



主流大数据处理开源架构的分析及对比评测

方艾,徐雄,梁冰,张玉忠,杨翊平

(中国电信股份有限公司广州研究院 广州 510630)

摘要: 结合电信增值业务领域中对大数据处理的实际需求,对现有主流的分布式大数据处理架构(Hive、Impala、Spark)的核心进行分析与实测,比较它们在大数据处理过程中的优劣及适用的场景,从而为大数据分析所面临的架构适用性选型提供参考。

关键词: 大数据;Hive;MapReduce;Impala;Spark

doi: 10.11959/j.issn.1000-0801.2015174

Comparison of Open-Source Distributed Computing Framework for Big Data

Fang Ai, Xu Xiong, Liang Bing, Zhang Yuzhong, Yang Yiping

(Guangzhou Research Institute of China Telecom Co., Ltd., Guangzhou 510630, China)

Abstract: A comparison of three open source distributed computing frameworks for big data (Hive, Impala and Spark) was conducted. Tests were run to evaluate the performance aiming at real business demands. The cost of implementation to meet business requirements was also discussed.

Key words: big data, Hive, MapReduce, Impala, Spark

1 引言

分布式并行计算是当前对大数据处理分析的主要方式,基于Google提出的分布式文件系统GFS(Google file system)和MapReduce算法,Hadoop开源实现了一个通用的计算框架^[1],通过Hadoop,各种计算需求可分解成一系列MapReduce过程,Hadoop框架具有通用、灵活方便、扩展性好、吞吐量大等优点。Hadoop的MapReduce发展到2.0版本,进一步优化了资源调度与利用,加强了容错性、可用性等。Hive等工具正是利用这一框架,将SQL类型的查询与计算转化为MapReduce过程,实现了对海量数据的处理。

MapReduce是一个批处理的过程,难以满足查询的实时性和交互性。为了应对实时要求高的计算,一些工具避开了MapReduce过程,如Impala^[2],更适合于实时的交互式查询,给数据分析人员提供了快速实验、验证想法的大数据分析工具。

为了实现高效率的计算,Spark^[3]采用了基于有向图(DAG)的计算框架,将计算任务分解成为可同时进行的任务,使用RDD(弹性分布数据集)数据模型,将中间数据放进内存并可共享,避免频繁的磁盘读写以提高处理速度。同样作为一个通用性的计算框架,Spark具有高容错、高性能等优点,特别适合社交网络分析、机器学习、流处理等。

收稿日期:2014-12-12;修回日期:2015-07-08

论文引用格式:方艾,徐雄,梁冰等.主流大数据处理开源架构的分析及对比评测.电信科学,2015174

Fang A,Xu X, Liang B, et al. Comparison of open-source distributed computing framework for big data. Telecommunications Science, 2015174

在电信的大数据分析处理的实际需求中,考虑集群配置(包括内存、CPU、网络环境等)限制条件,这3类处理框架的实际性能表现如何,各自适用于什么样的应用场景?需通过实际业务驱动的方式验证。本文以实际运营中产生的电信产品日志处理及分析为基础,紧贴实际业务需求设计案例进行测试对比,对不同场景下不同框架的吞吐量、响应时间、资源消耗等情况进行综合评价,为不同架构的适用性分析和技术选型提供参考。

2 核心环节对比

2.1 MapReduce

MapReduce 的基本原理是将任务分解成适合并行执行的 map-shuffle-reduce(映射-整理-归纳)过程,分布在不同机器上运行。如图1所示,mapper将输入数据映射为成对的(key,value)格式,对 key 和 value 进行操作(计数、累加、去重等),产生中间数据,经过重新排列组合,相同的 key 被发送(shuffle)到同一个 reducer,进行汇总,最后输出结果。由此可见,MapReduce 是一个批处理过程,上一个环节结束后才能执行下一个,如果一个子任务的中间环节出错,则这个任务需重新开始。复杂的任务需要被分解为更多的 MapReduce 过程,因此导致了高时延,加上 shuffle 时需将数据写入文件系统,硬盘读写与节点间网络传输也增加了时延。需要迭代运算的任务因此变得特别慢。

Hive、Hadoop streaming 都基于 MapReduce,能够很好地应对大规模离线数据的清洗、统计、分析等任务,但始终未能突破 MapReduce 机制的限制,计算的时间成本高。

2.2 Impala

为应对时效要求高的应用场景,如实时查询、交互式查询等,Impala^[4]抛开 MapReduce 机制,使用分布式查询引擎直接从 HDFS 或 HBase 中查询数据,从而大大降低时延,从客户端使用角度看,Impala 与 Hive 有很多的共同之处,如数据表元数据、ODBC/JDBC 驱动、SQL 语法、灵活的文件格式、存储资源池等。Impala 与 Hive 的关系如图2所示。

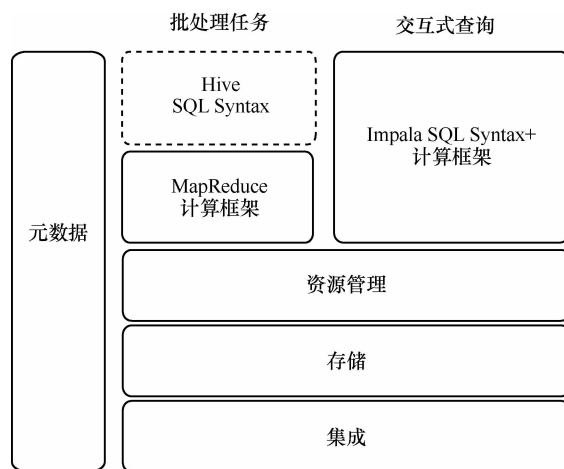


图2 Impala 与 Hive 关系

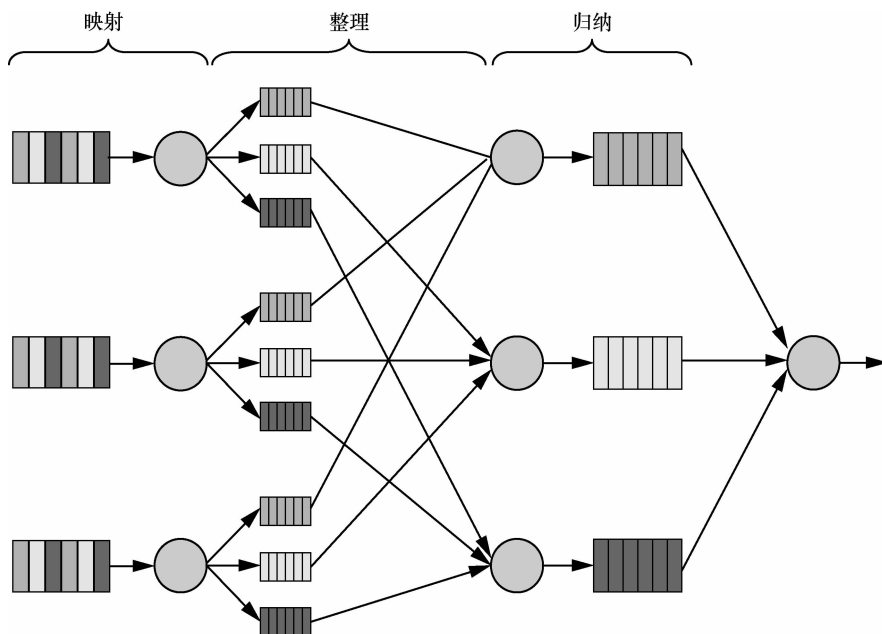


图1 MapReduce 过程示意



Impala 在每个节点运行一个 Impalad 服务(如图 3 所示),用于响应外部请求并完成实际的查询处理。Impalad 由 query planner(查询计划器)、query coordinator(查询协同器)和 query exec engine(查询执行引擎)组成。查询计划器将来自 SQL App 和 ODBC 的查询转换成多个子查询,查询协同器将这些子查询分发到各个节点上,由各节点上的查询执行引擎负责子查询的执行,最后返回子查询的结果,这些中间结果经过聚集之后最终返回给用户。

Impala 借鉴了 Google Dremel^[5]的树状模型(如图 4 所示),客户端发起请求后,根节点收到请求并将其分解到枝叶,一直到数据所在的叶子节点,叶子节点扫描处理数据,又不断往回汇总到根节点。

不过,目前的 Impala 版本还不支持容错,当一个子查询失败时,最终结果将失败。Impala 对内存的要求也比较高,在内存要求超出系统容量时,任务将失败。

2.3 Spark

Spark 是由 UC Berkeley AMP 实验室发布的开源通用

并行计算框架^[6],相对 Hadoop MapReduce2,有两大不同。

- 用有向图(DAG)数据流模型取代了 MapReduce2 固定的数据流模型,支持复杂的、可迭代的作业,多个 DAG 可同时启动,避免了 MapReduce2 中麻烦的同步问题,提高执行速度的同时也使任务分解更简单灵活。
- 引入弹性分布数据集^[7](RDD)模型,相比 MapReduce2 的(key,value)模型,RDD 适合分布式内存运算并且容错能力强,弹性分布数据集模型将数据抽象成一系列 RDD 对象,通过其数据依赖关系可以计算出另一个 RDD,因此,当某个 RDD 丢失时,不一定要去文件系统或另一个节点读取备份,只需根据在内存中的上一个依赖直接计算便可获得,也无需从头开始计算,避免了大量磁盘时延和网络传输时延。此外,DAG 之间可共享 RDD 数据。RDD 的主要操作有转换(map, filter, join)、动作(count, collect, save),如图 5 所示的错误日志查询作业,line、error、HDFS error、time filed 都是 RDD 对

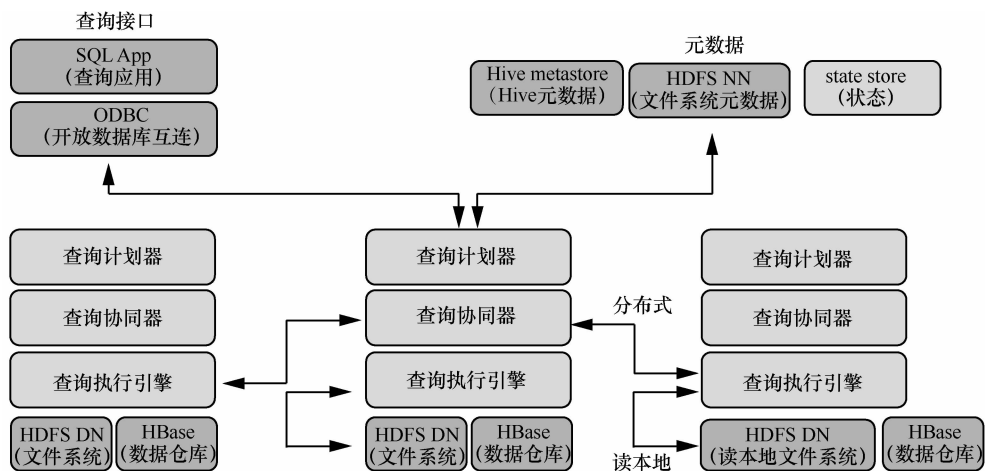


图 3 Impala 架构

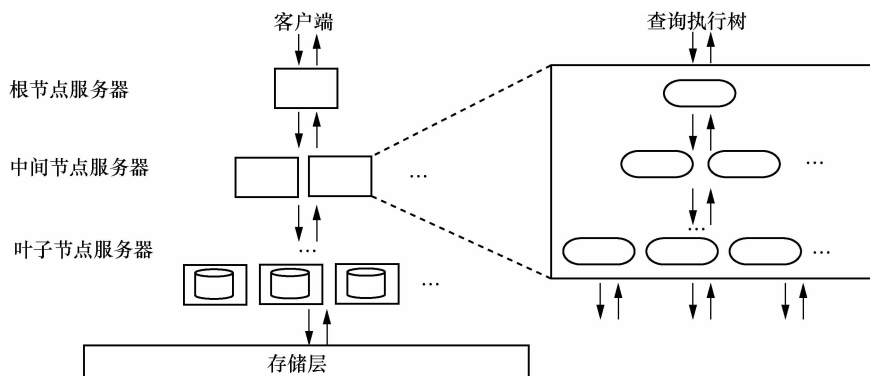


图 4 Google Dremel 查询示意



图5 Spark RDD 举例示意

象, line 从磁盘转换得到, error 由 line 转换 (filter) 得到, HDFS error 由 error 转换 (filter) 得到, time field 由 HDFS error 转换 (map) 得到, 最终可通过 collect 动作获取 time field。RDD 位于内存中, 并且只在动作 (例如 count) 触发时才进行计算。在计算过程中, 如果某个节点的某个环节出错, 导致 (如 HDFS error) 丢失了, 这时并不一定需要从磁盘重新读取, 而是可以从该节点的上一个可用的 RDD (如 error) 计算。

Spark 特别适用于社交网络分析, 机器学习等需要大量迭代的计算, 例如常用的梯度下降、最优、逻辑回归等。

3 实测对比

MapReduce 是现阶段被广泛应用的大数据处理框架核心, Hive、Hadoop streaming 是作者日常使用最多的工具; 实时性、交互性的要求使得 Impala 被纳入使用; 基于机器学习的大数据分析需要的出现, 使得 Spark 也被纳入考察。因此, 本文针对实际应用需求设计了测试场景, 对 Hive、Impala、Spark 在现有软硬件环境进行实测对比。

3.1 测试环境

针对上述几种不同类型的计算, 使用统一的测试环境, 利用存储在 HDFS 的同一份数据, 对比 MapReduce、Impala、Spark3 种工具的处理速度和资源利用率。

测试平台由 12 台 PC 服务器组成, 其中 8 台 Dell 2950、4 台 HP DL580G5, 每台服务器配置了 2 路 4 核 CPU, 其中 3 台内存 32 GB, 两台 8 GB, 其余 16 GB; 使用 1 000 Mbit/s 带宽的局域网, 安装 CentOS 6.3 64 位操作系统。

集群部署了 CDH5 试用版, 其中, 10 台服务器作为 HDFS 的数据节点 (DataNode), 存储了测试数据, 同时运行了 Yarn 的节点管理进程 (NodeManager)、Impala 的核心进程 (Impalad) 和 Spark 的节点进程 (Worker), 其余两台作为管理主节点 (NameNode) 以及 Yarn 的资源管理 (resource manager)、Spark 的主节点 (master)。

本次测试的软件版本分别是: CDH 5.1.2、MapReduce 2.3.0、Hive 0.12.0、Impala 1.4.0、Spark 1.0.0。

3.2 测试场景与结果

运营商的某个音乐产品每日产生 4 000 万行以上的日志, 历史数据数十亿行。运营报表要求每日、每周、每月的 UV (访问用户数)、PV 以及针对业务的排序、关联分析、用户价值挖掘等数据分析, 相关的计算可总结为如下几个主要类型:

- 去重与累计, 如计算日 UV、日 PV 等;
- 表关联, 如留存率、活跃度、用户与业务的关联分析等;
- 循环迭代, 如用户社交网络关系分析、用户重要度排名等。

3.2.1 场景一: UV (访问用户数) 统计

特点: 计算日志中不重复的 ID 数量, 对应 SQL 的 count (distinct id) 语句, 分别计算 1 天、3 天、7 天的 UV。UV 统计性能对比如图 6 所示。

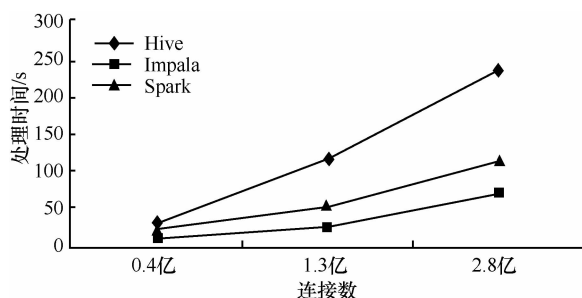


图6 UV 统计性能对比

原始日志每日约 4 000 万行记录 (未压缩 12 GB), 每行记录 25 个字段。计算得出每日 UV 约为 100 万。从图 6 的结果看, 处理时间随着数据量的加大而增长; Impala 的查询速度最优, Spark 次之, Hive 由于转化成 MapReduce, 中间的 shuffle 过程存在网络传输和磁盘读写操作, 时延较大。

3.2.2 场景二: 1~N 天每天用户留存率

特点: 计算某一天的新增用户, 在后续的第一、第二、一直到第 N 天中, 每天到访的比例。首先需统计新增用户数作为分母, 再按天关联继续到访的用户并统计出数量作为分子。

- 算法 1: 每天行为表与用户表进行关联 (重复使用用户表), 循环计算后续天数。
- 算法 2: 多天行为表与用户表进行关联 (无重复使用), 再按天汇总。



算法 1 采用迭代方式,需要重复使用用户表,符合 Spark 的特点,但不适合 Hive,因为 Hive 每次都需要从磁盘读取该表,因此,测试中 Spark 有算法 1、算法 2 两种, Hive、Impala 则使用了算法 2。结果如图 7 所示: Impala 速度最快。实际上,由于每次迭代时间相对固定,随着计算天数的增多,耗时成比例地增加,使用 Spark 时可将两种算法结合起来,选取适当的步长以达到最佳的效率。受限于测试环境配置,在进行数据量较大关联(join)操作时,Spark 的性能反而下降,本次测试的是 Spark 版本(1.0.0),在任务提交时如果没有申请足够的内存资源,任务可能失败,单个 executor 核心数较少时,任务执行速度也较慢。

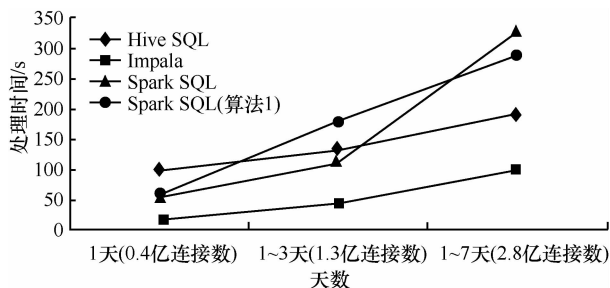


图7 留存率统计性能对比

3.2.3 场景三:基于 PageRank 算法^[9]的用户重要度排名

特点:需进行多重迭代,反复更新用户的排名得分(rank)。

(1) 相同数据量,不同迭代次数的对比

对同一份数据(约 10 万个节点,120 万个连接),分别进行 10 次、20 次、30 次和 40 次迭代,得到结果图 8 所示。

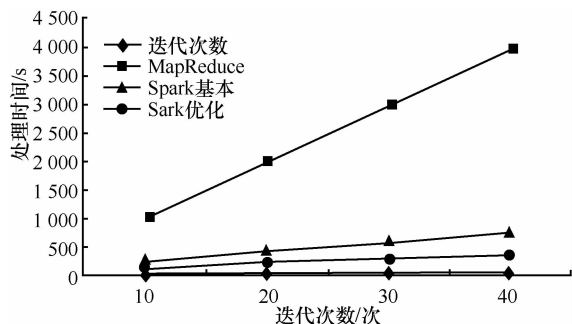


图8 重要度排名性能对比(不同迭代次数)

在 MapReduce 处理过程中,每次迭代需从 HDFS 读取 links 和 ranks,完成时将 ranks 写回 HDFS,加上中间 shuffle 过程也需要读写磁盘,每次迭代约需 100 s。而 Spark 进行迭代时,无需将结果写回磁盘,每次迭代只需 16 s;对 links 进行分区优化,即将 ID 相同的 links 和 ranks 分布在同一

个节点上,避免 join 时需要跨节点传输数据,总体上还可提高一倍的性能。

(2) 相同迭代次数,不同数据量的对比

选取同样的迭代次数(10 次),分别对大、中、小 3 种数据量进行测试,数据来源于微博用户关系的实际样例^[9]。

- 数据 1:约 1 万行,包含约 68 万个节点,133 万个连接。
- 数据 2:约 5 万行,包含约 335 万个节点,838 万个连接。
- 数据 3:约 36 万行,包含约 1 680 万个节点,4 890 万个连接。
- 数据 4:约 75 万行,包含约 3 260 万个节点,1 亿个连接。

MapReduce 的性能表现非常稳定,同一份数据中的每次迭代耗时接近。处理小数据量的数据 1 时,耗时约 880 s(每次迭代约需 88 s),随着数据量的加大,处理时间有所增加。中等数据量(数据 2)每次迭代耗时约 107 s;大数据量(数据 3)每次迭代耗时约 170 s,更大数据量(数据 4)迭代约需 230 s,如图 9 所示。

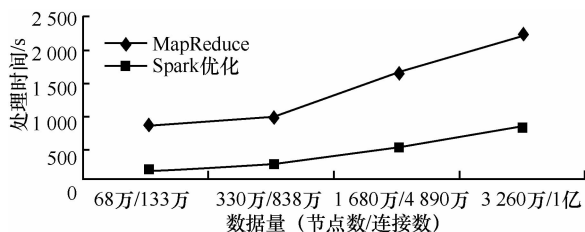


图9 重要度排名性能对比(不同数据量)

Spark 的性能则高出很多,处理数据 1 时,10 次迭代只需 138 s,处理数据 2 时 10 次迭代只需 260 s;随着数据量加大,耗时明显增加,原因是测试平台的内存配置较低,处理大量数据(数据 3、数据 4)时,需将 RDD 写入磁盘,影响了性能,10 次迭代耗时分别达到 532 s 和 885 s。如果强制不写入磁盘,则出现内存不足导致任务失败。在数据量增大的情况下,Spark 的稳定性也遇到一些问题,Spark 1.0.0 版本在数据 3 和数据 4 的测试中发生 StackOverflow,需要升级到 Spark 1.1.0 版本。

服务器资源的使用上,以测试场景一为例,如图 10 所示(横轴 10:30 前后为 Hive 执行时间,10:40 前后为 Impala 执行时间,10:45 之后为 Spark 执行时间),Hive 与 Impala 使用的 CPU 相对 Spark 少。各种工具执行时都需要从 HDFS 读取原始文件,因此在磁盘 I/O 和 HDFS I/O 上都表现出大量的读和少量的写。网络 I/O 上 Impala 相对很少,Spark 比 Hive 少。

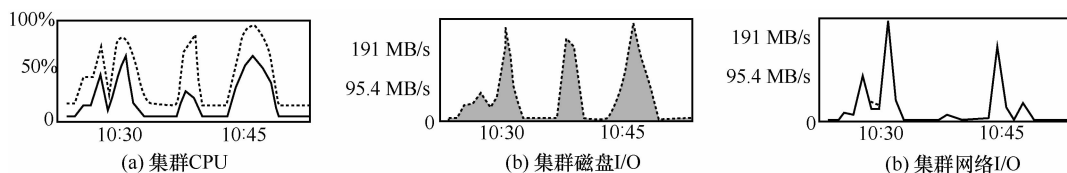


图 10 资源使用情况

4 结束语

Hadoop MapReduce 仍然是目前较为成熟完善的架构,适用于时效要求不高的各种作业,其对集群的配置要求也不高。对于数据量大、逻辑复杂的处理过程仍然是低成本的选择。

Impala 的响应速度非常快,相比 Hive 需要分钟级的任务,Impala 可在秒级完成,适合用于交互式的应用,但对内存大小要求高,在操作大表时对内存消耗大且内存不足时需要交换到磁盘,可能会影响性能。Impala 适合与 Hive 配合使用,对 Hive 的结果数据集进行实时分析。

Spark 在数据量小于或接近内存总量时表现优异,但当数据量远超内存容量时存在一些稳定性的问题,Spark 的新版本对此做了大量改进,但仍有待提高。Spark 性能的表现依赖集群的可用资源,特别是在本测试环境中,由于机器配置不高,内存较少,大数据量时需要进行内外存交换,导致性能下降。对于需要进行机器学习等应用的场合,Spark 有先天的优势,期待 Spark 的进一步发展完善。

参考文献

- 1 Hadoop. <http://hadoop.apache.org/>, 2014
- 2 耿益锋, 陈冠诚. Impala: 新一代开源大数据分析引擎. 程序员, 2013(8)
Geng Y F, Chen G C. Impala: next-gen open-source big data analytic engine. Programmer, 2013(8)
- 3 Spark. <http://www.cnblogs.com/jerrylead/archive/2013/04/27/Spark.html>, 2013
- 4 Hive 与 Impala 比较. <http://www.linuxeden.com/html/news/20130714/141332.html>, 2013
Comparison between Impala and Hive. <http://www.linuxeden.com/html/news/20130714/141332.html>, 2013
- 5 Dremel. <http://www.yankay.com/google-dremel-rationale/>, 2012
- 6 Spark API. <http://spark.apache.org/docs/latest/api/python/index.html>, 2014
- 7 Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. http://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf, 2012
- 8 PageRank. <http://en.wikipedia.org/wiki/PageRank>, 2014

9 微博 3.5 亿用户关系数据 (样例). <http://www.datatang.com/data/46248>, 2014

Relationship data of 350 million weibo users (samples). <http://www.datatang.com/data/46248>, 2014

[作者简介]



方艾,男,中国电信股份有限公司广州研究院工程师,主要研究方向为大数据、机器学习。



徐雄,男,中国电信股份有限公司广州研究院工程师,主要研究方向为互联网应用。



梁冰,女,中国电信股份有限公司广州研究院工程师、架构师,主要研究方向为软件开发。



张玉忠,男,中国电信股份有限公司广州研究院工程师,主要研究方向为数据挖掘、云计算。

杨翊平,男,中国电信股份有限公司广州研究院工程师,主要研究方向为搜索引擎技术、云计算、互联网应用。