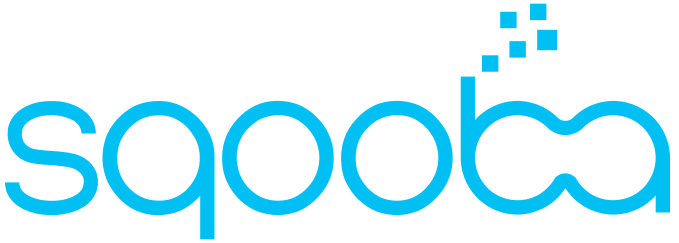


Data Locality in Kafka

„Kafka Meetup“ Bern, 30.01.2020

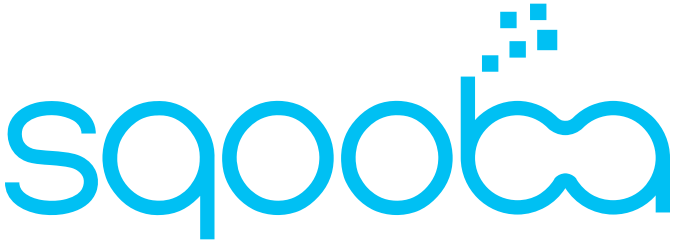
Agenda

- Data locality
- Partition (re)assignment
- Consumer group
- KIP 392
- Wrap up

Who am I

Benoit Perroud, CTO and Founder of Sqooba AG

- Data Platform Infrastructure Plumber (multi PB Hadoop (since 2010), multi TB Kafka (since 2013), …)
- Apache Committer and open source Contributor (mostly Cassandra, Hadoop, Kafka, …)
- Mostly on operation' side of the Force



Data Locality

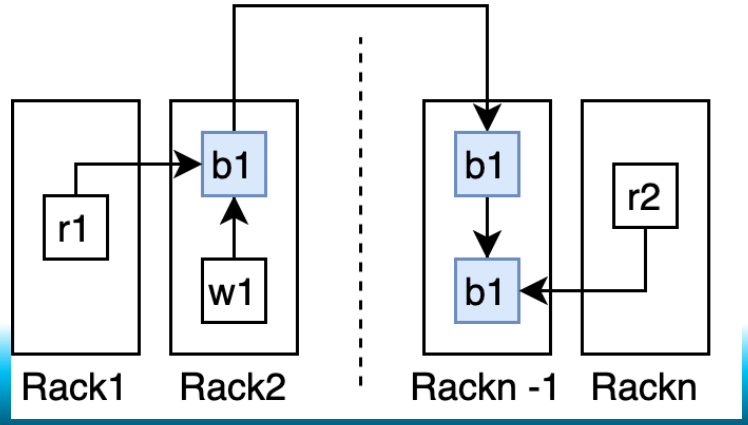
Data Locality

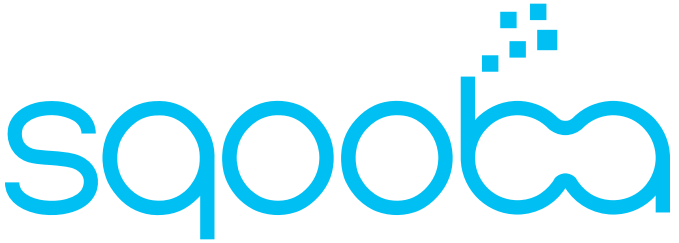
- Know where your data is stored
- Understand the underlying physical infrastructure (and its bottlenecks)
- Take advantage of the distance to the data

- Data affinity: read/write data close to the current location to ensure lower latency and/or higher throughput
- Data anti-affinity: read/write data far from the current location to ensure higher availability

Hadoop has data locality at root

- Hadoop value proposition is data locality: send the processing to the data.
- Every nodes have location information such as dc, az / rack, host
- Data Locality is usually measured as a distance between the data producer or consumer in term of host, ack / az, dc.
- Reads are directed to the closest host, rack / az, dc from the caller, in order of priorities, to ensure performance (and avoid network bottleneck)
- Writes are replicated to another dc, az / rack, host (i.e. to another failure domain, to ensure high availability of the data)





Kafka

Kafka is all about remote

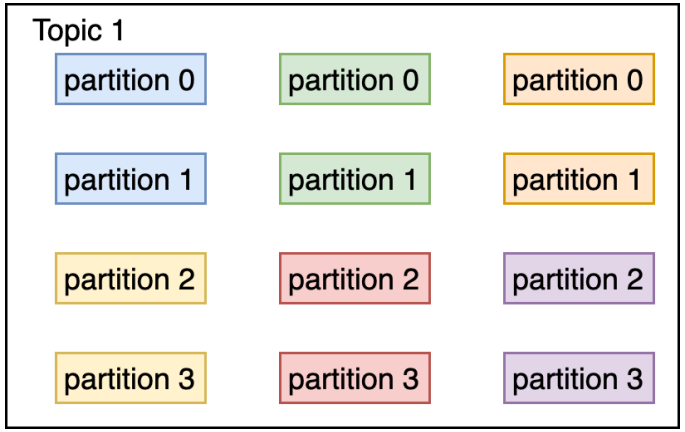
- Kafka changed the game to data locality: remote write and remote reads.

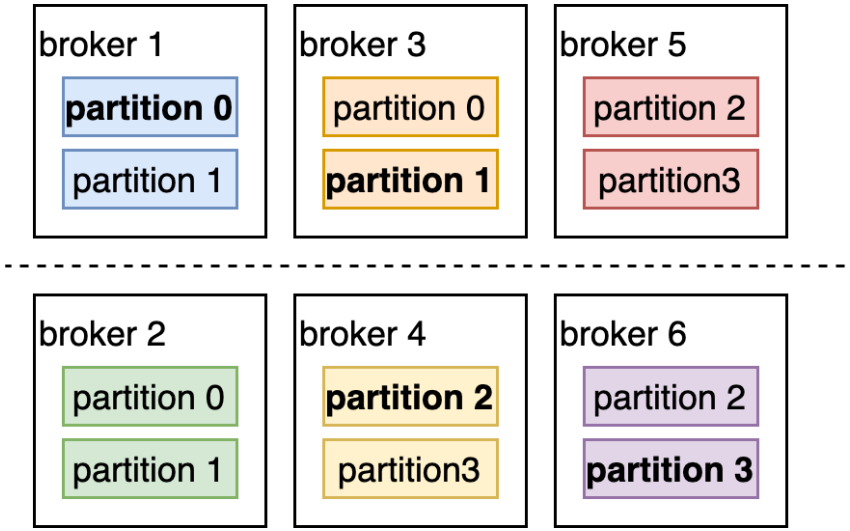
  Kafka is ~10 years younger than Hadoop, and network is no longer the clear bottleneck.

So why do we still care about Data Locality?

Kafka Model

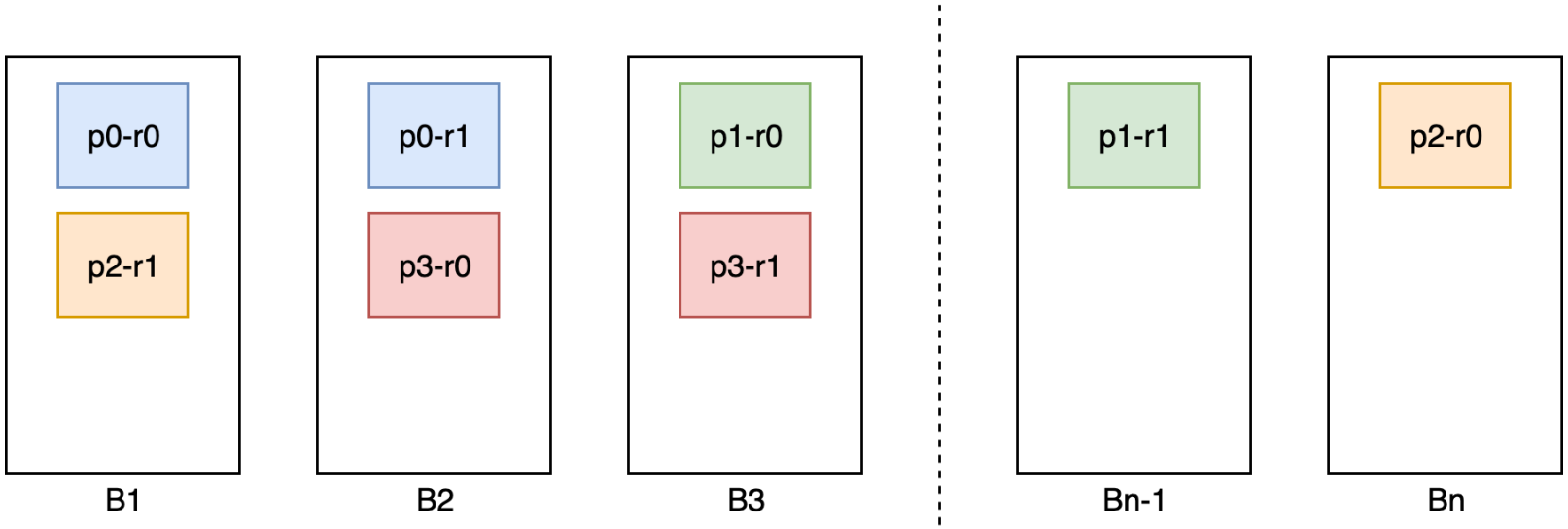
- Kafka defines topics, which are composed of 1 or more partitions
- Partitions have 1 or more replicas
- Partitions' replica are assigned to a broker and physically stored on disks.
- Every partition elect a leader amongst the replica, from which reads and writes happen.





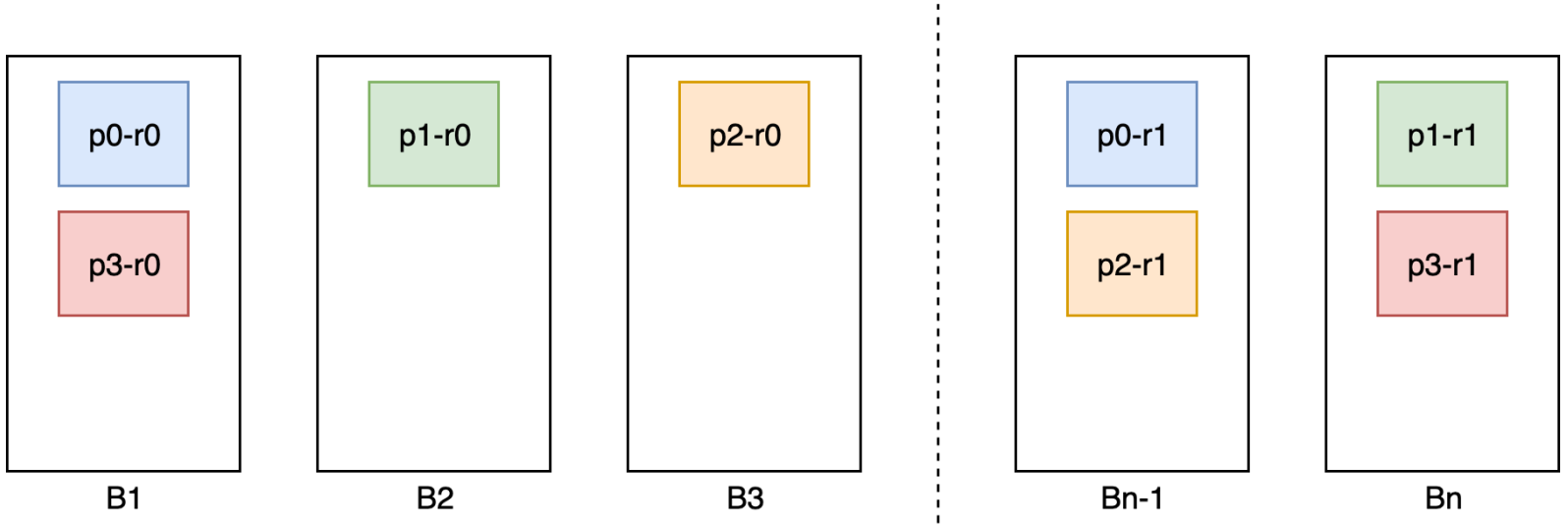
Replica assignment

- When creating a topic, if not told otherwise, Kafka brokers decide where goes which replica for each partitions.
- As an example, if you have a topic with 4 partition and a replication factor of 2:



KIP 36

- KIP-36, Rack aware replica assignment, can enforce replicas to be placed in different racks (anti-affinity)
- broker.rack must be set



Kafka Assignment Optimizer

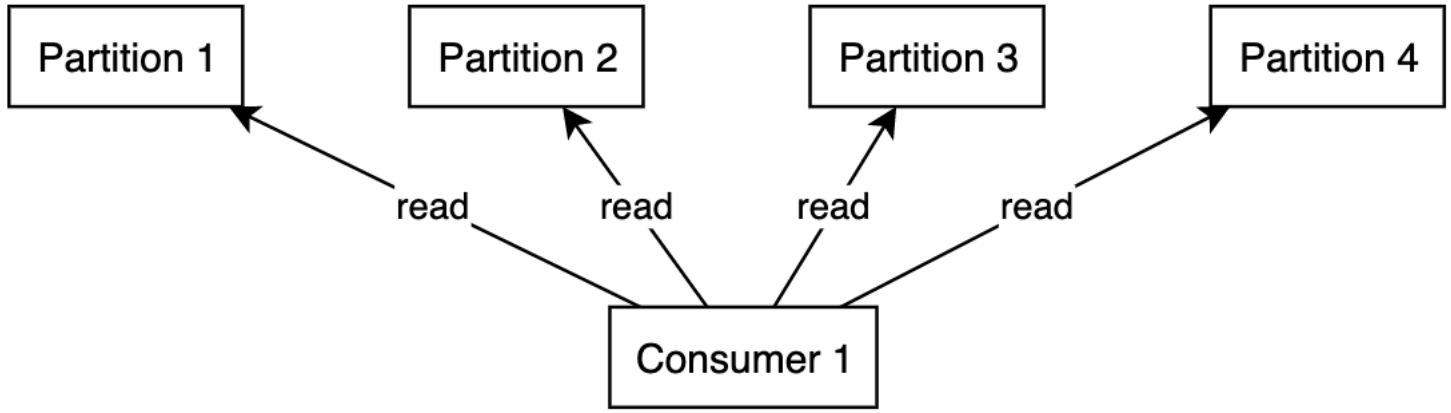
- On a long running kafka cluster, when adding/removing a node in the cluster, we'll need to reassign partitions.
- There is a famous tool, `kafka-reassign-partitions.sh`
- But it does not minimize partition movement to reach a balanced cluster state (i.e. it is rather rough)
- At Sqooba, we're using linear programming to ensure
  - rack and host distributed/balanced partitions
  - rack and host distributed/balanced leaders
  - minimal partition movements
- → https://kafka-optimizer.sqooba.io/

```
// Optimization function, based on current assignment
max: 1 t1b12p5 + 4 t1b19p6_l + 2 t1b14p8 + 2 t1b11p2_l + 1 t1b21p2 ...

// Constrain on replication factor for every partition
t1b1p0 + ... + t1b19p0_l + ... + t1b32p0_l = 2;
t1b1p1 + ... + t1b19p1_l + ... + t1b32p1_l = 2;
...

// Constraint on having one and only one leader per partition
t1b1p0_l + ... + t1b19p0_l + ... + t1b32p0_l = 1;
t1b1p1_l + ... + t1b19p1_l + ... + t1b32p1_l = 1;
...

// Constraint on min/max replicas per broker
t1b1p0 + t1b1p0_l + ... + t1b1p9 + t1b1p9_l <= 2;
t1b1p0 + t1b1p0_l + ... + t1b1p9 + t1b1p9_l >= 1;
...

// Constraint on min/max leaders per broker
t1b1p0_l + t1b1p1_l + ... + t1b1p8_l + t1b1p9_l <= 1;
t1b2p0_l + t1b2p1_l + ... + t1b2p8_l + t1b2p9_l >= 0;
...
```

Consumer Group

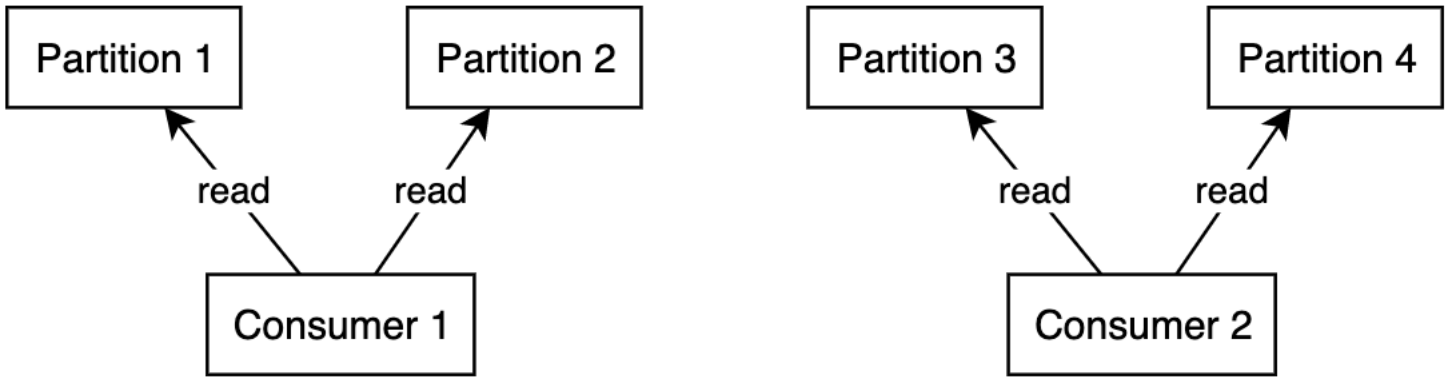
Consumer Group

- A consumer group is composed of 1 or more processes, participating (coordinating) in consuming (hopefully all) the messages of one or more topics.
- Assume we have a topic with 4 partitions and one consumer. The consumer is fetching messages from the 4 different partitions:



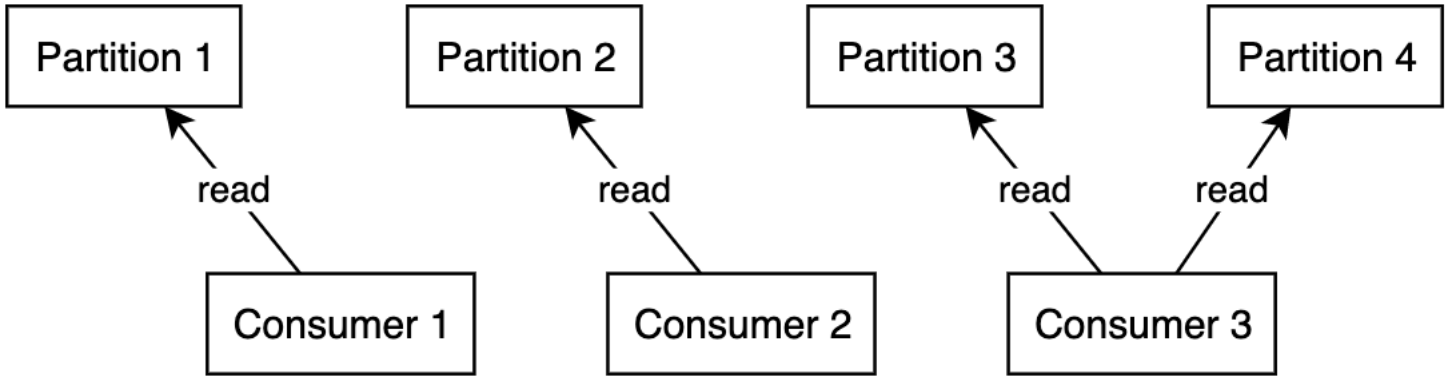
Consumer Group

- When another consumer joins the group, the assignment is modified so that ideally both consumers are fetching messages from 2 partitions:



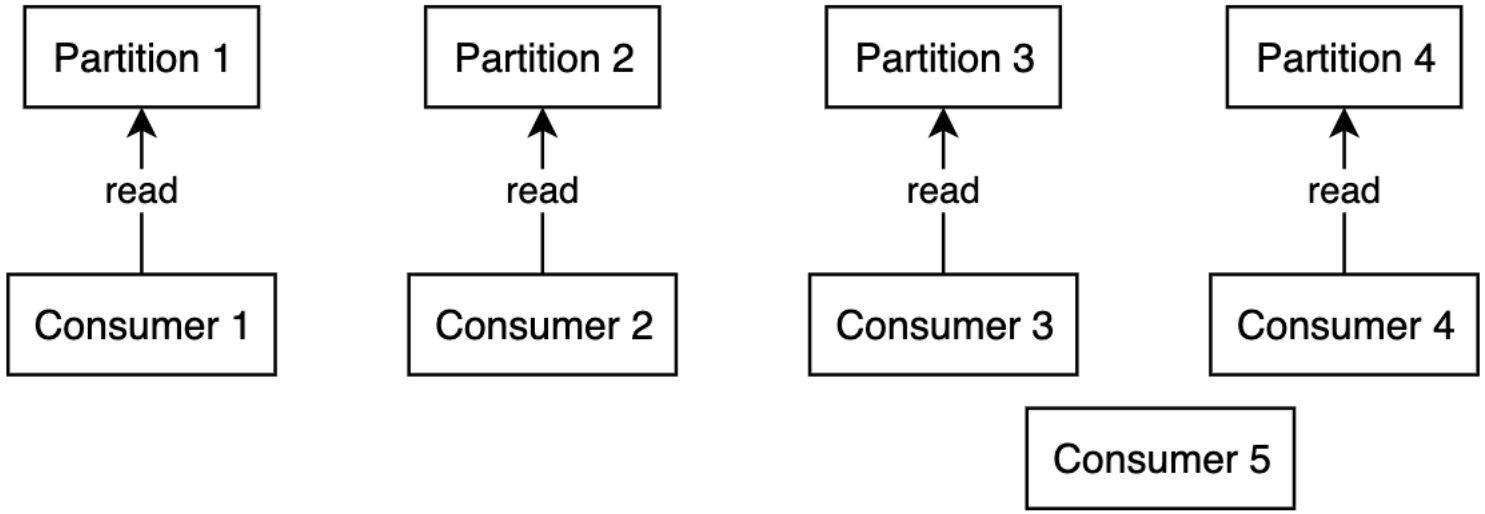
Consumer Group

- Let’s add a third consumer. The group is again updating its state to share the work



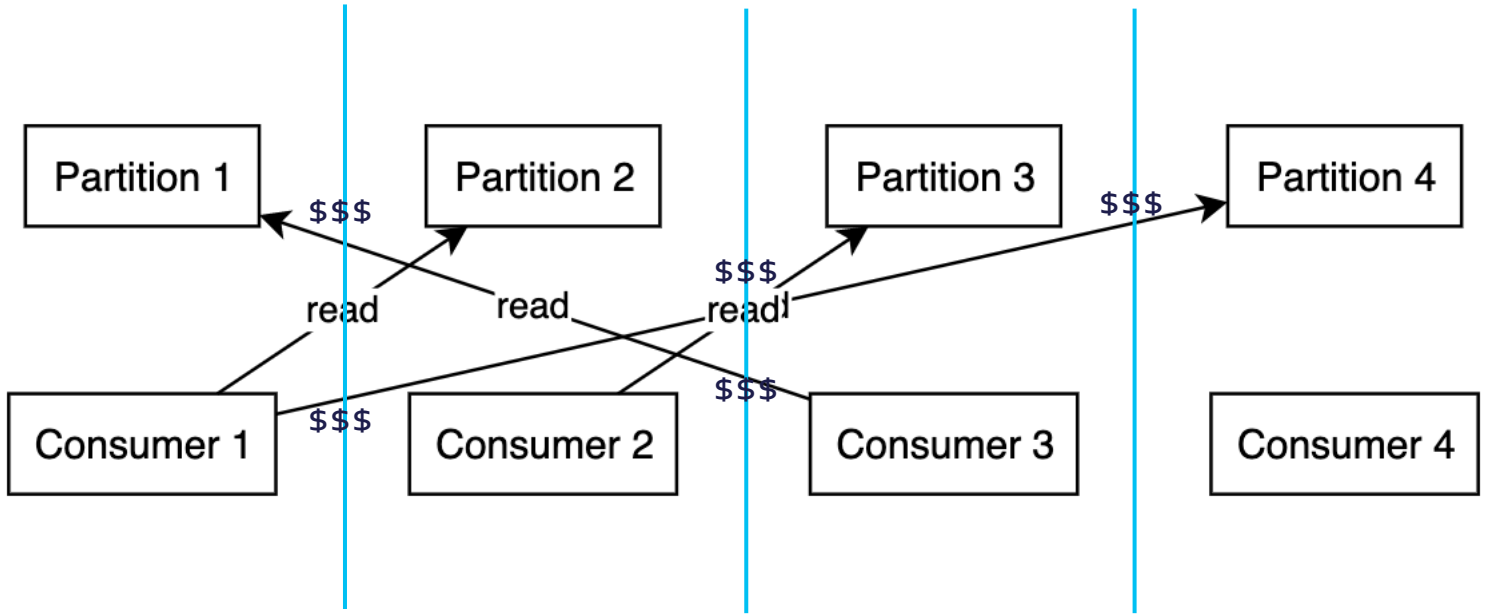
Consumer Group

- **Now in Kafka, one partition can only be read by one consumer within a group at the time. If we have 5 consumers for 4 partitions, one will be idle (standby consumer).**

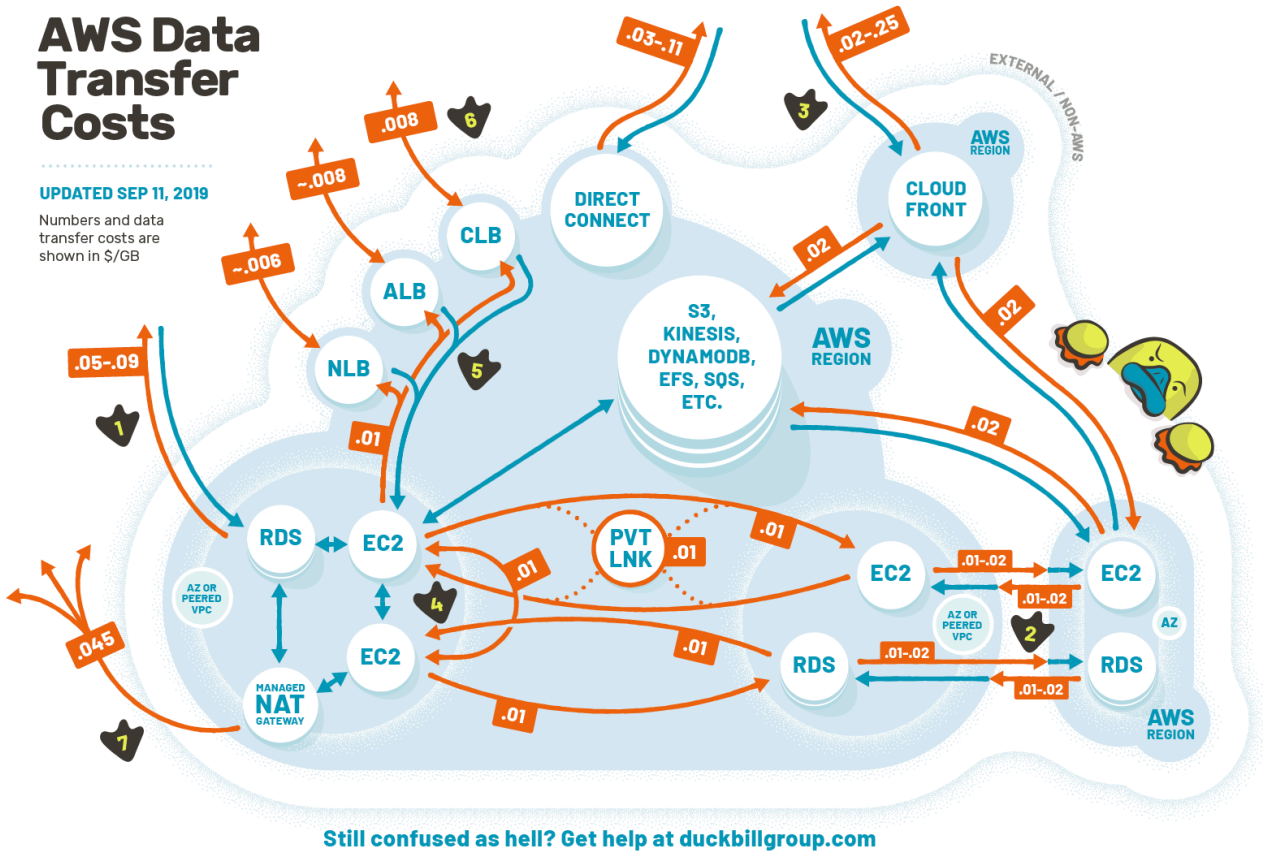


Consumer Group – cross rack reads / writes

- My diagrams are nicely aligned for sake of clarity, but in real life it looks more like that:



[side note] AWS Data transfer costs



1. Inbound traffic is typically free – outbound is not. Some (but not all) internal traffic is **free.**
2. Direct outbound data starts at **$.09/GB** for less than 10TB, and discounts with volume. **First 1GB is free.**
3. Region-to-region traffic is **$.02/GB** when it exits a region for indicated services except between us-east-1 and us-east-2, where it's **$.01/GB.** Even data wants to get out of Ohio.
4. Outbound CloudFront prices are highly variable by geography and regional edge cache and start at **$.085/GB** in US/Canada.
5. Internal traffic via public or elastic IPs incurs **additional fees** in both directions.
6. Cross-AZ EC2 traffic within a region costs half as much as region-to-region! ELB-EC2 traffic is **free** except outbound crossing AZs.
7. Elastic Load Balancing: Classic and Network LB is priced per GB. Application LB costs are in LCUs, not $/GB.
8. Traffic via Managed NAT Gateway – regardless of destination – costs an additional **$.045/GB** on top of other transfer, including internal transfer (S3, Kinesis, etc.).

Inspired by Open Guide to AWS's data transfer diagram
github.com/open-guides/og-aws

Image courtesy: https://www.lastweekinaws.com/wp-content/uploads/2019/09/dbg010-datatxfr-infographic-20190911-1.png

PartitionAssignor interface (KAFKA-2464)

- "client-side assignment for new consumer", i.e. a set of interfaces to manage how consumers assign their partitions.
- Default (dummy) implementations, RangeAssignor (the default) and RoundRobinAssignor.
- But:

https://github.com/apache/kafka/blob/trunk/clients/src/main/java/org/apache/kafka/clients/consum

Search

```
34   *
35   * In some cases, it is useful to forward additional metadata to the assignor in order to make
36   * assignment decisions. For this, you can override {@link #subscription(Set)} and provide custom
37   * userData in the returned Subscription. For example, to have a rack-aware assignor, an implementation
38   * can use this user data to forward the rackId belonging to each member.
39   *
40   * This interface has been deprecated in 2.4, custom assignors should now implement
41   * {@link org.apache.kafka.clients.consumer.ConsumerPartitionAssignor}. Note that maintaining compatibility
42   * for an internal interface here is a special case, as {@code PartitionAssignor} was meant to be a public API
43   * although it was placed in the internals package. Users should not expect internal interfaces or classes to
44   * not be removed or maintain compatibility in any way.
45   */
46  @Deprecated
47  public interface PartitionAssignor {
48
```

It looks like Netflix is running a rack-aware consumer assignor!

SmartAssignor

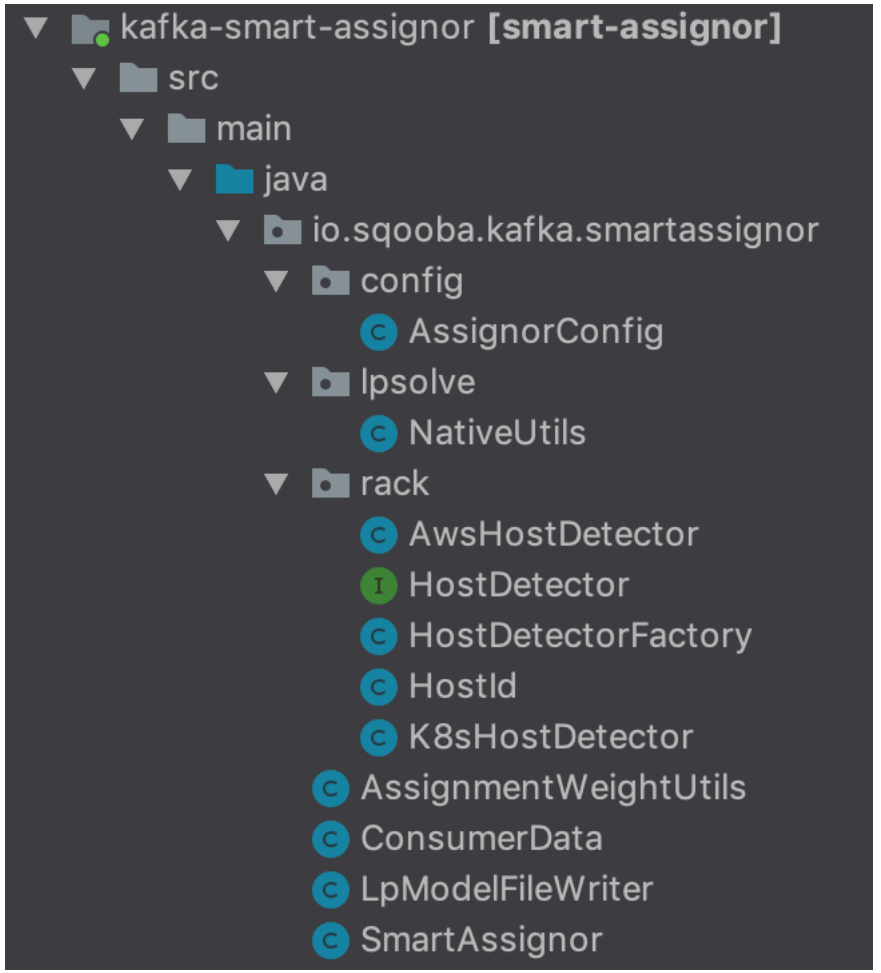
Custom PartitionAssignor with data affinity between replicas and consumers

- **Optimize assignments based on partitions' distance using linear programming:**

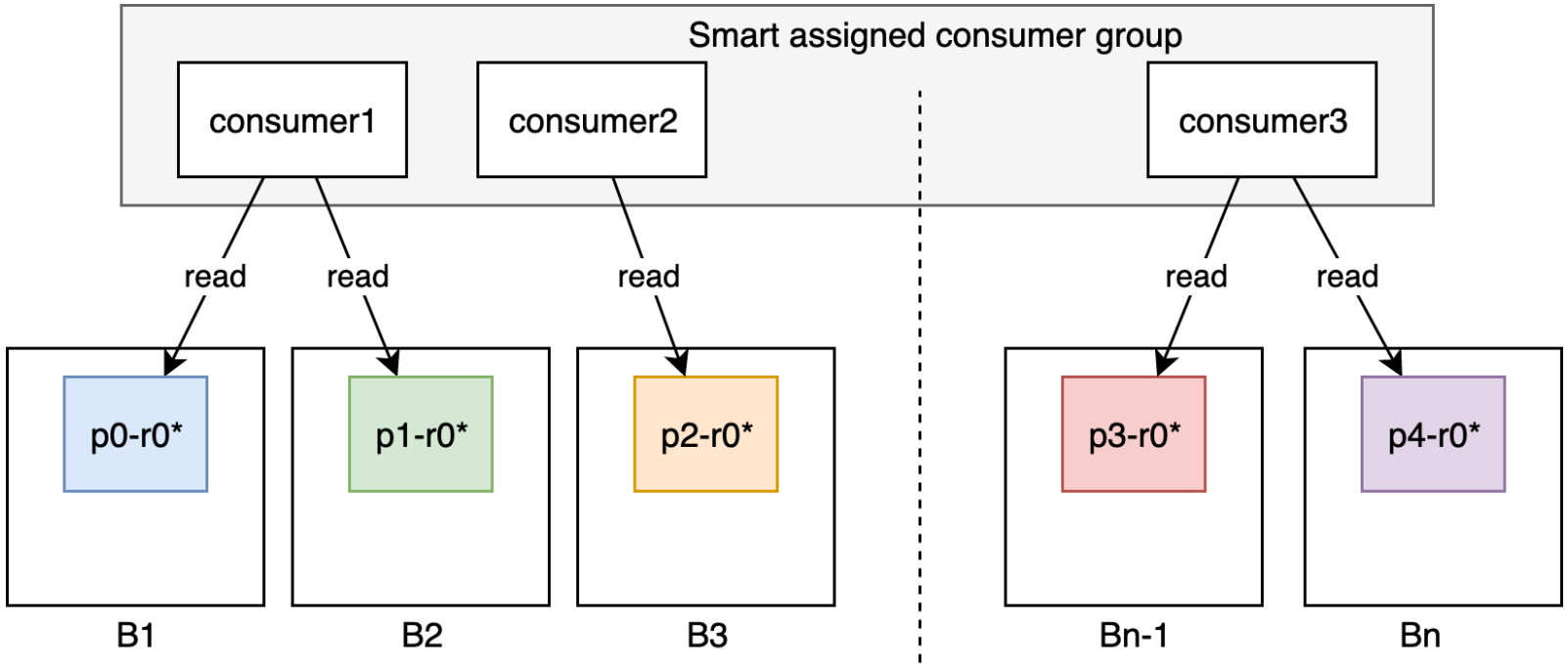
  - multi-level location such as /eu-central-1b/kafka-01
  - client location can be looked up from the metadata (k8s, aws, ...)

- **Preserve current assignments when no changes**

- **Benefits mostly are**
  - inter-availability zones traffic cost reduction
  - less burden when a consumer group member is entering the group.



SmartAssignor (cont'd)



All what I told before was right, until KIP-392…

KIP 392: Allow consumers to fetch from closest replica

- KIP-392 was released s part of Kafka 2.4 (Dec 2019) … as a task…

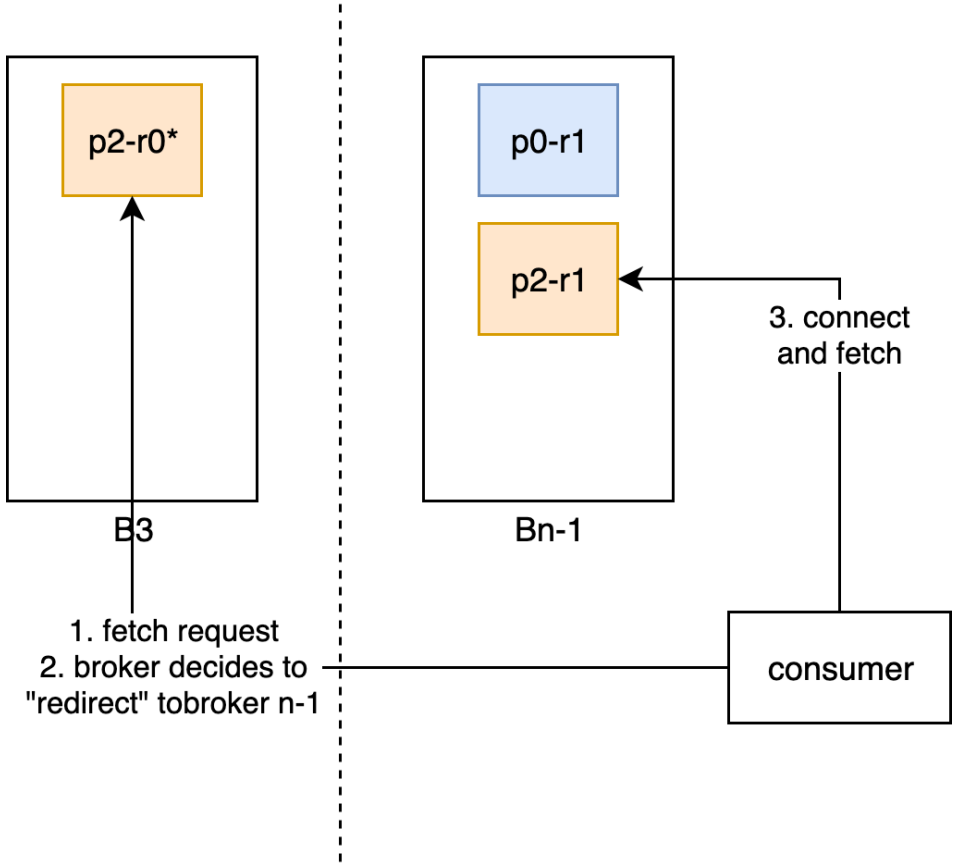
Task

- [KAFKA-8443] - Allow broker to select a preferred read replica for consumer
- [KAFKA-8765] - Remove "unstable" annotations
- [KAFKA-9072] - Add Section to Streams Developer Guide for Topology Nam

- Defaults to current behavior (choose the leader of the partition)
- Turn on in broker's configuration:
  replica.selector.class=RackAwareReplicaSelector
  broker.rack=rack01
- And in consumer's configuration:
  client.rack=rack01
  this value should somehow match what is in broker.rack.

Write to the leader, read from closest replica

- Leader decide if consumer can read from somewhere else
- Latency might be slighty higher
- Throughput can be multiplied by the number of replica
- Consumer locality to replica can save data transfer cost.



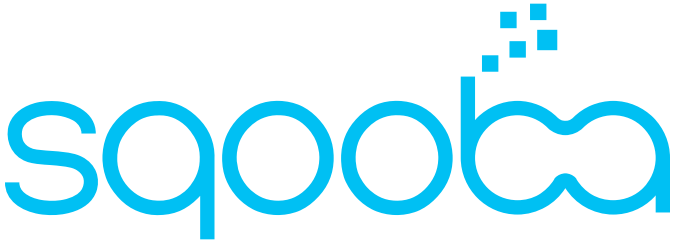
Take Away

Data locality in Kafka

- **You'd better know where your data is stored**
- **You should design your cluster with data localization in mind**
- **Kafka is getting better at providing options to take advantage of data locality (thanks Confluent Cloud!!)**
- **There is lot of room for improvement, depending on the use-case – be creative!**

Pointers

- KIP 392: https://cwiki.apache.org/confluence/display/KAFKA/KIP-392%3A+Allow+consumers+to+fetch+from+closest+replica
- https://www.apache.org/dist/kafka/2.4.0/RELEASE_NOTES.html
- https://www.confluent.io/blog/multi-region-data-replication/
- https://kafka-optimizer.sqooba.io/



Thanks!
Questions?

BTW we're hiring, reach out to diver@sqooba.io or @sqooba_io if you enjoy deep diving into Kafka!