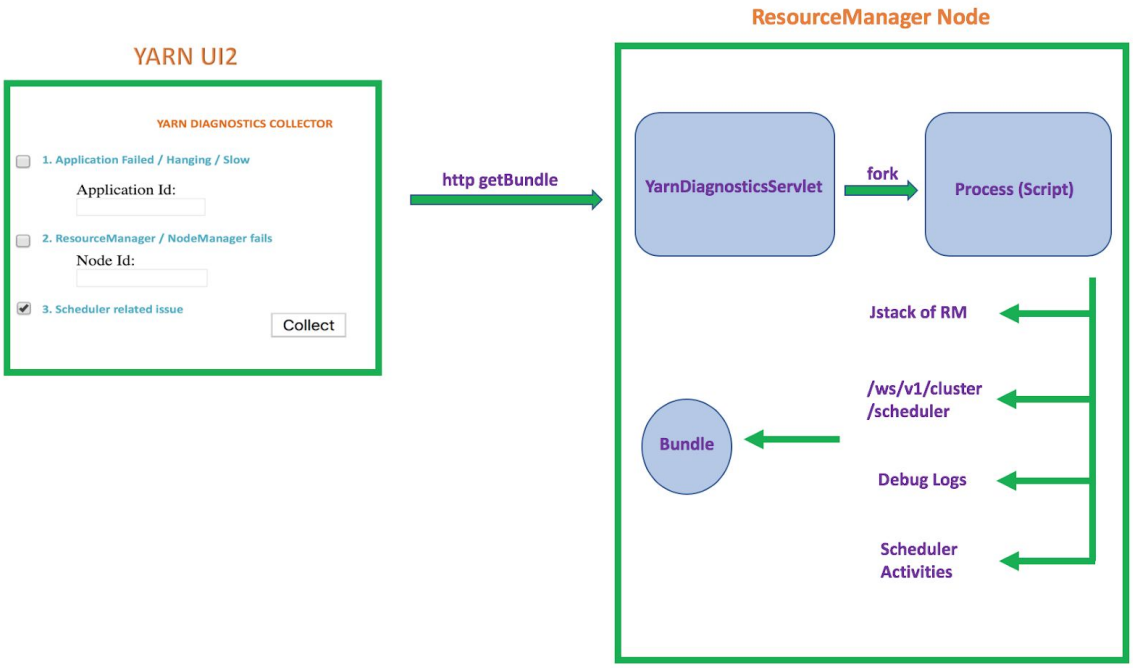
YARN-10323 Design of Diagnostics Collector

1. Overview

There is often a back and forth collection of diagnostics between Customer and Support / Engineering. We also fail to collect the right set of diagnostics for an issue at the very first time. This affects both Mean Time To Resolve (MTTR) issues and Customer Satisfaction. This can be improved by having a Diagnostics Collector Tool in YARN which lists common sets of issues. Customers can collect the diagnostics bundle from this tool when there is an issue. This bundle is shared to Support when creating a Support Case. This tool should capture all possible diagnostics for an issue so that we have complete diagnostics at the very first time.

2. Design Proposal



YARN UI2 will have a seperate tab / page which lists a set of common issues. This is served by YarnDiagnosticsServlet running inside ResourceManager Daemon. The servlet code allows

only a single http thread at a time and returns immediately for other threads saying “Diagnostics Collection in Progress”. This will avoid putting load on ResourceManager. The servlet forks a separate process executing a script. The script can be either in shell or python. The script will collect all the diagnostics and save it in a bundle. Once done, the servlet will send the bundle to the UI2.

Below are the few set of diagnostics which tool collects for various issues:

1. Application Hanging:
    - Application Logs
    - Find the Hanging Container and get Multiple Jstack.
    - ResourceManager logs during job duration.
    - NodeManager logs from NodeManager where hanging containers of jobs run during the duration of containers.
    - Job Configuration from MapReduce HistoryServer, Spark HistoryServer, Tez History URL.

2. Application Failed:
    - Application Logs
    - ResourceManager logs during job duration.
    - NodeManager logs from NodeManager where failed containers of jobs run during the duration of containers.
    - Job Configuration from MapReduce HistoryServer, Spark HistoryServer, Tez History URL.
    - Job Related Metrics like Container, Attempts.

3. Scheduler Related Issue:
    - ResourceManager Scheduler Logs with DEBUG enabled for 2 minutes.
    - Multiple Jstack of ResourceManager
    - YARN and Scheduler Configuration
    - Cluster Scheduler API /ws/v1/cluster/scheduler and Cluster Nodes API /ws/v1/cluster/nodes response
    - Scheduler Activities /ws/v1/cluster/scheduler/bulkactivities response (YARN-10319)

4. ResourceManager / NodeManager Daemon Failure to start:
    - ResourceManager and NodeManager out and log file
    - YARN and Scheduler Configuration

Any new diagnostics which we find useful later can be added into the script. Script allows customers / support engineers to add new diagnostics easily.

3. Implementation Details

ResourceManager adds a new servlet YarnDiagnosticsCollector with path /diagnostics/*. This Servlet handles only one jetty thread and returns a wait message for other parallel threads. This can be done using an AtomicBoolean. The timeout of the thread is configurable.

A Script which provides the list of Common Issues like 1. Application Failed, 2. Application Hanging and so on. During Class Load, YarnDiagnosticsCollector reads the list of common issues from the script and keeps it in memory. On every startup of YARN Ui2 diagnostics page, it fetches the list from the servlet and displays them.

YarnDiagnosticsCollector checks if the script is modified when there is a request from YARN UI2, if so it reads again and refreshes the list of common issues in memory before serving YARN UI2. This way, Users can easily plug new categories without any UI2 or Servlet code change.

Users can select one of the categories from UI2 and request for collection of diagnostics. This will request a YarnDiagnosticsCollector with the category number. The servlet executes the script using ShellCommandExecutor with the proper UGIs. The script collects all the diagnostics required for that issue category and saves it into a configurable directory as a compressed tar file.

Once done, YarnDiagnosticsCollector will stream the file to UI2 if the file size is lesser than configured threshold size. If not, will share the tmp path where the bundle is stored.

Below is the sample script which provides list of common issues and collect list of diagnostics for each category.

```
if [$1 = "listcommonissues"]
    echo "1, Application Failed"
    echo "2, Application Hanging"
    echo "3, Scheduler Related Issue"
    echo "4, RM failure to start"
    echo "5, NM failure to start"
elif [$1 = "collect"]
    if [$2 == 1]
      appId = $3
      mkdir /tmp/$appId
      yarn logs -applicationId $appId > /tmp/$appId/joblogs
```

```
    curl <JHS>/{appId}/conf > /tmp/$appId/conf
    curl <RM>/logs | grep container > /tmp/$appId/rmlogs
    curl <NM>/logs | grep container > /tmp/$appId/nmlogs
    outputpath = /tmp/$appId
  elif
    ...
  elif
    ...
  fi
 tar and compress outputpath.
```

To get the list of common issues from the script:

```
diagnostics_collector.sh listcommonissues

1, Application Failed
2, Application Hanging
3, Scheduler Related Issue
4, RM failure to start
5, NM failure to start
```

To collect diagnostics for category 1 - Application Failed

```
diagnostics_collector.sh collect --application-failed
```