[Design Doc] [YARN-8283] MaWo - A Master Worker framework on top of YARN Services

Yesha Vora, Xuan Gong, Sapin Amin with inputs from Wangda Tan, Vinod Kumar Vavilapalli

Introduction

There is a need for an application / framework to handle Master-Worker scenarios. There are existing frameworks on YARN which can be used to run a job in distributed manner such as Mapreduce, Tez, Spark etc. But master-worker use-cases usually are force-fed into one of these existing frameworks which have been designed primarily around data-parallelism instead of generic Master Worker type of computations.

In this JIRA, we'd like to introduce MaWo - a YARN Service based framework that achieves this goal. The overall goal is to create an app that can take an input job specification with tasks, their expected durations and have a Master dish the tasks off to a predetermined set of workers. The components will be responsible for making sure that the tasks and the overall job finish in specific time durations.

We have been using a version of the MaWo framework for running unit tests of Hadoop in a parallel manner on an existing Hadoop YARN cluster. What typically takes 10 hours to run all of Hadoop project's unit-tests can finish under 20 minutes on a MaWo app of about 50 containers!

YARN-3307 was an original attempt at this but through a first-class YARN app. In this JIRA, we instead build it on top of YARN Service so that our code can focus on the core Master Worker paradigm and let YARN Service take care of the orchestration details.

MaWo Application Components

MaWo application is a YARN service application. This application has mainly two component types - Master and Worker.

The Master component is responsible for distributing tasks to workers and monitoring the lifecycle of the tasks. This includes

- Assigning Tasks to Workers
- Monitor the lifecycle of Tasks
- Log the task status in a log file

The Worker Component is responsible for performing the actual work in the form of tasks.

- Periodically sends heartbeats to the Master
- Performs the assigned task

Job description file

The Job description file is a JSON file consisting of definitions of a setup task, the list of tasks and a teardown task. This file is used as input to the MaWo application and read by the Master component.

Here, a Task can be defined by 3 entities.

- Taskcmd:
    - Taskcmd is a command line which need to be executed as a task
- TaskEnv:

    - TaskEnv is a list of environment variables which need to be set while executing task
- TaskTimeout:
    - TaskTimeout is a timeout defined in seconds. If the task can not finish within defined timeout, it will be aborted and marked as failed.

Setup Task

Setup Task is a type of Task which gets executed on each worker on registering with the Master. This task is defined to perform any setup required by Tasks. Think of this as a “Job Setup but per each worker”.

Tasks

Tasks is a set of Task objects.

Teardown Task

Teardown Task is a type of Task which gets assigned to all workers after all the task execution is finished. This task can be defined to perform cleanup, log collection etc.

Example Job description file

Each Task can be defined with TaskCmd, TaskEnv, TaskTimeout. A simple task can be defined as below.

```
TaskCmd = “printenv”
TaskEnv = {“TEST_VAR”:”test_var”}
TaskTimeout = 10
```

Below is the full and working JSON file for executing sample job in MaWo framework.

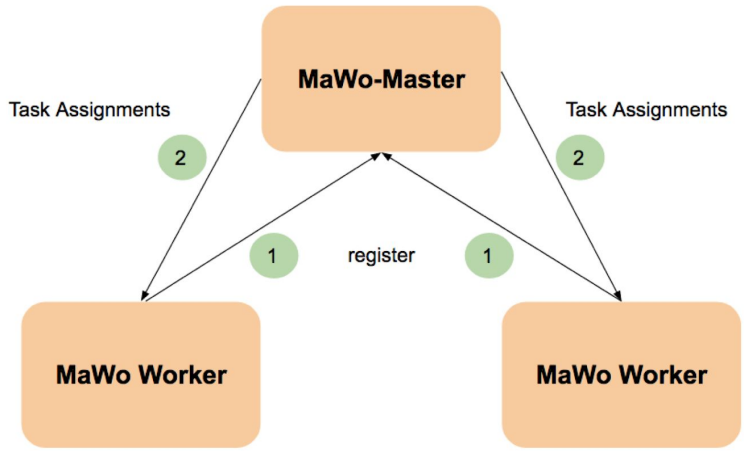
Filename= sample-job-description.json

```
{
        "SetupTask": {
                "TaskCmd": "echo Running SetupTask on this Worker",
            "TaskTimeout": 30
        },
```

```
        "Tasks": [{
                "TaskCmd": "printenv",
                "TaskEnv": {
                        "TEST_VAR": "test_var"
                },
                "TaskTimeout": 30
        }],
        "TeardownTask": {
                "TaskCmd": "echo Running Teardown Task on this Worker",
           "TaskTimeout": 30
        }
}
```

MaWo application lifecycle

MaWo application can have one instance of Master and n instance of Workers.



1) MaWo application is launched with job description file.
2) When Master instance is launched, Master component reads job description file and initializes setup task, teardown task and list of tasks.
3) When Worker instance is launched, It sends heartbeat to Master to register itself.
4) On registering, Master sends Setup task to the worker.
5) Worker keeps sending heartbeat to Master at regular interval.
6) Master keeps the list of Tasks. When worker finishes setup task, It assigns a task to worker.

7) Master also keep track of the status and logs the taskstatus in a log file.
8) Workers perform the task.
9) Step7 & Step8 repeats until all tasks are finished
10) On completion of all tasks, Master sends Teardown task to all workers.
11) On completion of teardown task on all nodes, Application exists.

Relation to existing frameworks

There are a variety of existing frameworks on YARN which can be used to run a job in distributed manner such as Mapreduce, Spark etc. However, using these frameworks to handle the special needs of master worker based scenario such as setup/teardown task, different task env for different tasks, making sure that the tasks are distributed equally enough & meet the deadline is cumbersome.

Proposal for source-code layout

MaWo Application can be put inside hadoop-yarn-project/hadoop-yarn/hadoop-yarn-applications/hadoop-yarn-applications-mawo dir along with other applications that we already have.

```
HW10278:hadoop yvora$ ls -l hadoop-yarn-project/hadoop-yarn/hadoop-yarn-applications
total 16
drwxr-xr-x  5 yvora  staff   170 Apr 25 16:53 hadoop-yarn-applications-distributedshell
drwxr-xr-x  5 yvora  staff   170 Apr 25 16:53
hadoop-yarn-applications-unmanaged-am-launcher
-rw-r--r--  1 yvora  staff   574 Oct 13  2017 hadoop-yarn-applications.iml
drwxr-xr-x  7 yvora  staff   238 Apr 26 11:11 hadoop-yarn-services
drwxr-xr-x  5 yvora  staff   170 Apr 25 16:54 hadoop-yarn-services-api
```

Design considerations

Why a YARN Service ?

MaWo Application should be able to run a variety of user specified jobs. Thus, It's important to give significant control to the users.

- Ability to pass job description file to MaWo application
- Ability to shrink/expand worker nodes as per user requirement
- Ability to define different resource type for Master and Worker component

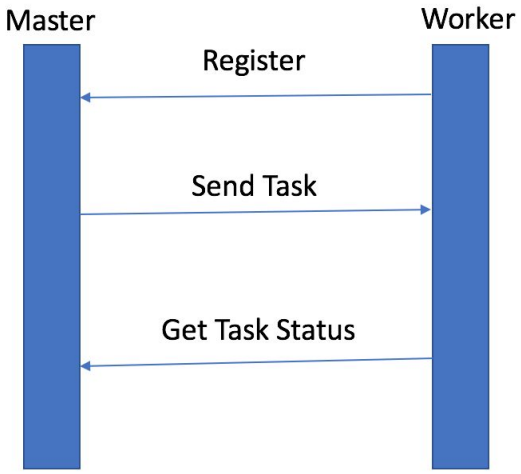
- Ability to use docker images to run the workload
- Fault tolerance for master and worker nodes

Given the above, YARN service seems like a perfect fit for Master Worker application.

Hadoop RPC for communication

Master and Worker components need to communicate Tasks and Task status in order to execute a job.

- Worker node should be able to register itself with Master
- Master should be able to send new task to worker
- Worker should be able to communicate current status of the task



The calls between Master and Worker can be best explained as Verbs such as SendTask, GetTaskStaus etc. Thus, RPC protocol is picked to communicate between master and worker component.