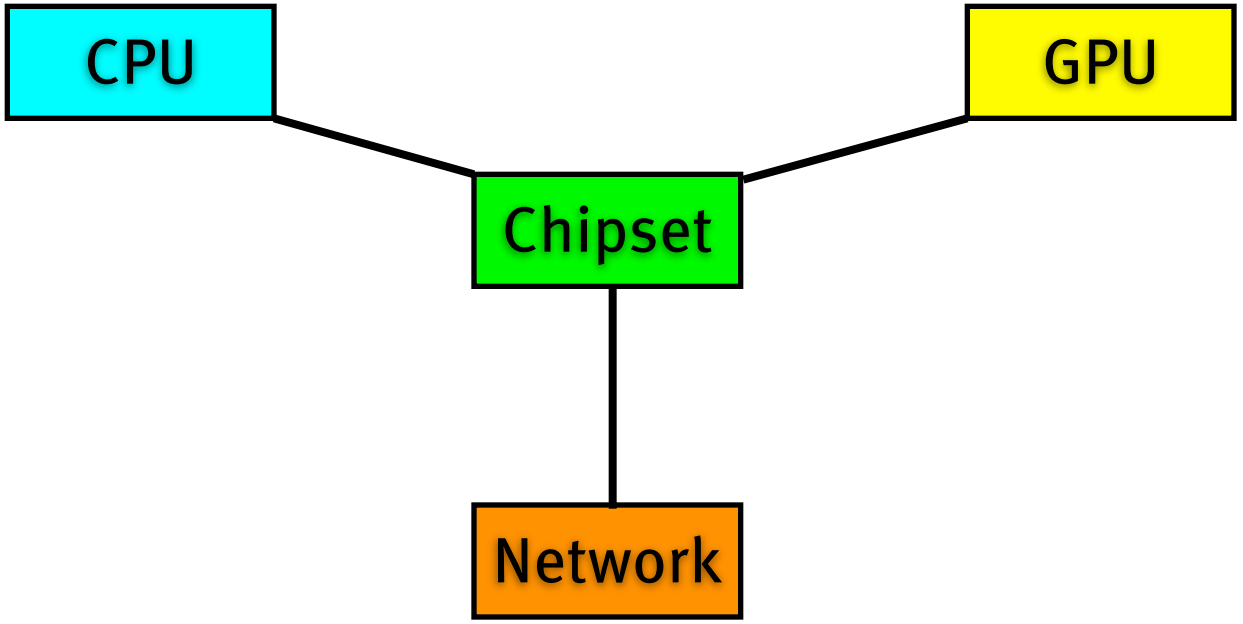
Multi-GPU Computing and GPU MapReduce

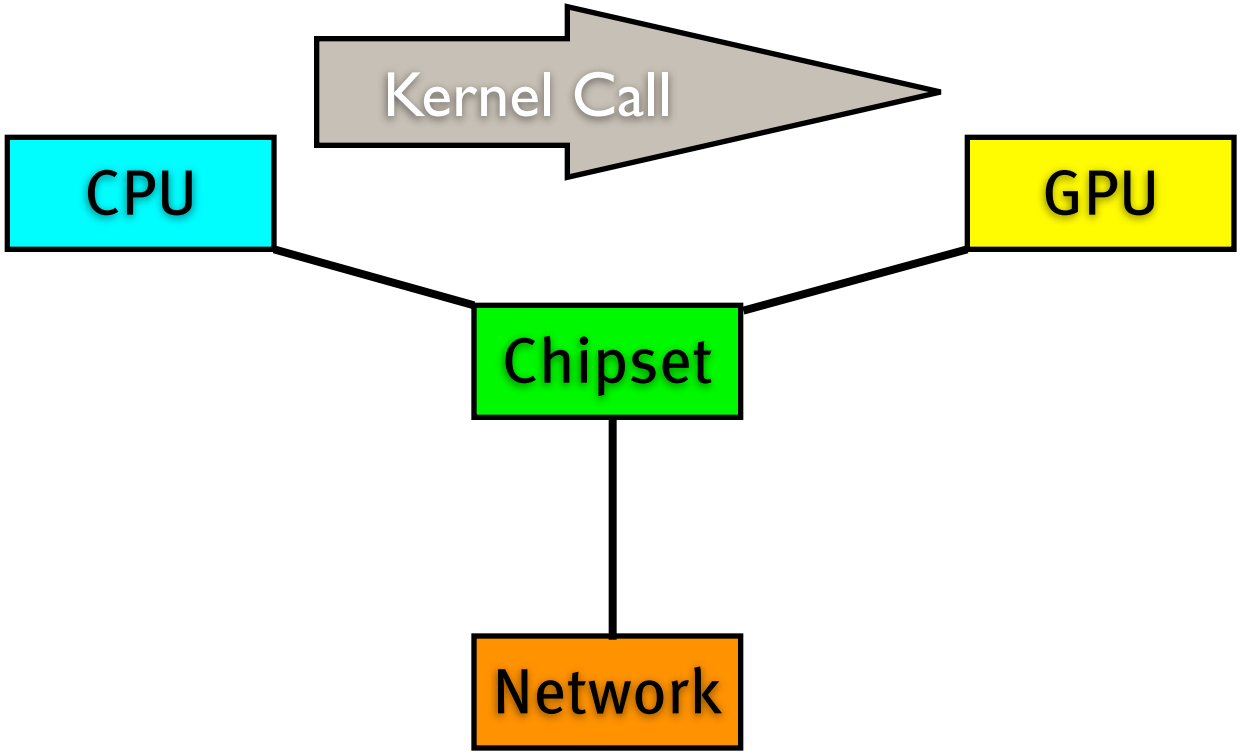
Joint work with Jeff Stuart, UC Davis

John Owens
Associate Professor, Electrical and Computer Engineering
SciDAC Institute for Ultrascale Visualization
University of California, Davis

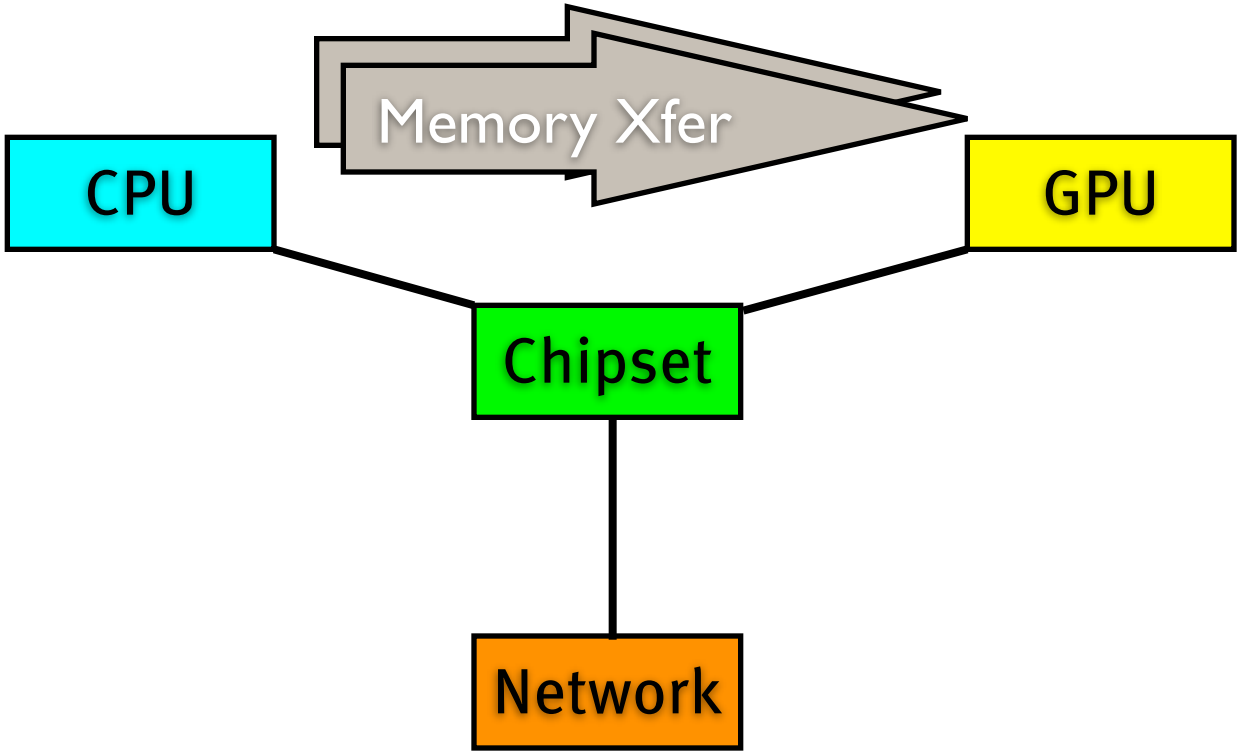
A Modern Computer



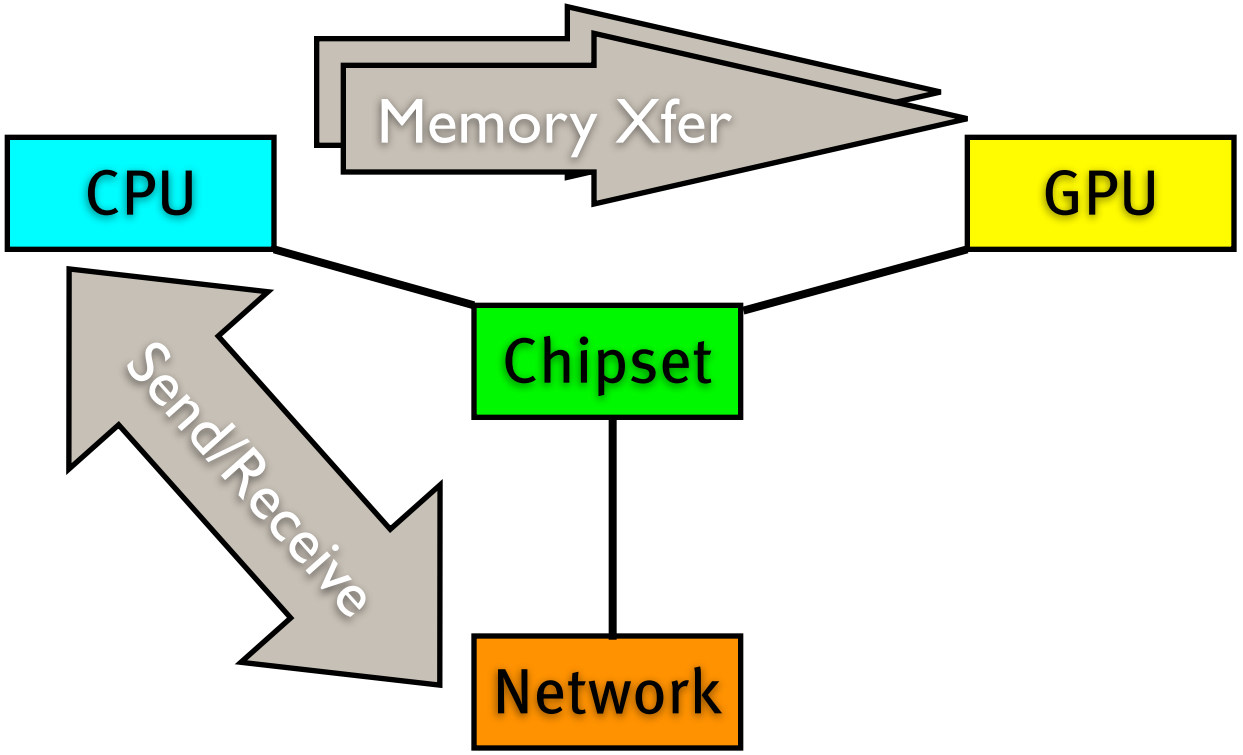
A Modern Computer



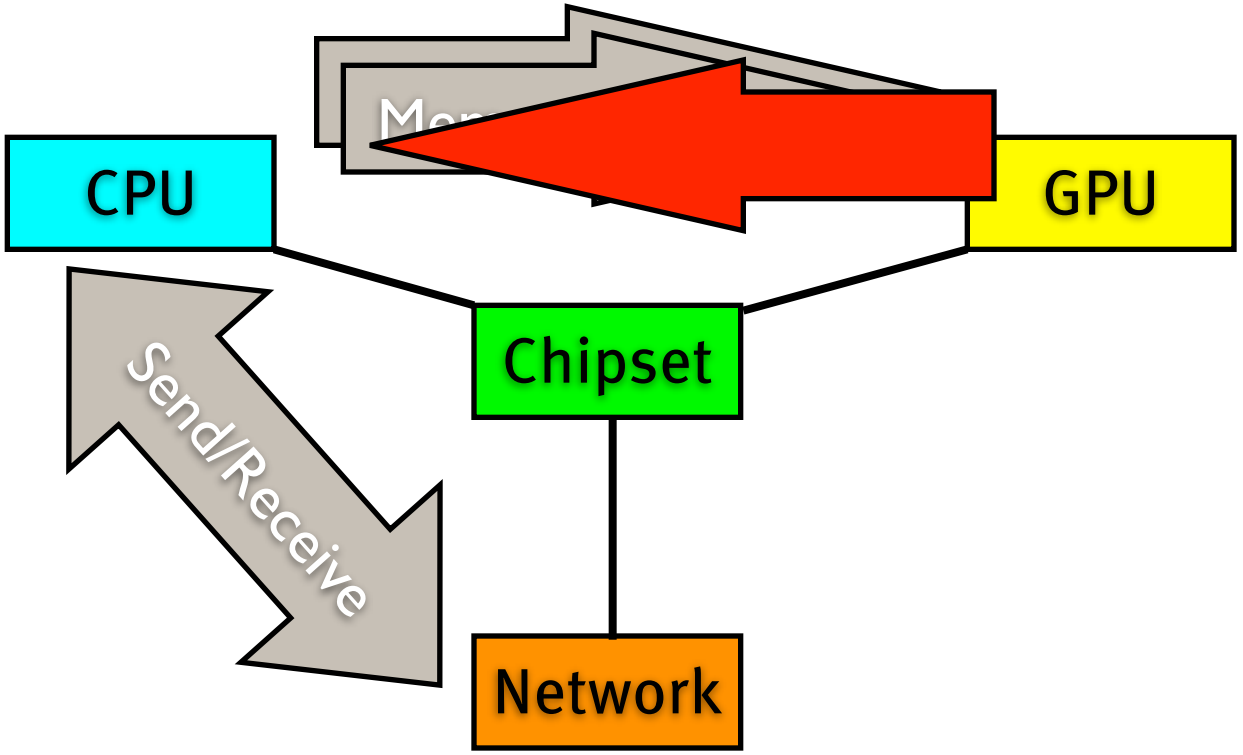
A Modern Computer



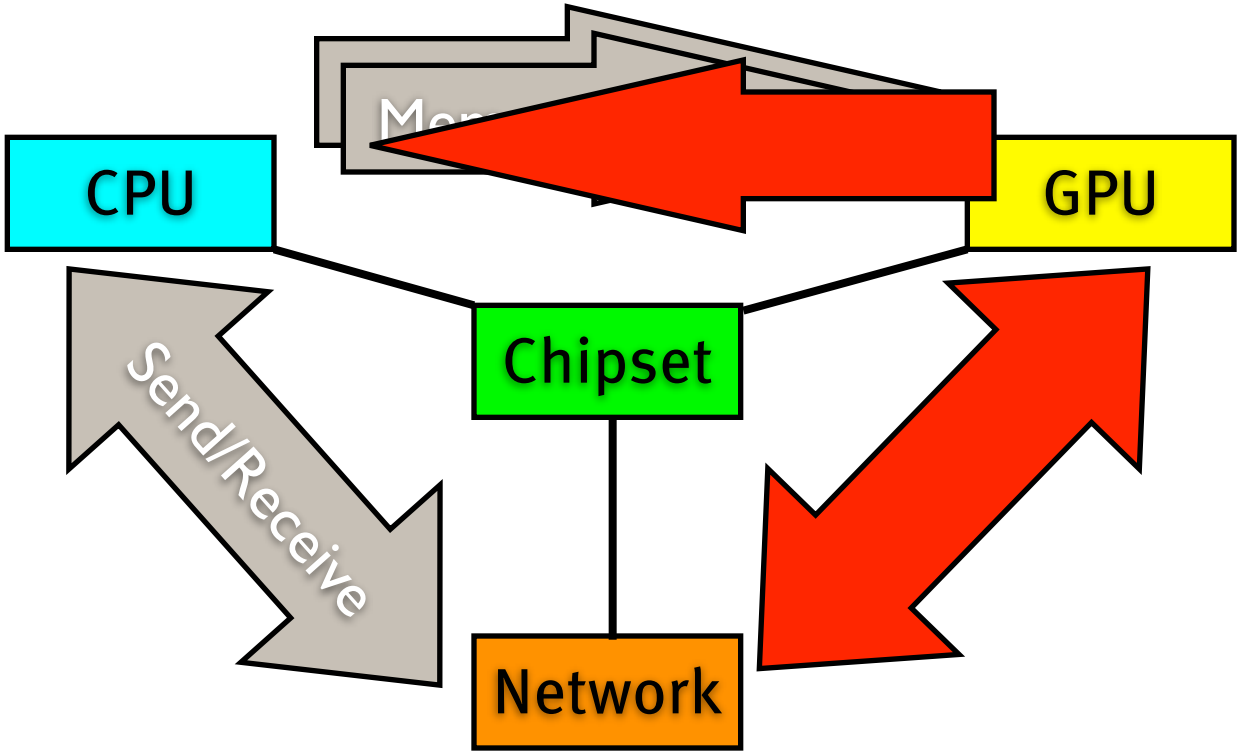
A Modern Computer



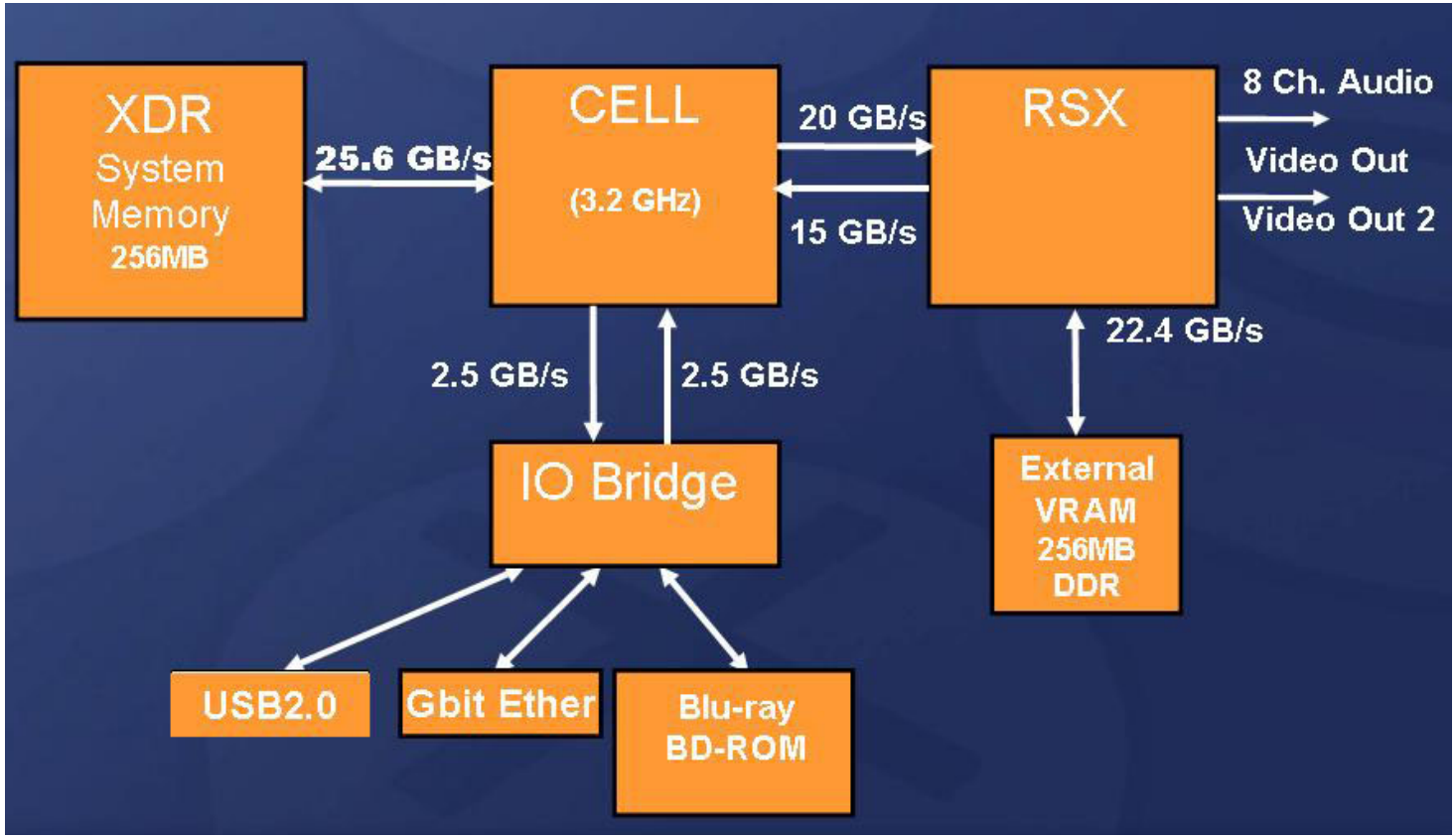
A Modern Computer



A Modern Computer



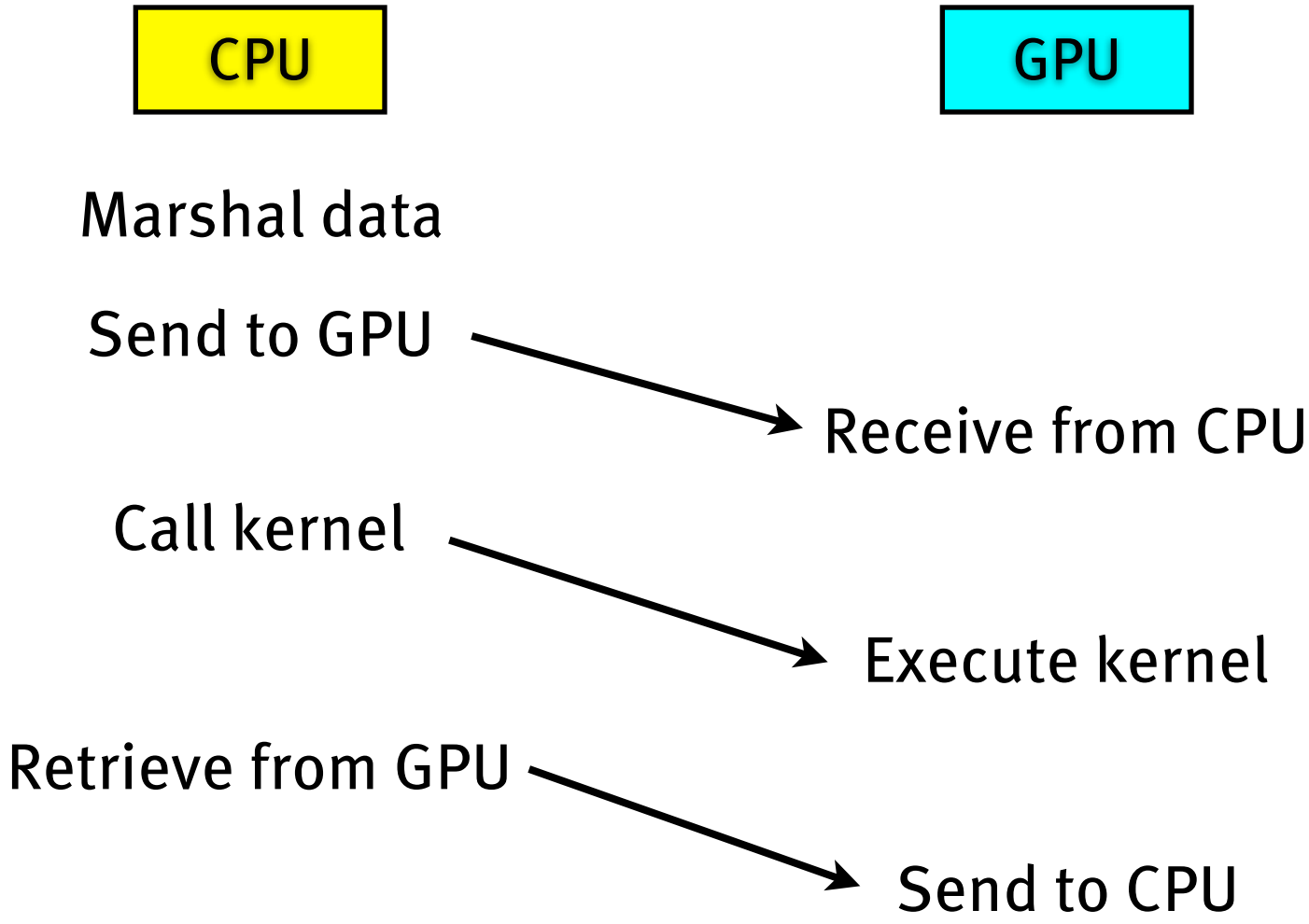
Fast & Flexible Communication



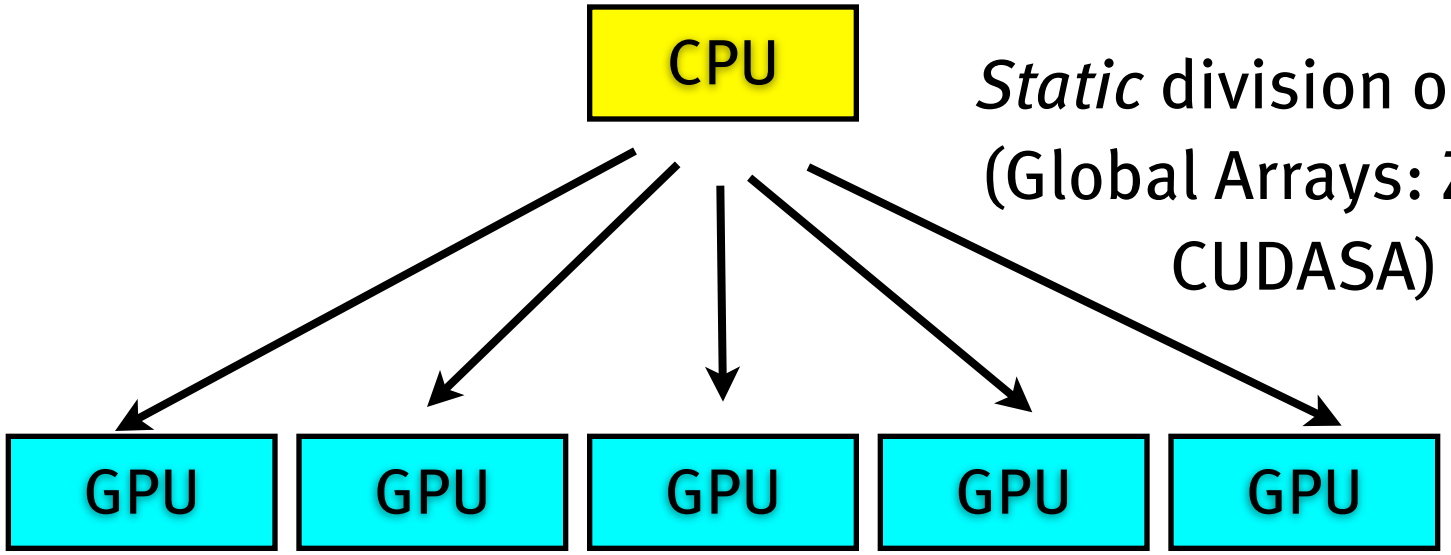
- CPUs are good at creating & manipulating data structures?
- GPUs are good at accessing & updating data structures?

http://www.watch.impress.co.jp/game/docs/20060329/3dps303.jpg

Structuring CPU-GPU Programs

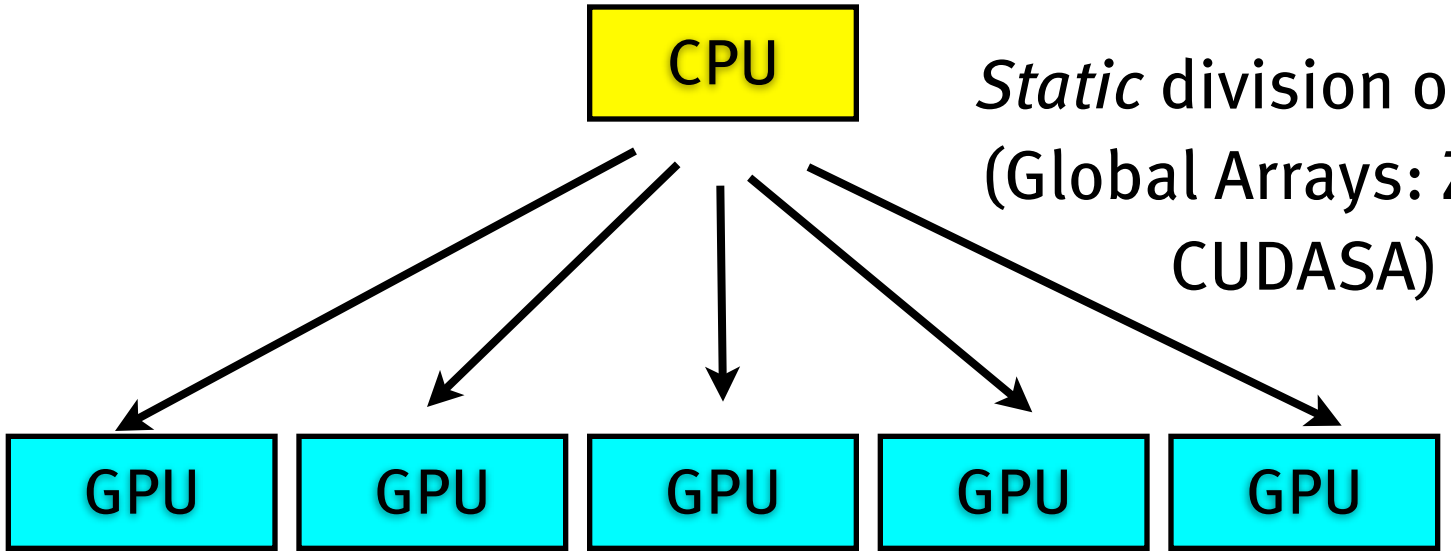


Structuring Multi-GPU Programs



Static division of work (Global Arrays: Zippy, CUDASA)

Structuring Multi-GPU Programs

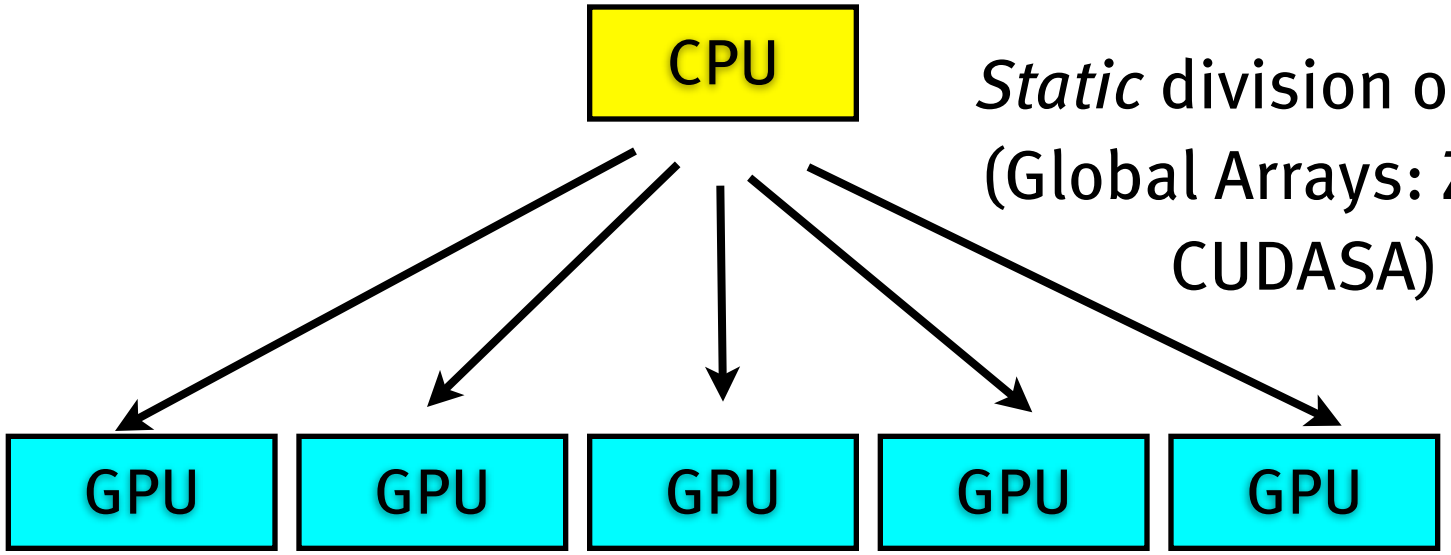


Static division of work (Global Arrays: Zippy, CUDASA)

Want to run on GPU:

```
if (foo == true) {
  GPU[x][bar] = baz;
} else {
  bar = GPU[y][baz];
}
```

Structuring Multi-GPU Programs



Static division of work (Global Arrays: Zippy, CUDASA)

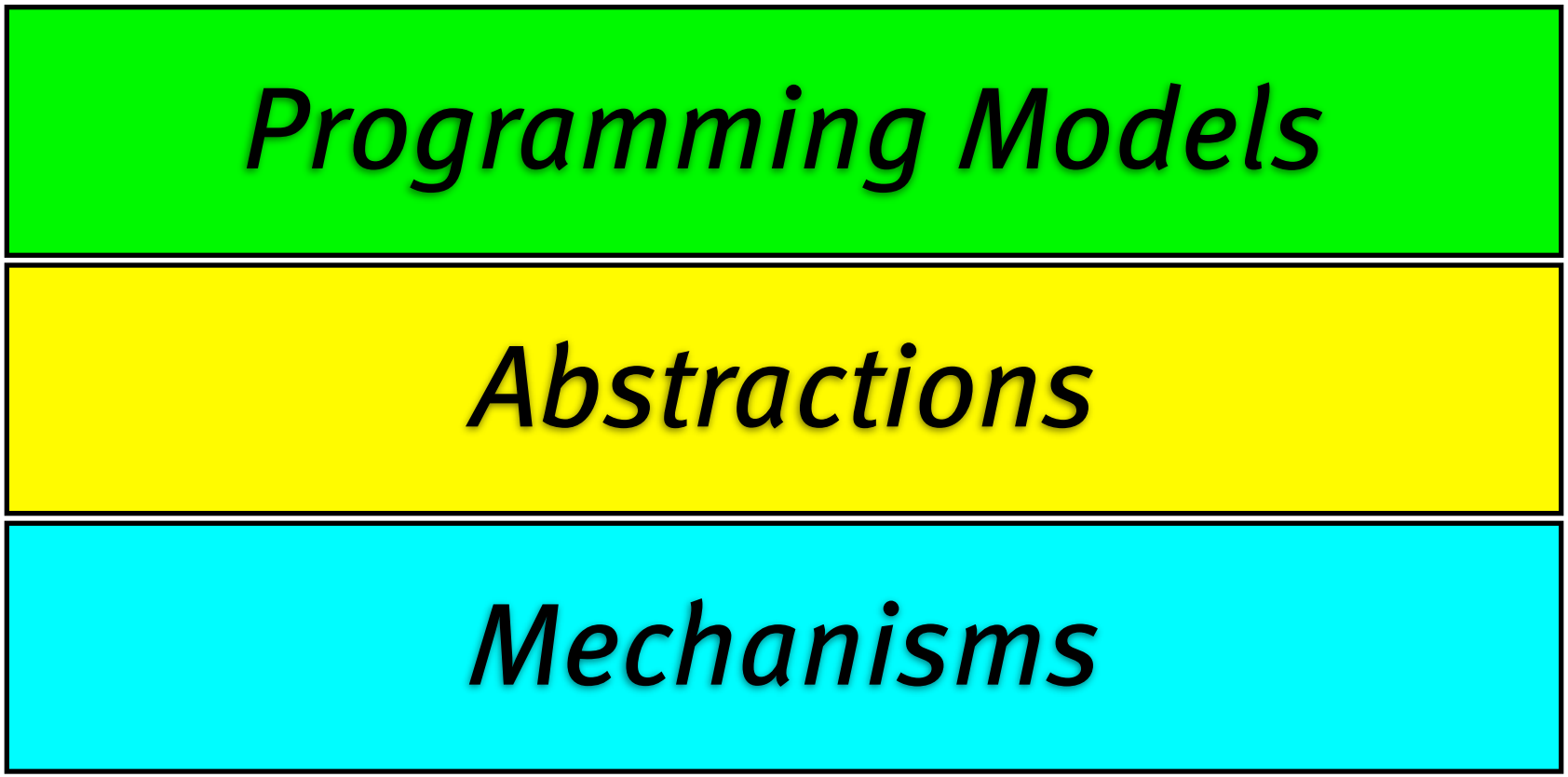
Want to run on GPU:

```
if (foo == true) {
  GPU[x][bar] = baz;
} else {
  bar = GPU[y][baz];
}
```

Instead, *GPU as slave*.

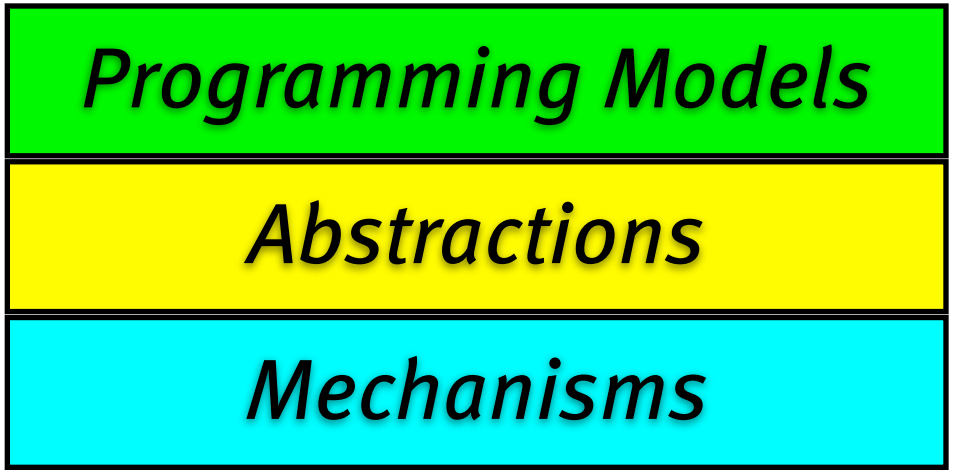
Goal: GPU as first-class citizen.

Our Research Program



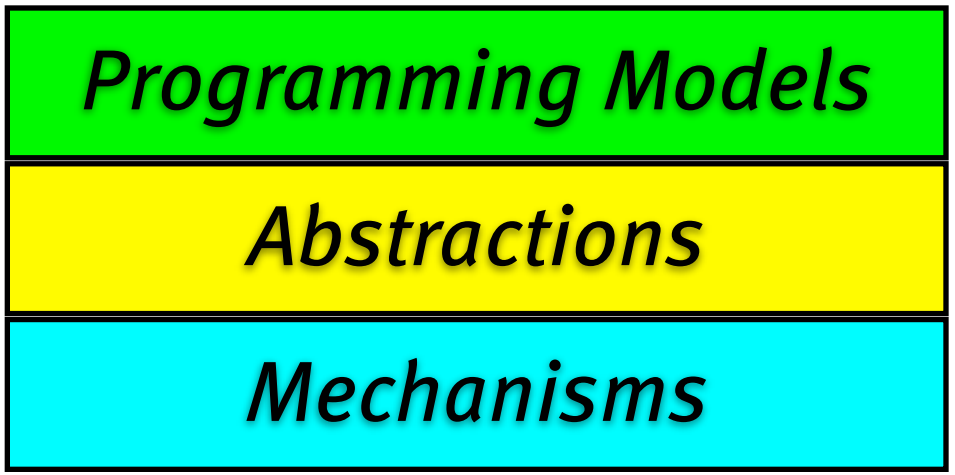
Example

- Abstraction: GPU initiates network send
- Problems:
  - GPU can't communicate with NI
  - GPU signals CPU



Example

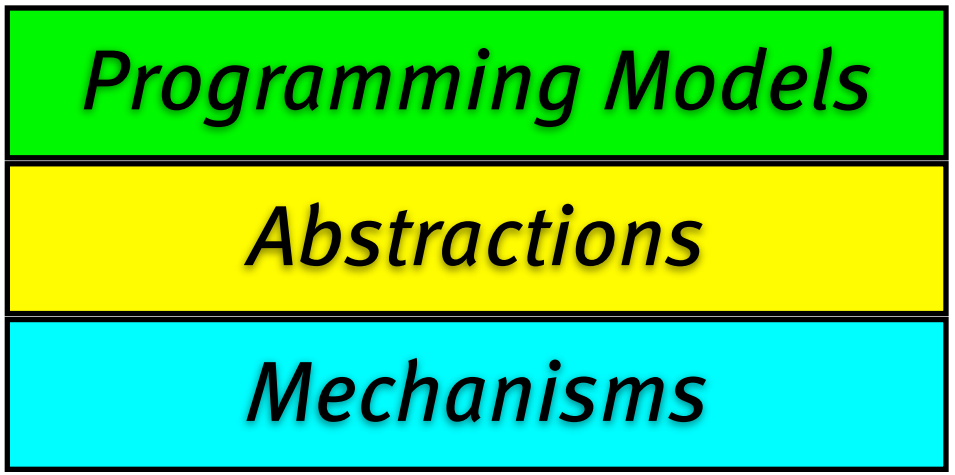
- Abstraction: GPU initiates network send
- Solution:
  - CPU allocates "mailbox" in GPU mem
  - GPU sets mailbox to initiate network send
  - CPU polls mailbox



Example

Take-home: Abstraction does not change even if underlying mechanisms change

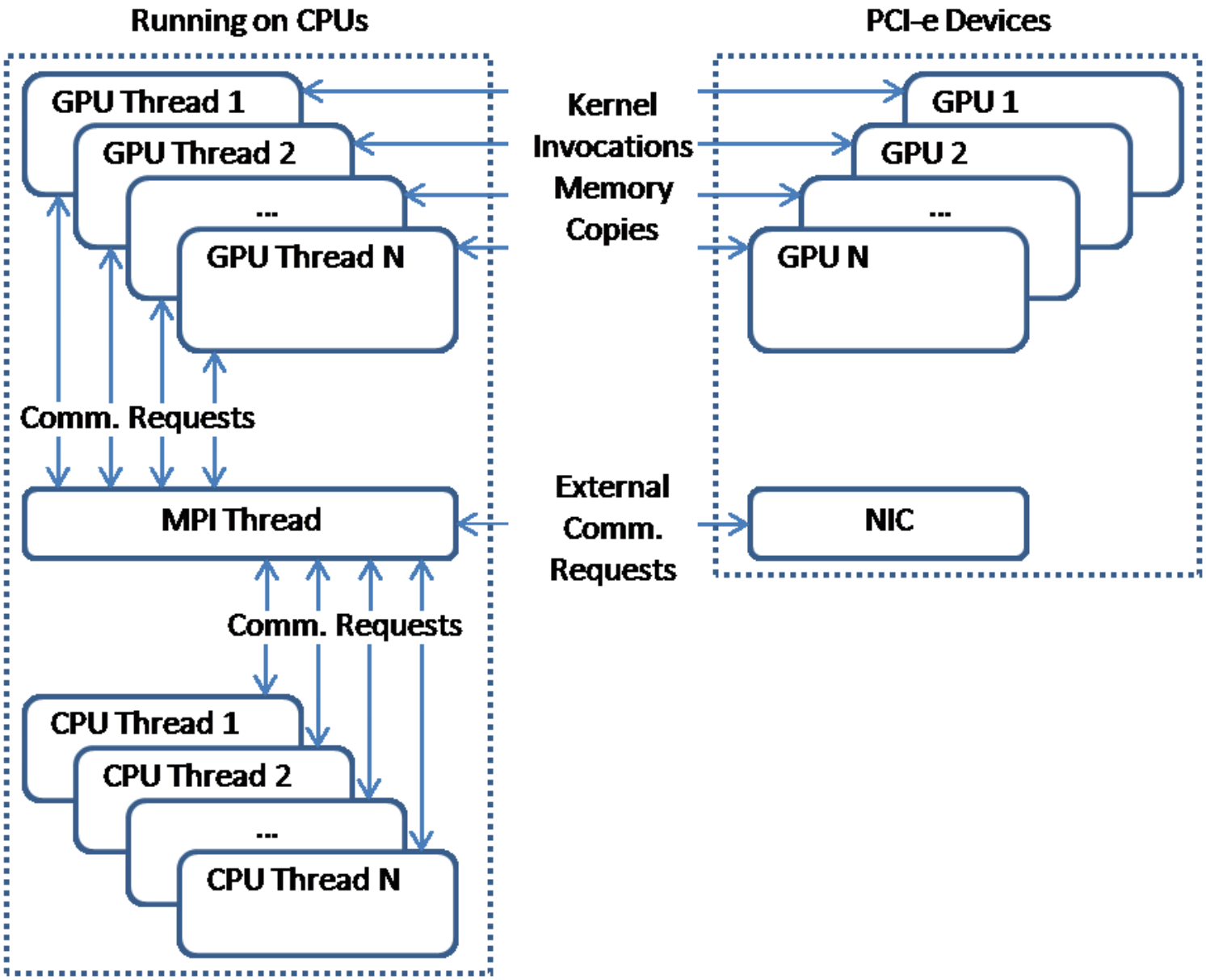
- Abstraction: GPU initiates network send
- Solution:
  - CPU allocates "mailbox" in GPU mem
  - GPU sets mailbox to initiate network send
  - CPU polls mailbox



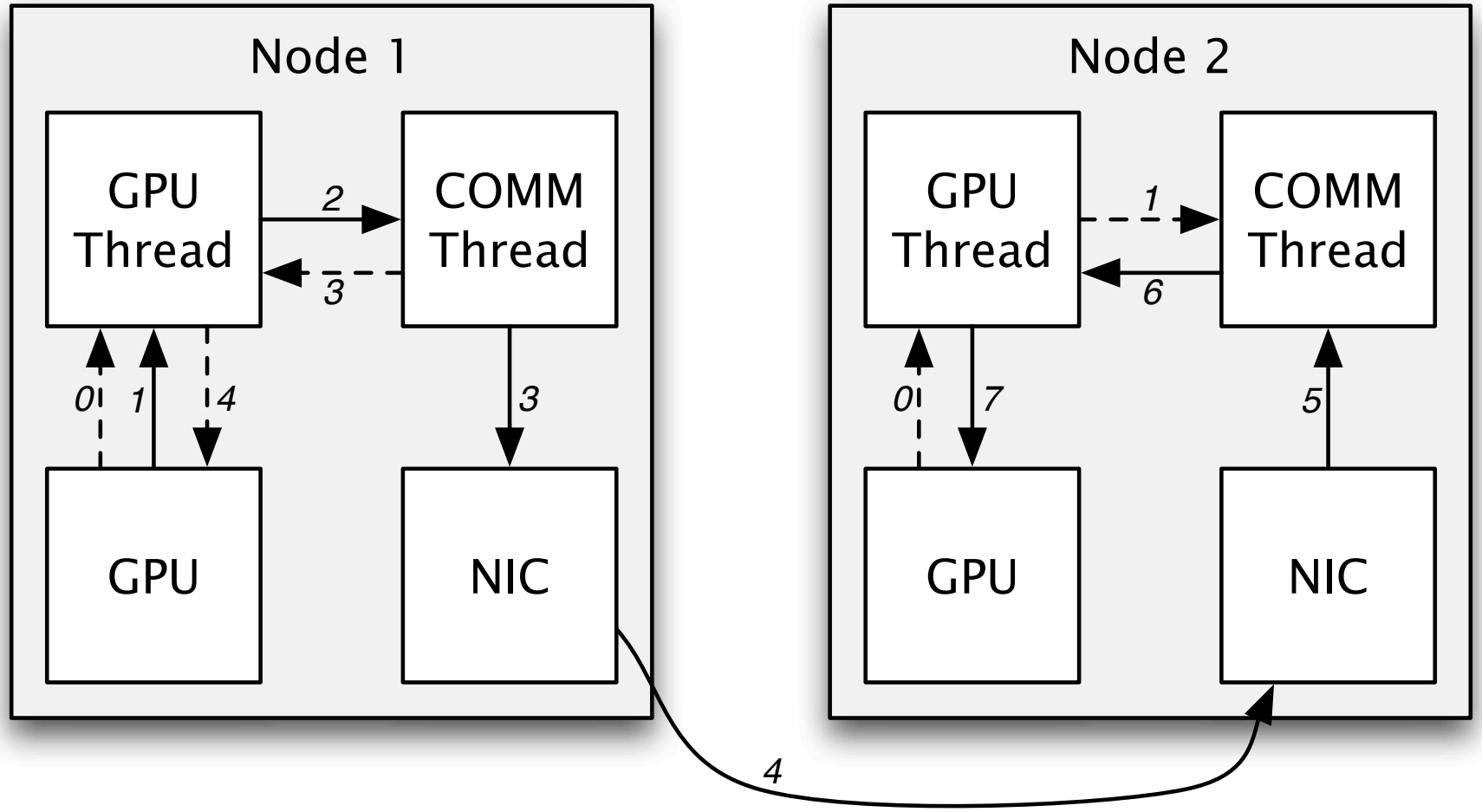
DCGN: MPI-Like Programming Model

- Distributed Computing for GPU Networks (DCGN, pronounced decagon)
  - MPI-like interface
- Allows communication between all CPUs and GPUs in system
  - Allow GPU to source/sink communication
  - Multithreaded communication via MPI
  - Both synchronous and asynchronous (‹- overlap!)
  - Collectives
  - Multiplex MPI addresses (“slots”)

Architecture



Node-to-Node Send



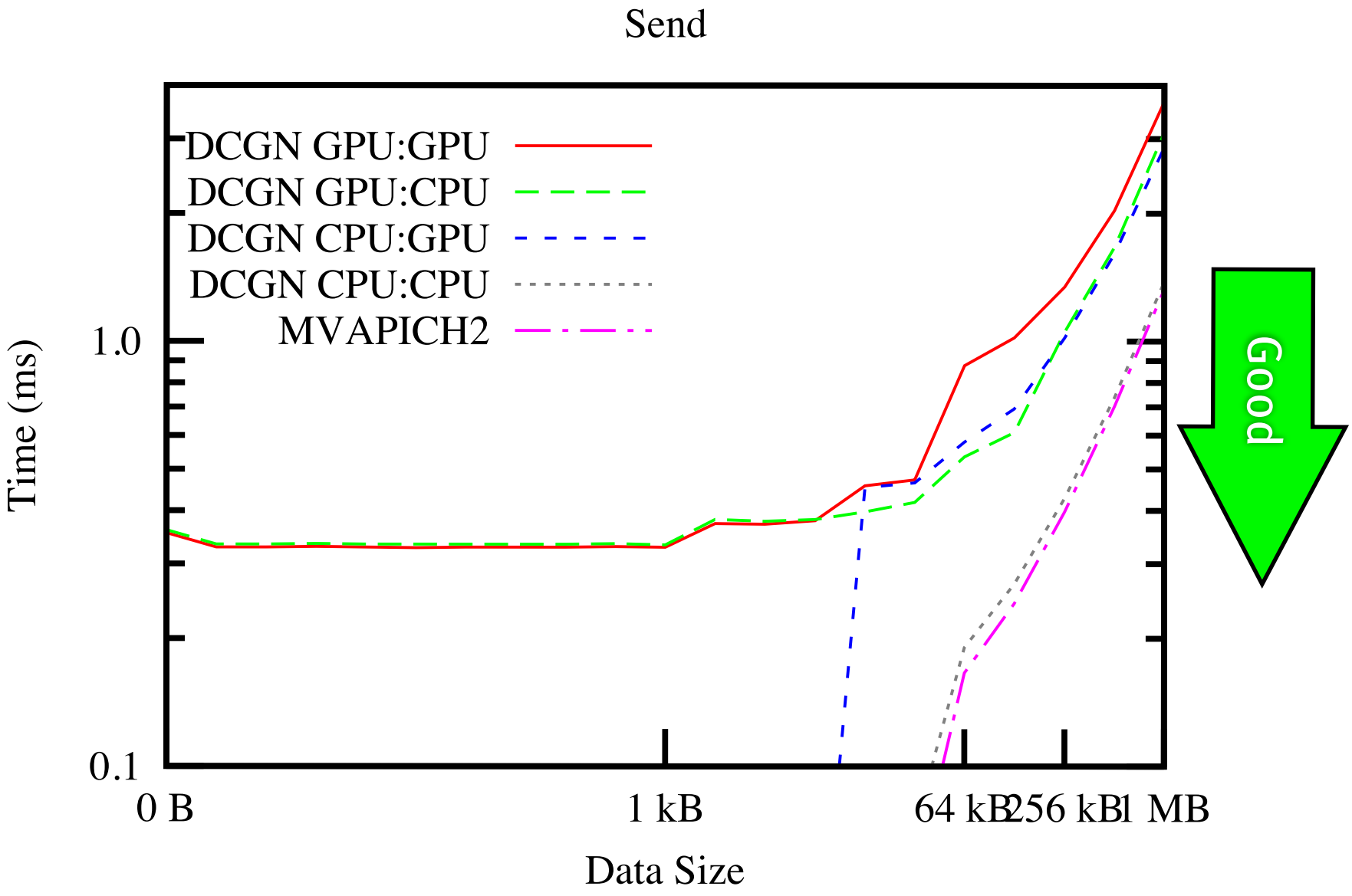
Complexities with GPU Threads

- In MPI, each processor has many (1? 10s? 100s?) of active threads/processes
- GPUs have thousands to millions of active threads at one time
  - Are those threads all cooperating on the same piece of work (logically communicating with one CPU thread)?
    - Uniform computation across many threads
  - Are those threads all doing separate pieces of work (logically communicating with many CPU threads)?
    - Small percentage of threads take 1000x longer

DCGN: Slots

- Each GPU is given 1–*n* slots for *n* GPU threads, specified by the user
- All communication requests have an additional “slot” parameter
- No implicit synchronization of slots
- Not part of MPI (this is why we didn’t use MPI)

Microbenchmarks



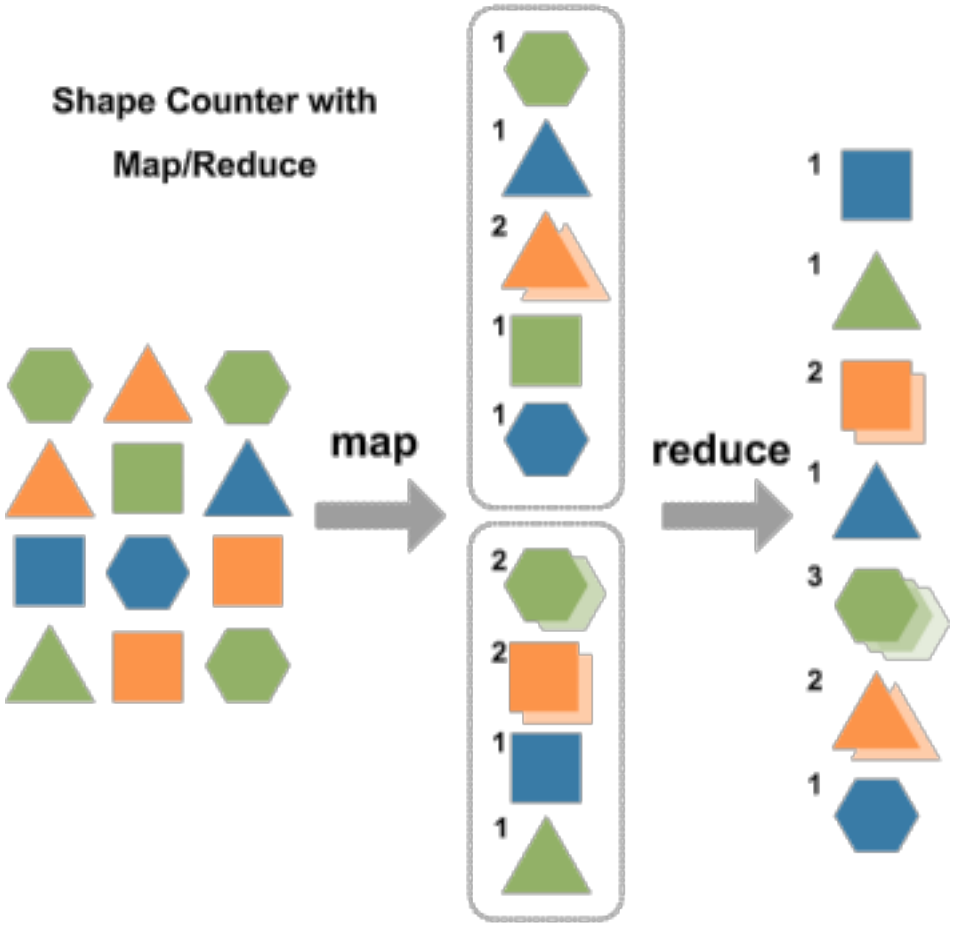
App Results

- N-body (one-to-all broadcast)
- Cannon's matrix multiplication (simultaneous communication)
- Mandelbrot set (unpredictable communication)
- All three had at least 90% of the performance of GPU-as-slave

DCGN Conclusions

- So why DCGN?
  - Future-proof code: use abstractions, underlying mechanisms can improve
  - DCGN code is at a higher level of abstraction
- HW improvements
  - Ability for GPU to initiate communication
  - Direct GPU-GPU and/or GPU-NIC desirable
  - Faster/lower-latency CPU-GPU communication
    - Upcoming Fusion-style CPU-GPU hybrids

MapReduce



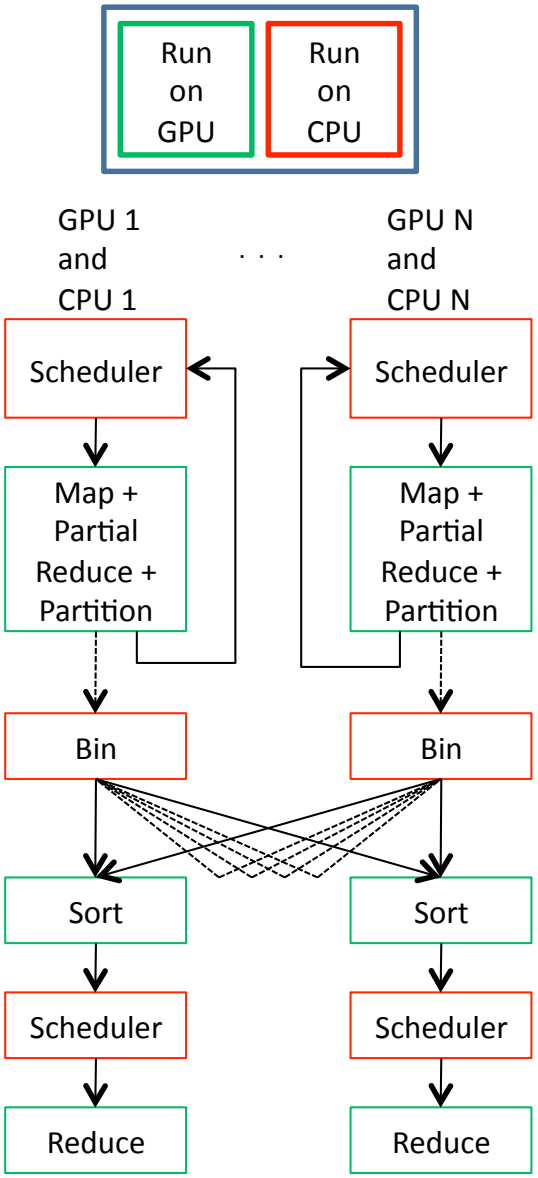
http://m.blog.hu/dw/dwbi/image/2009/Q4/mapreduce_small.png

Why MapReduce?

- Simple programming model
- Parallel programming model
- Scalable

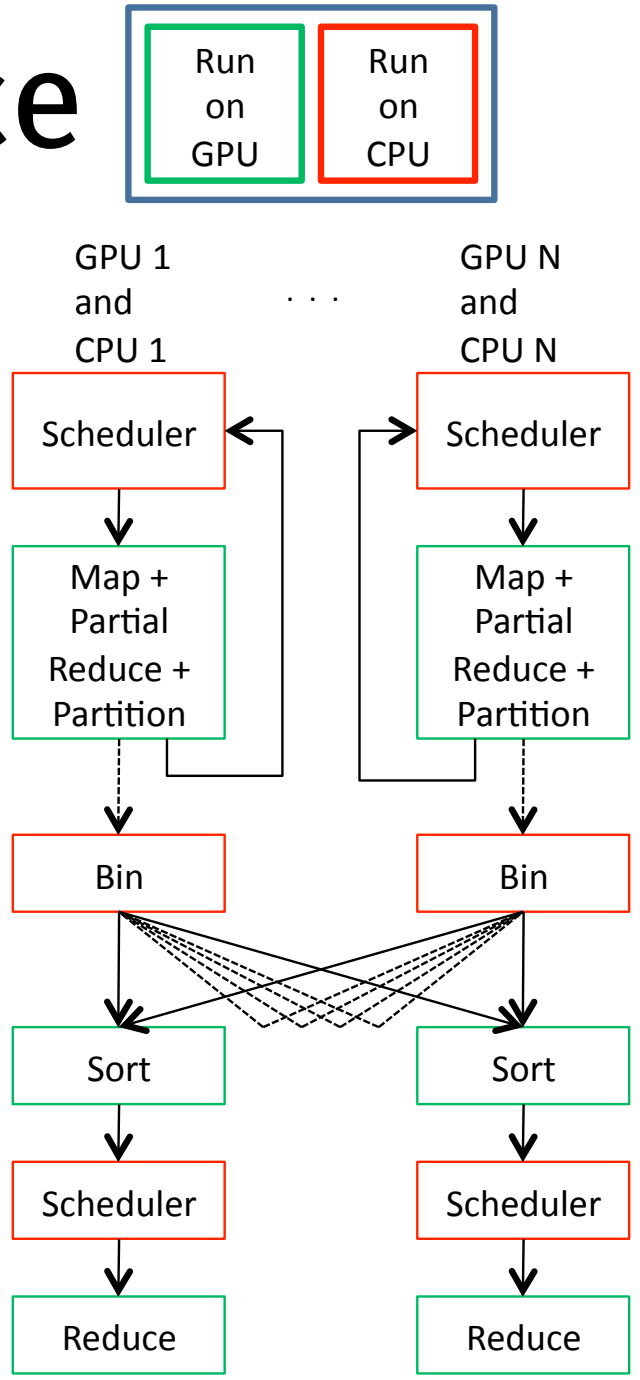
- Previous GPU work: neither multi-GPU nor out-of-core

Block Diagram



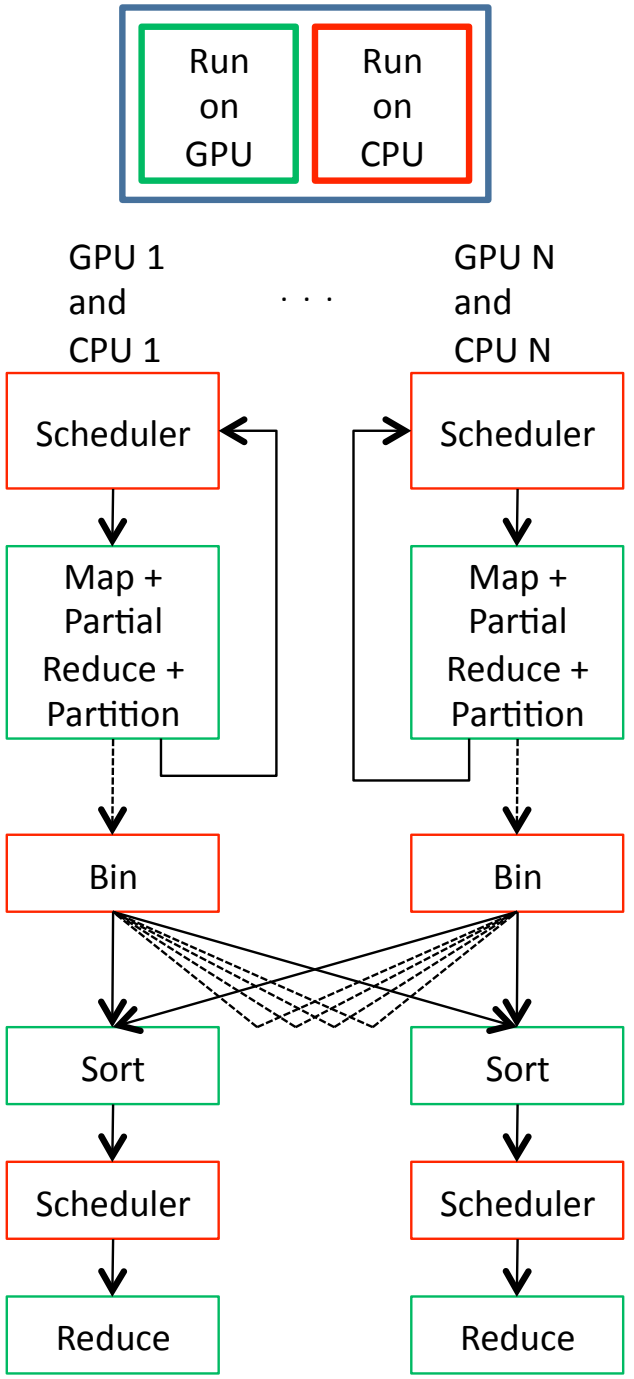
Keys to Performance

- Process data in chunks
  - More efficient transmission & computation
  - Also allows out of core
- Overlap computation and communication
- Accumulate
- Partial Reduce



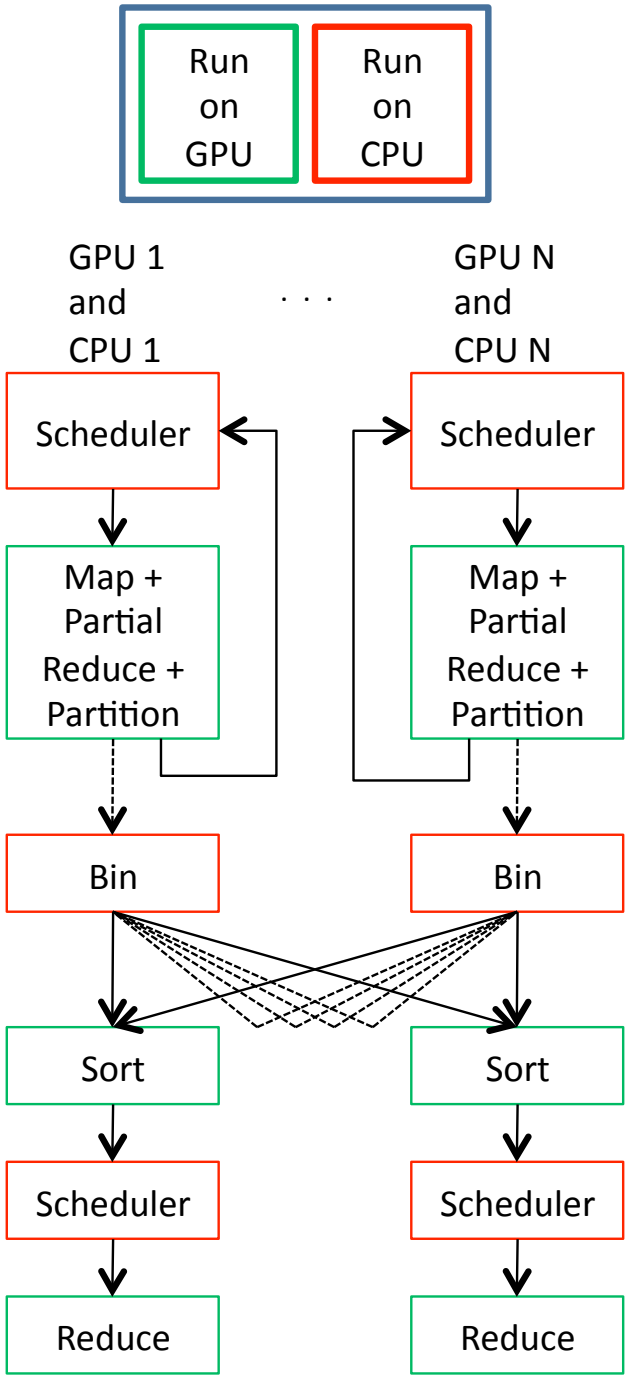
Partial Reduce

- User-specified function combines pairs with the same key
- Each map chunk spawns a partial reduction on that chunk
- Only good when cost of reduction is less than cost of transmission
- Good for ~larger number of keys
- Reduces GPU->CPU bw



Accumulate

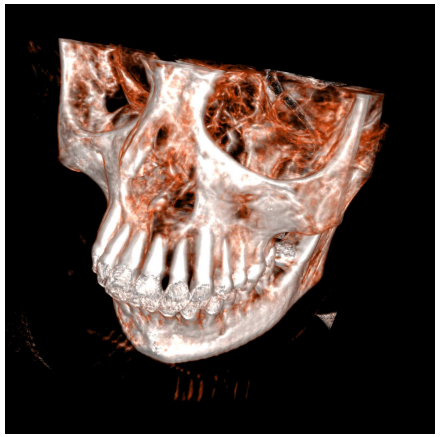
- Mapper has explicit knowledge about nature of key-value pairs
- Each map chunk accumulates its key-value outputs with GPU-resident key-value accumulated outputs
- Good for small number of keys
- Also reduces GPU->CPU bw



Benchmarks—Which

- Matrix Multiplication (MM)
- Word Occurrence (WO)
- Sparse-Integer Occurrence (SIO)
- Linear Regression (LR)
- K-Means Clustering (KMC)

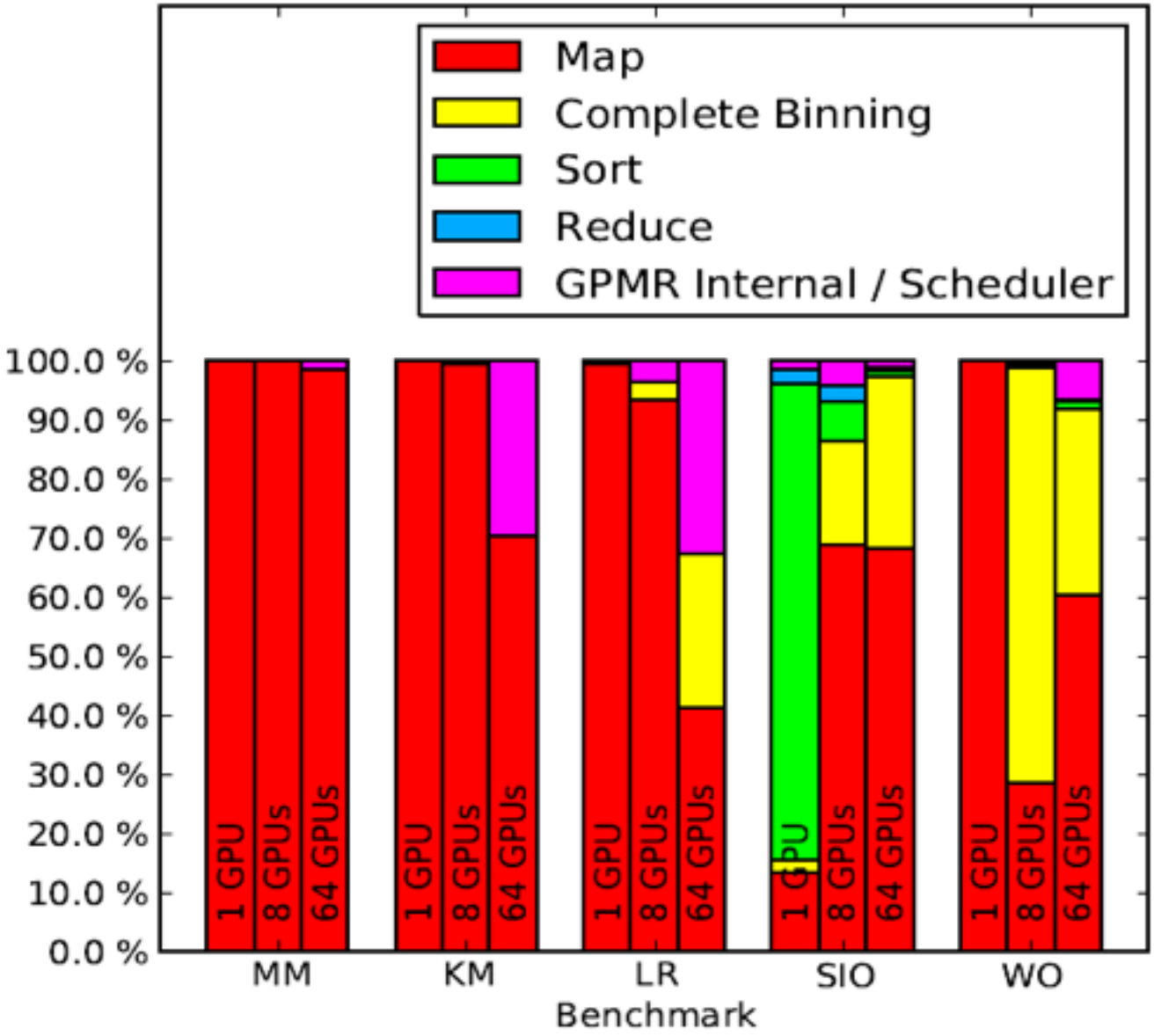
- (Volume Renderer—presented 90 minutes ago @ MapReduce '10)



Benchmarks—Why

- Needed to stress aspects of GPMR
  - Unbalanced work (WO)
  - Multiple emits/Non-uniform number of emits (LR, KMC, WO)
  - Sparsity of keys (SIO)
  - Accumulation (WO, LR, KMC)
  - Many key-value pairs (SIO)
  - Compute Bound Scalability (MM)

Benchmarks—Results



Benchmarks—Results

vs. CPU

	MM	KMC	LR	SIO	WO
1-GPU Speedup	162.712	2.991	1.296	1.450	11.080
4-GPU Speedup	559.209	11.726	4.085	2.322	18.441

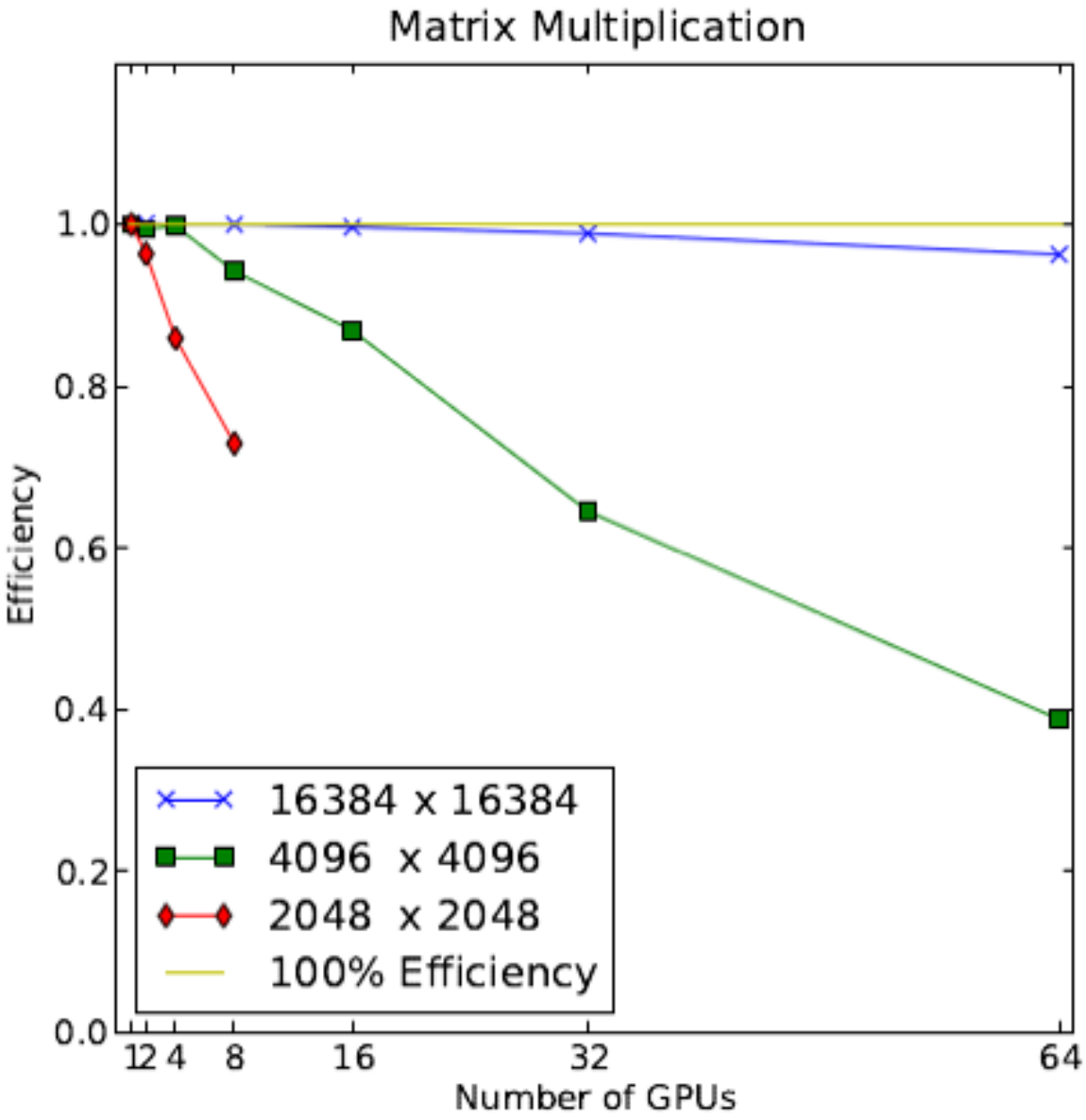
TABLE 2: Speedup for GPMR over Phoenix on our large (second-biggest) input data from our first set. The exception is MM, for which we use our small input set (Phoenix required almost twenty seconds to multiply two 1024×1024 matrices).

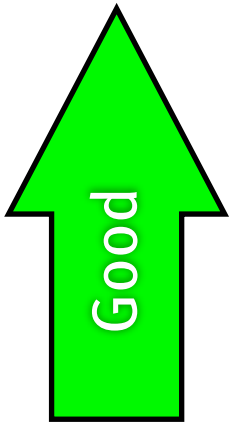
vs. GPU

	MM	KMC	WO
1-GPU Speedup	2.695	37.344	3.098
4-GPU Speedup	10.760	129.425	11.709

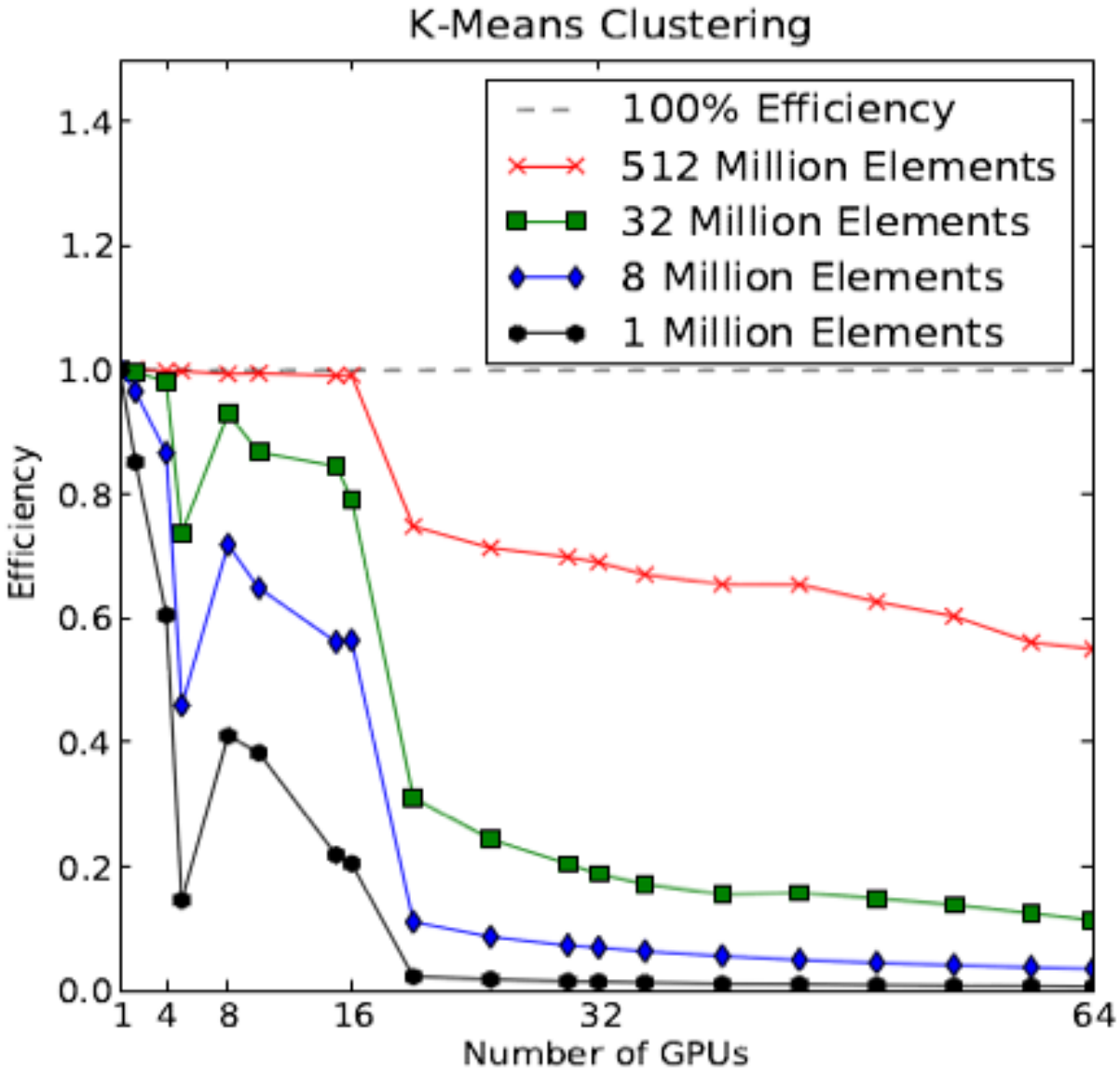
TABLE 3: Speedup for GPMR over Mars on 4096×4096 Matrix Multiplication, an 8M-point K-Means Clustering, and a 512 MB Word Occurrence. These sizes represent the largest problems that can meet the in-core memory requirements of Mars.

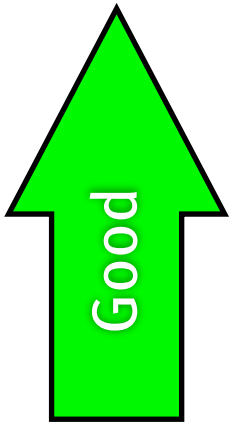
Benchmarks - Results



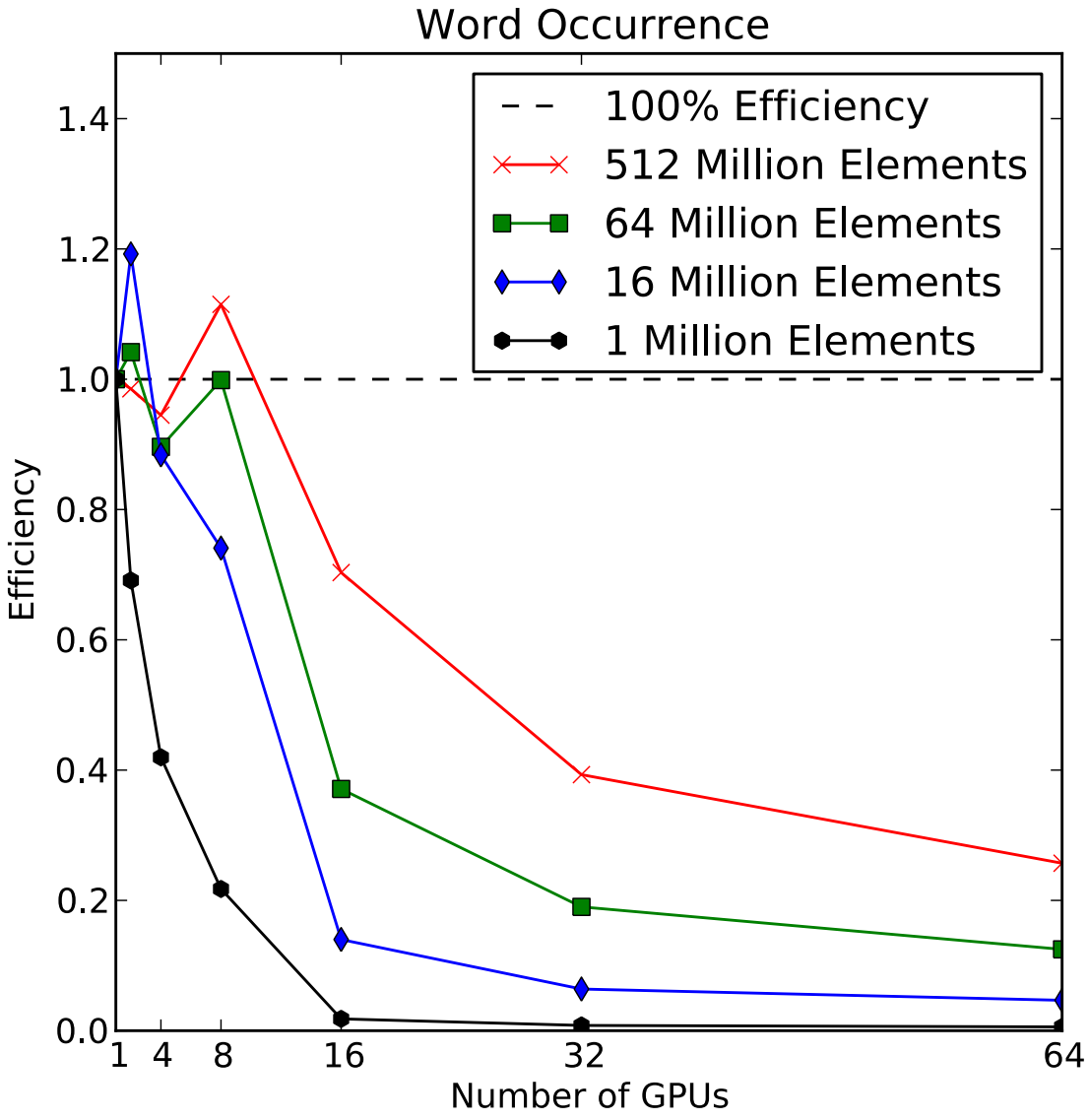


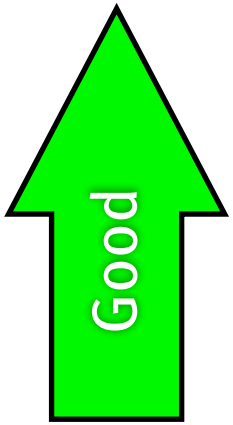
Benchmarks - Results





Benchmarks - Results





Conclusions

- Time is right to explore diverse programming models on GPUs
  - Few threads -› many, heavy threads -› light
- Lack of {bandwidth, GPU primitives for communication, access to NIC} are challenges
- What happens if GPUs get a lightweight serial processor?
- Future hybrid hardware is exciting

Thanks to …

- University of Illinois / NCSA and Argonne for cluster access
- NVIDIA for hardware donations
- Funding agencies: Department of Energy (SciDAC Institute for Ultrascale Visualization, Early Career Principal Investigator Award), NSF, LANL, BMW, NVIDIA, HP, Intel, UC MICRO, Microsoft, ChevronTexaco, Rambus