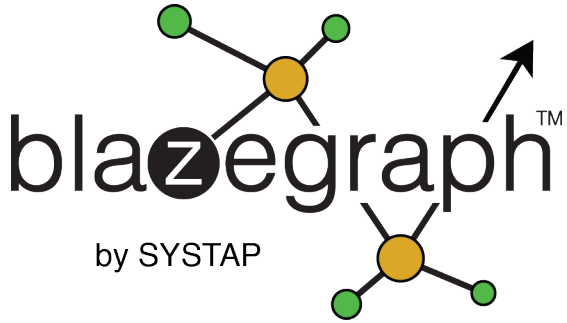


Overcoming the Barriers of Graphs on GPUs: Delivering Graph Analytics 100X Faster and 40X Cheaper

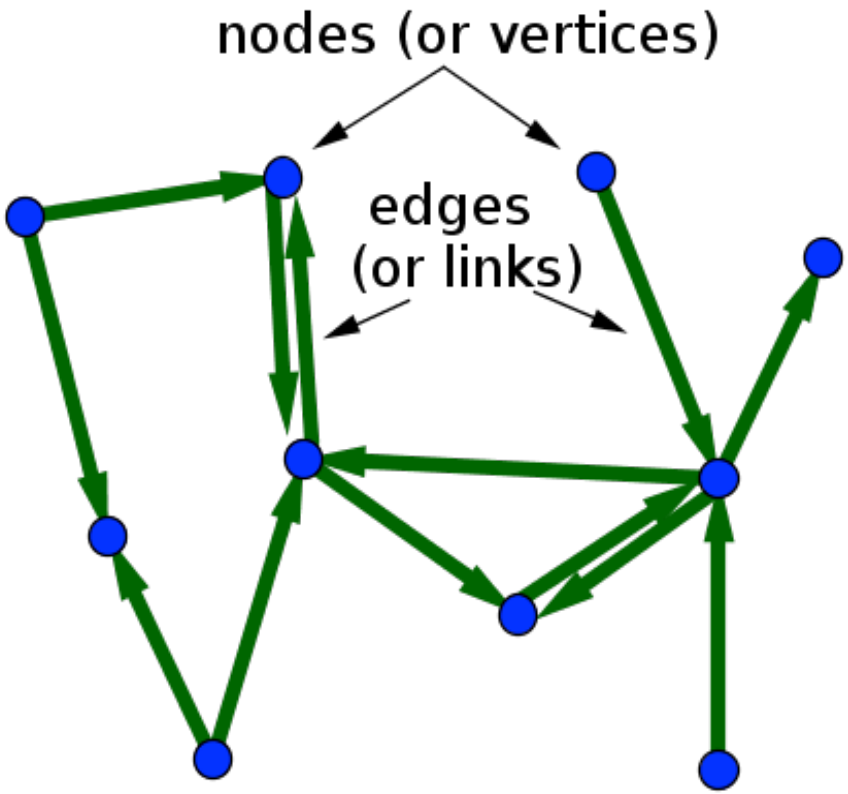
November 18, 2015

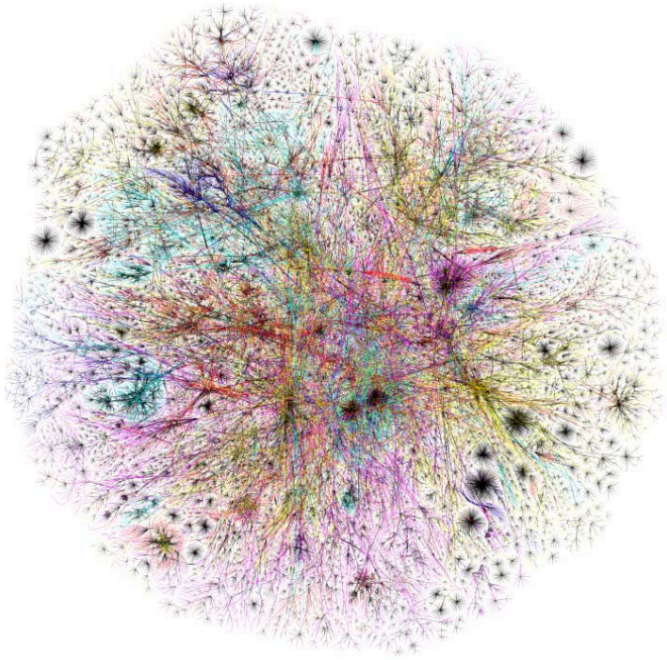
Super Computing 2015

http://blazegraph.com/

blazegraph™
by SYSTAP

The Amount of **Graph Data** is **Exploding**

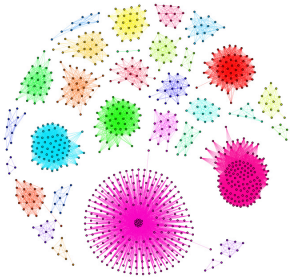




Billion+ Edges

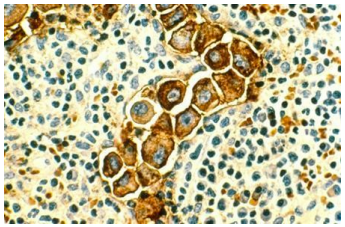
blazegraph™ by SYSTAP

Graph Applications are **Everywhere**



- Community Detection / Clustering
- Recommendation Systems

- Fault Prediction in Industrial and Internet of Things (IoT)



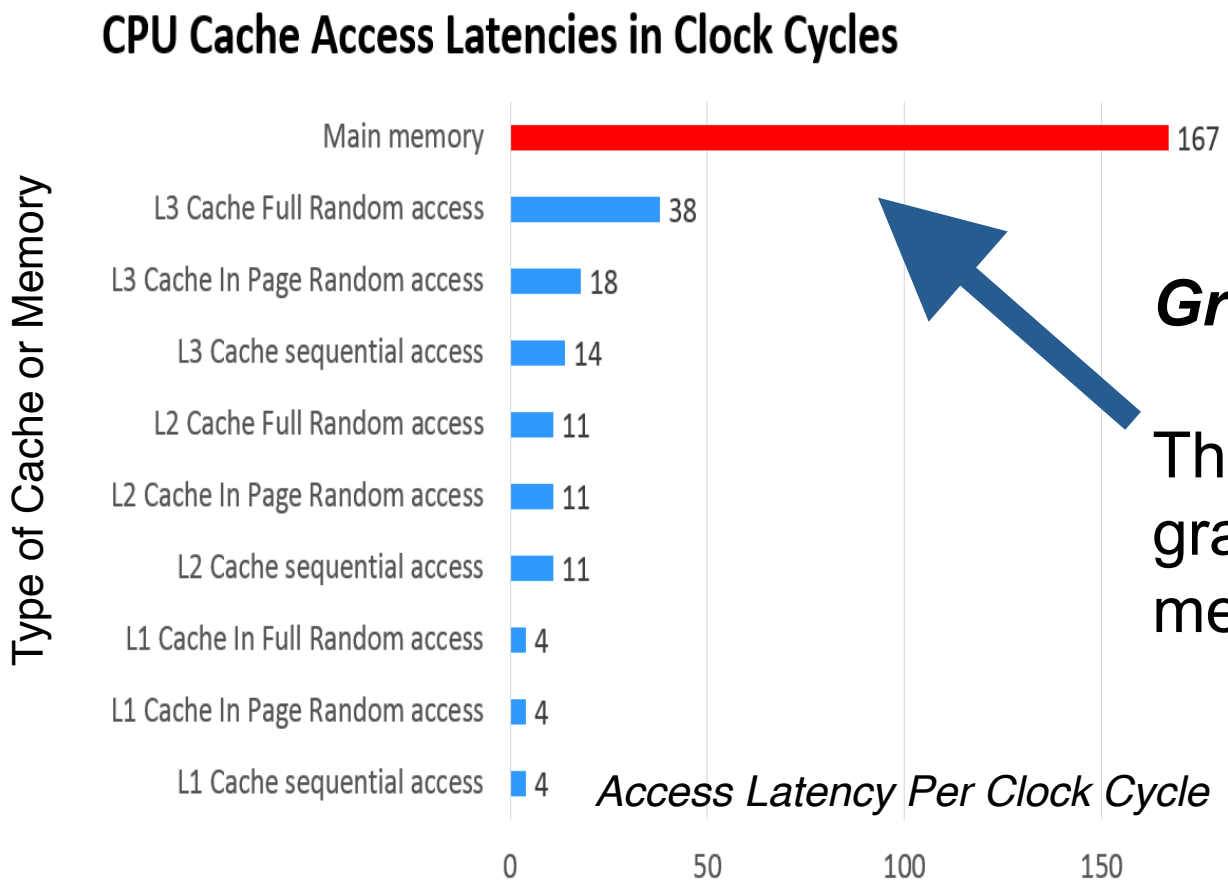
- Drug Discovery / Repurposing
- Precision Medicine / Genomics

- Cyber
- Defense / Security

- Fraud Detection
- Time Series, Compliance

blazegraph™ by SYSTAP

Graphs are **different**. You need the **right paradigm** and **hardware** to **scale**



Graph Cache Thrash

The CPU just waits for graph data from main memory...

https://datatake.files.wordpress.com/2015/09/latency.png

blazegraph™ by SYSTAP

Solutions to the Graph Scaling Problem Using **Graph Databases** and **GPUs**

blazegraph™ by SYSTAP

- **Embedded**
- **High Availability**
- **Scale-out**

- **GPU Acceleration**
  - **100s of Times Faster** than CPU main memory-based systems
  - Up to **40X Cheaper**
  - **10,000X Faster** than disk-based technologies

Blazegraph DASL : Extreme **Scale,** **40X** more **Affordable**!

Large **Hadoop** Cluster
$~18M / GTEP

1 GTEP = 1 Billion Traversed Edges Per Second

Cray XMT-2
$~180K / GTEP

Blazegraph with **GPU Clusters**
$16K / GTEP (K40 - Today) ← *10X!*
$4K / GTEP (Pascal 2016) ← *40X!*

blazegraph™ by SYSTAP

amazon web services™

Future **Blazegraph** SaaS **On-demand**

Available now!

SOFTLAYER® an IBM Company

Cirrascale

Coming soon!

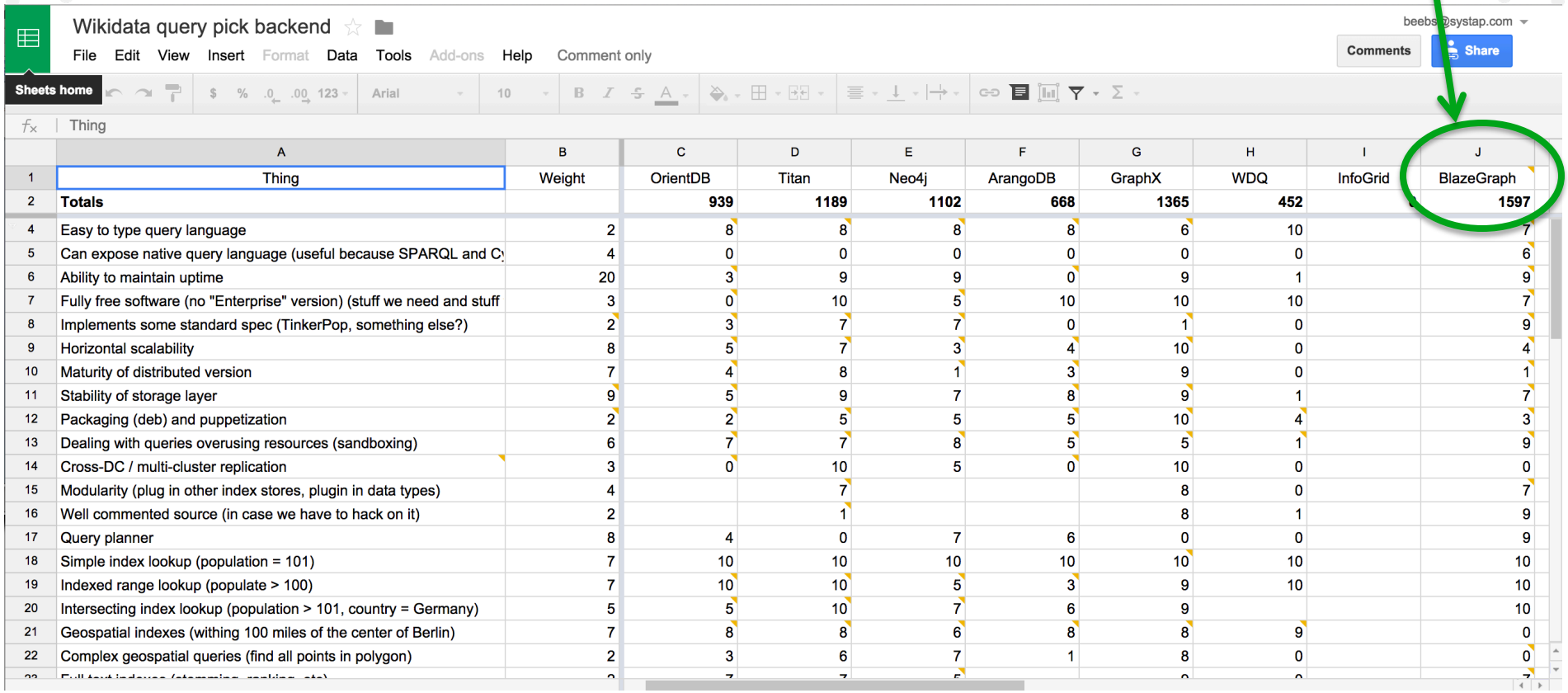
Powering Their Graphs with Blazegraph™

Information Management / Retrieval

Genomics / Precision Medicine

Defense, Intel, Cyber

Blazegraph™ stands out!



Wikidata query pick backend

File Edit View Insert Format Data Tools Add-ons Help Comment only

Comments Share

Sheets home

Thing

	A	B	C	D	E	F	G	H	I	J
1	Thing	Weight	OrientDB	Titan	Neo4j	ArangoDB	GraphX	WDQ	InfoGrid	BlazeGraph
2	**Totals**		**939**	**1189**	**1102**	**668**	**1365**	**452**		**1597**
4	Easy to type query language	2	8	8	8	8	6	10		7
5	Can expose native query language (useful because SPARQL and C	4	0	0	0	0	0	0		6
6	Ability to maintain uptime	20	3	9	9	0	9	1		9
7	Fully free software (no "Enterprise" version) (stuff we need and stuff	3	0	10	5	10	10	10		7
8	Implements some standard spec (TinkerPop, something else?)	2	3	7	7	0	1	0		9
9	Horizontal scalability	8	5	7	3	4	10	0		4
10	Maturity of distributed version	7	4	8	1	3	9	0		1
11	Stability of storage layer	9	5	9	7	8	9	1		7
12	Packaging (deb) and puppetization	2	2	5	5	5	10	4		3
13	Dealing with queries overusing resources (sandboxing)	6	7	7	8	5	5	1		9
14	Cross-DC / multi-cluster replication	3	0	10	5	0	10	0		0
15	Modularity (plug in other index stores, plugin in data types)	4		7			8	0		7
16	Well commented source (in case we have to hack on it)	2		1			8	1		9
17	Query planner	8	4	0	7	6	0	0		9
18	Simple index lookup (population = 101)	7	10	10	10	10	10	10		10
19	Indexed range lookup (populate > 100)	7	10	10	5	3	9	10		10
20	Intersecting index lookup (population > 101, country = Germany)	5	5	10	7	6	9			10
21	Geospatial indexes (withing 100 miles of the center of Berlin)	7	8	8	6	8	8	9		0
22	Complex geospatial queries (find all points in polygon)	2	3	6	7	1	8	0		0

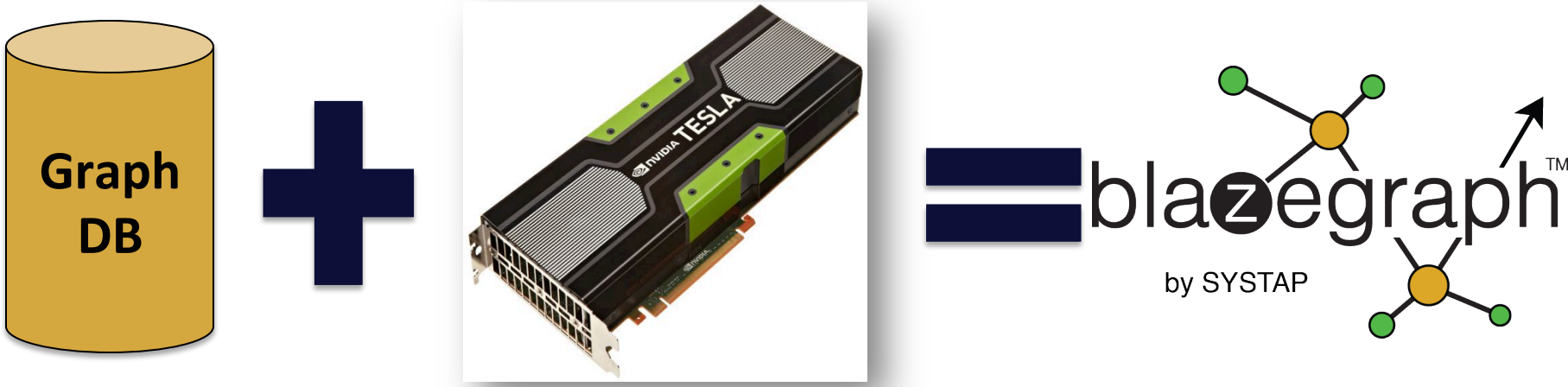
- Wikimedia Evaluation: https://docs.google.com/a/systap.com/spreadsheets/d/1MXikljoSUVP77w7JKf9EXN40OB-ZkMqT8Y5b2NYVKbU/edit#gid=0

How do I use GPUs to scale graphs?

- **Parallel Processing** on GPU Clusters for **Trillion+** Edge Graphs
- **High-Level API**
- **Partitioning** and **Overlapping** Communications
- **HPC** and **DARPA** Pedigree

http://blazegraph.com/

Blazegraph GPU: **Ridiculously Fast** for Graphs



Blazegraph™ **plug-in for GPU Acceleration** with familiar graph APIs – 200-300X acceleration with almost no code changes.

W3C SPARQL

RDF

Apache TinkerPop

Gremlin $G = (V, E)$

Just add GPUs --Testing Shows 200-300X Speed-up

LUBM Query #9 U1000 (167,697,263 Edges w/Inference)

290X

```
SELECT ?x ?y ?z
WHERE {
      ?x a ub:Student .
      ?y a ub:Faculty .
      ?z a ub:Course .
      ?x ub:advisor ?y .
      ?y ub:teacherOf ?z .
      ?x ub:takesCourse ?z .
}
```

172,632 results

53,960ms on Blazegraph CPU

```
SELECT ?x ?y ?z
WHERE {
      ?x a ub:Student .
      ?y a ub:Faculty .
      ?z a ub:Course .
      ?x ub:advisor ?y .
      ?y ub:teacherOf ?z .
      ?x ub:takesCourse ?z .
}
```

172,632 results

187ms on Blazegraph GPU

Did you hear the one about the Analytics Developer who is also a Parallel Programming expert?

Algorithm Developer

To find the shortest path between two arbitrary vertices from the graph A, call the source v and the destination w, we can:

1. Find c(v) and c(w), if c(v) == c(w) the shortest path can be found by searching over just the local cluster k= c(v)
2. If level.next is not NULL, find N(c(v)) = neighbors of cluster k on coarser level (each vertex on this level corresponds to a cluster in level.previous)
3. If level.next is null, perform SSSP on this level as normal.
4. Check if c(w) is in any of the neighbor clusters if yes, do two shortest path searches, one from v to the edge connection that neighbor, the other from the edge entering that neighbor to vertex w and sum the length of those paths to find the overall shortest path distance. If no, level = level.next
5. Goto 1

Parallel Programmer

```
   // Allocate problem on GPU
   int num_gpus = 1;
   typedef GASengine::CsrProblem<bfs, VertexId, SizeT, Value, g_mark_predecessor, g_
with_value> CsrProblem;
   CsrProblem csr_problem(cfg);

   if (csr_problem.FromHostProblem(g_stream_from_host, csr_graph.nodes,
                                   csr_graph.edges,
                                   csr_graph.column_indices, csr_graph.row_offsets,
                                   csr_graph.edge_values,
                                   csr_graph.row_indices, csr_graph.column_offsets,
                                   num_gpus,
                                   directed, device_id, rank_id))
      exit(1);

   printf("MPI Rank: %d, Loaded graph to GPU.\n", rank_id);

   if (cfg.getParameter<int>("stats")) {
               // Show statistics on the graph if requested.
               SizeT maxDegree = 0;
               VertexId maxDegreeVertex = csr_graph.getMaximumDegree(maxDegree);
               if (cfg.getParameter<int>("origin") == 1) {
                       maxDegreeVertex++; // 0-based index
               }
               fprintf(stdout, "vertexId=%lld has maximum degree of %lld\n",
                               (long long) maxDegreeVertex, (long long) maxDegree);
               csr_graph.PrintHistogram();
   }


   const bool INSTRUMENT = true;
   cudaError_t retval = cudaSuccess;

   if(profile) cudaProfilerStart(); // START PROFILER

   GASengine::EnactorVertexCentric<CsrProblem, bfs, INSTRUMENT> vertex_centric(cfg,
g_verbose);
```

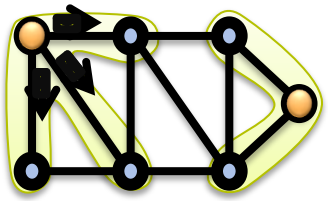
Need: Simpler, smarter algorithms that run efficiently on GPUs without requiring knowledge of the parallel programming or device-specific optimization.

Blazegraph DASL -- “Dazzle”

(Super Computing 2015)

Imagine the ease of use of Spark and Scala for graph algorithms and predictive analytics…

Spark Scala



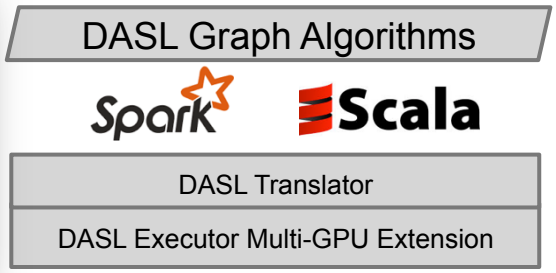
…With the speed of CUDA and GPUs.

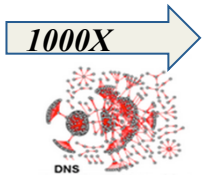
blazegraph™ by SYSTAP

Blazegraph DASL – A Domain specific language for graphs with Accelerated Scala using Linear algebra

You can see why we call it “DASL”…

DASL your analytics





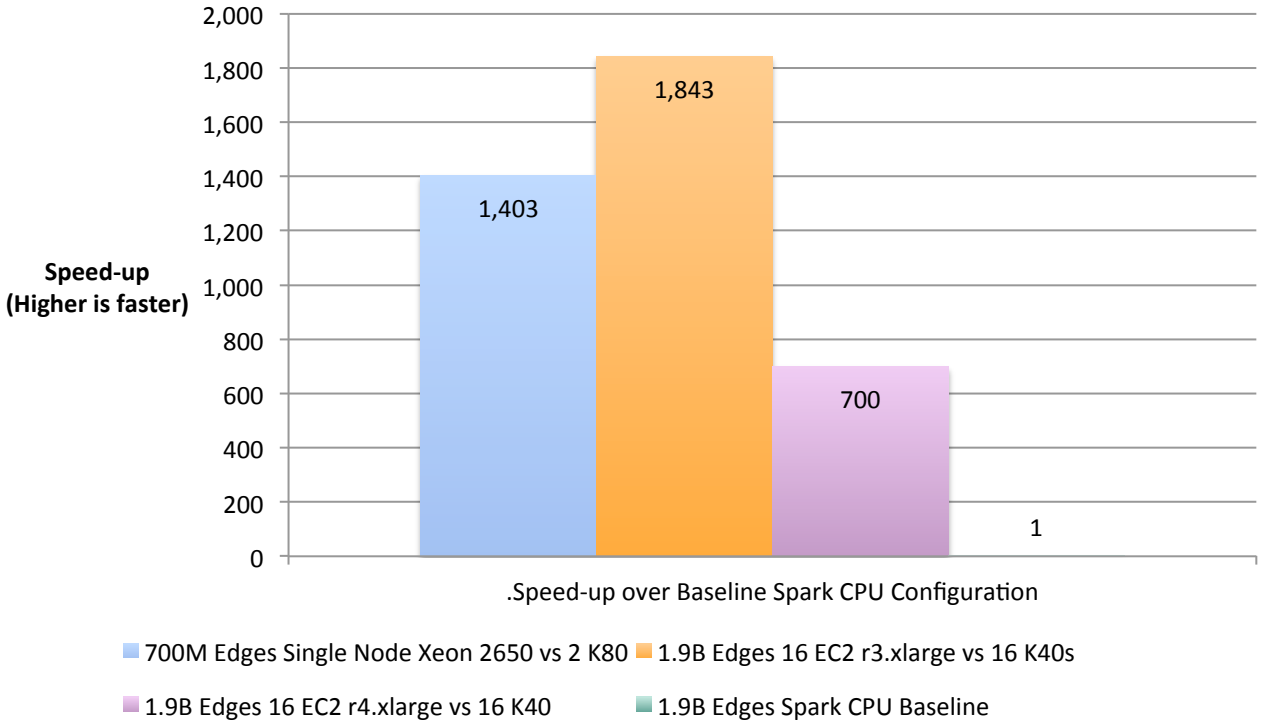
- DASL is functional, domain-specific language (DSL) for graph and machine learning algorithms
- Provides of linear algebra primitives supporting the DSL designed for efficient, multi-GPU execution
- Reduce the barrier of graphs on GPUS, enabling simpler and smarter algorithms
- Co-opt existing data ecosystems, such as Apache Spark and Hadoop, for ease of adoption and mission impact

GPU-Accelerated DASL Algorithms

- Graph
  - Hierarchical partitioning / graph coarsening
  - Louvain modularity
  - Jaccard Similarity
  - Triangle counting
- Collaborative Filtering (Recommendation Systems)
  - User-Based, Item-Based
  - Matrix Factorization with ALS (alternating least squares)
  - Weighted Matrix Factorization, SVD++
- Many others…

- Supervised neural network techniques
  - Hidden Markov Models
  - Multilayer Perceptron
- Clustering
  - Canopy, k-Means, Fuzzy k-Means, Streaming k-Means
  - Spectral Clustering
- Dimensionality Reduction
  - Singular Value Decomposition (top-k)
  - Lanczos Algorithm
  - Stochastic SVD
  - QR Decomposition
  - Non-Negative matrix factorization (NNMF)
  - Distance Geometry

Let's compare to Spark on a 1.9B Edge Twitter Graph



GPUs 700X-1800X faster than graphs in all cases.

Twitter ICWSM 2010

- "Data from our ICWSM 2010 paper is available from the link below. Our datasets have been anonymized to protect the privacy of the users themselves. At this moment, we are only releasing information about the Twitter link structure"
  - http://twitter.mpi-sws.org/data-icwsm2010.html
  - 2^17 max depth; 1.7 Average connectivity
  - Full data set: 1,963,263,824 Edges and 36 GB uncompressed
  - 1M Edge Sample: 15M uncompressed
  - 700M Edge Sample: 13G uncompressed

Not Connected	38.31%
Degree 2^0:	15.33%
Degree 2^1:	13.61%
Degree 2^2:	17.32%
Degree 2^3:	7.88%
Degree 2^4:	3.28%
Degree 2^5:	1.90%
Degree 2^6:	1.18%
Degree 2^7:	0.64%
Degree 2^8:	0.40%
Degree 2^9:	0.13%
Degree 2^10:	0.03%
Degree 2^11:	0.01%
Degree 2^12:	0.00%
Degree 2^13:	0.00%
Degree 2^14:	0.00%
Degree 2^15:	0.00%
Degree 2^16:	0.00%
Degree 2^17:	0.00%

Experiment Configurations (Spark / CPU)

Name	Qty	Total RAM (G)	Total CPU Cores	Instance Type	CPU Cores	Node RAM (G)	Storage (G)	Physical Processor	Clock (GHZ)	Chip Memory Bandwidth (G/s)
Softlayer CPU	1	128	8	Softlayer CPU	8	128	1 X 1000 SSD	Intel Xeon E5-2690	2.5	51.2
16-r3.xlarge	16	488	640	EC2 r3.xlarge	40	30.5	1 x 80 SSD	Intel Xeon E5-2670 v2 (Ivy Bridge)	2.5	59.7
16-r3.4xlarge	16	1952	2560	EC2 r3.4xlarge	160	122	1 x 320 SSD	Intel Xeon E5-2670 v2 (Ivy Bridge)	2.5	59.7

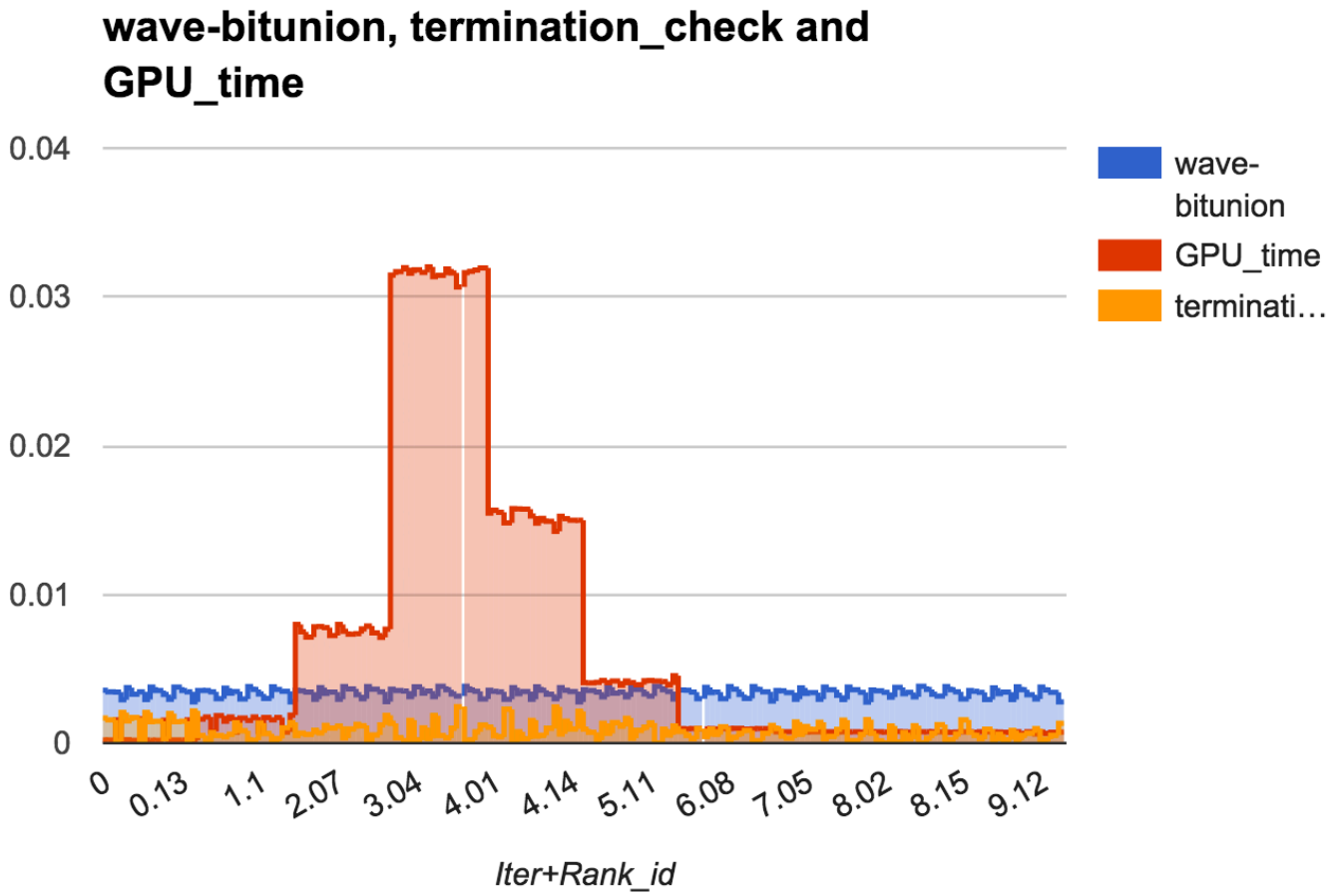
- Spark 1.5.1
- Centos 6.6
- Hadoop 2.7

GPU Experiment Configurations

		Total RAM (G)	Total CUDA Cores	Model	CUDA Cores	GPU RAM (G)	GPU Chip	Memory Bandwidth (G/s)
Softlayer GPU	2	48	9984	K80	4992	24	GK210 (2)	(240 / GK210) 480
PSG-16K40	16	192	46080	K40	2880	12	GK110B	288

- PSG K40s have Infiniband connectivity between the nodes.
- CUDA 7.0
- MVAPICH2 2.1

Detailed Observations (16 K40)

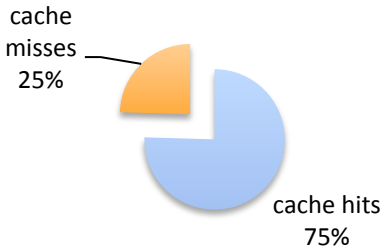


Detailed Observations Spark Cache

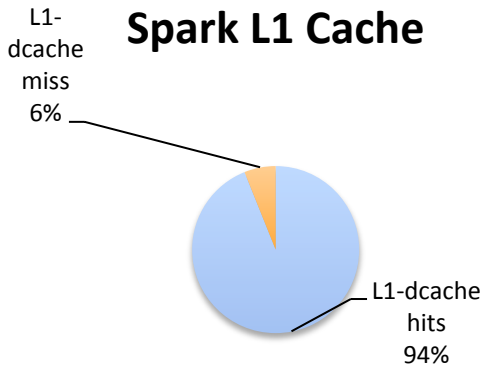
- Softlayer Spark Machine
- 700m Edge Run
- Missing cache about 25% of the time

Data	Label	Notes
867,301,015,332	cycles:u	[66.73%]
528,176,517,726	instructions:u	# 0.61 insns per cycle [83.37%]
2,380,301,262	cache-misses	# 24.544 % of all cache refs [83.31%]
9,698,182,996	cache-references	[83.30%]
1,287,086	context-switches	
145,907,875,938	L1-dcache-loads	[83.33%]
9,448,086,819	L1-dcache-load-misses	# 6.48% of all L1-dcache hits [83.35%]
565.3839499	seconds time elapsed	

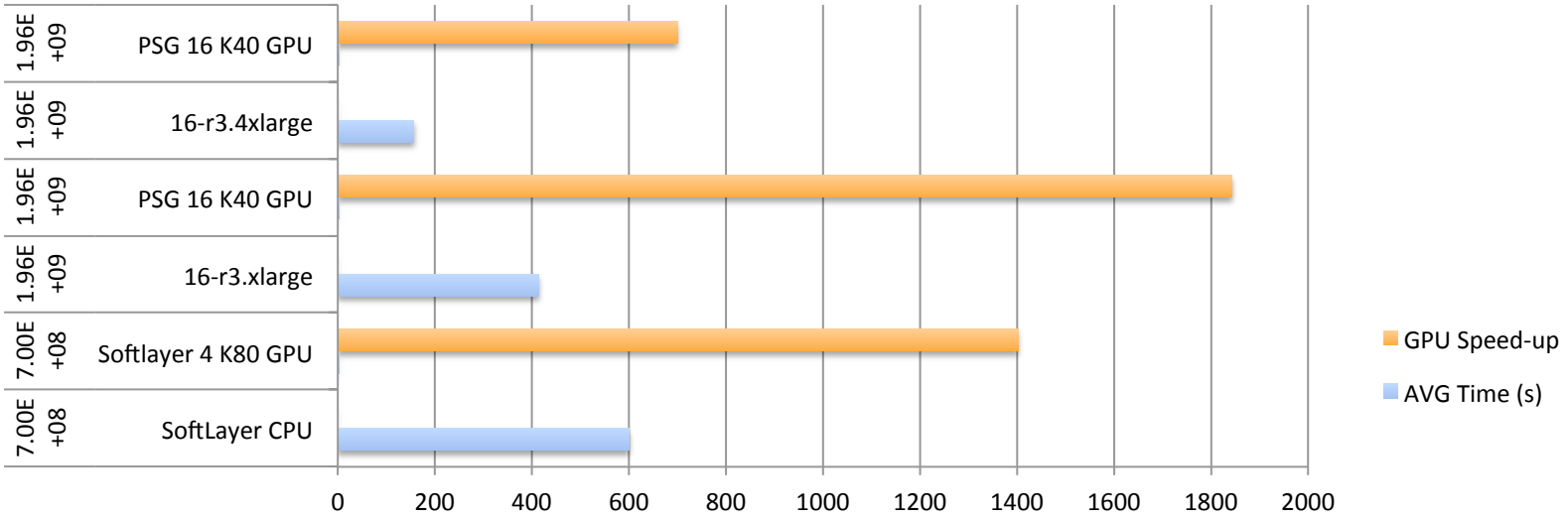
Spark cache hits/misses



Spark L1 Cache



Summary Results (Review)



	7.00E+08 SoftLayer CPU	7.00E+08 Softlayer 4 K80 GPU	1.96E+09 16-r3.xlarge	1.96E+09 PSG 16 K40 GPU	1.96E+09 16-r3.4xlarge	1.96E+09 PSG 16 K40 GPU
GPU Speed-up	0	1402.69	0	1843.16	0	700.38
AVG Time (s)	601.34	0.4284	413.46	0.2242	157.25	0.2242

- GPUs 700X to 1800X faster in all cases.

Pain Points and GPU Solutions

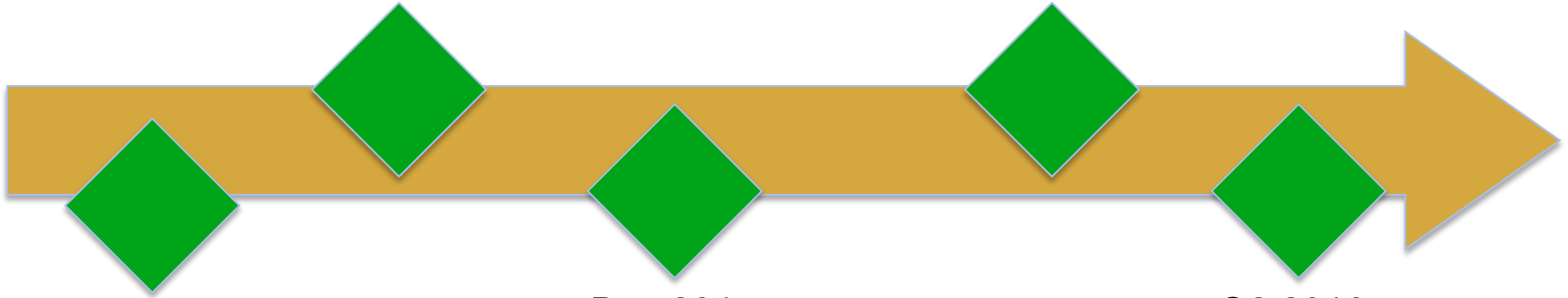
- Acceleration of existing graph database applications – Blazegraph GPU
  - Licensed Software
  - PaaS Offering
  - HW-based Appliance
- Large Scale Graph Analytics – Blazegraph DASL
  - Spark or Hadoop-based Graph Analytics that are performing too slowly to meet real-time analytic needs.
  - Scala-based language maps easily into existing Spark installations

Product Timeline / Roadmap

DASL Early Release
Opportunity for very large graph clusters.

Nov 2015

Q1 2016



Today

Blazegraph GPU

Proof of Concept / Pilots with key customers.

DASL
Proof of Concept / Pilots with key customers

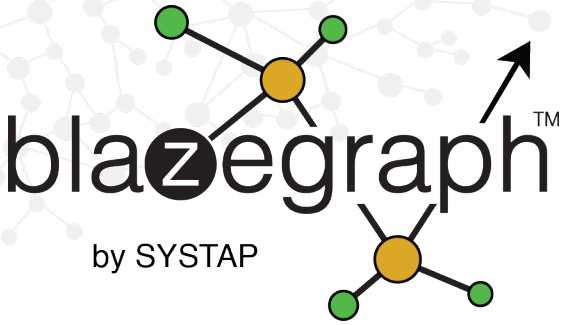
Dec 2015

Blazegraph GPU
1.0 Release
Appliance Available

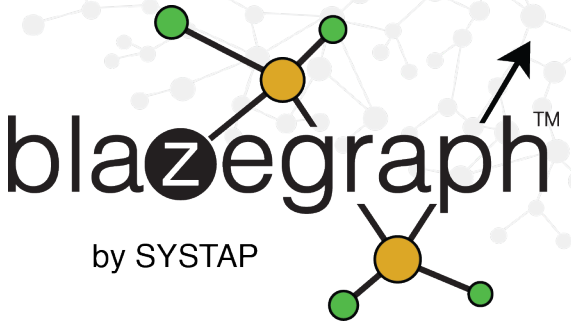
Q2 2016

GTC 2016
Initial Production Launch

blazegraph™ by SYSTAP



Stay in Touch



Technologies to solve the challenge of graphs at scale.

Brad Bebee, CEO

beebs@systap.com

http://blazegraph.com

@blazegraph