decreases, static power increases exponentially \( \rightarrow \) switch to 3D transistors was mainly motivated to control leakage power.
- Noise also limits reducing \( V_{th} \)
2. Dynamic power

- \( P_{\text{dynamic}} = \alpha f CV^2 \)
  - \( \alpha \) is fraction of clock cycles when gate switches
- At a particular design & technology nodes, higher frequency demands higher voltage.
- If chip size grows then total power grows
- Non-ideal scaling of \( V_t \) \( \rightarrow \) Non-ideal scaling of \( V \) \( \rightarrow \) non-ideal scaling of power.
- Power dissipation leads to heat generation \( \rightarrow \) when heat is not removed appropriately, it causes thermal hot spots \( \rightarrow \) problems to reliability and leakage power.

[Reda et al, TComp 11]
Reasons for the power wall

- Scaling rules from one technology node to another
  - Area scales by $\frac{1}{2}$
  - C scales by $\frac{1}{\sqrt{2}}$

- Let $N$ be #cores at 45 nm and $p$ be power per core. Total power = $Np$

- Assuming same frequency and total chip area at 32 nm,
  power at 45 nm = $2N\frac{p}{\sqrt{2}S}$, where $S=$ (old voltage/new voltage)$^2$.

- 45 nm voltage = 1.1 V, 32 nm voltage = 1.0 V \(\rightarrow S = 1.21 \rightarrow \text{less than} \sqrt{2} \rightarrow \text{power density increases.}\)
Combined metrics for performance and power

- Energy: cost paid by the user \[ E = \int_{\text{start}}^{\text{finish}} p(t) dt \]
- Joules per instruction (EPI)
- Energy delay product (EDP)
- MIPS/W and FLOPS/W
- For clusters: Power Utilization Effectiveness (PEU) = Total facility power / IT equipment power

[datacenterexperts.com]
Summary

1. Trends: power wall, memory wall, parallelism wall. Frequency increases → power wall → multi-core → parallelism wall → fusion

2. Quantifying performance: response time and throughput, computing means (arithmetic, geometric, harmonic)

3. Quantifying power: static and dynamic, origins of the power wall.