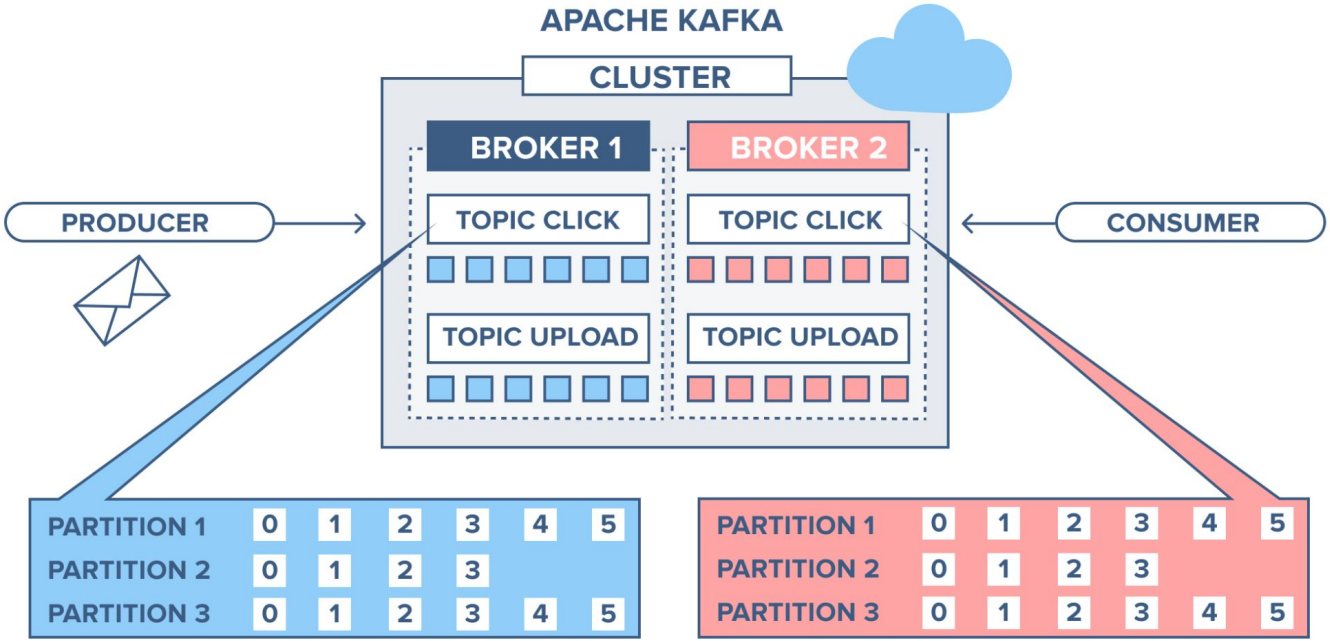
Nil-Ext* Interface for Kafka

Akshat Sinha, Himanshu Sagar, Kaushik Kota

Kafka

- Brokers
- Topics
- Partitions
- Replication
- Producers
- Consumers



Kafka Overview

- Message: An message is an atomic unit of kafka.
- Kafka organizes and durably stores messages across its system as collections of messages called Topics.
- Kafka Producer publishes messages to a Kafka Topic.
- Kafka Cluster receives the message and is responsible to for storing across the replicas.
- Kafka Consumer subscribes to a topic, read and process the messages.

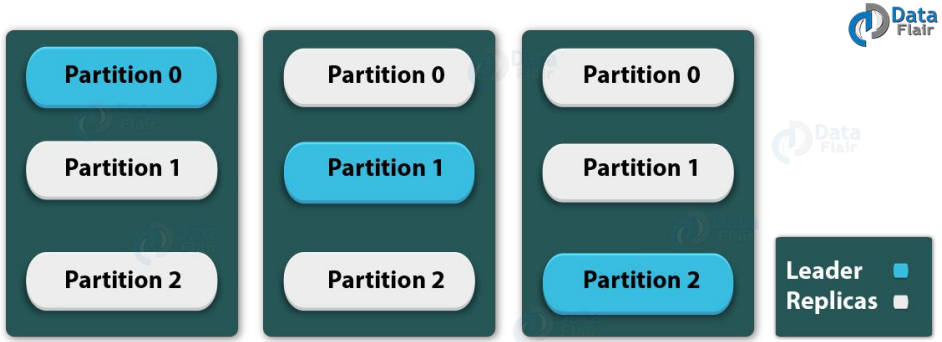
Replication and Partitioning of Topic in Kafka

Each topics is divided into partitions across replicas.

Each partition can be either a leader or follower for each topic.

Reads and Writes are sent to leader of the partition.

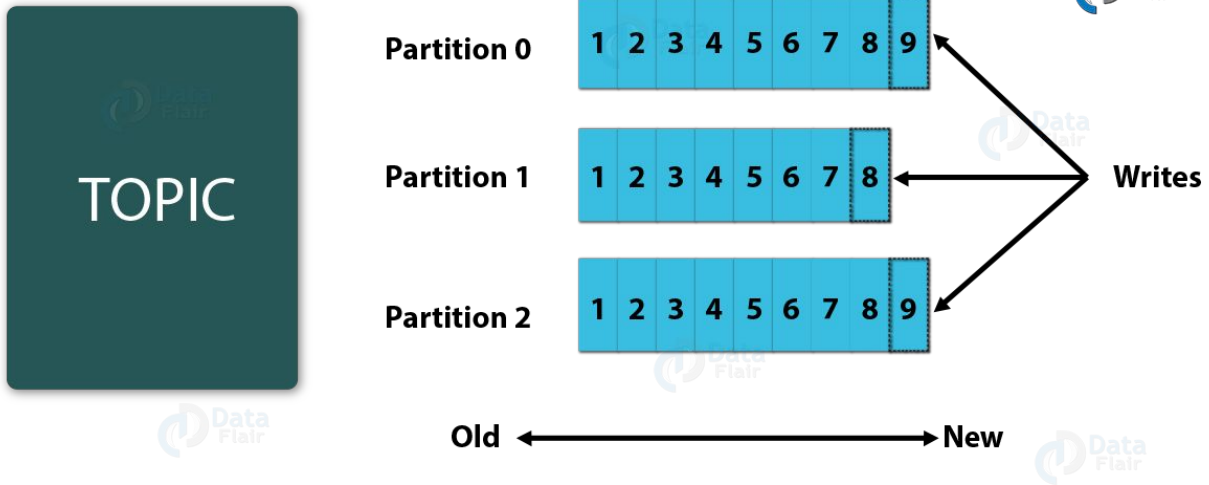
Eg. Topic with 3 Partitions.



Kafka Producer

Client Applications (Producer) publish messages to Kafka

Producer Partitioner can map each message to a particular partition and producer sends request to leader of that partition.

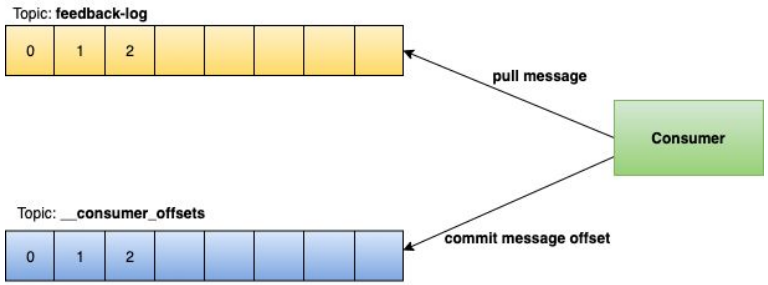


Kafka Consumer and Consumer Groups

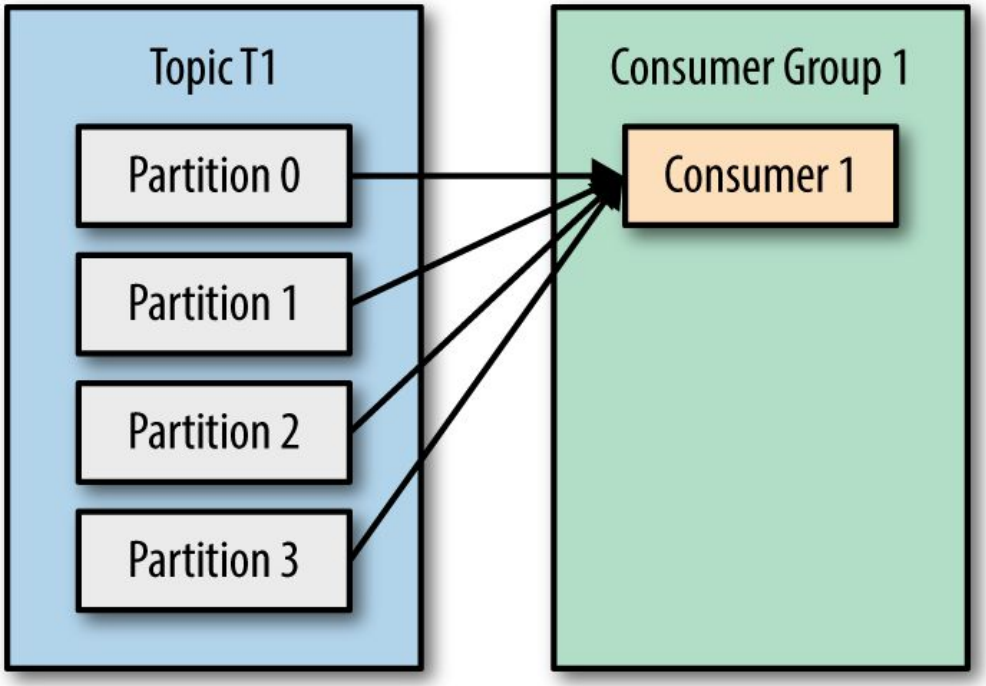
Kafka consumers subscribe to multiple topics, read and process messages.

Each consumer reads messages and keep track of last read message and updates the consumer_offset for the topic.

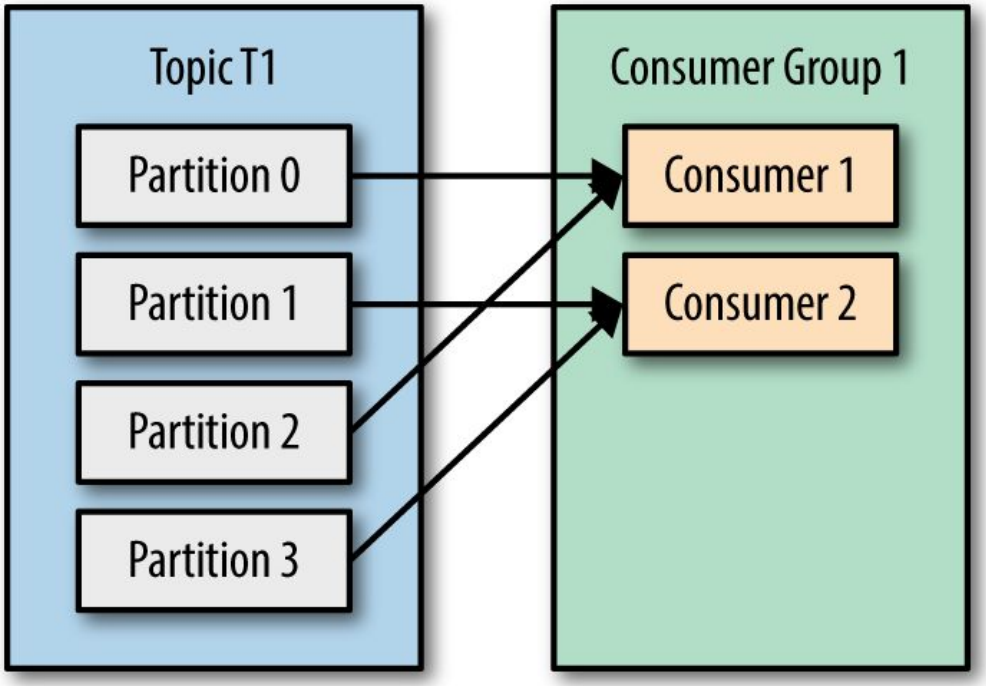
This ensures we don’t read a message again if consumer goes down.



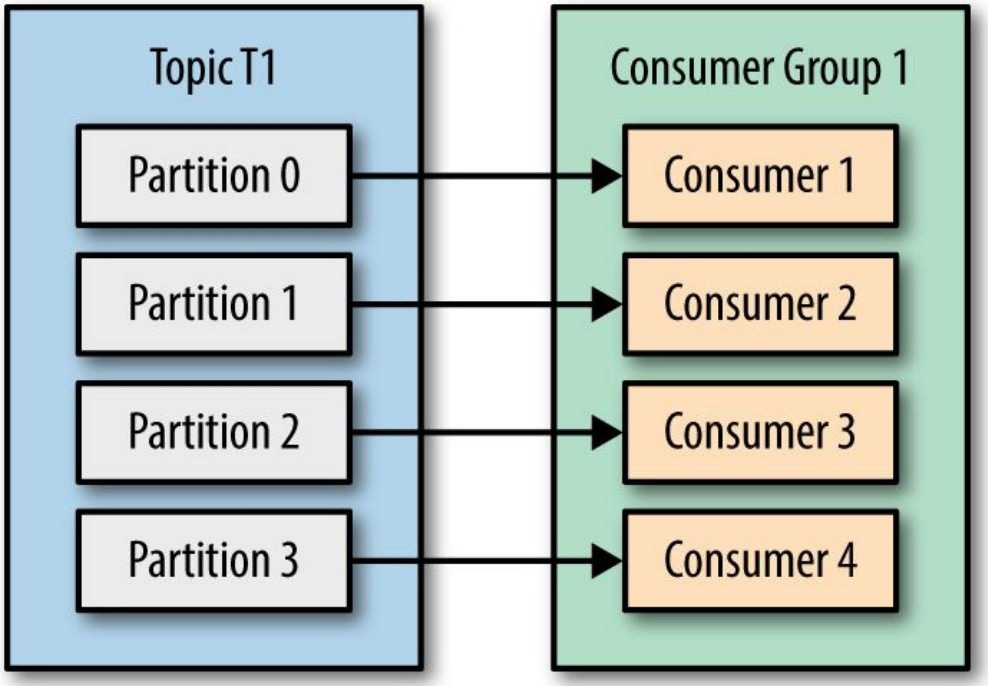
Consumer Group



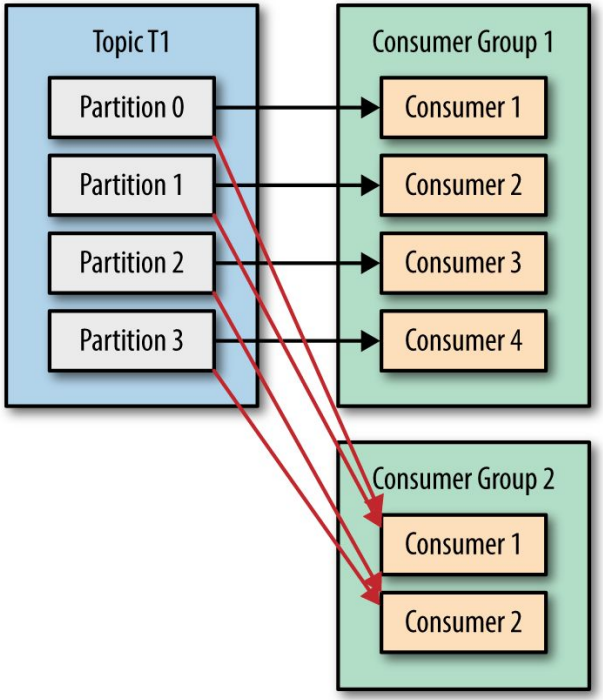
Consumer Group



Consumer Group



Consumer Group



Consumer Offsets

Allowed to change offset values of an entire group to:

- latest
- earliest
- slide by some offset

But, we cannot do random reads message of a particular offset in a partition of topic. Hence, we cannot consume randomly for a partition.

Kafka Interface

All Write message requests for a Kafka Topic are handled leader of that partition.

The acks setting in Producer Configuration denotes number of brokers that must receive the record before considering a write successful.

It takes a total of 2 ‘RTTs’ in ack = ‘all’ mode

Mode	Condition
all	Ack from all replicas
1	Ack from leader
0	Ack from none

Our Work - Almost Nil-Ext Interface

In nil-ext interface, each producer sends message request to all replicas and wait for the acknowledgment of message.

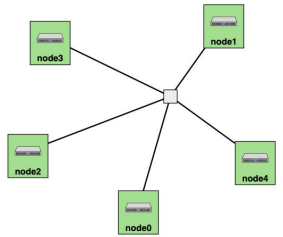
Producers considers a write successful when:

1. Receives acknowledgment from supermajority of the followers(store in a hash map).
2. Write is successful at the leader(non-nilext)
3. Order of messages is sent asynchronously

We are able to serve the write request in 1 RTT and ALL reads in 1 RTT.

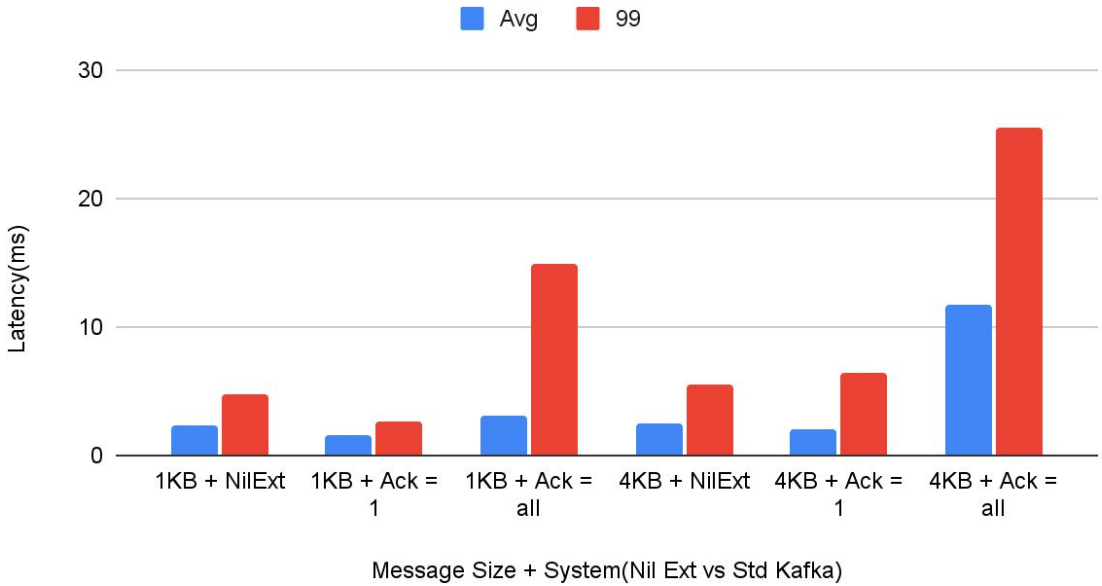
Experimental Setup

- 5 CloudLab Nodes
- Hardware Type: c220g2
  - CPU: Two Intel E5-2660 v3 10-core CPUs at 2.60 GHz
  - RAM: 160GB ECC Memory (10x 16 GB DDR4 2133 MHz dual-rank RDIMMs)
  - Disk: One Intel DC S3500 480 GB 6G SATA SSDs + Two 1.2 TB 10K RPM 6G SAS SFF HDDs
  - NIC: Dual-port Intel X520 10Gb NIC (PCIe v3.0, 8 lanes) + Onboard Intel i350 1Gb
- Link Bandwidth(reduced): 100MB/s otherwise 1GB/s



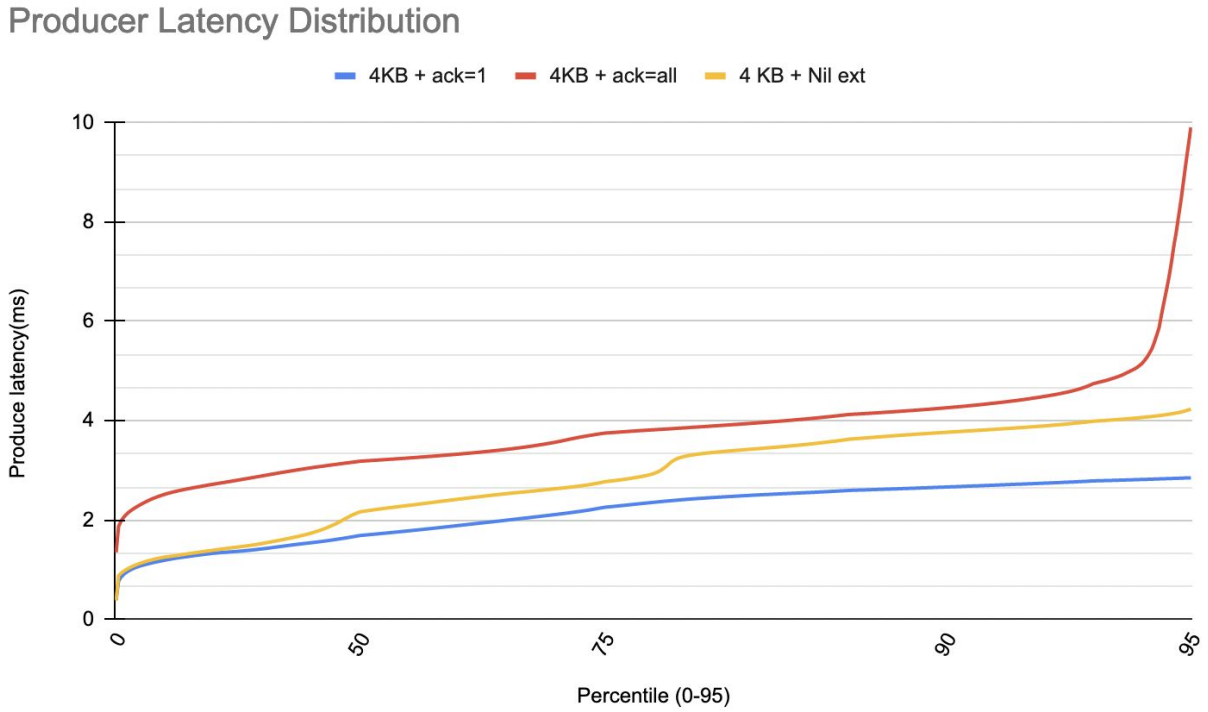
Experiment 1

Latencies for diff msg sizes and modes.



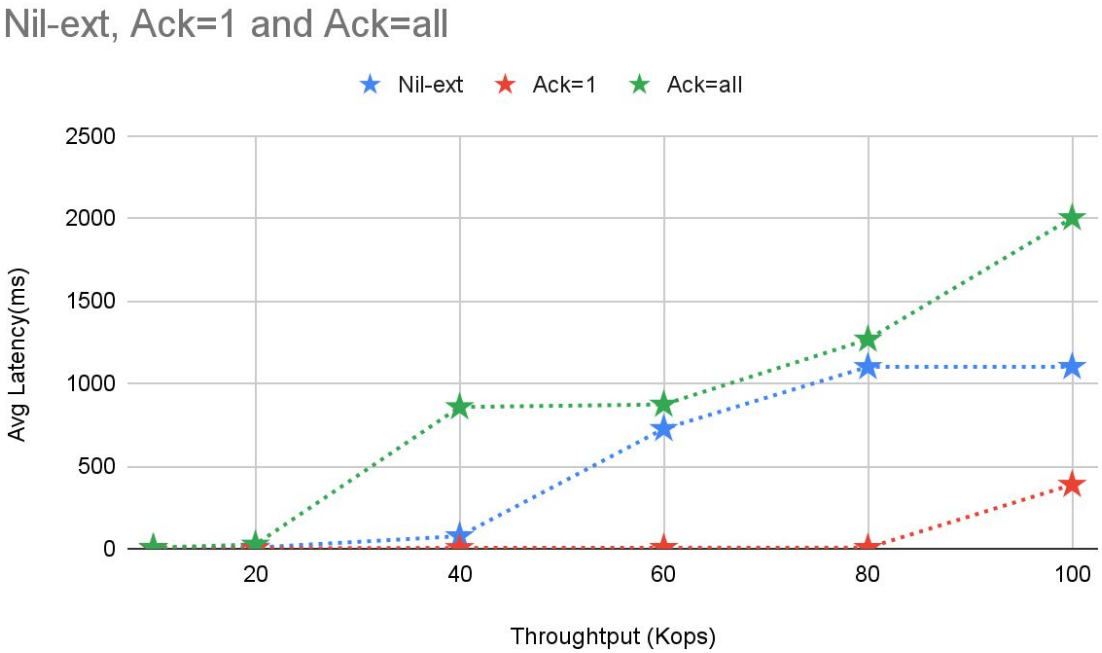
Observe impact of message sizes and ack=1,all and nil-ext modes.

Experiment 2



Latencies for diff msg sizes and modes.

Experiment 3

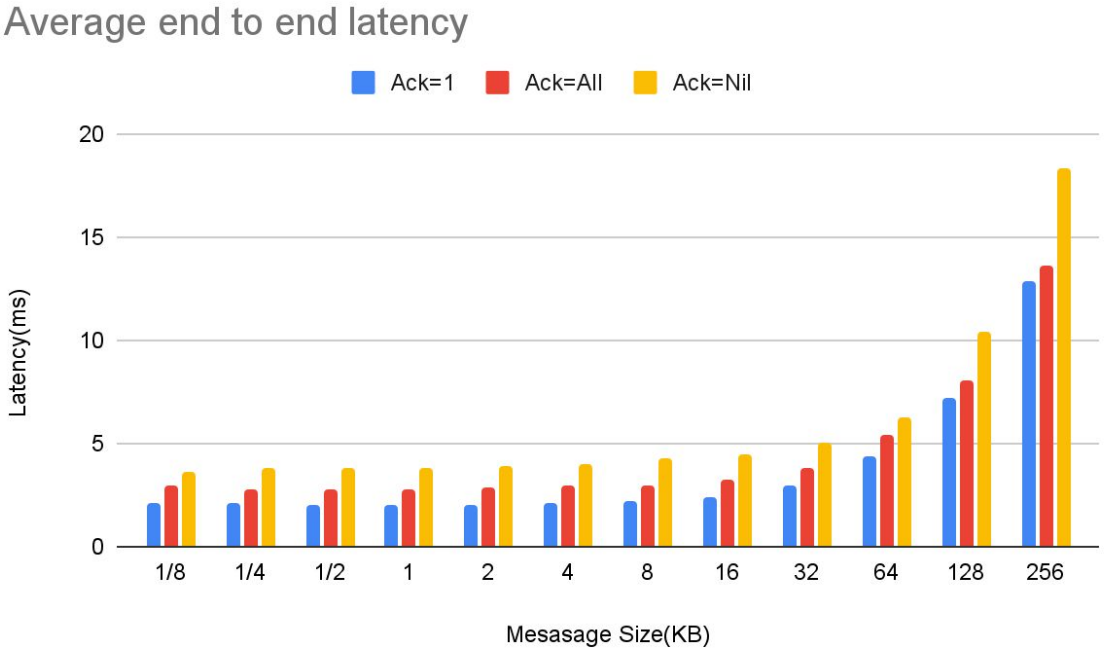


Effect of increasing throughput(Kops) on latency.

Message size: 1KB

Reduced network bandwidth: 100 MB/s

Experiment 4



Effect of message size(KB) on average end to end latency.

Negative results.

More results in Report

Slow node’s impact on performance

Scalability Testing: Vary Replications

Experiment with Read and Write Heavy workloads comparable to YCSB

- Vary producers and Consumer Groups to simulate read/write heavy
- Issue: Benchmark has open issues with increasing consumer groups

Implementation Details

- Exactly Once: Used Produce Sequence and Leader epoch
- Linearizability is guaranteed in a partition
- Producer requests are marked successful after getting acks from super majority
- Holes of messages on followers are handled
- Failure of followers is handled using Hole filling logic
- New code stats:
  - 102 commits
  - 54 files changed, **4,000 additions** and **157 deletions**.

Future Work

Handle Leader failure

Load balance reads from followers

Thank You