Accelerating Data Warehousing Applications Using General Purpose GPUs

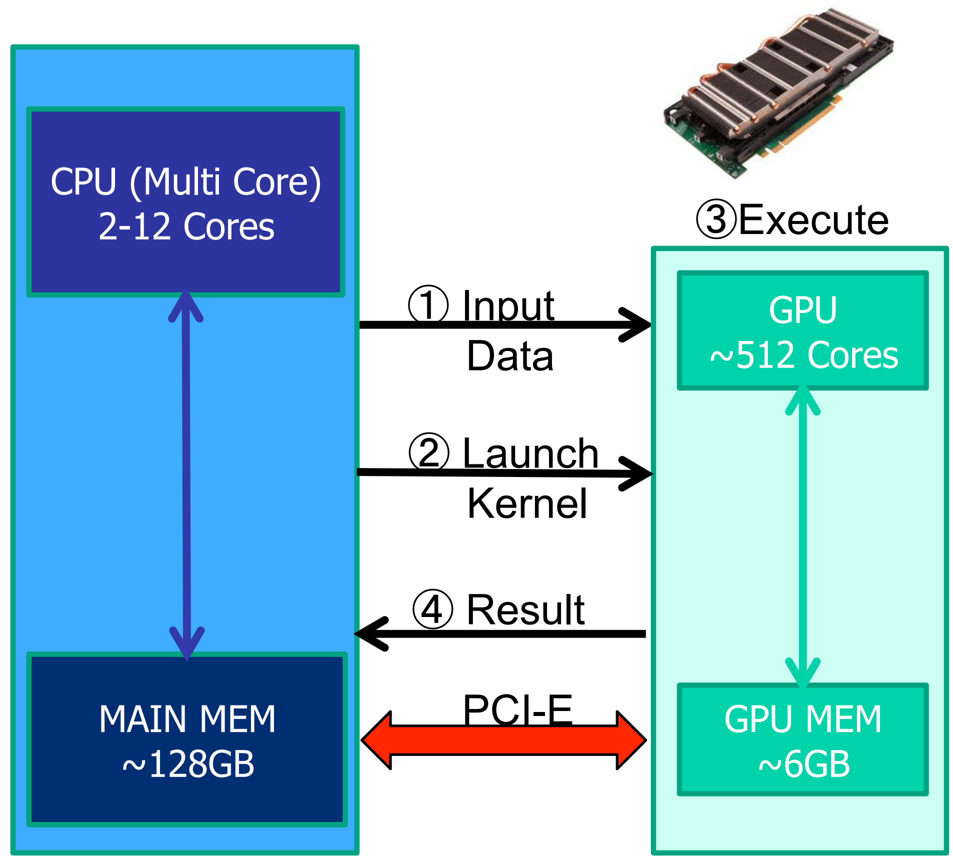
Haicheng Wu, Gregory Diamos, Jeffrey Young, and Sudhakar Yalamanchili

Computer Architecture and Systems Laboratory
Center for Experimental Research in Computer Systems
School of Electrical and Computer Engineering
Georgia Institute of Technology

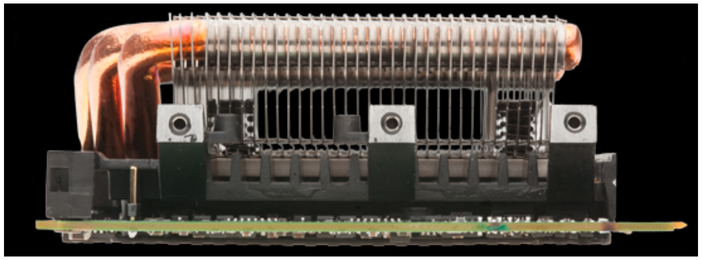
Sponsors: *National Science Foundation, LogicBlox Inc. , IBM, and NVIDIA*

The General Purpose GPU

- GPU is a many core co-processor
  - 10s to 100s of cores
  - 1000s to 10,000s of concurrent threads
  - CUDA and OpenCL are the dominant programming models
- Well suited for data parallel apps
  - Molecular Dynamics, Options Pricing, Ray Tracing, etc.
- Commodity: led by NVIDIA, AMD, and Intel



Enterprise: Amazon EC2 GPU Instance



NVIDIA Tesla

Amazon EC2 GPU Instances

Elements	Characteristics
OS	CentOS 5.5
CPU	2 x Intel Xeon X5570 (quad-core "Nehalem" arch, 2.93GHz)
GPU	2 x NVIDIA Tesla "Fermi" M2050 GPU Nvidia GPU driver and CUDA toolkit 3.1
Memory	22 GB
Storage	1690 GB
I/O	10 GigE
Price	$2.10/hour

Data Warehousing Applications on GPUs

- The good
  - Lots of potential data parallelism
  - If data fits in GPU mem, 2x—27x speedup has been shown
- The bad
  - Very large data set (will not even fit in host memory)
  - I/O bound (GPU has no disk)
  - PCI data transfer takes 15–90% of the total time*

Order	Price	Discount
0	10	10%
1	20	20%
2	10	15%
3	51	14%
4	33	13%
5	22	10%
……	……	……

* B. He, M. Lu, K. Yang, R. Fang, N. K. Govindaraju, Q. Luo, and P. V. Sander. Relational query co-processing on graphics processors. In TODS, 2009.

This Work

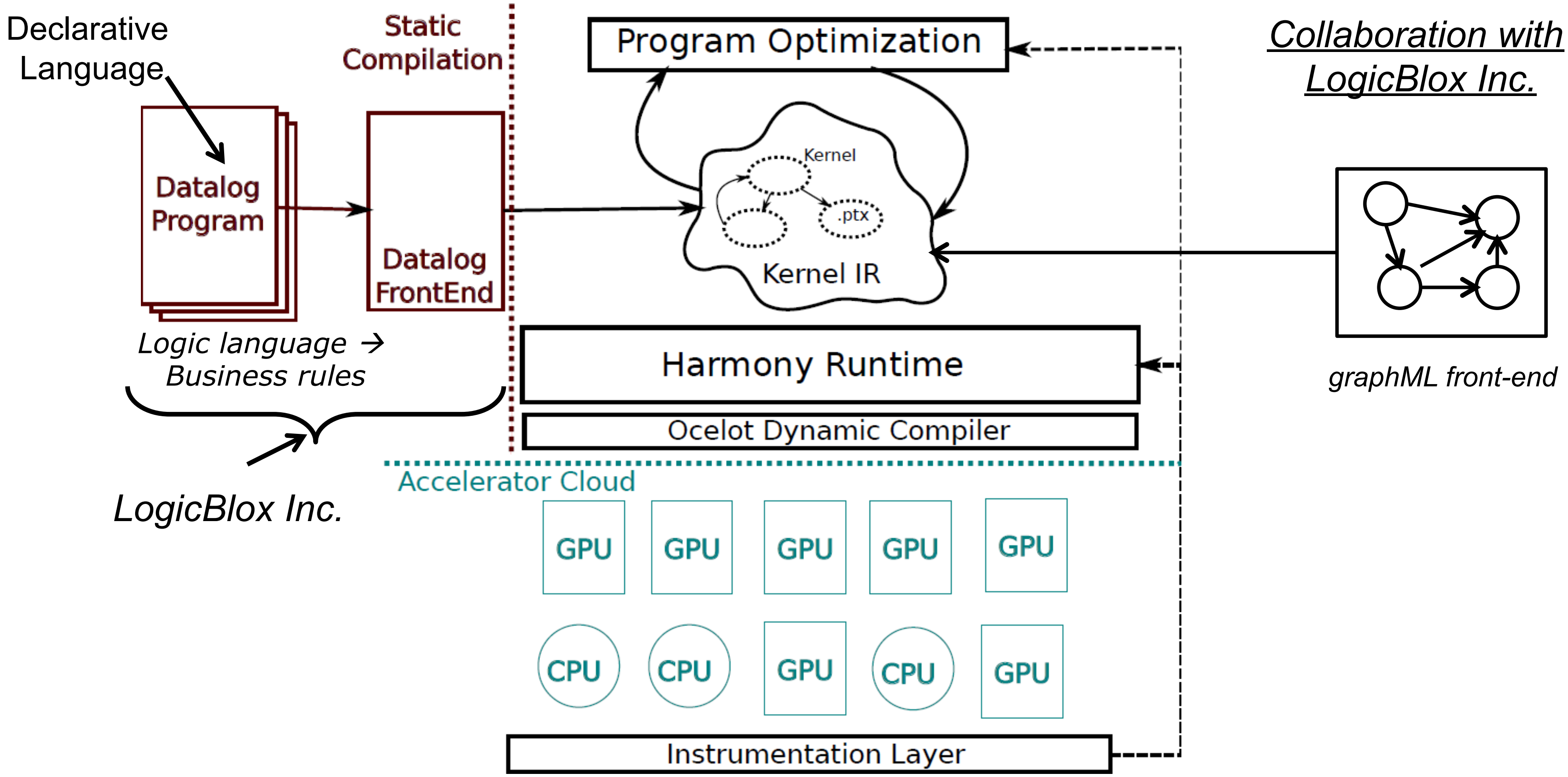
- Goal: Enable Large data warehousing applications on GPUs

- Assumptions
  - In-memory system
    - Host memory, not GPU memory
  - Not OLTP (*Online Transaction Processing*) type simple queries
    - Focus on data analysis instead of data entry/retrieval

Research Thrusts

- I: Optimized implementations of primitives
  - Relational algebra (RA)
  - Data management within the GPU memory hierarchy
- II: Data movement optimizations
  - Between host and accelerators
  - Within an accelerator
- III: In-core processing
  - Cluster wide memory aggregation techniques
  - Change the ratio of host memory size to accelerator memory size

Red Fox: Execution Environment for the Enterprise



- Bridge the x86-based Database Enterprise platform and Database backend with NVIDIA accelerators
- 10x-100x targeted improvement in application speedup

Thrust I: Optimized Primitives

- Optimized implementation of each relational algebra (RA) operator
  - Synthesized from micro-primitives
  - Implemented as a CUDA/PTX kernel template and available as a library
- The Redfox compiler synthesizes an application by instantiating templated skeletons of these primitives
  - Provides a framework for optimizations (e.g. kernel fusion)

Relational Algebra Operators in GPU

Operator	NVIDIA C2050	Phenom 9570	Speedup
Inner join	26.4-32.3 GB/s	0.11-0.63 GB/s	> 42x
Select	104.2 GB/s	2.55 GB/s	41x
Set operators	45.8 GB/s	0.72 GB/s	64x
Projection	54.3 GB/s	2.34 GB/s	23x
Cross product	98.8 GB/s	2.67 GB/s	37x

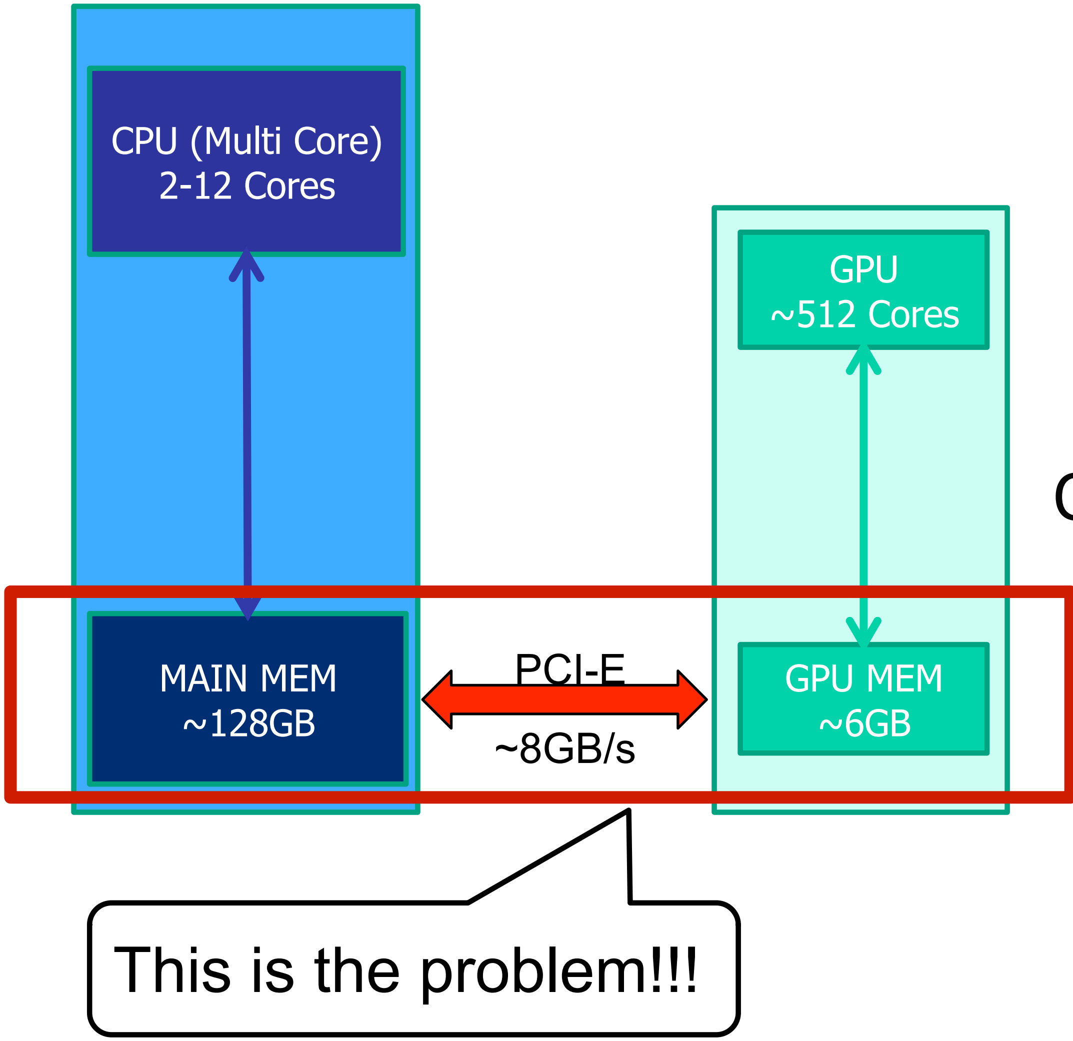
- 10 Datalog microbenchmarks running
- Metrics based on random data sets, compressed rows and 16M tuple relations
- Cost of initial sort not included

Status

- Moving Red Fox to the Amazon EC2
- Robustness extensions across
  - Scale and size of tables
  - Size and diversity of data types
- Performance extensions
  - Single node and multi-node implementations
    - Ocelot remote device interface
    - Using multiple GPU configurations

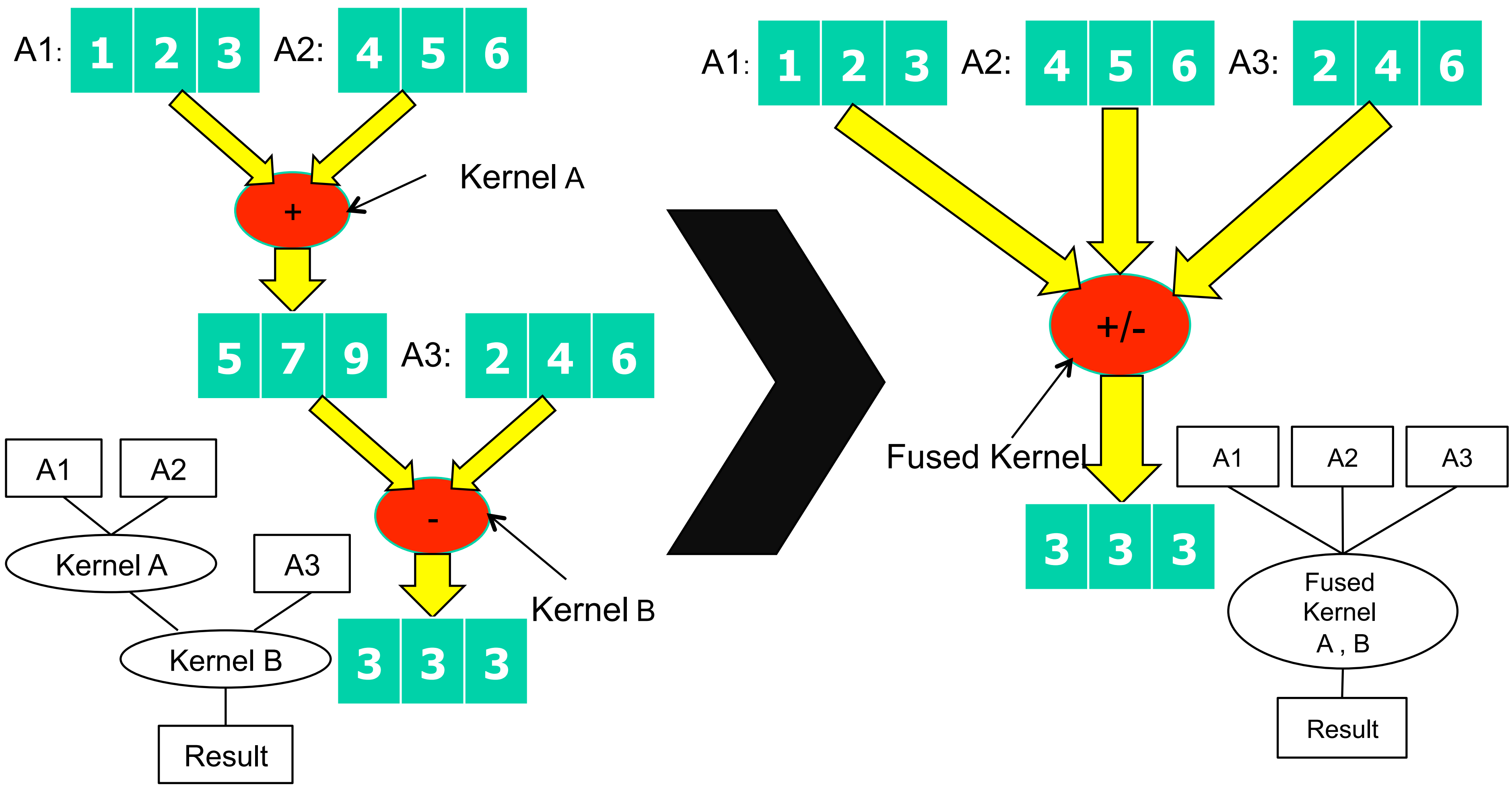
Thrust II: Optimization of Data Movement

Collaboration with NEC Inc.



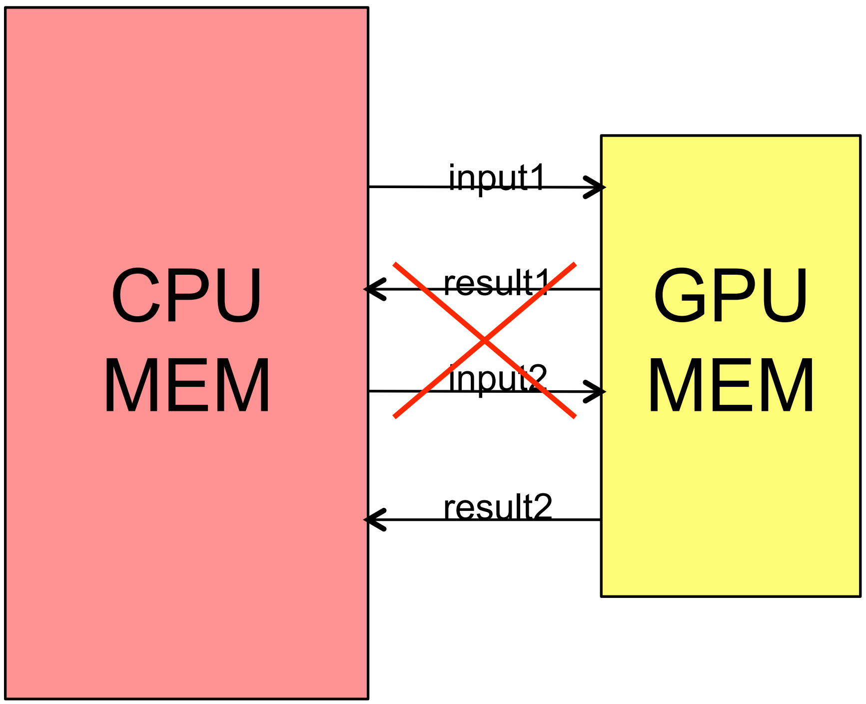
Our solution is Kernel Fusion

Kernel Fusion

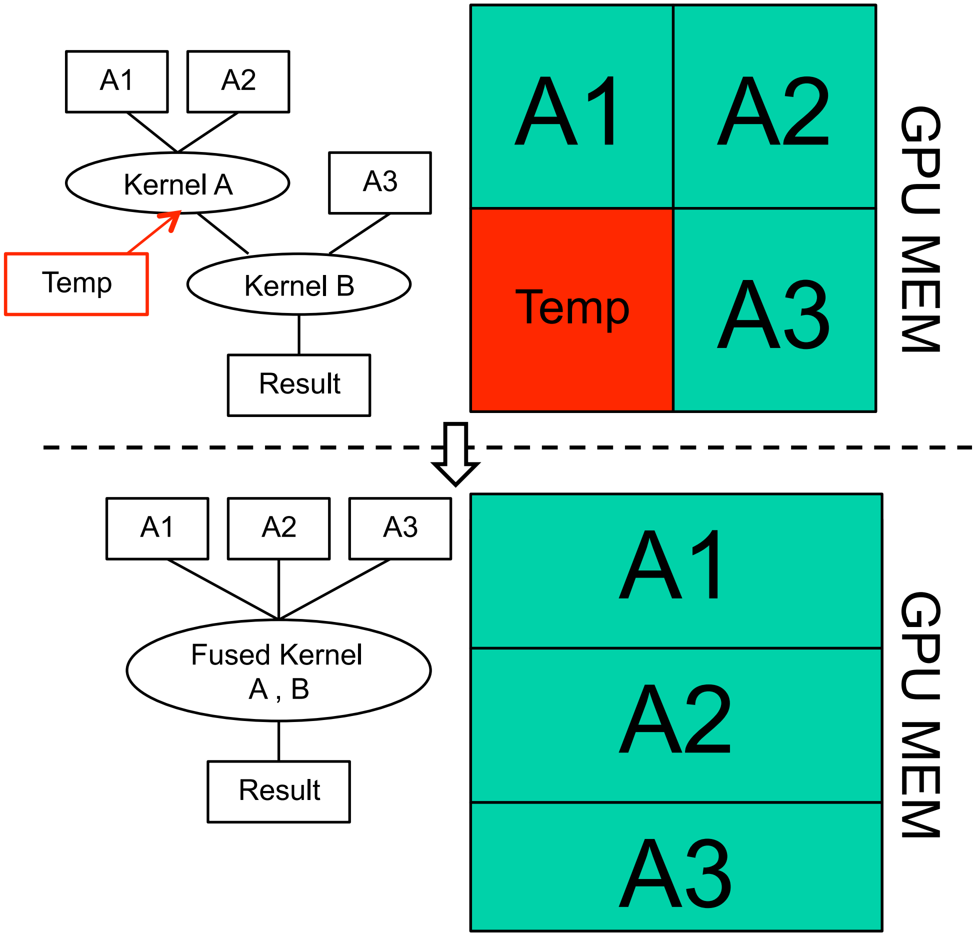


Benefits of Kernel Fusion-1

Reduce Data Transfer

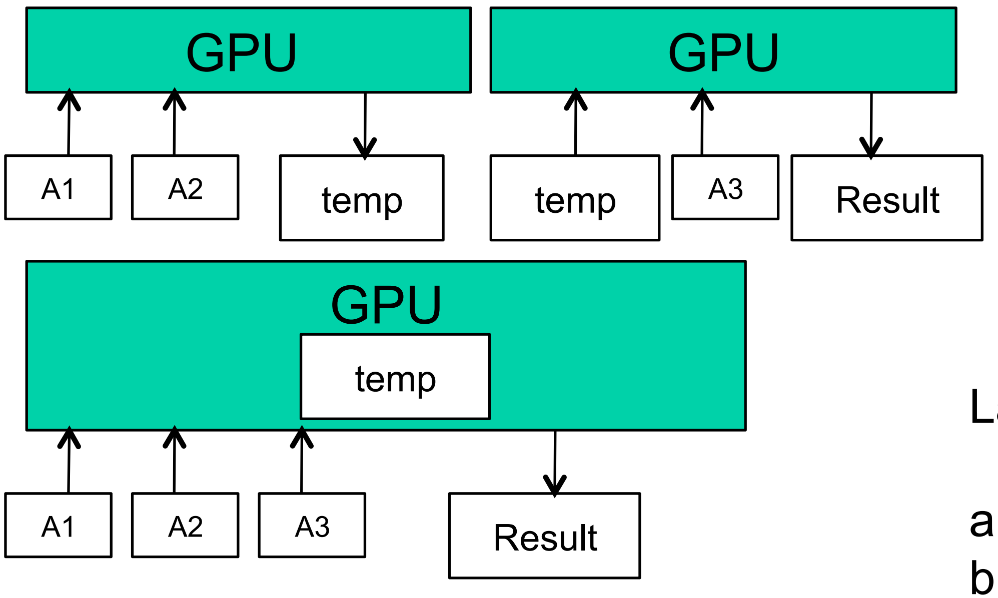


Reduce Temp Storage



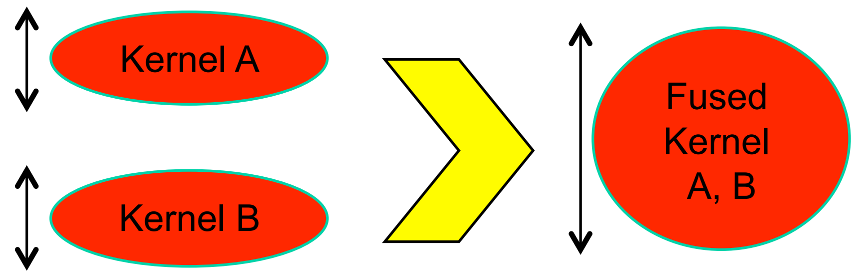
Benefits of Kernel Fusion-2

Faster Computation



Traverse the data only ONCE

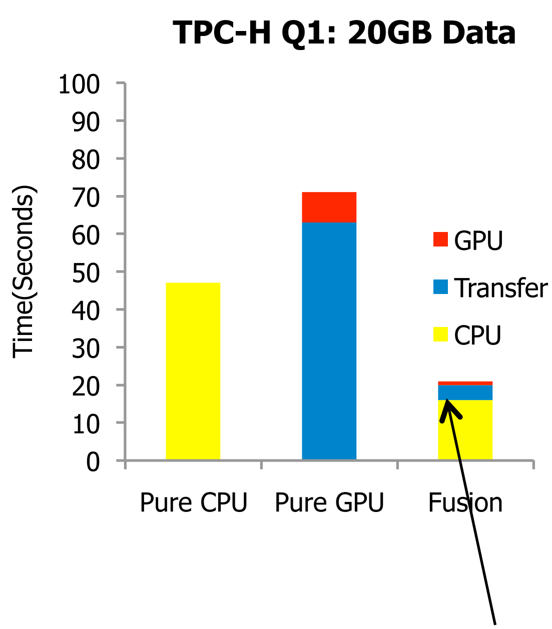
Enable More Optimization

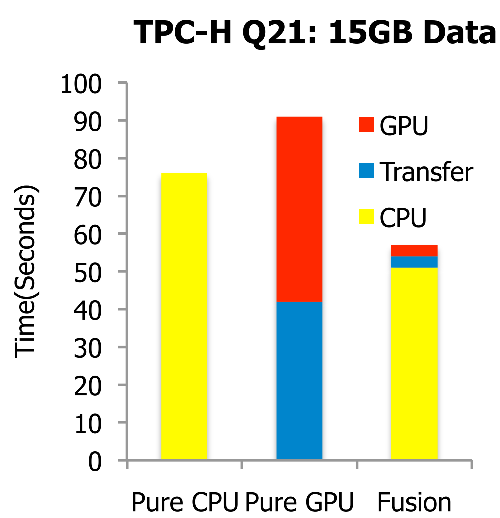


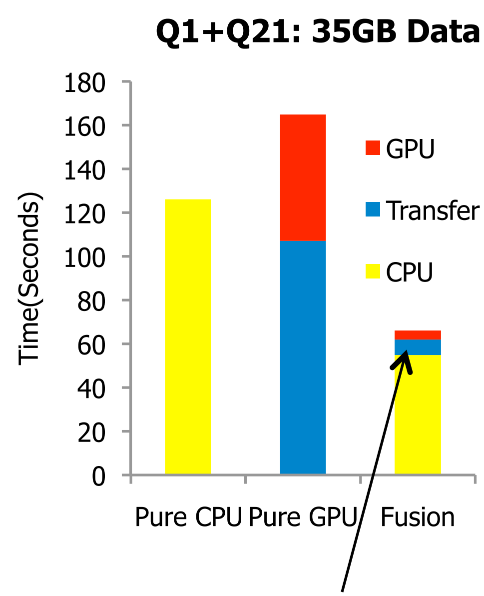
Larger code is good for other optimizations:

a) instruction scheduling,
b) register assignment,
c) constant propagation
……

Preliminary Result (2 Quad-Core CPU, C2070 GPU)



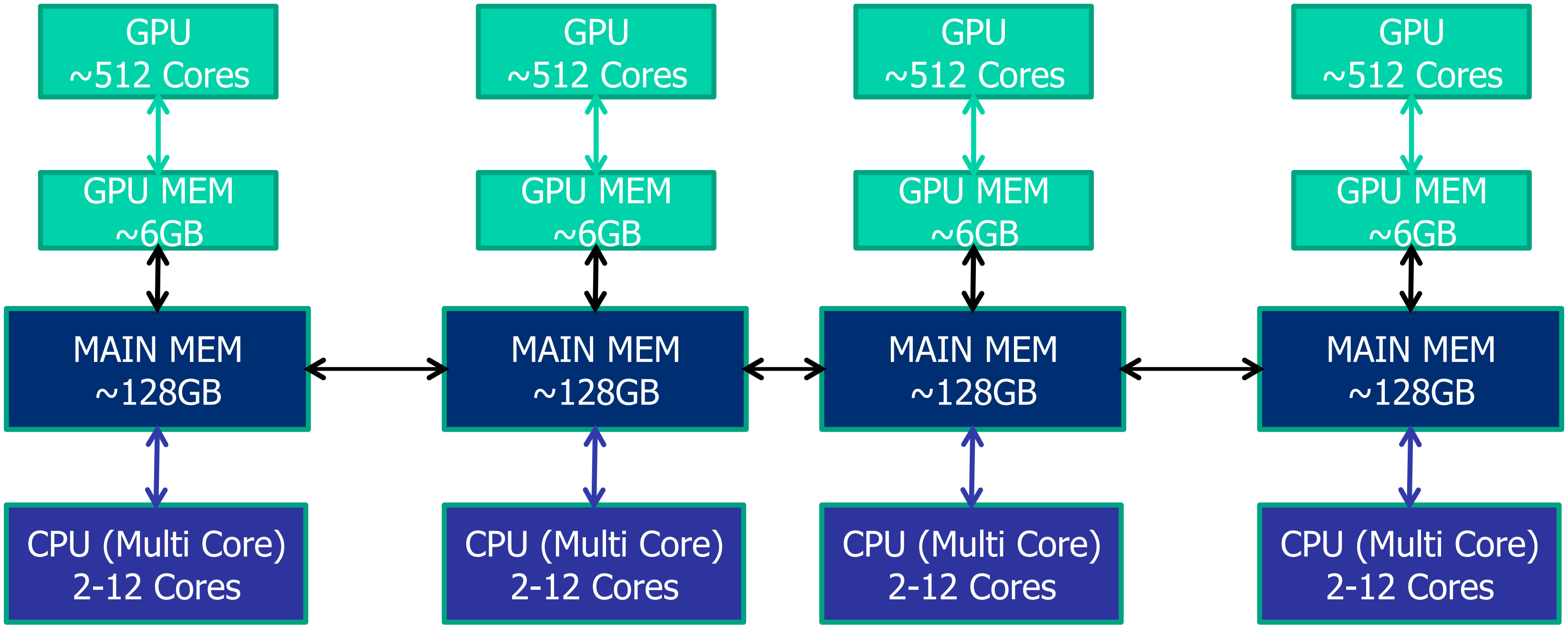




Fused across Queries

- Part of the query is run on CPU
- Transfer and GPU Computation time is much smaller

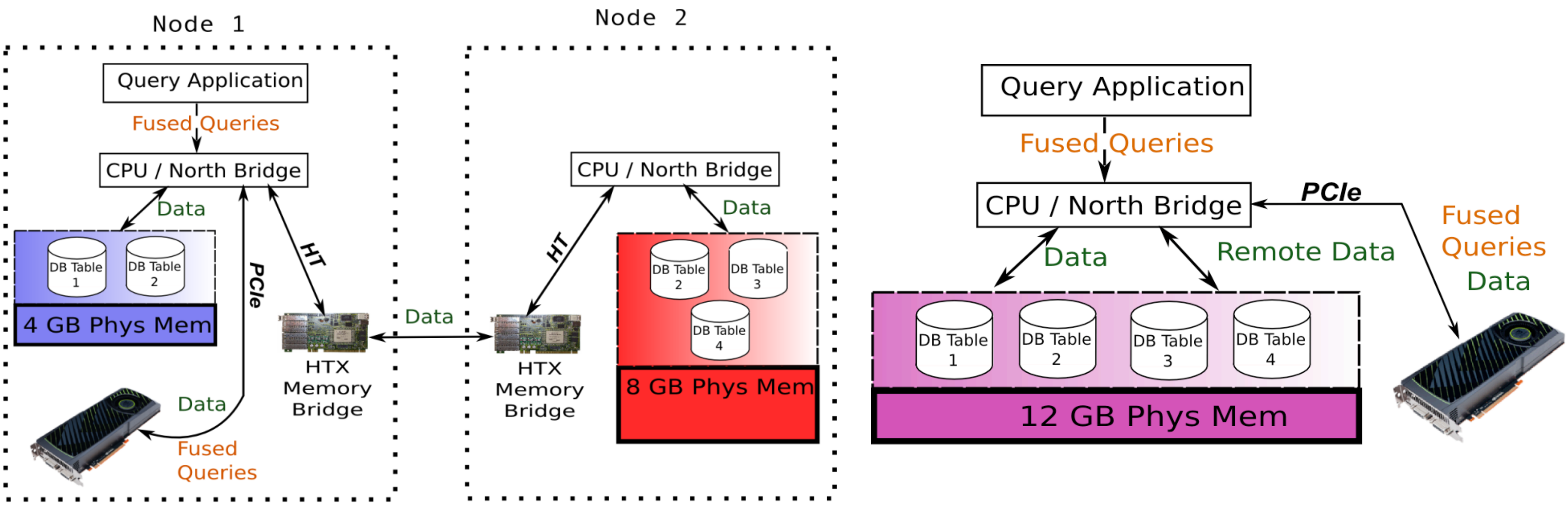
Thrust III: Cluster-based Memory Aggregation



- Hardware support for global non-coherent, physical address space system
- Change the ratio of *host-memory* : *GPU-memory*

Global Address Space Support for In-Core Databases

Collaboration with AIC Inc. & University of Heidelberg



- Use of low-latency, commodity network (HyperTransport) allows global, non-coherent access to remote memory
- Query app sees one large database / host memory from the application level
- Global address support can be extended in the future to support GPU memory
  - Applications could remotely read/write a remote GPU's memory without needing to involve its OS or CPU

Conclusions

- Fast GPU implementations of RA operators provide opportunity to run large data warehousing applications on GPU.

- Data movement optimization (Kernel Fusion) saves the memory transfer time and speeds up the computation time.

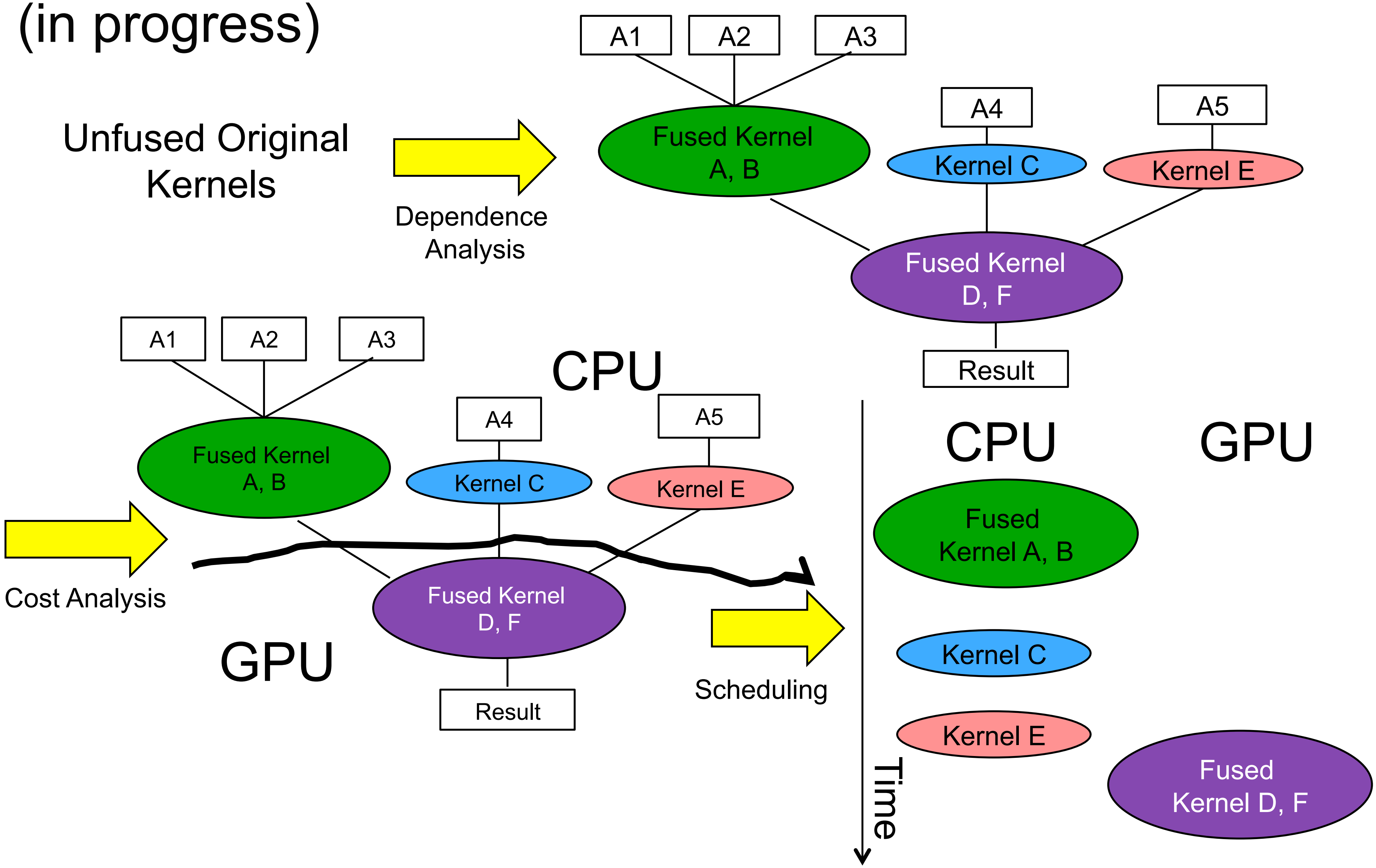
- New Memory Hierarchy (GAS) offers a larger logical memory for GPU database system.

Thank You

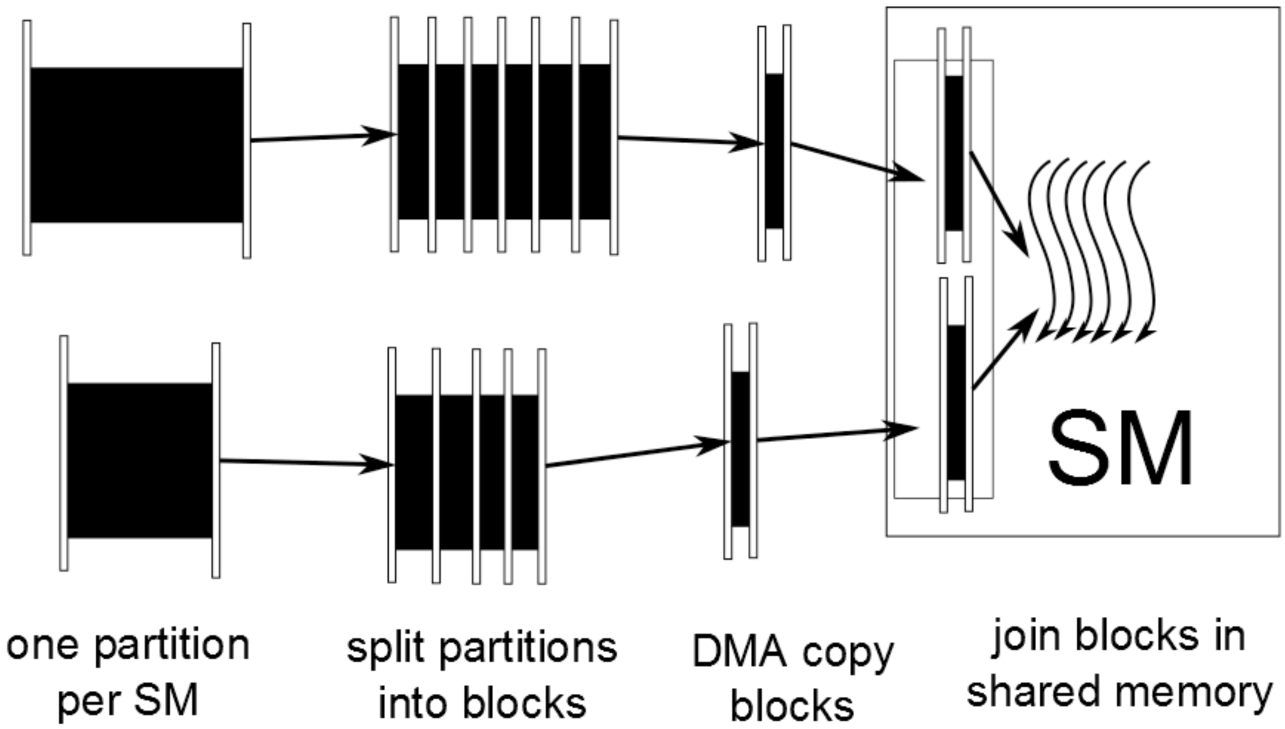
Questions?

Backup

Efficient Data Movement – Intelligent Scheduling (in progress)



Inner Join



Blocking into pages, shared memory buers, and transaction sized chunks makes memory accesses ecient.