Lect. 15: Real-World Concurrent Programming

Xin Liu

Florida State University
xliu15@fsu.edu

CIS 5370 Computer Security
https://xinliulab.github.io/cis5370.html

Concurrent Programming in the Age of AI

NVIDIA DGX-1 (2016)

8 × Tesla V100: The Computational Core of DGX-1

- **DGX-1 is a complete AI supercomputer** designed by NVIDIA.
- It integrates **8 Tesla V100 GPUs** into a single system.
- These GPUs are interconnected using **NVSwitch**, providing high-speed GPU-to-GPU communication (300GB/s).
- Compared to standalone GPUs, DGX-1 includes:
  - **2 × Intel Xeon CPUs** for coordination.
  - **512GB DDR4 RAM** for system memory.
  - **15TB NVMe SSD** for high-speed storage.
  - Optimized power and cooling system (3.2kW power consumption).
- **Performance:** 170 TFLOPS @ 3.2kW
- **Comparison:** CRAY-1: 138 MFLOPS @ 115kW

NVIDIA DGX B200 (2024)

8 × Blackwell GPUs: The Computational Core of DGX B200

- **DGX B200 is the latest AI supercomputer** designed by NVIDIA.
- It integrates **8 Blackwell GPUs** into a single system.
- These GPUs are interconnected using **NVLink and NVSwitch**, providing ultra-high-speed communication.
- Compared to standalone GPUs, DGX B200 includes:
  - **Optimized AI acceleration** for large-scale training and inference.
  - **High-bandwidth memory (HBM)** for faster data access.
  - Advanced **power and cooling solutions** for efficient operation.
- **Performance:**
  - **72 PFLOPS (Training), 144 PFLOPS (Inference)** @ 14.3kW

Computation Behind Large Language Models

"Attention Is All You Need"

LLM Visualization

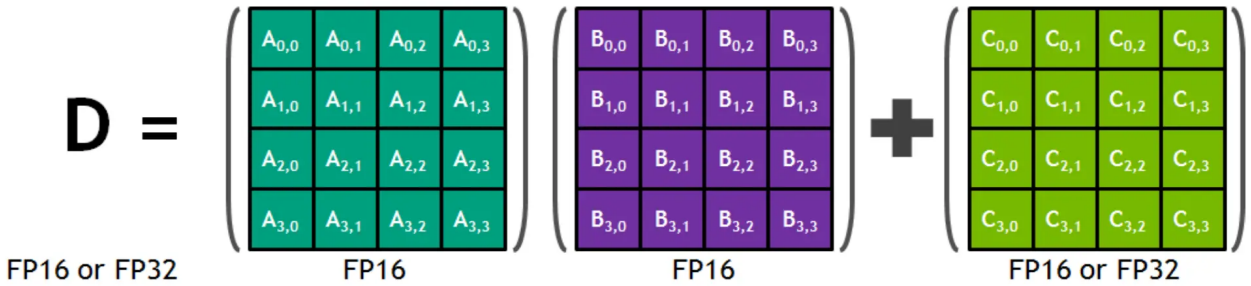
Single Compute-Intensive Slice (1): SIMD

Single Instruction, Multiple Data

- **Tensor Instructions (Tensor Core):** Mixed Precision

$$A \times B + C$$

  - A single instruction performs a 4×4 matrix operation.



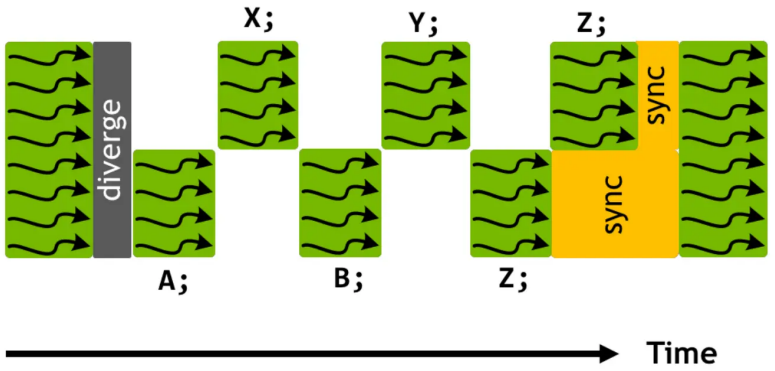
- **x86 SIMD Evolution:**
  - MMX (MultiMedia eXtension, 64-bit MM) $\rightarrow$ SSE (Streaming SIMD Extensions, 128-bit) $\rightarrow$ AVX (Advanced Vector eXtensions, 256-bit) $\rightarrow$ AVX512 (512-bit)

Single Compute-Intensive Slice (2): SIMT

Single Instruction, Multiple Threads

- **One PC (Program Counter)** controls **32 execution flows simultaneously**.
  - The number of logical threads can be even larger.
- Each execution flow has its own **registers**.
  - Three registers (x, y, and z) are used to store the ”thread ID”.
- Then, **a massive number of threads!**

```
if (threadIdx.x < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
__syncwarp()
```



Takeaways

- Most of the synchronization problems you will face are just variations of the **Producer-Consumer problem**.
- Mastering **condition variables** is enough to handle most real-world scenarios.
- The rest is just icing on the cake.

Takeaways

- Web
  - Focus: Usability
  - Pattern: Single Thread + Event Loop
  - Technologies: Promise
- High-Performance Computing
  - Focus: Task Decomposition
  - Pattern: Producer-Consumer
  - Technologies: MPI / OpenMP
- Data Centers
  - Focus: System Calls
  - Pattern: Threads-Coroutines
  - Technologies: Goroutine
- AI
  - Focus: Parallel Computation & Scalability
  - Pattern: Data Parallelism + Model Parallelism
  - Technologies: SIMIT / CUDA / TensorRT