Computer Architecture III Study Guide Introduction

Maximillien Courchesne-Mackie (mcour010@uottawa.ca)

I. Performance Analysis

Due to the large number of different systems in existence, metrics for measuring performance across these systems must exist. These metrics include the speedup of one system compared to another and it's efficiency. Additionally, we can observe the degree of a system's parallelism. Scaling or growing a system to be better than another comes with a set of challenges, which will also be discussed.

A. Speedup

Speedup is a measure of how fast a parallel algorithm is compared to a serial one. This can be calculated using the following formula:

$$S_p = \frac{T(1)}{T(n)}$$

Where S_p is the speedup, $T(1)$ is the execution time of the serial algorithm, $T(n)$ is the execution time of the parallel algorithm and n is the number of processors.

When a communication overhead exists in a system, the speedup can be defined as:

$$S = \frac{n}{xn + (1-x) + \frac{nt_c}{T(1)}}$$

Where t_c is the synchronization and communication overhead.

1) Amdahl's Law: Amdahl's law can be used to determine the maximum speedup that can be achieved when only part of a system can be improved. A core concept of this law is that a system's speedup is limited by it's serial parts. Adding several processors can improve the parallel part, but it will never be able to help the serial part.

$$T(n) = xT(1) + \frac{(1-x)T(1)}{n}$$
$$S = \frac{n}{xn + (1-x)}$$

Where x is the fraction of the system that operates in a fully sequential mode and n is the number of processors.

2) Gustafson's Law:

B. Efficiency

Efficiency is a measure of the speedup achieved per processor. It expresses how well-utilized the processors are in solving an algorithm. Algorithms with linear speedup have an efficiency of 1.

$$E_p = \frac{S}{n} = \frac{T(1)}{pT(n)}$$

C. Degree of Parallelism

For each time period, the number of processors being used is known as the degree of parallelism (DOP). When the DOP is plotted as a function of time, we get a parallelism profile of a given program or algorithm.

The average parallelism is computed using the following formula:

$$A = \frac{\sum\limits_{i=1}^{m} it_i}{\sum\limits_{i=1}^{m} t_i}$$

Where m is the maximum parallelism in a profile and t_i is the total amount of time that $DOP = i \sum_{i=1}^{m} t_i = t_2 - t_1$

D. Performance Challenges

System designs are often challenged by certain technological limitations. These include power, energy, reliability, wire delays complexity and the CMOS endpoint.

Dynamic power is the amount of energy consumed by a gate when it switches states. This is the limiting factor for the clock frequency of processors and is why the current trend is towards slower multiprocessor systems.

The **static power** is energy that is continuously dissipated by an electronic circuit. Seeing as transistors are not perfect on/off gates, some power is let through when the transistor is supposed to be in an off state. In the past, these numbers were negligible however, as the threshold voltage decreases, the power leakage is getting worse.

As processors and their components get smaller and smaller, the **reliability** of the system decreases. Notably, this is a problem in memory where transient, temporary, intermittent and permanent faults occur. To combat these faults, failsafes must be added to multiprocessing systems.

While transistors switch faster due to being smaller, the **wire delays** do not scale as well. The propagation delay of electricity in a wire is governed by its resistance. Signals propagate faster in small, thin wires and slower in wires with larger cross-sections. This makes components such as caches and register files have access times that are dominated by wire delays. To combat this, these components should have their access pipelined or their size reduced to match the delay of the clock.

The **design complexity** of systems is increasing at a very fast rate, which is making the manufacturer's verification process much more expensive and prone to mistakes. Due to this, when presented with two possible designs for a system, the simpler one must prevail (assuming equivalent performance).

The **CMOS endpoint** is fast approaching and will ultimately be limited by the nature of quantum physics. As transistors get smaller, their feature size decreases. Soon the behaviour of a transistor will be probabilistic instead of a deterministic. In the world of binary logic, this is clearly unusable.

II. Processor and Architectures

The Flynn's Taxonomy method of classifying computer architectures is based on the amount of concurrent instruction and data streams are available to the processing units.

However, Flynn's Taxonomy falls short in a couple of ways. First of all, it is difficult to distinguish different kinds of MIMD systems using this categorization. Also, the ability to compare different systems in this taxonomy is limited.

A. Single Instruction Stream, Single Data Stream (SISD)

SISD processors exploit no parallelism in either instruction or data streams. The control unit will fetch a single instruction stream, which is then translated into control signals for a single processing element to operate on a single data stream.

The colloquial example of an SISD is the uniprocessor that used to be used in consumer PCs.

B. Single Instruction Stream, Multiple Data Streams (SIMD)

An SIMD computer sends a single instruction stream to multiple processing units which operate on multiple stream of data. Notable examples of SIMD computers are array processors (vector processors) and GPUs.

C. Multiple Instruction Streams, Single Data Stream (MISD)

MISD computers are usually only used for fault tolerance. In this architecture, multiple instruction streams are sent to multiple processing units which operate using a single data stream.

D. Multiple Instruction Streams, Multiple Data Streams (MIMD)

MIMD computers have each processing unit working on different streams of data with a different instruction stream. Notable examples are multi-core superscalar processors.