Advanced Computer Architecture

Exercise 1

Prof. Dr.-Ing. Axel Hunger
Department of Electrical Engineering and Information Technology
Institute for Computer Engineering

Dipl.-Ing. M.A. Lebedev
Institute for Computer Engineering
BB 321, Tel: 0203 – 379 - 1019
E-mail: michail.lebedev@uni-due.de

Link for downloading ACA foil

Table of content

1. Speedup, throughput, efficiency
2. Amdahl's Law
3. Scalability
4. Speedup by pipelining
5. Multiple choice (examination preparation)

Literature

• Books
1. Multiple Choice: General Questions

1. Who is regarded as the founder of Computer Architecture?
   a) Alan Turing
   b) Konrad Zuse
   c) John von Neumann
   d) John William Mauchly
   e) None of the answers above is correct

2. What is characteristic for the organization of a computer architecture?
   a) Size
   b) Dynamic behaviour
   c) Static behaviour
   d) Speed
   e) None of the answers above is correct

3. What is usually regarded as the von Neumann Bottleneck?
   a) Processor/memory interface
   b) Control unit
   c) Arithmetic logical unit
   d) Instruction set
   e) None of the answers above is correct

4. How does the number of transistors per chip increase according to Moore’s law?
   a) Quadratically
   b) Linearly
   c) Cubicly
   d) Exponentially
   e) None of the answers above is correct
1. Multiple Choice: General Questions

5. Who is regarded as the founder of Computer Science?
   - a) Alan Turing
   - b) Konrad Zuse
   - c) J. Presper Eckert
   - d) John William Mauchly
   - e) None of the answers above is correct

6. Which is the fastest storage unit in a usual memory hierarchy?
   - a) Cache
   - b) Main memory
   - c) Hard disk
   - d) Register
   - e) None of the answers above is correct

7. Which cache miss does not occur in case of a fully associative cache?
   - a) Conflict miss
   - b) Capacity miss
   - c) Compulsory miss
   - d) Cold start miss
   - e) None of the answers above is correct

8. Which miss even occurs in infinite caches?
   - a) Coherence miss
   - b) Capacity miss
   - c) Conflict miss
   - d) Cold start miss
   - e) None of the answers above is correct
1. Multiple Choice: General Questions

9. What is stored in a Translation Lookaside Buffer?
   - a) System dumps
   - b) Physical addresses
   - c) Program data
   - d) Operating system log files
   - e) None of the answers above is correct

---

### Execution time, Throughput, Speedup

<table>
<thead>
<tr>
<th>Aeroplane</th>
<th>NY to Paris</th>
<th>Speed</th>
<th>Passengers</th>
<th>Throughput (Persons/h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing 747</td>
<td>6.5 h</td>
<td>610 mph</td>
<td>470</td>
<td>72.3</td>
</tr>
<tr>
<td>Concorde</td>
<td>3 h</td>
<td>1350 mph</td>
<td>132</td>
<td>44.0</td>
</tr>
</tbody>
</table>

- Execution time T (response time, latency) [sec], [h], ...
- Throughput X (bandwidth) [1/sec], [1/h], ....

→ Question not precise!
**Definition of Speedup**

**Speedup S (Acceleration):**

\[
A \text{ is } S \text{ times faster than } B
\]

\[
S = \frac{T(B)}{T(A)} = \frac{6.5h}{3h} = 2.167
\]

Speedup is a measure for the judgement of the processing of a single task (passenger).

**Throughput** is a measure for the judgement of the processing of the whole workload (with what aeroplane type can an airline transport more passengers?).

---

**Performance Goal**

- **Speedup**

\[
\text{speedup}(p \text{ processors}) = \frac{\text{performance}(p \text{ processors})}{\text{performance}(\text{1 processor})}
\]

- Scientific computing: performance=1/time

\[
\text{speedup}(p \text{ processors}) = \frac{\text{time(\text{1 processor})}}{\text{time(\text{p processors})}}
\]

- **Efficiency**

\[
\text{efficiency}(p \text{ processors}) = \frac{\text{speedup}(p \text{ processors})}{p}
\]
Speedup based on Throughput

- Performance = throughput = transactions / minute

\[ \text{speedup}(p \text{ processors}) = \frac{\text{tpm}(p \text{ processor})}{\text{tpm}(1 \text{ processor})} \]

Amdahl’s Law (1)

The performance gain that can be obtained by improving some portion of a computer can be calculated using Amdahl’s Law. Amdahl’s Law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.

Amdahl’s Law defines the speedup that can be gained by using a particular feature. What is speedup? Suppose that we can make an enhancement to a machine that will improve performance when it is used.

Speedup is the ratio of the performance for the entire task using the enhancement when possible to the performance for the entire task without using the enhancement.

\[ \text{Speedup} = \frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}} \]

Alternatively,

\[ \text{Speedup} = \frac{\text{Execution time for entire task without using the enhancement}}{\text{Execution time for entire task using the enhancement when possible}} \]

Speed tells us how much faster a task will run using the machine with the enhancement as opposed to the original machine.
Amdahl’s Law (2)

Amdahl’s Law gives us a quick way to find the speedup from some enhancement, which depends on two factors:

1. The fraction of the computation time in the original machine that can be converted to take advantage of the enhancement
2. The improvement gained by the enhanced execution mode; that is, how much faster the task would run if the enhanced mode were used for the entire program

The execution time using the original machine with the enhanced mode will be the time spent using the unenhanced portion of the machine plus the time spent using the enhancement:

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

The overall speedup is the ratio of the execution times:

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

Amdahl’s Law (3)

In 1967, Gene Amdahl (developer of the IBM 360/xx computer) defined the performance increase of a program with fixed problem size for parallel processing as:

\[
\text{Speedup} \quad S(p) = \frac{T_s}{f \times T_s + (1 - f) \times T_s/p}
\]

with

- \(T_s\) : Execution time for sequential processing of the whole task
- \(f\) : Fraction of the execution time for program segments which cannot run in parallel \((f = 0..1)\)
- \(p\) : Number of parallel processing elements (processors)

For \(p \to \infty\) : \(S(p) = 1 / f\), or for \(f \to 0\) : \(S(p) = p\)
For efficient parallel processing it is necessary to achieve speedups that are close to the number of processors used.

\[ S(p) = p \text{ (ideal)} \]
\[ S(p) < p \text{ (real)} \]
### Definition of Efficiency

Efficiency defines the ratio of speedup and number of processors used. Efficiency indicates which share of the processor performance can be utilized.

\[
E(p) = \frac{S(p)}{p} = \frac{T_{\text{sequential}}}{p \times T_{\text{parallel}}}
\]

with \(0 < E(p) \leq 1\)

### 2. Multiple Choice: Speedup and Amdahl's law

1. Which value has the speedup of a parallel program that achieves an efficiency of 75% on 32 processors?

   a) 18  
   b) 24  
   c) 16  
   d) 20  
   e) None of the answers above is correct

2. Which speedup could be achieved according to Amdahl’s law for infinite number of processors if 2.5% of a program is sequential and the remaining part is ideally parallel?

   a) Infinite speedup  
   b) 40  
   c) 4  
   d) 400  
   e) None of the answers above is correct
2. Multiple Choice: Speedup and Amdahl's law

3. Which speedup could be achieved according to Amdahl's law for 9 processors if 10% of a program is sequential and the remaining part is ideally parallel?

   a) $5 T_s$
   b) 5
   c) 1.098
   d) None of the answers above is correct

4. Which is the speedup that can be obtained on 100 processors if 93% of the program is ideally parallel, the remaining part is sequential and the sequential execution time is 10000 s?

   a) 12.6
   b) 10.6
   c) 1.075
   d) None of the answers above is correct

5. Suppose that we are considering an enhancement to the processor of a server system used for Web serving. The new CPU is 10 times faster on computation in the Web serving application than the original processor. Assuming that the original CPU 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?

   a) 1
   b) 2
   c) 1.56
   d) None of the answers above is correct

$$\text{Fraction}_{\text{enhanced}} = 0.4$$
$$\text{Speedup}_{\text{enhanced}} = 10$$
$$\text{Speedup}_{\text{overall}} = \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} \approx 1.56$$
2. Multiple Choice: Speedup and Amdahl's Law

5. A common transformation required in graphics engines 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 graphics processor run faster by a factor of 1.6; FP instructions are responsible for a total of 50% 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.

We can compare these two alternatives by comparing the speedups:

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

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

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

---

Summary of the benefits of Amdahl's Law (1)

Amdahl’s Law expresses the law of diminishing returns: The incremental improvement in speedup gained by an additional improvement in the performance 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, 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!
Summary of the benefits of Amdahl's Law (2)

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 CPU design alternatives.

Scalability

A computer architecture or a program is scalable if the efficiency of program processing remains constant for increasing processor number.

In general, this is only possible for a simultaneous increase of the problem size.

A program (an algorithm) is perfectly scalable if a linear increase of \( n \) is enough in case of a linear increase of \( p \) to achieve constant efficiency.

**Speedup is usually reduced by additional parallel overhead:**

\[
V(p) = p * T(p) - T(seq)
\]
Definition of mean parallel overhead:

\[ \nabla(p) = \frac{V(p)}{p} \]

Causes:
- Startup costs of an event (process or communication start)
- Costs for the distribution/administration of shared data
- Costs for synchronization

What is better?
- Less communication by bigger work packages for fewer processors (from fine to coarse granularity)
- Smaller work packages distributed to more processors

3. Multiple Choice: Scalability

1. Which is the sequential execution time of a program with mean parallel overhead 2 s and parallel execution time 7 s on 100 processors?
   - a) 603 s
   - b) 797 s
   - c) 500 s
   - d) None of the answers above is correct

2. The sequential execution time of a program is 500 s, its parallel execution time is 8 s and the mean parallel overhead of the system is 3 s. How many processors are in the system?
   - a) 163
   - b) 63
   - c) 100
   - d) None of the answers above is correct
Problem 6: Scalability

3. Which is the sequential execution time of a program with mean parallel overhead 3s and parallel execution time 8s if 10% of the program is sequential and the remaining part is ideally parallel? (The speedup is considered as 5)
   a) 65 s
   b) 45 s
   c) 42.5 s
   d) None of the answers above is correct

4. Which is the parallel execution time of a program with mean parallel overhead 1 s and sequential execution time 90 sec for 10 processors?
   a) 10 s
   b) 9 s
   c) 11 s
   d) 8 s
   e) None of the answers above is correct

Invention of the Assembly Line (Pipeline Principle)

In 1913, Henry Ford was the first to introduce the pipeline principle into industrial fabrication with production start of Tin Lizzy.

Principle:
A complex task is distributed into a sequence of simple partial tasks.

Advantage:
• Higher throughput (Lizzies / h)
• Simplification (and thus faster processing) of partial tasks (CISC → RISC)
Exercise 1

Sequential Laundry

The sequential laundry needs 6 hours for 4 tasks.

PipeLine Laundry

The pipeline laundry needs 3.5 hours for 4 tasks.
Exercise 1

What does Pipelining Teach Us?

- Pipelining does not improve the execution time of a single task; it improves the throughput of the total work load.
- Pipeline speed is limited by the slowest pipeline stage.
- Several tasks are processed simultaneously.
- Possible speedup = Number of pipeline (pipe) stages
- Speedup reduction by imbalanced lengths of the pipe stages
- Time for filling and emptying of the pipeline must be considered.

Speedup for Pipelining

- For efficient pipeline use, the n number of tasks (processes) to be processed should be as high as possible
- For a single task, a pipeline works purely sequentially
- For sequential processing of a task, a pipeline with k equal stages and execution time T per stage needs

For a single task: \[ T_1 = k \times T \]

For n tasks without pipelining: \[ T_{seq} = n \times T_1 = n \times k \times T \]

For n tasks and pipelining:

\[ T_{pipe} = (k - 1) \times T + n \times T = (k - 1 + n) \times T \]
### Exercise 1: Speedup for Pipelining

\[
S = \frac{T_{seq}}{T_{pipe}} = \frac{n \cdot k}{k - 1 + n}
\]

For \( n = 1 \) (only one task) \( \Rightarrow S = 1 \), i.e. same behaviour as sequential processing.

For \( n \gg k \Rightarrow S = k \), i.e. speedup grows for pipes which are always well filled with the number of stages.

If we define a mean overhead of a pipe on the analogy of \( V(p) \) we obtain:

\[
V(k) = \frac{(k \cdot T_{pipe} - T_{seq})}{k} = (k - 1) \cdot T
\]

### Multiple Choice: Speedup by Pipelining

1. Which is the speedup of pipelining that can be obtained with 7 tasks if the mean overhead of a pipeline is considered as 5 and an execution time per stage of 1 cycle?
   - a) 3.5
   - b) 3
   - c) 6
   - d) None of the answers above is correct

2. Which is the mean overhead of a pipeline with 5 stages and an execution time per stage of 1 cycle?
   - a) 2 cycles
   - b) 3 cycles
   - c) 4 cycles
   - d) None of the answers above is correct
3. Multiple Choice: Speedup by Pipelining

3. What is the execution time per stage of a pipeline that has 5 equal stages and a mean overhead of 12 cycles?

   a) 2 cycles
   b) 3 cycles
   c) 4 cycles
   d) None of the answers above is correct

Summary

1. Speedup, throughput, efficiency
2. Amdahl’s Law
3. Scalability
4. Speedup by pipelining
5. Multiple choice (examination preparation)

Next Lecture: Pipelines