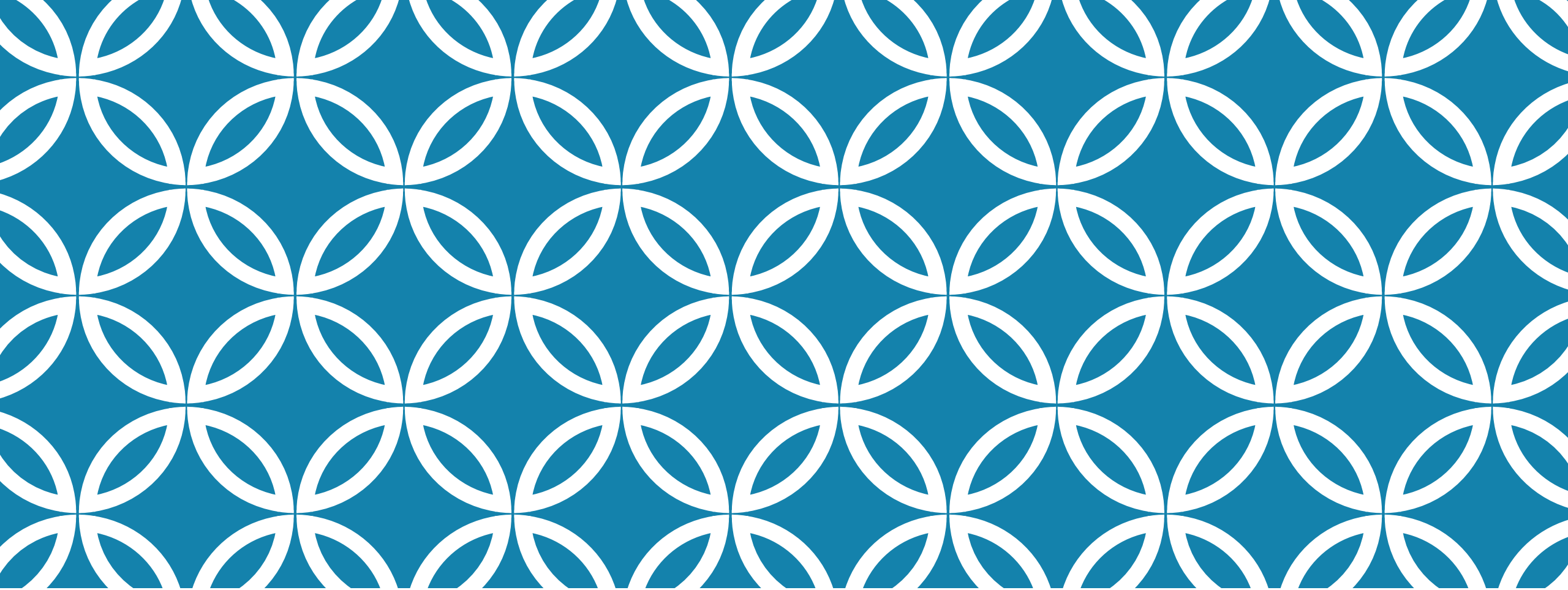


# CS5412 / LECTURE 24 HOW FACEBOOK REPRESENTS SOCIAL NETWORKING DATA

Ken Birman

Spring, 2021

# TODAY… A "BIG DATA" TOPIC

The last few lectures have looked at computing on sharded big data. But not all big data is actually stored in this form.

Today, we will look at the way that Facebook creates and manages its TAO database for the social networking graph. This is a central form of big data in Facebook, and most cloud platforms have a similar system.

IoT will cause us to create new kinds of social network graphs (for example to track Covid spread). This is a huge opportunity area for researchers.

# WHAT IS "BIG DATA" USUALLY ABOUT?

Early in the cloud era, research at companies like Google and Amazon made it clear that people respond well to social networking tools and smarter advertising placement and recommendations.

The idea is simple: "People with Ken's interest find this store fantastic." "Anne really like Eileen Fisher and might want to know about this 15% off sale on spring clothing." "Sarah had a flat tire and needs new ones."

# THEY HAD A LOT OF CUSTOMERS AND DATA

Web search and product search tools needed to deal with billions of web pages and hundreds of millions of products.

Billions of people use these modern platforms.

So simple ideas still involve enormous data objects that simply can't fit in memory on modern machines. And yet in-memory computing is far faster than any form of disk-based storage and computing!

# WHAT ARE THE BIG DATA FILES?

Snapshot of all the web pages in the world, updated daily.

Current product data & price for every product Amazon knows about.

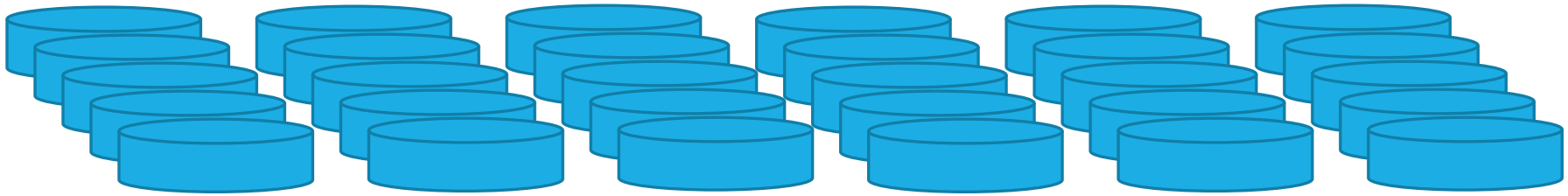
Social networking graph for all of Facebook

# MANY CHALLENGES

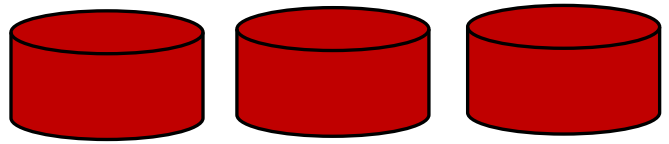
XenonStack.com

# VISUALIZING THE PIPELINE

Data starts out sharded over servers



Early pipeline stages are extremely parallel: they *extract, transform, summarize*



Eventually we squeeze our results into a more useful form, like a trained machine-learning model. The first stages can run for a long time before this converges

Copy the model to wherever we plan to use it.

# FOR WEB PAGES THIS IS "EASY"

The early steps mostly extract words or phrases, and summarize by doing things like counting or making lists of URLs.

The computational stages do work similar to sorting (but at a very large scale), e.g. finding the "most authoritative pages" by organizing web pages in a graph and then finding the graph nodes with highest weight for a given search.

When we create a trained machine-learning model, the output is some sort of numerical data that parameterizes a "formula" for later use (like to select ads).

# WHAT ABOUT FOR SOCIAL NETWORKS?

Here we tend to be dealing with very large and very dynamic graphs.

The approaches used involve specialized solutions that can cope with the resulting dynamic updates.

Facebook's TAO is a great example, we'll look closely at it.

facebook

# TAO

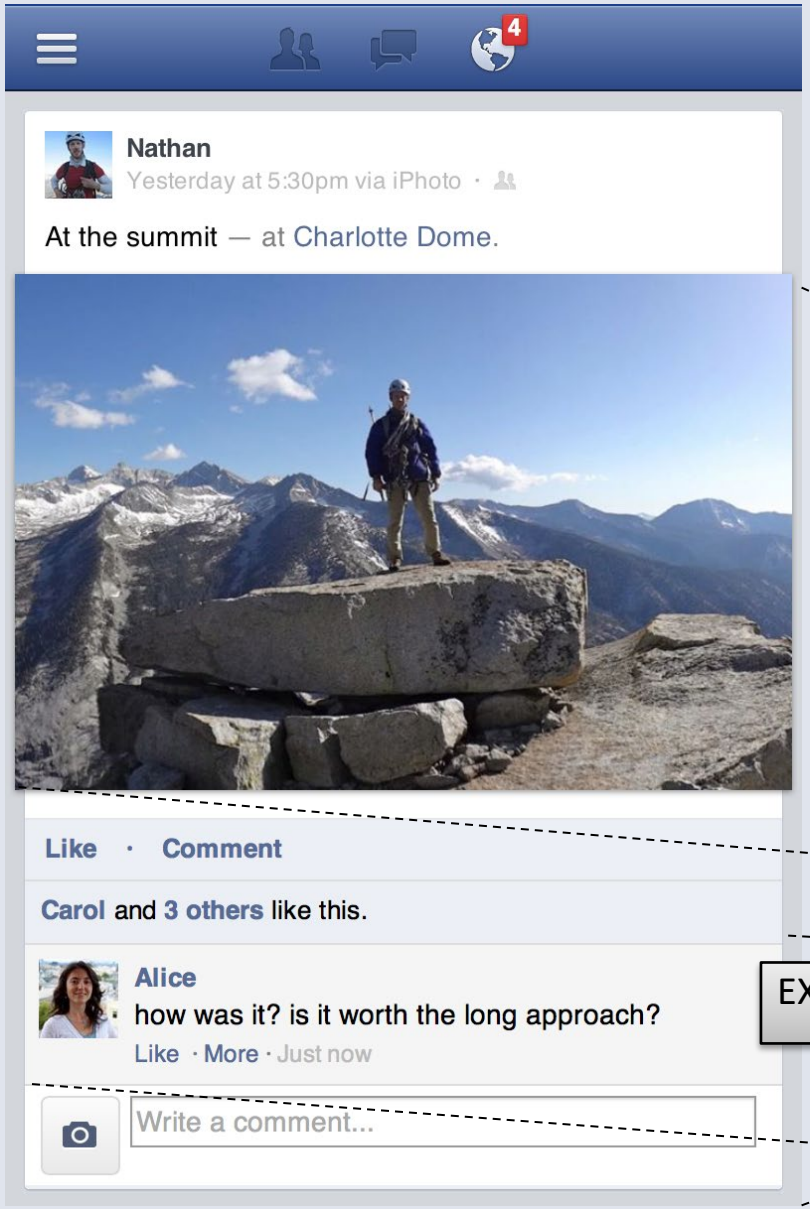
## Facebook's Distributed Data Store for the Social Graph

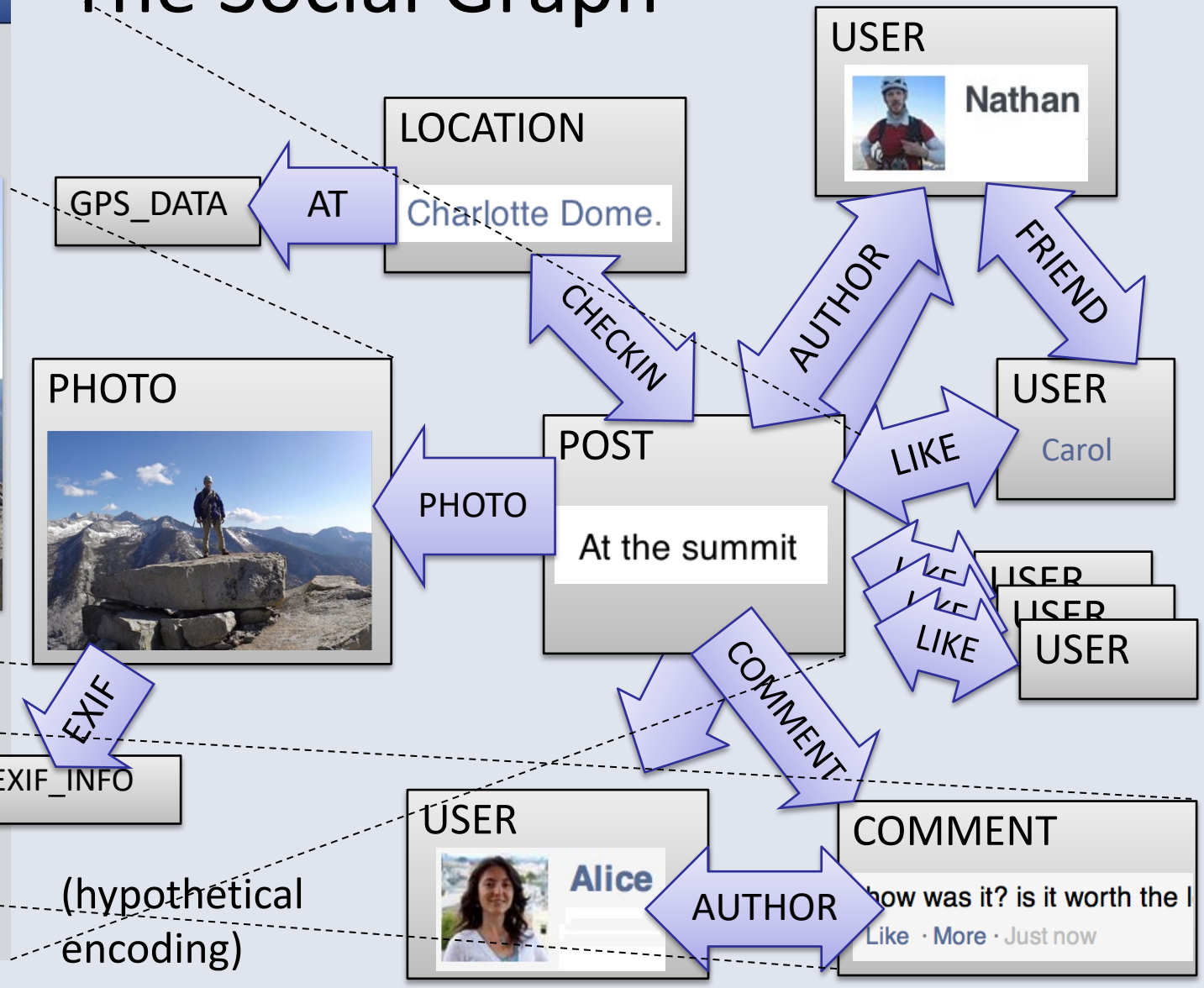
Cornell PhD who worked with Professor van Renesse. Graduated in 2010
Now one of several people with the title "Director of Engineering"
He owns the distributed systems area: the Facebook "edge"

**Nathan Bronson**, Zach Amsden, George Cabrera, Prasad Chakka, Peter Dimov, Hui Ding, Jack Ferris, Anthony Giardullo, Sachin Kulkarni, Harry Li, Mark Marchukov, Dmitri Petrov, Lovro Puzar, Yee Jiun Song, Venkat Venkataramani

Presented at USENIX ATC – June 26, 2013

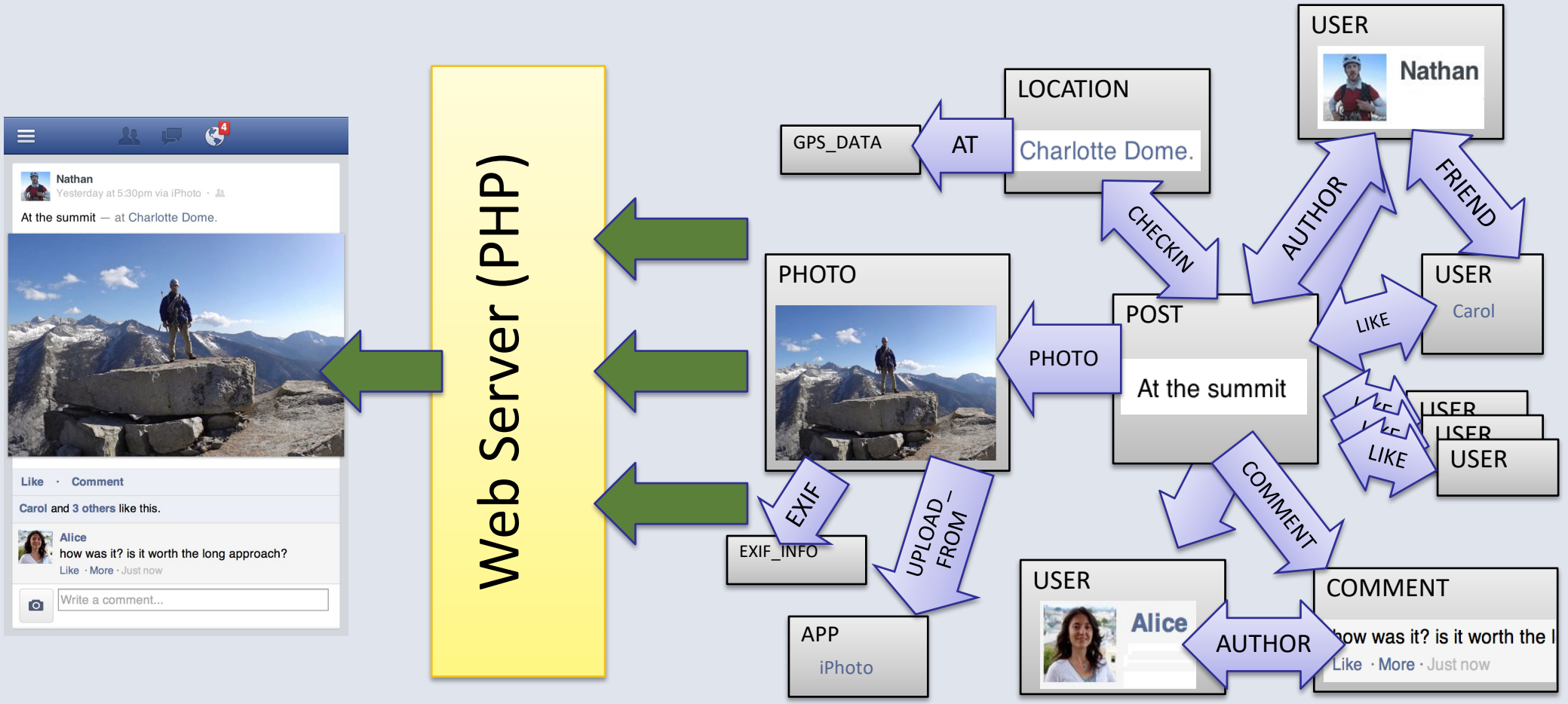
# The Social Graph



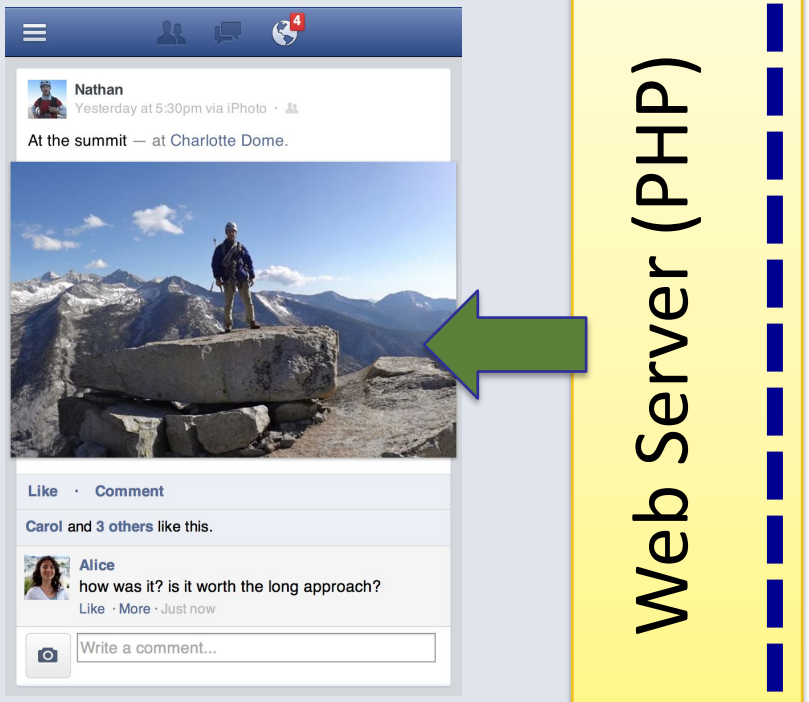




(hypothetical encoding)

# Dynamically Rendering the Graph

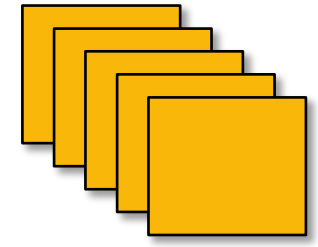


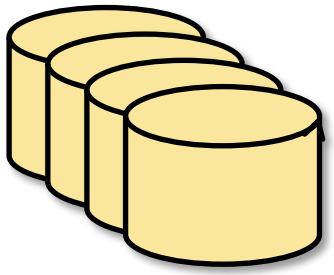
# Dynamically Rendering the Graph

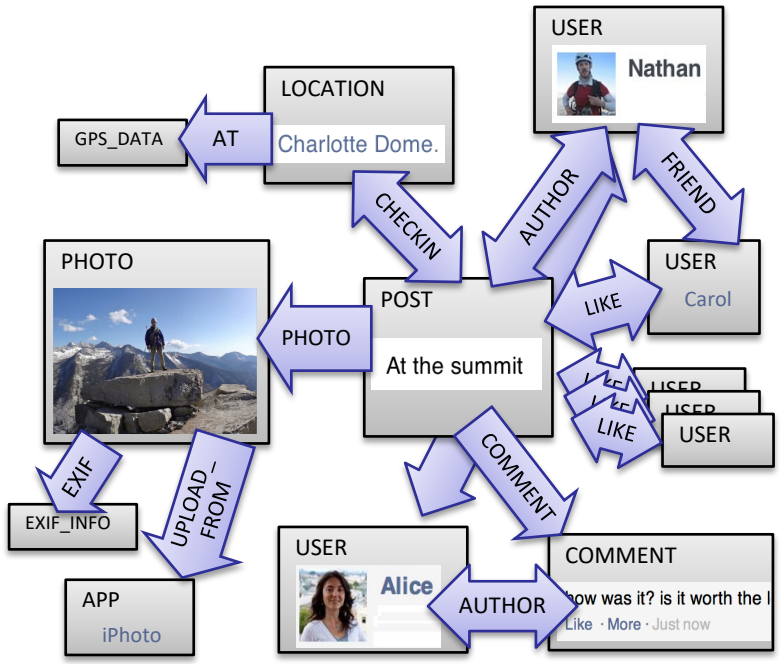



Web Server (PHP)

## TAO








- 1 billion queries/second
- many petabytes of data

# TAO opts for NoSQL Model

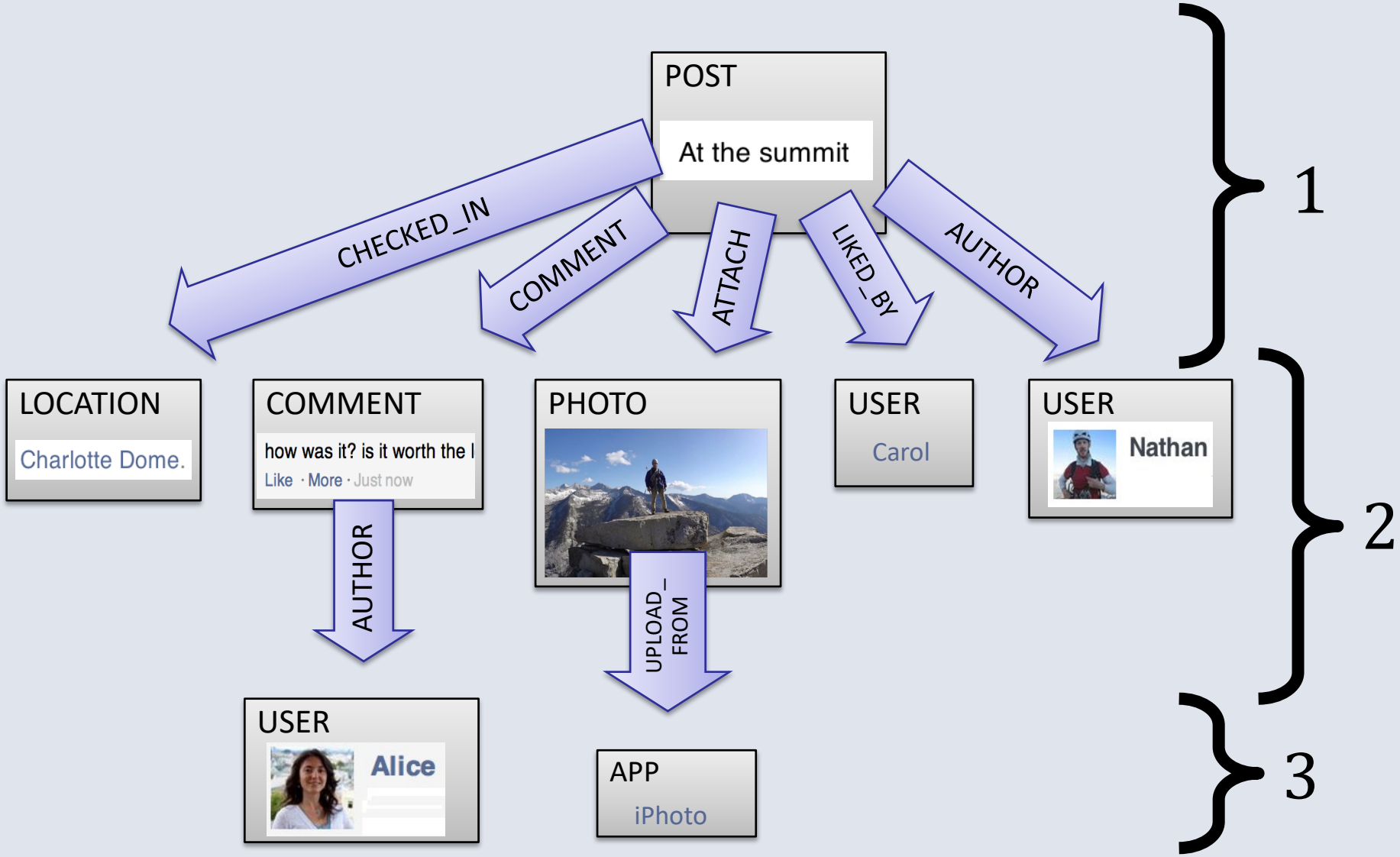
- Most TAO applications treat the graph like a very restricted form of SQL database: it looks like SQL.
- But first, they limit the operations: it isn't full SQL.
- And then they don't guarantee the ACID properties.

- In fact the *back end* of TAO actually is serializable, but it runs out of band, in a batched and high-volume way (BASE: eventually, consistency happens).
- The only edge consistency promise is that they try to avoid returning broken association lists, because applications find such situations hard to handle.

# What Are TAO's Goals/Challenges?

- Efficiency at scale

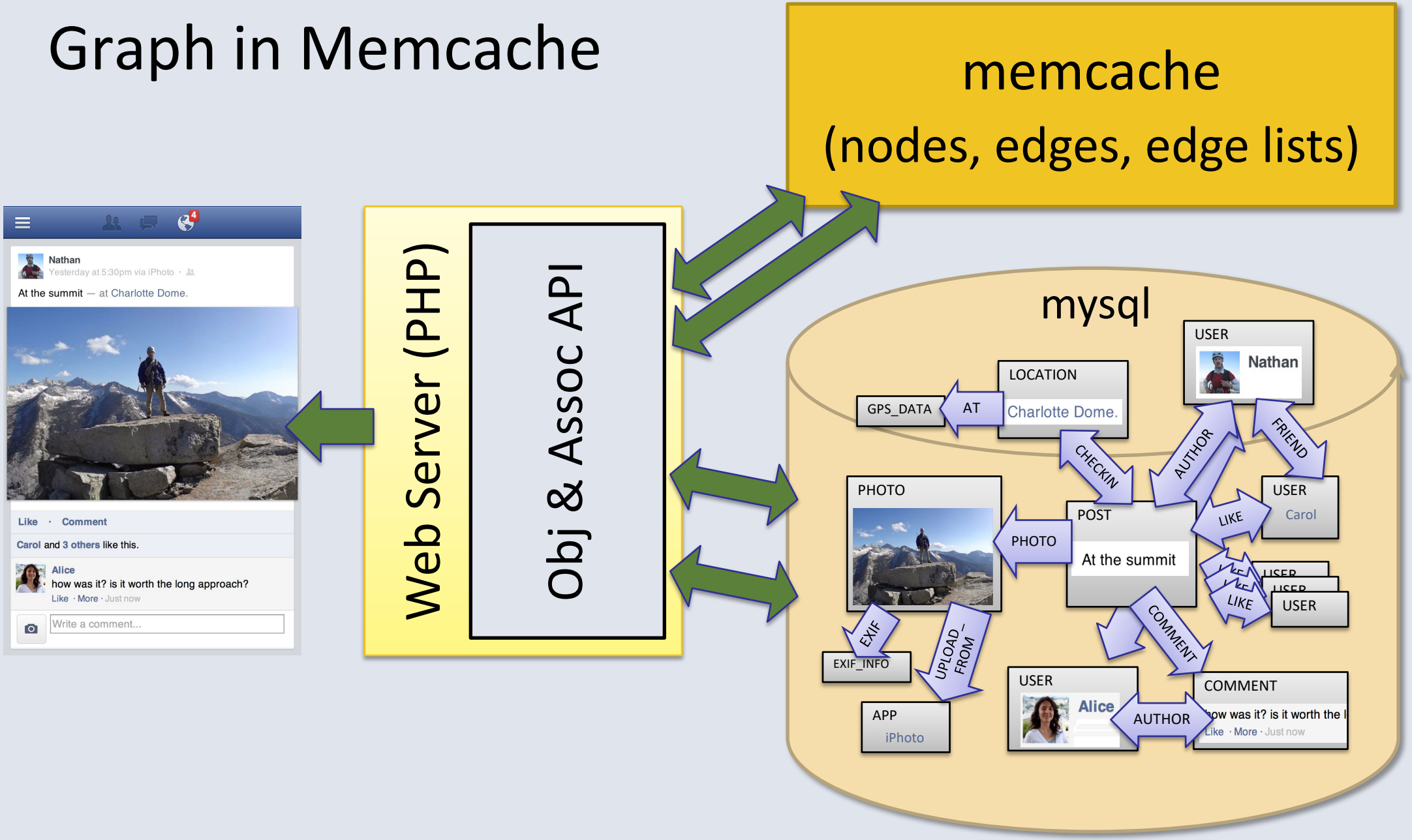
# Dynamic Resolution of Data Dependencies



# What Are TAO's Goals/Challenges?

- Efficiency at scale
- Low read latency
- Timeliness of writes
- High Read Availability

# Graph in Memcache

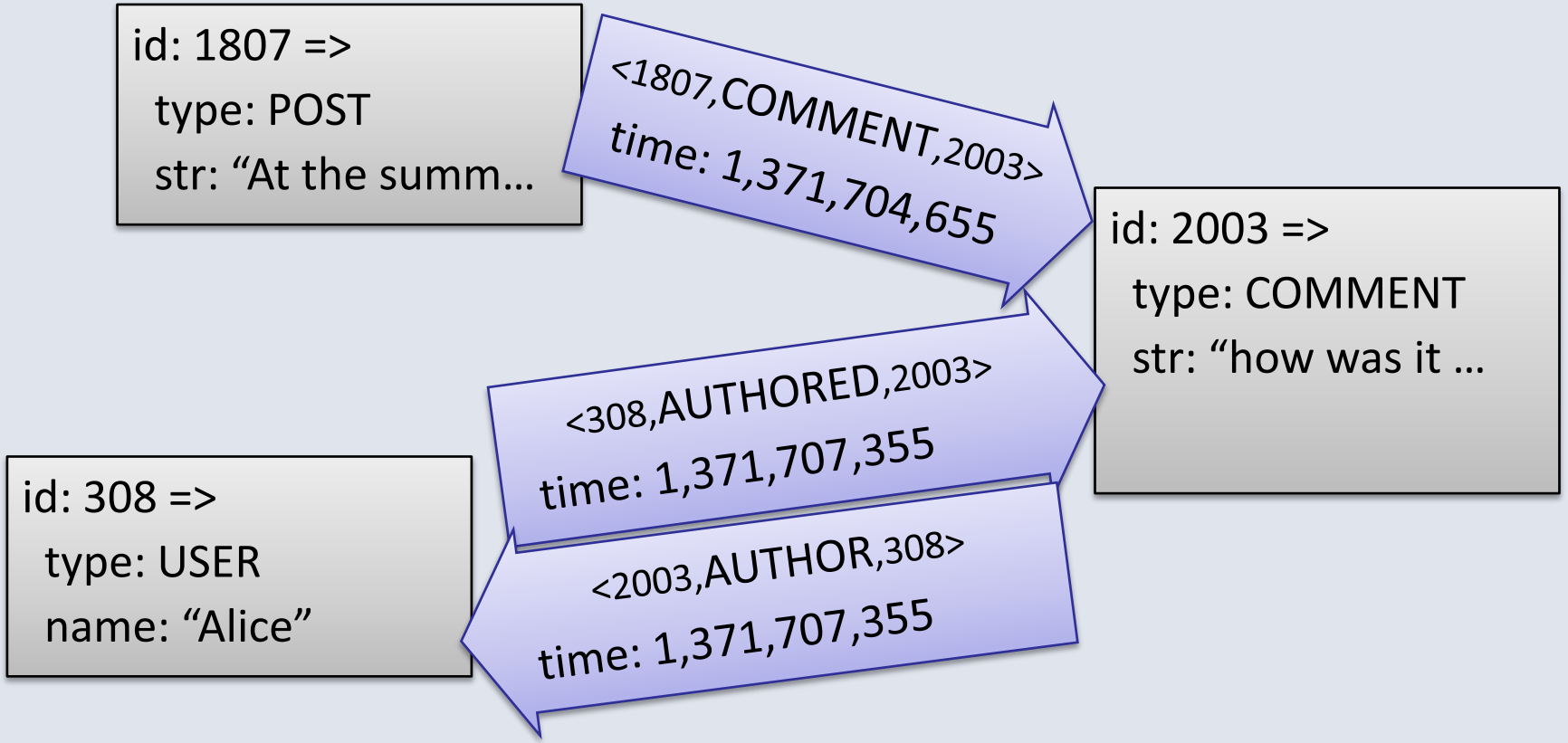


# Objects = Nodes

- Identified by unique 64-bit IDs
- Typed, with a schema for fields

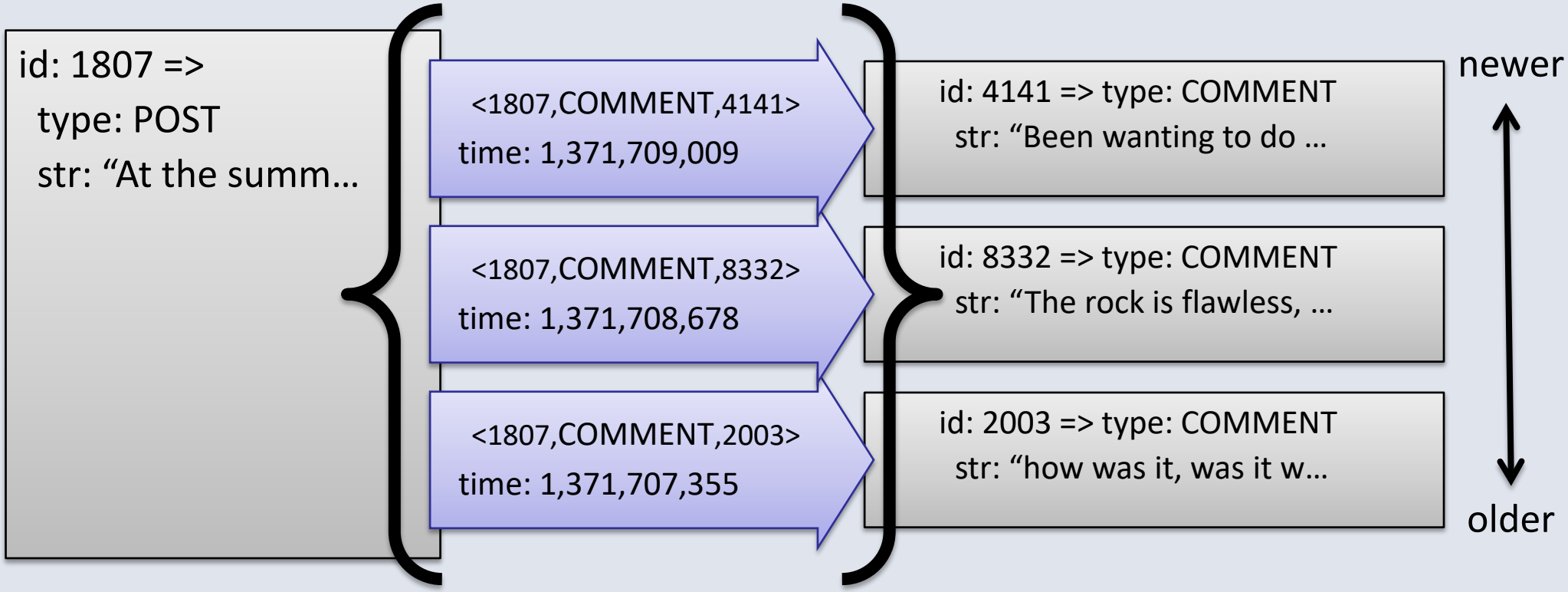
# Associations = Edges

- Identified by <id1, type, id2>
- Bidirectional associations are two edges, same or different type



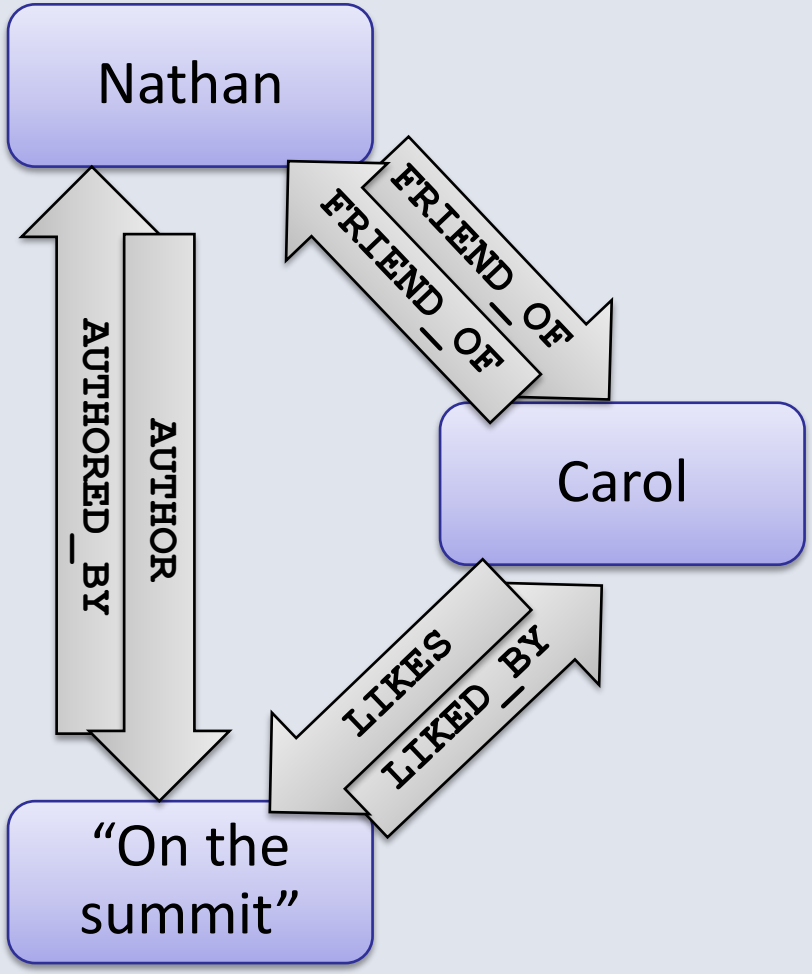
# Association Lists

- <id1, type, *>
- Descending order by time
- Query sublist by position or time
- Query size of entire list



# Inverse associations

- Bidirectional relationships have separate *a*→*b* and *b*→*a* edges
  - inv_type(**`LIKES`**) = **`LIKED_BY`**
  - inv_type(**`FRIEND_OF`**) = **`FRIEND_OF`**
- Forward and inverse types linked only during write
  - TAO **`assoc_add`** will update both
  - Not atomic, but failures are logged and repaired



# Objects and Associations API

## Reads – 99.8%

- Point queries
  - **obj_get** 28.9%
  - **assoc_get** 15.7%
- Range queries
  - **assoc_range** 40.9%
  - **assoc_time_range** 2.8%
- Count queries
  - **assoc_count** 11.7%

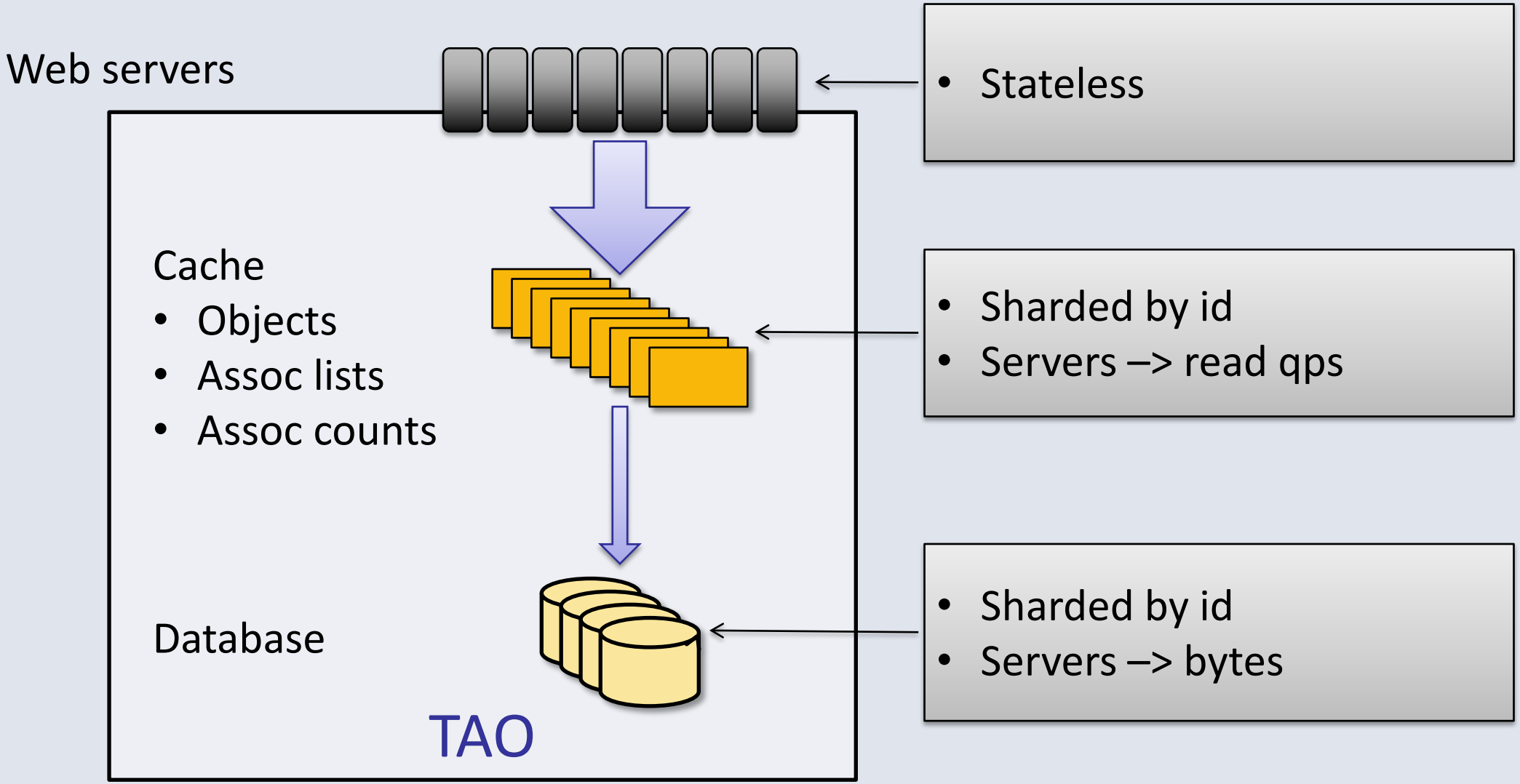
## Writes – 0.2%

- Create, update, delete for objects
  - **obj_add** 16.5%
  - **obj_update** 20.7%
  - **obj_del** 2.0%
- Set and delete for associations
  - **assoc_add** 52.5%
  - **assoc_del** 8.3%

# What Are TAO's Goals/Challenges?

- Efficiency at scale
- Low read latency
- Timeliness of writes
- High Read Availability

# Independent Scaling by Separating Roles

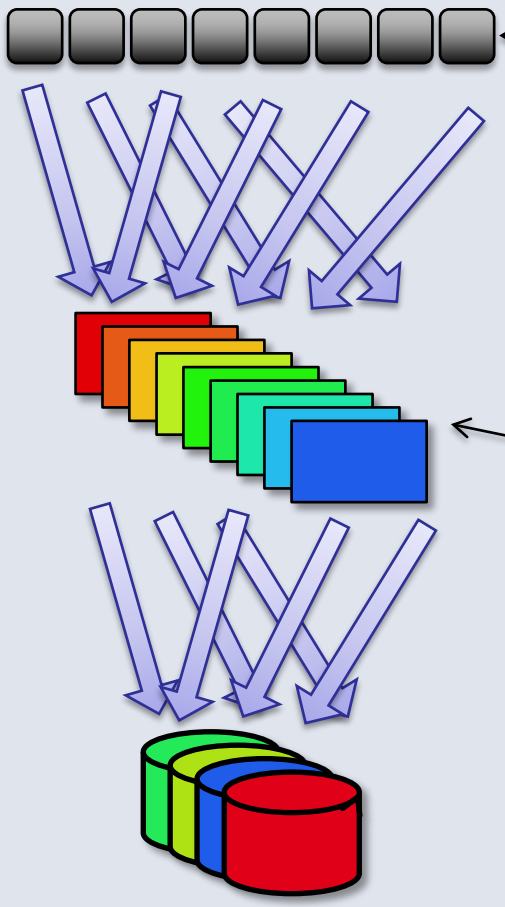


# Subdividing the Data Center

Web servers

Cache

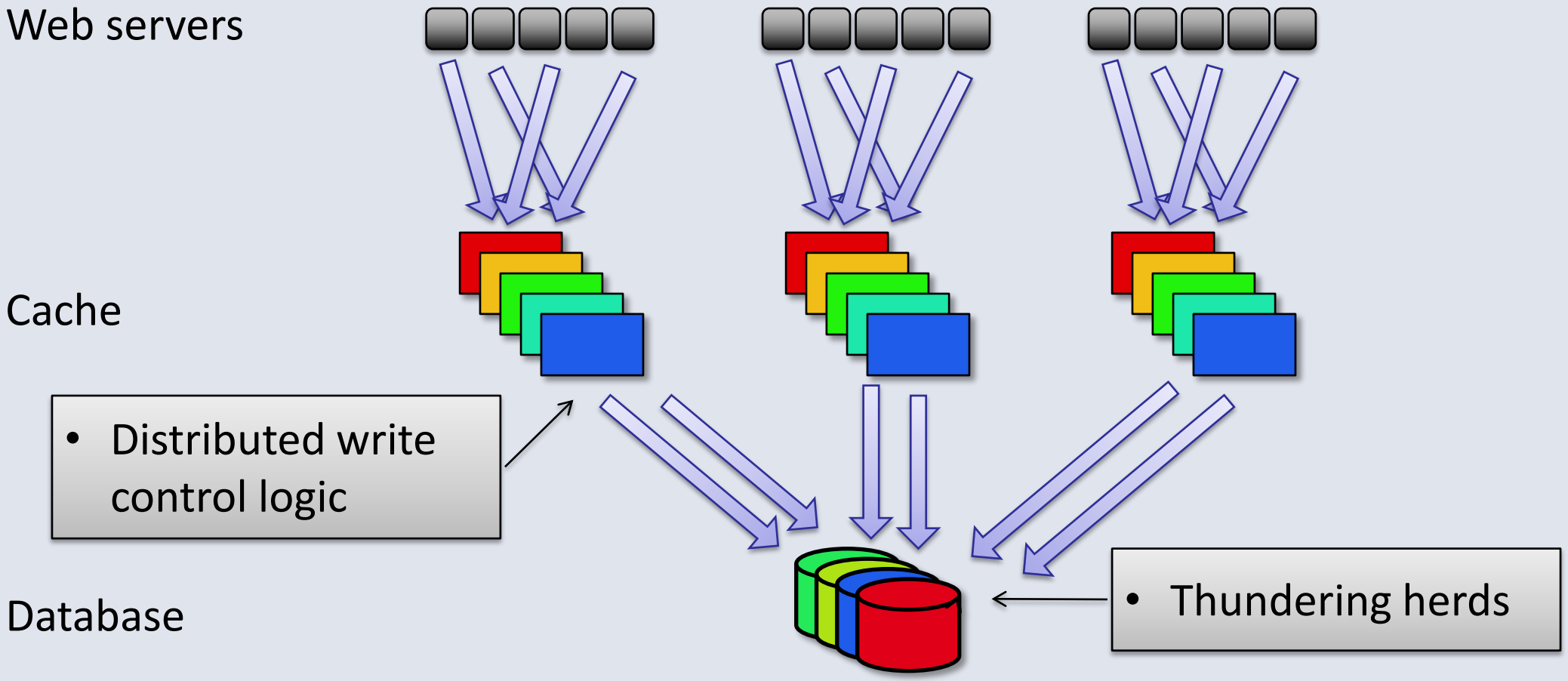
Database



- Inefficient failure detection
- Many switch traversals

- Many open sockets
- Lots of hot spots

# Subdividing the Data Center



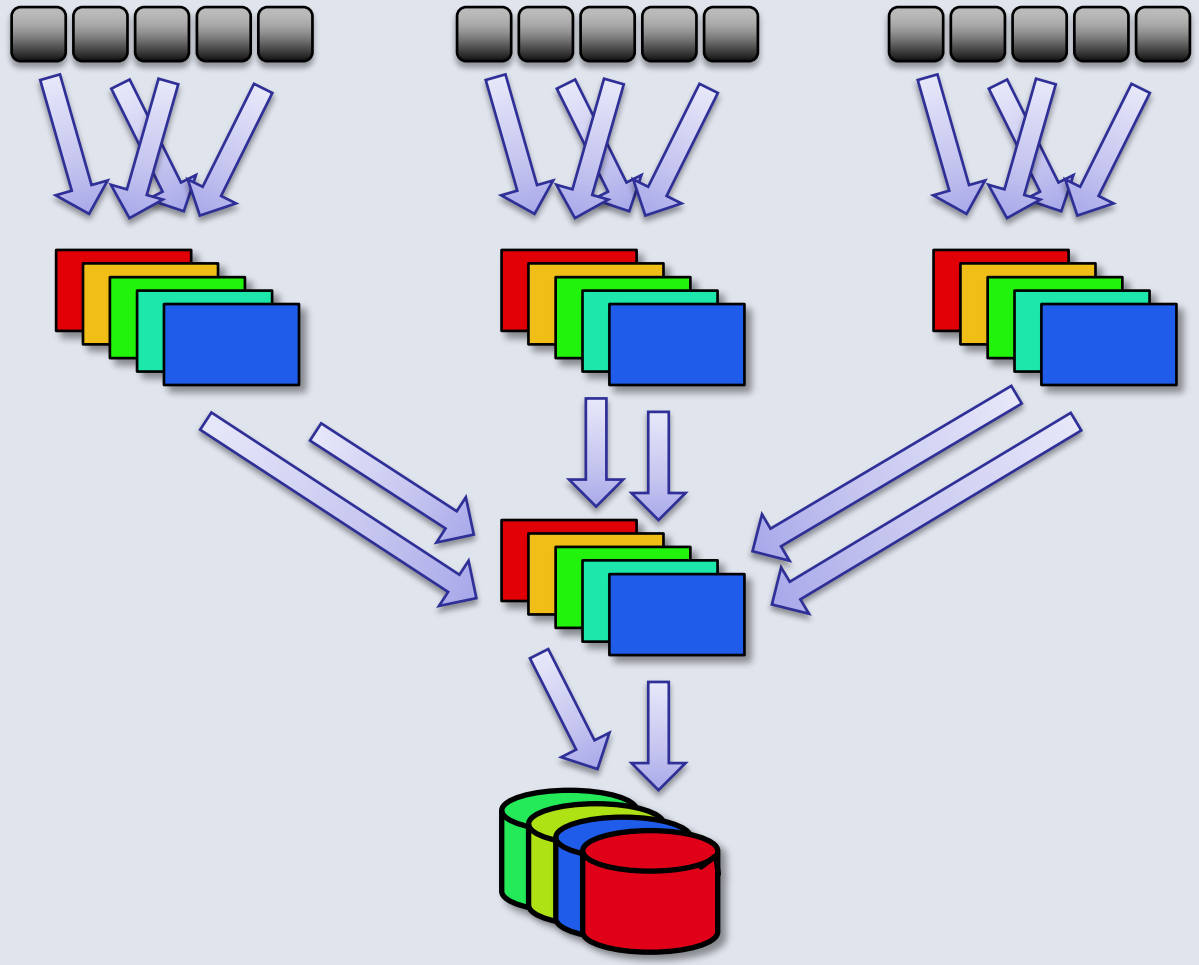
# Follower and Leader Caches

Web servers

Follower cache

Leader cache

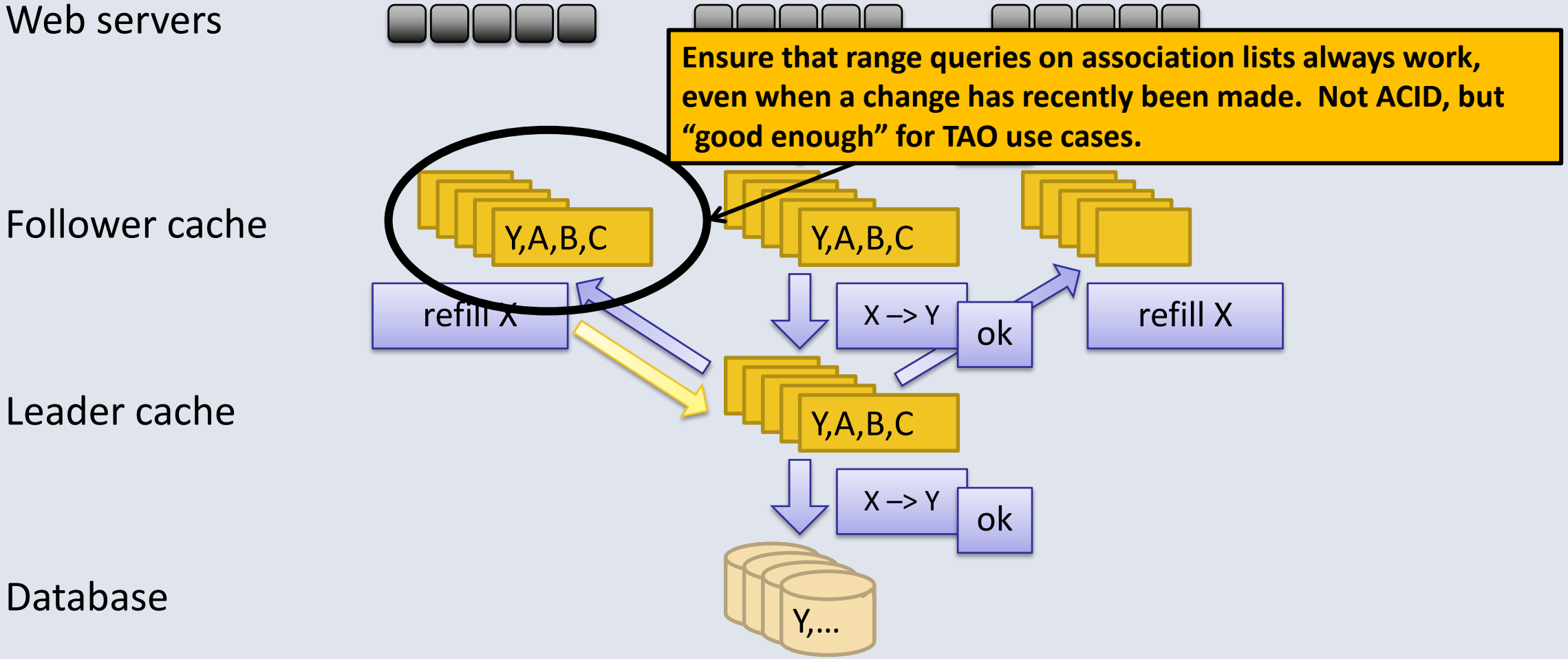
Database



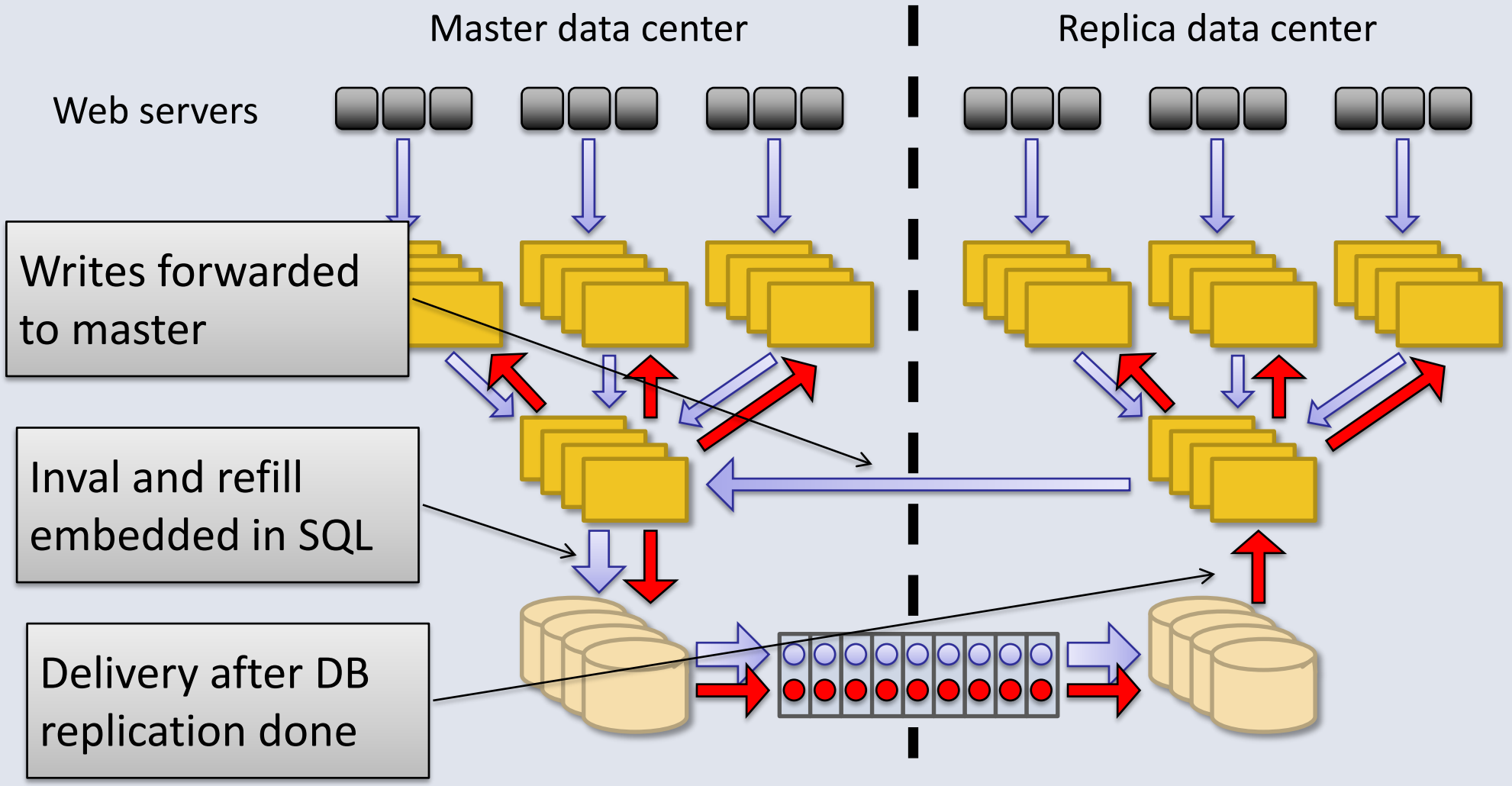
# What Are TAO's Goals/Challenges?

- Efficiency at scale
- Low read latency
- Timeliness of writes
- High Read Availability

# Write-through Caching – Association Lists



# Asynchronous DB Replication



# What Are TAO's Goals/Challenges?

- Efficiency at scale
- Low read latency
- Timeliness of writes
- High Read Availability

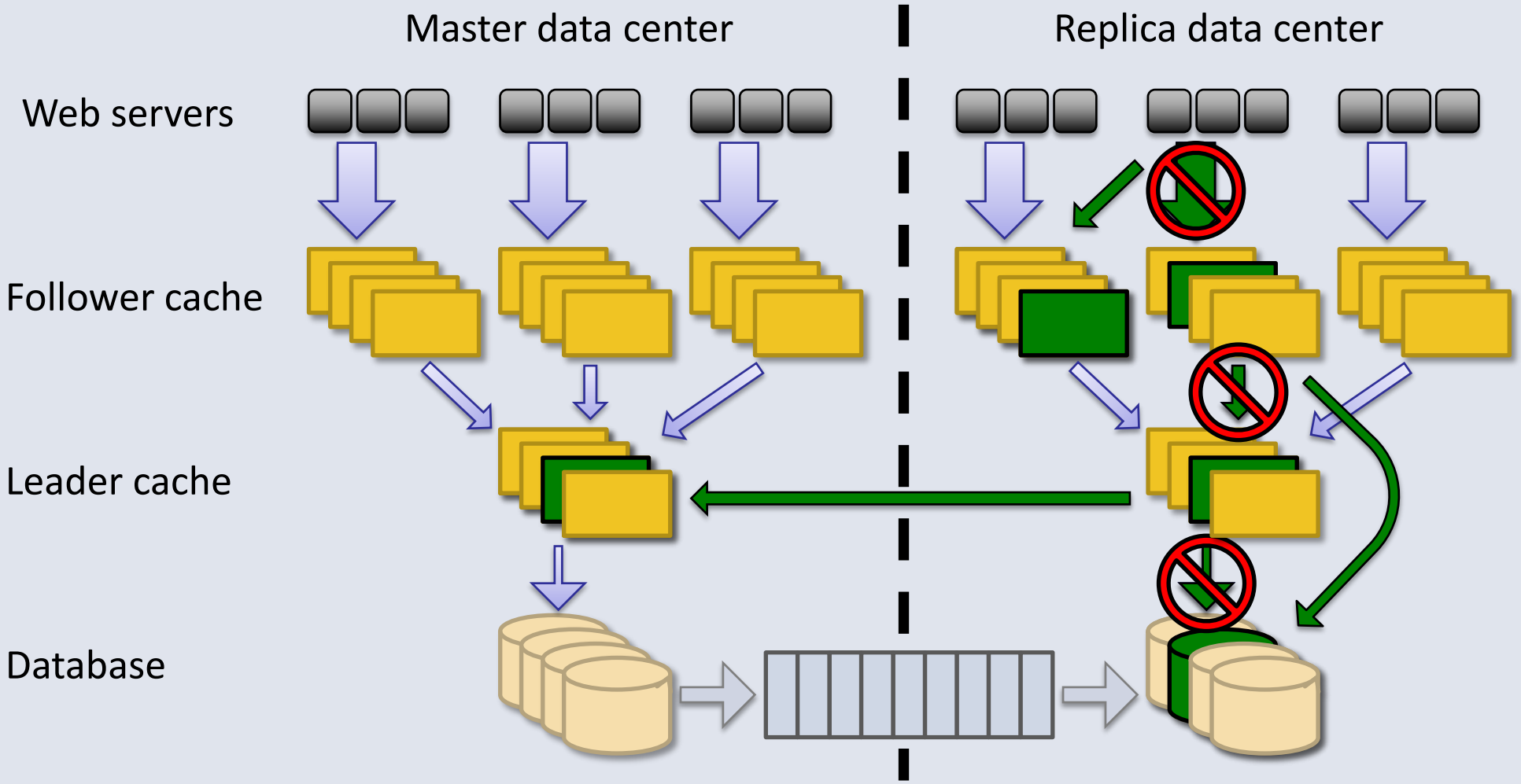
# Key Ideas

- TAO has a “normal operations” pathway that offers pretty good properties, very similar to full ACID.

- But they also have backup pathways for almost everything, to try to preserve updates (like unfollow, or unfriend, or friend, or like) even if connectivity to some portion of the system is disrupted.

- This gives a kind of self-repairing form of fault tolerance.  It doesn’t promise a clean ACID model, yet is pretty close to that.

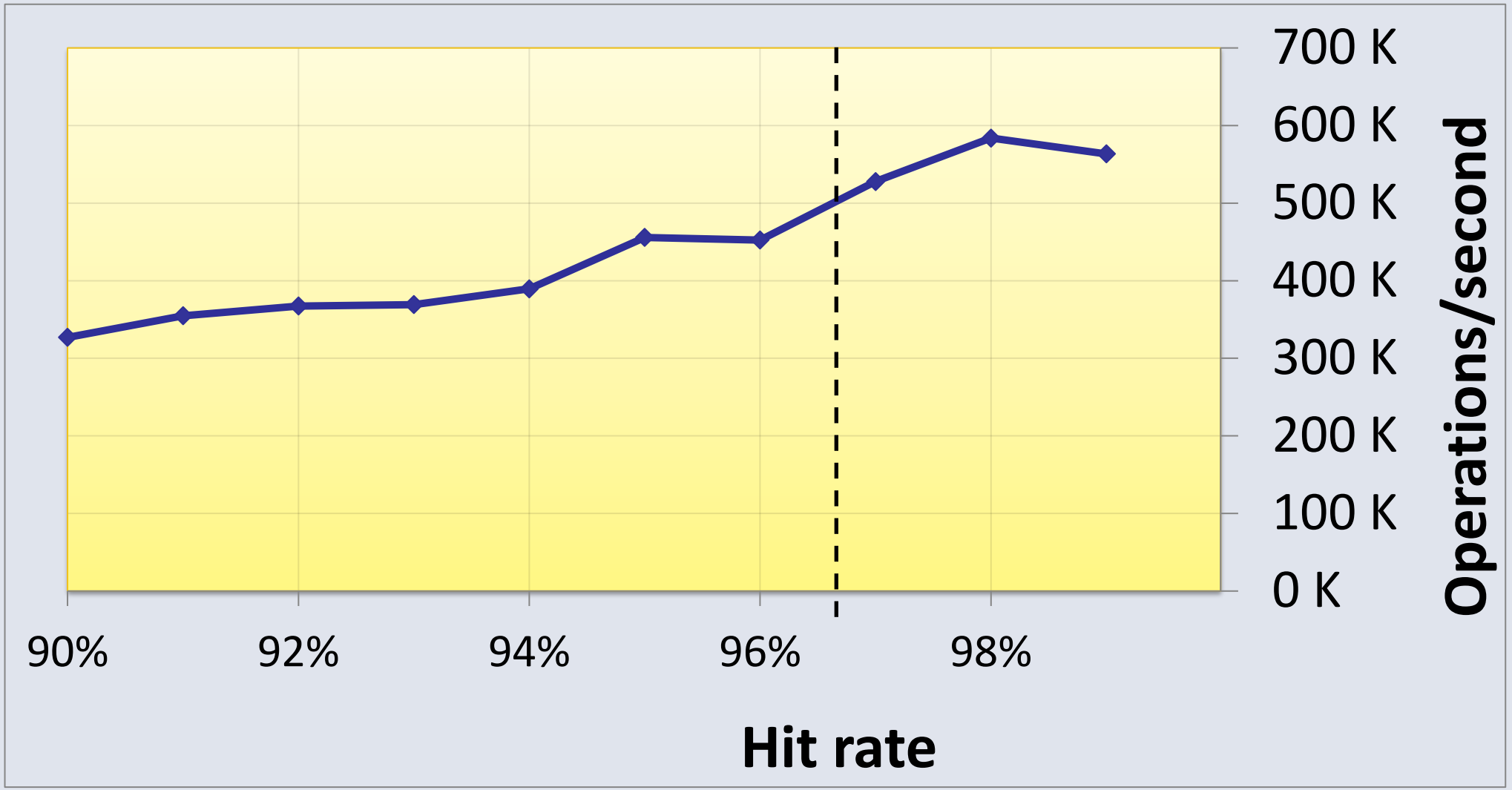
# Improving Availability: Read Failover



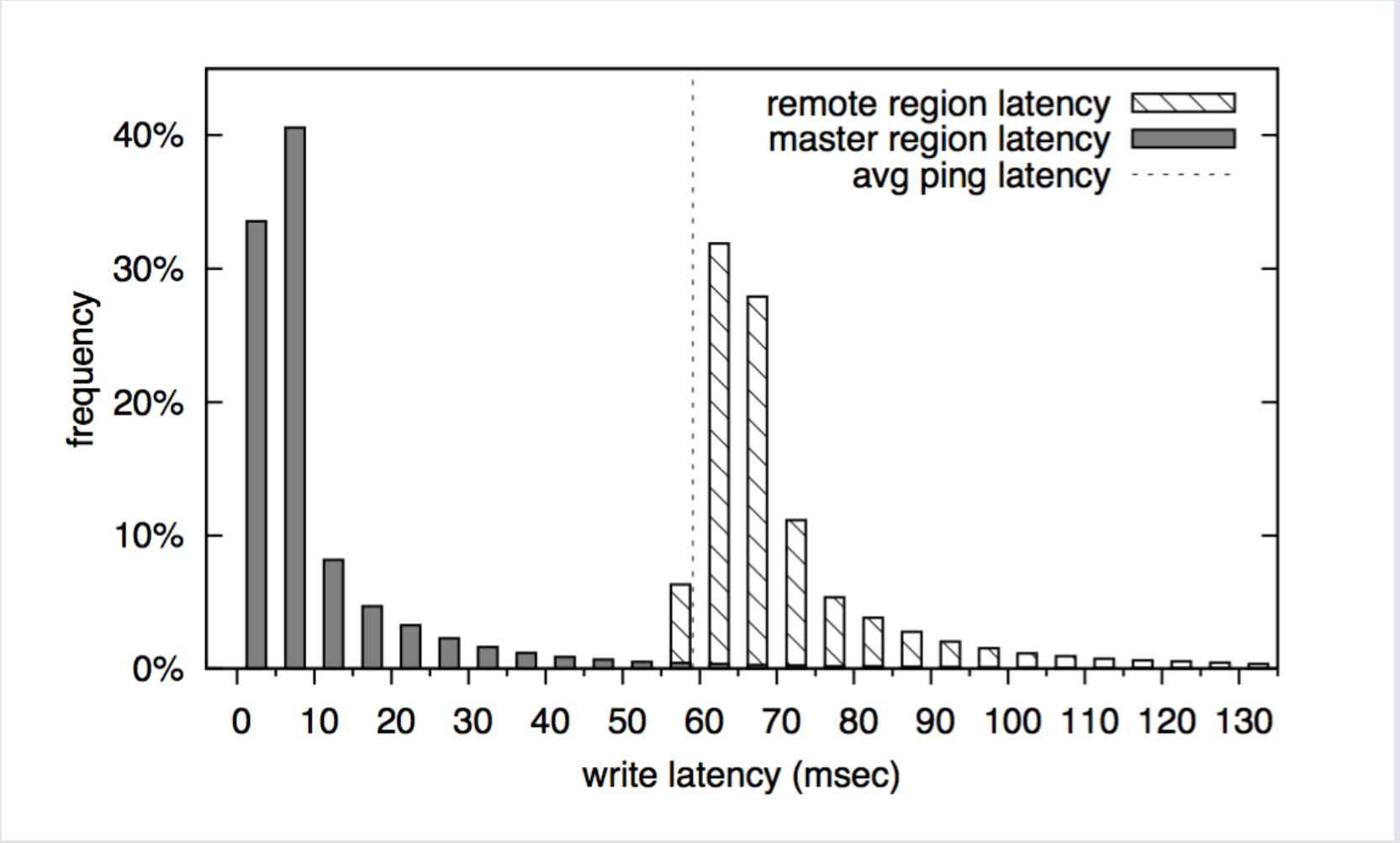
# TAO Summary

| | |
|---|---|
| Efficiency at scale<br>Read latency | • Separate cache and DB<br>• Graph-specific caching<br>• Subdivide data centers |
| Write timeliness | • Write-through cache<br>• Asynchronous replication |
| Read availability | • Alternate data sources |

# Single-server Peak Observed Capacity

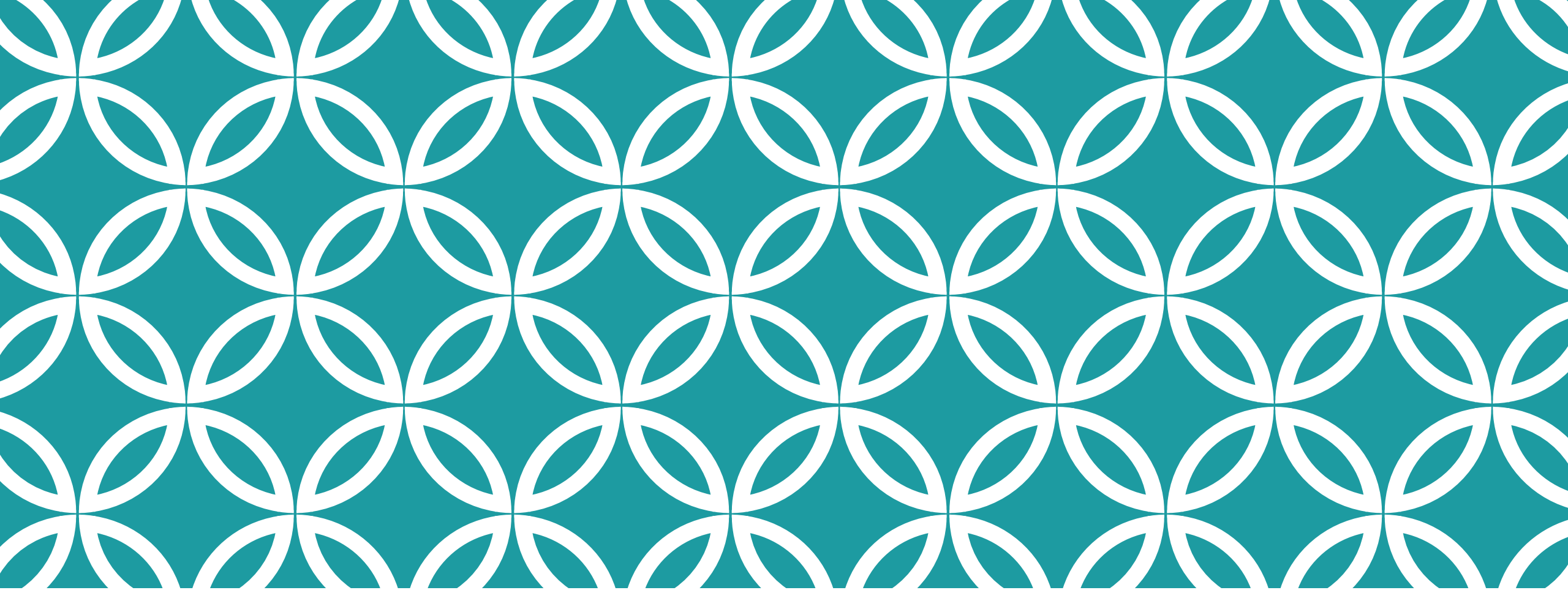


# Write latency



# More In the Paper

- The role of association time in optimizing cache hit rates
- Optimized graph-specific data structures
- Write failover
- Failure recovery
- Workload characterization



# BACK TO OUR CORE TOPIC

(End-of-TAO)

# WHAT HAVE WE LEARNED?

The broad pattern remains constant: sharded, massive structures.

Facebook TAO shows us how we can even maintain and update such a structure at runtime.

Big data "analytic" frameworks are used to develop completely new machine learning models by looking for structure under human guidance.

# BIG DATA ANALYTICS

These are frameworks for efficiently doing massively parallel, "always sharded" computing, aimed at producing useful knowledge.

The data starts out sharded, and often even the intermediary states and results are sharded too. But the ultimate goal is a chart or some other human-useful output.

We'll look at some of the tools (like MapReduce) in upcoming lectures.

# MACHINE LEARNING

We can understand this as the task of taking some kind of model and fitting it to the data – computing parameters that match the numerical model to the input.

Some models and tools are very basic

But the "higher levels" machine-learning models are powerful data summarization engines, and can make predictions/classifications.

# HOW DOES IoT CHANGE THIS GAME?

All of these infrastructures evolved with slowly changing data, and run on huge banks of computers as "back-end" compute tasks. The data is massive but "right there".

With IoT we will see situations where most of the data actually lives in the real world. The sensors capture glimpses, but even that data can't be downloaded in full detail. Worse, sensing devices have very limited compute / battery power.

We usually start by downloading thumbnails and meta-data. Will this suffice?

# THE FIRST WAVE OF SUCCESS STORIES?

IoT's version of big data will be shaped by early successes: research papers that show how to extract useful information from a mix of

- Very small data objects from lots of sensors (the meta-data)
- A few selectively downloaded high-resolution of images and videos, which even so will need to be immediately compressed, deduplicated, segmented and tagged, etc.
- Even the most basic steps will break new ground for the cloud! But each success story will leave us with technology to enable next steps.

# DIFFERENCES RELATIVE TO NORMAL BIG DATA?

The data is out there, not in here.

Part of the task is to figure out which information is worth downloading.

Many parts of IoT operate under time pressure and won't have a lot of time to decide what to do. IoT images/videos quickly become stale.

# WHERE WILL IoT TOOLS RUN?

Any form of real-time learning or decision making may need to run "close" to the function server, which is where the IoT Hub delivers incoming events.

Today you can already place company servers in that part of the cloud, as part of the "hybrid cloud" model. Those same tools and capabilities can be reused to place new intelligent μ-services you might build there.

But an open question is whether these new μ-services would have a way to access accelerators like GPU and TPU clusters or FGA kernels, and at what cost. These are deployed in a very cost-effective way in back-end machine learning platforms, and they play big roles there.

# WHERE WILL THE NEW TOOLS BE INVENTED?

This first wave of IoT solutions will happen at companies that have a handle on the emerging market and see opportunities to monetize it.

So you'll see work by Google and Amazon, because of Google Nest and Alexa, Microsoft because of Azure IoT, Waymo for self-driving cars, etc.

Gradually, this will give us tomorrow's IoT big data infrastructure. You can be a part of all that if you love this sort of challenge!

# WHAT IoT USES CREATE GRAPHICAL DATA?

Covid spread would be an example of an IoT data problem in which we need a graph to track contacts. Quiz 9 is on this topic, but doesn't actually build or study the graph.

Social networking may be relevant even for farms: for example, vineyards in California are struggling to understand spread of "red leaf blotch". Perhaps a networking perspective would clarify (is this spread by insects? By vineyard workers or equipment? On the wind?)

# CONCLUSIONS: BIG IoT DATA

The big data world will be evolving rapidly under demand from IoT uses.

We should look for existing solutions and ask how much of the infrastructure can be reused for these IoT cases. Example: Hybrid cloud can increase our confidence that a μ-service model is genuinely viable.

But we should be "cautious" and not change lots of things all at once. The cloud is surprisingly delicate and not everything would scale, be easy to manage, be cost-effective, be performant, etc.

Follow the money to understand which aspects will mature most quickly.