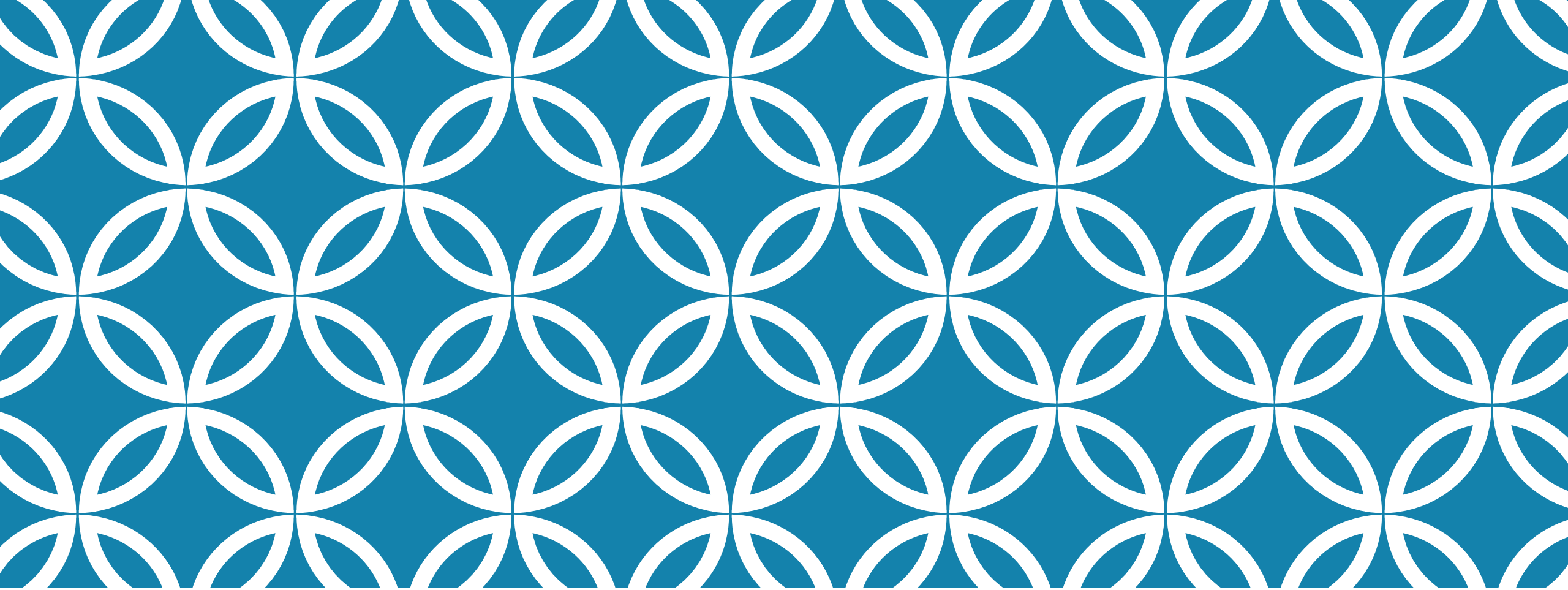


# CS5412 / LECTURE 25 SOCIAL NETWORK GRAPHS

Ken Birman

Fall, 2022

# 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.

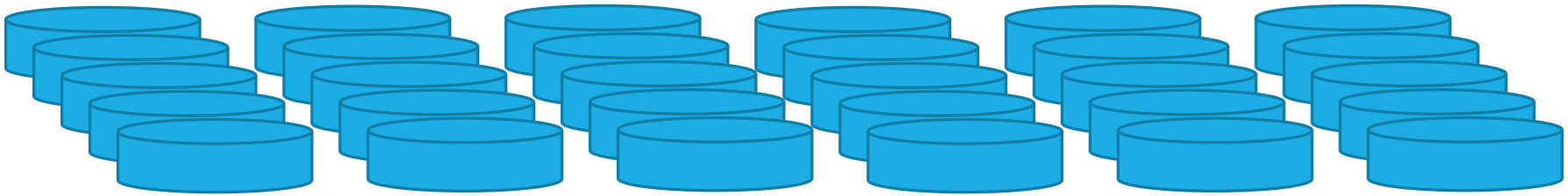
Social networking graph for all of Facebook

# MANY CHALLENGES

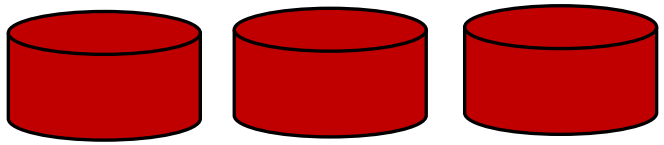

XenonStack.com

# 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

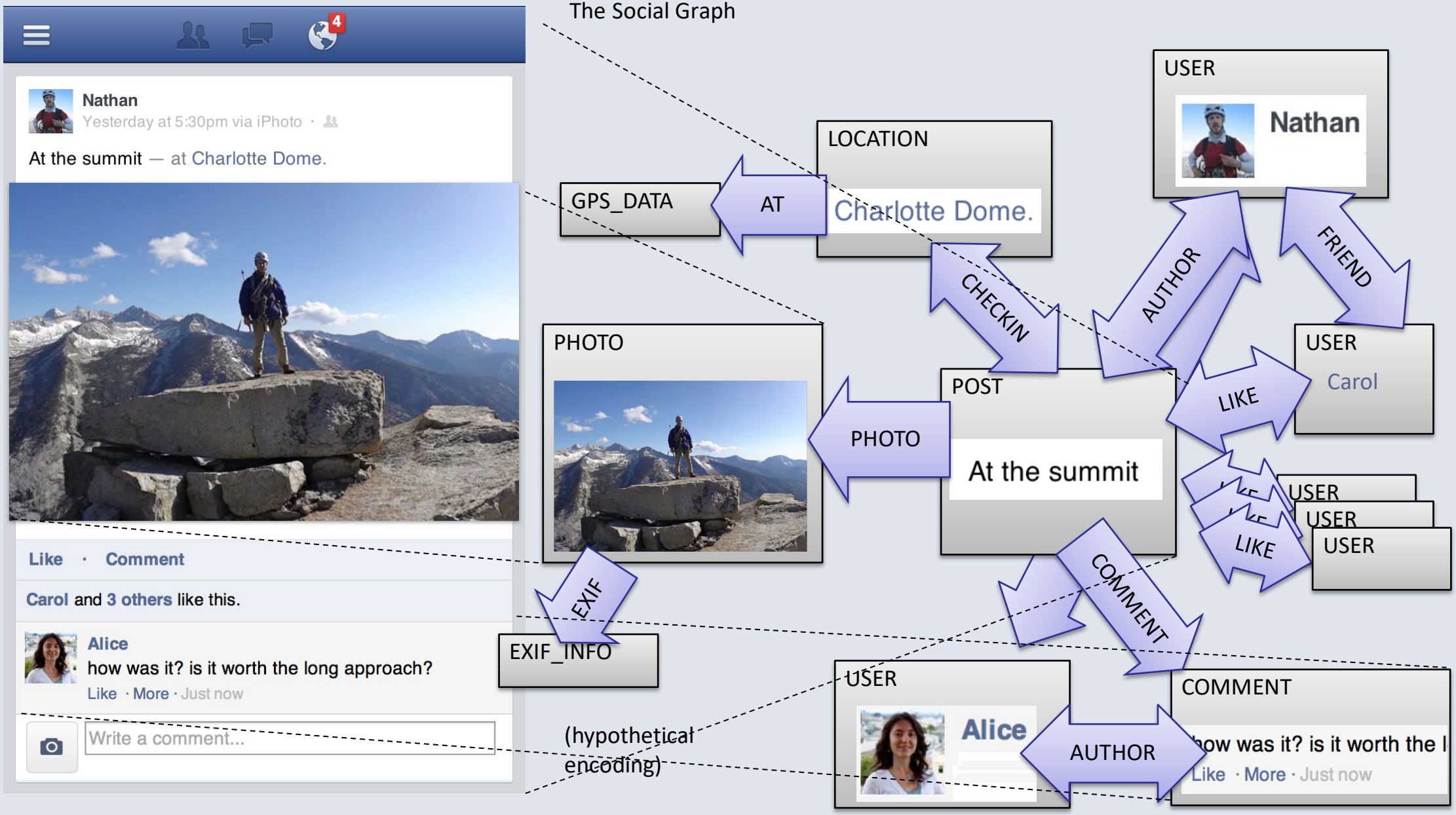
# TAO

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

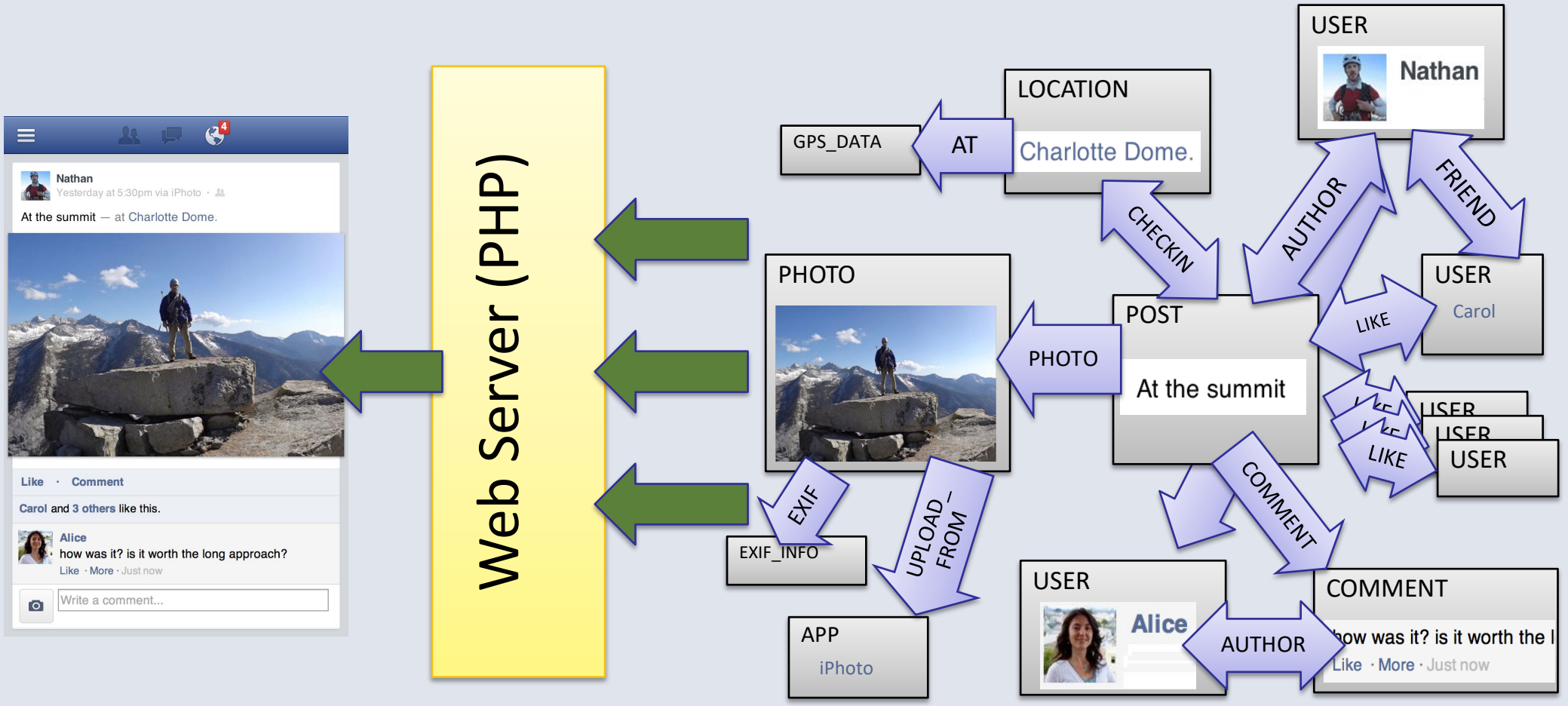
Cornell PhD who worked with Professor van Renesse. Graduated in 2010
Became director of engineering at Meta, then recently joined a startup.

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

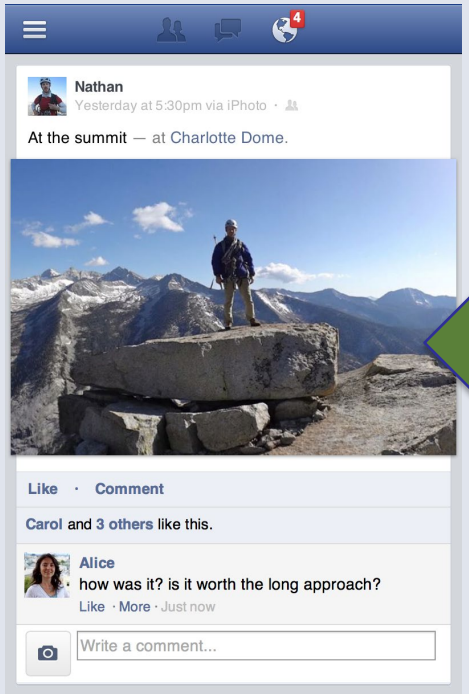
Presented at USENIX ATC – June 26, 2013


The Social Graph
Nathan
Yesterday at 5:30pm via iPhoto
At the summit — at Charlotte Dome.
Like · Comment
Carol and 3 others like this.
Alice
how was it? is it worth the long approach?
Like · More · Just now
Write a comment...
USER
Nathan
LOCATION
Charlotte Dome.
GPS_DATA
AT
CHECKIN
AUTHOR
FRIEND
PHOTO
PHOTO
POST
At the summit
LIKE
USER
Carol
USER
USER
USER
LIKE
EXIF
EXIF_INFO
COMMENT
USER
Alice
AUTHOR
COMMENT
how was it? is it worth the l
Like · More · Just now
(hypothetical encoding)

# Dynamically Rendering the Graph

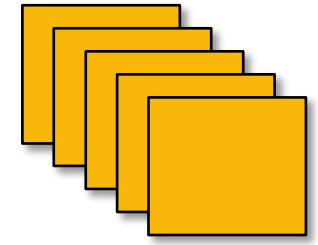


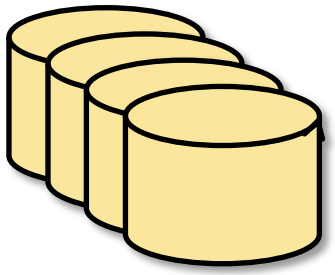
# Dynamically Rendering the Graph

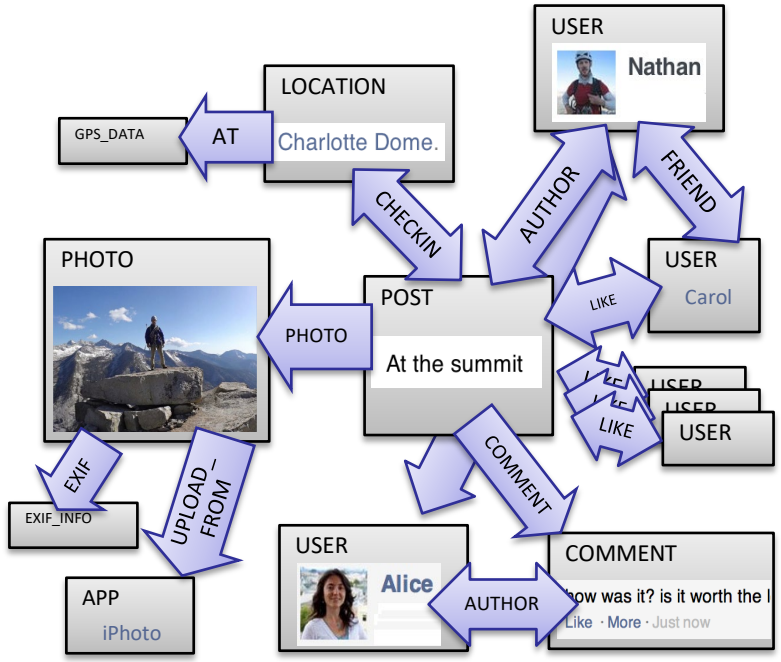



Web Server (PHP)

## 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.

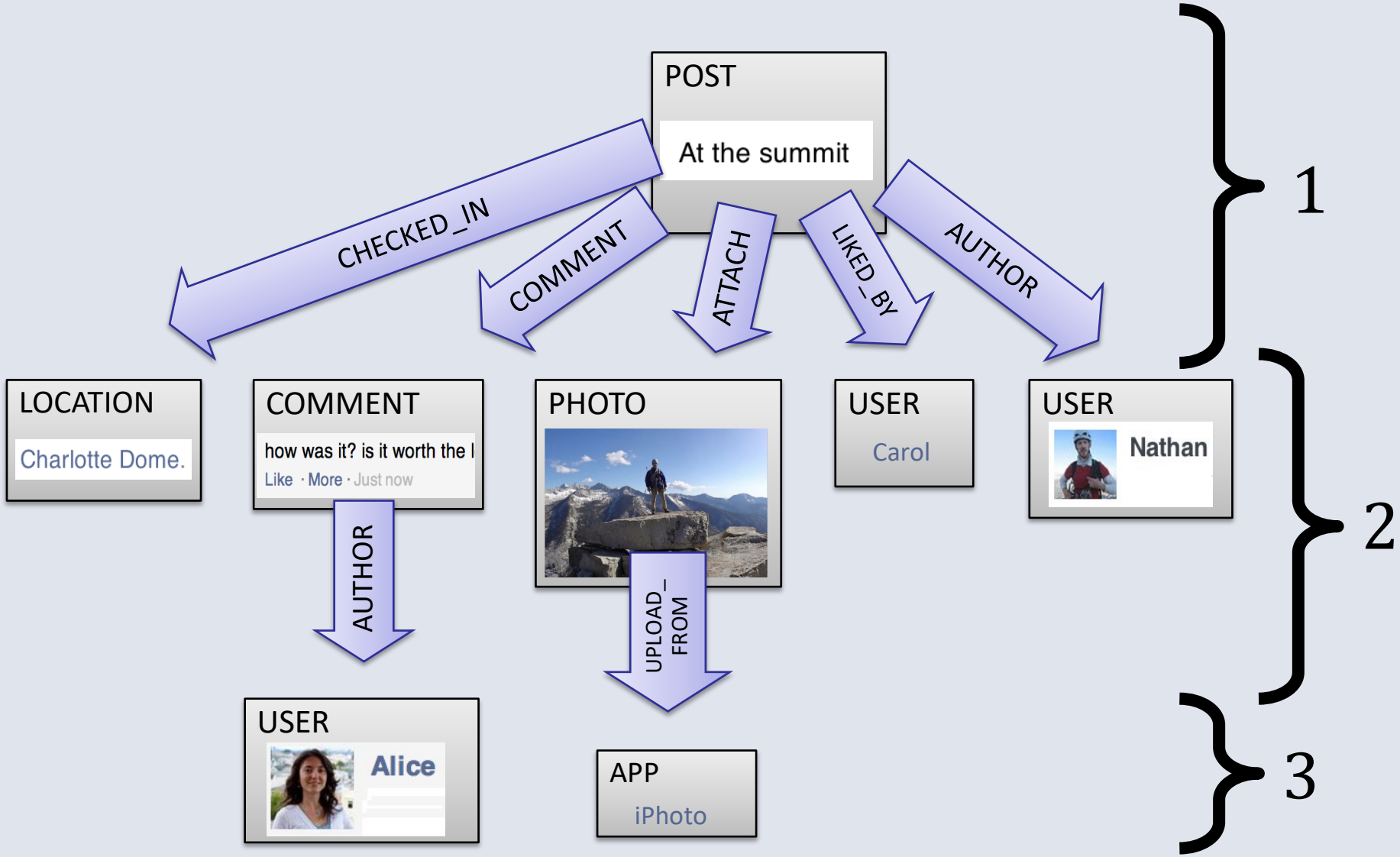
# Concept: A *data dependency*

- For TAO, if a node has an edge to some other node, we say that there is a data dependency: the origin of that directed edge "depends" on the destination

- They think of the graph as "broken" if a dependent node can't find the node it depends upon.
  - It would show up as a "?" on the Facebook feed, and they don't like that
  - Ideally, you only see a "?" if they deleted some sort of spam, fake news, etc

- The graph has a depth and so we can talk about dependencies at depth *d*

# Dynamic Resolution of Data Dependencies



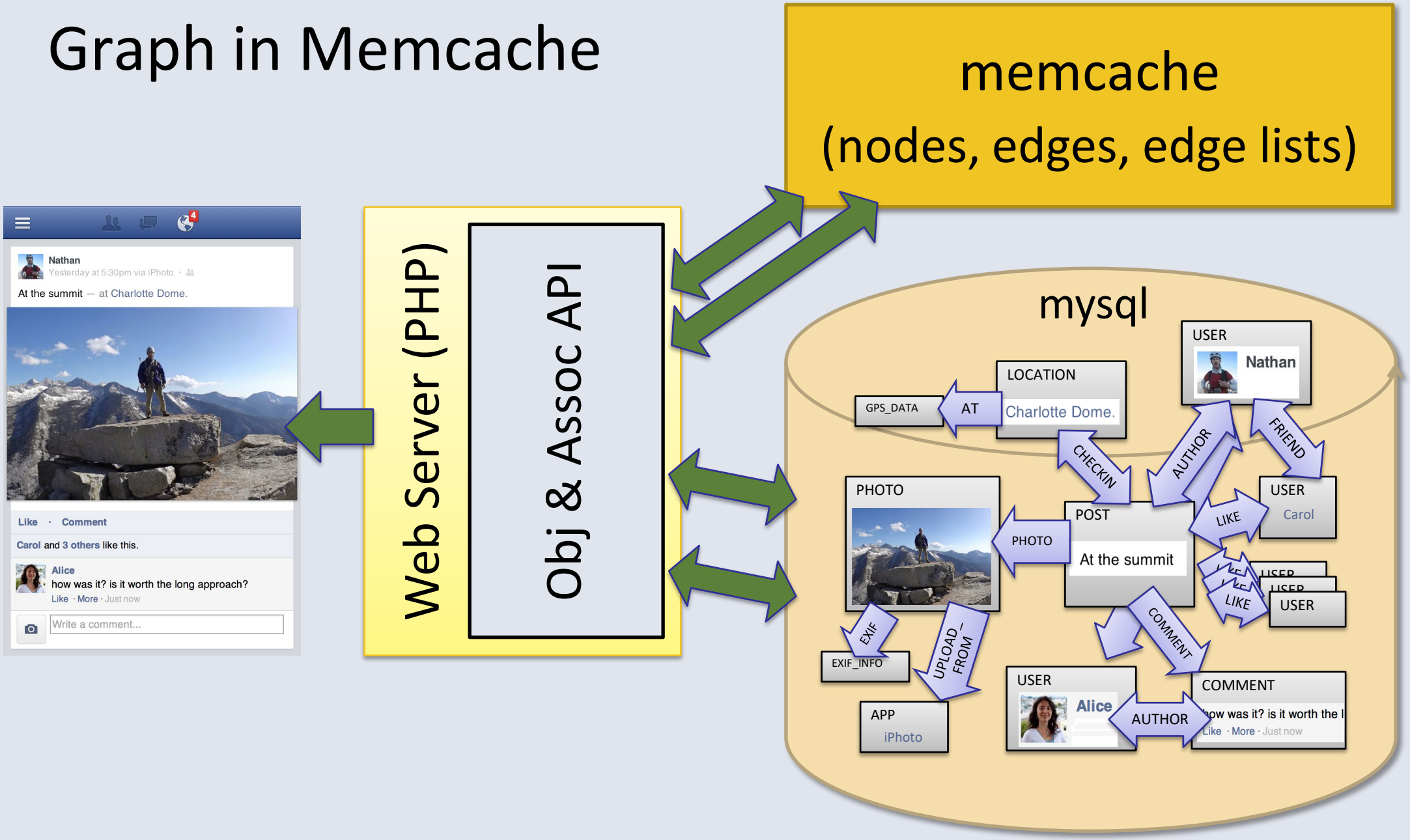
# What Are TAO's Goals/Challenges?

- Efficiency at scale

# What Are TAO's Goals/Challenges?

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

# Graph in Memcache

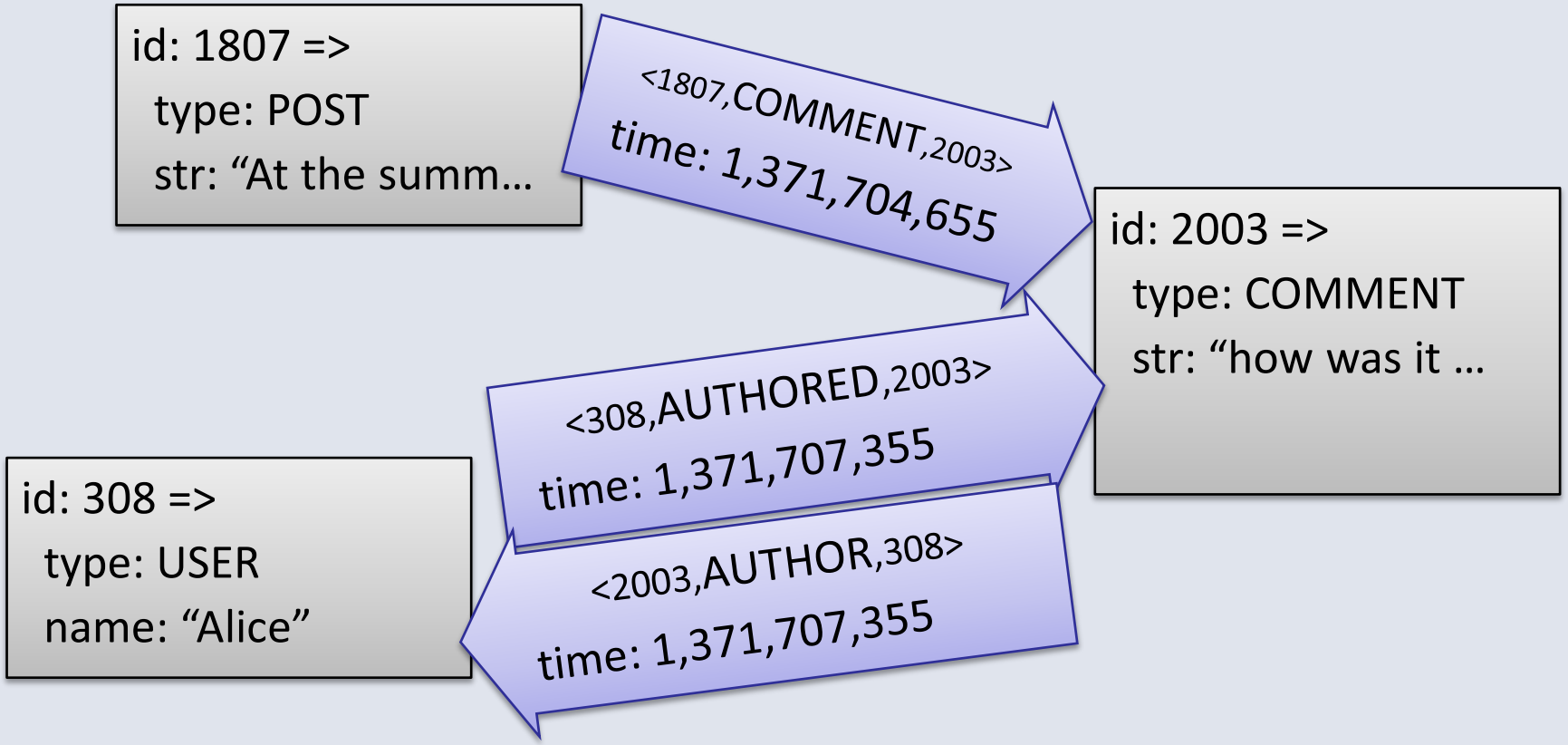


# Objects = Nodes

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

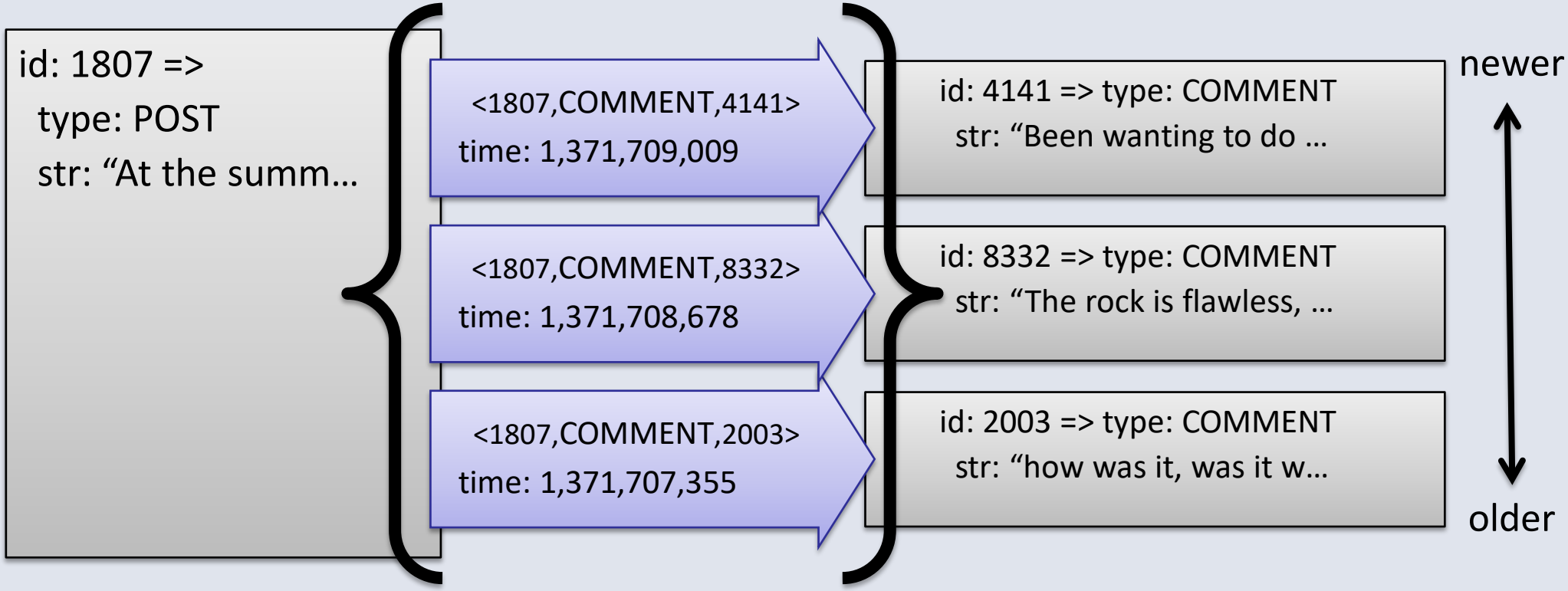
# Associations = Edges

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



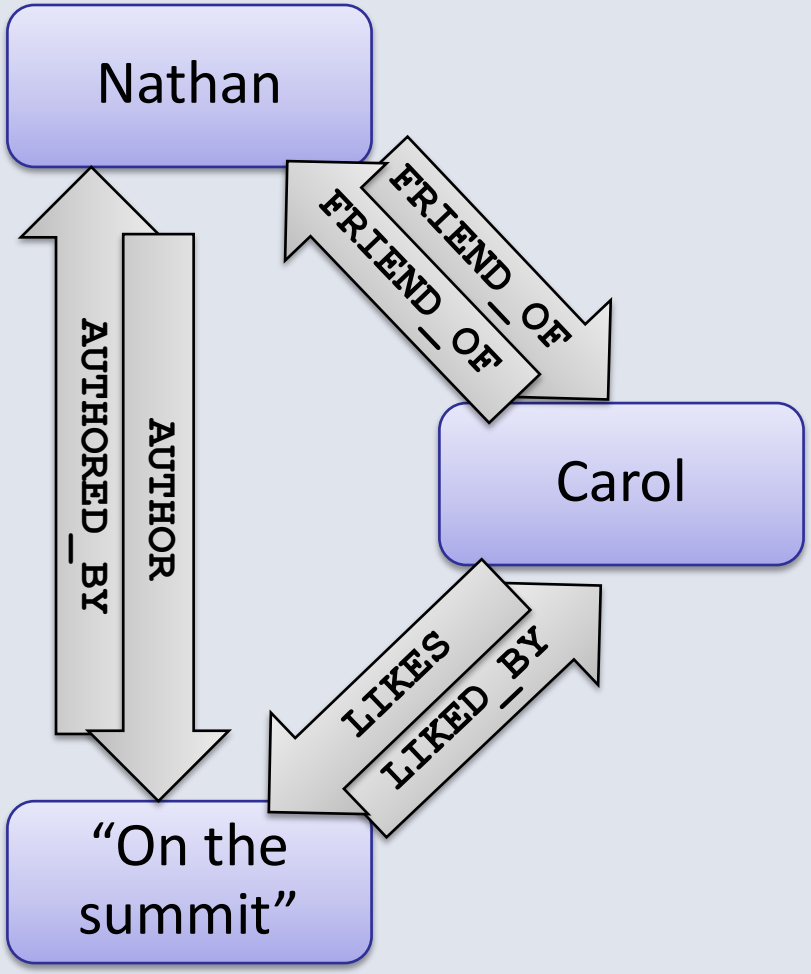
# Association Lists

- <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%

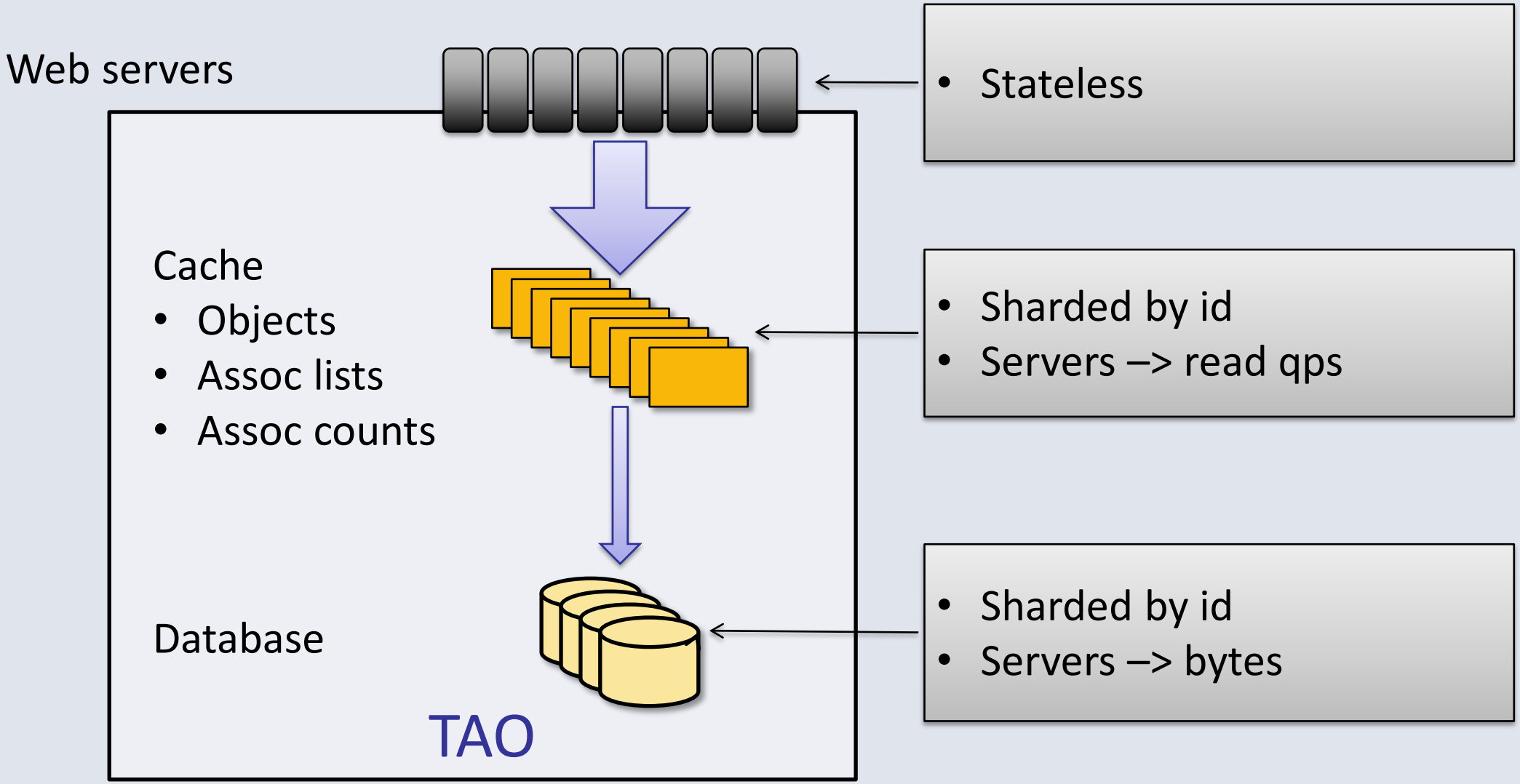
## Writes – 0.2%

- 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

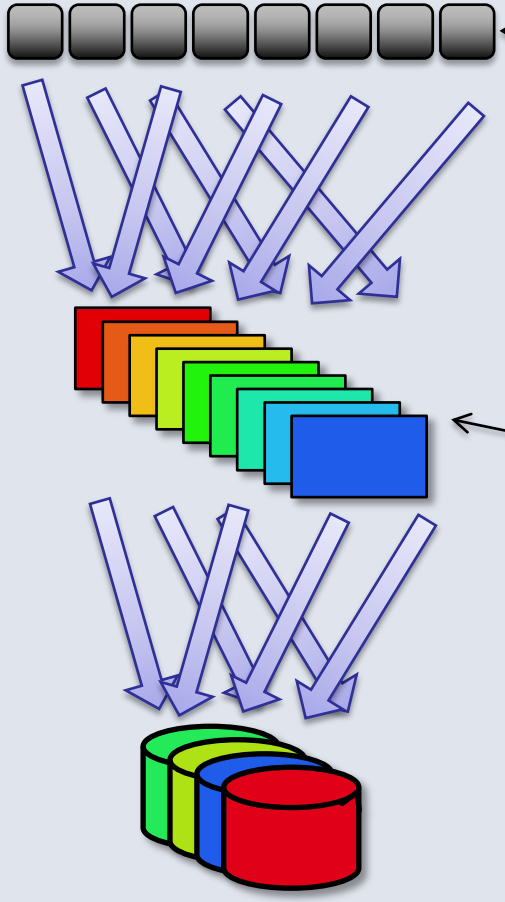


# Subdividing the Data Center

Web servers

Cache

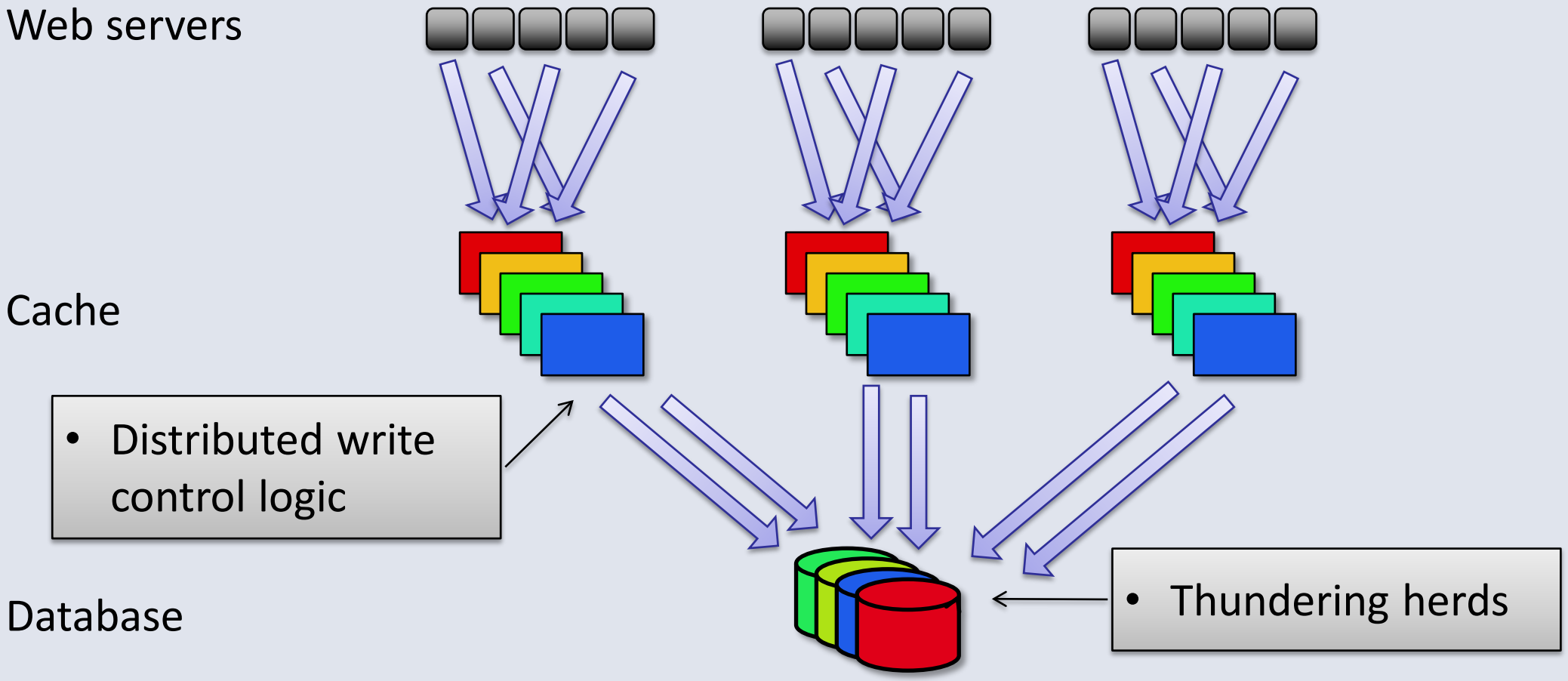
Database



- Inefficient failure detection
- Many switch traversals

- Many open sockets
- Lots of hot spots

# Subdividing the Data Center



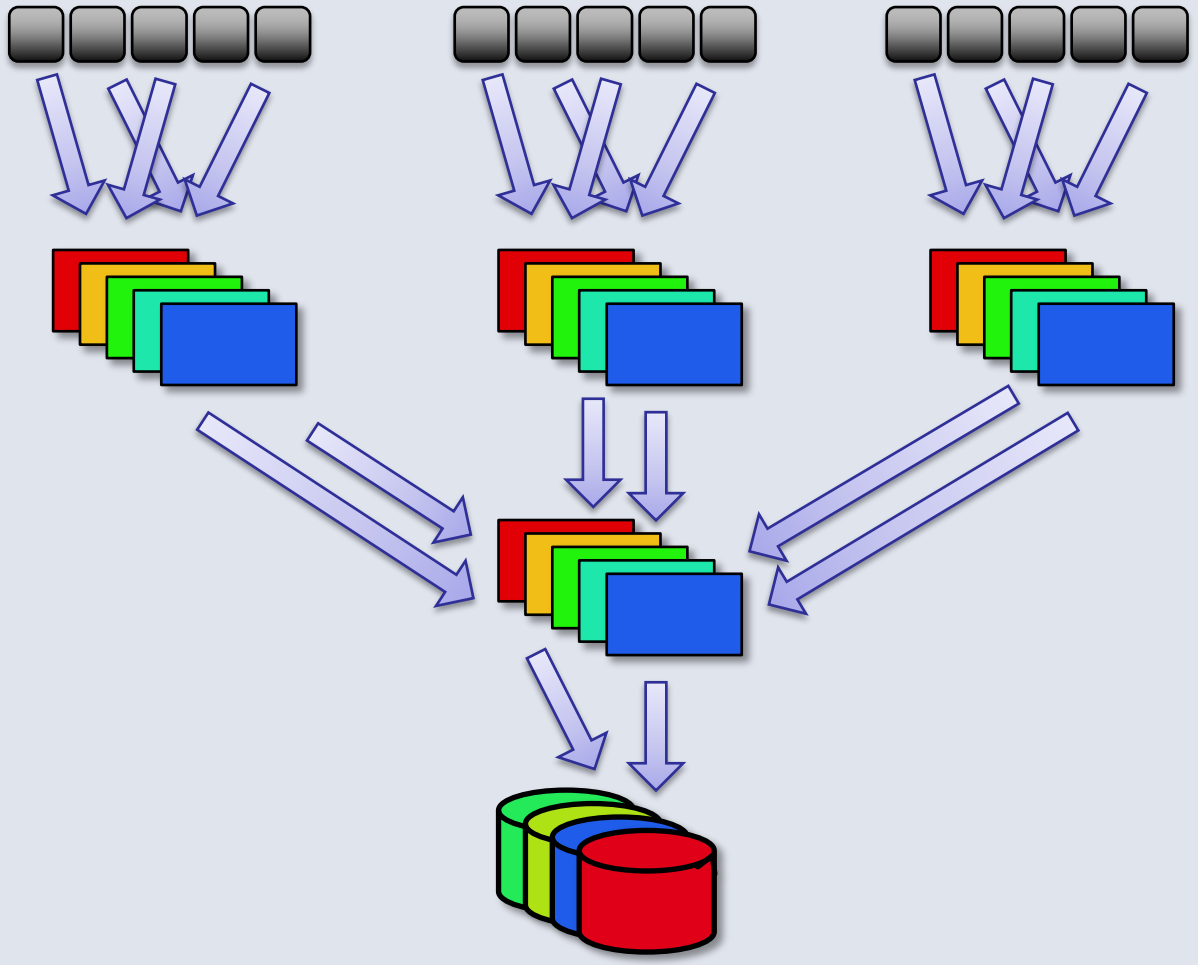
# Follower and Leader Caches

Web servers

Follower cache

Leader cache

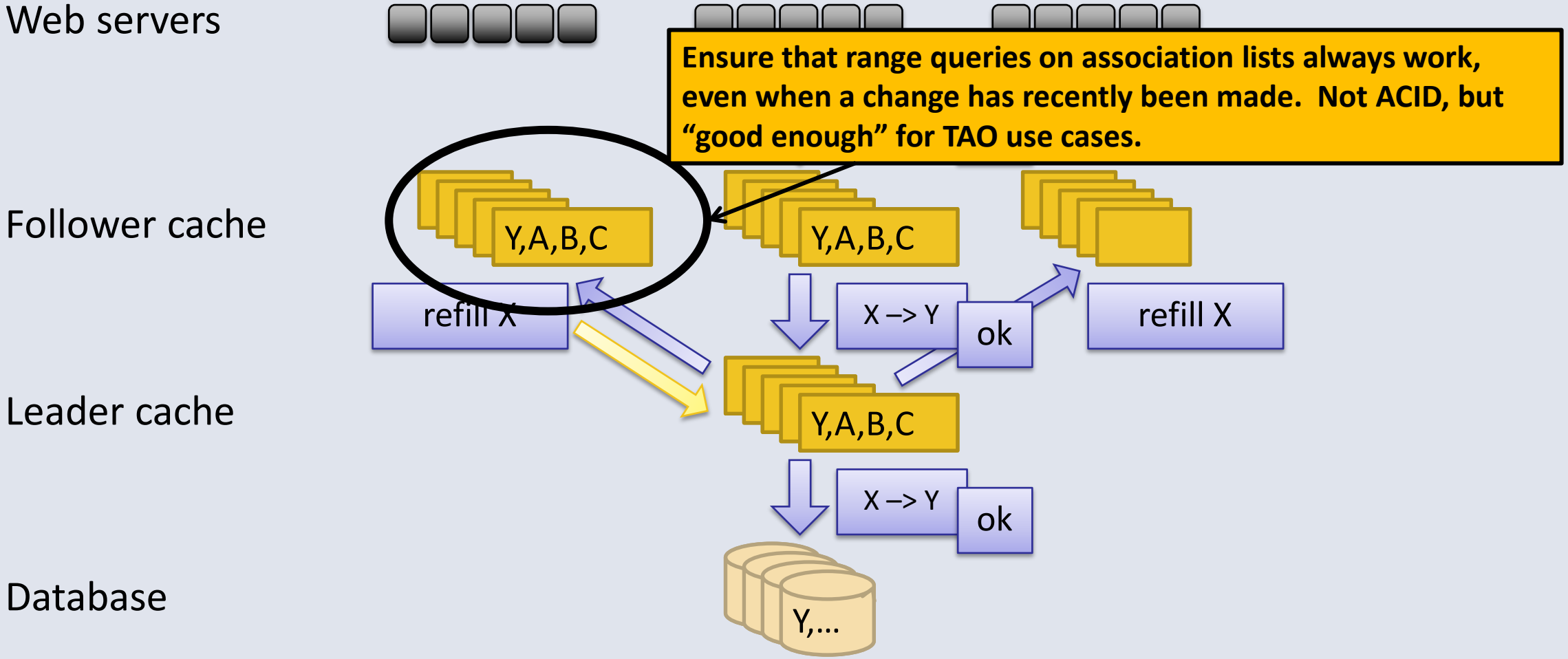
Database



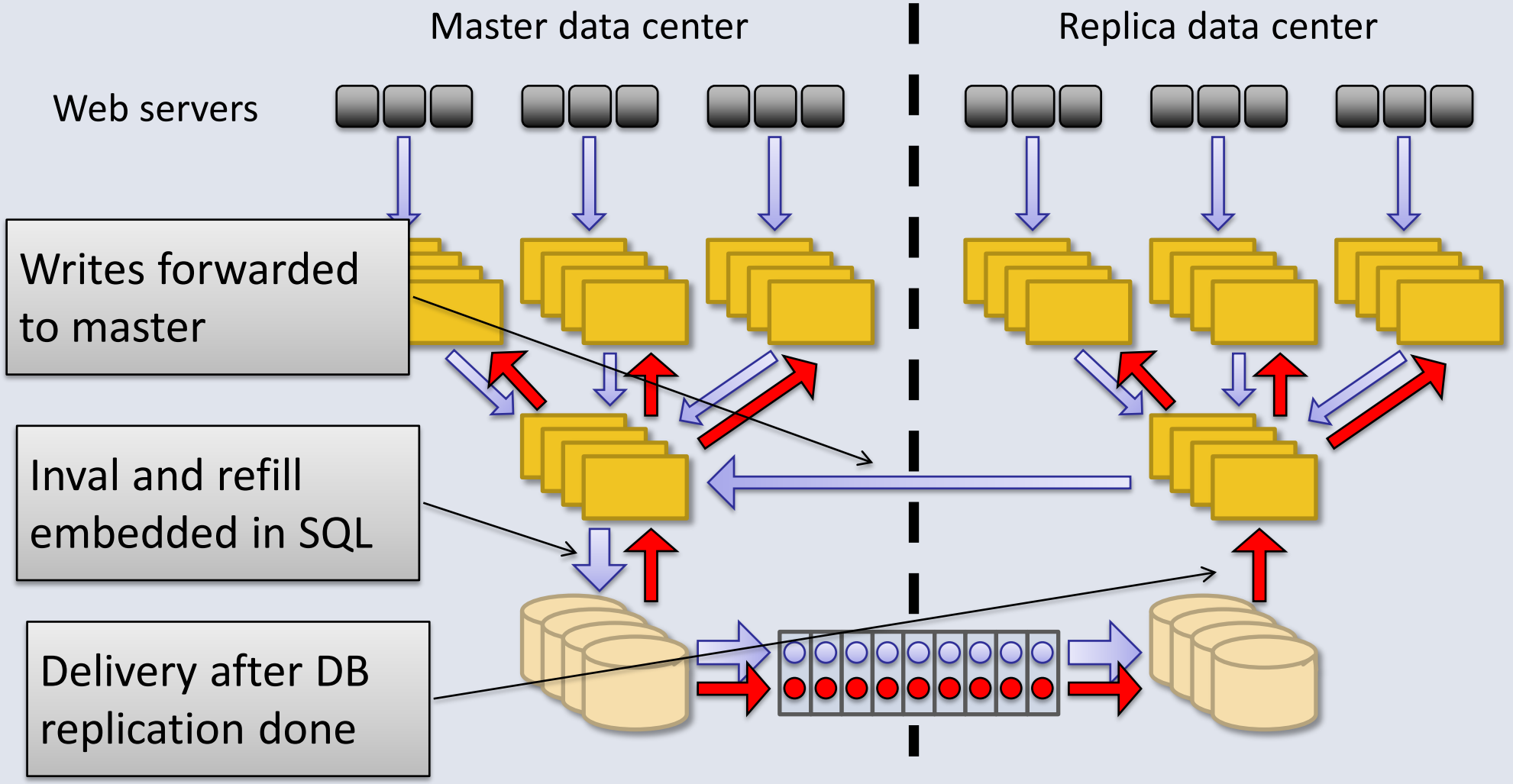
# What Are TAO's Goals/Challenges?


- 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

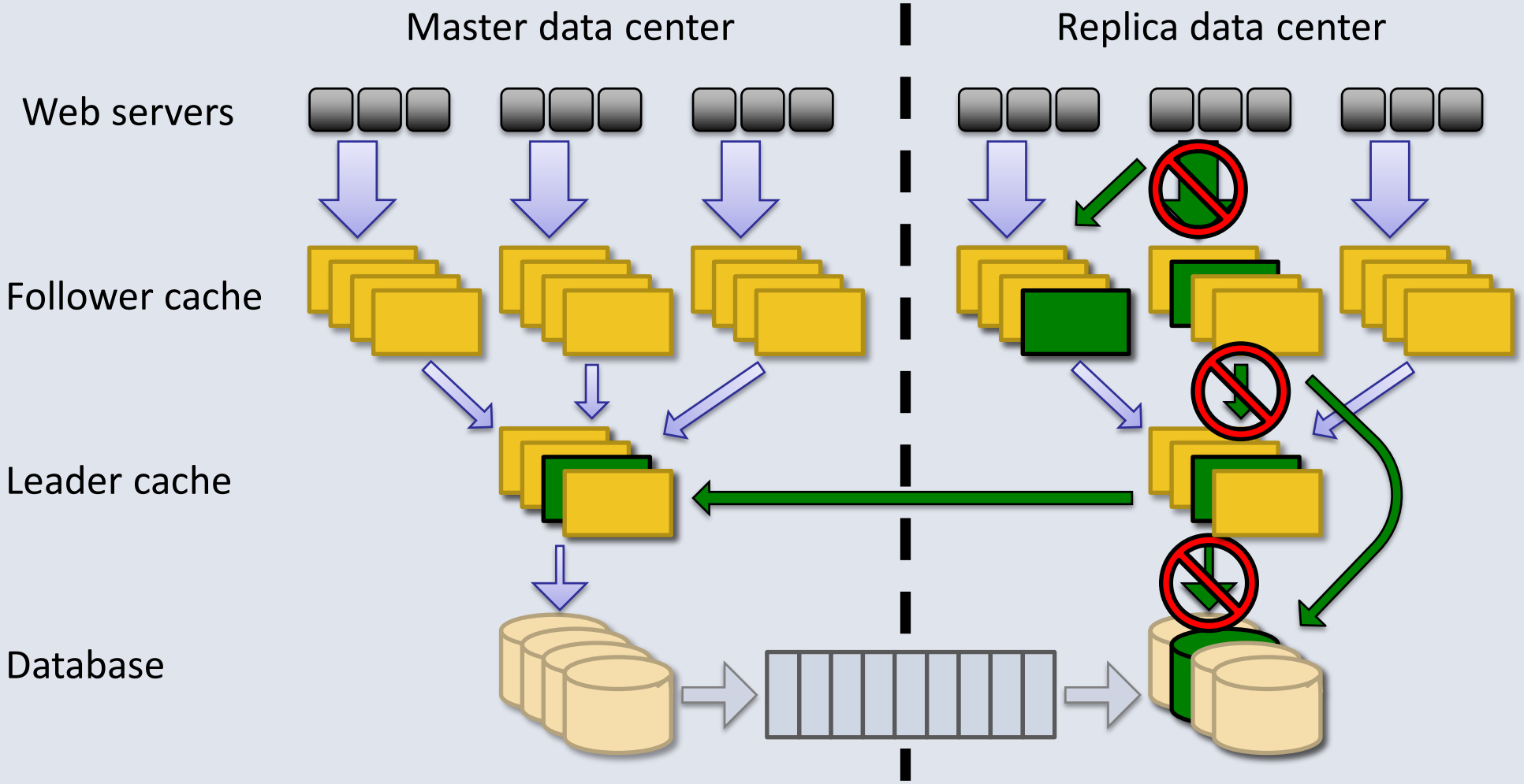
# Key Ideas

- 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.

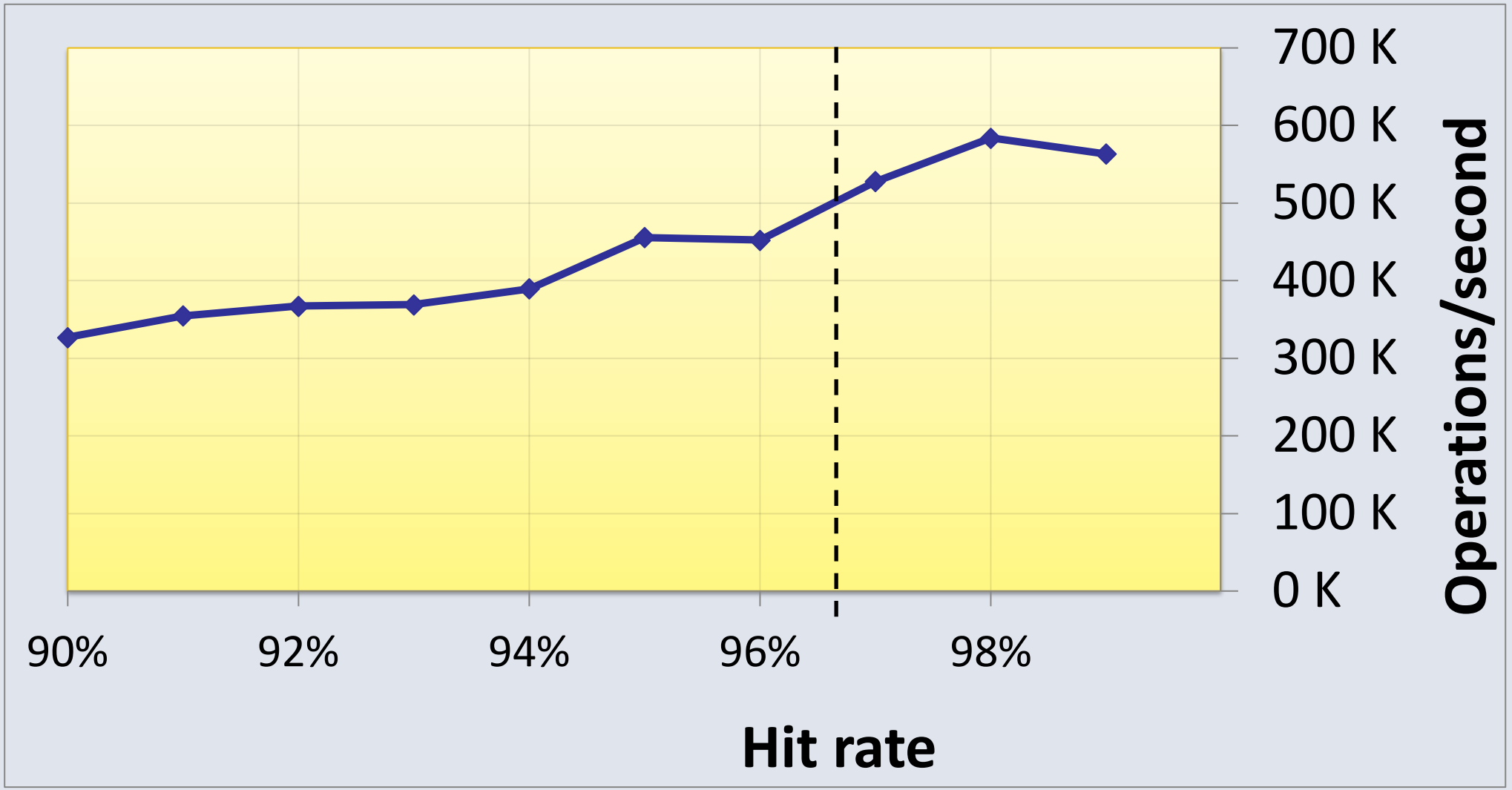
# Improving Availability: Read Failover



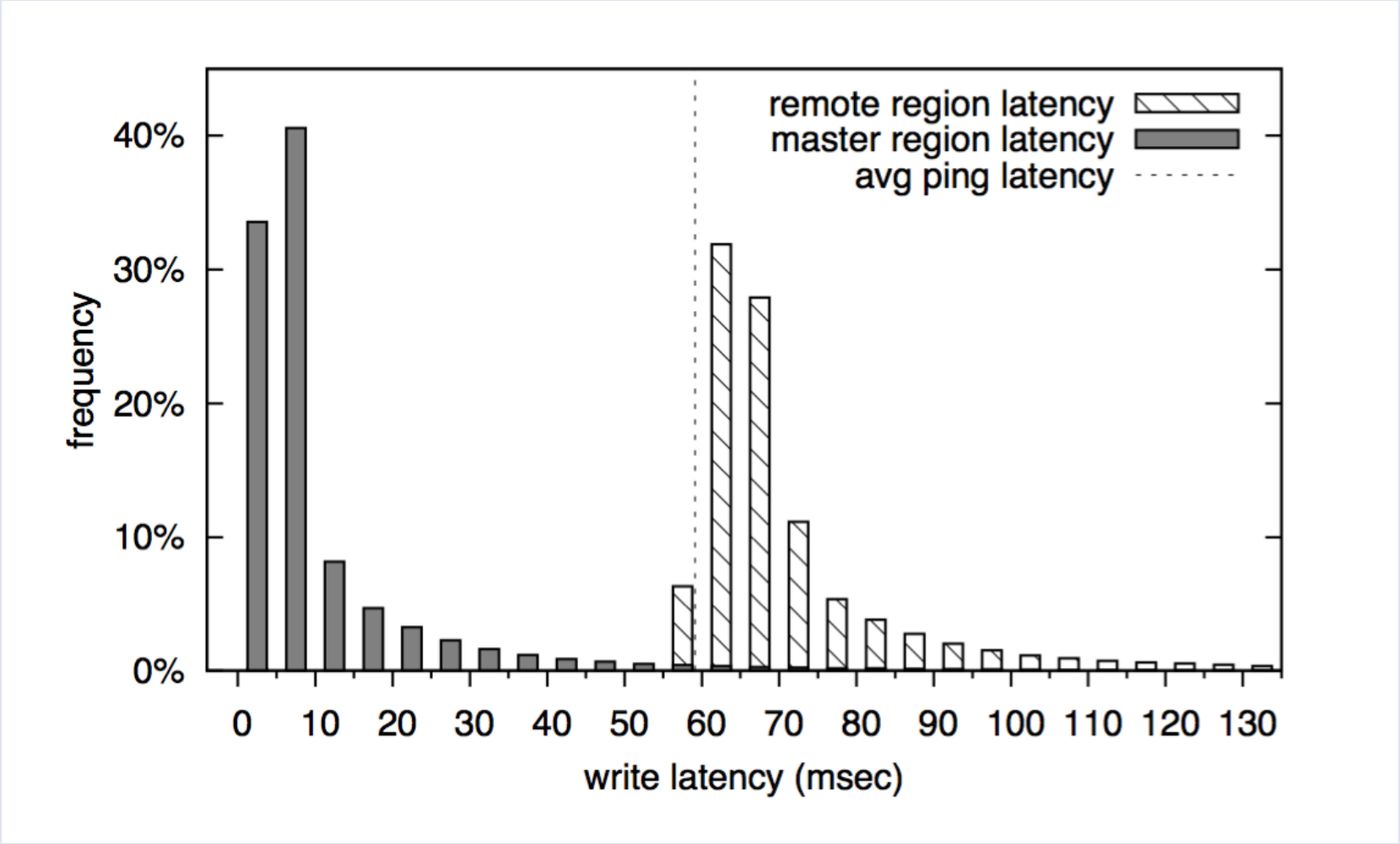
# TAO Summary

| | |
|---|---|
| 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 Cache Node Peak Observed Capacity

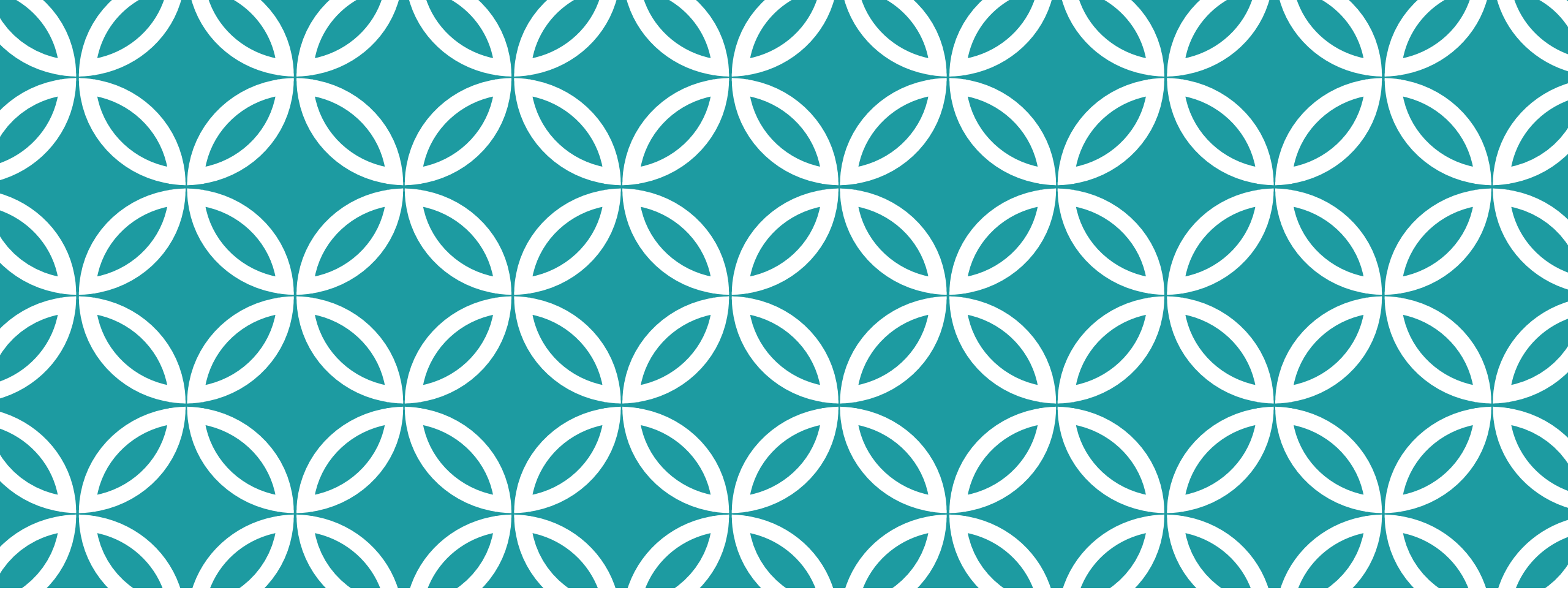


# Write latency



# Not on this slide set

- *Much* more data about performance under various conditions
- The role of association time in optimizing cache hit rates
- Optimized graph-specific data structures
- Write failover
- Failure recovery
- Workload characterization



# WHAT WILL COME NEXT?

(End-of-TAO)

# AS WE LOOK BEYOND SPARK AND TAO…

TAO showed us that the Hadoop style of computing on sharded data isn't really enough.

We also need a data ingress infrastructure optimized for the expected workload and customized for the specifics of the target setting.

Because Facebook has such a social-network centrism, it makes sense that the company would innovate on social network graph infrastructure tools

# THE EDGE AND "REAL WORLD" NETWORKS

Edge computing will demand new architectures and tools, as well!

Up to now we have treated edge computing as if it would be mostly a job for standard big-data analytics.

But in fact edge applications have a "social structure" of their own, emerging from the roles the various elements are playing

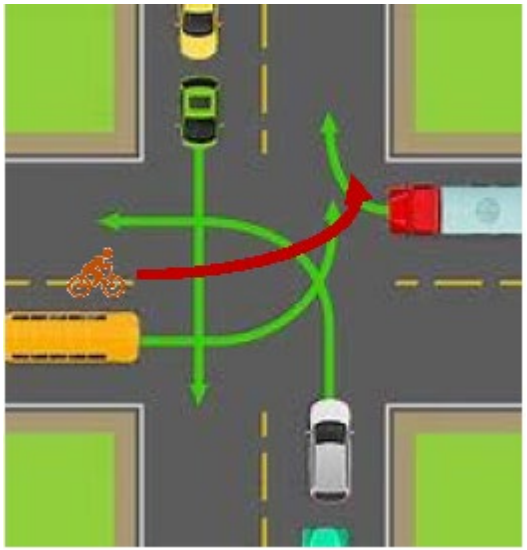
# THINK OF SMART CARS IN A MODERN CITY

Each has its own on-board computing infrastructure and knowledge of its current task, goals, plans.  Even true of non-self-driving cars!

Around it are smart urban support solutions, like smart traffic intersections and people carrying smart devices

The next generation of applications will need to put those together to create high value for consumers (and high value companies to provide those solutions)

# TWO OF MY FAVORITE EXAMPLES



**A smart intersection or roadway must send warnings within 25-30ms!**

# ARE THERE "GRAPHS" HERE?

No, if we mean specifically Facebook TAO social networking graphs

But the information of relevance to a vehicle or a pedestrian is a kind of graph in which the nodes are other entities and the edges represent interactions, and the queries we might use have to run on that graph.

We want to warn "Watch out for that bicycle!" but do that we have to query vehicle paths and proximity and know which ones are impacted

# TAO COULD INSPIRE THOSE NEXT SOLUTIONS

Like TAO they need to be geo-scalable, timely, graph-oriented

One big difference is that we can rapidly discard the past: these smart solutions often care mostly about current state, although they may have learned "characteristics" from past observations (like, "Trinity is an insane motorcycle driver," but "Joe the Uber driver is slow and careful")

We could start with a TAO-like model and build from there…

# WHAT WILL MOTIVATE THE DEVELOPERS?

Facebook is motivated to serve their customers amazingly well but also cost-effectively.

They also earn revenue by placing advertisements in smart ways that lead to click-throughs and purchases.

Future developers need a paycheck. So edge and 5G cellular solutions will align with revenue opportunities. Always ask: "Who pays, and why?"

# CONCLUSIONS: SOCIAL NETWORK 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 edge-computing 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.