Managing Datastore Locality at Scale with Akkio

Muthukaruppan Annamalai, Kaushik Ravichandran, Harish Srinivas, Igor Zinkovsky,
Luning Pan, Tony Savor, David Nagle and Michael Stumm

FACEBOOK, UNIVERSITY OF TORONTO*

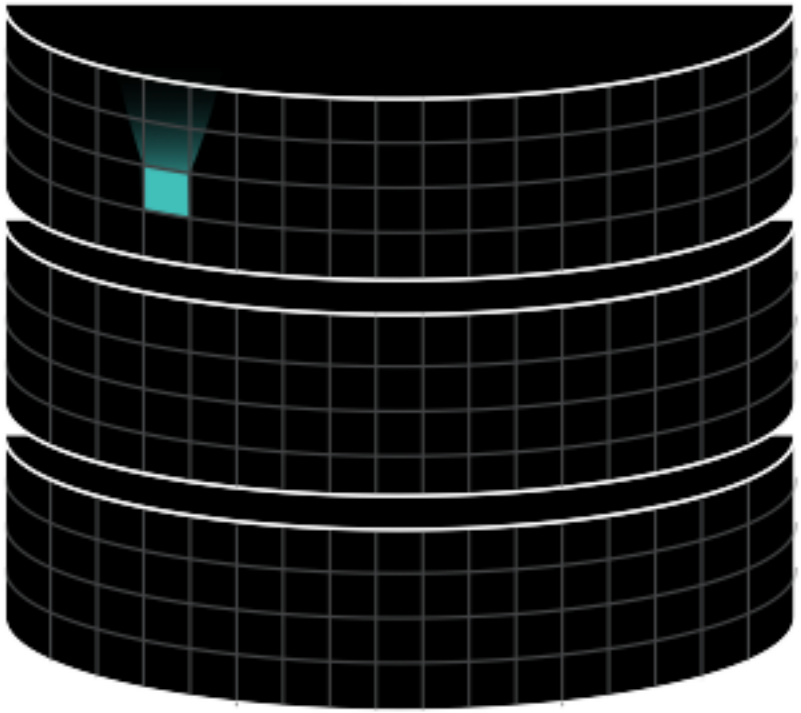
1


Motivation
ALTOONA, IA
PRINEVILLE, OR

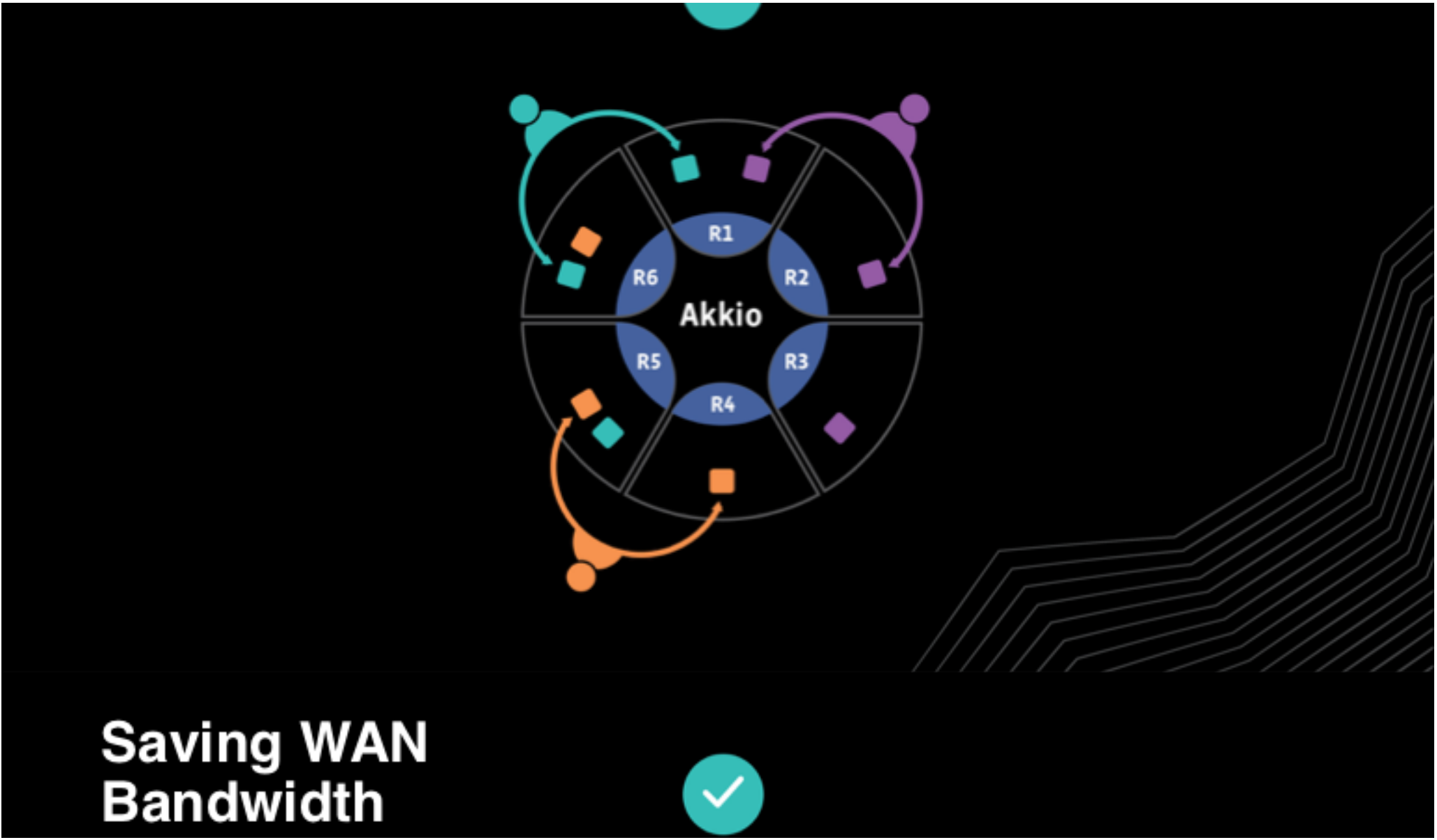
Motivation

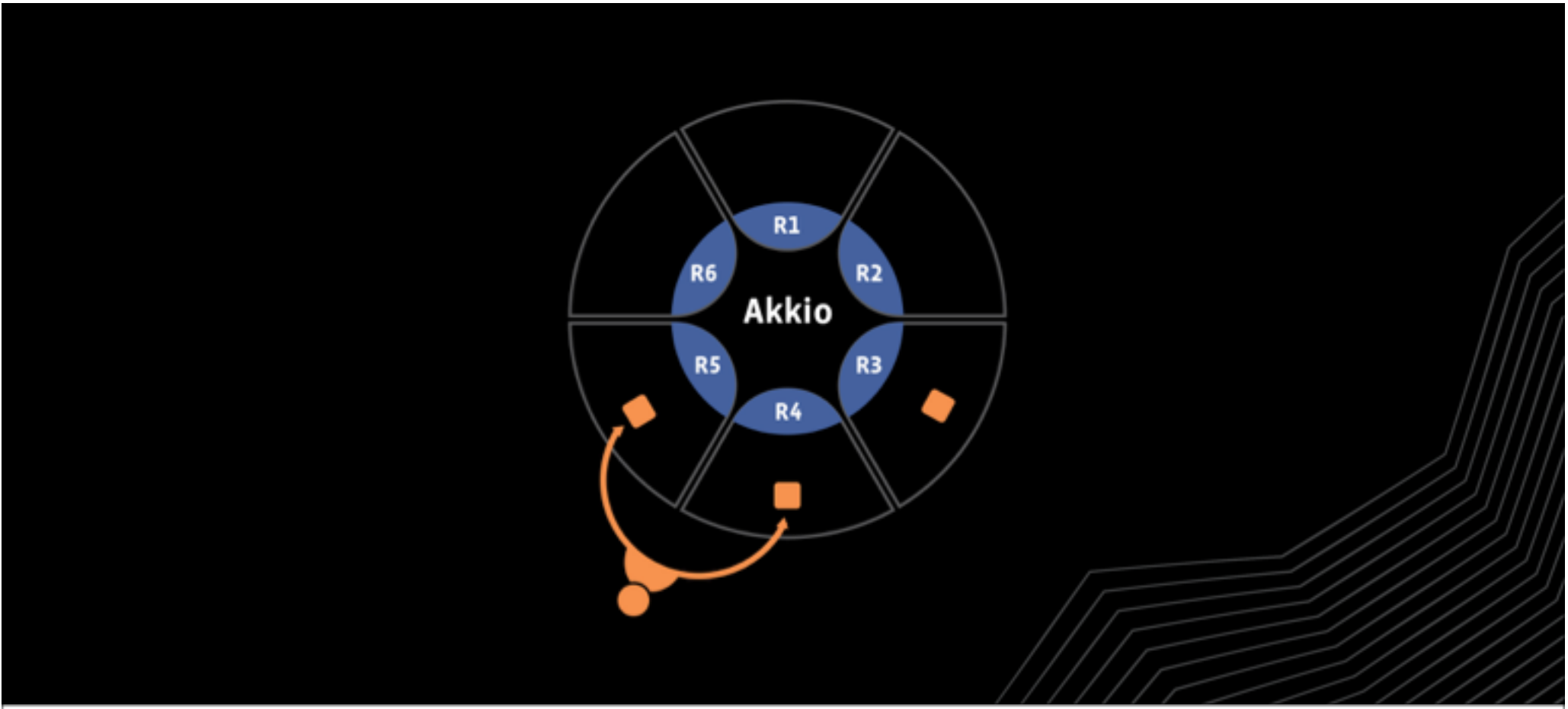

LULEÅ, SWEDEN
PAPILLION, NE
EAGLE MOUNTAIN,
CLONEE, IRELAND
ALTOONA, IA
PRINEVILLE, OR
NEW ALBANY, OH
HENRICO COUNTY, VA
ODENSE, DENMARK
LOS LUNAS, NM
FOREST CITY, NC
FORT WORTH, TX
HUNTSVILLE, AL
SINGAPORE
NEWTON COUNTY, GA

μ -shard



Saving Space


R1
R2
R3
R4
R5
R6
Akkio
Saving WAN
Bandwidth


R1
R2
R3
R4
R5
R6
Akkio


Strong consistency
is not expensive
Better serves
workloads with low
read-write ratios

Flexible

Supports five different durable storage systems

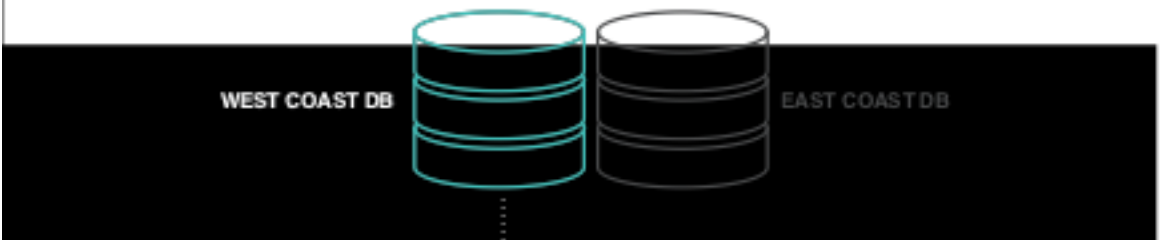
Scalable

100's of millions of data access requests per second

OF CROSS DC TRAFFIC DAILY

OF DATA STORED

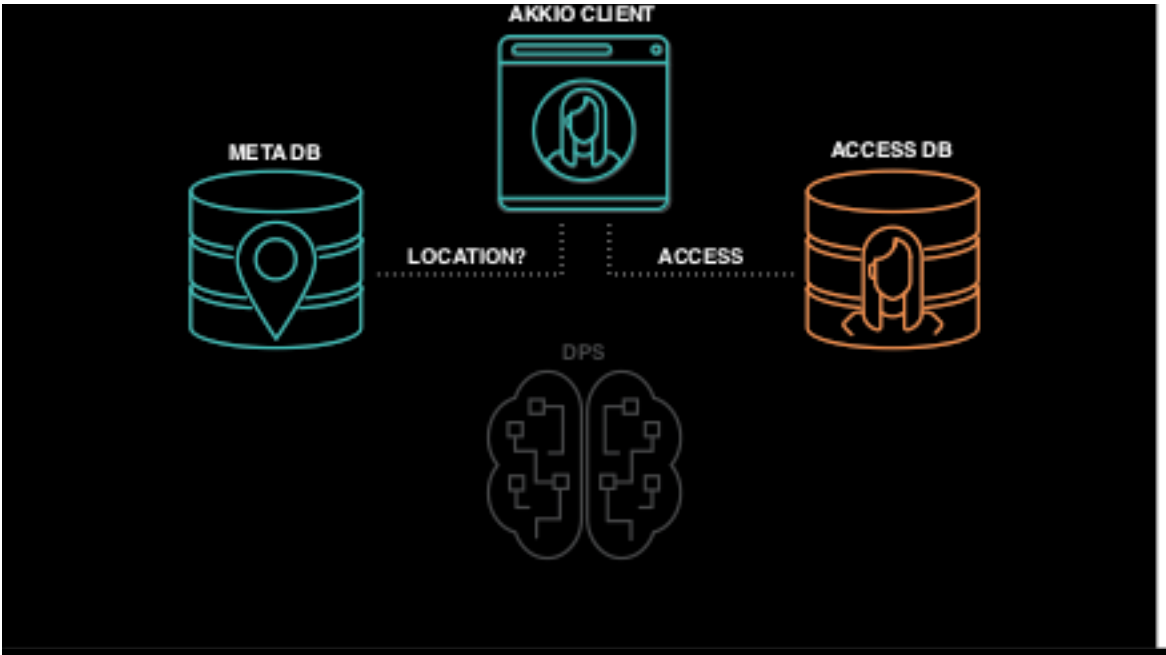
2



Metadata DB

- Stores µ-shard -> placement mapping
- A cache in front to absorb most reads

- Stale lookups are prevented through means like per-µ-shard Access Control Lists (ACLs)
- Typical footprint is ~300GB

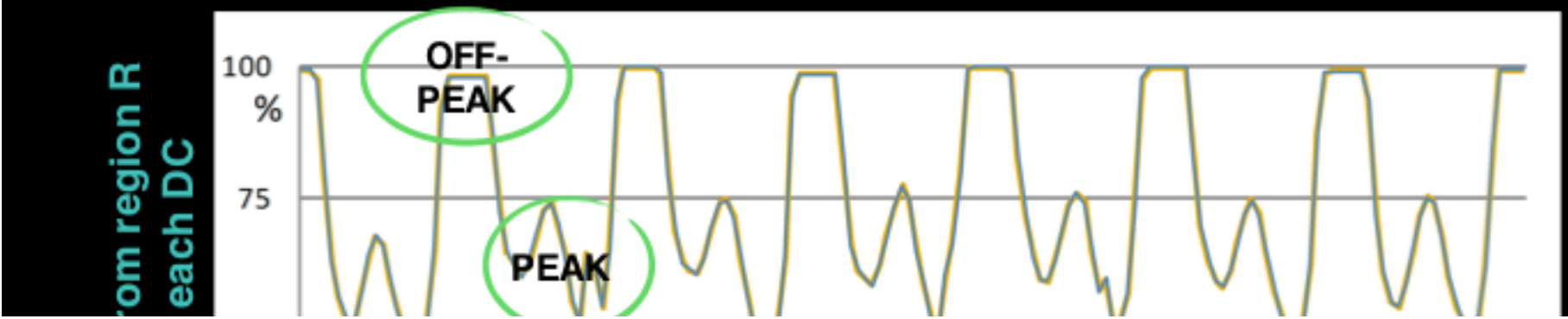


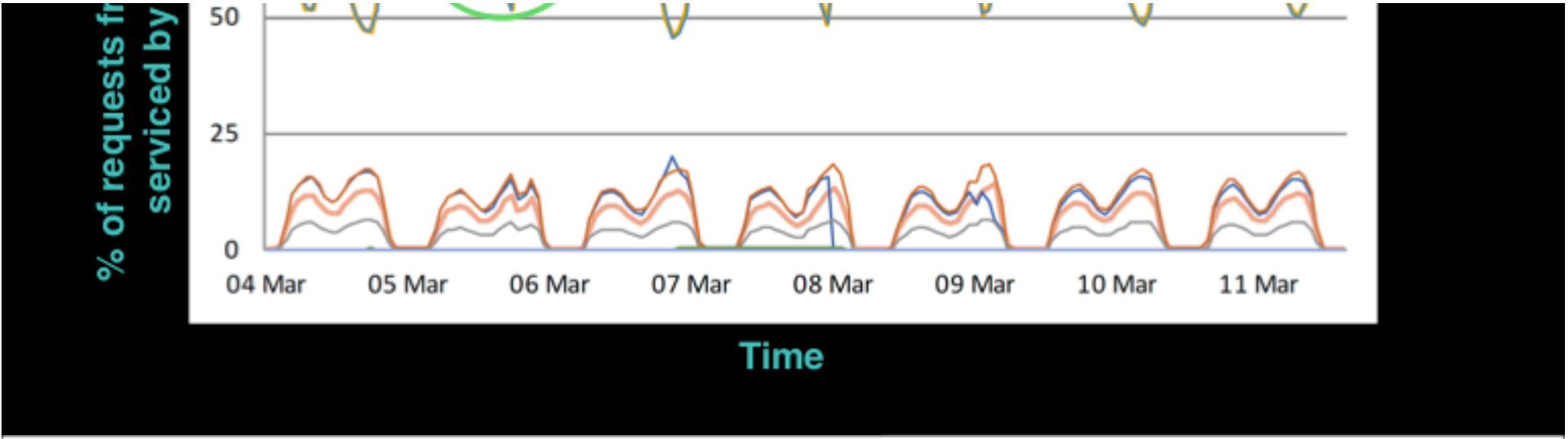
Access DB

- Fundamentally a time-windowed, counter service
- One counter per-<μ-shard, datacenter> tuple

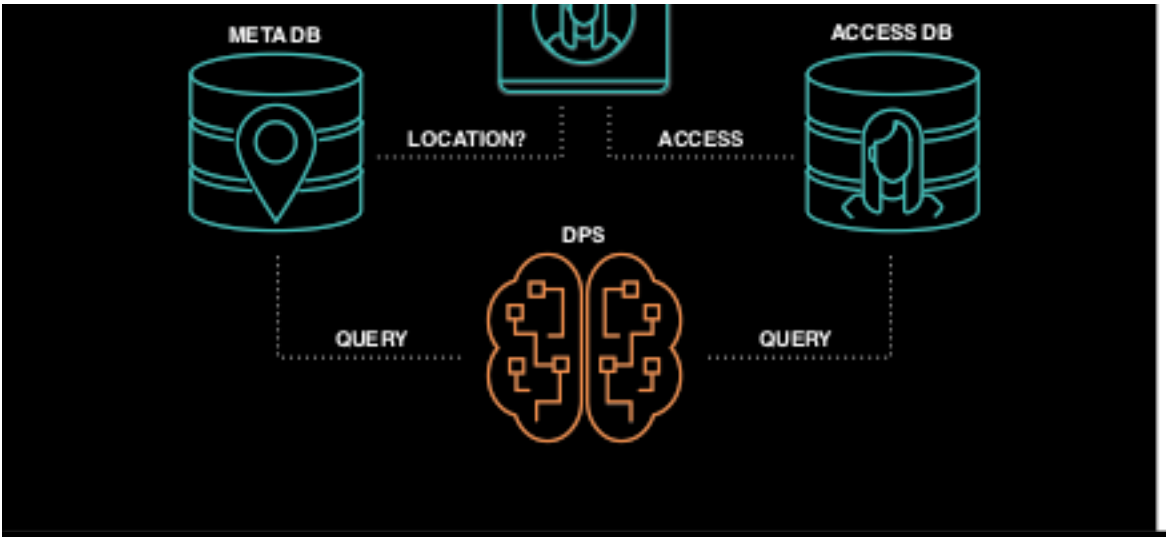
- Typical footprint is ~300GB
- Similar services can share access data
- Routing layer is a complete black box

Routing Layer Load Balancing? No problem!




% of requests fr
serviced by
50
25
0
04 Mar
05 Mar
06 Mar
07 Mar
08 Mar
09 Mar
10 Mar
11 Mar
Time


WEST COAST DB
EAST COAST DB
DATA ACCESS
AKKIO CLIENT



Data Placement Service

- Data Placement Service knows list of possible placements and associated data centers
- Receives hints on sub-optimally placed µ-shards from Akkio client
- New µ-shards are provisioned through the Data Placement Service

- Runs a set of policies to determine migration eligibility
- Identifies an optimal location using a configurable scoring policy

Default Scoring Policy

- Eliminate any placement possibilities with hot data centers
- Generate per-data center score
  - Number of times a µ-shard was accessed from a data center in the last N days
    - N is configurable

- Generate per-placement score
  - Sum of corresponding data center scores
- Pick placement with highest score
  - If tied, break tie using resource usage information

Fault Tolerance

- Migrations are serialized through a per-μ-shard lock on the Metadata DB
- A work item in the Metadata DB is created to track any work
  - Atomically created and deleted with the lock
  - Used to recover migrations
  - Monotonically increasing epoch to fence writes

Migration

- Every backend has a custom Data Placement Service plug-in to enable reads/writes to backend
- Configurable migration strategy
  - Depends on capabilities of underlying store
    - Native timestamps
    - Fine-grained ACLs
    - Read-modify-write transactions

Read-modify-write transactions

Simple Migration Strategy

1. [Metadata DB] Atomically grab per-µ-shard lock & create work item
2. [Source] Set per-µ-shard ACL to R/O, drain writes, read data
3. [Destination] Write data, set per-µ-shard ACL to R/O
4. [Metadata DB] Update placement
5. [Source] Delete data & per-µ-shard ACL

6. [Destination] Set per-μ-shard ACL to R/W

7. [Metadata DB] Atomically delete lock & work item

3

Use Case

40%

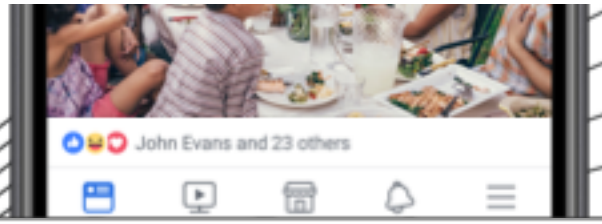
Storage Capacity Win

75%

Reduction in WAN B/W

68%

Savings in p95 Latencies

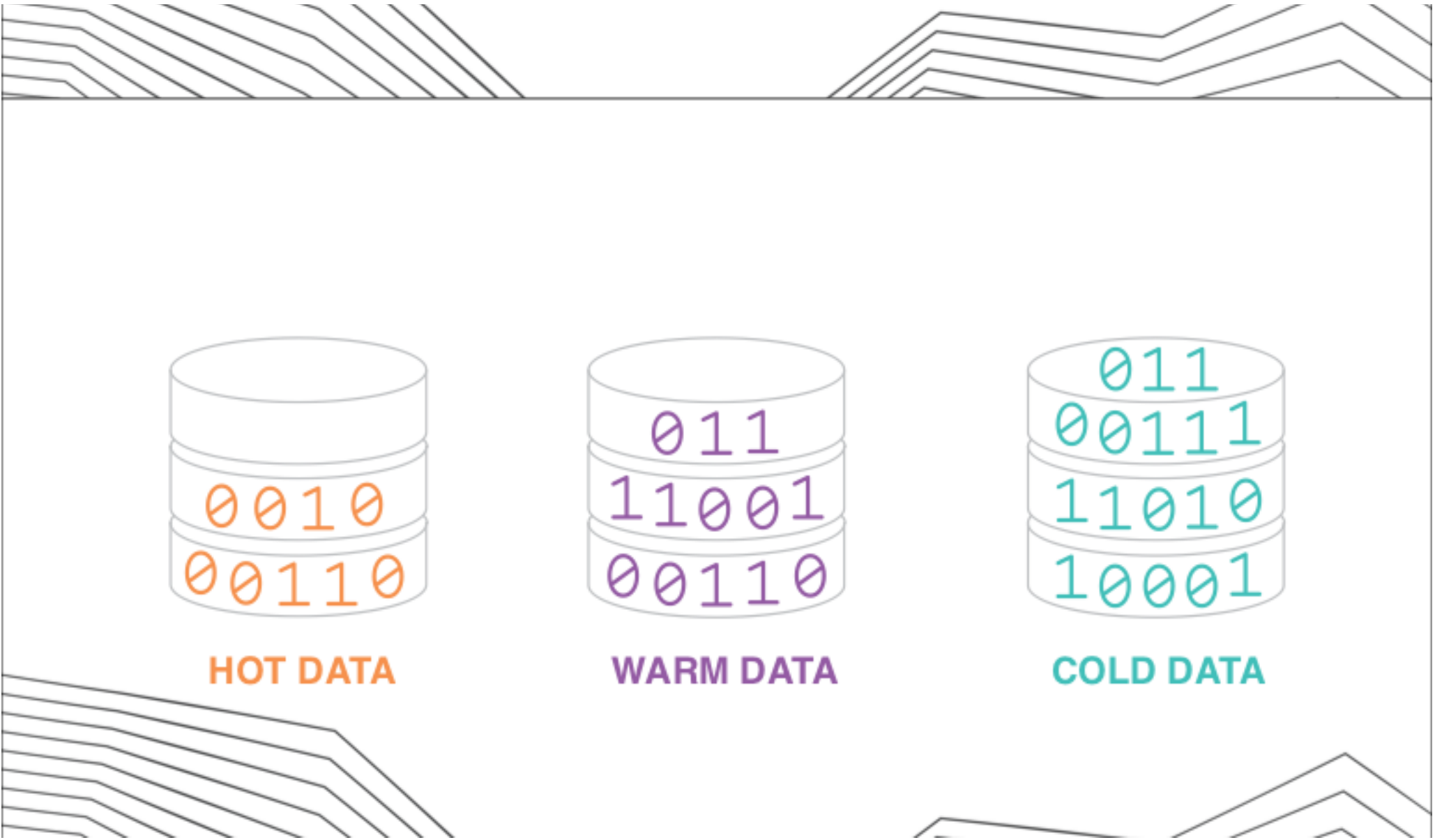


Typically its cheaper to move compute

Data does not need to be present everywhere

Savings on WAN bandwidth and latency wins


0 0 0 1 1 0 1 0 0 0 1 1 0
1 1 0 1 0 1 0 1 0 1 1 0 1
0 0 1 0 0 1 0 1 0 0 1 1 1


00110
HOT DATA
011
11001
00110
WARM DATA
011
00111
11010
10001
COLD DATA

Locality at fine-granularity works!

THANK YOU

POSTER BOOTH #9