
AWS
re:Invent

DAT404

Deep dive on Global Database for Amazon Aurora

Tobias Ternstrom

Director, Product Management,
Amazon RDS and Amazon Aurora
AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Quick recap on database internals

Cloud native database architecture

Global Database

Global Database write forwarding

Global Database managed RPO

Global Database managed failover



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Quick recap: Database internals



AWS re:Invent

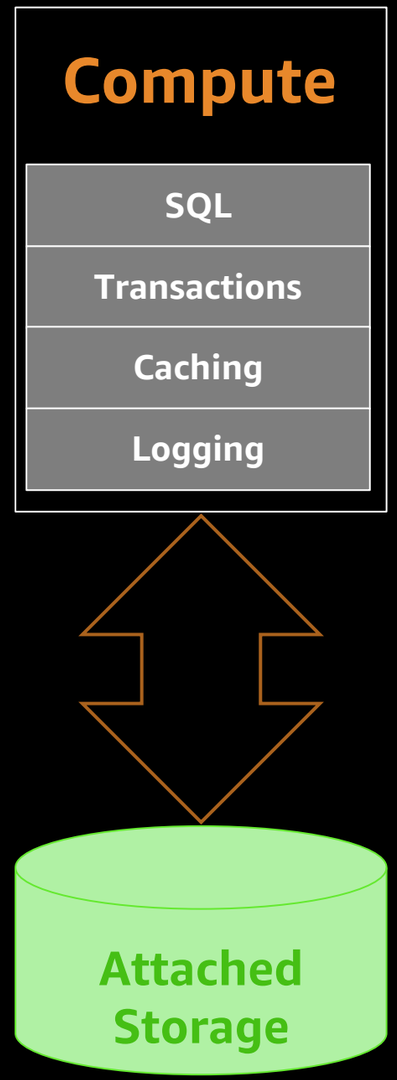
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Traditional database architecture

Databases are all about I/O

Design principles for 40+ years

- Increase I/O bandwidth
- Decrease number of I/Os

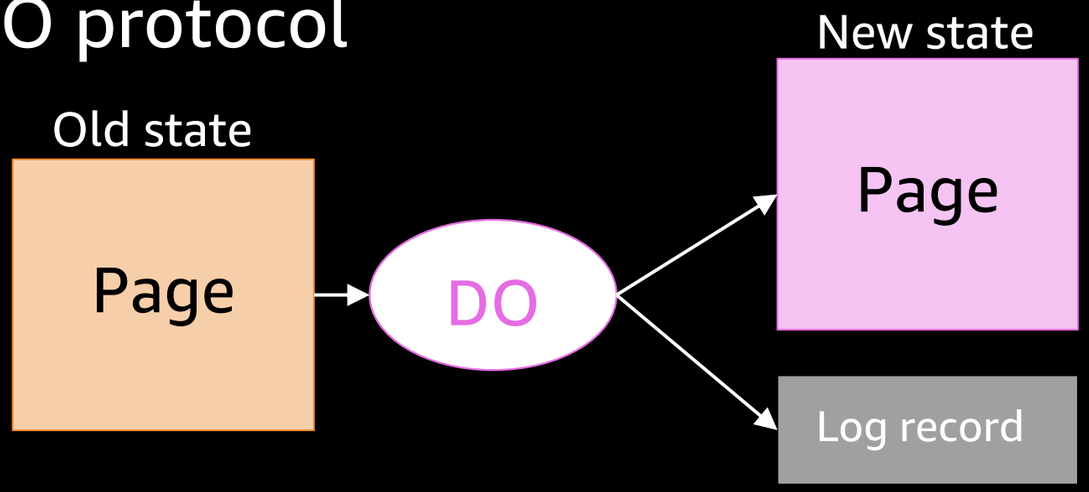


Quick recap: Database internals

Data is organized as fixed-size pages and is kept in main memory as “buffer pool” and is persisted on durable storage using periodic “checkpoints”

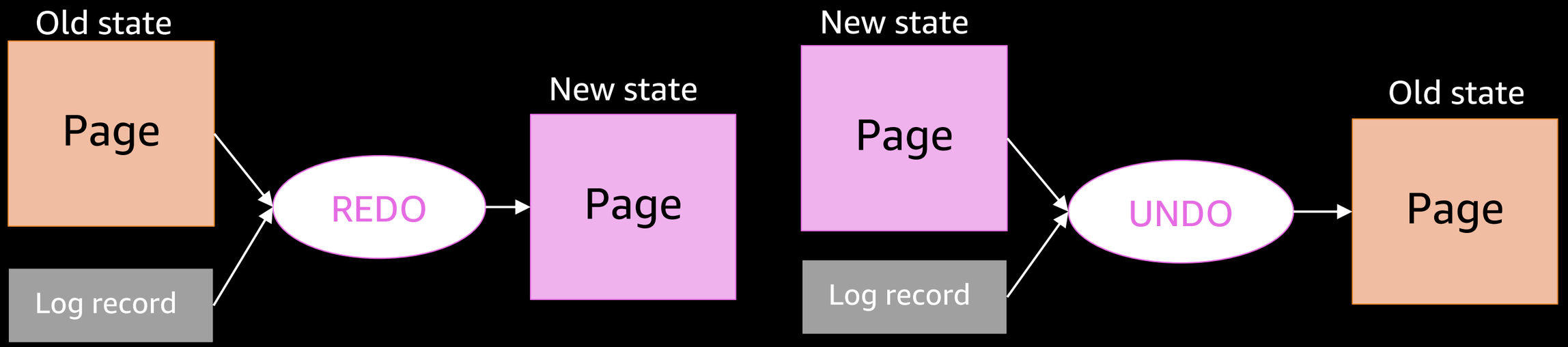
Data is modified in place in the buffer pool using DO-UNDO-REDO protocol with before images and after images stored in write-ahead log (WAL)

DO-UNDO-REDO protocol



Quick recap: Database internals

DO-UNDO-REDO protocol continued…



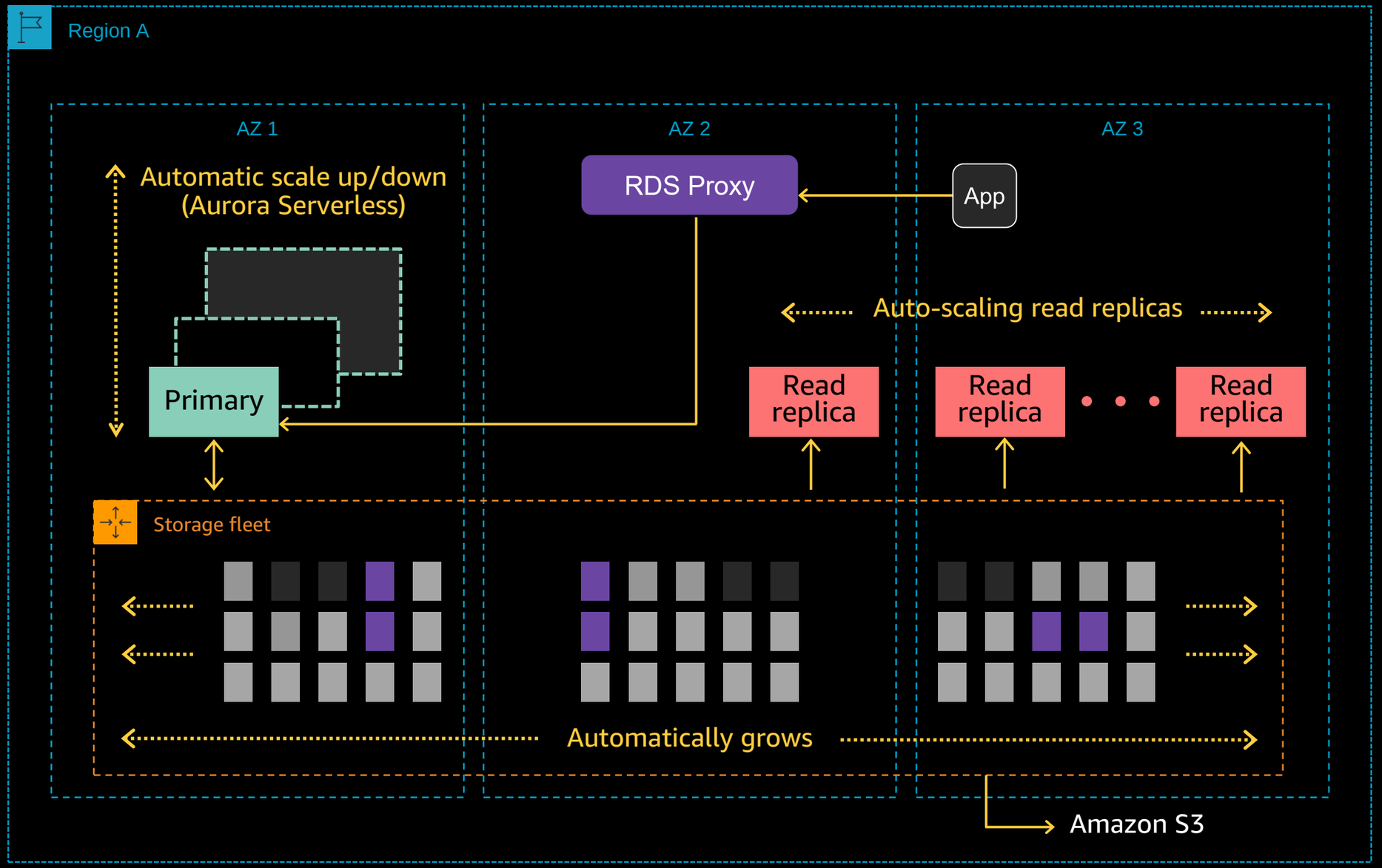
Cloud-native database architecture



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Aurora architecture

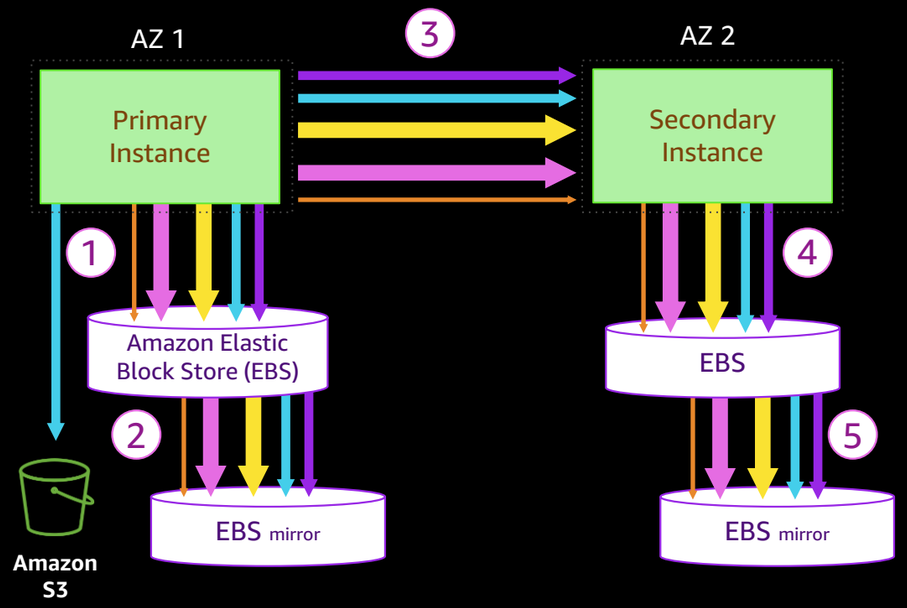


Offload redo processing to distributed storage

- Database instance writes redo log records to storage
- Database instance reads pages on demand from storage, which uses the same redo log applicator as the database instance to produce the correct page image
- Storage materializes the database pages in background
- Storage performs continuous backups of redo logs and pages without impacting database instance
- Storage is replicated for durability and availability

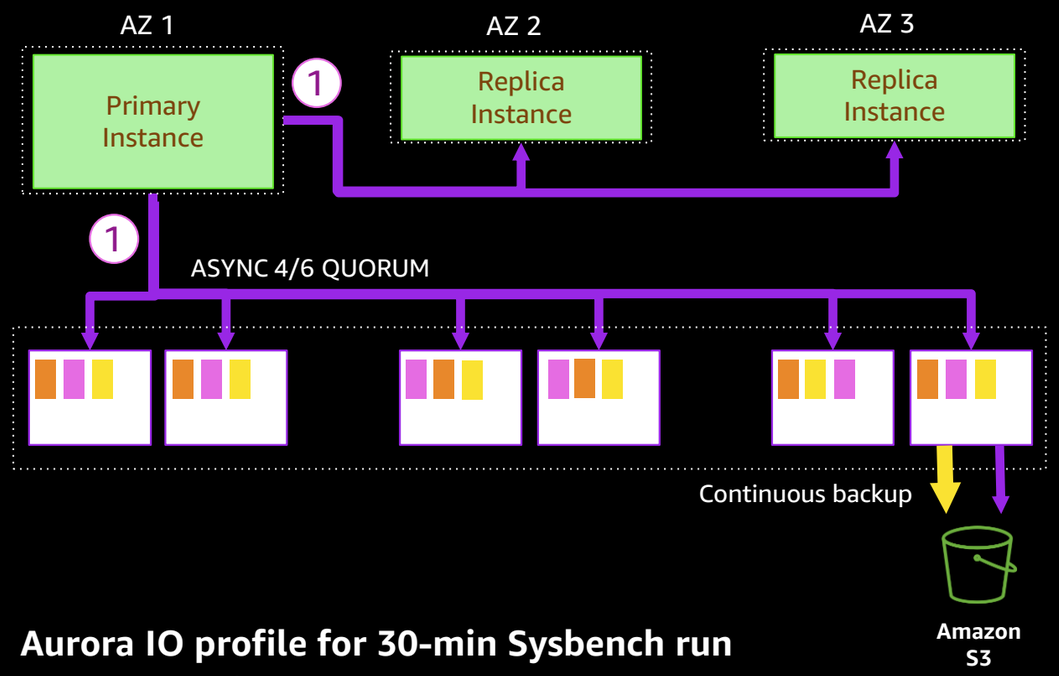
Aurora I/O profile

RDS MySQL with standby instance



MySQL I/O profile for 30-min Sysbench run
780K transactions
Average 7.4 I/Os per transaction

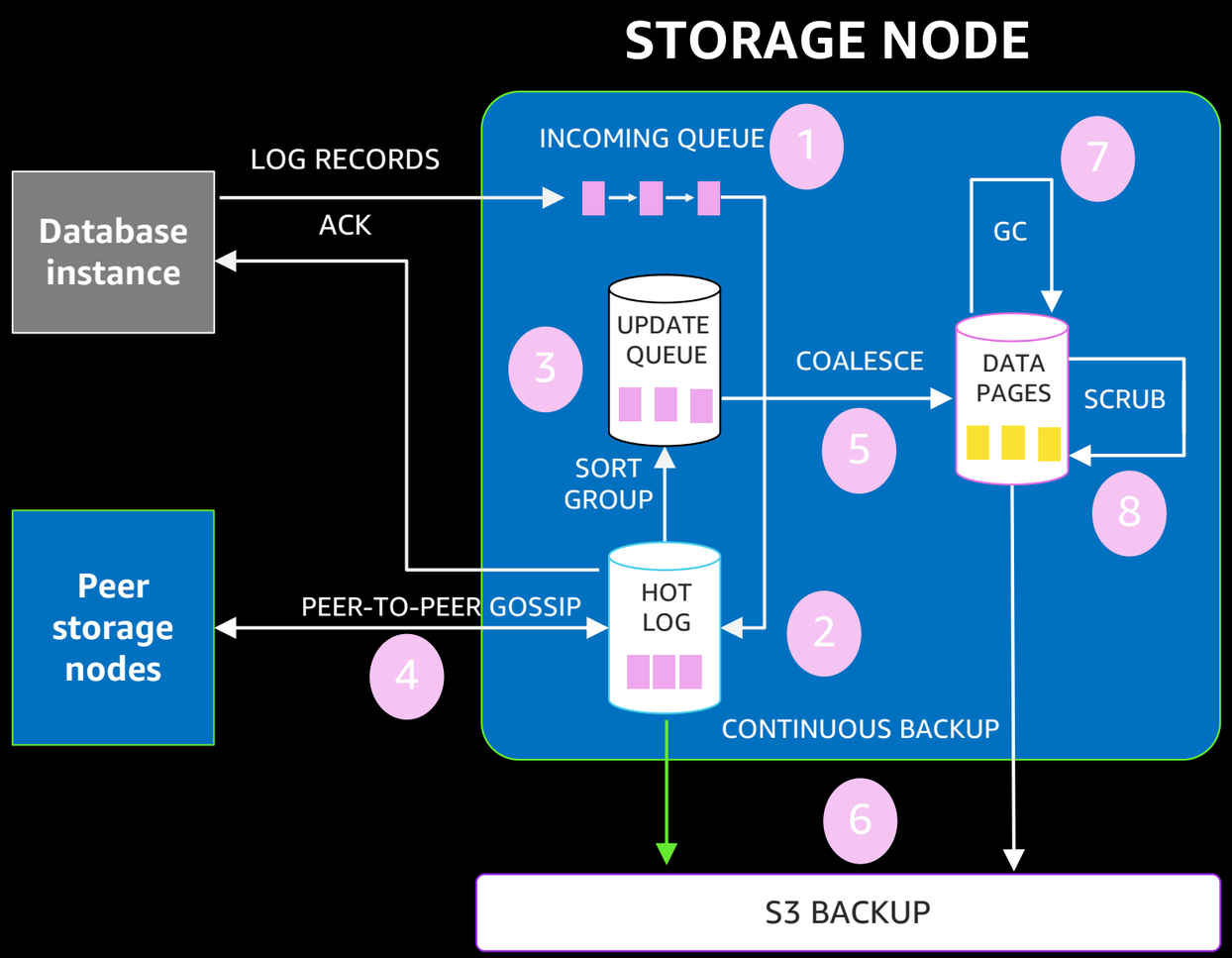
Aurora with read replicas



Aurora IO profile for 30-min Sysbench run
27M transactions – **35X MORE**
0.95 I/Os per transaction (6X amplification) – **7.7X LESS**

Log Binlog Data Double-write Frm files

IO flow in Amazon Aurora storage node



① Receive log records and add to in-memory queue
② Durably persist log records and ACK
③ Organize records and identify gaps in log
④ Gossip with peers to fill in holes
⑤ Coalesce log records into new page versions
⑥ Periodically stage log and new page versions to S3
⑦ Periodically garbage collect old versions
⑧ Periodically validate CRC codes on blocks

Note:
All steps are asynchronous
Only steps 1 and 2 are in IO latency path

Global Database



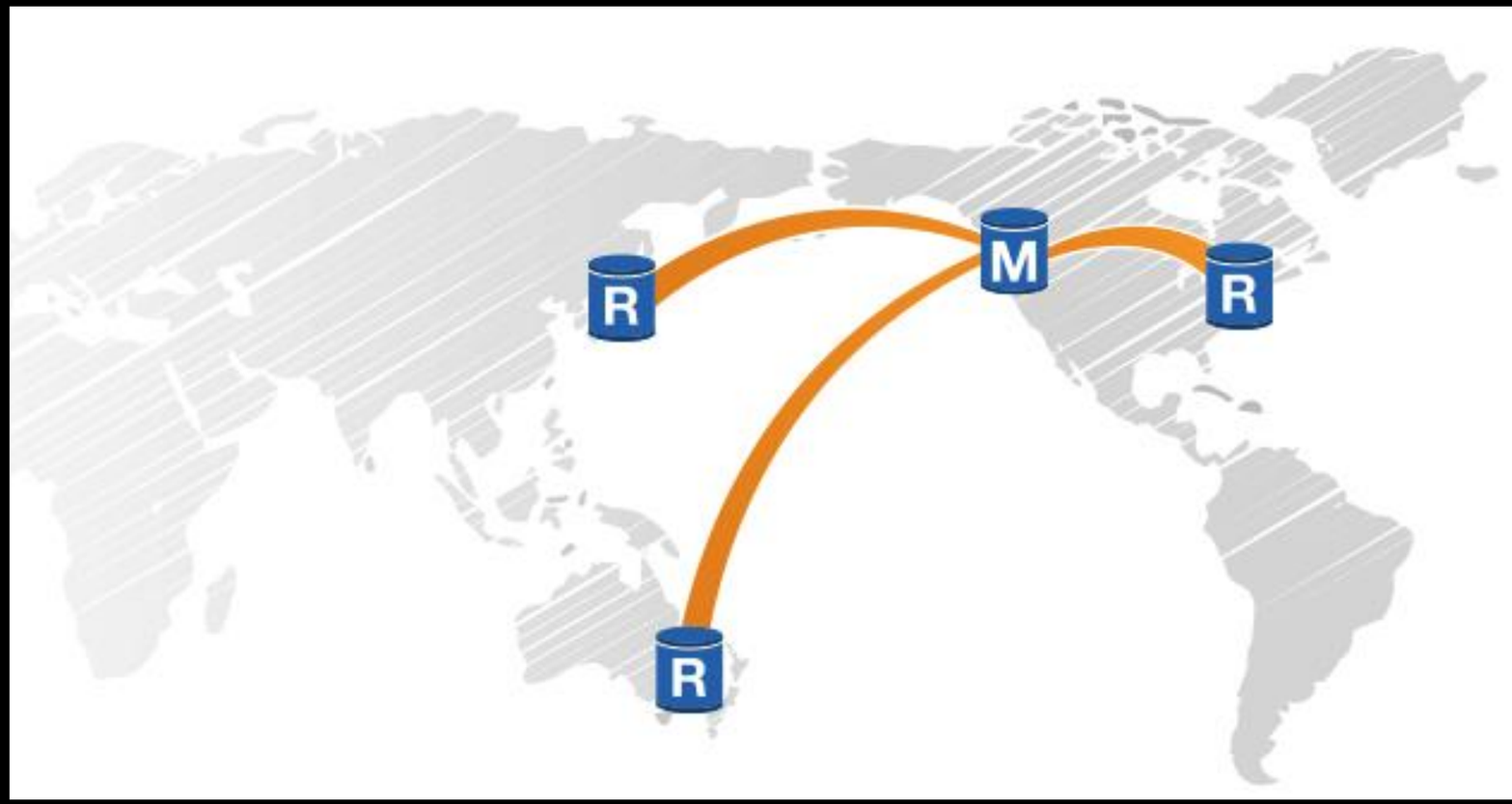
AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

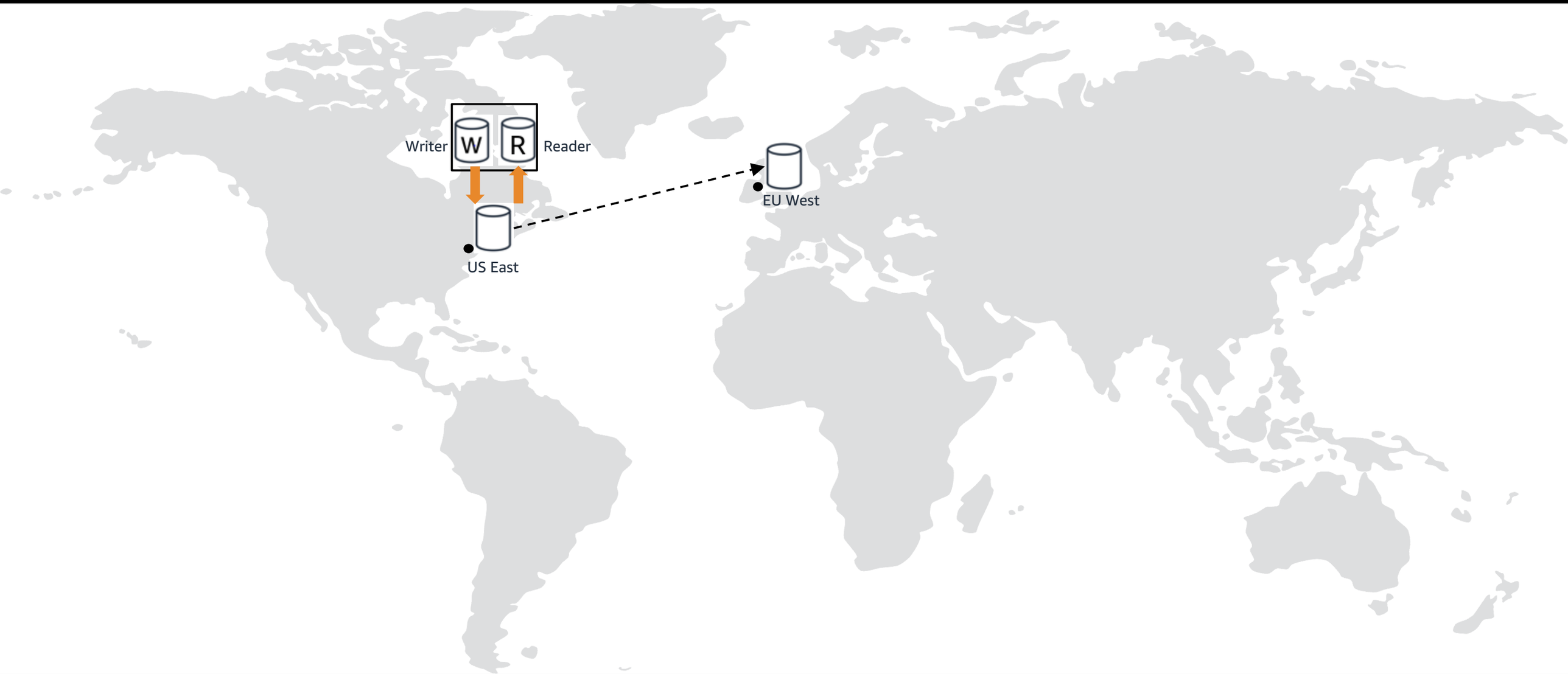
Use cases

- **Disaster recovery**
  Promote remote databases to a primary for faster recovery in the event of disaster

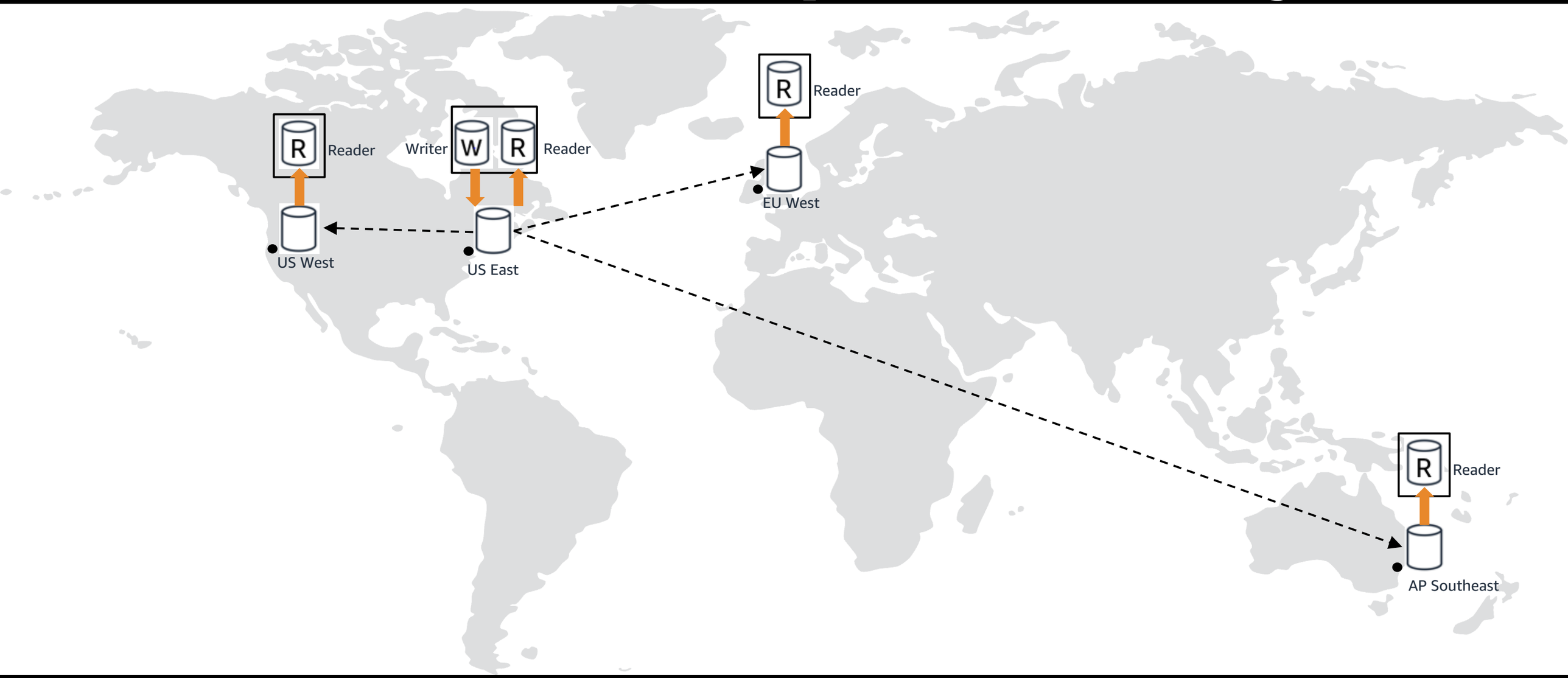
- **Data locality**
  Bring data closer to users in different regions to enable faster reads



Fast cross-region disaster recovery

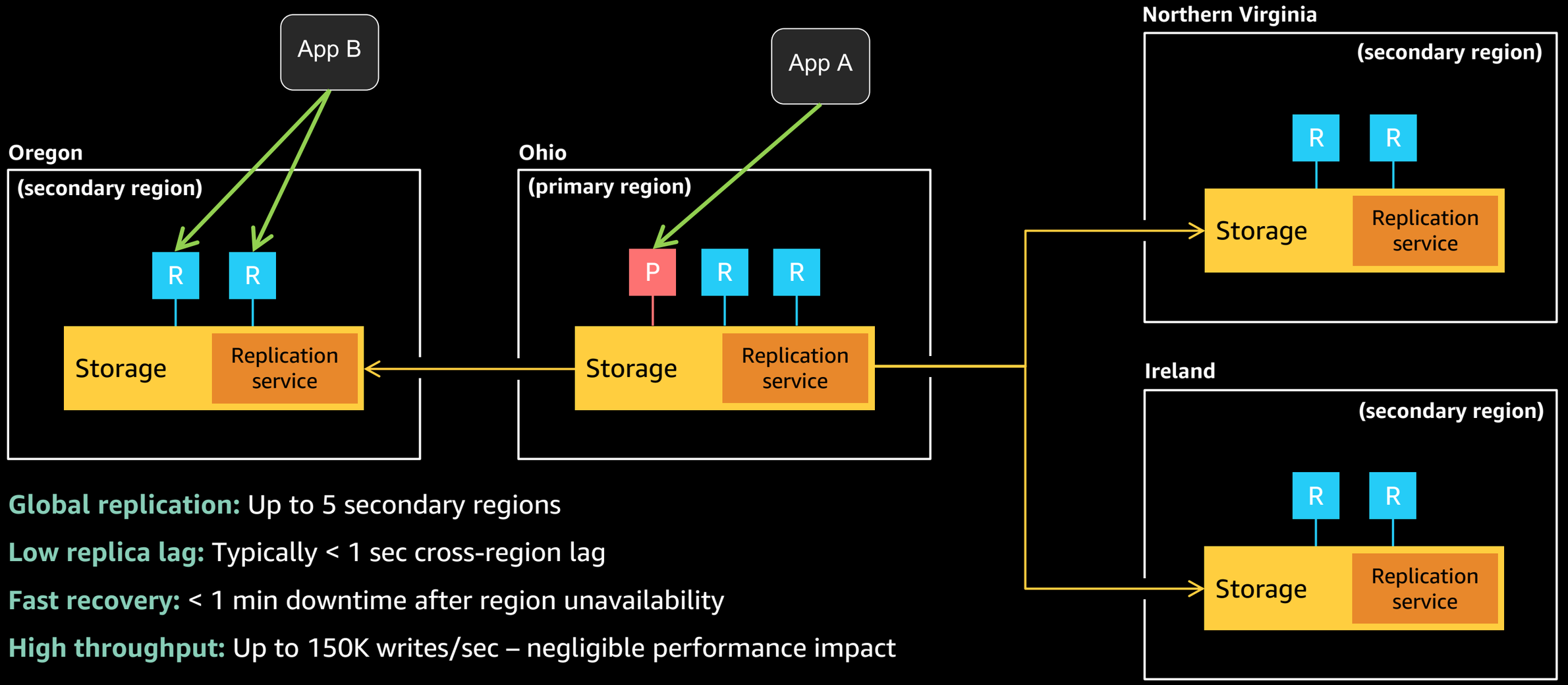


Global reads with low-replication latency



Aurora Global Database

FASTER DISASTER RECOVERY AND ENHANCED DATA LOCALITY



Global replication: Up to 5 secondary regions

Low replica lag: Typically < 1 sec cross-region lag

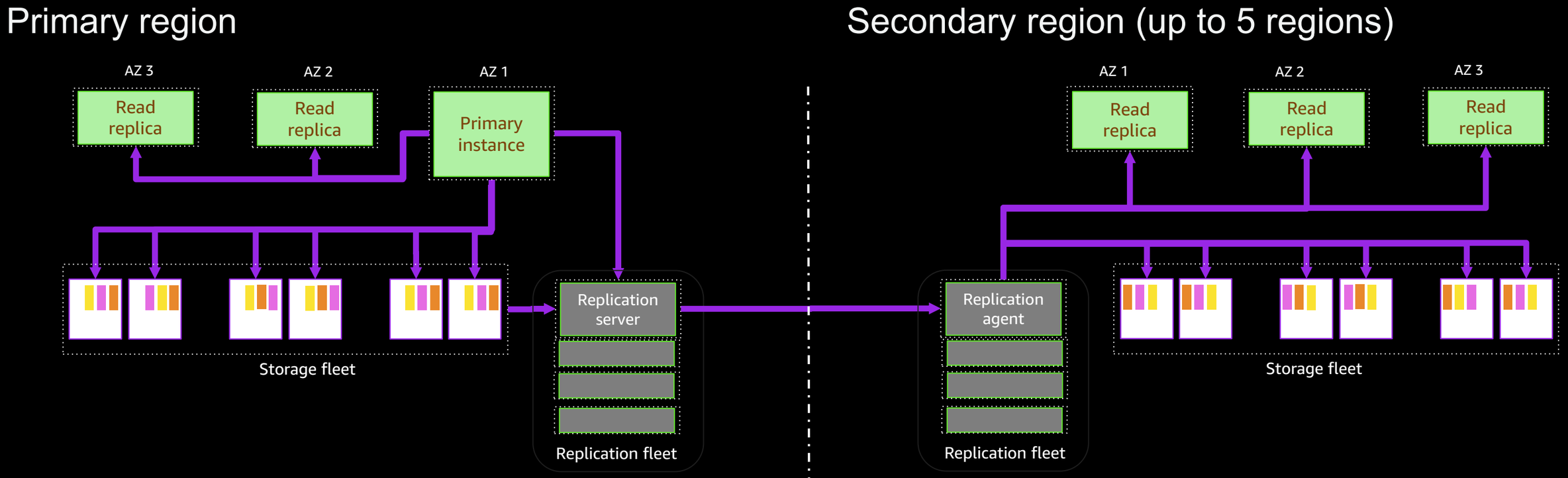
Fast recovery: < 1 min downtime after region unavailability

High throughput: Up to 150K writes/sec – negligible performance impact

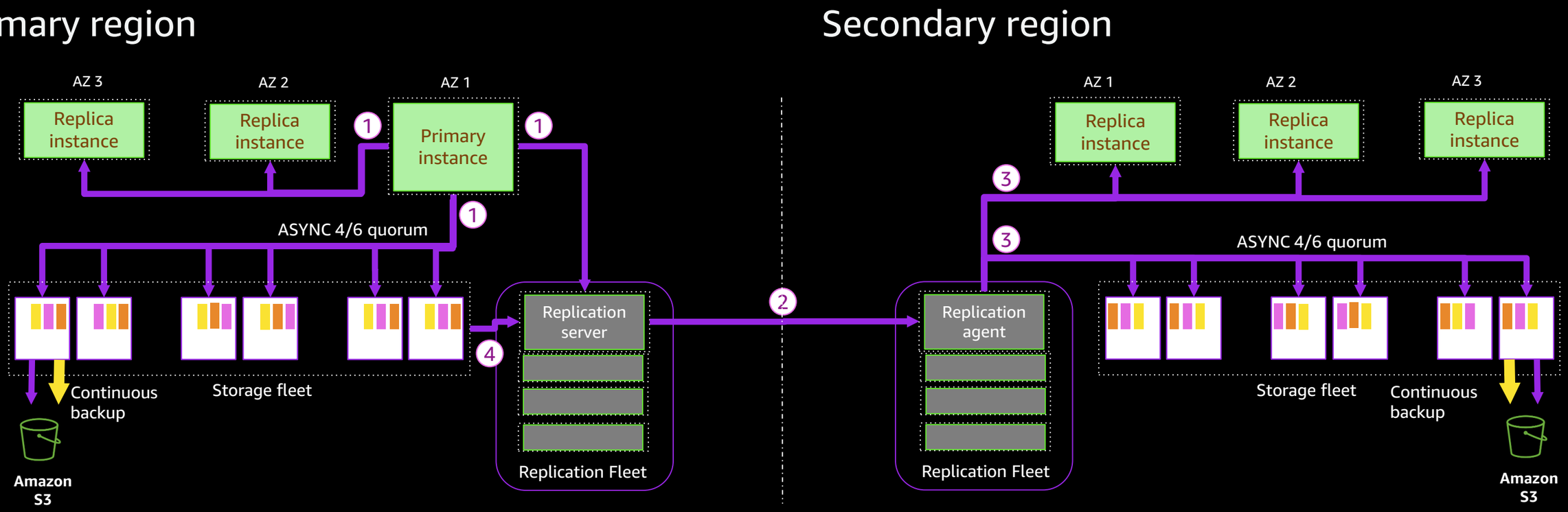
Global reader instances: Up to 90

READ SCALABILITY WITH GLOBAL DATABASES

Aurora Global Database supports up to 90 reader instances spread across 6 regions without impacting the database writer performance in the primary region



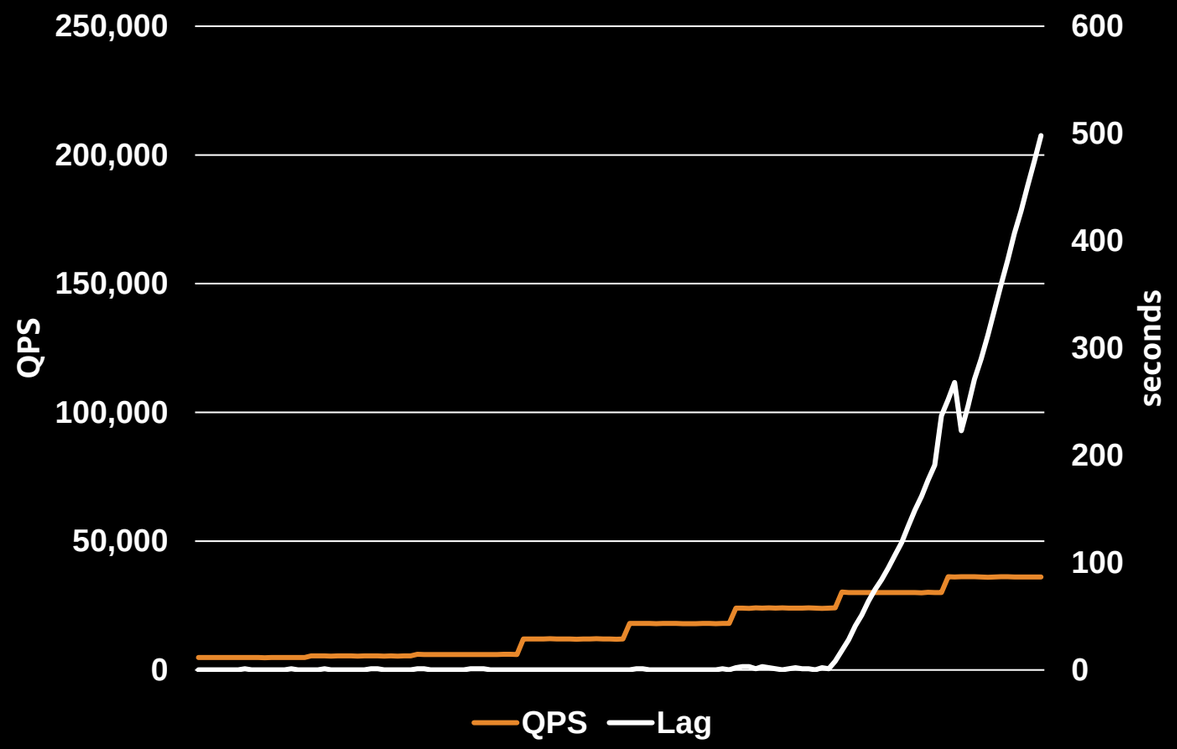
Global replication



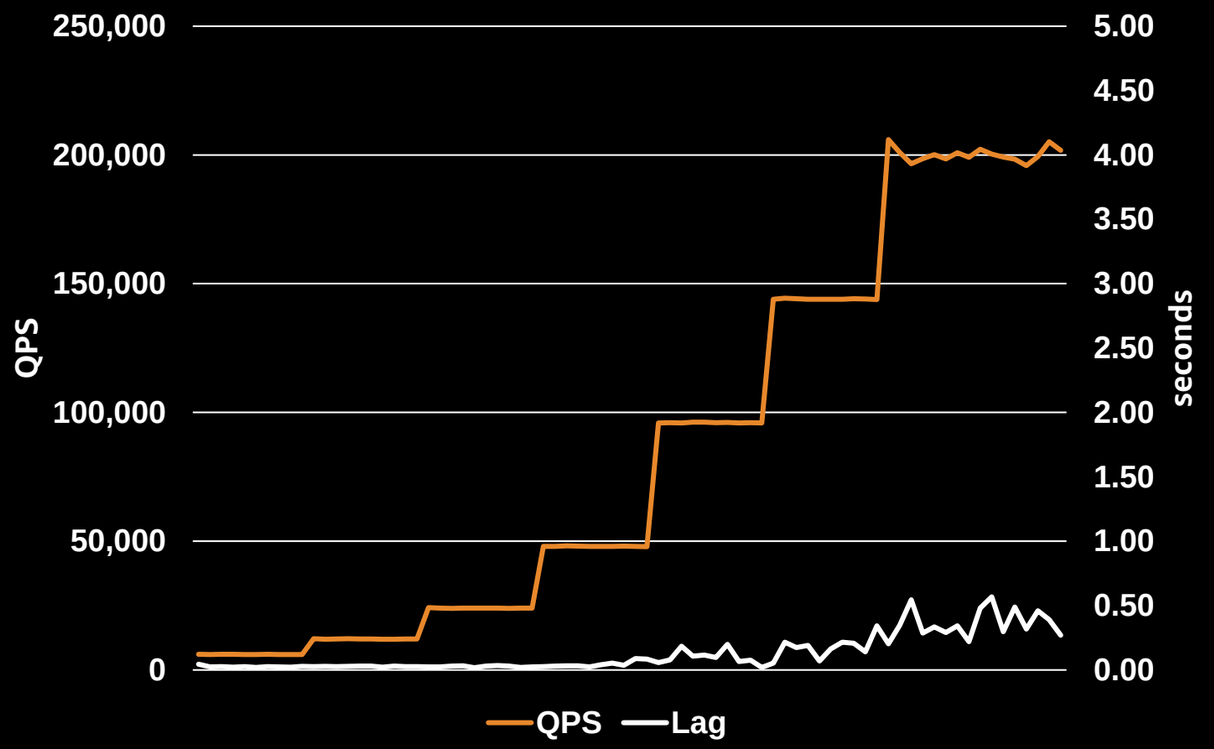
① Primary instance sends log records in parallel to storage nodes, replica instances, and replication server
② Replication server streams log records to replication agent in secondary region
③ Replication agent sends log records in parallel to storage nodes and replica instances
④ Replication server pulls log records from storage nodes to catch up after outages

Global replication performance

Logical vs. physical cross-region replication



Logical replication (MySQL binlog)



Physical replication (log record)

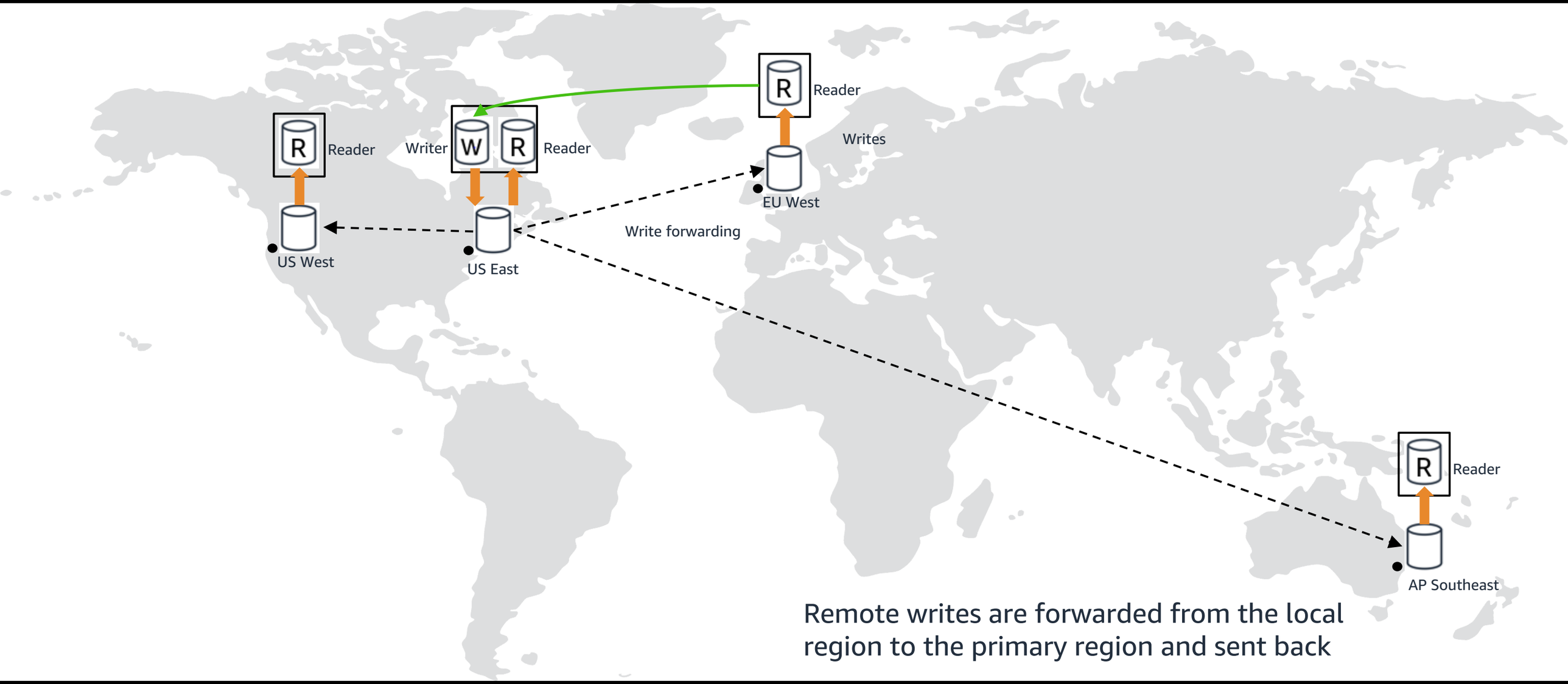
Global Database write forwarding



AWS re:Invent

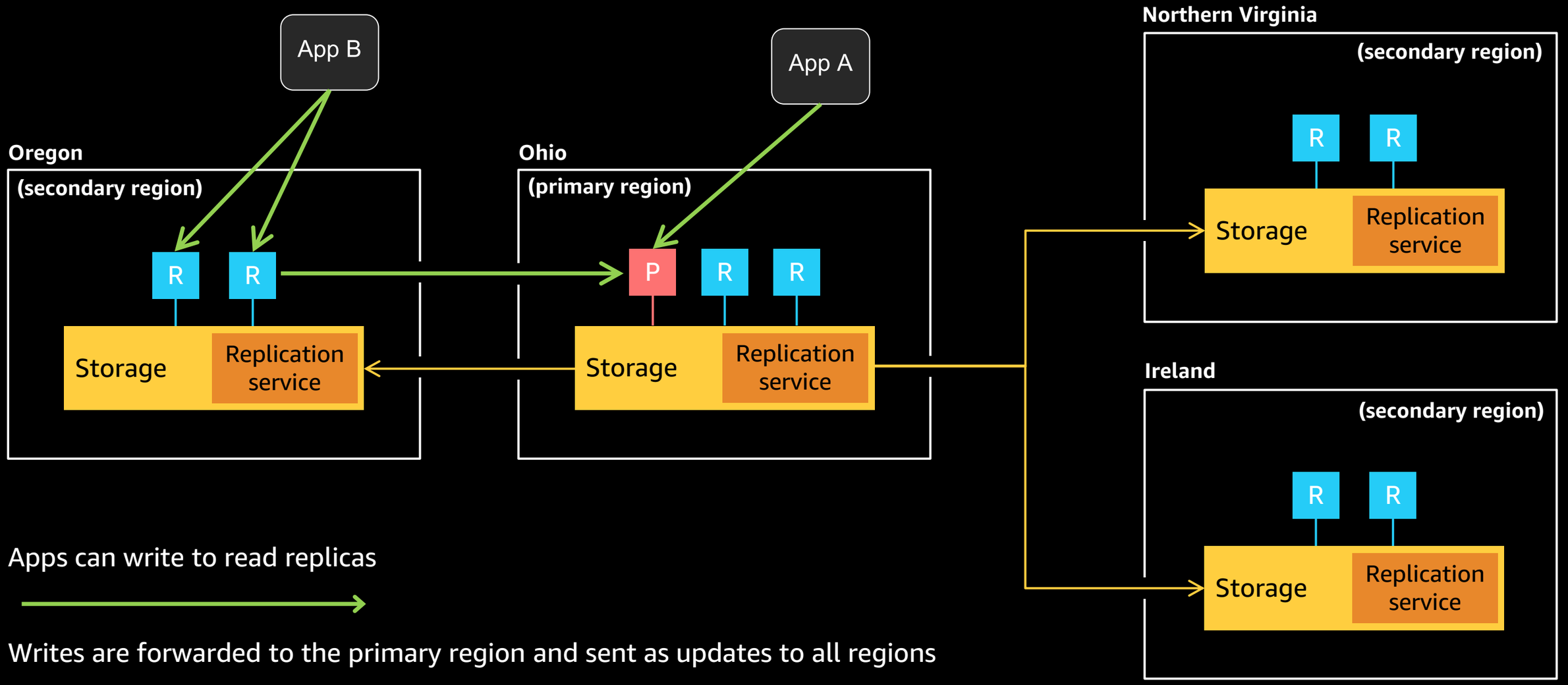
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Use case: Easier application development



Remote writes are forwarded from the local region to the primary region and sent back

Aurora Global Database write forwarding



Apps can write to read replicas

Writes are forwarded to the primary region and sent as updates to all regions

Any region can be promoted to be the primary

Global Database managed RPO



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Use case

Managing Recovery Point Objective (RPO)

- Global Database replication is asynchronous
- Replicas typically lag primary by < 1 second
- Data at risk in case of geo-disaster == Replication lag
- What if a failure (e.g., network) causes replication to fall further behind?
- Application needs protection from replica lag that is too high

Aurora Global Database: Managed RPO

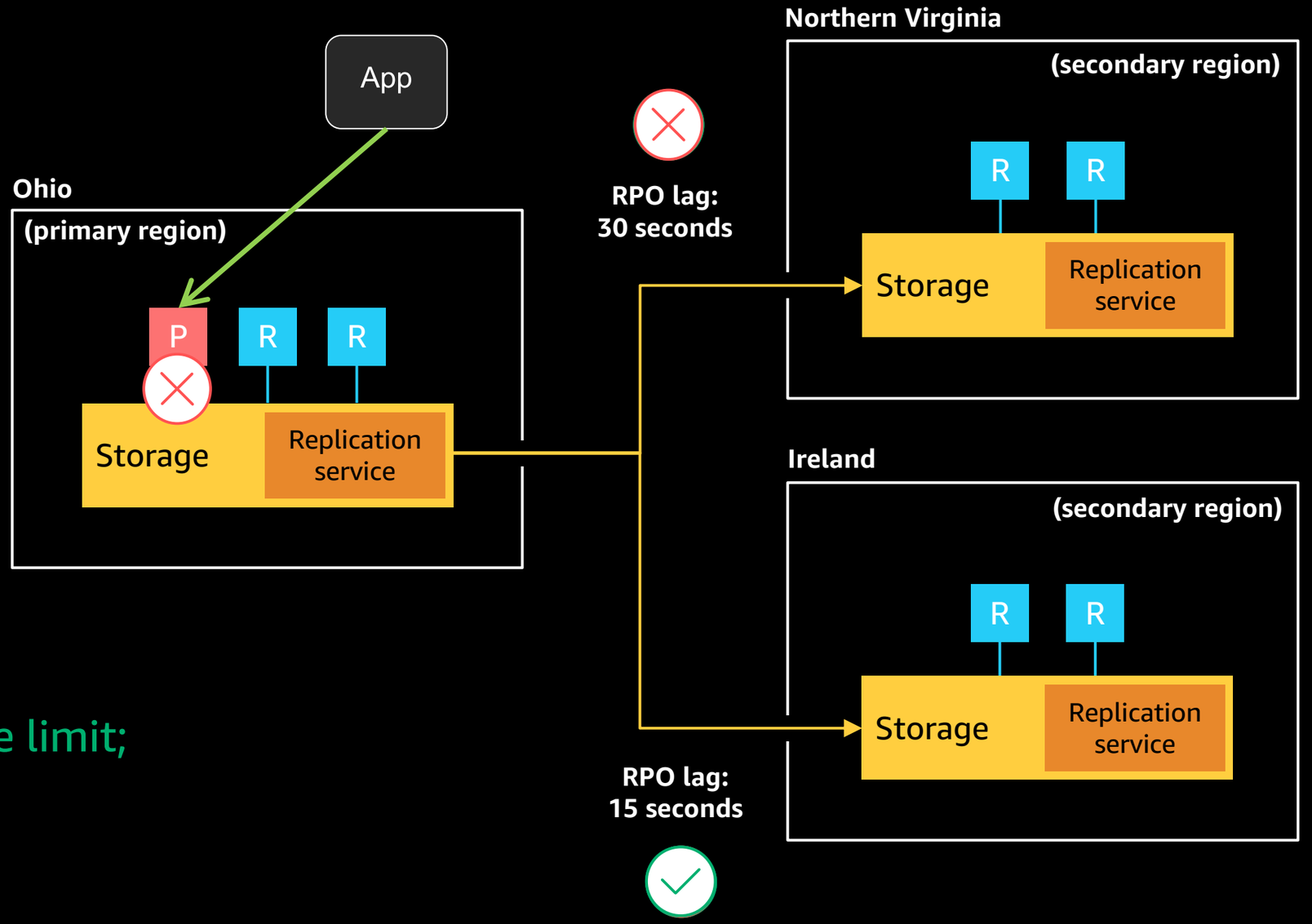
For applications with **critical RPO requirements**

You define the maximum RPO that you allow

If RPO lag in **all** secondary regions exceeds the limit, Aurora pauses writes until **at least one** region catches up

Let's see an example where we set RPO = 20 seconds

→ Lag in Ireland back under the limit; writing has been resumed



Global Database managed planned failover



On the road map

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Use case

On the road map

Global Database managed planned failover

- An easy way to test your DR setup
- An easy way to relocate your primary region
- Promote a secondary region to be the primary
  - In a completely automated manner
  - Without destroying your global database topology
  - With RPO = 0
  - Without having to replicate previous data
  - Without interrupting your DR capability
- Point your application to the new primary and you’re done
- Available for Aurora MySQL and PostgreSQL editions

Aurora Global Database: Managed failover

On the roadmap

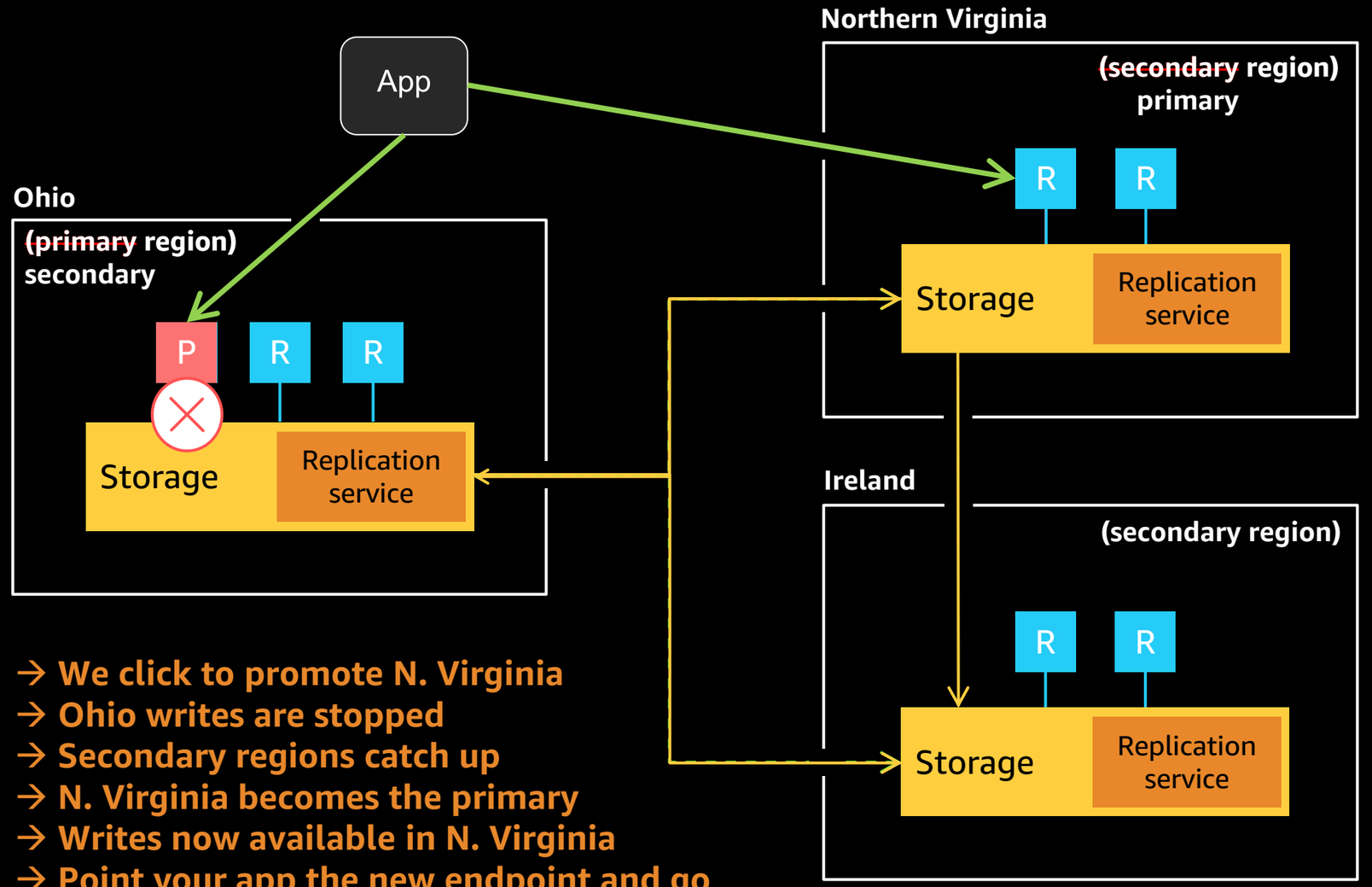
Automated way to promote secondary region to primary

For testing DR setup or relocating primary region

Fully maintains the topology

RPO = 0; writes are stopped until new primary catches up

In this example we'll promote N. Virginia to primary



→ We click to promote N. Virginia
→ Ohio writes are stopped
→ Secondary regions catch up
→ N. Virginia becomes the primary
→ Writes now available in N. Virginia
→ Point your app the new endpoint and go

Related sessions

DAT210

Transforming Hilton's reservation system with Amazon Aurora PostgreSQL

DAT401

Amazon Aurora storage demystified

DAT403

How Amazon Aurora helps you protect your data from mistakes

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Tobias Ternstrom

@tobiassql

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.