KIP-611: Improved Handling of Abandoned Connectors and Tasks

Status

Current state: *Abandoned*

Discussion thread: Here

JIRA: Unable to render Jira issues macro, execution error.

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

Background

In an ideal world, every connector and task behaves itself and responds promptly to requests made by the framework to initialize, process data, shut down, etc. Unfortunately, sometimes connectors or tasks block, either indefinitely or for an abnormally long period of time.

If this happens for a connector, the worker hosting that connector will become severely degraded; most of the functional parts of its REST API will become unavailable, it will fail to pick up rebalances, and will eventually fall out of the group and its connectors and tasks will be reassigned to other workers in the cluster. There is currently work in progress to reduce the impact of a blocked connector on the worker hosting it, which causes a blocked connector to be treated more like a blocked task.

If a task becomes blocked, the results are less severe. The framework gives each task its own dedicated thread, and all interactions with the task (such as invoking `start` , `stop` , and `poll` / `put`) take place on that thread. If a task becomes blocked during any of these methods, the impact is limited to the task itself. If the worker has to shut down a task, the task is given a graceful shutdown period to allow it to de-allocate resources, finish committing data, etc. If the shutdown timeout is exceeded, the task is abandoned and the framework moves on. No more calls to `put` or `poll` are made on that task, so the only major problem posed by this now-abandoned task is that it may still hold onto resources. Most of these resources are task-specific and include things like file descriptors, network connections, or resource locks. However, there are also some worker resources that are also still retained, including the thread for the task itself, the memory allocated for it, and the producer or consumer set up by the framework on its behalf.

Motivation

The goal of this KIP is to add some lightweight improvements to the Connect framework that should make it easier to deal with these abandoned connectors and tasks. These improvements include:

- JMX metrics that allow for real-time, on-demand monitoring of a worker's abandoned connectors and tasks
- A new worker configuration property that allows for a user-specified timeout period for connectors
- Logging additions to grant total insight into the worker's interactions with its connectors and tasks

Public Interfaces

New JMX Metrics

Some metrics will be added to help gauge the overall health of the the worker:

MBean name: kafka.connect:type=connect-worker-metrics

Metric/Attribute Name	Description
abandoned-connectors-current	How many connectors that the worker has abandoned and which have not yet shut down
abandoned-connectors-total	How many connectors that the worker has had to abandon over its lifetime, including connectors that both have and have not yet completed shutdown
abandoned-tasks-current	How many tasks that the worker has abandoned and which have not yet shut down
abandoned-tasks-total	How many tasks that the worker has had to abandon over its lifetime, including tasks that both have and have not yet completed shutdown

And some other metrics will be added to help diagnose which connectors or tasks are blocking:

MBean name: kafka.connect:type=abandoned-connector-metrics,connector=([-.\w]+)

Metric/Attribute Name	Description
current	How many instances of this connector that the worker has abandoned and which have not yet shut down
total	How many instances of this connector that the worker has had to abandon over its lifetime, including instances that both have and have not yet completed shutdown

MBean name: kafka.connect:type=abandoned-task-metrics,connector=([-.\w]+),task=([\d]+)

Metric/Attribute Name	Description
current	How many instances of this task that the worker has abandoned and which have not yet shut down
total	How many instances of this task that the worker has had to abandon over its lifetime, including instances that both have and have not yet completed shutdown

Worker Configuration Properties

A new configuration property will be introduced that will allow users to fine-tune how long connectors are allowed to take during shutdown.

Name	Type	Default	Importance	Doc
`connector.shutdown.graceful.timeout.ms`	`LONG`	`5000`	LOW	Amount of time to wait for connectors and tasks to shutdown gracefully. This is the total amount of time, not per connector or task. All connectors and tasks have shutdown triggered, then they are waited on sequentially. If specified, will take priority over the now-deprecated task.shutdown.graceful.timeout.ms property

Note that the default value for this property matches the current default for the `task.shutdown.graceful.timeout.ms` property.

Additionally, the `task.shutdown.graceful.timeout.ms` property will be deprecated.

Logging Additions

Tracking method calls

In order to provide a complete history of the worker's interactions with connector and task instances, the framework will log whenever it is about to invoke a connector or task method, and whenever that invocation completes (successfully or otherwise). The level of these new messages will be `TRACE`, and their format will be:

```
[About to invoke|Finished invoking|Failed to invoke] method <class>.<method>
```

The initial text will be "About to invoke" before the method is invoked, "Finished invoking" if the method returns without throwing an exception, or "Failed to invoke" if the method throws an exception. The simple class name of the connector or task will be substituted in for the "<class>" placeholder, and the name of the method will be substituted in for the "<method>" placeholder.

New context information

In addition, every connector and task instance will be given a sequence number that uniquely identifies it across the lifetime of the worker. The sequence number will start at one and increment monotonically every time a new instance of that connector or task is created. This sequence number will be added to the worker's connector and task logging contexts, which were introduced added in KIP-449.

The new format will be `[<connectorName>|scope|seqno]<sp>`, where the sequence number for the task or context is substituted in for the "seqno" text. This will only take place for connectors started by users (as opposed to connector objects instantiated by the framework to perform configuration validation, gather version information, etc.), and for tasks outside the context of periodic source offset commits.

Here are some before/after examples of how log contexts will be altered:

Creating a connector, before:

```
[2019-04-02 17:01:38,315] INFO [local-file-source|worker] Creating connector local-file-source of type
FileStreamSource (org.apache.kafka.connect.runtime.Worker:227)
```

Creating a connector, after:

```
[2019-04-02 17:01:38,315] INFO [local-file-source|worker|1] Creating connector local-file-source of type
FileStreamSource (org.apache.kafka.connect.runtime.Worker:227)
```

Creating a task, before:

```
[2019-04-02 17:01:38,320] INFO [local-file-source|task-0] Creating task local-file-source-0 (org.apache.kafka.
connect.runtime.Worker:395)
```

Creating a task, after:

```
[2019-04-02 17:01:38,320] INFO [local-file-source|task-0|6] Creating task local-file-source-0 (org.apache.kafka.
connect.runtime.Worker:395)
```

Validating

Proposed Changes

We will add the relevant metrics, changes to worker configuration properties, and logging messages as specified in the Public Interfaces section.

Compatibility, Deprecation, and Migration Plan

New JMX Metrics

The proposed metrics are brand-new and therefore fully backwards-compatible.

Worker Configuration Properties

The `task.shutdown.graceful.timeout.ms` property will be deprecated and scheduled for removal at the next major release. At the time of writing, the next major release is 3.0.

The additional `connector.shutdown.graceful.timeout.ms` property is a worker configuration property with a default value. As a result, the only potential compatibility concerns are that it may conflict with a property used by existing REST extensions. The name of this property is long and specific enough that the chances for this conflict are low enough to be acceptable.

Logging Additions

The new method logging should be fully backwards-compatible as it does not modify any existing logs.

The changes to the logging context are not fully backwards-compatible; it is possible that users are parsing worker logs and expecting the format of the connector context to contain only a certain number of "|" characters, for example. We may consider adding the sequence number as an additional logging key that can be added to a commented-out line in the default `config/connect-log4j.properties` file if these concerns are severe enough.

Rejected Alternatives

Tracking Blocked Connector/Task Methods

Summary: in addition to tracking the current and total number of abandoned instances of a specific connector or task, we might also try to deduce which methods of that instance are blocking. This could be accomplished by exposing the last method to be invoked by the framework on an instance before it had to be abandoned.

Rejected because: JMX metrics are useful for proactively monitoring the health of a worker, but not as useful for debugging actual issues as they do not give a complete history of the lifetime of the worker and instead can only provide a snapshot. Once it's clear that there's a problem with a worker, it's easier to check its logs to diagnose what that issue is, hence the proposed logging additions.

Keeping task.shutdown.graceful.timeout.ms

Summary: instead of deprecating the `task.shutdown.graceful.timeout.ms` property, retain it, and apply the timeout specified by the `connector.shutdown.graceful.timeout.ms` exclusively to connectors.

Rejected because: there is no clear advantage to being able to specify separate timeout periods for connectors and tasks, and it'd be a configuration burden to have to specify timeouts for both even if a single timeout is all that is desired.

Co-Opting task.shutdown.graceful.time.ms

Summary: instead of deprecating the `task.shutdown.graceful.timeout.ms` property and introducing the `connector.shutdown.graceful.timeout.ms` property, keep the `task.shutdown.graceful.timeout.ms` property and apply it to both connectors and tasks.

Rejected because: "task" has a very specific meaning with Connect and it's important to respect that definition. "Task" only ever refers to a connector task and not the connector itself, whereas "connector" can be used to refer to a connector and all of its tasks (the "logical" connector that the user configures, as opposed to the "physical" Java Connector object that the framework brings up).

Replacing "worker" Log Context Scope with "connector"

Summary: because connectors may be given their own dedicated threads if/when work on

is complete, the scope "worker" is a bit of a misnomer, and it would probably be more intuitive to users to replace it with "connector".

Rejected because: breaks backwards compatibility and doesn't add any extra information to the logs emitted by workers since the new term would be used in the exact same places the old one currently is. Originally, the term "connector" may have been preferable, but it's not worth it now to change things and risk breaking users' log collection tools.

Using Existing Connector and Task Metrics to Report on Abandoned Instances

Summary: instead of adding the new `kafka.connect:type=abandoned-connector-metrics,connector=([-.\w]+)` and `kafka.connect:type=abandoned-task-metrics,connector=([-.\w]+),task=([\d]+)` MBeans to report on abandoned instances of specific connectors and tasks, use the existing `kafka.connect:type=connector-metrics,connector=([-.\w]+)` and `kafka.connect:type=connector-task-metrics,connector=([-.\w]+)`,task=([\d]+) MBeans.

Rejected because: the lifetimes of the existing MBeans are directly tied to the lifetimes of the connectors and tasks that they report on. Retaining those MBeans even after a connector or task has apparently been deleted (but has in reality only been abandoned by the worker after failing to shut down in time) may break existing tooling that relies on that correlation. Also, a new namespace provides more room to add new connector- and task-specific metrics for abandoned instances if we'd like to.