Massive Database Analysis on the Cloud with D4M

Jeremy Kepner

William Arcand, William Bergeron, Chansup Byun, Matthew Hubbell, Ben Landon, Andrew McCabe, Craig McNally, Peter Michaleas, Andrew Prout, Tony Rosa, Delsey Sherrill, Albert Reuther, and Chuck Yee

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Outline

- **Introduction**
  - *Webolution*
  - *As is, is OK*
  - *D4M*
- Technologies
- Results
- Demo
- Summary

Primordial Web

Kepner & Beaudry 1992, Visual Intelligence Corp (now GE Intelligent Platforms)

Browser (html):		Server (http):		Database (sql):
NCSA Mosaic™ 3.0	http put ⇄ http get	Gopher Language: perl	SQL ⇄ data	SYBASE
Client		**Server**		**Database**
				

- Browser GUI? HTTP for files? Perl for analysis? SQL for data?
- A lot of work just to view data.
- Won't catch on.

Cambrian Web

Browser (html): | **Server (http):** | **Database (sql):**

http put / http get

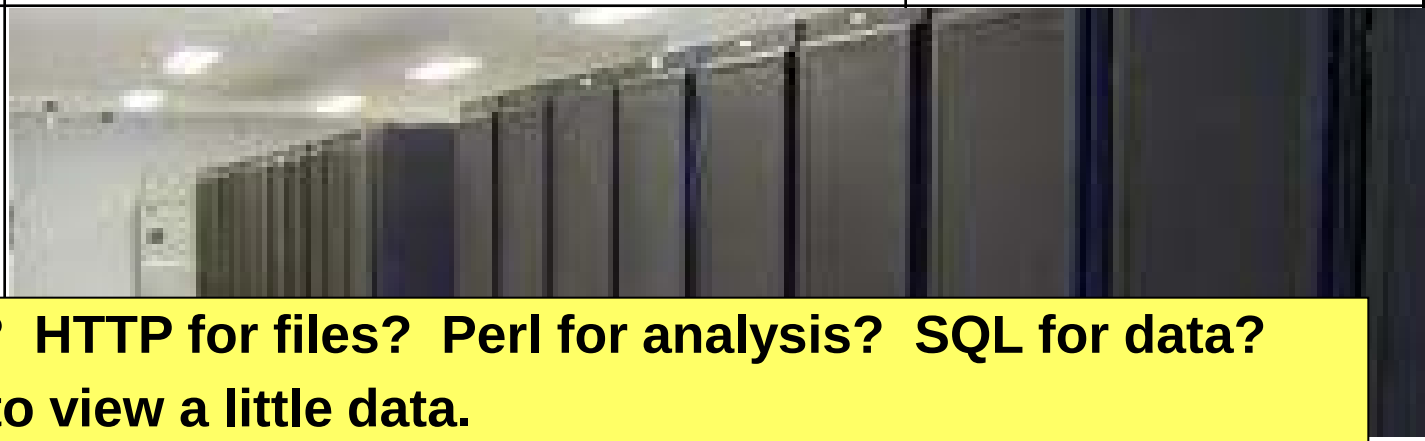
SQL / data

Apache Windows Server

Language: Java, php, perl, python, Ruby

SYBASE, ORACLE, IBM DB2, Microsoft SQL Server, MySQL

Client | **Server** | **Database**



- Browser GUI? HTTP for files? Perl for analysis? SQL for data?
- A lot of work to view a little data.
- ~~Won't catch on.~~

Modern Web

Game (data):

http put

http get

Server (http):

Apache Windows Server

Language:

Java php perl python Ruby

java

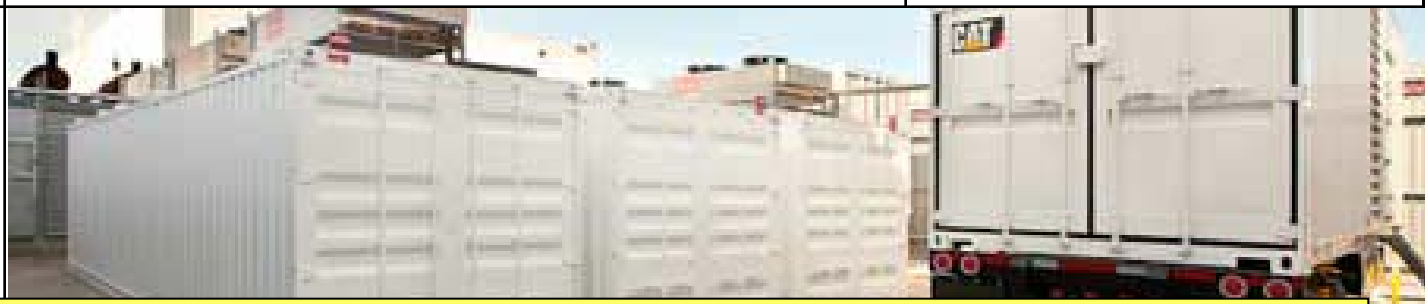
data

Database (triples):

BigTable riak Cassandra HBASE

Client

Server

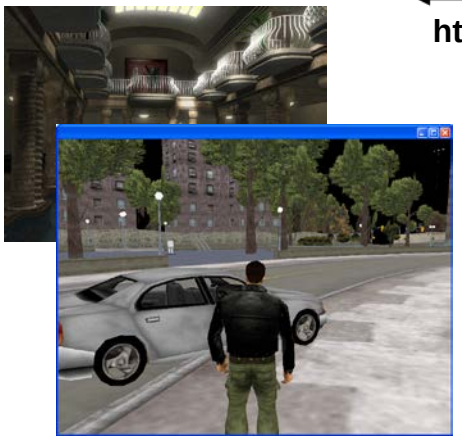


Database

- Game GUI! HTTP for files? Perl for analysis? Triples for data!
- A lot of work to view a lot of data.
- Great view. Massive data.

Modern Web

Game (data):



http put

http get

Server (http):

Apache Windows Server

Language:

Java php perl python Ruby

java

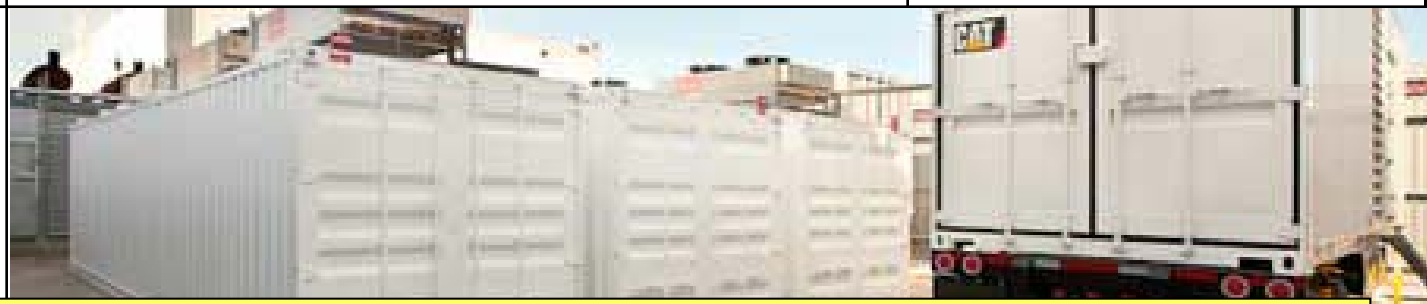
data

Database (triples):

BigTable riak Cassandra HBASE

Client

Server

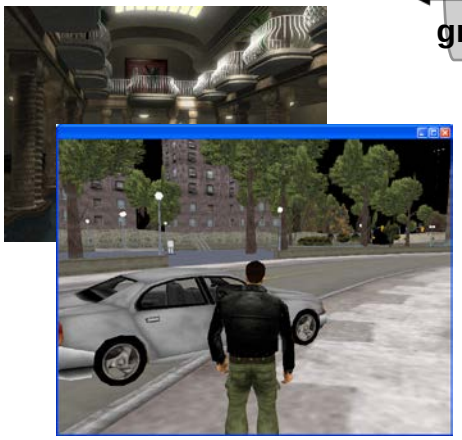


Database

- Game GUI! HTTP for files? Perl for analysis? Triples for data!
- A lot of work to view a lot of data. Missing middle.
- Great view. Massive data.

Future Web?

Game (data):



graphs

graphs

portal

Server (files):

lustre®

Language:

graphs

graphs

portal

Database (triples):

BigTable

riak

Cassandra

HBASE

Client

Server

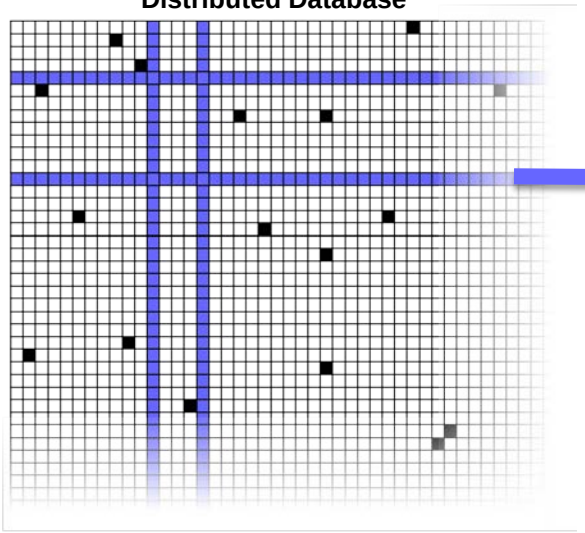
Database

- Game GUI! Fileserver for files! D4M for analysis! Triples for data!
- A little work to view a lot of data. Securely.
- Great view. Massive data.

D4M: "Databases For Matlab"

Triple Store

Distributed Database



Triple store are high performance distributed databases for heterogeneous data

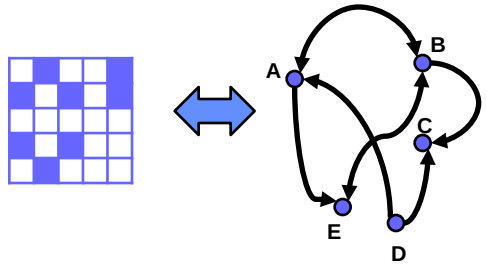
D4M

Dynamic Distributed Dimensional Data Model

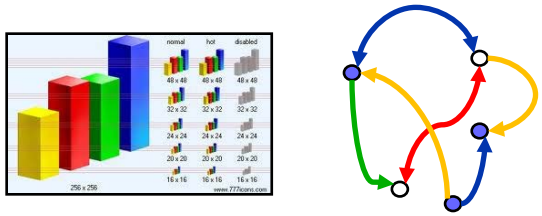
Query:
Alice
Bob
Cathy
David
Earl

Associative Arrays

Numerical Computing Environment



A D4M query returns a sparse matrix or graph from Cloudbase…



…for statistical signal processing or graph analysis in MATLAB

D4M binds Associative Arrays to Triple Store, enabling rapid prototyping of data-intensive cloud analytics and visualization

Outline

- Introduction
- **Technologies**
  - ***Cloud Software***
  - *D4M*
- Results
- Demo
- Summary

Recap: Cloud Computing Concepts

Data Intensive Computing

- Compute architecture for large scale data analysis
  - Billions of records/day, trillions of stored records, petabytes of storage
    - Google File System 2003
    - Google MapReduce 2004
    - Google BigTable 2006
- Design Parameters
  - Performance and scale
  - Optimized for ingest, query and analysis
  - Co-mingled data
  - Relaxed data model
  - Simplified programming
- Community:

Utility Computing

- Compute services for outsourcing IT
  - Concurrent, independent users operating across millions of records and terabytes of data
    - IT as a Service
    - Infrastructure as a Service (IaaS)
    - Platform as a Service (PaaS)
    - Software as a Service (SaaS)
- Design Parameters
  - Isolation of user data and computation
  - Portability of data with applications
  - Hosting traditional applications
  - Lower cost of ownership
  - Capacity on demand
- Community:

Recap: Cloud Computing Concepts

Data Intensive Computing

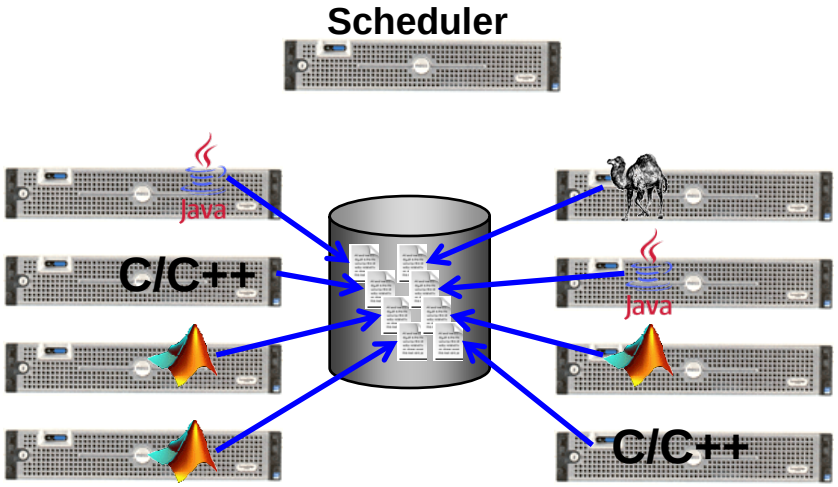
- Compute architecture for large scale data analysis
  - Billions of records/day, trillions of stored records, petabytes of storage
    - Google File System 2003
    - Google MapReduce 2004
    - Google BigTable 2006
- Design Parameters
  - Performance and scale
  - Optimized for ingest, query and analysis
  - Co-mingled data
  - Relaxed data model
  - Simplified programming
- Community:

Utility Computing

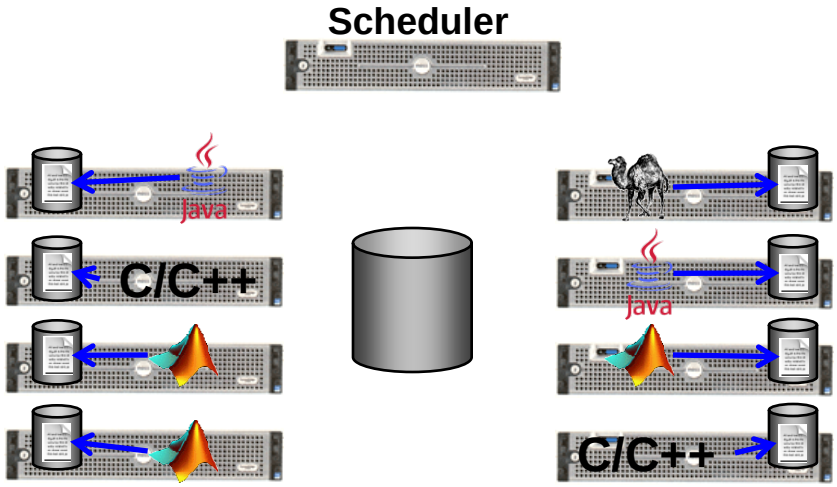
- Compute services for outsourcing IT
  - Concurrent, independent users operating across millions of records and terabytes of data
    - IT as a Service
    - Infrastructure as a Service (IaaS)
    - Platform as a Service (PaaS)
    - Software as a Service (SaaS)
- Design Parameters
  - Isolation of user data and computation
  - Portability of data with applications
  - Hosting traditional applications
  - Lower cost of ownership
  - Capacity on demand
- Community:

Advantages of Data Intensive Cloud

Traditional:
Data from central store to compute nodes

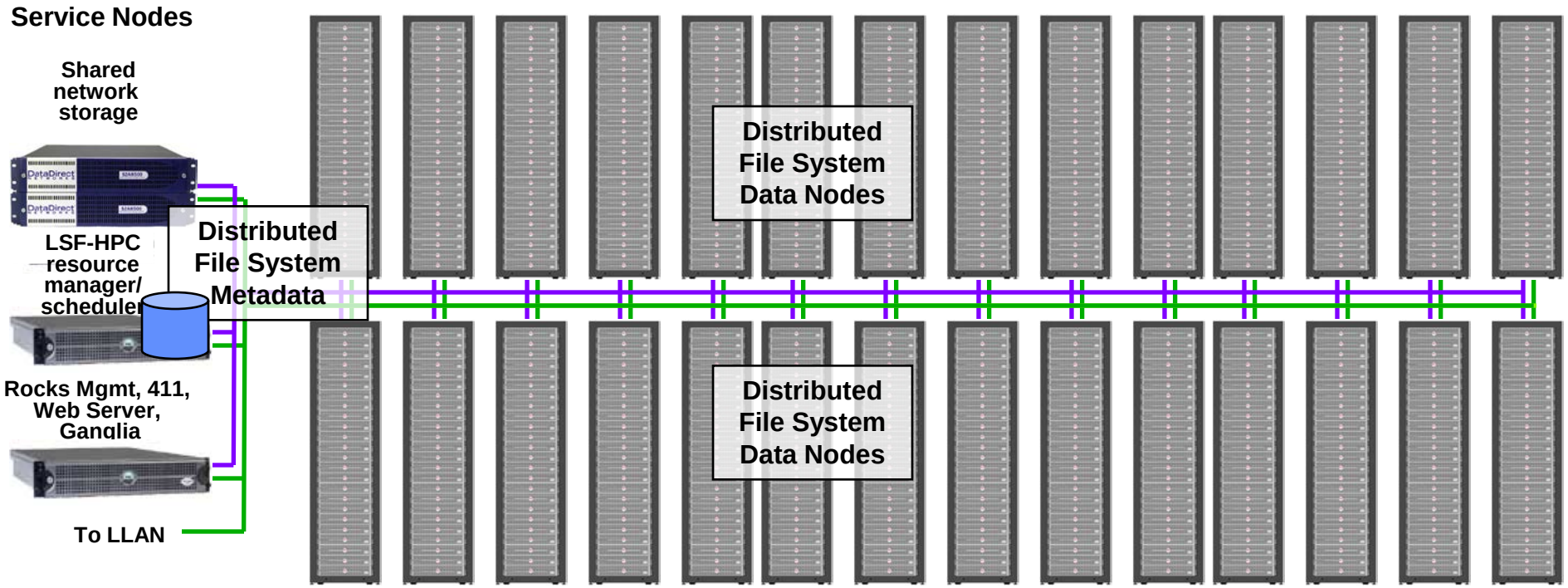


Cloud:
Data replicated on nodes, computation sent to nodes



- Cloud computing moves computation to data
  - Good for applications where time is dominated by reading from disk
- Replaces expensive shared memory hardware and proprietary database software with cheap clusters and open source
  - Scalable to hundreds of nodes

Distributed Cloud File Systems on TX-2500 Cluster



432 DELL PowerEdge 2850

Dual 3.2 GHz EM64-T Xeon (P4)
8 GB RAM memory
Two Gig-E Intel interfaces
Infiniband interface
Six 300-GB disk drives

- 432+5 Nodes
- 864+10 CPUs
- 3.4 TB RAM
- 0.78 PB of Disk
- 28 Racks

MIT-LL Cloud	Hadoop DFS	Sector
Number of nodes used	350	350
File system size	298.9 TB	452.7 TB
Replication factor	3	2

Outline

- Introduction
- **Technologies**
  - *Cloud Software*
  - ***D4M***
- Results
- Demo
- Summary

Multi-Dimensional Associative Arrays

- **Extends associative arrays to 2D and mixed data types**

```
A('alice ','bob ') = 'talked '
```

or

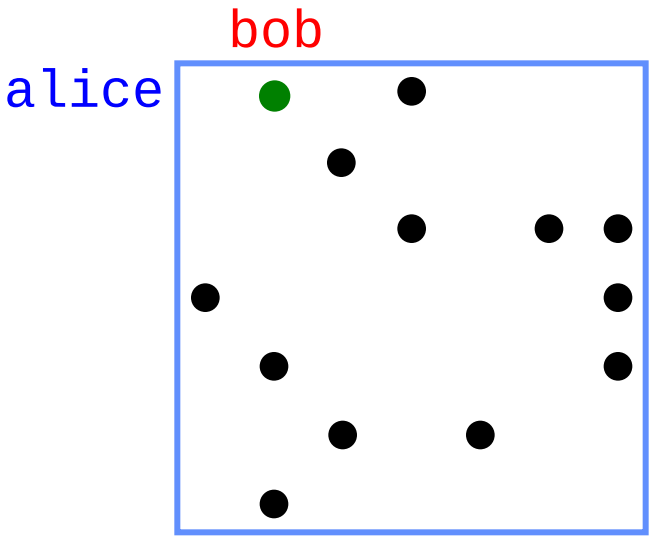
```
A('alice ','bob ') = 47.0
```

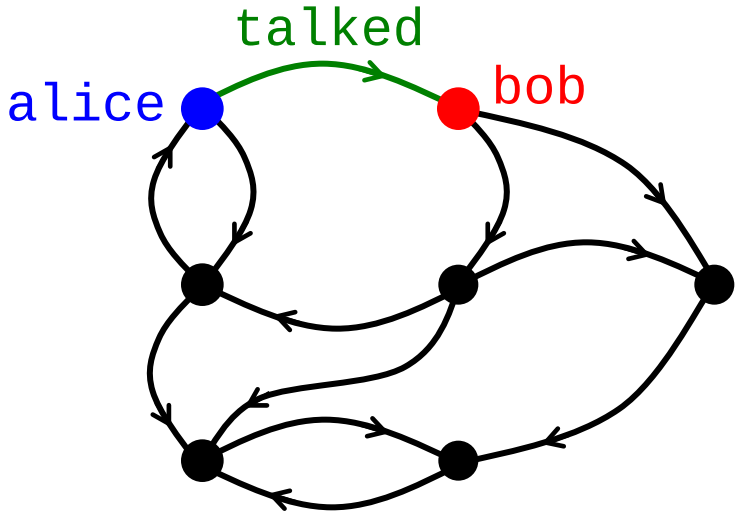
- **Key innovation: 2D is 1-to-1 with triple store**

```
('alice ','bob ','talked ')
```

or

```
('alice ','bob ',47.0)
```





- **Associative arrays unify four viewpoints into one concept**

Composable Associative Arrays

- **Key innovation: mathematical closure**
  - **all associative array operations return associative arrays**

- **Enables composable mathematical operations**

```
A + B      A - B      A & B      A|B      A*B
```

- **Enables composable query operations via array indexing**

```
A('alice bob ',:)    A('alice ',:)   A('al* ',:)
A('alice : bob ',:)  A(1:2,:)        A == 47.0
```

- **Simple to implement in a library (~2000 lines) in programming environments with: 1st class support of 2D arrays, operator overloading, sparse linear algebra**

- **Complex queries with ~50x less effort than Java/SQL**
- **Naturally leads to high performance parallel implementation**

Associative Array Algebra

- Keys and values are from the infinite strict totally ordered set $\mathbb{S}$
- Associative array $A(\mathbf{k}) : \mathbb{S}^d \rightarrow \mathbb{S}$, $\mathbf{k}=(k^1,\ldots,k^d)$, is a partial function from d keys (typically 2) to 1 value, where

  $A(\mathbf{k}_i) = v_i$ and $\varnothing$ otherwise
- Binary operations on associative arrays $A_3 = A_1 \oplus A_2$, where $\oplus = \cup_{f()}$ or $\cap_{f()}$, have the properties
  - If $A_1(\mathbf{k}_i) = v_1$ and $A_2(\mathbf{k}_i) = v_2$, then $A_3(\mathbf{k}_i)$ is

    $v_1 \cup_{f()} v_2 = f(v_1,v_2)$ or $v_1 \cap_{f()} v_2 = f(v_1,v_2)$
  - If $A_1(\mathbf{k}_i) = v$ or $\varnothing$ and $A_2(\mathbf{k}_i) = \varnothing$ or v, then $A_3(\mathbf{k}_i)$ is

    $v \cup_{f()} \varnothing = v$ or $v \cap_{f()} \varnothing = \varnothing$

- **High level usage dictated by these definitions**
- **Deeper algebraic properties set by the collision function f()**
- **Frequent switching between "algebras"**

Universal "Exploded" Schema

Input Data

Time	src_ip	domain	dest_ip
2001-01-01	a		a
2001-01-02	b	b	
2001-01-03		c	c

Triple Store Table: Ttranspose

	2001-01-01	2001-01-02	2001-01-03
src_ip/a	1		
src_ip/b		1	
domain/b		1	
domain/c			1
dest_ip/a	1		
dest_ip/c			1

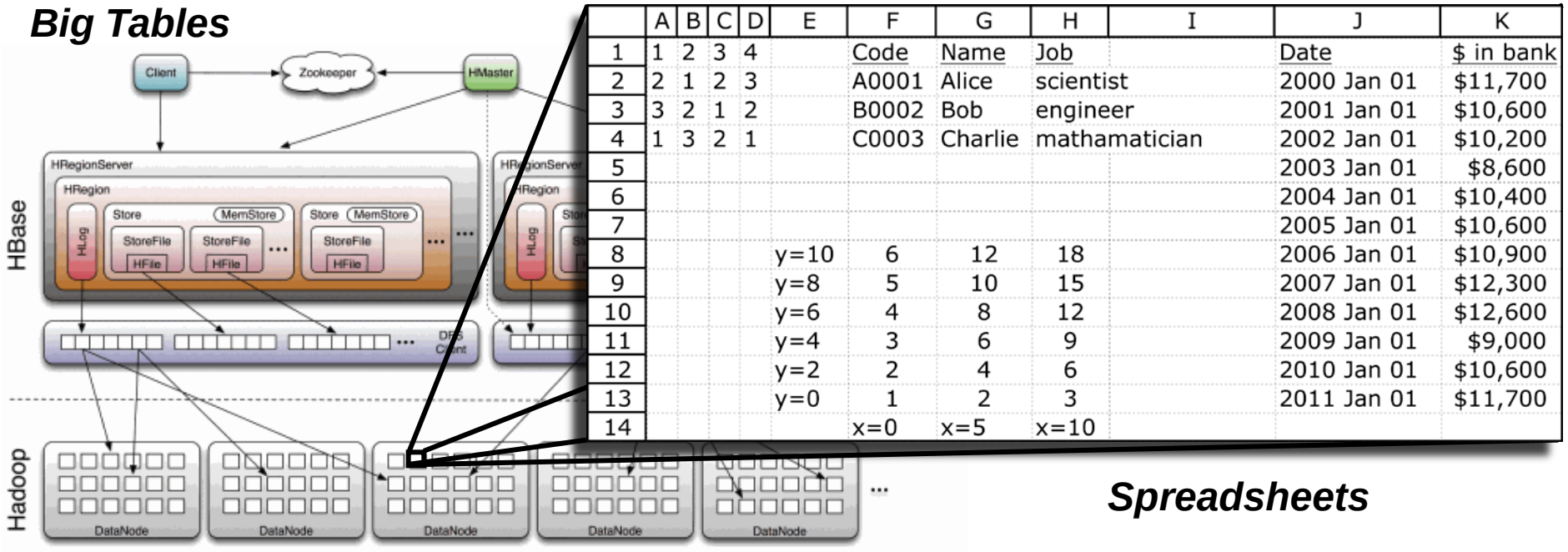
Triple Store Table: T

	src_ip/a	src_ip/b	domain/b	domain/c	dest_ip/a	dest_ip/c
2001-01-01	1				1	
2001-01-02		1	1			
2001-01-03				1		1

Key Innovations

- Handles all data into a *single* table representation
- Transpose pairs allows quick look up of *either* row or column

Unifies Spreadsheets and Big Tables



	A	B	C	D	E	F	G	H	I	J	K
1	1	2	3	4		Code	Name	Job		Date	$ in bank
2	2	1	2	3		A0001	Alice	scientist		2000 Jan 01	$11,700
3	3	2	1	2		B0002	Bob	engineer		2001 Jan 01	$10,600
4	1	3	2	1		C0003	Charlie	mathamatician		2002 Jan 01	$10,200
5										2003 Jan 01	$8,600
6										2004 Jan 01	$10,400
7										2005 Jan 01	$10,600
8					y=10	6	12	18		2006 Jan 01	$10,900
9					y=8	5	10	15		2007 Jan 01	$12,300
10					y=6	4	8	12		2008 Jan 01	$12,600
11					y=4	3	6	9		2009 Jan 01	$9,000
12					y=2	2	4	6		2010 Jan 01	$10,600
13					y=0	1	2	3		2011 Jan 01	$11,700
14						x=0	x=5	x=10			

- ***Spreadsheets*** **are the most commonly used analytical structure on Earth (100M users/day?)**
- **Big Tables (Google, Amazon, Facebook, …) store most of the analyzed data in the world (Exabytes?)**
- **Simultaneous diverse data: strings, dates, integers, reals, …**
- **Simultaneous diverse uses: matrices, functions, hash tables, databases, …**

Outline

- Introduction
- Technologies
- **Results**
  - ***Network Monitoring***
  - ***Text Query***
  - ***Insert Performance***
- Demo
- Summary

Stats Diagram

Triple Store Table: T

Row	Key (time)	src_ip/a	src_ip/b	src_ip/c	src_ip/d	domain/a	domain/b	domain/c	domain/d	dest_ip/a	dest_ip/b	dest_ip/c	dest_ip/d	Recv/a	Recv/b	Recv/c	Recv/d	Recv/e
1	2001-10-01 01 01 00	■				■	■	■	■									
2	2001-10-01 01 02 00			■	■					■	■	■	■					
3	2001-10-01 01 03 00	■												■	■	■	■	■
4	2001-10-01 01 04 00			■														
5	2001-10-01 01 05 00	■				■	■	■	■									
6	2001-10-01 01 06 00			■														

Associative Array: A

- Copy a set of rows from T into associative array A
- Perform the following statistical calculations on A
  - Column count: how many times each column appears in A
  - Column type count: how many times each column type appears in A
  - Column covariance: how many times a each pair of columns in A appear in the same row together
  - Column covariance: how many times a each pair of column types in A appear in the same row together

- Good for identifying column types, gaps, clutter, and correlations

Stats Implementation

- **Define a set of rows**

```
r = '2001-01-01 01 02 00,2001-01-01 01 03 00, 2001-01-01 01 04 00,'
```

- **Copy rows from table to associative array and convert `'1'` to `1`**

```
A = double(logical(T(r,:)))
A = A(:,'src_ip/ *,domain/ *,dest_ip/ *,')
```

- **Find popular columns counts**

```
sum(A,1) > 200
```

- **Find popular pairs**

```
A' * A > 200
```

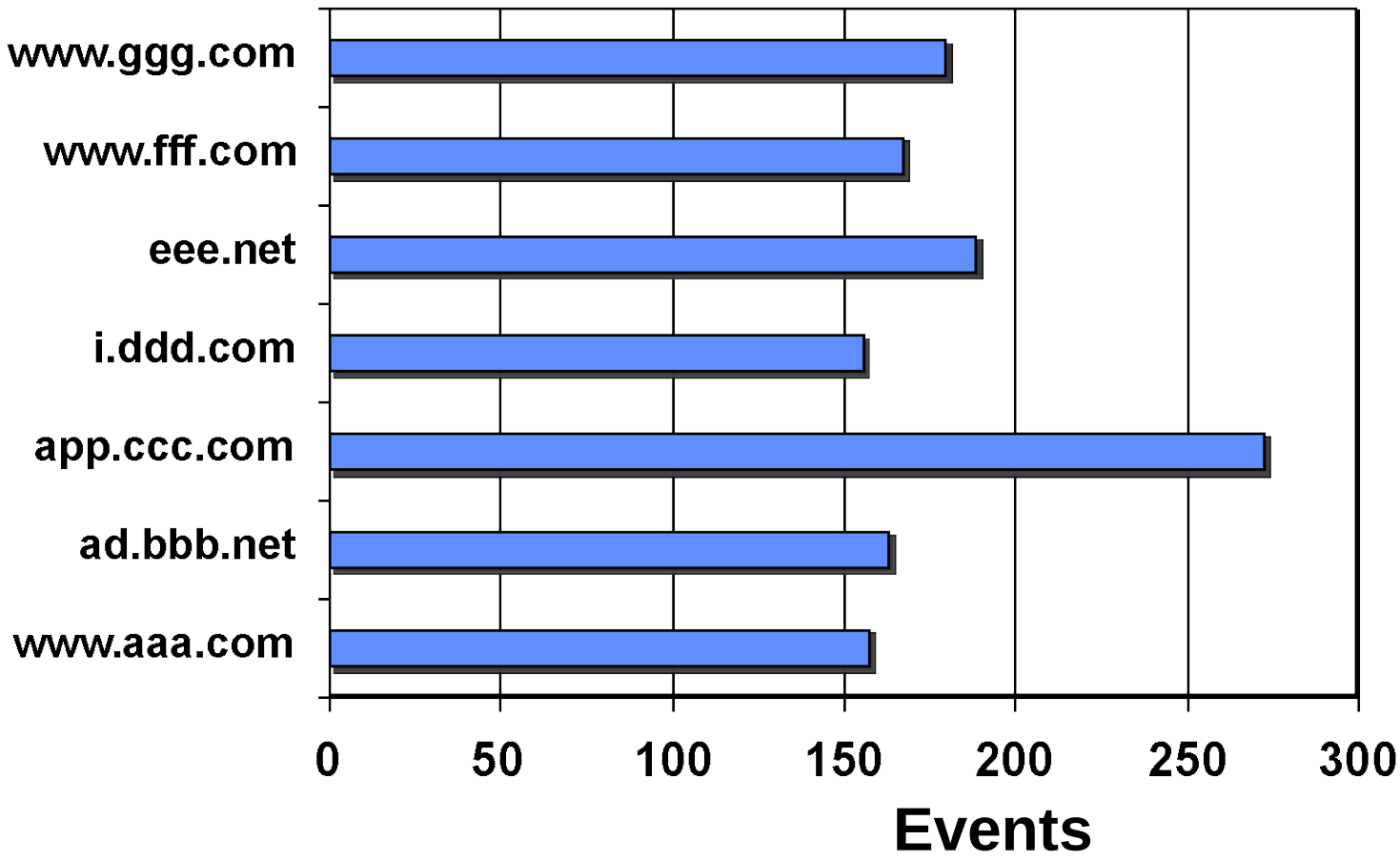
or

```
sqIn(A) > 200
```

- **Find domains with many dest IPs**

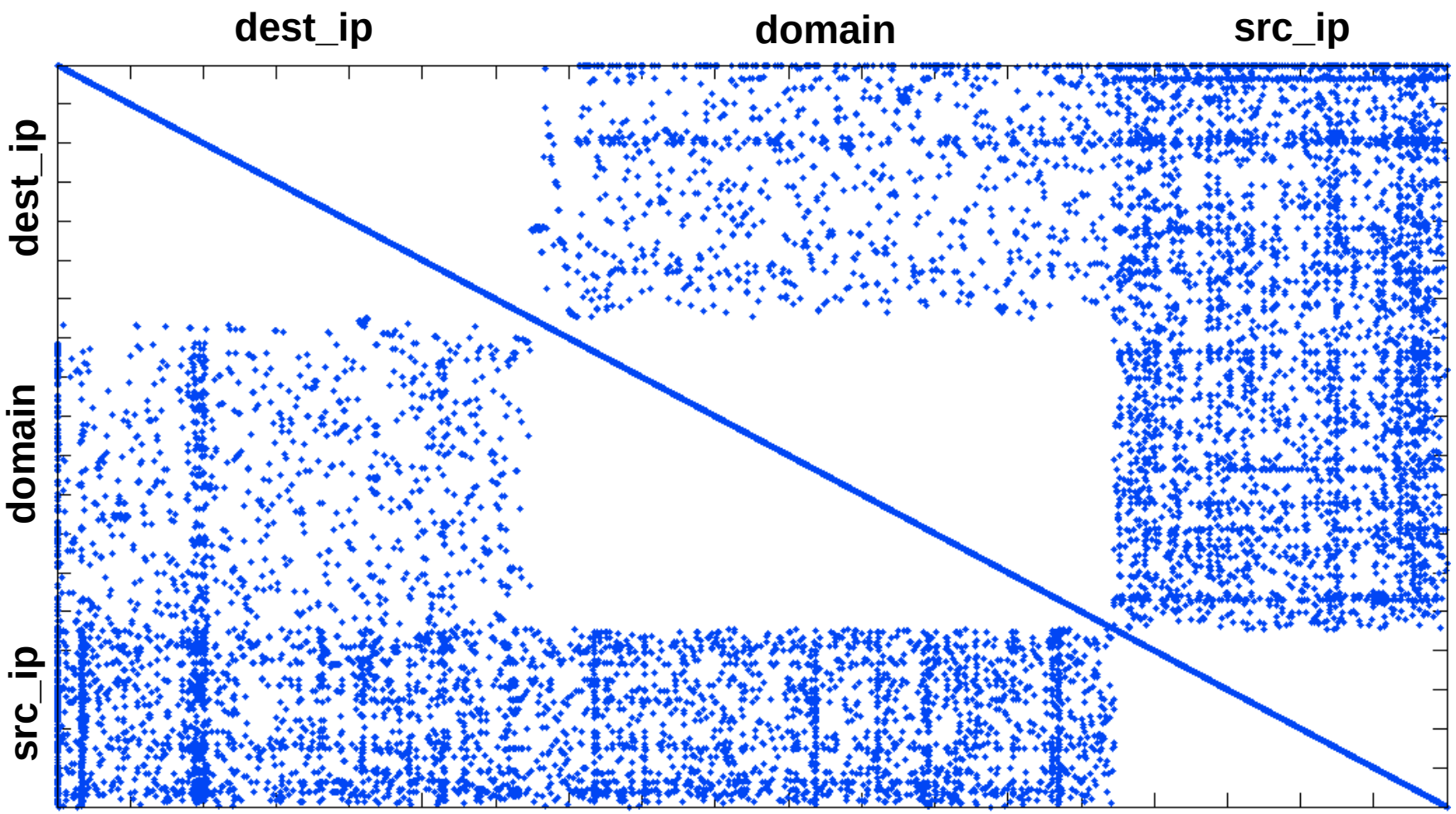
```
sum(double(logical(sqIn(A))),2) > 3
```

Count



- Very easy to get elementary count info necessary for finding clutter and anomalies

Covariance

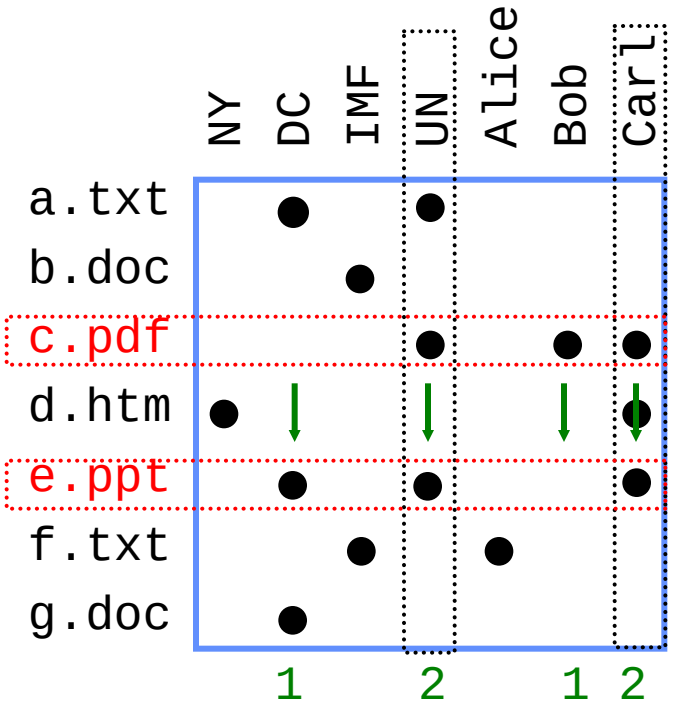


- Adjacency matrix a natural result of covariance calculation

Facet Search

- Core analytic of SKS
- Gives keyword distribution of a set of documents that share a common keyword(s)
  - Provides useful guide to what keyword to select next
- Currently implemented with several hundreds of lines of Java/SQL
- Associative array implementation has 1 line

Facet Search Algorithm



- Associative array relates documents to place, org and person entities

$$A(x,y) : \mathbf{S}^{NxM} \rightarrow \mathbf{R}$$

- Facets y_1=UN, y_2=Carl
- Documents that contain both

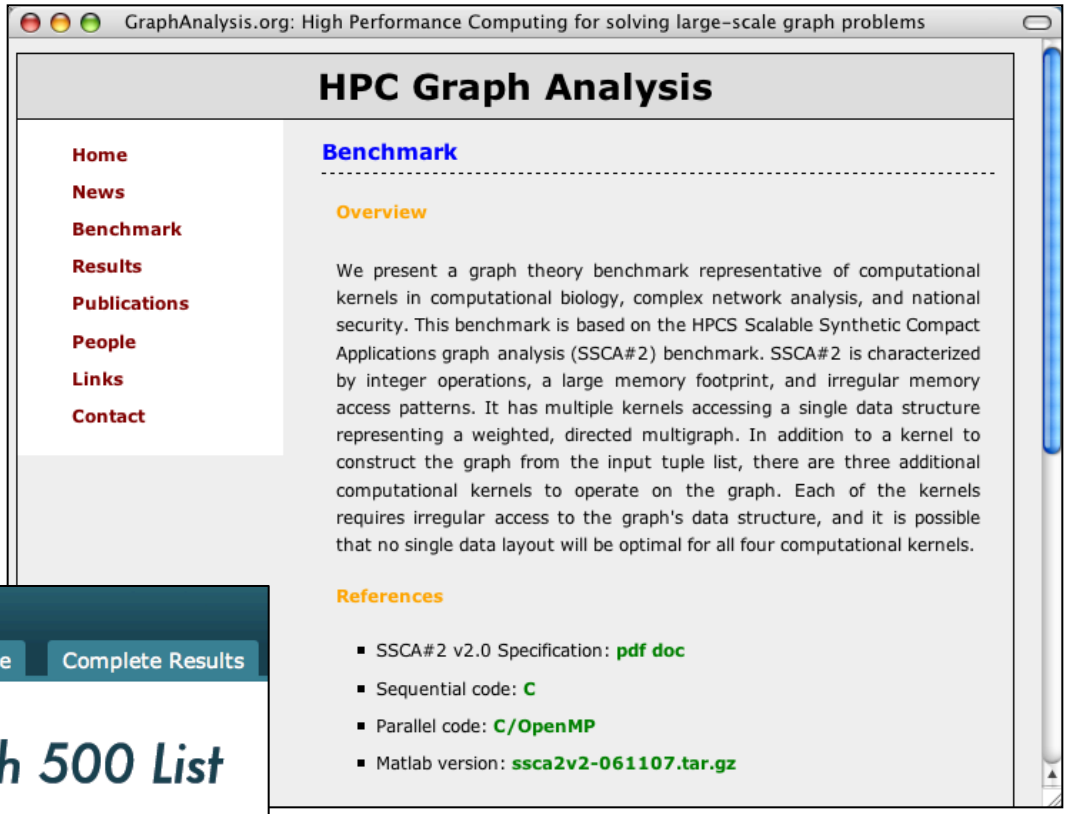
$$\underline{A}(:,y_1) \;\&\; \underline{A}(:,y_2)$$

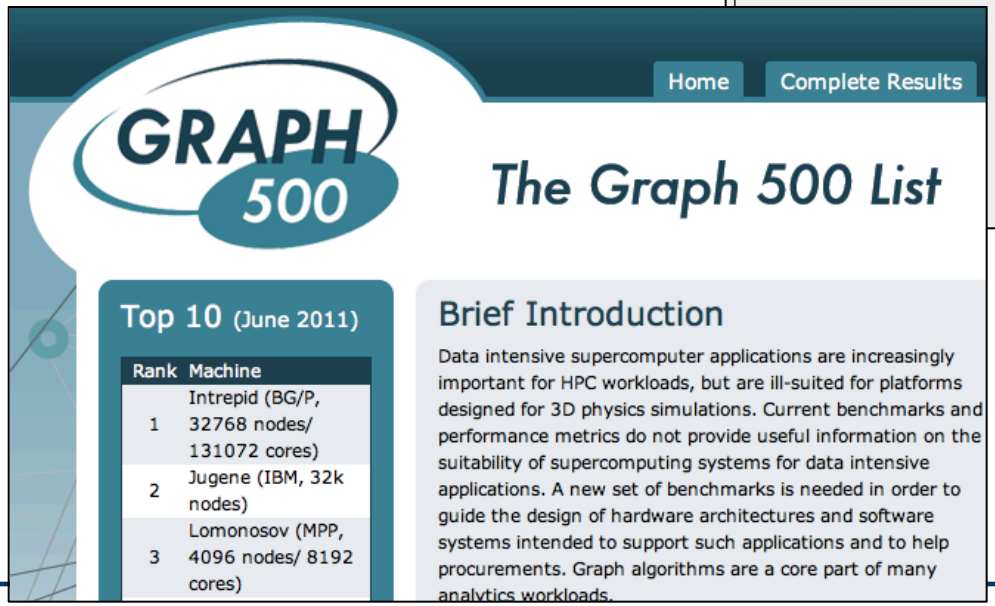
- Entity counts in the above set of documents obtained via matrix multiply

$$(\underline{A}(:,y_1) \;\&\; \underline{A}(:,y_2))^t \; A$$

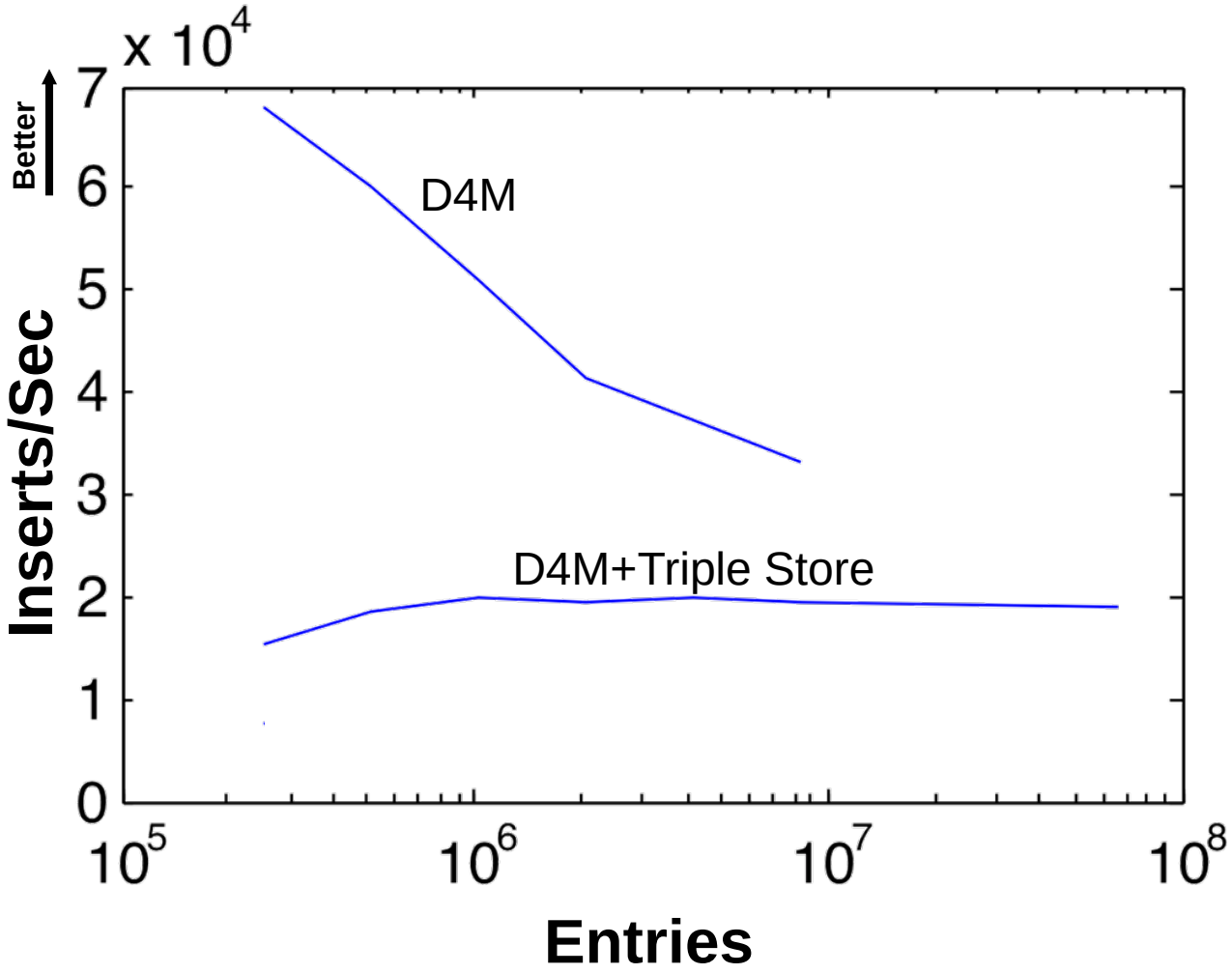
Graph Analysis/Graph500 Benchmark

- **Scalable benchmark specified by graph community**
- **Goal**
  - **Stress parallel computer architecture**
- **Key data**
  - **Very large Power law graph**
- **Kernels**
  - **Data Generator**
  - **Ingest**

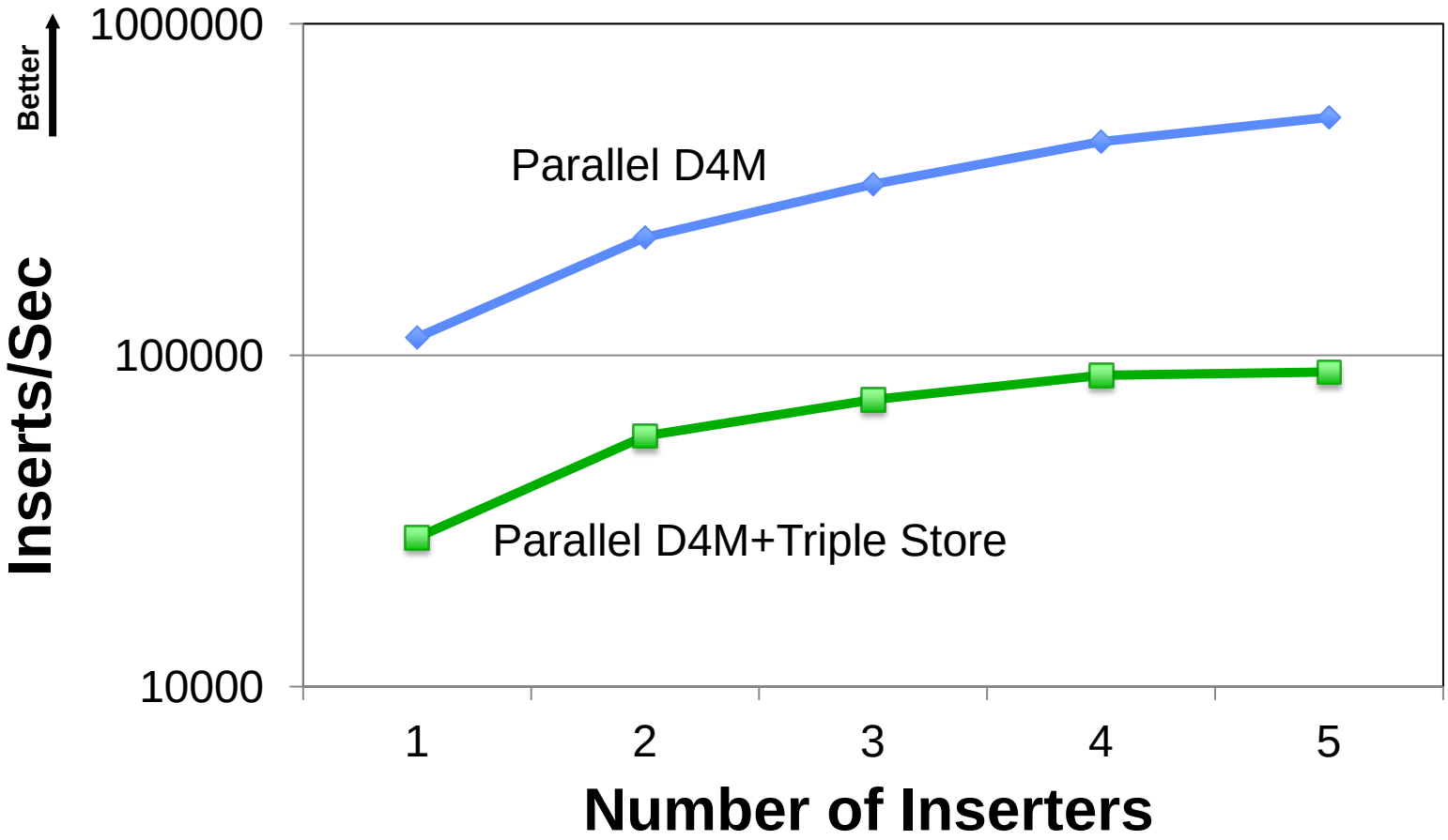




Single Node Insert Rate



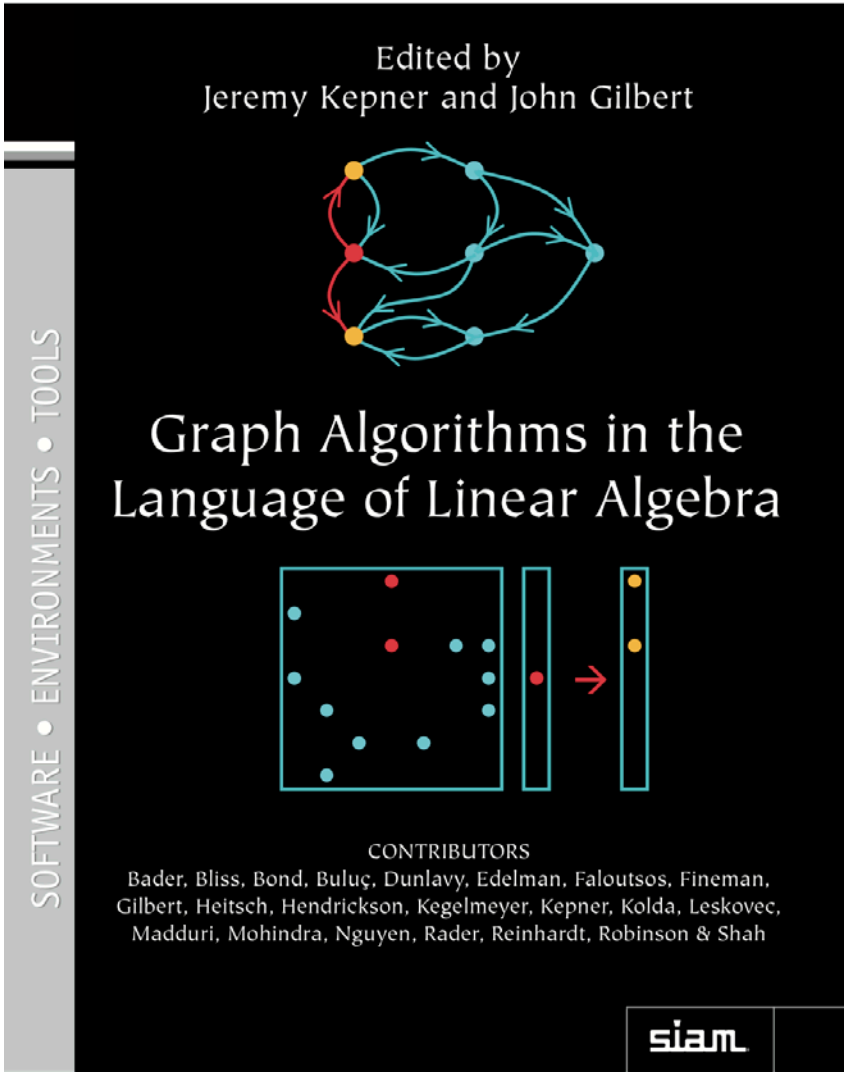
Parallel D4M + Single Node Triple Store Graph500 insert rate



- **Parallel D4M provides 500K memory inserts/sec on 1 node**
- **Parallel D4M + Triple Store provides 100K disk inserts/sec on 1 node**
- **Parallel D4M enables exploiting full power of Triple Store from D4M**

Reference

- **Book: "Graph Algorithms in the Language of Linear Algebra"**
- **Editors: Kepner (MIT-LL) and Gilbert (UCSB)**
- **Contributors**
  - **Bader (Ga Tech)**
  - **Bliss (MIT-LL)**
  - **Bond (MIT-LL)**
  - **Dunlavy (Sandia)**
  - **Faloutsos (CMU)**
  - **Fineman (CMU)**
  - **Gilbert (UCSB)**
  - **Heitsch (Ga Tech)**
  - **Hendrickson (Sandia)**
  - **Kegelmeyer (Sandia)**
  - **Kepner (MIT-LL)**
  - **Kolda (Sandia)**
  - **Leskovec (CMU)**
  - **Madduri (Ga Tech)**
  - **Mohindra (MIT-LL)**
  - **Nguyen (MIT)**
  - **Rader (MIT-LL)**
  - **Reinhardt (Microsoft)**
  - **Robinson (MIT-LL)**
  - **Shah (UCSB)**



Summary

- Web evolution has resulted in a new class of technologies for
  - Display (game interfaces)
  - Analysis (D4M)
  - Storage (triple stores)

- D4M is a novel technology that allows complex analytics to be implement with significantly less effort that traditional approaches

- D4M is built on composable associative arrays which admit linear algebraic manipulation