Optimizing Performance and Productivity on Heterogeneous Processors

Sudhakar Yalamanchili
School of Electrical and Computer Engineering
Georgia Institute of Technology

With: M. Gupta, C. Kersey, H. Kim, I. Saeed, S. H. Shon, J. Young, H. Wu, and LogicBlox Inc.

http://www.istc-cc.cmu.edu/

Intel Science & Technology Center for Cloud Computing

Overview

- Drivers
- High Performance Relational Computing
- Benchmark Repository
- Near Memory Processing

2

Post-Dennard Performance Scaling

$$Perf\left(\frac{ops}{s}\right) = Power(W) \times Efficiency\left(\frac{ops}{joule}\right)$$

W. J. Dally, Keynote IITC 2012

Memory Cost + Operator_cost + Data_movement_cost

Specialization → heterogeneity and asymmetry

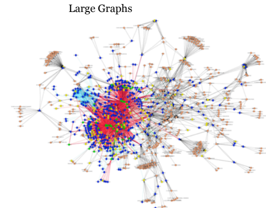
Three operands x 64 bits/operand

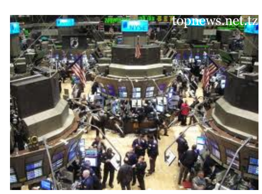
$$Energy^{*} = \#\,bits \times dist - mm \times energy - bit - mm$$

*S. Borkar and A. Chien, "The Future of Microprocessors, *CACM*, May 2011

3

A Data Rich World



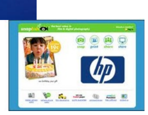


Mixed Modalities and levels of parallelism

Irregular, Unstructured Computations and Data

Pharma

conventioninsider.com

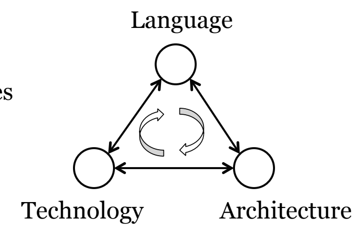


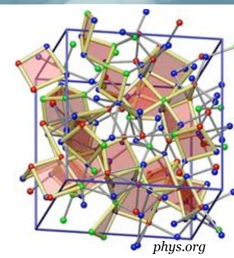
Images from math.nist.gov, blog.thefuturescompany.com,melihsozdinler.blogspot.com

Trend analysis

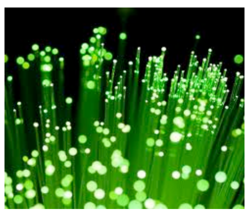
Diversity is Mainstream

Amazon EC2 GPU Instances





Phase Change Memory



Photonics

Tianhe-2

Overview

- Drivers
- High Performance Relational Computing
  - The software stack
  - Relational algebra and arithmetic kernels
  - Multi-Predicate Join
- Benchmark Repository
- Near Memory Processing

6

Relational Queries and Data Analytics

- The Opportunity
  - Significant potential data parallelism
  - High memory bandwidth and compute bandwidth of accelerators [1]
- The Problem
  - Need to process 1-50 TBs of data [2]
  - Fine grained computation
  - 15–90% of the total time spent in data movement [1] (for accelerators)

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

[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*.

Goal and Strategy

```
tr(x,y,z) ← E(x,y),E(y,z),E(x,z),x<y<z.
```

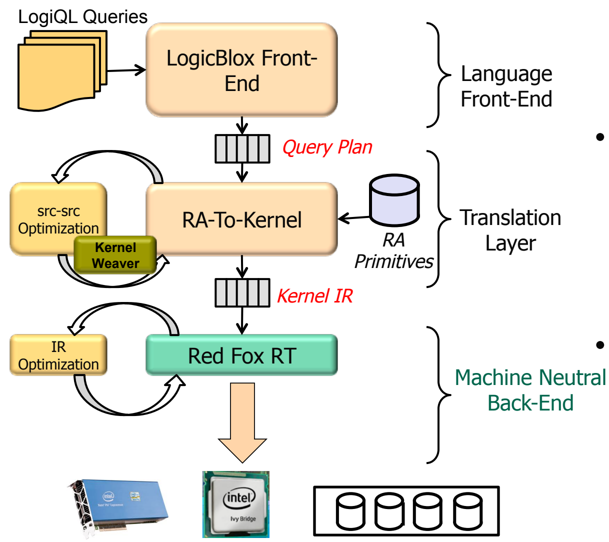
- GOAL
  - Build a compilation chain to bridge the semantic gap between ***Relational Queries*** and ***data parallel*** execution models (vector and BSP)
- Strategy

1. Optimized Primitive Design – Relational Algebra
2. Data Movement Optimizations – Compiler/Node
3. Query Level Optimization
4. Cluster-Level Data Management
5. Out-of-Core Data Management

Domain Specific Compilation: Red Fox

```
tr(x,y,z) ← E(x,y),E(y,z),E(x,z),x<y<z.
```

Joint with LogicBlox Inc.

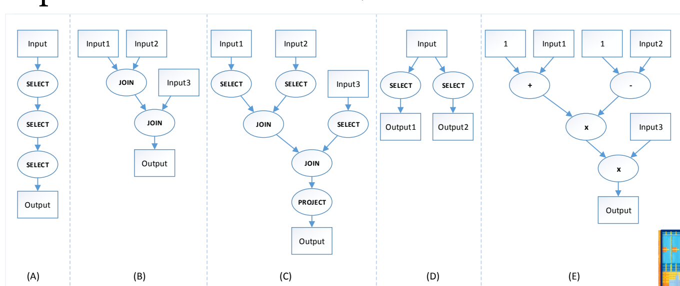


- Targeting Accelerator Clouds for meeting the demands of data warehousing applications
- In-core databases

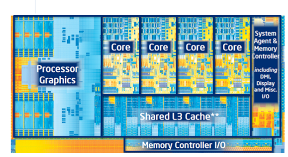
To Date

The frequently occurring patterns of operators in the TPC-H benchmark suite

Ifrah Saeed

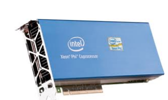


OpenCL: Multicore x86 CPUs, Gen, and Phi



Relational Algebra Operators

- PROJECT
- INNER JOIN
- SET Family (SET INTERSECTION, SET UNION, SET DIFFERENCE)
- REDUCE BY KEY
- SORT
- SELECT
- CROSS PRODUCT
- REDUCE
- UNIQUE



Multi-Predicate Join Algorithm

$tr(x,y,z) \leftarrow E(x,y), E(y,z), E(x,z), x<y<z.$

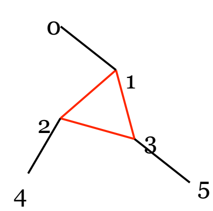
- Goal: Implementation of Leapfrog Triejoin (LFTJ) on GPU
  - A worst-case optimal multi-predicate join algorithm
  - Details (e.g., complexity analysis) in T. L. Veldhuizen, *ICDT 2014*
- Benefits
  - Smaller memory footprint and data movement
  - No data reorganization (e.g. sorting or rebuilding hash table) after changing join key
- Approach
  - CPU version
  - CPU-Friendly GPU version
  - Customized GPU version

Example: Graphs as Relations

Haicheng Wu

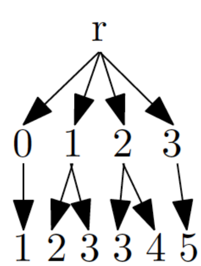
- Finding cliques
  - triangle(x,y,z)<-E(x,y),E(y,z),E(x,z), x<y<z.
  - 4cl(x,y,z,w)<-E(x,y),E(x,z),E(x,w),E(y,z),E(y,w),E(z,w), x<y<z<w.

Multi-predicate Join



Edge:

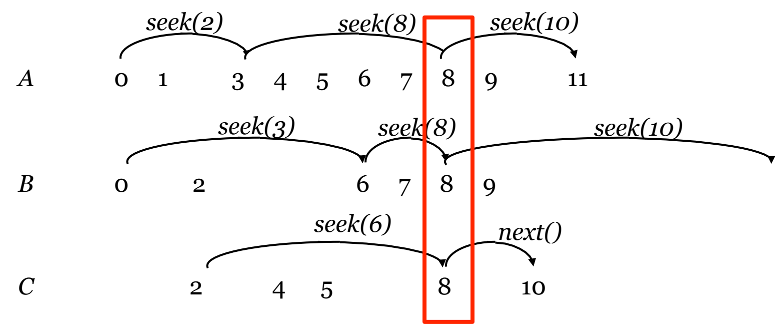
From	To
0	1
1	2
1	3
2	3
2	4
3	5



H. Wu, D. Zinn, M. Aref, and S. Yalamanchili, "Multipredicate Join Algorithms for Accelerating Relational Graph Processing on GPUs," *Proceedings of ADMS*, September 2014

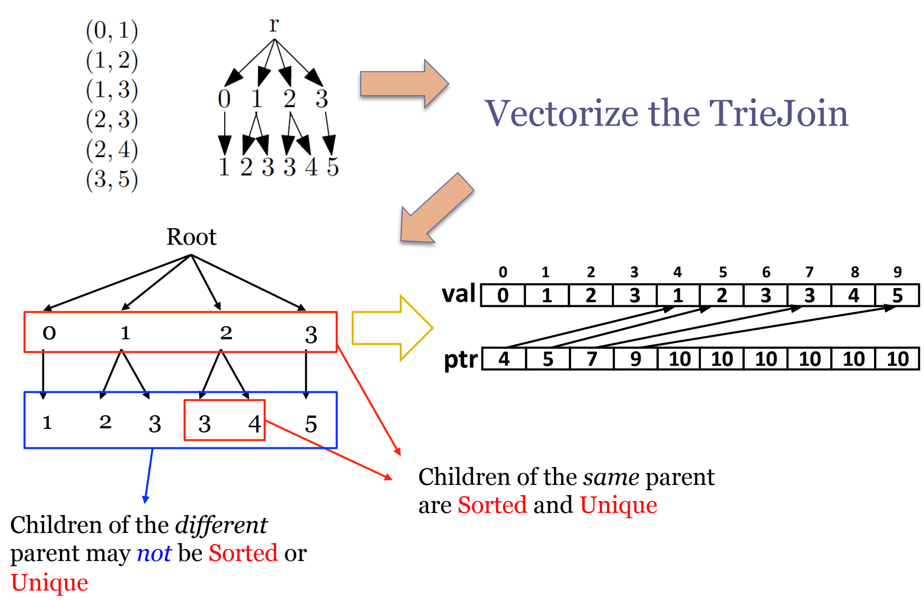
Leapfrog Triejoin: Principle

- Example on unary relations
- Essentially multi-way-intersections
- Basic primitives: *seek()*, *next()*



T. Veldhuizen, "Leapfrog Triejoin: A Simple, Worst-Case Optimal Join Algorithm," *ICDT 2014*

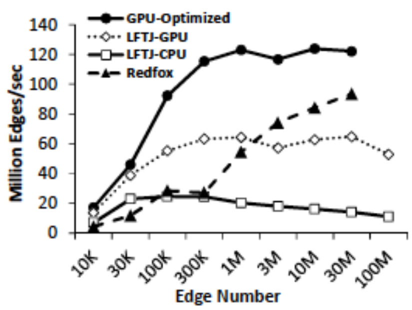
Extensions to General Relations

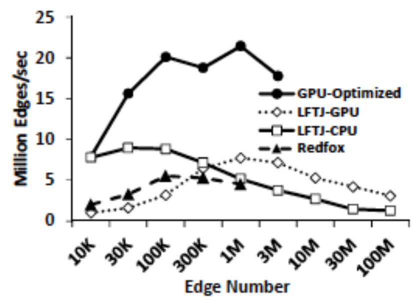


Performance

- Collaboration with LogicBlox Inc.
- CPU-Friendly GPU version
- Customized GPU Version

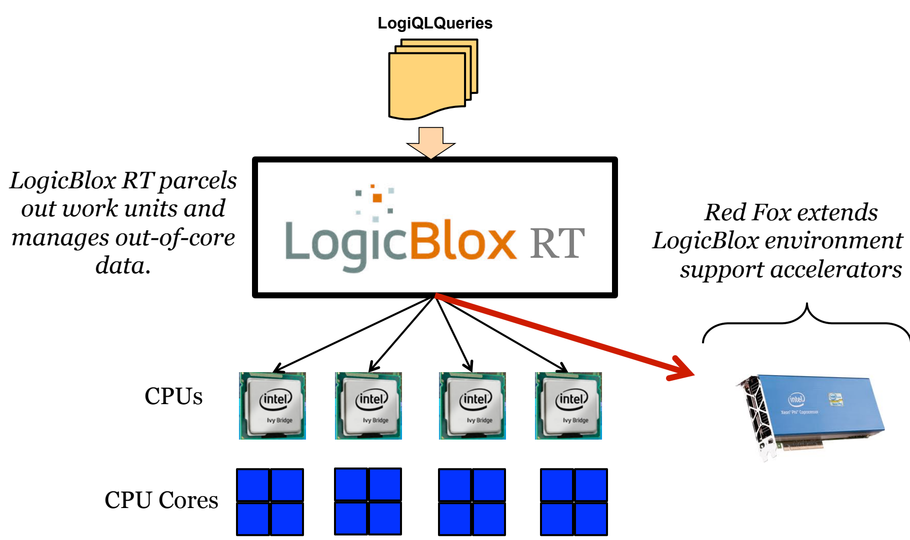
```
tr(x,y,z) ← E(x,y),E(y,z),E(x,z),x<y<z.
```





15

Getting to Out of Core Data Sets



16

Overview

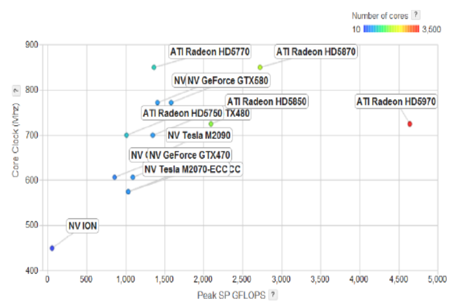
- Drivers
- High Performance Relational Computing
- Benchmark Repository
- Near Memory Processing

The Scalable Heterogeneous Computing Benchmark Suite

Courtesy: Oak Ridge National Laboratories *Jeffery Young*

https://github.com/vetter/shoc/wiki

- Early focus on scientific computing workloads
- Kernels implemented in CUDA, OpenCL MIC port was developed in collaboration with Intel
  - System and stability tests
  - Multi-accelerator & cluster scale support



<u>Max FLOPS Benchmark from SHOC</u>

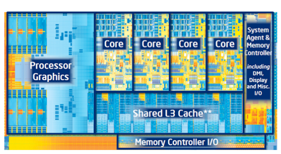
- Our current efforts → adding Red Fox kernels, TPC-H microbenchmarks, and TPC-H queries

-Danalis, et al, The Scalable HeterOgeneous Computing (SHOC) Benchmark Suite, GPGPU '10
-https://github.com/vetter/shocOur

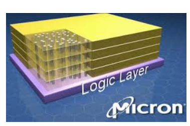
Overview

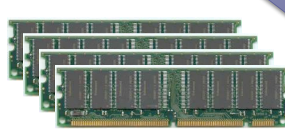
- Drivers
- High Performance Relational Computing
- Benchmark Repository
- Near Memory Processing

Disaggregated Compute



Place Compute Near the Data



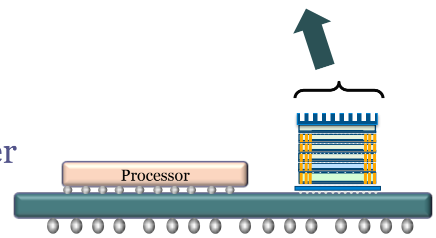


Explore novel programming models and abstractions

Near Memory Data Intensive Computing

H. Kim (CS), S. Yalamanchili (ECE)
Collaborative Discussions with Intel Labs (N. Carter)

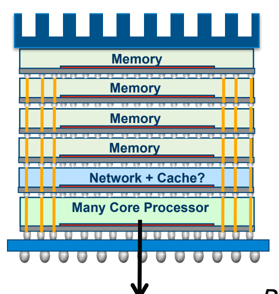
- Move Analytics Primitives (RA) into the memory system
  - Data movement optimization
  - Data locality optimizations
- Explore novel compute and memory architectures
  - Memory consistency and coherency models
  - Integrated thermal and power management



Heterogeneous Architecture Research Prototype (HARP)

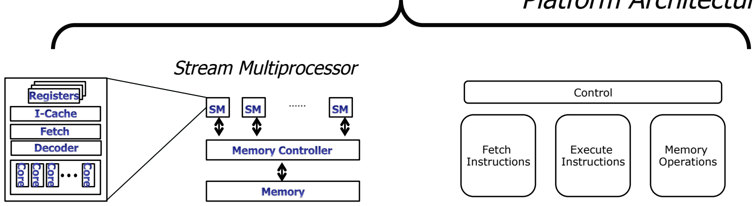
Chad Kersey
Meghana Gupta

- Parametric, C++ processor generator environment
- Harmonica v2 in Altera FPGAs
- Assembler, emulator, and linker
- **OpenCL programming environment and Compiler** (in progress)



- RISC ISA
- C++ generator flow
- gcc compilation support
- Basic multicore/ multithreaded support
- Testing with cycle level simulators (in progress)

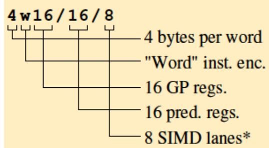
Platform Architecture



HARP Core *RISC Core*

Heterogeneous Architecture Research Prototype (HARP)

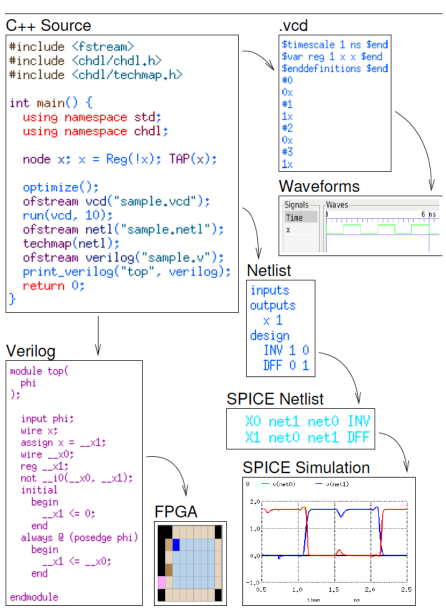
```
/* Return in r0 dot product of vectors
   of real values pointed to by r0 and
   r1, length in r2 */
dotprod: ldi %r3, #0;
dploop:  ld %r4, %r0, #0;
         ld %r6, %r1, #0;
         subi %r2, %r2, #1;
         addi %r0, %r0, __WORD;
         addi %r1, %r1, __WORD;
         rtop @p0, %r2;
         fmul %r4, %r4, %r6;
         fadd %r3, %r3, %r4;
   @p0 ? jmpi dploop;
         ori %r0, %r3, #0;
         jmpr %r5;
```



*ignored by assembler/linker

- Customizable, multithreaded, SIMD soft core
  - Generated from an architecture specification
- Set of associated instruction set architectures for
- Supported by a generated HARP Tool assembler/linker/emulator
- Small - ~1500 lines of C++

Tool Chain



- Vertically integrated CAD environment
- Strong emphasis on code reusability
- Same model for generating both gate level and system level simulations
- CAD tool interfaces
- v2 running in Altera FPGAs

Project Elements

Thread-I: Understanding Architecture Trade-offs and Application Analysis
- Cycle-level timing modeling
- Energy modeling
- Application Profiling Techniques

→

Thread-II: Simple PIM FPGA
- HARP Core design
- HARP SOC design
- HARP Kernel based evaluations
- HARP Kernel Programming

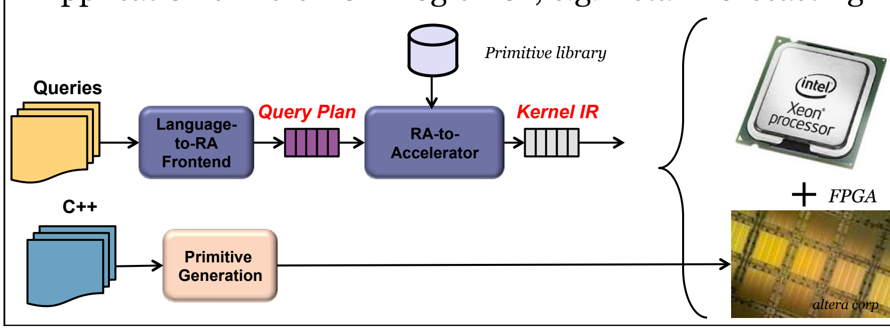
→

Thread-III: PIM with big data/ server workloads
- Benchmarks conversion with OpenCL
- OpenCL→ HARP compiler
- PIM run-time system
- PIM driver
- HARP-core +PCI-E

Accelerator Customization

Collaborative Discussions with Intel Labs (N. Carter)

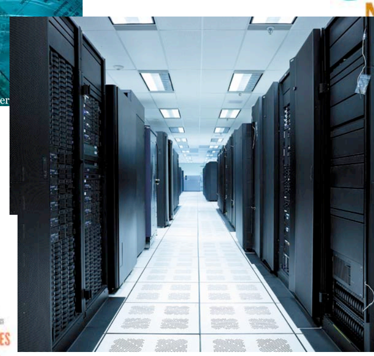
- Custom implementations of data analytics primitives
  - On-demand acceleration
  - C++ development flow
- Customization of new higher level primitives
  - E.g., graph primitives via multi-predicate operators
- Optimized execution of query plans
- Application drivers from LogicBlox, e.g. Retail Forecasting



Contributing Students

- Red Fox:
  - Haicheng Wu (Algorithms & Compiler)
  - Se Hoon Shon (Algorithms)
  - Ifrah Saeed (OpenCl RA Primitives and TPC-H Microbenchmarks)
- Processor Near Memory:
  - Meghana Gupta (OpenCL compiler)
  - Chad Kersey (HARP architecture and Tool Chain)
  - Troy O’Neal (HARP Architecture and Tool Chain)
- SHOC Benchmarks
  - Jeffery Young (Post Doc): Oak Ridge National Laboratories

The Future is Acceleration



Thank You

Large Graphs