Workload Characterization and Optimization of TPC-H Queries on Apache Spark

#### 目标

通过调节JVM参数和OS参数来提高spark的性能

#### 硬件与环境配置

Node: 2 processor(12 cores per processor),192worker threads

1TB RAM

1TB RAID5 disk

Ubuntu 14.10, Hadoop2.6.0, Spark1.5 IBM J9 JVM

#### 数据集：

TPC-H

100GB Hive table on HDFS（block size = 128M）

Lineitem 75GB

Orders 17GB

Partsupp 12GB

Customer 2.3GB

Part 2.3GB

Supplier 137MB

Nation 2.2KB

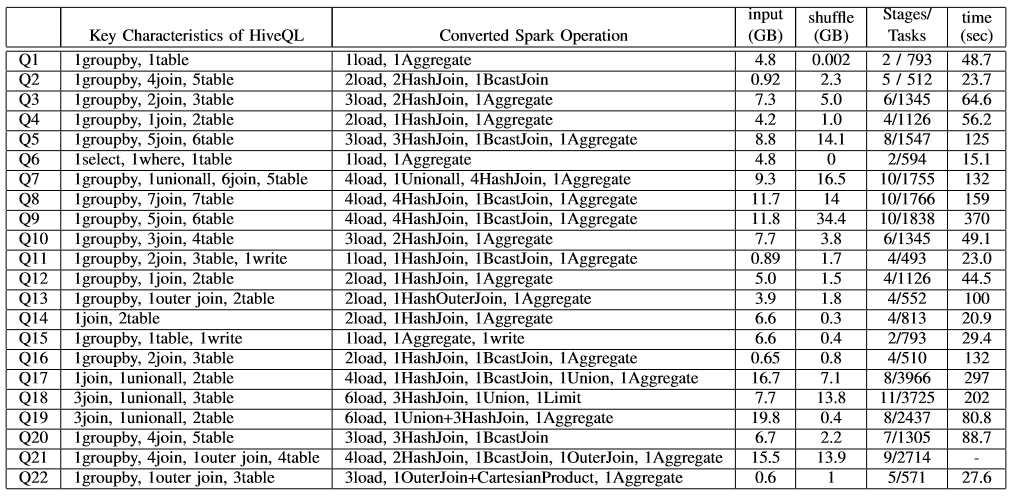
Region 389B

#### 执行结果

1. 修改了某些语句（主要是浪费了大量时间在建表上的语句）
2. 运行策略：每个查询迭代6次，取最小时间的那次迭代。

和取平均值得比较思考：

1. 目的调优，最小时间更加接近当前配置的理想情况
2. sql结果的确定性
3. 测试结果



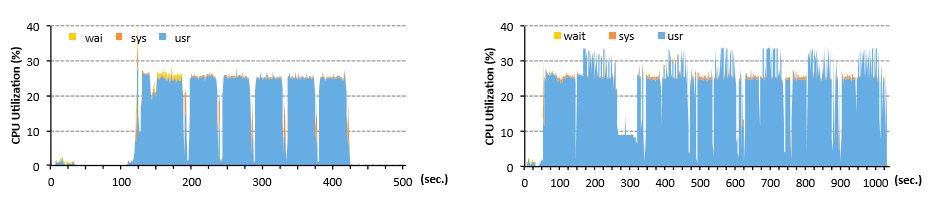
结论：shuffle data 的大小对运行时间有巨大的影响

Q1，Q6，Q19 little shuffle big input data

Q5，Q8，Q9 huge shuffle

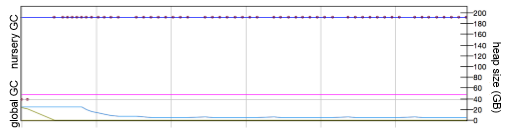
Two class：Shuffle light and shuffle heavy

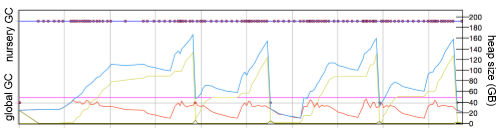
Q1和Q5 的cpu利用率的比较



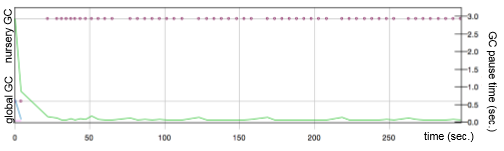
1. 第一次迭代的时间远大于后几次，原因是jvm的jit（Just in time）机制,在第一次迭代的时候热方法（hot method）的编译还未完成。
2. Q5每次迭代后的cpu利用率突增是heavy shuffle的gc过程

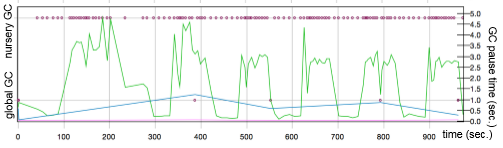
Q1和Q5的heap size比较

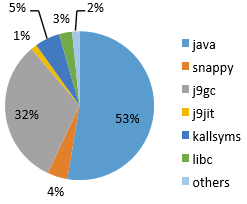




Q1和Q5的gc比较

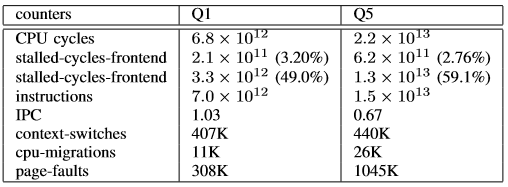






Q5引起了更多的堆内存中新生代和老年代的交换和更多的gc过程，gc过程所占时间很大

Q1和Q5的PMU（性能监视单元）比较（略）



#### 优化策略

1. 减少gc时间
2. 改变新生代space size
3. 调整JVM options
4. 保持总得堆内存和工作线程不变，改变JVM个数（Executor）

2．提高IPC,减少install cycles（停滞周期）

（1）改变SMT（同步多线程技术）的模式和工作线程个数

（2）使用NUMA亲和策略

#### 优化效果

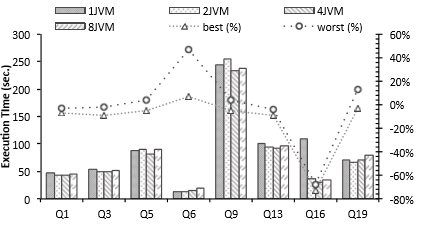
（1）增大新生代堆内存（48g default->96g,128g）：对于light shuffle的语句没有影响，对于heavy shuffle的语句，提高了10%-40%的效率，因为它减少了对象从新生代复制到老年代的操作。

（2）配置JVM参数：策略是每次添加一个参数配置，然后看结果比较。

-XlockReservation：作用于锁优化，对sql的序列化和反序列化过程提升了效率

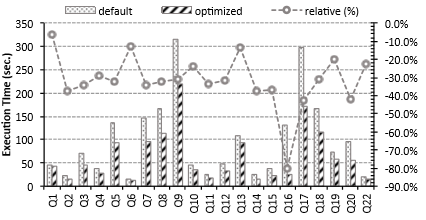
-XX:-RuntimeInstrumentation：作用于jit过程，对于某些sql有效，但有些有反作用

（3）改变JVM个数（在总heap和threads不变的情况下）：对于light shuffle的sql,越少的JVM越适合;对于heavy shuffle，2-4个JVM表现出了更好的性能



（4）采用NUMA亲和策略等：稍微有点效果（2%-3%）

采取所有优化策略之后的结果



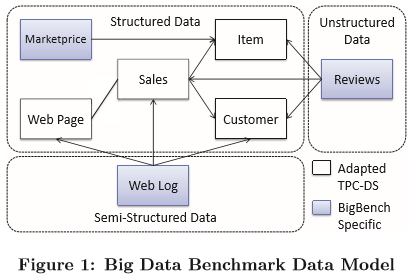
Bigbench：Towards an Industry Standard Benchmark for Big Data Analytics

#### 目标：

设计一个端到端的大数据基准测试bigbench

#### 内容

1. 数据模型设计



基于tpc-ds设计，拓展了半结构化数据和非结构化数据。

各个表之间的关系与规模设计

1. 数据生成方法

基于PDGF数据生成器生成随机数据，基于Markov Chain设计非结构化数据（商品评论）

1. workload设计

和McKinsey Report设计基于基于大数据零售业的workload，共30个

1. 评估

根据30条语句的时间综合评估

C:\Users\lenovo\AppData\Roaming\Tencent\Users\276627327\QQ\WinTemp\RichOle\J)@2AS0S1G~8J852OD4HA~G.png

T\_L: loading time

T\_D: the total time for queries in declarative processing

T\_P: the total time for queries in procedural processing

T\_B: the total time for queries in both

1. 验证

测试平台：Teradata Aster DBMS

Performance Evaluation of Spark SQL Using BigBench

#### 摘要：

In this paper we present the initial results of our work to execute BigBench on Spark. First, we evaluated the scalability behavior of the existing MapReduce implementation of BigBench. Next, we executed the group of 14 pure HiveQL queries on Spark SQL and compared the results with the respective Hive ones. Our experiments show that: (1) for both Hive and Spark SQL, BigBench queries perform with the increase of the data size on average better than the linear scaling behavior and (2) pure HiveQL queries perform faster on Spark SQL than on Hive.

1. 用BigBench来评估现行的MapReduce框架的扩展性
2. 比较SparkSQL和Hive运行14条HQL语句的性能

#### 内容介绍

• Scripts to automate the execution and validation of query results.

编写自动测试和验证结果的脚本

• Evaluation of the query scalability on the basis of four different scale factors.

评估4中不同数据规模下的扩展性

• Comparison of the Hive and Spark SQL query performance.

比较Hive和SparkSQL的查询性能

• Resource utilization analysis of set of seven representative queries.

7种代表性查询语句的资源利用率分析

#### 集群

4 nodes

master node: 2 \* 6 cores, 32GB RAM, 1TB disk

worker node: 6 cores, 32GB RAM, 4\*1TB disk

OS: Ubuntu Server 14.04

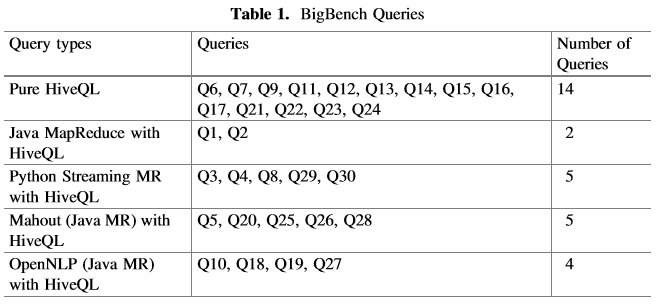
Hive version 0.13.1

Spark version 1.4.0

#### 数据集

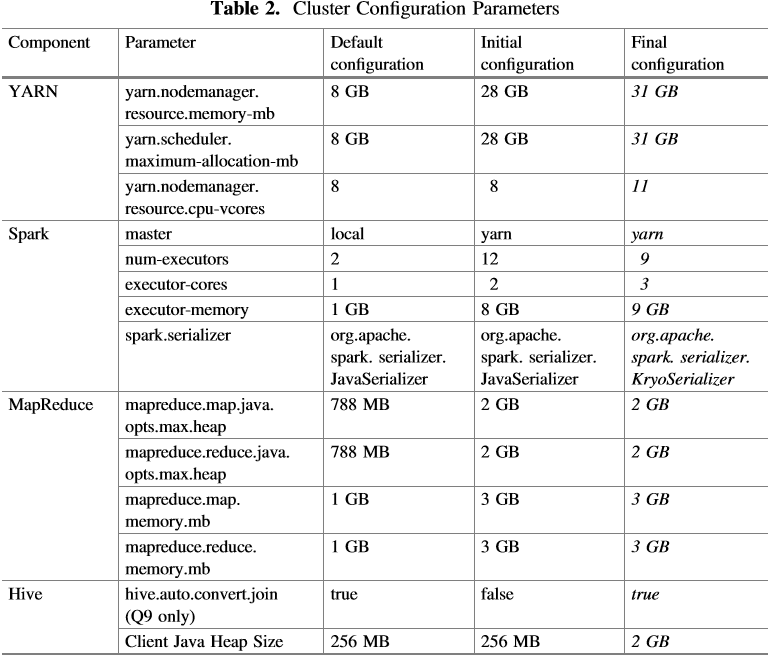
100G，300G，600G，1000 GB

Bigbench的30个查询语句



#### 参数

Ryza, S.: How-to: tune your apache spark jobs (Part 2) | Cloudera Engineering Blog, 30March 2015

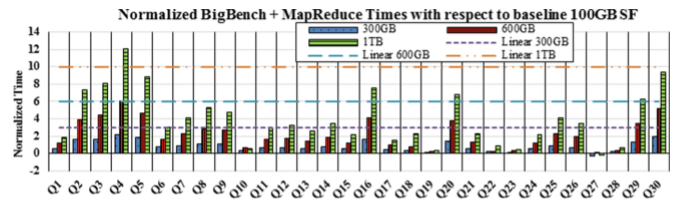


Final 和 initial的比较

#### 实验结果：

1. bigbench在mapreduce上的扩展性

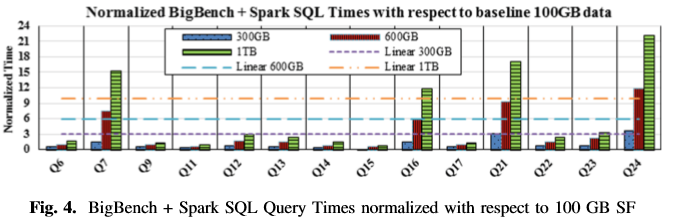
每个query运行3遍取平均值



以100G的数据为基准（baseline）标准化后的图

分析：Longer normalized times indicate that the execution became slower with the increase of the data size, whereas shorter times indicate better scalability with the increase of the data size

1. bigbench在spark上的扩展性

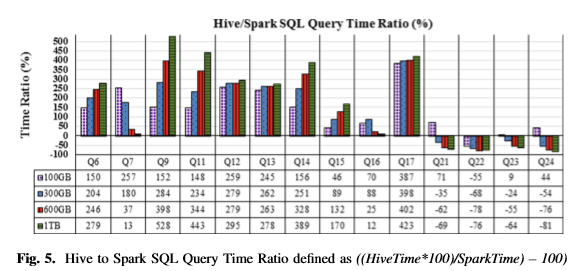


以100G的数据为基准（baseline）标准化后的图

结论：with the increase of the data size the BigBench queries perform on average better than the linear scaling behavior for both the Hive and Spark SQL executions

Q7，Q16，Q21，Q24所表现出性能（spark-5791）

Hive 和spark sql比较



Q7, Q16, Q21, Q22, Q23 and Q24所表现的性能（spark-5791）

#### 资源利用率

Tool：Intel’s Performance Analysis Tool

Metric：CPU utilization, disk I/O, memory utilization and network I/O

结合各种query的语义逻辑，来解释它们在这些metric上所表现的性能

Summary

1. a subset of the OpenNLP (MR) queries (Q19, Q10) scale best with the increase of the data size, whereas a subset of the Python Streaming (MR) queries (Q4, Q30, Q3) show the worst scaling behavior
2. For 14 pure HiveQL queries, both Hive and Spark SQL queries achieve on average better than linear data scaling behavior（except Q7, Q16, Q21, Q22, Q23 and Q24）
3. Find a join issue [20] in Spark SQL
4. for the stable pure HiveQL queries (Q6, Q9, Q11, Q12, Q13, Q14, Q15 and Q17), Spark SQL performs between 1.5 and 6.3 times faster than Hive
5. the different of cpu, io ,memory performance between spark sql and Hive