Red Fox: An Execution Environment for Relational Query Processing on GPUs

Haicheng Wu[1], Gregory Diamos[2], Tim Sheard[3], Molham Aref[4], Sean Baxter[2], Michael Garland[2], Sudhakar Yalamanchili[1]

1. Georgia Institute of Technology
2. NVIDIA
3. Portland State University
4. LogicBlox Inc.

System Diversity Today

Amazon EC2 GPU Instances

Mobile Platforms (DSP, GPUs)

Hardware Diversity is Mainstream

Keeneland System (GPUs)

Cray Titan (GPUs)

Relational Queries on Modern GPUs

- The Opportunity
  - Significant potential data parallelism
  - If data fits in GPU memory, 2x—27x speedup has been shown [1]

- The Problem
  - Need to process 1-50 TBs of data[2]
  - Fine grained computation
  - 15–90% of the total time spent in moving data between CPU and GPU [1]

[1] 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.

[2] Independent Oracle Users Group. A New Dimension to Data Warehousing: 2011 *IOUG Data Warehousing Survey*.

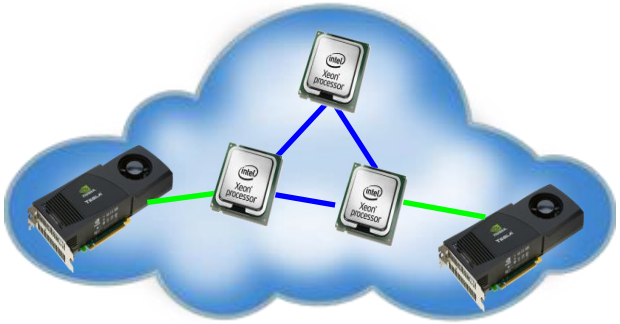
The Challenge

Candidate Application Domains

New Applications and Software Stacks

New Accelerator Architectures

```
.....
LargeQty(p) <-
 Qty(q), q > 1000.
.....
```



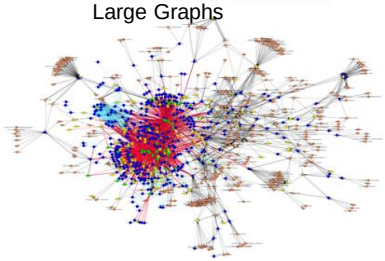
Walmart

amazon.com and you're done.

NASDAQ

facebook

Large Graphs



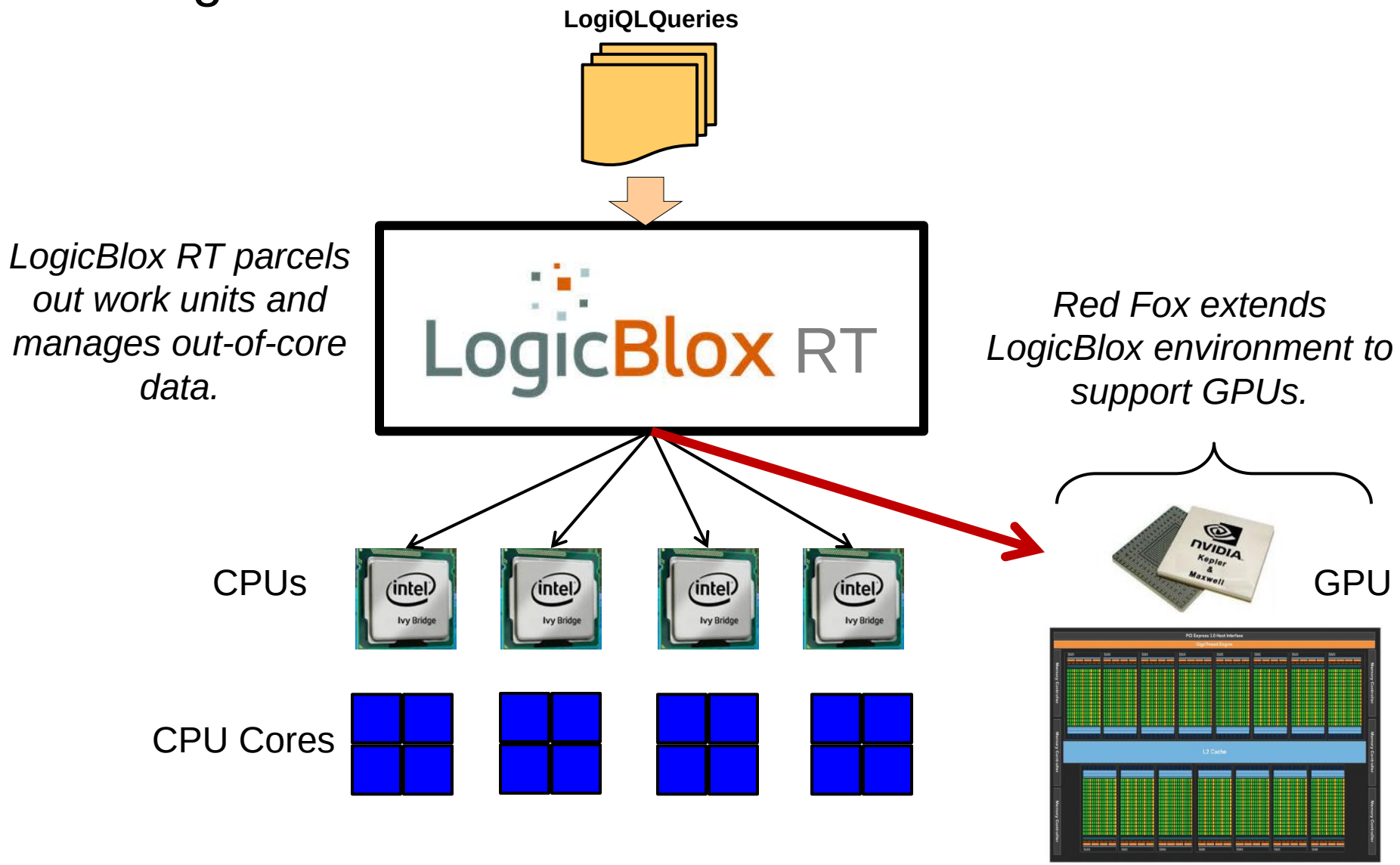
Relational Computations Over Massive Unstructured Data Sets: Sustain 10X – 100X throughput over multicore

Goal and Strategy

- GOAL
  - Build a compilation chain to bridge the semantic gap between ***Relational Queries*** and ***GPU*** execution models
    - 10x-100X speedup for relational queries over multicore

- Strategy
  1. Optimized Primitive Design
     - Fastest published GPU RA primitive implementations (PPoPP2013)
  2. Minimize Data Movement Cost (MICRO2012)
     - Between CPU and GPU
     - Between GPU Cores and GPU Memory
  3. Query level compilation and optimizations (CGO2014)

The Big Picture



LogicBlox Domain Decomposition Policy

- Sand, *Not* Boxes
  - Fitting boxes into a shipping container => hard (NP-Complete)
  - Pouring sand into a dump truck => dead easy

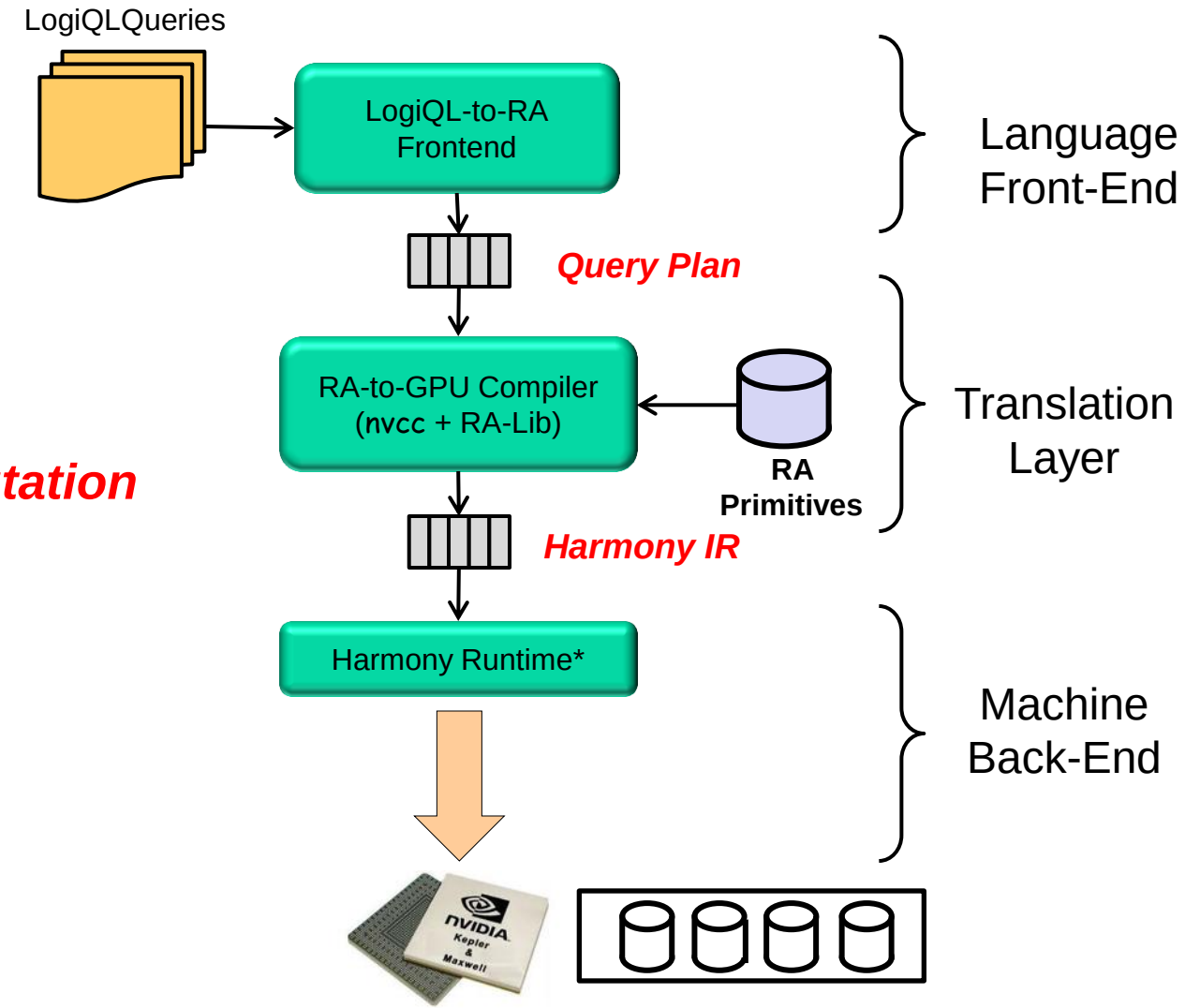
- Large query is partitioned into very fine grained work units
  - Work unit size should fit GPU memory
  - GPU work unit size will be larger than CPU size
  - Still many problems ahead, e.g. caching data in GPU

- Red Fox: Make the GPU(s) look like very high performance cores!

Domain Specific Compilation: Red Fox

First thing first,

mapping the computation
to GPU



* G. Diamos, and S. Yalamanchili, “Harmony: An Execution Model and Runtime for Heterogeneous Many-Core Processors,” *HPDC*, 2008.

Source Language: LogiQL

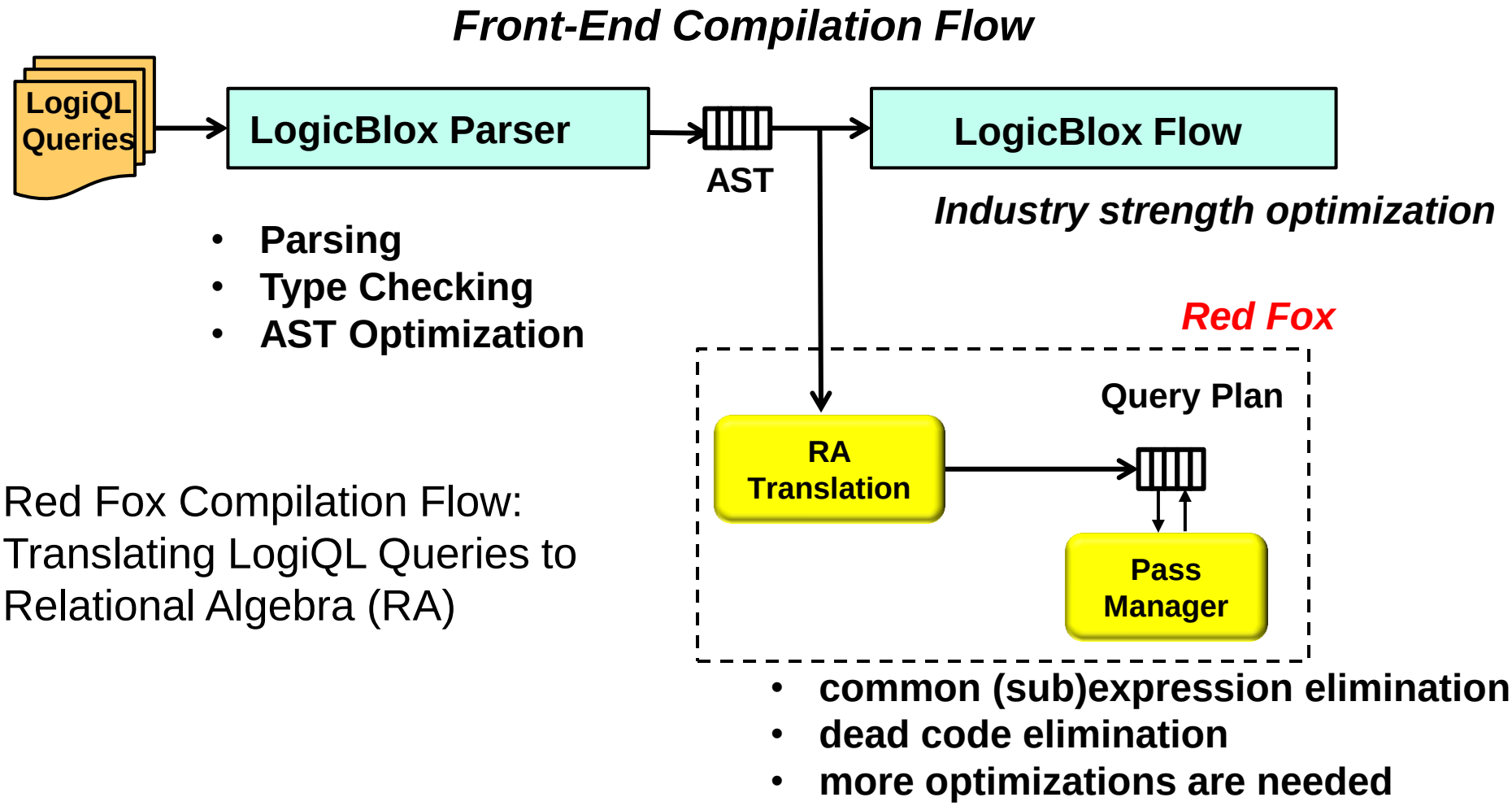
- LogiQL is based on Datalog
  - A declarative programming language
  - Extended Datalog with aggregations, arithmetic, etc.
- Find more in http://www.logicblox.com/technology.html
- Example

ancestor(x,y)<-parent(x,y).

ancestor(x,y)<-ancestor(x,t),ancestor(t,y).

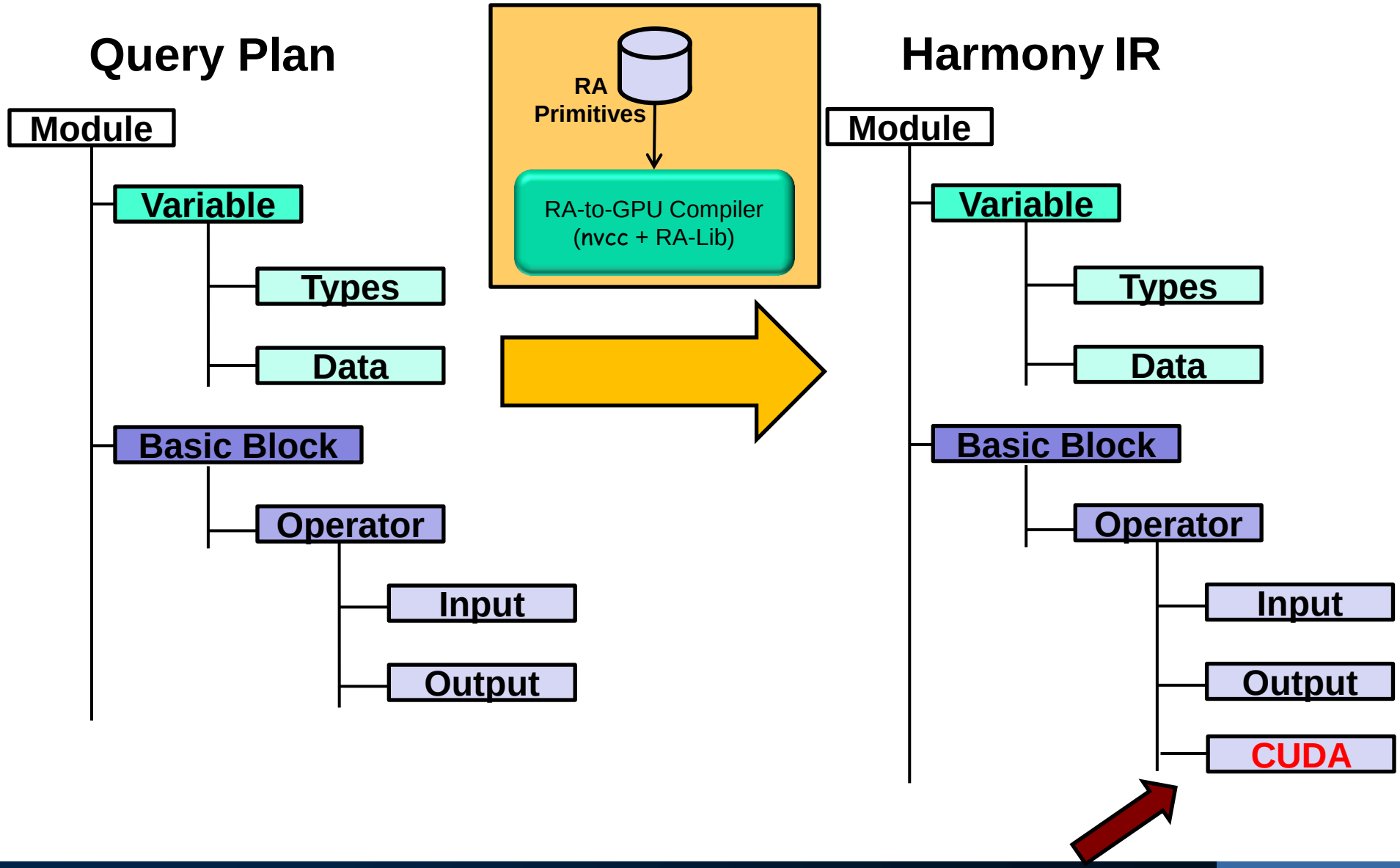
recursive definition

Language Front-end

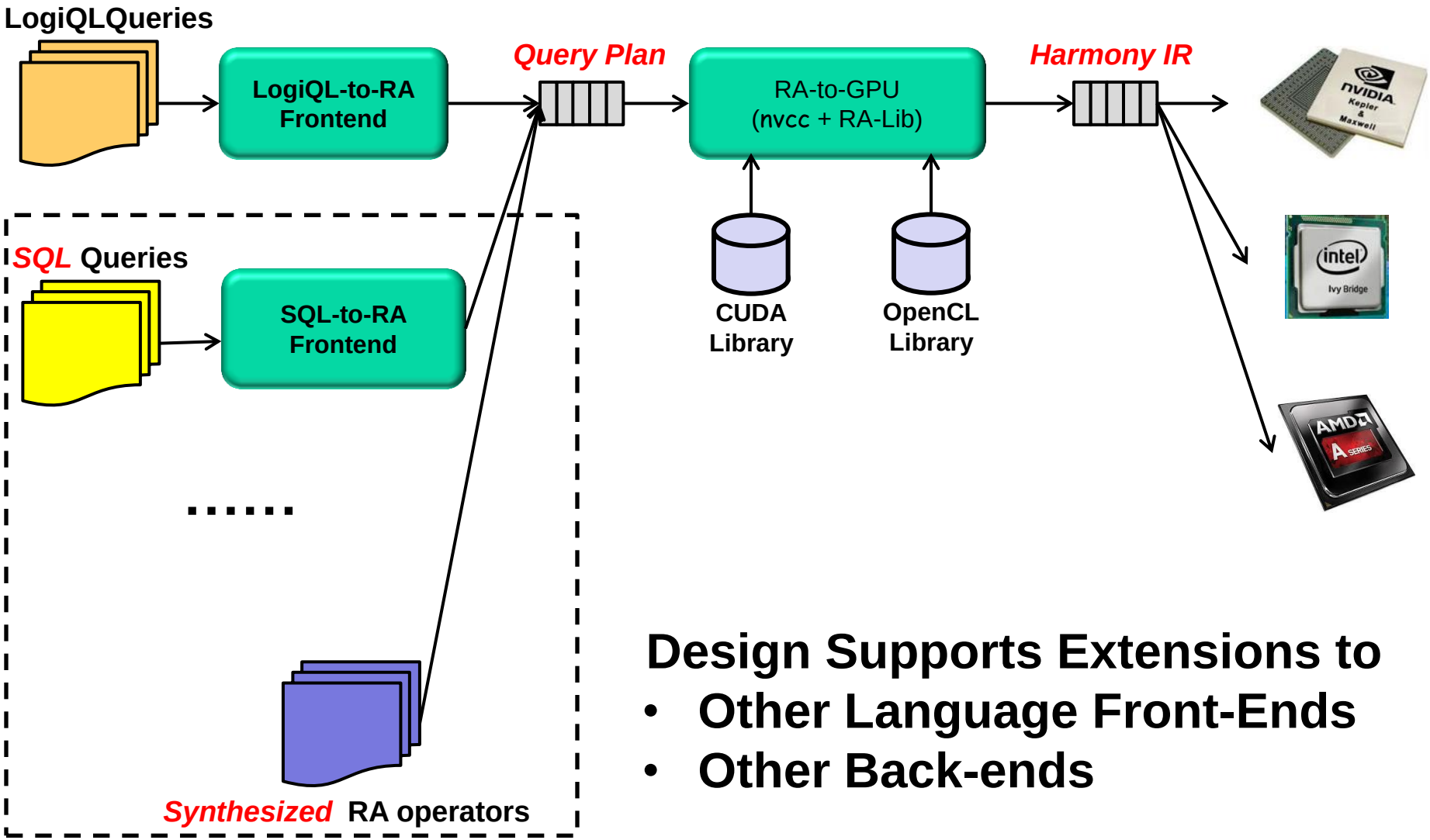


Red Fox Compilation Flow: Translating LogiQL Queries to Relational Algebra (RA)

Structure of the Two IRs:



Two IRs Enable More Choices

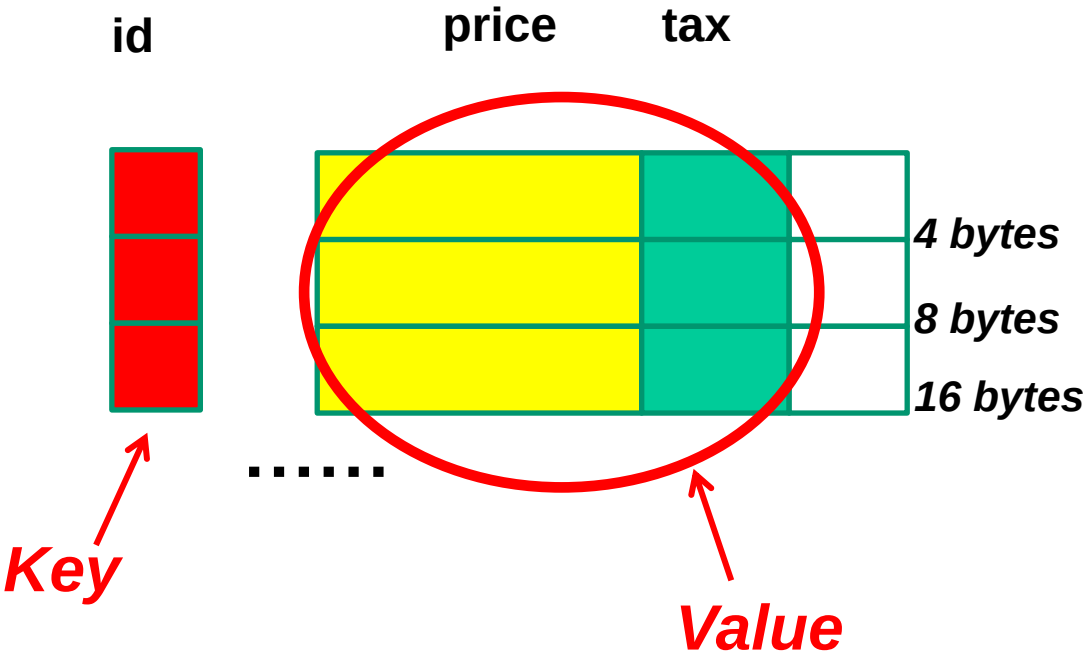


Design Supports Extensions to

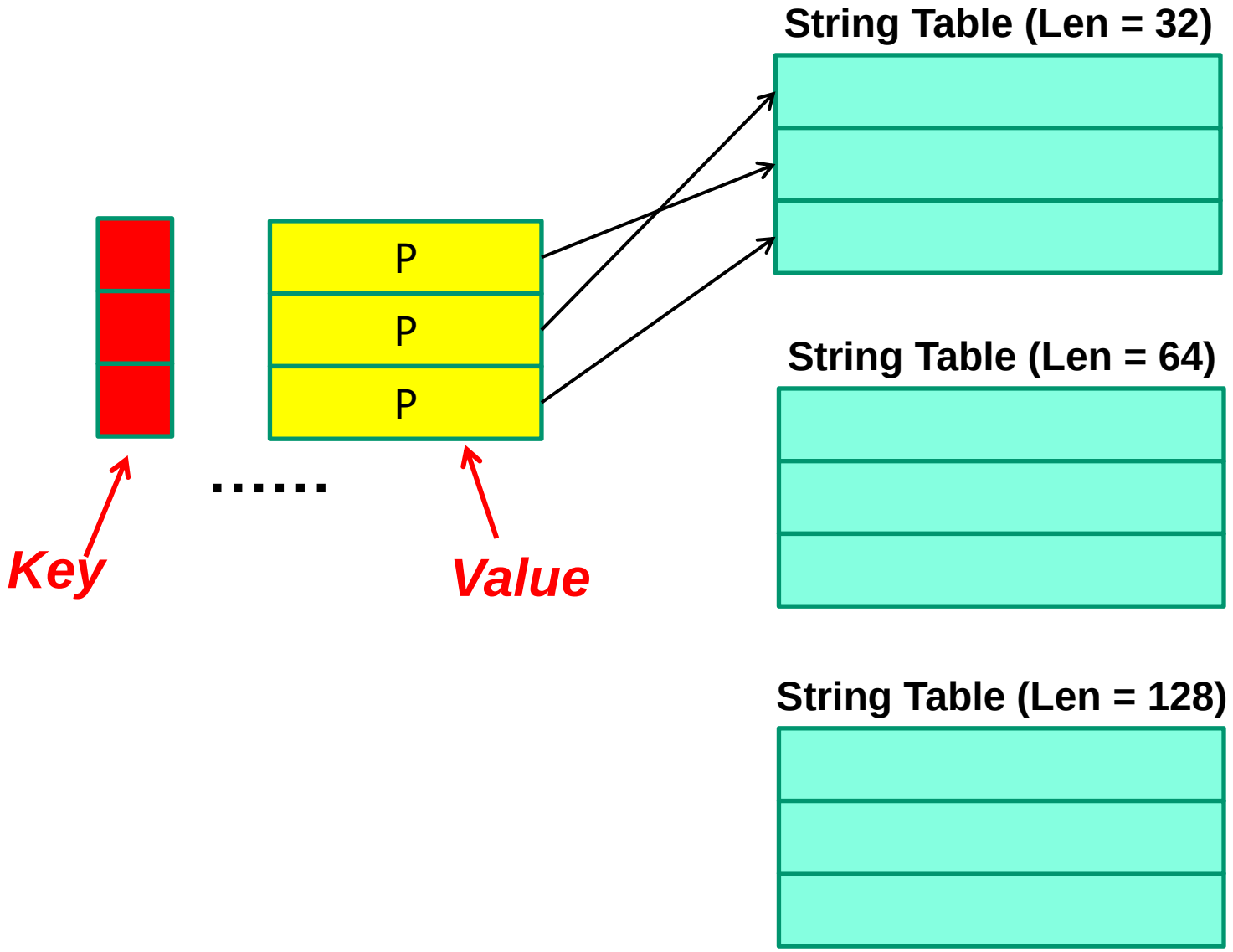
- **Other Language Front-Ends**
- **Other Back-ends**

Primitive Library: Data Structures

- **Key-Value Store**
  - Arrays of densely packed tuples
  - Support for up to 1024 bit tuples
  - Support int, float, string, date



Primitive Library: When Storing Strings



Primitive Library: Performance

Stores the GPU implementation of following primitives

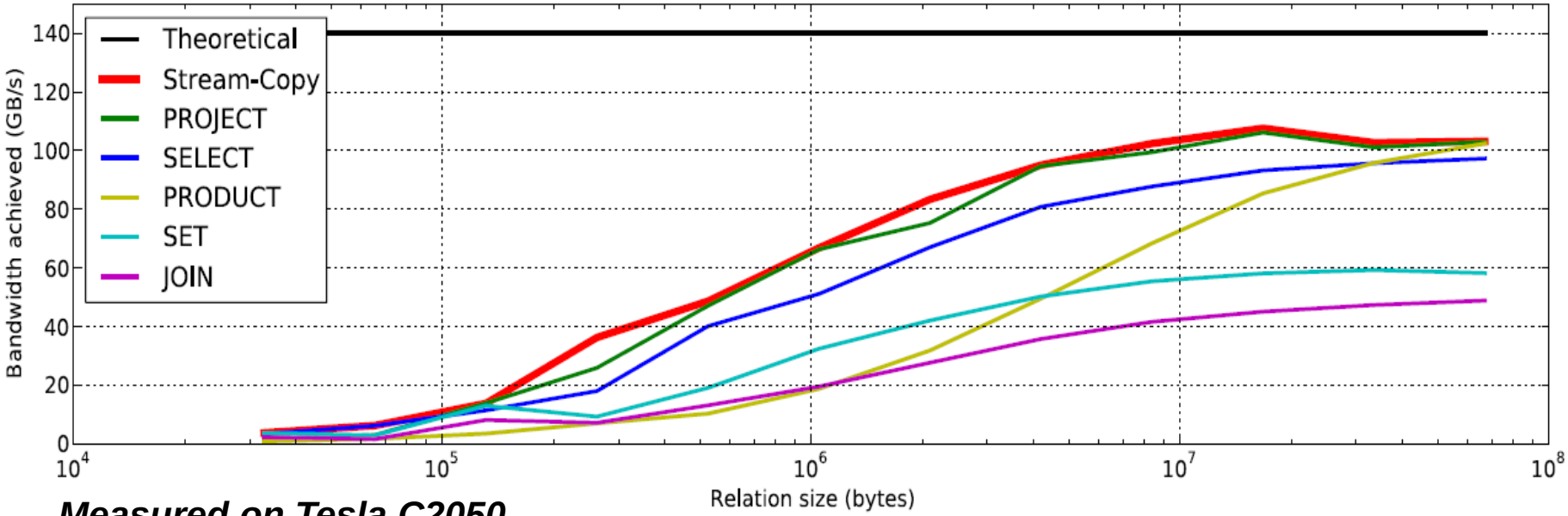
Relational Algebra
- PROJECT
- PRODUCT
- SELECT
- JOIN
- SET

Math
- Arithmetic: + - * /
- Aggregation

Built-in
- String
- Datetime

Others
- Sort
- Unique

RA performance on GPU (PPoPP 2013)*



Measured on Tesla C2050
Random Integers as inputs

* G. Diamos, H. Wu, J. Wang, A. Lele, and S. Yalamanchili. Relational Algorithms for Multi-Bulk-Synchronous Processors. In *PPoPP*, 2013.

Forward Compatibility: Primitive Library Today

- Use best implementations from the state of the art
- Easily integrate improved algorithms designed by 3rd parties

Relational Algebra
- **PROJECT**
- **PRODUCT**
- **SELECT**
- **JOIN**
- **SET**

Math
- **Arithmetic: + - * /**
- **Aggregation**

Built-in
- **String**
- **Datetime**

Others
- **Merge Sort**
- **Radix Sort**
- **Unique**

Red: Thrust library
Green: ModernGPU library[1]
- ***Merge Sort***
- ***Sort-Merge Join***

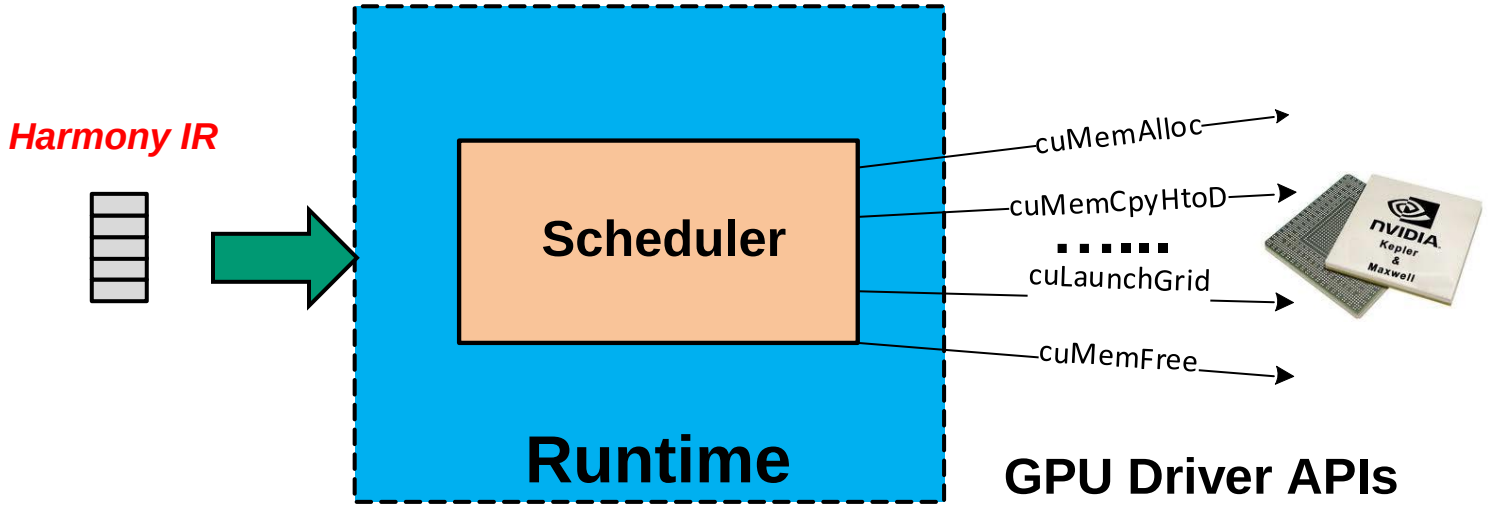
Purple: Back40Computing[2]
Black: Red Fox Library

[1]S. Baxter. Modern GPU, http://nvlabs.github.io/moderngpu/index.html

[2]D. Merrill. Back40Computing, https://code.google.com/p/back40computing/

Harmony Runtime

Schedule GPU Commands on available GPUs



Current scheduling method attempts to minimize memory footprint

Allocate j_1

j_1:=

⋮

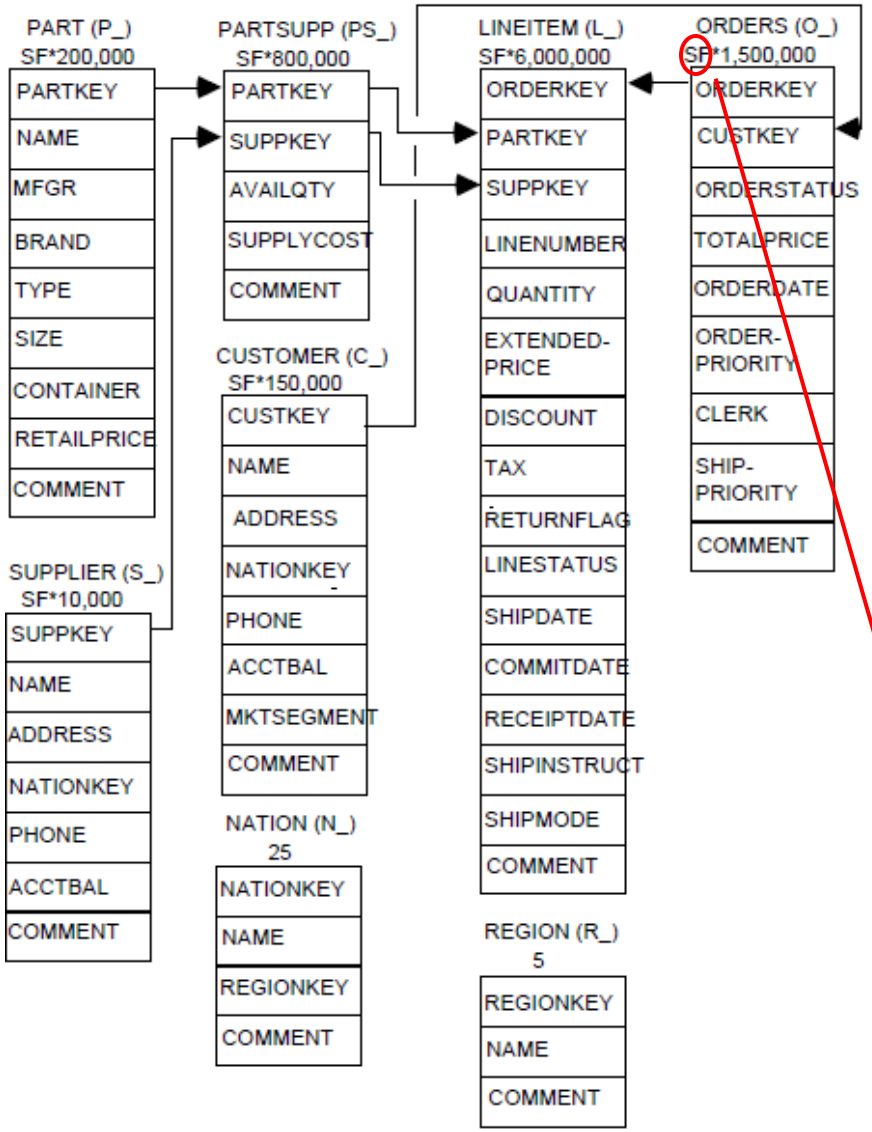
Allocate p_1
Free j_1

p_1:= PROJECT j_1

Complex Scheduling such as speculative execution is also possible*

*G. Diamos, and S. Yalamanchili, “Speculative Execution On Multi-GPU Systems,” *IPDPS*, 2010.

Benchmarks: TPC-H Queries



- A popular decision making benchmark suite
- Comprised of 22 queries analyzing data from 6 big tables and 2 small tables
- Scale Factor parameter to control database size
  - SF=1 corresponds to a 1GB database

Courtesy: O'Neil, O'Neil, Chen. Star Schema Benchmark.

Experimental Environment

Red Fox

CPU	Intel i7-4771 @ 3.50GHz
GPU	Geforce GTX Titan (2688 cores, $1000 USD)
PCIe	3.0 x 16
OS	Ubuntu 12.04
G++/GCC	4.6
NVCC	5.5
Thrust	1.7

LogicBlox 4.0

Amazon EC2 instance cr1.8xlarge

- **32 threads run on 16 cores**
- **CPU cost - $3000 USD**

Red Fox TPC-H (SF=1) Comparison with CPU

On average (geo mean)

GPU w/ PCIe : Parallel CPU = 11x

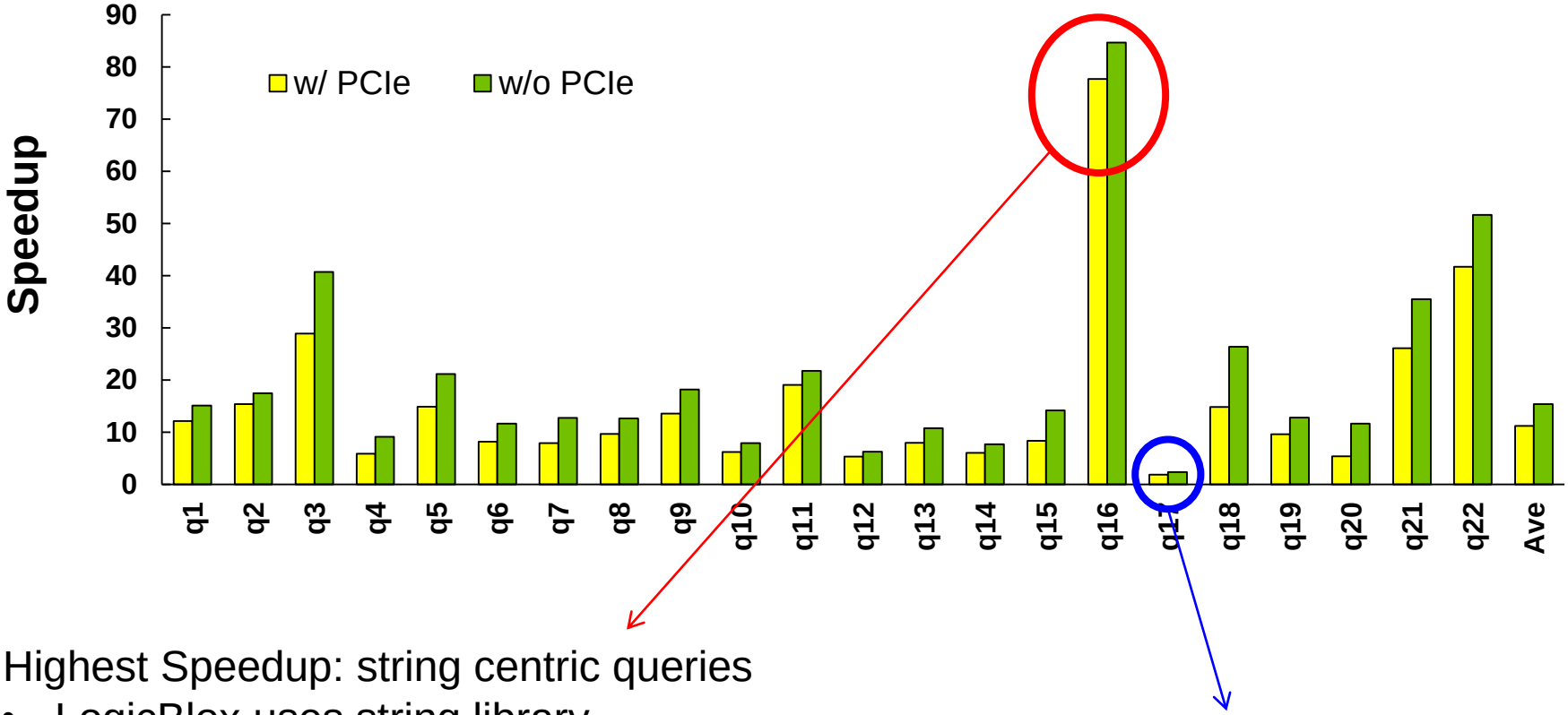
GPU w/o PCIe : Parallel CPU = 15x

This performance is viewed as *lower bound* - more improvements are coming

Find latest performance and query plans in

http://gpuocelot.gatech.edu/projects/red-fox-a-compilation-environment-for-data-warehousing/

Red Fox TPC-H (SF=1) Comparison with CPU

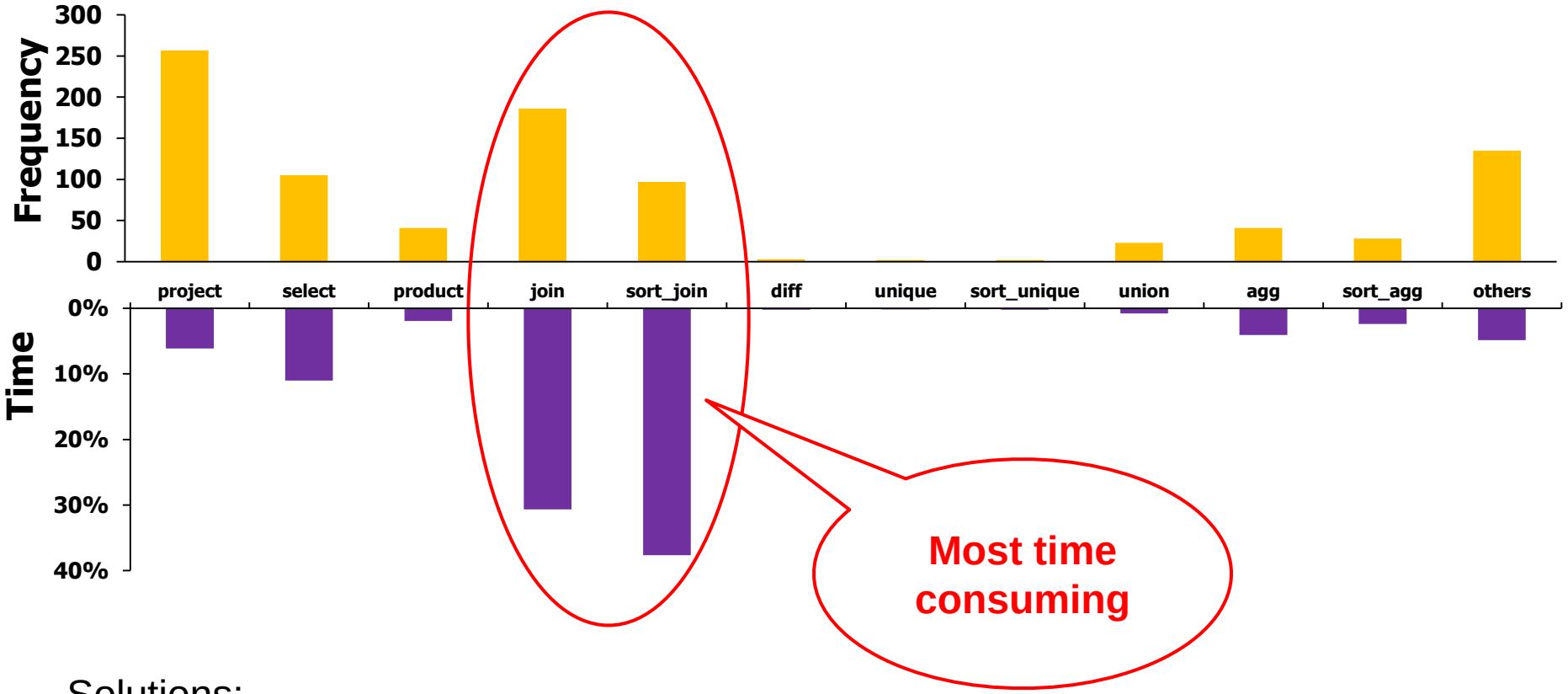


Highest Speedup: string centric queries

- LogicBlox uses string library
- Red Fox re-implements string ops
  - 1 thread manages 1 string
  - Performance depends on string contents
  - Branch/Memory Divergence

Lowest Speedup: poor query plan

Performance of Primitives



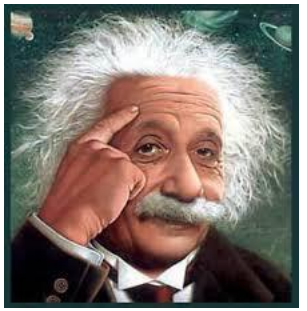
Solutions:

- Better order of primitives
- New join algorithms, e.g. hash join, multi-predicate join
- More optimizations, e.g. kernel fusion, better runtime scheduling method

Next Steps: Running Faster, Smarter, Bigger…..

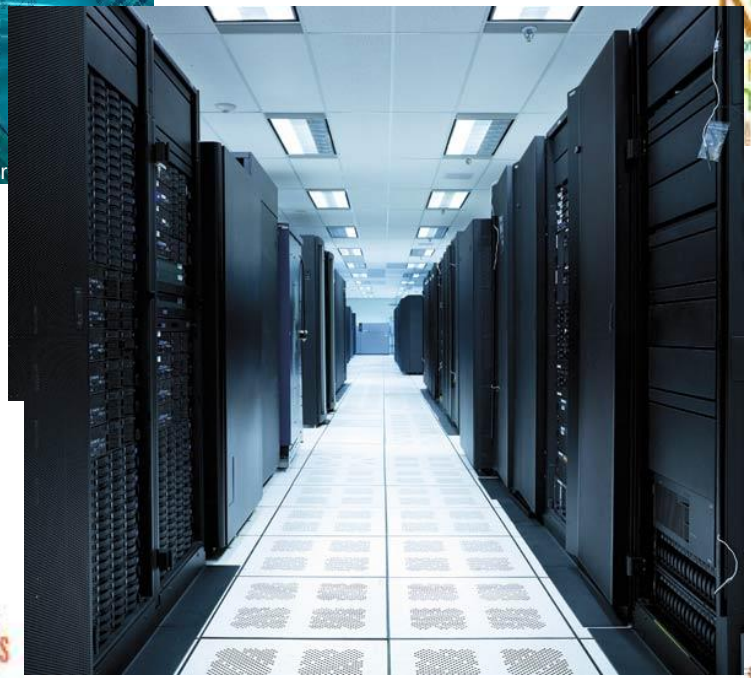
- Running Faster
  - Additional query optimizations
  - Improved RA algorithms
  - Improved run-time load distribution

- Running Smarter:
  - Extension to single node multi-GPU
  - Extension to multi-node multi-GPU

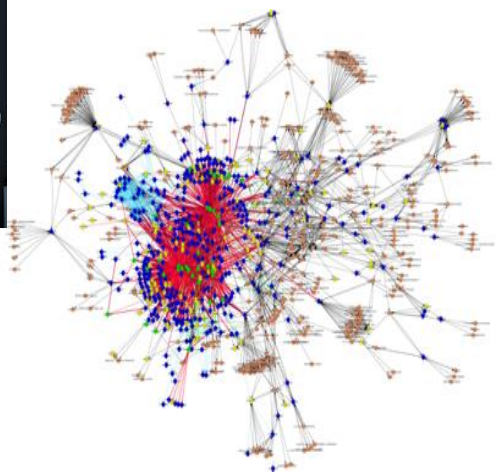


- Running Bigger
  - From in-core to out-of-core processing

The Future is Acceleration



Large Graphs



Thank You