Blazegraph GPU-Accelerated Graph Query for Cyber Applications

- Blazegraph DASL

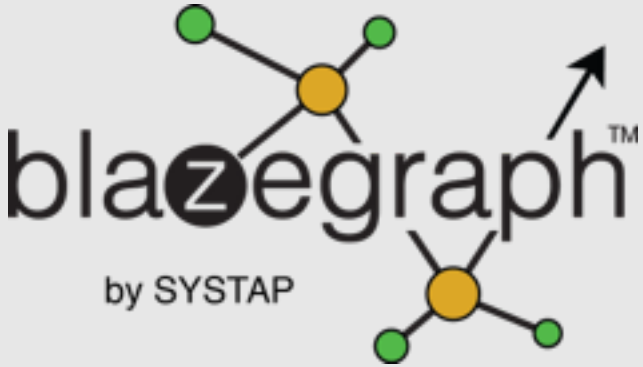
Jim Carbonaro, Sr Software Engineer, Blazegraph

April 2016

http://blazegraph.com/

Founded in
2006

Release Open Source Solution
"Bigdata"



"

We enable Fortune 500 companies, startups, governments and research organizations achieve business objectives with graph data.

Industry Recognition

2015 Finalist
Emerging Companies Summit

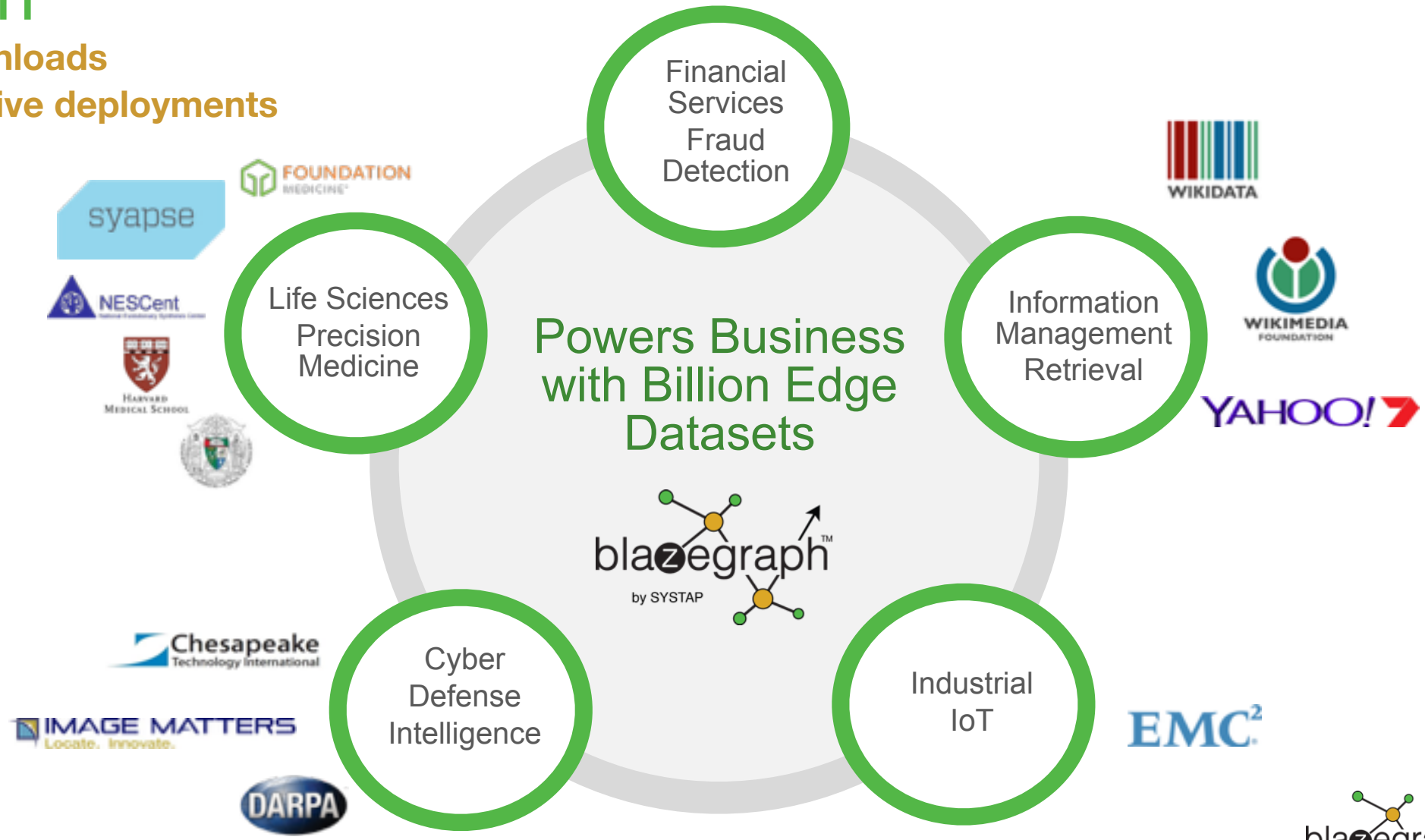
Big Data Startup Award Winner
2015 Big Data
Innovations Summit

innovation enterprise
Big Data ie.

Blazegraph™ Opensource Platform: Proven and in Production

200+ weekly downloads

Thousands of active deployments



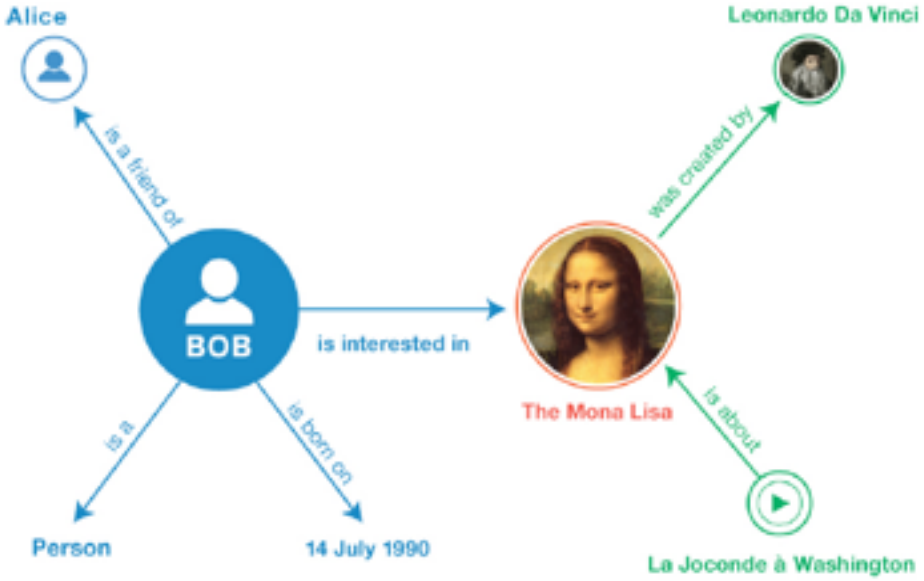
http://blazegraph.com/

blazegraph™ by SYSTAP

Exploding Data Volumes Requires New Approach for Relationship Insight

Graph databases designed to analyze diverse entities and relationships.
Today's datasets have billions of edges and nodes.

A (very small) Knowledge Graph



http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140225/example-graph.jpg

metaphacts

RESEARCH SPACE

blazegraph™ by SYSTAP

Scalability Challenge is the Blazegraph Opportunity

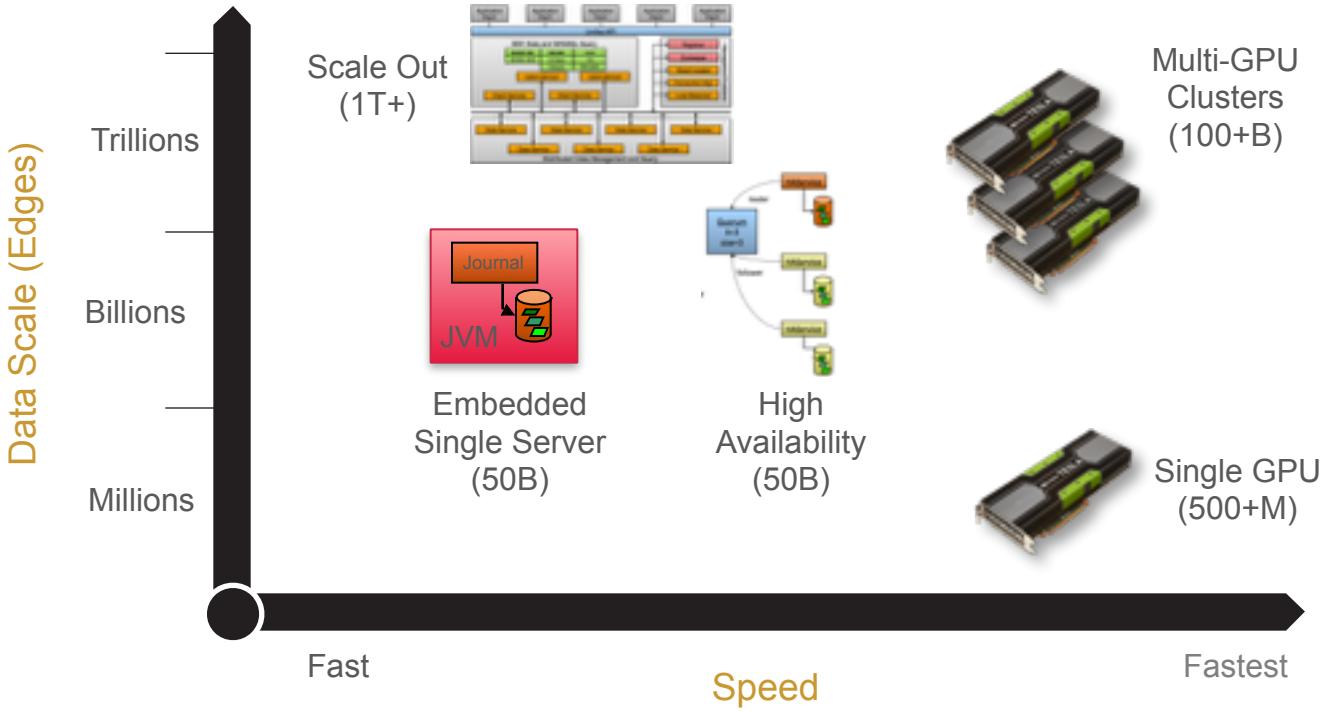
The Blazegraph Platform

Embedded and
Single Server Deployments

Enterprise
High Availability

Enterprise
Scale-out Architecture

Speed & Scale



blazegraph™ by SYSTAP

Blazegraph DASL for GPU Acceleration

Automation Enabling Analytics Coders to Optimize for GPU

Delivering ease of use of Spark and Scala for graph algorithms and predictive analytics… your analytic code can work on the Tesla GPUs and Blazegraph.

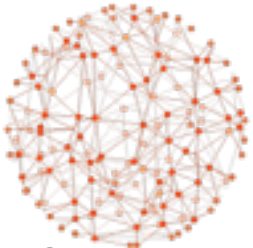
…with the speed of CUDA…

DASL Graph Algorithms

Spark Scala

DASL Translator

DASL Executor Multi-GPU Extension



…Graph and Machine Learning Algorithms…

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

…you can see why we call it "DASL"

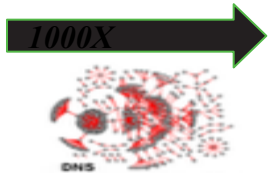
DASL your analytics

DASL Graph Algorithms

Spark Scala

DASL Translator

DASL Executor Multi-GPU Extension



Graph and Machine Learning Algorithms

- 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

Case Study - NetFlow Data Analysis

- What is NetFlow data?
  - NetFlow is a network protocol developed by Cisco for collecting IP traffic information and monitoring network traffic. http://www.solarwinds.com/what-is-netflow.aspx
- Sample NetFlow record

{"start_time":"2007-08-01 14:31:02.946",”duration":0.000, “src_addr":"122.166.71.110", “dst_addr":"9.155.118.136", "src_port":10822,"dst_port":13567,"protocol":"UDP “,"tcp_flags":".....","input_packets":1,"input_bytes":46}

- How is NetFlow collected?
  - Generated by routers and forwarded to collection point
  - Processed out of pcap records using tools such as nfcapd/nfdump

DASL GPU-Accelerated Algorithms Applicable to NetFlow

- Path algorithms (BFS, SSSP, APSP, CC) and their extensions (closeness, betweenness, etc)
- Ranking algorithms (cardinality, PageRank, BadRank, SecureRank)
- Clustering algorithms (canopy, k-means, Jaccard, community detection, etc)
- Many others!

Challenges of Network Flow Analysis

- The volume of NetFlow packets could not previously be efficiently analyzed or visualized to identify threats in real time
  - One hour of collection on a modest network can easily generate more than 100M NetFlow records
  - Visualizing this data to identify points of interest is nearly impossible due to the “hairball effect”, even after time windowing the data
  - These issues largely led to attempts at batch processing the data on large clusters of machines using Hadoop and/or Spark
  - Pushing the problem into the batch space puts organizations in the unenviable position of hoping to identify why and by whom their network was attacked last month rather than being able to identify an attack as it is occurring.

How does Blazegraph DASL overcome these challenges?

- Graph algorithms are inherently iterative. CPU architectures lack the memory bandwidth to efficiently run iterative algorithms over large data sets.
- DASL overcomes this by pushing the computation to the GPU where bandwidth can be 10x that available on CPU architectures.
- DASL makes the algorithm development process easily accessible to non-CUDA specialists

Anatomy of a DASL algorithm

```scala
/**
 * An interface for an SSSP algorithm implementation.
 *
 * @param wmat The graph, with weighted edges.
 *
 * @param srcId The starting vertex for the traversal.
 */
abstract class SSSPWorker(wmat: Matrix[Float], srcId: Int) extends DASLAlgorithm[Vector[Float]] {

    /** Return true if the algorithm assumes zero-weight self-edges */
    def assumesDiagonalIsZero() : Boolean

    /**
     * Initialize the current-distance vector such that it uses the semiring
     * (Float,min,plus) as default semiring for the following computation
     * populate the current-distance vector as follows: shortest path to the
     * source vertex is 0, all other elements will be set to the additive
     * identity of of the semiring, that is, Float.PositiveInfinity.
     */
    def init(daslContext : DASLContext ) : Vector[Float] = {
      daslContext.vectorGenerator .createVector(
                size         = wmat.numOfRows,
                dfltSemiRing = wmat.dfltSemiRing, // MinPlusSemiRing
                dfltValue    = wmat.dfltSemiRing.addIty, // PositiveInfinity
                data         = ( srcId, 0F ) )
    }

    /** Run the SSSP algorithm, returning the shortest paths. */
    def run( daslContext : DASLContext ) : Vector[Float]

}
```

```scala
/**
 * This SSSP algorithm is essentially the algebraic formulation of the
 * Bellman-Ford algorithm as given in Kepner and Gilbert's book "Graph
 * Algorithms in the Language of Linear Algebra" (see Algorithm 5.1 on
 * page 47 in the book)
 */
class SSSPWorker1(wmat: Matrix[Float], srcId: Int) extends SSSPWorker(wmat, srcId) {

    def assumesDiagonalIsZero() = true

    def run( daslContext : DASLContext ) : Vector[Float] =
    {
        val d = init( daslContext )
        var dd = d
        var i = 1
        while (i < wmat.numOfRows) {

            val oldDD = dd

            dd = dd _min_plus_ wmat  // general vector-matrix product
                                     // (using the min-plus semiring)

            daslContext.release( oldDD )

            if(log.isDebugEnabled()) log.debug("iteration=" + i)

            i = i + 1

        }

        return dd
    }
}
```

http://blazegraph.com/

blazegraph by SYSTAP

DASL vs Spark GraphX (PageRank)

Spark CPU Configuration

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)
16-r3.4xlarge	1	122	16	EC2 r3.4xlarge	16	122	1 x 320 SSD	Intel Xeon E5-2670 v2 (Ivy Bridge)	2.5	59.7

DASL vs Spark GraphX (PageRank)

DASL GPU Configuration

		Total RAM (G)	Total CUDA Cores	Model	CUDA Cores	GPU RAM (G)	GPU Chip	Memory Bandwidth (G/s)
Ubuntu Workstation (k40)	1	12	2880	K40	2880	12	Tesla K40c	288

DASL vs Spark GraphX (PageRank)

PageRank over Netflow Data (140051582 edges)

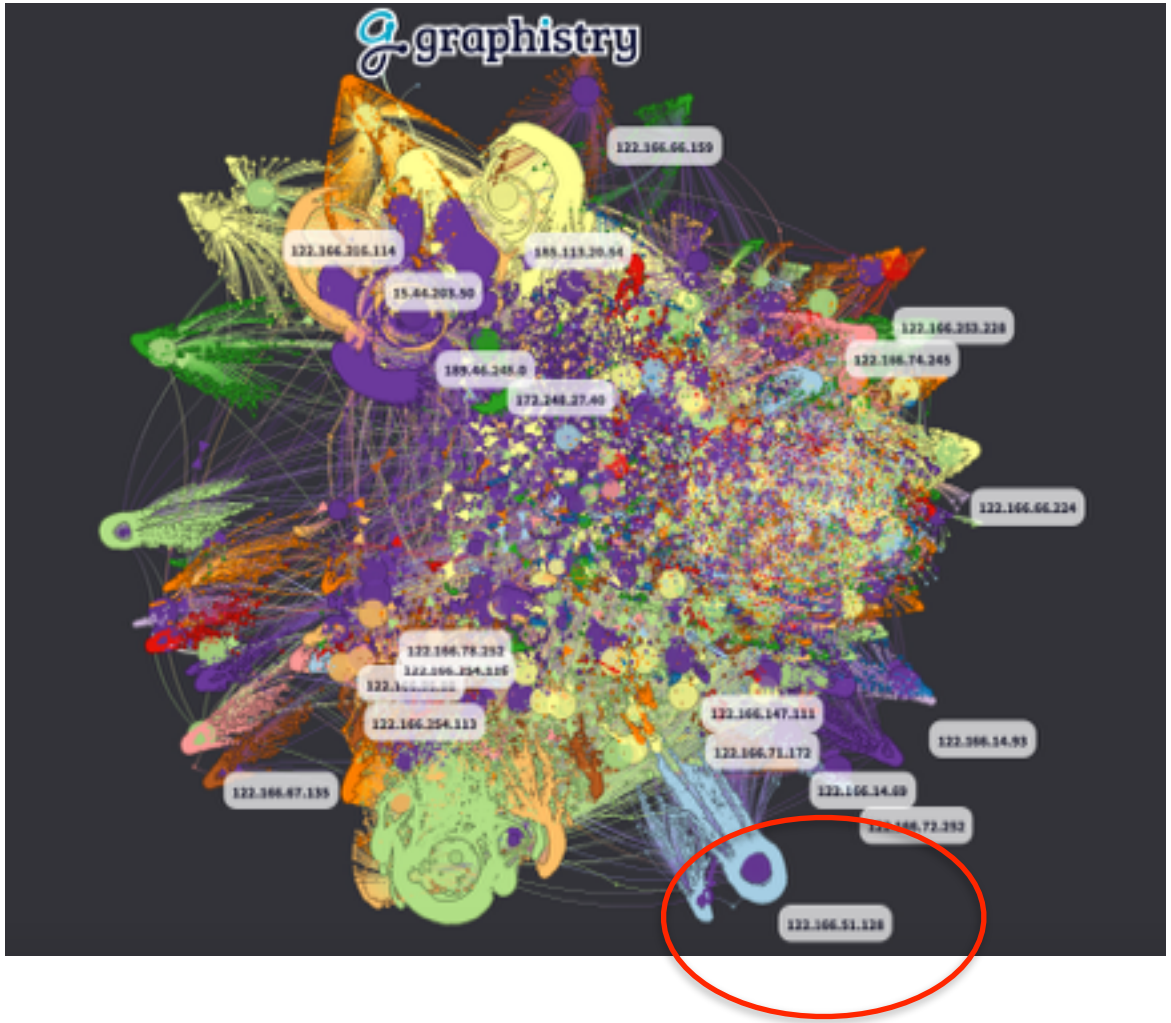
	Time(ms)	*Speedup ("DASL PR is faster than")*
DASL PR	***12841.00***	***0.000%***
Spark PR	***102836.93***	***800.84%***

https://traces.simpleweb.org/traces/netflow/netflow1/

Visualizing NetFlow Data

- Graphistry uses GPU acceleration to allow graphs containing 1M+ edges to be visualized and manipulated by an analyst
- Combining the capabilities of Graphistry and Blazegraph into an integrated suite can provide the capability to more quick identify anomalies over wider spans of time while maintaining contextual awareness

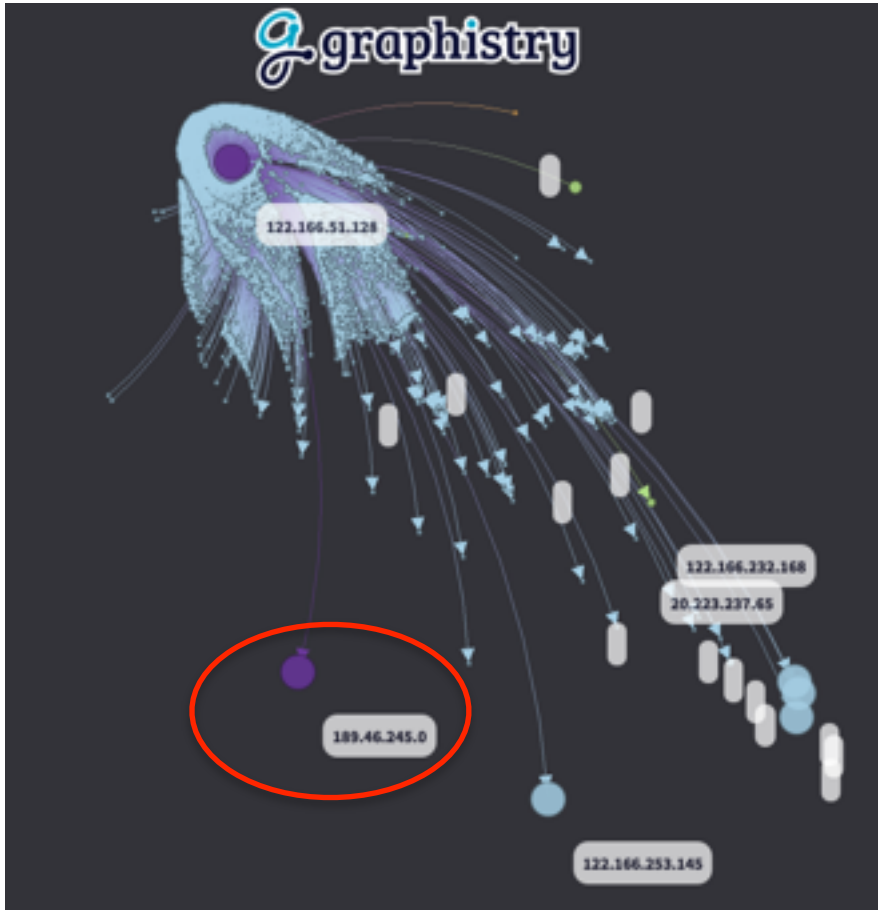
Initial Graph Filtered by PageRank using DASL

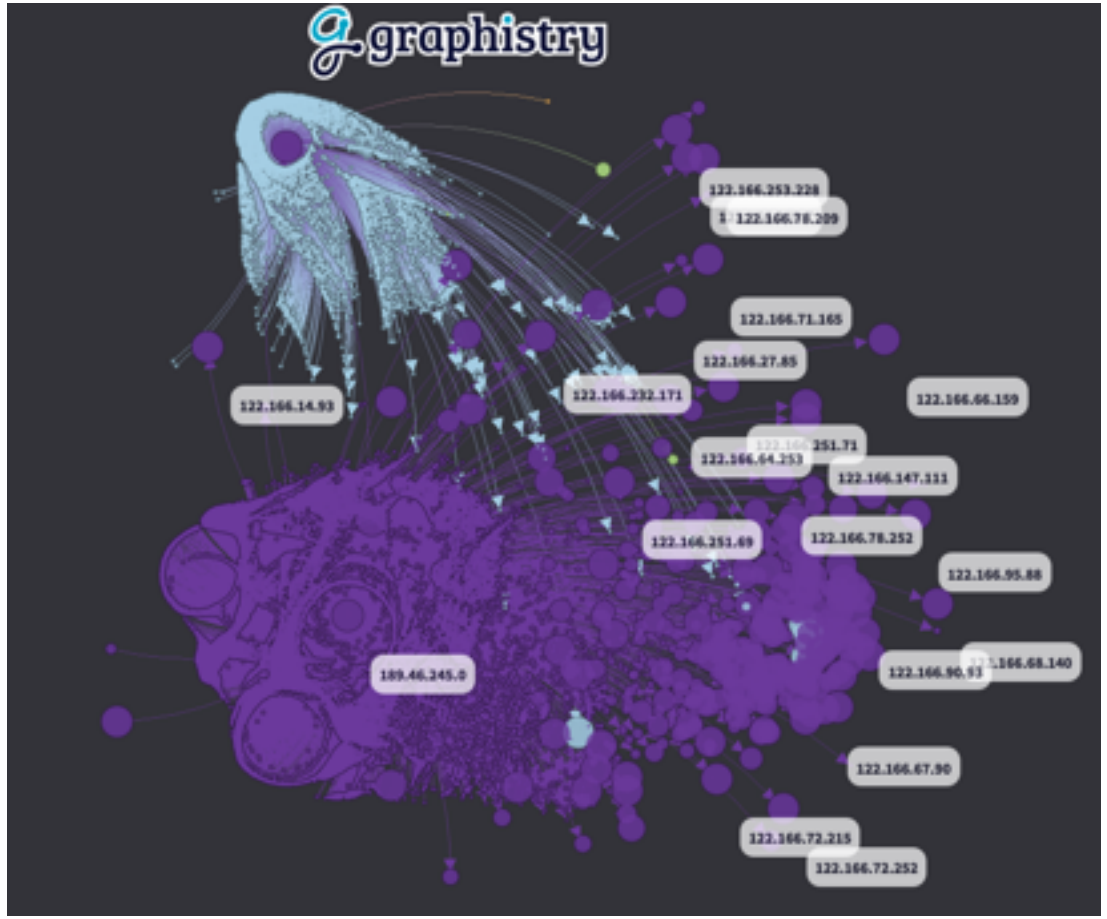


http://blazegraph.com/

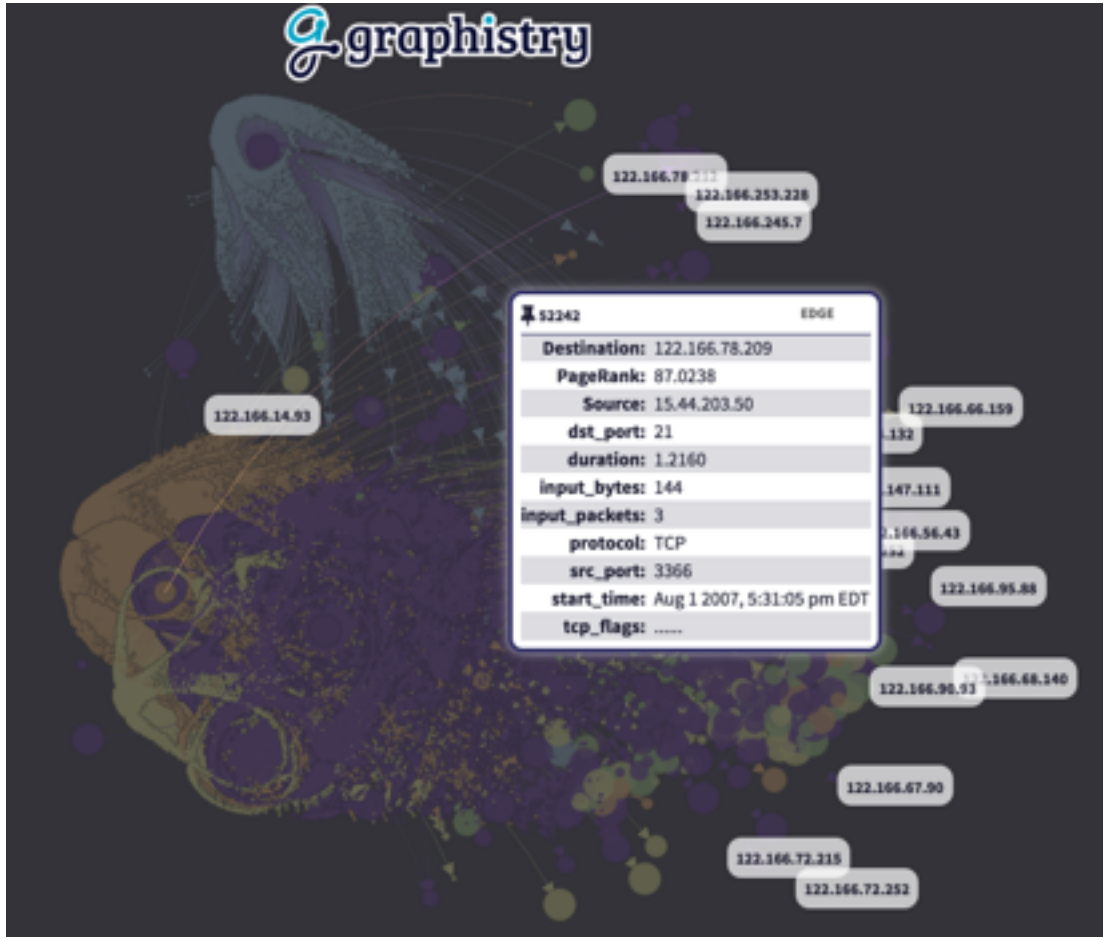
blazegraph by SYSTAP

Interactive Visual Query Session





Visual Analysis - Identification of Exfiltrated Traffic



http://blazegraph.com/

Stay in Touch

Jim Carbonaro,

Sr Software Engineer

jcarbonaro@blazegraph.com

http://blazegraph.com

@blazegraph