PDP-3: Support for Flink

~~PNDA-4005~~ - Integrate Flink as event streaming solution DONE

Motivation

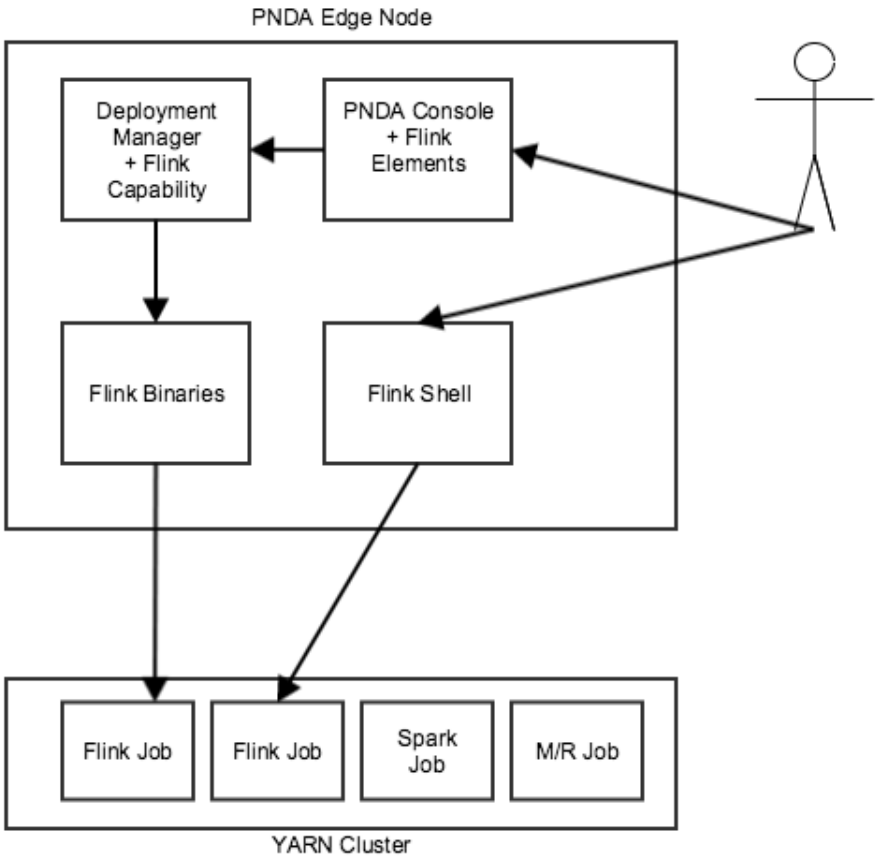
- Apache Flink® is an open-source stream processing framework for distributed, high-performing, always-available, and accurate data streaming applications.
- Flink allows for stream processing at event level granularity and/or very low latency with more advanced windowing and other stream processing features that Spark doesn't currently offer.
- Users of PNDA have requested support for Flink in addition to Spark to implement use cases that are more suited to the features provided by Flink.

Proposal

Flink will be introduced to PNDA as an alternative engine for data processing.

At a high level this involves:

- Deploying the Flink binaries out-of-the-box on a PNDA cluster.
- Supporting Flink components in the PNDA Application Deployment Manager.
- Making Flink visible in the PNDA Console, with links to user interfaces, health information and documentation in the same way as other components of the PNDA system.
- Flink applications will follow the same model as Spark Streaming applications with each job running on YARN as an independent single-use Flink session.
- Prototyping and ad-hoc work on Flink programs will be supported via the Flink CLI shell because support for Flink in Jupyter is currently not suitable for use.



Design

The following section discusses the changes required to each PNDA component.

Flink YARN Mode

Flink jobs will be submitted as single jobs with no need to have a Flink cluster. More details of Flink YARN integration are here

PNDA Mirror

Flink resources and any other dependencies will be hosted on the PNDA mirror. The mirror build script will need to include these in the appropriate mirror section.

Flink Components in PNDA

A Flink history server will be run on one of the PNDA manager nodes.

Flink client libraries will be installed on all nodes with a hadoop EDGE role.

Deployment Manager

Support will be added for deploying Flink components in application packages.

A Flink component plugin will be created that will run Flink applications in the same way as the Spark Streaming plugin currently does. A supervisor will be set up on the PNDA edge node that will call the Flink CLI to submit the job.

We will not include support for Flink in the Deployment manager application information service as this is currently under review, and the implementation is likely to change.

Platform tests

A health metric will be generated for Flink "hadoop.flink.health" which will show the health status of the Flink history server.

The presence of a running, healthy History Server will be verified by calling API “/joboverview” on the history server.

Flink application metrics will be gathered as described here and in the post http://mnxfst.tumblr.com/post/136539620407/receiving-metrics-from-apache-flink-applications and associated with the relevent application and component by prefixing with "application.kpi.<applicaton-name>.<component-name>".

Candidate metrics: https://github.com/apache/flink/tree/master/flink-metrics

Console

The PNDA console dashboard page will be modified to include add Flink blocks under both Stream & Batch sections.

The Flink block will link to the history server UI.

The Flink block will link to a help pop-up with include help text and a link to the Flink documentation.

Logging

The Flink history server log will be aggregated by the PNDA logshipper.

Flink application logs will be aggregated by the PNDA logshipper by configuring the logshipper to collect logs from the YARN container folder on each node manager. To match spark streaming, this will gather the stdout and stderr logs and a flink specific log called flink.log that the user can configure their application to write to if they want to.

Platform libraries

A utility library will be provided that offers scala functions for reading the PNDA master dataset into Flink objects/streams.

Any Kafka and Avro libraries required to read/write to Kafka from Flink will be supplied as part of PNDA.

Jupyter

Prototyping and exploration using Flink with Jupyter shows that the support which is there right now is not good enough. We will monitor this and include support when it is fit-for-purpose.

More details and open issues of Jupyter integration with Flink are here

Example applications

An example application that demonstrates the benfits of Flink over Spark in that case - e.g. Stateful continuous processing (non-batch mode).

Candidates: https://flink.apache.org/usecases.html

PNDA Guide

Sections of guide will need creating or updating to reference Flink

Plan

Phase 1

- Add Flink software to PNDA build and mirror processes
- Deploy & configure Flink and start Flink History server service
- Support applications through pyflink, scala-shell
- Handle mapping of users to queues to give functional parity with Spark
- Add basic platform test & console elements to represent Flink in PNDA
- Logging

Phase 2

- Deployment Manager
    - Packaging and deploying basic Flink applications in similar way to Spark
        - Supervised Flink stream processing jobs
        - Scheduled Flink jobs via Oozie
- Application status monitoring
    - display the flink job status similar to spark (Component, Application type, Id & Status) and provide a link to flink dashboard for more information.
- Documentation
- Data management & cluster houskeeping
    - Clean-up completed-jobs
    - Clean-up temporary streaming directory
    - Rolling file mechanism for growing log files

Phase 3

- Interactive Flink based data exploration in Jupyter
- Plaform libraries support for common operations
- Savepoints
    - Triggering, Listing & Resuming & Disposing savepoints from Deployment Manager for a given application.
- Example applications
    - Illustrate how to build and deploy Flink applications with PNDA (in Java, Scala & Python)
        - executing batch job similar to spark-batch-python
        - executing streaming job similar to spark-streaming-python
    - Illustrate how to use Flink Accumulators, application metrics
    - Use stateful continuous processing (non-batch mode) making it a different case from Spark streaming examples
- Metrics
    - configure Graphite as a default metrics reporter in flink.

TBD

- Operations (restart of nodes, restart of services, losing a node, scale up/down)
- Metrics export

Interfaces

- DM Interface to support execution of flink applications along with the ability to override default values
- New interfaces to be documented for creating & deploying flink applications and Flink-yarn-session/Flink-cluster

Compatibility

- Flink applications and spark applications can co-exist

Alternatives

Expose Flink as a runner to Apache Beam, and move to Apache Beam as primary SDK for application development. This is something under review and could be an incremental step forward from basic Flink support (it doesn't need to be an either/or).