DOI：10.13439/j.cnki.itsc.2017.03.014

Python与Spark集群在收费数据分析中的应用

张雷
（重庆高速公路集团有限公司联网收费结算中心，重庆 401121）

摘要：随着智慧交通时代的到来，高速公路运营管理越来越依赖于大数据分析，如何有效利用现有硬件资源，更好地满足日益增长的数据分析需求是高速公路信息化建设中的重要课题。本文通过对数据库查询、Python内存计算和Spark集群的效率实测分析，将可弹性扩展的内存计算架构应用于高速公路数据分析，为运营大数据分析提供了平台基础。

关键词：数据分析；Python；Spark

在智慧交通时代，提升高速公路营运管理、运行管理与公众出行服务水平，将越来越依靠大量交通数据的分析。在营运管理方面，随着路网的不断扩大，联网收费相关业务也不断发展，特别在多义性精确清分和全国联网后，需要利用原始记录、二义性标识点数据进行精确清分和结算，ETC、二义性、小年票、区域优惠等业务也在不断新增和升级，路网对日常查询、联机事务处理、报表查询等需求越来越多。在运行管理方面，帮助管理者及时直观地了解目前的道路及交通状况，并提供公众出行服务，需要把大量繁复的道路和交通数据转化成直观的现有路况信息，形成交通流量短期和长期预测模型，并在模型基础上进行重要参数的敏感分析、交通安全分析等大量运算工作。因此，要想有效地利用各种生产数据更好地为路网运营管理服务，需要提高大数据应用分析能力。本文通过实例对比了传统的基于数据库的查询分析与基于内存计算的Python和Spark集群的数据分析效率，搭建了应用于收费数据分析的大数据平台，为运营大数据分析应用打下了平台基础。

一、基于数据库的查询

结算中心核心数据库服务器系统采用SAN架构，Informix数据库运行在小型机上，数据存储在共享磁盘阵列。原有的生产应用和管理查询均在同一系统，常常导致查询异常缓慢，还会出现生产数据入库积压、丢失的情况，难以满足日益迫切的生产和管理需求，在新的大数据分析平台建设中，将生产和管理应用分离，以解决上述应用瓶颈。新平台的数据库服务器由4台DL580G7组成，配置：内存128G，CPU E7-4820 2*8核2GHz；共享存储：EVA P6500 15K SAS硬盘；操作系统：CentOS 6.8；数据库：Oracle11g RAC。

新的管理数据库平台在原有架构上做了纵向扩展，即在硬件架构基本不变的基础上，提升服务器和数据库的数据处理能力。以下模拟2个实际应用查询，测试系统的响应时间。

应用一：简单的多用户并发查询。在300万条记录中同时查询不同站点收费金额。通过改变查询日期和站点，模拟从1条语句到50条并发的查询。查询全索引字段时，每条平均用时8.98秒；非索引字段查询并发量为1、10、50时，用时分别为5.37秒、41.76秒、214.56秒，用时与并发量呈线性增长。通常情况下，很难为所有查询字段建立索引，因此，对于高并发而言，查询效率将明显降低。

应用二：复杂的大数据量查询。对比展示若干收费站，不同车型非年票客车一段时间内通行量和收费金额汇总，去除节假日免费并按照ETC和非ETC车道分别展示。当运行1800万条记录的单表查询时，用时29秒；当数据量达到1.1亿，需要连接多表时，耗时8.11分钟。

数据库对于处理结构化数据有着简洁快速的优点，但当数据和并发量增大时也面临性能瓶颈：数据库的并发、索引及分表机制等因素制约着大数据查询分析性能的提高。在传统的数据库架构下，通过服务器的横向扩展提升性能也面临瓶颈，在面对多用户大数据量查询时尤其明显。

二、Python内存计算

Python的pandas数据分析包，在查询分析时将外部数据读入

内存，通过内存计算的方式，使得分析效率远高于基于外置存储的查询分析；同时，pandas提供了大量的快速便捷地处理数据的函数和方法，能够高效地进行数据预处理和分析。

在OD分析中，常常需要统计站点车流量。以收费站车流量排名与分钟车流量统计为例，在DL580服务器部署Python2.7及数据分析包进行测试。在统计分析前，需要对数据进行预处理，即ETL：抽取相关分析记录，对相应字段进行转换处理后，再进行整合。伪代码如下：

```
ex15=sql.read_sql('select extime,exstation from exlist2015',conn)
date=ex15.extime.apply(lambda:x,x.strftime('%Y%m%d')
hour=ex15.extime.apply(lambda:x,x.strftime('%H')
minute=ex15.extime.apply(lambda:x,x.strftime('%M')
s=pd.DataFrame({'exstation':ex15.exstation,'date':date,'hour':hour,'minute':minute}, index=None) #形成新的DataFrame
```

加载路网车流量数据后，可方便地分站、分时段统计车流量，如统计时段内各收费站通行量：

```
s.groupby('exstation').size().sort_values(ascending=False)
```

查询某站特定时间段内车流量明细：

```
s[(s['hour']==hour)&(s['date']==date)&(s['exstation']==exstation)].groupby('minute').size()
```

通过Python matplotlib绘图包显示，如图1、图2所示。

在脚本中加入计时器duration_time=time.time() - start_time，可统计出各步骤的运行时间。Python分析处理数据主要时间开销在于从外设一次性读入内存，1800万和1.1亿条记录加载分别用时2.13分钟和12.8分钟。数据载入内存后，计算效率显著提高，1800万条数据和1.1亿条记录内存计算用时如图3所示。

运用Python分析数据的优势在于内存计算的速度快，一次性读入内存后，每次分析无需重新加载数据，还可复用之前的分组排序结果，比如按照站点和时间的分组，使得后续分析效率进一步提高。由于Python数据分析包在数据处理上的灵活高效性，当内存对于所分析数据量足够大时，适用于各种营运分析与报表展示，能够大大提升分析效率。通过图3对比，可以发现分析效率随着数据增大而显著降低。用Python进行大数据分析的主要挑战在于大量数据一次性载入本地内存，数据读取和处理难以横向扩展，性能的提升面临瓶颈。

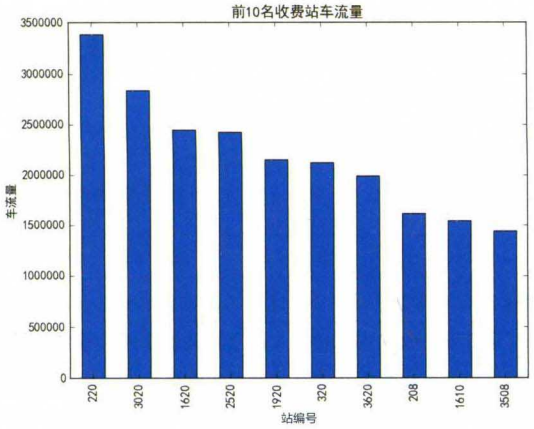

图1 车流量统计

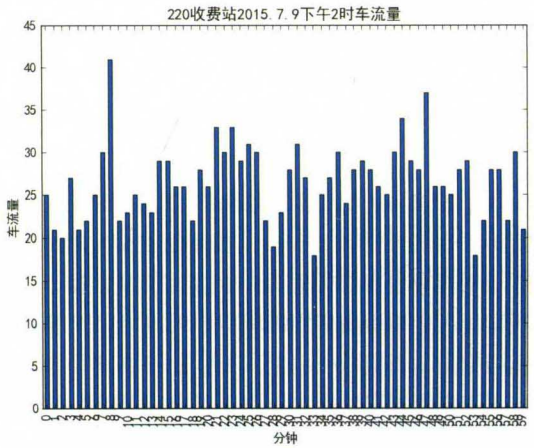

图2 站车流量分析

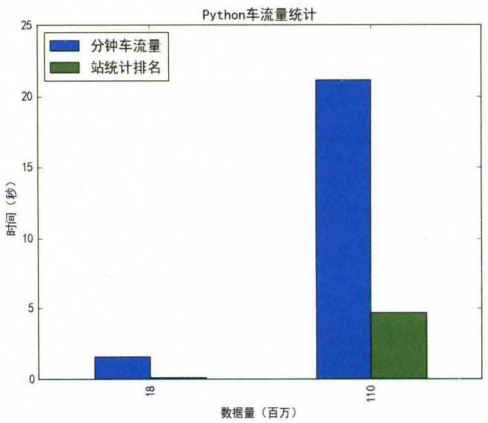

图3 Python数据分析性能

三、基于Spark集群的计算

Spark作为专门的大数据分析平台，通过统一抽象的数据结构RDD，实现分布式计算。数据分析者可以通过Spark做大数据ETL；通过SparkSQL进行大数据查询，应用SQL语句对结构化数据进行查询；通过导入pyspark包进行Python语言编程，因此具有很好的通用性。Spark分布式集群，能够很好地整合利用现有服务器资源。在平台测试中，通过千兆交换机连接的服务器能够轻易实现横向扩展，利用多台服务器的计算能力提高数据分析效率。

（下转第132页）

等，都加入了轴组式动态称重设备的选择行列，这也充分说明了轴组式称重设备的安全性、适用性和性价比更好。

四、动态称重设备发展展望

计重收费期望动态称重设备的计量性能能够达到静态称重设备（静态电子汽车衡）的准确度等级，以减少称重误差和称重争议，但是在现有技术条件下，动态称重设备还达不到这样的要求。从现阶段来看，动态称重准确度低于国标1级的动态称重设备正在逐渐退出应用，许多省份已经开始使用整车式或轴组式动态称重设备。虽然这两种设备的动态称重准确度达到了国标1级、防作弊效果好，但是，轴组式动态称重设备由于具有连续称重精度高、安全性好、投资少、施工快、维护便捷等优势，应该成为公路计重收费的首选称重设备。另外，轴组式动态称重设备长度合理，相对整车式设备来说，对安装环境要求更低、安全性更好、维护更简便，能够更好满足今后货车ETC的称重需要。目前，有些省份把轴组式和整车式动态称重设备进行联合应用，即普通车道采用轴组式，超宽车道采用整车式，利用整车式的静态称重性能，通过精准称重，快速解决计重收费的重量纠纷问题，这种模式充分发挥了两种设备的优势。

五、总结

由于公路计重收费连续称重、快速通行的特点，要求称重设备必须具有良好的动态称重性能，才能满足计重收费的需要。轴组式称重设备的出现，代表着现阶段动态称重的最新技术，因其具有高准确度、高防作弊性能和高性价比等优点，是计重收费与货车ETC称重的不二选择。同时，希望具有更高动态称重准确度、更好稳定性、更简便维护性的动态称重设备能够早日出现。

参考文献

[1] JJG 907-2006《动态公路车辆自动衡器检定规程》

[2] GB/T 21296-2007《动态公路车辆自动衡器》

[3] GB/T 7723-2008《固定式电子衡器》

[4] GB 1589-2016《汽车、挂车及汽车列车外廓尺寸、轴荷及质量限值》

责任编辑：刘睿健

（上接第123页）

以前面提到的收费数据查询需求为例，通过多进程并发读取从数据库抽取出来的数据文件，利用Python并结合Spark SQL在终端输出报表。伪代码如下：

```
spark=SparkSession.builder.getOrCreate() #生成spark运行环境
schema=StructType([StructField("exstation", IntegerType(), True),\,其它数据对应字段]) #生成表结构
spark.read.csv(txtpath,sep="|",schema=schema,dateFormat='yyyy/MM/dd')
s.createOrReplaceTempView('df') #注册为临时表
sql=对应SQL查询语句;s1=spark.sql(sql);s1.show()#显示查询结果
```

对比Spark Standalone模式下单机和3台服务器集群查询1个月、半年和1年数据时的效率，结果如图4所示。

以上统计时间包含了从读取数据文件到形成分析结果的全过程。由于分区和延迟计算的特性，Spark通过并发读入需要分析的数据，加载效率远高于纯Python编程。结果表明，Spark分布式集群能够很好地利用新加入的服务器内存和计算资源，在服务器资源横向扩展时明显提高数据分析效率。

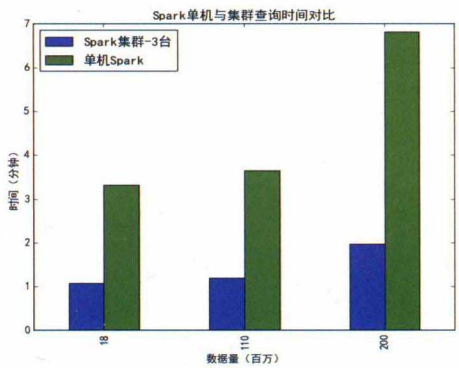

图4 Spark数据分析性能对比

四、总结

Python内存计算能够显著提高数据分析效率，并且使得中间结果可复用。而利用Spark集群，能够在利用原有Python和SQL语言优势的基础上，方便地扩展计算资源，在充分利用现有服务器资源的同时满足大数据应用分析需求。同时，Spark集成了MLlib机器学习包和Streaming实时计算框架，可应用于营运稽查及车流量预测等领域，为高速公路营运及运行大数据分析提供了很好的平台支撑。

责任编辑：刘睿健