Item 195
git_comments:

1. * * Acquires permits for the rate limiter.
2. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
3. * * Sets the desired rate for the rate limiter. * @param rate
4. * * An interface to create a ratelimiter * * <p>The ratelimiter is configured via {@link #setRate(long)} and * created via {@link #open(RuntimeContext)}. * An example implementation can be found {@link GuavaFlinkConnectorRateLimiter}. *
5. * * A method that can be used to create and configure a ratelimiter * based on the runtimeContext. * @param runtimeContext
6. * * Creates a rate limiter with the runtime context provided. * @param runtimeContext
7. * Runtime context. *
8. * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
9. Ensure permits > 0
10. * * Set the global per consumer and per sub-task rates. * @param globalRate Value of rate in bytes per second.
11. * RateLimiter. *
12. * Rate in bytes per second per subtask of the consumer.
13. * * An implemetation of {@link FlinkConnectorRateLimiter} that uses Guava's RateLimiter for rate limiting.
14. * Rate in bytes per second for the consumer on a whole.
15. If a rateLimiter is set, then call rateLimiter.open() with the runtime context.
16. If a rateLimiter is set, then call rateLimiter.open() with the runtime context.
17. * * RateLimiter to throttle bytes read from Kafka. The rateLimiter is set via * {@link #setRateLimiter(FlinkConnectorRateLimiter)}.
18. * * Set a rate limiter to ratelimit bytes read from Kafka. * @param kafkaRateLimiter
19. * * * @param records List of ConsumerRecords. * @return Total batch size in bytes, including key and value.
20. * * Get records from Kafka. If the rate-limiting feature is turned on, this method is called at * a rate specified by the {@link #rateLimiter}. * @return ConsumerRecords
21. -- Rate limiting methods ---
22. If a ratelimiter was created, make sure it's closed.
23. Null is an allowed value for the key
24. * Ratelimiter.
25. ---------- Consumer from Kafka in a ratelimited way -----------
26. Approximate bytes/second read based on job execution time.
27. * * Kafka09 specific RateLimiter test. This test produces 100 bytes of data to a test topic * and then runs a job with {@link FlinkKafkaConsumer09} as the source and a {@link GuavaFlinkConnectorRateLimiter} with * a desired rate of 3 bytes / second. Based on the execution time, the test asserts that this rate was not surpassed. * If no rate limiter is set on the consumer, the test should fail.
28. ------- Assertions --------------
29. ---------- Produce a stream into Kafka -------------------
30. 1 byte
31. ---------- RateLimiter config ------------- bytes/second
32. --- ratelimiting properties ---
33. * * A testable KafkaConsumer thread to test the ratelimiting feature using user-defined properties. * The mockConsumer does not mock all the methods mocked in {@link TestKafkaConsumerThread}.
34. -- mock Handover and logger ---
35. -------- new partitions with defined offsets --------
36. Sleep for one second in each consumer.poll() call to return 24 bytes / second
37. Wait for 4 seconds to ensure atleast 2 calls to consumer.poll()
38. In a period of 5 seconds, no more than 3 calls to poll should be made. The expected rate is 1 byte / second and we read 4 bytes in every consumer.poll() call. The rate limiter should thus slow down the call by 4 seconds when the rate takes effect.
39. -------- setup mock KafkaConsumer with test data --------
40. -- Test Kafka Consumer thread ---
41. Ratelimiting dependencies

git_commits:

1. **summary:** [FLINK-11501] [kafka] Add ratelimiting to Kafka consumer (#7679)
   message: [FLINK-11501] [kafka] Add ratelimiting to Kafka consumer (#7679)

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
   body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not

available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

2. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
   body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
3. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
   body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
4. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
   body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
5. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer

body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

6. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
   body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
7. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
   body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
8. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
   body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does

this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

9. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
   body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
10. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
   body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
11. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
   body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
12. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
   body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with

`@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

13. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here] (https://issues.apache.org/jira/browse/FLINK-11501? focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
14. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here] (https://issues.apache.org/jira/browse/FLINK-11501? focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
15. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here] (https://issues.apache.org/jira/browse/FLINK-11501? focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
16. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here] (https://issues.apache.org/jira/browse/FLINK-11501?

focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

17. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
    label: code-design
18. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
19. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
20. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a

`getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

21. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
22. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
23. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
24. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature

is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

25. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
26. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
27. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
28. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer

body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

29. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
    label: code-design
30. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
31. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing,

Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

32. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
33. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
label: code-design
34. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
35. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). -

Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
label: code-design

36. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
37. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
    label: code-design
38. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
39. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a

`getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

40. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
41. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
42. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
label: code-design
43. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not

available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

44. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
45. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
46. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
label: code-design

47. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
48. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
49. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
50. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing,

Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
label: code-design

51. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
52. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
53. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
54. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). -

Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
label: documentation

55. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here] (https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
56. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here] (https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
57. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here] (https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
58. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call.

Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
label: code-design

59. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

60. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

61. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

62. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature

is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

63. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)
    label: code-design
64. **title:** [FLINK-11501][Kafka Connector] Add ratelimiting to Kafka consumer
    body: ## What is the purpose of the change This pull request adds a ratelimiting feature to the Flink Kafka consumer. There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is a useful feature. ## Brief change log - This feature is set by using a feature flag - `kafka.consumer.ratelimiting.enabled` - A`RateLimiterFactory` is used to configure and create a Guava [RateLimiter](https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html) with a desired rate. - The `consumer.poll()` part of the `run()` loop in the `KafkaConsumerThread` is modularized into a `getRecordsFromKafka()` method. - The rate is controlled by setting the bytes received from Kafka as the parameter to the `acquire()` call. ## Verifying this change This change added tests and can be verified as follows: - Added a `testRateLimiting()` test in the `KafkaConsumerThreadTest` class. - Manually verified the change using a test application and screenshots of results are added [here](https://issues.apache.org/jira/browse/FLINK-11501?focusedCommentId=16762965&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-16762965). - Added a `testRateLimitedConsumer()` to `Kafka09ITCase` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (yes / **no** / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) ## Documentation - Does this pull request introduce a new feature? (**yes** / no) - If yes, how is the feature documented? (not applicable / docs / **JavaDocs** / not documented)

github_pulls_comments:

1. Thanks a lot for your contribution to the Apache Flink project. I'm the @flinkbot. I help the community to review your pull request. We will use this comment to track the progress of the review. ## Review Progress * âœ… 1. The [description] looks good. - Approved by @tweise [committer] * âœ… 2. There is [consensus] that the contribution should go into to Flink. - Approved by @tweise [committer] * â “ 3. Needs [attention] from. * âœ… 4. The change fits into the overall [architecture]. - Approved by @tweise [committer] * âœ… 5. Overall code [quality] is good. - Approved by @tweise [committer] Please see the [Pull Request Review Guide](https://flink.apache.org/reviewing-prs.html) for a full explanation of the review process.<details> The Bot is tracking the review progress through labels. Labels are applied according to the order of the review items. For consensus, approval by a Flink committer of PMC member is required <summary>Bot commands</summary> The @flinkbot bot supports the following commands: - `@flinkbot approve description` to approve one or more aspects (aspects: `description`, `consensus`, `architecture` and `quality`) - `@flinkbot approve all` to approve all aspects - `@flinkbot approve-until architecture` to approve everything until `architecture` - `@flinkbot attention @username1 [@username2 ..]` to require somebody's attention - `@flinkbot disapprove architecture` to remove an approval you gave earlier </details>
2. @flinkbot approve description
3. @flinkbot approve consensus
4. @rmetzger the flinkbot is awesome!
5. Thank you :) There's a cool new feature coming up in the pipeline: Labeling of the PR progress.
6. Thanks for taking a look @becketqin and @tweise. I think adding an interface that can extend to other connectors and also not restricting consumers to use Guava's RateLimiter makes sense. Maybe something simple like below will work for the interface? ``` public interface FlinkConnectorRateLimiter { public void open(StreamingRuntimeContext runtimeContext); public <T> T create(long rate); public void

setRate(long rate); public void close(); } ``` And connector specific RateLimiters can be defined to implement this interface. Also, point taken on keeping these discussions on the mailing list in the future.
7. @tweise @becketqin I updated the code to include a `FlinkConnectorRateLimiter` and also included a `DefaultKafkaRateLimiter` that implements the interface using Guava's RateLimiter. Let me know what you think.
8. @tweise I added a test to the `Kafka09ITCase` that runs a job that uses the ratelimiter (and asserts an approximate rate). I also updated the `GuavaFlinkConnectorRateLimiter` to be serializable (which was a bug that I discovered on writing the IT case). Let me know your thoughts!
9. As per discussion on the [mailing list] (https://lists.apache.org/thread.html/3de9d2353cf22aea0448fb744314103b5f88195216acc3bff449354a@%3Cdev.flink.apache.org%3E), I have moved `FlinkConnectorRateLimiter` and `GuavaFlinkConnectorRateLimiter` to `org.apache.flink.api.common.io.ratelimiting`. The latest commit includes changes to update references. One slightly unfortunate consequence of doing this is including the shaded guava dependency in the `flink-core/pom.xml`. I don't know if that's acceptable? @tweise PTAL when you have a chance and let me know what you think!
10. @flinkbot approve all
11. @glaksh100 thanks for the contribution!
12. @tweise Thank you for the patient and thorough review :)

github_pulls_reviews:

1. Do we need this prefix? Can all the source and sink share the same configuration nameï¼Ÿ If we do need the prefix, can we put this in somewhere like `FlinkKafkaConsumerBase`?
2. Do we want to directly import class from the shaded jar?
3. It is a little unfortunate that we have to make an additional iteration over the records to get the sizes. Another option is putting the throttling logic in the `AbstractFetcher`.
4. Can we move this class to a separate package so it can be shared by all the other connectors? For example, a `flink-connector-throttling` module in `flink-connectors`.
5. **body:** Personally, I prefer independent configurations if possible. I am wondering if the following configuration names would be clear enough: ``` CONSUMER_MAX_BYTES_PER_SECOND_CONFIG="consumer.max.bytes.per.second" CONSUMER_MAX_RECORDS_PER_SECOND_CONFIG="consumer.max.records.per.second" ``` In the future we can add configurations for the producers. The default value of the above configurations could be either be set to Long.MAX_VALUE, or -1 to indicate not throttling.
   label: code-design
6. There is not need for this prefix. Rate limiting should be configured per consumer. I have a more general comment on that.
7. There is no need for this configuration. When a rate limiter is set on the consumer, then it is "enabled", otherwise it is null and no rate limiting takes place.
8. I would suggest to remove this property based configuration and have instead properties on the rate limiter that the user sets directly.
9. This import should not be here at all after we find the appropriate abstraction. But in general, we should use the shaded dependency or not?
10. Makes sense to enable ratelimiting based on whether or not a consumer's ratelimiter is null. To @becketqin's point, I think we would still need to pass the rate value itself from the consumer to its ratelimiter. Perhaps we could standardize that property across connectors?
11. Yes, I think it makes sense to move this logic to a separate package within the `flink-connectors` module. I will do that once we reach an agreement on the discussion. Thanks for suggesting.
12. The reason for doing the iteration here was because we needed the record sizes prior to deserialization and I believe the records are already deserialized at the `AbstractFetcher` level?
13. Yeah, I think the consumer prefix is not adding any value here. Makes sense to not have it.
14. yes, if you want to use something from guava then you should use the shaded dependency. The only exception is if a another dependency exposes a guava-related API.
15. @zentol @tweise Just for my own curiosity. I agree that we should shade the jar to avoid interference with other user imports. What I am wondering is how should that be done. Whether it should be done in the pom.xml or from the import explicitly. I am not sure about the difference between those two ways. But I did not see such explicit imports in Flink anywhere else. Am I missing something?
16. The `AbstractFetcher` has two subclasses - `Kafka09Fetcher` and `Kafka08Fetcher`. The serialization is done inside them. `Kafka09Fetcher` is sort of an abstract fetcher for Kafka 0.9+. `Kafka08Fetcher` is more of a legacy. Maybe we can just do that in `Kafka09Fetcher`?
17. **body:** I have not made this change yet. Perhaps we could do this as a part of augmenting the consumer and making the code more customizable (in another PR i.e.) ?
    label: code-design
18. I ended up passing the rate value to the setter that sets the ratelimiter on the consumer. Let me know if that looks alright.
19. Why the second parameter? That should be a setter on `DefaultKafkaRateLimiter`
20. Since there is nothing Kafka specific here, but the implementation is specific to Guava, would `GuavaFlinkConnectorRateLimiter` be a better name?
21. **body:** This is confusing, the rate limiter was already created by the user. Why is `acquire` not defined here?
    label: code-design
22. I think that is a better name. Thanks for the suggestion.
23. **body:** This does look confusing. Essentially, we need `runtimeContext` to be initialized to be able to determine the per-subtask rate. Perhaps, only the `open()` method should suffice (and all related logic can be implemented there)? As for `acquire()` not being here, that was my bad. I have fixed this to include `acquire()` and completely removed the Guava import in the `KafkaConsumerThread` class (which was not the case before).
    label: code-design
24. Fixed.
25. **body:** It looks like `create` is redundant. The work done in that method can be moved to `open`.
    label: code-design
26. The calls to close are missing.
27. Thanks for catching. Updated to call `close()` in the `KafkaConsumerThread` both in the `finally{}` block of the `run()` method and in the `shutdown()` method.
28. Updated to only have `open()` and also moved the `create()` logic in the `GuavaFlinkConnectorRateLimiter` to reflect this.
29. Why repeat this here (already defined in `FlinkKafkaConsumer09`)?
30. **body:** Fair point. I removed the field and added a getter to access the `rateLimiter` from the super class. Let me know if that doesn't seem reasonable.
    label: code-design

31. I opened a mailing list thread here to discuss where this module should live: https://lists.apache.org/thread.html/3de9d2353cf22aea0448fb744314103b5f88195216acc3bff449354a@%3Cdev.flink.apache.org%3E
32. This has now been moved to `org.apache.flink.api.common.io.ratelimiting`
33. remove
34. **body:** Unless there is precedence let's remove this comment - it is pretty obvious what these methods are there for :)
    label: code-design
35. remove
36. remove
37. remove
38. **body:** indent
    label: code-design
39. Thanks for writing this additional test. Just to confirm, would this test generate a higher rate without the limit?
40. remove
41. javadoc - mention what this method would be useful for
42. **body:** Added a javadoc.
    label: documentation
43. Yes. That is correct. The rate is close to 45 bytes/second when run without setting the ratelimiter.
44. Sounds good, removed :)
45. FlinkConnectorRateLimiter must be serializable, so we should move that up.
46. **body:** OK, that should be good. I would make a note of that in the test though. What makes this a bit shaky is the fact that you could get a false positive when the initialization/execution overhead is just enough to hide a faster read. I'm not sure how that can be done in a much better way though. Perhaps measure the time in the deserializer? Otherwise we can leave it as is.
    label: code-design
47. Moved up.
48. I added details to the javadoc with the test expectation. Moving it to deserializer is a good idea. I updated the test. The gain is ~ 200 ms (not all that much but closer than what I had before). Thanks for the suggestion.
49. remove this and `endTime` below
50. @becketqin I'm not familiar with shaded dependency but let's say that it is a common case we use such explicit imports. You can just search "import org.apache.flink.shaded.guava18` and there are over 100 use points.
51. **body:** It seems you break the code style. `rateLimiter);` should in the next line.
    label: code-design

jira_issues:

1. **summary:** Add a ratelimiting feature to the FlinkKafkaConsumer
   description: There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is useful feature. The approach is essentially involves using Guava's [RateLimiter|https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html] to ratelimit the bytes read from Kafka (in the [KafkaConsumerThread|https://github.com/apache/flink/blob/master/flink-connectors/flink-connector-kafka-0.9/src/main/java/org/apache/flink/streaming/connectors/kafka/internal/KafkaConsumerThread.java]) More discussion here: [https://lists.apache.org/thread.html/8140b759ba83f33a22d809887fd2d711f5ffe7069c888eb9b1142272@%3Cdev.flink.apache.org%3E]
2. **summary:** Add a ratelimiting feature to the FlinkKafkaConsumer
   description: There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is useful feature. The approach is essentially involves using Guava's [RateLimiter|https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html] to ratelimit the bytes read from Kafka (in the [KafkaConsumerThread|https://github.com/apache/flink/blob/master/flink-connectors/flink-connector-kafka-0.9/src/main/java/org/apache/flink/streaming/connectors/kafka/internal/KafkaConsumerThread.java]) More discussion here: [https://lists.apache.org/thread.html/8140b759ba83f33a22d809887fd2d711f5ffe7069c888eb9b1142272@%3Cdev.flink.apache.org%3E]
3. **summary:** Add a ratelimiting feature to the FlinkKafkaConsumer
   description: There are instances when a Flink job that reads from Kafka can read at a significantly high throughput (particularly while processing a backlog) and degrade the underlying Kafka cluster. While Kafka quotas are perhaps the best way to enforce this ratelimiting, there are cases where such a setup is not available or easily enabled. In such a scenario, ratelimiting on the FlinkKafkaConsumer is useful feature. The approach is essentially involves using Guava's [RateLimiter|https://google.github.io/guava/releases/19.0/api/docs/index.html?com/google/common/util/concurrent/RateLimiter.html] to ratelimit the bytes read from Kafka (in the [KafkaConsumerThread|https://github.com/apache/flink/blob/master/flink-connectors/flink-connector-kafka-0.9/src/main/java/org/apache/flink/streaming/connectors/kafka/internal/KafkaConsumerThread.java]) More discussion here: [https://lists.apache.org/thread.html/8140b759ba83f33a22d809887fd2d711f5ffe7069c888eb9b1142272@%3Cdev.flink.apache.org%3E]

jira_issues_comments:

1. Adding some screenshots of results from testing the Guava RateLimiter approach on the Kafka consumer. !RateLimiting-1.png|width=981,height=305! !Ratelimiting-2.png|width=981,height=305!