Hibench 大数据安装测试手册

Hibench 作为一个测试 hadoop 的基准测试框架，提供了对于 hive：（aggregation，scan，join），排序（sort，TeraSort），大数据基本算法（wordcount，pagerank，nutchindex），机器学习算法（kmeans，bayes），集群调度（sleep），吞吐（dfsio），以及新加入 5.0 版本的流测试，是一个测试大数据平台非常好用的工具
它支持的框架有：hadoopbench、sparkbench、stormbench、flinkbench、gearpumpbench。
HiBench 是一套基准测试套件，用于帮助我们评估不同的大数据框架性能指标（包括处理速度、吞吐等）的负载指标，可以评估 Hadoop、Spark 和流式负载等，
其中Hadoop bench 具体的工作负载有：

- Sort
- WordCount
- TeraSort
- Sleep
- SQL
- PageRank
- Nutch indexing
- Bayes
- Kmeans
- NWeight
- enhanced DFSIO
- 等等

工作负载

对这些工作负载进行分类记录如下，总体可以分为 6 大类：micro、ml（ machine learning）、sql、graph、websearch 和 streaming。
Micro 基准

- Sort

使用RandomTextWriter 生成测试数据，Sort 工作负载对它的文本输入数据进行排序来进行基准测试

- WordCount

使用 RandomTextWriter 生成测试数据，WordCount 工作负载对输入数据中每个单词的出现情况进行统计

- TeraSort

是由 Jim Gray 创建的标准基准。其输入数据由 Hadoop TeraGen 示例程序生成。

- Sleep

使每个任务休眠一定的时间来测试调度框架

- enhanced DFSIO (dfsioe)

  增强的 DFSIO 通过生成大量执行写入和读取的任务来测试 Hadoop 集群的 HDFS 吞吐量。它测量每个map 任务的平均 I/O 速率、每个 map 任务的平均吞吐量以及 HDFS 集群的聚合吞吐量。

  注:这个工作负载不支持 Spark。

Machine Learning 基准

- 贝叶斯分类 (Bayes)

  朴素贝叶斯是一种简单的多类分类算法，具有独立于每一对特征的假设。这个工作负载是在 spark.mllib 中实现并使用自动生成的文档，这些文档的单词遵循 zipfian 分布。关键字用于文本生成/usr/share/dict/linux.words.ords 也从默认的 linux 文件。

- k-means 聚类(Kmeans)

  这个工作负载测试是在spark.mllib 中实现的K-means(一种著名的知识发现和数据挖掘的聚类算法)。输入数据集是由基于均匀分布和 Guassian 分布的 GenKMeansDataset 生成的。

- 逻辑回归(LR)

  逻辑回归(Logistic Regression, LR)是预测分类响应的常用方法。这个工作负载是在 spark.mllib 中实现， LBFGS 优化器和输入数据集是 LogisticRegressionDataGenerator 基于随机生成决策树的平衡。它包含三种不同类型的数据类型，包括分类数据、连续数据和二进制数据。

- 交替最小二乘(ALS)

  交互最小二乘法(ALS)算法是一种著名的协同过滤算法。这个工作负载是在 spark.mllib 中实现和输入数据集是由 RatingDataGenerator 为产品推荐系统生成的。

- 梯度增加树(GBT)

  梯度增强树(GBT)是一种使用决策树组合的流行的回归方法。这个工作负载是在 spark.mllib 中实现， GradientBoostingTreeDataGenerator 生成的输入数据集。

- 线性回归(线性)

  线性回归(线性回归)是一个在 spark.mllib 中实现的工作负载。mllib SGD 优化器。输入数据集是由 LinearRegressionDataGenerator 生成的。

- 潜在狄利克雷分配(LDA)

  潜在的 Dirichlet 分配(LDA)是一个主题模型，它从一个文本文档集合中推断主题。这个工作负载是在 spark.mllib 中实现和输入数据集由 LDADataGenerator 生成。

- 主成分分析(PCA)

  主成分分析(PCA)是一种寻找旋转的统计方法，使得第一个坐标有最大的方差，而每个后续的坐标都有最大的方差。PCA 在降维方面得到了广泛的应用。这个工作负载是在 spark.mllib 中实现。输入数据集由PCADataGenerator 生成。

- 随机森林(RF)

  随机森林(RF)是决策树的集合。随机森林是最成功的分类和回归机器学习模型之一。为了降低过度拟合的风险，他们联合了许多决策树。这个工作负载是在 spark.mllib 中实现，RandomForestDataGenerator 生成的输入数据集。

- 支持向量机(SVM)

  支持向量机(SVM)是大规模分类任务的标准方法。这个工作负载是在 spark.mllib 中实现和输入数据集由SVMDataGenerator 生成。

- 奇异值分解(SVD)

  奇异值分解(SVD)将矩阵分解成三个矩阵。这个工作负载是在 spark.mllib 中实现及其输入数据集由 SVDDataGenerator 生成。

SQL 基准

- 扫描(扫scan),连接(join),聚合(aggregation)

  这些工作负载是基于 SIGMOD 09 论文“对大规模数据分析方法的比较”和 HIVE-396 进行开发的。它包含用于执行本文描述的典型 OLAP 查询的Hive 查询(聚合和连接)。它的输入也会自动生成带有超链接的网络数据。

Websearch 基准

- PageRank

  这个工作负载基准PageRank 算法在 Spark-MLLib/Hadoop 中实现(在 pegasus 2.0 中包含一个搜索引擎排名基准)。数据源是由Web 数据生成的，其超链接遵循 Zipfian 分布。

- Nutch 索引(nutchindexing)

  大规模搜索索引是 MapReduce 最重要的用途之一。这个工作负载测试 Nutch 中的索引子系统，这是一个流行的开源(Apache 项目)搜索引擎。工作负载使用自动生成的 Web 数据，其超链接和单词都遵循 Zipfian 分布和相应的参数。用来生成网页文本的命令是默认的 linux 命令文件。

Graph 基准

- NWeight(NWeight)
  NWeight 是由Spark GraphX 和 pregel 实现的一种迭代的图形并行算法。该算法计算两个 n-hop 的顶点之间的关联。

Streaming 基准

- 身份(Identity)
  这个工作负载从Kafka 读取输入数据，然后立即将结果写入Kafka，不涉及复杂的业务逻辑。
- 重新分区(Repartition)
  这个工作负载从 Kafka 读取输入数据，并通过创建更多或更少的分区来更改并行度。它测试了流框架中的数据洗牌效率。
- 有状态Wordcount(Wordcount)
  每隔几秒就会收到Kafka 的词汇量。这将测试流框架中的有状态操作符性能和检查点/Acker 成本。
- Fixwindow(Fixwindow)
  工作负载执行基于窗口的聚合。它在流框架中测试窗口操作的性能

Hibench 包含几个 hadoop 的负载

- micro benchmarks

Sort:使用 hadoop randomtextwriter 生成数据，并对数据进行排序。
Wordcount:统计输入数据中每个单词的出现次数，输入数据使用 hadoop randomtextwriter 生成。
TeraSort：输入数据由 hadoop teragen 产生，通过 key 值进行排序。

- hdfs benchmarks

增强行的 dfsio：通过产生大量同时执行读写请求的任务测试 hadoop 机群的 hdfs 吞吐量

- web search bench marks

Nutch indexing:大规模收索引擎，这个是负载测试 nutch（apache 的一个开源搜索引擎）的搜索子系统，使用自动生成的 web 数据，web 数据中的连接和单词符合 zipfian 分布（一个单词出现的次数与它在频率表的排名成反比）

Pagerank:这个负载包含在一种在 hadoop 上的 pagerank 的算法实现，使用自动生成的 web 数据，web 数据中的链接符合 zipfian 分布。(对于任意一个 term 其频度（frequency)的排名（rank）和 frequency 的乘积大致是一个常数）

- machine learning benchmarks

Mahout bayesian classification(bayes):大规模机器学习，这个负载测试 mahout（apache 开源机器学习库）中的 naive bayesian 训练器，输入的数据是自动生成的文档，文档中的单词符合 zipfian 分布。

Mahout k-means clustering(kmeans):测试 mahout 中的 k-means 聚类算法，输入的数据集由基于平均分布和高斯分布的 genkmeansdataset 产生。

- data analytics benchmarks

Hive query benchmarks(hivebench):包含执行的典型olap 查询的hive 查询（aggregation 和 join），使用自动生成的 web 数据，web 数据的链接符合 zipfian 分布。

测试时安装在 hadoop 集群的一台 linux 服务器即可

下载 Hibench

Github 地址：https://github.com/intel-hadoop/HiBench

注意事项：1、Python 2.x(>=2.6) is required.

2、bc is required to generate the HiBench report.（如没有 bc 工具，执行 yum install bc）

3、Supported Hadoop version: Apache Hadoop 2.x, CDH5.x, HDP

4、Build HiBench according to build HiBench.

5、Start HDFS, Yarn in the cluster.

3、编译 Hibench

首先执行节点要安装 maven，如没有，需要先安装

这里我只对 Hibench 的 hadoopbench 和 sparkbench 框架进行了编译，也可以对某个需要测试模块进行编译：

可以参考 Github 文档：

https://github.com/intel-hadoop/HiBench/blob/master/docs/build-hibench.md

mvn -Phadoopbench -Psparkbench -Dspark=1.6 -Dscala=2.10 clean package

4、修改配置

进入 conf 目录：

cp hadoop.conf.template hadoop.conf

修改 conf/hadoop.conf 文件：

[root@cdh-agent1 conf]# vi hadoop.conf
Hadoop home
hibench.hadoop.home /opt/cloudera/parcels/CDH/lib/hadoop

The path of hadoop executable
hibench.hadoop.executable ${hibench.hadoop.home}/bin/hadoop

Hadoop configraution directory
hibench.hadoop.configure.dir ${hibench.hadoop.home}/etc/hadoop

The root HDFS path to store HiBench data 这里就是生成测试数据存放的 hdfs 负目录，注意要有写权限
hibench.hdfs.master hdfs://10.x.x.x:8020/user/hibench
parafs://192.168.1.41:4500

Hadoop release provider. Supported value: apache, cdh5, hdp
hibench.hadoop.release cdh5 _->apache

这里以 wordcount 为例，conf/benchmarks.lst 测试项目
conf/frameworks.lst 配置语言，这两个文件可以不作修改即可。
在 conf/hibench.conf 文件中配置了一些测试的相关参数，比如 MR 并行度，报告路径，名称等。
其中 hibench.scale.profile 为对应的生成的数据量大小，他的值对应于 conf/workloads/micro/wordcount.conf 中设置的数值
[root@cdh-agent1 conf]# cat hibench.conf
Data scale profile. Available value is tiny, small, large, huge, gigantic and bigdata.
The definition of these profiles can be found in the workload's conf file i.e. conf/workloads/micro/wordcount.conf
hibench.scale.profile huge
Mapper number in hadoop, partition number in Spark
hibench.default.map.parallelism 8

Reducer nubmer in hadoop, shuffle partition number in Spark
hibench.default.shuffle.parallelism 8
#===
Report files
#===
default report formats
hibench.report.formats "%-12s %-10s %-8s %-20s %-20s %-20s %-20s\n"

```
# default report dir path
hibench.report.dir                      ${hibench.home}/report

# default report file name
hibench.report.name                     hibench.report

# input/output format settings. Available formats: Text, Sequence.
sparkbench.inputformat                  Sequence
sparkbench.outputformat                 Sequence

# hibench config folder
hibench.configure.dir                   ${hibench.home}/conf
```

进入 conf/workloads/micro/,wordcount.conf 文件配置的是生成的数据量大小

```
[root@cdh-agent1 micro]# more wordcount.conf
#datagen
hibench.wordcount.tiny.datasize                 32000
hibench.wordcount.small.datasize                320000000
hibench.wordcount.large.datasize                3200000000
hibench.wordcount.huge.datasize                 32000000000
hibench.wordcount.gigantic.datasize             320000000000
hibench.wordcount.bigdata.datasize              1600000000000

hibench.workload.datasize
${hibench.wordcount.${hibench.scale.profile}.datasize}

# export for shell script
hibench.workload.input                  ${hibench.hdfs.data.dir}/Wordcount/Input
hibench.workload.output                 ${hibench.hdfs.data.dir}/Wordcount/Output
```

执行测试脚本

在 bin/run_all.sh 该脚本为测试所有的测试基准模块(将运行所有在 conf/benchmarks.lst 和 conf/frameworks.lst 中的 workloads)；

这里还是以 wordcount 为例，

①生成测试数据 bin/workloads/micro/wordcount/prepare/prepare.sh

②运行 wordcount 测试例子 bin/workloads/micro/wordcount/hadoop/run.sh

③生成的测试数据在 conf/hadoop.conf 中 hibench.hdfs.master 项配置，我的是在 /user/hibench/HiBench 目录下

查看测试报告

测 试 报 告 位 置 :report/hibench.report

```
[root@cdh-agent1 report]# cat hibench.report
Type                    Date                Time            Input_data_size             Duration(s)
Throughput(bytes/s) Throughput/node
HadoopWordcount  2018-03-07 09:53:15  35891                       43.457              825
825
HadoopWordcount    2018-03-07    10:21:22    3284906140                      283.518
11586234                11586234
HadoopSort          2018-03-07    10:41:19    328492770                       49.502
6635949                 6635949
HadoopJoin          2018-03-07    14:08:00    1919260193                      264.432
7258048                 7258048
```

补充

在 bin/workloads 目录下：对应着不同的大数据测试点，比如 sql、ml、graph、streaming 等

```
[root@cdh-agent1 bin]# cd workloads/
[root@cdh-agent1 workloads]# ll
总用量 24
drwxr-xr-x  3 root root 4096 3 月   6 14:45 graph
drwxr-xr-x  7 root root 4096 3 月   6 14:45 micro
drwxr-xr-x 13 root root 4096 3 月   6 14:45 ml
drwxr-xr-x  5 root root 4096 3 月   6 14:45 sql
drwxr-xr-x  6 root root 4096 3 月   6 14:46 streaming
drwxr-xr-x  4 root root 4096 3 月   6 14:46 websearch
```

进入这些目录后：有对应不同的测试小项目，根据大数据平台需要，对其进行相应的测试。

```
[root@cdh-agent1 workloads]# cd micro/
[root@cdh-agent1 micro]# ll
总用量 20
drwxr-xr-x 4 root root 4096 3 月   6 14:45 dfsioe
drwxr-xr-x 5 root root 4096 3 月   6 14:45 sleep
drwxr-xr-x 5 root root 4096 3 月   6 14:45 sort
drwxr-xr-x 5 root root 4096 3 月   6 14:45 terasort
drwxr-xr-x 5 root root 4096 3 月   6 14:45 wordcount
```

目前，hibench 使用 7.0
修改配置：

- vim conf/hadoop.conf

hibench.hadoop.home：hadoop 安装目录
hibench.hadoop.executable ： 你 的 bin/hadoop 所 在 目 录 ， 一 般 是 {HADOOP_HOME}/bin/hadoop
hibench.hadoop.configure.dir ： hadoop 配 置 文 件 所 在 目 录 ， 一 般 位 于 HADOOP_HOME}/etc/hadoop
hibench.hdfs.master ： hdfs 上 存 储 Hibench 数 据 的 目 录 ， 如 ： hdfs://localhost:8020/user/hibench
hibench.hadoop.release：hadoop 发行版提供者，支持 value：apache, cdh5, hdp

- 修改 HiBench-master/conf/hibench.conf

hibench.scale.profile tiny # 数据量有 tiny, small, large, huge, gigantic and bigdata

hibench.home /home/hadoop/HiBench-master # hibench.home 路径

- 修改 HiBench-master/conf/frameworks.lst

根据需要选择一个或全选 ，前面加 # 代表不选择
hadoop
#spark

- 修改 HiBench-master/bin/workloads/micro/dfsioe/prepare/prepare.sh

在 run_hadoop_job 行的上一行添加下面配置

INPUT_HDFS=hdfs://mycluster/HiBench/Dfsioe/Input # hdfs 的路径，不带端口

- 修改 HiBench-master/bin/workloads/micro/dfsioe/hadoop/run_read.sh

在 SIZE=`dir_size $INPUT_HDFS` 行的上一行添加下面配置

INPUT_HDFS=hdfs://mycluster/HiBench/Dfsioe/Input # hdfs 的路径，不带端口

- 修改 HiBench-master/bin/workloads/micro/dfsioe/hadoop/run_write.sh

在 # pre-running 行的上一行添加下面配置

OUTPUT_HDFS=hdfs://mycluster/HiBench/Dfsioe/Onput # hdfs 的路径，不带端口

INPUT_HDFS=hdfs://mycluster/HiBench/Dfsioe/Input # hdfs 的路径，不带端口

- 目录结构

/root/apps/HiBench-master/conf

-rw-r--r-- 1 root root 245 Dec 12 21:30 benchmarks.lst
-rw-r--r-- 1 root root 332 Aug 13 02:34 flink.conf.template
-rw-r--r-- 1 root root 13 Dec 12 21:30 frameworks.lst
-rw-r--r-- 1 root root 246 Aug 13 02:34 gearpump.conf.template
-rw-r--r-- 1 root root 471 Dec 13 20:30 hadoop.conf
-rw-r--r-- 1 root root 448 Aug 13 02:34 hadoop.conf.template
-rw-r--r-- 1 root root 6600 Dec 18 04:57 hibench.conf
-rw-r--r-- 1 root root 1640 Dec 18 22:42 spark.conf
-rw-r--r-- 1 root root 1655 Aug 13 02:34 spark.conf.template
-rw-r--r-- 1 root root 942 Aug 13 02:34 storm.conf.template
drwxr-xr-x 8 root root 109 Dec 14 02:49 workloads #里面有对应配置产生的数据量配置

举例：micro
[root@gsafety1 conf]# ll /root/apps/HiBench-master/conf/workloads/micro
total 20
-rw-r--r-- 1 root root 1920 Aug 13 02:34 dfsioe.conf
-rwxr-xr-x 1 root root 805 Aug 13 02:34 sleep.conf
-rw-r--r-- 1 root root 657 Aug 13 02:34 sort.conf
-rw-r--r-- 1 root root 571 Dec 18 04:40 terasort.conf
-rwxr-xr-x 1 root root 658 Dec 14 02:38 wordcount.conf

目录结构：

/root/apps/HiBench-master/bin/workloads 下面有：
graph
micro#选这个！！测试
ml
sql
streaming
websearch

测试算法包：
/opt/diyu/hadoop-system/HiBench-7.0/conf/workloads/micro/dfsioe# hdfsio 测试
/opt/diyu/hadoop-system/HiBench-7.0/conf/workloads/micro/sleep
/opt/diyu/hadoop-system/HiBench-7.0/conf/workloads/micro/sort
/opt/diyu/hadoop-system/HiBench-7.0/conf/workloads/micro/wordcount

cd /opt/diyu/hadoop-system/HiBench-7.0/conf/workloads/micro/terasort#排序准备数据启动：
/opt/diyu/hadoop-system/HiBench-7.0/conf/workloads/micro/terasort/prepare/prepare.sh
启动 mr 任务：

/opt/diyu/hadoop-system/HiBench-7.0/conf/workloads/micro/terasort/hadoop/run.sh
启动 spark 任务：
/opt/diyu/hadoop-system/HiBench-7.0/conf/workloads/micro/terasort/spark/run.sh

测试报告：

/root/apps/HiBench-master/report

```
drwxr-xr-x 4 root root     44 Dec 14 03:58 bayes
-rw-r--r-- 1 root root 6651 Dec 19 01:39 hibench.report
drwxr-xr-x 3 root root     28 Dec 13 23:56 sort
drwxr-xr-x 4 root root     44 Dec 13 03:45 terasort
drwxr-xr-x 5 root root     61 Dec 12 04:11 wordcount
```

- 全部测试：
  运行 bin/run-all.sh
  查看结果
  在当前目录会生成 hibench.report 文件，内容如下

Type	Date	Time	Input_data_size	Duration(s)	Throughput(bytes/s)	Throughput/node
WORDCOUNT	2015-05-12 19:32:33	251.248				
DFSIOE-READ	2015-05-12 19:54:29	54004092852		463.863	116422505	38807501
DFSIOE-WRITE	2015-05-12 20:02:57	27320849148		498.132	54846605	18282201
PAGERANK	2015 05-12 20:27:25	711.391				
SORT	2015 05-12 20:33:21	243.603				
TERASORT	2015-05-12 20:40:34	10000000000	00	266.796	37481821	12493940
SLEEP	2015-05-12 20:40:40	0	.177	0	0	

- 单个测试 ，测试的场景都在 HiBench-master/bin/workloads 目录下
举测试 hadoo 的 WordCount 场景为例，其它场景类似。

在 HiBench-master/bin/workloads/micro/wordcount/prepare 目录下执行
./prepare.sh

在 HiBench-master/bin/workloads/micro/wordcount/hadoop 目录下执行
./run.sh