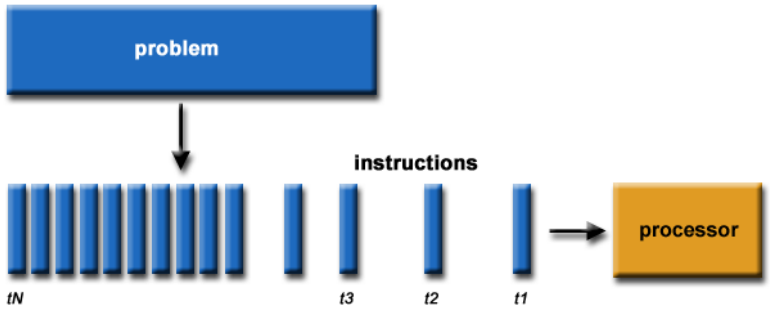
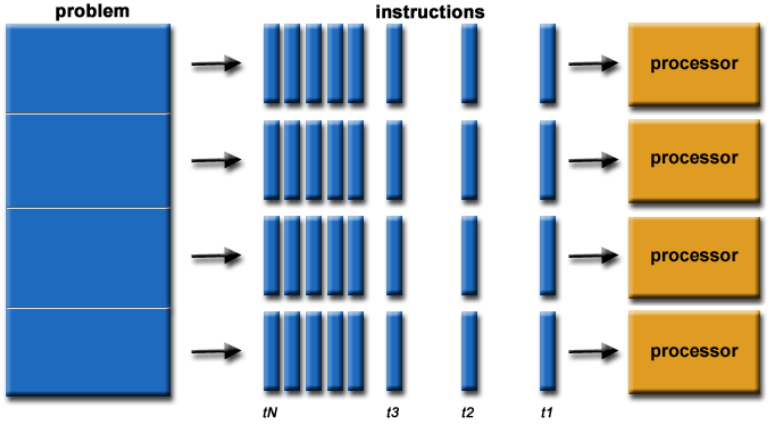
Lecture 01: Introduction to MPPRS

Vivek Kumar
Computer Science and Engineering
IIIT Delhi
vivekk@iiitd.ac.in

What is Parallel Programming?



- Serial
  - One instruction at a time



- Parallel
  - Multiple instructions in parallel

Picture source: https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial

Andy and Bill's Law

"What Andy giveth, Bill taketh away"

"The Free Lunch is Now Over!"

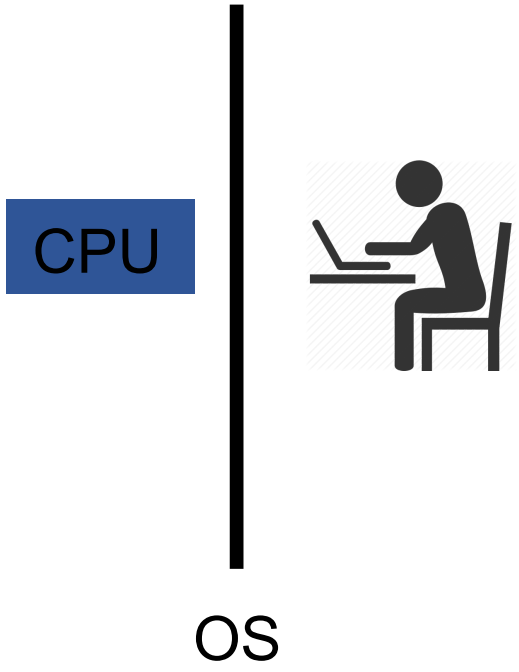
Herb Sutter, Dr. Dobb's Journal, March 2005

Motivations for Parallel Programming

- Technology push
- Application push

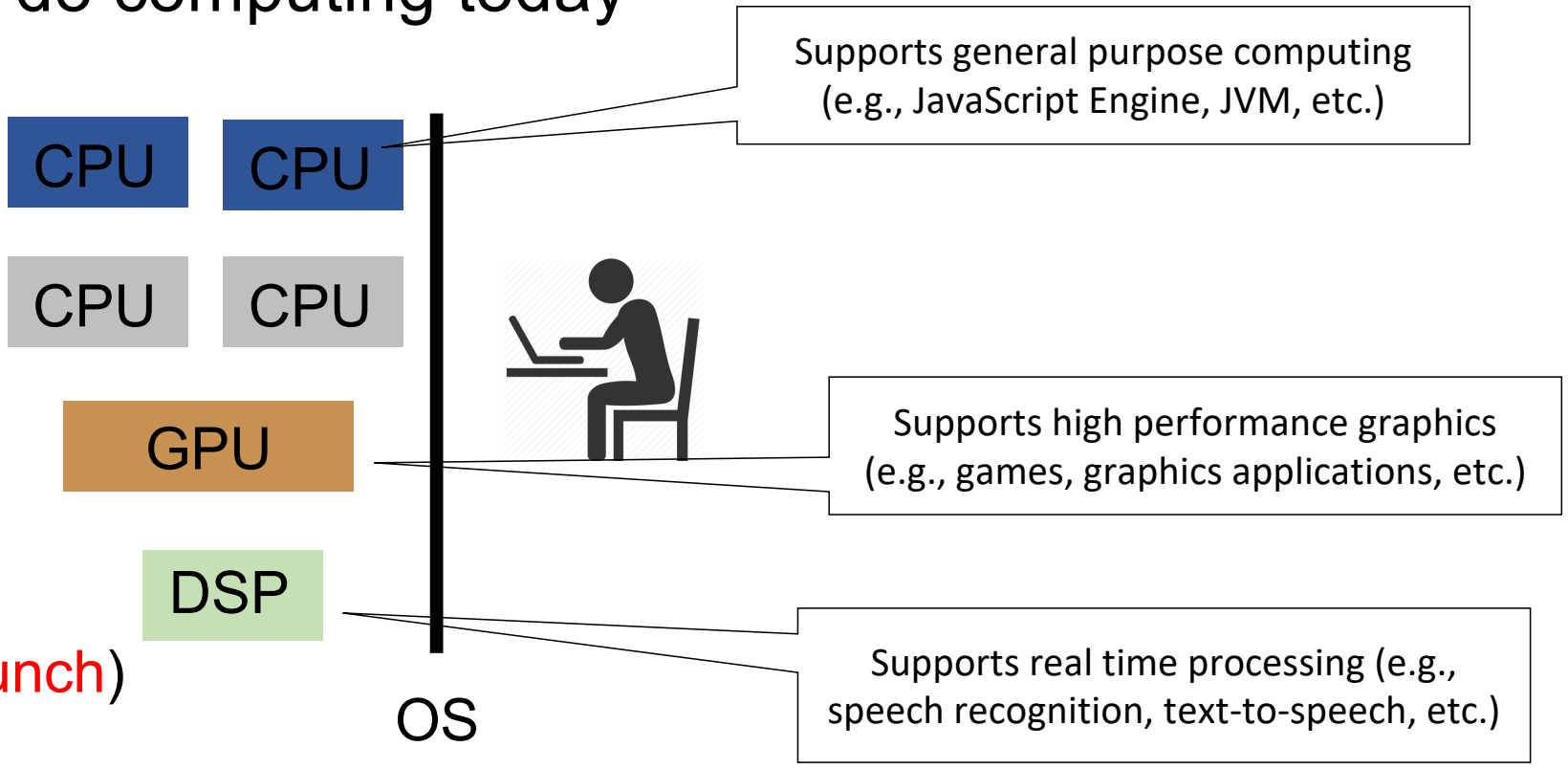
Technology Push

- How I did computing during my undergraduate studies (early 2000s)



Technology Push

- How you do computing today

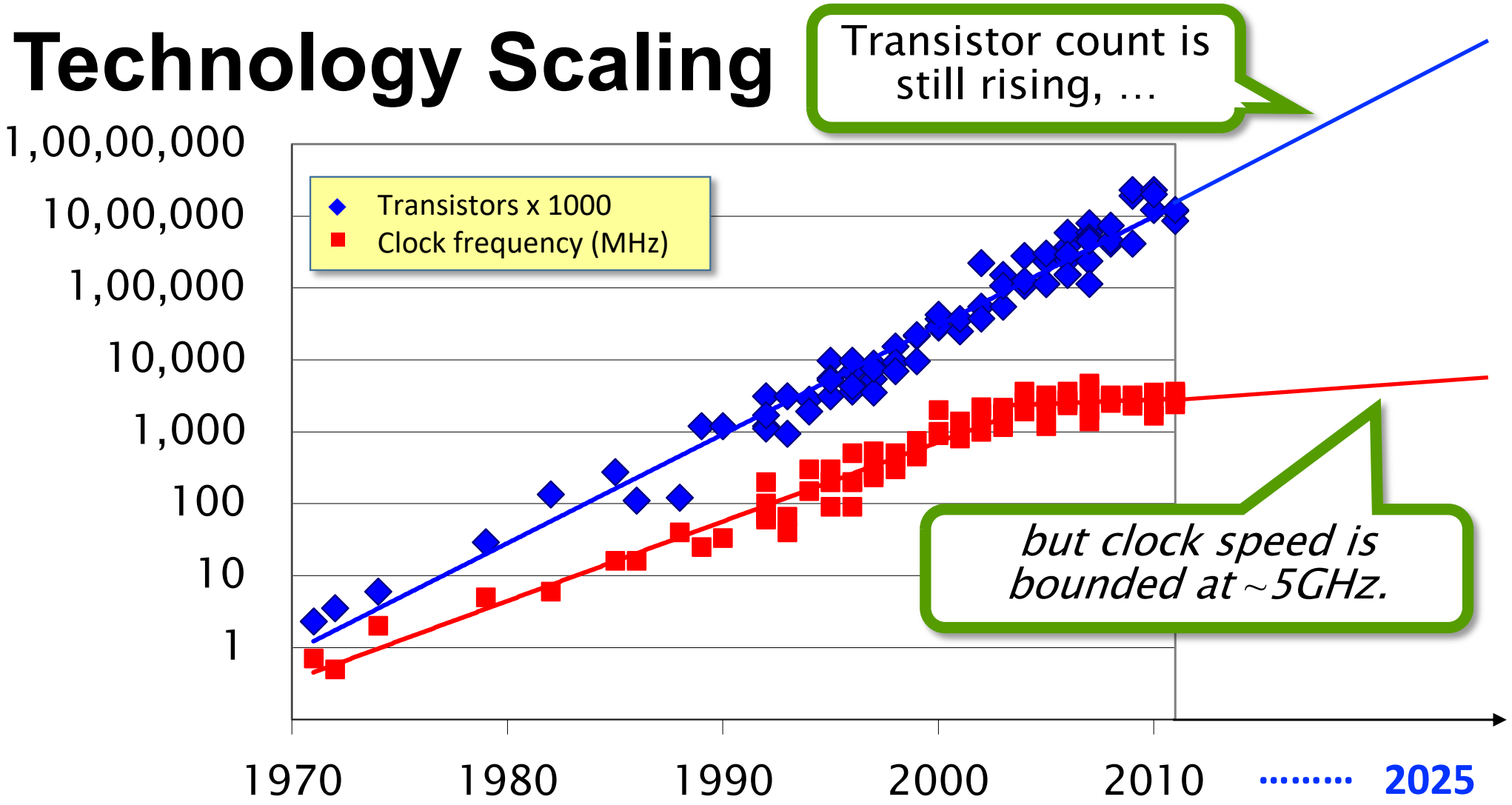


Today
(no more free lunch)

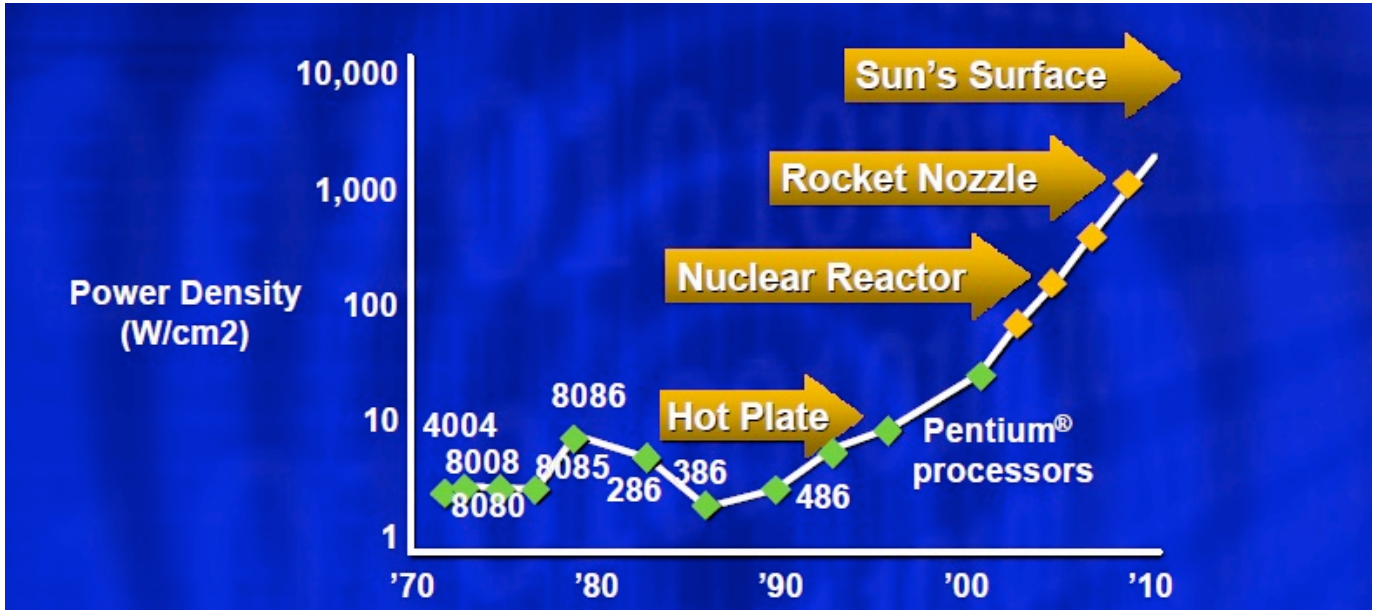
Let's Pause and Recap

- **Why the hell do we even need multicores ??**
- Moore's law (1965)
  - Area of transistors halves roughly every two years
    - Total transistors on processor chip gets doubled roughly every two years
    - Smaller transistors can switch at higher speed, hence increased single core performance
- Dennard scaling (1974)
  - Power required to flip a transistor scales with its area
    - Implies that power for a fixed chip area remains almost constant as transistors grow smaller

Technology Scaling



Power Density

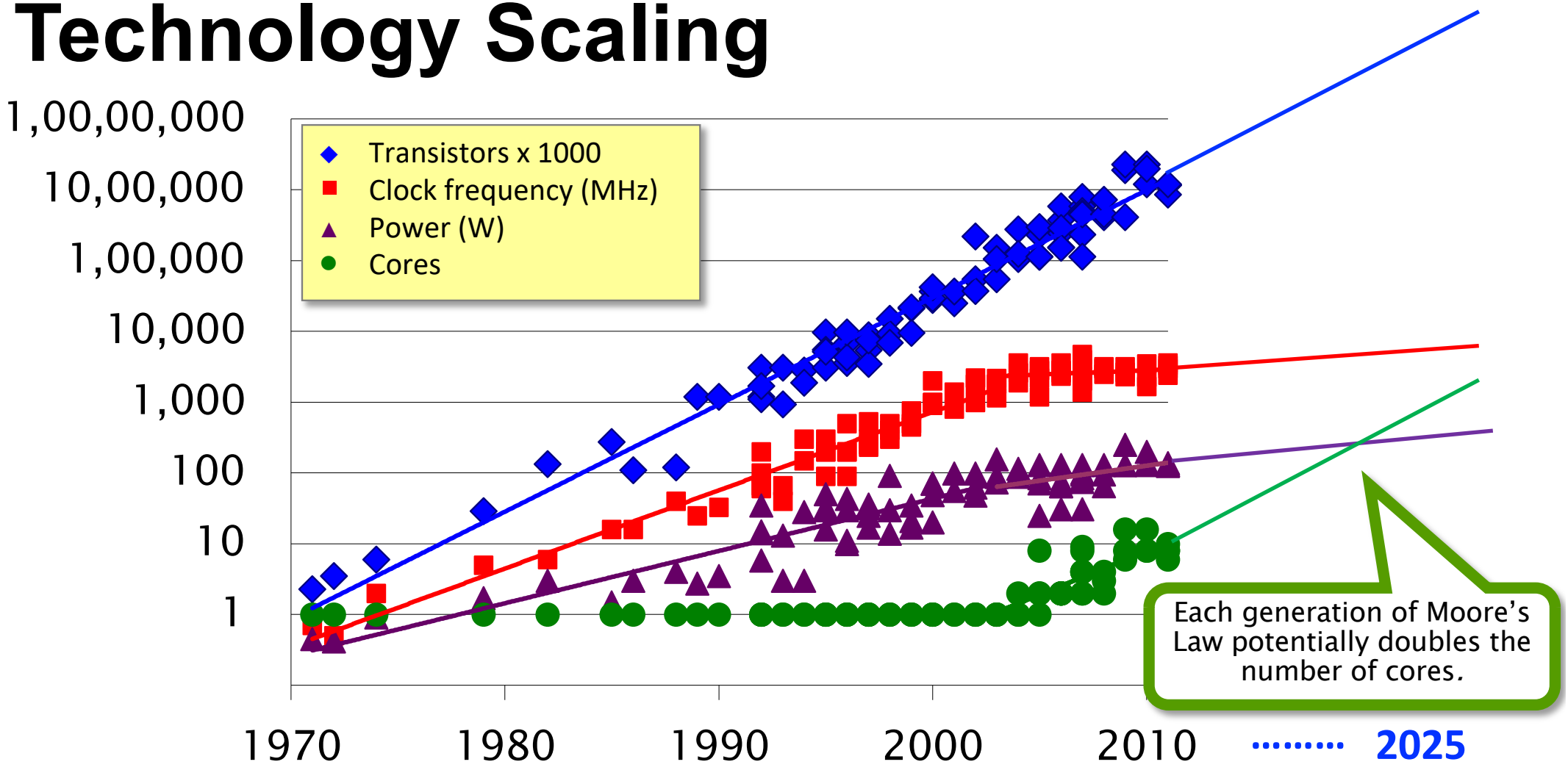


- No more free lunch!
  - Dynamic power is proportional to cube of frequency
    - Power = C x V^2 x f
      - As, V ∝ f
      - Power ∝ f^3
  - Thermal wall hit around 2004
    - Heat dissipation
    - Leakage current

Power density, had scaling of clock frequency continued its trend of 25%-30% increase per year.

Source: Patrick Gelsinger, Intel Developer's Forum, Intel Corporation, 2004.

Technology Scaling



Multicore Saves Power

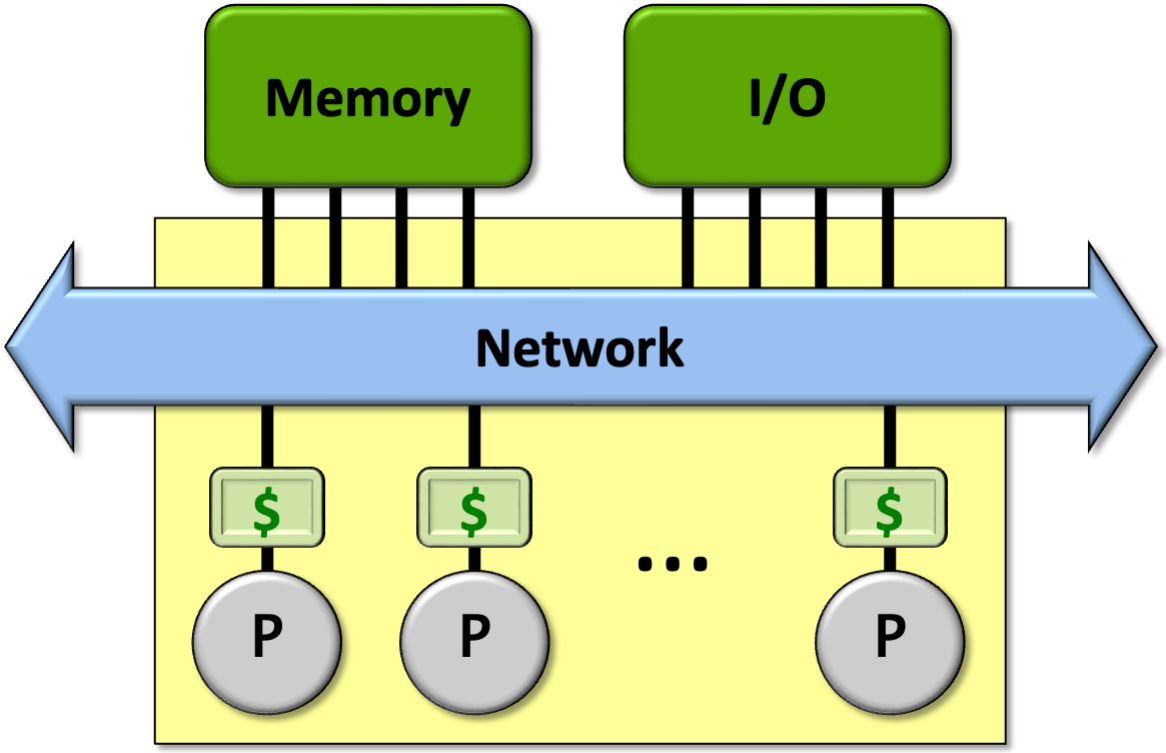
- Nowadays (post Dennard Scaling)
  - Power ~ (Capacitance) * (Voltage)^2 * (Frequency) and maximum Frequency is capped by Voltage
  - *Power is proportional to (Frequency)^3*
- Baseline example: single 1GHz core with power P
  - Option A: Increase clock frequency to 2GHz
    - Power = 8P
  - Option B: Use 2 cores at 1 GHz each
    - Power = 2P
- Option B delivers same performance as Option A with 4x less power … provided software can be decomposed to run in parallel !!

Source: https://wiki.rice.edu/confluence/download/attachments/4435861/comp322-s16-lec1-slides.pdf?version=1&modificationDate=1452732285045&api=v2

A Real World Example

	Intel Core-i7-970 (released 2010)	Intel Core i9-13900K (released 2022)
Number of Cores	6	24
Base Clock Frequency	3.2 GHz	3 GHz (P-core) / 2.2 GHz (E-core)
Power	130 Watts	125 Watts

Abstract Multicore Architecture



Latest Intel Xeon Processors

Product Name	Launch Date	Total Cores	Max Turbo Frequency	Processor Base Frequency	Cache	TDP
Intel® Xeon® 6776P-B Processor (288M Cache, 2.30 GHz)	Q4'25	72	3.5 GHz	2.3 GHz	288 MB	325 W
Intel® Xeon® 6766P-B Processor (256M Cache, 2.30 GHz)	Q4'25	64	3.5 GHz	2.3 GHz	256 MB	305 W
Intel® Xeon® 6756P-B Processor (256M Cache, 2.20 GHz)	Q4'25	64	3.5 GHz	2.2 GHz	256 MB	325 W
Intel® Xeon® 6768P-B Processor (256M Cache, 2.20 GHz)	Q4'25	64	3.5 GHz	2.2 GHz	256 MB	325 W
Intel® Xeon® 6718P-B Processor (160M Cache, 2.50 GHz)	Q4'25	40	3.5 GHz	2.5 GHz	160 MB	235 W
Intel® Xeon® 6548P-B Processor (128M Cache, 2.00 GHz)	Q4'25	32	3.5 GHz	2 GHz	128 MB	195 W

Source: https://www.intel.com/content/www/us/en/products/details/processors/xeon.html

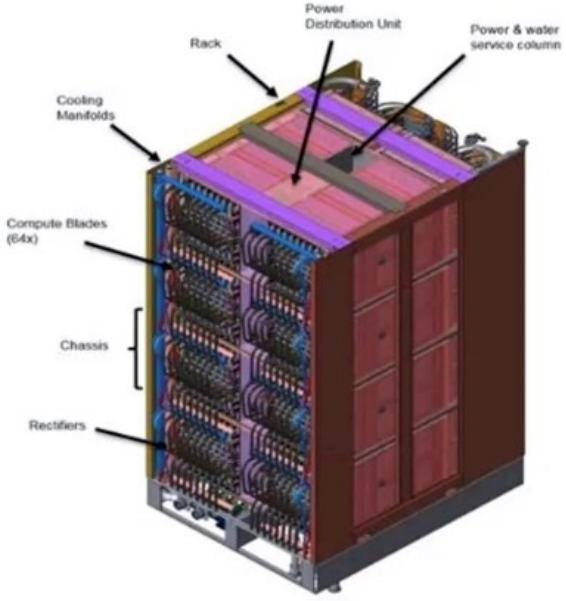
Modern Supercomputer (June 2024)

System

- 2 EF Peak DP FLOPS
- 74 compute racks
- 29 MW Power Consumption
- 9,408 nodes
- 9.2 PB memory
  (4.6 PB HBM, 4.6 PB DDR4)
- Cray Slingshot network with dragonfly topology
- 37 PB Node Local Storage
- 716 PB Center-wide storage
- 4000 ft^2 foot print

Olympus rack

- 128 AMD nodes
- 8,000 lbs
- Supports 400 KW



AMD node

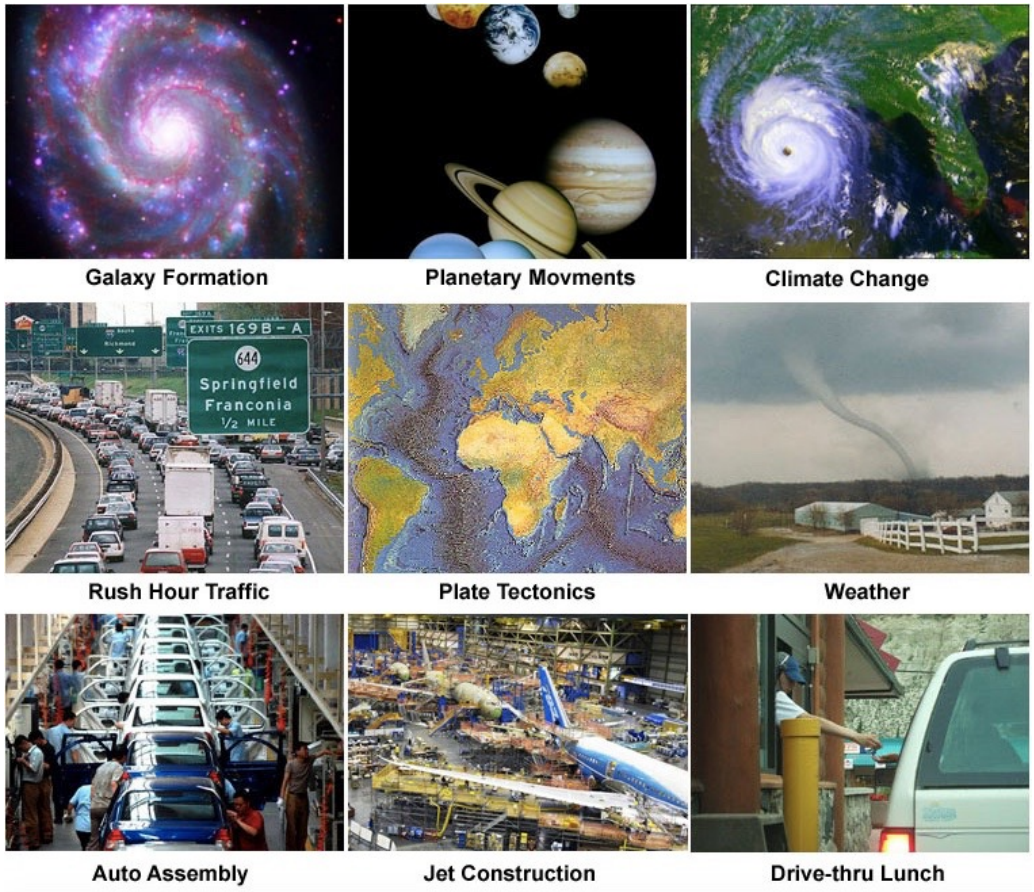
- 1 AMD "Trento" CPU
- 4 AMD MI250X GPUs
- 512 GiB DDR4 memory on CPU
- 512 GiB HBM2e total per node
  (128 GiB HBM per GPU)
- Coherent memory across the node
- 4 TB NVM
- GPUs & CPU fully connected with AMD Infinity Fabric
- 4 Cassini NICs, 100 GB/s network BW

Compute blade

- 2 AMD nodes

Source: https://www.nextplatform.com/wp-content/uploads/2022/03/oak-ridge-al-geist-frontier-specs.jpg

Parallel Programming: Application Push



ChatGPT

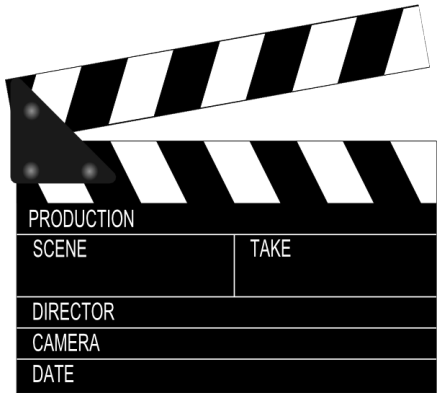
Gemini

PyTorch

TensorFlow

Performance of a Computer for Scientific Calculations

Movie Hits

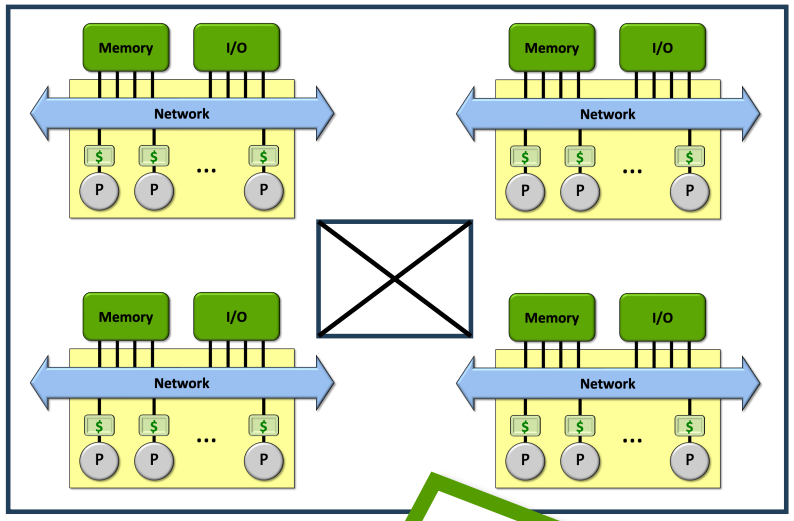


Computer FLOPS

Source: All images obtained from https://images.google.com/

Floating Point Operations per Second (FLOPS)

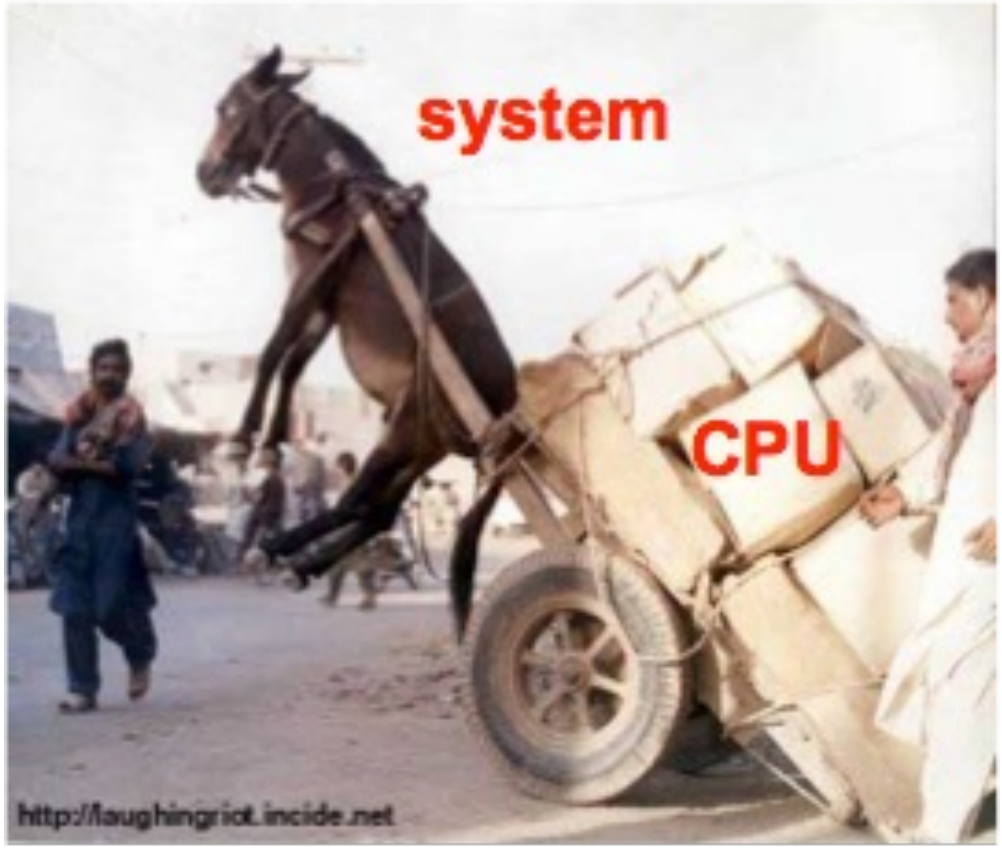
- Measure of a computer's theoretical peak performance
- FLOPS = (Total Cores) x (Clock) x (FLOPS per cycle)



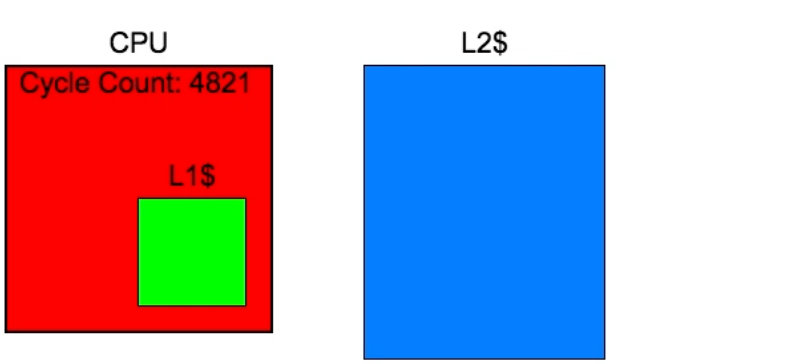
One of our lab servers with four Intel Xeon Gold 5318H processors

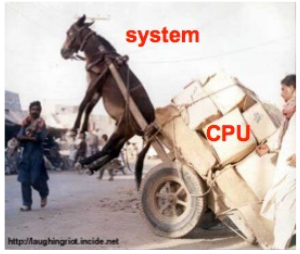
- Single core base frequency = 2.1GHz
- FLOPS (double precision) per cycle = 32
- Total cores / processor = 18
- Total cores = 4 x 18 = 72
- FLOPS = ?

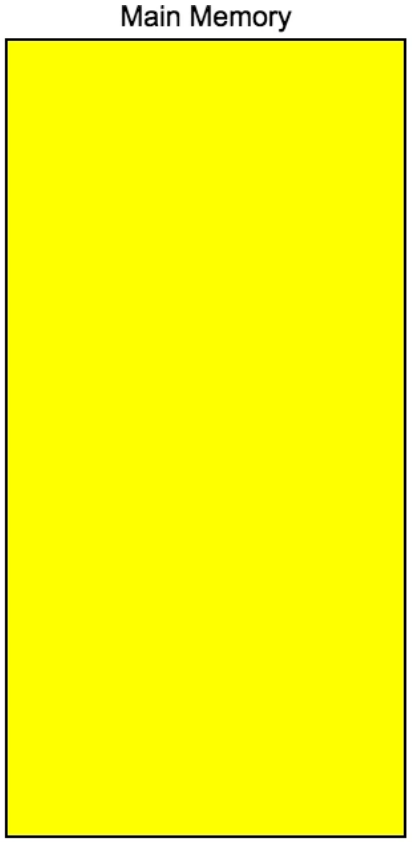
FLOPS is Just Theoretical Peak..



Latency in Memory Hierarchy







- Computation is just part of the picture
- Another analogy
  - Normalizing with L1 latency, and assuming one seconds is equal to 4 cycles
    - L1 = one second
    - L2 = 7.5 seconds
    - Main memory = 2.5 minutes
    - Hard drive = in several days!

Animation source: https://overbyte.com.au/misc/Lesson3/CacheFun.html

Lets do an Analysis

- Single core processor, clock cycle = 1GHz
- FLOPS per cycle = 4
- No caches
- Peak performance = 4 GFLOPS ??

DRAM access latency = 100ns

(=100 cycles, i.e. 10 MHz)

Peak performance limited to 4 MFLOPS

Example taken from the book: Introduction to Programming by Grama et. al. 2nd edision

Parallel Programming Dilemma

```
uint64_t array_sum(uint64_t* array, uint64_t size) {
  uint64_t sum = 0;
  for(uint64_t i=0; i<size; i++) {
    sum = sum + array[i];
  }
  return sum;
}
```

```
uint64_t fib(uint64_t n) {
  if (n < 2) {
    return n;
  } else {
    uint64_t x = fib(n-1);
    uint64_t y = fib(n-2);
    return (x + y);
  }
}
```

- How to convert a sequential program into parallel program with minimal effort
- How to customize the parallel program for the underlying processor such that it achieves the best performance

Parallel Programming Dilemma

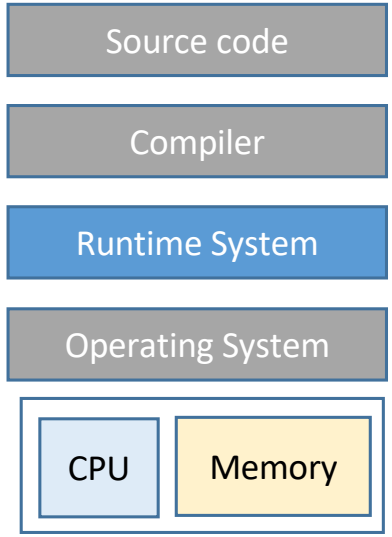
```
uint64_t array_sum(uint64_t* array, uint64_t size) {
  uint64_t sum = 0;
  for(uint64_t i=0; i<size; i++) {
    sum = sum + array[i];
  }
  return sum;
}
```

```
uint64_t fib(uint64_t n) {
  if (n < 2) {
    return n;
  } else {
    uint64_t x = fib(n-1);
    uint64_t y = fib(n-2);
    return (x + y);
  }
}
```

- Solution
  - Write parallel program once but run it anywhere
  - How?
    - With the help of a **Runtime System**

What are Runtime Systems?

- Runtime systems helps in the execution of a program by helping it interact with with the underlying computing resources such as CPU and memory
  - Abstracts away the challenges with the modern hardware



Multicore Parallel Programming and Runtime Systems (MPPRS) – Topics

- Part-1: High **productivity** parallel programming
  - Tasks based programming
    - Async-finish, futures and promises, etc.
  - Traditional programming model
    - Pthreads, OpenMP
  - Heterogeneous parallel programming
    - Boost C++ library for programming over SIMD registers, GPUs
- Part-2: High **performance** parallel runtime system
  - Dynamic load balancing runtime
  - Minimizing runtime overheads
  - Achieving locality
  - Achieving energy efficiency

Course Evaluation

- Quiz: 10%
  - Total **5** quizzes throughout the semester
  - Will be held during lecture hours (around 20mins duration)
  - **(N-1)** policy (e.g., missing due to medical issue, laziness, etc.)
- Group project: 20%
  - Total **3** deadlines (one before/after midsem)
  - Group of one or two students only
  - **Use of LLM is allowed!**
- Labs: 20%
  - Total **6-7** labs throughout the semester
  - **Closed book** and to be done in Btech lab machines
  - **(N-2)** policy (e.g., missing due to medical issue, laziness, etc.)
- Semester exams: 20% + 30%
- Bonus marks based on attendance (e.g., preparing notes): 4.4%

Important Information

1. I will not upload mid/end semester solution/rubric on Google Classroom
   - Although, I will discuss it in class
2. I will **not** be marking attendance but if you miss the lecture then you are on your own!
   - We will not provide the lecture recordings

Course Prerequisites

- **Programming in C/C++ is a must!**
  - If you don't know C/C++ then you should be confident that you can pick it up on your own
- Basics of Operating Systems

We will strictly follow the IIITD **plagiarism policy**. No excuses if you get caught in plagiarism

Reference Materials

- Course material derived from multiple sources
- Course notes / references will be provided depending on the lecture
- References will also be mentioned on the last slide in each lecture
- Relevant text books will be mentioned during the lectures

Next Class

- Introduction to parallel programming

HiPeC Lab: https://hipec.github.io/