CMPSCI 630 Systems **Spring 2023**

Lecture 3

Lecturer: Emery Berger *Scribe: Bryan McCaffery*

3.1 Lecture Notes

Ahmadal's Law
Some tasks must be done sequentially, so there will not be a 1 to 1 ratio between number of processors and performance. There will be a serial part of the program that can only be done on a single processor. As such no matter how many processors you use, there will be a limit to how much faster you can make it. Parallelizing a program is bound by the sequential part.

Parallel Programming Programs Split into two parts

- Part Parallelizable - Can be perfectly parallelized
- Serial Part - Sequential, can only be done on one core

Totally Parallel = T / P
where T is time, and P is number of cores
Totally Serial = T
$MaximumSpeedUp = \frac{T}{\frac{ParallelPart}{P}+S}$

Where T is total time, Parallel Part is time for Parallel Part and S is time for sequential part

Gustafsson's Law
As the number of processors grow, the amount of work we want to get done grows at least linearly. The idea of gustafsson's law is built upon, "If you build it they will come". In essence as processing power increases, user will develop larger problems to use the increase in processing power.

Chip Dependence analysis
The processor will attempt to parrallelize non dependent parts of the program. Using one of three processes ILP, SMT or SMP. SMP is no longer used in most machines, as it was designed upon utilizing multiple processors which no longer is common.

- ILP - Instruction Level Parallelism
- SMT - Simultaneous Multi Threading
- SMP - Simultaneous Multi Processors (Came before multi threading, when multiple processors used on same board)

Memory Wall
Processors are limited by retrieving data from memory which can take over a hundred cycles. As such, processors are attempt to retrieve memory before it is necessary or keep relevant data in the cache hierarchies. Two of the processes that are used are batching and speculation.

Batching is the process of retrieving N elements at a time instead of one. For example rather than retrieving one element, retrieve a whole array if you know that it is needed. If you know for certain data is needed retrieving it all at once is highly effective, if you have space in the cache.

Speculation attempts to retrieve data before it is necessary, by predicting/guessing what memory will be needed in the future. Predicting what will be needed is not an exact science, rather it use past patterns to predict what will be needed in the future. These patterns use very primitive machine learning, using past history as data. If speculation is accurate, it is highly effective, however over aggressive prefetching can lead to poor utilization rates. A poor utilization rate, can lead to wasted time and slowing down of the program.

Memory Visibility
Memory should be observable, however the cache is not observable. The cache is meant to be private, meaning that it is hard to observe the effectiveness of different memory techniques. The most effective way to monitor how effective the cache is, is to monitor how often memory is acquired. Security researchers did find a way to decipher things in the cache by monitoring the latency of memory, however this exposed private information and as such systems added processes to slow down things to hide information. This slow down added 15 to 20 percent speed reductions, but prevented information from being exposed.