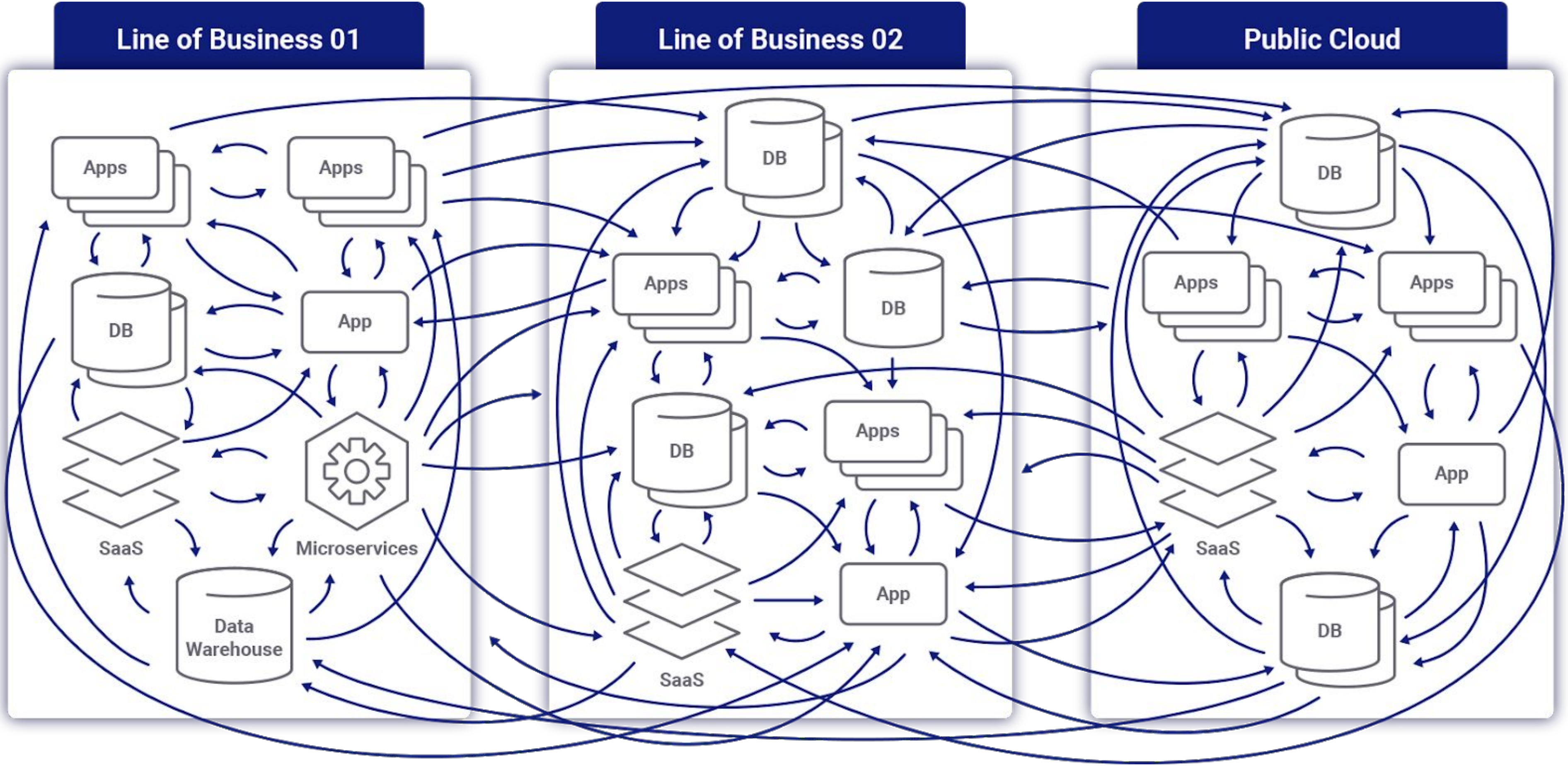
Kafka Performances

Brice Leporini brice@confluent.io
Florent Ramière florent@confluent.io


Line of Business 01
Line of Business 02
Public Cloud
Apps
Apps
DB
App
SaaS
Microservices
Data Warehouse
DB
Apps
DB
DB
Apps
SaaS
App
DB
Apps
Apps
SaaS
App
DB

confluent

Founded by the creators of **Apache Kafka**

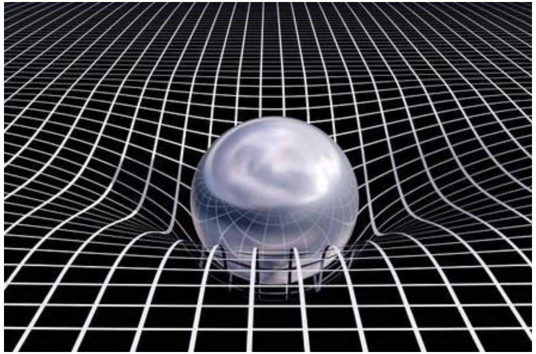
Technology Developed while at **LinkedIn**

Largest Contributor and tester of Apache Kafka

- Founded in 2014
- Raised $205M from Benchmark, Index, Sequoia
- +950 Employees
- Offices in 20 countries
- Hundreds of enterprise subscription customers

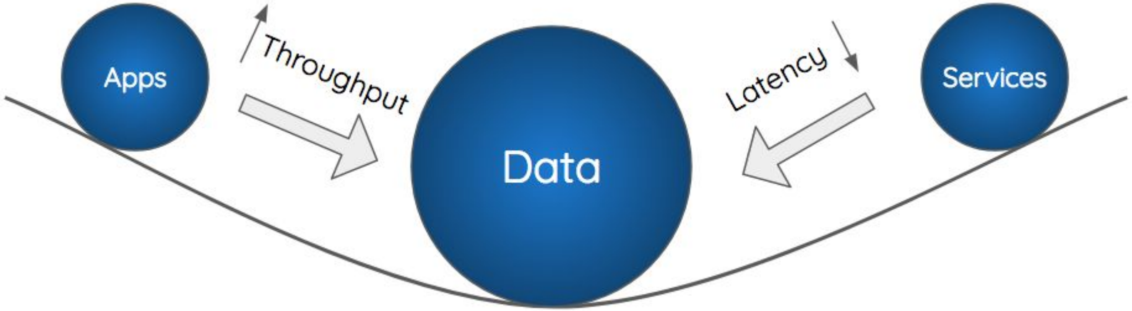
BENCHMARK Index Ventures SEQUOIA

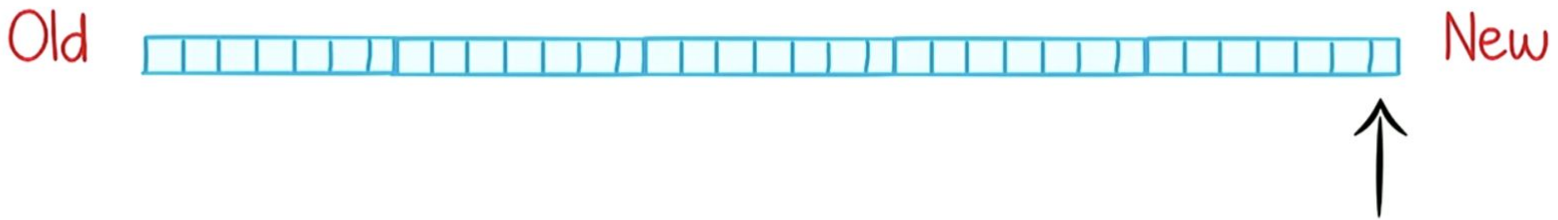
Silos explained by Data Gravity concept



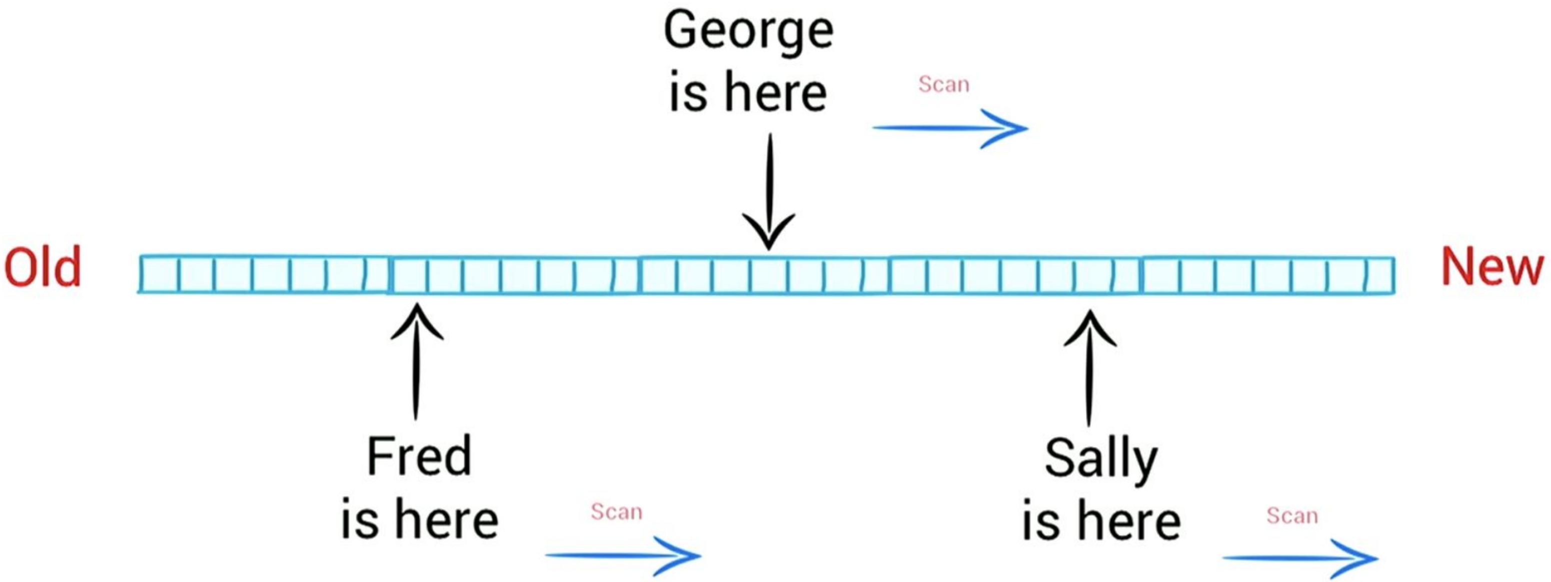
As data accumulates (builds mass) there is a greater likelihood that additional services and applications will be attracted to this data.

This is the same effect gravity has on objects around a planet. As the mass or density increases, so does the strength of gravitational pull.





Messages are added at the end of the log


George
is here
Scan
Old
New
Fred
is here
Scan
Sally
is here
Scan

Mobile

Cloud

Microservices

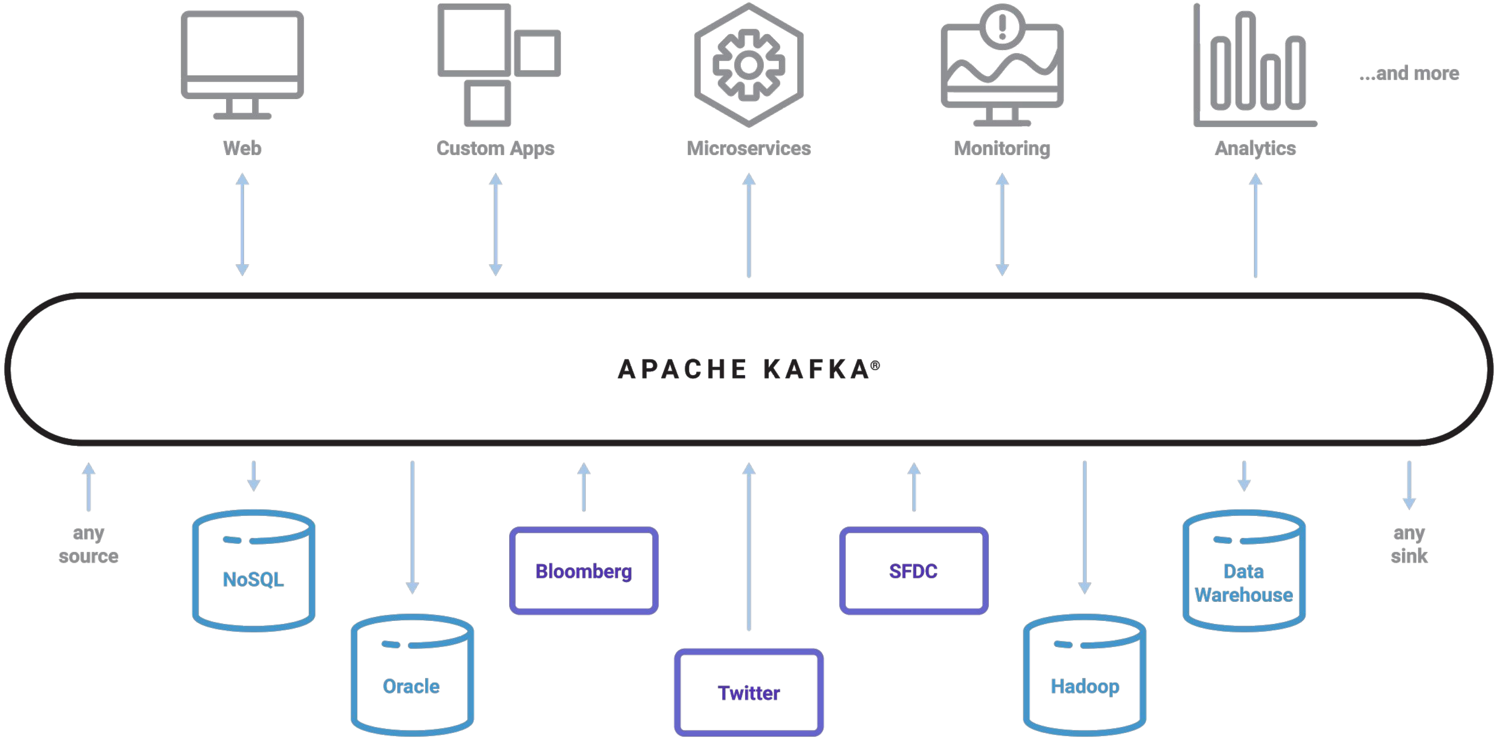
Internet of Things

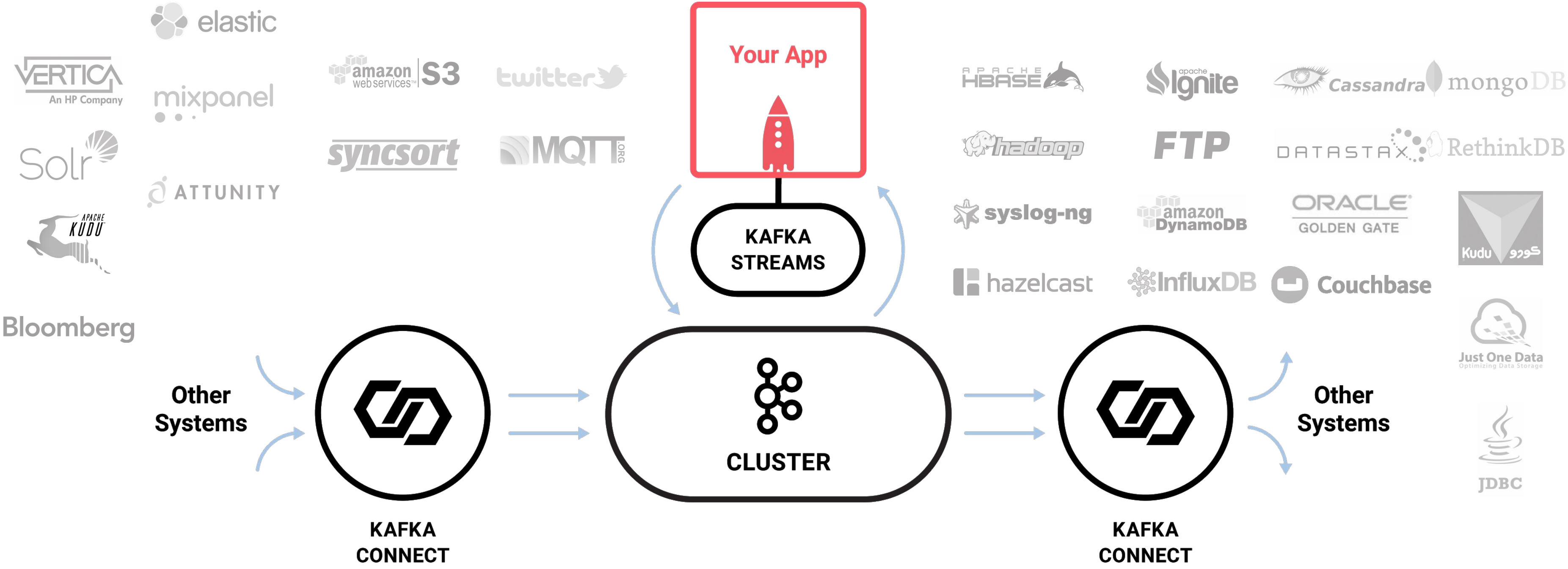
Machine Learning

Massive volumes of new data generated every day

Distributed across apps, devices, datacenters, clouds

Structured, unstructured polymorphic


Web
Custom Apps
Microservices
Monitoring
Analytics
...and more
APACHE KAFKA®
any source
NoSQL
Oracle
Bloomberg
Twitter
SFDC
Hadoop
Data Warehouse
any sink



elastic
VERTICA An HP Company
mixpanel
amazon web services S3
twitter
Solr
syncsort
MQTT.ORG
ATTUNITY
APACHE KUDU
Bloomberg
Your App
KAFKA STREAMS
Other Systems
KAFKA CONNECT
CLUSTER
KAFKA CONNECT
Other Systems
APACHE HBASE
apache Ignite
Cassandra
mongoDB
hadoop
FTP
DATASTAX
RethinkDB
syslog-ng
amazon DynamoDB
ORACLE GOLDEN GATE
Kudu
hazelcast
InfluxDB
Couchbase
Just One Data Optimizing Data Storage
JDBC

Publish & Subscribe

Store & ETL

Process

- Scalability
- Retention
- Durability
- Replication
- Security
- Resiliency
- Throughput
- Ordering
- Exactly Once Semantic
- Transaction
- Idempotency
- Immutability
- …

NETFLIX

criteo

lyft

LinkedIn

And many more...

What means Kafka performance?

Depends on **your** context!

Do you need throughput?

Do you need to optimize storage / bandwidth?

Do you need low latency ?

What is your SLA ?

No SLA ?

Stop ! And gather them right away!

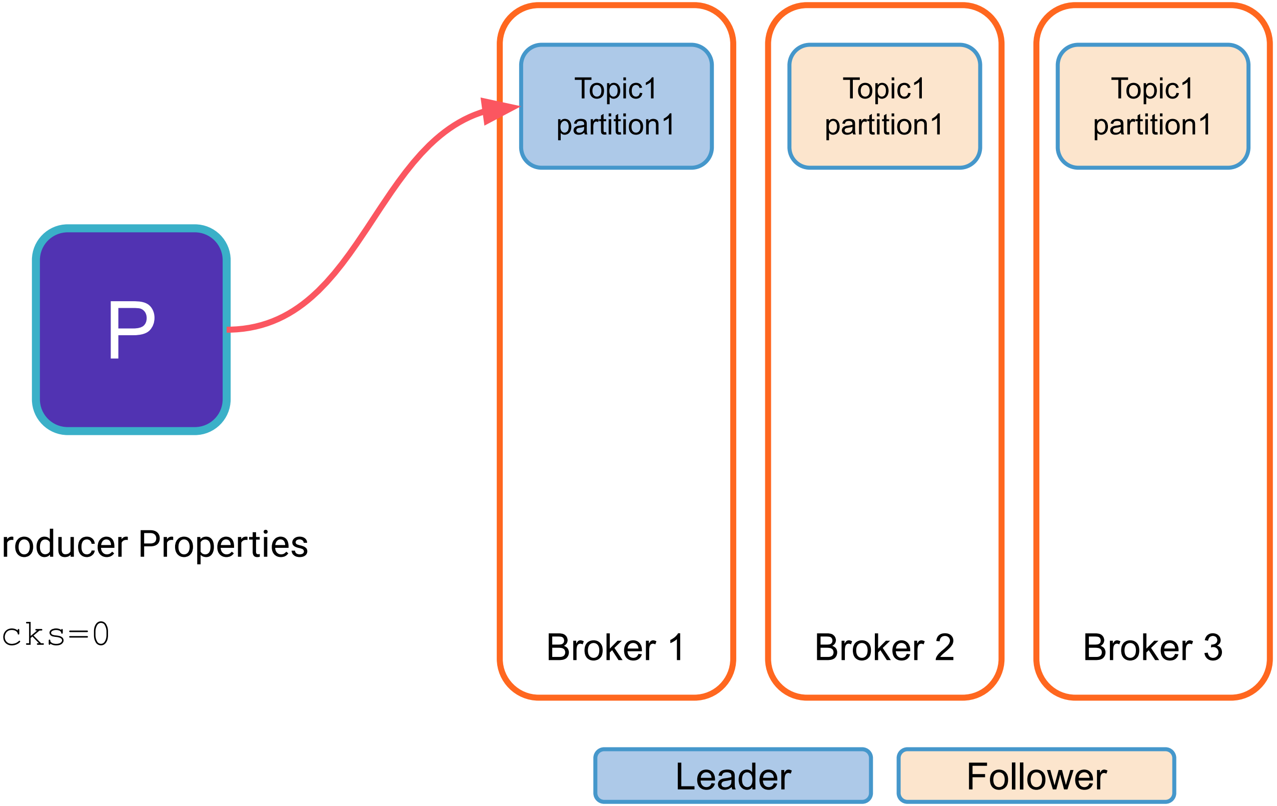
But first ...

Kafka in bash !

https://github.com/framiere/kafka-in-bash

On the producer side

Producer Guarantees



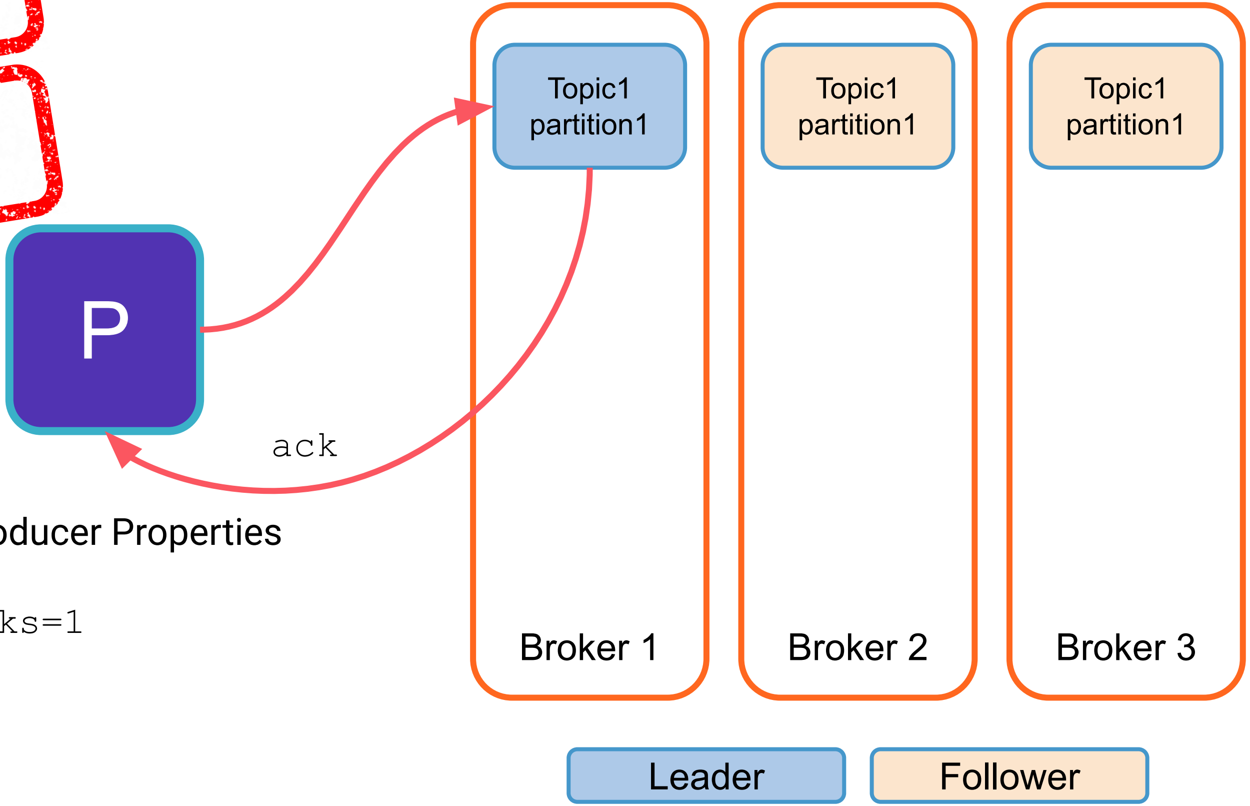
Producer Properties

```
acks=0
```

Producer Guarantees

THROUGHPUT

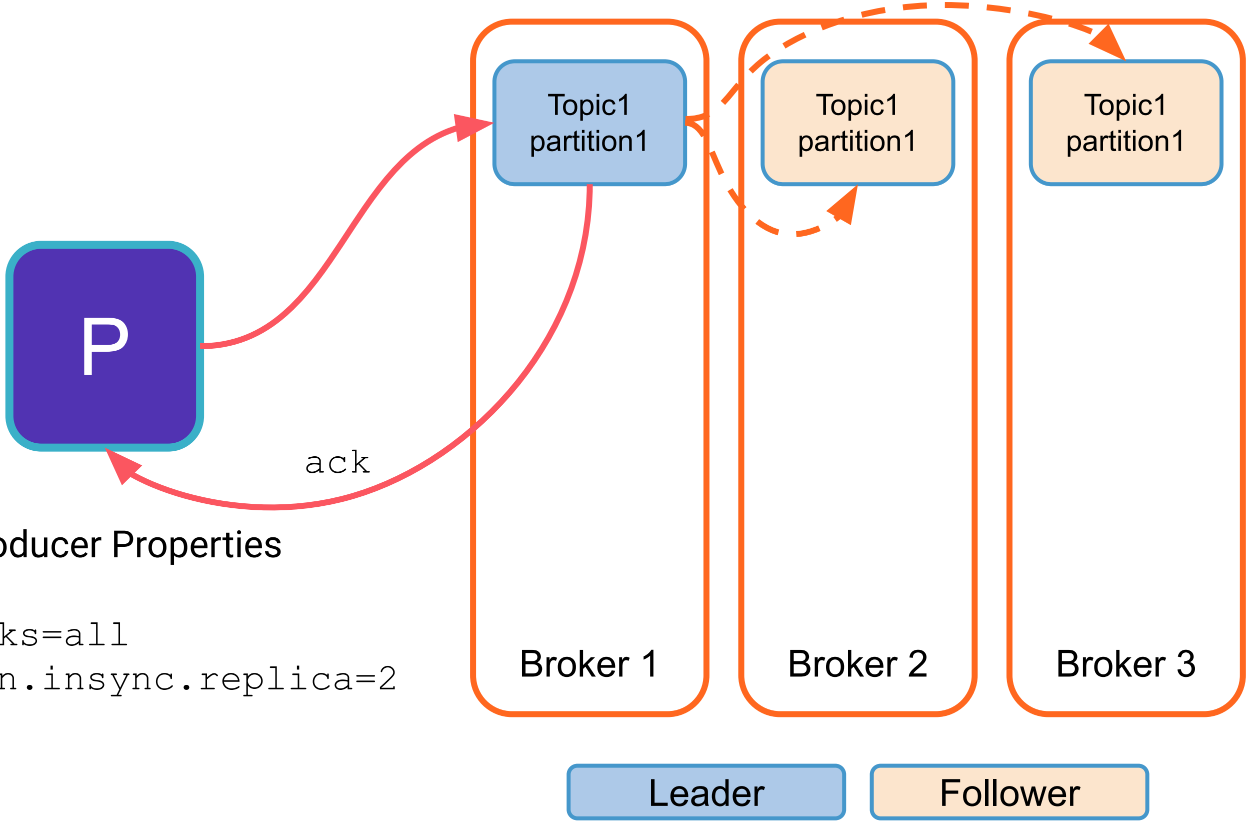
LATENCY



Producer Properties

```
acks=1
```

Producer Guarantees



Producer Properties

```
acks=all
min.insync.replica=2
```

batch.size

Default: 16KB

The producer will attempt to batch records together into fewer requests whenever multiple records are being sent to the same partition.

THROUGHPUT

but... (yes there is a but)

linger.ms

Default: **0 ms**

LATENCY

This parameter may overrule the batch size, especially with the default value...

max.in.flight.requests .per.connection

Default: 5

The maximum number of unacknowledged requests the client will send on a single connection before blocking.

THROUGHPUT

but… (yes there is a but)

max.in.flight.requests .per.connection

Default: 5

Sending parallel requests to the broker avoids ordering guarantee.

but... (yes there is a but)

enable.idempotence

Default: **false**

retries

Default: **2147483647**

Enabling idempotence avoids any duplicate message and restore the ordering guarantee

compression.type

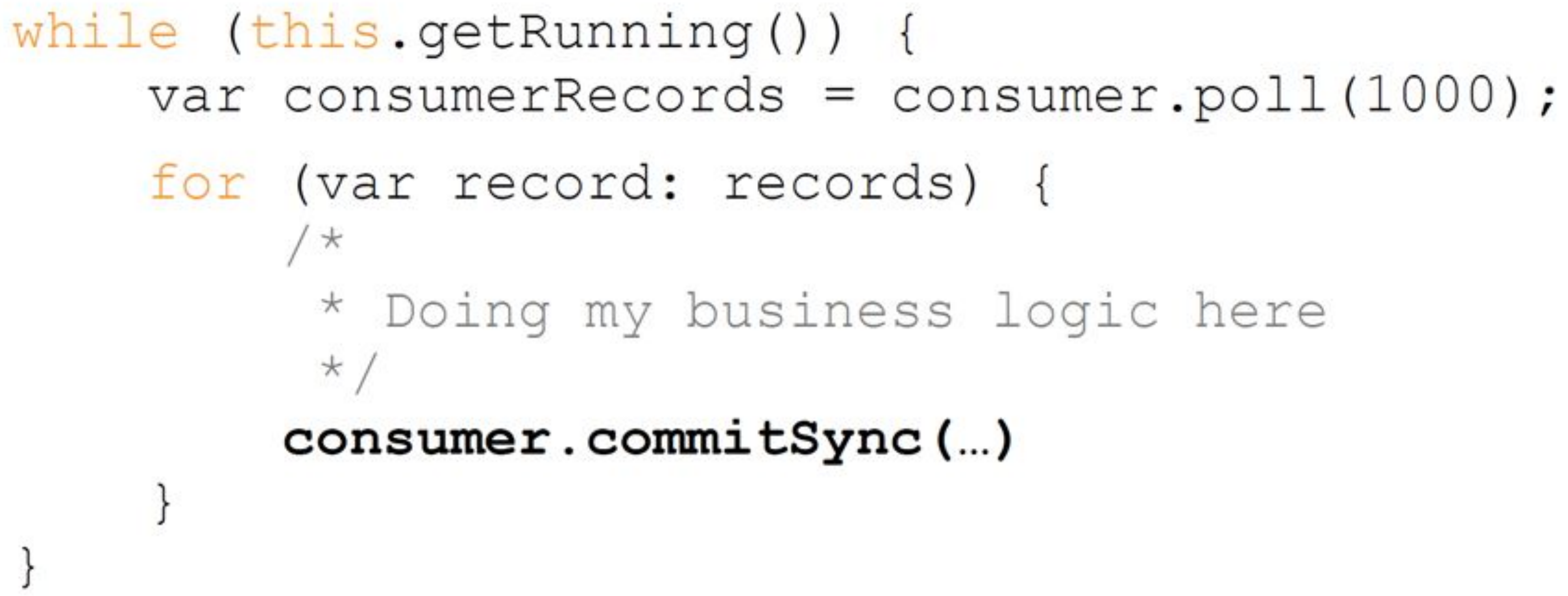
Default value: **none**

STORAGE / IO

Supported compression algorithms are gzip, snappy, lz4, or zstd.

“Compression is of full batches of data, so the efficacy of batching will also impact the compression ratio“

On the consumer side



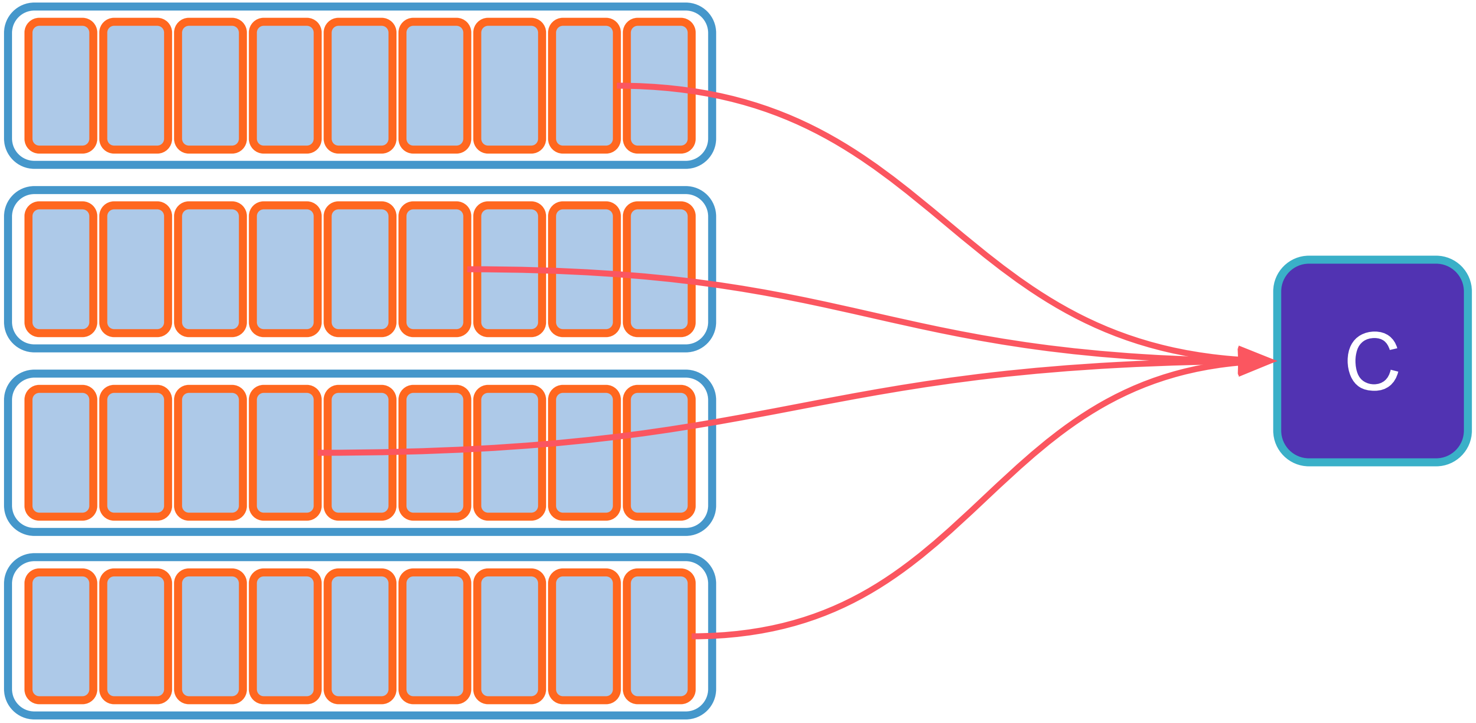
```
while (this.getRunning()) {
    var consumerRecords = consumer.poll(1000);
    for (var record: records) {
        /*
         * Doing my business logic here
         */
        consumer.commitSync(...)
    }
}
```

commit

Manually committing aggressively…

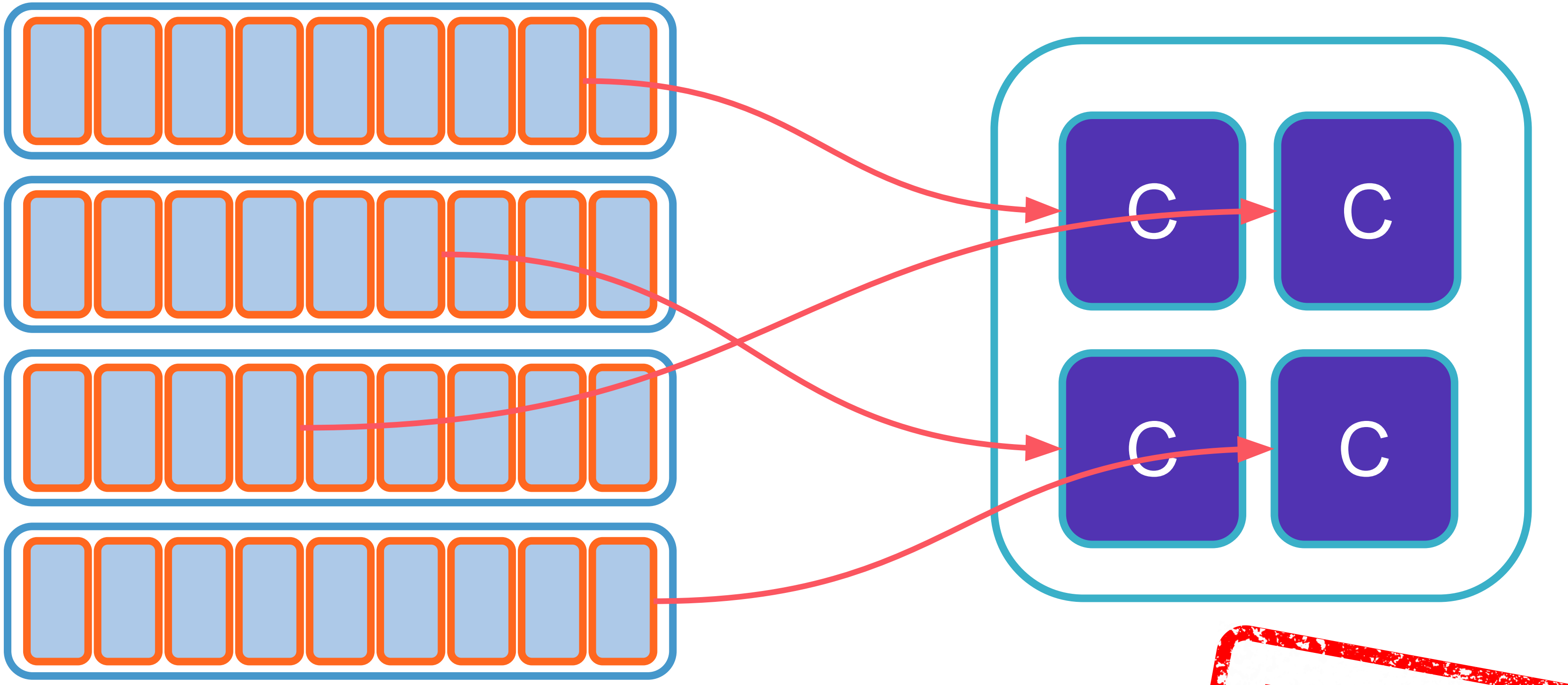
Add a huge workload on Apache Kafka

Consuming From Kafka - Single Consumer



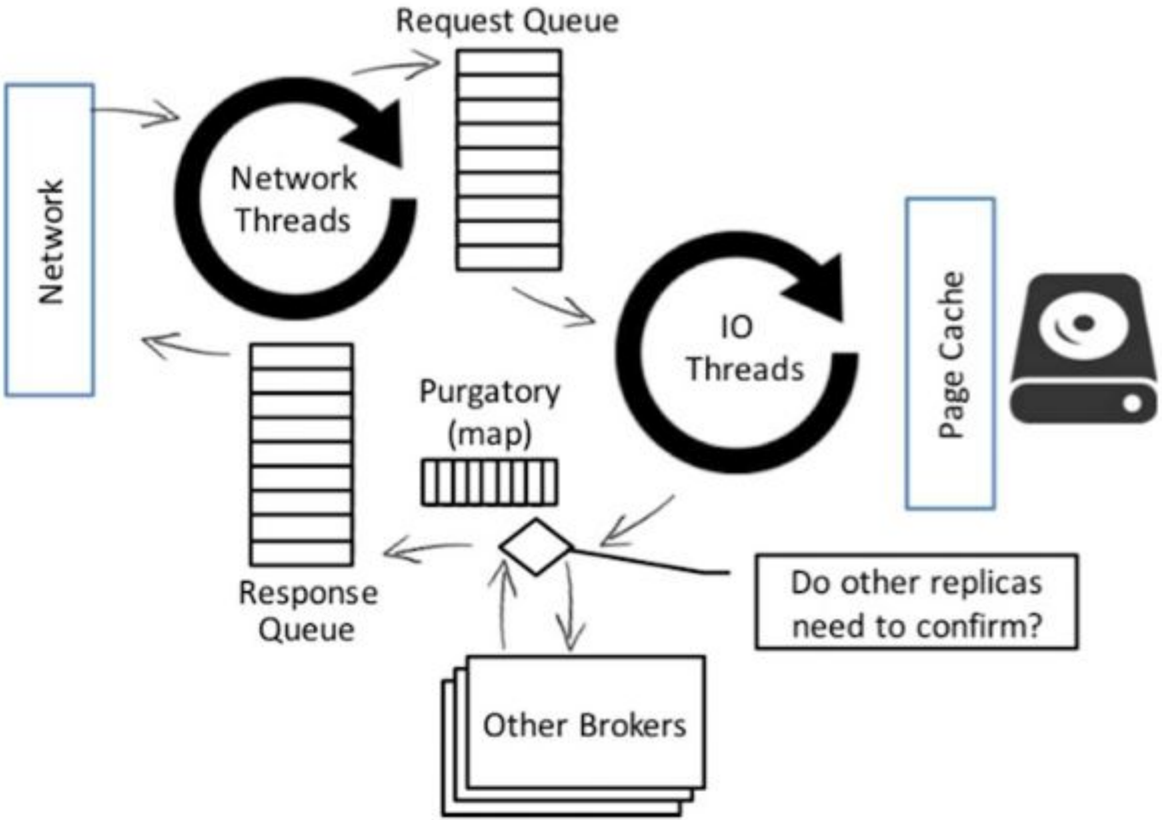
confluent

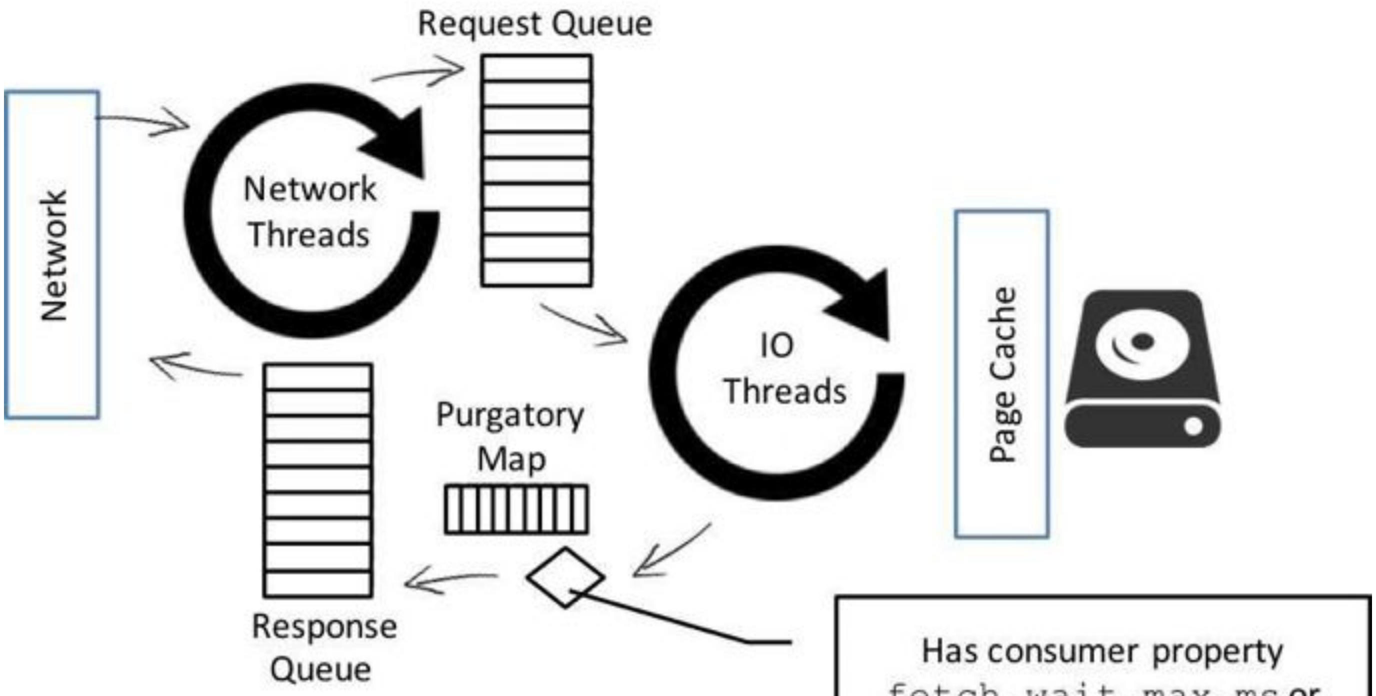
Consuming From Kafka - Grouped Consumers



THROUGHPUT

On the broker side


Request Queue
Network
Network Threads
IO Threads
Page Cache
Purgatory (map)
Response Queue
Do other replicas need to confirm?
Other Brokers



Has consumer property `fetch.wait.max.ms` or `fetch.min.bytes` been exceeded?

Never forget that

“Premature optimization is the root of all evil (or at least most of it) in programming.”

Donald E. Knuth

Monitoring producers

Detect application code performance smells

Monitor Producer time spent in user processes

`io-ratio`: fraction of time the I/O thread spent doing I/O

`io-wait-ratio`: fraction of time the I/O thread spent waiting

User processing time =

```
    1 - io-ratio - io-wait-ratio
```

If user processing time is high, the single Producer I/O thread may be busy.

Check Producer code, including the callback which is invoked when messages have been acknowledged.

Check your tuning efficiency

`batch-size-avg`

`compression-rate-avg`

Small `batch-size-avg` : `linger.ms` too low?

Poor compression rate: `batch.size` too small?
`linger.ms` too low?

Check broker response time

`request-latency-avg`

A high request latency on the producer side may be the first clue of a deeper investigation!

Check the client efficiency

`buffer-available-bytes`

`buffer-total-bytes`

`waiting-threads`

A high buffer usage may be a clue that your code is producing at a very high pace. Consider the following options:

- Running more application instances, beware of ordering concerns
- Digging deeper: network → broker → disks
- `acks=1` if compliant with the business

Monitoring consumers

Infer consumer performance

`records-lag-max`

`records-lag`

The maximum lag in terms of number of records for any partition in this window.

An increasing value over time is your best indication that the consumer group is not keeping up with the producers.

Monitoring brokers

Monitoring leaders and partitions

`LeaderCount`

`PartitionCount`

These gauges are critical to monitor performance.

Give clues about how balanced is the load across brokers and may explain if one broker seems to face more load than another.

I/O and Network

`RequestHandlerAvgIdlePercent`

I/O metric

`NetworkProcessorAvgIdlePercent`

Network metric

Values are between 0 and 1:

- 0 indicates all resources are used
- 1 indicates all resources are available

Values below 0.4 should trig alerts and raise questions:

- Over usage?
- Infrastructure issue?
- Capacity planning to update?
- Thread pool sizing?

Monitoring Requests on the broker

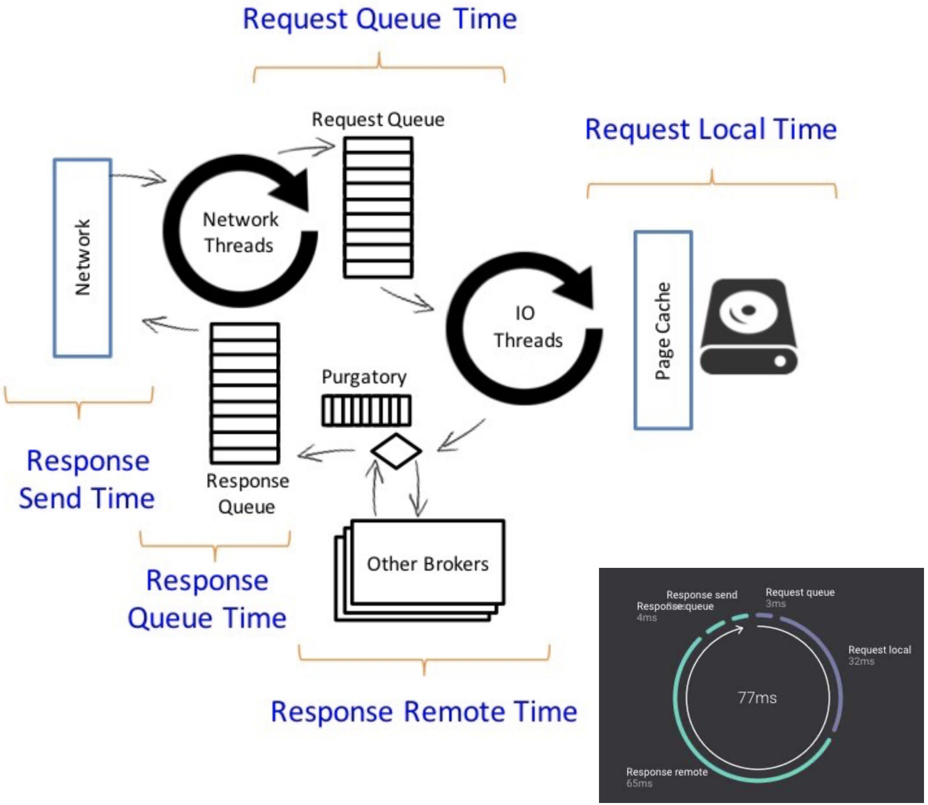
`RequestQueueTimeMs`: Time the request waits in the request queue

`LocalTimeMs`: Time the request is processed at the leader

`RemoteTimeMs`: Time the request waits for the follower. This is non-zero for produce requests when acks=all

`ResponseQueueTimeMs`: Time the request waits in the response queue

`ResponseSendTimeMs`: Time to send the response



Monitoring the broker host OS

Disk usage > 60%

CPU usage > 60%

Network IO usage > 60%

File handle usage > 60%

⇒ Warning!

Hosting

Zookeeper

Critical component but not facing a huge traffic nor holding a lot of data.

Sensitive to latency

⇒ Fast drive (SSD)

⇒ Single digit network latency between nodes

In VMs, provision dedicated resources.

Broker

Brokers are designed to take benefit of the Kernel Page Cache

⇒ Keep half of the RAM free to allow Kernel to leverage it

Avoid sharing disks with other applications or brokers.

User several disks to maximize throughput:

- Through a RAID controller, except RAID5/6
- With multiple mount points

Avoid hosting brokers and Zookeeper on the same host as their running profiles are completely different.

Under a profiler

https://github.com/framiere/a-kafka-performance-story

Optimizing Your Kafka Deployment

Throughput, Latency, Durability, and Availability

https://www.confluent.io/white-paper/optimizing-your-apache-kafka-deployment/

THANK YOU

@framiere - florent@confluent.io
@blep - brice@confluent.io

cnfl.io/blog

confluent