KIP-705: Selectively Disable Topology Optimizations

Status

Current state: *Under Discussion*

Discussion thread: *here*

JIRA: *KAFKA-12192*

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

Motivation

Historically, the `topology.optimizations` configuration for Kafka Streams was meant to help ensure backwards compatibility (see KIP-295: Add Streams Configuration Allowing for Optional Topology Optimization) and assumed that there would be only two valid options: none and all. There are some optimizations under that umbrella, however, that may have unintended side-effects. Specifically, the source-changelog optimization that reuses the source topic as a changelog has two issues:

First, it assumes that the serde being used is symmetric (i.e. serialize(deserialize(bytes)) == bytes). Because of this assumption, Kafka Streams is able to pass the source topic name (instead of the changelog topic name, which doesn't exist) into the serializer that writes to the RocksDB store (see KAFKA-10179). To illustrate why this is not always desirable, consider the following use case:

- The serde is not symmetric (e.g. an AVRO serde with different read/write schemas)
- The serializer has side effects (e.g Confluent Schema Registry serializers, which will register a new schema version to a subject if unknown)

In this situation, Kafka Streams will register a new schema to the source topic subject in Schema Registry, even though no messages with that schema exist in the source topic.

Second, it assumes that there are no "dropped" messages in the deserialization exception handler (see KAFKA-8037 for more details).

For the two reasons above, some users may want to disable the source-changelog optimization while keeping other optimizations under the `topology. optimizations` config enabled. In the future, there may also be more optimizations that users may want to selectively enable/disable.

Public Interfaces

We will add the following configuration to `StreamsConfig`:

```
TOPOLOGY_OPTIMIZATION_DISABLED_LIST_CONFIG = "topology.optimization.disabled.list";
TOPOLOGY_OPTIMIZATION_DISABLED_LIST_DOC =
    "If topology.optimization" is set to \"all\", this configuration allows users to selectively disable some
optimizations. "
    + "Currently, users can toggle the following optimizations: " + SOURCE_CHANGELOG;

SOURCE_CHANGELOG_OPTIMIZATION = "source_changelog";

...

.define(TOPOLOGY_OPTIMIZATION_DISABLED_LIST_CONFIG,
        Type.LIST,
        "",
        ValidList.in(SOURCE_CHANGELOG_OPTIMIZATION),
        Importance.LOW,
        TOPOLOGY_OPTIMIZATION_DISABLED_LIST_DOC)
```

Proposed Changes

If the topology.optimization.disabled.list contains "source_changelog", then the InternalStreamsBuilder class will not call the optimizeKTableSourceTopics() method as part of maybePerformOptimizations.

Compatibility, Deprecation, and Migration Plan

- *What impact (if any) will there be on existing users?*
  - N/A - this will not affect any existing users
- *If we are changing behavior how will we phase out the older behavior?*
  - N/A - this does not phase out older behavior
- *If we need special migration tools, describe them here.*
  - N/A
- *When will we remove the existing behavior?*
  - N/A

Rejected Alternatives

- Add a allow-list of optimizations as a valid option to topology.optimization config (in addition to "all" and "none"). This was rejected because most of the optimizations do not have a known valid reason to disable (unlike the source_changelog optimization) and would require users to know about all the optimizations that Kafka Streams decides to implement.
- Add a separate configuration for each disabled optimization. This seems more difficult to track and maintain than the single list of disabled optimizations, and is more difficult to extend.
- Require users to disable the source-changelog optimizaiton by adding a dummy materialized mapValues step. This requires users to understand the underlying implementation details of Kafka Streams, and may sometimes result in unnecessary materialization altogether (see https://github.com/confluentinc/ksql/issues/6750).