Quanta: Quora's Hierarchical Counting System On HBase

Nikhil Garg

@nikhilgarg28

Chun-Ho Hung

@chhung6

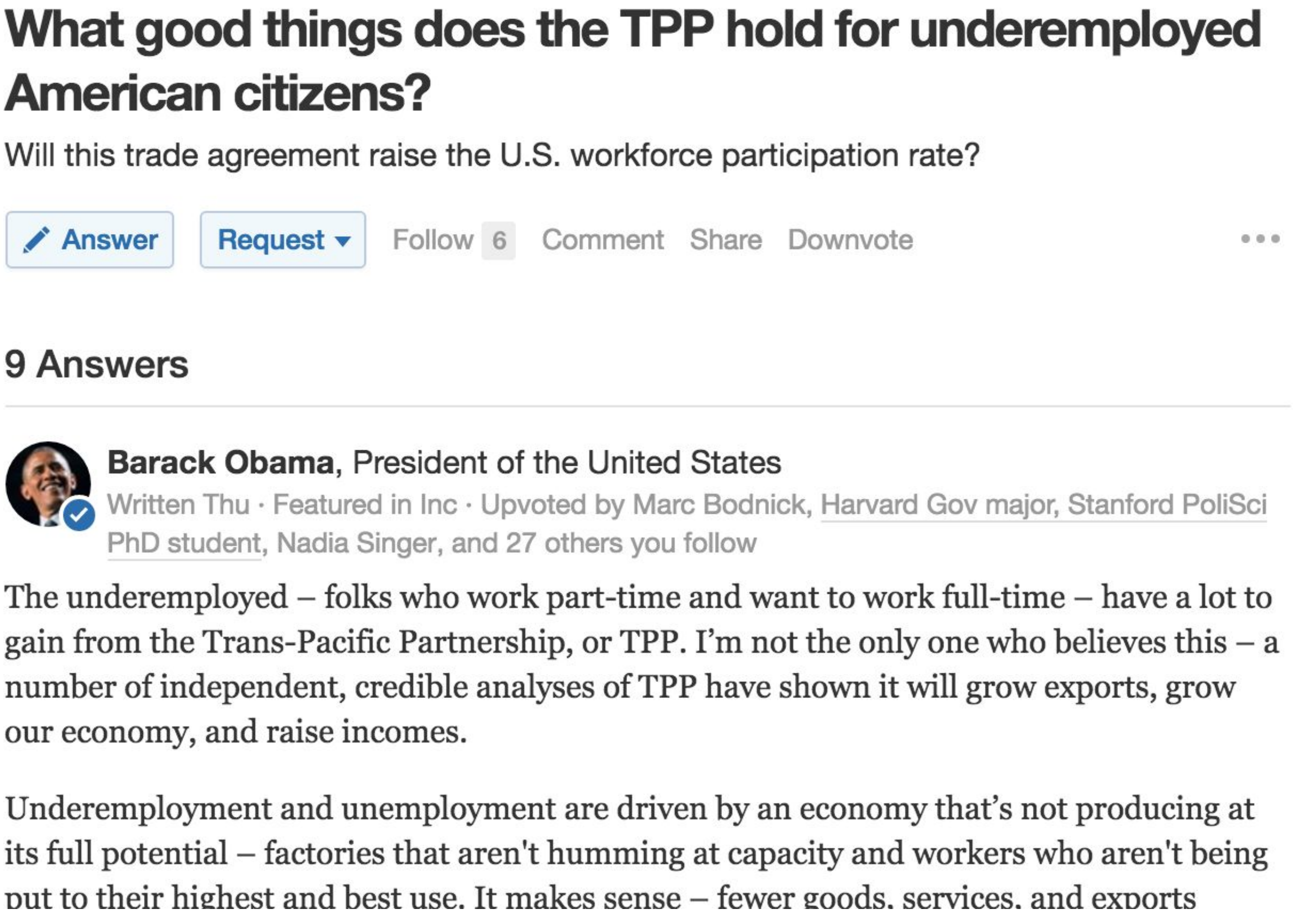
@Quora

HBaseCon 6/12/17

Quora

The best answer to any question

To Grow And Share World’s Knowledge



What good things does the TPP hold for underemployed American citizens?

Will this trade agreement raise the U.S. workforce participation rate?

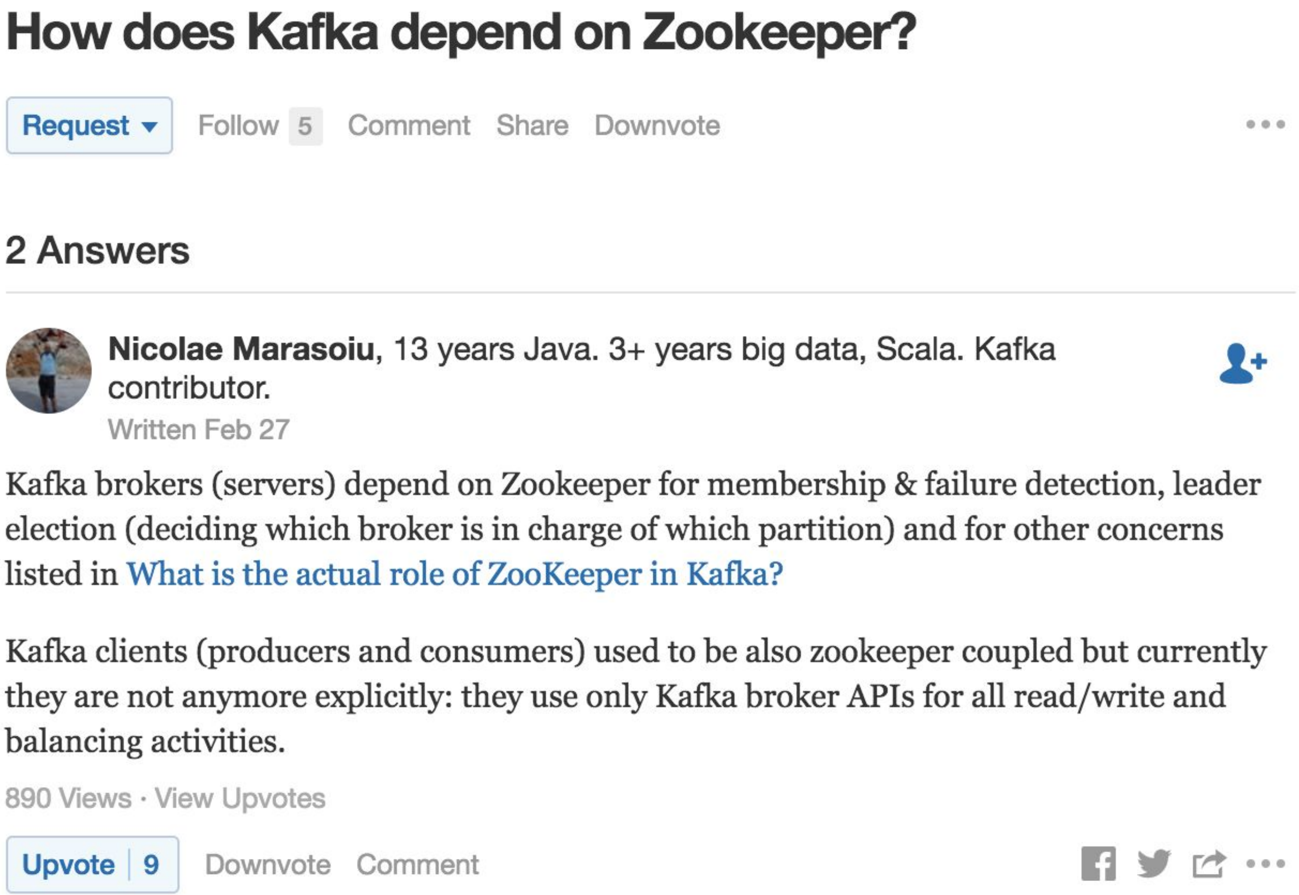
9 Answers

Barack Obama, President of the United States

Written Thu · Featured in Inc · Upvoted by Marc Bodnick, Harvard Gov major, Stanford PoliSci PhD student, Nadia Singer, and 27 others you follow

The underemployed – folks who work part-time and want to work full-time – have a lot to gain from the Trans-Pacific Partnership, or TPP. I'm not the only one who believes this – a number of independent, credible analyses of TPP have shown it will grow exports, grow our economy, and raise incomes.

Underemployment and unemployment are driven by an economy that's not producing at its full potential – factories that aren't humming at capacity and workers who aren't being put to their highest and best use. It makes sense – fewer goods, services, and exports



How does Kafka depend on Zookeeper?

2 Answers

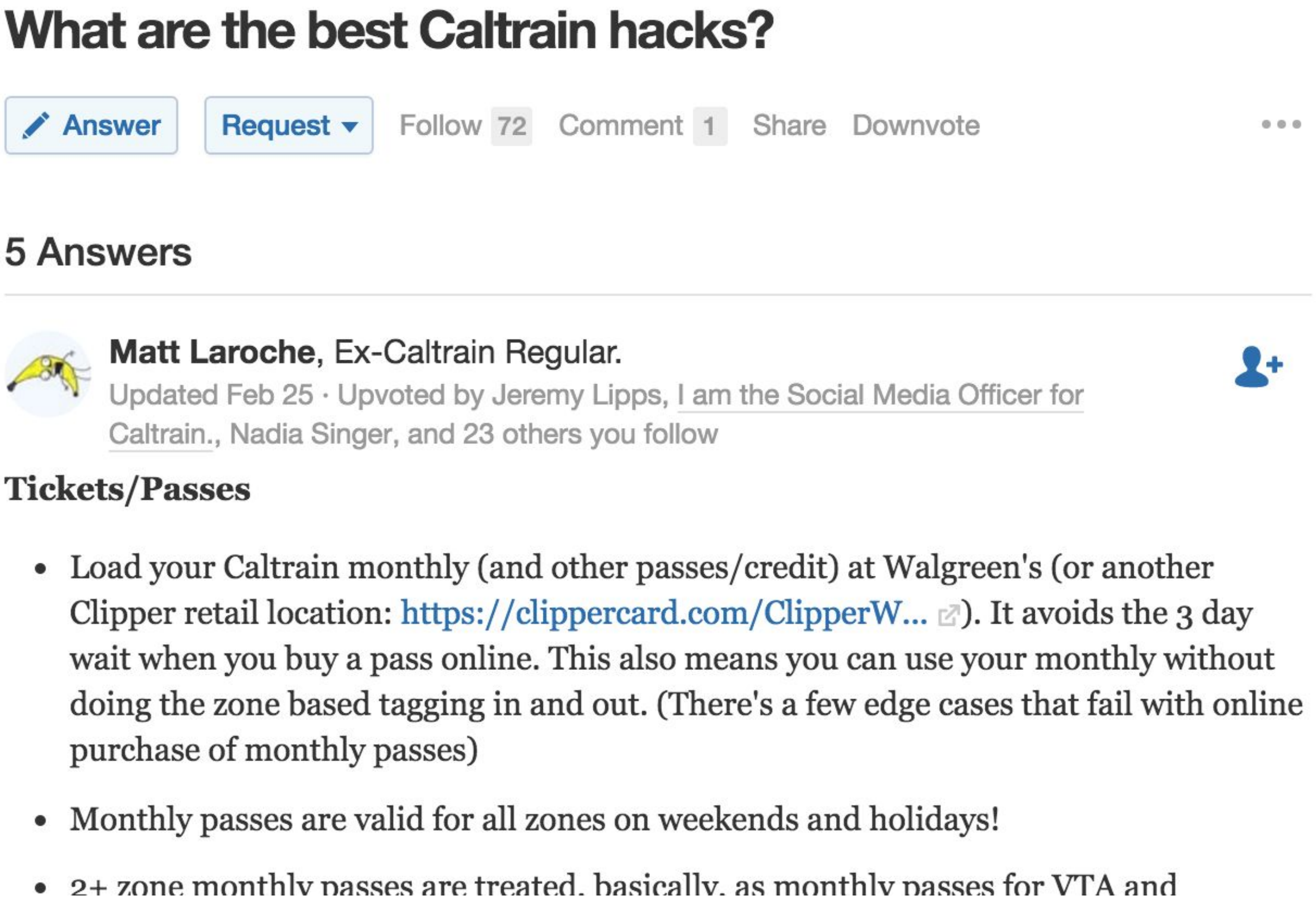
Nicolae Marasoiu, 13 years Java. 3+ years big data, Scala. Kafka contributor.

Written Feb 27

Kafka brokers (servers) depend on Zookeeper for membership & failure detection, leader election (deciding which broker is in charge of which partition) and for other concerns listed in What is the actual role of ZooKeeper in Kafka?

Kafka clients (producers and consumers) used to be also zookeeper coupled but currently they are not anymore explicitly: they use only Kafka broker APIs for all read/write and balancing activities.

890 Views · View Upvotes



What are the best Caltrain hacks?

5 Answers

Matt Laroche, Ex-Caltrain Regular.

Updated Feb 25 · Upvoted by Jeremy Lipps, I am the Social Media Officer for Caltrain., Nadia Singer, and 23 others you follow

Tickets/Passes

- Load your Caltrain monthly (and other passes/credit) at Walgreen's (or another Clipper retail location: https://clippercard.com/ClipperW...). It avoids the 3 day wait when you buy a pass online. This also means you can use your monthly without doing the zone based tagging in and out. (There's a few edge cases that fail with online purchase of monthly passes)
- Monthly passes are valid for all zones on weekends and holidays!
- 2+ zone monthly passes are treated, basically, as monthly passes for VTA and

Around 200 million monthly uniques

Millions of questions & answers

In hundreds of thousands of topics

Supported by < 100 engineers

Quora’s Big Data In Cloud

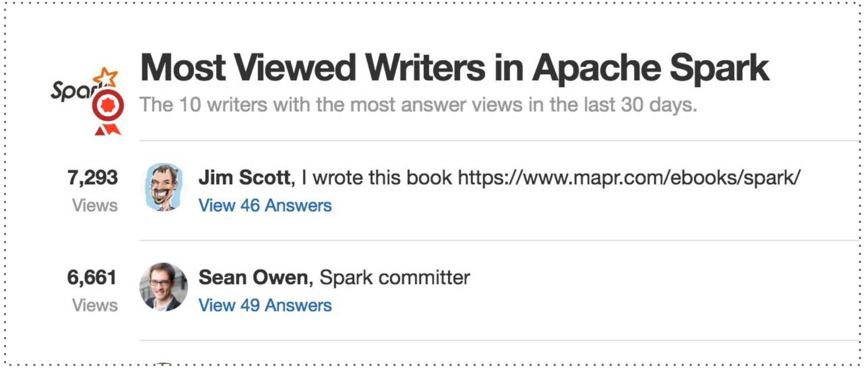
- All infrastructure is on AWS from t = 0
- MySQL and HBase are online production databases
- Redshift: ds2.8xlarge boxes, hundreds of terabytes of data
- Spark: d2.2xlarge, runs HDFS + YARN_MR + Spark, hundreds of terabytes of data

Quanta

Over 1 Billion Realtime Updates/Reads Everyday

Users of Quanta

- All features based on content views
- All Ads monitoring and tracking
- Exploring:
  - Internal Dashboards
  - Deduplication service
  - Count based ML Features
  - Monitoring native app performance/reliability

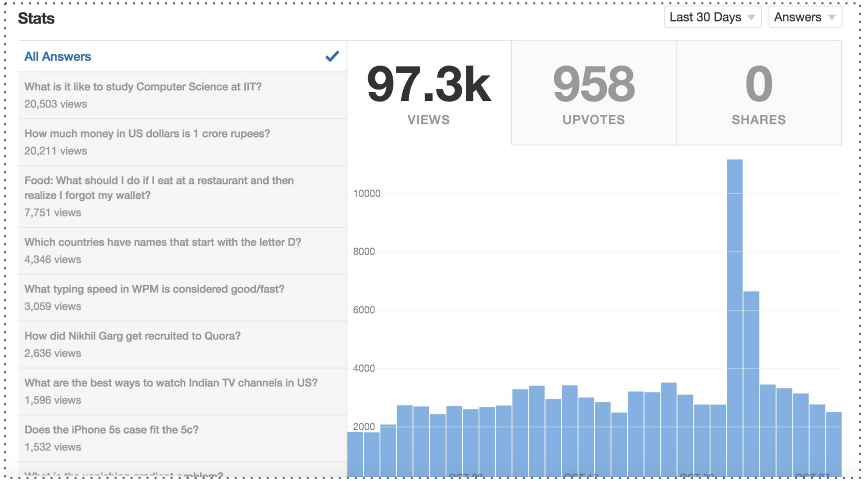


Most Viewed Writers in Apache Spark

The 10 writers with the most answer views in the last 30 days.

7,293 Views — **Jim Scott**, I wrote this book https://www.mapr.com/ebooks/spark/
View 46 Answers

6,661 Views — **Sean Owen**, Spark committer
View 49 Answers



Stats

Last 30 Days ▾ Answers ▾

All Answers ✓

What is it like to study Computer Science at IIT?
20,503 views

How much money in US dollars is 1 crore rupees?
20,211 views

Food: What should I do if I eat at a restaurant and then realize I forgot my wallet?
7,751 views

Which countries have names that start with the letter D?
4,346 views

What typing speed in WPM is considered good/fast?
3,059 views

How did Nikhil Garg get recruited to Quora?
2,636 views

What are the best ways to watch Indian TV channels in US?
1,596 views

Does the iPhone 5s case fit the 5c?
1,532 views

What is the vanishing gradient problem?

97.3k VIEWS

958 UPVOTES

0 SHARES

1. **Design Constraints**

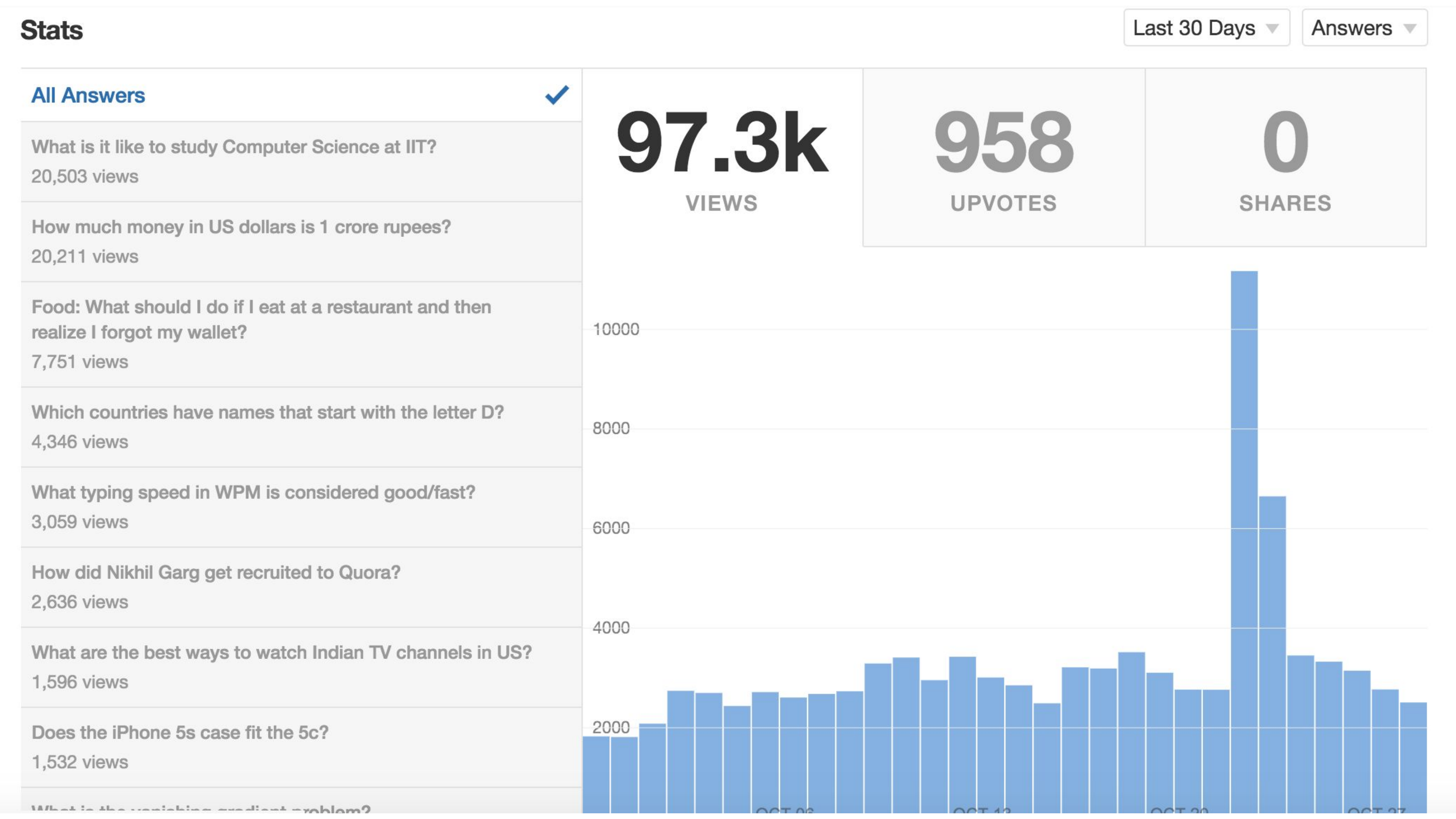
2. **Update Algorithm**

3. **Macro Architecture**

4. **Towards Horizontal Scalability**

5. **Alternatives**

1. **Design Constraints**
2. Update Algorithm
3. Macro Architecture
4. Towards Horizontal Scalability
5. Alternatives



Stats

All Answers ✓

What is it like to study Computer Science at IIT?
20,503 views

How much money in US dollars is 1 crore rupees?
20,211 views

Food: What should I do if I eat at a restaurant and then realize I forgot my wallet?
7,751 views

Which countries have names that start with the letter D?
4,346 views

What typing speed in WPM is considered good/fast?
3,059 views

How did Nikhil Garg get recruited to Quora?
2,636 views

What are the best ways to watch Indian TV channels in US?
1,596 views

Does the iPhone 5s case fit the 5c?
1,532 views

What is the vanishing gradient problem?

97.3k	958	0
VIEWS	UPVOTES	SHARES

Design Constraints

- An explicit commit point after which updates are never lost.
- Maintain billions of counters -- not bound by a single machine.
- Very high volume of writes (> 100K/sec) and reads (>1M/sec).

Design Constraints

- An explicit commit point after which updates are never lost.

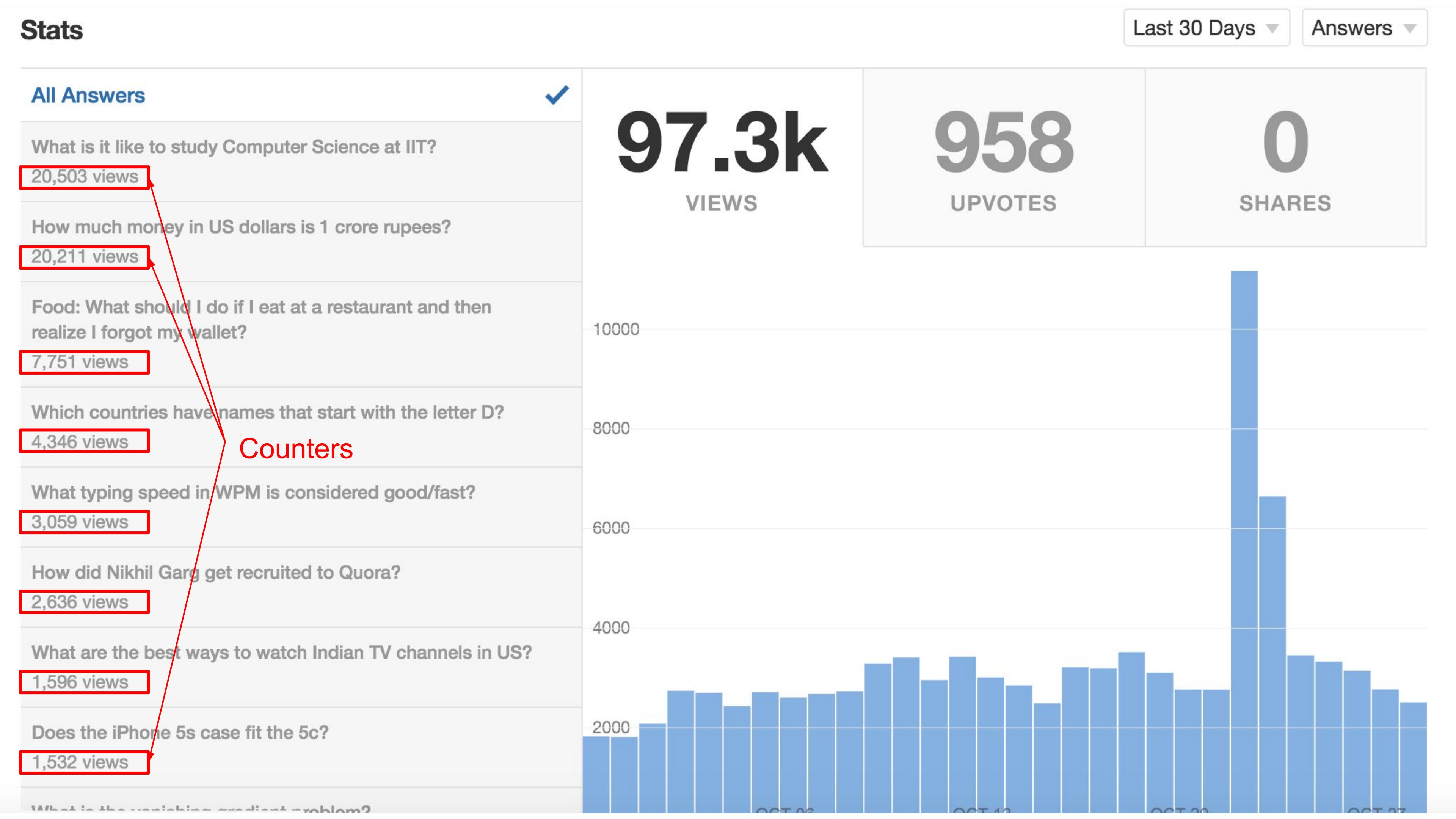
  ⇒ Persistent on disk, replicated

- Maintain billions of counters -- not bound by a single machine.

  ⇒ Distributed on multiple machines

- Very high volume of writes (> 100K/sec) and reads (>1M/sec).

  ⇒ Append-only writes



Stats

Last 30 Days ▾ Answers ▾

All Answers ✓

What is it like to study Computer Science at IIT?
20,503 views

How much money in US dollars is 1 crore rupees?
20,211 views

Food: What should I do if I eat at a restaurant and then realize I forgot my wallet?
7,751 views

Which countries have names that start with the letter D?
4,346 views

What typing speed in WPM is considered good/fast?
3,059 views

How did Nikhil Garg get recruited to Quora?
2,636 views

What are the best ways to watch Indian TV channels in US?
1,596 views

Does the iPhone 5s case fit the 5c?
1,532 views

What is the vanishing gradient problem?

97.3k VIEWS

958 UPVOTES

0 SHARES

Counters

Design Constraints

- Low latency reads on a single counter and some “related” counters (~1ms)

Design Constraints

- Low latency reads on a single counter and some “related” counters (~1ms)

    ⇒ avoid random reads in the same request

    ⇒ heavy in-memory caching

    ⇒ storing in range-sorted format

Design Constraints

- Low latency reads on a single counter and some “related” counters (~1ms).

  ⇒ avoid random reads in the same request

  ⇒ heavy in-memory caching

  ⇒ storing in range-sorted format

- Writes lag behind by at most a couple of minutes.

Design Constraints

- Low latency reads on a single counter and some “related” counters (~1ms).

  ⇒ avoid random reads in the same request

  ⇒ heavy in-memory caching

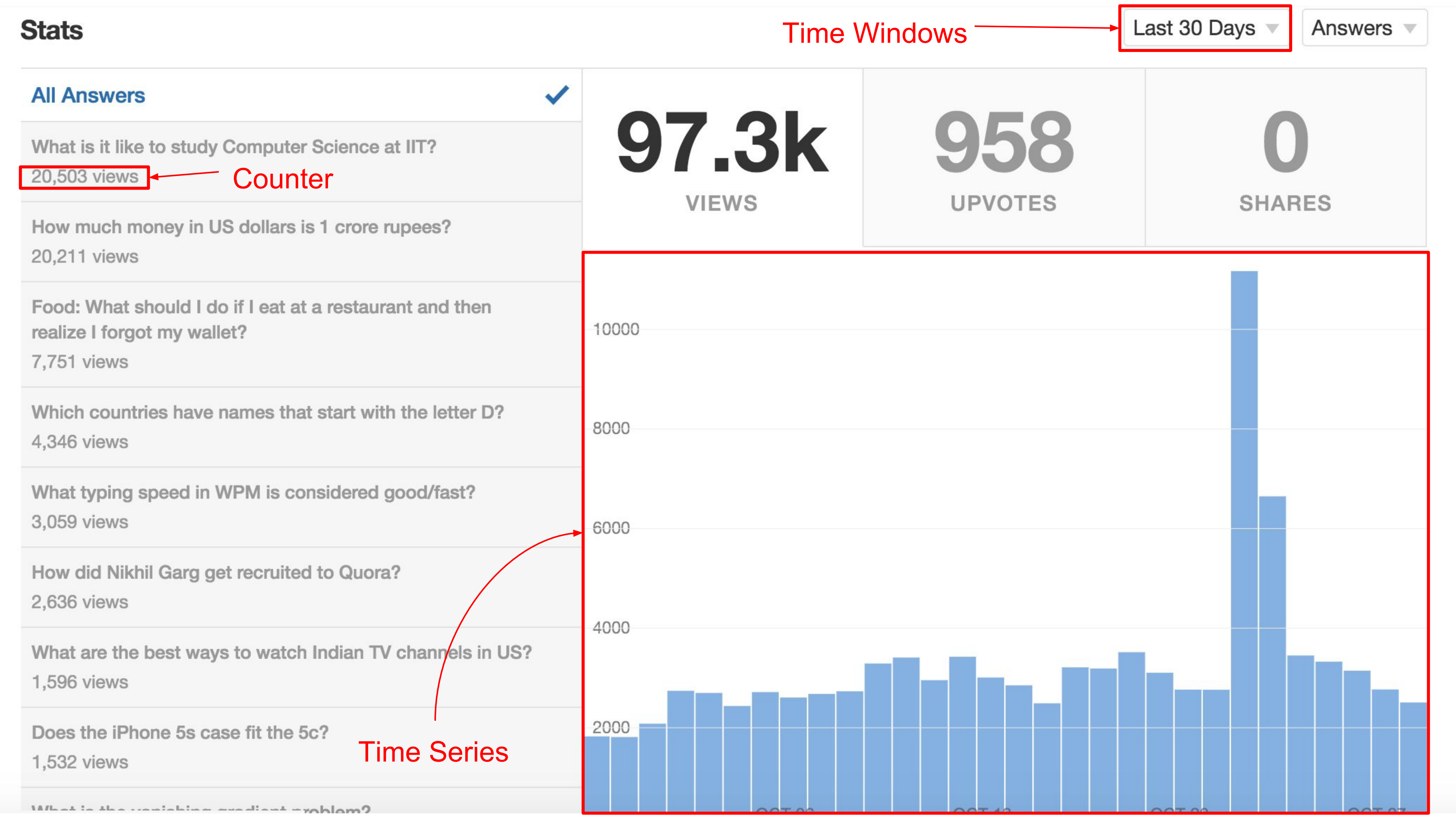
  ⇒ storing in range-sorted format

- Writes lag behind by at most a couple of minutes.

  ⇒ can do asynchronous writes

Design Constraints

- Should be highly available for both reads and writes.
- Handle 2x the load with 2x the resources, should scale to 20-50x.
- Should be as cheap as possible.



Stats

Time Windows

Last 30 Days ▾

Answers ▾

All Answers ✓

What is it like to study Computer Science at IIT?
20,503 views

Counter

How much money in US dollars is 1 crore rupees?
20,211 views

Food: What should I do if I eat at a restaurant and then realize I forgot my wallet?
7,751 views

Which countries have names that start with the letter D?
4,346 views

What typing speed in WPM is considered good/fast?
3,059 views

How did Nikhil Garg get recruited to Quora?
2,636 views

What are the best ways to watch Indian TV channels in US?
1,596 views

Does the iPhone 5s case fit the 5c?
1,532 views

What is the vanishing gradient problem?

97.3k VIEWS

958 UPVOTES

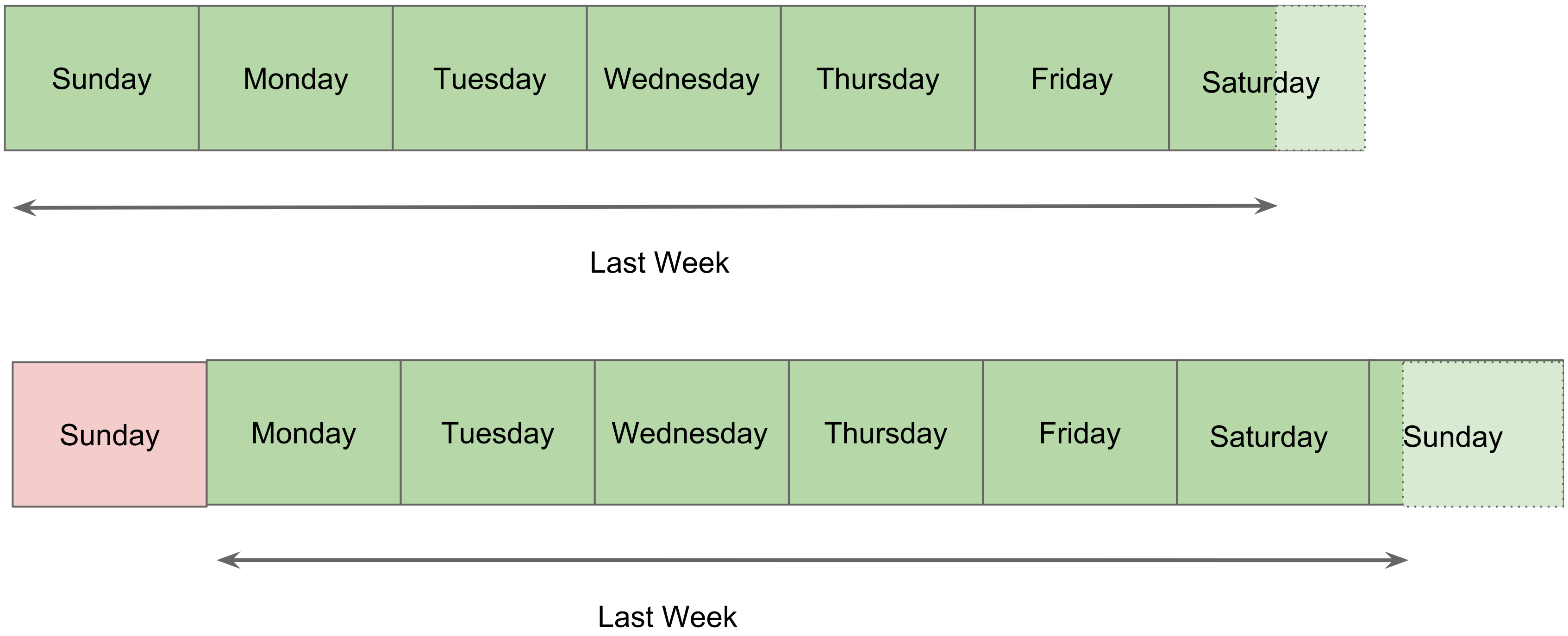
0 SHARES

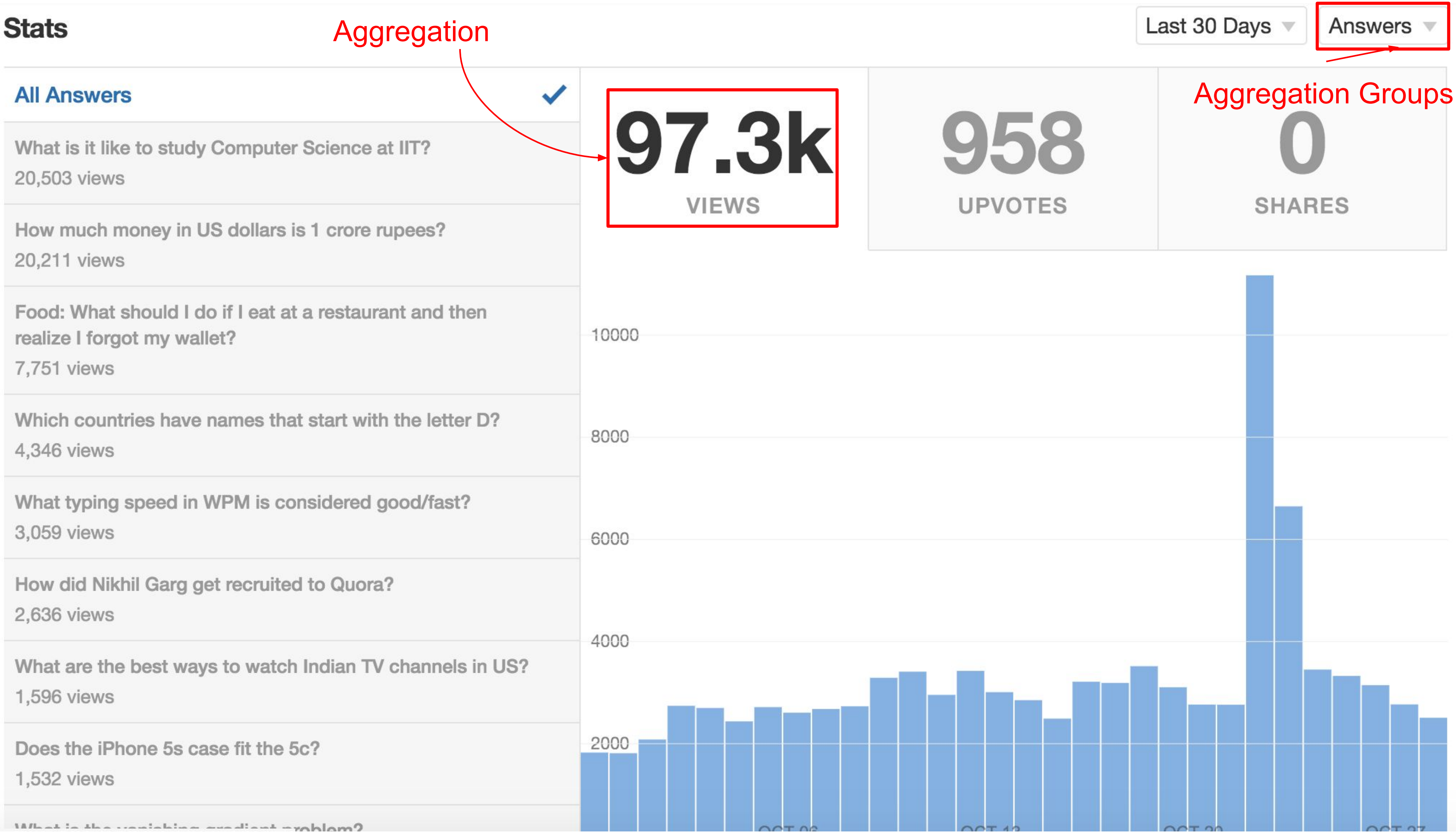
Time Series

Arbitrary Time Bucketing

- Should support counting in rolling windows -- “How many times did this happen in **last X**”?
- Should support storing timeseries -- “How many times did this happen **every X** in the **last Y**?”
- **X** and **Y** can be arbitrary time units.

Configurable Rolling Errors





Stats

Aggregation

Aggregation Groups

All Answers ✓

What is it like to study Computer Science at IIT?
20,503 views

How much money in US dollars is 1 crore rupees?
20,211 views

Food: What should I do if I eat at a restaurant and then realize I forgot my wallet?
7,751 views

Which countries have names that start with the letter D?
4,346 views

What typing speed in WPM is considered good/fast?
3,059 views

How did Nikhil Garg get recruited to Quora?
2,636 views

What are the best ways to watch Indian TV channels in US?
1,596 views

Does the iPhone 5s case fit the 5c?
1,532 views

What is the vanishing gradient problem?

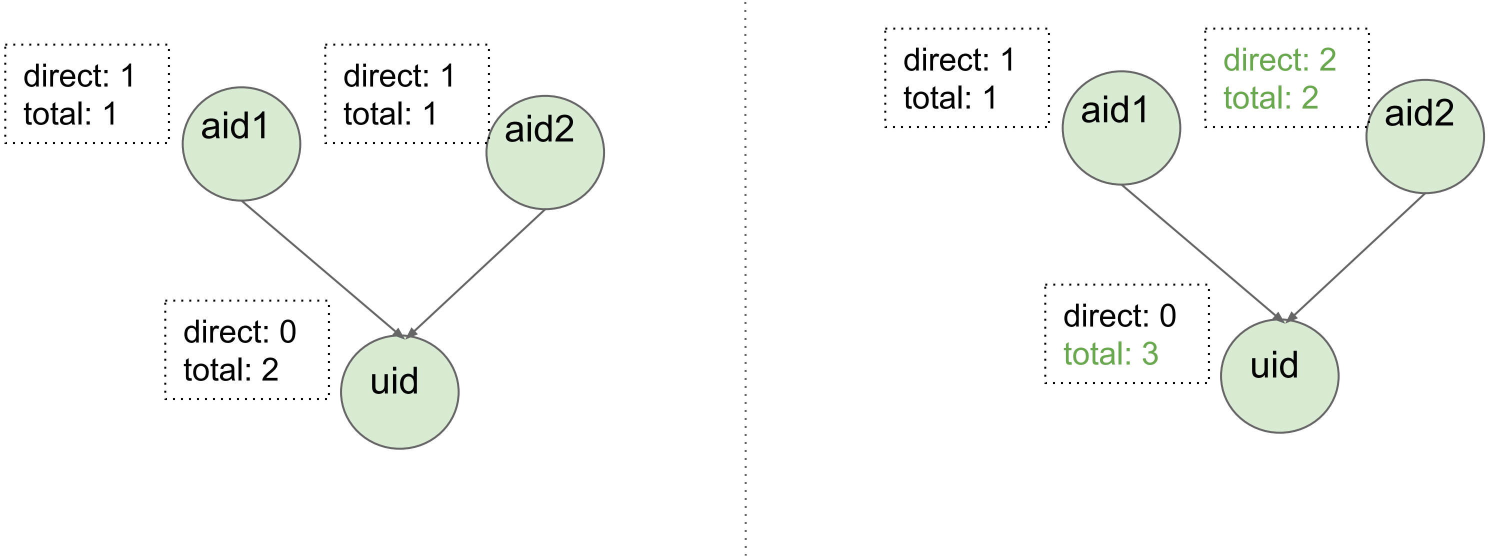
97.3k
VIEWS

958
UPVOTES

0
SHARES

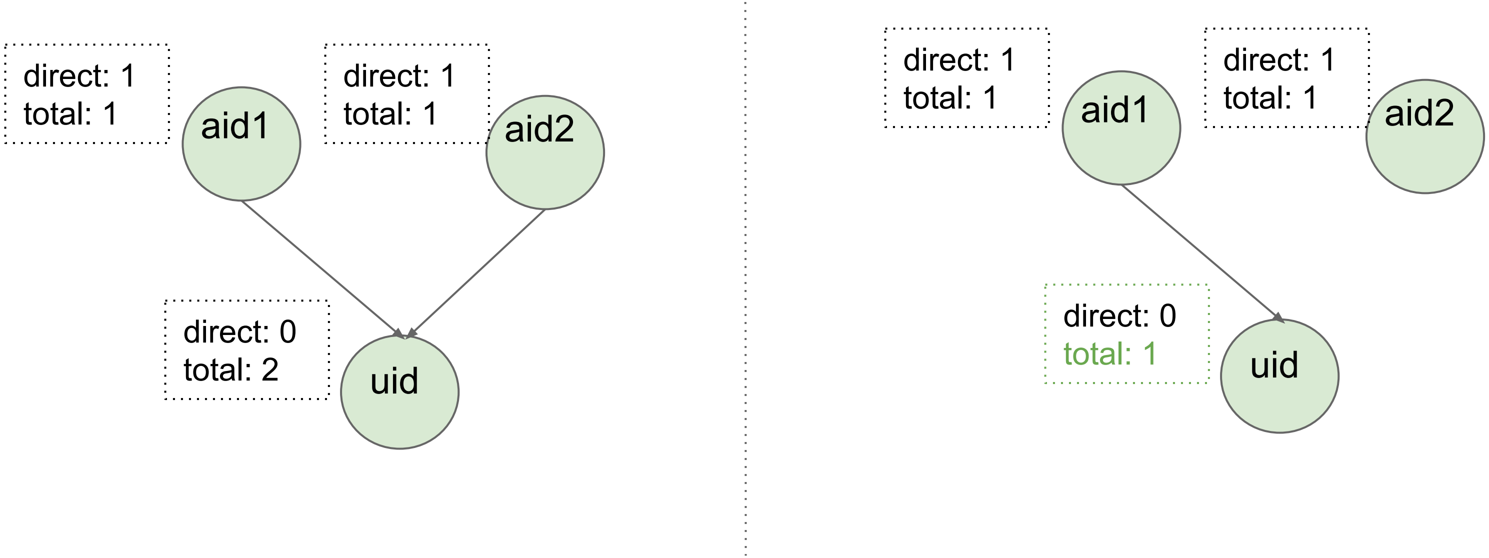
Dynamic Hierarchical Aggregation

Increments on aid1 and aid2 should be propagated along the edges to uid

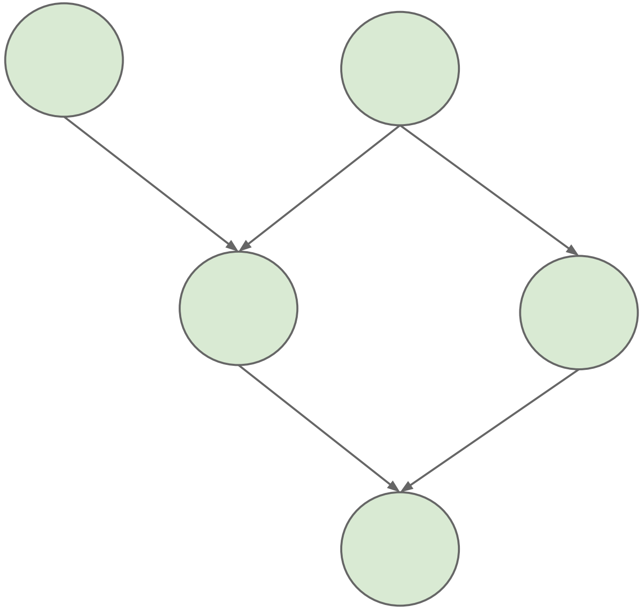


Dynamic Hierarchical Aggregation

When edge aid2 → uid is removed, we should pull back all previously propagated counts



Arbitrary DAGs



1. Design Constraints
2. **Update Algorithm**
3. Macro Architecture
4. Towards Horizontal Scalability
5. Alternatives

Lazy Vs Eager Updates

Lazy Updates

- We don’t do anything when the graph changes.
- At query time, we find out all the ancestors of a counter and return the sum of their values.

Eager Updates

- When the graph changes, we change the counter information right away.
- At query time, we don’t have to read the graph -- can just return the value of the counter.

We Need Eager Updates

- Doing a lot of processing at query time isn’t acceptable for latency.

- However it’s okay if the writes take longer, they are aysnc after all.

- Graph updates are extremely rare compared to queries.

Update Algorithm V1

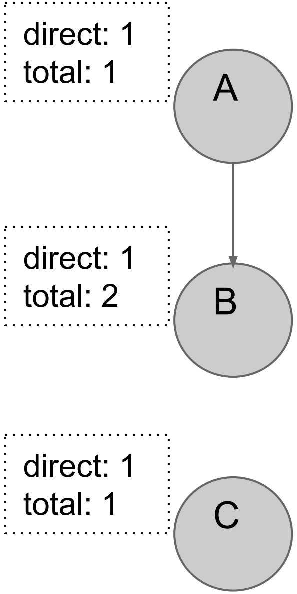
Increment (u, delta)

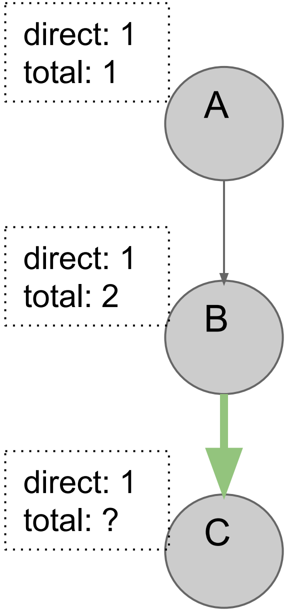
- Increment counter u by delta.
- Read descendents of u from the graph.
- Increment all these descendents by delta as well.

Add Edge (u, v)

- Read all direct counts of u.
- Read all descendants of v for the graph.
- Add direct counts of u to all descendents of v.
- Add edge u -> v in the graph.

Update Algorithm V1 On Multi-Level Graphs

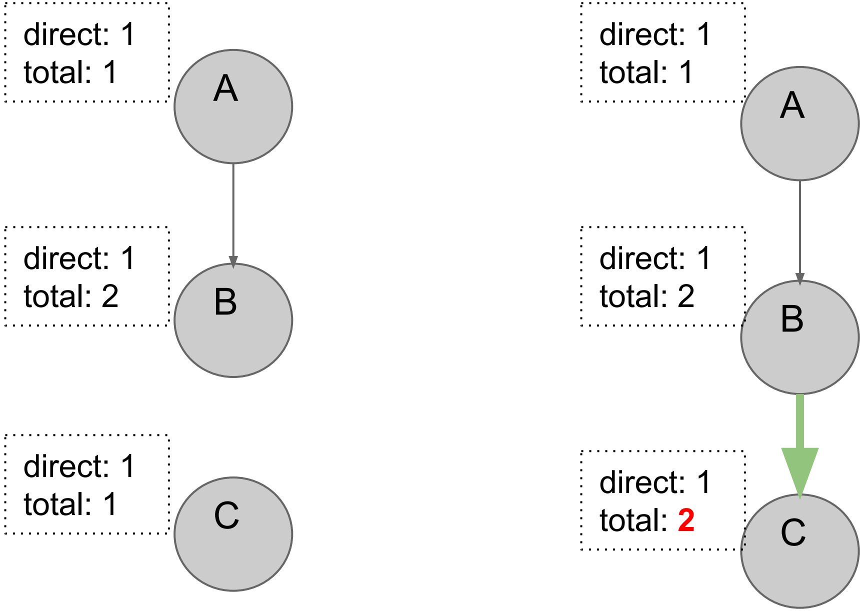




Add Edge (u, v)

- Read all direct counts of u.
- Read all descendants of v for the graph.
- Add direct counts of u to all descendents of v.
- Add edge u -> v in the graph.

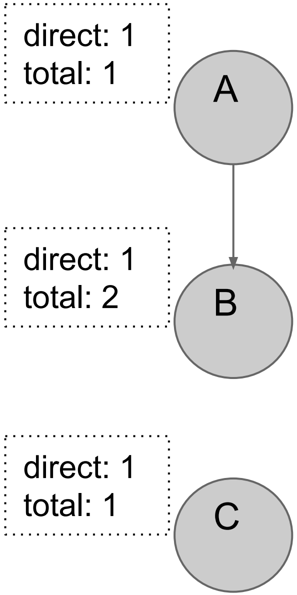
Update Algorithm V1 Fails On Multi-Level Graphs

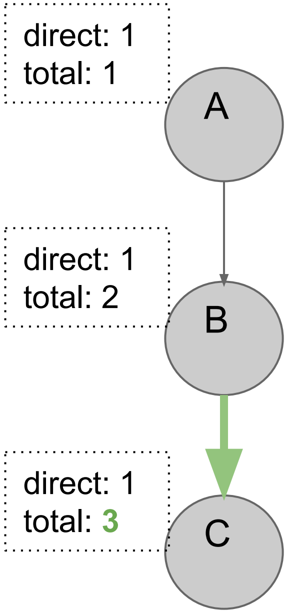


Add Edge (u, v)

- Read all direct counts of u.
- Read all descendants of v for the graph.
- Add direct counts of u to all descendents of v.
- Add edge u -> v in the graph.

Update Algorithm V2

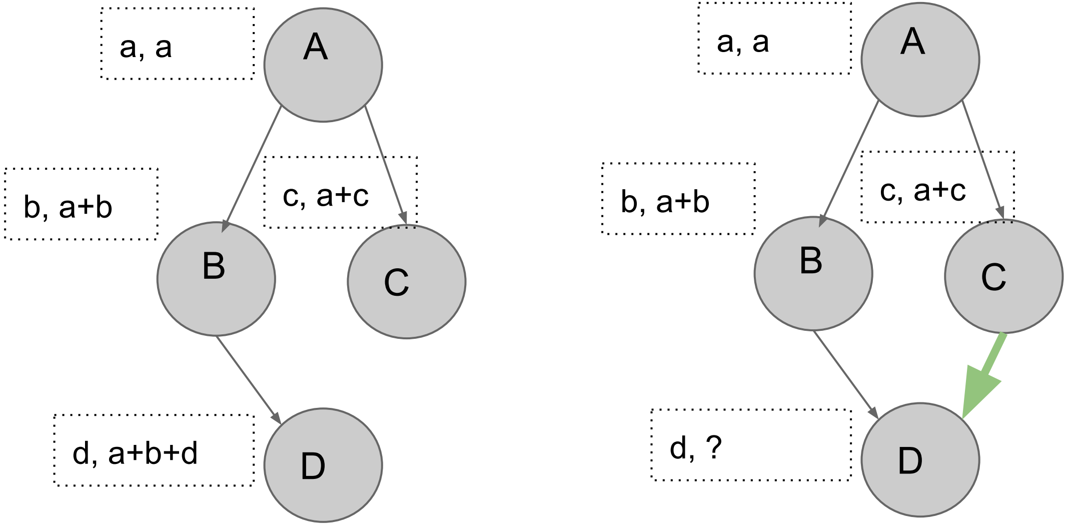




Add Edge (u, v)

- Read all direct counts of **all ancestors of** u.
- Read all descendants of v for the graph.
- Add direct counts of **ancestors of u** to all descendents of v.
- Add edge u -> v in the graph.

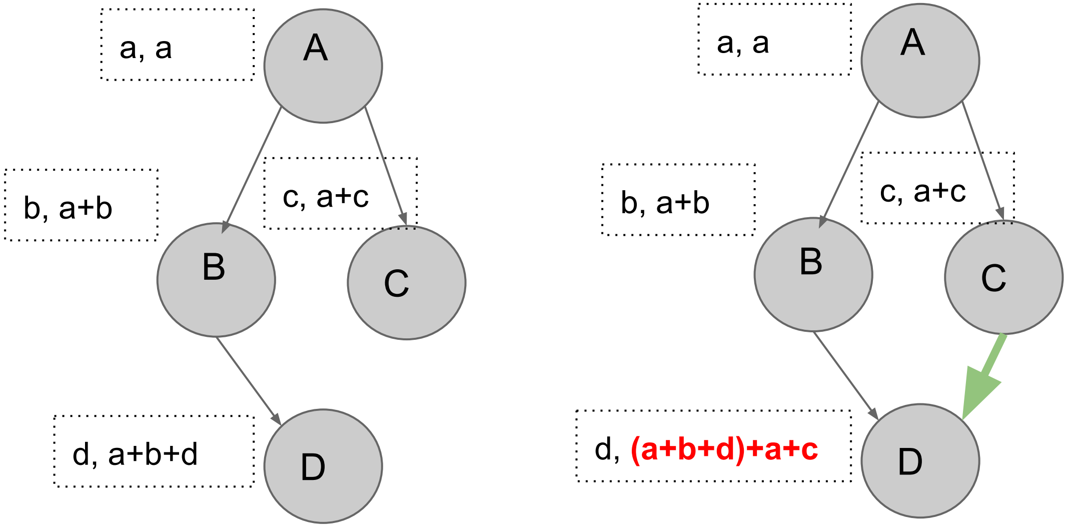
Update Algorithm V2 Fails On “Diamonds”



Add Edge (u, v)

- Read all direct counts of all ancestors of u.
- Read all descendants of v for the graph.
- Add direct counts of ancestors of u to all descendents of v.
- Add edge u -> v in the graph.

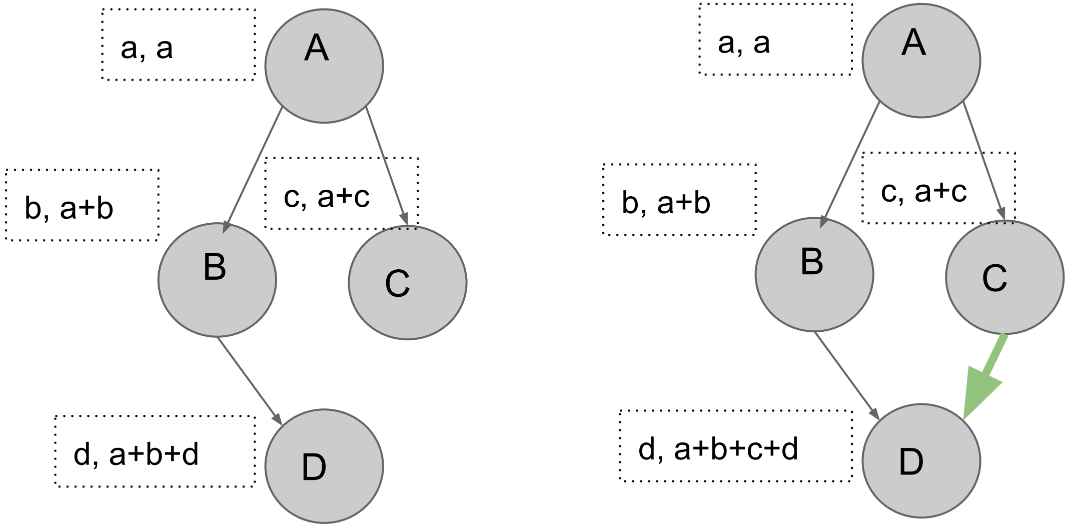
Update Algorithm V2 Fails On “Diamonds”



Add Edge (u, v)

- Read all direct counts of all ancestors of u.
- Read all descendants of v for the graph.
- Add direct counts of ancestors of u to all descendents of v.
- Add edge u -> v in the graph.

Update Algorithm V3



Add Edge (u, v)

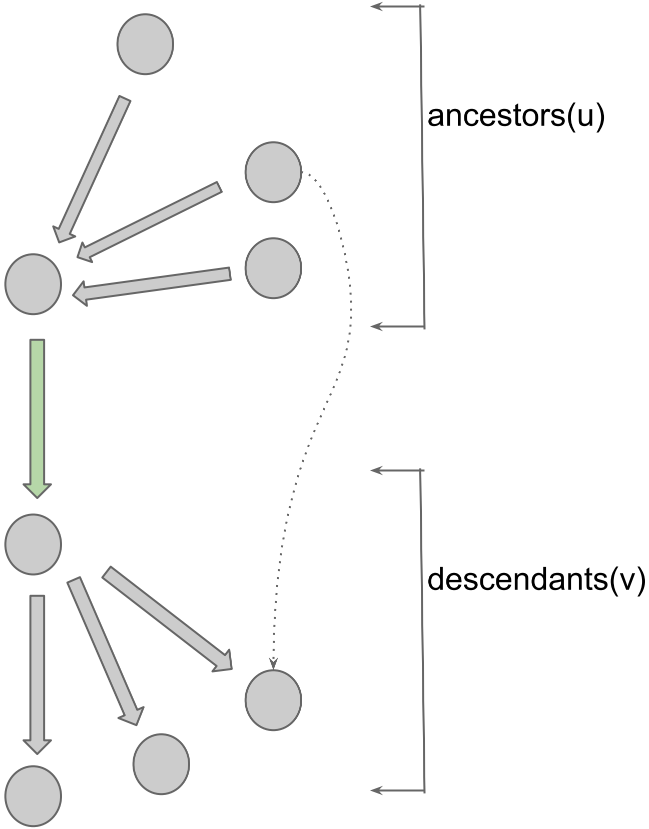
- Read all direct counts of all ancestors of u.
- Read all descendants of v for the graph.
- Add direct counts of ancestors of u to all descendents of v **if not already added.**
- ...

Final Graph Update Algorithm

Add Edge (u, v)

```
for a in ancestors(u):
    for d in descendants(v):
        if d not in descendants(a):
            total[d] += direct[a]

add_edge_in_graph(u, v)
```



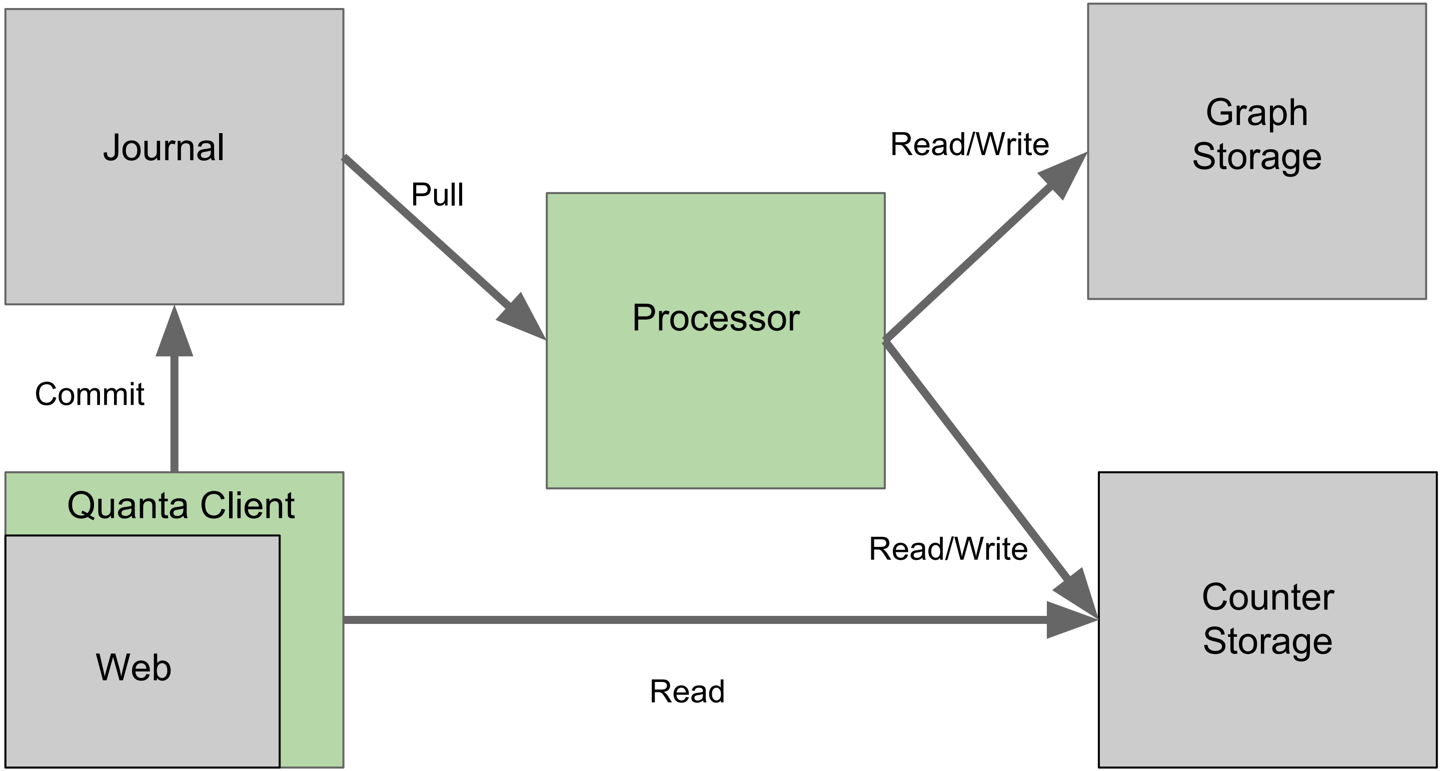
Final Graph Update Algorithm: Ordering

- Commutative with increments → can process increments and graph updates separately

- Can reorder graph updates as long as the final graph structure is correct.

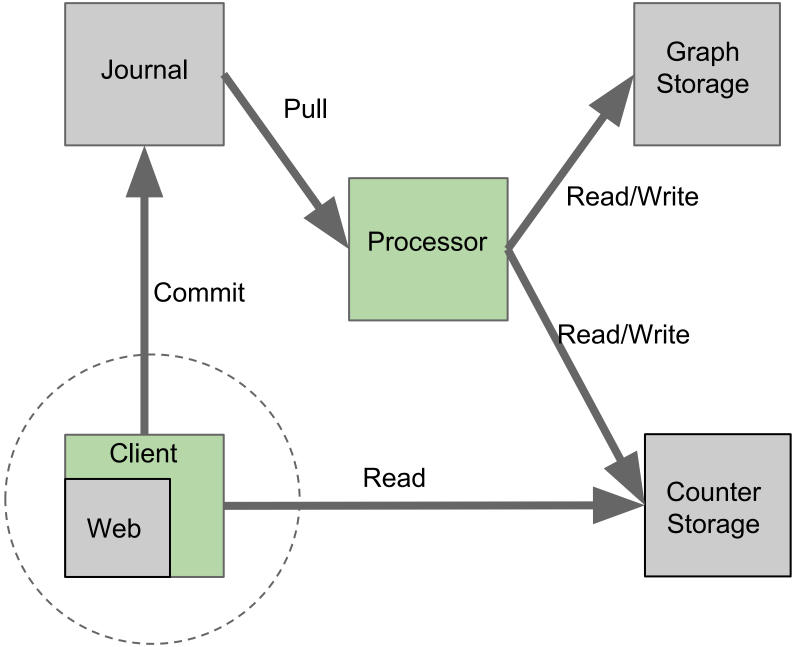
- Can run the graph algorithm in a batched mode

1. Design Constraints
2. Update Algorithm
3. **Macro Architecture**
4. Towards Horizontal Scalability
5. Alternatives


Journal
Graph Storage
Read/Write
Pull
Processor
Commit
Quanta Client
Web
Read/Write
Counter Storage
Read

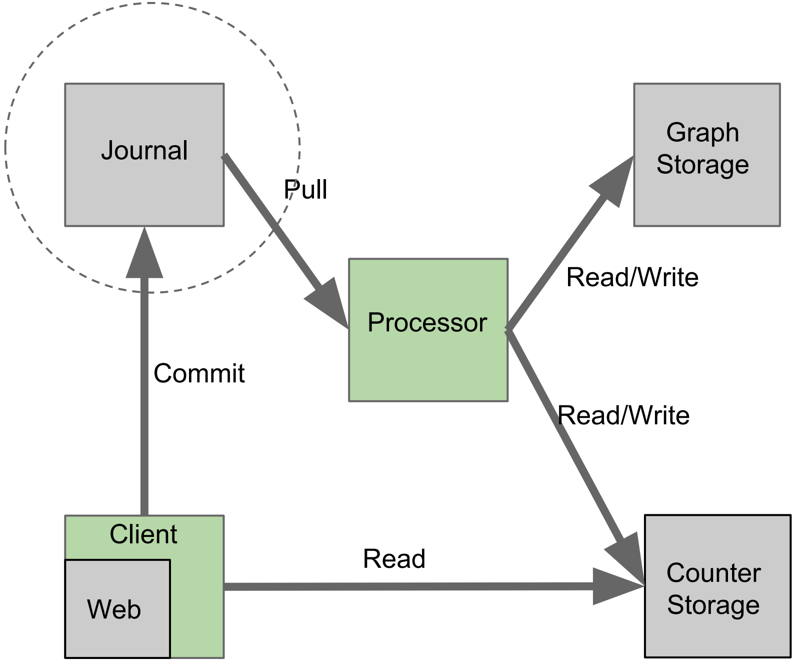
Quanta Client

- Very thin client, knows how to talk to Journal and Counter storage.
- Commits updates to Journal.
- All updates are thrift messages, so clients can be written in any language.
- Reads count data directly from ‘Counter Storage’



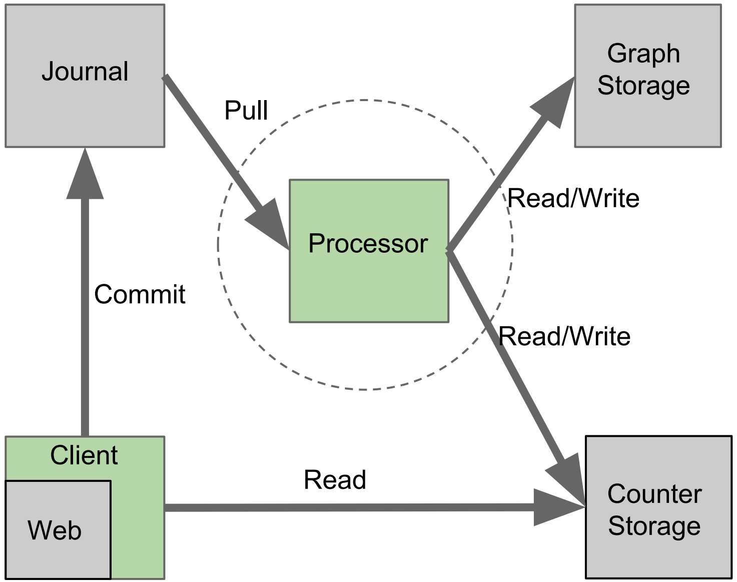
Journal

- Just a replicated persistent commit log. Replication makes writes available.
- Stores updates temporarily till processor pulls them to process
- At-least-once processing semantics.
- Any persistent commit log can work as Journal. Currently using Scribe but soon moving to Kafka.



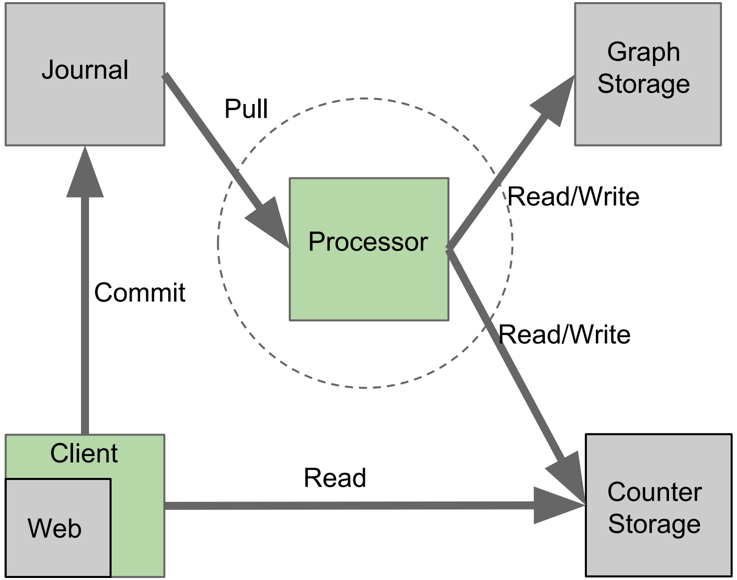
Processor

- Reads updates from Journal, and processes them in batches by reading/updating graph/counter storage.
- Graph updates also processed in batches -- two rounds of BFS to "download" graph
- Processing updates asynchronously in batches is the single most important idea in this architecture.



Processor: Batched Asynchronous Updates

- Processor becomes stateless → don't have to worry about its availability.
- Can easily absorb spikes
- Reduce volume by deduplicating increments
- Can batch IO requests.



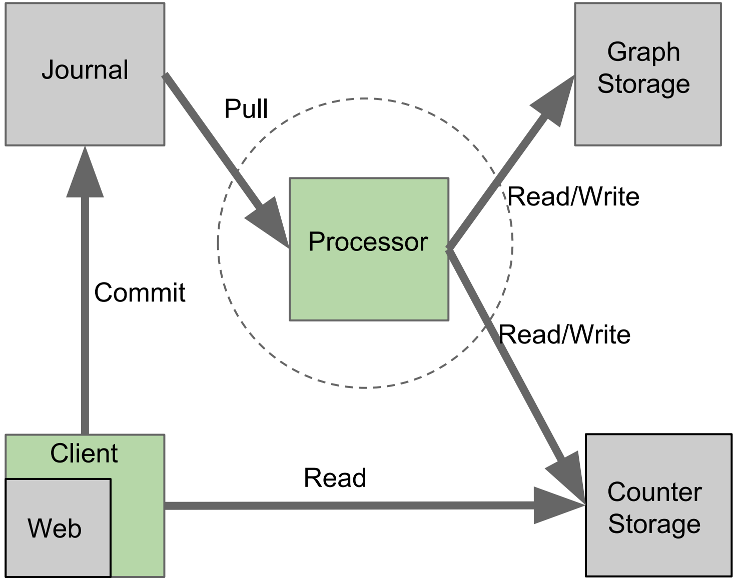
Processor: Batched Asynchronous Updates

- Does some optimizations based on the graph structure

- Currently implemented as system of stateless minutely crons, on top of our distributed cron architecture

- Lag of few minutes comes from implementing on top of minutely crons

- Can reduce lag by processing data in smaller batches



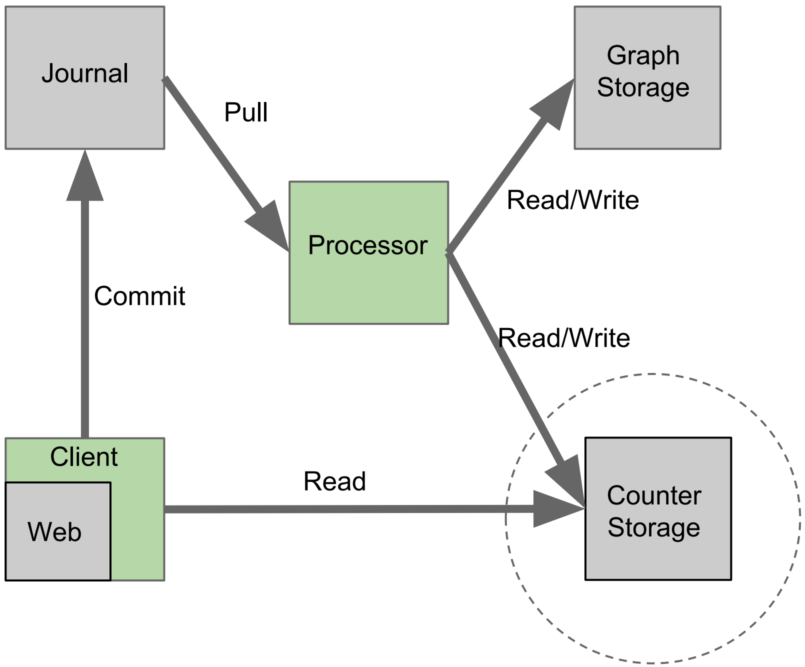
Counter Storage

- Implemented on top of HBase.

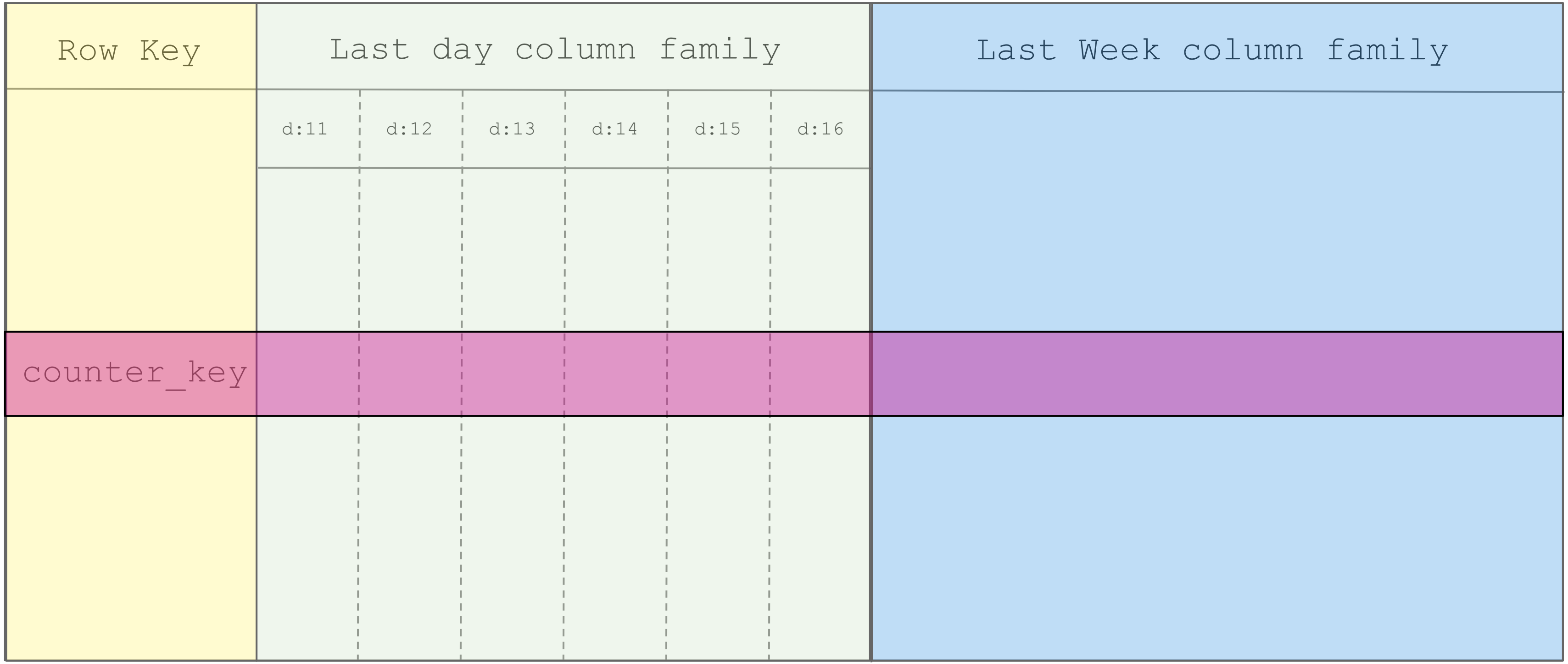
- Each row denotes a single counter.

- Column family corresponds to a time window (e.g "last week")

- Columns in this column family correspond to time buckets (either rolling or static).



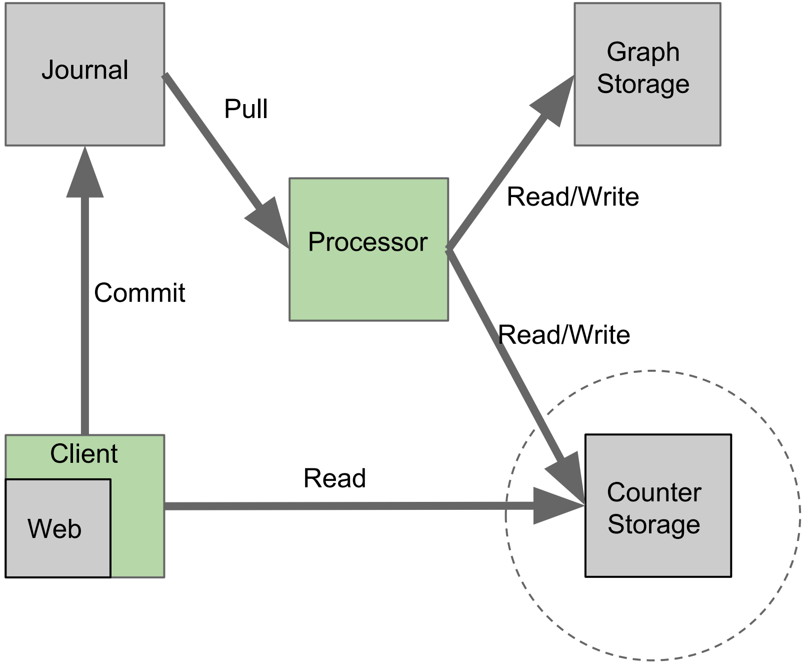
Counter Storage - HBase Schema



Row Key	Last day column family						Last Week column family
	d:11	d:12	d:13	d:14	d:15	d:16	
counter_key							

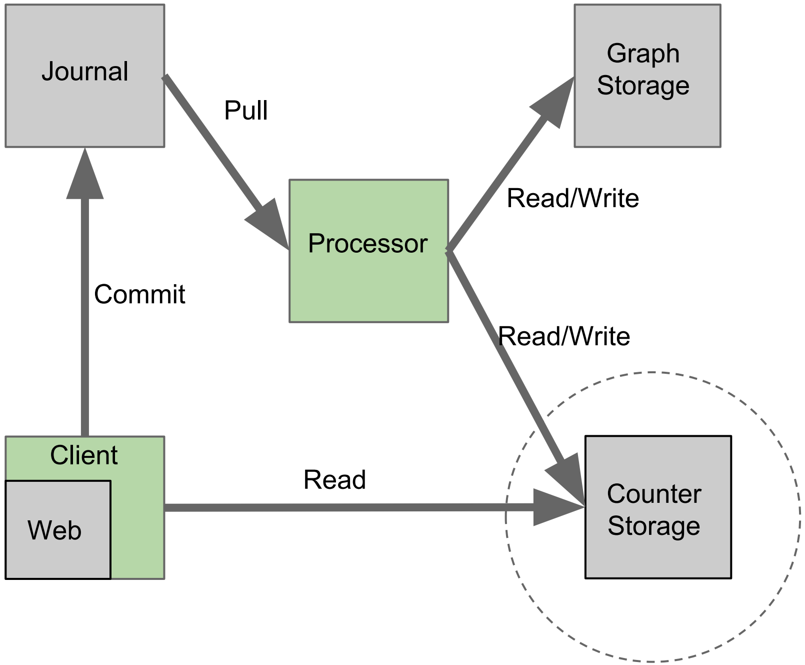
Counter Storage: Why HBase

- Hbase supports high write throughput
- Allows range scans -- can get data for related counters in a sequential disk read
- Supports variable number of columns -- great for both time-series and rolling windows
- Supports automatic garbage collection of old data by setting TTL per column family.



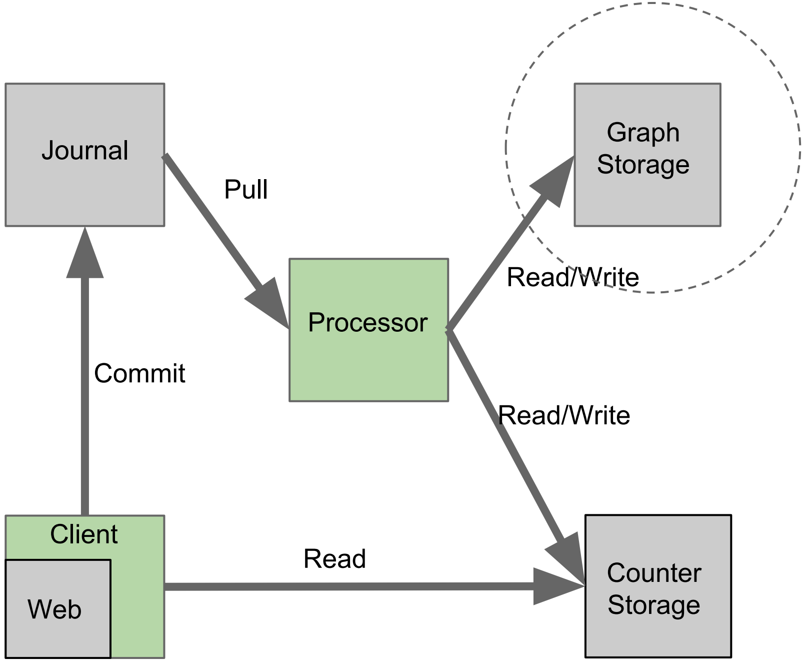
Counter Storage: Why HBase

- Bulk upload is very useful to backfill data into Quanta
- High availability of Quanta reads (due to HBase availability from multi-way replication)



Graph Storage

- Built a general purpose graph datastore called Graphiq on top of HBase.
- Data stored as sparse adjacency matrix.
- Schema -- 2 column families, one for incoming edges, and one for outgoing edges
- HBase can serve negative queries through Bloom Filters → good fit for graph traversals



1. Design Constraints
2. Update Algorithm
3. Macro Architecture
4. **Towards Horizontal Scalability**
5. Alternatives

Sharding Quanta Processor

- Must shard processor to reach the desired throughput

- But sharding isn’t easy because there can be be race conditions between graph updates and counter updates.

Revisiting The Algorithm

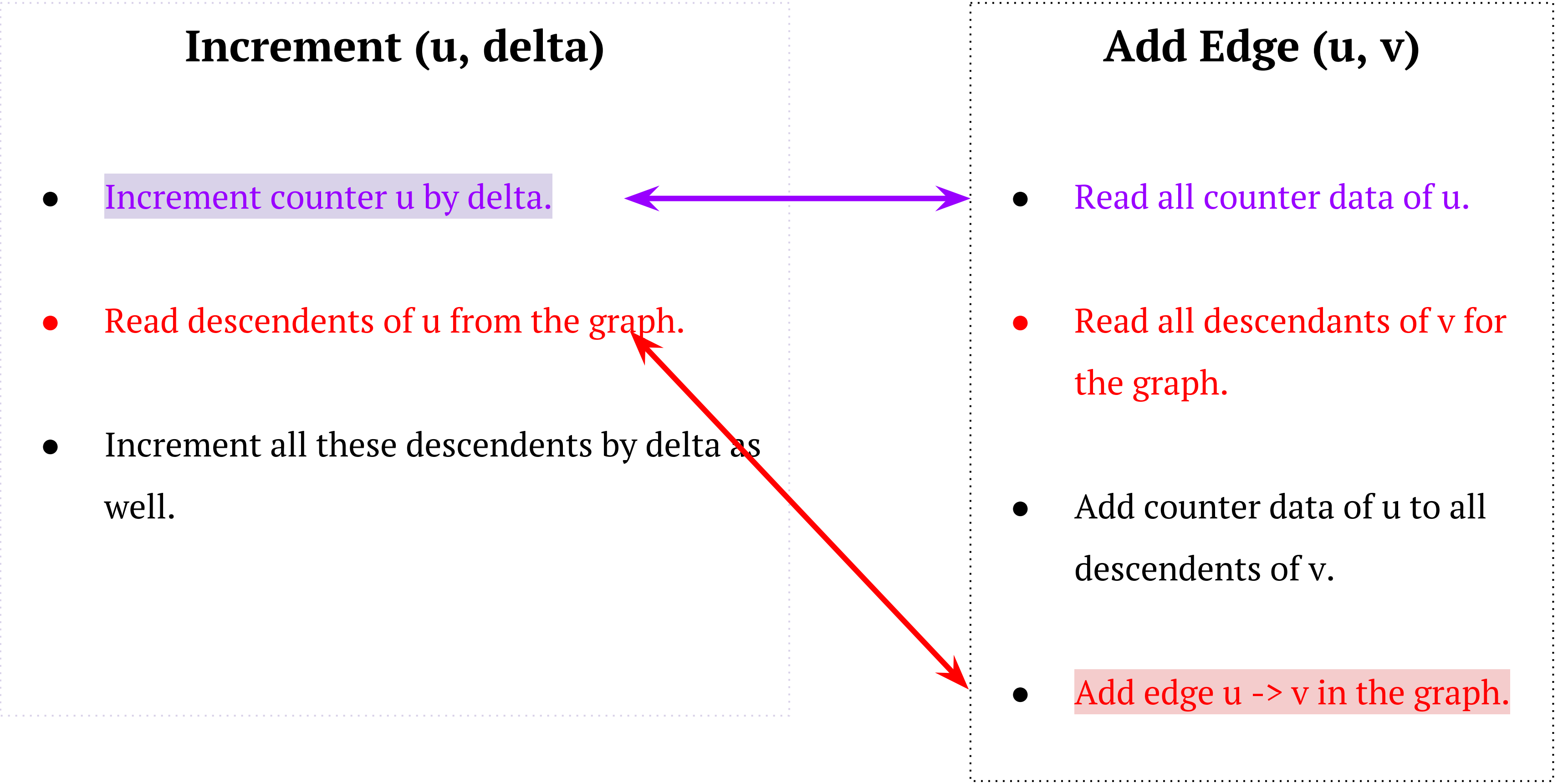
Increment (u, delta)

- Increment counter u by delta.
- Read descendents of u from the graph.
- Increment all these descendents by delta as well.

Add Edge (u, v)

- Read all counter data of u.
- Read all descendants of v for the graph.
- Add counter data of u to all descendents of v.
- Add edge u -> v in the graph.

Race Conditions



Increment (u, delta)

- Increment counter u by delta.
- Read descendents of u from the graph.
- Increment all these descendents by delta as well.

Add Edge (u, v)

- Read all counter data of u.
- Read all descendants of v for the graph.
- Add counter data of u to all descendents of v.
- Add edge u -> v in the graph.

More Observations

- Counter updates don’t create a race condition with any other counter updates (assuming increments are “atomic”)

- Volume of counter updates is 2-3 orders of magnitude higher than graph updates.

- Sharding Solution --
  - Partition processor in two independent parts -- one for graph and other for counter
  - Shard counter processor in as many shards as required
  - Synchronize all processors using an instance wide shared/exclusive lock

Design 1: Sharding With Shared/Exclusive Locks

Design -- single graph processor, sharded counter processor, shared/exclusive lock to synchronize

- Graph update volume is low → could scale to 20x on this design.
- Lock starvation for graph processor as ‘counter’ shards grow in the number
- Would ideally prefer to eliminate locks completely.

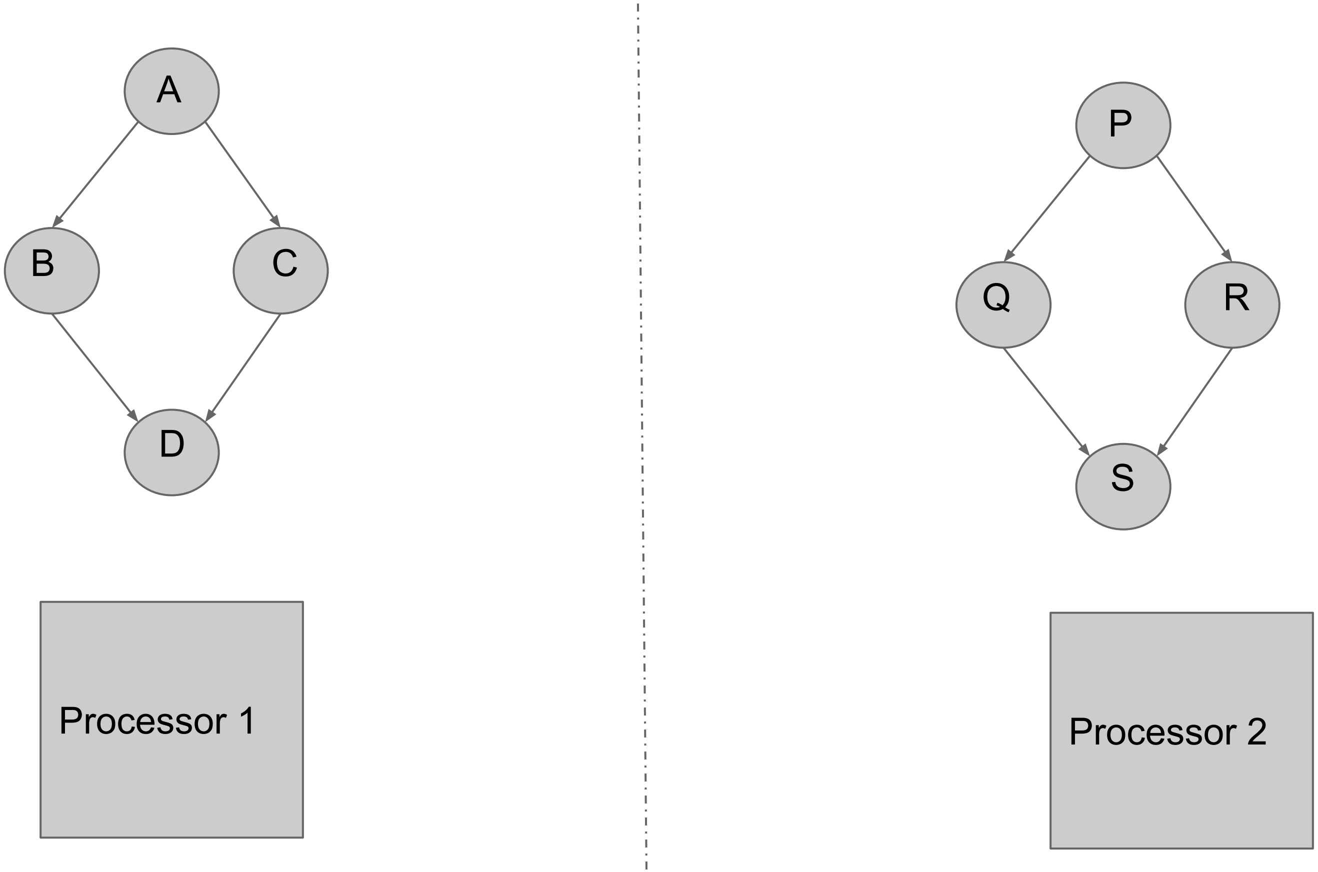
Sharding Quanta Processor With Shared/Exclusive Locks

Design -- single graph processor, sharded counter processor, shared/exclusive lock to synchronize

- Graph update volume is low → could scale to 20x on this design.
- Lock starvation for graph processor as ‘counter’ shards grow in the number
- Would ideally prefer to eliminate locks completely.

Decided to find ways of sharding without these problems.

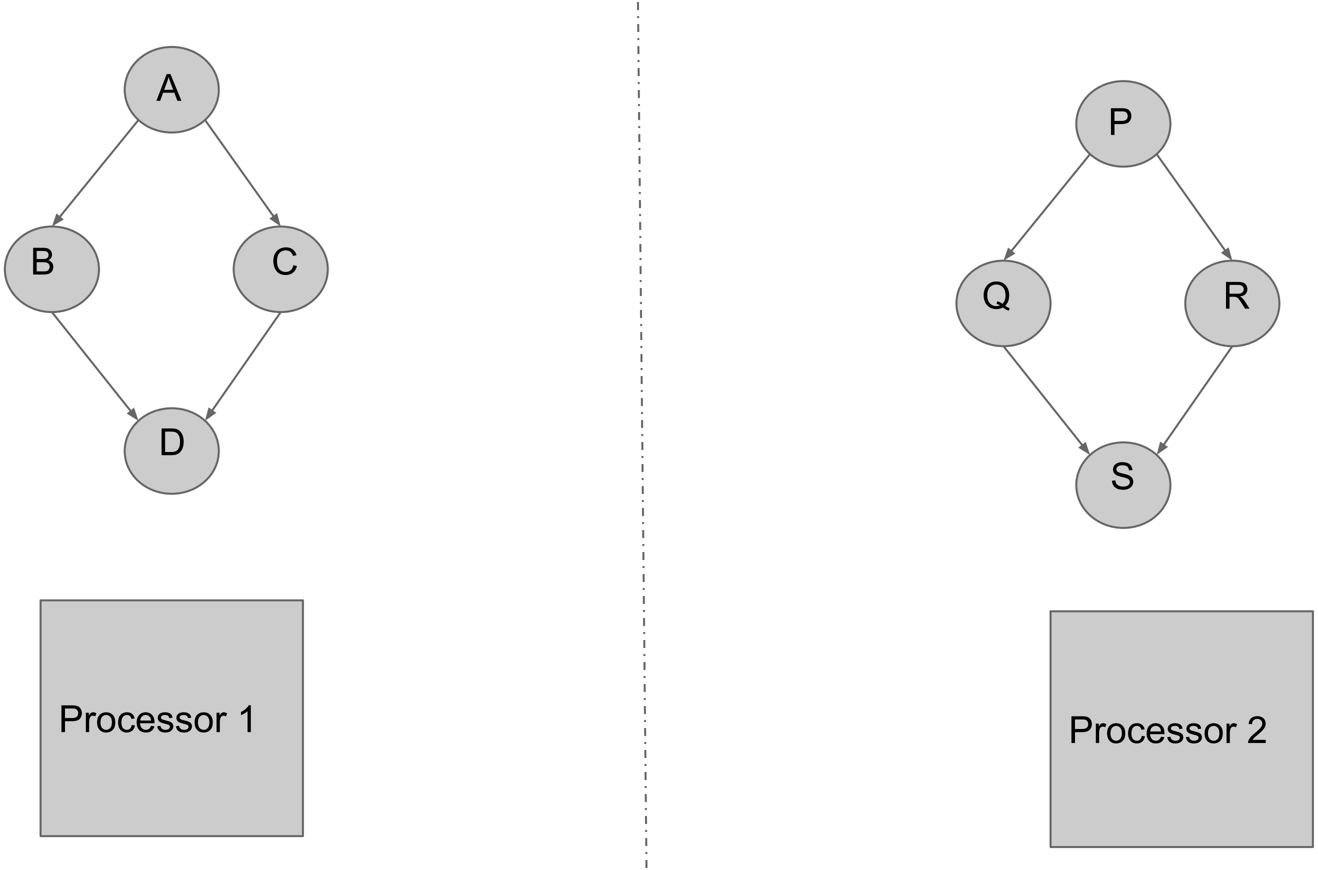
Observation: Connected Components Are Isolated



Processor 1

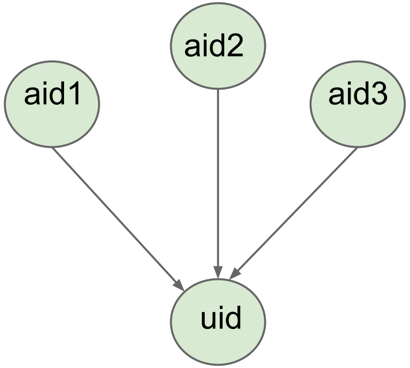
Processor 2

Co-sharding counter/graph on connected components



Design 2: Co-Sharding Counter/Graph Processor

- Application owners define a shard key (by overriding a function) for their graphs.
- We can now run counter/graph updates of a shared sequentially and eliminate locks.
- However, connected components is a much stronger condition than required.



Design 3: Out Of Box Co-Sharding

- Can provide out-of-box support for co-sharding without users telling us the shard key.

- Key idea -- propagating information in the graph level-by-level.

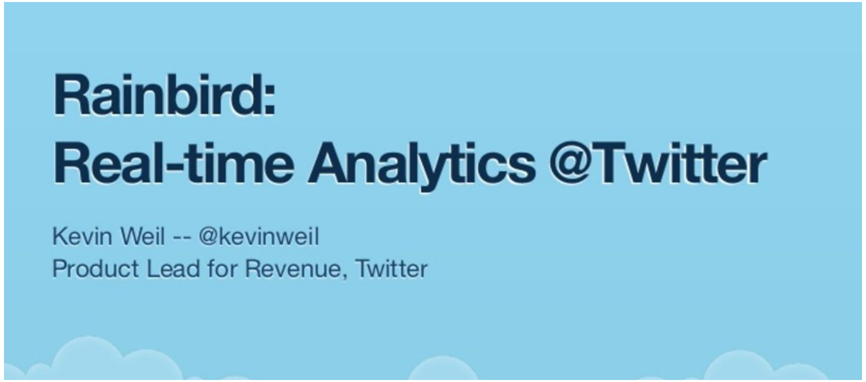
- Will free up Quanta from closely managing partitioning and delegate it to someone else. Could then easily move onto Spark.

- Not yet implemented, still on our TODO list.

1. Design Constraints
2. Update Algorithm
3. Macro Architecture
4. Towards Horizontal Scalability
5. **Alternatives**

Alternatives: Twitter’s RainBird

- Distributed counting with aggregation and temporal bucketing -- similar to Quanta.
- No dynamic hierarchical aggregation, only static
- Latency guarantees at ~100ms (Quanta is ~1ms)
- Built atop Cassandra

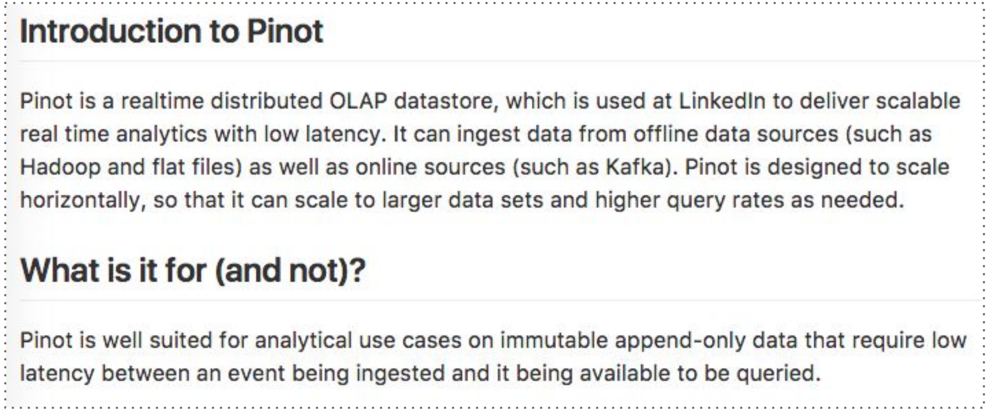


Rainbird:
Real-time Analytics @Twitter

Kevin Weil -- @kevinweil
Product Lead for Revenue, Twitter

Alternatives: LinkedIn’s Pinot

- Powers LinkedIn’s product features that are related to profile views.
- SQL-like language support -- aggregation, filtering, group by etc.
- Lots of moving parts & configuration state -- high operational cost.



Introduction to Pinot

Pinot is a realtime distributed OLAP datastore, which is used at LinkedIn to deliver scalable real time analytics with low latency. It can ingest data from offline data sources (such as Hadoop and flat files) as well as online sources (such as Kafka). Pinot is designed to scale horizontally, so that it can scale to larger data sets and higher query rates as needed.

What is it for (and not)?

Pinot is well suited for analytical use cases on immutable append-only data that require low latency between an event being ingested and it being available to be queried.

Alternatives: Facebook's Insights System

- Almost identical to Quanta -- does similar things, built atop HBase, Scribe etc..

- Except, doesn't support dynamic hierarchical aggregation, only static

- Many of our design decisions were inspired by this system.

Building Realtime Insights

By Alex Himel on Tuesday, March 15, 2011 at 1:38pm

Social plugins have become an important and growing source of traffic for millions of websites over the past year. We released a new version of Insights for Websites last week to give site owners better analytics on how people interact with their content and to help them optimize their websites in real time.

Summary

- Quanta is a service built on top of HBase to do counting on dynamic graphs.
- Supports defining arbitrary time windows and configurable rolling error.
- Super low latency: 1ms.
- Currently doing 1 Billion updates/reads everyday. Built to scale to 20-50x.
- Powering all ads reporting as well as features based on content views. Many more users lined up.

Thank you!

Nikhil Garg
@nikhilgarg28

Chun-Ho Hung
@chhung6