Building and Running HiBench on AArch64 Platform

- Introduction
- Build
- Run Benchmark
  - Prerequisites
  - Hadoop
    - Setup
    - Configure HiBench
    - Run Workload
    - View Report
    - Tuning Benchmark
  - Spark
    - Setup
    - Configure HiBench
      - Configure Hadoop
      - Configure Spark
    - Run Workload
    - View Report
    - Tuning Benchmark
- References

Introduction

HiBench is an open sourced and Apache licensed big data benchmark suite that helps evaluate different big data frameworks in terms of speed, throughput and system resource utilizations.

It contains a set of Hadoop, Spark and streaming workloads, including Sort, WordCount, TeraSort, PageRank, Bayes, Kmeans, enhanced DFSIO, etc. It also contains several streaming workloads for Spark Streaming, Storm and Samza.

Build

NOTE: Following steps are tested on Ubuntu-16.04.

Prerequisites

```
apt install -y maven
```

Build

```
# get source of latest release
git clone https://github.com/intel-hadoop/HiBench -b HiBench-7.0
cd HiBench
# build all modules in HiBench
mvn -Dspark=2.2 -Dscala=2.11 clean package
# if you just want to build for hadoop and spark
mvn -Phadoopbench -Psparkbench -Dspark=2.2 -Dscala=2.11 clean package
```

Run Benchmark

Prerequisites

```
apt install -y bc python2.7 python-setuptools openssh-server
service start ssh
```
Hadoop

Setup

- A working hadoop cluster with HDFS, and YARN
  - To setup pseudo-distributed cluster, pls refer this link (Hadoop 2.x) or this link (Hadoop 3.0).
  - To setup multi-node cluster, please refer this link (Hadoop 2.x) or this link (Hadoop 3.0)
- Start up SSH service

You may encounter two problems:

1. Passphraseless ssh
   Hadoop requires a certain account to login to nodes without passphrase. This account should be setup in each node. To setup this account, please refer following cmds.
   
   ```
   mkdir -p ~/.ssh
   rm -f ~/.ssh/id_rsa*
   # scan and save target fingerprints
   ssh-keyscan -t ecdsa-sha2-nistp256 -H ${HOSTNAME} > ~/.ssh/known_hosts
   ssh-keyscan -t ecdsa-sha2-nistp256 -H localhost >> ~/.ssh/known_hosts
   ssh-keyscan -t ecdsa-sha2-nistp256 -H 0.0.0.0 >> ~/.ssh/known_hosts
   # generate key
   ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
   cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
   chmod 0600 ~/.ssh/authorized_keys
   ```

2. Hadoop user privilege
   It is recommended to run hadoop services as a non-root user. Usually a user, hdfs, is created to run HDFS and YARN services. If "root" is a must option, following cmds are required
   
   ```
   USER=$(whoami)
   export HDFS_NAMENODE_USER=${USER}
   export HDFS_DATANODE_USER=${USER}
   export HDFS_SECONDARYNAMENODE_USER=${USER}
   export YARN_RESOURCEMANAGER_USER=${USER}
   export YARN_NODEMANAGER_USER=${USER}
   ```

Configure HiBench

HiBench requires Hadoop configuration info to setup and run test workloads. The default configuration is `<HIBENCH_ROOT_DIR>/conf/hadoop.conf`. A template configuration file can be used as start point.

````
cp conf/hadoop.conf.template conf/hadoop.conf
```'

Usually these two fields should be modified to match Hadoop settings:

- **hibench.hadoop.home**: point to hadoop root directory
- **hibench.hdfs.master**: point to hdfs service uri. This uri can be found at `<HADOOP_ROOT_DIR>/etc/hadoop/core-site.xml:fs.defaultFS`.

A detail description for fields in hadoop.conf are listed as following:

<table>
<thead>
<tr>
<th>Property</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>hibench.hadoop.home</td>
<td>The Hadoop installation location</td>
</tr>
<tr>
<td>hibench.hadoop.executable</td>
<td>The path of hadoop executable. For Apache Hadoop, it is/YOUR/HADOOP/HOME/bin/hadoop</td>
</tr>
<tr>
<td>hibench.hadoop.configure.dir</td>
<td>Hadoop configuration directory. For Apache Hadoop, it is/YOUR/HADOOP/HOME/etc/hadoop</td>
</tr>
<tr>
<td>hibench.hdfs.master</td>
<td>The root HDFS path to store HiBench data, i.e. hdfs://localhost:8020/user/username</td>
</tr>
<tr>
<td>hibench.hadoop.release</td>
<td>Hadoop release provider. Supported value: apache, cdh5, hdp</td>
</tr>
</tbody>
</table>

Run Workload

HiBench's workload usually have two parts: `prepare` and `run`. For example, to run "wordcount",
The prepare.sh launches a Hadoop job to generate the input data on HDFS. The run.sh submits a Hadoop job to the cluster.

**View Report**

When benchmark is done, the report is outputted to `<HIBENCH_ROOT_DIR>/report/hibench.report`. It is a summarized workload report, including workload name, execution duration, data size, throughput per cluster, throughput per node.

The report directory also includes further information for debugging and tuning.

- `<workload>/hadoop/monitor.html`: System utilization monitor results.
- `<workload>/hadoop/conf/<workload>.conf`: Generated environment variable configurations for this workload.

**Tuning Benchmark**

- change input data size:
  - set `hibench.scale.profile` in `conf/hibench.conf`. Available values are tiny, small, large, huge, gigantic and bigdata.
- change parallelism
  - Change the below properties in `conf/hibench.conf` to control the parallelism.

<table>
<thead>
<tr>
<th>Property</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>hibench.default.map.parallelism</td>
<td>Mapper number in hadoop</td>
</tr>
<tr>
<td>hibench.default.shuffle.parallelism</td>
<td>Reducer number in hadoop</td>
</tr>
</tbody>
</table>

**Spark**

**Setup**

- A working HDFS service
- A working YARN service, if Spark is tested in YARN mode
- Working Spark: Spark can be configured to work in either "standalone mode" or "YARN mode". ("Mesos mode" is not counted in as Mesos is not deployed when we run HiBench)
  - Standalone mode: it is the easiest to set up and will provide almost all the same features as the "YARN mode" if only Spark is running.
  - YARN mode:
- Start SSH service

**Configure HiBench**

**Configure Hadoop**

Hadoop is used to generate the input data of the workloads. Create and edit `conf/hadoop.conf`

```
cp conf/hadoop.conf.template conf/hadoop.conf
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>hibench.hadoop.home</td>
<td>The Hadoop installation location</td>
</tr>
<tr>
<td>hibench.hadoop.executable</td>
<td>The path of hadoop executable. For Apache Hadoop, it is /YOUR/HADOOP/HOME/bin/hadoop</td>
</tr>
<tr>
<td>hibench.hadoop.configure.dir</td>
<td>Hadoop configuration directory. For Apache Hadoop, it is /YOUR/HADOOP/HOME/etc/hadoop</td>
</tr>
<tr>
<td>hibench.hdfs.master</td>
<td>The root HDFS path to store HiBench data, i.e. hdfs://localhost:8020/user/username</td>
</tr>
<tr>
<td>hibench.hadoop.release</td>
<td>Hadoop release provider. Supported value: apache, cdh5, hdp</td>
</tr>
</tbody>
</table>

**Configure Spark**

Create and edit `conf/spark.conf`
Set the below properties properly:

- hibench.spark.home: The Spark installation location
- hibench.spark.master: The Spark master, i.e. `spark://xxx:7077`, `yarn-client`

**Run Workload**

HiBench's workload usually have two parts: *prepare* and *run*. For example, to run "wordcount",

```bash
bin/workloads/micro/wordcount/prepare/prepare.sh
bin/workloads/micro/wordcount/spark/run.sh
```

The prepare.sh launches a Hadoop job to generate the input data on HDFS. The run.sh submits a Spark job to the cluster.

**View Report**

Same as "Hadoop benchmark", the report is outputed to `<HIBENCH_ROOT_DIR>/report/hibench.report`.

Meanwhile, detail information is generated for debugging and tuning:

- `<workload>/spark/bench.log`: Raw logs on client side.
- `<workload>/spark/monitor.html`: System utilization monitor results.
- `<workload>/spark/conf/<workload>.conf`: Generated environment variable configurations for this workload.
- `<workload>/spark/conf/sparkbench/<workload>/sparkbench.conf`: Generated configuration for this workloads, which is used for mapping to environment variable.
- `<workload>/spark/conf/sparkbench/<workload>/spark.conf`: Generated configuration for spark.

**Tuning Benchmark**

- change input data size:
  - set `hibench.scale.profile` in `conf/hibench.conf`. Available values are tiny, small, large, huge, gigantic and bigdata.
- change parallelism

<table>
<thead>
<tr>
<th>Property</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>hibench.default.map.parallelism</td>
<td>Partition number in Spark</td>
</tr>
<tr>
<td>hibench.default.shuffle.parallelism</td>
<td>Shuffle partition number in Spark</td>
</tr>
</tbody>
</table>

- change Spark job properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>hibench.yarn.executor.num</td>
<td>Spark executor number in Yarn mode</td>
</tr>
<tr>
<td>hibench.yarn.executor.cores</td>
<td>Spark executor cores in Yarn mode</td>
</tr>
<tr>
<td>spark.executor.memory</td>
<td>Spark executor memory</td>
</tr>
<tr>
<td>spark.driver.memory</td>
<td>Spark driver memory</td>
</tr>
</tbody>
</table>

**References**

Sample log: **Hadoop terasort**