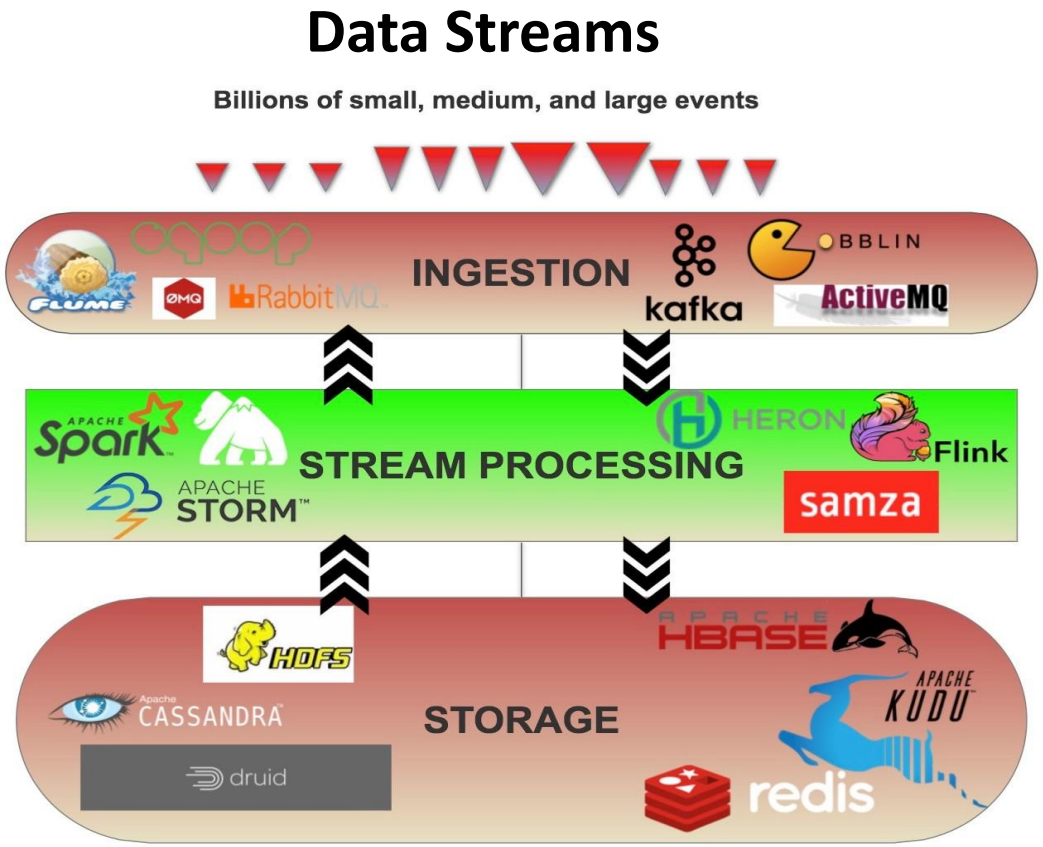
Virtual Log-Structured Storage for High-Performance Streaming

Ovidiu-Cristian Marcu, University of Luxembourg
Alexandru Costan, University of Rennes, Inria, France
Bogdan Nicolae, Argonne National Laboratory, USA
Gabriel Antoniu, University of Rennes, Inria, France

Fast Data: Stream Storage and Processing

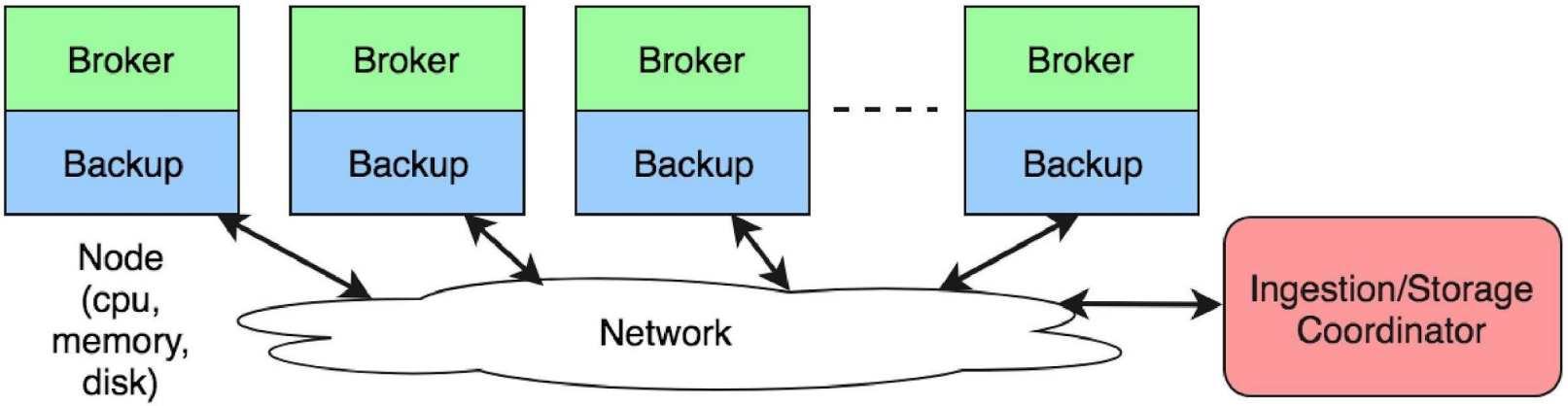


Low-latency
High-throughput

Fault-tolerant
Scalability

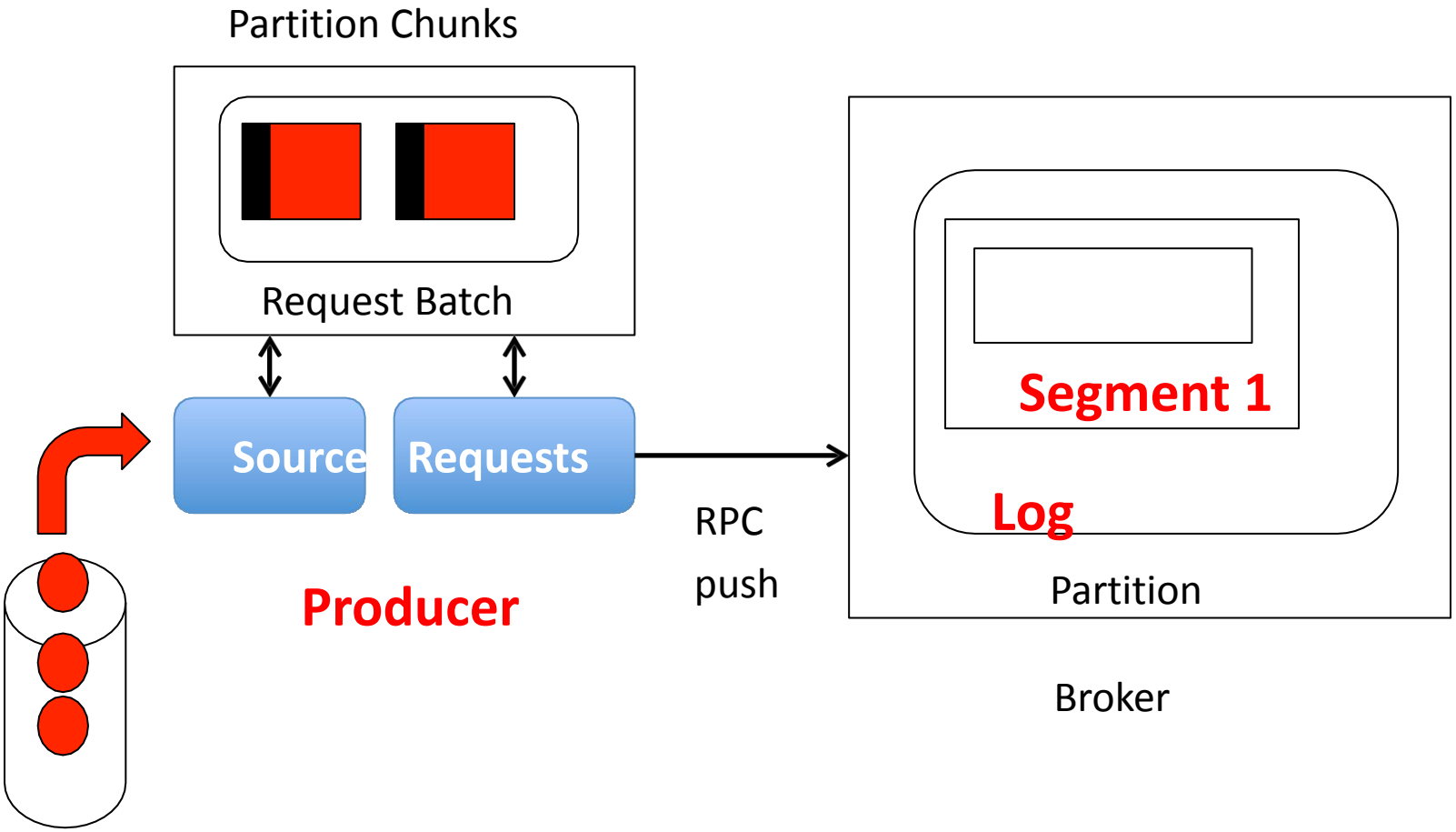
Durability
Consistency

Stream Ingestion and Storage Architecture

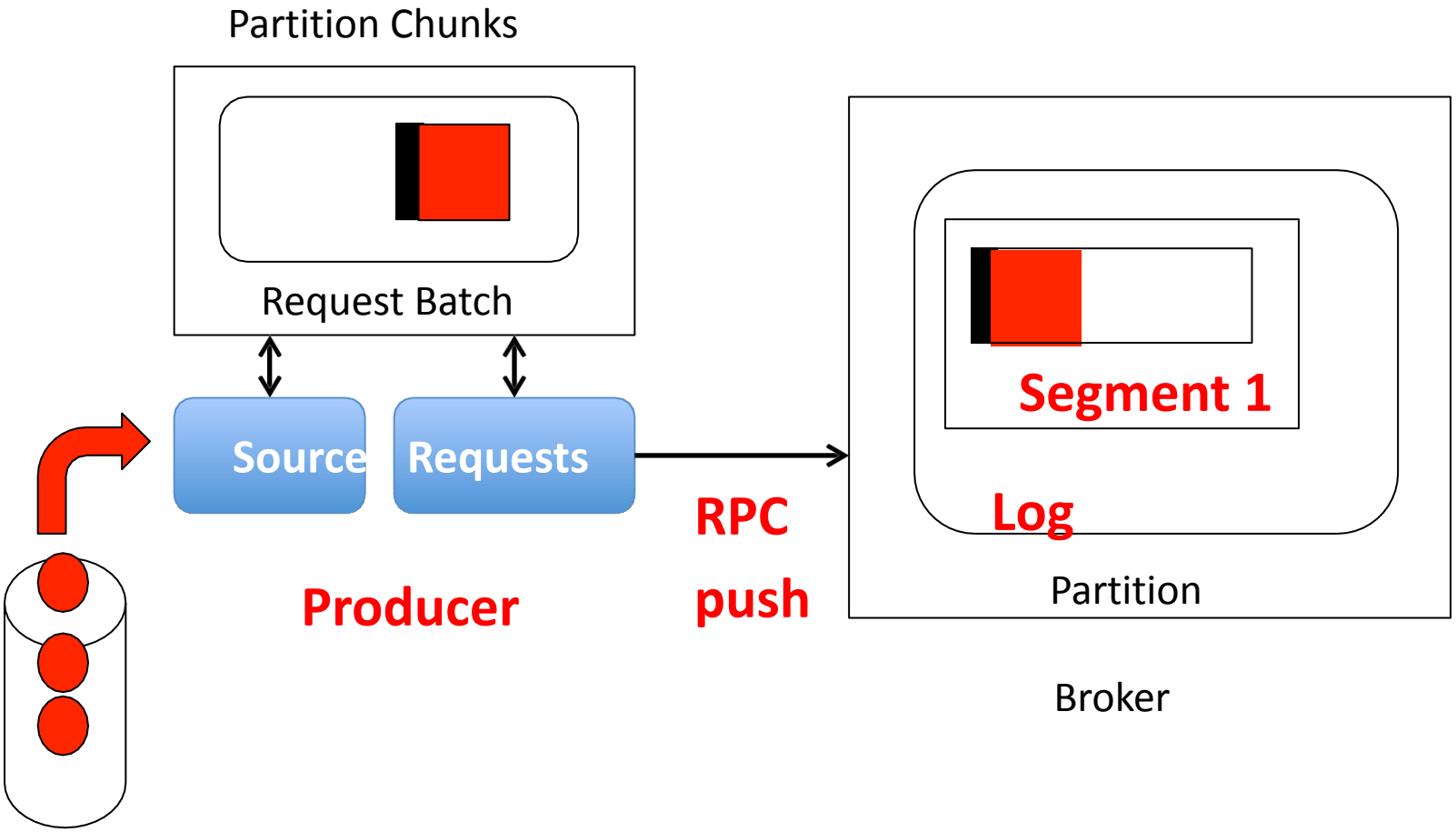


Clients (producers, consumers) query the coordinator for partition metadata
Clients communicate directly with Brokers

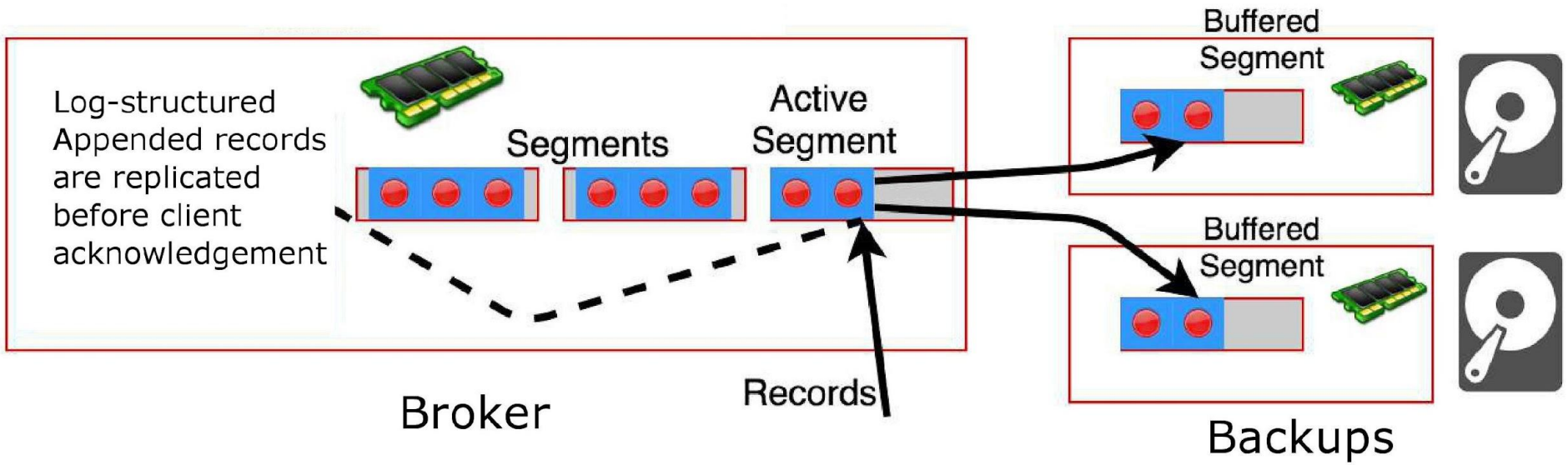
Producer Architecture: Chunk Ingestion



Producer Architecture: Chunk Ingestion



Stream Partitioning Through Log-structured Storage Implementation



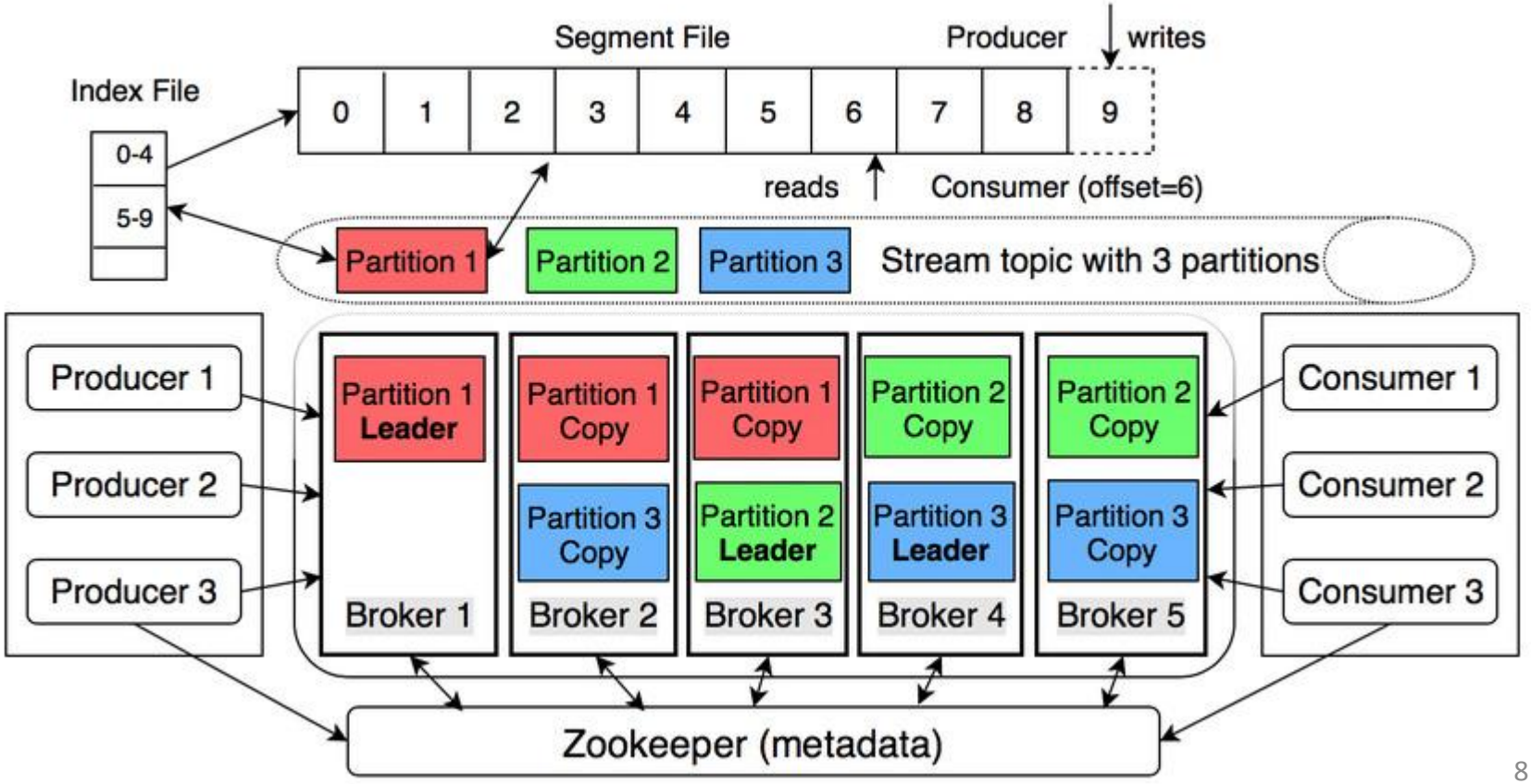
A stream partition represented by a replicated log

How to organize stream partitions for faster replication and better throughput compared to using one replicated log per partition approach?

How to Replicate?

Option 1: per partition replicated log approach in Kafka

Issues: Metadata, IOs



How to Replicate?

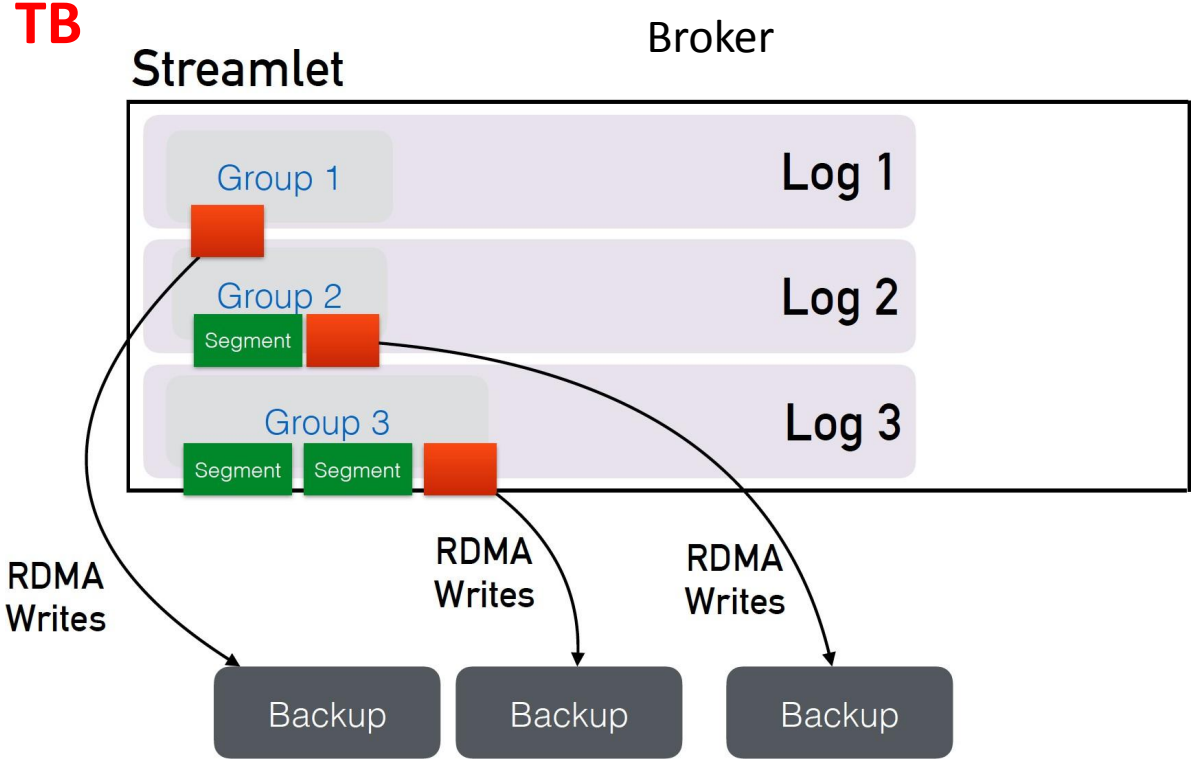
Option 2: the multi-log approach in KerA

Additional Issue: backups in-memory storage

Memory required = 80 TB

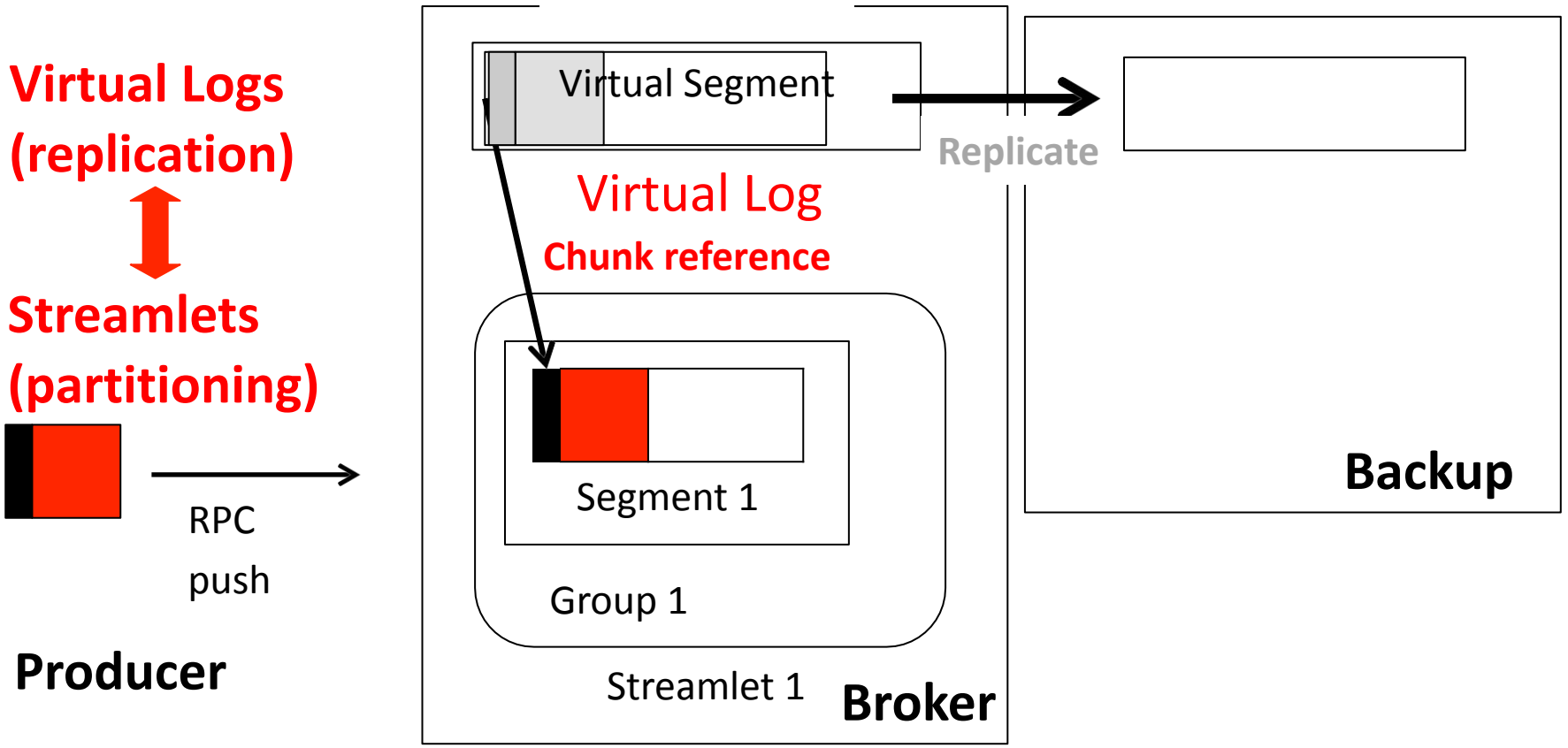
100 Brokers 10K Streams
32 Streamlets Q=16 Groups
Replication 3

Backups
256 Segments 8 MB



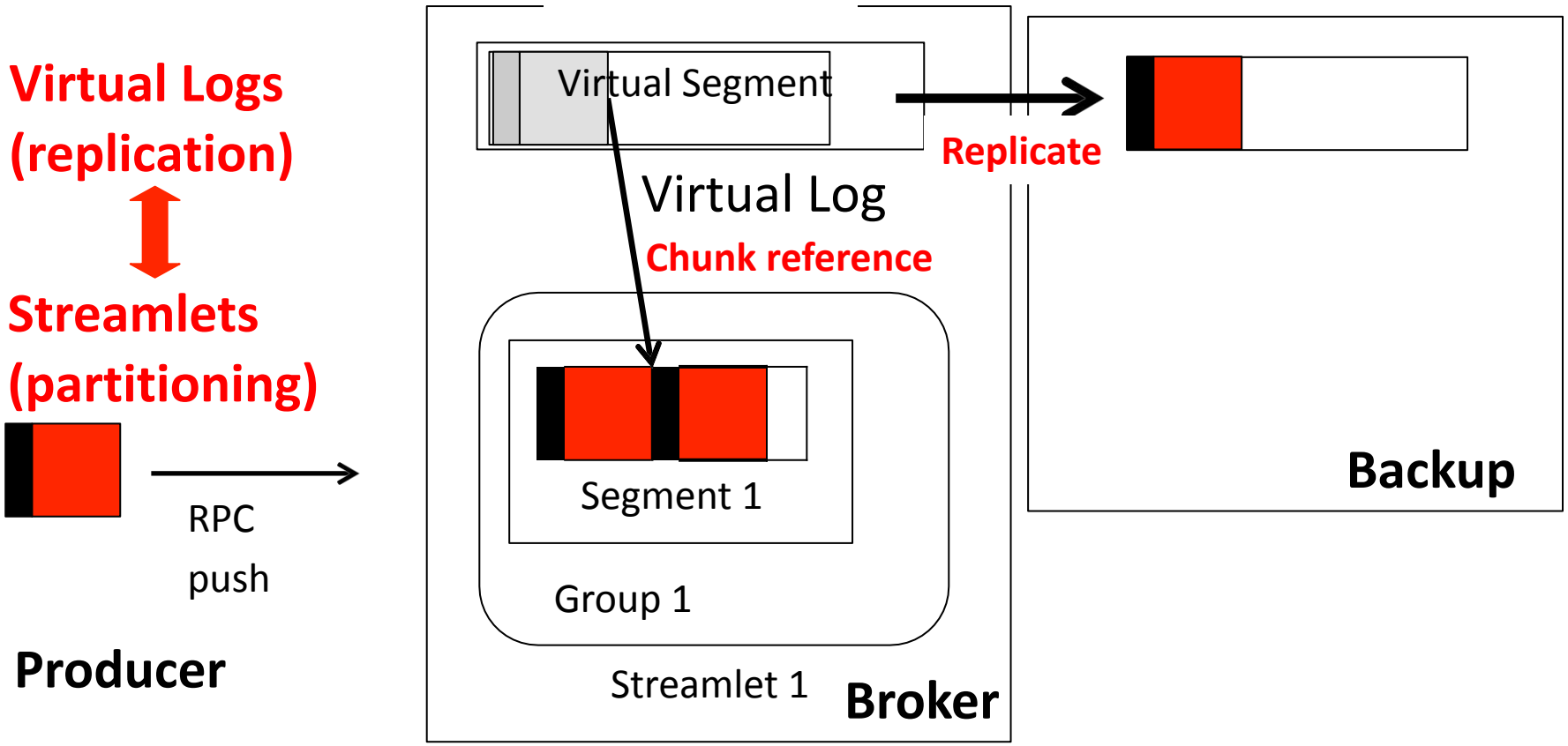
Credits: Yacine Taleb

Transparent and Fine-grained Replication: Distributed Virtual Logs



Ensuring consistent ingestion: chunk metadata + distributed virtual log

Transparent and Fine-grained Replication: Distributed Virtual Logs

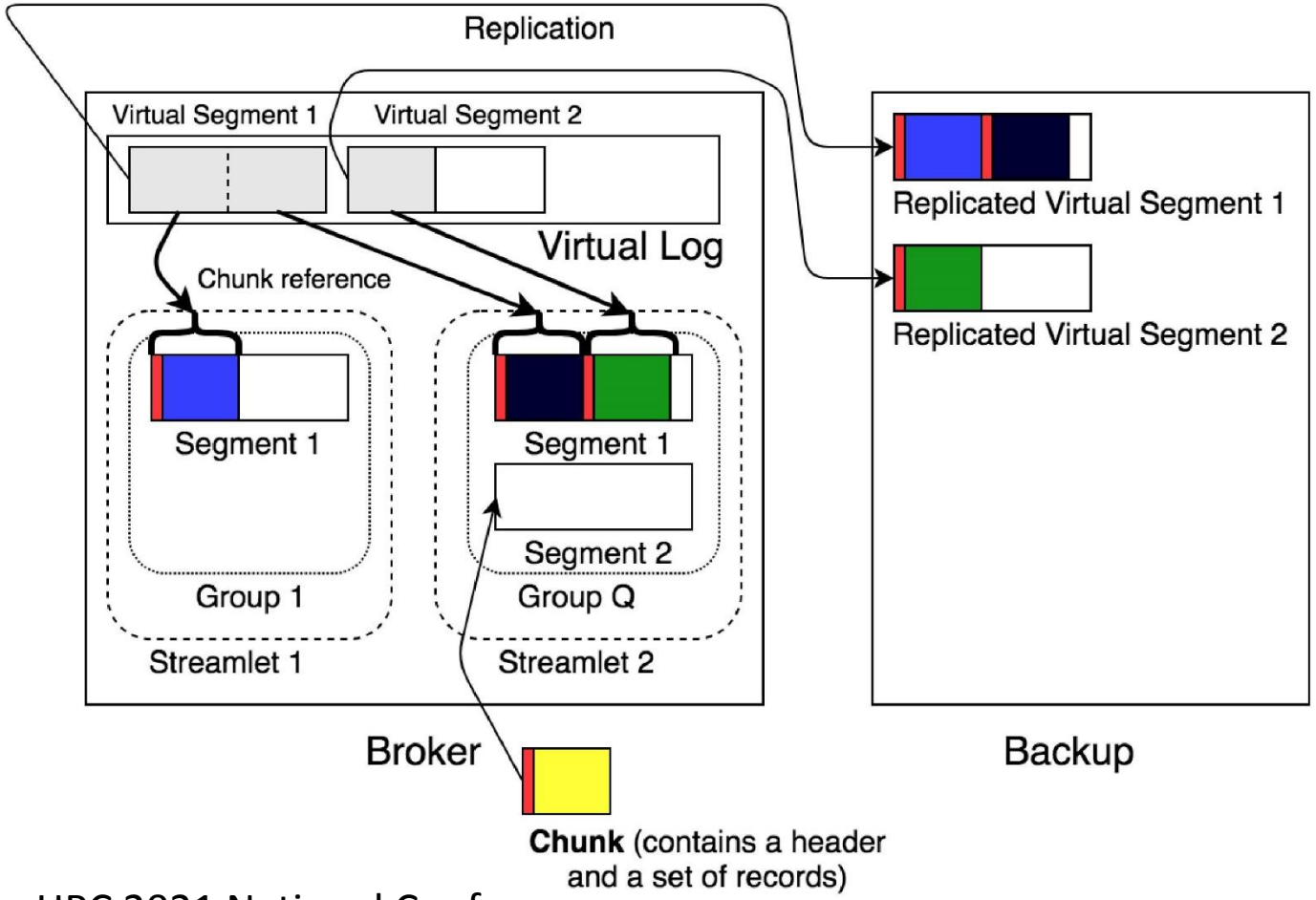


Multiple virtual logs can be associated with one to many streamlets

Our Proposal: The Virtual Log

Separate partitioning and replication

Stream Partitions are associated with replicated shared virtual logs



Experimental Setup

Methodology

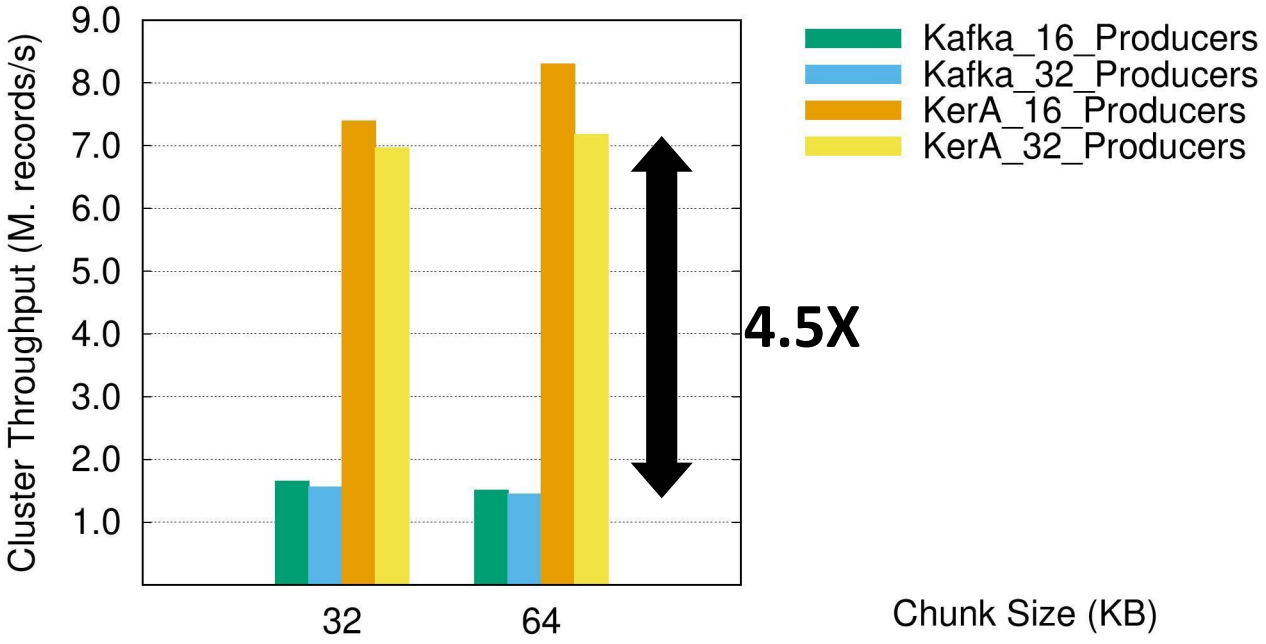
- Metric: aggregated clients' throughput
- How: clients issue write/read requests over TCP
- What:
  - Record size: 100 B
  - Partitioning strategy: round-robin
- Where: Grid5000 cluster

Parameters

- Number of streamlets (stream partitions)
- Chunk size (gives request size)
- Number of brokers
- Number of clients (producers and consumers)

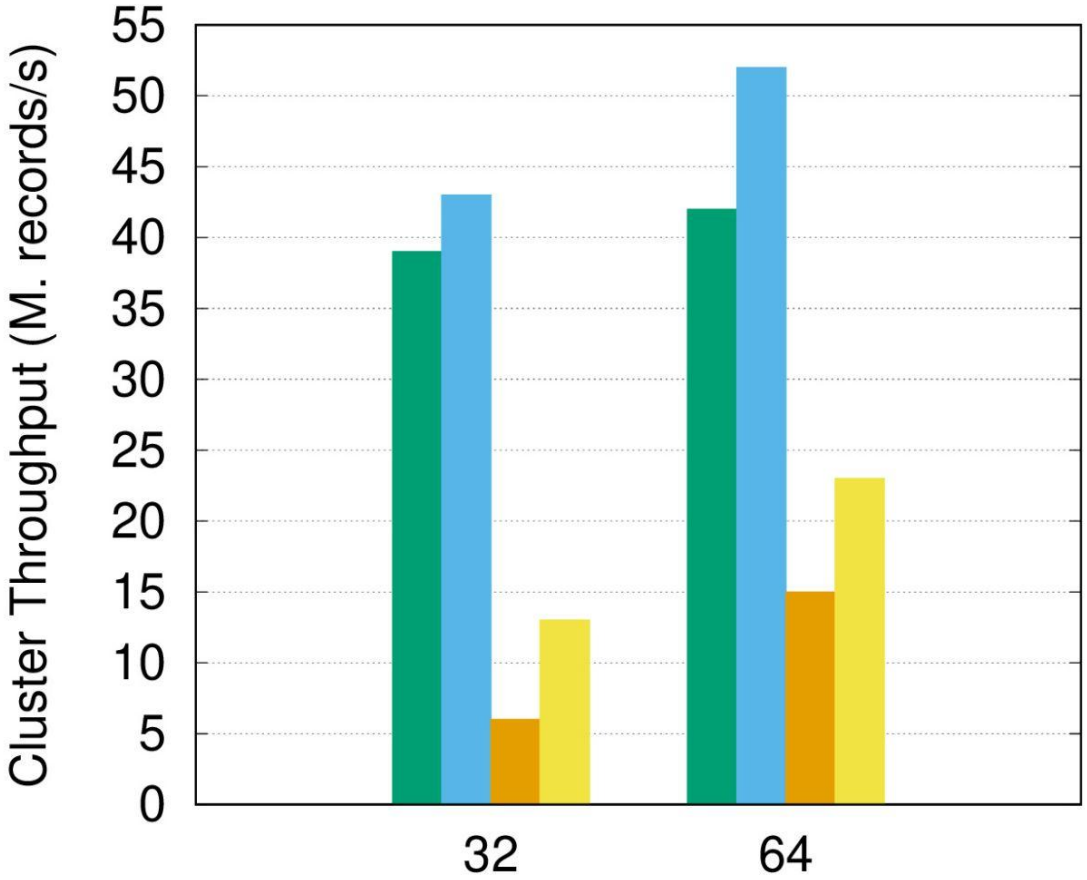
4 Brokers
128 GB RAM
16 CPU Cores
10 Gbps Network

Replicated KerA versus Kafka: High-throughput Configuration



Replication factor 3, 32 partitions per stream in Kafka,
32 streamlets with 4 groups in KerA (one virtual log per group)

In-memory Replication Infiniband vs TCP

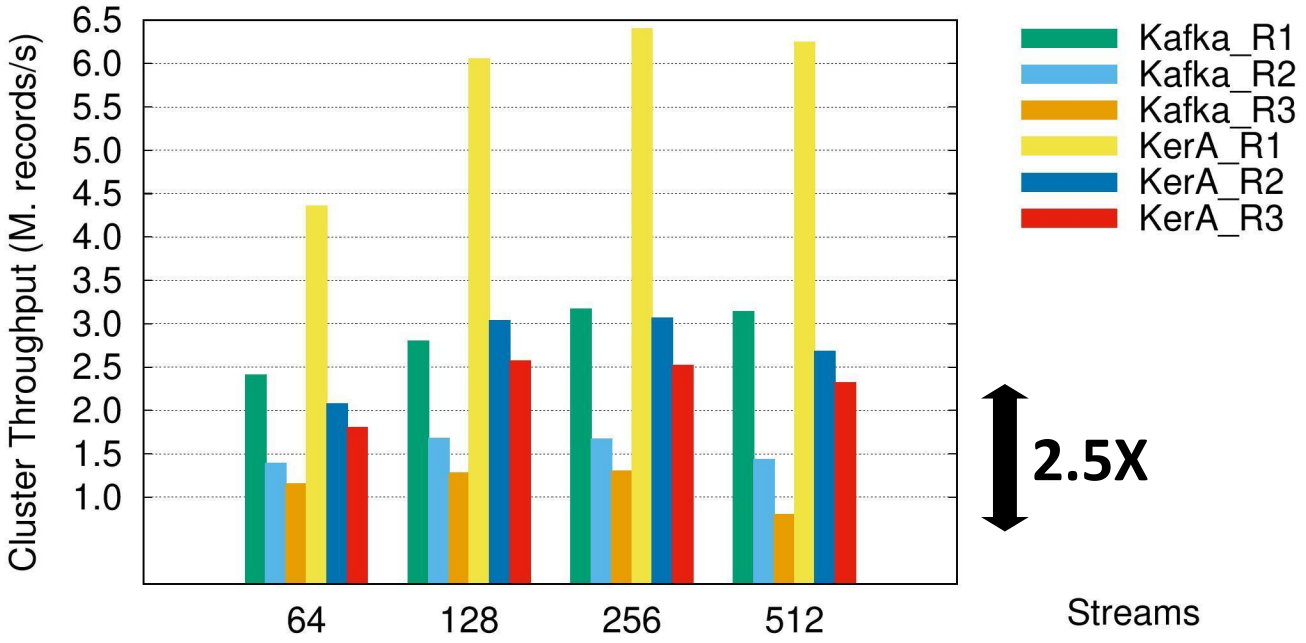


2 AMD Epyc ROME 7H12 CPU (64 cores), 256 GB of RAM

Replication factor 3, 32 streamlets with 4 groups in KerA (8 virtual logs per broker)

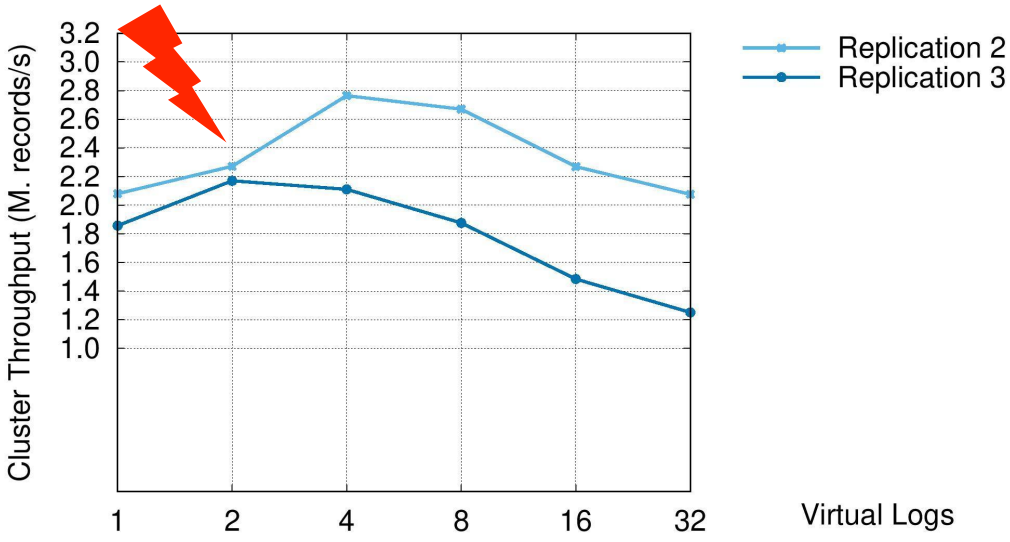
HPC aion uni.lu, 100Gb Infiniband network, Singularity containers (RC QP)

Replicated KerA versus Kafka: Increasing the Number of Streams

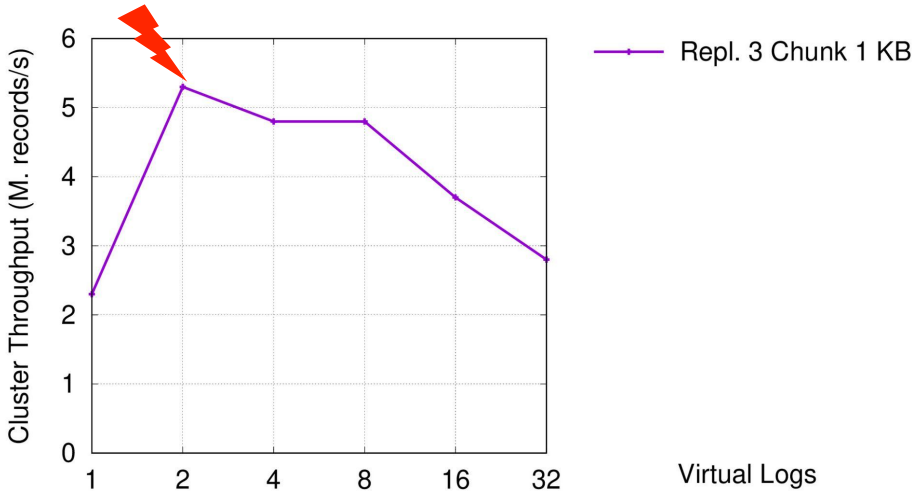


Chunk size 1 KB, 1 Partition per Stream, 4 virtual logs, 4 producers

Impact of the Virtual Log on Performance

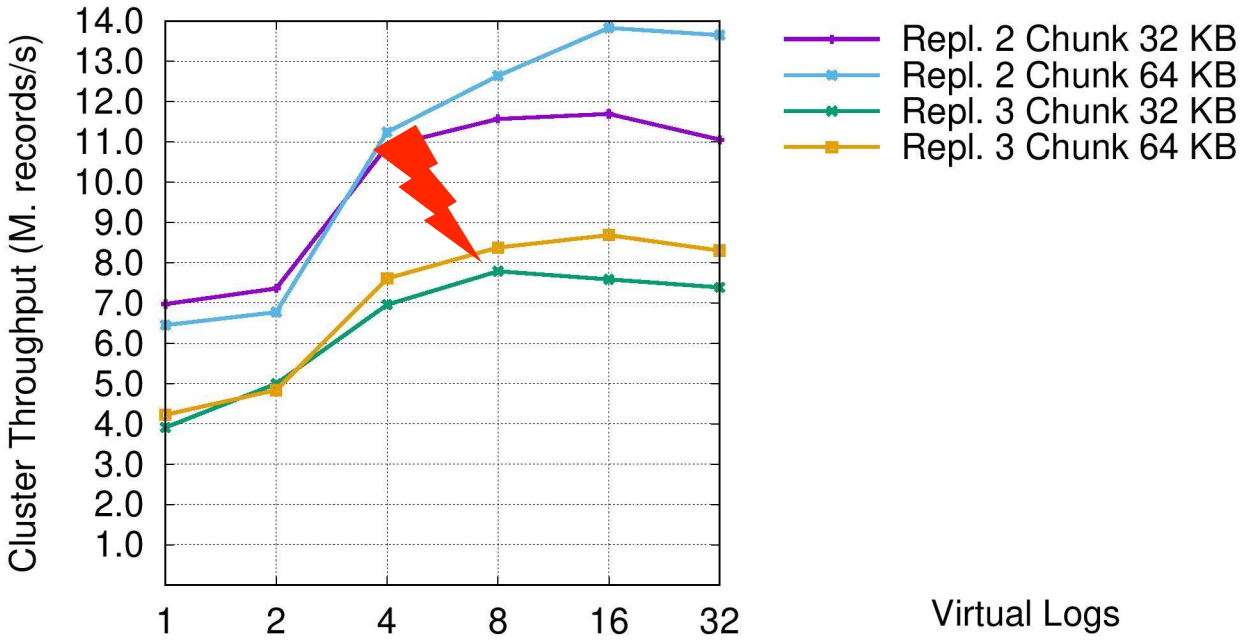


KerA low-latency, 512 streams, chunk 1 KB, 8 producers with 8 consumers



On HPC AION uni.lu cluster with Singularity and 100Gb Infiniband

Impact of the Virtual Log on Performance



KerA high-throughput, 32 partitions per stream, 8 producers with 8 consumers

Conclusion

Separating Stream Partitioning and Replication helps increase the Cluster Throughput
Tuning the Virtual Log replication capacity depends on workload and cluster configurations/available hardware.

Future Work:
Latency versus Throughput (auto-tuning)
Crash recovery
Co-located stream storage and processing

Deployments on HPC with Singularity and Infiniband

Q&A

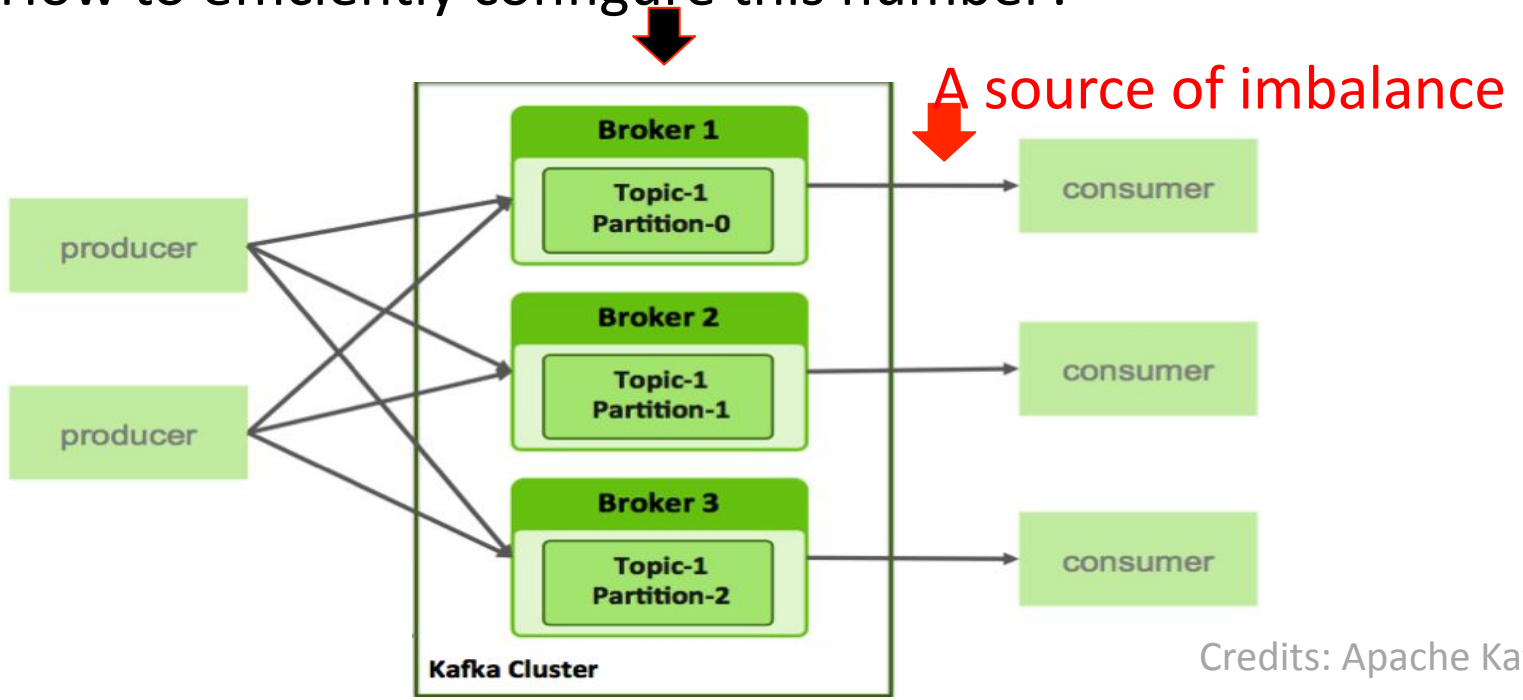
Open source:
https://gitlab.uni.lu/omarcu/zettastreams

Contact: ovidiu-cristian.marcu@uni.lu

Static Partitioning in Apache Kafka

In Apache Kafka, the number of partitions is configured statically

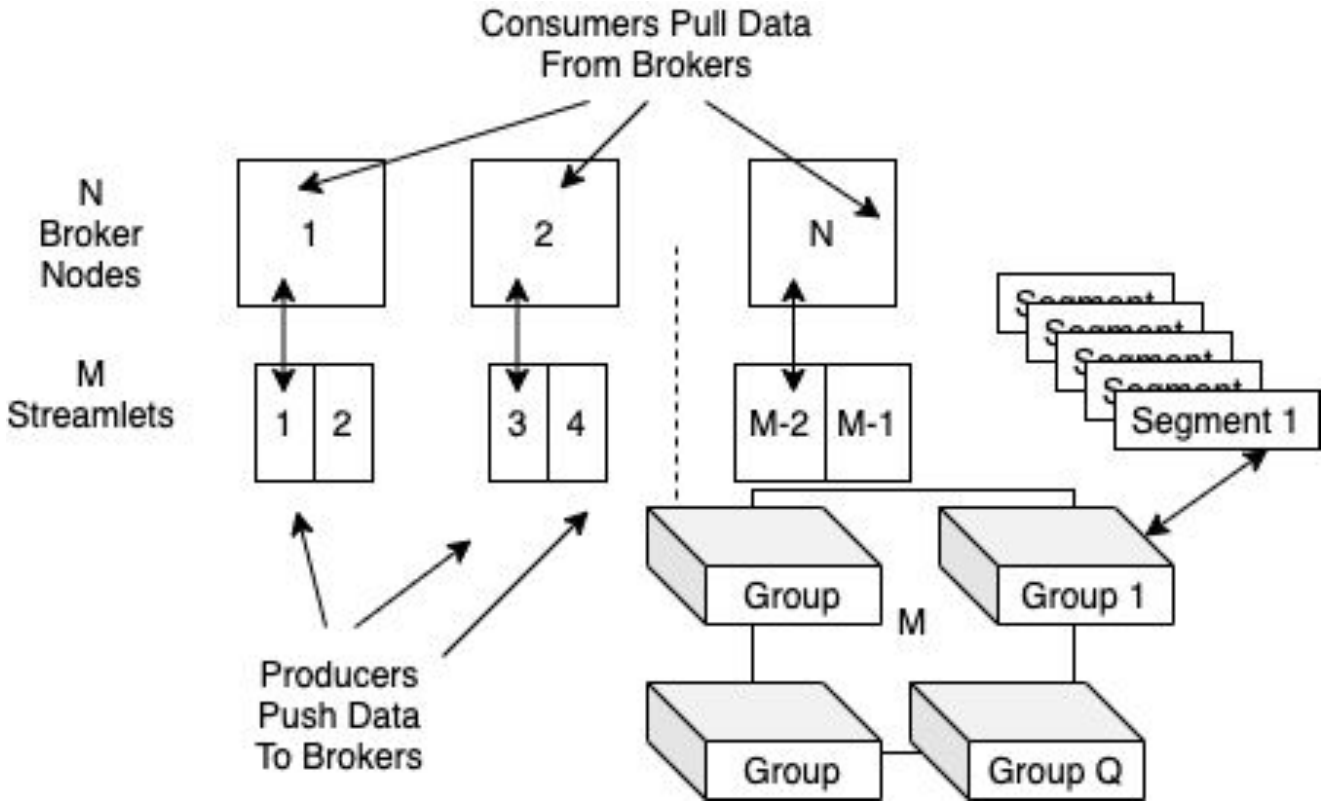
- How to efficiently configure this number?



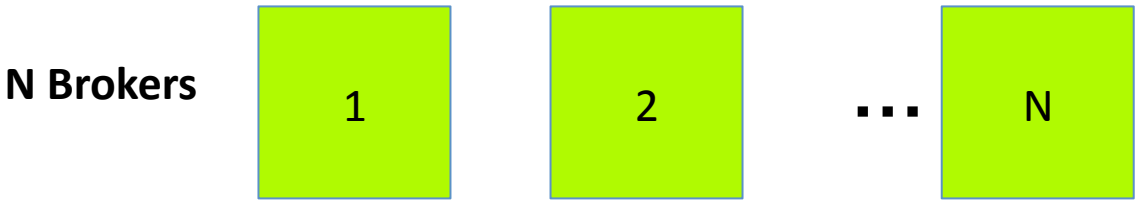
Credits: Apache Kafka

Need for dynamic stream partitioning

Dynamic Stream Partitioning in KerA



Dynamic Stream Partitioning in KerA



Create Stream:

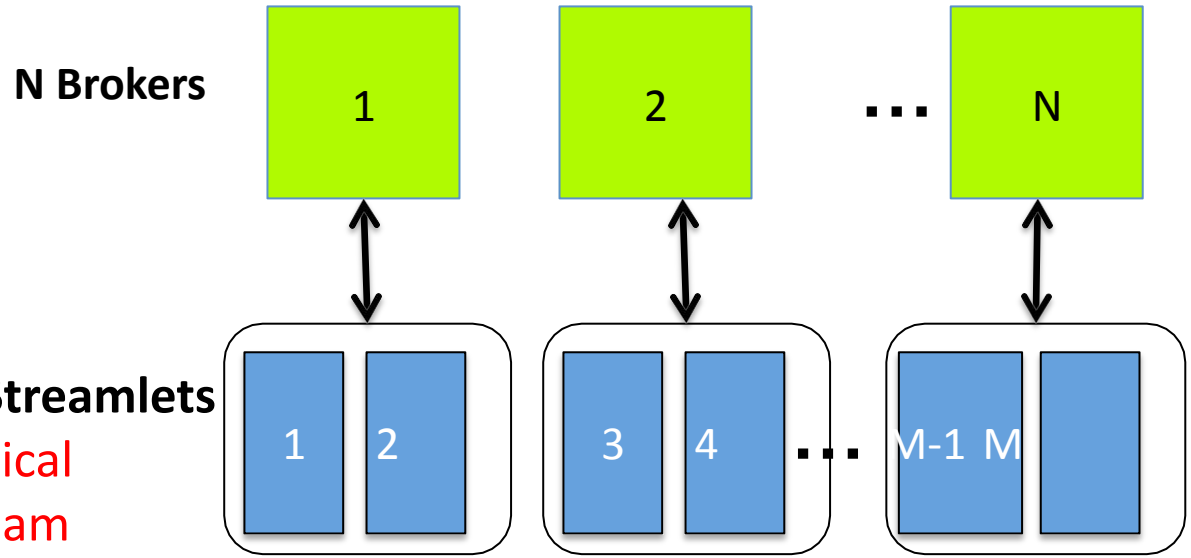
Topic name,

N Brokers,

M Streamlets,

M>=N

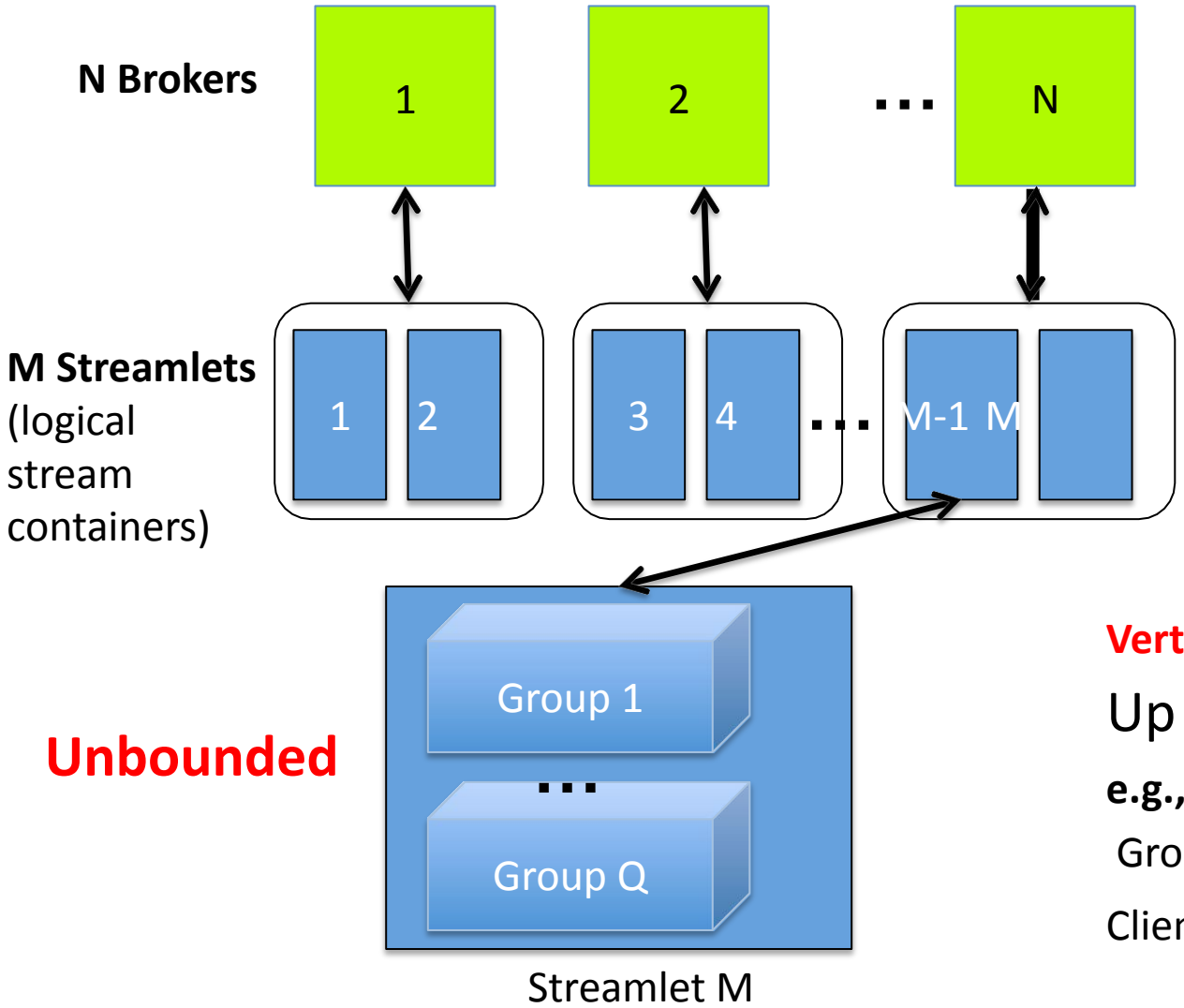
Dynamic Stream Partitioning in KerA



Create Stream:
Topic name,
N Brokers,
M Streamlets,
M>=N

Horizontal Scaling
Up to M brokers

Dynamic Stream Partitioning in KerA



Create Stream:
Topic name,
N Brokers,
M Streamlets,
M>=N

Horizontal Scaling
Up to M brokers

Unbounded

Vertical Scaling

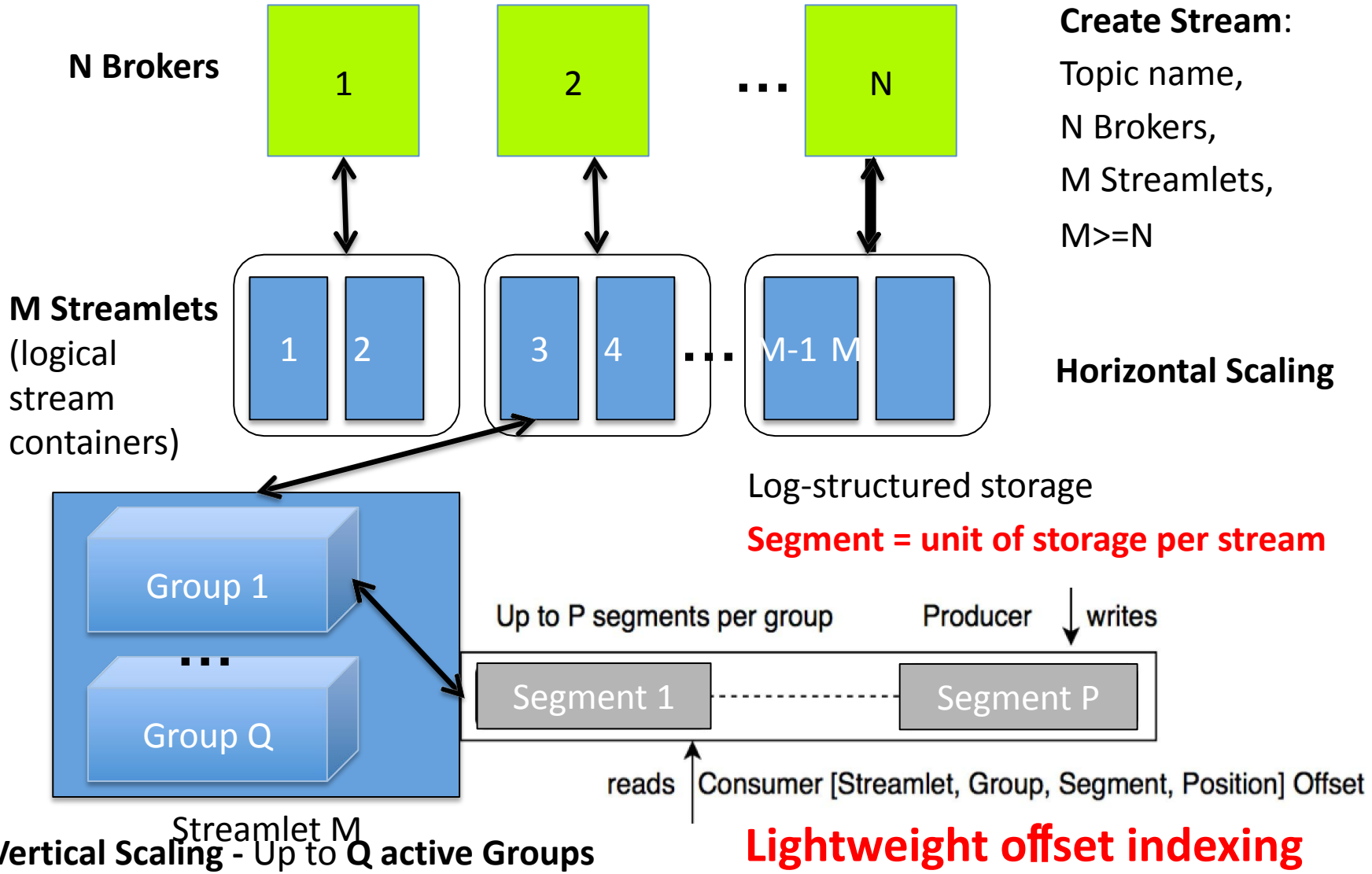
Up to **Q active Groups**

e.g., Q = number cores

Groups = fixed sub-partitions

Clients parallelism!

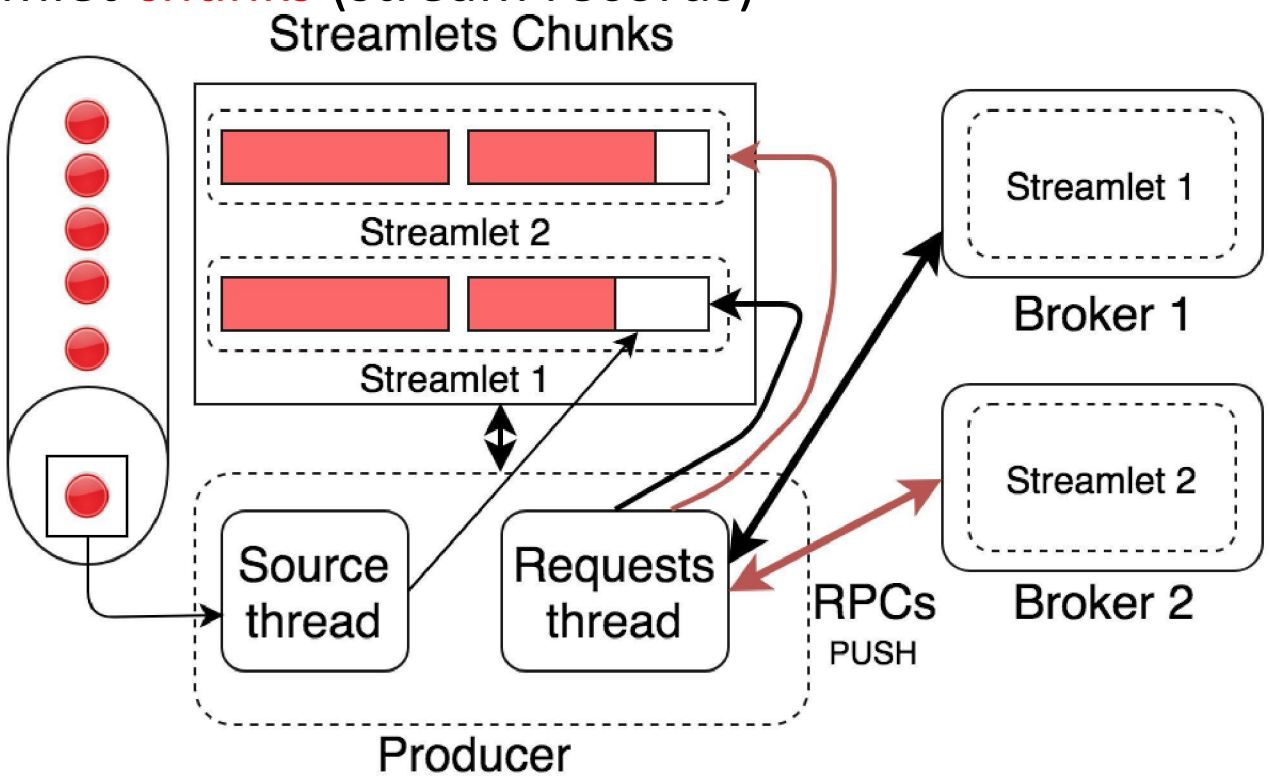
Dynamic Partitioning in KerA



Producer Architecture

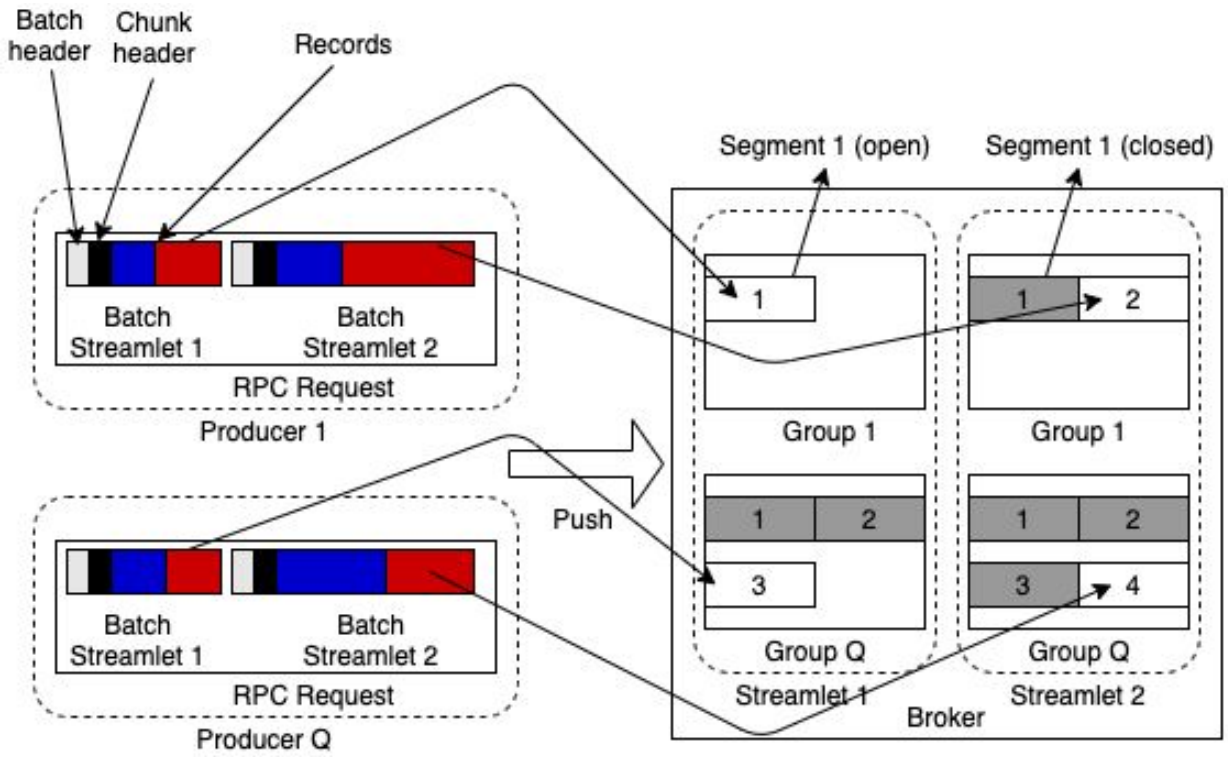
Streamlet chunks (stream records)

Latency vs Throughput



Chunk size, Request size, Chunk timeout, RPC requests

Writing Chunks from Producers to Brokers



How to organize stream partitions for faster replication and better throughput compared to using one replicated log per partition approach?