Kafka at Scale

Multi-Tier Architectures

Todd Palino

Staff Site Reliability Engineer

LinkedIn, Data Infrastructure Streaming

You may remember me from such talks as…

“Apache Kafka Meetup”

And

“Enterprise Kafka: QoS and Multitenancy”

Who Am I?

- Kafka, Samza, and Zookeeper SRE at LinkedIn
- Site Reliability Engineering
  - Administrators
  - Architects
  - Developers
- Keep the site running, always

What Will We Talk About?

- Tiered Cluster Architecture
- Kafka Mirror Maker
- Performance Tuning
- Data Assurance
- What's Next?

Kafka At LinkedIn

- ~~300+ Kafka brokers~~
- ~~Over 18,000 topics~~
- ~~140,000+ Partitions~~

- ~~220 Billion messages per day~~
- ~~40 Terabytes In~~
- ~~160 Terabytes Out~~

- ~~Peak Load~~
  - ~~3.25 Million messages/sec~~
  - ~~5.5 Gigabits/sec Inbound~~
  - ~~18 Gigabits/sec Outbound~~

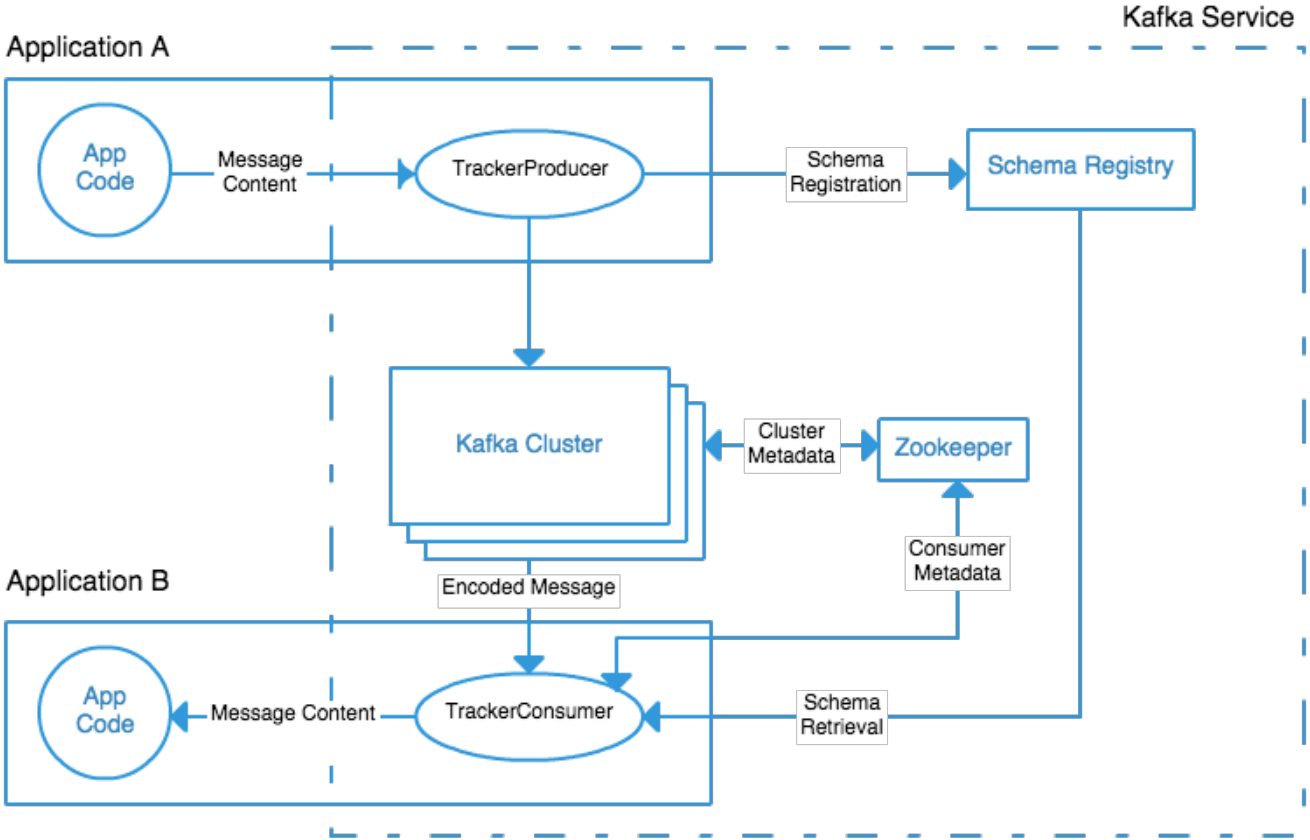
- 1100+ Kafka brokers
- Over 31,000 topics
- 350,000+ Partitions

- 675 Billion messages per day
- 150 Terabytes In
- 580 Terabytes Out

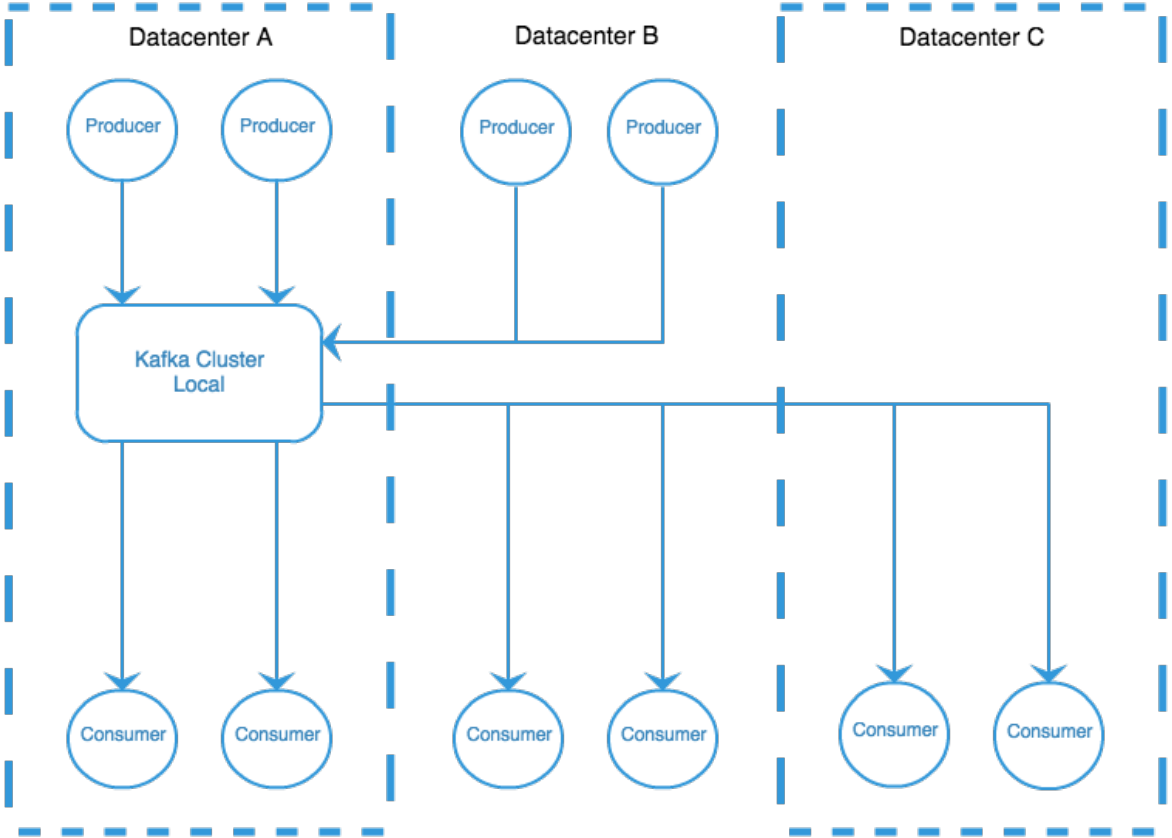
- Peak Load
  - 10.5 Million messages/sec
  - 18.5 Gigabits/sec Inbound
  - 70.5 Gigabits/sec Outbound

Tiered Cluster Architecture

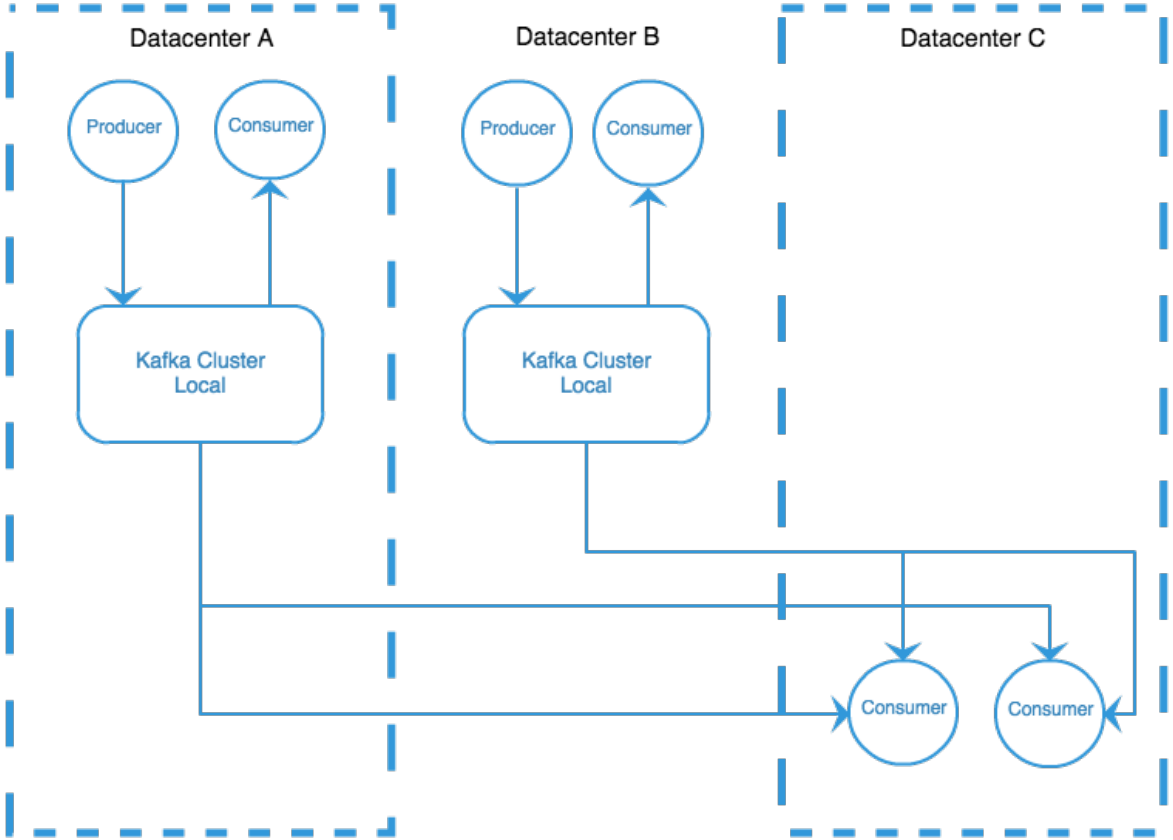
One Kafka Cluster



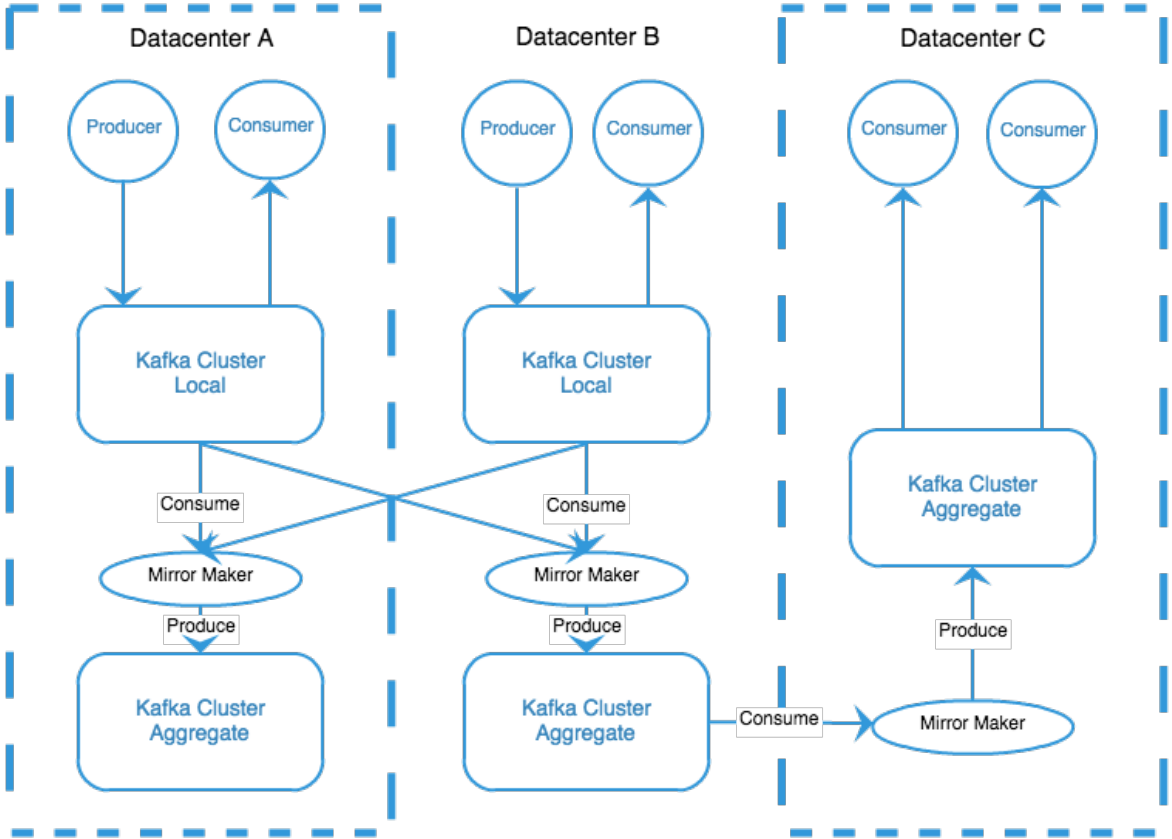
Single Cluster – Remote Clients



Multiple Clusters – Local and Remote Clients



Multiple Clusters – Message Aggregation



Why Not Direct?

- Network Concerns
  - Bandwidth
  - Network partitioning
  - Latency
- Security Concerns
  - Firewalls and ACLs
  - Encrypting data in transit
- Resource Concerns
  - A misbehaving application can swamp production resources

Kafka Mirror Maker

Kafka Mirror Maker

- Consumes from one cluster, produces to another
- No communication from producer back to consumer
- Best practice is to keep the mirror maker local to the target cluster
- Kafka does not prevent loops

Rules of Aggregation

- NEVER produce to aggregate clusters

NEVER produce to aggregate clusters!

Rules of Aggregation

- NEVER produce to aggregate clusters

- Not every topic needs to be aggregated
  - Log compacted topics do not play nice
  - Most queuing topics are local only
- But your whitelist/blacklist configurations must be consistent
  - If you have a topic that is aggregated, make sure to do it from all source clusters to all aggregate clusters
- Carefully consider if you want front-line aggregate clusters
  - It can encourage creating single-master services
  - Sometimes it is necessary, such as for search services

Mirror Maker Concerns

- Adding a site increases the number of mirror maker instances
  - Solution: Multi-consumer mirror makers
- Mirror maker can lose messages like any producer
  - Solution: reduce inflight batches and acks=-1
- Mirror maker has to decompress and recompress every batch
  - Possible solution: flag compressed batches for keyed messages
- Message partitions are not preserved
  - Possible solution: an identity mirror maker

Performance Tuning

Kafka Cluster Sizing

- How big for your local cluster?
  - How much disk space do you have?
  - How much network bandwidth do you have?
  - CPU, memory, disk I/O

- How big for your aggregate cluster?
  - In general, multiple the number of brokers by the number of local clusters
  - May have additional concerns with lots of consumers

Topic Configuration

- Partition Counts for Local
  - Many theories on how to do this correctly, but the answer is “it depends”
  - How many consumers do you have?
  - Do you have specific partition requirements?
  - Keeping partition sizes manageable
- Partition Counts for Aggregate
  - Multiply the number of partitions in a local cluster by the number of local clusters
  - Periodically review partition counts in all clusters
- Message Retention
  - If aggregate is where you really need the messages, only retain it in local for long enough to cover mirror maker problems

Mirror Maker Sizing

- Number of servers and streams
  - Size the number of servers based on the peak bytes per second
  - Co-locate mirror makers
  - Run more mirror makers in an instance than you need
  - Use multiple consumer and producer streams

- Other tunables to look at
  - Partition assignment strategy
  - In flight requests per connection
  - Linger time

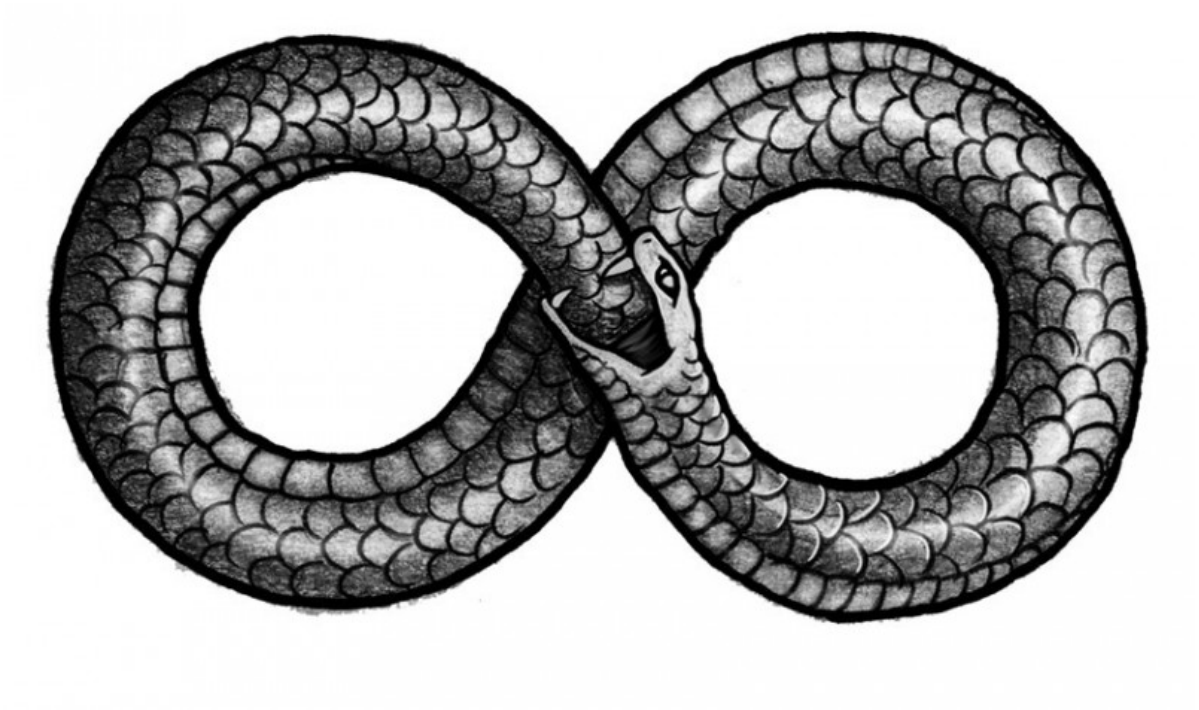
Segregation of Topics

- Not all topics are created equal
- High Priority Topics
  - Topics that change search results
  - Topics used for hourly or daily reporting
- Run a separate mirror maker for these topics
  - One bloated topic won't affect reporting
  - Restarting the mirror maker takes less time
  - Less time to catch up when you fall behind

Data Assurance

Monitoring

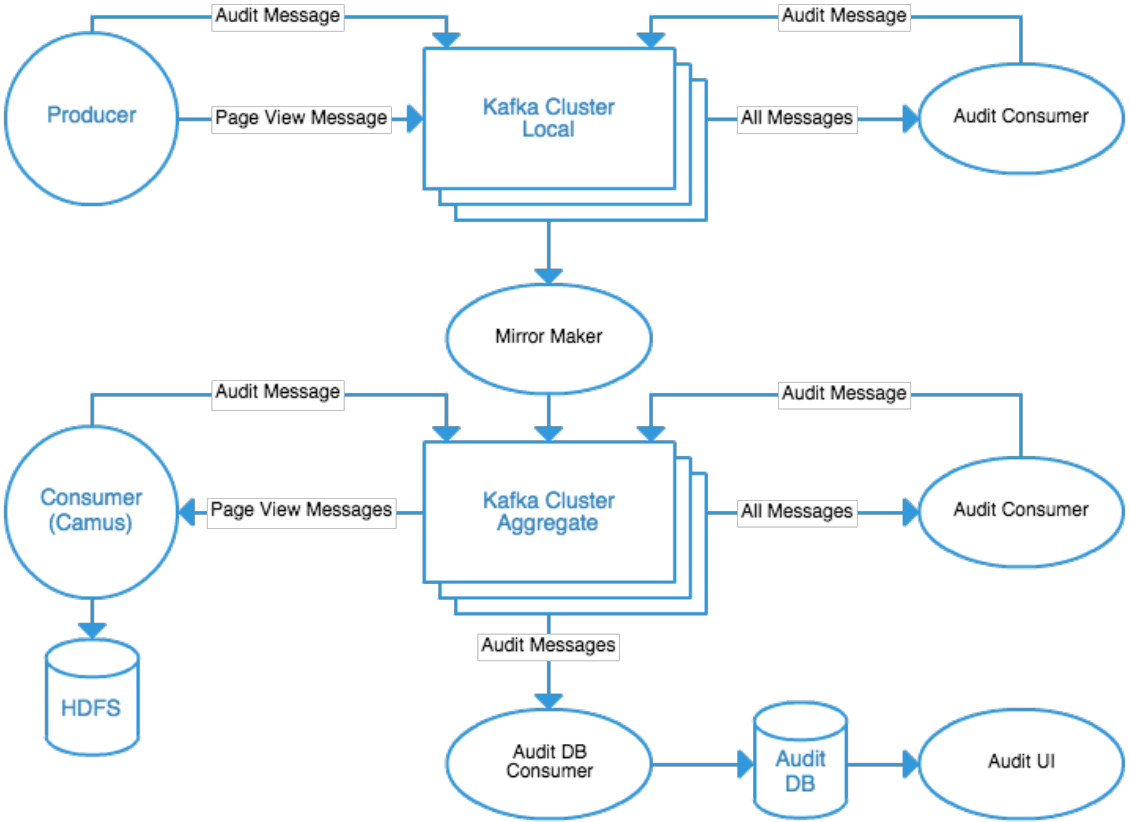
- Kafka is great for monitoring your applications



Monitoring

- Have a system for monitoring Kafka components that does not use Kafka
  - At least for critical metrics
- For tiered architectures
  - Simple health check on mirror maker instances
  - Mirror maker consumer lag
- Is the data intact?

Auditing Message Flows



Audit Content

- Message audit header
  - Timestamp
  - Service and hostname
- Audit messages
  - Start and end timestamps
  - Topic and tier
  - Count

Audit Concerns

- We are only counting messages
  - Duplication of messages can hide losses
  - Using the detailed service and host audit criteria, we can get around this
- We can’t audit all consumers
  - The relational DB has issues keeping up with bootstrapping clients
  - This can be improved with changes to the database backend
- We cannot handle complex message flows
  - The total number of messages has to appear in each tier that the topic is in
  - Multiple source clusters must have the same tier name

Conclusion

Work Needed in Kafka

- Access controls
- Encryption
- Quotas
- Decompression improvements in mirror maker

Getting Involved With Kafka

- http://kafka.apache.org
- Join the mailing lists
  - users@kafka.apache.org
  - dev@kafka.apache.org
- irc.freenode.net - #apache-kafka
- Meetups
  - Apache Kafka - http://www.meetup.com/http-kafka-apache-org
  - Bay Area Samza - http://www.meetup.com/Bay-Area-Samza-Meetup/
- Contribute code