KIP-726: Make the "cooperative-sticky, range" as the default assignor

Status

Current state: *"Accepted"*

Discussion thread: here

JIRA: *KAFKA-12473*

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Now that 3.0 is coming up, we can change the default ConsumerPartitionAssignor to something better than the **RangeAssignor**. The original plan was to switch over to the StickyAssignor, but now that we have incremental cooperative rebalancing, we should consider using the new **CooperativeStickyAssignor** instead: this will enable the consumer group to follow the COOPERATIVE protocol, improving the rebalancing experience OOTB.

In Kafka, we currently support the following assignors:

	Supported strategy	Feature	Drawback
RangeAssignor	Eager	Current default value. The 1st assignor we have	possible to generate heavily skewed assignments when the consumer topic subscriptions are not identical
RoundRobinAssignor	Eager	Improvement for RangeAssignor to have balanced assignment	Didn't consider the overheads of reassignment
StickyAssignor	Eager	Improvement for RoundRobinAssignor/RangeAssignor to preserve the existing assignments to reduce some of the overheads of a reassignment	Will have stop-the-world issue when doing rebalance
CooperativeStickyAssignor	Eager, Cooperative	To be default value in 3.0 as in this KIP described, by having multiple rounds of rebalance to avoid the stop-the-world issue as described in KIP-429	

As above table showed, we should change the default assignor now to enable cooperative rebalancing by default. However we must be careful to do so in a backwards compatible way that does not require any specific steps to be taken by the user for them to safely upgrade, as is currently the case for cooperative rebalancing, which requires a double rolling bounce. So in addition to changing the default partition assignor, we must also improve the rebalancing protocol selection mechanism. This will allow the consumer group to remain on the old protocol during the rolling upgrade, and only complete the upgrade to cooperative rebalancing once all members are safely on the new bytecode.

Public Interfaces

With this KIP, the default value of *`partition.assignment.strategy`* changes from `"RangeAssignor"` to `"CooperativeStickyAssignor, RangeAssignor".` Since the assignor list is ordered by preference, this means that the "CooperativeStickyAssignor" will become the new default partition assignor (once all members have been upgraded/used the new default).

However this in itself would not effect the rebalancing protocol, as this is chosen during startup as "the highest protocol which is commonly supported by all assignors". So in addition to changing the default assignor, this static rebalance protocol selection will be replaced by a dynamic protocol that depends on the actual assignor that was selected for use in the rebalance. This means that cooperative rebalancing will also become enabled by default.

Note that this change will also automatically be inherited by sink connectors, like any other application that uses Kafka consumers, as long as a consumer assignor is not explicitly defined in their configuration.

Proposed Changes:

If the consumers rely on the default value of *`partition.assignment.strategy`*, they will automatically enable cooperative rebalancing for new applications, and once all members of the group have been upgraded for existing ones. During the upgrade, it will continue to follow the EAGER protocol with the RangeAssignor. Once all members support the CooperativeStickyAssignor, this will be selected for use by the group coordinator, and reported back to each member. At this point the consumer is allowed to enable the COOPERATIVE protocol, and hold onto its owned partitions during a rebalance.

Still, there are scenarios in which some members may be on COOPERATIVE while others in the group are not. For example if some member(s) on the old bytecode missed a rebalance, or the operator accidentally started up a consumer that only supports RangeAssignor, or during a downgrade. The danger in mixing EAGER and COOPERATIVE rebalancing, and the reason for the original double rolling bounce upgrade strategy, is that some partitions may be claimed by two consumers at once. This can occur if some members are using COOPERATIVE and don't revoke their partitions prior to the rebalance, while an EAGER-only assignor is chosen which will freely reassign those un-revoked partitions. At the heart of the dynamic protocol mechanism is the insight that the danger this situation presents is due to the possibility of both consumers committing offsets for that partition, not that both believe to be owners at the same time – in fact that situation may occur already, if a member has dropped out of the group. We therefore take a similar approach here: in the case of some members upgrading to COOPERATIVE but seeing an EAGER assignor be selected, they will invoke the onPartitionsLost callback and disable committing until they have rejoined the group.

With this proposal

- New applications will enable cooperative rebalancing by default
- Existing applications which don’t set an assignor can safely upgrade using a normal rolling bounce, and will automatically transition to cooperative rebalancing
- Existing applications which do set an assignor that uses EAGER can likewise upgrade their applications to COOPERATIVE with a single rolling bounce (by adding the "cooperative-sticky" assignor to the beginning of the assignors list)
- Once on new version (ex: V3.0), applications can safely go back and forth between EAGER and COOPERATIVE, based on the configured assignors
- Applications can safely downgrade

Detailed implementation can be found in KAFKA-12477.

Compatibility, Upgrade path

No special upgrade path is necessary, and this change should be transparent to the user – they will just see that cooperative rebalancing is enabled.

Rejected Alternatives

1. Some cooperative-sticky related defects might not freed before V3.0
   We've marked important defects as blocker for V3.0, ex: KAFKA-12896. Please raise any important defect if you found any.
2. cooperative-sticky assignor is also very new for C/C++ users in librdkafka, so not many in that community have tried incremental cooperative yet. And bugs are still recently being worked out there too.
   I checked this library and found currently only 1 cooperative-sticky related bug open, which is good (10 bugs are fixed). Anyway, I think the clients can always change the assignor to other assignors if there are still bugs in the library.