Spark企业级大数据项目实战 第6课

DATAGURU专业数据分析社区

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

本课内容

- ElasticSearch核心基础
- Spark 整合Elasticsearch要点、案例实操
- EasticSearch实现Exactly-once语义
- 四种方案对比
- Spark 整合Elasticsearch性能优化

ElasticSearch核心基础

ElasticSearch是什么？

1. 基于Lucene的面向文档的搜索引擎
2. 分布式、可扩展
3. 提供了支持JSON的RESTful接口

ElasticSearch集群的角色

Elasticsearch是主从架构，集群的节点主要有主节点（master ）和数据节点（data）。

但是Elasticsearch集群又是具备去中心化的特性， 因为可以通过任意节点都可以同Elasticsearch集群通信， 并且是等价的。

1.2 安装ElasticSearch及HEAD插件

ElasticSearch安装

1. 下载ElasticSearch
2. 配置ElasticSearch
3. 启动ElasticSearch

安装过程错误解决

Head插件

1. Head插件下载
2. 安装node
3. 安装grunt
4. 修改Elasticsearch配置
5. 启动Head插件和ES服务

1.3 观念转换--类比关系型数据库

1. Index（索引）

类似关系型数据库的Schema。
索引可以动态创建和删除

2. Type（类型）

相同格式文档的集合。 类比关系型数据库的表

3. Document（文档）

类似关系型数据库的一条记录（行）

4. Field（字段）

关系型数据库的字段， 对应列

1.3 Elastisearch相关概念--节点类型

Elasticsearch是主从架构， 但是Elasticsearch集群又是具备去中心化的特性， 因为可以通过任意节点都可以同Elasticsearch集群通信， 并且是等价的。

Elasticsearch集群的节点主要有**主节点（master ）和数据节点（data）**。

主节点

主节点负责集群的状态变更， 包括：增删节点， index（索引）、mapping（映射）的管理， 副本的管理， 分片的重分配等。为了防止脑裂， 通常设置多个主节点。

在生产中，主节点是独占一台服务器， 这个服务器不会存储数据。 因为如果主节点的负载过重， 有可能导致主节点不能提供服务， 甚至导致脑裂。

主节点独占一台服务器的配置：

```
node.master: true
node.data:false
```

数据节点

数据节点用于存放数据， 也就是存放lucene索引的。 数据节点在生产环境也是配置成独占的， 配置如下：

```
node.master: false
node.data:true
```

1.3 Elastisearch相关概念--节点发现

Elasticsearch集群是通过cluster.name配置集群的名称， 所有具有相同cluster.name的节点组成一个集群。
Elasticsearch通过discover的方式自动发现节点， 并把节点加入到集群中。

节点发现有两种方式: 组播和单播。

□ 组播： 每个节点实例向指定的多播组和端口发送多播的ping请求， 每个节点响应请求， 当找到主节点， 就将这个节点实例接入集群。 如果多播没有发现主节点， 集群会选择一个主节点。
在生产环境中， 一般不推荐使用组播的方式， 因为可能会将不相干的节点加入到集群。
discovery.zen.ping.multicast.enabled=false

□ 单播： 只向配置好的主机列表和端口发送请求。

```
discovery.zen.ping.unicast.hosts:
  - 192.168.1.10:9300
  - 192.168.1.11
  - seeds.mydomain.com
```

注意： 不需要把集群中所有的节点都配置上， 新的节点会根据配置的主机和端口通信， 只要发现了某个集群， 就会加入到这个集群。 但是从高可用考虑，节点也不能配置的太少。

1.3 Elastisearch相关概念--分片/副本

分片（Shard）

Elasticsearch提供了将索引划分成多片的能力，这些片叫做分片。当你创建一个索引的时候，你可以指定你想要的分片的数量。每个分片本身也是一个功能完善并且独立的“索引”，这个“索引”可以被放置到集群中的任何节点上。

一个索引可以有多个分片（shard）， 每个shard就是一个lucene索引。

从集群分布式角度， 可以把分片类比Kafka中topic的分区（partition）。

分片很重要， 主要有两个方面的原因：

（1）、允许你水平分割/扩展你的内容容量

（2）、允许你在分片（位于多个节点上）之上进行分布式的、并行的操作，进而提高性能/吞吐量

副本

在一个网络/云的环境里，失败随时都可能发生。在某个分片/节点因为某些原因处于离线状态或者消失的情况下，故障转移机制是非常有用且强烈推荐的。为此， Elasticsearch允许你创建分片的一份或多份拷贝，这些拷贝叫做复制分片，或者直接叫复制。

复制之所以重要，有两个主要原因：

（1）、在分片/节点失败的情况下，复制提供了高可用性。复制分片不与原/主要分片置于同一节点上是非常重要的。

（2）、因为搜索可以在所有的复制上并行运行，复制可以扩展你的搜索量/吞吐量

1.3 Elastisearch相关概念--分片/副本

默认情况下，Elasticsearch中的每个索引分配5个主分片和1个复制。这意味着，如果你的集群中至少有两个节点，你的索引将会有5个主分片和另外5个复制分片（1个完全拷贝），这样每个索引总共就有10个分片。

注意： 分片可以在创建index的时候指定，指定后就不能修改。 副本可以修改。

每个shard的最大数据量：2,147,483,519 （20亿）
链接： https://www.elastic.co/guide/en/elasticsearch/reference/5.6/_basic_concepts.html

分片设置：

```
curl  -XPUT  http://spark1234:9200/myshard/ -d '{
    "settings": {"number_of_shards":2, "number_of_replicas":0}
}'
```

设置副本数：

```
curl -XPUT  http://spark1234:9200/myindex/_settings -d '{
    "number_of_replicas": 0
}'
```

1.4 Elastisearch相关概念--倒排索引

- 正排索引：

文档ID	文档内容	分词
1	spark provides high-level APIs in Java, Scala, Python and R	spark,provides,high-level,APIs,in,Java,Scala,Python,and,R
2	Scala and Java users can include Spark in their projects using Java Maven coordinates	Scala,and,Java,users,can,include,Spark,in,their,projects,using,its,Maven,coordinates
3	spark currently provides several options for spark deployment	spark,currently,provides,several,options,for,deployment

通过文档的id， 可以确定这个文档包含了哪些单词。 但是， 如果需要查看某个单词（比如spark）在哪个文档出现过， 那就需要遍历所有文档了。

1.4 Elastisearch相关概念--倒排索引

- 倒排索引：

单词	文档ID
spark	1,2,3
Java	1,2
Scala	1,2
provides	1,2
Python	1
high-level	1
APIs	1
in	1,
and	1,2
...	...

文档ID	文档内容	分词
1	spark provides high-level APIs in Java, Scala, Python and R	spark,provides,high-level,APIs,in,Java,Scala,Python,and,R
2	Scala and Java users can include Spark in their projects using Java Maven coordinates	Scala,and,Java,users,can,include,Spark,in,their,projects,using,its,Maven,coordinates
3	spark currently provides several options for spark deployment	spark,currently,provides,several,options,for,deployment

可以快速定位单词在哪个文档里面出现和位置信息。

1.5 Elastisearch操作实践

基本操作

创建文档、 插入数据、获取文档、更新文档、创建索引

Mapping

Mapping操作、动态Mapping、显式Mapping

索引模板

创建模板、删除模板、根据模板创建index

ElasticSearch查询语句

URL、match_all、terms、Boolean、match、multi_match、range、wildcard、过滤器查询

聚合分析

terms聚合、 histogram聚合、range聚合、avg/sum聚合、嵌套聚合

Spark 整合Elasticsearch要点、案例实操

maven配置

```
<dependency>
  <groupId>org.elasticsearch</groupId>
  <artifactId>elasticsearch-spark-13_2.10</artifactId>
  <version>5.6.4</version>
</dependency>
```

2.1 Spark 整合Elasticsearch-配置

配置ES相关的Spark属性

1. 在代码中配置

```
import org.apache.spark.SparkConf
val conf = new SparkConf().setAppName(appName).setMaster(master)
conf.set("es.index.auto.create", "true") // 或者  conf.set("spark.es.index.auto.create", "true")
```

2. 提交时候传入配置

```
spark-submit \
  --conf  "spark.es.resource=index/type"  \
  --conf  "spark.es.index.auto.create=true"  \
...
```

注意： 通过spark-shell或者spark-submit设置属性，spark只接受以”spark”前缀开头的属性， 忽略非spark前缀的属性。

因此通过**spark**配置**elasticsearch-hadoop**属性， 必须在属性前面加上**spark.**前缀。

上面的例子， **es.index.auto.create**属性变成了**spark.es.index.auto.create**

2.2 Spark 整合Elasticsearch-写入

使用elasticsearch-hadoop，只要RDD能被转换成Elasticsearch的文档（documents）， 这个RDD就可以写入Elasticsearch。

实际上， 能写入ES的RDD的类型是如下几种：

1. Map(无论是使用Java还是Scala)
2. JavaBean
3. Scala的Case Class

Spark的DataFrame也可以写入Elasticsearch

2.2 Spark 整合Elasticsearch-写入案例1

```
import org.elasticsearch.spark._

sparkConf.set("spark.es.nodes","spark1234") // 配置es的主机名
  .set("spark.es.port","9200").set("spark.es.index.auto.create", "true") //  可以去掉spark.前缀

 val sc = new SparkContext(sparkConf)

val numbers = Map("one" -> 1, "two" -> 2, "three" -> 3)
val airports = Map("arrival" -> "Otopeni", "SFO" -> "San Fran")
sc.makeRDD(Seq(numbers, airports)).saveToEs("spark/docs")
```

2.2 Spark 整合Elasticsearch-写入案例2

```
import org.elasticsearch.spark.rdd.EsSpark

sparkConf.set("es.nodes","spark1234")
      .set("es.port","9200").set("es.index.auto.create", "true")

    val sc = new SparkContext(sparkConf)

    // define a case class
    case class Trip(deptid: String, departure: String, arrival: String)

    val upcomingTrip = Trip("d001", "OTP", "SFO")
    val lastWeekTrip = Trip("d002", "MUC", "OTP")

    val rdd = sc.makeRDD(Seq(upcomingTrip, lastWeekTrip))
   EsSpark.saveToEs(rdd, "spark1/docs")
```

_index	_type	_id	_score ▲	deptid	departure	arrival
spark1	docs	AWHsv20vppGQjnoWYdJj	1	d002	MUC	OTP
spark1	docs	AWHsv203ppGQjnoWYdJk	1	d001	OTP	SFO

指定文档ID

对于需要指定文档的id（或其他元数据字段，如ttl或timestamp）的情况，可以通过设置元数据属性字段es.mapping.id来实现。 在前面的例子之后，为了向Elasticsearch指示使用字段id作为文档ID，需要更新RDD配置（也可以SparkConf上设置属性， 但是由于它的全局效果， 因此不推荐使用这种方式）：

```
EsSpark.saveToEs(rdd, "spark2/docs",Map("es.mapping.id" -> "deptid"))
```

查询 5 个分片中用的 5 个. 2 命中. 耗时 0.002 秒

_index	_type	_id	_score ▲	deptid	departure	arrival
spark2	docs	d001	1	d001	OTP	SFO
spark2	docs	d002	1	d002	MUC	OTP

2.2 Spark 整合Elasticsearch-写入案例4

Json入Elasticsearch: 通过专用的saveJsonToEs方法索引JSON数据

```
import org.elasticsearch.spark._

sparkConf.set("es.nodes","spark1234")
.set("es.port","9200").set("es.index.auto.create", "true")

val json1 = """{"reason" : "business", "airport" : "SFO"}"""
val json2 = """{"participants" : 5, "airport" : "OTP"}"""

new SparkContext(sparkConf).makeRDD(Seq(json1, json2)) .saveJsonToEs("sparkjson/json-trips")
```

查询 5 个分片中用的 5 个. 2 命中. 耗时 0.045 秒

_index	_type	_id	_score ▲	reason	airport	participants
sparkjson	json-trips	AWHsxSanppGQjnoWYdJs	1	business	SFO	
sparkjson	json-trips	AWHsxSe2ppGQjnoWYdJt	1		OTP	5

2.2 Spark 整合Elasticsearch-写入案例5

动态/多资源写入Elasticsearch:

如果写入Elasticsearch的数据， 需要基于数据的内容索引到不同的buckets， 可以使用es.resource.write字段，该字段在运行时接受从文档内容解析的模式。

```
val game = Map("media_type"->"game","title" -> "FF VI","year" -> "1994")
 val book = Map("media_type" -> "book","title" -> "Harry Potter","year" -> "2010")
 val cd = Map("media_type" -> "music","title" -> "Surfing With The Alien")

 sc.makeRDD(Seq(game, book, cd)).saveToEs("my-collection/{media_type}")
```

数据基于字段media_type写入到不同的ES的不同type下， 需要注意的是所有的文档（document）都需要有media_type这个字段。

_index	_type	_id	_score ▲	media_type	title	year
my-collection	music	AWHszYYQppGQjnoWYdJy	1	music	Surfing With The Alien	
my-collection	game	AWHszYaappGQjnoWYdJz	1	game	FF VI	1994
my-collection	book	AWHszYYQppGQjnoWYdJx	1	book	Harry Potter	2010

2.2 Spark 整合Elasticsearch-写入案例6-1

处理文档元数据:

Elasticsearch允许每个document都有自己的元数据。 通过各种mapping选项可以定义这些参数。 在Spark中，elasticsearch扩展了该功能， 允许使用pair RDD的方式在文档本身之外提供元数据。 简单的说， 就是使用二元组的方式提供原数据， 第一个元素是元数据信息， 第二个数据是文档数据。

元数据使用org.elasticsearch.spark.rdd包下的Metadata枚举类描述， 可以标识的类型有：ID,PARENT,ROUTING,TTL,TIMESTAMP,VERSION,VERSION_TYPE。

Pair RDD的二元组， 第一个元素定义每个文档的元数据， 可以是Map类型或者非Map类型。如果不是Map类型， Elasticsearch将该对象视为文档的ID。

2.2 Spark 整合Elasticsearch-写入案例6-2

处理文档元数据: 手动指定文档ID

```
 val otp = Map("iata" -> "OTP", "name" -> "Otopeni")
val muc = Map("iata" -> "MUC", "name" -> "Munich")
val sfo = Map("iata" -> "SFO", "name" -> "San Fran")

val airportsRDD = sc.makeRDD(Seq((1, otp), (2, muc), (3, sfo)))
airportsRDD.saveToEsWithMeta("airports/2015")
```

手动指定每个文档的id， 只需传入RDD中的Object（不是Map类型）， 这里sc.makeRDD(Seq((1, otp), (2, muc), (3, sfo)))是一个键值对RDD; 它通过元组的Seq创建， 分别代表id和与之关联的文档，换句话说， 文档otp的id是1， 文档muc的id是2。

查询 5 个分片中用的 5 个. 3 命中. 耗时 0.008 秒

_index	_type	_id	_score ▲	iata	name
airports	2015	2	1	MUC	Munich
airports	2015	1	1	OTP	Otopeni
airports	2015	3	1	SFO	San Fran

2.2 Spark 整合Elasticsearch-写入案例6-3

处理文档元数据: 指定更多属性

```
val otpMeta = Map(ID -> 1, TTL -> "3h")
val mucMeta = Map(ID -> 2, VERSION -> "23")
val sfoMeta = Map(ID -> 3)
val airportsRDD = sc.makeRDD(Seq((otpMeta, otp), (mucMeta, muc), (sfoMeta, sfo)))
airportsRDD.saveToEsWithMeta("airports2/2015")
```

```
[hadoop@spark1234 ~]$ curl -XGET http://spark1234:9200/airports2/2015/2?pretty
{
  "_index" : "airports2",
  "_type" : "2015",
  "_id" : "2",
  "_version" : 23,
  "found" : true,
  "_source" : {
    "iata" : "MUC",
    "name" : "Munich"
  }
}
```

2.2 Spark 整合Elasticsearch-写入案例7

Spark SQL的支持： 使用DataFrame的方式写入Elasticsearch

也可以将json的数据写入elasticsearch, 使用spark的外部数据源将json格式的数据转换成DataFrame即可：sqlContext.read.format(“json”).load(json文件的路径或者json的RDD)

```
val dataSet = List(("021", "54657", "Apple.com"), ("023", "64780", "hp.com"), ("010", "23567", "Google.com"))
import sqlContext.implicits._
val df = sc.parallelize(dataSet).map(x=>(x._1, x._2, x._3)).toDF("areacode", "code", "companyname")
df.saveToEs("company/info")
```

2.3 Spark 整合Elasticsearch-读取1

从Elasticsearch读取数据

使用spark读取elasticsearch数据， 应定义Elasticsearch的RDD， 将数据传输给spark

- 方式一： 通过RDD的方式读取Elasticsearch数据
  - 为索引radio/artists创建专用的elasticsearch RDD

```
val RDD = sc.esRDD("radio/artists")
```

  - 指定额外的查询或者添加一个Map的配置

为索引radio/artists创建一个从所有文档字段匹配me*的数据

```
sc.esRDD("radio/artists", "?q=me*")
```

默认情况下， Elasticsearch的文档返回的是一个二元组Tuple2， 第一个元素是文档id， 第二个字段是用scala集合表示的实际文档， 即Map[String, Any], 其中key为字段名称， 值是字段的存储的值。

RDD读取数据示例：

```
(1,Map(iata -> OTP, name -> Otopeni))
(3,Map(iata -> SFO, name -> San Fran))
(2,Map(iata -> MUC, name -> Munich))
```

2.3 Spark 整合Elasticsearch-读取2

从Elasticsearch读取数据

使用spark读取elasticsearch数据， 应定义Elasticsearch的RDD， 将数据传输给spark

- 方式二： 以Json的方式读取数据

如果从Elasticsearch读取的数据需要转换为json格式（通常是将读取的数据发送到其他的系统），可以使用专有的esJsonRDD方法。 当使用这个方法， connector将从Elasticsearch读取的RDD（scala的RDD[(String, String)] 或者Java中的JavaPairRDD[String, String] ）的文档内容返回不做任何处理。其中key为文档id， 值为json格式的文档的实际内容

JSON格式读取数据示例：

```
(1,{"iata":"OTP","name":"Otopeni"})
(2,{"iata":"MUC","name":"Munich"})
(3,{"iata":"SFO","name":"San Fran"})
```

2.3 Spark 整合Elasticsearch-外部数据源读取1

1. 外部数据源读取样式

```
val options = Map("pushdown" -> "true", "es.nodes" -> "spark1234", "es.port" -> "9200")

val spark14DF = sqlContext.read.format("org.elasticsearch.spark.sql").options(options).load("company/info")

sqlContext.read.format("es").load("company/info")
```

2. 将数据源作为table

```
sqlContext.sql(
      "CREATE TEMPORARY TABLE myIndex    " +
       "USING org.elasticsearch.spark.sql " +
       "OPTIONS (resource 'company/info', scroll_size '20')" )
    sqlContext.sql("select * from myIndex").show
```

2.3 Spark 整合Elasticsearch-外部数据源读取2

pushdown

是否将Spark SQL语句翻译（下推）为Elasticsearch Query DSL.

使用push down， 在数据源就将数据过滤掉，这样只有需要的数据才被传回Spark， 这样能显著提高性能，并最大限度地减少Spark和Elasticsearch集群的CPU、内存、IO。

```
// as a DataFrame
val df = sqlContext.read().format("org.elasticsearch.spark.sql").load("spark/trips")

df.printSchema()
// root
//|-- departure: string (nullable = true)
//|-- arrival: string (nullable = true)
//|-- days: long (nullable = true)
val filter = df.filter(df("arrival").equalTo("OTP").and(df("days").gt(3))
```

```
{
  "query" : {
    "filtered" : {
      "query" : {
        "match_all" : {}

      },
      "filter" : {
        "and" : [{
            "query" : {
              "match" : {
                "arrival" : "OTP"
              }
            }
          }, {
            "days" : {
              "gt" : 3
            }
          }
        ]
      }
    }
  }
}
```

或者语句：

```
CREATE TEMPORARY TABLE trips USING org.elasticsearch.spark.sql OPTIONS (path "spark/trips")
SELECT departure FROM trips WHERE arrival = "OTP" and days > 3
```

2.3 Spark 整合Elasticsearch-使用DataFrame读取

使用**esDF 读取ES数据**

```
val sqlContext = new SQLContext(sc)
val company = sqlContext.esDF("company/info")

// check the associated schema
println(company.schema.treeString)
// root
// |-- areacode: string (nullable = true)
// |-- code: string (nullable = true)
// |-- companyname: string (nullable = true)
```

```
# include
es.read.field.include = *name, address.*
# exclude
es.read.field.exclude = *.created
```

指定参数查询：

```
val company = sqlContext.esDF("company/info", "?q=023" )
```

指定返回的列：

```
val company = sqlContext.esDF("company/info", "?q=023", Map("es.read.field.include" -> "companyname") )
```

2.4 Spark 整合Elasticsearch-使用HadoopAPI读取

DATAGURU炼数成金

1. 使用Hadoop的API读取Elasticsearch数据

```
sparkConf.set("spark.serializer",
      classOf[KryoSerializer].getName)

    val sc = new SparkContext(sparkConf)

    val conf = new JobConf()
    conf.set("es.resource", "company/info")
    //conf.set("es.query", "?q=me*")
    val esRDD = sc.hadoopRDD(conf,
      classOf[EsInputFormat[Text, MapWritable]],
      classOf[Text], classOf[MapWritable])
    val docCount = esRDD.count()
    println("docCount:" + docCount)
    esRDD.take(10).foreach(println)
```

2. 使用Hadoop的New API读取Elasticsearch数据

```
sparkConf.set("spark.serializer",
      classOf[KryoSerializer].getName)

    val sc = new SparkContext(sparkConf)

    val conf = new Configuration()
    conf.set("es.resource", "company/info")
    //conf.set("es.query", "?q=me*")
    val esRDD = sc.newAPIHadoopRDD(conf,
      classOf[EsInputFormat[Text, MapWritable]],
      classOf[Text], classOf[MapWritable])

    val docCount = esRDD.count();
    println("docCount:" + docCount)
    esRDD.take(10).foreach(println)
```

Elasticsearch的Spark-Streaming支持

当使用elasticsearch-hadoop Spark Streaming支持时，可以将Elasticsearch作为输出位置，将Spark Streaming作业中的数据索引到ES中，其方式与持久化RDD结果的方式相同。

Elasticsearh-hadoop的Spark Streaming对ES做了专门的优化， 在Spark的Executors上允许为非常小的处理窗口保留网络资源。 因此， 推荐使用Spark Streaming专门的方式整合ES， 而不是在DStream上使用从foreachRDD返回的RDD上调用SaveToES方法。

类似RDD， 使用Dstream，只要Dstream的内容能被转换成Elasticsearch的文档（documents）， 这个Dstream就可以写入Elasticsearch。

实际上， 能写入ES的Dstream的类型是如下几种：

1. Map(无论是使用Java还是Scala)
2. JavaBean
3. Scala的Case Class

2.5 Spark 整合Elasticsearch-Spark Streaming

Elasticsearch的Spark-Streaming支持

- Map方式写入
- Case Class写入
- 指定元数据
- 动态/多资源写入
- Json方式写入

```
val ssc = new StreamingContext(sc, Seconds(10))

val game = Map("media_type"->"game","title" -> "FF VI","year" -> "1994")
val book = Map("media_type" -> "book","title" -> "Harry Potter","year" -> "2010")
val cd = Map("media_type" -> "music","title" -> "Surfing With The Alien")

val batch = sc.makeRDD(Seq(game, book, cd))
val microbatches = mutable.Queue(batch)
ssc.queueStream(microbatches).saveToEs("streaming-collection/{media_type}")

ssc.start()
```

写入方式类似RDD的方式， 具体参见案例演示

3. EasticSearch实现Exactly-once语义

id设计：

```
val md5pre = md5(topic + "|" + groupName + "|" + part)
 val id = md5pre + "|" + String.format("%020d",java.lang.Long.valueOf(offset))
```

使用两种方式实现：

1. Spark Streming的foreachRDD方式

2. ElasticSearch-Hadoop的Streaming方式

详细参见代码实操

4. 四种方案对比

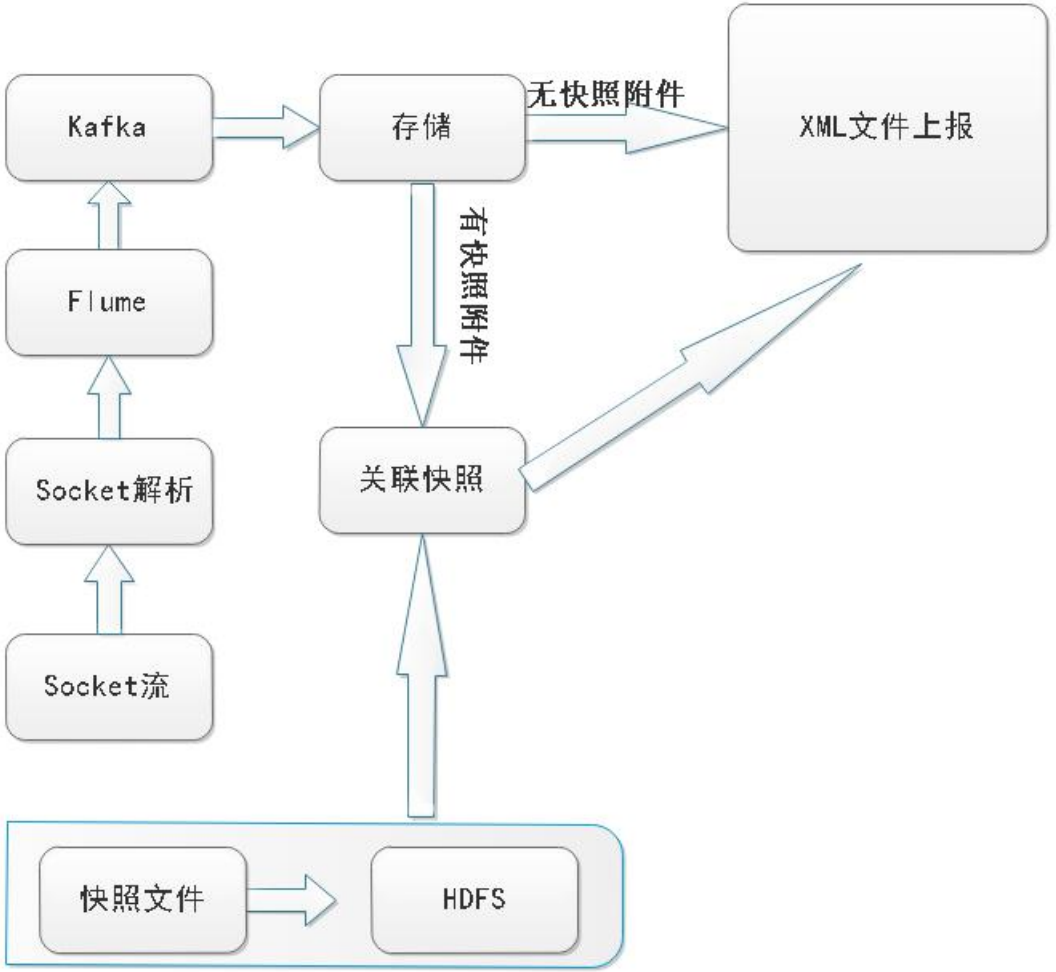
幂等写入（ idempotent writes）

rowkey设计

Kafka每条消息的partition、offset等获取

四种存储选择： Kafka、Oracle、Hbase、Elasticsearch

先写数据 ---》 保存offset



5. Spark + ElasticSearch性能优化1

- ElasticSearch生产集群配置

集群主机配置相近
建议使用SSD硬盘替换传统的机械硬盘
ES内存配置不要超过32G
集群名称和节点名称必须要配置
减少swap内存交换
副本、分片数量的规划
避免脑裂的配置
集群启动数据恢复相关参数配置
数据导入优化
关闭不必要的index
事务刷新阈值
索引缓存大小
使用独占的主节点和数据节点

5. Spark + ElasticSearch性能优化2

Spark整合Elasticsearch优化

- 读取性能

  增加shard数量是否可以增加读取性能？

  增加节点， 数据更好的跨越多个节点， 理想情况下数据跨越多个节点。

  如果一个index的多个shard在同一个机器上， 即仅仅是数据虚拟分区， 硬件保持不变。 虽然可以增加客户端读取的并行度， 因为更多的shard意味着更多的任务可以同时从elasticsearch读取数据， 但是从elasticsearch角度，没有实际效益， 因此性能可能保持不变。

- 写性能

  写入Elasticsearch是由spark的分区数量驱动。 elasticsearch-hadoop检测要写入主分片的数量， 并在这些shard之间分配写入。 因此， 可用的shard越多， 写入Elasticsearch的并行度越高。

  Elasticsearch-hadoop跨越所有的task并行写数据。 提升写性能的一个关键方面是Elasticsearch能无压力摄入数据的最大速率。 这取决于很多因素（数据大小、硬件、当前负载等）， 因此Elasticsearch集群自身的优化也非常重要。

5. Spark + ElasticSearch性能优化3

Spark整合Elasticsearch优化

- 减小bulk的大小

假设有T个任务，配置为B字节和N个文档（其中d是平均文档大小），则在给定时间点批量写入请求的最大数量可以是T * B个字节或 T * N个文档（T * N * d以字节为单位）。

对于具有5个任务的工作，使用默认值（1mb或1000文档）意味着最多5mb / 5000文档批量大小（散布在shard中）。 如果处理时间超过1-2秒，则无需减少bulk的大小。

elasticsearch-hadoop允许为每个任务配置批量写入Elasticsearch的条目数量和大小。

Kafka配置限速参数等。

5. Spark + ElasticSearch性能优化4

Spark整合Elasticsearch优化

- 限制写入Elasticsearch的任务数

使用elasticsearch-hadoop写数据到Elasticsearch，如果是RDD或者表的join会生成多个任务，这可能导致产生大量的Task, 即导致用户计划用于Elasticsearch的任务数与实际的任务数之间的数量不成比例， 有时可能高出1-2个数量级。

如果ES的集群只有几个节点，同时处理这么多的task将导致写非常缓慢。

解决： 减少Task数量， 减小源端的小文件， 使用coalesce或者repartition参数等。

coalesce或者repartition参数的使用， 在后续离线项目中介绍。

Thanks

FAQ时间