FLIP-74: Flink JobClient API

Discussion thread	https://lists.apache.org/thread/hyqf650dbrjxc0d0v9x5h328xyb2sftb https://lists.apache.org/thread/fdwlgkoth3f9dhyvv1skzgklsvx9vtlt
Vote thread	
JIRA	~~FLINK-14392~~ - Introduce JobClient API(FLIP-74) CLOSED
Release	1.10

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

Motivation

More and more users ask for client APIs for Flink job managements. Currently users are only able to achieve these functions by REST API. However, because of its string(JSON) return type, REST API is hard to program with.

Goal

1. Introduce a public user-facing class JobClient as job management handler of which users can make use to get job status, cancel job, trigger savepoint and so on.
2. Due to the natural of asynchronous network, we support asynchronous job management operations.

Non Goal

1. JobClient doesn't support job status listener(hook) in this proposal. JobClient itself can be extended to register job listener and call back on job status changed. But it requires to extend Dispatcher to notify client on job changed.
2. JobClient cannot be used for cluster management, i.e., submit job, list jobs and so on. JobClient is only used for managing a specific job and you get it from env.execute or ClusterClient#submitJob.

Description

In this document we introduce a public user-facing class JobClient for job management. The relationship between different level clients and their responsibility is as below

client	responsibility
ClusterDescriptor(external cluster level client)	communicate with external resource manager such as YARN, mesos, k8s, etc.; responsible for deploying Flink application or retrieve ClusterClient.
ClusterClient(Flink application cluster level client)	communicate with Flink application cluster(Dispatcher); responsible for operations on Flink cluster level such as submit job, list job, request cluster status, etc.
JobClient(Flink job level client)	communicate with Flink job manager(for implementation, now with Dispatcher which forwards messages to JM); responsible for operations on Flink job level such as get job status, trigger savepoint and so on.

Narrow to this proposal, as for implementation aspect, JobClient is a thin encapsulation of current ClusterClient with an associated job id on constructed, so that users need not and should not pass JobID for the similar functions of ClusterClient.

JobClient

Interface

Overall interfaces of JobClient is as below.

```
public interface JobClient {
  CompletableFuture<JobResult> getJobResult(classLoader);
  CompletableFuture<JobStatus> getJobStatus();

  CompletableFuture<Acknowledge> cancel();
  CompletableFuture<savepoint path> stop(savepointDir, advanceToEndOfEventTime)

  CompletableFuture<savepoint path> triggerSavepoint(savepointDir);
  CompletableFuture<Map<acc name, acc value>> getAccumulators(classLoader);
}
```

These interfaces come from current interfaces of ClusterClient. JobClient itself is extensible for further requirement. An example is we can easily expose REST API get job details with a method

```
CompletableFuture<JobDetails> getJobDetails();
```

Retrieval

There are two ways to retrieval a JobClient.

1. compose job submission future returned by ClusterClient, encapsulate ClusterClient with JobID
2. retrieved from a configuration object. Specifically, building ClusterDescriptor, retrieving ClusterClient, encapsulated to JobClient with job id.

The former is used when submit job, while the latter is used when perform job management operations in Flink manage platform(instead of from within user program).

Proposed Changes

ClusterClient

All synchronous job management operations would be replaced with their asynchronous version. Specifically, operations below would be replaced.

```
void cancel(jobId);
Map<acc name, acc value> getAccumulators(jobID, classLoader);
String stopWithSavepoint(jobId, advanceToEndOfEventTime, savepointDir)
```

Detached Mode

Since all operations are asynchronous now, detached mode switch is meaningless. Detached mode inside ClusterClient will be removed.

JobStatus & JobDetails

As proposed by Aljoscha, it's better to move these classes to flink-core as common classes, or provide their user-facing variants.

Executors(FLIP-73)

Executors introduced by FLIP-73 will include a method Executor#execute return a JobClient.

Compatibility, Deprecation, and Migration Plan

- Users previously programming directly against ClusterClient should adjust to changes of ClusterClient. However, since ClusterClient is an internal interface, it isn't regarded as compatibility issue.

Test Plan

- Port job management part of existing tests to using JobClient API in order to ensure that JobClient API works as expect.
- No other dedicated system tests needed.

Rejected Alternatives

JobClient#cancelWithSavepoint

We don't include this method in JobClient because this function is deprecated from REST API. Note that it has nothing to do with current support, users can still use the function as they usually do, but not via JobClient.

Future Works

Expose JobClient

Recur the second scenario of retrieval, for example, said we want to trigger savepoint from command line, JobClient should be generated from command line arguments.

Based on current codebase, we achieve this by

1. build a CustomCommandLine
2. call CustomCommandLine#createClusterDescriptor
3. call ClusterDescriptor#retrieve: ClusterClient
4. construct JobClient from ClusterClient and JobID(parsed from args)

Because CustomCommandLine and ClusterDescriptor are internal concepts, there is no public interface that downstream project developers can program with.

Since this FLIP is mainly aimed at introduce the interface JobClient, it is future works about alternative ways of exposing the JobClient. Candidates includes

1. multi-layered clients https://lists.apache.org/x/thread.html/240582148eda905a772d59b2424cb38fa16ab993647824d178cacb02@%3Cdev.flink.apache.org%3E