Domain Specific Architectures
CSCE 4013/5013

David Andrews
Rm 527 JBHT
dandrews@uark.edu
CSCE University of Arkansas

Agenda

- Technology Trends
- Types of Parallelism
- Measuring Performance

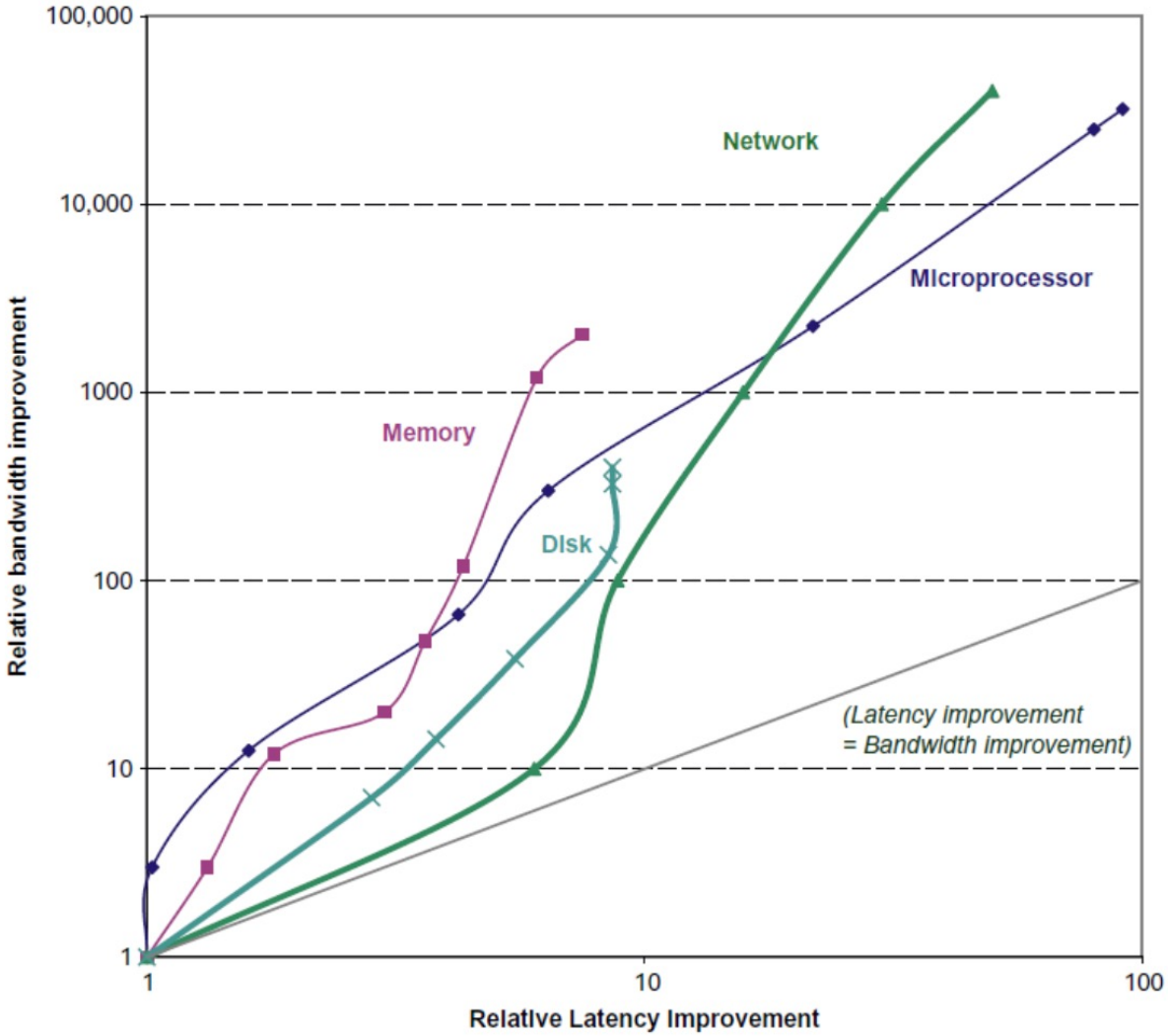
Trends in Technology

- DRAM capacity: 25-40%/year (slowing)
  - 8 Gb (2014), 16 Gb (2019), possibly no 32 Gb
- Flash capacity: 50-60%/year
  - 8-10X cheaper/bit than DRAM
- Magnetic disk capacity: recently slowed to 5%/year
  - Density increases may no longer be possible, maybe increase from 7 to 9 platters
  - 8-10X cheaper/bit then Flash
  - 200-300X cheaper/bit than DRAM
- Emerging NVRAM technologies:
  - PCM, STTRAM, Memristors

What are we Optimizing ?

- Prior Design Goal: Keep processor(s) busy:
  - Focus: Bandwidth or throughput
    - Total work done in a given time
    - 32,000-40,000X improvement for processors
    - 300-1200X improvement for memory & disks
- New Design Goal: Machine Learning Latency
    - Latency or response time
    - Time between start and completion of an event
    - 50-90X improvement for processors
    - 6-8X improvement for memory and disks

Bandwidth and Latency



Log-log plot of bandwidth and latency milestones

How Do We Compare Performance?

- (Amdahl's Law) Compute Speedup (S_p) of X relative to Y
  Execution time$_Y$ / Execution time$_X$
  - Unitless Ratio
- What is Execution Time ?
  - Depends on What You Need to Evaluate.
  - Choose to Remove Unknowns-focus on Only 1 Variable
  - Examples:
    - Wall clock time: includes all system overheads
    - CPU time: only computation time
- Use Meaningful Benchmarks
  - Kernels (e.g. matrix multiply)
  - Toy programs (e.g. sorting)
  - Synthetic benchmarks (e.g. Dhrystone)
  - Benchmark suites (e.g. SPEC06fp, TPC-C)

Fan Favorite: Amdahl's Law

$$\text{Speedup } S_p = \frac{T_{old}}{T_{new}} = \frac{Execution\ Time_{old}}{Execution\ Time_{new}}$$

$$Execution\ time_{new} = Execution\ time_{old} \times \left((1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right)$$

$$Speedup_{overall} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

How Do We Increase Speedup ?

- Historically through Moore's Law
  - Increased transistor densities supported innovation
  - Feature shrinkage enabled clock increases referred to as Dennard Scaling.
    - Ended…….
  - Moore's Law has slowed but is still kicking
- Exploit Parallelism
  - Exploit what algorithm has to offer
- More Efficient Designs
  - Yes, Domain Specific Architectures !

Performance Is Thru Parallelism !

- Cannot Clock Faster so Do More In Parallel
  - Apply Transistors to Exploit Parallelism
  - Parallelism Exists at Different "Granularities"
  - Circuit, Data, Instruction, Procedural, Program….
  - *Continued increases rely on Moore's Law: Why ?*
- Implicit Parallelism within Single Processor
  - Out of Order Instruction-Level parallelism (ILP)
  - Speculation
  - *Pretty much squeezed this lemon dry*
- Domain Specific Parallelism
  - Thread-level parallelism (TLP)
  - Data-level parallelism (DLP)
    - *Our Focus for ML*

Flynn's Taxonomy

Single instruction stream, single data stream (SISD)

=> Single instruction stream, multiple data streams (SIMD)

- Vector architectures
- Multimedia extensions
- Graphics processor units

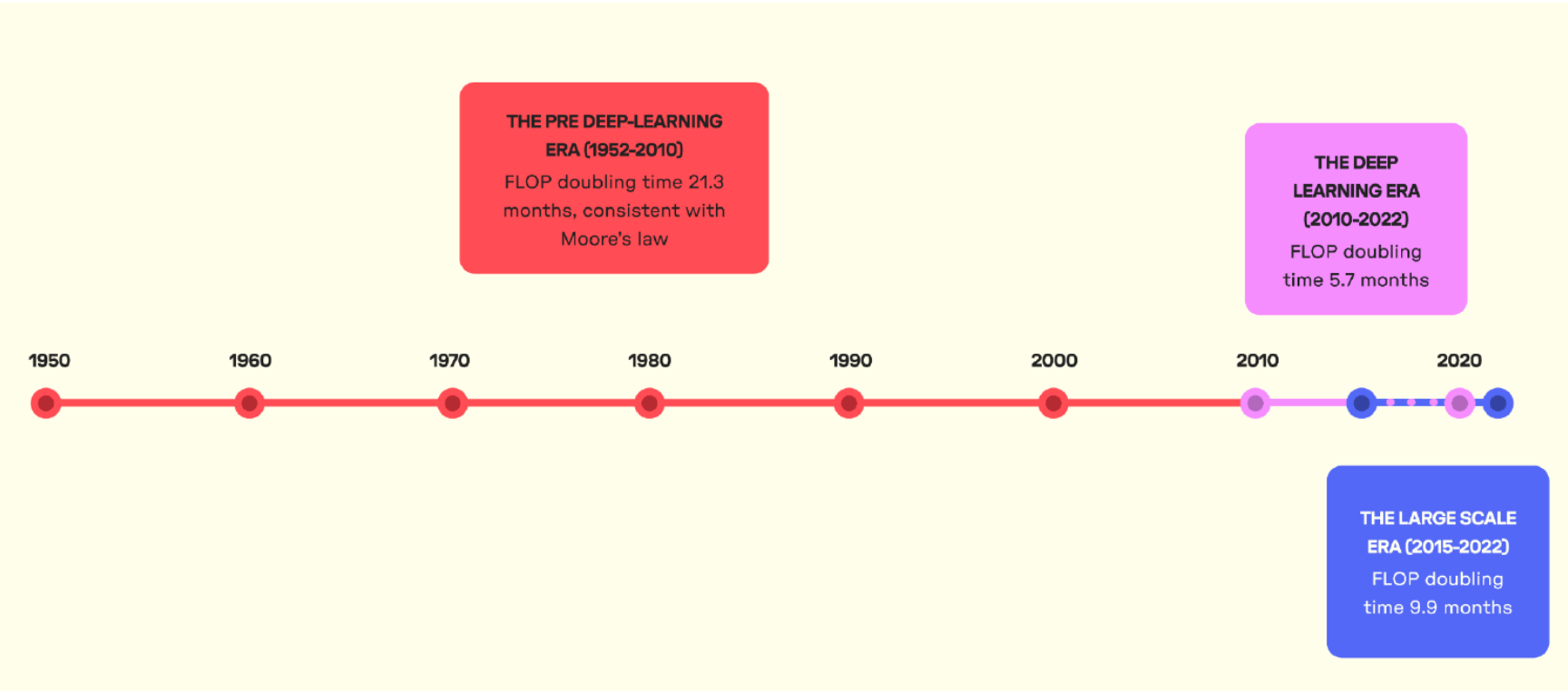
Multiple instruction streams, single data stream (MISD)

- No commercial implementation

Multiple instruction streams, multiple data streams (MIMD)

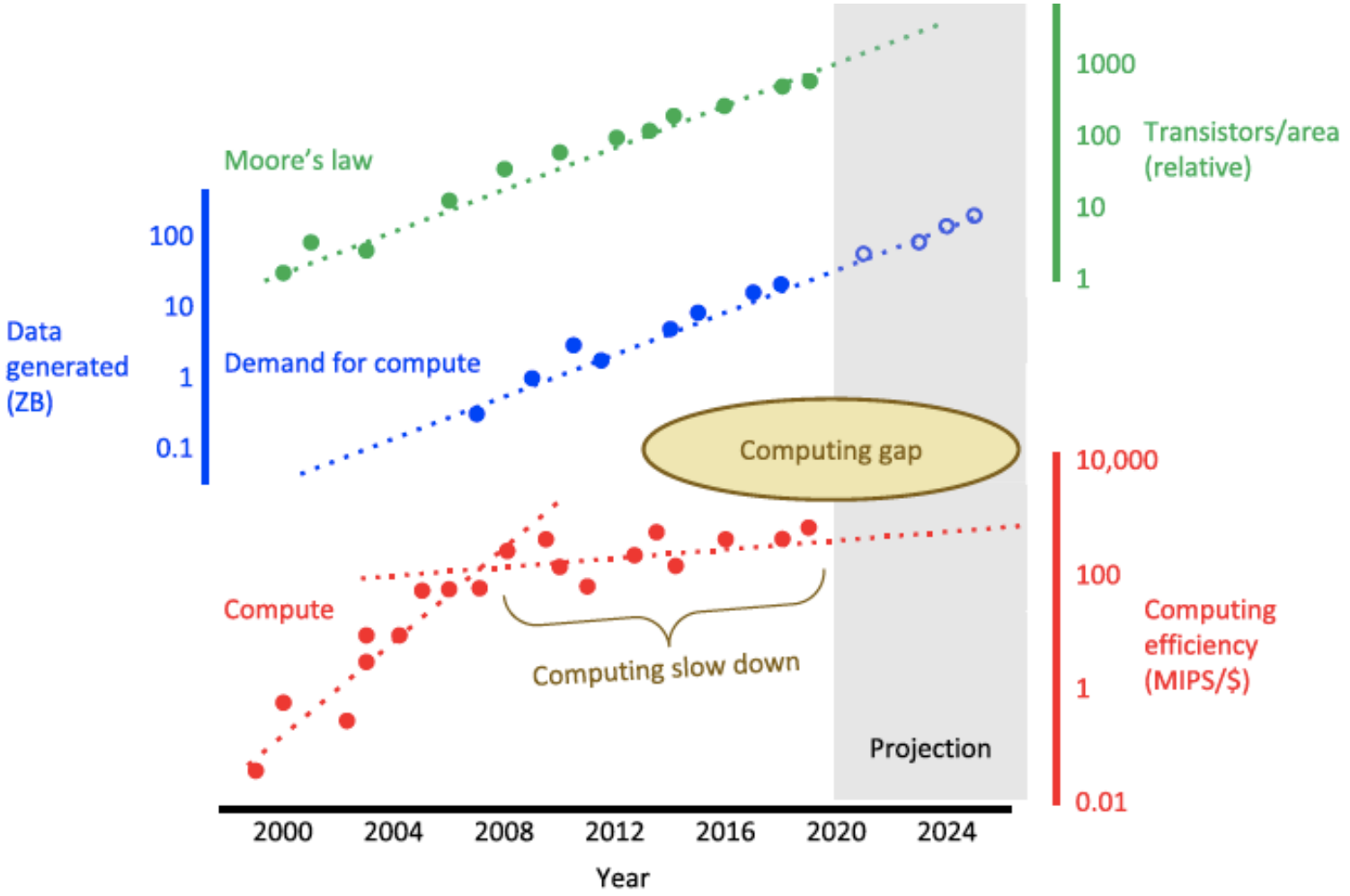
- Tightly-coupled MIMD
- Loosely-coupled MIMD

With ML we are Chasing a Speeding Train…



Computational Power and AI, Jai Vipra Sarah West, AI Now

We Created a Monster That Needs Continual Feeding !!!



J. D. Kendall and S. Kumar, "The building blocks of a brain-inspired computer," Applied Physics Reviews, vol. 7, no. 1, 2020.

Next time Review of

- Moore's Law,
- Dennard Scaling,
- Power and Energy
- See you then!

Review CSCE 2214: Processor Performance Equation

$$CPU\ time = CPU\ clock\ cycles\ for\ a\ program \times Clock\ cycle\ time$$

$$CPU\ time = \frac{CPU\ clock\ cycles\ for\ a\ program}{Clock\ rate}$$

$$CPI = \frac{CPU\ clock\ cycles\ for\ a\ program}{Instruction\ count}$$

$$CPU\ time = Instruction\ count \times Cycles\ per\ instruction \times Clock\ cycle\ time$$

$$\frac{Instructions}{Program} \times \frac{Clock\ cycles}{Instruction} \times \frac{Seconds}{Clock\ cycle} = \frac{Seconds}{Program} = CPU\ time$$

Review CSCE 2214: Processor Performance Equation

- Different instruction types having different CPIs

$$CPU\ clock\ cycles = \sum_{i=1}^{n} IC_i \times CPI_i$$

$$CPU\ time = \left(\sum_{i=1}^{n} IC_i \times CPI_i \right) \times Clock\ cycle\ time$$

Review CSCE 2214: Processor Performance Equation

- Different instruction types having different CPIs

$$CPU\ clock\ cycles = \sum_{i=1}^{n} IC_i \times CPI_i$$

$$CPU\ time = \left(\sum_{i=1}^{n} IC_i \times CPI_i \right) \times Clock\ cycle\ time$$