18-742:
Computer Architecture & Systems

NVIDIA Tesla: A Unified Graphics and Computing Architecture

Prof. Phillip Gibbons

Spring 2026, Lecture 18

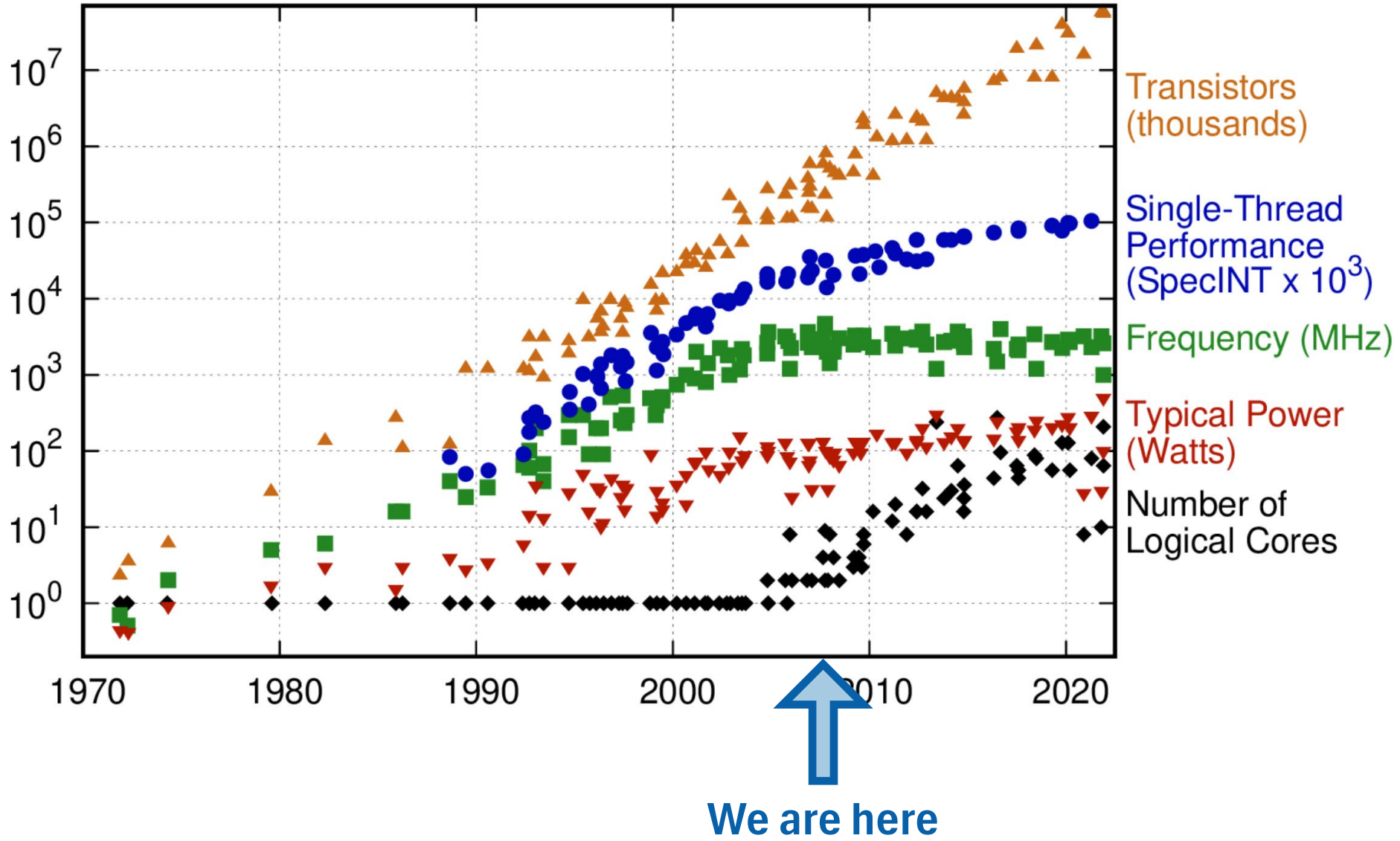
“NVIDIA Tesla: A Unified Graphics and Computing Architecture”

Erik Lindholm, John Nicholls, Stuart Oberman, John Montrym 2008

- **Erik: Distinguished engineer => Senior Distinguished Architect**
  - Master Inventor, co-architected the first GPU. Recently retired
- **John N: Director => Chief Compute Architect for GPUs**
  - Died 2011: “Without John Nickolls, there'd be no CUDA”
- **Stuart: Design manager => VP GPU ASIC Engineering**
  - IEEE Fellow 2024
- **John M: Chief Architect**




50 Years of Microprocessor Trend Data



Road to Unification

- **Vertex processors: low-latency, high-precision math operations**
- **Pixel processors: high-latency, lower-precision texture filtering**
- **3:1 ratio of Pixel to Vertex processors, but workload varies**
- **Unification advantages:**
  - Enables dynamic load balancing
  - Intro new graphics shader stages (e.g., geometric shaders)
  - New GPU parallel-computing capability

Tesla Architecture: GeForce 8800

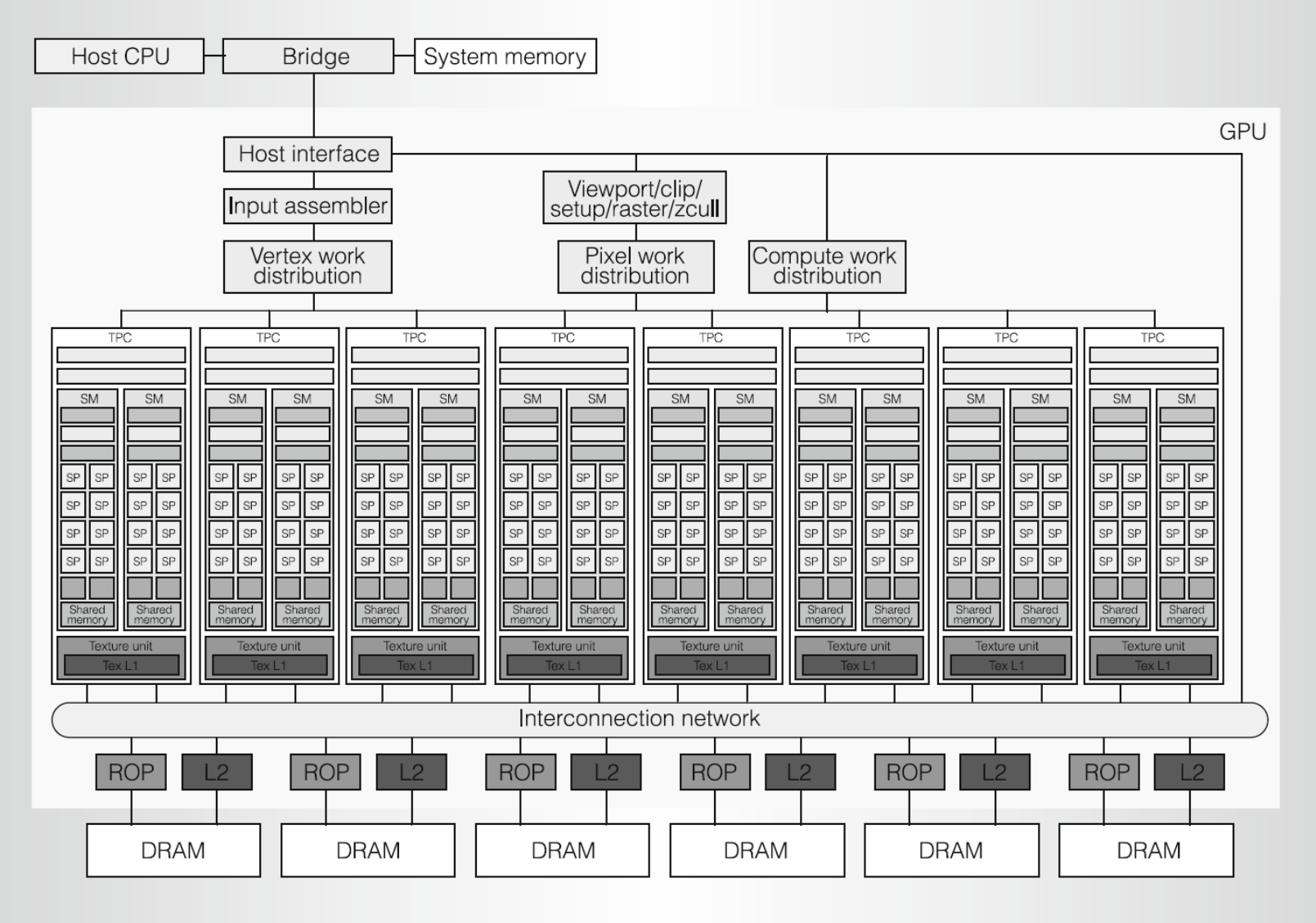
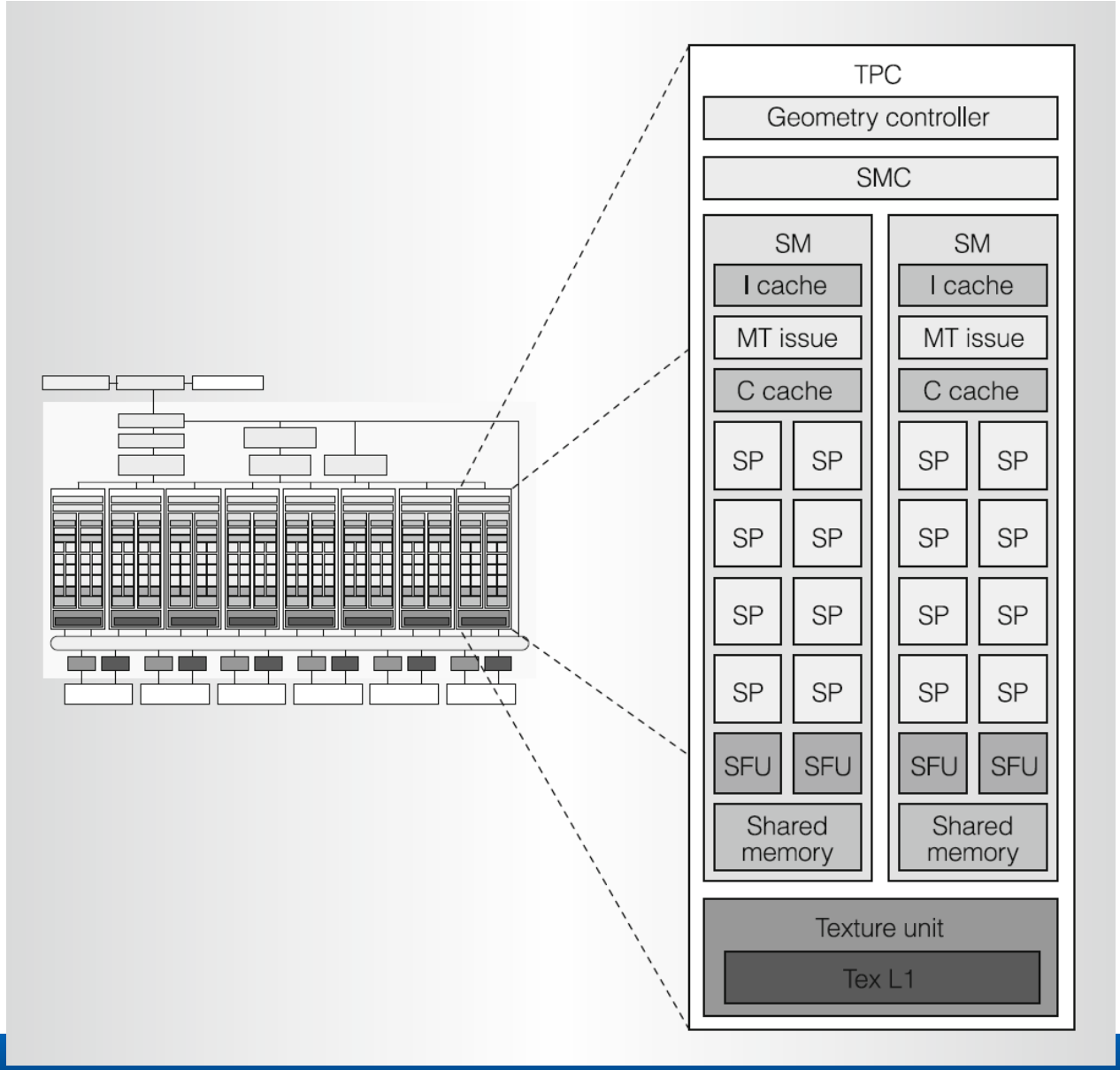
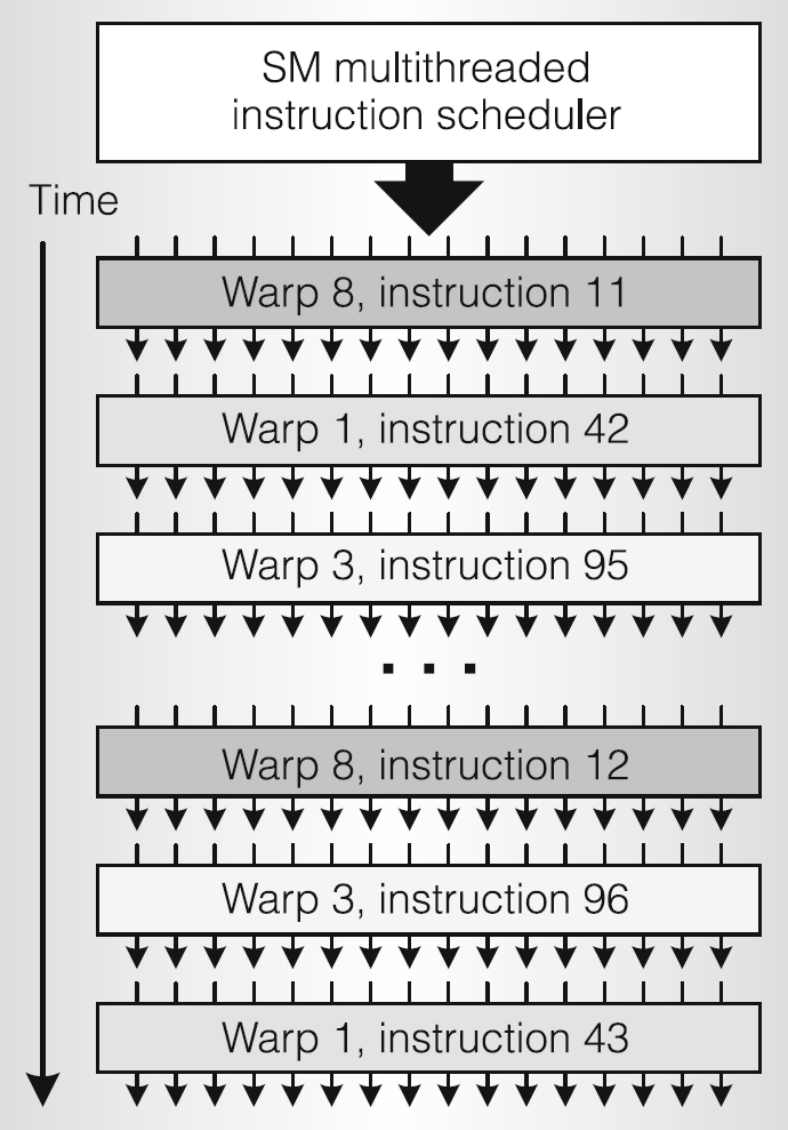


Figure 1. Tesla unified graphics and computing GPU architecture. TPC: texture/processor cluster; SM: streaming multiprocessor; SP: streaming processor; Tex: texture, ROP: raster operation processor.

Texture/processor cluster (TPC)



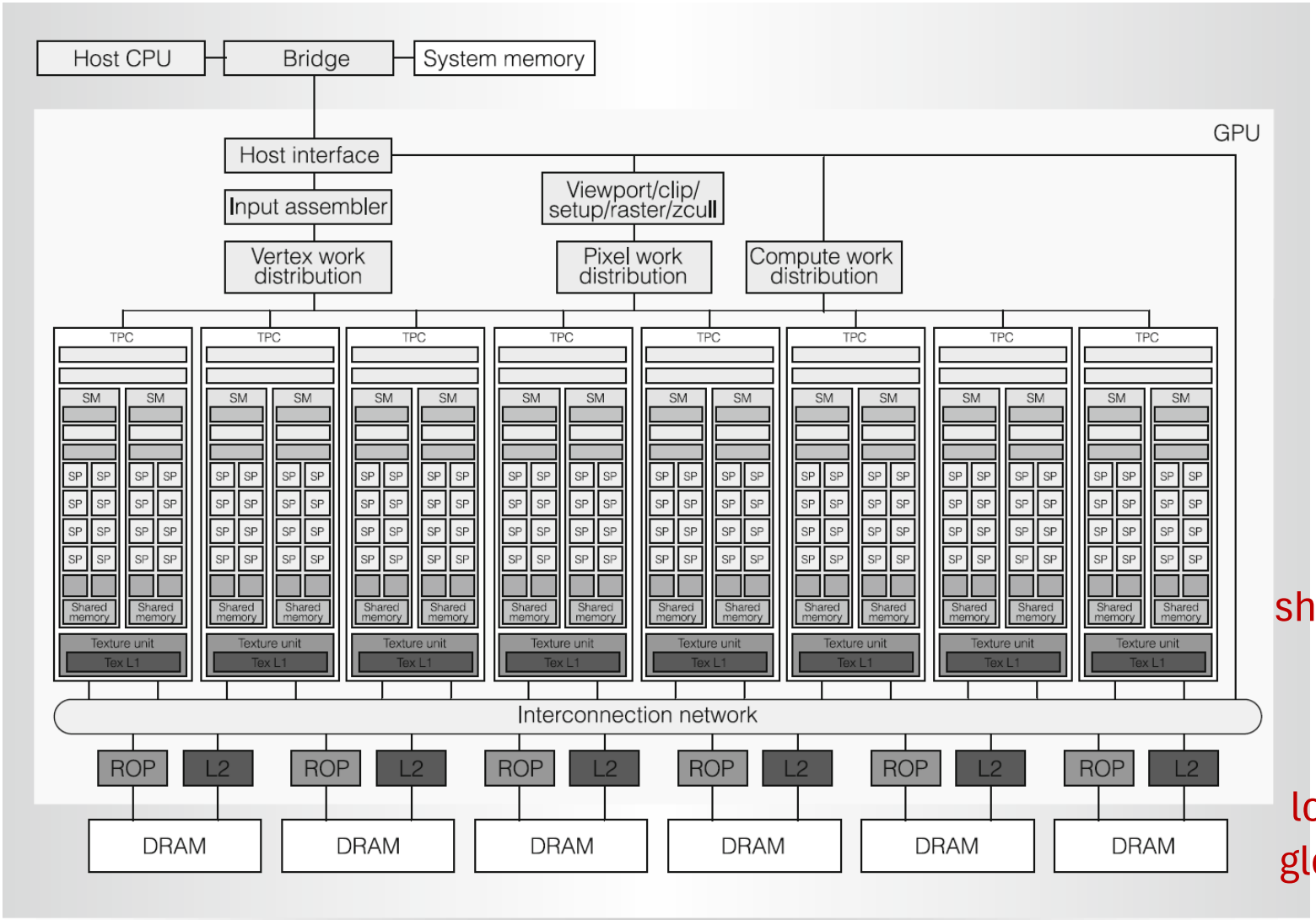
Single-instruction, Multiple-thread (SIMT)



Warp is 32 parallel threads of the same type

Performance issue: Branch divergence within a warp

General Compute: Memory Spaces



shared memory

local memory
global memory

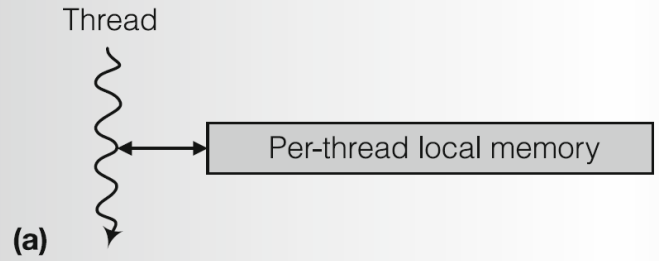
Figure 1. Tesla unified graphics and computing GPU architecture. TPC: texture/processor cluster; SM: streaming multiprocessor; SP: streaming processor; Tex: texture, ROP: raster operation processor.

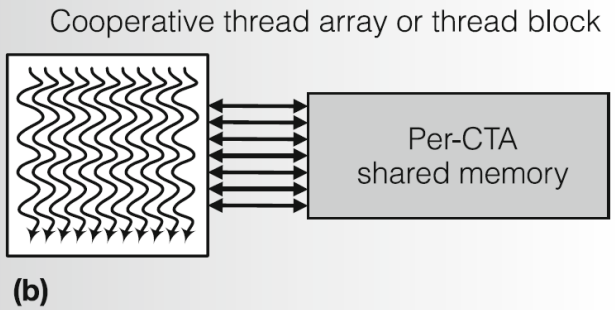
Virtual address space + page tables + hierarchy of TLBs

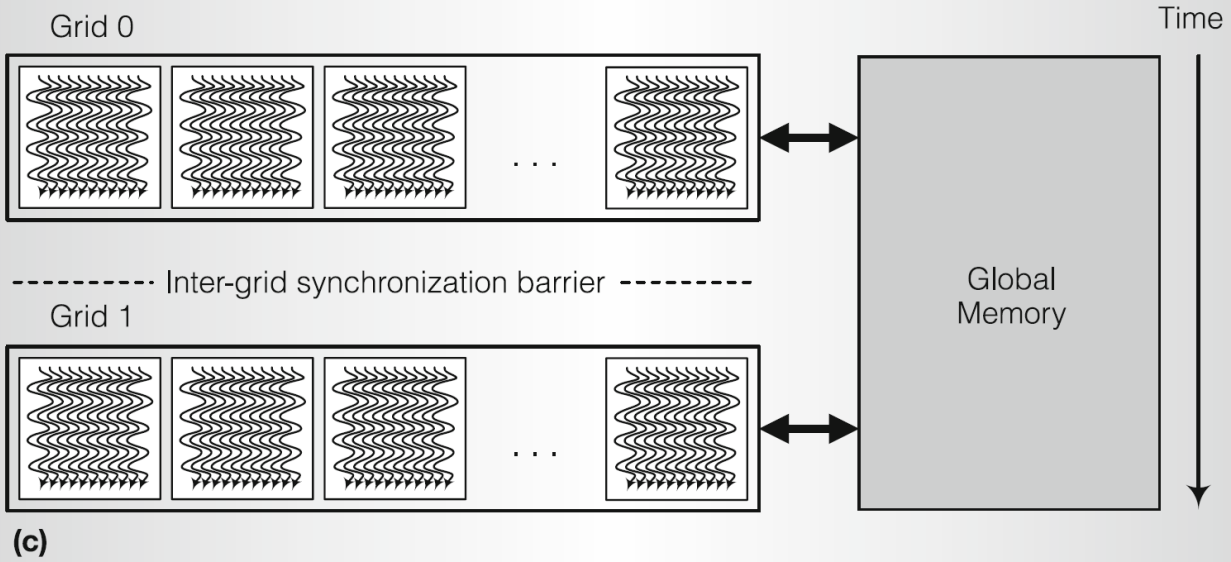
Cooperative Thread Array (Thread Block)

- **An array of concurrent threads executing the same thread program**
  - Can cooperate to compute a result
  - Each thread has a unique TID (1D, 2D, or 3D)
  - Share data in global or shared memory

Levels of Parallel Granularity







A GPU computing program executes on any size of GPU w/o recompiling

Discussion: Summary Question #1

What Did the Paper Get Right?

State the 3 most important things the paper says.

These could be some combination of the motivations, observations, interesting parts of the design, or clever parts of the implementation.

Throughput Computing: Properties

- Extensive data parallelism
- Modest task parallelism
- Intensive FP arithmetic
- Latency tolerant
- Streaming data flow with little reuse
- Modest inter-thread synchronization/communication

C vs. CUDA

```
void addMatrix
    (float *a, float *b, float *c, int N)
{
    int i, j, idx;
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            idx = i + j*N;
            c[idx] = a[idx] + b[idx];
        }
    }
}
void  main()
{
    . . .
    addMatrix(a, b, c, N);
}
```

C code

```
__global__ void addMatrixG
    (float *a, float *b, float *c, int N)
{
    int i = blockIdx.x*blockDim.x + threadIdx.x;
    int j = blockIdx.y*blockDim.y + threadIdx.y;
    int idx = i + j*N;
    if (i < N && j < N)
        c[idx] = a[idx] + b[idx];
}

void  main()
{
    dim3 dimBlock (blocksize, blocksize);
    dim3 dimGrid (N/dimBlock.x, N/dimBlock.y);
    addMatrixG<<<dimGrid, dimBlock>>>(a, b, c, N);
}
```

CUDA code

GeForce 8800 Ultra: Specs

- 681M transistors, 470 mm^2 in 90-nm CMOS
- 128 SP cores in 16 SMs
- 12,288 processor threads
- 1.5 GHz processor
- Peak 576 Gflops
- 768 MB GDDR3 DRAM
- 384-pin DRAM interface, 1.08 GHz DRAM clock
- 104 GB/s peak DRAM BW
- Typical power: 150 W at 1.3 V

Discussion: Summary Question #2

What Did the Paper Get Wrong?

Describe the paper's single most glaring deficiency.

Every paper has some fault. Perhaps an experiment was poorly designed or the main idea had a narrow scope or applicability.

“A Case for Speculative Address Translation with Rapid Validation for GPUs”

Junhyeok Park, Osang Kwon, Yongho Lee, Seongwook Kim, Gwangeun Byeon, Jihun Yoon, Prashant Nair, Seokin Hong 2024

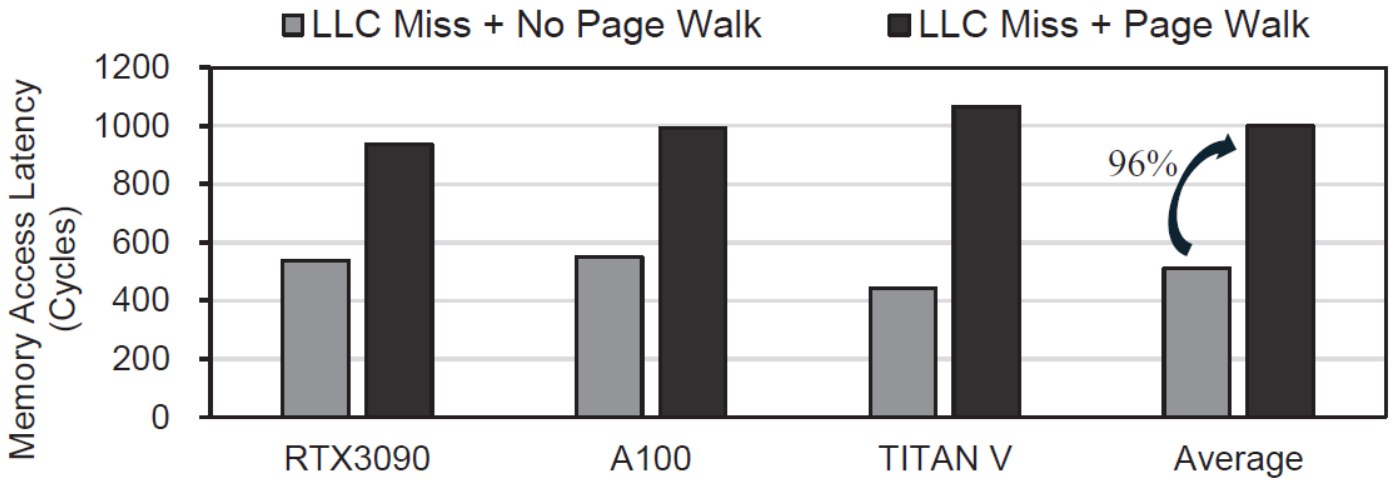


Fig. 1: The latency of page walks on memory access latency in commodity GPUs [48], [53], [54]. On average, by using micro-benchmarks, we see that commodity GPUs have up to 1.96× higher memory access latency (nearly 1000 cycles) due to page walks.

“A Case for Speculative Address Translation with Rapid Validation for GPUs”

Junhyeok Park, Osang Kwon, Yongho Lee, Seongwook Kim, Gwangeun Byeon, Jihun Yoon, Prashant Nair, Seokin Hong 2024

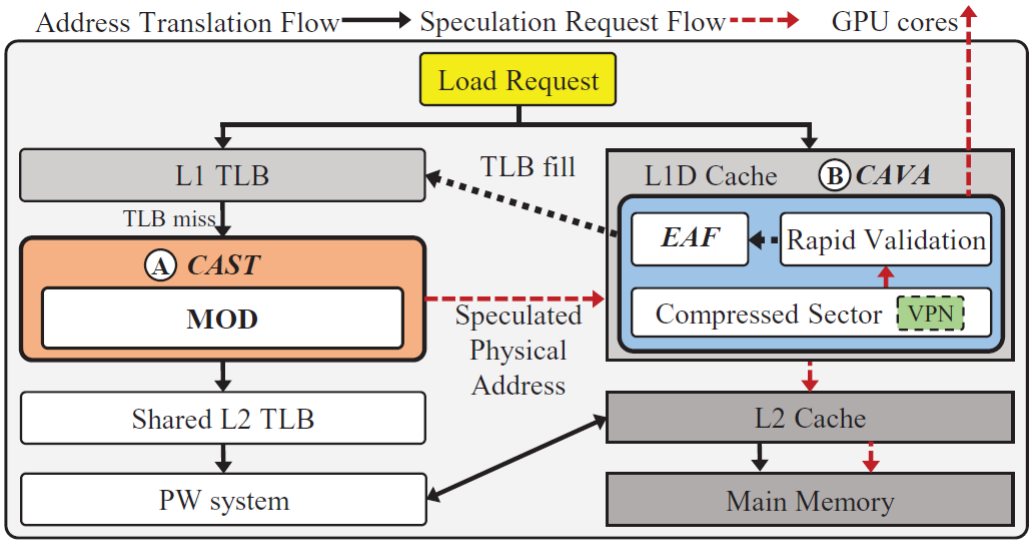


Fig. 6: An overview of Avatar.

CAST: Contiguity-aware Speculative Translation

CAVA: In-Cache Validation

To Read for Wednesday

"Splitwise: Efficient Generative LLM Inference Using Phase Splitting"

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Inigo Goiri, Saeed Maleki, Ricardo Bianchini 2024

Optional Further Reading:

"Neural Acceleration for General-Purpose Approximate Programs"

Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, Doug Burger 2012