PROJECT CHRONO

Parallel Computing Aspects

Relevant in the Context of Chrono/Computational Dynamics

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

UNIVERSITÀ DEGLI
STUDI DI PARMA

Discussion Points

- Computing speed issues, general remarks

- GPU computing and relevance to Chrono

- Multi-core CPU computing and relevance to Chrono

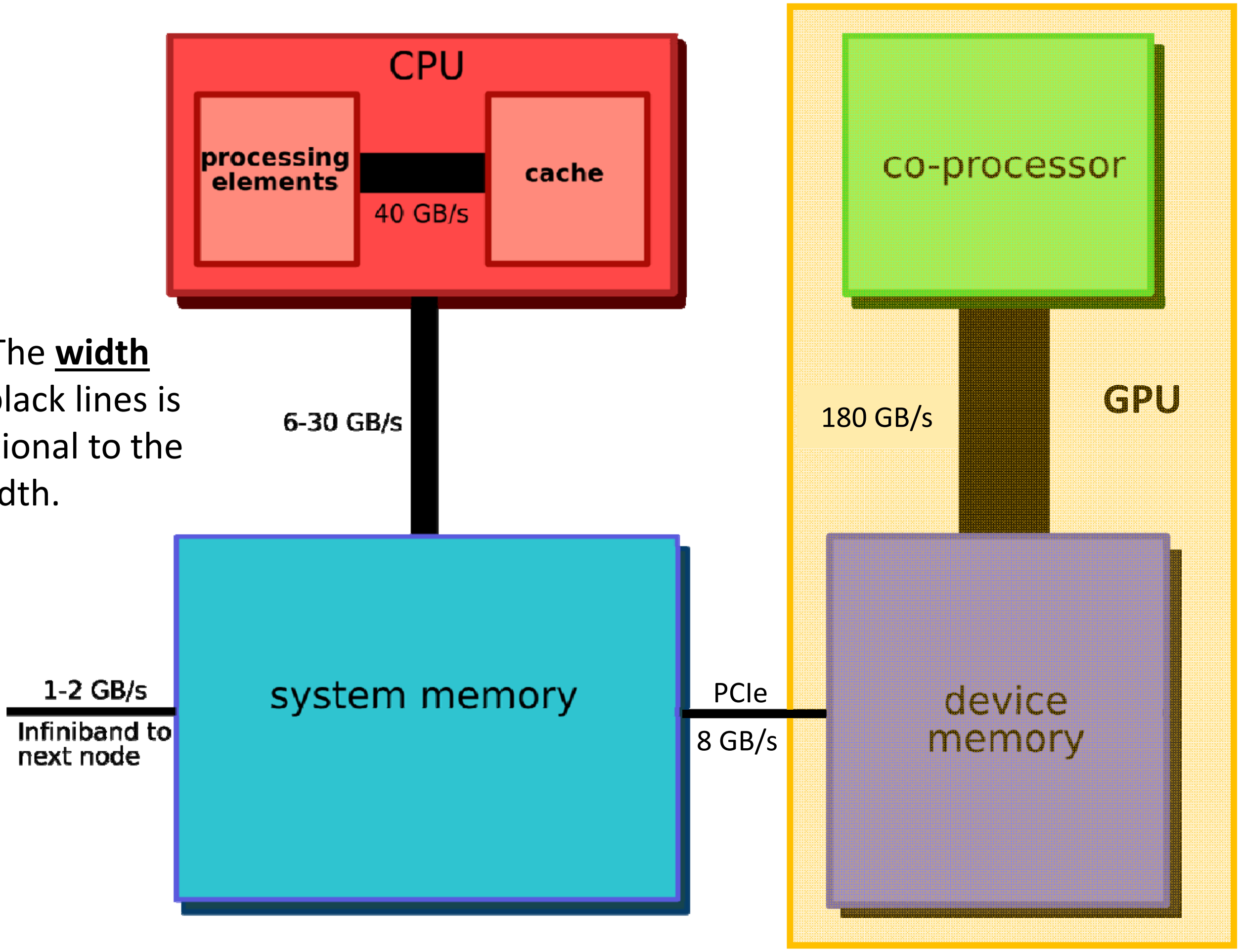
- Distributed-memory computing and relevance to Chrono

The Price of 1 Gflop/second

- How much would you have to pay to perform 1,000,000,000 operations in one second?
  - 1961:
    - Combine 17 million IBM-1620 computers
    - At $64K apiece, when adjusted for inflation, this would cost $8.3 trillion
  - 2000:
    - About $1,000
  - 2015 (January):
    - 8 cents

[wikipedia]→

Commodity CPU-GPU System

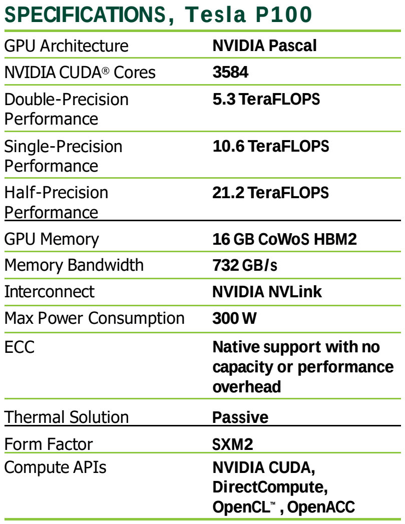


NOTE: The **width** of the black lines is proportional to the bandwidth.

Beyond “commodity”: What’s a Powerful Workstation Today?

- Price tag: $50K

- IBM Power S822LC
  - Two Power8 CPUs
  - Four to eight Pascal P100 GPUs
  - 0.5 TB of system memory
  - NVLink (instead of PCIe bus)



SPECIFICATIONS, Tesla P100

GPU Architecture	**NVIDIA Pascal**
NVIDIA CUDA® Cores	**3584**
Double-Precision Performance	**5.3 TeraFLOPS**
Single-Precision Performance	**10.6 TeraFLOPS**
Half-Precision Performance	**21.2 TeraFLOPS**
GPU Memory	**16 GB CoWoS HBM2**
Memory Bandwidth	**732 GB/s**
Interconnect	**NVIDIA NVLink**
Max Power Consumption	**300 W**
ECC	**Native support with no capacity or performance overhead**
Thermal Solution	**Passive**
Form Factor	**SXM2**
Compute APIs	**NVIDIA CUDA, DirectCompute, OpenCL™ , OpenACC**

Today’s Workstation Is Yesterday’s Supercomputer

- Today one can get about 25-45 TFlops out of a powerful workstation
  - 25 thousands billion operations per second

- 2006: Sandia National Lab Supercomputer
  - 43.5 TFlops
  - Hardware provider: Cray
  - 9th fastest supercomputer in the world
  - Required of the order of tens of KW of power

[http://images.nvidia.com/content/tesla/pdf/nvidia-tesla-p100-datasheet.pdf]→

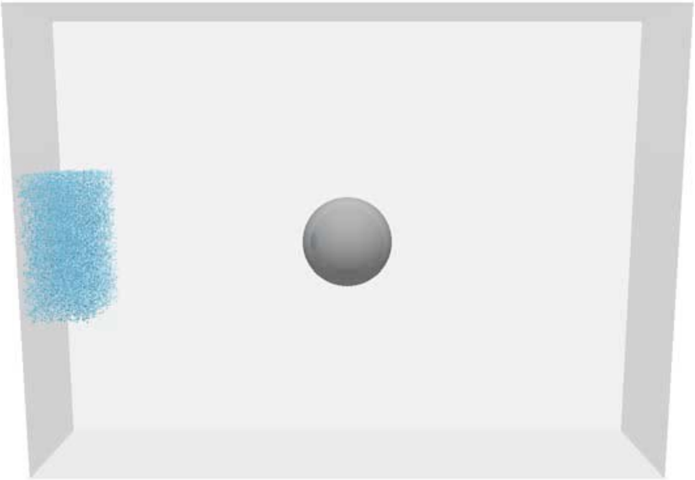
Computational Dynamics: Putting Things in Perspective

- We can count on lots of compute power

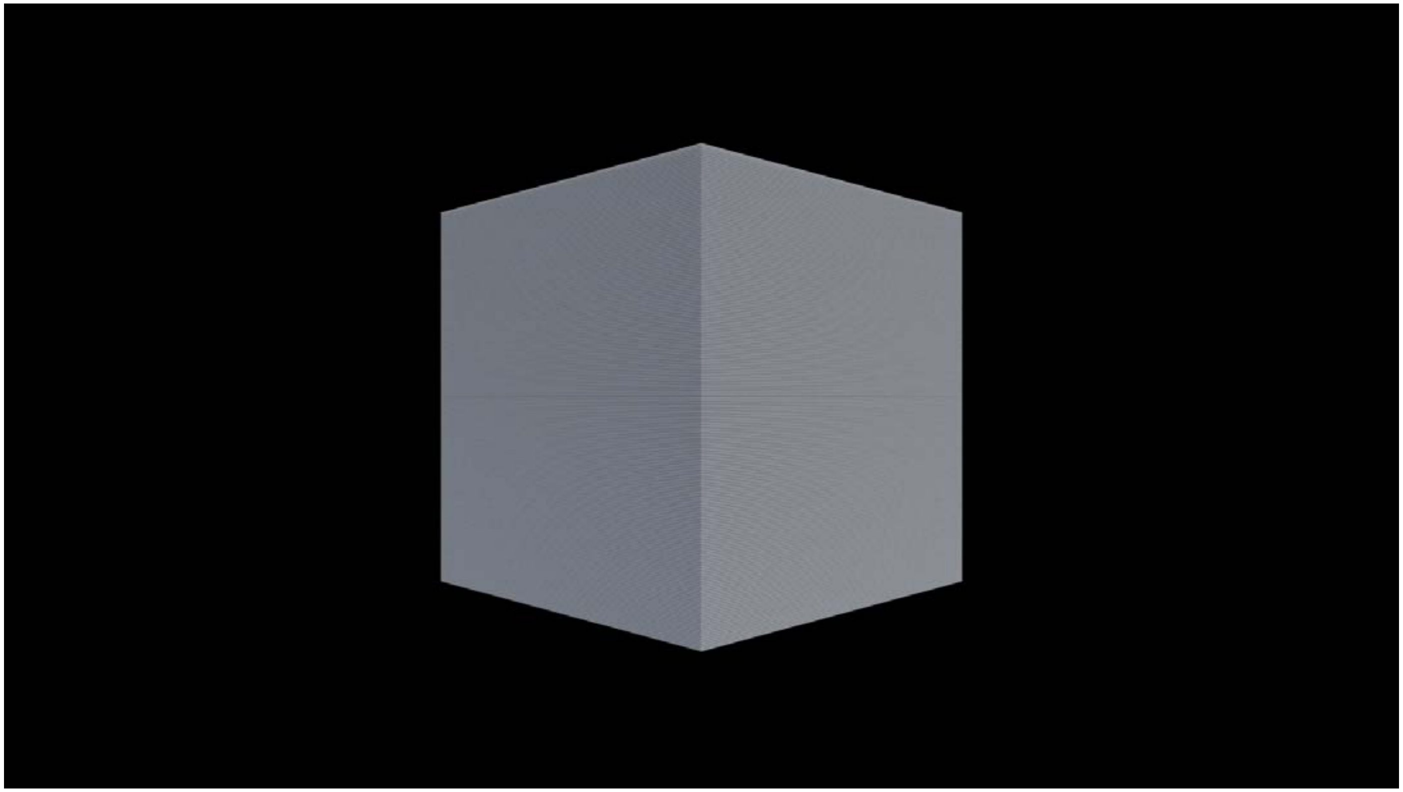
  - Bottom line
    - More accurate models
    - Bridging of space and/or time multiple scales

  - Emerging hardware takes us half way there. Also need
    - Modeling methods + suitable solution algorithms
    - Targeted software, platform specific

Progress over Time
Year: 2009 – 1 million bodies [GPU Granular Dynamics]



Year: 2015 – 10 million bodies



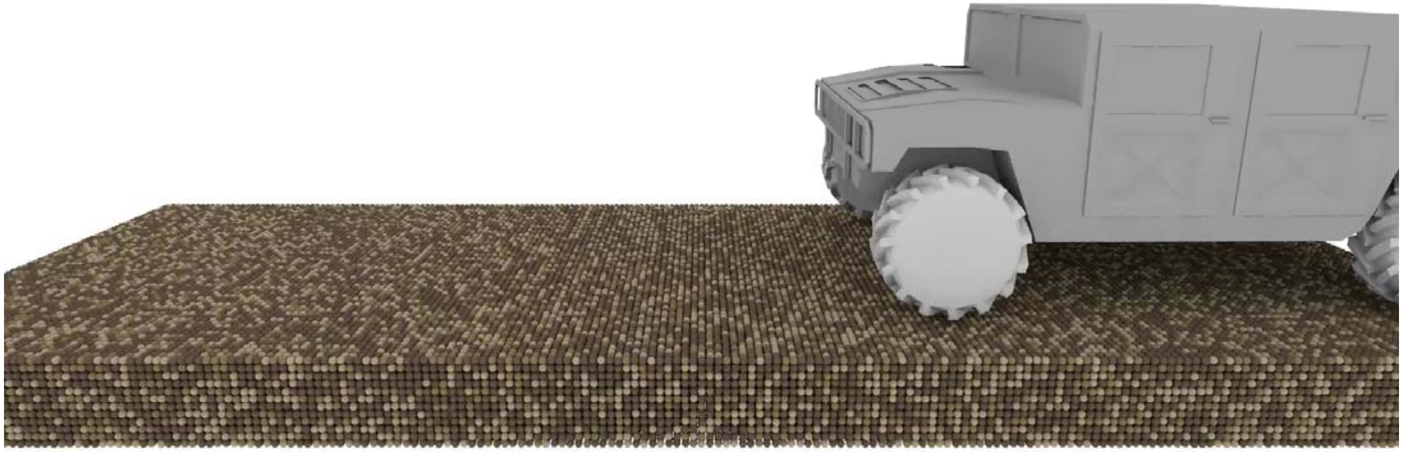
GPU Tank Wave Simulation

- 2009 Implementation
  - 1 million rigid frictionless spheres
  - Integration time step: 0.01s
  - 24 sec per step
  - Hardware used: GPU
    - NVIDIA Tesla C1060
  - Velocity based complementarity
  - 20 seconds long simulation
  - Simulation Time: ~2 days

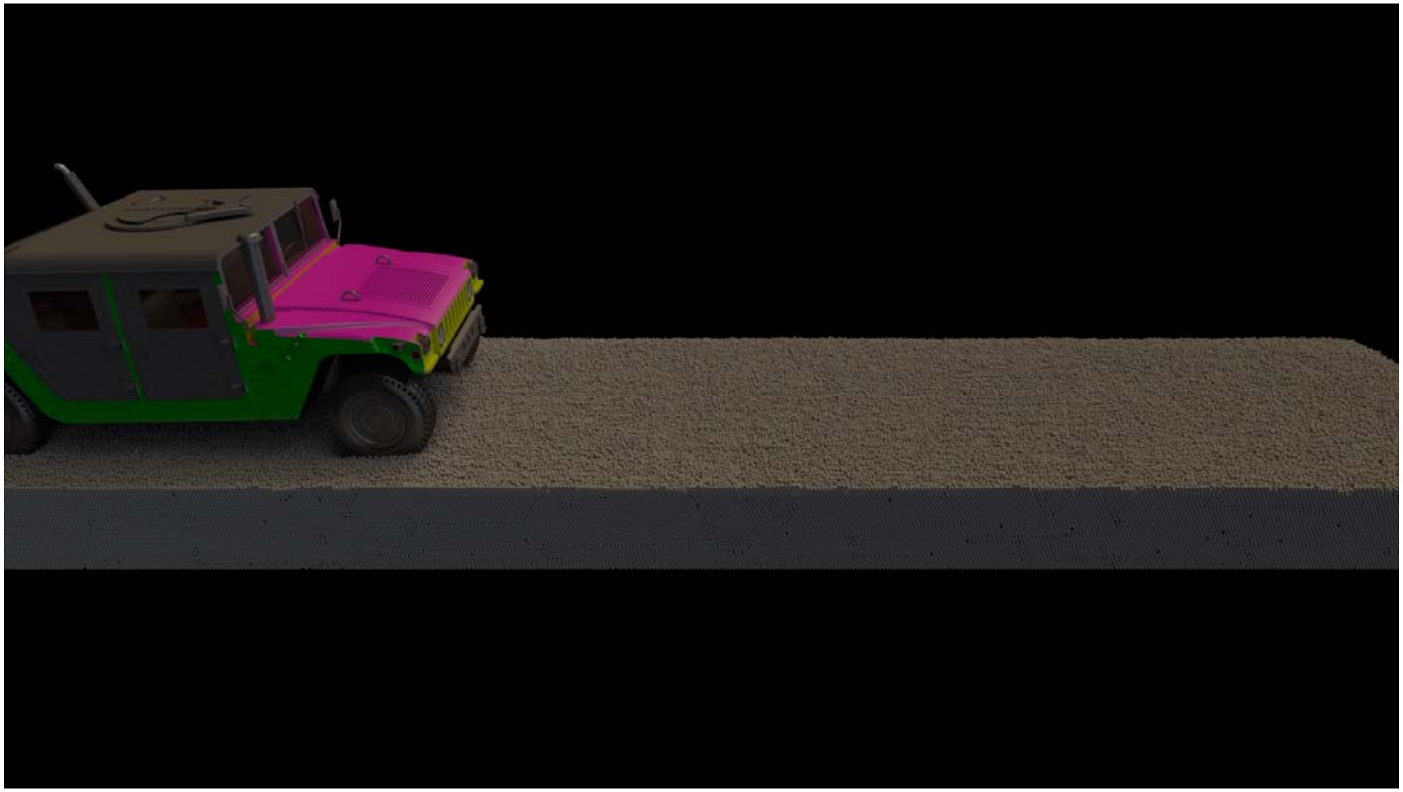
- 2015 Implementation
  - 10,648,000 rigid spheres
  - Integration time step: 0.00025s
  - 1 second per step
  - Hardware Use: GPU
    - NVIDIA Tesla K40x
  - Position based complementarity
  - 10 seconds long simulation
  - Simulation Time: 11 hours

2015 Simulation: given the number of bodies, about 20 times faster

Vehicle on Deformable Terrain. Year: 2012



Vehicle on Deformable Terrain. Year: 2015



Vehicle on Deformable Terrain

- 2012
  - 300k rigid spheres
  - Length of simulation: 15 seconds
  - Hardware used: CPU (Intel)
    - Multicore, based on OpenMP
  - Integration time step: 0.001s
  - Velocity Based Complementarity
  - 17 seconds per time step
  - Simulation time: ~2.5 days

- 2015
  - ~1.5 million rigid spheres
  - Length of simulation: 15 seconds
  - Hardware: GPU (NVIDIA)
    - Tesla K40X
  - Integration time step: 0.0005s
  - Position Based Dynamics
  - 0.3 seconds per time step
  - Simulation time: ~2.5 hours

2015 Simulation: although 5X more bodies, runs about 25 times faster

Parallel Computing on the GPU

- NVIDIA GPU Computing Architecture
  - Via a separate HW interface
  - In laptops, desktops, workstations, servers

Pascal P100

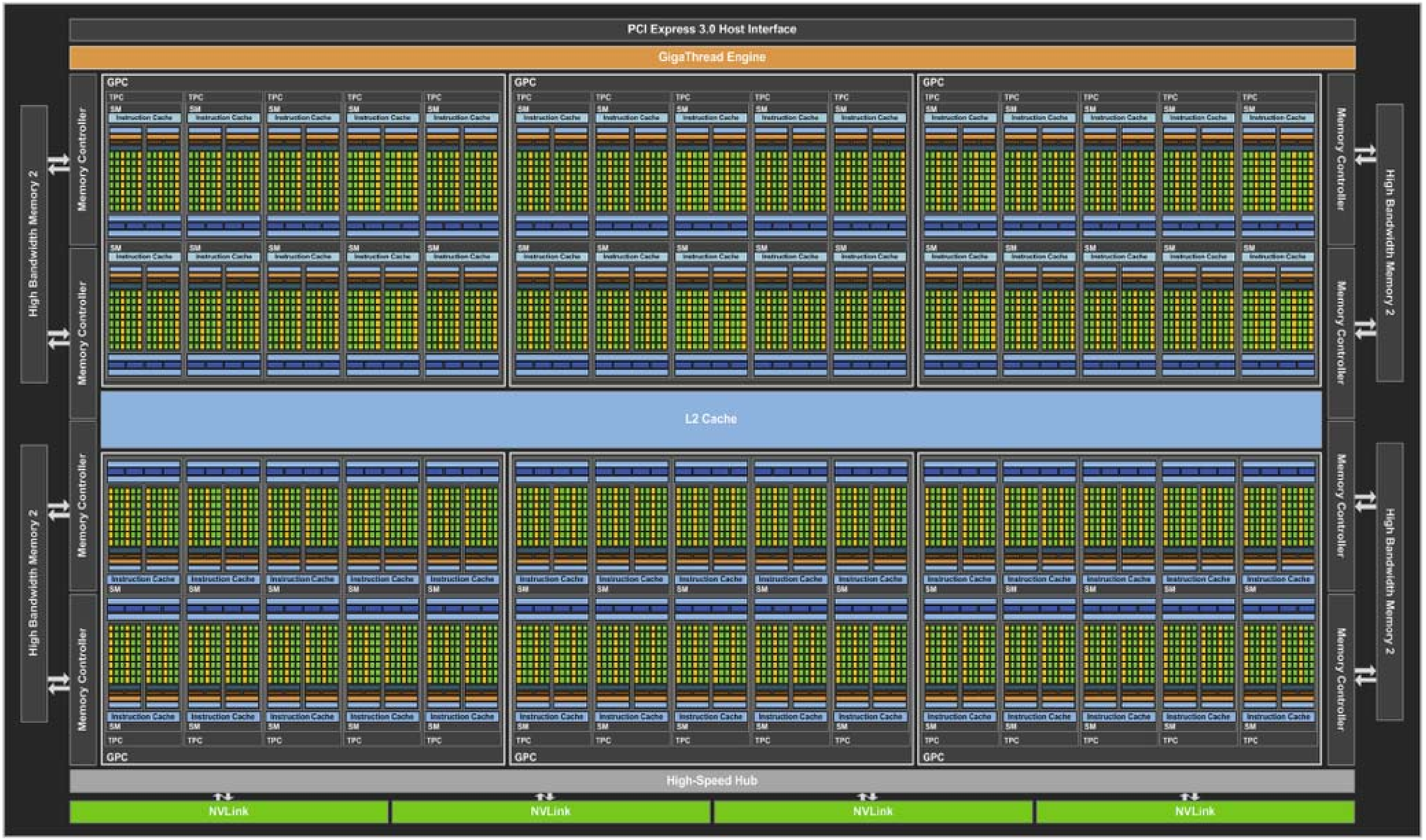
- Multithreaded SIMT model uses application data parallelism and thread parallelism

Kepler K20X

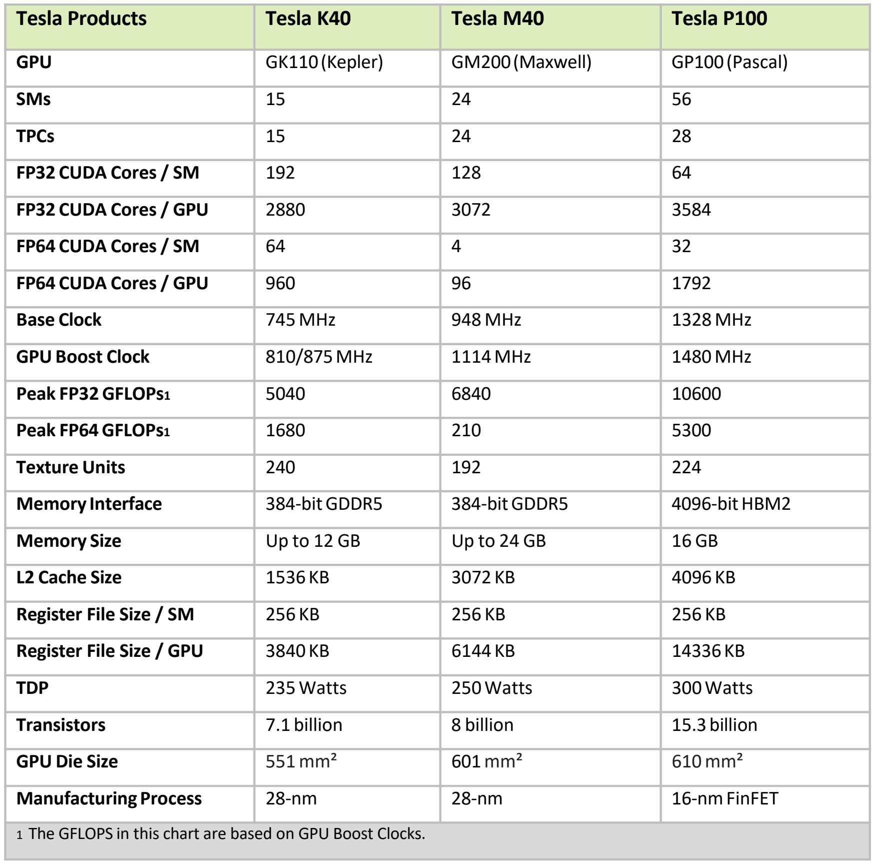
L x W x H: 18 x 8 x 12 inches

- Programmable in C with CUDA tools
  - “Extended C”

Latest NVIDIA GPU Architecture [Tesla P100]



Moore’s Law At Work



Tesla Products	Tesla K40	Tesla M40	Tesla P100
GPU	GK110 (Kepler)	GM200 (Maxwell)	GP100 (Pascal)
SMs	15	24	56
TPCs	15	24	28
FP32 CUDA Cores / SM	192	128	64
FP32 CUDA Cores / GPU	2880	3072	3584
FP64 CUDA Cores / SM	64	4	32
FP64 CUDA Cores / GPU	960	96	1792
Base Clock	745 MHz	948 MHz	1328 MHz
GPU Boost Clock	810/875 MHz	1114 MHz	1480 MHz
Peak FP32 GFLOPs[1]	5040	6840	10600
Peak FP64 GFLOPs[1]	1680	210	5300
Texture Units	240	192	224
Memory Interface	384-bit GDDR5	384-bit GDDR5	4096-bit HBM2
Memory Size	Up to 12 GB	Up to 24 GB	16 GB
L2 Cache Size	1536 KB	3072 KB	4096 KB
Register File Size / SM	256 KB	256 KB	256 KB
Register File Size / GPU	3840 KB	6144 KB	14336 KB
TDP	235 Watts	250 Watts	300 Watts
Transistors	7.1 billion	8 billion	15.3 billion
GPU Die Size	551 mm^2	601 mm^2	610 mm^2
Manufacturing Process	28-nm	28-nm	16-nm FinFET

[1] The GFLOPS in this chart are based on GPU Boost Clocks.

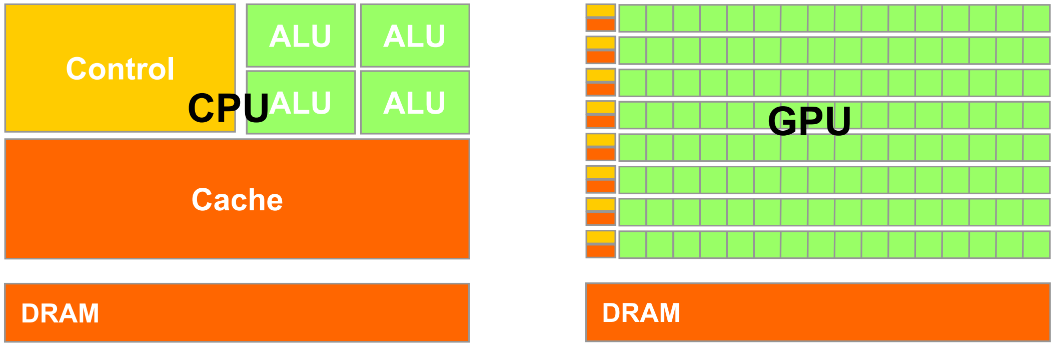
Data vs. Task Parallelism

- Data parallelism
  - You have a large amount of data elements and each data element needs to be processed to produce a result
  - When this processing can be done in parallel, we have data parallelism
  - Example:
    - Filtering a picture; i.e., processing each pixel out of a million in a snapshot

- Task parallelism
  - You have a collection of tasks that need to be completed
  - If these tasks can be performed in parallel you are faced with a task parallel job
  - Examples:
    - Microwave a soup, make a salad, boil pasta, bake a cake
      - All of the above can happen at the same time

Why is the GPU Fast?

- The GPU is specialized for compute-intensive, highly data parallel computation (owing to its graphics rendering origin)
  - More transistors devoted to data processing rather than data caching and control flow
  - Where are GPUs good: high arithmetic intensity (the ratio between arithmetic operations and memory operations)



- Video gaming industry exerts strong economic pressure that fuels constant innovation in GPU computing

When Are GPUs Good?

- Ideally suited for data-parallel computing (SIMD)

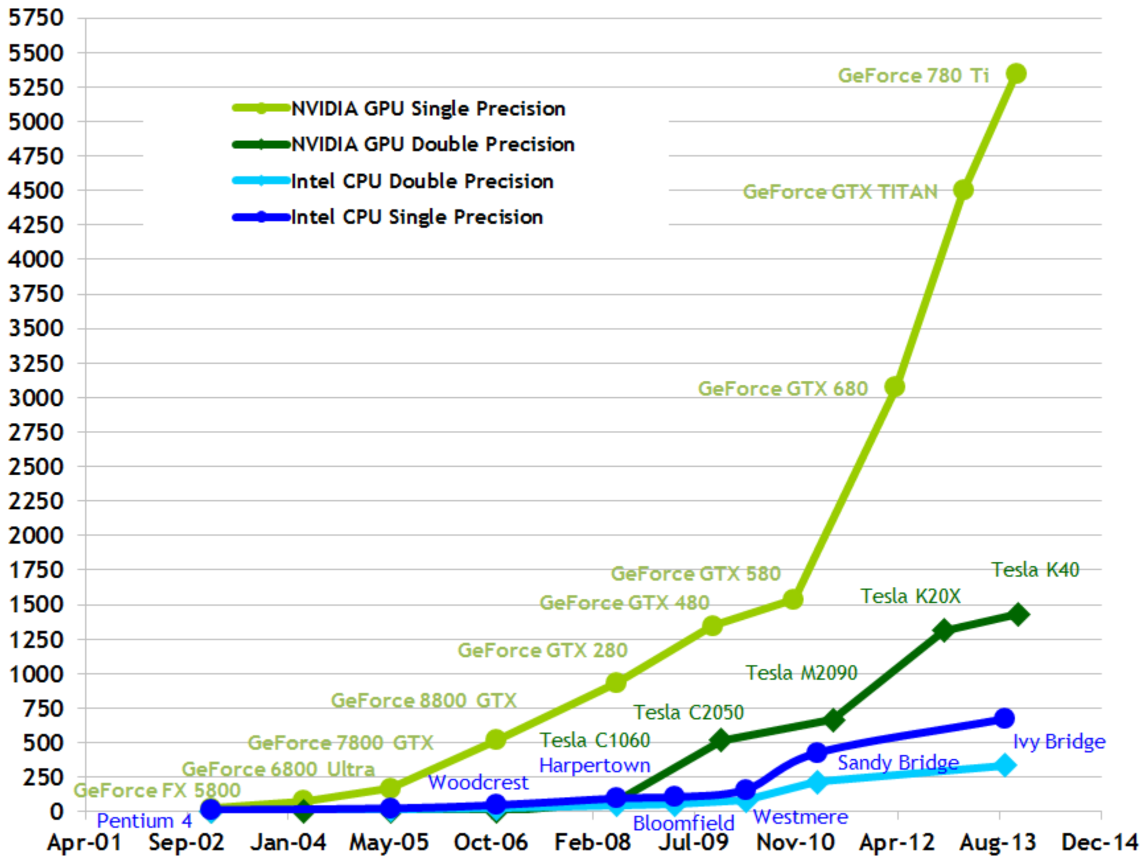
- Moreover, you want to have high arithmetic intensity
  - Arithmetic intensity: ratio or arithmetic operations to memory operations

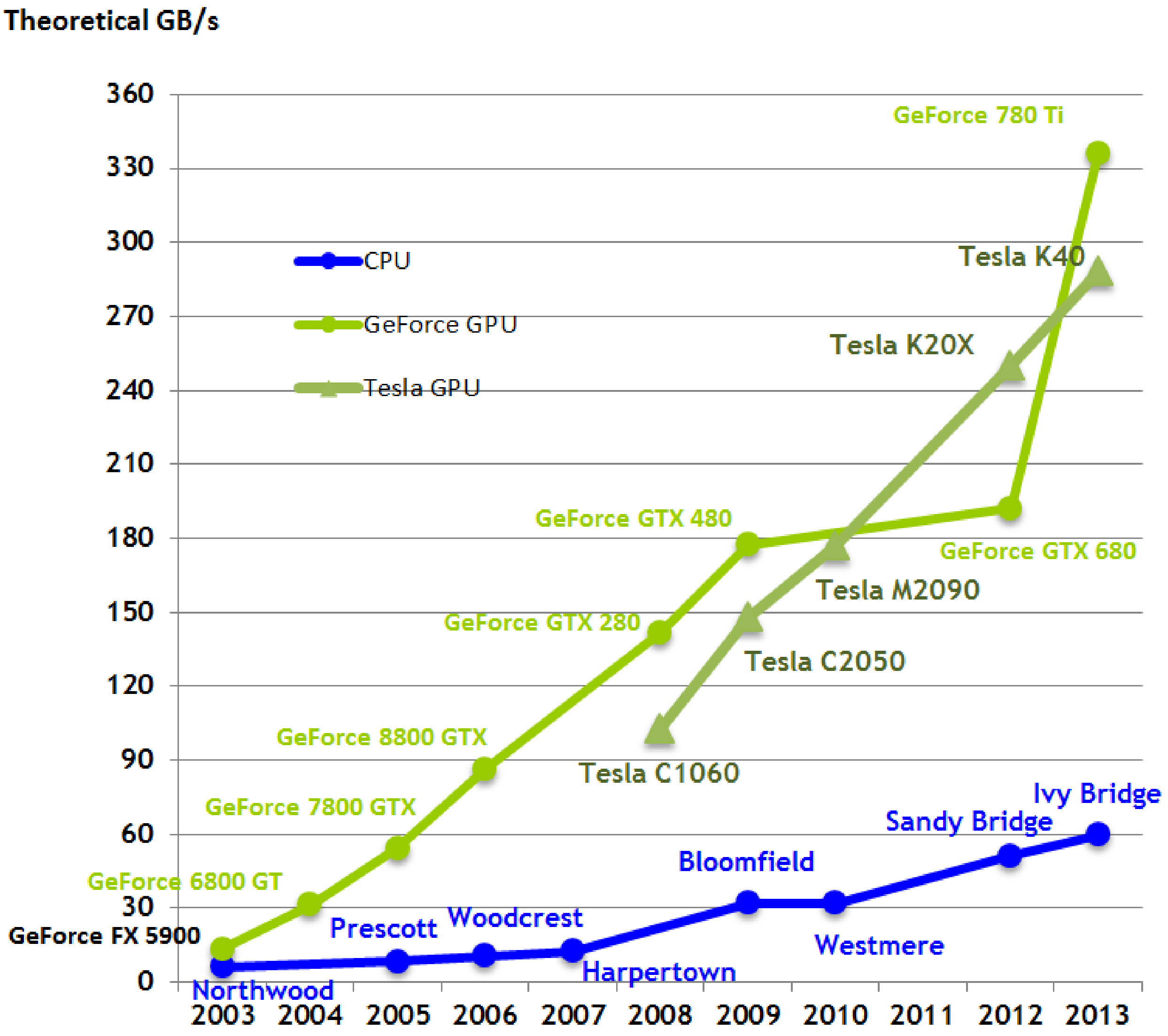
- You are off to a good start with GPU computing if you can do this…
  - GET THE DATA ON THE GPU AND KEEP IT THERE
  - GIVE THE GPU ENOUGH WORK TO DO
  - FOCUS ON DATA REUSE WITHIN THE GPU TO AVOID MEMORY BANDWIDTH LIMITATIONS

Very Important Point

- Almost 100% of our applications bound by memory speed
  - Most often, the cores (or SPs) idle waiting for data
  - What's important is how fast you can move data
    - You want high memory bandwidth

Theoretical GFLOP/s




Theoretical GB/s
360
330
300
270
240
210
180
150
120
90
60
30
0
CPU
GeForce GPU
Tesla GPU
GeForce 780 Ti
Tesla K40
Tesla K20X
GeForce GTX 480
GeForce GTX 680
Tesla M2090
GeForce GTX 280
Tesla C2050
GeForce 8800 GTX
Tesla C1060
GeForce 7800 GTX
Ivy Bridge
Sandy Bridge
Bloomfield
GeForce 6800 GT
GeForce FX 5900
Prescott
Woodcrest
Westmere
Harpertown
Northwood
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013

Memory Speed:

Widening of the Processor-DRAM Performance Gap

- The processor: So fast it left the memory behind
  - A system (CPU-Memory duo) can’t move as fast as you’d like (based on CPU top speeds) with a sluggish memory

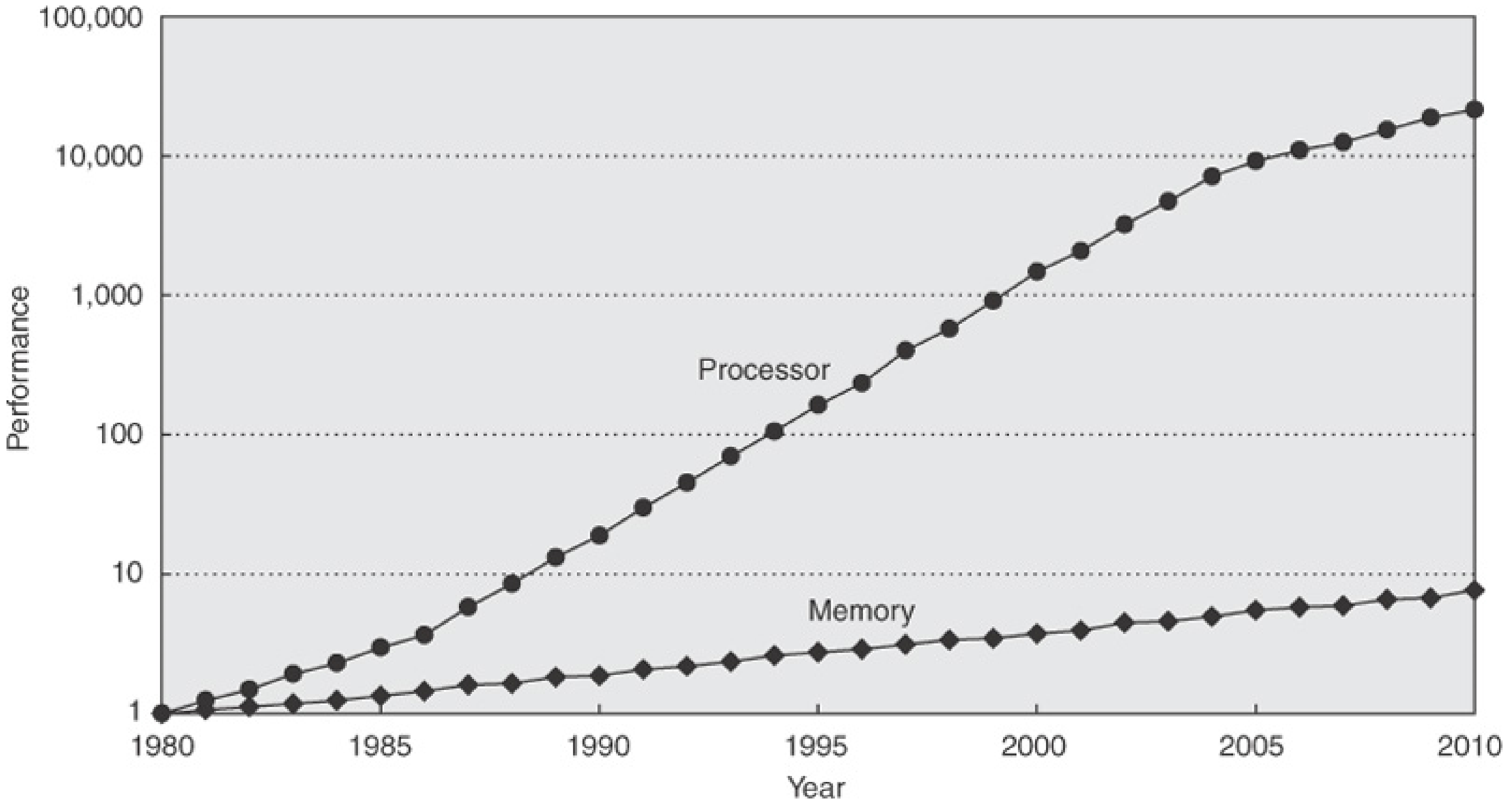
- Plot on next slide shows on a *log* scale the increasing gap between CPU and memory

- The memory baseline: 64 KB DRAM in 1980

- Memory speed increasing at a rate of approx 1.07/year
  - However, processors improved
    - 1.25/year (1980-1986)
    - 1.52/year (1986-2004)
    - 1.20/year (2004-2010)

Memory Speed:

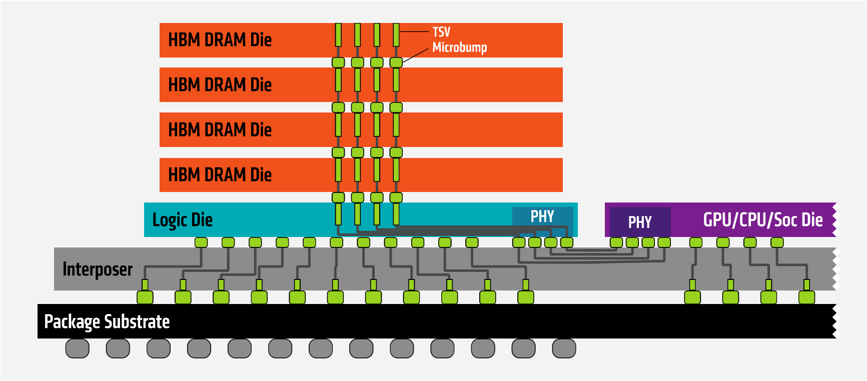
Widening of the Processor-DRAM Performance Gap



Courtesy of Elsevier, Computer Architecture, Hennessey and Patterson, fourth edition

3D Stacked Memory
[immediate future looks bright]

- SK Hynix's High Bandwidth Memory (HBM)
  - Developed by AMD and SK Hynix

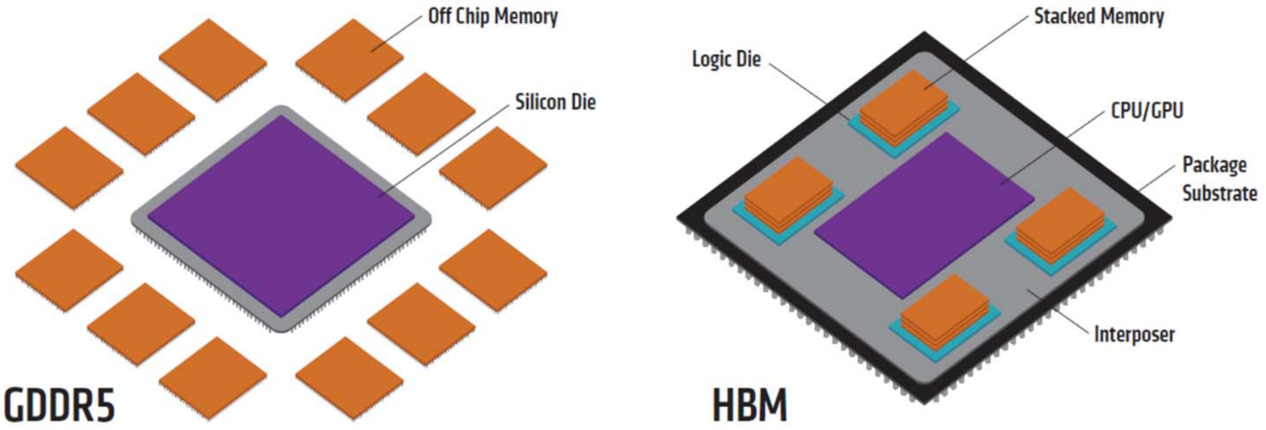


- 1st Generation (HBM1) introduced in AMD Fiji GPUs
  - 1GB & 128GB/s per stack
  - AMD Radeon R9 Fury X: had four stacks → 4GB & 512GB/s

- 2nd Generation (HBM2) will be used in NVIDIA Pascal and AMD Arctic Island GPUs
  - 2 GB & 256GB/s bandwidth per stack
  - NVIDIA Pascal: close to 1TB/s memory bandwidth

3D Stacked Memory

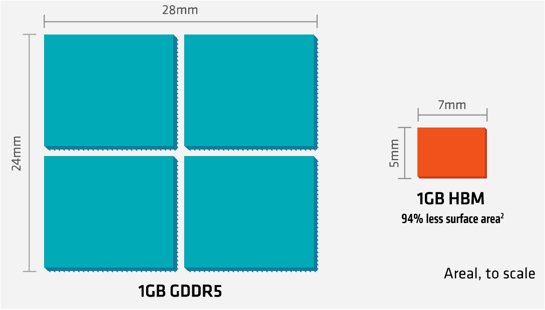
[immediate future looks bright]

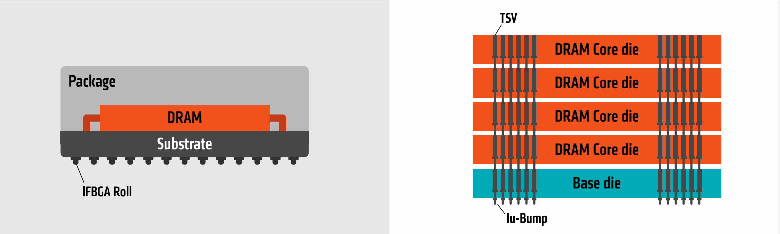


This is a big deal.

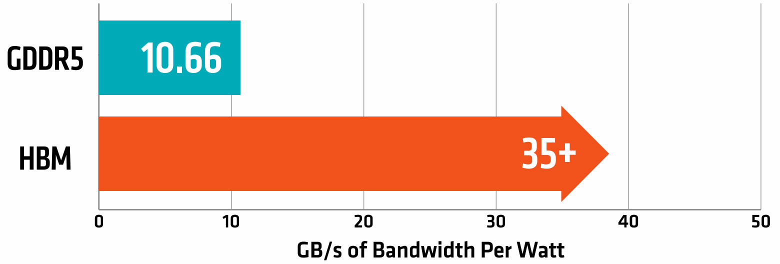
3D Stacked Memory

- More power efficient
- Shorter distances
  - Less power wasted moving data
- Smaller memory footprint
  - More memory can be packed into space



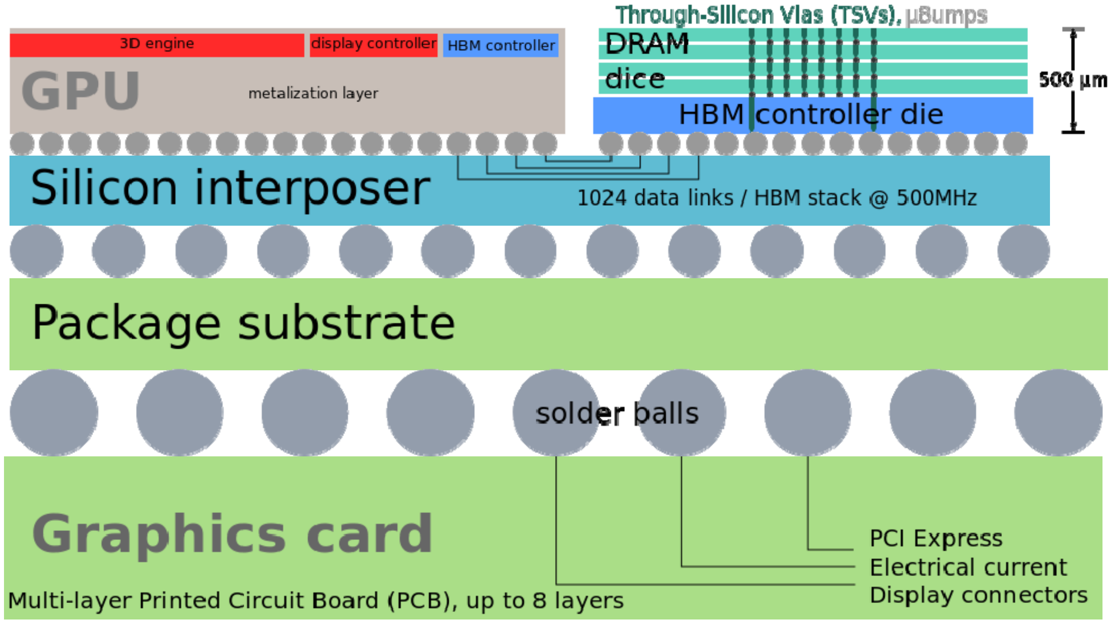


GDDR5	Per Package	HBM
32-bit	Bus Width	1024-bit
Up to 1750MHz (7GBps)	Clock Speed	Up to 500MHz (1GBps)
Up to 28GB/s per chip	Bandwidth	>100GB/s per stack
1.5V	Voltage	1.3V



GPU w/ HBM

[cut-through]



The Advanced Computing Landscape

Supercomputing (distributed memory)	🟩	Group of nodes communicate through fast interconnect	🟩	MPI, Charm++, Chapel
Multi-socket Parallel Computing	🟩	Group of CPUs operate together on the same node	🟩	OpenMP, MPI
Acceleration (GPU/Phi)	🟩	Compute devices accelerating parallel computation on one node	🟩	CUDA, OpenCL + OpenMP, MPI
Multi-Core Parallel Computing	🟩	Communication through shared caches	🟩	OpenMP, TBB, pthreads
Vectorization	🟩	Higher operation throughput via special/fat registers	🟩	AVX, SSE
Pipelining	🟨	Sequence of instruction sharing functional units	🟨	Assembly
Superscalar	🟨	Non-sequence instructions sharing functional units	🟨	Assembly

We have full control → 🟩

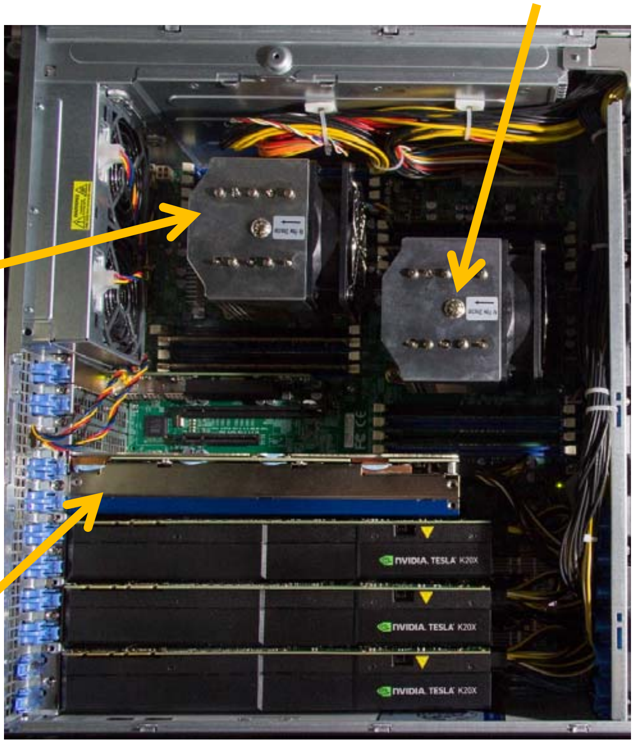
We have little to no control → 🟨

New Topic:
Looking beyond GPU computing

- Much more to parallel computing than GPU acceleration

- Intel: shared memory parallelism
  - “Parallel multicore” computing

- Where are the multiple cores?



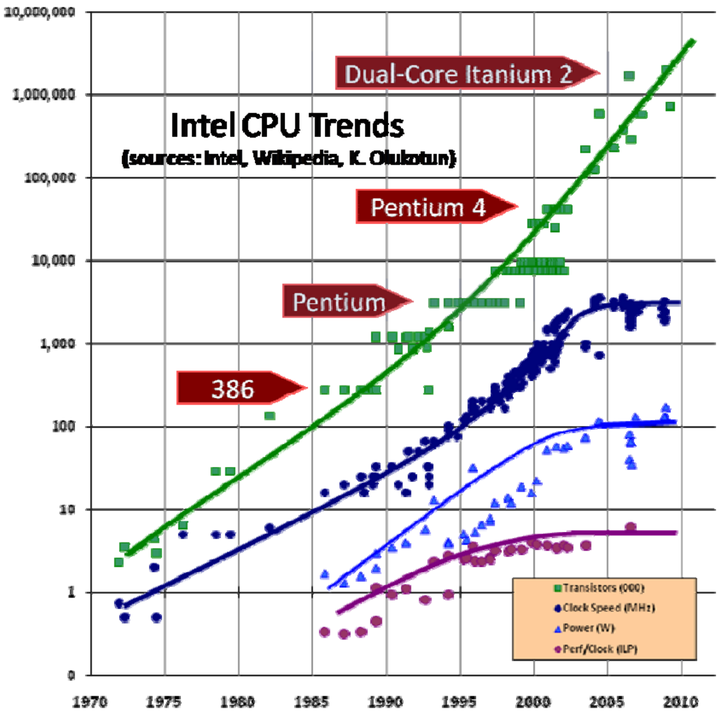
10 cores (20 virtual)
E5-2690 v2 Ivy Bridge-EP
3.0GHz 25MB L3 Cache

10 cores (20 virtual)
E5-2690 v2 Ivy Bridge-EP
3.0GHz 25MB L3 Cache

60 cores
Intel® Xeon Phi™ 5110P (8GB)

Moore’s Law

- Number of transistors per unit area has been steadily going up
- ILP and Clock Speed have stagnated



Transistor Densities Still Going Up For Now

- Consequence: lots of cores in one chip
  - CPU Cores: 18 today, probably **32** by 2018
  - Intel Xeon Phi : 61 today, very likely close to **200** in 2017
- This trend leveraged in Chrono by use of OpenMP

What Is OpenMP?

- Portable, shared-memory threading API
  - Bindings: Fortran, C, and C++
  - Multi-vendor support for both Linux and Windows

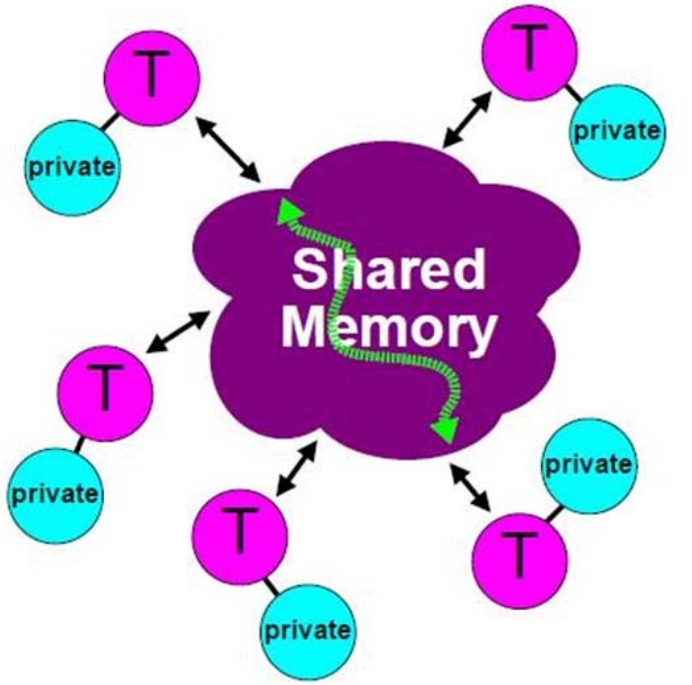
- Standardizes task & loop-level parallelism
- Very good at coarse-grained (task) parallelism
- Combines serial and parallel code in single source
- Standardizes ~ 25 years of compiler-directed threading experience

- Current spec is OpenMP 4.0
  - Released in 2013
  - http://www.openmp.org
  - More than 300 Pages

[IOMPP]→

Shared Memory Multi-Processing [SMMP]

- Threads have access to large pool of shared memory
- Threads can have private data
  - Not accessible by other threads
- Data transfer/access transparent to programmer
- Synchronization is implicit but can be made explicit as wel



[CodeProject]→

OpenMP: Directives-based API

- Most OpenMP constructs are compiler directives or pragmas

  - For C and C++, the pragmas take the form:

    ```
    #pragma omp construct [clause [clause]…]
    ```

  - For Fortran, the directives take one of the forms:

    ```
    C$OMP construct [clause [clause]…]
    !$OMP construct [clause [clause]…]
    *$OMP construct [clause [clause]…]
    ```

[IOMPP]→

Work Sharing

- **Work sharing**: term used in OpenMP to describe distribution of work across threads

- Three primary avenues for work sharing in OpenMP:
  - `omp for` construct
  - `omp sections` construct
  - `omp task` construct

Each of them automatically divides work among threads

Intel Roadmap: The Full Story

- 2013 – 22 nm Tick: Ivy Bridge – Tock: Haswell
- 2015 – 14 nm Tick: Broadwell – Tock: Skylake
- 2016 – 14nm "Refresh" Kabylake
- 2017 – 10 nm Tick: Cannonlake (delayed to 2nd Half 2017)
- 2019 – 7 nm
- 2021 – 5 nm
- 2023 – ??? (carbon nanotubes?)

- Happening now: Moore's law moving from 18-24 month cycle to 24-30 month cycle for the first time in 50 years
- In 5 years, end of Moore's law: can't increase transistor density on unit area after 2021

Why's OpenMP Faltering?

- Cache Coherence
  - Many cores see the same memory space and write all over the place
  - Memory manager unit (MMU): the watchdog making sure Thread X running on Core X does not work with data that is stale (since modified by Thread Y running on Core Y)

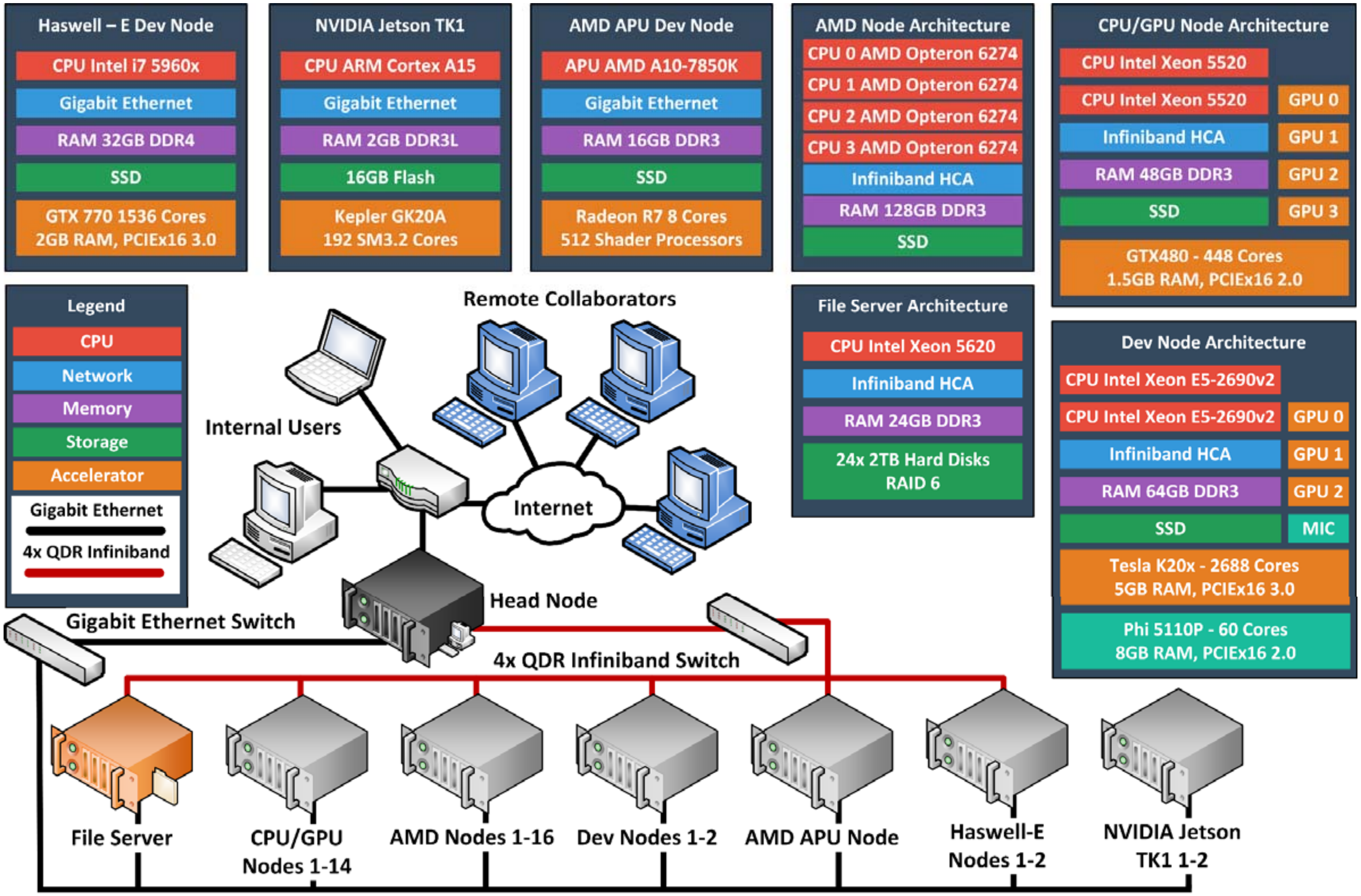
- NUMA: Non-uniform memory access
  - Some memory banks closer to some cores than to some others
  - Long wait time if a Thread keeps looking for data stored in a far-away memory bank

Scaling, with Lots of Cores: Via Distributed Memory

- To scale, a different parallel programming paradigm needed: distributed memory

- Pros, distributed memory
  - Eliminates cache coherence issues
  - Can attack very large problems – lots of memory available to user

- Cons, distributed memory
  - Calls for major code re-write
  - Very often high data access latencies
    - 1000X higher than accessing memory on a workstation



Haswell – E Dev Node
- CPU Intel i7 5960x
- Gigabit Ethernet
- RAM 32GB DDR4
- SSD
- GTX 770 1536 Cores 2GB RAM, PCIEx16 3.0

NVIDIA Jetson TK1
- CPU ARM Cortex A15
- Gigabit Ethernet
- RAM 2GB DDR3L
- 16GB Flash
- Kepler GK20A 192 SM3.2 Cores

AMD APU Dev Node
- APU AMD A10-7850K
- Gigabit Ethernet
- RAM 16GB DDR3
- SSD
- Radeon R7 8 Cores 512 Shader Processors

AMD Node Architecture
- CPU 0 AMD Opteron 6274
- CPU 1 AMD Opteron 6274
- CPU 2 AMD Opteron 6274
- CPU 3 AMD Opteron 6274
- Infiniband HCA
- RAM 128GB DDR3
- SSD

CPU/GPU Node Architecture
- CPU Intel Xeon 5520
- CPU Intel Xeon 5520 | GPU 0
- Infiniband HCA | GPU 1
- RAM 48GB DDR3 | GPU 2
- SSD | GPU 3
- GTX480 - 448 Cores 1.5GB RAM, PCIEx16 2.0

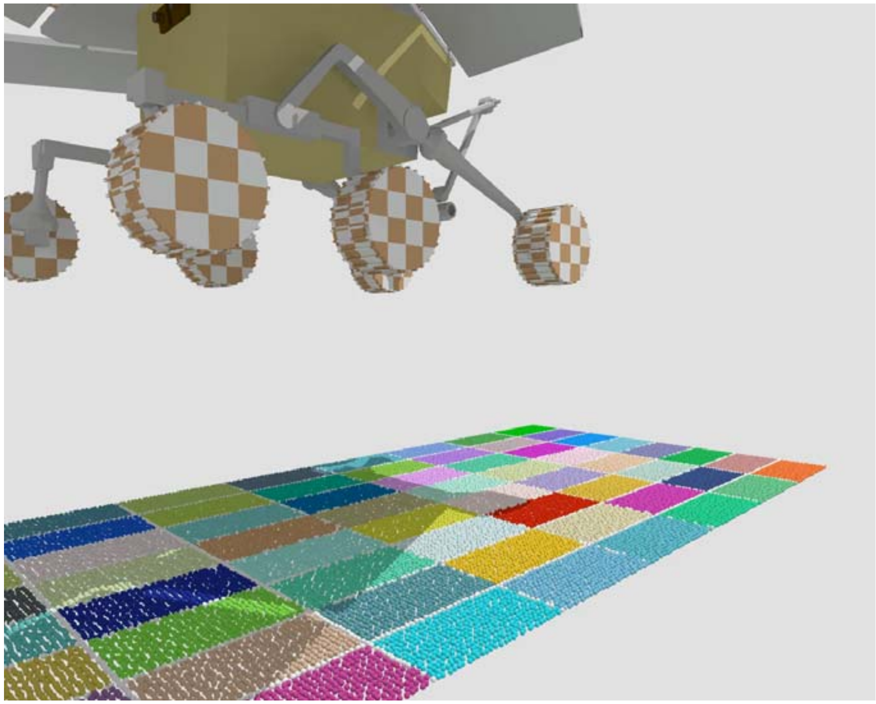
Legend
- CPU
- Network
- Memory
- Storage
- Accelerator
- Gigabit Ethernet
- 4x QDR Infiniband

File Server Architecture
- CPU Intel Xeon 5620
- Infiniband HCA
- RAM 24GB DDR3
- 24x 2TB Hard Disks RAID 6

Dev Node Architecture
- CPU Intel Xeon E5-2690v2
- CPU Intel Xeon E5-2690v2 | GPU 0
- Infiniband HCA | GPU 1
- RAM 64GB DDR3 | GPU 2
- SSD | MIC
- Tesla K20x - 2688 Cores 5GB RAM, PCIEx16 3.0
- Phi 5110P - 60 Cores 8GB RAM, PCIEx16 2.0

EULER – Lab's heterogeneous CPU/GPU Cluster.

Distributed Memory Simulation in Chrono



Distributed Memory, Good for the Long Run Though

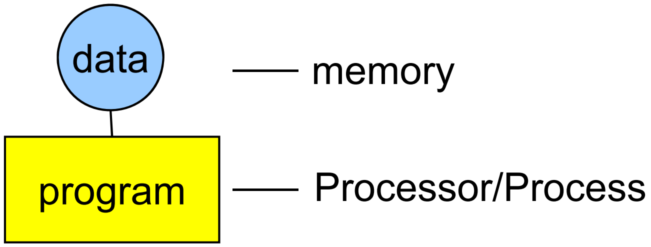
- Five to six years from now, it’s not clear what will replace Moore’s law
- No technology yet to continue the steady increase in core count we have experienced
  - Can’t improve anymore speeds by use of more cores on one workstation
- Distributed memory is the path towards running by drawing on multiple workstations
  - Called “nodes”

Distributed Memory Solution

- Distributed Memory Solution = Supercomputer
- Tianhe, IBM BlueGene, Cray, Linux cluster, etc.
- Almost all of these machines rely on the Message Passing Interface (MPI)
- MPI not quite friendly to large scale multibody dynamics simulation

The Message-Passing Programming Paradigm

- Sequential Programming Paradigm



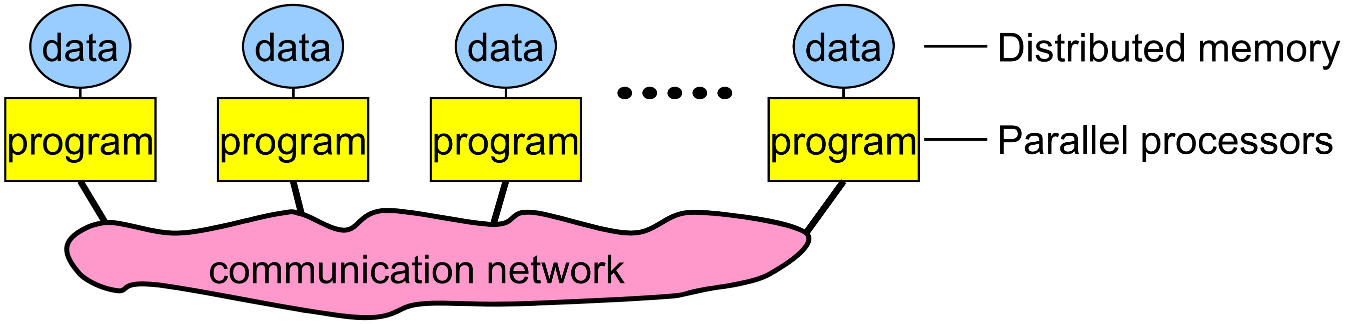
data

program

memory

Processor/Process

- Message-Passing Programming Paradigm



data data data ..... data

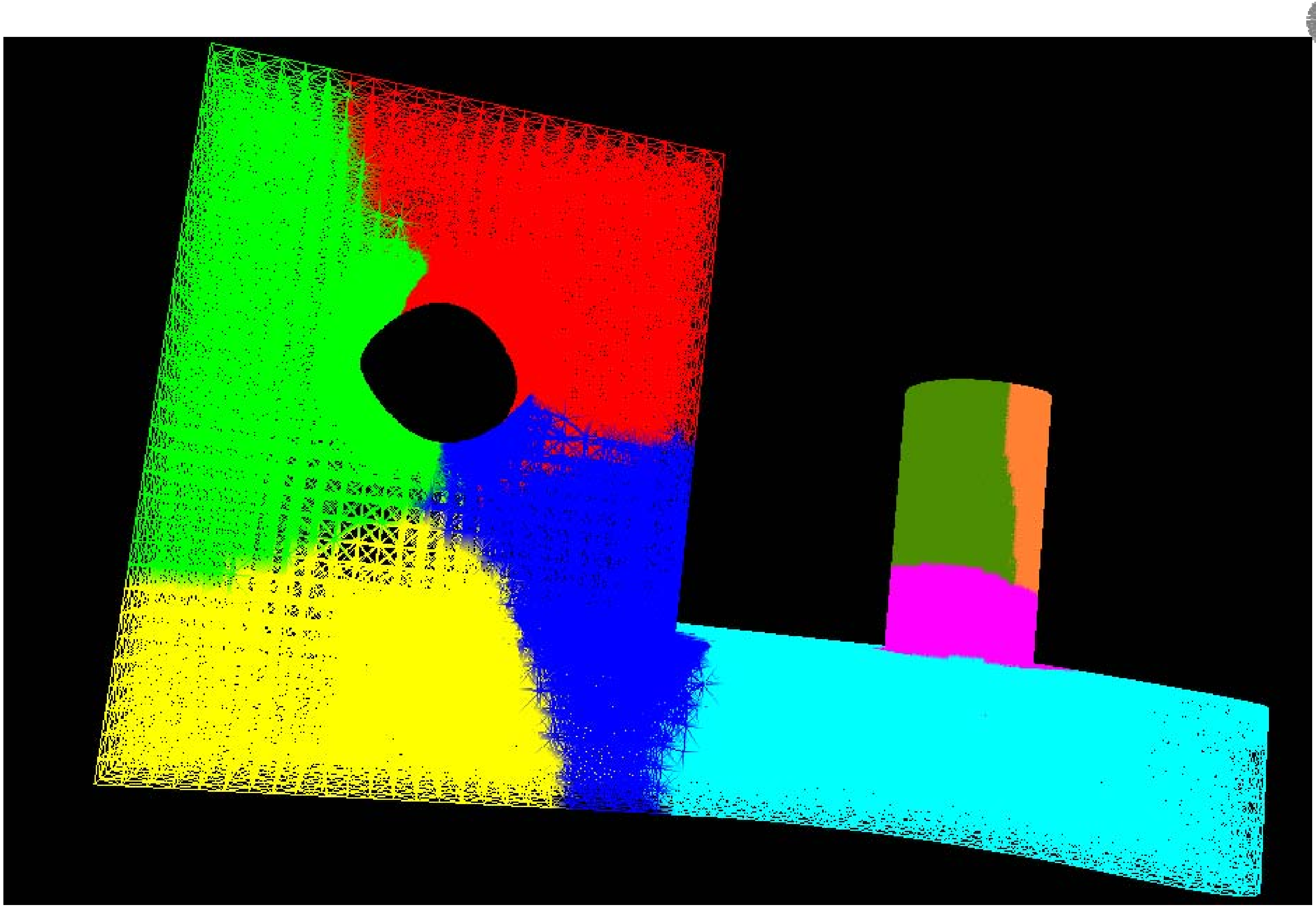
program program program program

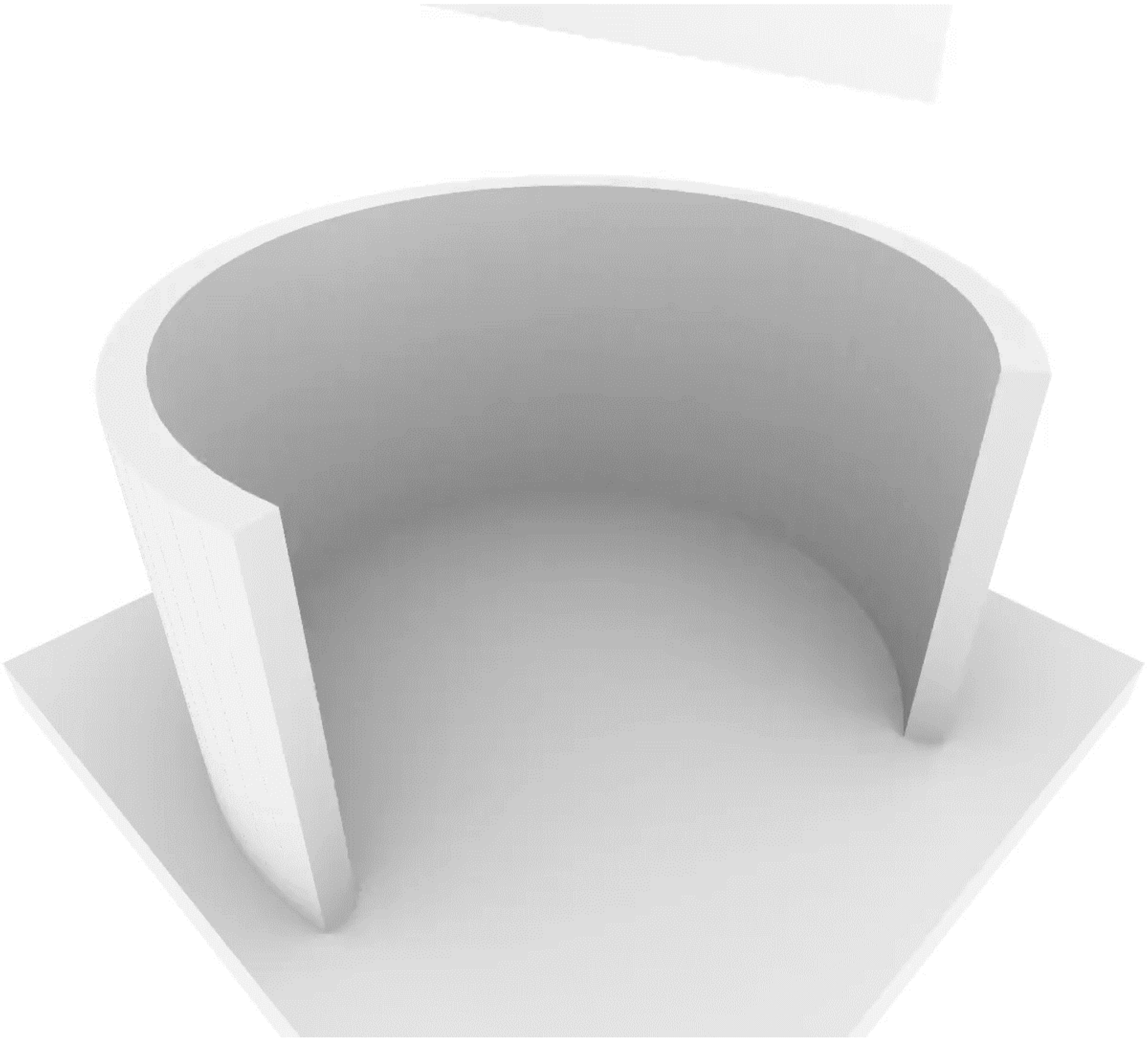
Distributed memory

Parallel processors

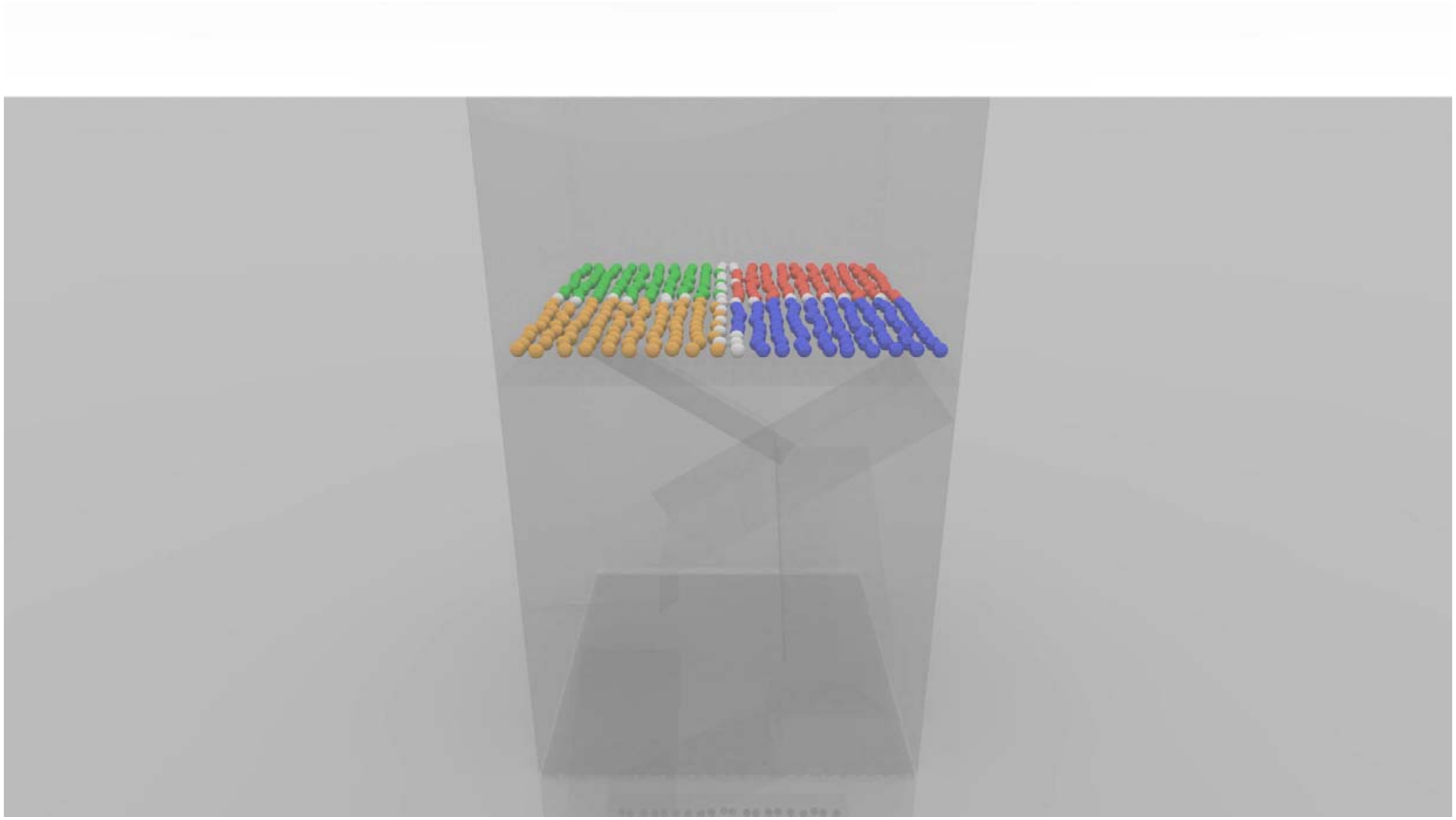
communication network

[ICHEC]→





Computation Using Multiple CPUs & MPI

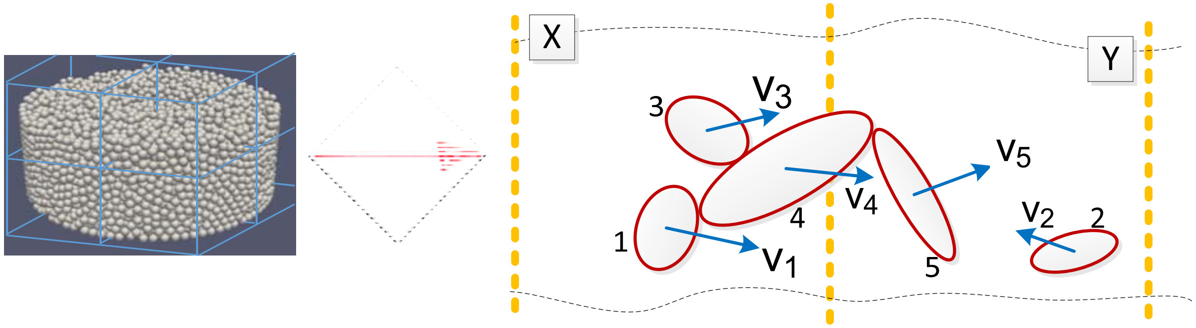


Breaking Up Dynamics for Distributed Computing

- Simulation divided into chunks executed on different cores

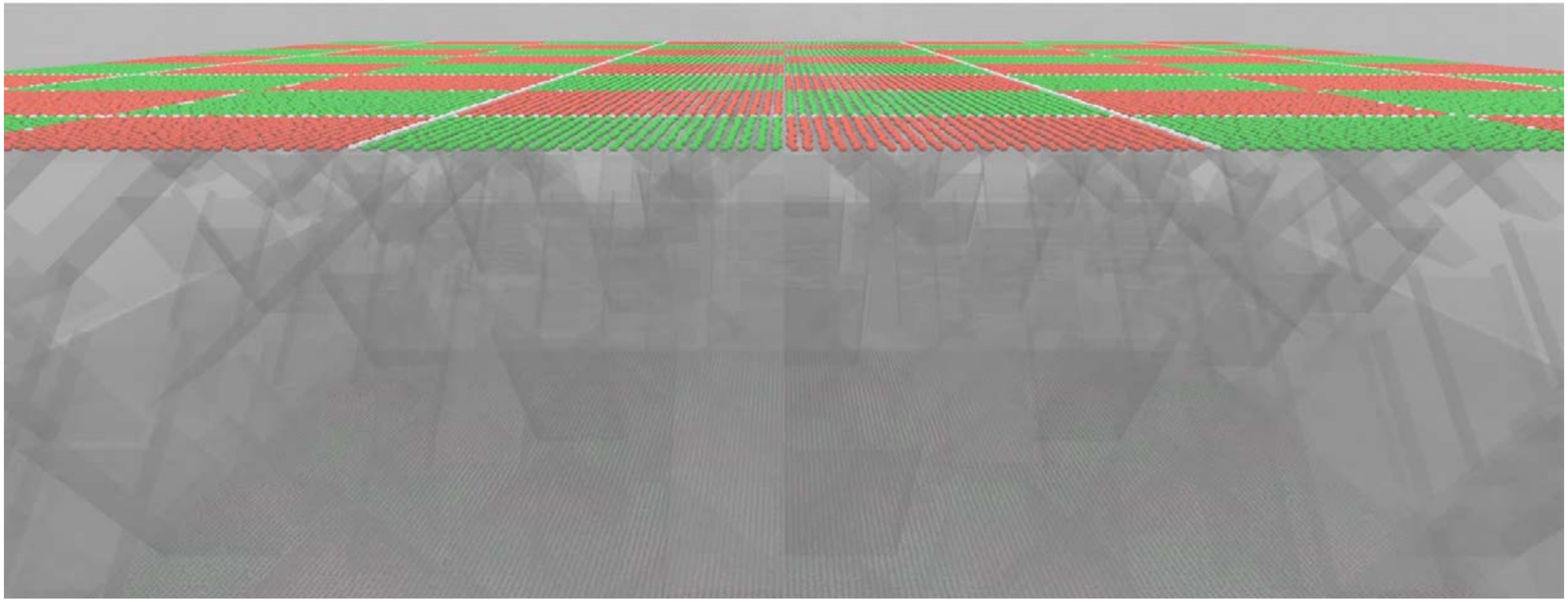
- Elements leave one chunk (subdomain) to move to a different one

- Key issues:
  - Dynamic load balancing
  - Establish a dynamic data exchange (DDE) protocol to implement migration at run time



0.5 Million Bodies on 64 Cores

[Penalty Approach, MPI-based]



Wrapping Things Up

- In the midst of a commoditization of parallel computing
  - Consequence of Moore’s law

- Many alternatives, landscape very fluid
  - Lots of hardware options: GPU, CPU (Intel/AMD), Phi, clusters, FPGAs, etc.
  - Lots of software ecosystems: CUDA, OpenCL, OpenMP, OpenACC, MPI, Charm++, etc.

- Parallel computing here to stay
  - Can pursue new physics + higher fidelity + bridge space/time scales

Chrono Reliance on Parallel Computing

- GPU Computing
  - Granular material
  - Terrain modeling
  - Fluid-solid interaction (CFD)

- Multi-core parallel computing
  - Workhorse in Chrono::Parallel
  - Granular material
  - Terrain modeling

- Distributed memory parallel computing
  - Chrono::Vehicle
  - Pilot implementation, Chrono::FSI
  - Pilot implementation, terrain simulation

- GPU Computing + Multi-core
  - Pilot implementation, meshless methods for mud/silt

Looking Ahead, Chrono Reliance on Parallel Computing

- Distributed memory for terrain

- GPU computing for FSI (fording, sloshing, underwater robotics)