Agenda

Accelerated Computing
FireHose
PageRank end-to-end
nvGRAPH
Coming Soon
Conclusion
ACCELERATED COMPUTING
10x Performance & 5x Energy Efficiency

GPU Accelerator

- Summit & Sierra: U.S. Announces Two Pre-Exascale Supercomputers Powered by GPU & NVLink
- Breakthrough in HIV Research: World’s Largest Simulation of Virus Uncovers New Discovery
- Oak Ridge TITAN: World’s Fastest Supercomputer
- Top500: 3 of Top 5 Supercomputers with Tesla GPUs
- Deep Learning: Univ. of Toronto Team Uses GPUs to Win Image-Net Competition, Google Acquires Team
- Tsubame: World’s First GPU Supercomputer
- NVIDIA Launches CUDA

# GPU Developers

2007 2008 2009 2010 2011 2012 2013 2014
PERFORMANCE GAP CONTINUES TO GROW

Peak Double Precision FLOPS

Peak Memory Bandwidth

GFLOPS

GB/s


M1060 M2090 K20 Pascal


M1060 M2090 K20 K80 Pascal

NVIDIA GPU

x86 CPU
Volume and Velocity of some big data tasks do not allow for store & analyze.

Strong need to analyze on-the-fly, continuous stream of data, without trip to disk first.

Applications in Network monitoring - Social and Cyber - are growing in importance.

Firehose

Suite of tasks measuring best-effort processing of UDP packets at high data rates.

Compare processing software and hardware
Quantitative & Qualitative
Firehose Parts

Generator streams UDP packets
- not throttled by Analyzer
- only a few ops per datum
Analyzer may not be able to keep up, measure success rate

3 Firehose Benchmarks

- Power-law anomaly detection
- Active power-law anomaly detection
- Two-level anomaly detection
FIREHOSE CUDA IMPLEMENTATION

Analytics are implemented as pThreads + CUDA apps

N worker threads for each UDP socket

Threads read data and insert into double-buffered queue.

When buffer filled it is sent to one of K GPUs for processing
FIREHOSE PROCESSING RATE

CPU: Dual Xeon, 16 core X5690
PAGERANK PIPELINE BENCHMARK

Graph Analytics Benchmark

Proposed by MIT LL.

Apply supercomputing benchmarking methods to create scalable benchmark for big data workloads.

Four different phases that focus on data ingest and analytic processing.

Reference code for serial implementations available on GitHub.

https://github.com/NVIDIA/PRBench
PAGERANK PIPELINE BENCHMARK
4 Stage Graph Analytics Benchmark

Stage 1 - Generate graph  (not timed)
Stage 2 - Read graph from disk, sort edges, write back to disk
Stage 3 - Read sorted edge list, generate normalized adjacency matrix for graph
Stage 4 - Run 20 iterations of Pagerank algorithm (power method)

Stage 2 tests I/O
Stage 3 tests I/O + compute
Stage 4 tests compute
# SPEEDUP VS REFERENCE C++

<table>
<thead>
<tr>
<th>Scale</th>
<th>K0(99%)</th>
<th>K1(90%)</th>
<th>K2(80%)</th>
<th>K3(0%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>2.8x</td>
<td>1.1x</td>
<td>4.6x</td>
<td>5.9x</td>
</tr>
<tr>
<td>17</td>
<td>2.6x</td>
<td>1.3x</td>
<td>5.0x</td>
<td>10.3x</td>
</tr>
<tr>
<td>18</td>
<td>2.9x</td>
<td>1.3x</td>
<td>6.3x</td>
<td>14.1x</td>
</tr>
<tr>
<td>19</td>
<td>3.2x</td>
<td>1.5x</td>
<td>7.6x</td>
<td>14.0x</td>
</tr>
<tr>
<td>20</td>
<td>3.3x</td>
<td>1.5x</td>
<td>8.7x</td>
<td>11.8x</td>
</tr>
<tr>
<td>21</td>
<td>3.2x</td>
<td>1.5x</td>
<td>9.2x</td>
<td>9.8x</td>
</tr>
<tr>
<td>22</td>
<td>3.4x</td>
<td>1.5x</td>
<td>9.9x</td>
<td>9.8x</td>
</tr>
</tbody>
</table>
PAGERANK PIPELINE RESULTS

Single P100 GPU

Speed up versus MIT LL reference C++ implementation

GPU: Tesla P100
CPU: Intel Xeon E5 2690 3.0 GHz

<table>
<thead>
<tr>
<th>Stage</th>
<th>GPU Speed Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAGE 2</td>
<td>1.8X</td>
</tr>
<tr>
<td>STAGE 3</td>
<td>11X</td>
</tr>
<tr>
<td>STAGE 4</td>
<td>44X</td>
</tr>
</tbody>
</table>
PAGERANK PIPELINE RESULTS

Comparing Across GPUs

Memory bandwidth most important

GTEPS = billions edges/sec

# edges = E * 2^S

Stage 4 - Pagerank computation

E = 16   S = 21

<table>
<thead>
<tr>
<th>GPU</th>
<th>GTEPS</th>
<th>Aggregate Memory BW</th>
</tr>
</thead>
<tbody>
<tr>
<td>K40</td>
<td>2.45</td>
<td></td>
</tr>
<tr>
<td>M40</td>
<td>3.23</td>
<td></td>
</tr>
<tr>
<td>P100</td>
<td>9.88</td>
<td></td>
</tr>
<tr>
<td>DGX-1</td>
<td>37.9</td>
<td></td>
</tr>
</tbody>
</table>

CPU
DSSD+NVIDIA PageRank Results

Key Takeaways

• Out of the box (no change to test code) D5 BLK speed is comparable to RAM Disk
• Using API (minimal code change) D5 is \textit{2.7x faster} than RAM Disk
• D5 Advantages:
  • Device speed
  • Direct-to-device API
  • Direct transfer to GPU
  • Shared between machines
  • High Capacity
DSSD+NVIDIA PageRank Results

**Complete Runtime**

<table>
<thead>
<tr>
<th>Storage Type</th>
<th>Complete Runtime (seconds)</th>
<th>Relative Speed Compared to HDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDD</td>
<td>21.6036</td>
<td>1.0x</td>
</tr>
<tr>
<td>DSSD BLKDEV</td>
<td>6.6846</td>
<td><strong>3.2x</strong> faster than HDD</td>
</tr>
<tr>
<td>DSSD API</td>
<td>1.9719</td>
<td><strong>2.1x</strong> faster than RAM Disk, <strong>11x</strong> faster than HDD</td>
</tr>
<tr>
<td>DSSD API GPU Direct</td>
<td>1.5913</td>
<td><strong>2.7x</strong> faster than RAM Disk, <strong>13.6x</strong> faster than HDD</td>
</tr>
<tr>
<td>RAM Disk</td>
<td>4.2278</td>
<td></td>
</tr>
</tbody>
</table>

**Date:** 4 Aug 2016 18:32 GMT  
**Host:** shaded DSSD/NVIDIA machine  
**Test:** mpirun -np 1 --bind-to none ./prbench -S 21 -E 16 --no-rcache
DSSD+NVIDIA PageRank Results

- **Device Read**
- **Delivery to GPU**

### Total read time

- **HDD**: 8.1326
  - 2.9x faster than HDD
- **DSSD BLKDEV**: 8.8054
  - 2.9x faster than HDD (orange bar)
  - 1.2x faster than RAM Disk, 11.5x faster than HDD (red bar)
- **DSSD API**: 0.7677
  - 2.3x faster than RAM Disk, 22.9x faster than HDD (blue bar)
- **DSSD API GPU Direct**: 0.3838
- **RAM Disk**: 0.8967

**Date:** 4 Aug 2016 18:32 GMT

**Host:** shaded DSSD/NVIDIA machine

**Test:** mpirun -np 1 --bind-to none ./prbench -S 21 -E 16 --no-rcache

**time in seconds**
DSSD+NVIDIA PageRank Results

- 3.9x faster than HDD, and 90% speed of RAM Disk
- 4.2x faster than RAM Disk, 18.2x faster than HDD
- 4.2x faster than RAM Disk, 18.2x faster than HDD

Date: 4 Aug 2016 18:32 GMT
Host: shaded DSSD/NVIDIA machine
Test: mpirun -np 1 --bind-to none ./prbench -S 21 -E 16 --nocrace
# Graphs Are Fundamental
Tight connection between data and graphs

<table>
<thead>
<tr>
<th>Data View</th>
<th>Graph View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Element/ Entity</td>
<td>Graph Vertex</td>
</tr>
<tr>
<td>Entity Attributes</td>
<td>Vertex labels</td>
</tr>
<tr>
<td>Binary Relation (1 to 1)</td>
<td>Graph Edge</td>
</tr>
<tr>
<td>N-ary Relation (many to 1)</td>
<td>Hypergraph edge</td>
</tr>
<tr>
<td>Relation Attributes</td>
<td>Edge labels</td>
</tr>
<tr>
<td>Group of relations over entities</td>
<td>Sets of Vertices and Edges</td>
</tr>
</tbody>
</table>
NVGRAPH
Easy Onramp to GPU Accelerated Graph Analytics

- GPU Optimized Algorithms
- Reduced cost & Increased performance
- Standard formats and primitives
  - Semi-rings, load-balancing
- Performance Constantly Improving
nvGRAPH
Accelerated Graph Analytics

nvGRAPH for high performance graph analytics

Deliver results up to 3x faster than CPU-only

Solve graphs with up to 2 Billion edges on a single GPU (M40)

Accelerates a wide range of graph analytics applications:

<table>
<thead>
<tr>
<th>PageRank</th>
<th>Single Source Shortest Path</th>
<th>Single Source Widest Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>Robotic Path Planning</td>
<td>IP Routing</td>
</tr>
<tr>
<td>Recommendation Engines</td>
<td>Power Network Planning</td>
<td>Chip Design / EDA</td>
</tr>
<tr>
<td>Social Ad Placement</td>
<td>Logistics &amp; Supply Chain Planning</td>
<td>Traffic sensitive routing</td>
</tr>
</tbody>
</table>

nvGRAPH: 3.4x Speedup

PageRank on Twitter 1.5B edge dataset

2x12 Core Xeon E5 v2

nvGRAPH on P100

GraphMat on 2 socket 12-core Xeon E5-2697 v2 CPU, @ 2.70 GHz

developer.nvidia.com/nvgraph
Motivating example

Power law graph: wiki2003.bin

455,436 vertices (n)
2,033,173 edges (nnz)
sparsity = 4.464234

Cusparse csr mv time: 8.05 ms

Merge Path csr mv time: 1.08 ms

~7.45x faster!

PSG Cluster, K40
SEMI-RINGS
Definition / Axioms

Set $\mathbb{R}$ with two binary operators: $+$ and $\ast$ that satisfy:

1. $(\mathbb{R}, +)$ is associative, commutative with additive identity $0$
   
   $(0 + a = a)$

2. $(\mathbb{R}, \ast)$ is associative with multiplicative identity $1$
   
   $(1 \ast a = a)$

3. Left and Right multiplication is distributive over addition

4. Additive identity $0 = \text{multiplicative null operator}$
   
   $(0 \ast a = a \ast 0 = 0)$
# SEMI-RINGS

## Examples

<table>
<thead>
<tr>
<th>SEMIRING</th>
<th>SET</th>
<th>PLUS</th>
<th>TIMES</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>$\mathbb{R}$</td>
<td>+</td>
<td>*</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MinPlus</td>
<td>$\mathbb{R} \cup {-\infty, \infty}$</td>
<td>min</td>
<td>+</td>
<td>$\infty$</td>
<td>0</td>
</tr>
<tr>
<td>MaxMin</td>
<td>$\mathbb{R} \cup {-\infty, \infty}$</td>
<td>max</td>
<td>min</td>
<td>$-\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>Boolean</td>
<td>${0, 1}$</td>
<td>$\lor$</td>
<td>$\land$</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Applications

Pagerank (+, *)

- Ideal application: runs on web and social graphs
- Each iteration involves computing: \( y = A x \)
- Standard csr\(m\)v
- PlusTimes Semiring
- \( \alpha = 1.0 \) (multiplicative identity)
- \( \beta = 0.0 \) (multiplicative nullity)

://sw/gpgpu/naga/src/pagerank.cpp
APPLICATIONS
Single Source Shortest Path (min, +)

Common Usage Examples:

Path-finding algorithms:
- Navigation
- Modeling
- Communications Network

Breadth first search building block

Graph 500 Benchmark
APPLICATIONS
Widest Path (max,min)

Common Usage Examples:

- Maximum bipartite graph matching
- Graph partitioning
- Minimum cut
- Common application areas:
  - power grids
  - chip circuits
PROPERTY GRAPHS
Many simple graphs overlaid
SUBGRAPH EXTRACTION

Focus on a specific area

Explicit Graph

Developer's FOAF Import Graph

Friend-Of-A-Friend Graph

Developer Imports Graph

Software Imports Graph

Friendship Graph

Developer Created Graph

Employment Graph

Friends at Work Graph
COMING SOON
Features in next release

Partitioning
Clustering
BFS
Graph Contraction
PARTITIONING AND CLUSTERING

Spectral Min Edge Cut Partition

Modularity maximization (spectral)

Example 2 partitions

\( B_{ij} = A_{ij} - k_i k_j / 2m \)
BREADTH FIRST SEARCH

Key subroutine in several graph algorithms, naturally leads to random access

MPI Version implementations: pack or use a bitmap to exchange frontier at end of each step

NVSHMEM version: directly updates the frontier map at target using atomics

Benefits with smaller graphs (likely behavior with strong scaling)

4 P100 GPUs alltoall connected with NVLink
Graph Contraction
Graph Contraction
DYNAMIC GRAPHS

cuSTINGER brings STINGER to GPUs

Oded Green presented at HPEC 2016

cuSTINGER: Supporting Dynamic Graph Algorithms for GPUs

https://www.researchgate.net/publication/308174457
nvGRAPH is working toward a full GraphBLAS implementation

Semi-rings are a start
**TOWARD REAL TIME BIG DATA ANALYTICS**

GPUs enable the next generation of in-memory processing

<table>
<thead>
<tr>
<th></th>
<th>DUAL BROADWELL SERVER</th>
<th>NVIDIA DGX-1 SERVER</th>
<th>GPU PERFORMANCE INCREASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregate Memory Bandwidth</td>
<td>150 GB/s</td>
<td>5760 GB/s</td>
<td>38 X</td>
</tr>
<tr>
<td>Aggregate SP FLOPS</td>
<td>4 TF</td>
<td>85 TF</td>
<td>21 X</td>
</tr>
</tbody>
</table>

*Single DGX-1 server provides the compute capability of dozens of dual-cpu servers*
ACCELERATED DATABASE TECHNOLOGY

Big data ISVs moving to the accelerated model

SQL

No SQL

Graph

IBM DB2 with BLU Acceleration

blazegraph™

kinetica

graphistry

ONU TECHNOLOGY

Adobe Research

Spark

MAPD
SUMMARY

GPUs for High Performance Data Analytics

GPU computing is not just for scientists anymore!

GPUs excel at in-memory analytics.
- Streaming - Firehose
- Graph - Pagerank Pipeline
- Analytics - nvGRAPH

Savings in infrastructure size & cost by using GPU servers versus standard dual-CPU server.
Database In-Memory performance
20-100x at practical scale
REFERENCES

*GPU Processing of Streaming Data: A CUDA implementation of the Firehose benchmark*
Mauro Bisson, Massimo Bernaschi, Massimiliano Fatica      IEEE HPEC 2016
NVIDIA corporation & Italian Research Council

*A CUDA Implementation of the Pagerank Pipeline Benchmark*
Mauro Bisson, Everett Phillips, Massimiliano Fatica      IEEE HPEC 2016
NVIDIA corporation