Parallel programming: Introduction to GPU architecture

Graphics processing unit (GPU)

or

- Graphics rendering accelerator for computer games
  - Mass market: low unit price, amortized R&D
  - Increasing programmability and flexibility
- Inexpensive, high-performance parallel processor
  - GPUs are everywhere, from cell phones to supercomputers
- General-Purpose computation on GPU (GPGPU)

GPUs in high-performance computing

- GPU/accelerator share in Top500 supercomputers
  - In 2010: 2%
  - In 2018: 22%

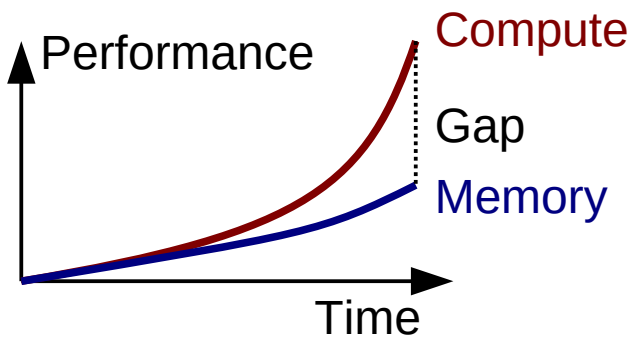
- 2016+ trend:
  Heterogeneous multi-core processors influenced by GPUs

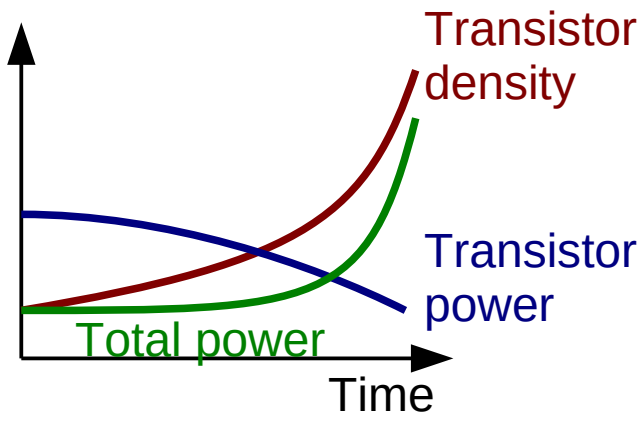
#1 Summit (USA)
4,608 × (2 Power9 CPUs + 6 Volta GPUs)

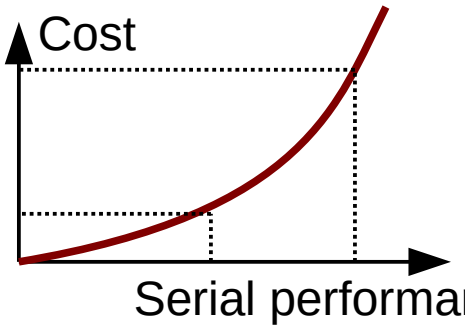
#3 Sunway TaihuLight (China)
40,960 × SW26010 (4 big + 256 small cores)

Technology evolution

- Memory wall
  - Memory speed does not increase as fast as computing speed
  - Harder to hide memory latency
- Power wall
  - Power consumption of transistors does not decrease as fast as density increases
  - Performance is now limited by power consumption
- ILP wall
  - Law of diminishing returns on Instruction-Level Parallelism
  - Pollack rule: cost ≃ performance²







Usage changes

- New applications demand **parallel processing**
  - Computer games : 3D graphics
  - Search engines, social networks… “big data” processing
- New computing devices are **power-constrained**
  - Laptops, cell phones, tablets…
    - Small, light, battery-powered
  - Datacenters
    - High power supply and cooling costs

Latency vs. throughput

- **Latency**: time to solution
  - Minimize time, at the expense of power
  - Metric: time
    e.g. seconds
- **Throughput**: quantity of tasks processed per unit of time
  - Assumes unlimited parallelism
  - Minimize energy per operation
  - Metric: operations / time
    e.g. Gflops / s
- CPU: optimized for latency
- GPU: optimized for throughput

Amdahl's law

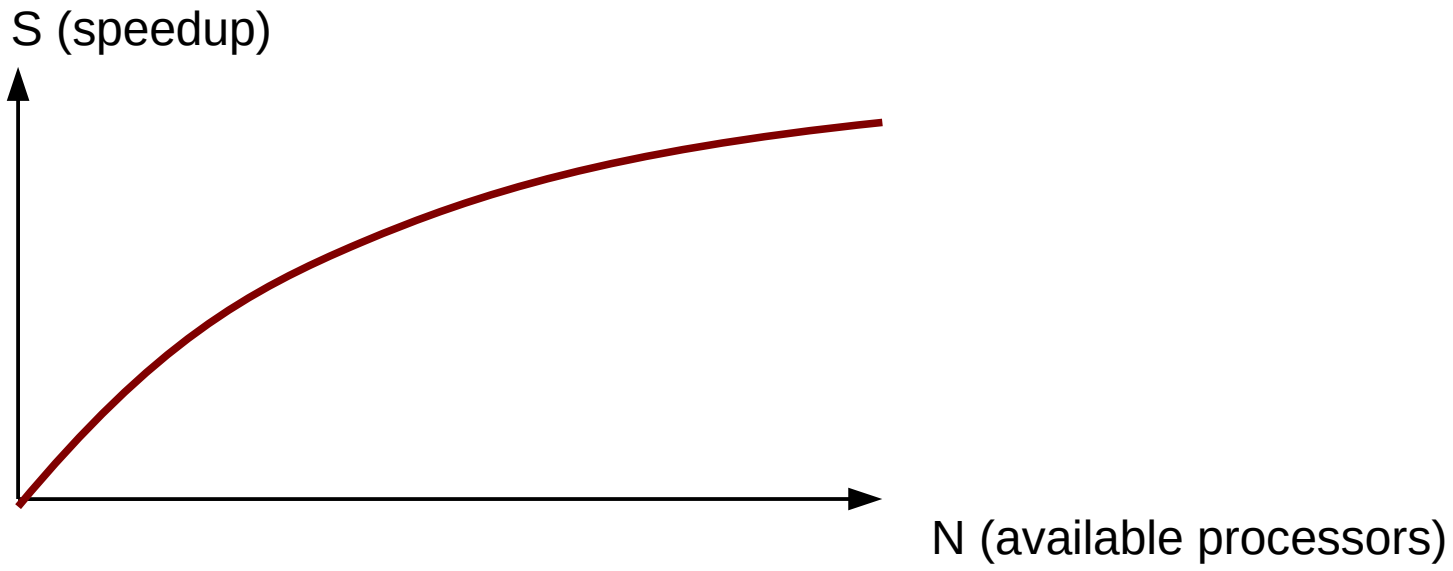
- Bounds speedup attainable on a parallel machine

$$S = \frac{1}{(1-P) + \frac{P}{N}}$$

Time to run sequential portions

Time to run parallel portions

S	Speedup
P	Ratio of parallel portions
N	Number of processors



G. Amdahl. Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. AFIPS 1967.

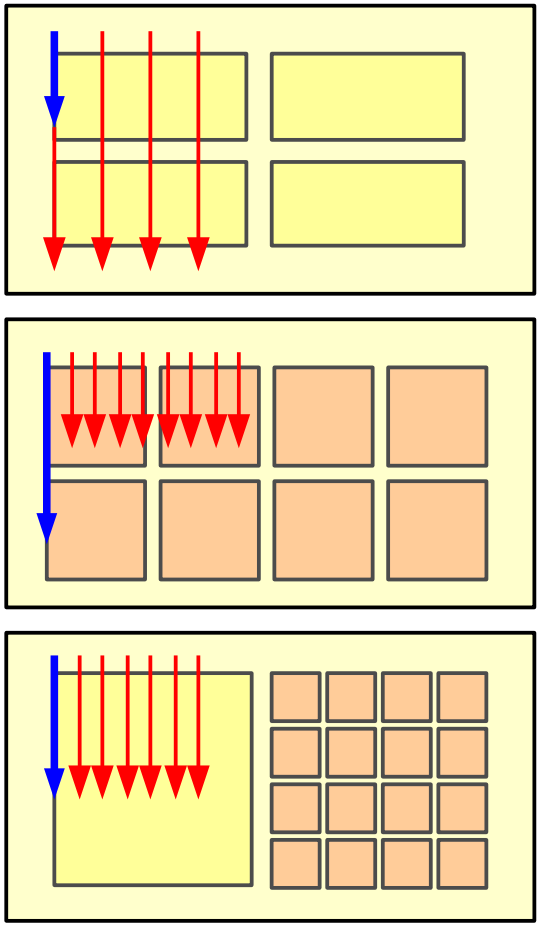
Why heterogeneous architectures?

$$S = \frac{1}{(1-P) + \frac{P}{N}}$$

Time to run sequential portions

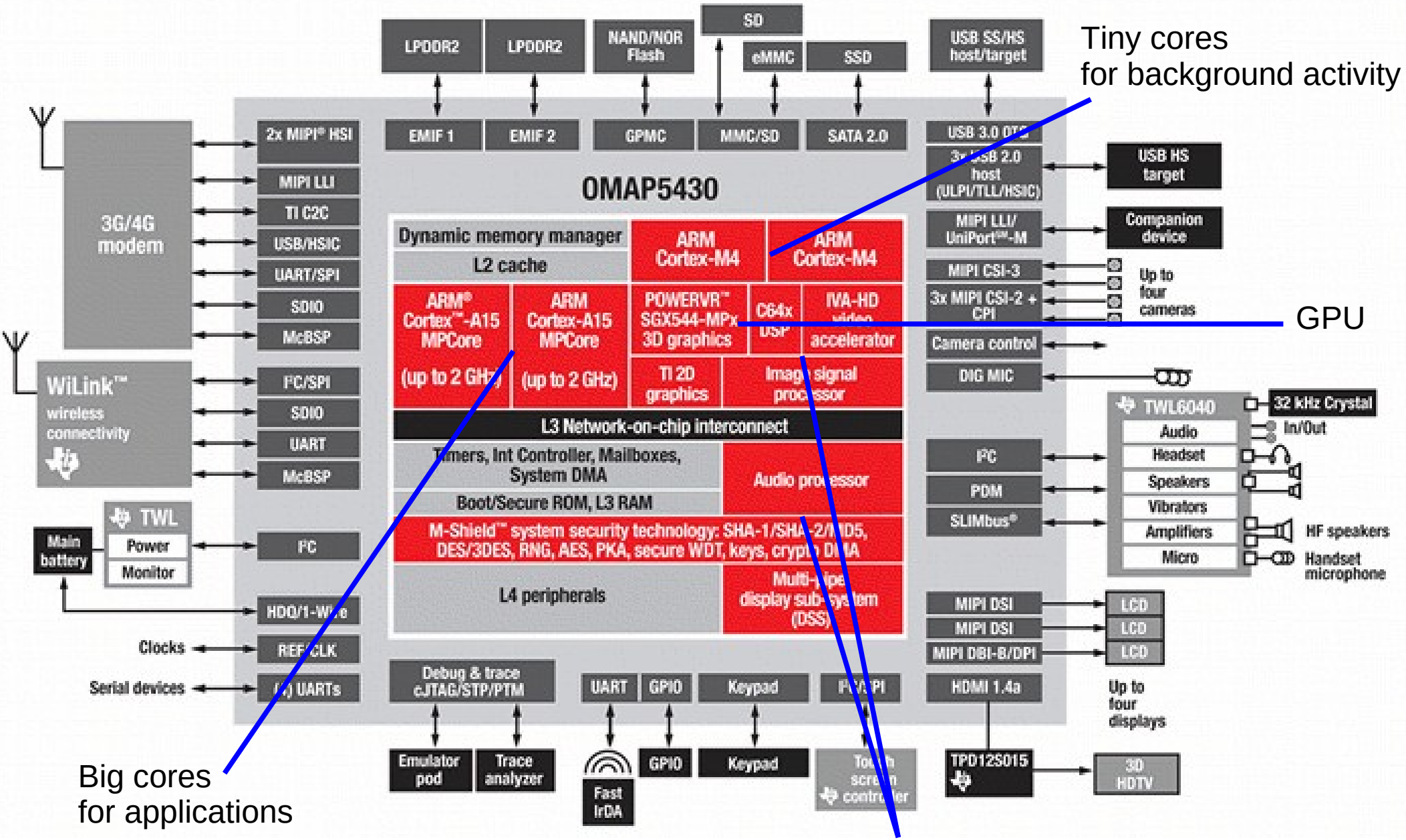
Time to run parallel portions

- Latency-optimized multi-core (CPU)
  - Low efficiency on parallel portions: spends too much resources
- Throughput-optimized multi-core (GPU)
  - Low performance on sequential portions
- Heterogeneous multi-core (CPU+GPU)
  - Use the right tool for the right job
  - Allows aggressive optimization for latency **or** for throughput

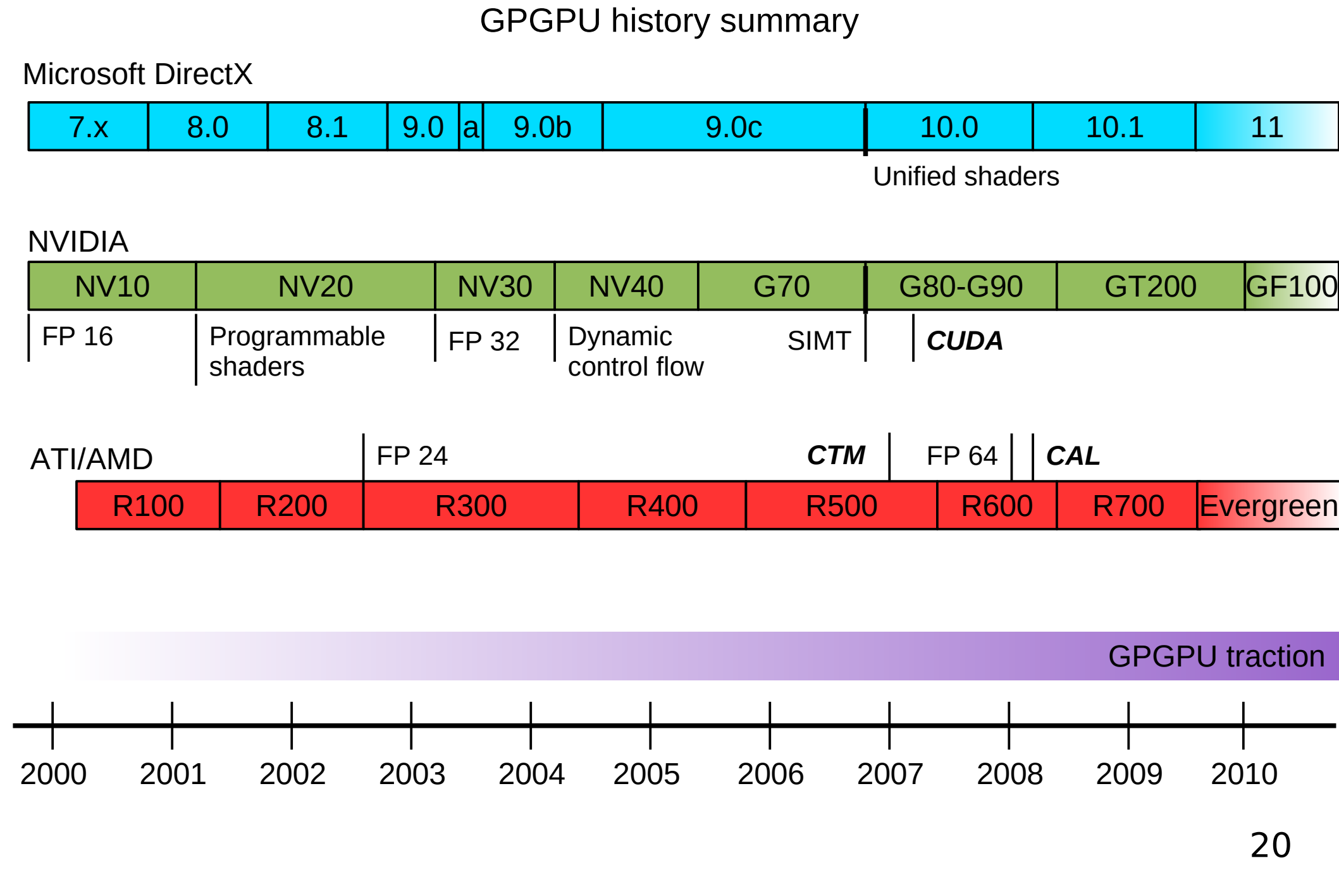


M. Hill, M. Marty. Amdahl's law in the multicore era. IEEE Computer, 2008.

Example: System on Chip for smartphone



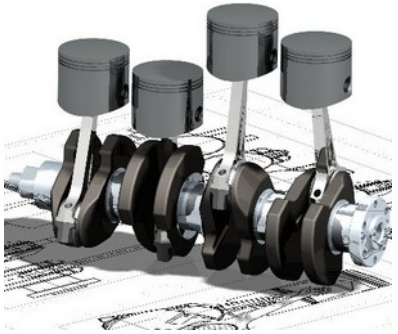
GPGPU: General-Purpose computation on GPUs

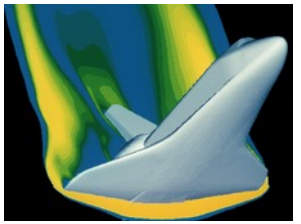


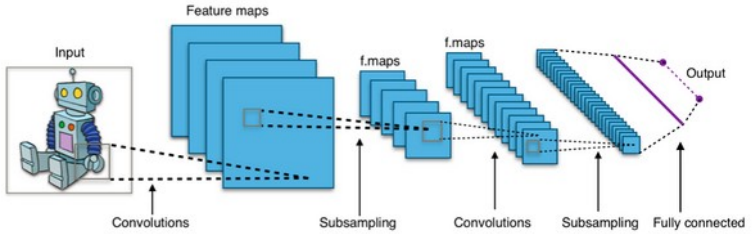
Today: what do we need GPUs for?

1. 3D graphics rendering for games
   - Complex texture mapping, lighting computations…
2. Computer Aided Design workstations
   - Complex geometry
3. High-performance computing
   - Complex synchronization, off-chip data movement, high precision
4. Convolutional neural networks
   - Complex scheduling of low-precision linear algebra

- One chip to rule them all
  - Find the common denominator

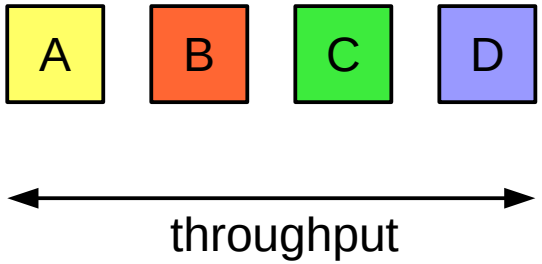




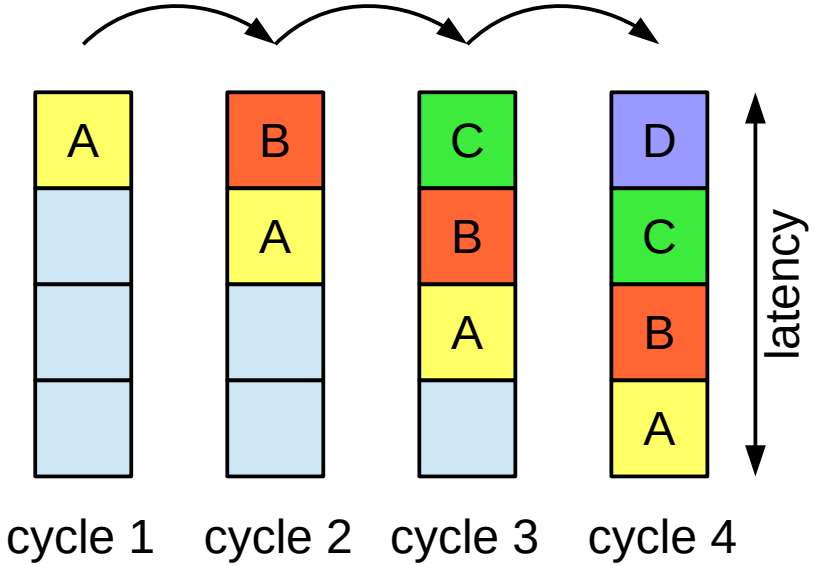


Uses of parallelism

- “Horizontal” parallelism for throughput
  - More units working in parallel



- “Vertical” parallelism for latency hiding
  - Pipelining: keep units busy when waiting for dependencies, memory



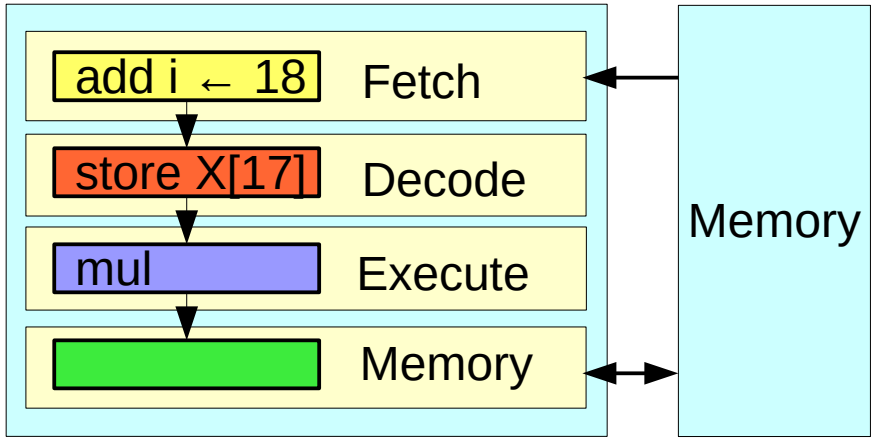
Sequential processor

```
for i = 0 to n-1
   X[i] ← a * X[i]
```

Source code

```
   move  i ← 0
loop:
   load  t ← X[i]
   mul   t ← a×t
   store X[i] ← t
   add   i ← i+1
   branch i<n? loop
```

Machine code

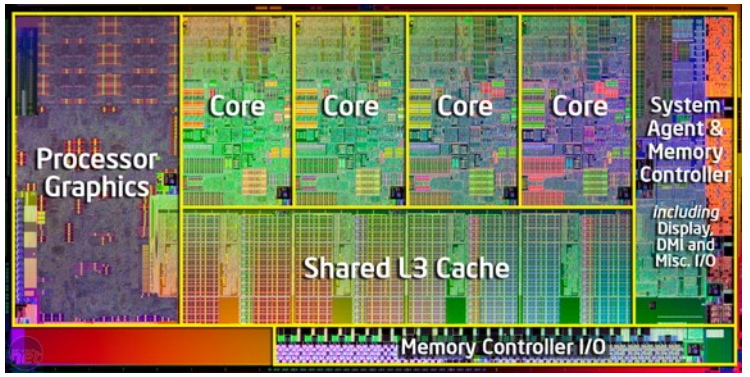


Sequential CPU

- Focuses on instruction-level parallelism
  - Exploits ILP: vertically (pipelining) and horizontally (superscalar)

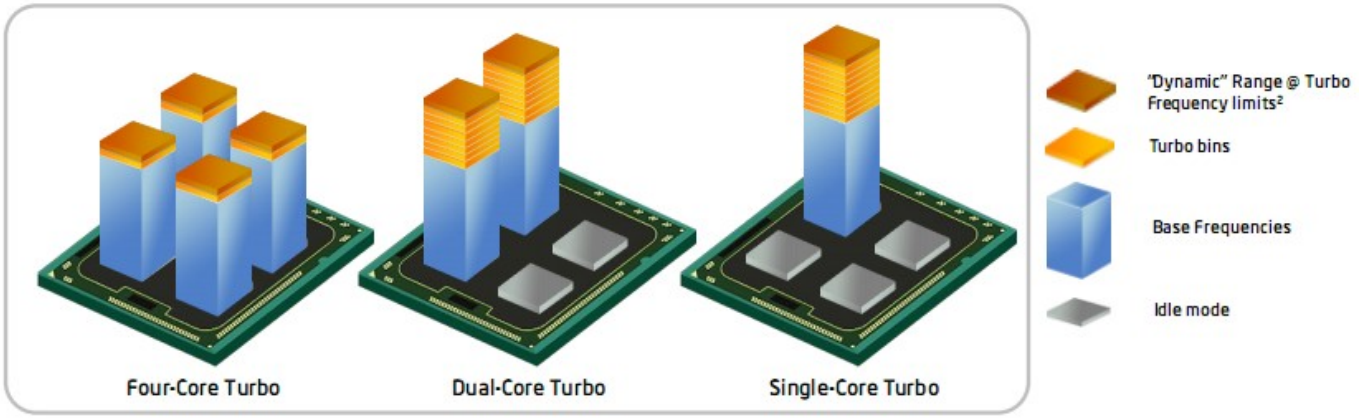
The incremental approach: multi-core

- Several processors
  on a single chip
  sharing one memory space



Intel Sandy Bridge

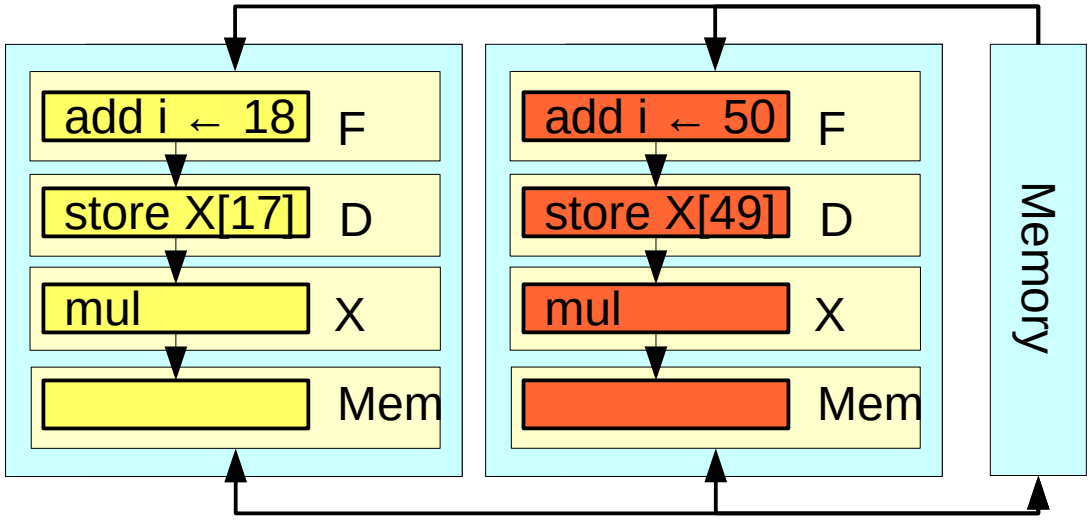
- Area: benefits from Moore's law
- Power: extra cores consume little when not in use
  - e.g. Intel Turbo Boost



Source: Intel

Homogeneous multi-core

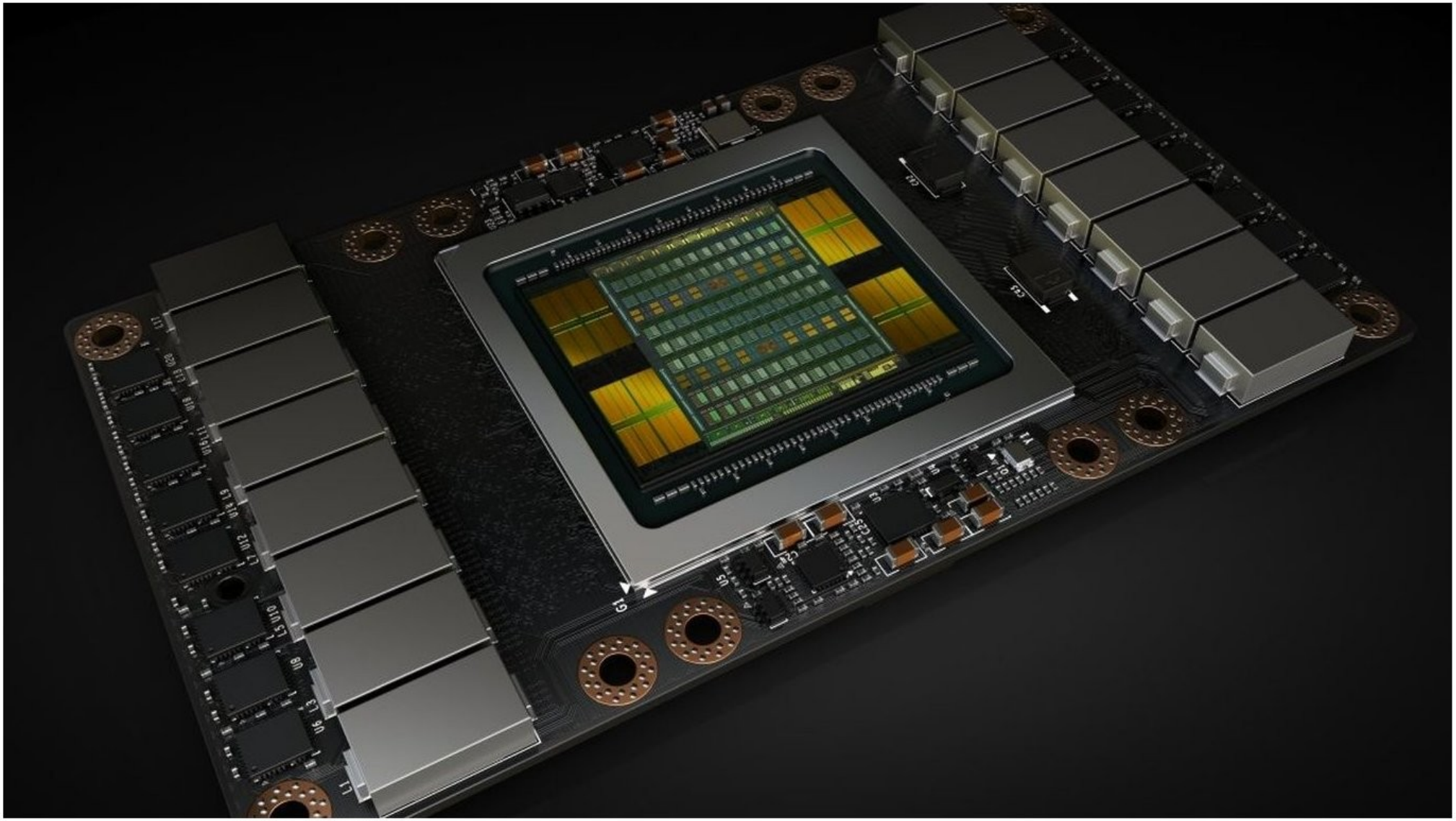
- Horizontal use of thread-level parallelism



Threads: T0 T1

- Improves peak throughput

What is inside a graphics card?



NVIDIA Volta V100 GPU. Artist rendering!

External memory: embedded GPU

Most GPUs today are integrated

- Same physical memory
- May support memory coherence
  - GPU can read directly from CPU caches
- More contention on external memory