GPU Basics

Introduction to Parallelization

S. Sundar and M. Panchatcharam

August 9, 2014

Outline

1. Introduction
2. Scientific Simulation
3. Introduction to Parallelization

School Problem

Let us look at the following simple school problem

Men and Time

12 men do a work in 36 days.In how many days can 18 men do the same work?

Key: The number of men is inversely proportional to the time taken to do the job

Solution: With basic assumptions that each man has the same efficiency, 18 men do the same work in 24 days.

School Problem

Let us look at the second example

Men and Time

40 tailors are working in a tailor shop. A tailor can stitch 100 shirts in 10 days. The shop got an order to stitch 100 school shirts for the next day delivery. How many persons has to be assigned to deliver the product on the next day?

Solution: 10 men can finish the task in a single day.

Draw lines

- Draw a line using one hand, 2s
- Draw two lines using one hand, 5s
- A multi tasking person, Draw two lines using two hands, 2s
- More than two lines ?
- A writing pad with 10 lines? 29s(one), 14s(two)
- If we have 10 hands!?

What is Parallelization?

Parallelization

The simultaneous use of more than one processors or system to solve a problem

Scientific Simulation

Scientific Simulation

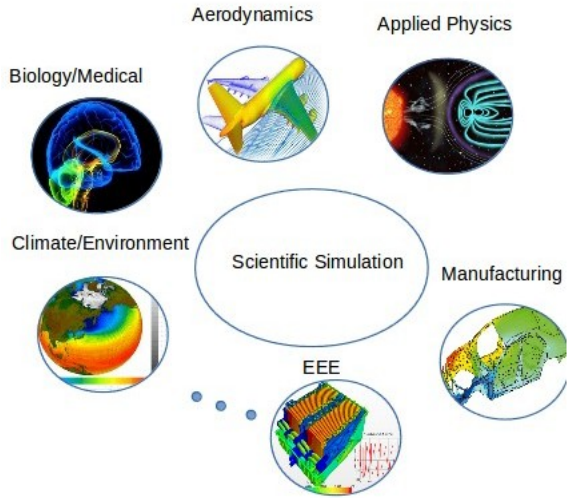
S. Sundar &
M. Panchatcharam

Introduction
Scientific Simulation
Introduction to Parallelization

Computing and Science

Computational modeling and simulation are among the most significant developments in the practice of scientific inquiry in the 20th Century. Within the last two decades, scientific computing has become an important contributor to all scientific disciplines. It is particularly important for the solution of research problems that are insoluble by traditional scientific theoretical and experimental approaches, hazardous to study in the laboratory, or time consuming or expensive to solve by traditional means.

Scientific Simulation Applications



Traditional Engineering

- Theory or Paper design
- Perform experiments in Lab,
- Build system

Limitations:

- Is it possible to build large wind tunnel by trial and error?
  – too difficult
- Is it possible to throw flight passengers as an accident experiment?
  – Too expensive
- Is it possible to wait for climate change or cyclone/tsunami to attack us to observe?
  – Too slow
- weapons, drug design, climate experimentation
  – Too dangerous

Third Pillar of Science : Simulation

Computational Science

- Use high performance computer systems to **simulate** the phenomenon
- Based on known physical laws and efficient numerical methods

Challenging Computations

- Science
  - Global warming
  - Biology: Cancer; protein folding; drug design
  - Astrophysics
  - Computational chemistry
  - Nano Sciences
- Engineering
  - Semiconductor
  - Earthquake model
  - CFD
  - Flight crash
- Business
  - Financial and economic modeling
  - Translation processing, web services
- Defense
  - Cryptography
  - Nuclear weapons

Introduction to Parallelization

Why do we need Parallelization?

- Serial computing is too slow
- To handle large amounts of data
- Consider the following problem:
  - Suppose our computer performs one billion(10^9) calculations per second. We want to predict the weather over India, for the next 10 days.
  - Area of India is 3.2 million sq. km.
  - model the atmosphere from sea level to 20km
  - make prediction of the weather at each cubical grid, with each cube measuring 0.1 km on each side.
  - To predict weather for each hour, each grid needs 100 calculations

Why do we need Parallelization?

Basics of HPC (High Performance Computing)

- Flop: Floating point operation
- Flops/s: floating point operation per second
- Bit: 0 or 1
- Bytes: Size of data (double precision floating point number is 8)
- Gflops $\approx 10^9$ flop/sec
- Tflops $\approx 10^{12}$ flop/sec
- Pflops $\approx 10^{15}$ flop/sec

Why do we need Parallelization?

Problem: Compute
$F(latitude, longitude, elevation, time) = Temperature/Pressure/\ldots$

- Discretize the domain
- Total number of points and calculations
  - $3.2 * 10^6 km^2 * 20 km * 10^3 cubes/km^3 = 6.4 * 10^{10}$ points
  - To predict weather for one hour, we need $6.4 * 10^{12}$ calculations
  - To predict weather for 10 days, we need $6.4 * 10^{12} * 240 = 1.5 * 10^{15}$ calculations
- If we use a normal PC(10^9), it will take $1.5 * 10^{15}/10^9 s = 1.5 * 10^6 s = 17 days$
- Predict hourly temperature of the globe???

Microprocessor Technology

Moore's Law

Over the history of computing hardware, the number of transistors in a dense integrated circuit doubles approximately every two years

Microprocessor(μP) Technology

- Chip density is increasing 2x every two years
- Clock speed is not increasing
- Number of cores have to double instead
- Parallelism must be managed by software

Three Barriers

- **Power**
  Improve power efficiency, to increase the performance of μP
- **Frequency**
  Conventional μP require increasingly deeper instruction pipelines to achieve higher operating frequencies.This technique has diminishing returns now, and even negative returns if power is taken into account
- **Memory**
  latency to DRAM memory is near 1000 cycles. So, program performance is dominated by data transfer between main memory and the processor.

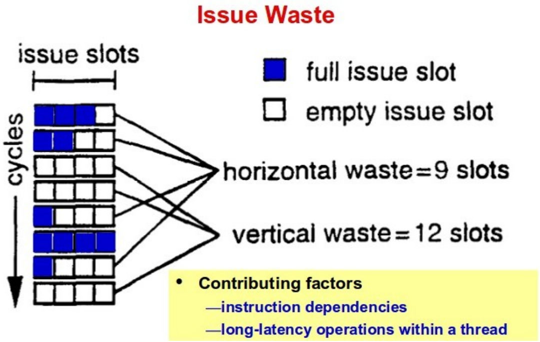
Power Barrier

- Power $\propto Voltage^2 \times Frequency$
- Frequency $\propto Voltage$
- Power $\propto Voltage^3$

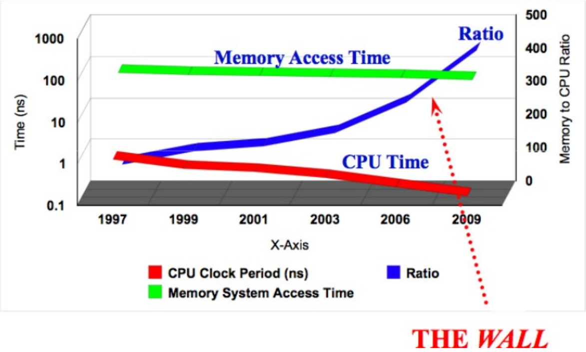
	Cores	Voltage	Freq	Performance	Power
SuperScalar	1	1	1	1	1
Increase	1x	1.5x	1.5x	1.5x	3.3x
Multicore	2x	0.75x	0.75x	1.5x	0.8x

50% more performance with 20% less power Advice: Use multiple slower devices, than one superfast device.

Frequency Barrier



Memory Barrier



Latency

Latency

Latency is the time taken to transfer a block of data from main memory.

- Transfer data as quickly as possible
- Latency is the time the CPU waits to obtain the data
- CPU clocks for caches
- nano-seconds for the main memory
- The latency of the main memory directly influences the efficiency of the CPU.
- Reducing wait time can be more important than increasing execution speed

Latency

Typical Latencies of today's world

Hierarchy	Processor clocks
Register	1
L1 Cache	2-3
L2 Cache	6-12
L3 Cache	14-40
Near Memory	100-300
Far Memory	300-900
Remote Memory	$O(10^3)$
Message-Passing	$O(10^3)$-$O(10^4)$

Speed of Light

- Can a serial computer execute the following code in one second?

```
for i=1:TRILLION
c[i]=a[i]*b[i]
```

  - To fetch *a* and *b*, it needs $2 * 10^{12}$ transaction/s
  - Assume data travels from memory to CPU at the speed of light
- Then in the chip, the data should fit in a square of side length $\approx 10^{-10}$ meters
- It is an atom size. Is it possible to design such chip?

Parallelism

Not possible so far. But Parallelism is the way to solve.

Types of Parallelism

- Bit Level Parallelism
  using floating point operations, etc
- Instruction Level Parallelism (ILP)
  multiple instruction execution per clock cycle
- Memory system parallelism
  overlap of memory operations with computation
- OS Parallelism
  multiple jobs run in parallel on commodity SMPs.

Let us see more about it in Lecture 2

Parallel Computing Principles

- Amdhal's Law
- Gustafson's Law
- Granularity
- Locality
- Load Balance
- Synchronization
- Performance modeling

These principles makes parallel programming even harder than sequential programming

Challenges in Parallelism

- It is so hard to develop algorithm
  –complexity of specifying and coordinating concurrent activities
- Software development is harder
  –lack of standardized and effective development tools
- Rapid place of change in computer system architecture
  –Today's parallel algorithm may not be suitable for tomorrow's parallel computer

First Glance

- Most People with IT degree and technology interested
- Most programmers develop applications based on university courses
- Parallel programming is scattered
- Parallel program beginners work with multicore CPUs, which is OS-based parallelism

First Glance

- Almost all desktop ship today with either dual or quad-core processor
- multicore processor works using threads

Thread

A thread is a separate execution flow within program that may diverge and converge as and when required with the main execution flow.

- In the background, we have context switching, which is an expensive operation

Context Switch

It is a swap in and out of registers. OS has to switch between tasks every time, in the multicore machine.

Serial/Parallel Issues

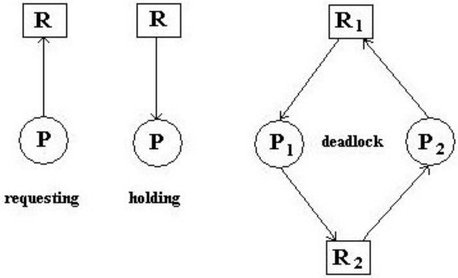
- The main issue is sharing the resources between thread
- To share resources, we have token.
- A thread with token can use the resource, where as other has to wait to release the token
- If there is a single token, there wont be a problem
- If there are two tokens, deadlock occur

Deadlock

Deadlock

If there are two tokens, thread 1 grabs token 1, thread 2 grabs token 2. Thread 1 now tries to grab token 2, while thread 2 tries to grab token 1. As neither thread releases the one token they already own, all threads wait forever. This situation is called deadlock.

Programmers responsibility to avoid deadlock

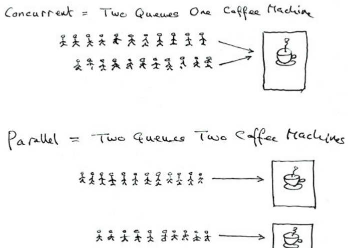


Concurrency

concurrency

It is a property of system in which several computations are executing simultaneously, and potentially interacting with each other.

For example, if a problem requires every data point to know about the value of it surrounding neighbours then the speedup will be limited. To avoid this bottleneck, throw more processors. But, computation slows down, because threads spend more time on sharing data



Locality

- In modern computers, we have multilevel caches (L1, L2, L3).
- Caches work on either spatial (close in the address space) or temporal (close in time).
- Temporal locality: Data accessed before, accessed again
- Spatial locality: Data close to last accessed data will be accessed in future

Understanding Cache

- Caches work well if the task is repeated many times
- A plumber (RAM)) with a toolbox (L1 cache) which can hold 4 tools.
- Useful if he uses the same four tools repeatedly (cache hit)
- If an important job requires additional tool, he takes it from the van (L2 cache)
- If he needs a special tool, he needs to leave the job and drive to local store (global memory)
- The last job may require more time
- Modern plumber comes with a tool box (L1 cache), van (L2 cache) and truck (L3 cache), where the truck carries more additional tools.

Locality...

- The programmer must deal with locality at first instance
- He/she has to think which memory locations (L1, L2, or L3) or data structures will be needed
- These details need to be collected in a single trip to the global memory and placed on chip memory

Amdhal's Law

Amdhal's Law

Given:

- $n \in \mathfrak{N}$, the number of threads of execution
- $f_s \in [0, 1]$, the fraction of the algorithm that is serial
- $f_p \in [0, 1]$, the fraction of the algorithm that is parallel
- $f_p = 1 - f_s$

then The time taken to finish when being executed on n threads of execution corresponds to

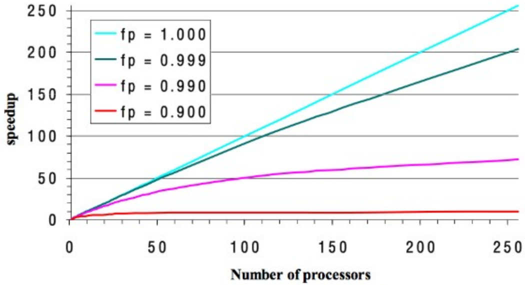
$$T(n) = T(1)(f_s + \frac{1}{n} f_p) \qquad (1)$$

The speedup that can be expected

$$S_n = \frac{1}{f_s + \frac{1}{n} f_p} \qquad (2)$$

Amdhal's Law

Even a small fraction of serial code content is enough to degrade the parallel performance. For example, even 0.001 is enough to decrease the speed up by 50% (see Figure)



Gustafson's Law

Gustafson's Law

Effect of multiple processor on run time of a problem with a fixed amount of parallel work per processor

$$S_P \leq P - \alpha.(P - 1) \tag{3}$$

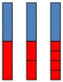
α is the fraction of non-parallelized code where the parallel work per processor is fixed (not the same as f_p from Amdhal's law
P is the number of processors

Gustafson's Law

Introduction
Scientific Simulation
41 Introduction to Parallelization

Amdahl : fixed work

$f_p = 0.5$



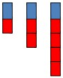
$$S \le \frac{1}{f_s + f_p / N}$$

$$S_2 \le \frac{1}{0.5 + 0.5/2} = 1.33$$

$$S_4 \le \frac{1}{0.5 + 0.5/4} = 1.6$$

Gustafson : fixed work per processor

$\alpha = 0.5$



$$S_p \le P - \alpha \cdot (P-1)$$

$$S_2 \le 2 - 0.5(2-1) = 1.5$$

$$S_4 \le 4 + 0.5(4-1) = 2.5$$

- Amdhal's law: Strong Scaling
- Gusafson's law: Weak Scaling

Overhead

- Cost of starting process
- Cost of communicating shared data
- Cost of synchronizing
- Extra computation

Tradeoff: Algorithm needs sufficiently large units of work to run fast in parallel but no so large that there is not enough parallel work

Load Imbalance

Load imbalance is the time that some processors are idle due to

- insufficient parallelism
- unequal task size

Example: Tree structured computations
Algorithm needs to balance load

THANK YOU