

Intra-cluster Replication for Apache Kafka

Jun Rao

About myself

- Engineer at LinkedIn since 2010
- Worked on Apache Kafka and Cassandra
- Database researcher at IBM

Outline

- Overview of Kafka
- Kafka architecture
- Kafka replication design
- Performance
- Q/A

What's Kafka

- A distributed pub/sub messaging system
- Used in many places
 - LinkedIn, Twitter, Box, FourSquare ...
- What do people use it for?
 - log aggregation
 - real-time event processing
 - monitoring
 - queuing

Example Kafka Apps at LinkedIn

All Updates · LinkedIn Coworkers · Shares · More ▾ Recent · Top · 



Peter Skomoroch



Ignition, Accel, Graylock Put \$40M in Apache Hadoop Distribution...
techcrunch.com

Like · Comment · Send a message · Share · 36 seconds ago

Trending in Venture Capital & Private Equity

Add a comment...



Peter Skomoroch



Common Crawl Foundation Announces 5 Billion Page Web Index,
Available... readwriteweb.com

Like · Comment · Send a message · Share · 2 minutes ago

Trending in Computer Software and Online Media

Add a comment...



Neha Desai is now connected to Eli Smaga

Send a message · 13 minutes ago



Ajay Choudhary is now connected to Yliad Wilson (LION)

Send a message · 30 minutes ago



Ashwin Ram via Twitter 3P

ashwinram RT @daniel_kraft : @tpostz at #FutureMed on the role feedback loops in
#health [#http://it.co/OOmzJw5l](http://it.co/OOmzJw5l) #in

Retweet · Favorite · Reply

Like · Comment · Share · 40 minutes ago



voldemort-server/cluster-2-dps-rw-throughput

Create snapshot: Create

Customize Your Selection

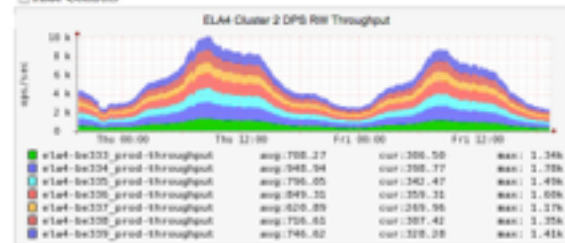
Duration: 2 days

Start Time: YYYYMMDD-HH:MM End Time: YYYYMMDD-HH:MM

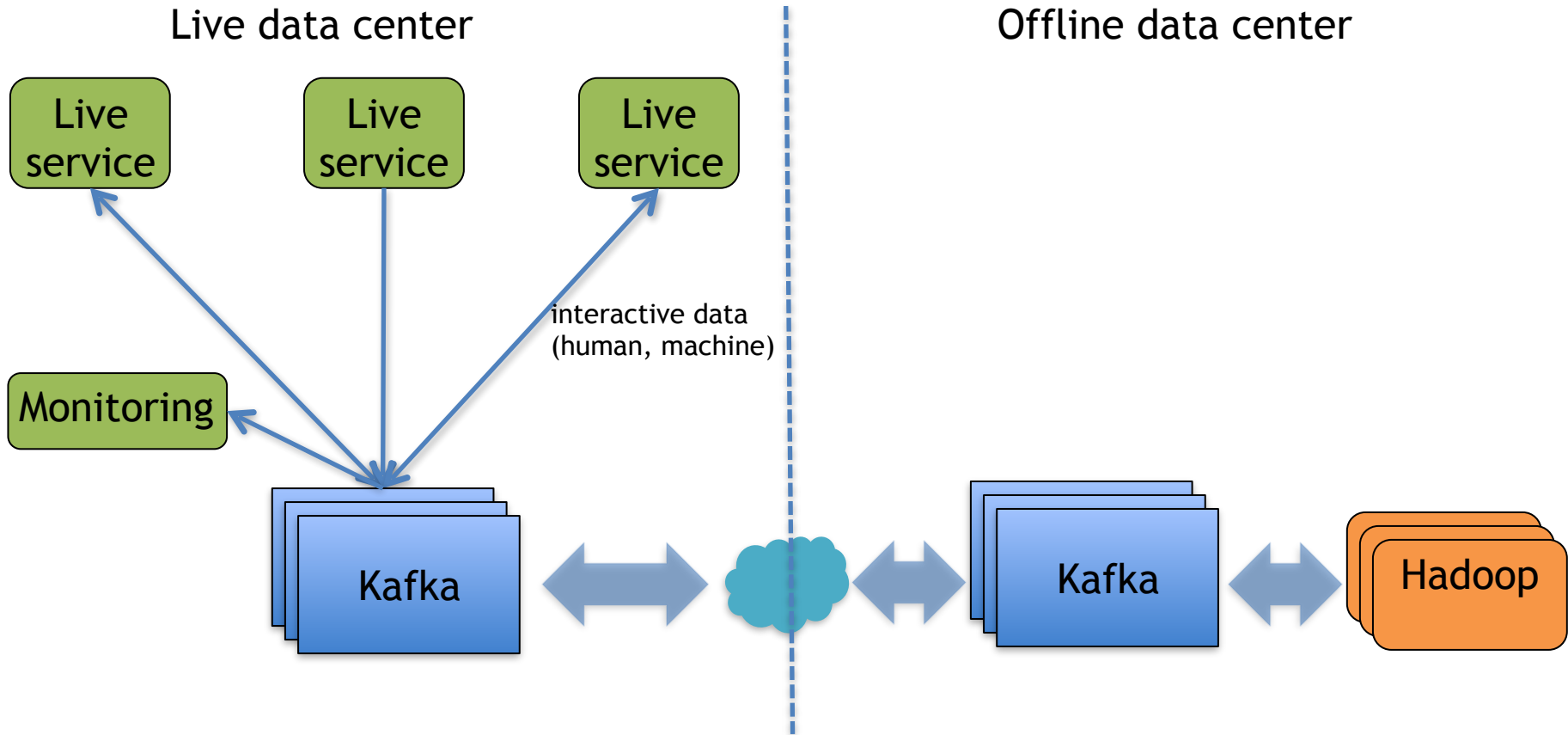
Timezone: Height: 100 px Width: 900 px Lower Limit: Upper Limit: 10000 Threshold:

Stack: ☒ Consolidate: Overlay: Legend: ☒ Alt Scale: ☐ Alt Y-axis: ☐ Auto-refresh:

Hide Controls



Kafka Deployment at LinkedIn



Per day stats

- writes: 10+ billion messages (2+TB compressed data)
- reads: 50+ billion messages

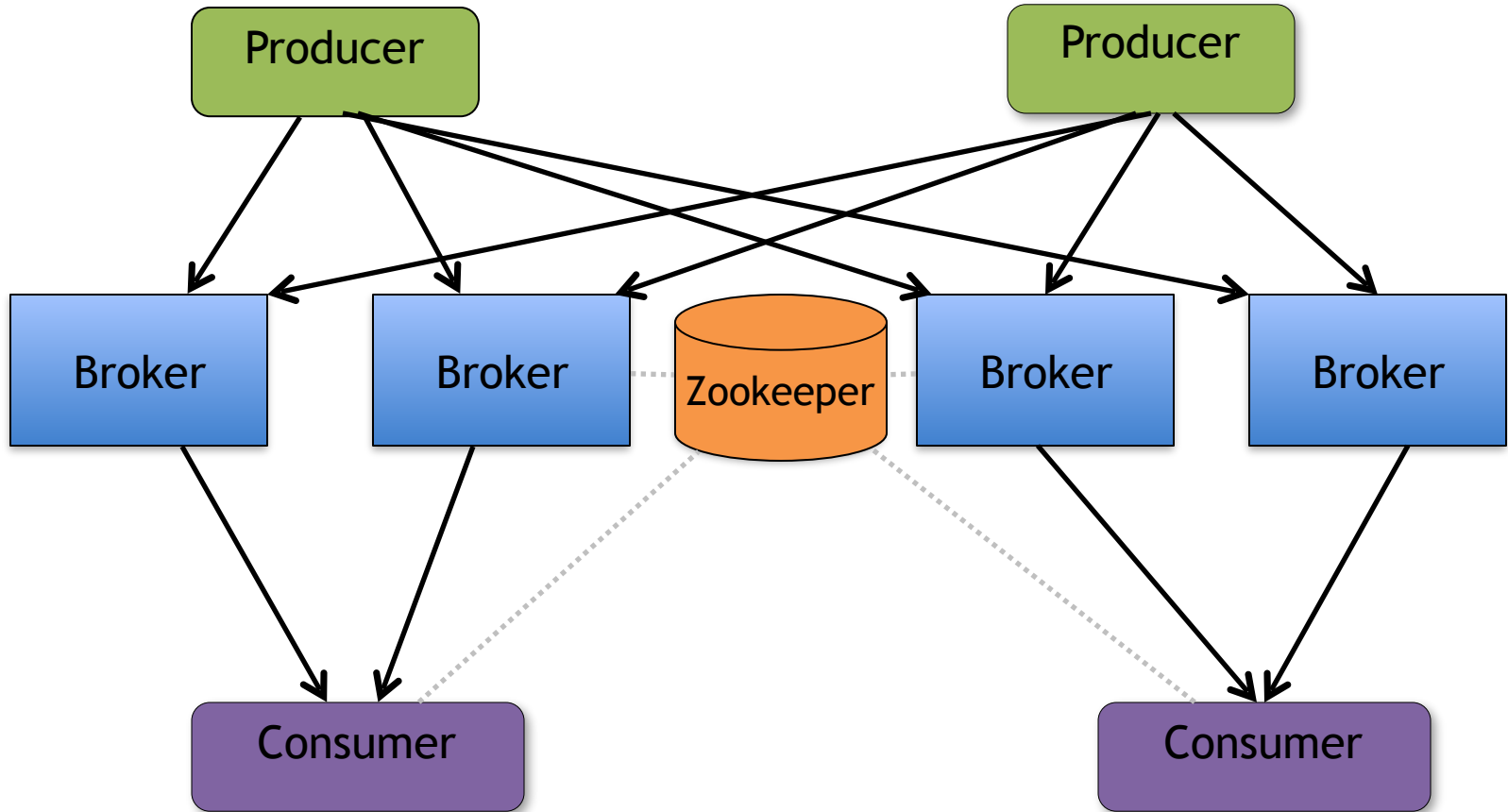
Kafka vs. Other Messaging Systems

- Scale-out from groundup
- Persistence to disks
- High throughput (10s MB/sec per server)
- Multi-subscription

Outline

- Overview of Kafka
- Kafka architecture
- Kafka replication design
- Performance
- Q/A

Kafka Architecture



Terminologies

- Topic = message stream
- Topic has partitions
 - partitions distributed to brokers
- Partition has a log on disk
 - message persisted in log
 - message addressed by offset

API

- **Producer**

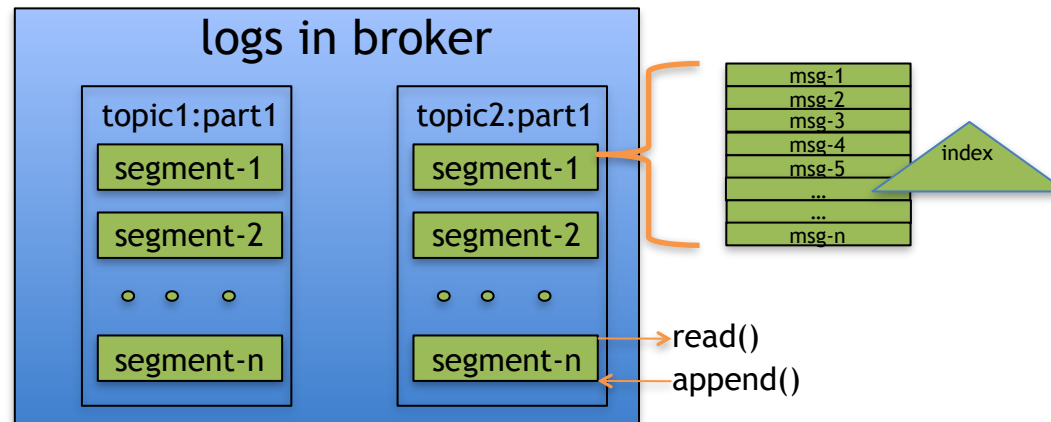
```
messages = new List<KeyedMessage<K,V>>();  
messages.add(new KeyedMessage("topic1", null, "msg1");  
send(messages);
```

- **Consumer**

```
streams[] = Consumer.createMessageStream("topic1", 1);  
  
for(message: streams[0]) {  
    // do something with message  
}
```

Deliver High Throughput

- Simple storage



- Batched writes and reads
- Zero-copy transfer from file to socket
- Compression (batched)

Outline

- Overview of Kafka
- Kafka architecture
- Kafka replication design
- Performance
- Q/A

Why Replication

- Broker can go down
 - controlled: rolling restart for code/config push
 - uncontrolled: isolated broker failure
- If broker down
 - some partitions unavailable
 - could be permanent data loss
- Replication → higher availability and durability

CAP Theorem

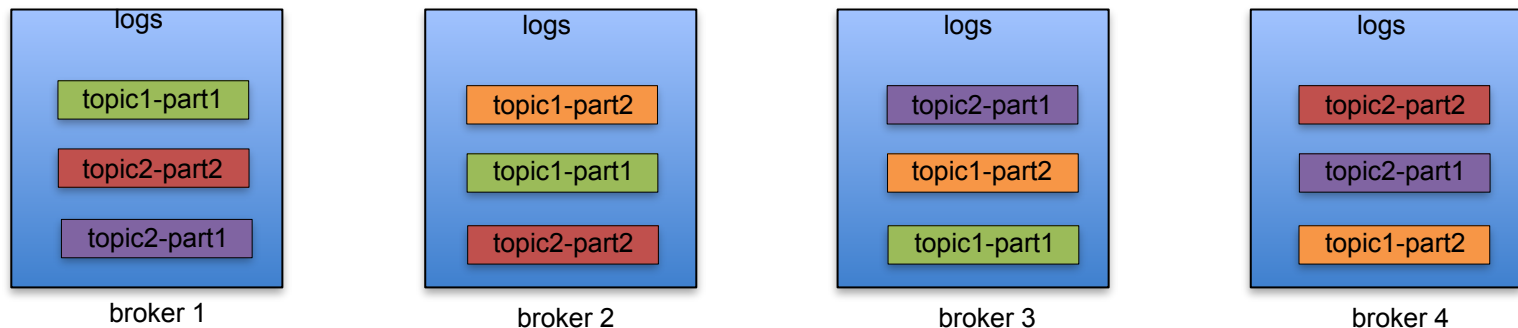
- Pick two from
 - consistency
 - availability
 - network partitioning

Kafka Replication: Pick CA

- Brokers within a datacenter
 - i.e., network partitioning is rare
- Strong consistency
 - replicas byte-wise identical
- Highly available
 - typical failover time: < 10ms

Replicas and Layout

- Partition has replicas
- Replicas spread evenly among brokers



Maintain Strongly Consistent Replicas

- One of the replicas is leader
- All writes go to leader
- Leader propagates writes to followers in order
- Leader decides when to commit message

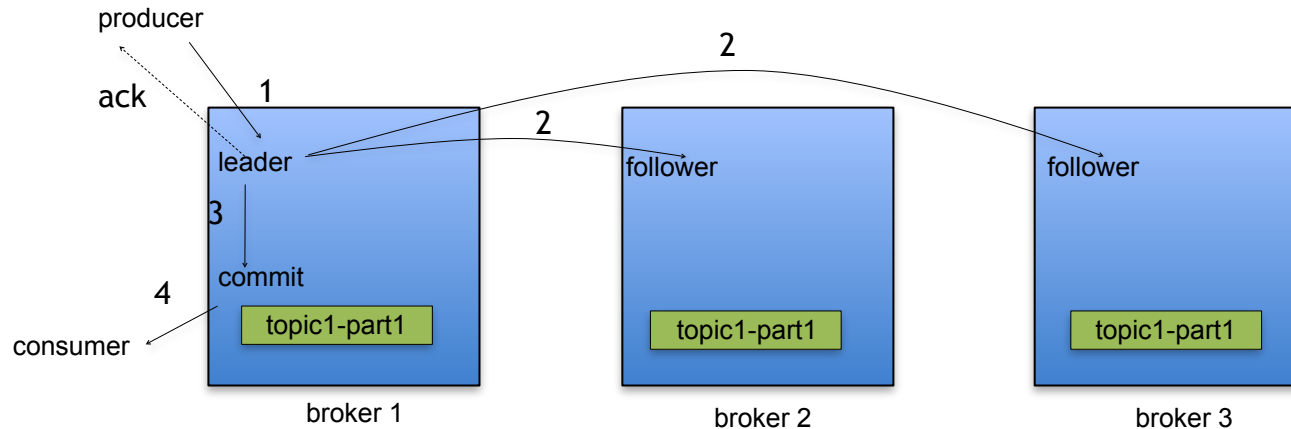
Conventional Quorum-based Commit

- Wait for majority of replicas (e.g. Zookeeper)
- Plus: good latency
- Minus: $2f+1$ replicas \rightarrow tolerate f failures
 - ideally want to tolerate $2f$ failures

Commit Messages in Kafka

- Leader maintains in-sync-replicas (ISR)
 - initially, all replicas in ISR
 - message committed if received by ISR
 - follower fails → dropped from ISR
 - leader commits using new ISR
- Benefit: f replicas → tolerate $f-1$ failures
 - latency less an issue within datacenter

Data Flow in Replication

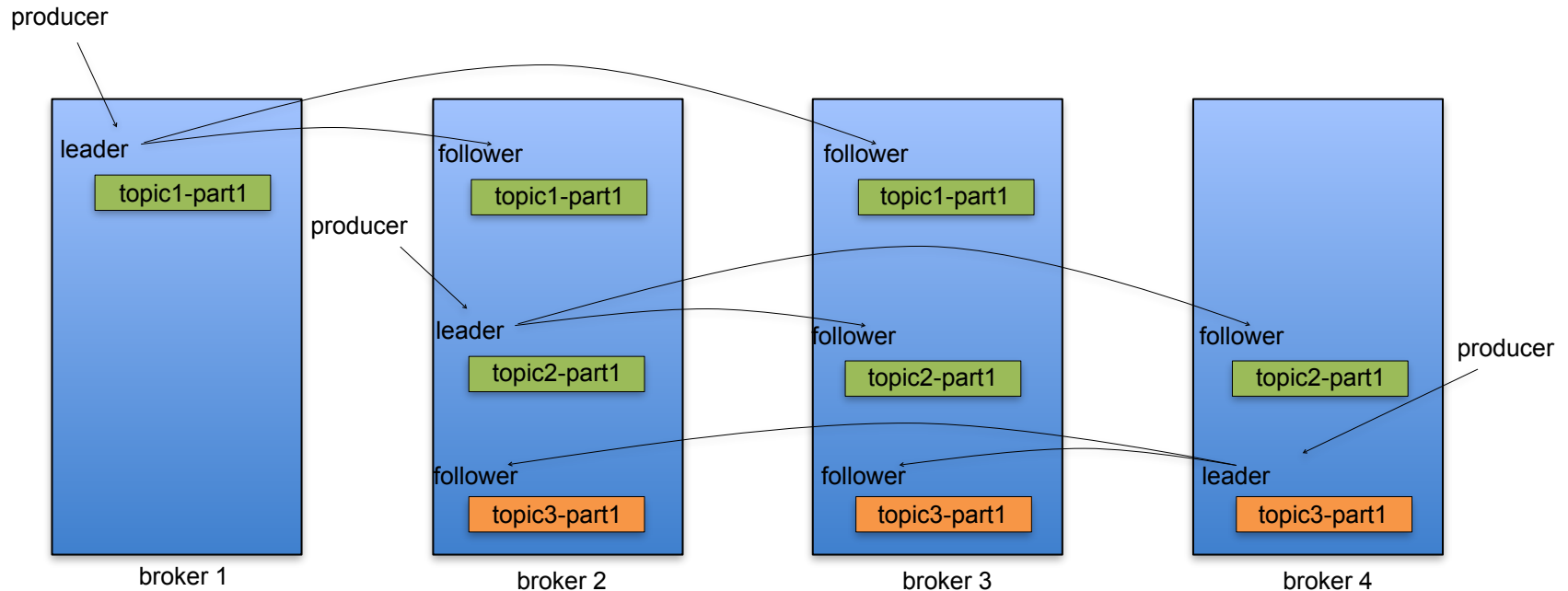


When producer receives ack	Latency	Durability on failures
no ack	no network delay	some data loss
wait for leader	1 network roundtrip	a few data loss
wait for committed	2 network roundtrips	no data loss

Only committed messages exposed to consumers

- independent of ack type chosen by producer

Extend to Multiple Partitions



- Leaders are evenly spread among brokers

Handling Follower Failures

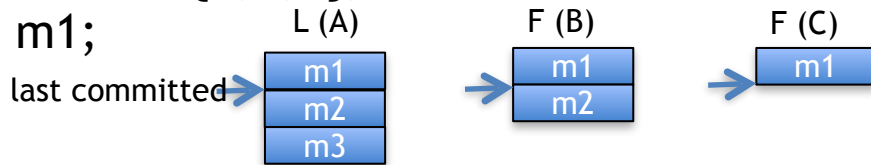
- Leader maintains last committed offset
 - propagated to followers
 - checkpointed to disk
- When follower restarts
 - truncate log to last committed
 - fetch data from leader
 - fully caught up → added to ISR

Handling Leader Failure

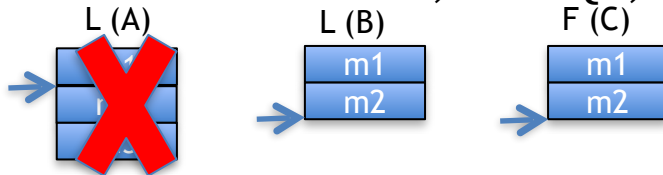
- Use an embedded controller (inspired by Helix)
 - detect broker failure via Zookeeper
 - on leader failure: elect new leader from ISR
 - committed messages not lost
- Leader and ISR written to Zookeeper
 - for controller failover
 - expected to change infrequently

Example of Replica Recovery

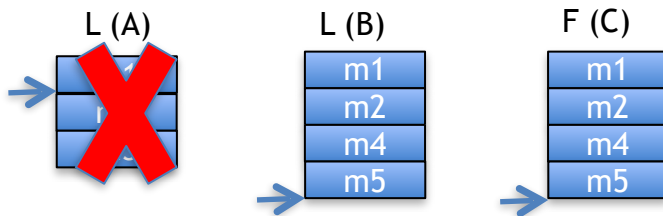
1. ISR = {A,B,C}; Leader A commits message m1;



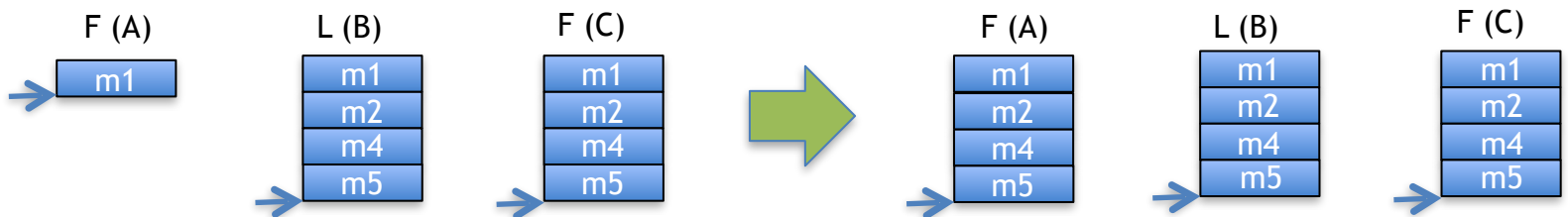
2. A fails and B is new leader; ISR = {B,C}; B commits m2, but not m3



3. B commits new messages m4, m5



4. A comes back, truncates to m1 and catches up; finally ISR = {A,B,C}



Outline

- Overview of Kafka
- Kafka architecture
- Kafka replication design
- Performance
- Q/A

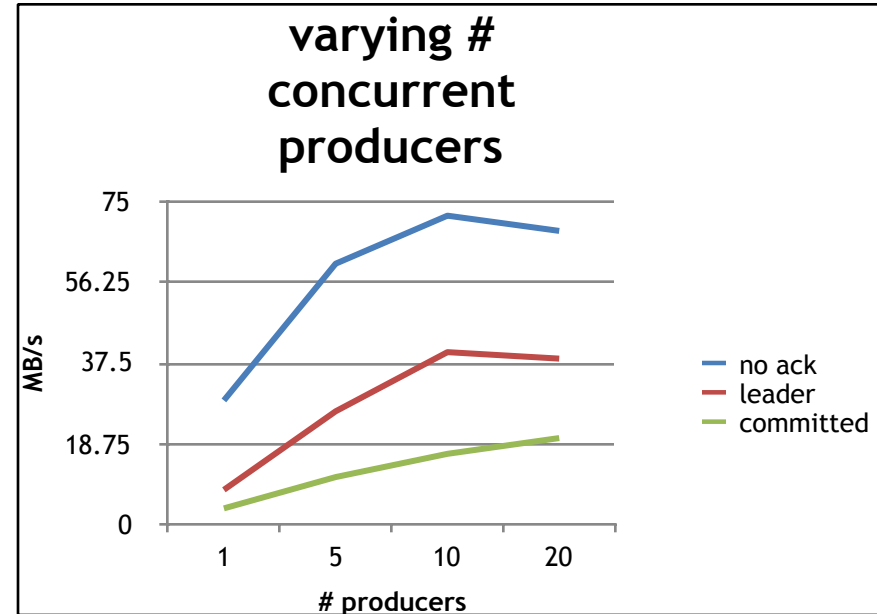
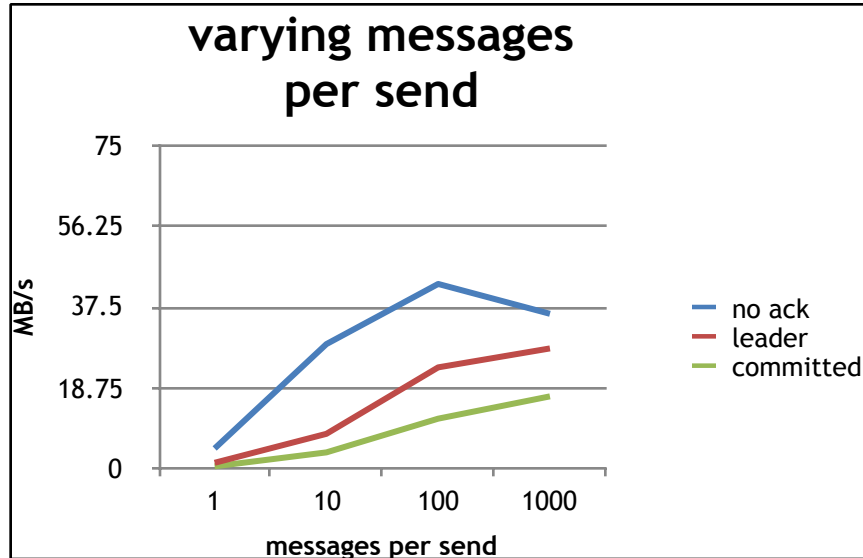
Setup

- 3 brokers
- 1 topic with 1 partition
- Replication factor=3
- Message size = 1KB

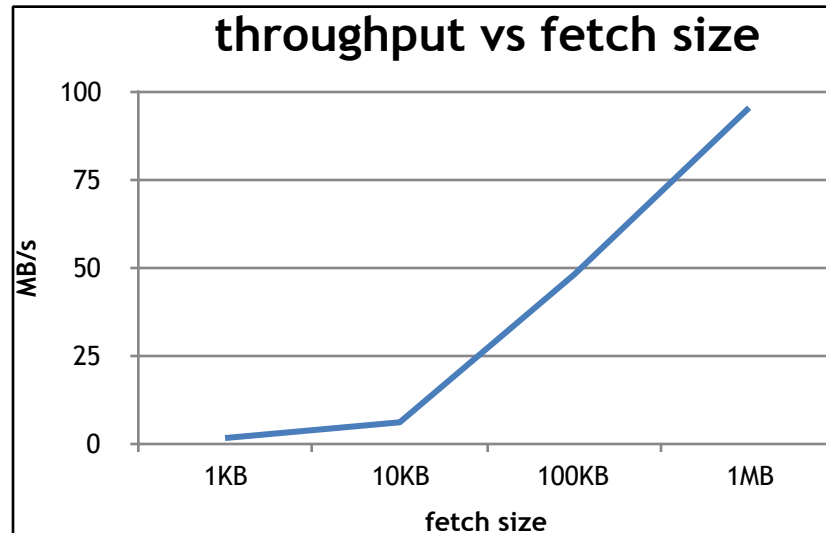
Choosing btw Latency and Durability

When producer receives ack	Time to publish a message (ms)	Durability on failures
no ack	0.29	some data loss
wait for leader	1.05	a few data loss
wait for committed	2.05	no data loss

Producer Throughput



Consumer Throughput



Q/A

- Kafka 0.8.0 (intra-cluster replication)
 - expected to be released in Mar
 - various performance improvements in the future
- Checkout more about Kafka
 - <http://kafka.apache.org/>
- Kafka meetup tonight