一种高效的Flink与MongoDB连接中间件的研究与实现

胡 程[1],叶 枫[1,2]
1.河海大学 计算机与信息学院,南京 211100
2.南京龙渊微电子科技有限公司 博后工作站,南京 211106

摘 要:为了提高大数据处理平台Flink与MongoDB之间的读写速率,提出并实现了一种高效的Flink与MongoDB连接中间件。基于Flink的并行化思想,通过对数据进行逻辑分片,调用Mongo-Java包中的接口实现并行化将数据读取和写入。以不同规模的水文传感器数据集作为实验数据,实验了在Java单线程操作、Hadoop与MongoDB连接器和提出的Flink与MongoDB连接中间件三种连接方式下数据的读写速度。结果表明,Flink并行读写数据效率较于单线程提高了1.5倍,验证了该连接中间件可以有效地提高对海量数据的读写速率。

关键词: Flink;MongoDB;Flink与MongoDB连接中间件;数据逻辑分片;并行读写
文献标志码:A **中图分类号:**TP391 doi:10.3778/j.issn.1002-8331.1808-0342

胡程,叶枫.一种高效的Flink与MongoDB连接中间件的研究与实现.计算机工程与应用,2019,55(23):64-69.
HU Cheng, YE Feng. Research and implementation of efficient connection middleware between Flink and MongoDB. Computer Engineering and Applications, 2019, 55(23):64-69.

Research and Implementation of Efficient Connection Middleware Between Flink and MongoDB

HU Cheng[1], YE Feng[1,2]
1.College of Computer and Information, Hohai University, Nanjing 211100, China
2.Postdoctoral Centre, Nanjing Longyuan Micro-Electronic Company, Nanjing 211106, China

Abstract: In order to improve the reading and writing rate between big data processing platform Flink and MongoDB, this paper proposes and implements an efficient connection middleware of Flink and MongoDB. Based on Flink's parallelization idea, by logically fragmenting the data, the interface in the Mongo-Java package is called to realize parallel reading and writing of data. With different scale of hydrological sensor datasets as experimental data, the reading and writing speeds of the data in Java single-threaded operation, Hadoop and MongoDB connector and the Flink and MongoDB connection middleware proposed in this paper are tested. The results show that the efficiency of using Flink to read and write data is 1.5 times higher than the single-threaded operation, which validates that the connection middleware can effectively improve the reading and writing speed of massive data.

Key words: Flink; MongoDB; Flink-MongoDB connector middleware; logically fragmenting data; parallel reading and writing

1 引言

随着大数据时代的到来,基于大数据处理平台对海量数据进行计算处理,基于NoSQL对海量数据进行存储已经成为主流趋势。以Flink、Spark为代表的大数据处理平台能为数据处理提供了超大规模的、并行的计算能力;以MongoDB为代表的NoSQL数据库则为海量数据的存储提供了支持。但是,连接于大数据处理平台与NoSQL存储之间的存取机制往往成为大数据处理的瓶颈,如何构建大数据处理平台NoSQL数据库之间的连接中间件,并实现快速、高效地进行数据读取已成为了大规模数据计算和处理的关键挑战。针对该问题,已有了一些有代表性的工作[1-9]提出了特定解决方案,例如:

基金项目:2017江苏省博士后科研资助计划(No.1701020C);2017江苏省"六大人才高峰"资助项目(No.XYDXX-078);中央高校基本业务费项目(No.2013B01814)。
作者简介:胡程(1995—),男,硕士研究生,研究领域为分布式计算、数据挖掘;叶枫(1980—),通讯作者,男,博士,讲师,研究领域为分布式计算,E-mail:yefeng1022@hhu.edu.cn。
收稿日期:2018-08-20 **修回日期:**2018-10-12 **文章编号:**1002-8331(2019)23-0064-06
CNKI**网络出版:**2018-12-28, http://kns.cnki.net/kcms/detail/11.2127.tp.20181227.1009.023.html

使用基于Hadoop与MongoDB连接器整合Hadoop应用与MongoDB数据库，基于mongo-spark-connector整合Spark应用与MongoDB数据库。随着流数据处理技术的发展，Apache Flink[10]作为新一代的大数据处理平台，它能够同时处理流数据和批数据。MongoDB作为典型的NoSQL数据库，适量级的内存存储机制使得热数据的读写变得十分迅速；易扩展的架构使得集群扩展变得十分简单；自身的Failover机制为集群提供了高容错机制；以Json和Bson的存储格式非常适合进行文档的存储和查询。在大数据时代下，以Flink作为大数据处理平台，MongoDB作为底层数据库进行存储，将非常适合当前的许多业务需求。但是，当前的Apache Flink的系统组件中并没有包括与MongoDB的连接器，这对于使用MongoDB存储业务数据并进行分析处理的用户而言，非常不方便。针对该问题，本文提出了一种Flink与MongoDB数据库连接中间件，并且在不同规模的水文数据集下进行了读写的实验，对比验证了在Java单线程操作MongoDB数据库、基于Hadoop与MongoDB的连接器和本文提出的Flink与MongoDB连接中间件下的数据读写效率，最后进行了总结和展望。

2　相关工作

随着数据量和复杂性的快速增长，结合大数据处理平台对海量数据进行计算和处理成为大数据时代的主流趋势。主流的大数据处理平台有：Hadoop、Spark、Flink。Hadoop[11]主要用于离线大批量任务的处理，具备高吞吐性和高可靠性等，但是延迟较高；Spark[12]是专为大规模数据处理而设计的快速通用的计算引擎，具备更快处理速度和通用性，并且可以用于数据的实时处理；Flink[10]作为新一代大数据处理平台，可以同时进行流处理和批处理。而传统的关系型数据库面对超大规模数据集和高并发的数据库读写能力略显不足，随之NoSQL数据库成为主流的数据存储工具，包括了：基于列存储的Cassandra[13]和HBase[14]数据库、文档型数据库MongoDB[15]数据库等。

当前已有很多基于大数据处理平台和数据库的整合方案。Lei等[1]提出了一种整合Hadoop和MongoDB数据库的基于云的矢量数据存储和分析系统的方法和系统；Dai等[2]提出了一种利用MongoDB和Hadoop的水平可扩展的云存储结构，为高性能私有云存储平台打下了坚实的基础；曾强等[5]对Hadoop和MongoDB进行合理的集成进行了详细的研究；Abouzeid等[7]对如何使用Hadoop和数据库构件实用的应用进行了描述；陈德森等[8]基于MongoDB数据库，结合Spark的机器学习库，进行了文本分类研究；Hajoui O、Talea M[9]比较了Spark和Hadoop结合MongoDB数据库的性能。

但是，如何在大数据处理平台和NoSQL数据库存储之间进行快速、高效的数据读取成为大规模数据计算和处理的关键挑战。Oracle为Hadoop与Oracle数据库整合提供了Oracle Big Data Connector[16-17]，构建了一个高性能的Hadoop与Oracle集成解决方案，容易直接对所有结构化和非结构化数据进行分析；10gen提供了MongoDB Connector for Hadoop[5,18]，用于整合MongoDB数据库和Hadoop，提供了使用MongoDB数据库代替HDFS作为MapReduce数据输入源和输出源；HORTONWORKS发布了基于HBase的Spark连接器Spark-on-HBase[19]，弥补了HBase键值存储和复杂关系SQL之间的鸿沟，使用户能够使用Spark在HBase之上执行复杂的数据分析；10gen为Spark和MongoDB的整合发布了MongoDB Connector for Apache Spark连接器[20]，利用MongoDB的聚合管道和丰富的索引来提取、筛选和处理所需要的数据的范围，从而最小化地实现跨集群的数据移动并减少等待时间；DataStax为Spark与Cassandra数据库整合实现了Spark-Cassandra-Connector[21-22]，该连接器允许将Cassandra表作为Spark RDDs，并将Spark RDDs写入到Cassandra表中，并且任意地执行CQL查询在Spark应用程序中；当前Flink支持使用Hadoop的数据类型、输入和输出格式从HDFS、Amazon S3、Alluxio、Avro、Microsoft Azure storage、MongoDB数据库[23]中读取或写入数据。

综上，在大数据处理平台和数据库之间使用连接中间件可以实现对海量数据进行快速、高效的读取。但是Flink作为新一代的大数据处理平台，在与典型的NoSQL数据库MongoDB之间进行读取中缺乏了专一、高效的连接中间件，因此本文提出并开展了该研究。

3　基于Flink连接MongoDB的中间件

3.1　中间件的体系结构

本文通过分析了解MapReduce的数据读取方式，设计了Flink与MongoDB数据读取的处理流程，如图1所示。

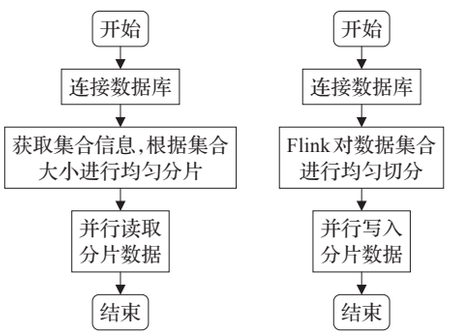

图1　读取、写入流程

从图1中可以看到与传统的读写数据不同的是，一次性读取一个庞大的数据集会严重堵塞方法的执行速

度,合理利用大数据处理平台提供了完全并行化处理的功能与数据库的高并发能力,多线程同时读取、写入不同的数据块,可以有效地提高读写。首先,通过初始化数据连接,创建可用的数据库连接池;然后,获取到集合的信息,并对数据集进行有效的划分,划分的方法可以是根据主键进行划分、可以根据不同的条件进行划分,该划分为逻辑意义上的划分;最后各个并行执行路径通过执行各自的数据子集进行读取和写入,实现了真正的并行读写流程。

通过重写 getSplits(),可以根据文档属性进行划分。本文根据数据集的特有属性 id 进行对数据集的切分,首先通过获取整个集合的信息,得到集合大小,然后通过 id 属性切分成多个子集,每个数据集分块都记录了各自子集的大小、游标信息等。多个并行任务通过读取不同的数据集分块最终合并为一个完整的数据集,可以不同程度地提高读取的效率。

基于上述的设计方案,本文提出了基于 Flink 与 MongoDB 数据库连接中间件系统,该系统由应用层、数据存储层、业务逻辑层组成。基于 Flink 与 MongoDB 数据库连接中间件系统结合了 Flink 大数据处理平台的优点,包括了并行化任务,提供了超大规模计算能力,底层数据存储层采用了 MongoDB 数据库,MongoDB 作为一种典型的 NoSQL 数据库,在超大规模数据集上有很好的并发读写能力,并且能够完成海量数据的高效存储,同时最主要应用在于 MongoDB 数据库提供了分区功能,开发者能够将 MongoDB 集群进行分区。上层应用层是基于 Flink 开发的应用程序,能够充分地利用大数据处理平台所提供的超大规模计算能力,提供良好的服务。中间层是连接器中间件系统,在上层的应用层和底层的数据存储层起到连接作用,包括提供了四大模块:配置模块、读模块、写模块、查询模块。系统架构图如图 2 所示。

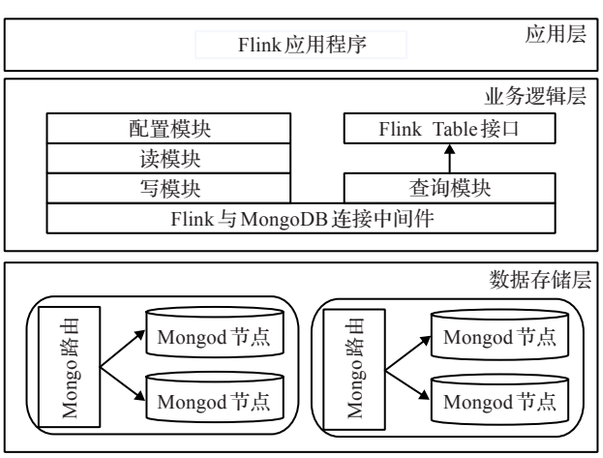

图 2　系统架构图

应用层的 Flink 应用提供了并行化计算框架和计算能力,MongoDB 集群数据库将数据进行切片管理,保证了中间件能够将并行化充分利用,提高整个数据的读写速度,中间件是为上层的应用程序提供了快速读写 MongoDB 数据库的保证。连接中间件作为整个架构的核心部分,基于 Flink 与 MongoDB 数据库的连接中间件的机制分为:初始化配置模块、读模块、写模块、查询模块。整个过程是并行化处理的过程.

3.2　中间件的运行机制

Flink 提供了一个 Hadoop 与 MongoDB 的连接器,通过配置 mongo.input.uri、mongo.output.uri 配置数据库地址,获取数据读写服务。该连接器使用 Hadoop 的序列化类 Writable 进行数据序列化和反序列化,读写过程需要进行数据类型转换工作,使得读写效率不高。本文提供的中间件基于 MongoDB 特有的存储结构 bson 进行对数据的序列化与反序列化。本文提出的中间件建立了配置模块,通过设置 servername、port、databasename、collectionname 即可连接到指定的数据集,方便进行配置。建立了读模块和写模块,实现 Flink 的通用读写接口 Inputformat 和 Outputformat,实现了并行化读写数据的功能,同时还增加了查询模块,查询模块是封装使用了 Flink Table 所给予的 sql 查询功能,当面对复杂的查询问题时,通过使用 FlinkTableAPI 可以简单地完成数据筛选的过程。具体有以下几个模块。

3.2.1　配置模块

配置模块包括了完成对存储系统的配置,中间件系统初始化创建了一个 MongoDB 客户端,用于与数据库之间建立连接客户端,提供集合的相关状态信息,包括:集合中的文档数、集合的状态、集合的大小等。该模块是其余模块的初始化模块,每个连接操作都需要初始化配置模块,才能进行相应的读、写操作等。配置信息包括了数据库服务器地址、数据库名字、数据集合名。配置模块伪代码如下所示:

```
输入:
//数据库地址、端口、数据库名、集合名称
servername,port,databasename,collectionname
初始化:
//根据 servername 和 port 获取 Mongo 客户端
mongoclient=MongoClient(servername,port);
//通过 mongoclient 和 databasename 获取指定数据库对象
mongodatabase=getDatabase(databasename);
//对指定的数据库通过 collectionname 获取指定数据集对象
Mongocollection=getCollection(collectionname)
输出:
//数据集对象
mongocollection
```

3.2.2　读模块

连接中间件系统支持高并发的读取数据,Flink 应用程序连接 MongoDB 通过读模块获取数据库中的数据

集。首先通过createInputSplites()按照集合的文档数通过平均切分将集合进行逻辑分片，分片保存了各个分片包含的数据的原信息包括了：集合信息、游标位置等。然后连接到MongoDB数据库各个分片，通过nextRecord()遍历整个分片，获取到分片集合的内容，每一次读取数据后，通过reachend()判断是否读取完所有数据，再进行下一次读取；最后读取完集合内所有的数据之后，关闭连接。Flink应用程序会自动地将所有读的记录归并到Flink数据集中。模块伪代码如下所示：

```
输入：//数据集对象
mongocollection
读取：
//获取数据集记录大小
nums=getCount(mongocollection)
//切分数据集，得到数据集分片
Splits[]=createInputSplits(collectionname，parall)
//读取数据集分片，得到记录
document=open(Inputsplit)
if reachEnd()//没有读取完成
   document=nextRecord()
else              //读取完成
   return；
输出：//数据集的所有记录
document
```

3.2.3　写模块

连接中间件系统支持高并发多线程的写数据，该模块完成了数据的写操作，可以将当前数据集插入到指定的数据库集合中。Flink程序会默认地保存数据存在的分区信息或将集合重新进行切分，通过调用writeRecord()，多分区同时进行写操作，实现并行化写操作，较于传统的单线程操作，在写模块中，充分地利用了Flink大数据处理平台并行化处理任务的能力，极大地加快了写数据的效率。写入模块伪代码如下所示：

```
输入：//写入的数据集名称、数据集
mongocollection，document
初始化：
//对指定的数据集进行写入数据
if reachEnd()  //没有写入完成
   writeRecord（document）//写入document
else              //写入完成
   return
输出：//无
```

3.2.4　查询模块

基于Flink与MongoDB的连接中间件，对于非结构化的数据库MongoDB，可以实现类似结构化数据库的SQL查询功能，该模块基于Flink的TableAPI提供的对数据集合的查询功能，集合相当于MySQL数据库中的表，可以直接对该集合使用SQL语句进行查询和对MySQL数据库查询一样，可以只关注SQL语句的书写，而且TableAPI也对SQL查询的一些方法进行了封装提供了接口方法，包括Select、Where、GroupBy等SQL的查询方法。查询模块是一个并行化执行过程，而当前的查询操作更多的是进行单线程操作，直接对全表进行查询，是一个串行化的执行过程。多线程对数据集合进行操作，极大地提高了查询的速率。

```
输入：//数据集
dataset
初始化：
//对数据集进行注册为一个表格
table=registerDataSet(dataset)
//对table进行Sql查询
querydataset=Sql(table，expression)；
输出：//查询后的数据集
querydataset
```

4　实验与结果分析

下面本文就以Flink连接MongoDB，对数据的导入和导出，对单机数据库、Flink官方提供的中间件和本文提出的Flink与MongoDB连接的中间件进行实验分析对比。

4.1　实验环境

实验数据集分别为100万条水位数据集、200万条水文数据、400万条水文传感器数据。

实验环境是单机节点的伪分布式集群模式，操作系统为Windows10环境下，处理器为Intel® Core™ i5-3230M CPU @ 2.60 GHz，内存是8 GB，硬盘是ST500LT012-9WS142（500 GB），显卡为华硕的NVIDIA GeForce GT 720M。

选用的Flink的版本为Flink1.4.1，MongoDB的版本为3.4。

4.2　实验内容

实验内容分为连接MongoDB数据库，对数据库进行插入数据和查询数据，所有的操作都分为在单机数据库下执行、基于Flink内置的MongoDB数据库连接中间件下执行以及基于本文提出的Flink连接MongoDB数据库连接中间件下执行。

（1）导入所有的数据到数据库耗时如图3所示。

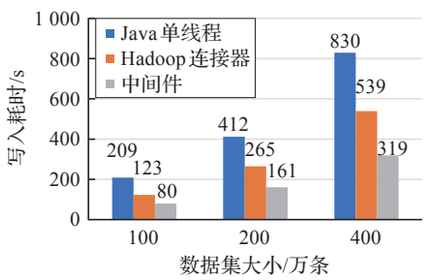

图3　数据导入数据库耗时

从图3中可以看到使用Java单线程操作导入100万条数据耗时为209 s,使用Hadoop连接器连接MongoDB耗时为123 s,而使用本文所提供的连接器只需要80 s,可以看到本文提供的Flink-MongoDB中间件在效率上具有明显的优势。当数据规模成倍增长时,导入数据的时间都成倍数增加,说明当前数据库的插入速度在100万条数据时进行插入已达到上限。

首先比较Java单线程和本文提供的基于Flink与MongoDB连接器中间件,单线程插入数据是串行方式,而本文的连接器中间件基于并行化的思想设计,由于MongoDB是支持高并发读写的NoSQL数据库,所有的线程是安全的,故采用并行的方式能够极大地提高写入数据库的效率。

再比较基于Flink官方提供的基于Hadoop的连接器,通过使用Hadoop连接器包装,该连接方式是并行化从数据库中读取数据,但是所耗费的时间依然要远大于本文提出的连接中间件。基于Hadoop的连接器耗费了大量的时间对数据进行并行化处理。由于基于Hadoop的连接器作为一个通用型的NoSQL数据库连接器,连接器提供了并行化的处理方式,它能够应对于其他的NoSQL数据库,所有的连接器获取的数据类型都是Hadoop的数据类型Writable,所以Hadoop连接器将花费大量的时间用于对数据类型的转换。而本文提出的基于Flink与MongoDB连接中间件针对于MongoDB数据库而进行设计的。基于Flink所提供的特性和MongoDB数据库类型的定义,可以充分地利用Flink提供的计算框架并行化地将数据库读写操作变得更加容易和简单,并且可以从实验数据上看到提升了近1.5倍的处理速度。

(2)查询水位值小于40的记录耗时如图4所示。

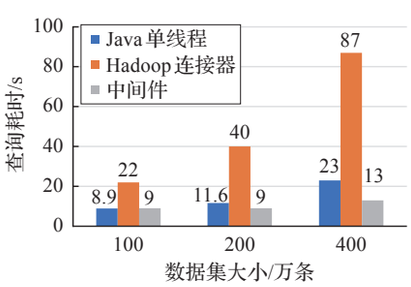

图4　查询水位值小于40的记录

从图4中可以看到单线程查询100万条记录中水位值小于40的记录耗时为8.9 s,使用本文提出的基于Flink与MongoDB连接中间件的方式耗时则为9 s;当数据量增大到200万时,单线程查询耗时11.6 s,中间件耗时为9 s,中间件耗时不变,但数据量增大到400万条时,单线程查询耗时为23 s,中间件耗时为13 s,而使用Hadoop连接器连接MongoDB查询水位值小于40的记录一直较高,在时间上可以看到基于Flink与MongoDB连接中间件的高效性。

首先比较单线程操作和基于Flink与MongoDB连接中间件。当数据集规模越来越大时,达到400万数据规模时,单线程操作在大数据集上直接进行条件查询,利用官方提供的MongoDB-Java-Connector中定义了对数据库的检索方式,通过使用过滤的方式,获取到所有的水位值小于40的记录。而本文提出的基于Flink与MongoDB连接中间件的方式则是通过读模块首先获取到所有的数据,不同于单机数据库串行化的执行任务,通过MongoDB分块技术,在读取数据的过程中,任务通过并行化执行,极大地缩减了读取所有数据所消耗的时间,再通过基于Flink自带的类似结构化数据库的查询方式,对所有数据进行查找。前者在检索数据库的过程中,进行查找,得到的数据集是符合条件筛选的数据集,后者则是首先通过读取所有数据,通过大数据平台所赋予的强大的计算能力,对结果进行筛选,得到结果集。从实验数据上可以看到时间提高了一倍之多。并且,基于MongoDB的数据库的检索方式所提供的API只有有限的一些条件查找方式,而使用Flink的类似结构化数据库的TableAPI中,可以提供包括SQL语句中所有的查找方式,并且可以通过直接写SQL语句对数据进行选择,极大地方便了在数据处理方面的工作。

再比较基于Flink与MongoDB连接中间件和使用Hadoop连接器连接MongoDB数据库。同样是通过并行化读取所有数据,再通过Flink提供的TableAPI进行筛选,基于Flink与MongoDB连接中间件的耗时总小于使用Hadoop连接器连接MongoDB数据库进行读取。基于Hadoop的连接器连接MongoDB数据库在读取过程中,同样通过逻辑分块并行读取,但是由于Hadoop连接器是一个通用型的连接器,内部存在了冗余的机制。在使用Hadoop连接器时调用了Mongo-Hadoop包中的输入输出格式,而Hadoop和Flink的计算框架不同,必然导致了运行效率的降低。而本文提出的Flink与MongoDB数据库连接中间件是结合了MongoDB-Java-Driver进行设计的,无需花费大量时间进行数据转换处理,同时不存在通用型连接器所存在的冗余问题,可以充分调用Flink所赋予的强大的计算能力,从实验数据上可以看到当数据规模越大时,查询效率将越来越高。

5　结束语

本文概括了基于Flink与MongoDB连接中间件系统的系统架构及各个组件的特性,包括了Flink大数据处理框架、MongoDB数据库和基于Hadoop的连接器连接MongoDB,并通过实验比较,可以看到本文提出了基于Flink与MongoDB连接中间件,从读、写、查询,整个过程完全并行化执行,效率得到显著提高。通过Flink

大数据平台所提供的超大规模计算能力，充分地利用了MongoDB所具备的特性，在Flink连接MongoDB数据库连接弥补了这片空白，并且本文提出的中间件不受操作系统的限制，且兼容了主流的Flink和MongoDB版本，使用了通用的接口实现，故本中间件具有很强的扩展性。

未来，基于Flink连接MongoDB中间件将扩展与Flink的流处理整合，并且会对该中间件与多种文档型数据库进行整合，提供通用的数据接口为Flink提供各种各样的数据存储服务。

参考文献：

[1] Lei D，Guo D，Chen C，et al.Vector spatial data cloud storage and processing based on MongoDB[J].Geo-Information Science，2014，16(4)：507-516.

[2] Dai C，Ye Y，Liu T J，et al.Design of high performance cloud storage platform based on cheap PC clusters using MongoDB and Hadoop[J].Applied Mechanics & Materials，2013：2050-2053.

[3] Grolinger K，Hayes M，Higashino W A，et al.Challenges for MapReduce in big data[C]//2014 IEEE World Congress on Services，2014：182-189.

[4] Plugge E，Hawkins T，Membrey P.The definitive guide to MongoDB：the NoSQL database for cloud and desktop computing[M].[S.l.]：Apress，2010.

[5] 曾强，缪力，秦拯.面向大数据处理的Hadoop与MongoD整合技术研究[J].计算机应用与软件，2016(2)：21-24.

[6] Abouzied A，Bajda-Pawlikowski K，Huang J，et al.HadoopDB in action：building real world applications[C]//ACM SIGMOD International Conference on Management of Data，2010：1111-1114.

[7] Abouzeid A，Bajda-Pawlikowski K，Abadi D，et al.HadoopDB：an architectural hybrid of MapReduce and DBMS technologies for analytical workloads[J].Proceedings of the VLDB Endowment，2009，2(1)：922-933.

[8] 陈德森，杨祖元.基于MongoDB的文本分类研究[J].无线互联科技，2017(5)：96-98.

[9] Hajoui O，Talea M.Which NoSQL database to-combine with spark for real time big data analytics?[J].International Journal of Computer Science & Information Security，2018，16(1).

[10] Junghanns M，Teichmann N，Rahm E.Analyzing extended property graphs with Apache Flink[C]//ACM SIGMOD Workshop on Network Data Analytics，2016.

[11] White T，Cutting D.Hadoop：the definitive guide[M].[S.l.]：O'Reilly Media Inc，2012.

[12] Zaharia M，Chowdhury M，Franklin M J，et al.Spark：cluster computing with working sets[C]//USENIX Conference on Hot Topics in Cloud Computing，2010.

[13] Lakshman A，Malik P.Cassandra：a decentralized structured storage system[J].ACM SIGOPS Operating Systems Review，2010，44(2)：35-40.

[14] George L.HBase：the definitive guide[M].[S.l.]：O'Reilly Media，Inc，2011.

[15] Chodorow K.MongoDB：the definitive guide[M].[S.l.]：O'Reilly Media，Inc，2013.

[16] Plunkett T，Macdonald B，Nelson B，et al.Oracle big data handbook[M].[S.l.]：McGraw-Hill Osborne Media，2013.

[17] Oracle.Oracle big data connectors[DB/OL].[2018-05-21].http：//www.oracle.com/technetwork/bdc/.

[18] 10gen.MongoDB connector for Hadoo[DB/OL].[2018-05-21].https：//docs.mongodb.com/ecosystem/tools/hadoop/.

[19] HORTONWORKS.Spark on HBase：data frame based HBase connector[DB/OL].[2018-05-21].https：//zh.hortonworks.com/blog/spark-hbase-dataframe-based-hbase-connector/.

[20] 10gen.MongoDB connector for Apache Spark[DB/OL].[2018-05-21].https：//www.mongodb.com/products/spark-connector.

[21] Farooqui S.Spark+Cassandra：technical integration details[Z].O'Reilly Media Free，Live Events，2018.

[22] DATASTAX.Accessing Cassandra from Spark in Java[DB/OL].[2018-05-21].https：//www.datastax.com/dev/blog/accessing-cassandra-from-spark-in-java.

[23] Deshpande T.Learning Apache Flink[M].Birmingham，UK：Packt Publishing Ltd，2017.