TITAN

BIG GRAPH DATA WITH CASSANDRA

#TITANDB #CASSANDRA12

MATTHIAS BROECHELER, CTO

AUGUST VIII, MMXII

AURELIUS

THINKAURELIUS.COM

Abstract

Titan is an open source distributed graph database build on top of Cassandra that can power real-time applications with thousands of concurrent users over graphs with billions of edges. Graphs are a versatile data model for capturing and analyzing rich relational structures. Graphs are an increasingly popular way to represent data in a wide range of domains such as social networking, recommendation engines, advertisement optimization, knowledge representation, health care, education, and security.

This presentation discusses Titan's data model, query language, and novel techniques in edge compression, data layout, and vertex-centric indices which facilitate the representation and processing of big graph data across a Cassandra cluster. We demonstrate Titan's performance on a large scale benchmark evaluation using Twitter data.

Titan Graph Database

- supports real time local traversals (OLTP)
- is highly scalable
  - in the number of concurrent users
  - in the size of the graph
- is open-sourced under the Apache2 license
- builds on top of Apache Cassandra for distribution and replication

I

The Graph Data Model

Hercules: demigod
Alcmene: human
Jupiter: god
Saturn: titan
Pluto: god
Neptune: god
Cerberus: monster

Entities

Name	Type
Hercules	demigod
Alcmene	human
Jupiter	god
Saturn	titan
Pluto	god
Neptune	god
Cerberus	monster

TABLE

Name:
Hercules
Type:
demigod

Name:
Alcmene
Type:
human

Name:
Jupiter
Type:
god

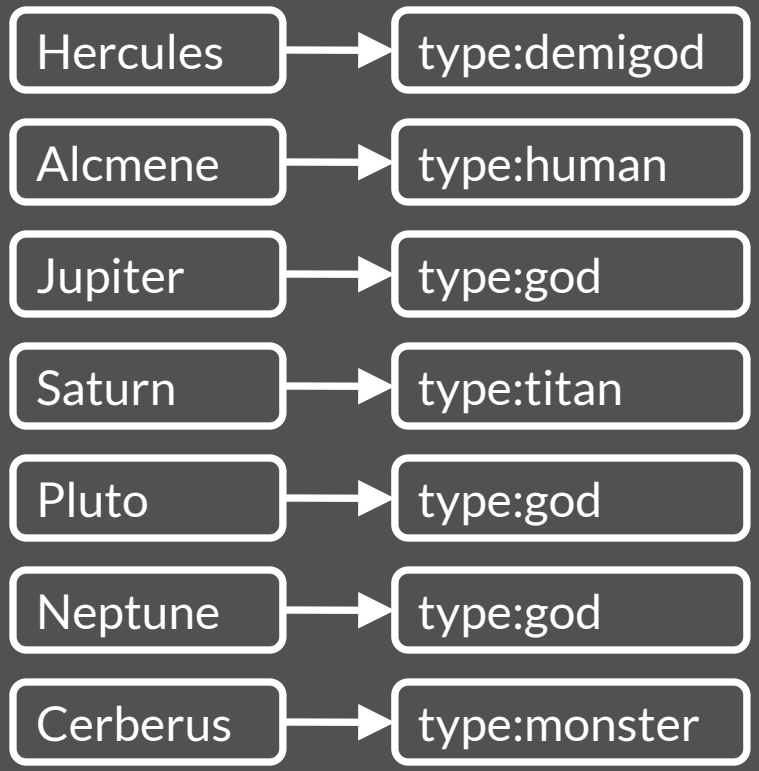
Name:
Saturn
Type:
titan

Name:
Pluto
Type:
god

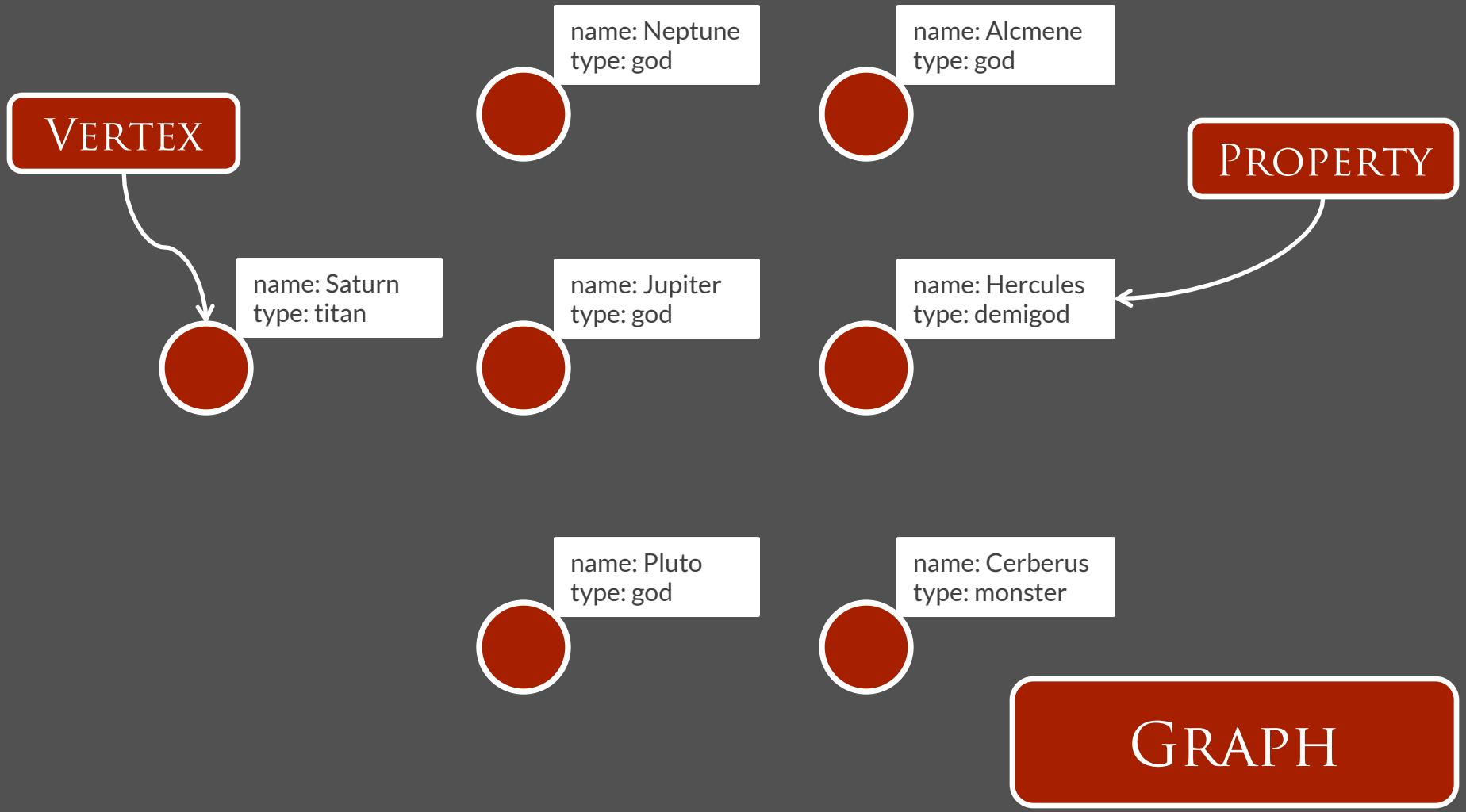
Name:
Neptune
Type:
god

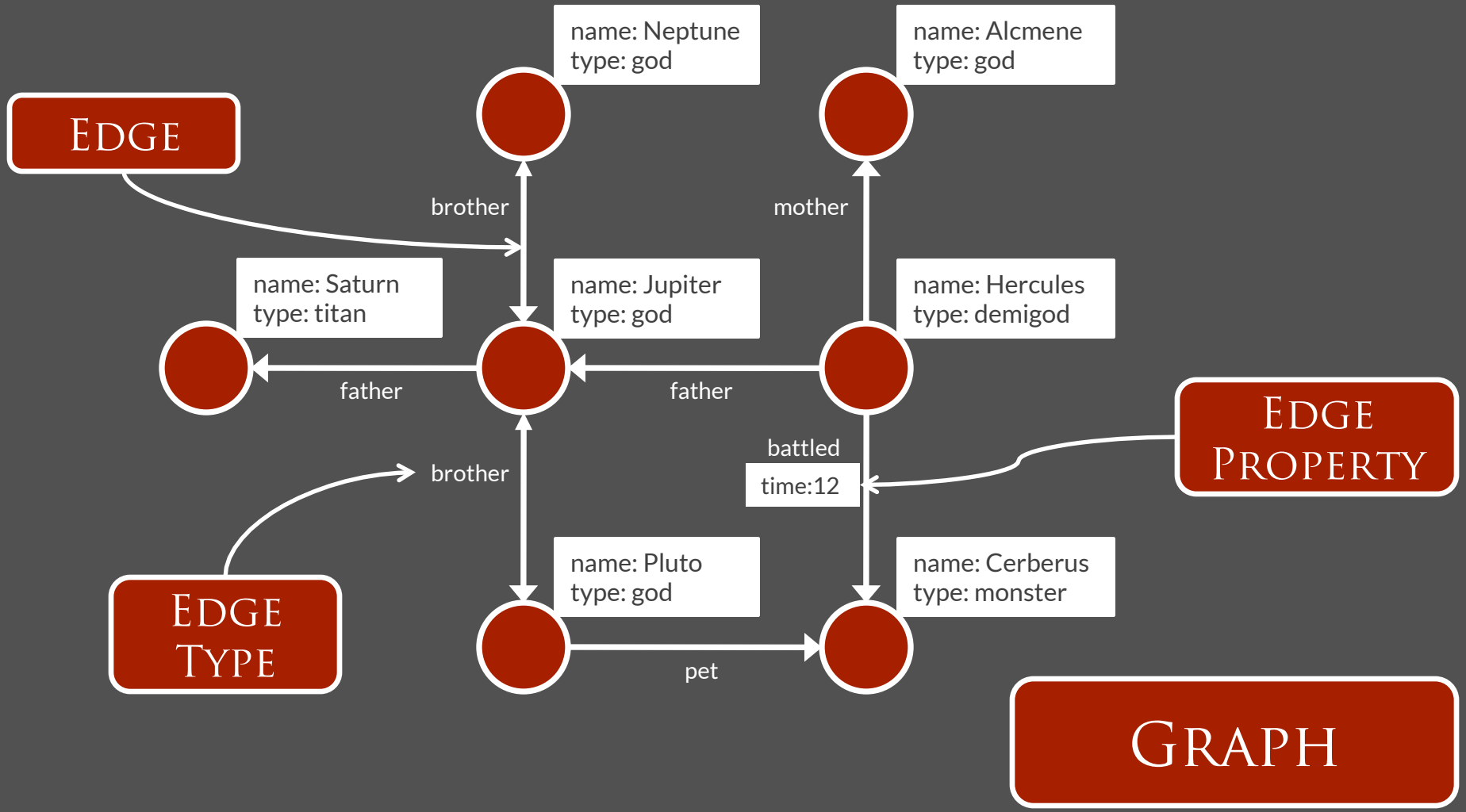
Name:
Cerberus
Type:
monster

Documents


Hercules
type:demigod
Alcmene
type:human
Jupiter
type:god
Saturn
type:titan
Pluto
type:god
Neptune
type:god
Cerberus
type:monster

Key->Value


Vertex
name: Neptune
type: god
name: Alcmene
type: god
Property
name: Saturn
type: titan
name: Jupiter
type: god
name: Hercules
type: demigod
name: Pluto
type: god
name: Cerberus
type: monster
Graph


Edge
name: Neptune
type: god
name: Alcmene
type: god
brother
mother
name: Saturn
type: titan
name: Jupiter
type: god
name: Hercules
type: demigod
father
father
battled
time:12
Edge Property
brother
Edge Type
name: Pluto
type: god
name: Cerberus
type: monster
pet
Graph

I

Graph = Agile Data Model

II

Graph Use Cases

AURELIUS
THINKAURELIUS.COM

Amazon.com: Networks: An Introduction (9780199206650): Mark Newman: Books

http://www.amazon.com/Networks-An-Introduction-Mark-Newman/dp/0199206651/ref=sr_1_1?ie=UTF8&qid=1344380760&sr=8-1&keywor

Amazon.com: Networks: An Intro...

Frequently Bought Together

Networks: An Introduction TOP

Price For All Three: $146.70

Add all three to Cart | Add all three to Wish List

Show availability and shipping details

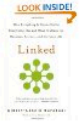
- **This item:** Networks: An Introduction by Mark Newman Hardcover $64.06
- Networks, Crowds, and Markets: Reasoning About a Highly Connected World by David E
- Social and Economic Networks by Matthew O. Jackson Paperback $45.00

Recommendations

Customers Who Bought This Item Also Bought

Page 1 of 15

Networks, Crowds, and Markets: Reasoning ...
David Easley
(6)
Hardcover
$37.64



Linked: How Everything Is Connected to ...
Albert-Laszlo Barabasi
(118)
Paperback
$10.05

Dynamical Processes on Complex Networks
Alain Barrat
(3)
Hardcover
$65.69

Understanding Social Networks: Theories, ...
Charles Kadushin
(1)
Paperback
$21.55

Social and Economic Networks
Matthew O. Jackson
(2)
Paperback
$45.00

Social Network Analysis: Methods and ...
Stanley Wasserman
(12)
Paperback
$42.57

Social Network Analysis: History, ...
Christina Prell
(4)
Paperback
$36.59

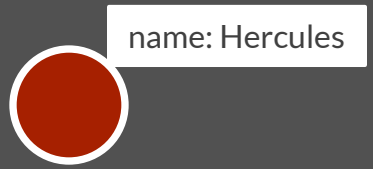
Editorial Reviews

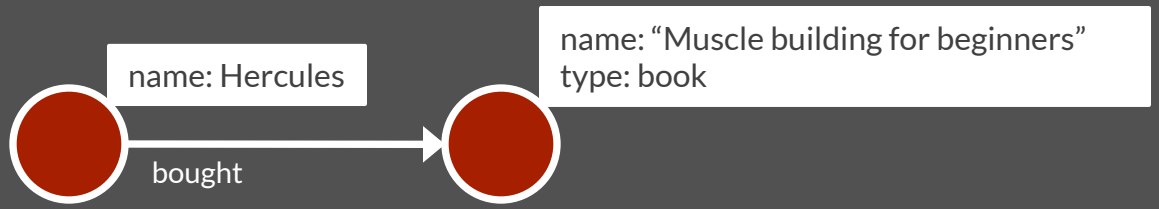
Review

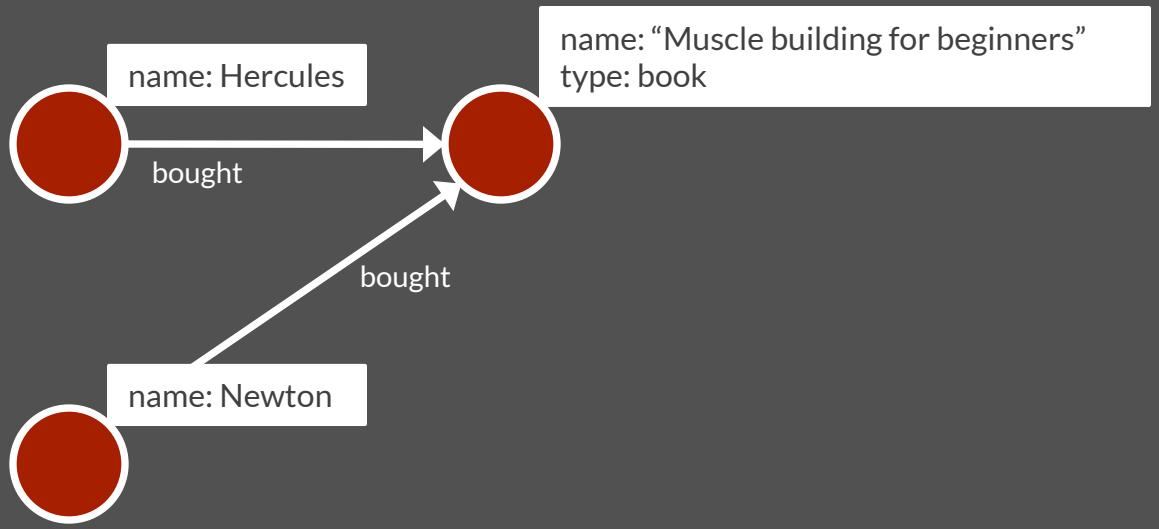
"[*Networks*] distinguishes itself from other network texts by its attention to the breadth of both the areas to which networks have been applied and the techniques for reasoning about them. It is likely to become the standard introductory textbook for the study of networks, and it is valuable as a desk-side reference for anyone who works with network problems." -- H. Van Dyke Parunak, *Computing Reviews*

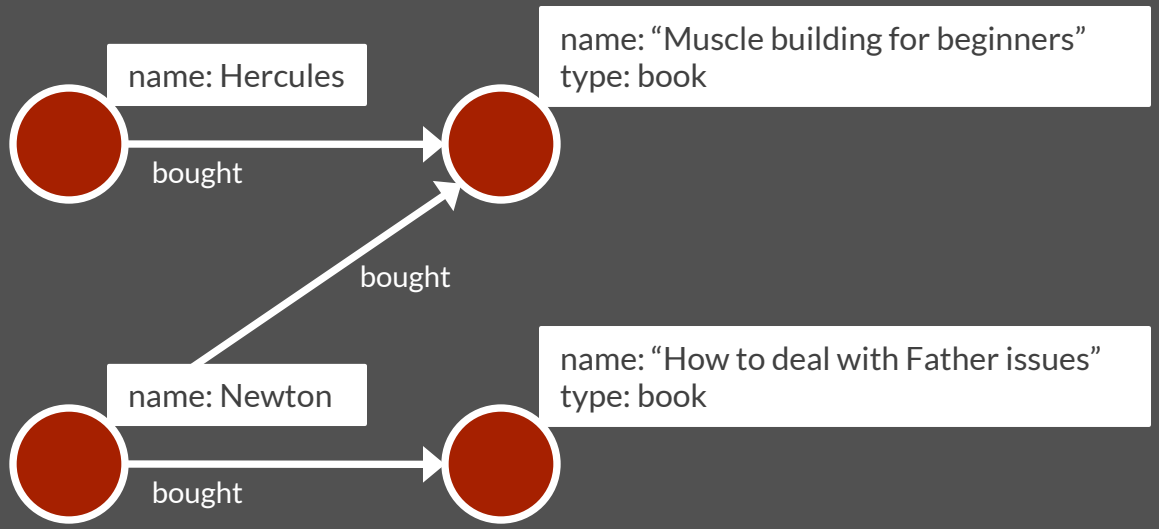
"An excellent textbook for the growing field of networks. It is cleverly written and suitable as both an introduction for undergraduate students and as a roadmap for graduate students. Furthermore, its more than 300 bibliographic references will guide readers who are interested in particular topics. Being highly self-contained, computer scientists and professionals from other fields can also use the book -- in fact, the author himself is a physicist. In short, this book is a delight for the inquisitive mind." -- Fernando Berzal, *Computing Reviews*

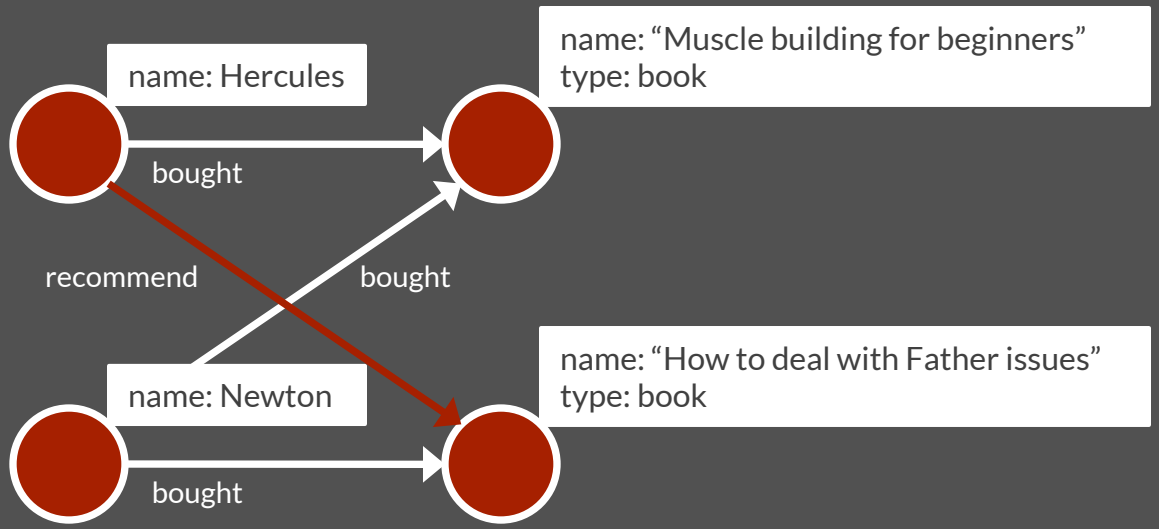
Recommendation?



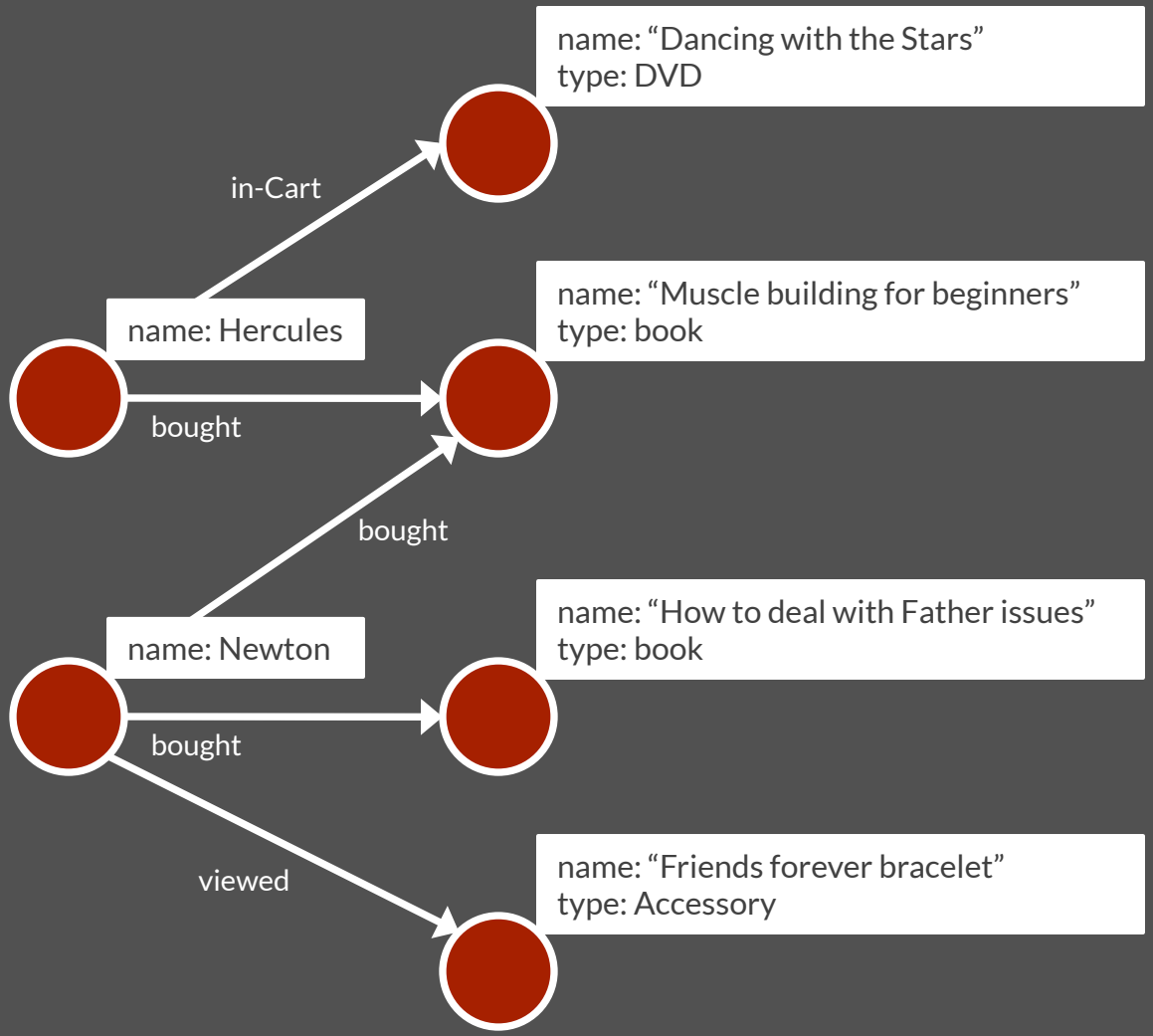

name: Hercules
bought
name: “Muscle building for beginners”
type: book

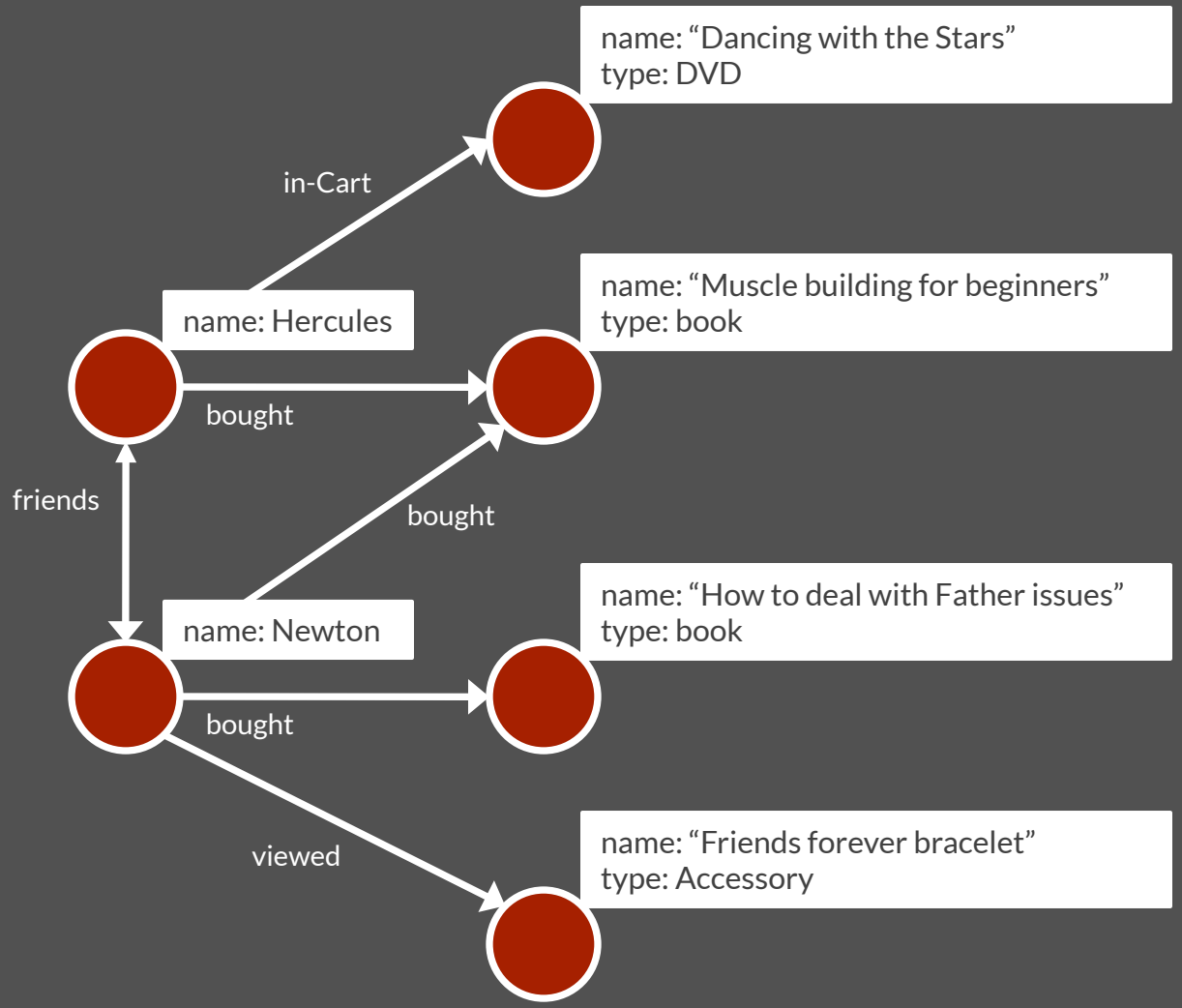

name: Hercules
name: “Muscle building for beginners”
type: book
bought
bought
name: Newton

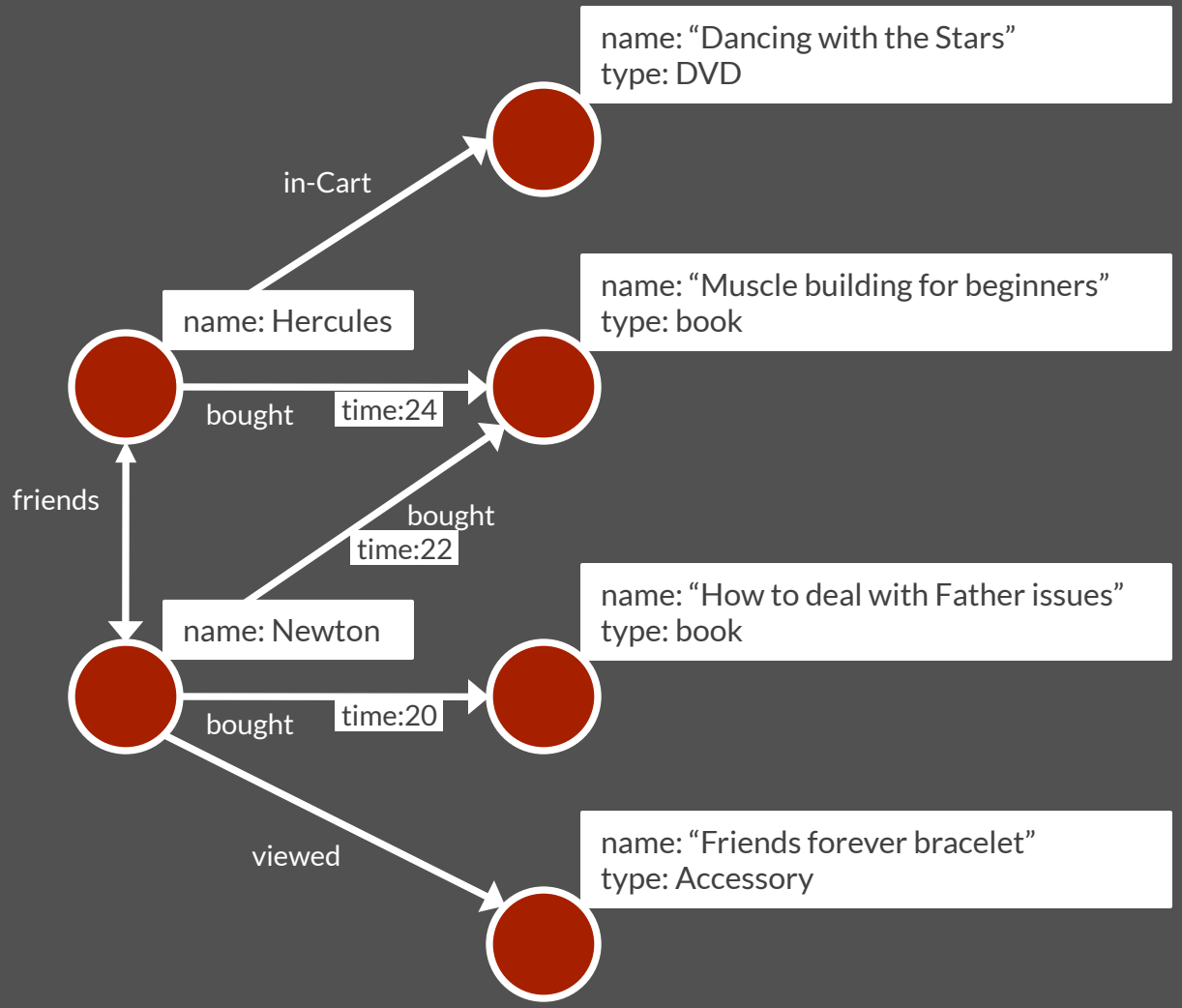

name: Hercules
name: “Muscle building for beginners”
type: book
bought
bought
name: Newton
name: “How to deal with Father issues”
type: book
bought


name: Hercules
name: "Muscle building for beginners"
type: book
bought
recommend
bought
name: Newton
name: "How to deal with Father issues"
type: book
bought

Traversal


name: “Dancing with the Stars”
type: DVD
in-Cart
name: Hercules
name: “Muscle building for beginners”
type: book
bought
bought
name: Newton
name: “How to deal with Father issues”
type: book
bought
viewed
name: “Friends forever bracelet”
type: Accessory


name: “Dancing with the Stars”
type: DVD
in-Cart
name: Hercules
name: “Muscle building for beginners”
type: book
bought
friends
bought
name: Newton
name: “How to deal with Father issues”
type: book
bought
viewed
name: “Friends forever bracelet”
type: Accessory


name: “Dancing with the Stars”
type: DVD
in-Cart
name: Hercules
name: “Muscle building for beginners”
type: book
bought
time:24
friends
bought
time:22
name: Newton
name: “How to deal with Father issues”
type: book
bought
time:20
viewed
name: “Friends forever bracelet”
type: Accessory

Recommendations

Path Finding

LinkedIn Account Type: Basic

Add Connections

Home | Profile | Contacts | Groups | Jobs | Inbox 3 | Companies

Add Connections | Colleagues | Classmates | People You May Kno

Your invitation was sent successfully.

Filter By...

Current Company
- All Companies
- LBi (13)
- Bigmouthmedia (9)
- Latitude (6)
- Just Search (5)
- Econsultancy (3)

Past Company
- All Companies
- Latitude (20)
- LBi (5)
- Bigmouthmedia (5)
- Microsoft (3)
- Diffiniti (3)

School
- All Schools
- Bournemouth University (4)
- Napier University (4)
- Staffordshire University (4)
- The University of Edinburgh (4)
- University of Leeds (4)

Import contacts »

It's easy to search your email contacts and quickly grow your network

Wizard of In — Connect
Wizard at LinkedIn

Sherlock Holmes — Connect
Detective at 221B Baker Street

John Watson — Connect
Dr. Watson at 221B Baker Street

Ernest Hemingway (3rd)
Author

Robin Hood
Activist/ Chief Fundraiser at Nottingham

the Hatter
Riddler at 6 o'clock productions

J. R. R. Tolkien (3rd) — Connect
Father of modern fantasy literature

Groucho Marx — Connect
Comedian

Albert Einstein — Connect
E=mc^2

Werner Heisenberg — Connect
This *may* be Heisenberg's profile

Ads by LinkedIn Members

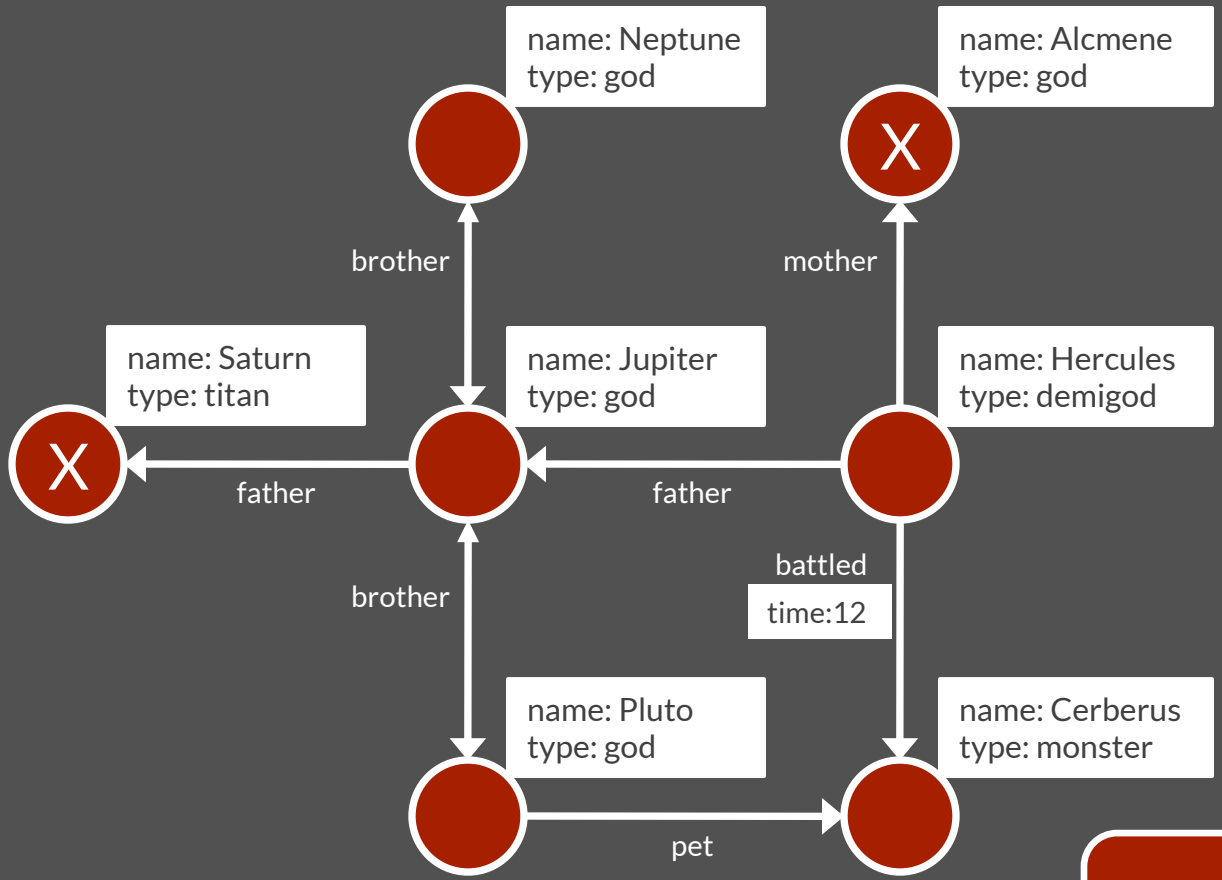
B2B Email Marketing Demo

View a demo of our specialist CRM Integrated B2B Email

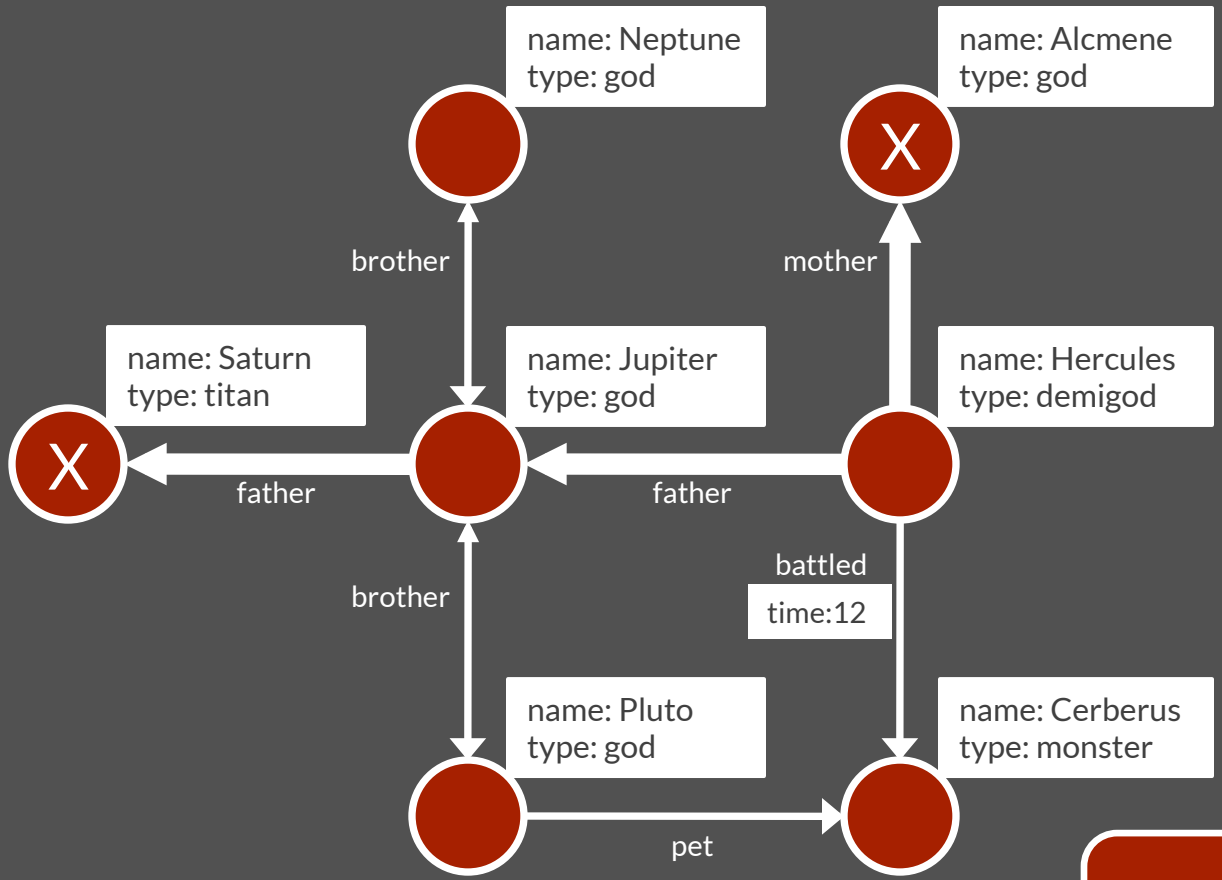
search white paper. Four key campaign success drivers.

From: Ashley Gregg

What's this?



Path Finding



Path Finding

titan graph database – Google Search

http://www.google.com/#hl=en&sugexp=eese&gs_nf=1&cp=8&gs_id=11&xhr=t&q=Titan+Graph+database&pf=p&output=search&sclient=psy

titan graph database – Google Search

+You Search Images Maps Play YouTube News Gmail Documents Calendar More

Titan Graph database

titan graph database

Sign in

Search

About 663,000 results (0.17 seconds)

Web
Images
Maps
Videos
News
Shopping
More

Oakland, CA
Change location

Show search tools

Ad related to **titan graph database** Why this ad?

NOSQL **Graph Database** 1 (866) 648 8043
www.objectivity.com/**databases**
Get the **database** that powers the world's most complex applications.

Titan: Distributed **Graph Database** - GitHub
thinkaurelius.github.com/**titan**/
Titan : A Highly Scalable, Distributed **Graph Database**.

Titan: A Highly Scalable, Distributed **Graph Database** | Hacker News
news.ycombinator.com/item?id=4112396
Jun 14, 2012 – Many **graph databases** have claimed high scalability and distributability but none of those claims have held up over time due to the ...

Titan: The Rise of Big **Graph** Data - Eventbrite
titanbiggraphdata.eventbrite.com/
He is the primary developer of the distributed **graph database Titan**. Matthias focuses most of his time and effort on novel OLTP and OLAP graph processing ...

Titan - A Flexible Distributed **Graph Database** - AMT Blog
blog.amt.in/**titan**-a-flexible-distributed-**graph-database**
Titan is an open source, distributed **graph database** optimized for processing large-scale graphs represented over a cluster of machines. The pluggable storage ...

Titan: new **graph database** - Large Scale Machine Learning and ...
bickson.blogspot.com/2012/06/**titan**-new-**graph-database**.html
Jun 17, 2012 – It looks like a very interesting new **graph database**. Anyone who is interested to learn more about **Titan** is welcome to our GraphLab workshop ...

yahoo.com

```
<html> …
</html>
```

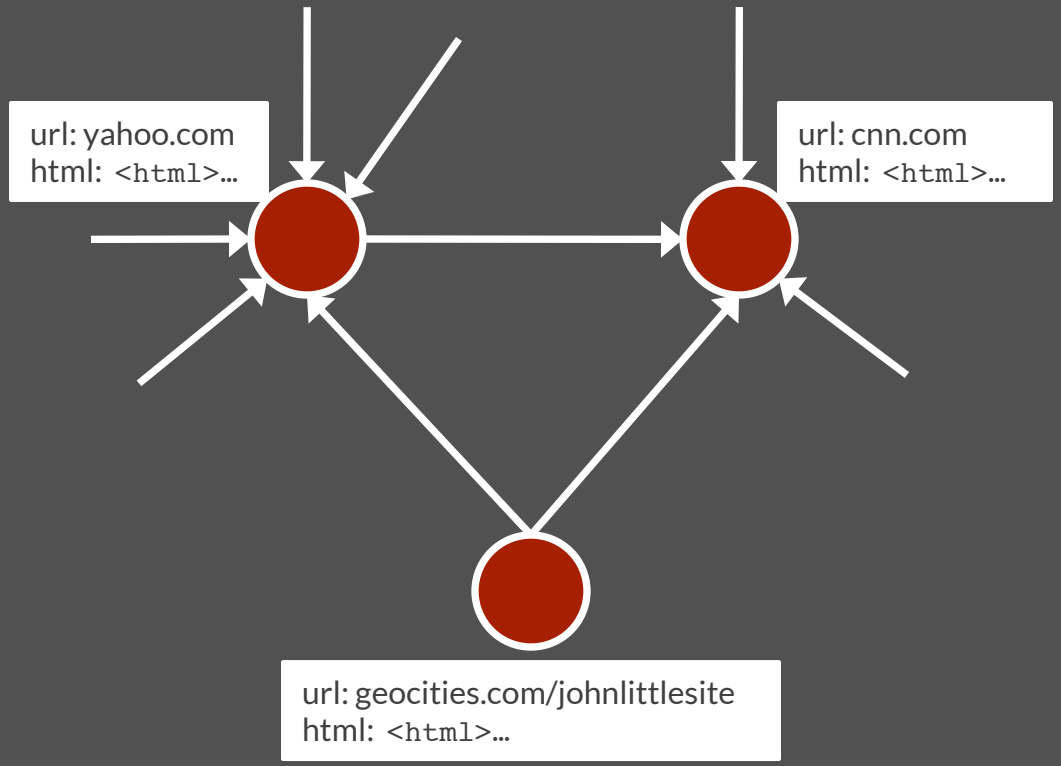
geocities.com
/johnlittlesite

```
<html> …
</html>
```

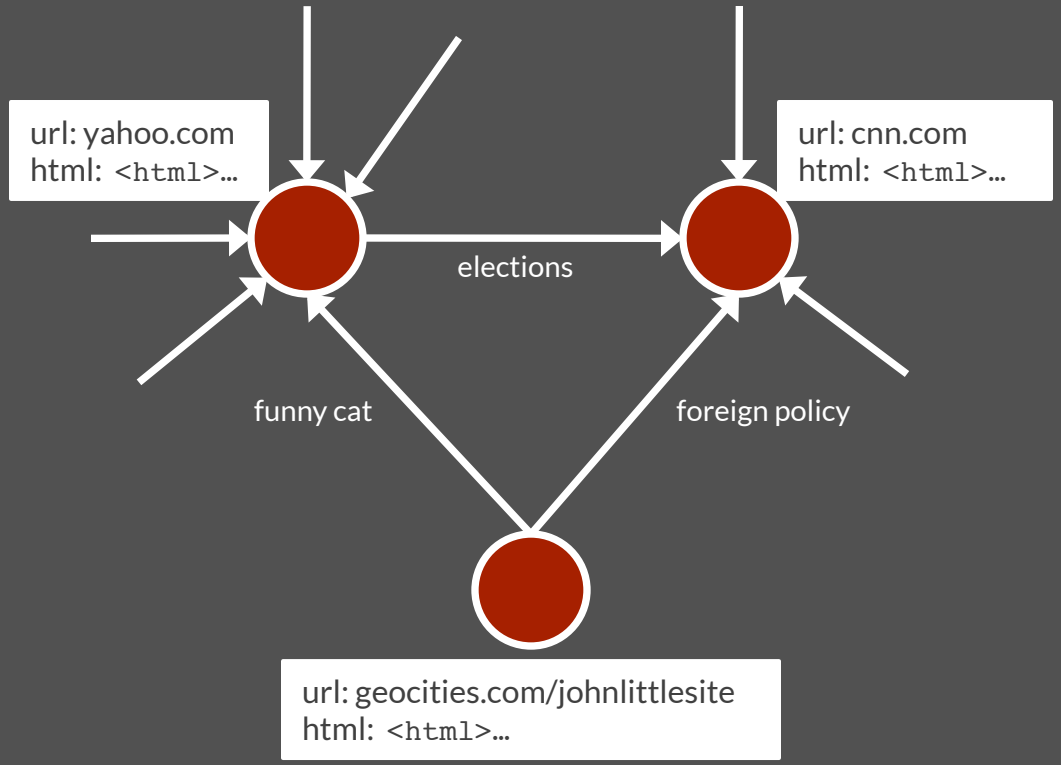
cnn.com

```
<html> …
</html>
```

CREDIBILITY?


url: yahoo.com
html: <html>...
url: cnn.com
html: <html>...
url: geocities.com/johnlittlesite
html: <html>...

Link Graph


url: yahoo.com
html: <html>...
url: cnn.com
html: <html>...
elections
funny cat
foreign policy
url: geocities.com/johnlittlesite
html: <html>...

Link Graph

II

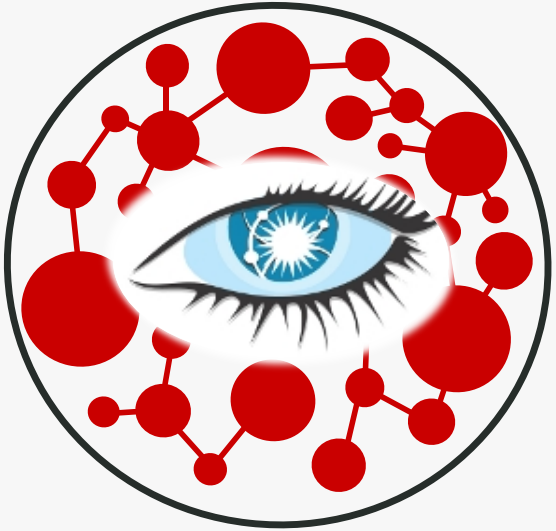
Graph = Value from Relationships

III

The Titan Graph Database

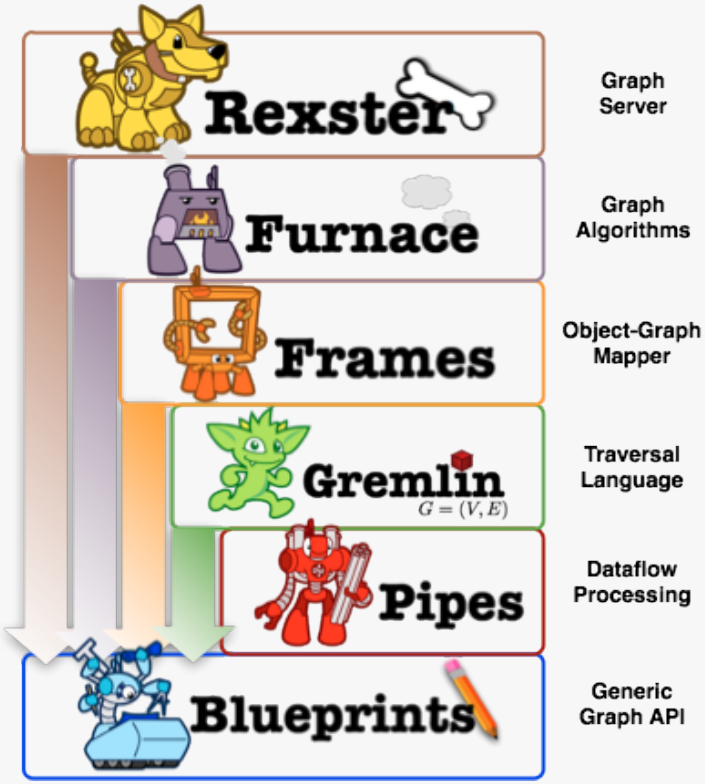
Titan Features

- numerous concurrent users
- real-time traversals (OLTP)
- high availability
- dynamic scalability
- build on Apache Cassandra



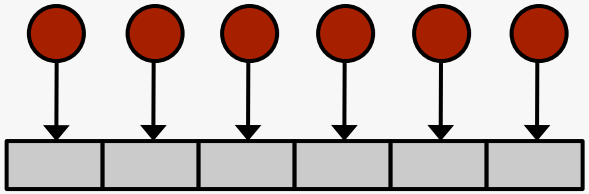
Titan Ecosystem

- Native Blueprints Implementation
- Gremlin Query Language
- Rexster Server
  - any Titan graph can be exposed as a REST endpoint

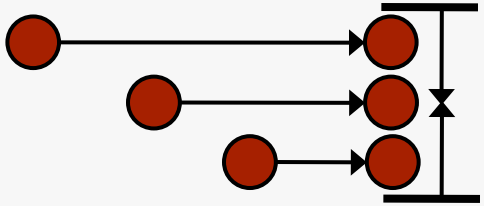


Titan Internals

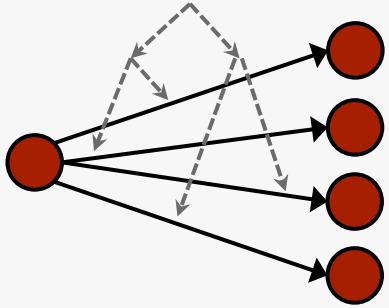
I. Data Management



II. Edge Compression

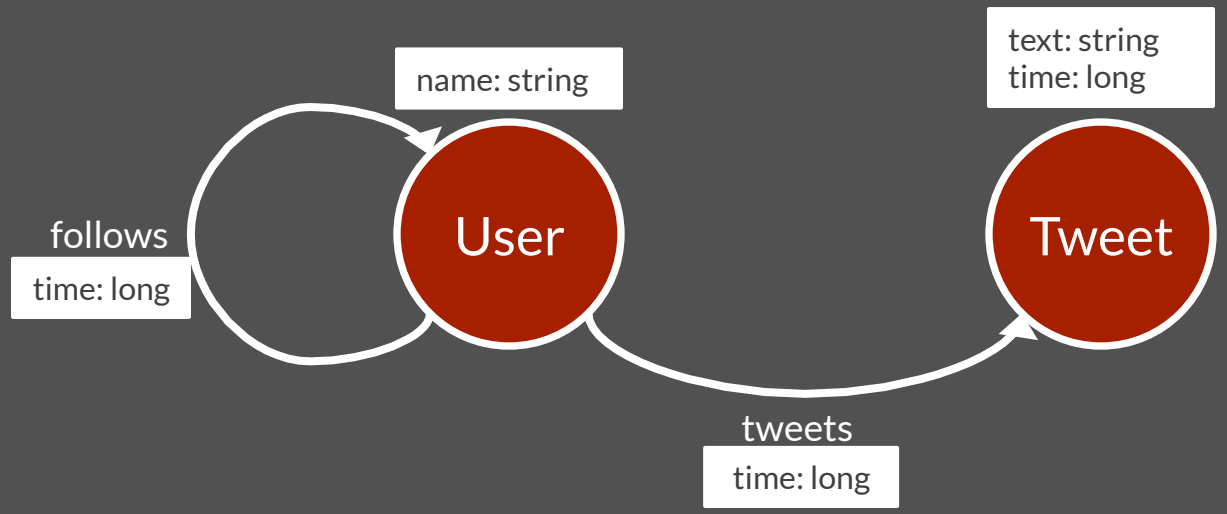


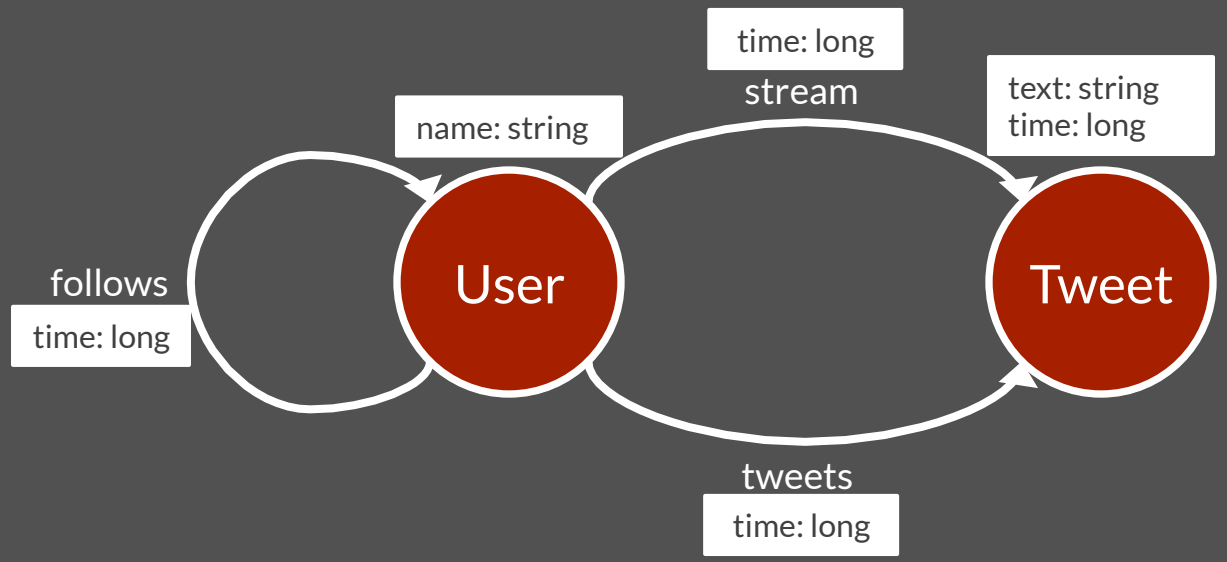
III. Vertex-Centric Indices



IV

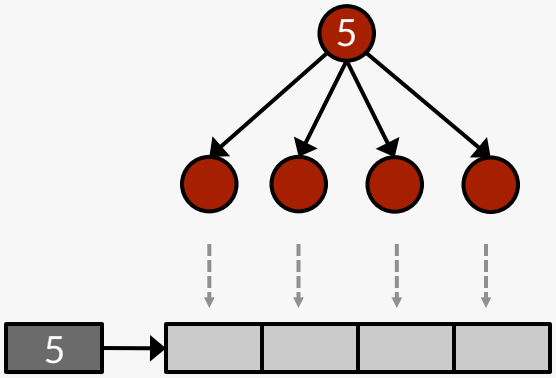
Rebuilding Twitter with Titan


name: string
text: string
time: long
follows
time: long
User
Tweet
tweets
time: long


time: long
stream
name: string
text: string
time: long
follows
time: long
User
Tweet
tweets
time: long

Titan Storage Model

- Adjacency list in one column family
- Row key = vertex id
- Each property and edge in one column
  - Denormalized, i.e. stored twice
- Direction and label/key as column prefix
  - Use slice predicate for quick retrieval



Connecting Titan

```
titan$ bin/gremlin.sh
         \,,,/
         (o o)
-----oOOo-(_)-oOOo-----
gremlin> conf = new BaseConfiguration();
==>org.apache.commons.configuration.BaseConfiguration@763861e6
gremlin> conf.setProperty("storage.backend","cassandra");
gremlin> conf.setProperty("storage.hostname","77.77.77.77");
gremlin> g = TitanFactory.open(conf);
==>titangraph[cassandra:77.77.77.77]
gremlin>
```

Defining Property Keys

```
gremlin> g.makeType().name("time").
         dataType(Long.class).
         functional().
         makePropertyKey();
gremlin> g.makeType().name("text").dataType(String.class).
         functional().makePropertyKey();
gremlin> g.makeType().name("name").dataType(String.class).
         indexed().
         unique().
         functional().makePropertyKey();
```

Defining Property Keys

```
gremlin> g.makeType().name("time").
         dataType(Long.class).
         functional().
         makePropertyKey();
gremlin> g.makeType().name("text").dataType(String.class).
         functional().makePropertyKey();
gremlin> g.makeType().name("name").dataType(String.class).
         indexed().
         unique().
         functional().makePropertyKey();
```

Each type has a unique name

The allowed data type

If a key is functional, each vertex can have at most one property for this key

Defining Property Keys

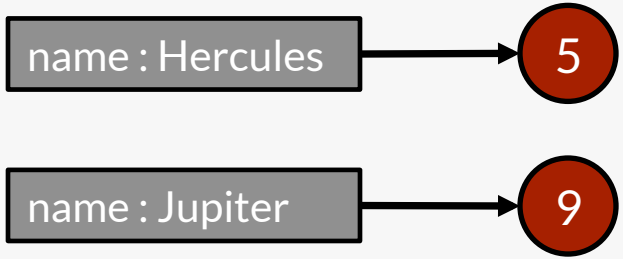
```
gremlin> g.makeType().name("time").
         dataType(Long.class).
         functional().
         makePropertyKey();
gremlin> g.makeType().name("text").dataType(String.class).
         functional().makePropertyKey();
gremlin> g.makeType().name("name").dataType(String.class).
         indexed().
         unique().
         functional().makePropertyKey();
```

Creates and maintains an index over property values

Ensures that each property value is uniquely associated with only one vertex by acquiring a lock.

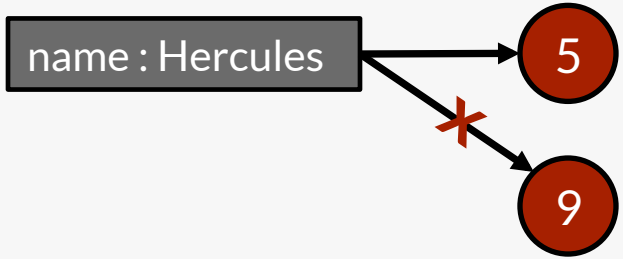
Titan Indexing

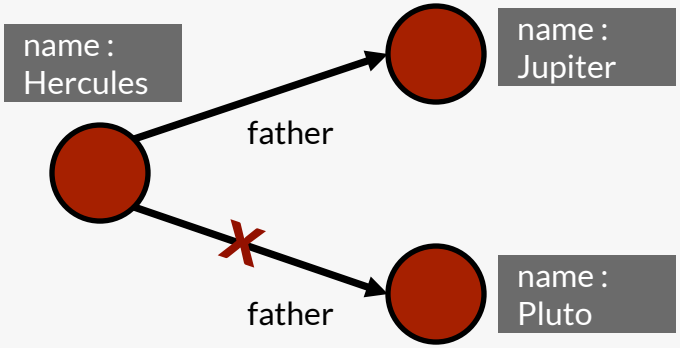
- Vertices can be retrieved by property key + value
- Titan maintains index in a separate column family as graph is updated
- Only need to define a property key as .index()



Titan Locking

- Locking ensures consistency when it is needed
- Titan uses time stamped quorum reads and writes on separate CFs for locking
- Uses
  - Property uniqueness: .unique()
  - Functional edges: .functional()
  - Global ID management





Defining Edge Labels

```
gremlin> g.makeType().name("follows").
         primaryKey(time).
         makeEdgeLabel();
gremlin> g.makeType().name("tweets").
         primaryKey(time).makeEdgeLabel();
gremlin> g.makeType().name("stream).
         primaryKey(time).
         unidirected().
         makeEdgeLabel();
```

Defining Edge Labels

```
gremlin> g.makeType().name("follows").
         primaryKey(time).
         makeEdgeLabel();
gremlin> g.makeType().name("tweets").
         primaryKey(time).makeEdgeLabel();
gremlin> g.makeType().name("stream).
         primaryKey(time).
         unidirected().
         makeEdgeLabel();
```

Sort/index key for edges of this label

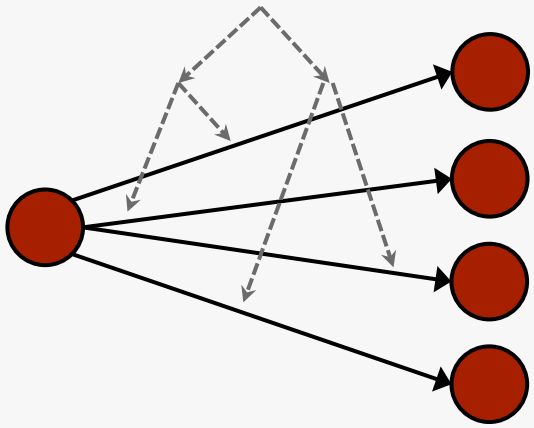
Defining Edge Labels

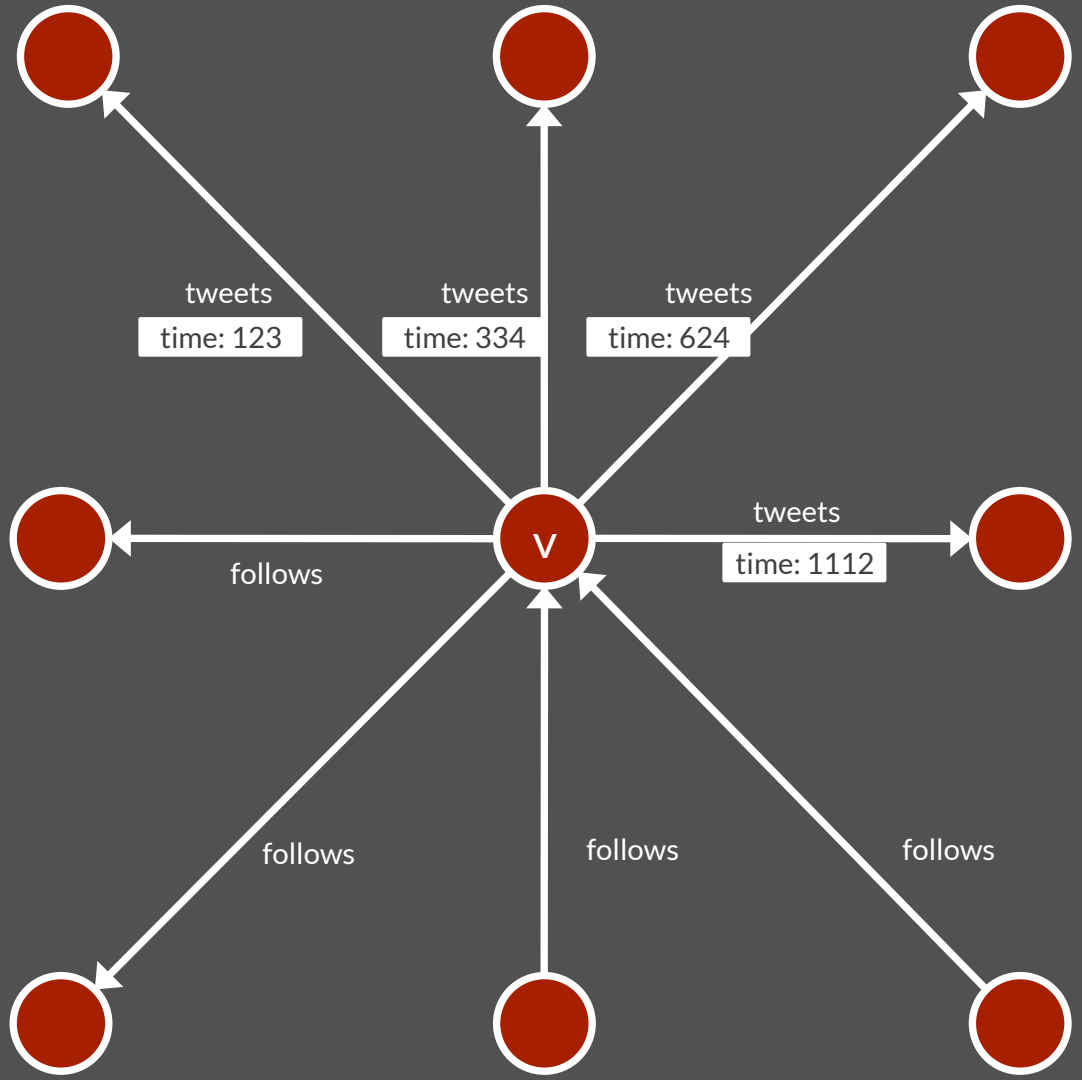
```
gremlin> g.makeType().name("follows").
         primaryKey(time).
         makeEdgeLabel();
gremlin> g.makeType().name("tweets").
         primaryKey(time).makeEdgeLabel();
gremlin> g.makeType().name("stream).
         primaryKey(time).
         unidirected().
         makeEdgeLabel();
```

Store edges of this label only in outgoing direction

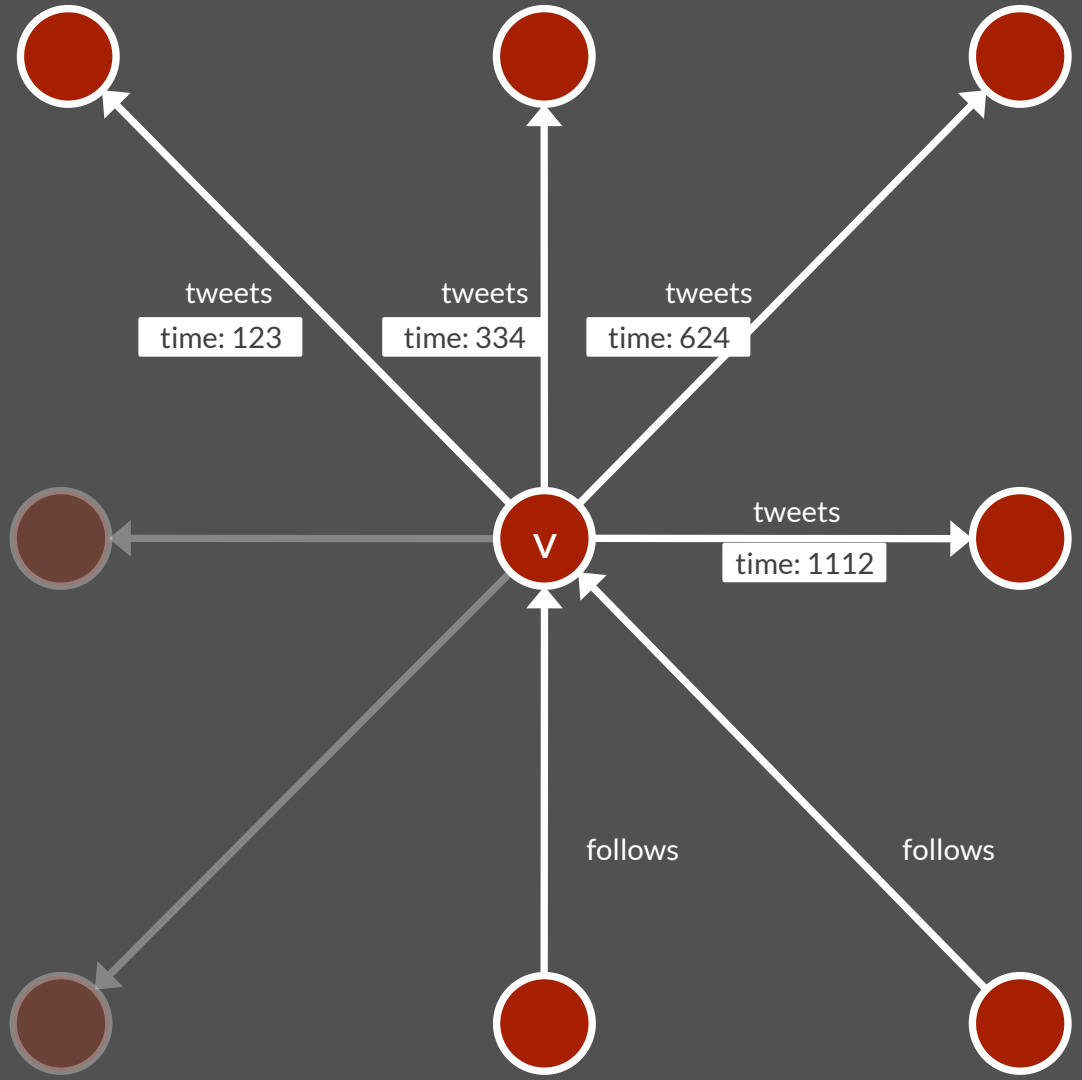
Vertex-Centric Indices

- Sort and index edges per vertex by primary key
  - Primary key can be composite
- Enables efficient focused traversals
  - Only retrieve edges that matter
- Uses slice predicate for quick, index-driven retrieval

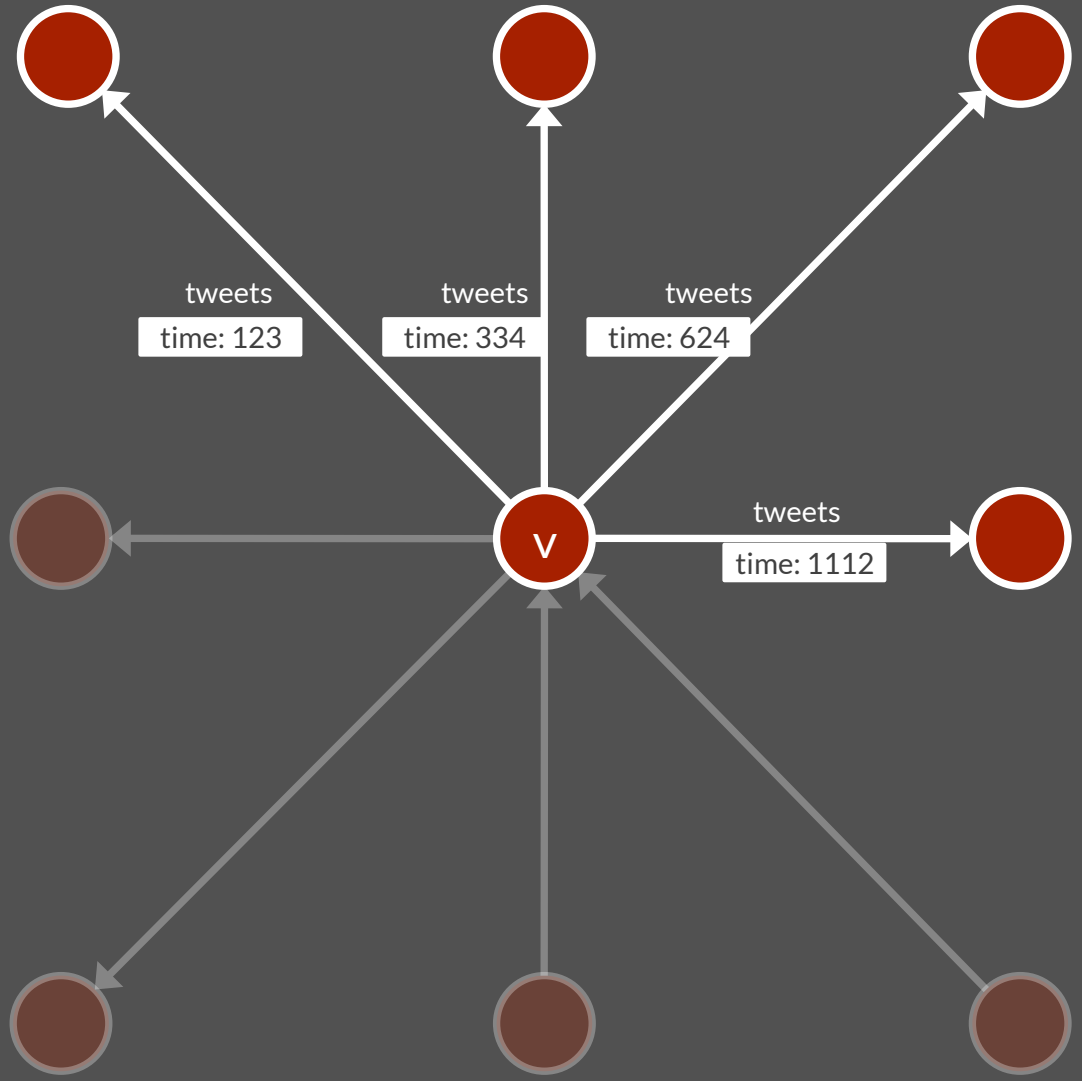



tweets
time: 123
tweets
time: 334
tweets
time: 624
v
tweets
time: 1112
follows
follows
follows
follows

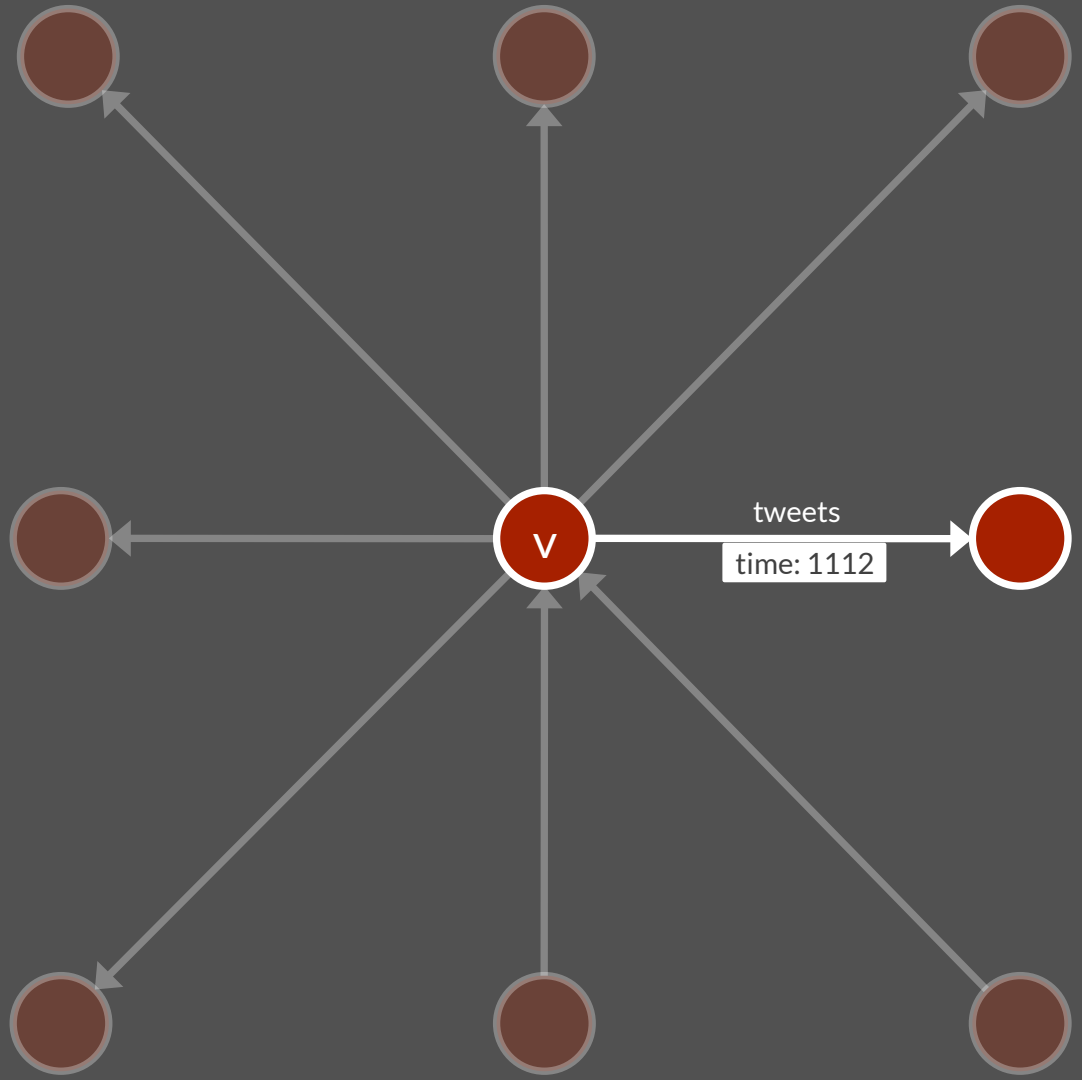
```
v.query()
```




```
v.query()
.direction(OUT)
```




```
v.query()
.direction(OUT)
.labels("tweets")
```

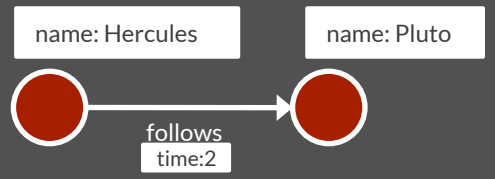



```
v.query()
.direction(OUT)
.labels("tweets")
.has("time",T.gt,1000)
```

Create Accounts

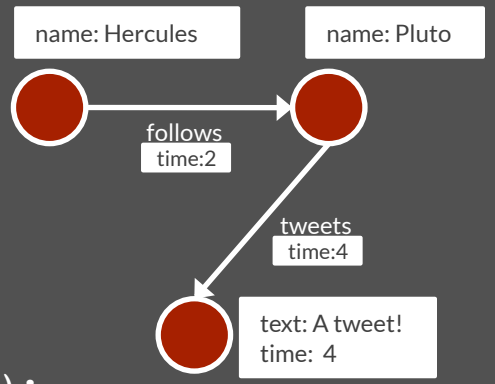

```
gremlin> hercules = g.addVertex(['name':'Hercules']);

gremlin> pluto = g.addVertex(['name':'Pluto']);
```

Add Followship



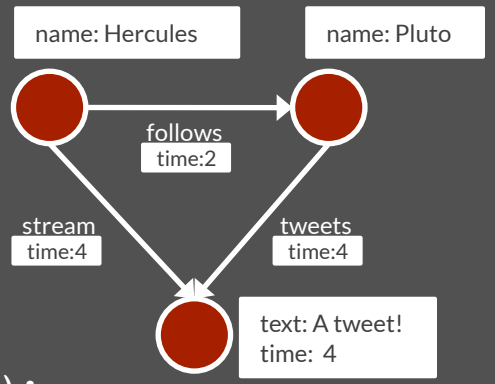

```
gremlin> hercules = g.addVertex(['name':'Hercules']);

gremlin> pluto = g.addVertex(['name':'Pluto']);

gremlin> g.addEdge(hercules,pluto,"follows",['time':2]);
```

Publish Tweet



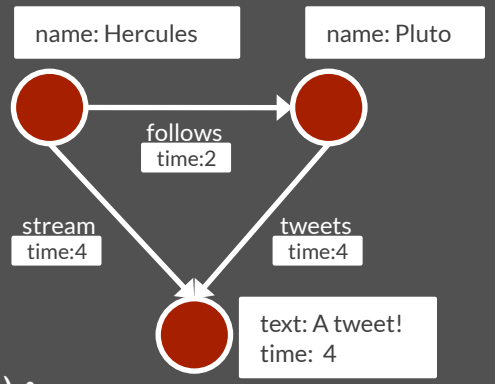

```
gremlin> hercules = g.addVertex(['name':'Hercules']);

gremlin> pluto = g.addVertex(['name':'Pluto']);

gremlin> g.addEdge(hercules,pluto,"follows",['time':2]);

gremlin> tweet = g.addVertex(['text':'A tweet!','time':4])

gremlin> g.addEdge(pluto,tweet,"tweets",['time':4])
```

Update Streams




```
gremlin> hercules = g.addVertex(['name':'Hercules']);

gremlin> pluto = g.addVertex(['name':'Pluto']);

gremlin> g.addEdge(hercules,pluto,"follows",['time':2]);

gremlin> tweet = g.addVertex(['text':'A tweet!','time':4])

gremlin> g.addEdge(pluto,tweet,"tweets",['time':4])

gremlin> pluto.in("follows").each{g.addEdge(it,tweet,"stream",['time':4])}
```

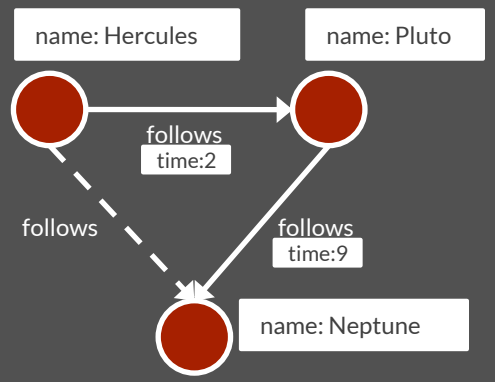
Read Stream




```
gremlin> hercules = g.addVertex(['name':'Hercules']);

gremlin> pluto = g.addVertex(['name':'Pluto']);

gremlin> g.addEdge(hercules,pluto,"follows",['time':2]);

gremlin> tweet = g.addVertex(['text':'A tweet!','time':4])

gremlin> g.addEdge(pluto,tweet,"tweets",['time':4])

gremlin> pluto.in("follows").each{g.addEdge(it,tweet,"stream",['time':4])}

gremlin> hercules.outE('stream')[0..9].inV.map
```

Sorted by time because its ‘stream’s primary key

Followship Recommendation




```
follows = g.V('name','Hercules').out('follows').toList()
follows20 = follows[(0..19).collect{random.nextInt(follows.size)}]
m = [:]
follows20.each
    { it.outE('follows'[0..29].inV.except(follows).groupCount(m).iterate() }
m.sort{a,b -> b.value <=> a.value}[0..4]
```

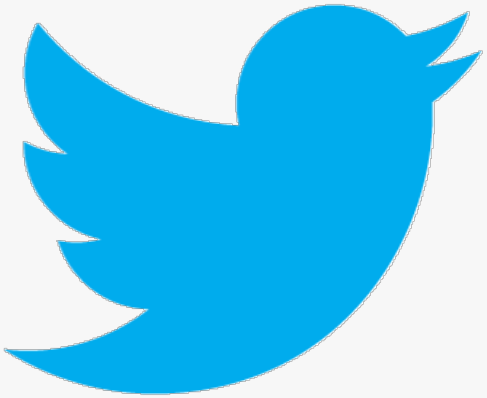
IV

Titan Performance Evaluation on Twitter-like Benchmark

AURELIUS
THINKAURELIUS.COM

Twitter Benchmark

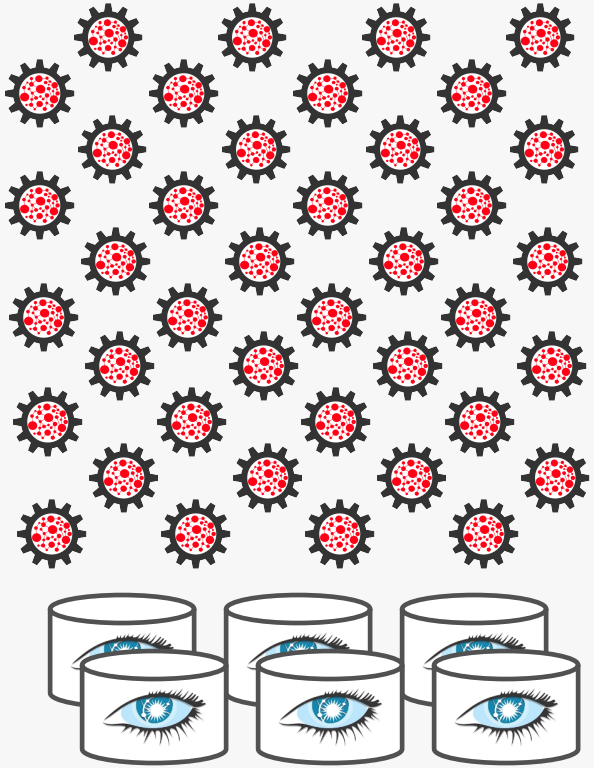
- 1.47 billion followship edges and 41.7 million users
  - Loaded into Titan using BatchGraph
  - Twitter in 2009, crawled by Kwak et. al
- 4 Transaction Types
  - Create Account (1%)
  - Publish tweet (15%)
  - Read stream (76%)
  - Recommendation (8%)
    - Follow recommended user (30%)



Kwak, H., Lee, C., Park, H., Moon, S., "What is Twitter, a Social Network or a News Media?," World Wide Web Conference, 2010.

Benchmark Setup

- 6 cc1.4xl Cassandra nodes
  - in one placement group
  - Cassandra 1.10
- 40 m1.small worker machines
  - repeatedly running transactions
  - simulating servers handling user requests
- EC2 cost: $11/hour



Benchmark Results

Transaction Type	Number of tx	Mean tx time	Std of tx time
Create account	379,019	115.15 ms	5.88 ms
Publish tweet	7,580,995	18.45 ms	6.34 ms
Read stream	37,936,184	6.29 ms	1.62 ms
Recommendation	3,793,863	67.65 ms	13.89 ms
Total	**49,690,061**		
Runtime	**2.3 hours**		

5,900 TX/SEC

High Load Results

Transaction Type	Number of tx	Mean tx time	Std of tx time
Create account	374,860	172.74 ms	10.52 ms
Publish tweet	7,517,667	70.07 ms	19.43 ms
Read stream	37,618,648	24.40 ms	3.18 ms
Recommendation	3,758,266	229.83 ms	29.08 ms
Total	**49,269,441**		
Runtime	**1.3 hours**		

10,200 TX/SEC

Benchmark Conclusion

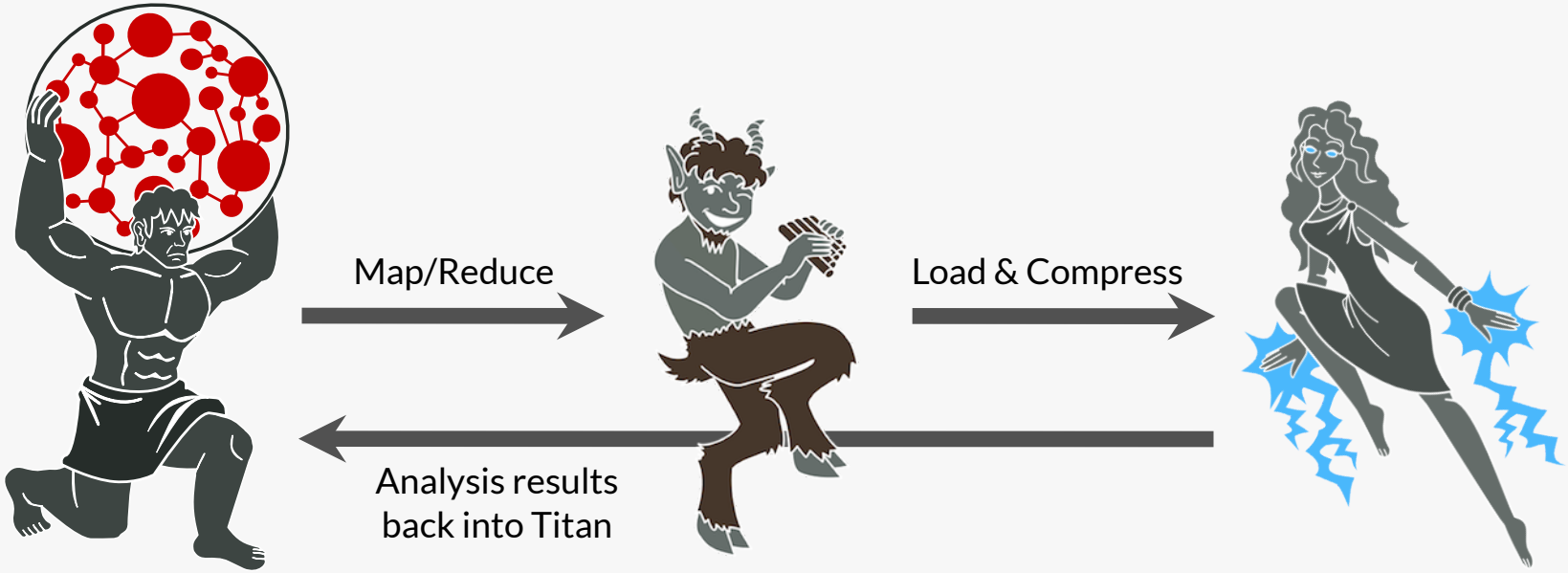
Titan can handle 10s of thousands of users with short response times even for complex traversals on a simulated social networking application based on real-world network data with billions of edges and millions of users in a standard EC2 deployment.

For more information on the benchmark:
http://thinkaurelius.com/2012/08/06/titan-provides-real-time-big-graph-data/

Future Titan

- Titan+Cassandra embedding
  - sending Gremlin queries into the cluster
- Graph partitioning together with ByteOrderedPartitioner
  - data locality = better performance
- Let us know what you need!

Titan goes OLAP



Stores a massive-scale property graph allowing real-time traversals and updates

Batch processing of large graphs with Hadoop

Runs global graph algorithms on large, compressed, in-memory graphs

III

Graph = Scalable + Practical

TITAN

THINKAURELIUS.GITHUB.COM/TITAN

AURELIUS

THINKAURELIUS.COM