Data Analytic Engine
数据科学引擎 Jarvis

目 录

CONTENT

01. 产品介绍

02. 性能指标

01 产品介绍

Jarvis 数据科学引擎是一个单机大数据、高性能数据分析工具。
满足您在百 G 级别数据量下，使用个人工作环境/工作站，高效地完成数据分析、机器学习的工作。

兼容主流工具的易用接口

同一套数据，提供 SQL like、Pandas like、SKLearn like 三套接口，无学习成本。

性能极致优化的单机计算引擎

通过并行计算，数据索引，核外计算等"黑科技"，实现百 G 数据量级下，相比Spark 5 -10 倍的性能提升。

多种数据源和数据格式支持

支持对接 HDFS，HIVE 等多种数据源，支持 CSV、Parquet、Iceberg 等多种数据格式，支持多数据源的联合分析。

极简安装

pip 一键安装，单机极简部署，无服务化，无需运维。

01 功能介绍-DataFrame & Machine Learning

[49]:
```
%%time

import blackhole.dataframe as pd
from blackhole.ml.ensemble import RandomForestRegressor
df = pd.read_csv("taxi_10g.csv")
df.dropna(inplace=True)
features, label = df.iloc[:, :15], df['fare_amount']
rf = RandomForestRegressor(n_estimators=100)
rf.fit(features, label)
```

```
CPU times: user 8.92 s, sys: 6.15 s, total: 15.1 s
Wall time: 57.2 s
```

[50]:
```
%%time

import pandas as pd
from sklearn.ensemble import RandomForestRegressor
df = pd.read_csv("taxi_10g.csv")
df.dropna(inplace=True)
features, label = df.iloc[:, :15], df['fare_amount']
rf = RandomForestRegressor(n_estimators=100)
rf.fit(features, label)
```

```
CPU times: user 27min 15s, sys: 7.68 s, total: 27min 23s
Wall time: 28min 16s
```

和主流开源工具使用接口兼容，接口覆盖率 93.6%，性能提升明显

01 功能介绍-SQL

```
from blackhole import sql

load_query = """
    create table test_load FROM 'data/demo_data.parquet'
    FORMAT Parquet
    ENGINE = MergeTree()
    PRIMARY KEY(`logId`, `logDate`)
"""
sql(load_query).show()
```

--

```
sql("select count(*) from test_load").show()
```

```
  count()
---------
  5785196
Read 1 rows, 1.00 B in 0.001316791 sec., 759 rows/sec., 759.42 B/sec.
Read 3 rows, 167.00 B in 0.00371202 sec., 808 rows/sec., 43.93 KiB/sec.
Read 1 rows, 4.01 KiB in 0.001681123 sec., 594 rows/sec., 2.33 MiB/sec.
```

- SQL like 语法
- 支持多种格式导入导出

```
[2]: from blackhole import sql

[3]: sql('select count(*) from pl_log_web_new').show()

     count()
  ----------
  1449147666
  Read 1 rows, 1.00 B in 0.003689657 sec., 271 rows/sec., 271.03 B/sec.
  Read 1 rows, 57.00 B in 0.001360881 sec., 734 rows/sec., 40.90 KiB/sec.
  Read 1 rows, 4.01 KiB in 0.001140996 sec., 876 rows/sec., 3.43 MiB/sec.

[4]: sql("select logDate, count(*) from pl_log_web_new group by logDate order by logDate desc").show()

  153 lines read, show first 100 lines

  logDate       count()
  ----------  ---------
  2021-10-31    9397966
  2021-10-30    9570364
  2021-10-29   10884826
  2021-10-28   11289618

  2021-07-26    8367550
  2021-07-25    8151171
  2021-07-24    8119428
  Read 1 rows, 1.00 B in 0.000767674 sec., 1302 rows/sec., 1.27 KiB/sec.
  Read 1 rows, 57.00 B in 0.001581119 sec., 632 rows/sec., 35.21 KiB/sec.
  Read 1449147666 rows, 2.70 GiB in 0.479869685 sec., 3019877502 rows/sec., 5.62 GiB/sec.
```

- 大数据亚秒级查询

```
[2]: from blackhole import sql

[3]: sql('select count(*) from pl_log_web_new').show()

     count()
  ----------
  1449147666
  Read 1 rows, 1.00 B in 0.003924401 sec., 254 rows/sec., 254.82 B/sec.
  Read 1 rows, 57.00 B in 0.001442862 sec., 693 rows/sec., 38.58 KiB/sec.
  Read 1 rows, 4.01 KiB in 0.001219085 sec., 820 rows/sec., 3.21 MiB/sec.

[4]: df = sql("select * from pl_log_web_new where logDate = '2021-09-30'").to_df()

  Read 1 rows, 1.00 B in 0.000854559 sec., 1170 rows/sec., 1.14 KiB/sec.
  Read 1 rows, 57.00 B in 0.001465005 sec., 682 rows/sec., 38.00 KiB/sec.
  Read 384018402 rows, 98.19 GiB in 19.6096461 sec., 19583137 rows/sec., 5.01 GiB/sec.

[5]: df.shape

[5]: (11350407, 15)

[6]: df.info()

  <class 'blackhole.dataframe.frame.DataFrame'>
  Index: 11350407 entries, 0 to 11350406
  Data columns (total 15 columns):
   #   Column              Non-Null Count     Dtype
  ---  ------              --------------     -----
   0   logId               11350407 non-null  object
   1   url                 11350407 non-null  object
   2   userId              11350407 non-null  object
```

- 查询结果可打通 DataFrame

02 性能指标-DataFrame & Machine Learning

DataFrame & ML 处理主流开源引擎中单机性能第一：

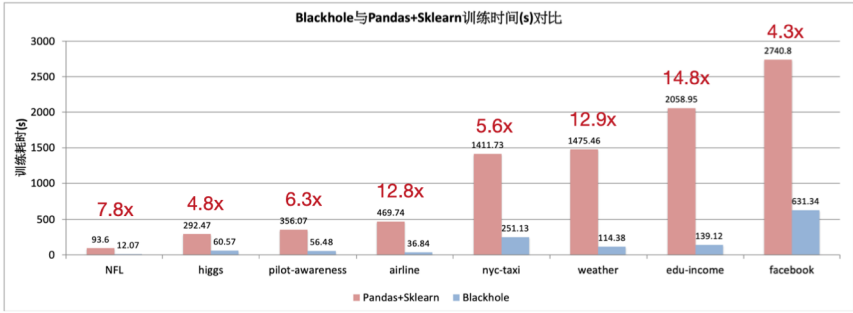
- 高频数据分析算子/机器学习算子 平均性能比主流工具快 17.3/7.42 倍
- 热门Kaggle任务 端到端场景平均比 Pandas+Sklearn 快 8.9 倍

数据分析：选取高频的7类数据分析算子[1]，统计不同数据量级下的平均耗时
结论：将最大可处理的数据量从128G 提升到 512G，总平均性能提升17.3倍

耗时(s)/耗时比	8G	32G	128G	256G
Jarvis	13.83秒/1倍	64.3秒/1倍	256.97秒/1倍	256.97秒/1倍
Pandas	230.7秒/16.68倍	1001.53秒/15.58倍	4557.56秒/17.74倍	Crash

机器学习：选取高频的6类机器学习算法[2]，统计不同数据集量级下的平均耗时
结论：最大数据处理量从十G级别提升到百G级别，总平均性能提升7.42倍

耗时(s)/耗时比	1G	8G	32G	64G
Jarvis	66.68秒/1倍	394.3秒/1倍	1603.36秒/1倍	3836.33秒/1倍
SKLearn	425.62秒/6.38倍	2964.58秒/7.52倍	11941.79秒/7.45倍	Crash



测试环境：CPU：48逻辑核，内存376G，Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz

加速倍数平均值为8.9倍 (Pandas 总时间 / Blackhole 总时间)

测试环境： CPU 28逻辑核， Intel(R) Xeon(R)；Gold 5117 CPU @ 2.00GHz；内存：256G；

[1] 数据分析算子包括：read_csv, sort, groupby, join, dropna, std
[2] 机器学习算子包括：Linear Regression, RandomForest Regressor, LogisticRegression, NativeBayes, RandomForestClassifier, Kmeans, PCA

02 性能指标-SQL

- 聚合算子组合测试：平均性能比 Spark 快 **2.15** 倍

  测试方法：基于 https://h2oai.github.io/db-benchmark/。预先将数据加载到Jarvis / Spark内存中，执行案例中的 Query，所有 Query 在不同量级数据下执行三次，取平均值。

  测试结果：

耗时(s)/耗时比	0.5G	5G	50G
Jarvis	2.24秒/1倍	17.28秒/1倍	110.58秒/1倍
Spark	17.31秒/7.73倍	29.69秒/1.72倍	228.86秒/2.07倍

- 端到端场景测试1（TPC-H）：平均性能比 Spark 快 **11.73** 倍

  测试方法：基于 https://clickhouse.com/docs/en/getting-started/example-datasets/star-schema/。预先在磁盘中生成大宽表，执行案例中的 Query，在不同量级数据下执行三次，取平均值。

  测试结果：

耗时(s)/耗时比	16G	128G
Jarvis	5.03秒/1倍	18.9秒/1倍
Spark	54.13秒/10.76倍	226.54秒/12倍

- 端到端场景测试2（TPC-DS）：建设中...

测试环境：48 逻辑核，376G内存，NVMe SSD盘

THANKS