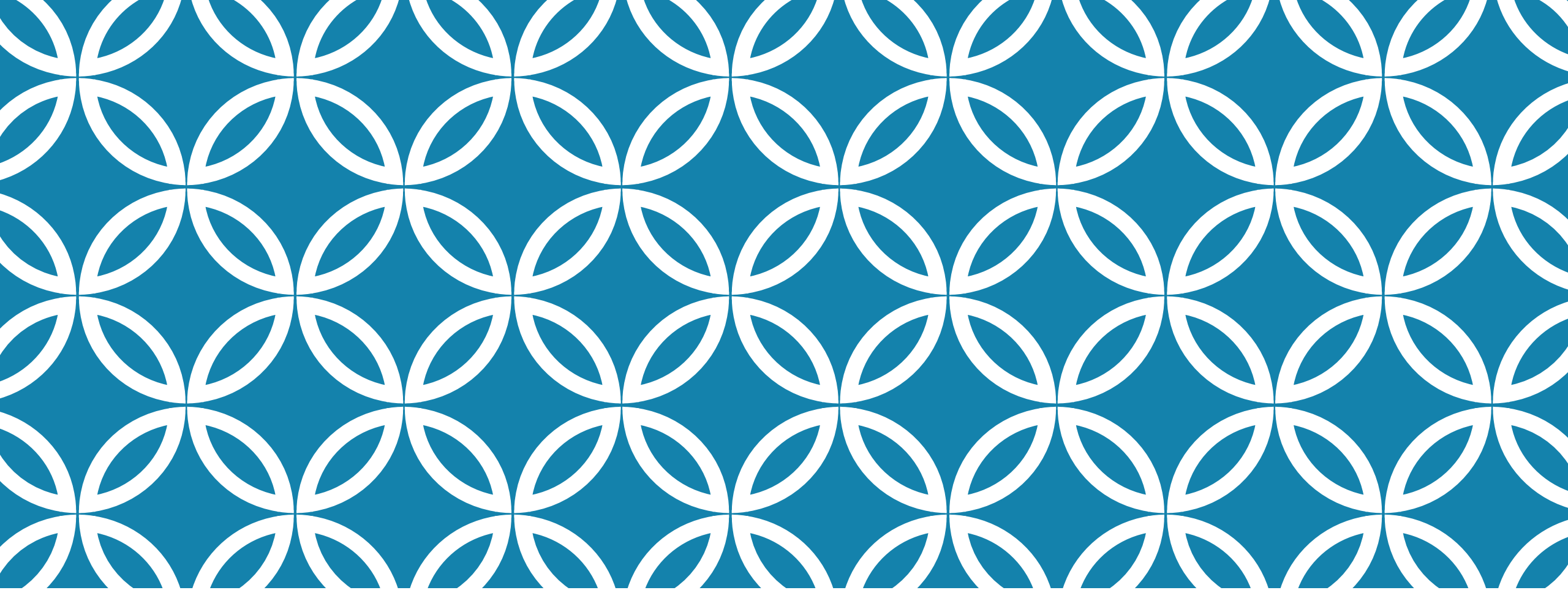


HOW FACEBOOK CACHES AND SERVES IMAGES AND VIDEO DATA

Professor Ken Birman

CS4414/5416 Lecture 17

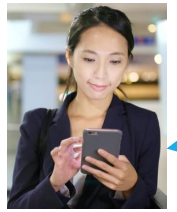
KEY IDEAS FOR TODAY

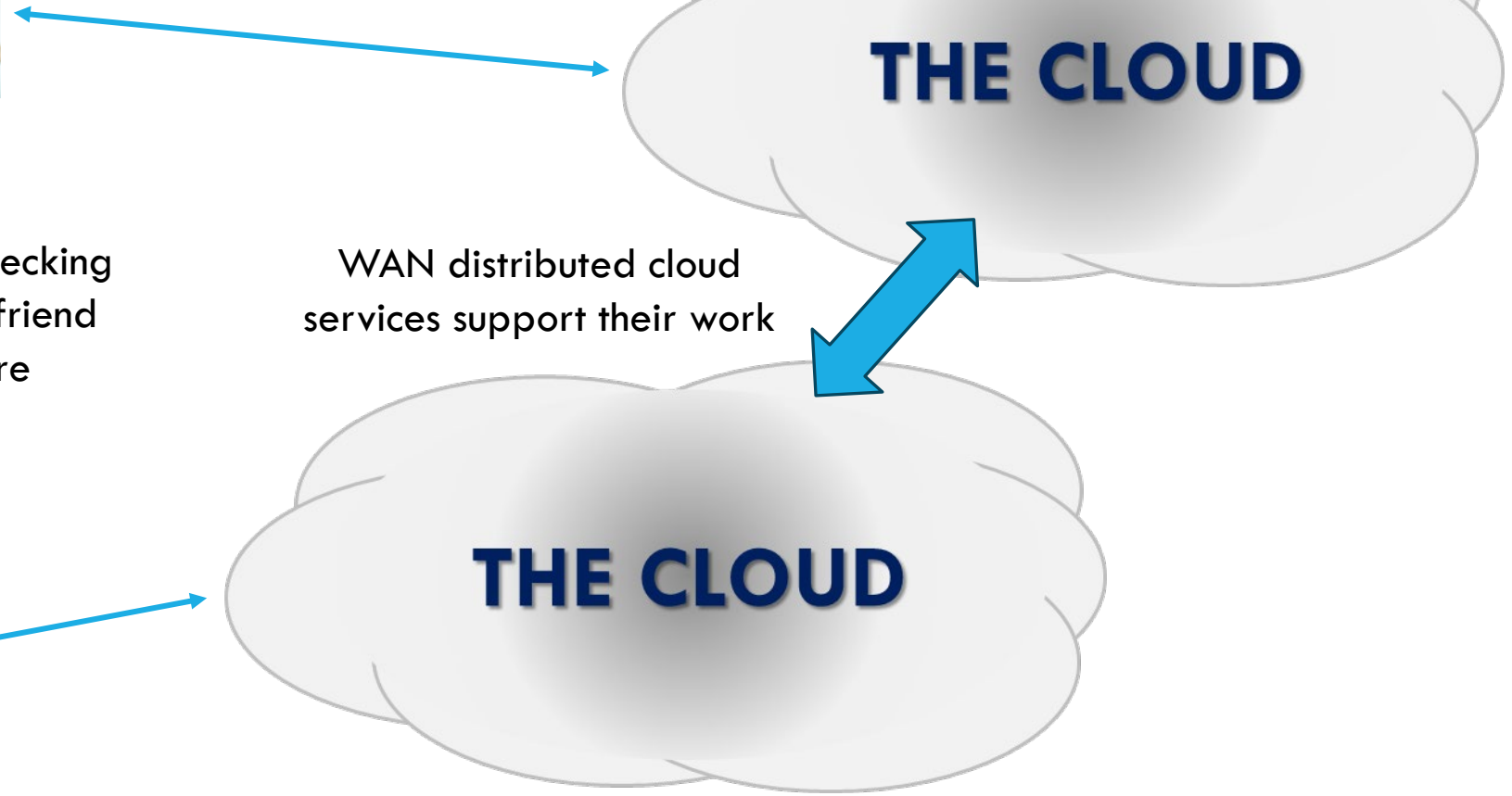
Now we know how data reaches the cloud

… but not how the cloud "serves" data

Today we will focus on Facebook at global scale!

A CLOUD WITH REGIONS





Rohan, based in Seattle, is checking out videos posted by his FB friend Anita, based in Singapore

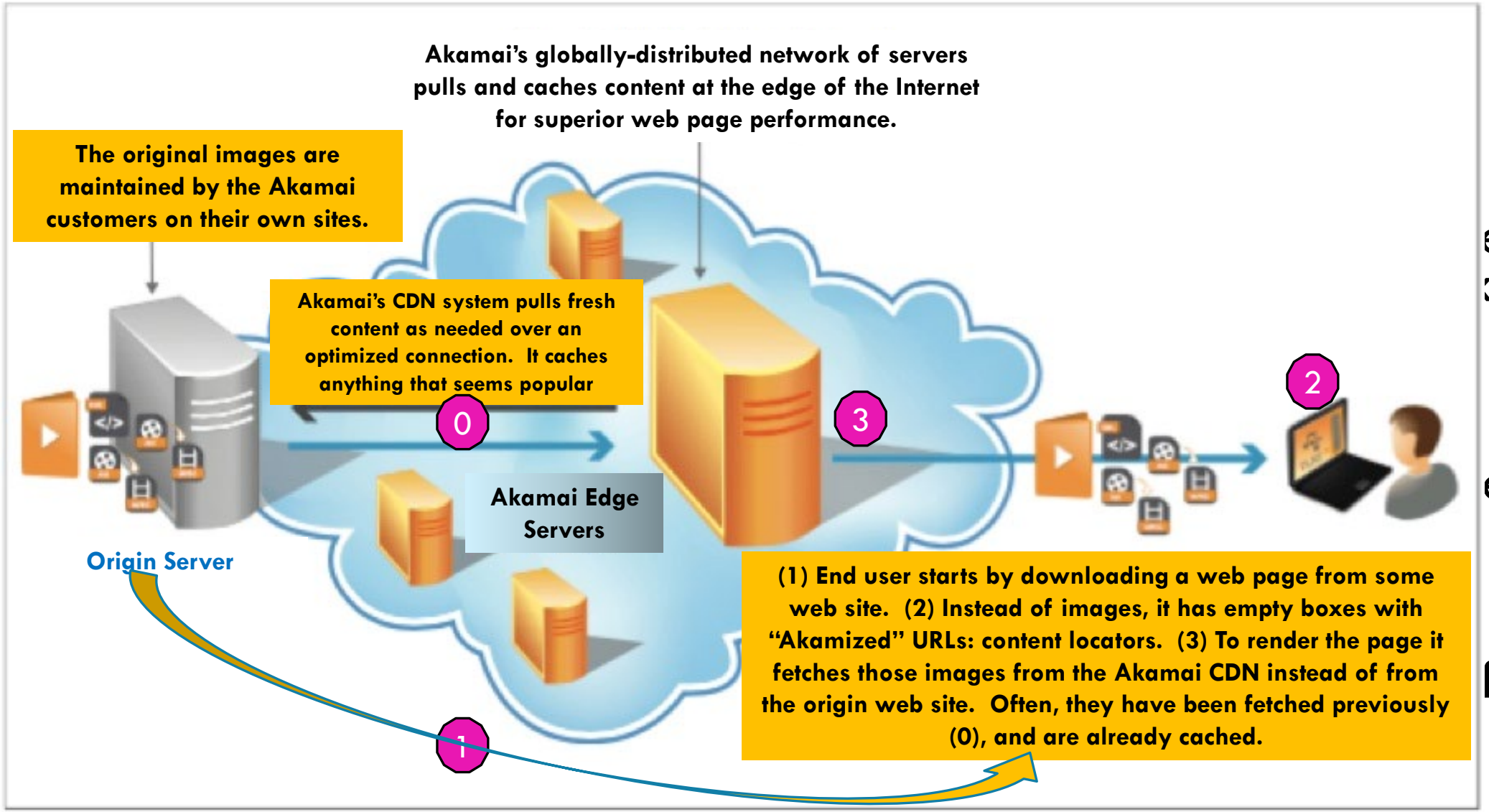
WAN distributed cloud services support their work

CONCEPT: A CONTENT DELIVERY NETWORK

Widely called CDNs

Role is to serve videos and images for end-users

Requirements include speed, scaling, fault-tolerance, self-management


Akamai's globally-distributed network of servers pulls and caches content at the edge of the Internet for superior web page performance.
The original images are maintained by the Akamai customers on their own sites.
Akamai's CDN system pulls fresh content as needed over an optimized connection. It caches anything that seems popular
0
3
2
Akamai Edge Servers
Origin Server
1
(1) End user starts by downloading a web page from some web site. (2) Instead of images, it has empty boxes with "Akamized" URLs: content locators. (3) To render the page it fetches those images from the Akamai CDN instead of from the origin web site. Often, they have been fetched previously (0), and are already cached.

FROM AKAMAI TO FACEBOOK

Akamai is best for relatively stable web page layouts, where we can predict days in advance that certain images or advertisements will be popular in certain places.

Facebook is a big customer of Akamai, one of the largest!

But Facebook is so dynamic that the Akamai solution wasn't (nearly) enough to provide their required performance levels.

FACEBOOK'S REMARKABLE SPEED

Think about the amazing performance of Facebook's CDN …

- You can scan up and down at will, and it renders **instantly**, at the perfect size
- … or search for things, or click links, and **those render instantly too**!

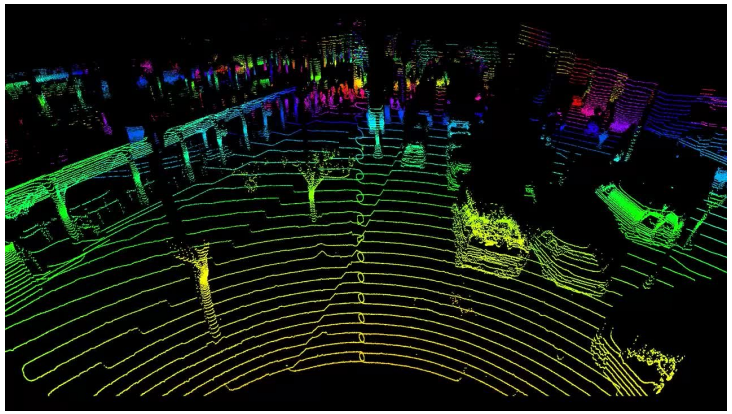
How do they do it?

- Today we will learn about Facebook's global architecture, caching.
- Haystack server, where all the data lives forever.
- TAO, the graph representing its "knowledge" about social networks

… THE DATA IS JUST "BLOBS"

Facebook image data is stored in "blobs": Binary Large Objects

- This includes original images, videos
- Resized versions, and ones with different playback quality
- Versions that have been processed to tag people, *augmented reality*



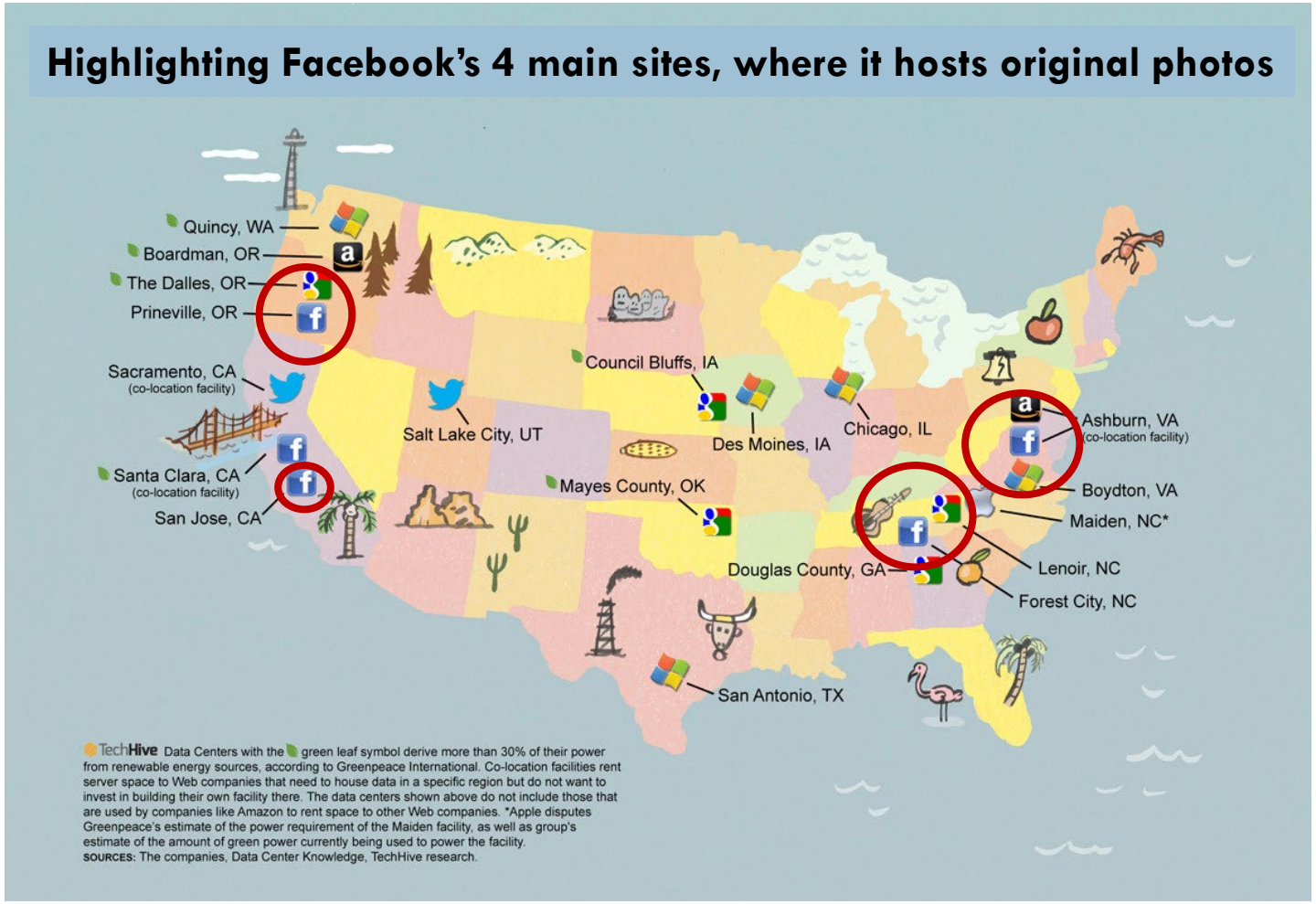
DATACENTERS VERSUS POINTS OF PRESENCE

Facebook doesn't really have a huge number of data centers that hold the main copies of things like image data and the social network graph

But it does have a very large number of smaller datacenters that do tasks like holding cached copies and computing "resized" versions

Called "points of presence". They are datacenters too, but just not as large, and not playing as broad a range of roles.

LARGEST US DATA CENTERS AS OF 2014



HAYSTACK

Holds the real image and video data in huge “film strips”, write-once.

Designed to retrieve any object with a single seek and a single read. Optimized for SSD (these have good transfer rates but are best for write-once, reread many loads, and have a long delay for starting a write).

Facebook doesn’t run a lot of copies

- One on the West Coast, one more on the East Coast
- Each has a backup right next to it.

Main issue: Haystack would easily get overloaded without caching

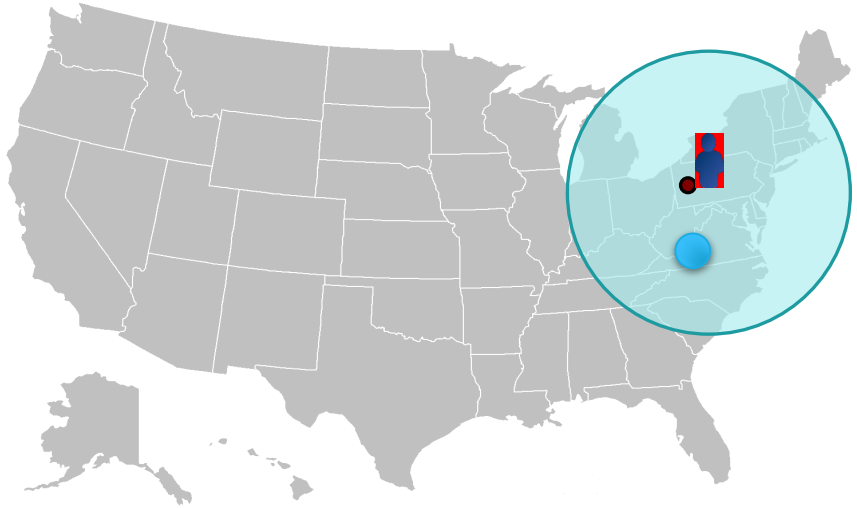
IMPORTANCE OF CACHING

The job of the Facebook image and video (blob) cache is to rapidly find the version of an image or video when needed.

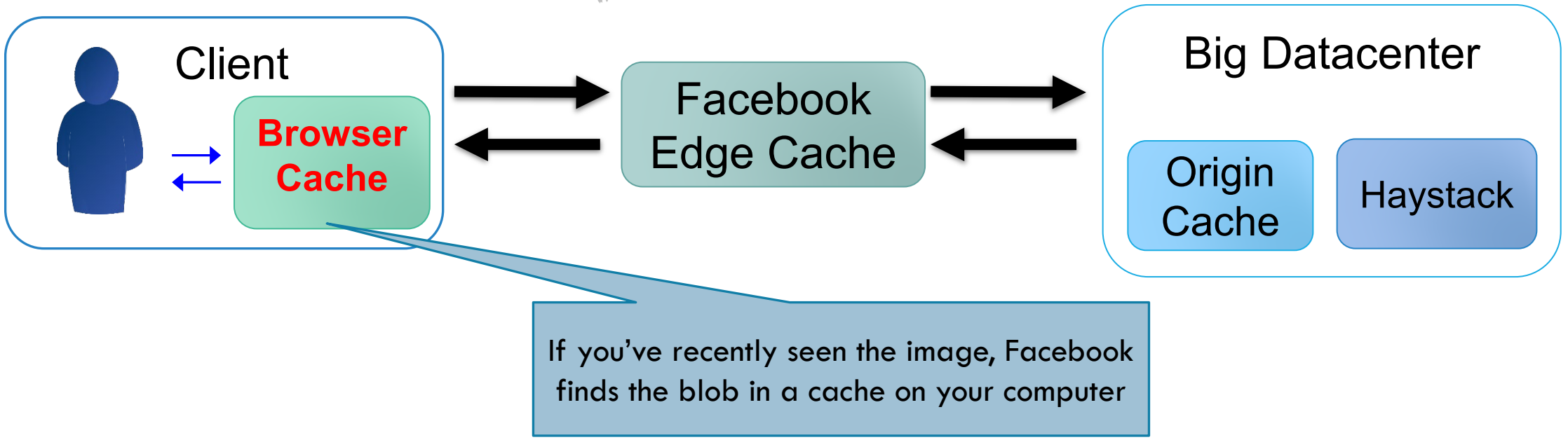
One thing that helps is that once captured, images and videos are *immutable*, meaning that the actual image won't be modified.

- Facebook often computes artifacts from an image, but it doesn't discard original versions. Even deleted images live forever (in the Haystack)
- Thus there is always a recipe to (re) create any needed object. But Facebook would rather not do that unless needed: it wastes resources.

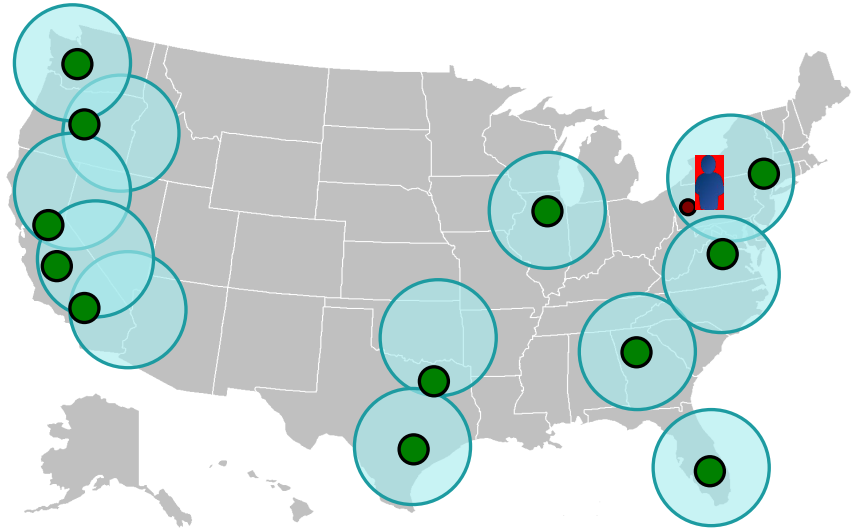
True data center computes personalized content for each user

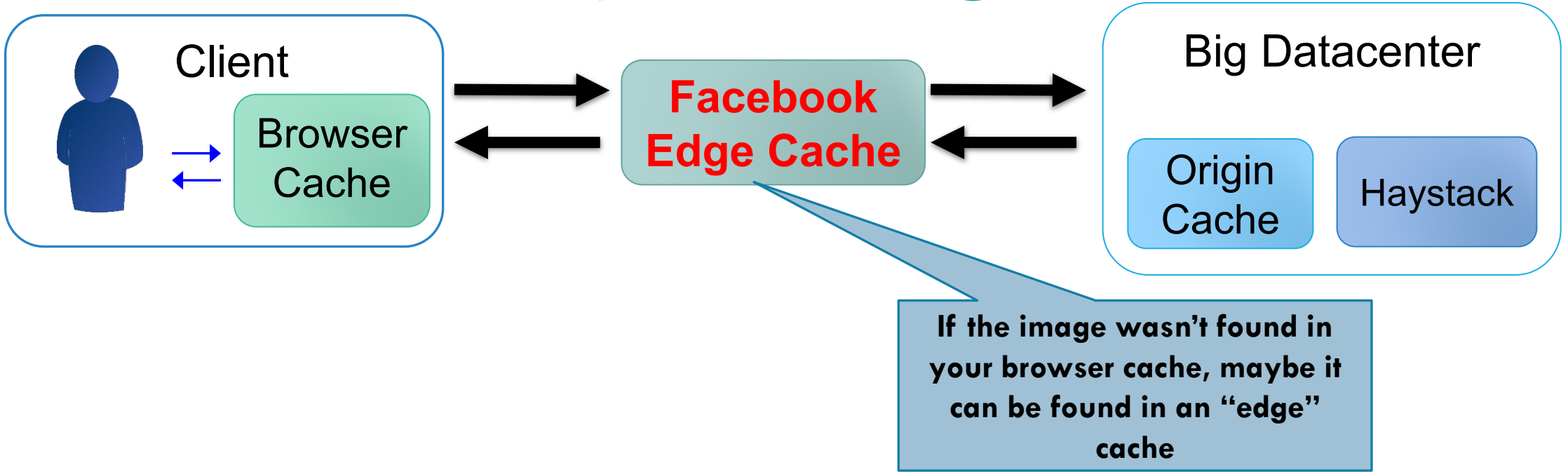


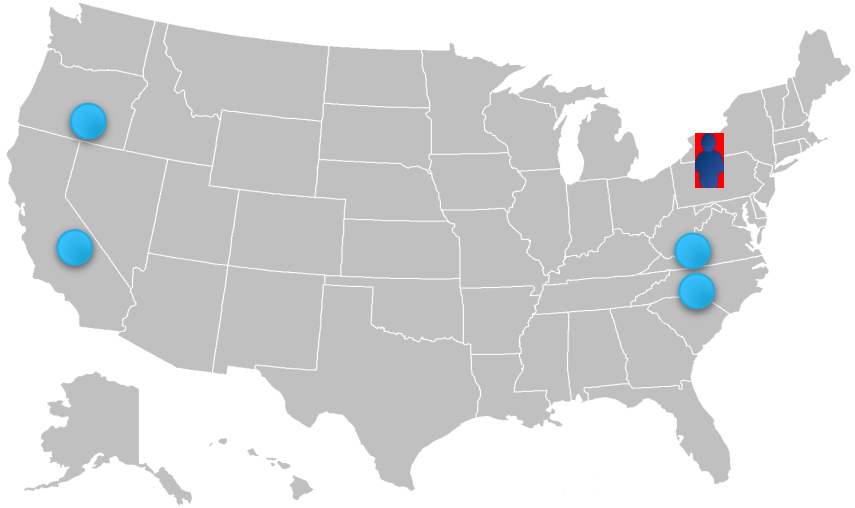
Web page is fetched from an actual data center. It has URLs for all the image content



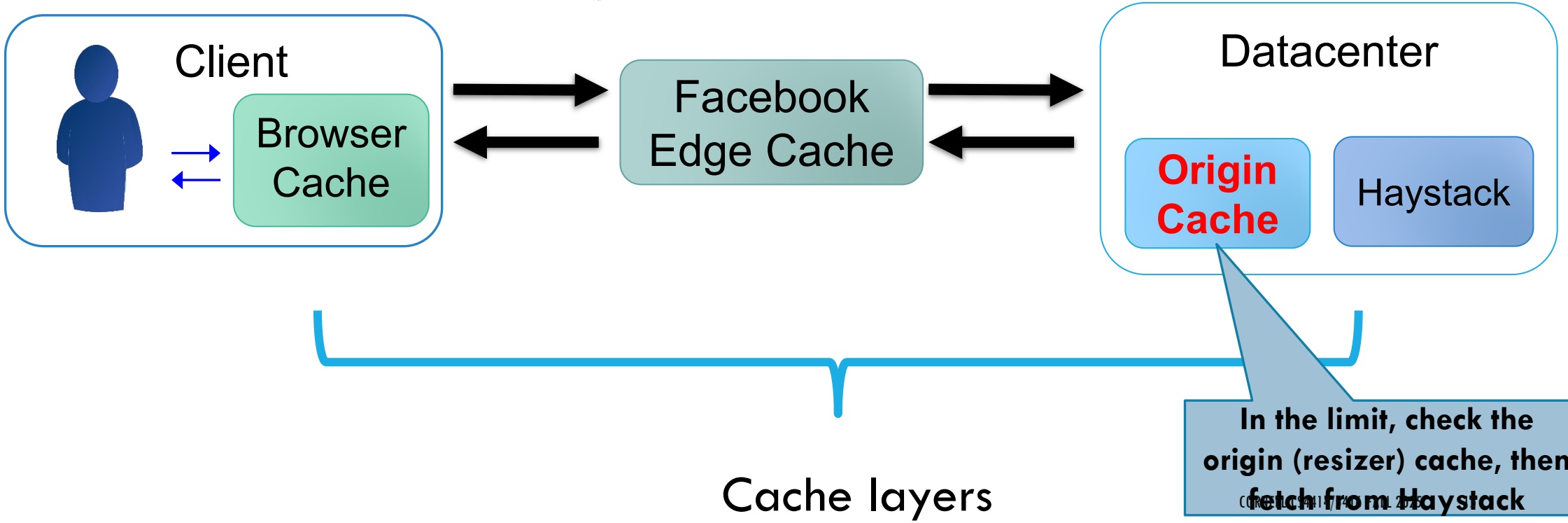
Points of presence:
Independent
FIFO
Main goal:
reduce bandwidth



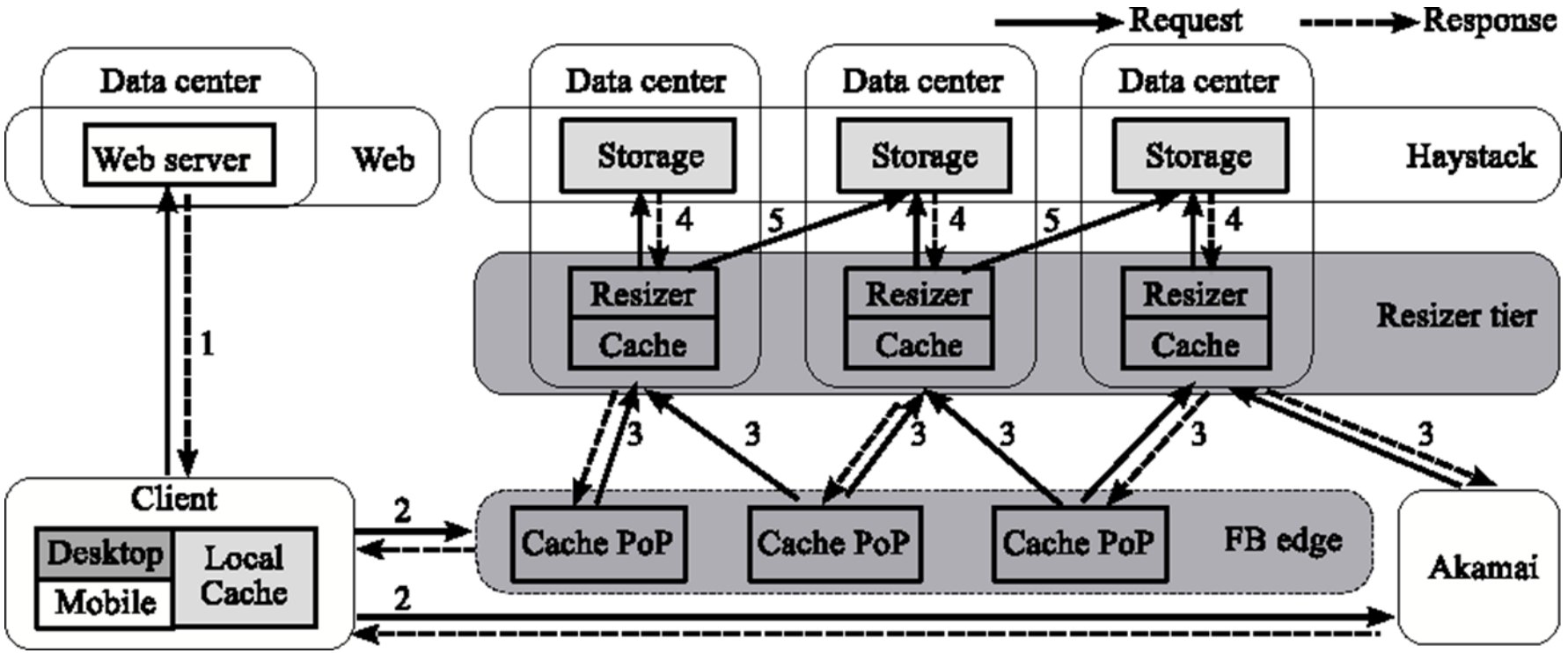




Origin:
Coordinated
FIFO
Main goal:
traffic sheltering



ARCHITECTURAL DETAIL (COMPLEX)
DARK GRAY: WE INSTRUMENTED IT
PALE GRAY: WE CAN FIGURE OUT ITS BEHAVIOR



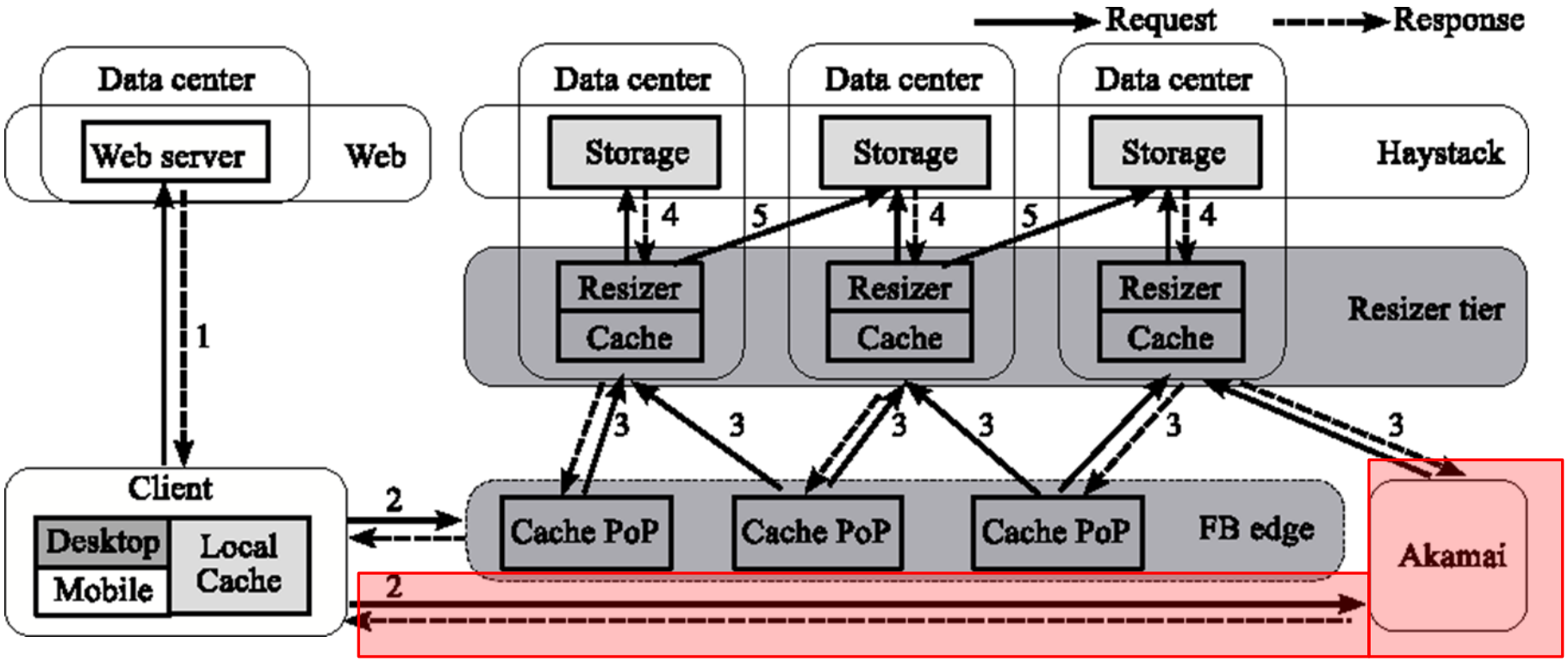
AKAMAI PUZZLE

Facebook creates the web pages (with embedded URLs), hence decides if you will fetch photos via Akamai, and then to the Facebook origin sites, or via Facebook "directly", without checking Akamai first.

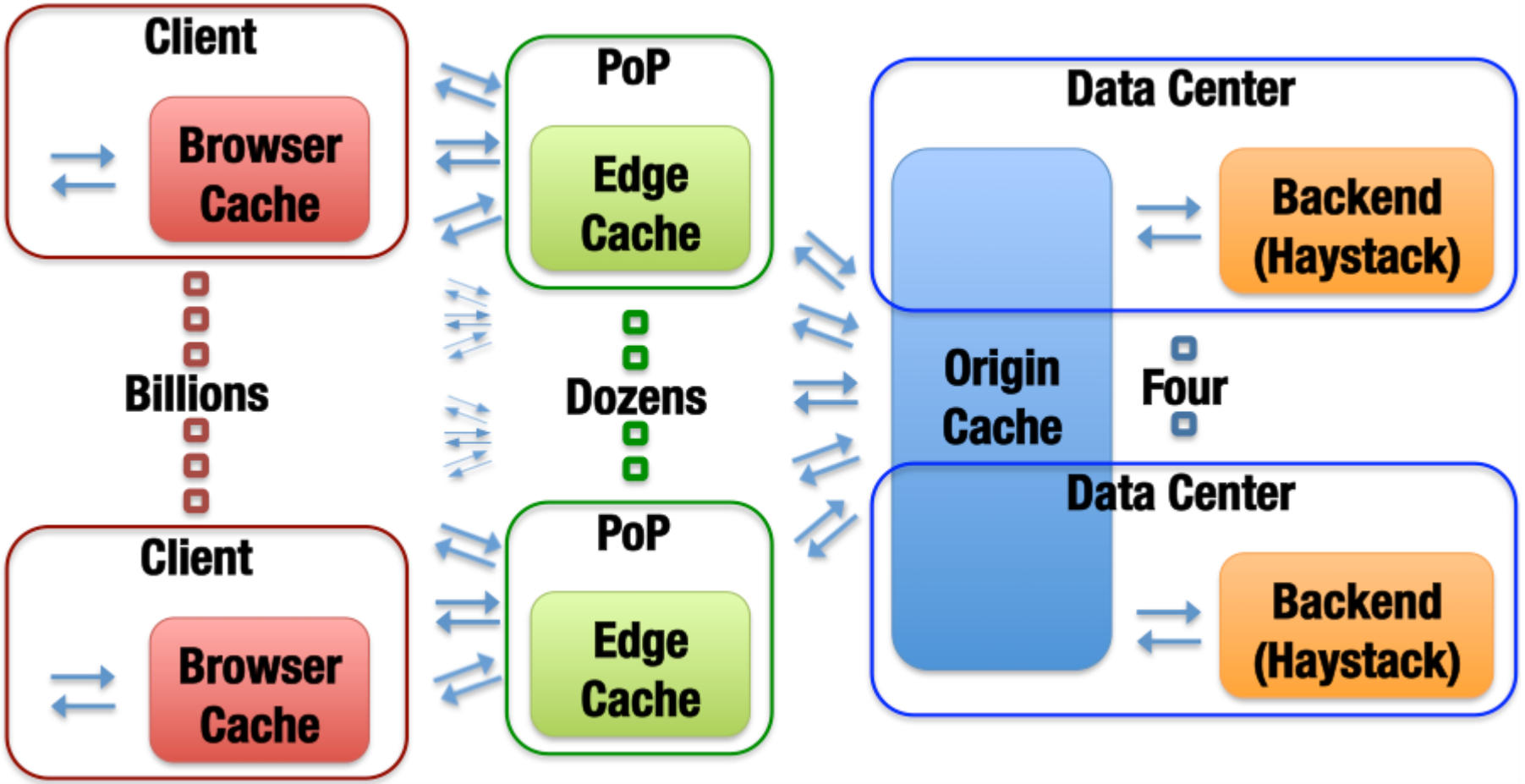
But we have no way to instrument Akamai, and no way to directly instrument the client browser cache. This makes Akamai a black box for us.

There are times when Facebook temporarily doesn't use the Akamai pathway in some regions. We did our experiments during such a period, to eliminate this confusion.

ARCHITECTURAL DETAIL (COMPLEX)

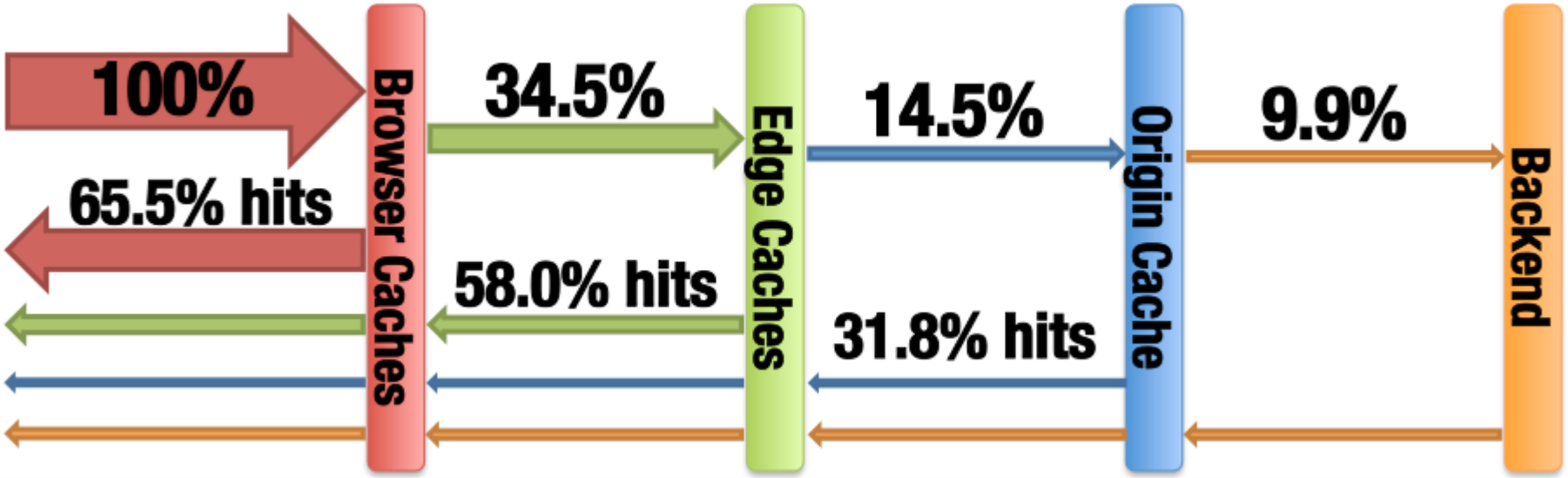


BLOB-SERVING STACK (AGAIN)



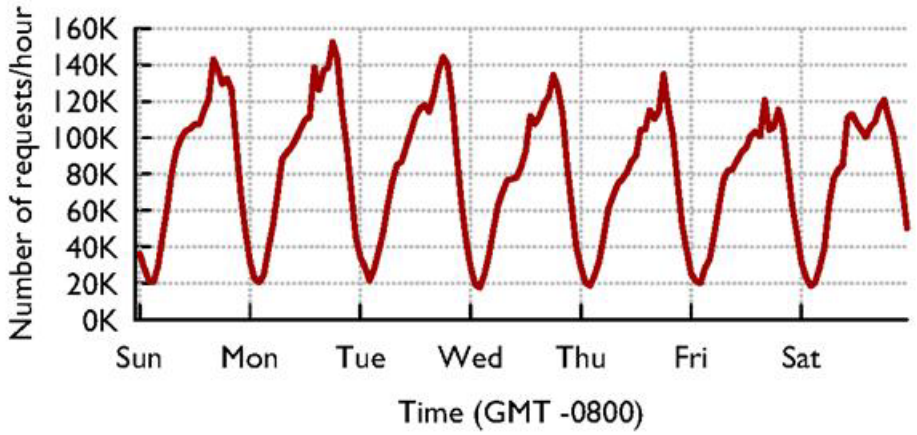
WHAT WE OBSERVED

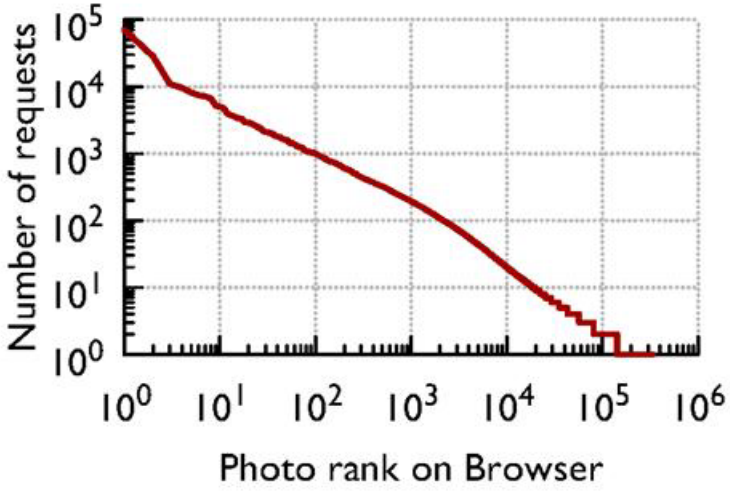
Month long trace of photo accesses, which we sampled and anonymized. Captures cache hits and misses at every level.



CACHES SEE "CIRCADIAN" PATTERNS

Accesses vary by time of day…



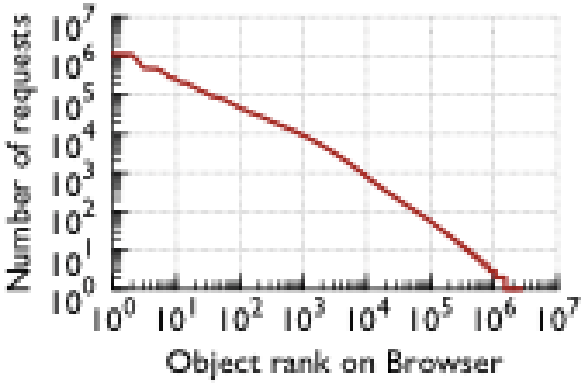


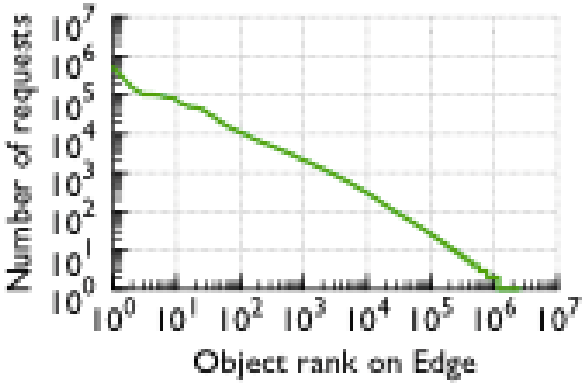
… and by photo: Some are far more popular than others

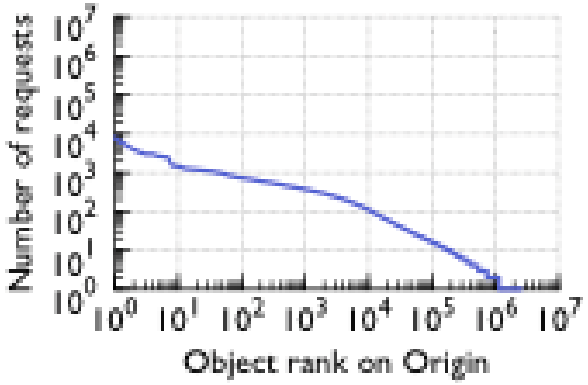
DIFFERENT CACHES HAVE SLIGHTLY DIFFERENT POPULARITY DISTRIBUTIONS

The most popular objects are mostly handled in the outer cache layers

Suggests that one caching policy *might* suffice, but it will need to be configured differently for each layer.







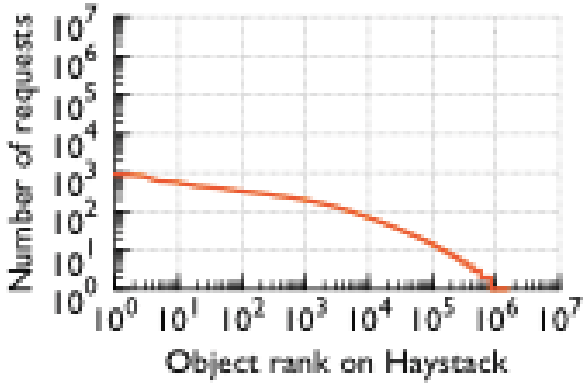
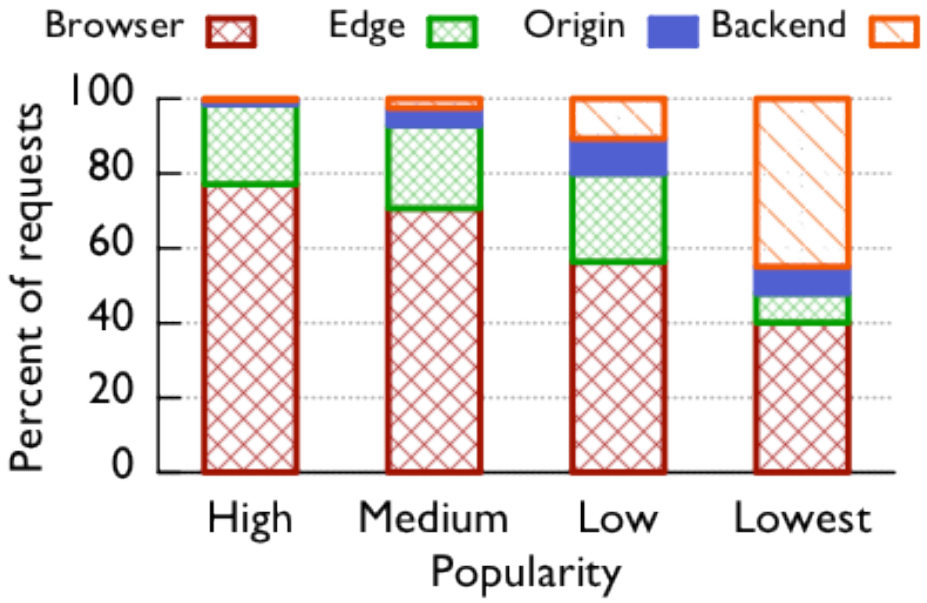


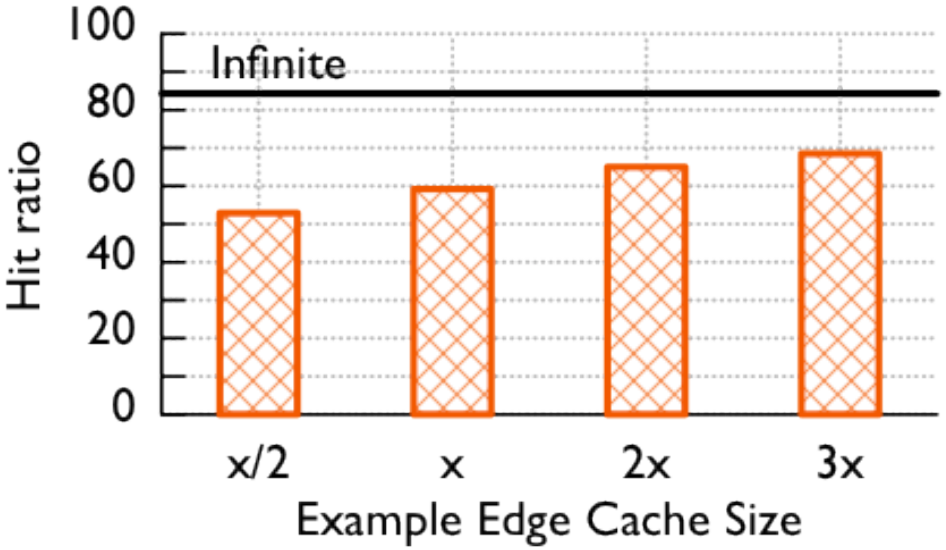
PHOTO CACHEABILITY BY POPULARITY

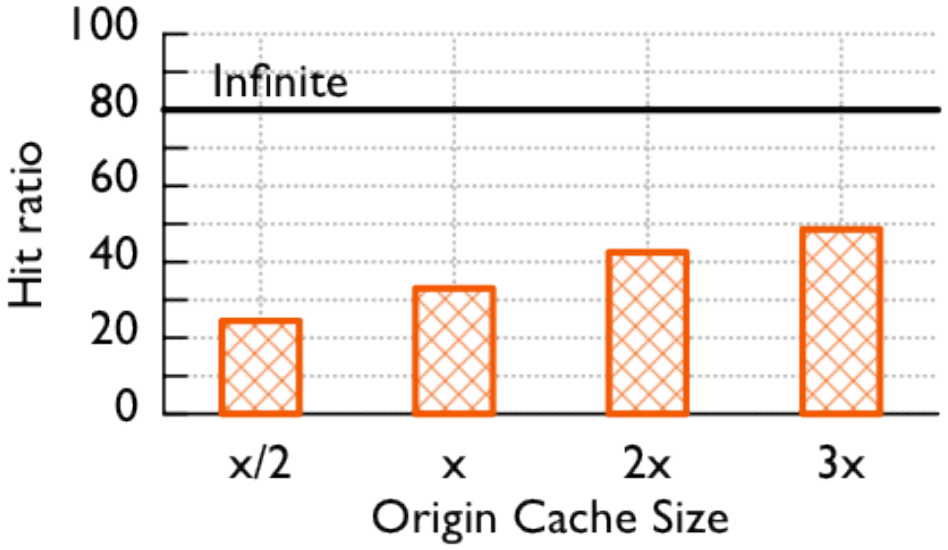
The main job of each layer is different.

This is further evidence that cache policy should vary to match details of the actual workload



LARGER CACHES ARE BETTER, BUT ANY REALISTIC SIZE IS STILL FAR FROM IDEAL





GEO-SCALE CACHING

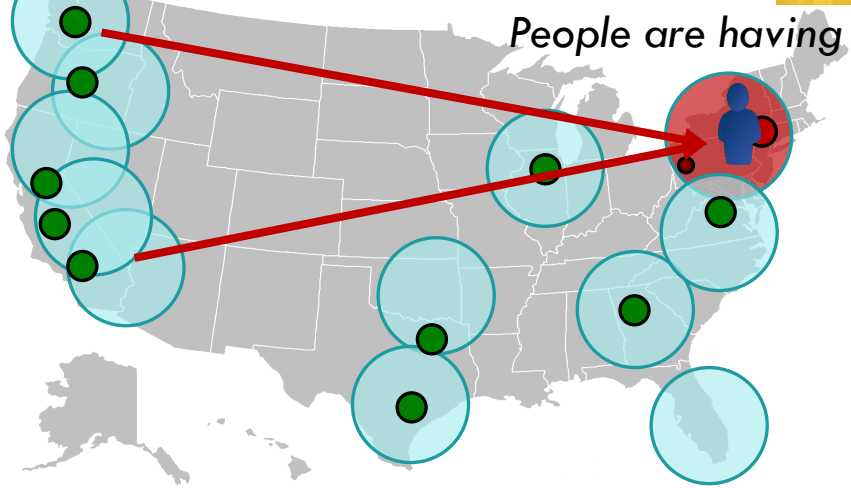
One way to do far better turns out to be for caches to collaborate at a WAN layer – some edge servers may find "flash popular" content earlier than others, and this would avoid regenerating it.

WAN collaboration between Edge caches is faster than asking for it from Haystack, and also reduces load on the Haystack platform.

Key insight: the Facebook Internet is remarkably fast and stable, and this gives better than scaling because the full cache can be exploited.

People are sleeping in Seattle

Remote fetch: cooperation between point-of-presence cache systems to share load and resources



People are having breakfast in Ithaca

In Ithaca, most of your content probably comes from a point of presence in the area, maybe the red one (near Boston)

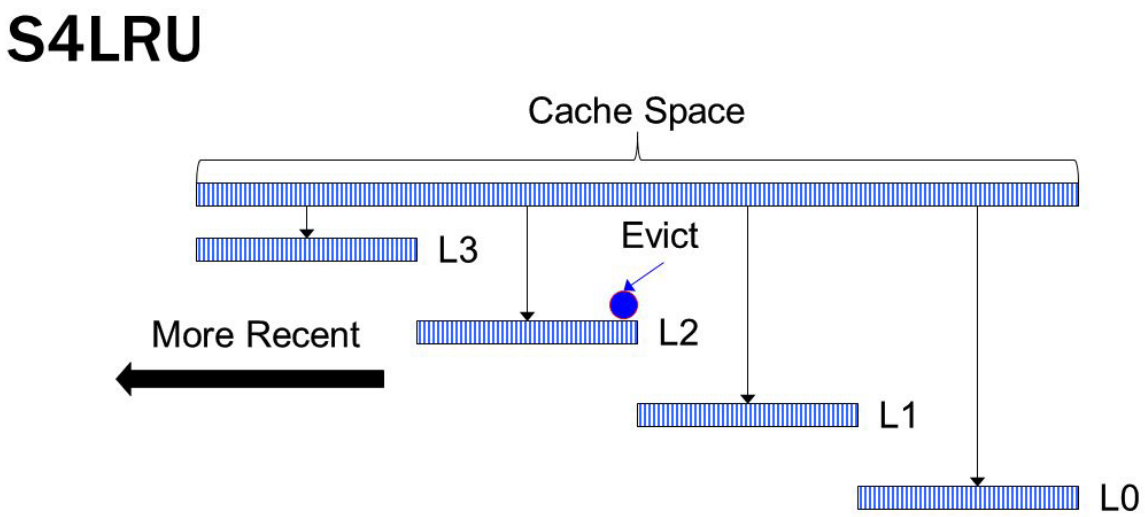
But if Boston doesn't have it or is overloaded, they casually reach out to places like Spokane, or Arizona, especially during periods when those are lightly loaded, like 5am PT!

CACHE RETENTION/EVICTION POLICY

Facebook was using an LRU policy.

We used our trace to evaluate a segmented scheme called S4LRU

It outperforms all other algorithms we looked at



HOW S4LRU WORKS

Popular content

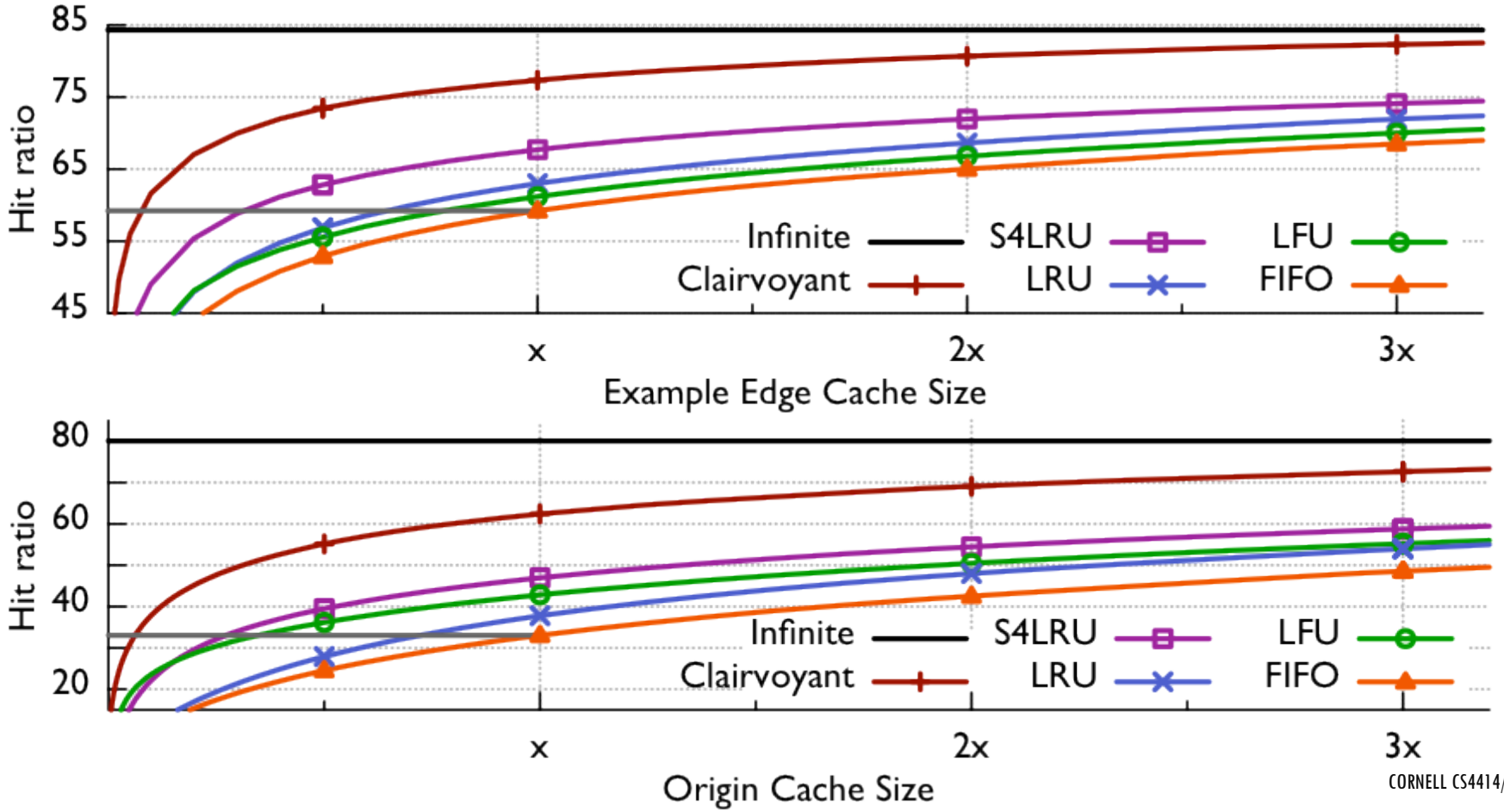
Less popular content

We create multiple slices of the cache

Each slice focuses on workloads with different popularity and duration of "hotness", from transient surges to long term obsession that fascinates the entire Facebook user community

Then we can use LRU in each slice

WITH FACEBOOK'S REAL TRACES, S4LRU PERFORMS BETTER THAN NORMAL LRU



SO… SWITCH TO S4LRU, RIGHT?

They decided to do so…

Total failure!

Why didn't it help?

S4LRU WORKED ON THE TRACE YET DIDN'T WORK WELL ON THE HARDWARE!

It turned out that the algorithm worked well *in theory* but created a pattern of reads and writes that were badly matched to flash memory storage devices of the kind Facebook used.

Resulted in a whole two year project to redesign the "operating system" layer for big SSD disk arrays based on flash memory.

Once this change was made, S4LRU finally worked as hoped!

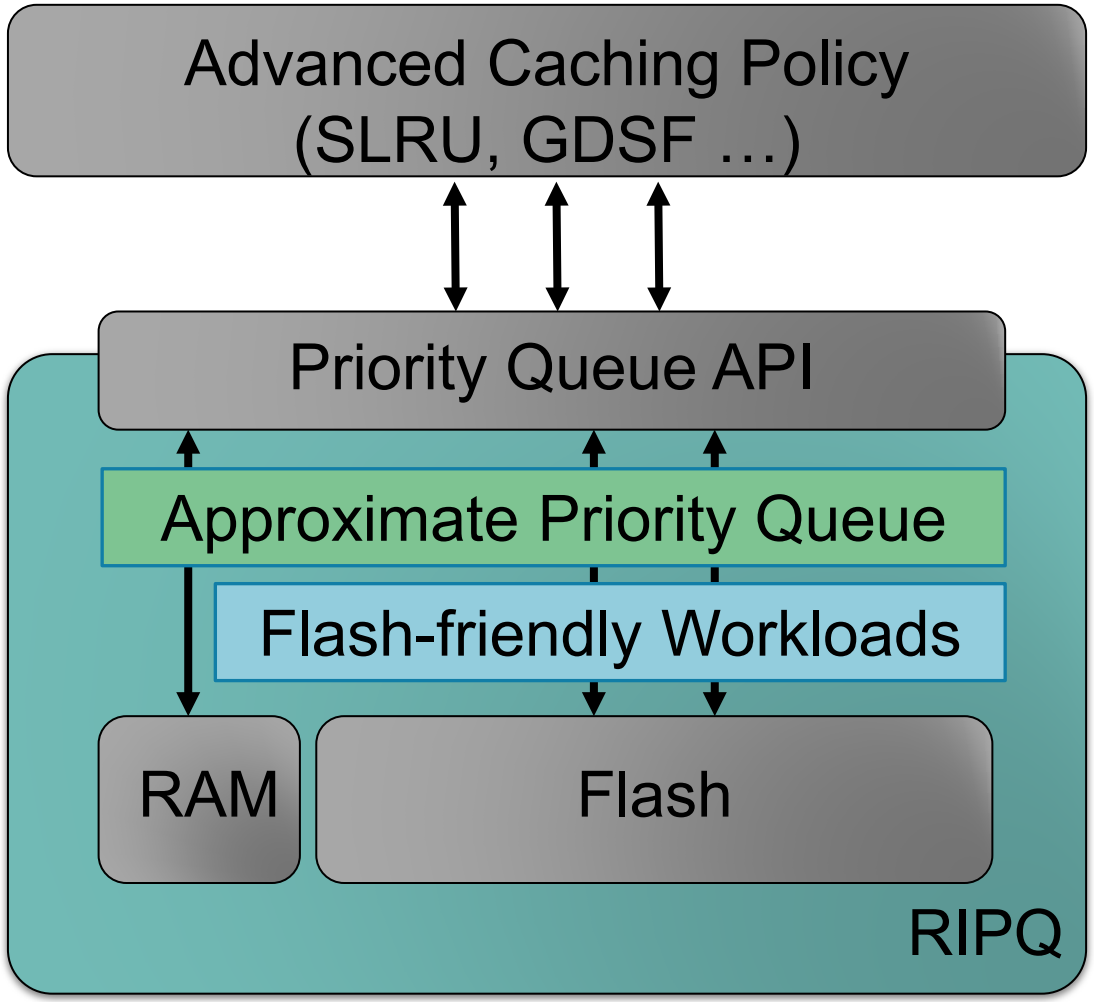
CENTRAL PROBLEM?

The amounts of data involved are huge and do not fit in memory, so even the cache server does a great deal of I/O

SSD disks only are fast if you write long strips of data. But many photos are fairly small. Haystack ran into this too.

Needed to create an S4LRU caching layer that does large I/Os

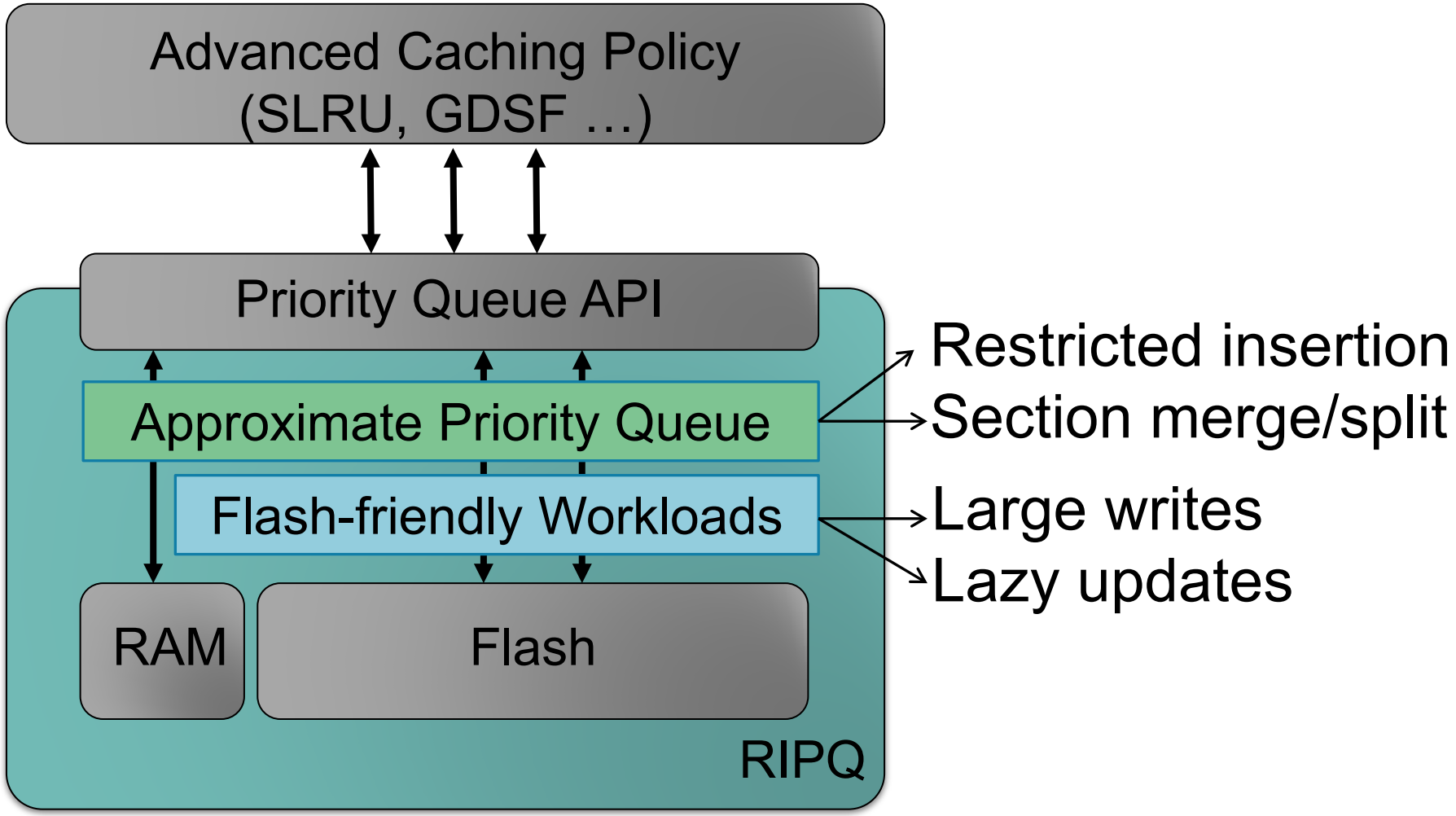
RIPQ ARCHITECTURE
(RESTRICTED INSERTION PRIORITY QUEUE)



Caching algorithms approximated as well

Efficient caching on flash

RIPQ ARCHITECTURE
(RESTRICTED INSERTION PRIORITY QUEUE)



START BY UNDERSTANDING THAT LRU IS BASED ON PRIORITY QUEUE DATA STRUCTURES

A priority queue is a list of items ordered using a sorting method

For example, in Facebook, an LRU would be a list of cached items and the sort order would be “time of most recent use”

With this priority queue, new items go to the front and evictions occur from the tail.

WHY S4LRU CREATES A PUZZLE

First, keep in mind that Facebook operates at **huge** scale

The S4LRU computer memory is totally filled just with pointers to content.

The actual image content lives on flash-memory disks, not in memory. It will be read from disk when a cache-hit occurs, and is written to disk when the first cache retention event occurs

WHY S4LRU CREATES A PUZZLE

Next, in S4LRU, items need an access count, not just time. And the sort order actually includes both attributes

When an item moves from layer to layer in S4LRU, it moves along with this history of access

Instead of being inserted at the front, an item might need to be inserted elsewhere in the LRU list of the next level up

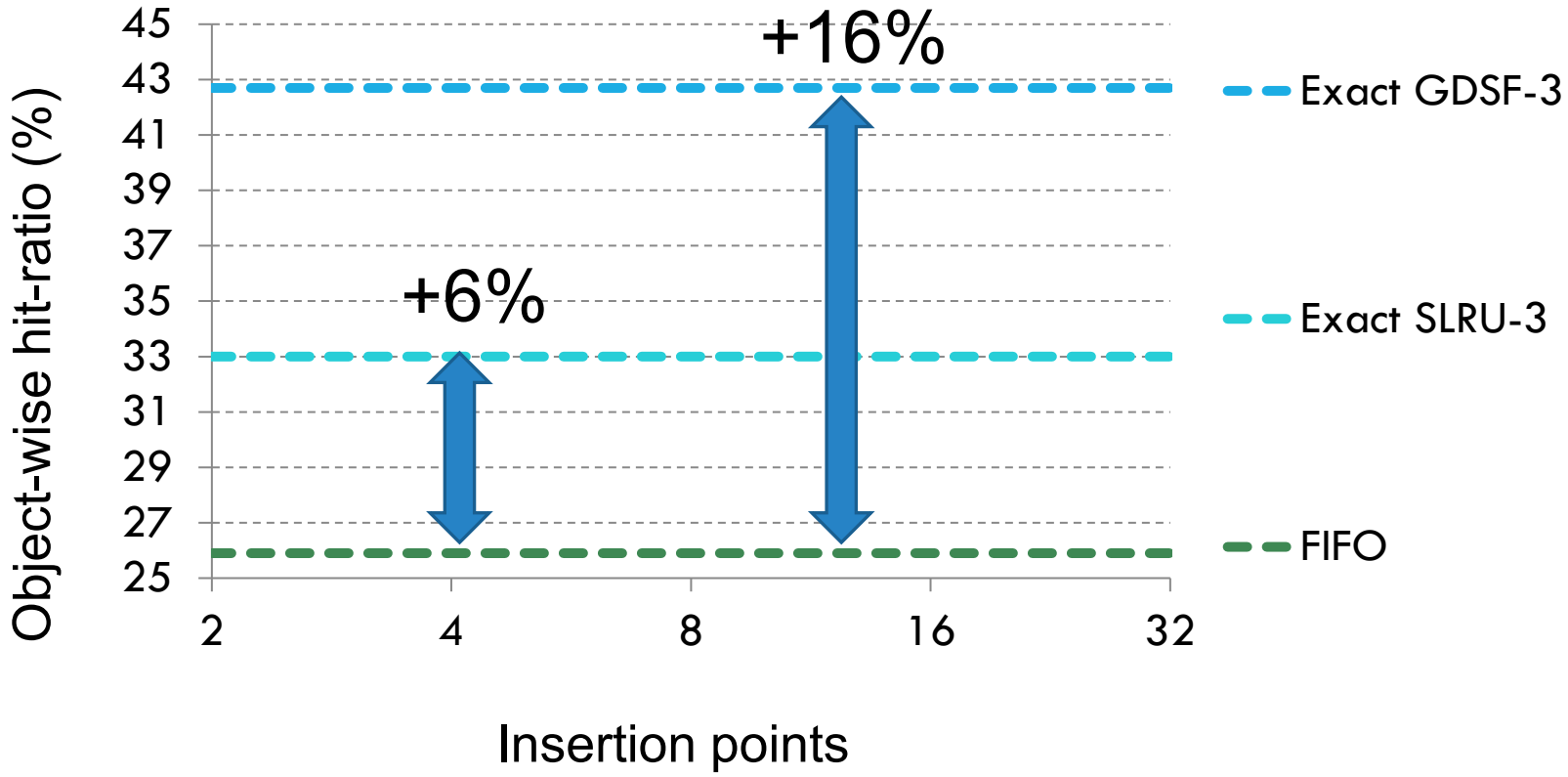
WHAT IS A "RESTRICTED INSERTION POINT"?

The **restriction** in RIPQ is on **where new items can be inserted** into the priority queue. Instead of allowing arbitrary insertions (which would lead to many small random writes), RIPQ restricts insertions to a **limited set of locations**. This design choice enables:

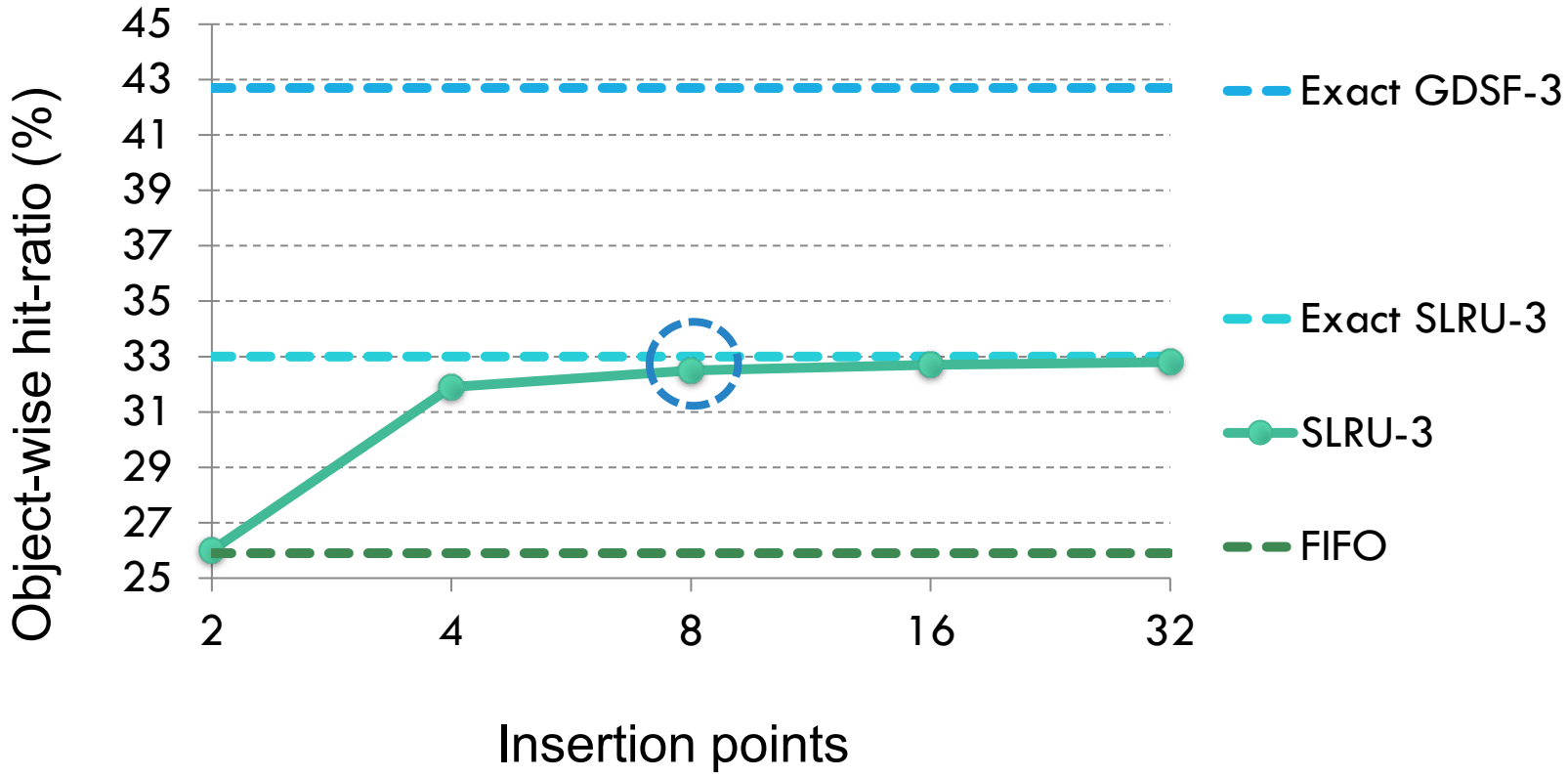
- **Aggregation of small writes** into larger, more flash-friendly operations.
- **Co-location of similarly prioritized content**, reducing fragmentation.

  Two items at the same S4LRU cache layer will be in the same strip of memory on the flash memory device
- **Lazy movement of updated content**, minimizing write amplification and garbage collection overhead.

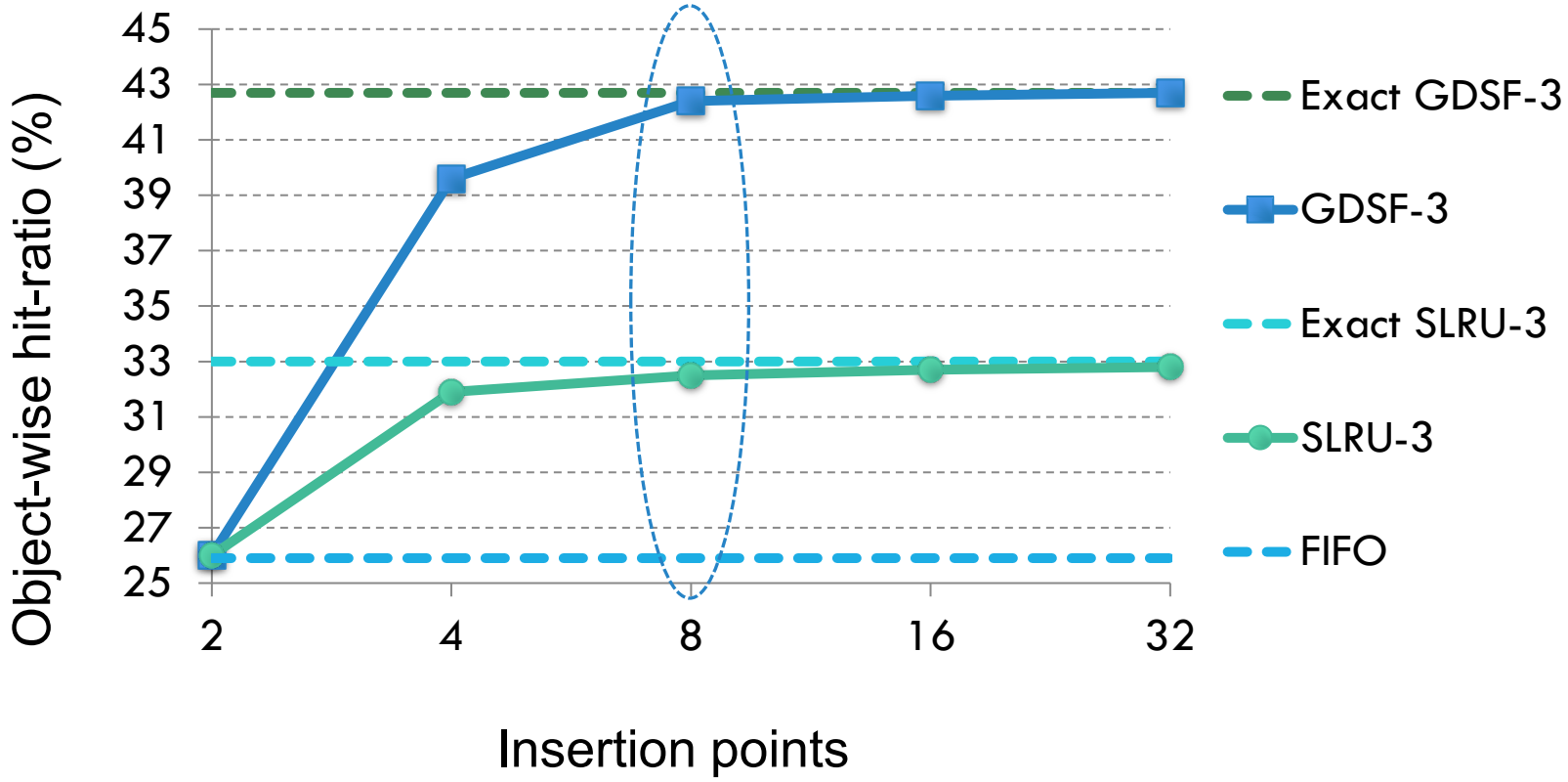
RIPQ NEEDS SMALL NUMBER OF INSERTION POINTS



RIPQ NEEDS SMALL NUMBER OF INSERTION POINTS

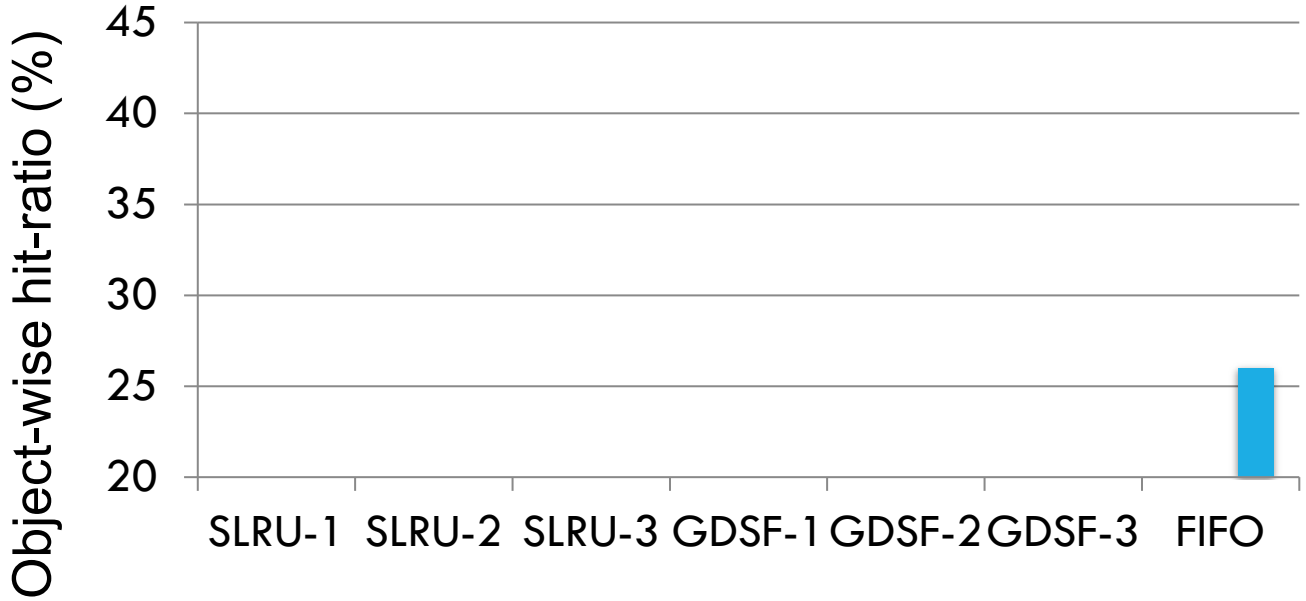


RIPQ NEEDS SMALL NUMBER OF INSERTION POINTS

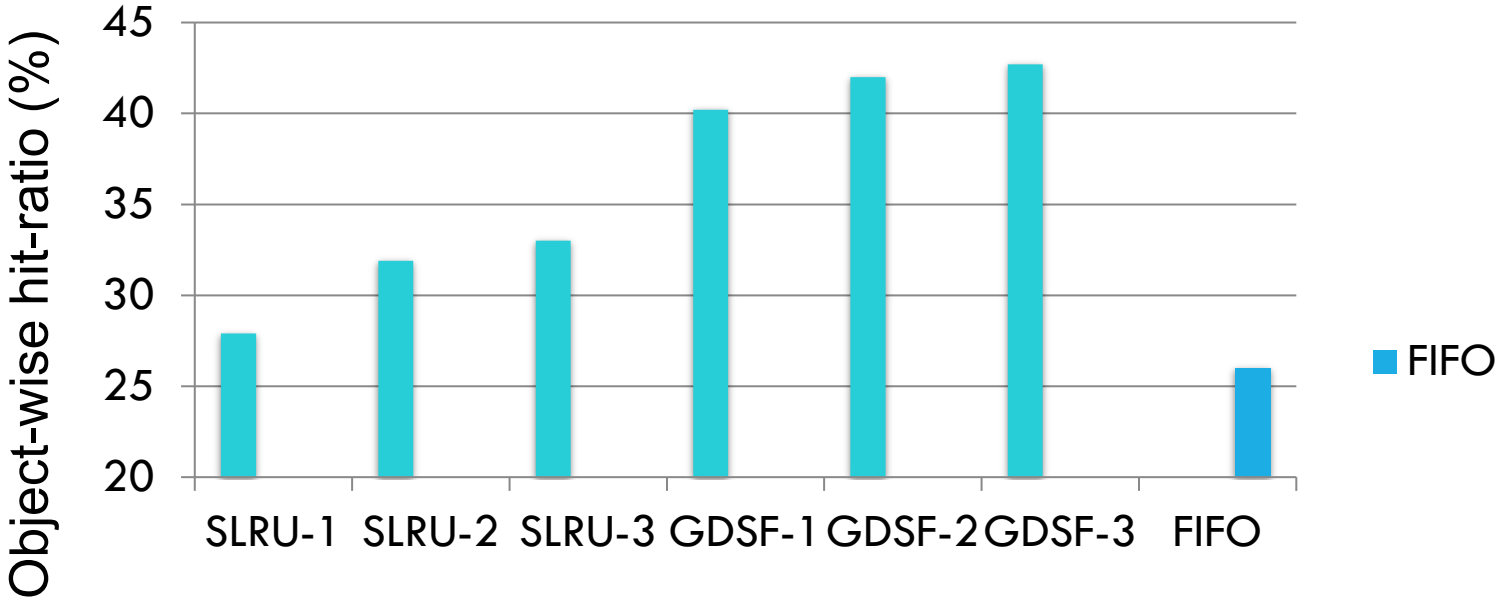


You don't need much RAM buffer (2GiB)!

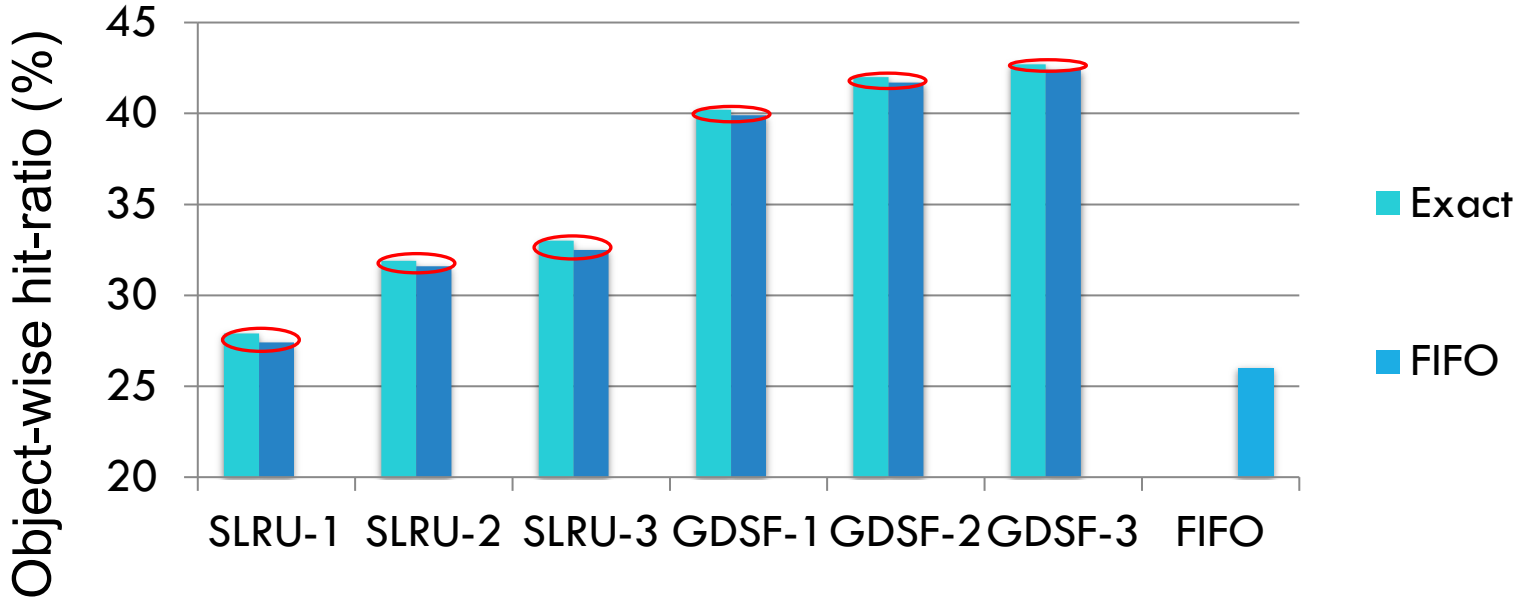
RIPQ HAS HIGH FIDELITY



RIPQ HAS HIGH FIDELITY

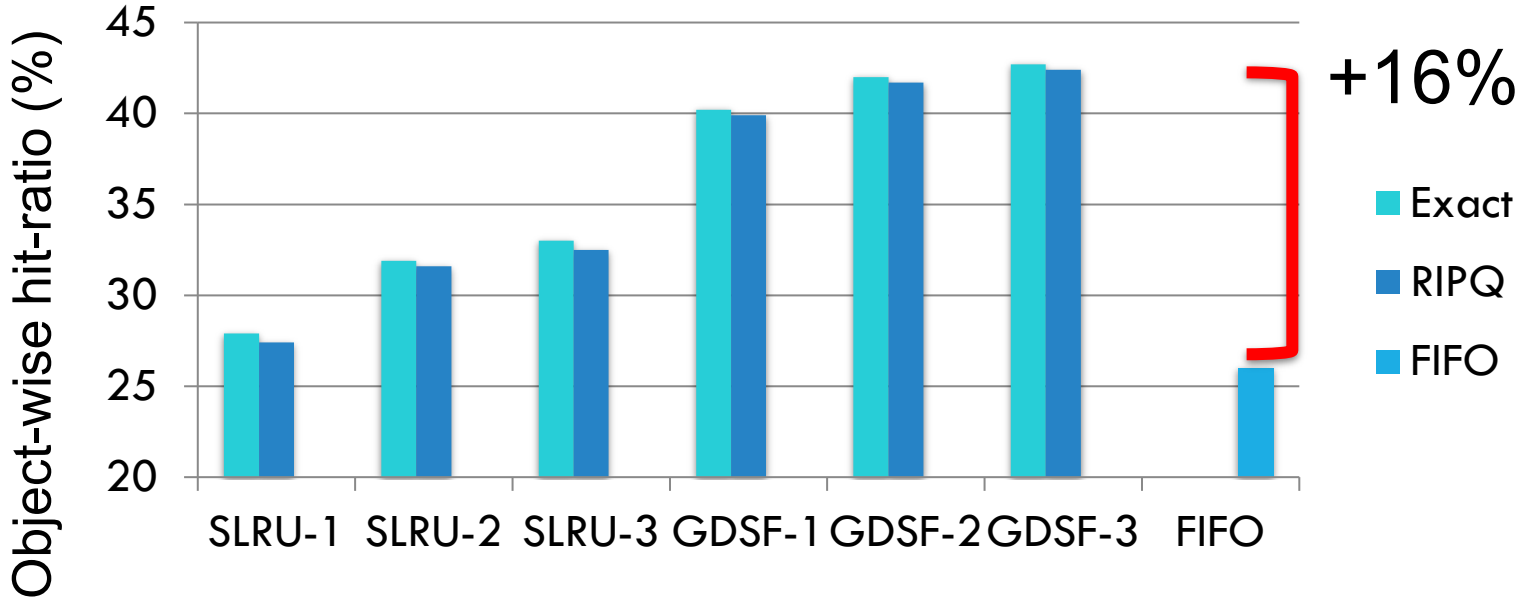


RIPQ HAS HIGH FIDELITY



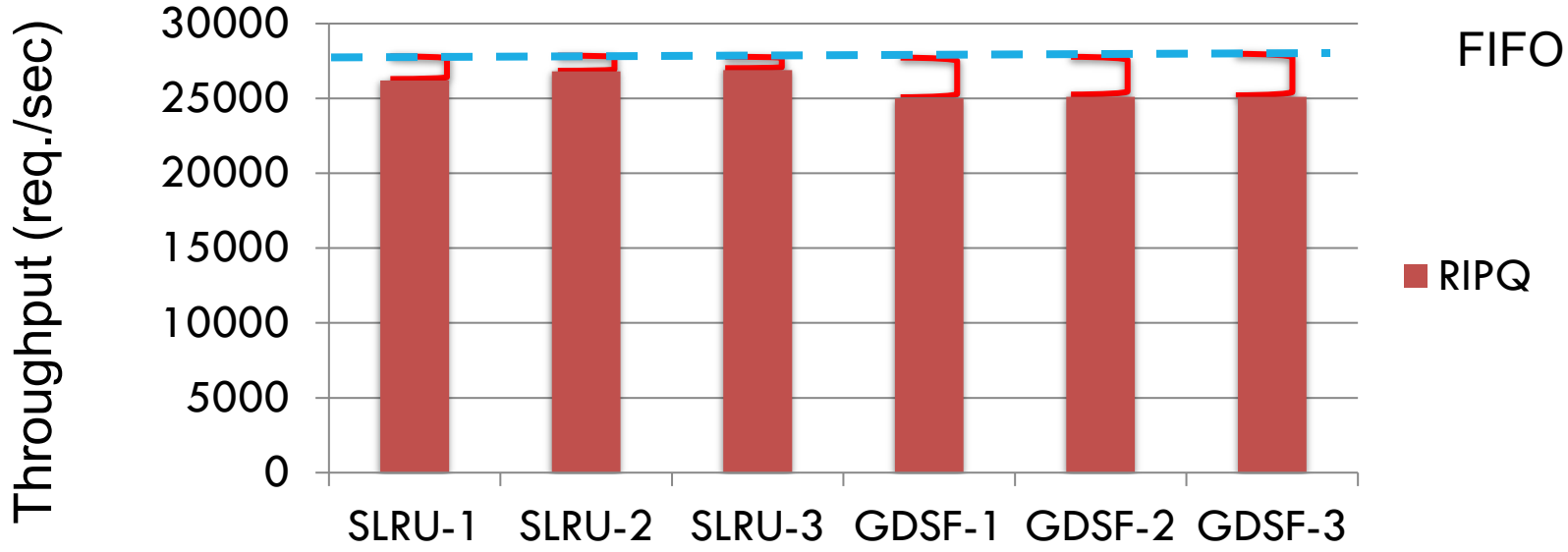
RIPQ achieves ≤0.5% difference for all algorithms

RIPQ HAS HIGH FIDELITY



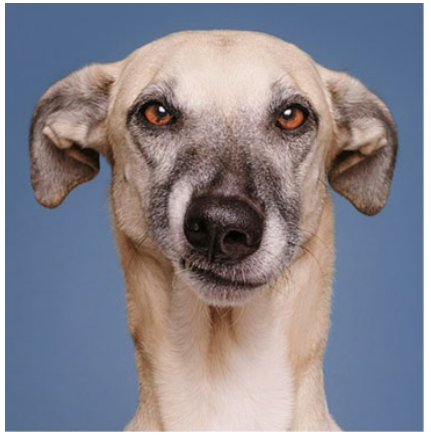
+16% hit-ratio ➔ 23% fewer backend IOs

RIPQ HAS HIGH THROUGHPUT



RIPQ throughput comparable to FIFO (≤10% diff.)

MASSIVE SAVINGS!



The total savings across all of this project was about 30-40% cost reduction and a huge improvement in Facebook latency – a smooth, stall-free feed.

For Facebook, this work was extremely impactful. It helped Facebook cement its hold over its market sector at the time.

Other clouds offer similar CDN frameworks, which ultimately enabled the emergence of other social networking solutions with similar characteristics, likeYoutube, Instagram and Tik Tok.

HOW TO FIT THIS CDN AND CACHING TOPIC INTO A BIGGER VIEW OF CLOUD COMPUTING

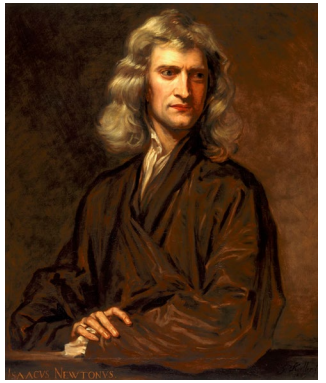
The cloud is an ecosystem: More like a collection of applications than like an operating system (in fact the cloud operating system is Linux). For many applications, blob serving is via some form of CDN

Applications are created by customizing or reusing collections of existing standard framework services.

So we can understand today's lecture as a glimpse of one important kind of framework service, and a detailed view of how it works.

CAN ONE PERSON LEARN ALL OF THEM?

In his era, Isaac Newton was a "polymath:" a universal scientist



Not really! There aren't any universal specialists today.

More typically, we specialize in some area and learn its tools.

ML applications will become an ecosystem of its own.

CONCLUSIONS?

At Facebook, cache and image I/O performance shapes the web browsing experience. Optimizing is a balancing act.

- Important in many kinds of applications
- Issues include the nature of the content: some is more cacheable, some less.
- Optimizing relative to the *actual* properties of the hardware is important!
- Sometimes your intuition deceives you. Ken was surprised that collaborative caching at geographic scale turns out to be a great idea.
- Cost of the overall solution matters a lot too.

SELF-TEST: T/F

Which kinds of objects would you find in Haystack:

- Original copies of images in the format originally uploaded
- Resized copies
- Images created using Facebook's photoshopping tools
- Videos on YouTube that were cross-posted to Facebook
- Comments that Facebook users typed on someone's feed

SELF-TEST: T/F

When a specially sized version of an image is needed

- Facebook resizes it for each new use
- Facebook searches first to see if it has a matching image in the geographic vicinity of the client downloading it
- Facebook resizes it and saves the new size back into Haystack for future reuse
- Facebook sends the original to Akamai, and Akamai resizes it

SELF-TEST

One hypothesis that makes intuitive sense is that if an influencer posts an image, it will be accessed globally. Knowing this, Facebook will anticipate and push that image to every cache.

A second hypothesis is that this form of anticipation has costs and might not always bring any reward.

Which is better supported by the data?

SELF-TEST

Why was S4LRU not an immediate success when Facebook switched their caching solutions to use it?

In your answer, think about the hardware constraints Facebook was coping with and relate the effort Facebook needed to do to the properties of the hardware Facebook was using.

Would S4LRU have worked "out of the box" for an old-style rotating magnetic disk?