Big Data Infrastructure

CS 489/698 Big Data Infrastructure (Winter 2016)

Week 11: Analyzing Graphs, Redux (2/2)

March 24, 2016

Jimmy Lin
David R. Cheriton School of Computer Science
University of Waterloo

These slides are available at http://lintool.github.io/bigdata-2016w/

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States
See http://creativecommons.org/licenses/by-nc-sa/3.0/us/ for details

Theme for Today:

How things work in the real world

(forget everything I told you…)

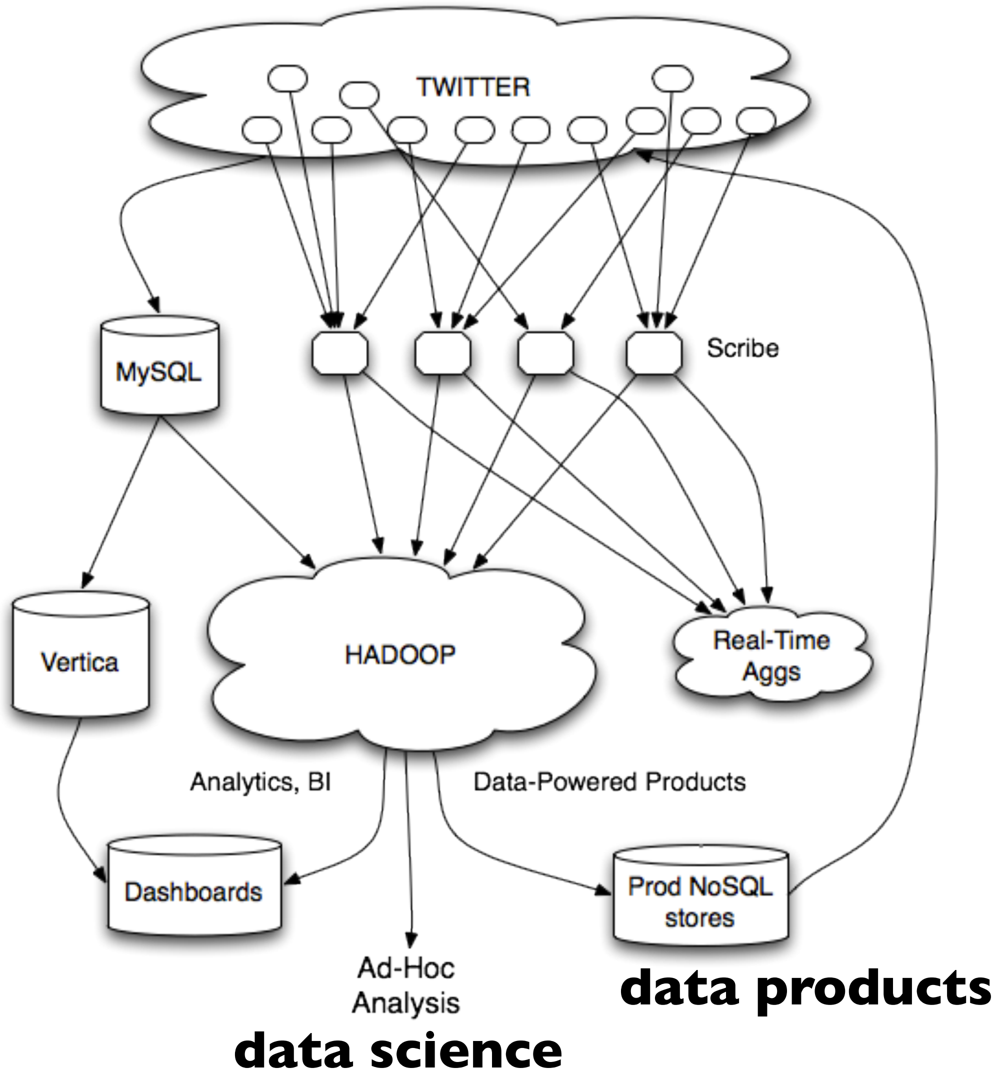
From the Ivory Tower...

Source: Wikipedia (All Souls College, Oxford)

... to building sh*t that works

Source: Wikipedia (Factory)

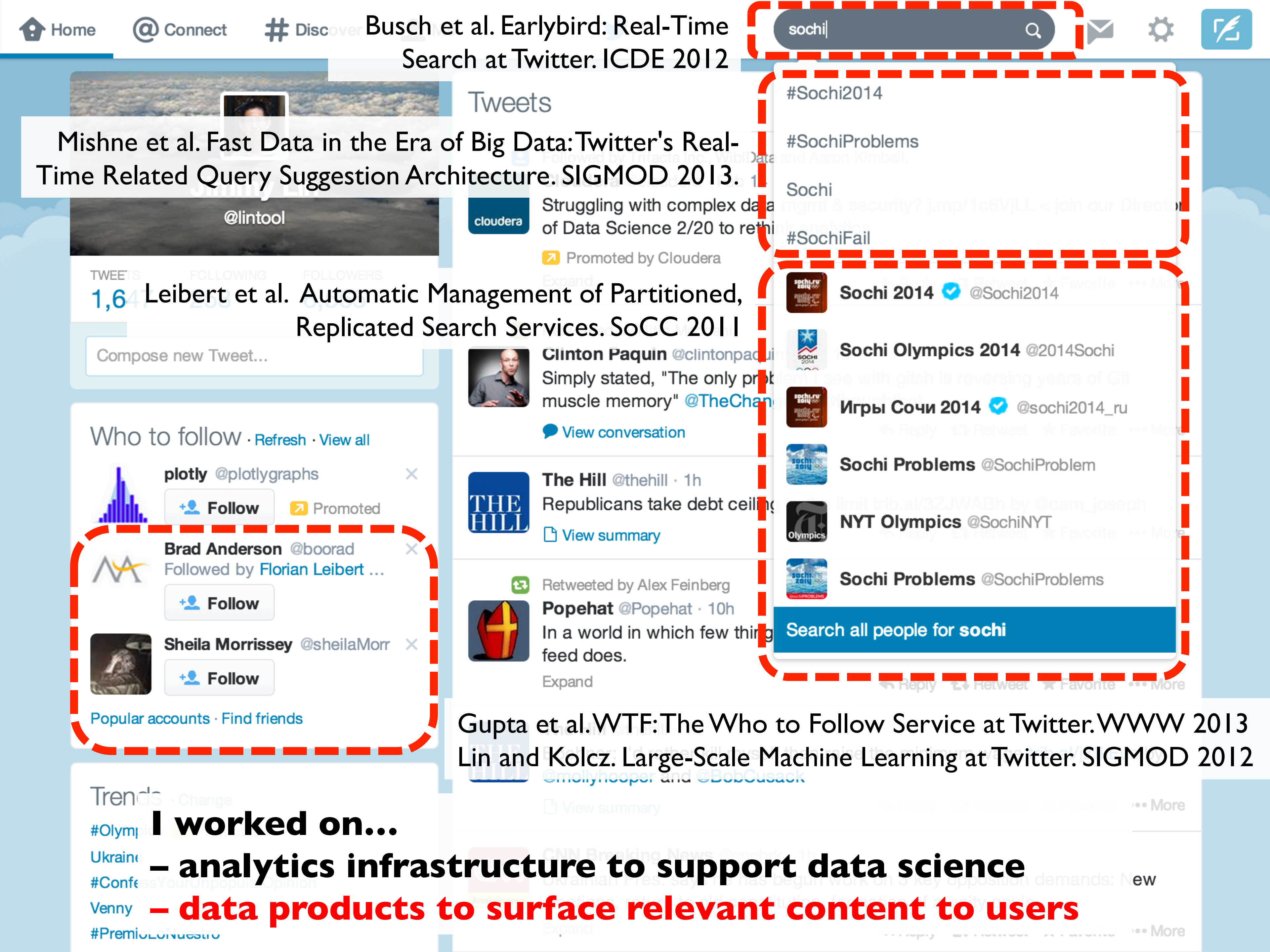
Source: Wikipedia (All Souls College, Oxford)



I worked on…

– analytics infrastructure to support data science

– data products to surface relevant content to users



Busch et al. Earlybird: Real-Time Search at Twitter. ICDE 2012

Mishne et al. Fast Data in the Era of Big Data: Twitter's Real-Time Related Query Suggestion Architecture. SIGMOD 2013.

Leibert et al. Automatic Management of Partitioned, Replicated Search Services. SoCC 2011

Gupta et al. WTF: The Who to Follow Service at Twitter. WWW 2013

Lin and Kolcz. Large-Scale Machine Learning at Twitter. SIGMOD 2012

I worked on…

– analytics infrastructure to support data science

– data products to surface relevant content to users

Source: https://www.flickr.com/photos/bongtongol/3491316758/

circa ~2010

~150 people total

~60 Hadoop nodes

~6 people use analytics stack daily

circa ~2012

~1400 people total

10s of Ks of Hadoop nodes, multiple DCs

10s of PBs total Hadoop DW capacity

~100 TB ingest daily

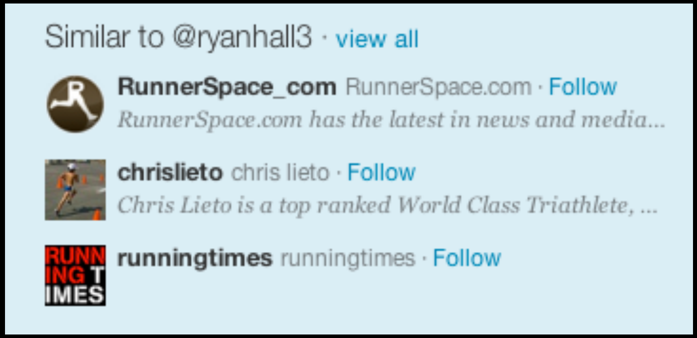
dozens of teams use Hadoop daily

10s of Ks of Hadoop jobs daily

WTF

(whom to follow)

(who to follow)



Launched summer 2010

MG Siegler @parislemon · Jul 27

@kevinweil @elizabeth OMG just seeing you secured @thirdweil for baby. Most amazing handle ever. Well played. (via @amy)

Reply Retweet Favorite More

Details

Follow

Elizabeth Weil
@elizabeth

@parislemon Got it in January. Then Twitter recommended the account to my uncle. Family guessed we were pregnant. Oops. Denied it all. ;)

Reply Retweet Favorite Share More

RETWEETS	FAVORITES
2	20

3:39 PM - 27 Jul 2014

#numbers

(Second half of 2012)

~175 million active users

~20 billion edges

42% edges bidirectional

Avg shortest path length: 4.05

40% as many unfollows as follows daily

WTF responsible for ~1/8 of the edges

Myers, Sharma, Gupta, Lin. Information Network or Social Network?
The Structure of the Twitter Follow Graph. WWW 2014.

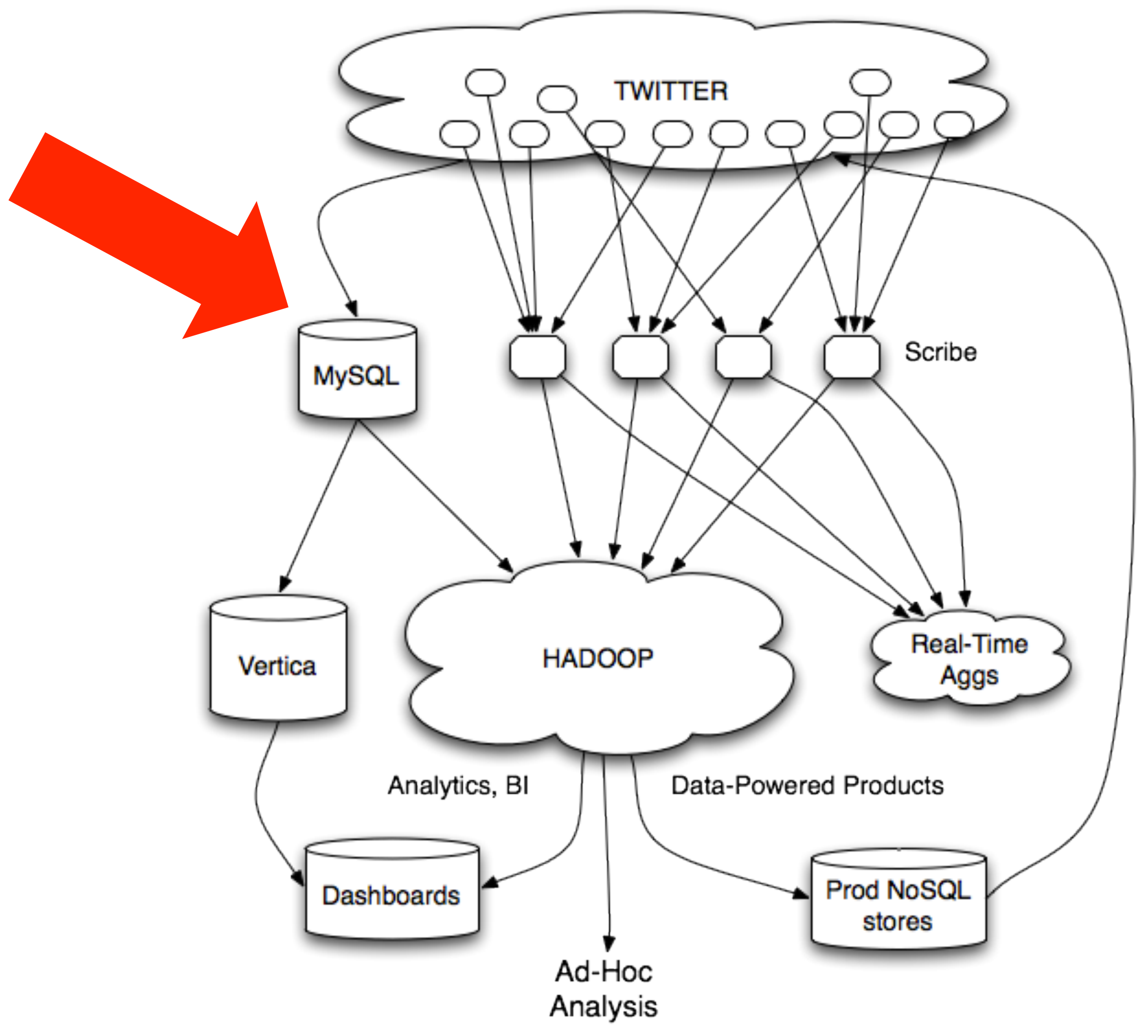
A talk in three episodes…

PROLOGUE

flockDB
(graph database)

Simple graph operations
Set intersection operations

Not appropriate for graph algorithms!


TWITTER
MySQL
Scribe
Vertica
HADOOP
Real-Time Aggs
Analytics, BI
Data-Powered Products
Dashboards
Prod NoSQL stores
Ad-Hoc Analysis

Use Hadoop!

MapReduce sucks for graph algorithms…

Java verbosity
Long task startup
Stragglers
Needless graph shuffling
Frequent checkpointing

What about?

HaLoop (VLDB 2010)
Twister (MapReduce Workshop 2010)
Pregel/Giraph (SIGMOD 2010)
Graphlab (UAI 2010)
PrIter (SoCC 2011)
Datalog on Hyracks (Tech report, 2012)
Spark/GraphX (NSDI 2012, arXiv 2014)
PowerGraph (OSDI 2012)
GRACE (CIDR 2013)
Mizan (EuroSys 2013)
…


A LONG TIME AGO IN A GALAXY FAR FAR AWAY...
STAR WARS
A NEW
~~HOPE~~
CASSOVARY

CIRCA 2010

MapReduce sucks for graph algorithms…
Let's build our own system!

Key design decision:

Keep entire graph in memory… on a single machine!

Nuts!

Why?

Because we can!
Graph partitioning is hard… so don't do it
Simple architecture

Right choice at the time!

Source: Wikipedia (Pistachio)

The runway argument

Source: Wikipedia (Heathrow)

Suppose: 10×10^9 edges

(src, dest) pairs: ~80 GB

18 × 8 GB DIMMS = 144 GB

18 × 16 GB DIMMS = 288 GB

12 × 16 GB DIMMS = 192 GB

12 × 32 GB DIMMS = 384 GB

Cassovary

- In-memory graph engine
  - Implemented in Scala
  - Compact in-memory representations
  - But no compression
  - Avoid JVM object overhead!
  - Open-source

Source: Wikipedia (Cassowary)

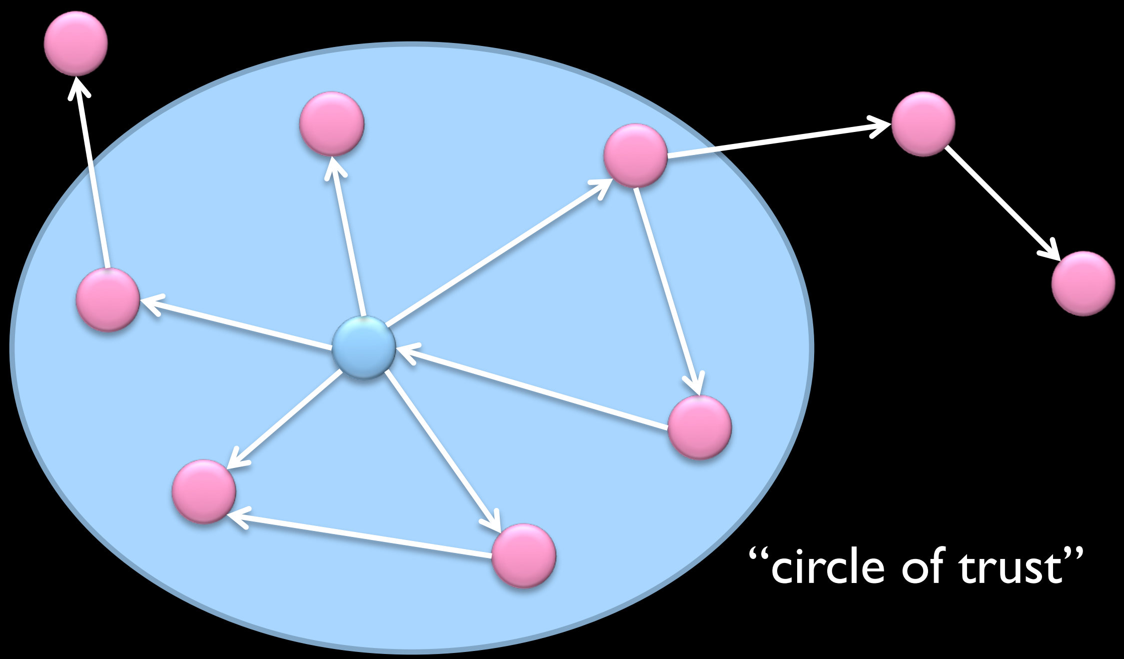
PageRank

- “Semi-streaming” algorithm
  - Keep vertex state in memory, stream over edges
  - Each pass = one PageRank iteration
  - Bottlenecked by memory bandwidth
- Convergence?
  - Don’t run from scratch… use previous values
  - A few passes are sufficient

“Circle of Trust”

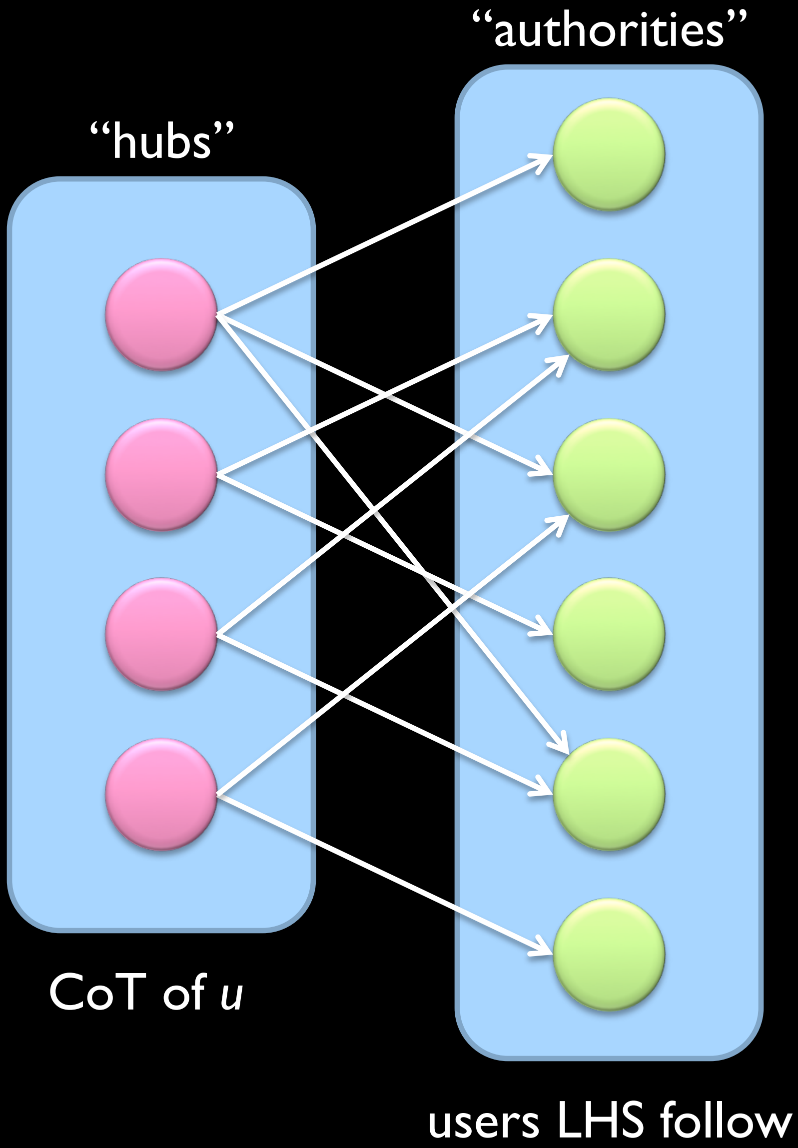
Ordered set of important neighbors for a user

- Result of egocentric random walk
- Computed online based on various input parameters



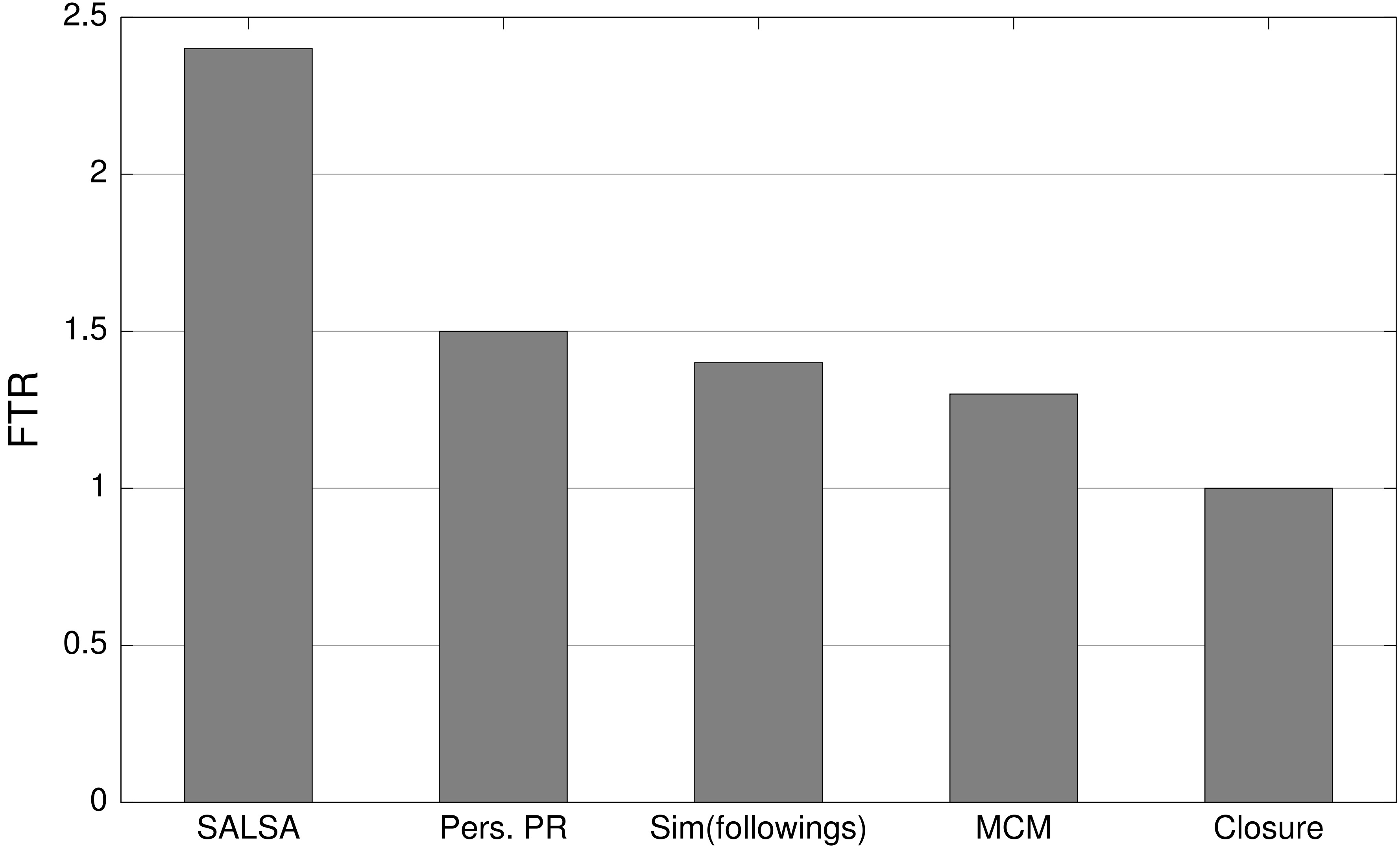
One of the features used in search

SALSA for Recommendations

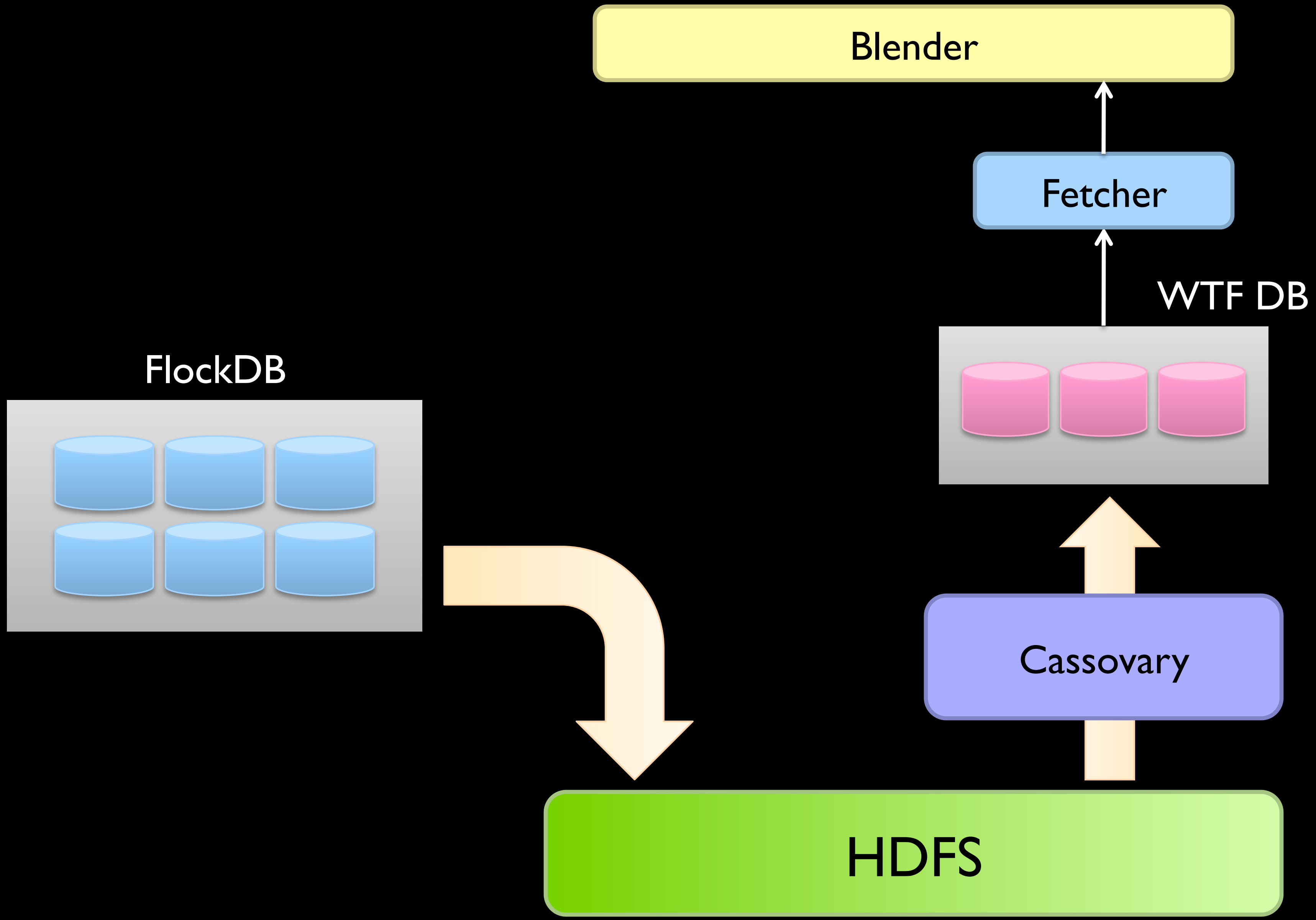


hubs scores:
similarity scores to *u*

authority scores:
recommendation scores for *u*



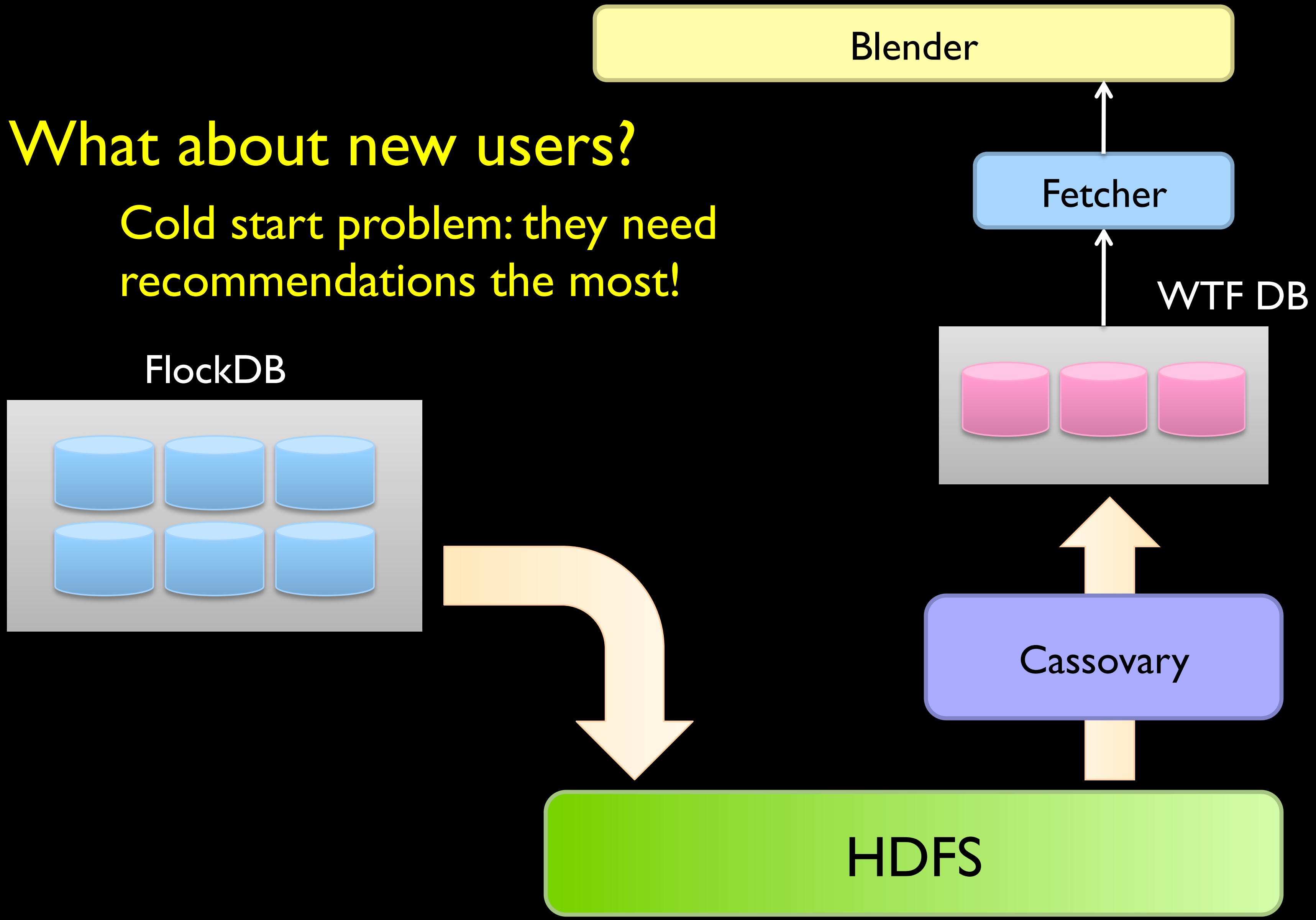
Goel, Lin, Sharma, Wang, and Zadeh. WTF: The Who to Follow Service at Twitter. WWW 2013

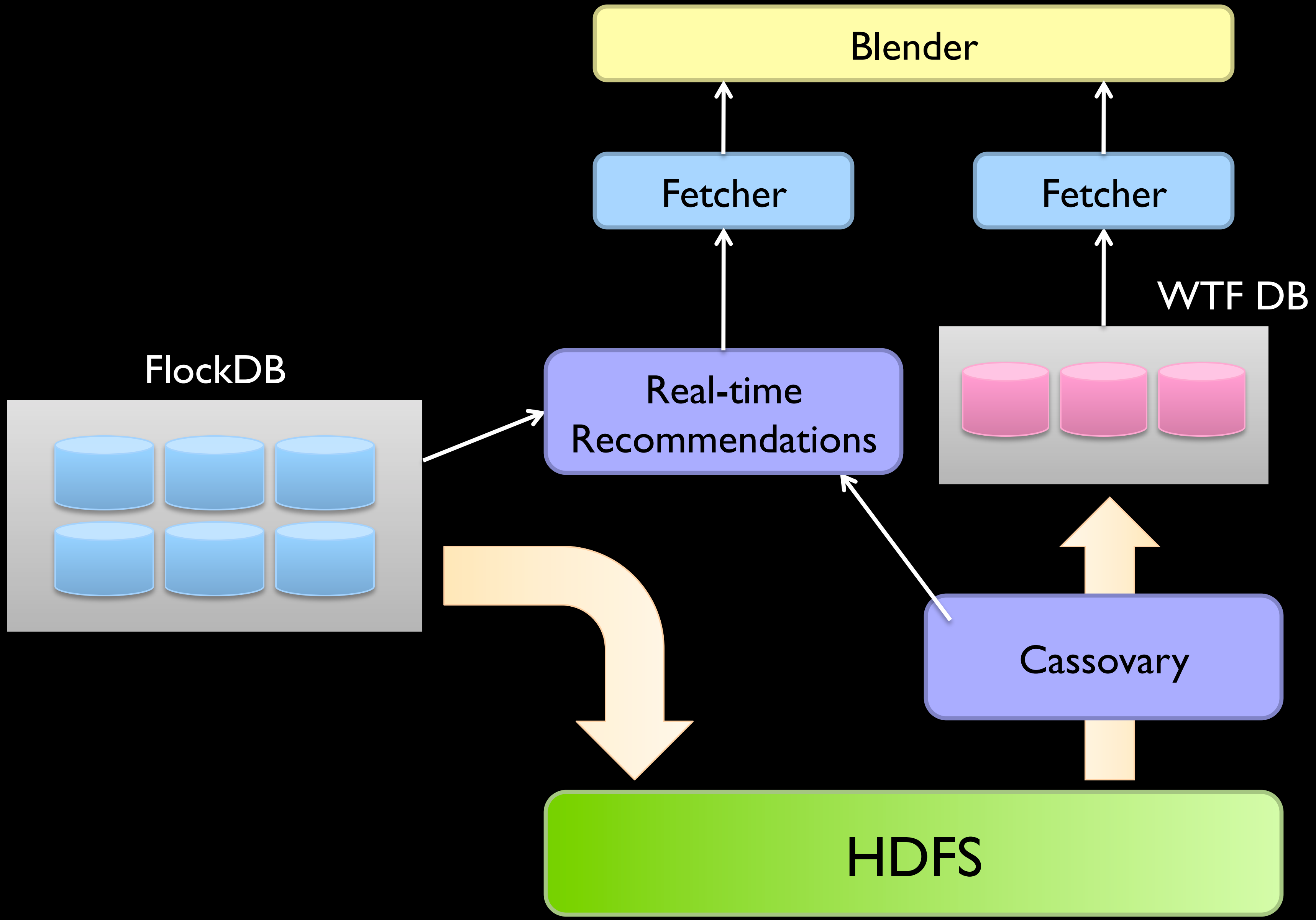


Blender
Fetcher
WTF DB
FlockDB
Cassovary
HDFS

What about new users?

Cold start problem: they need recommendations the most!




Blender
Fetcher
Fetcher
WTF DB
FlockDB
Real-time Recommendations
Cassovary
HDFS

Spring 2010: no WTF

seriously, WTF?

Summer 2010: WTF launched


THE STAR WARS SAGA CONTINUES...
HADOOP
THE ~~EMPIRE~~
STRIKES BACK
TWENTIETH CENTURY FOX LUCASFILM LTD. STAR WARS EPISODE V THE EMPIRE STRIKES BACK
MARK HAMILL HARRISON FORD CARRIE FISHER BILLY DEE WILLIAMS ANTHONY DANIELS C-3PO
DAVID PROWSE KENNY BAKER PETER MAYHEW FRANK OZ
IRVIN KERSHNER LEIGH BRACKETT LAWRENCE KASDAN GEORGE LUCAS GEORGE LUCAS JOHN WILLIAMS
THX
LUCASFILM
A LUCASFILM LTD. PRODUCTION
A TWENTIETH CENTURY RELEASE
WWW.STARWARS.COM

CIRCA 2012

Another “interesting” design choice:

We migrated from Cassovary back to Hadoop!

Source: Wikipedia (Pistachio)

Whaaaaa?

Cassovary was a stopgap!

Hadoop provides:

Richer graph structure

Simplified production infrastructure

Scaling and fault-tolerance “for free”

Right choice at the time!

Wait, didn't you say MapReduce sucks?

What exactly is the issue?

Random walks on egocentric 2-hop neighborhood

Naïve approach: self-joins to materialize, then run algorithm

The shuffle is what kills you!

Graph algorithms in MapReduce

Tackle the shuffling problem

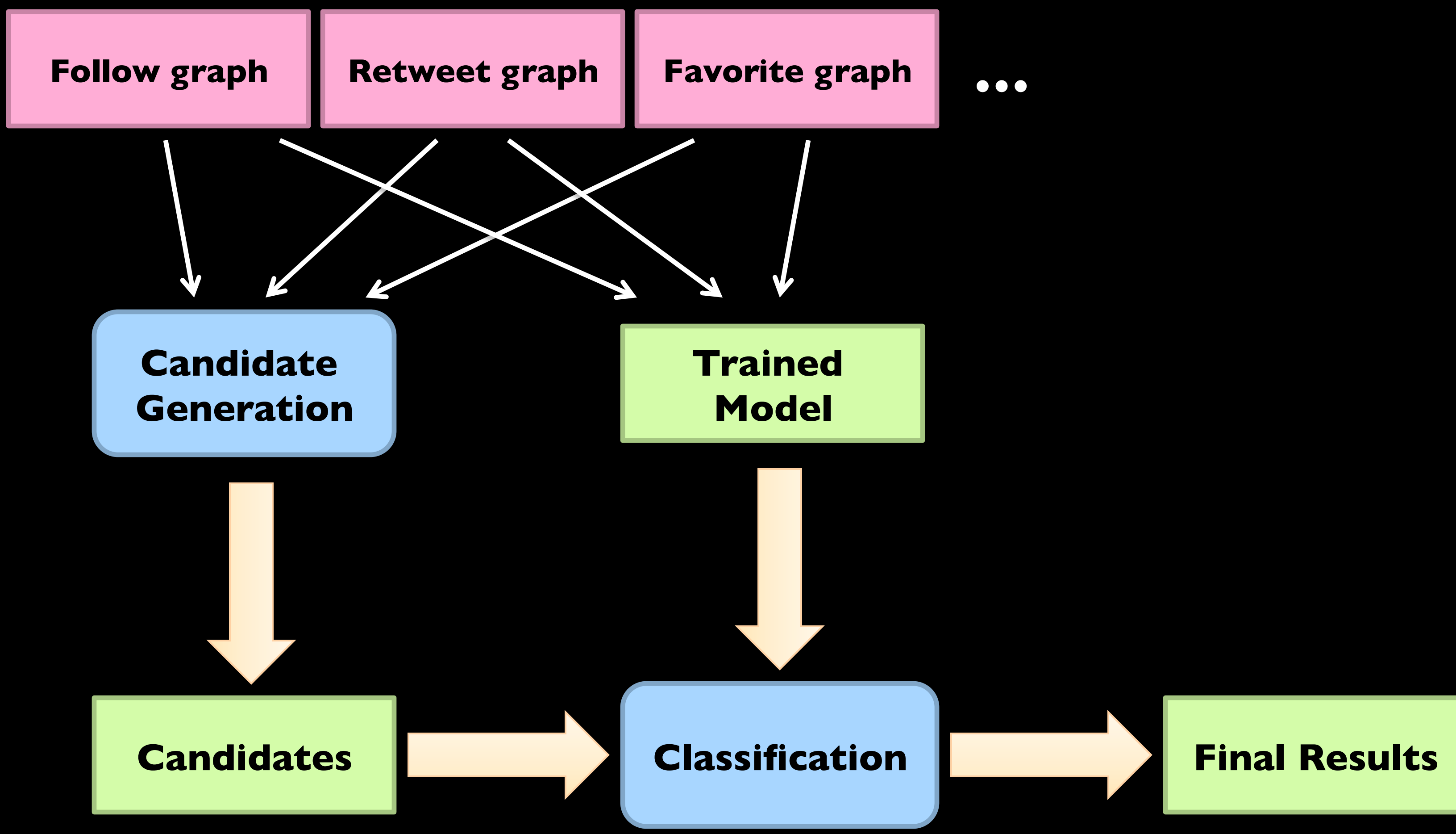
Key insights:

Batch and "stich together" partial random walks*

Clever sampling to avoid full materialization

* Sarma et al. Estimating PageRank on Graph Streams. PODS 200
Bahmani et al. Fast Personalized PageRank on MapReduce. SIGMOD 2011.

Throw in ML while we're at it…



Lin and Kolcz. Large-Scale Machine Learning at Twitter. SIGMOD 2012.

CUSTOM ARCHITECTURES

CIRCA 2013

Isn't the point of Twitter real-time?

So why is WTF still dominated by batch processing?

Source: Wikipedia (Motion Blur)

From batch to real-time recommendations:

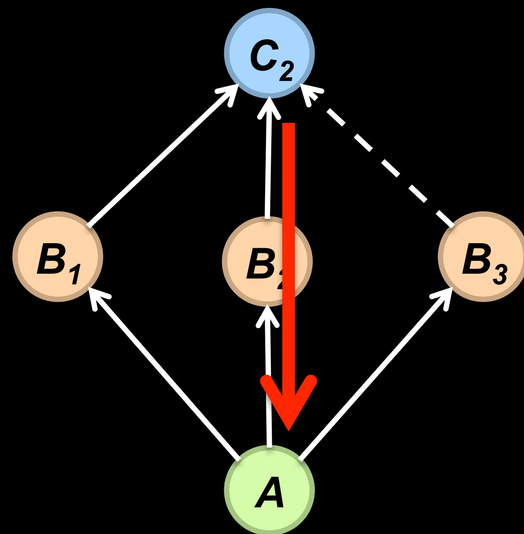
Recommendations based on recent activity

“Trending in your network”

Inverts the WTF problem:

For this user, what recommendations to generate?

Given this new edge, which user to make recommendations to?



Why does this work?

A follows B's because they're interesting

B's following C's because "something's happening"

(generalizes to any activity)

Gupta, Satuluri, Grewal, Gurumurthy, Zhabiuk, Li, and Lin. Real-Time Twitter Recommendation: Online Motif Detection in Large Dynamic Graphs. VLDB 2014

Scale of the Problem

$O(10^8)$ vertices, $O(10^{10})$ edges

Designed for $O(10^4)$ events per second

Naïve solutions:

Poll each vertex periodically

Materialize everyone's two-hop neighborhood, intersect

Production solution:

Idea #1: Convert problem into adjacency list intersection

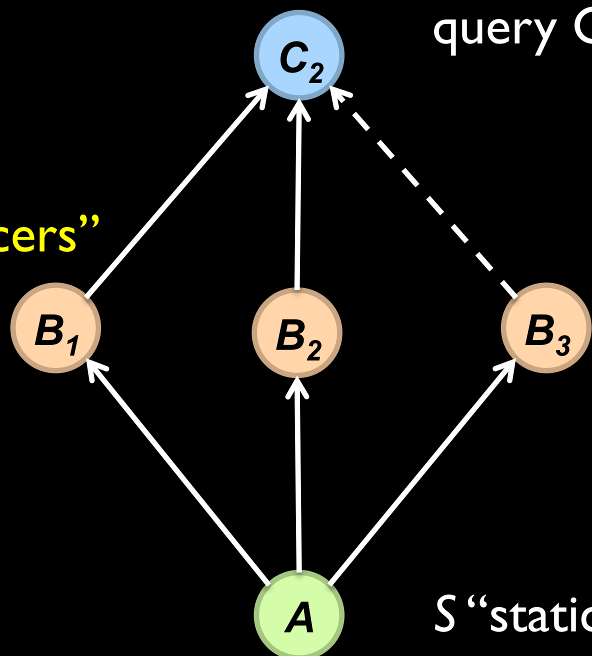
Idea #2: Partition graph to eliminate non-local intersections

Gupta, Satuluri, Grewal, Gurumurthy, Zhabiuk, Li, and Lin. Real-Time Twitter Recommendation: Online Motif Detection in Large Dynamic Graphs. VLDB 2014

Single Node Solution

Who we're recommending

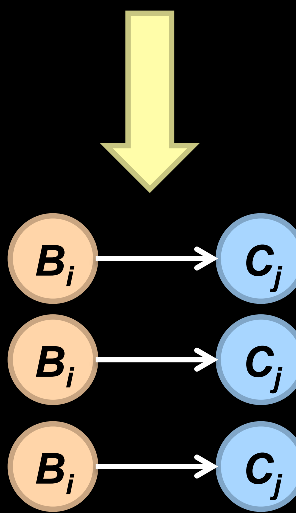
“influencers”



Who we're making the recommendations to

D “dynamic” structure:
stores inverted adjacency lists

query C, return all B's that link to it

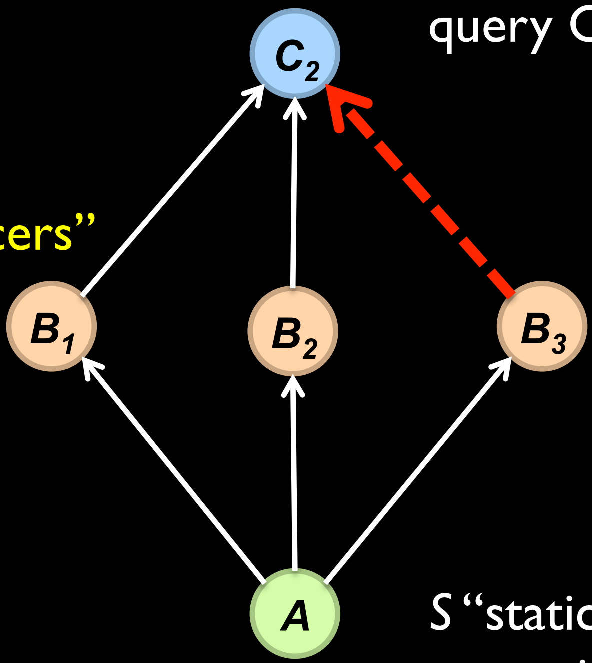


S “static” structure:
stores inverted adjacency lists

query B, return all A's that link to it

Algorithm

Who we're recommending



"influencers"

Who we're making the recommendations to

D "dynamic" structure:
stores inverted adjacency lists

query C, return all B's that link to it

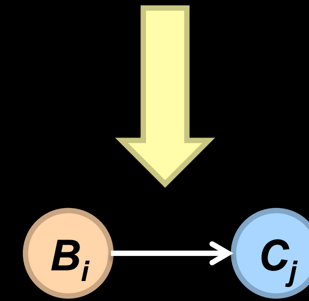
1. Receive B_3 to C_2
2. Query D for C_2, get B_1, B_2, B_3
3. For each B_1, B_2, B_3, query S
4. Intersect lists to compute A's

S "static" structure:
stores inverted adjacency lists

query B, return all A's that link to it

Idea #1: Convert problem into adjacency list intersection

Distributed Solution

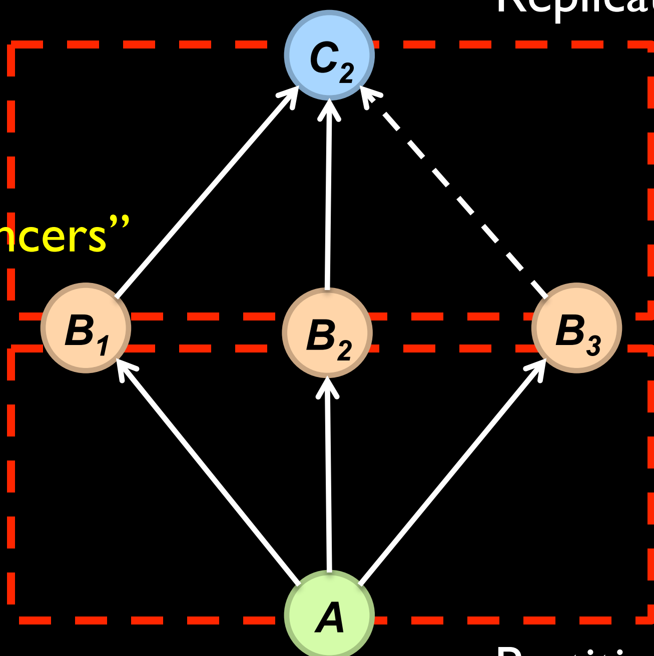


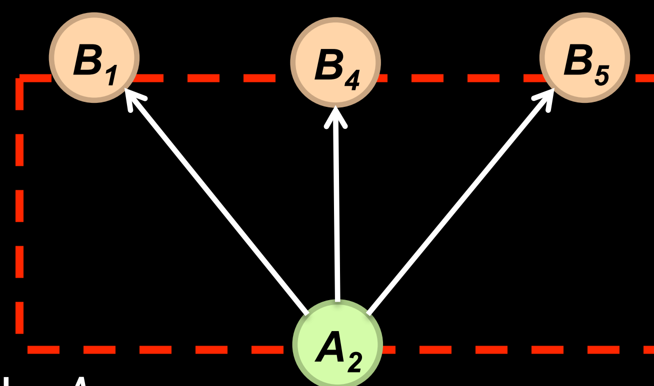
Who we're recommending

Replicate on every node

1. Fan out new edge to every node
2. Run algorithm on each partition
3. Gather results from each partition

“influencers”





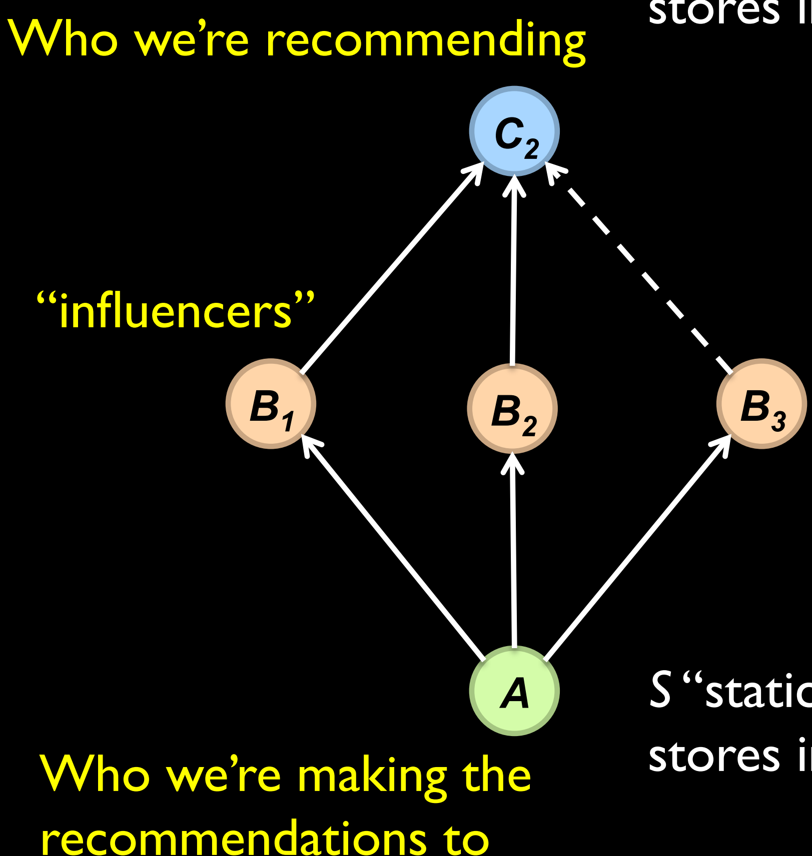
Partition by *A*

Who we're making the recommendations to

Idea #2: Partition graph to eliminate non-local intersections

Notes

D "dynamic" structure:
stores inverted adjacency lists



Memory pressure?

Why?

S "static" structure:
stores inverted adjacency lists

Production Status

Launched September 2013

Push recommendations to Twitter mobile users

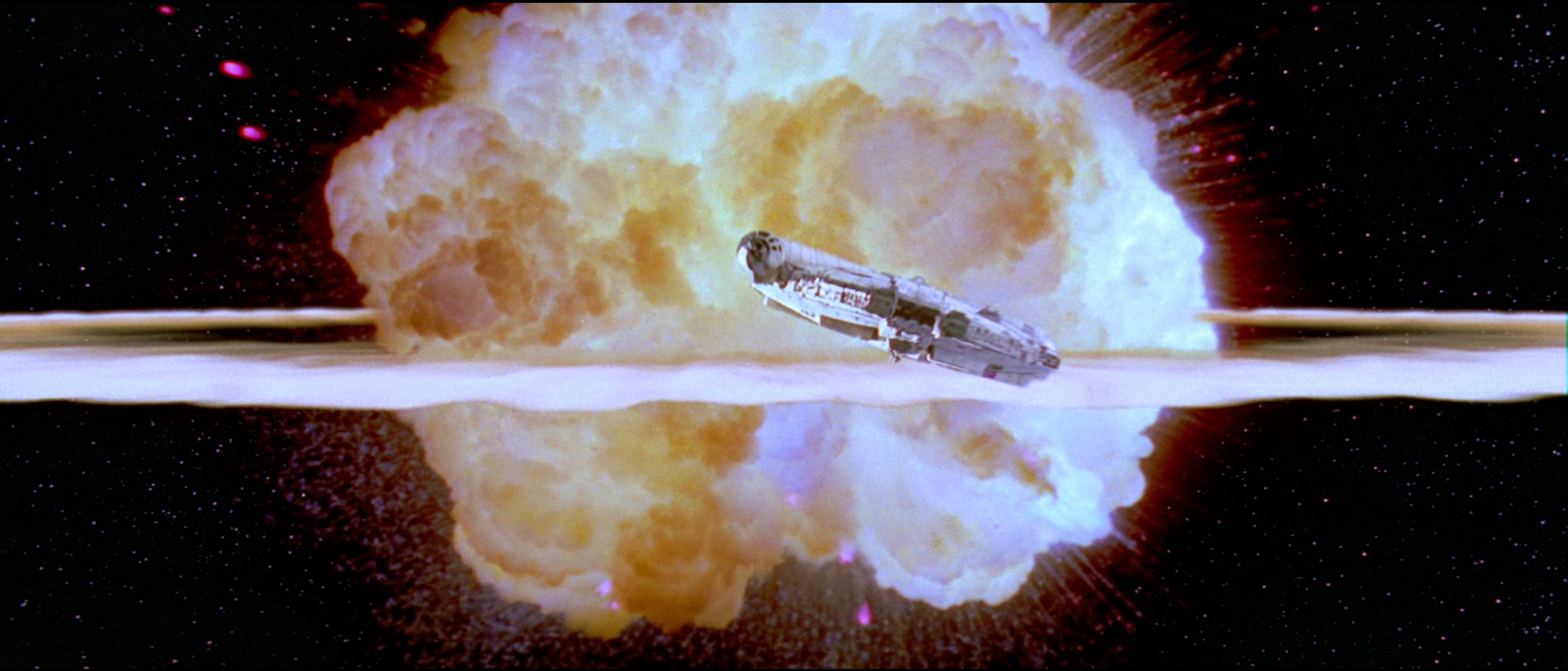
Billions of raw candidates, millions of push notifications daily

Performance

End-to-end latency (from edge creation to delivery):

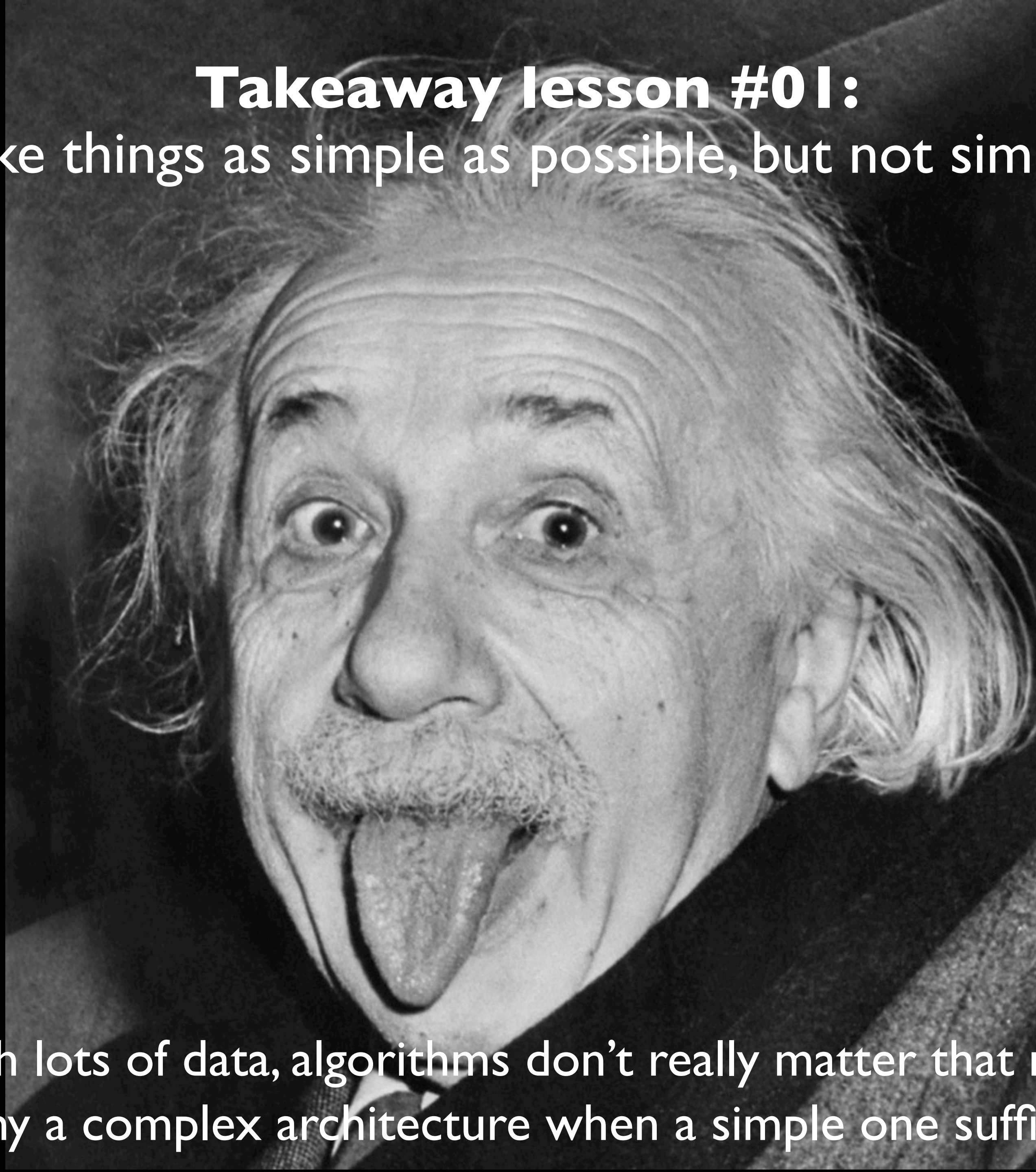
median 7s, p99 15s

Gupta, Satuluri, Grewal, Gurumurthy, Zhabiuk, Li, and Lin. Real-Time Twitter Recommendation: Online Motif Detection in Large Dynamic Graphs. VLDB 2014



Takeaway lesson #01:

Make things as simple as possible, but not simpler.



With lots of data, algorithms don’t really matter that much

Why a complex architecture when a simple one suffices?

Takeaway lesson #10:
Constraints aren't always technical.

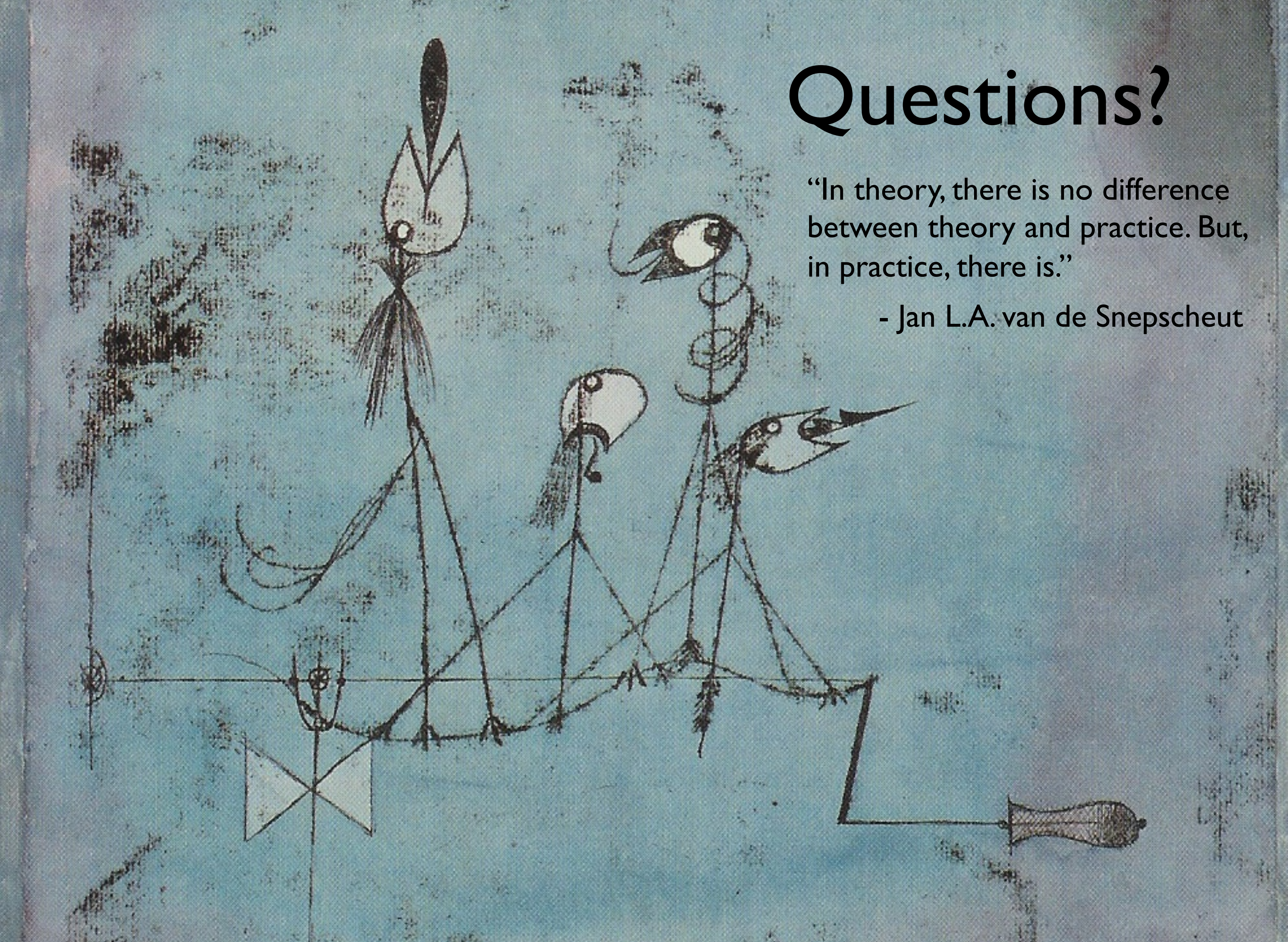
Source: https://www.flickr.com/photos/43677780@N07/6240710770/


Takeaway lesson #11:
Plumbing matters. A lot.
Source: Wikipedia (Plumbing)

Questions?

"In theory, there is no difference between theory and practice. But, in practice, there is."

- Jan L.A. van de Snepscheut



Twittering Machine. Paul Klee (1922) watercolor and ink