GPUs for Physics

Alan Gray

EPCC

The University of Edinburgh

Why?

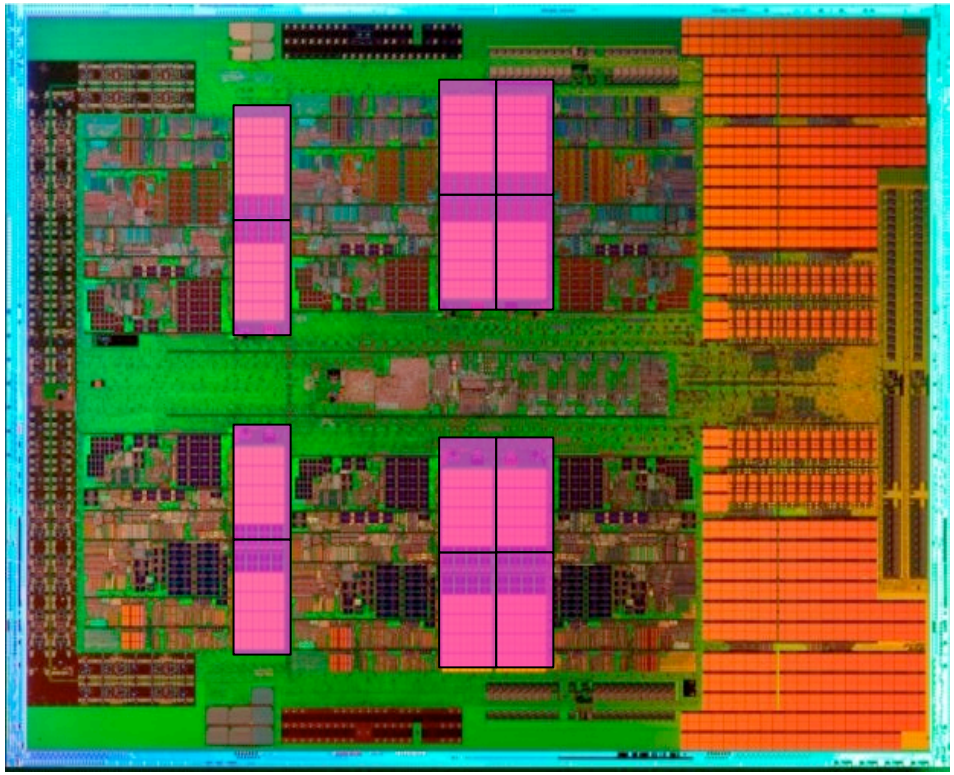
- The power used by a CPU core is proportional to Clock Frequency x Voltage^2
- In the past, computers got faster through increases to frequency
  - Voltage was decreased to keep power reasonable.
- Now, voltage cannot be decreased any further
  - 0s and 1s would become indistinguishable
- Performance increases now achieved through exploiting parallelism
  - Many cores and/or many operations per core
  - need to keep power per core as low as possible

Why?

- Much of the resource expended by CPU cores is on functionality not generally that useful for HPC
- It costs a huge amount of money to design and fabricate new chips
  - No longer feasible for relatively small HPC market
- Over the last few years, Graphics Processing Units (GPUs) have evolved for the highly lucrative gaming market
  - Largely possess the right characteristics for HPC
- GPU vendors have tailored existing GPU architectures to the HPC market

AMD 12-core CPU

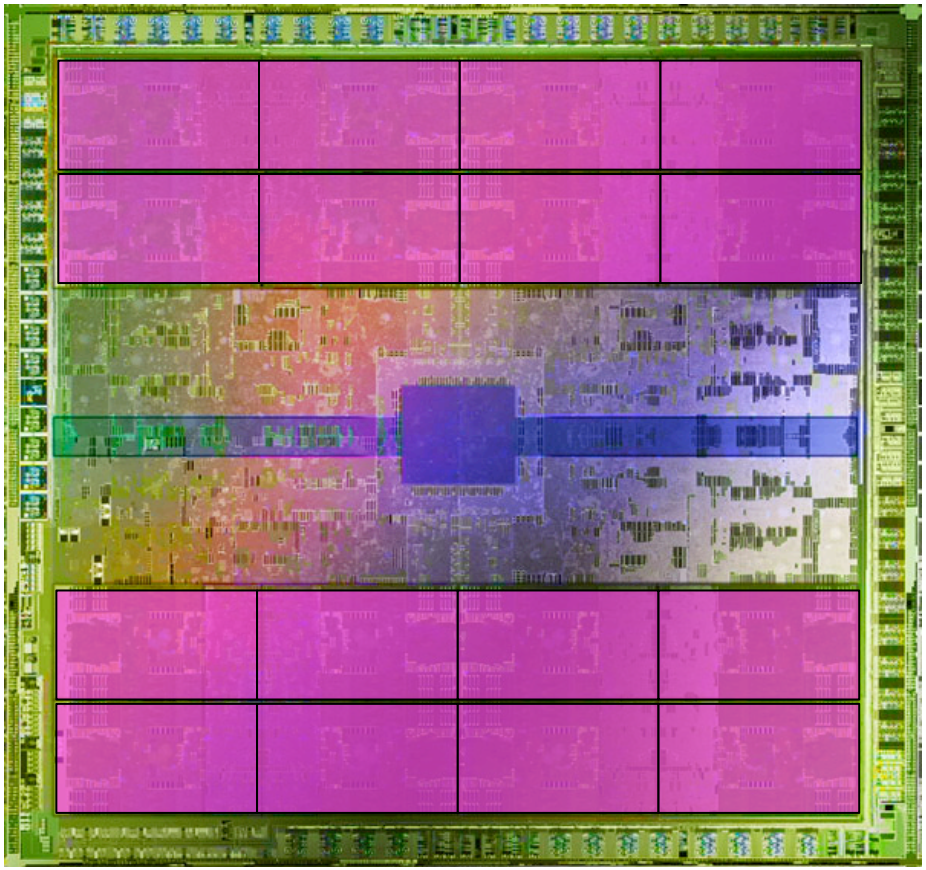
- Not much space on CPU is dedicated to compute



= compute unit
(= core)

NVIDIA Fermi GPU

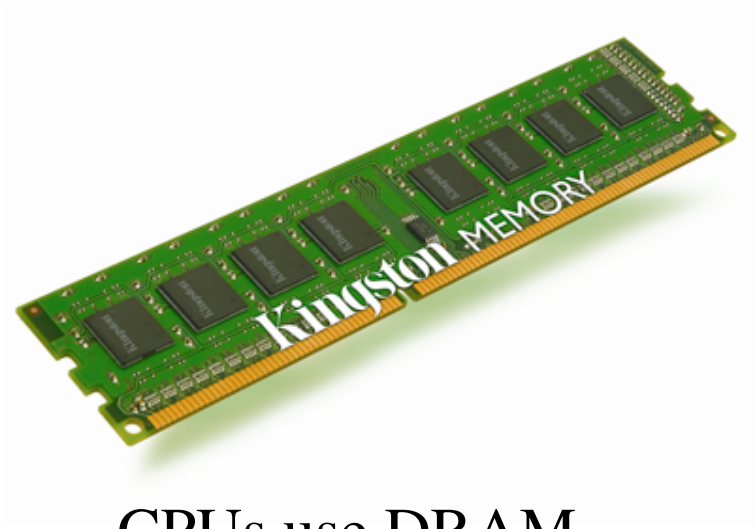
- GPU dedicates much more space to compute
  - At expense of caches, controllers, sophistication etc



= compute unit
(= SM
= 32 CUDA cores)

Memory

- GPUs use Graphics memory: much higher bandwidth than standard CPU memory



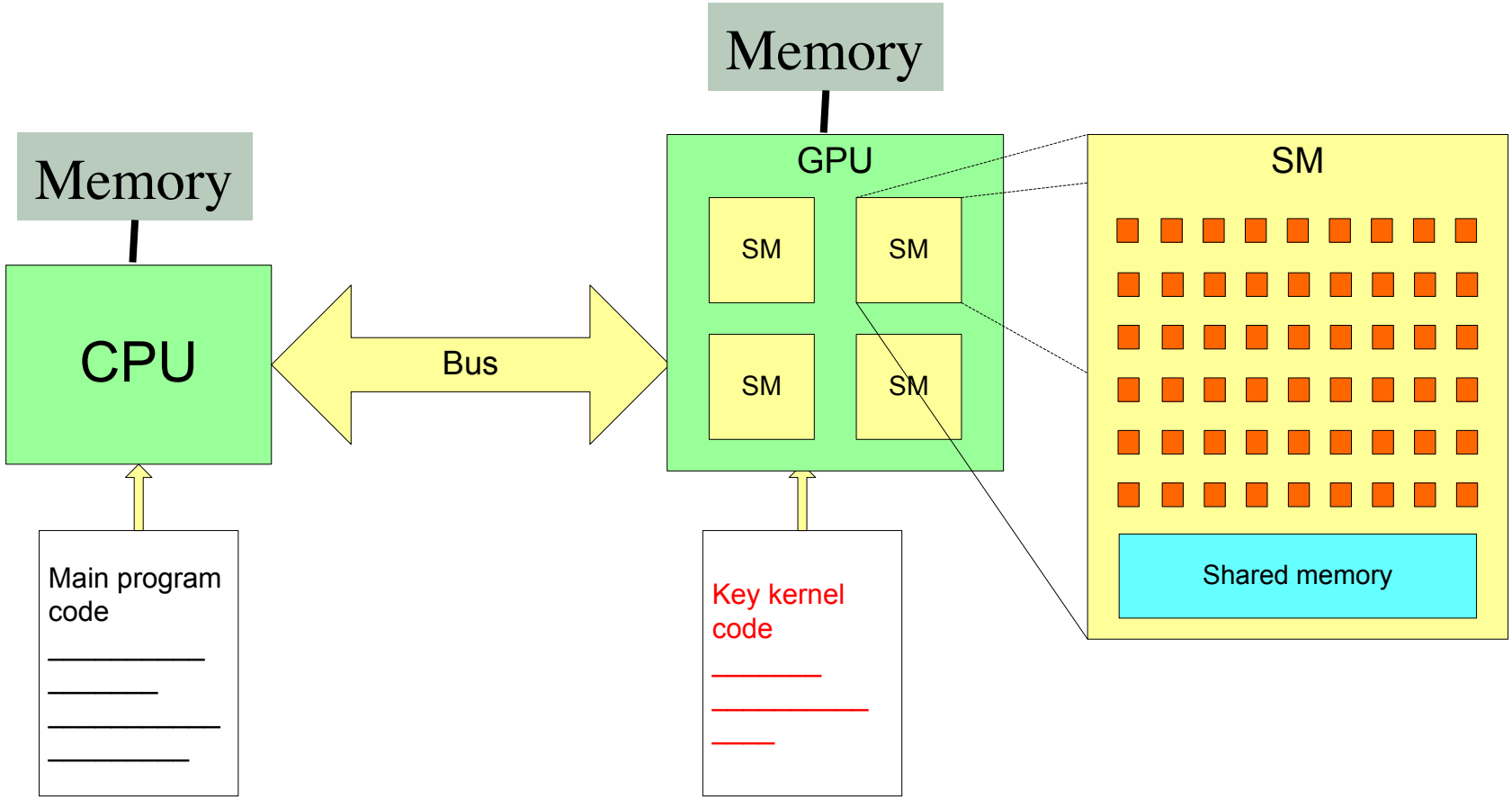
CPUs use DRAM

GPUs use Graphics DRAM

- For many applications, performance is very sensitive to memory bandwidth

Hardware

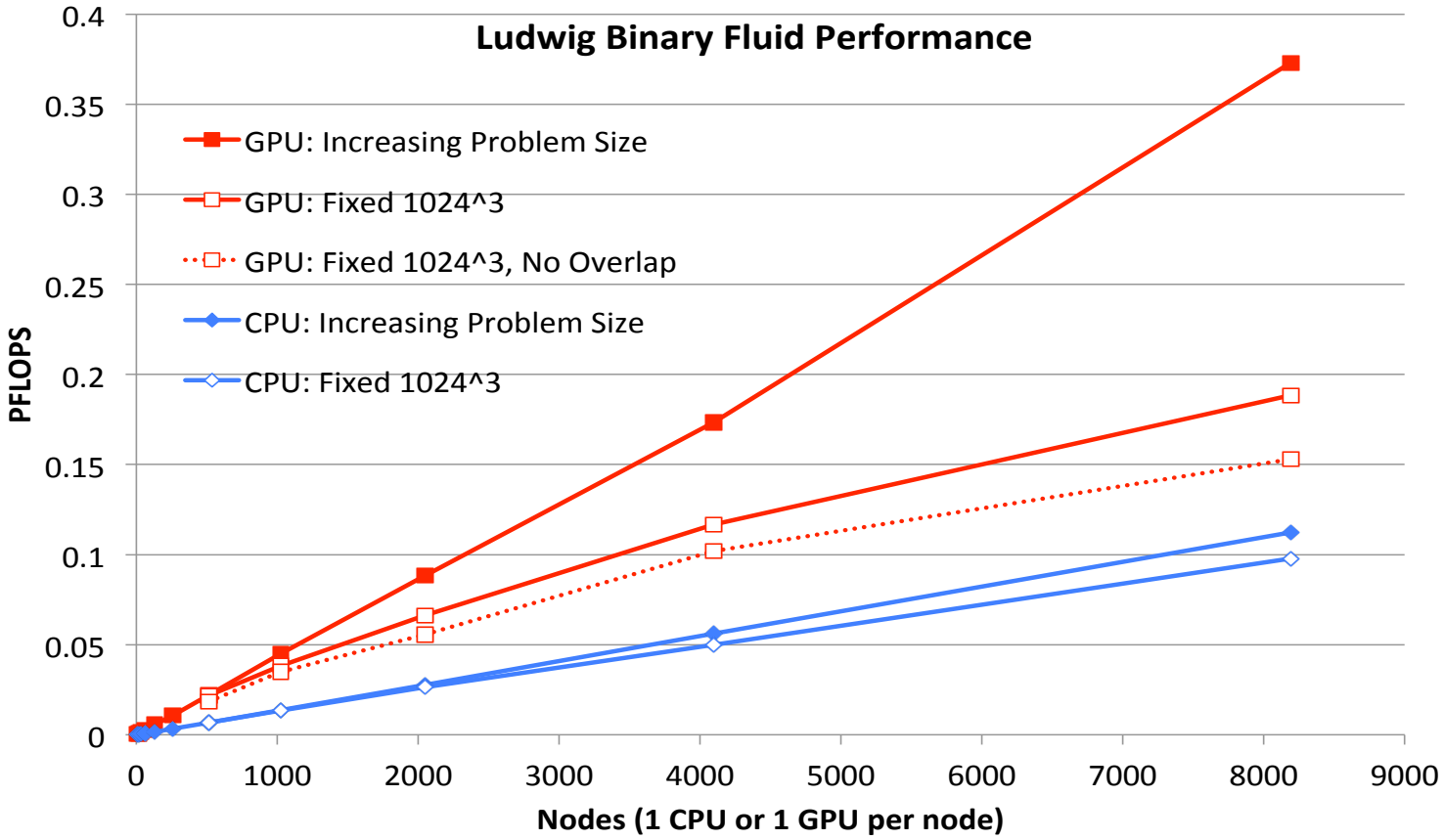
NVIDIA accelerated system:



Programming and Porting

- Programming GPU Accelerated systems involves managing distinct memory spaces and abstracting parallel hierarchy
  - CUDA is most widely used language for NVIDIA GPUs
- Performance achievable depends on problem
  - Many physics problems involve can be solved using grid-based finite difference techniques, and the grid-based parallelism can be mapped to the hardware parallelism
- Porting effort for legacy code depends on existing software engineering implementation
  - All computationally significant functions must be coded for GPU
  - All data accessed by these functions must be transferred to GPU
  - All other functions accessing such data should be moved to GPU to avoid communication overheads
  - This can be very difficult for heavily engineered applications, e.g those involving OO

Lattice Boltzmann Complex Fluid Performance



Lattice QCD on GPUs using Chroma and QUDA

Bálint Joó, Frank Winter
US DOE Jefferson Lab

Mike Clark, NVIDIA

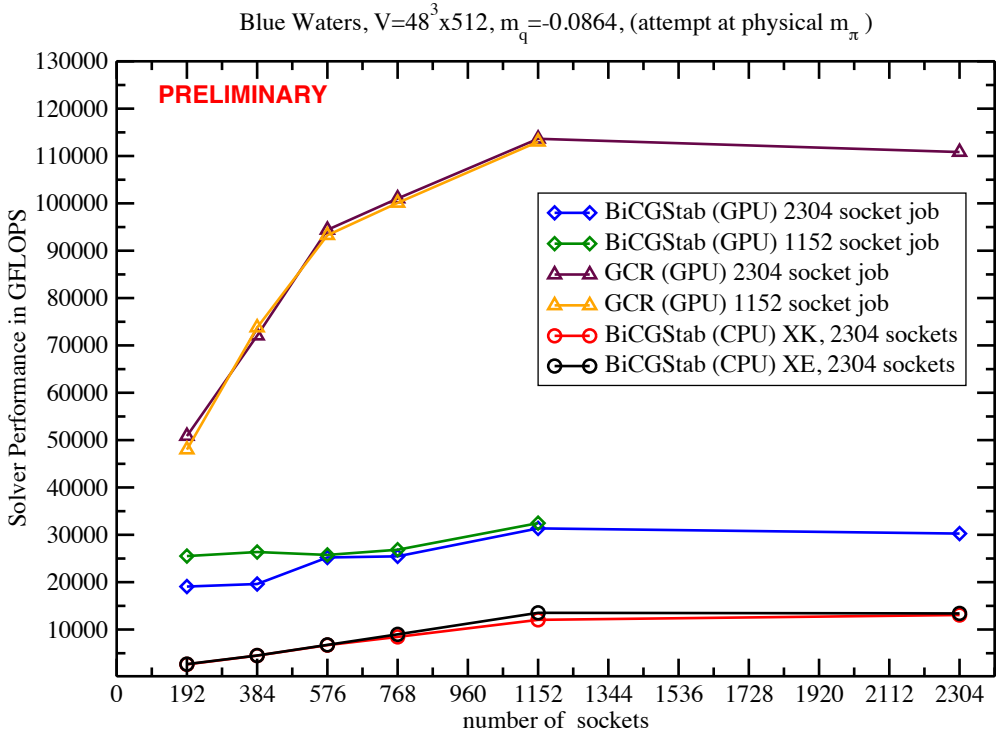
NVIDIA GPU Technology Conference
San Jose, CA
March 20, 2013

Challenges in Gauge Generation

- Amdahl's law effects
  - Unaccelerated code can drag down performance of accelerated code
  - Solution: move more code to GPU
  - Problem: how to preserve 10 year investment in Chroma
  - Solution: target QDP++ layer on which most of Chroma is built
    - QDP-JIT: Frank Winter's talk at this conference
- Strong Scaling
  - As node count increases, local problem size decreases
  - device occupancy is reduced, surface to volume ratio increases
  - latencies start to become important
  - Solution: hardware and software improvements

QUDA Solver Scaling: FLOPS

- DD+GCR solver in QUDA
  - GCR solver with Additive Schwarz domain decomposed preconditioner
  - no communications in preconditioner
  - extensive use of 16-bit precision
- 2011: 256 GPUs on Edge cluster
- 2012: 768 GPUs on TitanDev
- 2013: On BlueWaters
  - ran on up to 2304 nodes (24 cabinets)
  - FLOPs scaling up to 1152 nodes
- Titan results: work in progress



Summary

- GPUs offer performance advantages over CPUs
  - effective use of silicone
  - graphics memory
- Many physics problems can in principle take advantage of GPU architecture
- Porting existing codes can be challenging, especially if they use high-level software abstractions
- There are examples of physics codes scaling to many GPUs in parallel