departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Parallel Programming Models and Architectures

Concurrency and Parallelism — 2019-20

Master in Computer Science

(Mestrado Integrado em Eng. Informática)

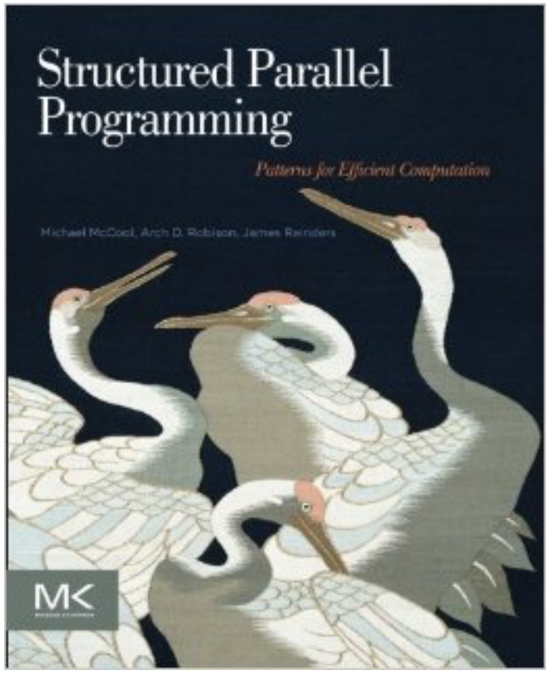
Joao Lourenço <joao.lourenco@fct.unl.pt>

Some slides and ideas take from:
http://cri.uchicago.edu/wp-content/uploads/2018/09/Intro-to-Parallel-Computing.pdf

Outline

- Parallel Programming Models
- Parallel Architectures

  – Bibliography:
    - **Chapters 1 and 2** of book
      McCool M., Arch M., Reinders J.;
      Structured Parallel Programming: Patterns for Efficient Computation;
      Morgan Kaufmann (2012);
      ISBN: 978-0-12-415993-8



Car crash simulation example

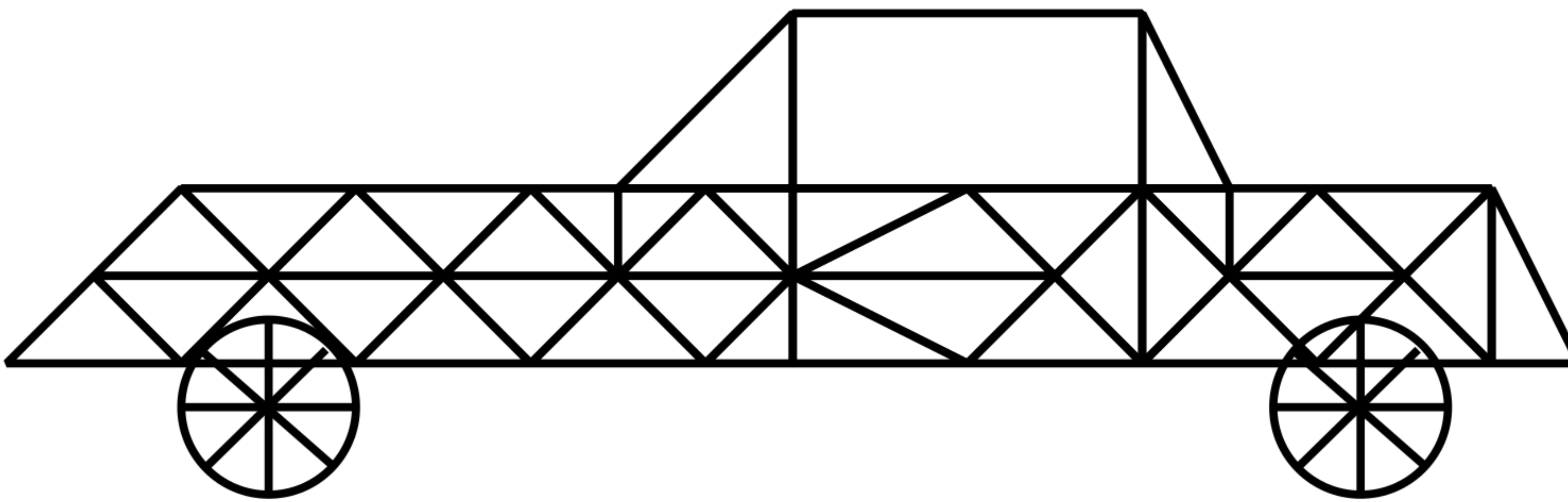
- Simplified model based on a crash simulation for the Ford Motor Company
- Illustrates various aspects common to many simulations and applications

- This example was provided by Q. Stout and C. Jablonowski of the University of Michigan

Finite Element Representation

- Car is modeled by a triangulated surface (elements)
- The simulation models the movement of the elements, incorporating the forces on the elements to determine their new position
- In each time step, the movement of each element depends on its interaction with the other elements to which it is physically adjacent
- In a crash, elements may end up touching that were not touching initially
- The state of an element is its location, velocity, and information such as whether it is metal that is bending

(Sequential) Car



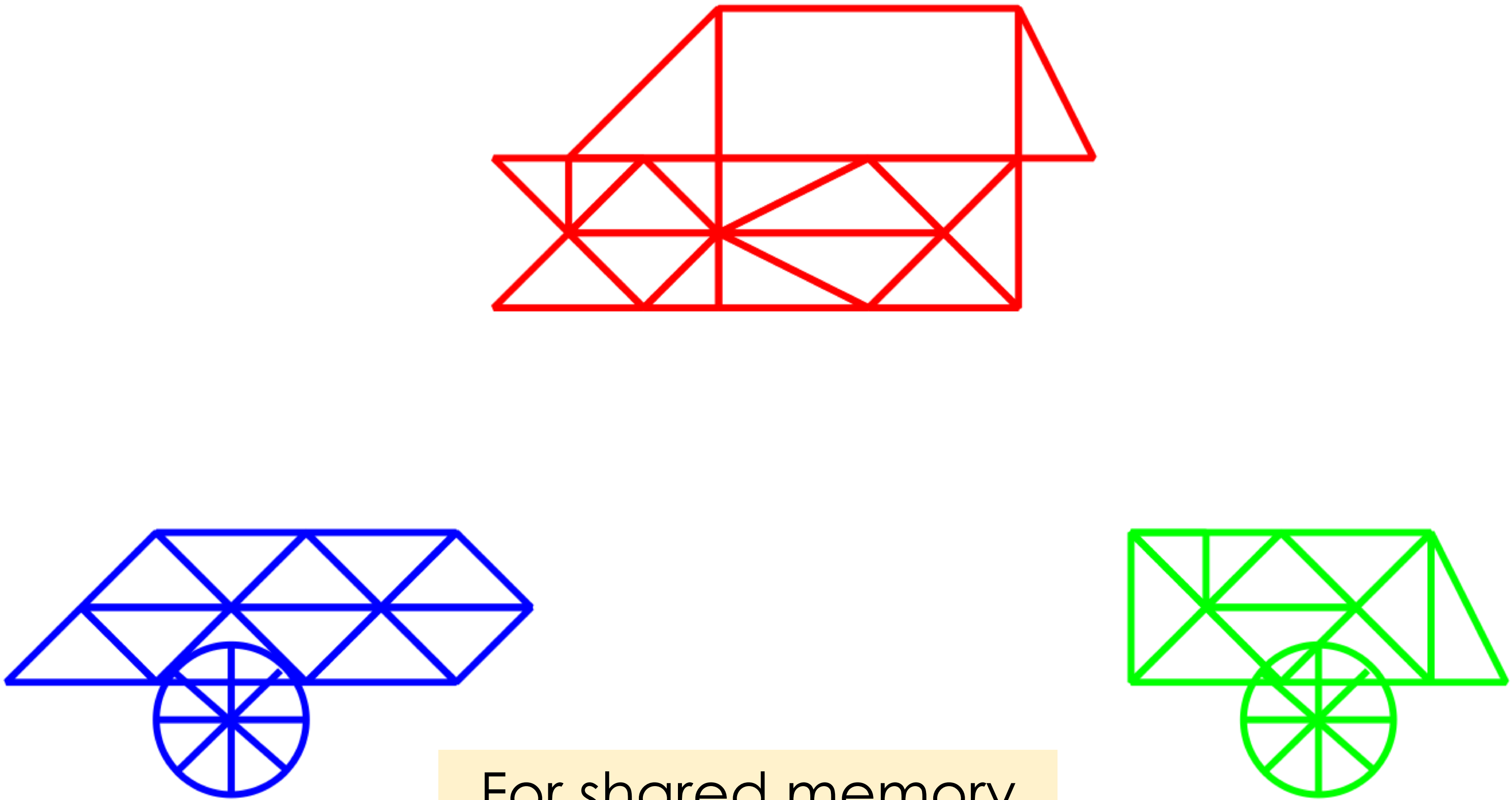
Serial Crash Simulation

```
for all elements
   read ( State(element), Properties(element), Neighbor_list(element) )
   for step=1 to end_of_simulation
      for element=1 to num_elements
         Compute State(element) for next step,
         based on the previous state of the element
         and its neighbors and the properties of the element
```

Simple Approach to Paralleization

- **Distributed Memory** – Parallel system based on processors linked with a fast network; processors communicate via messages
- **Owner Computes** – Distribute elements to processors; each processor updates its own elements
- **Single Program Multiple Data (SPMD)** – All machines run the same program on independent data; dominant form of parallel computing

Split Car



For shared memory

Basic Parallel Version

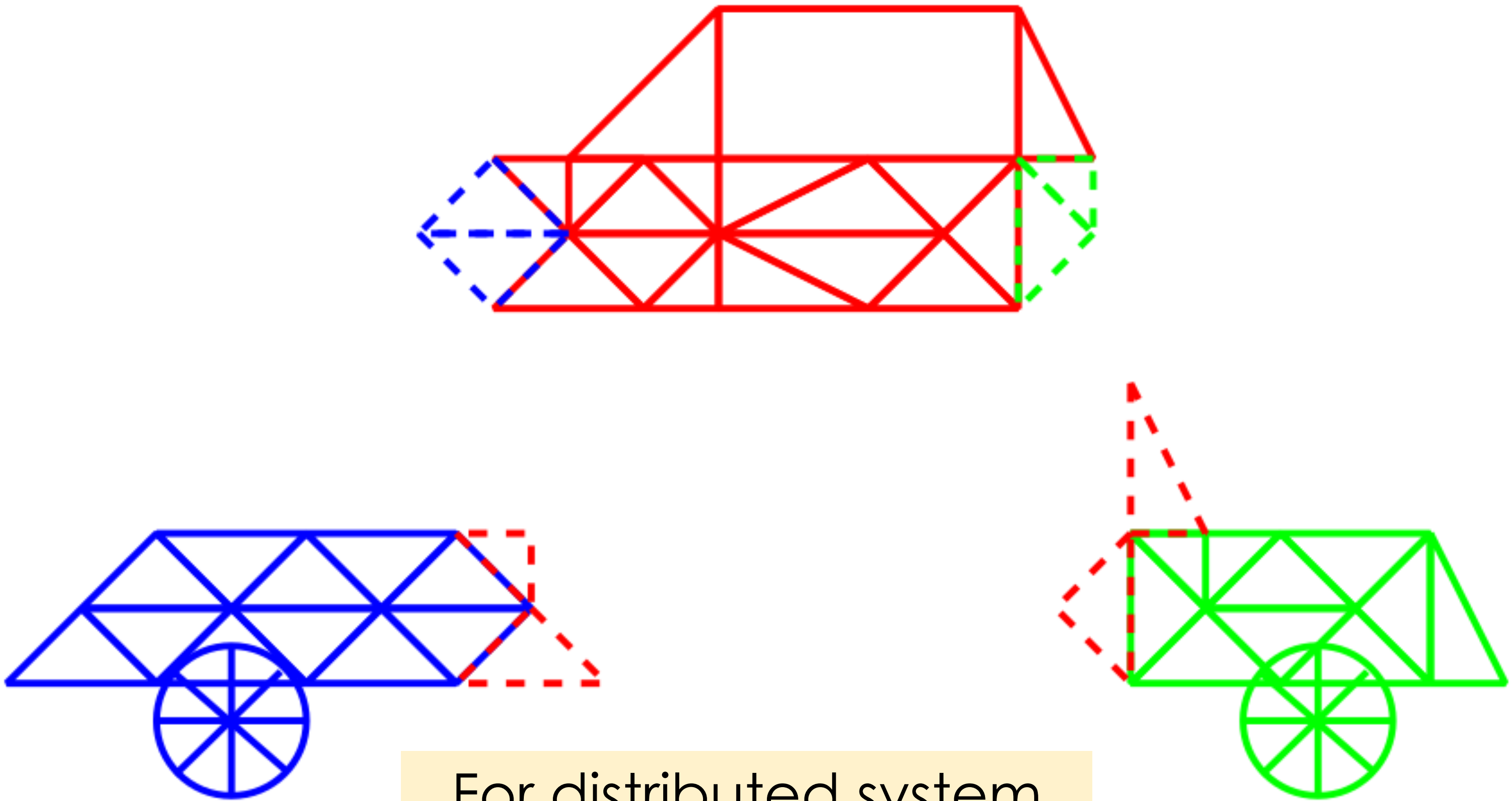
```
concurrently for all processors P
   for all elements assigned to P
      read ( State(element), ProperCes(element), Neighbor- list(element) )
      for step=1 to end_of_simulaCon
         for element=1 to num_elements_in_P
            Compute State (element) for next step, based on previous state
            of element and its neighbors, and on properties of the element
```

Notes

- Most of the code is the same as, or similar to, serial code
- High-level structure remains the same: a sequence of steps
  - The sequence is a serial construct, but
  - Now the steps are performed in parallel, but
  - Calculations for individual elements are serial

Question: In a distributed memory system, how does each processor keep track of adjacency info for neighbors in other processors?

Distributed Car (w/ ghost cells)



For distributed system

Parallel Architectures

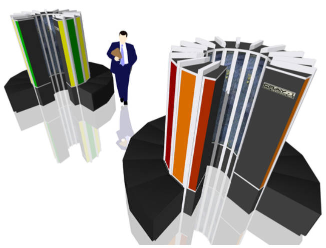
- Flynn's Taxonomy — basic concepts
- **Single Instruction (SI)** – System in which all processors execute the same instruction
- **Multiple Instruction (MI)** – System in which different processors may execute different instructions
- **Single Data (SD)** – System in which all processors operate on the same data
- **Multiple Data (MD)** – System in which different processors may operate on different data

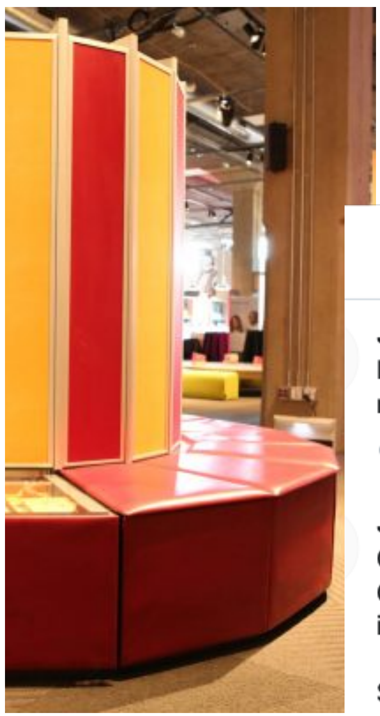
- M. J. Flynn. Some computer organizations and their effectiveness. IEEE Transactions on Computers, C-21(9):948–960, 1972.

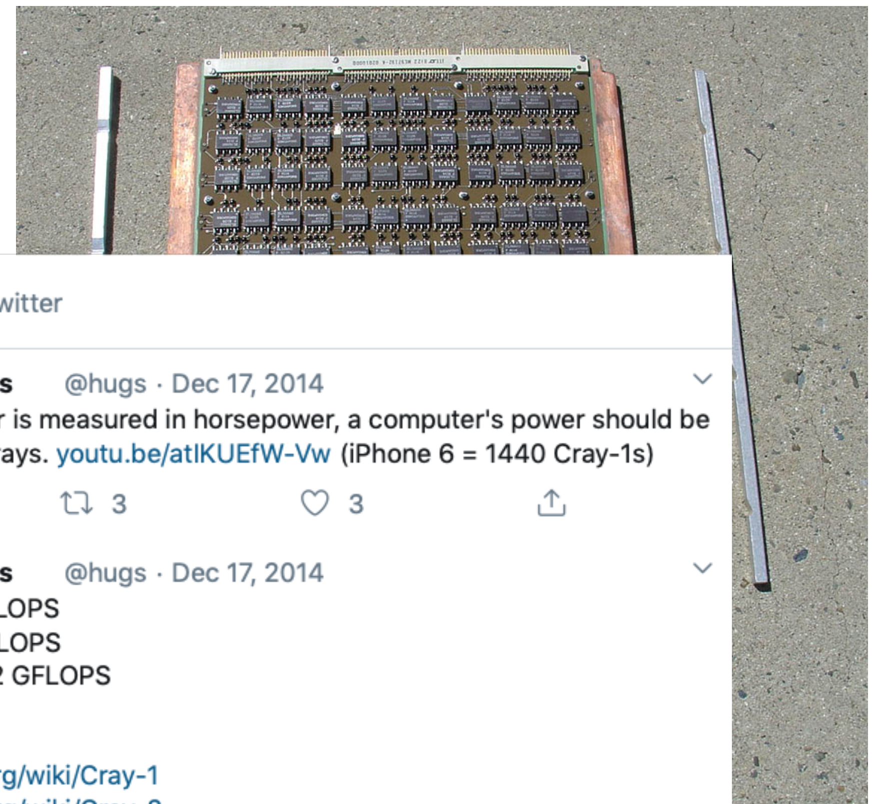
Flynn's Taxonomy

- **SISD** – Classic von Neumann architecture; serial computer
- **MIMD** – Collections of autonomous processors that can execute multiple independent programs; each of which can have its own data stream
- **SIMD** – Data is divided among the processors and each data item is subjected to the same sequence of instructions; GPUs, Advanced Vector Extensions (AVX)
- **MISD** – Very rare; systolic arrays; smart phones carried by Chupacabras 🤣

CRAY-1 Vector Machine (1976)

Cray-1	
 3D rendering of two Cray-1 with a figure as scale	
Design	
Manufacturer	Cray Research
Designer	Seymour Cray
Release date	1975
Units sold	Over 80
Price	US$7.9 million in 1977 (equivalent to $33.3 million in 2019)
Casing	
Dimensions	Height: 196 cm (77 in)[1] Dia. (base): 263 cm (104 in)[1] Dia. (columns): 145 cm (57 in)[1]
Weight	5.5 tons (Cray-1A)
Power	115 kW @ 208 V 400 Hz[1]
System	
Front-end	Data General Eclipse
Operating system	COS & UNICOS
CPU	64-bit processor @ 80 MHz[1]
Memory	8.39 Megabytes (up to 1 048 576 words)[1]
Storage	303 Megabytes (DD19 Unit)[1]
FLOPS	160 MFLOPS
Successor	Cray X-MP





Search Twitter

Jason Huggins @hugs · Dec 17, 2014

If a car's power is measured in horsepower, a computer's power should be measured in crays. youtu.be/atIKUEfW-Vw (iPhone 6 = 1440 Cray-1s)

2 3 3

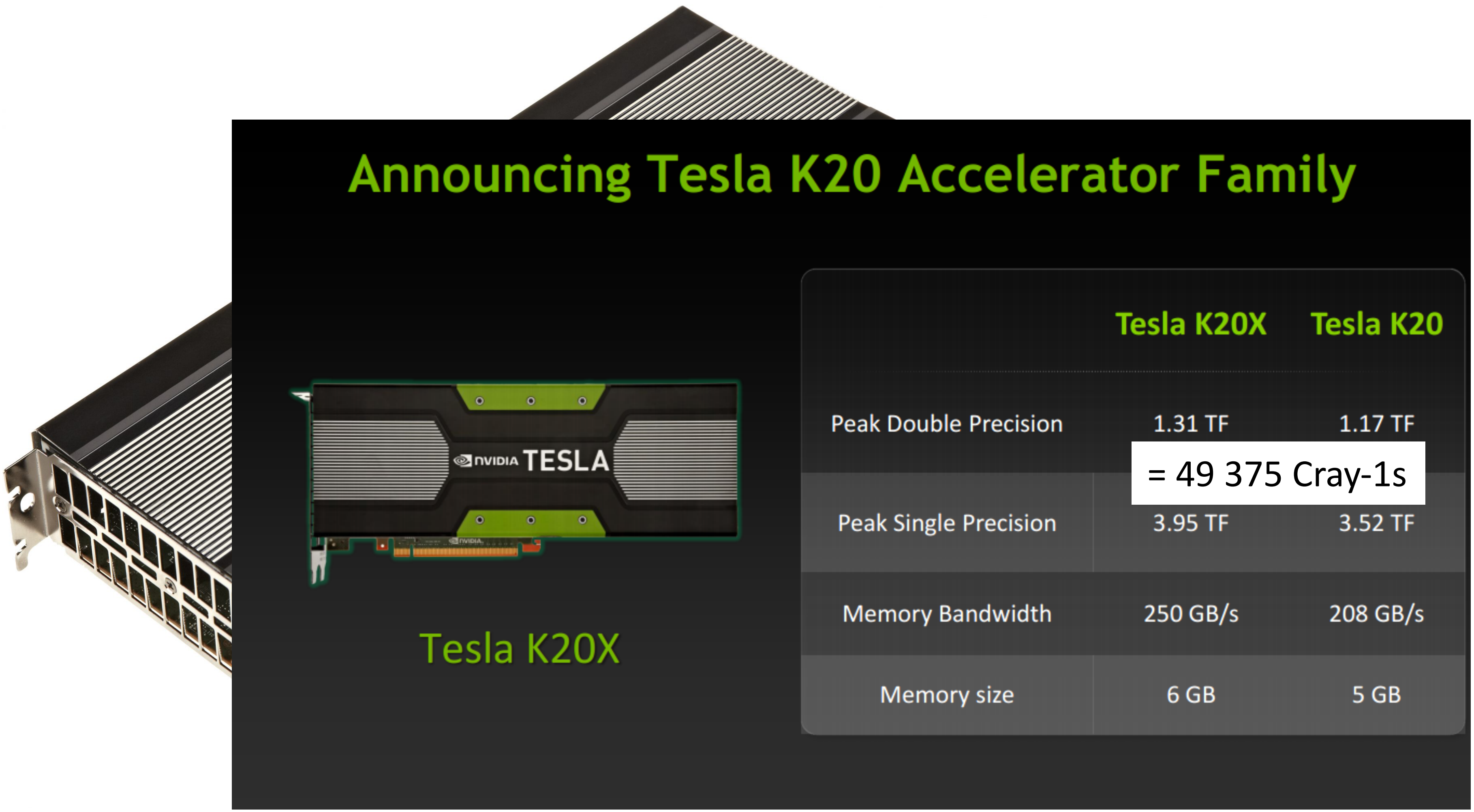
Jason Huggins @hugs · Dec 17, 2014

Cray-1: 80 MFLOPS
Cray-2: 1.9 GFLOPS
iPhone 6: 115.2 GFLOPS

Sources:
en.wikipedia.org/wiki/Cray-1
en.wikipedia.org/wiki/Cray-2
en.wikipedia.org/wiki/Apple_sys...

3 6 7

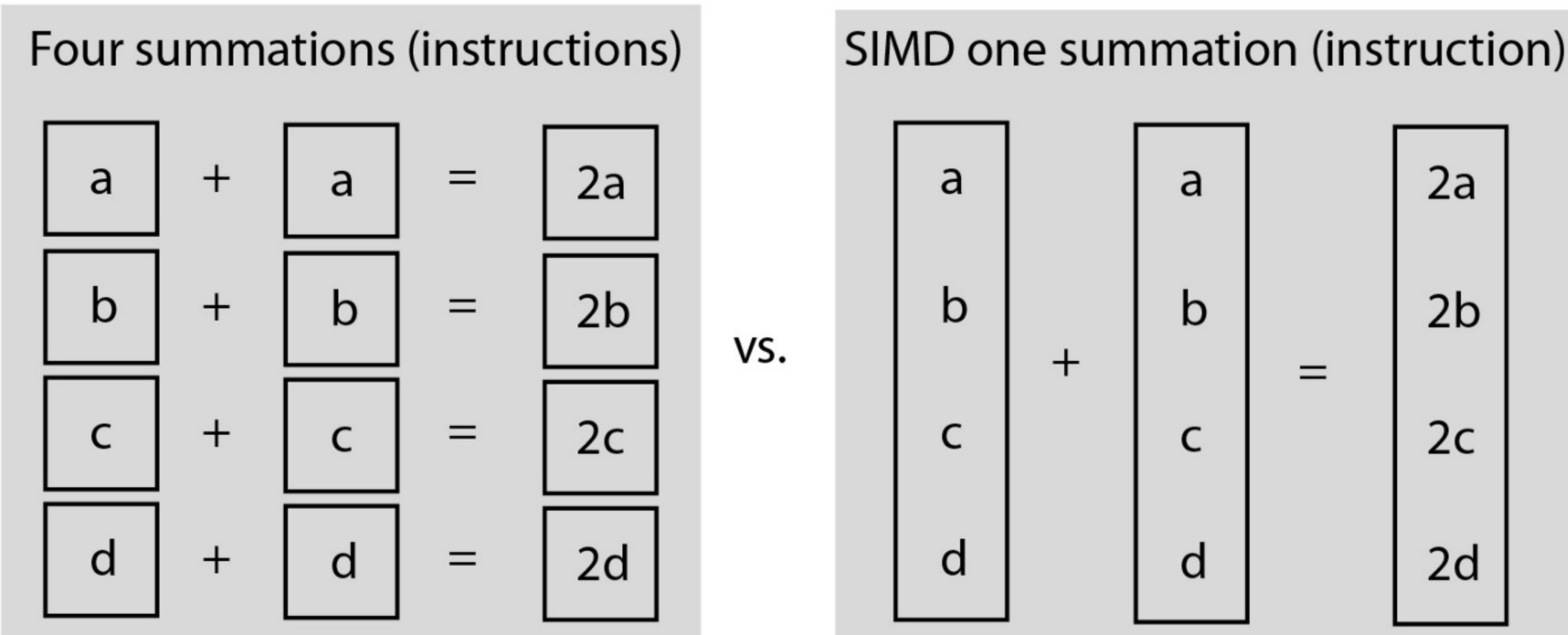
Vector Machines Today



Software Taxonomies

- Data Parallel (SIMD)
  - Parallelism that is a result of identical operations being applied concurrently on different data items; e.g., many matrix algorithms
  - Difficult to apply to complex problems
- Single Program, Multiple Data (SPMD)
  - A single application is run across multiple processes/threads on a MIMD architecture
  - Most processes execute the same code but do not work in lock-step
  - Dominant form of parallel programming

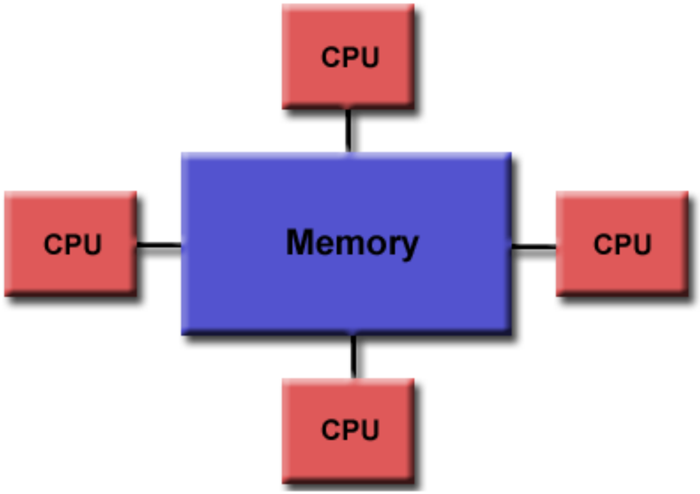
SISD vs. SIMD



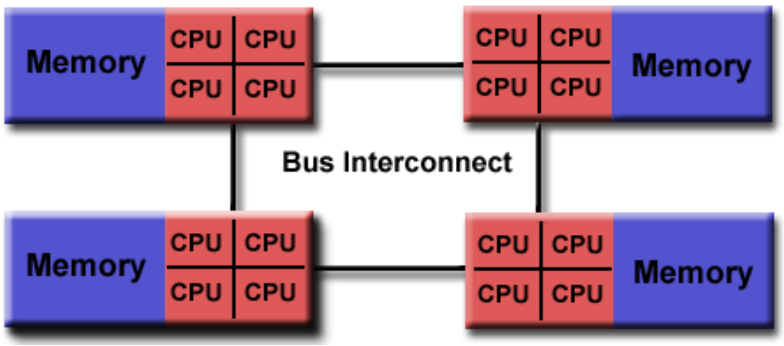
Four summations (instructions)
a
+
a
=
2a
b
+
b
=
2b
c
+
c
=
2c
d
+
d
=
2d
vs.
SIMD one summation (instruction)
a
b
c
d
+
a
b
c
d
=
2a
2b
2c
2d

MIMD Architectures (Shared Memory)

Uniform Memory Access (UMA)

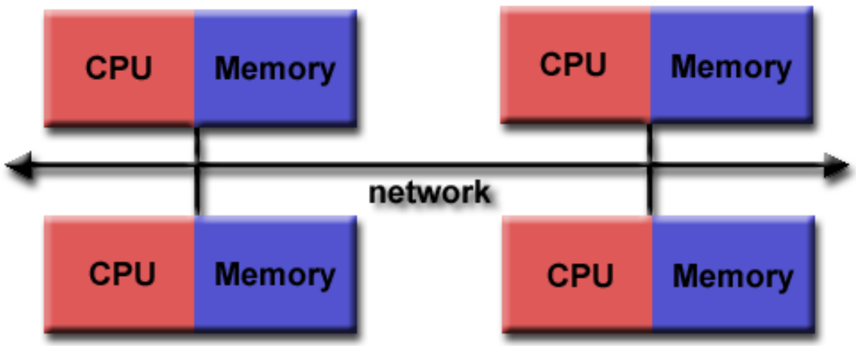


Non-Uniform Memory Access (NUMA)

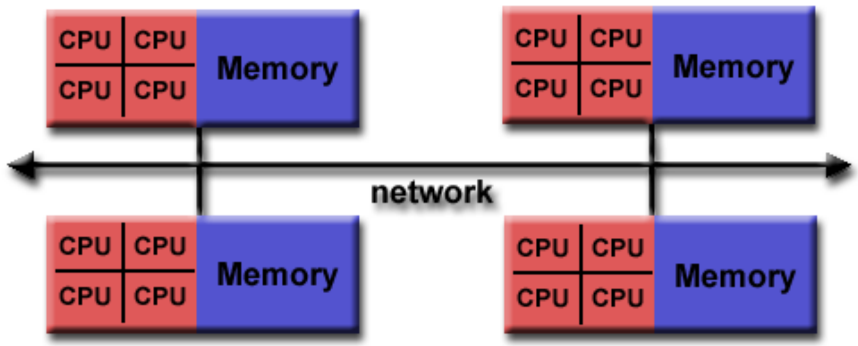


More MIMD Architectures

Distributed Memory



Hybrid Memory



Shared Memory (SM)

- Attributes:
  - Global memory space
  - Each processor will utilize its own cache for a portion of global memory; consistency of the cache is maintained by hardware
- Advantages:
  - User-friendly programming techniques (OpenMP and OpenACC)
  - Low latency for data sharing between tasks
- Disadvantages:
  - Global memory space-to-CPU path may be a bottleneck
  - Non-Uniform Memory Access
  - Programmer responsible for synchronization

Distributed Memory (DM)

- Attributes:
  - Memory is shared amongst processors through message passing
- Advantages:
  - Memory scales based on the number of processors
  - Access to a processor's own memory is fast
  - Cost effective
- Disadvantages:
  - Error prone; programmers are responsible for the details of the communication
  - Complex data structures may be difficult to distribute

Hardware/Software Models

- **Software and hardware models do not need to match**
- DM software on SM hardware:
  - Message Passing Interface (MPI) - designed for DM Hardware but available on SM systems
- SM software on DM hardware
  - Remote Memory Access (RMA) included within MPI-3
  - Partitioned Global Address Space (PGAS) languages
    - Unified Parallel C (extension to ISO C 99)
    - Coarray Fortran (Fortran 2008)

Difficulties

- Serialization causes bottlenecks
- Workload is not distributed
- Debugging is hard
- Serial approach doesn't parallelize

The END