CSC2/458 Parallel and Distributed Systems Machines and Models

Sreepathi Pai

January 23, 2018

URCS

Outline

Outline

Goals

What is the goal of parallel programming?

Scalability

Why is scalability important?

Outline

Speedup

$$\mathrm{Speedup}(n) = \frac{T_1}{T_n}$$

- T_1 is time on one processor
- T_n is time on n processors

Amdahl's Law

Let:

- T_1 be $T_{serial} + T_{parallelizable}$
- T_n is then $T_{serial} + \frac{T_{parallelizable}}{n}$, assuming perfect scalability

Divide both terms T_1 and T_n by T_1 to obtain serial and parallelizable ratios.

$$\text{Speedup}(n) = \frac{1}{r_{serial} + \frac{r_{parallelizable}}{n}}$$

Amdahl's Law – In the limit

$$\text{Speedup}(\infty) = \frac{1}{r_{serial}}$$

This is also known as *strong scalability* – work is fixed and number of processors is varied.

What are the implications of this?

Scalability Limits

Assuming infinite processors, what is the speedup if:

- serial ratio r_{serial} is 0.5 (i.e. 50%)
- serial ratio is 0.1 (i.e. 10%)
- serial ratio is 0.01 (i.e. 1%)

Current Top 5 supercomputers

- Sunway TaihuLight (10.6M cores)
- Tianhe 2 (3.1M cores)
- Piz Daint (361K cores)
- Gyoukou (19.8M cores)
- Titan (560K cores)

Source: Top 500

Weak Scalability

- Work increases as number of processors increase
  - Parallel work should increase linearly with processors
- Work $W = \alpha W + (1 - \alpha)W$
  - α is serial fraction of work
- Scaled Work $W' = \alpha W + n(1 - \alpha)W$
  - Empirical observation
  - Usually referred to as Gustafson's Law

Source:

`http://www.johngustafson.net/pubs/pub13/amdahl.htm`

Outline

Organization of Parallel Computers

Components of parallel machines:

- Processing Elements
- Memories
- Interconnect
    - how processors, memories are connnected to each other

Flynn’s Taxonomy

- Based on notion of “streams”
  - Instruction stream
  - Data stream
- Taxonomy based on number of each type of streams
  - Single Instruction - Single Data (SISD)
  - Single Instruction - Multiple Data (SIMD)
  - Multiple Instruction - Single Data (MISD)
  - Multiple Instruction - Multiple Data (MIMD)

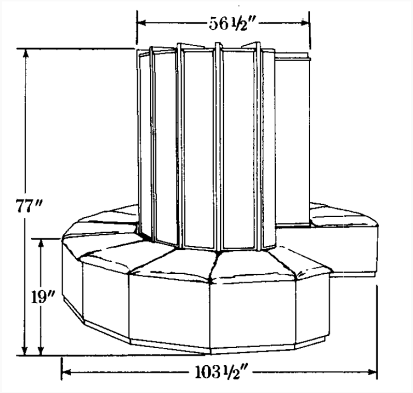
Flynn, J., (1966), “`http://ieeexplore.ieee.org/document/1447203/`Very High Speed Computing Systems”, Proceedings of the IEEE

SIMD Implementations: Vector Machines

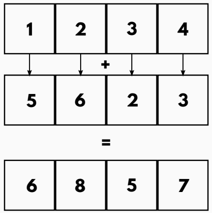
The Cray-1 (circa 1977):

- V*x* – vector registers
  - 64 elements
  - 64-bits per element
- Vector length register (V*len*)
- Vector mask register

Richard Russell, "The Cray-1 Computer System", Comm. ACM 21,1 (Jan 1978), 63-72



Vector Instructions – Vertical

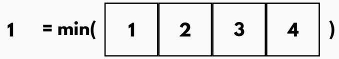


For $0 < i < \mathrm{V}len$:

```
dst[i] = src1[i] + src2[i]
```

- Most arithmetic instructions

Vector Instructions – Horizontal



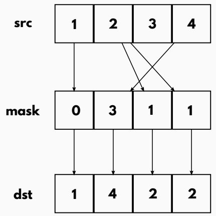
For $0 < i < \mathrm{V}\mathit{len}$:

```
dst = min(src1[i], dst)
```

Note that `dst` is a scalar.

- Mostly reductions (min, max, sum, etc.)
- Not well supported
    - Cray-1 did not have this

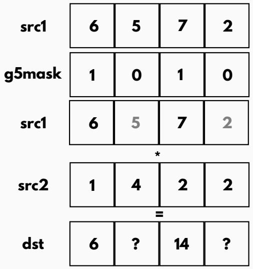
Vector Instructions – Shuffle/Permute



```
dst = shuffle(src1, mask)
```

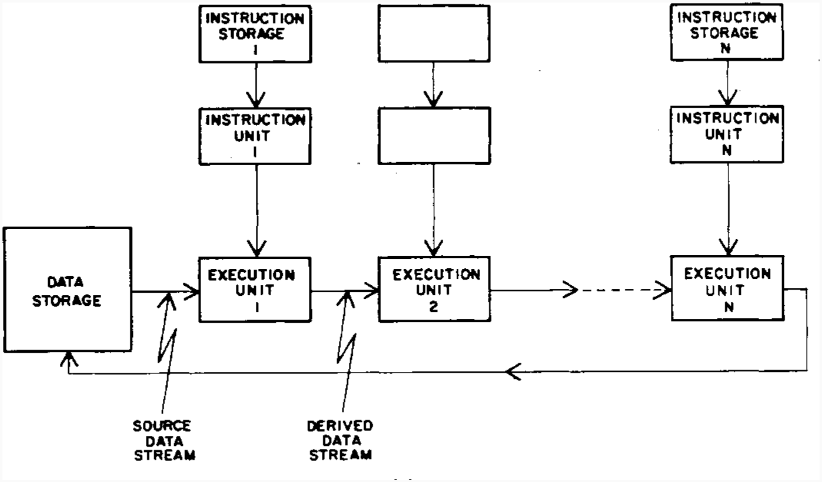
- Poor support on older implementations
- Reasonably well-supported on recent implementations

Masking/Predication



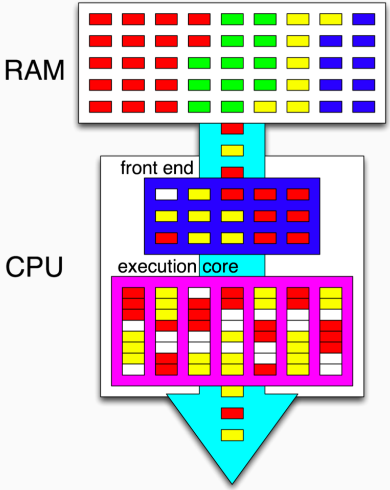
```
g5mask = gt(src1, 5)
dst = mul(src1, src2, g5mask)
```

MISD - ?



Flynn, J., (1966), "`http://ieeexplore.ieee.org/document/1447203/`Very High Speed Computing Systems", Proceedings of the IEEE

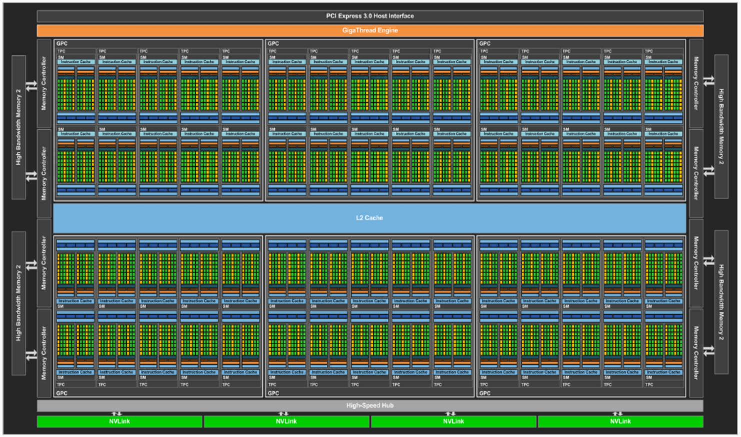
What type of machine is this? Hyperthreaded Core



Different colours in RAM indicate different instruction streams.

Source: https://en.wikipedia.org/wiki/Hyper-threading

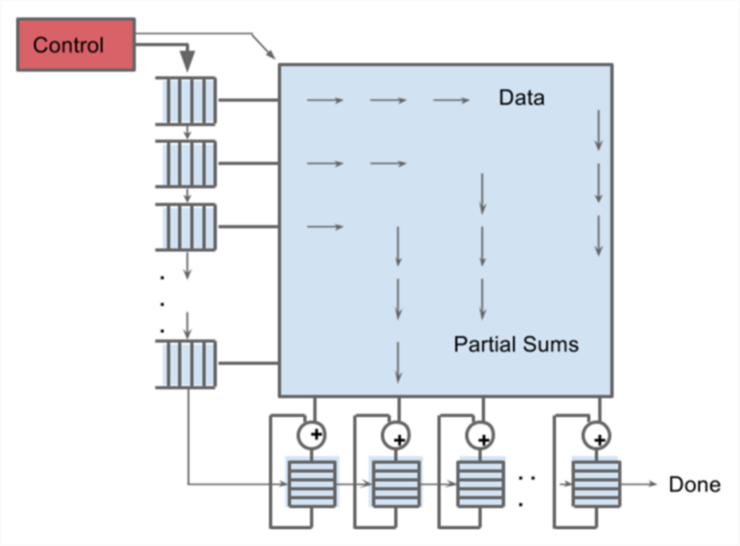
What type of machine is this? GPU



Each instruction is 32-wide.

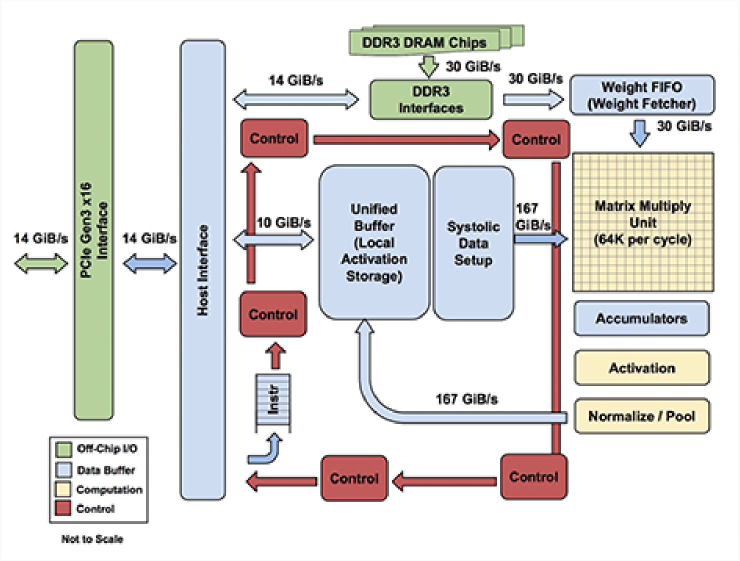
Source: https://devblogs.nvidia.com/inside-pascal/

What type of machine is this? TPU Matrix Multiply Unit



Source:
https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

TPU Overview



Source:
https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

Modern Multicores

- Multiple Cores (MIMD)
- (Short) Vector Instruction Sets (SIMD)
    - MMX, SSE, AVX (Intel)
    - 3DNow (AMD)
    - NEON (ARM)

Outline

Metrics we care about

- Latency
  - Time to complete task
  - Lower is better
- Throughput
  - Rate of completing tasks
  - Higher is better
- Utilization
  - Time “worker” (processor, unit) is busy
  - Higher is better
- Speedup
  - Higher is better

Reducing Latency

- Use cheap operations
- Which of these operations are expensive?
    - Bitshift
    - Integer Divide
    - Integer Multiply
- Latency fundamentally bounded by physics

Increasing Throughput

- Parallelize!
  - Lots of techniques, focus of this class
- Add more processors
- Need lots of work though to benefit

Speedup

- Measure speedup w.r.t. fastest serial code
    - Not parallel program on 1 processor
- Always report runtime
    - Never speedup alone
- Are superlinear speedups possible?