
在InfoQ的协调下，作者与国内多家公有云服务提供商的主要负责人进行了电话访谈，围绕团队建设、产品研发、服务运营这三个问题进行了讨论。除此之外，作者也在本文所探讨的所有公有云上都注册了账号，从用户体检的角度进行了一些小规模的测试。这篇文章的目的，便是从团队建设、产品研发、服务运营、用户体验等四个方面对中国的公有云服务发展状况做一个简要的综述。本篇是服务运营的上半部分。

浏览所有 文化 & 方法

- DevOps
  - 持续交付
  - 自动化操作
  - 云计算

特别专题 DevOps

2015年中国公有云服务发展报告——服务运营篇（上）


在InfoQ的协调下，作者与国内多家公有云服务提供商的主要负责人进行了电话访谈，围绕团队建设、产品研发、服务运营这三个问题进行了讨论。除此之外，作者也在本文所探讨的所有公有云上都注册了账号，从用户体检的角度进行了一些小规模的测试。这篇文章的目的，便是从团队建设、产品研发、服务运营、用户体验等四个方面对中国的公有云服务发展状况做一个简要的综述。本篇是服务运营的上半部分。

浏览所有 DevOps

- 架构
- 移动
- Docker
- 云计算
- 大数据
- 架构师
- 运维
- QCon
- ArchSummit
- 全球架构师峰会
- 极光
- 又拍云

全部话题

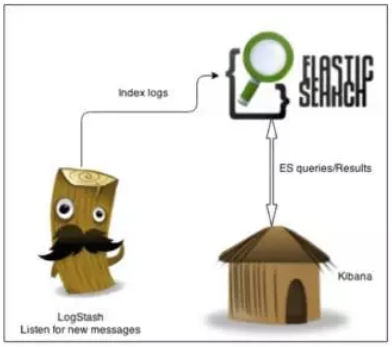
大数据杂谈微课堂|Elasticsearch 5.0新版本的特性与改进

作者 曾勇 发布于 2016年8月8日 | 被首富的“一个亿”刷屏？不如定个小目标，先把握住QCon上海的优惠吧！ 2 讨论

大家好，非常高兴能在这里给大家分享，首先简单自我介绍一下，我叫曾勇，是Elastic的工程师。Elastic将在今年秋季的时候发布一个Elasticsearch V5.0的大版本，这次的微信分享将给大家介绍一下5.0版里面的一些新的特性和改进。

5.0？ 天啦噜，你是不是觉得版本跳的太快了。

好吧，先来说说背后的原因吧。



相信大家都听说ELK吧，是Elasticsearch、Logstash、Kibana三个产品的首字母缩写，现在Elastic又新增了一个新的开源项目成员：Beats。

有人建议以后这么叫：ELKB?

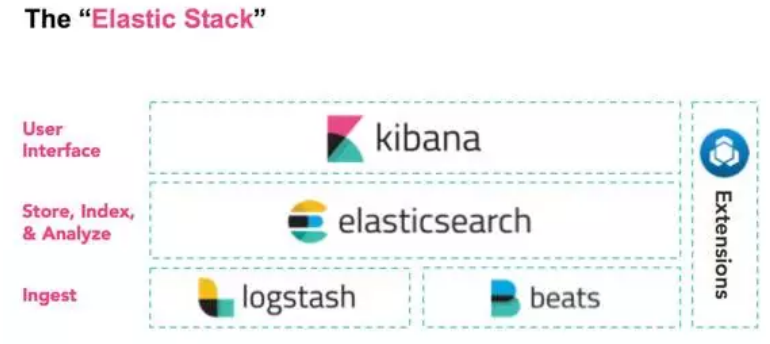
为了未来更好的扩展性:）ELKBS？ELKBSU？.....

所以我们打算将产品线命名为ElasticStack

同时由于现在的版本比较混乱，每个产品的版本号都不一样，Elasticsearch和Logstash目前是2.3.4；Kibana是4.5.3；Beats是1.2.3；

版本号太乱了有没有，什么版本的ES用什么版本的Kibana？有没有兼容性问题？

所以我们打算将这些的产品版本号也统一一下，即v5.0，为什么是5.0，因为Kibana都4.x了，下个版本就只能是5.0了，其他产品就跟着跳跃一把，第一个5.0正式版将在今年的秋季发布，目前最新的测试版本是：5.0 Alpha 4



目前各团队正在紧张的开发测试中，每天都有新的功能和改进，本次分享主要介绍一下Elasticsearch的主要变化。

Elasticsearch5.0新增功能

首先来看看5.0里面都引入了哪些新的功能吧。

首先看看跟性能有关的。

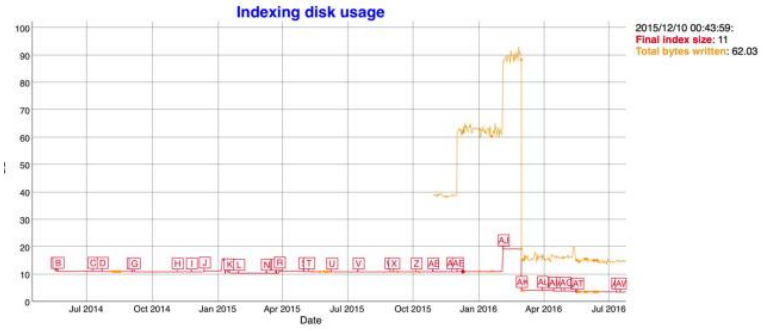
第一个就是Lucene 6.x 的支持。

Elasticsearch5.0率先集成了Lucene6版本，其中最重要的特性就是 Dimensional Point Fields，多维浮点字段，ES里面相关的字段如date, numeric，ip 和 Geospatial 都将大大提升性能。

这么说吧，磁盘空间少一半；索引时间少一半；查询性能提升25%；IPV6也支持了。

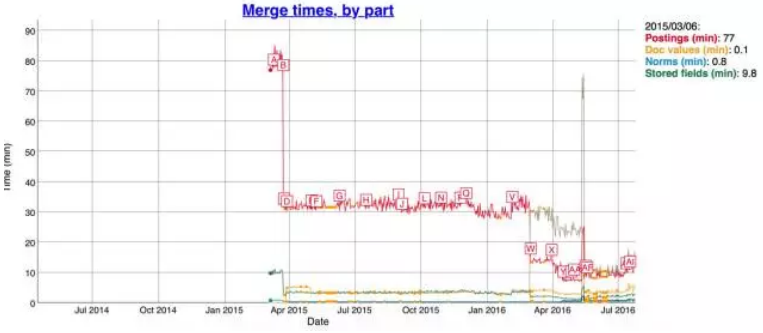
为什么快，底层使用的是Block k-d trees，核心思想是将数字类型编码成定长的字节数组，对定长的字节数组内容进行编码排序，然后来构建二叉树，然后依次递归构建，目前底层支持8个维度和最多每个维度16个字节，基本满足大部分场景。

说了这么多，看图比较直接。

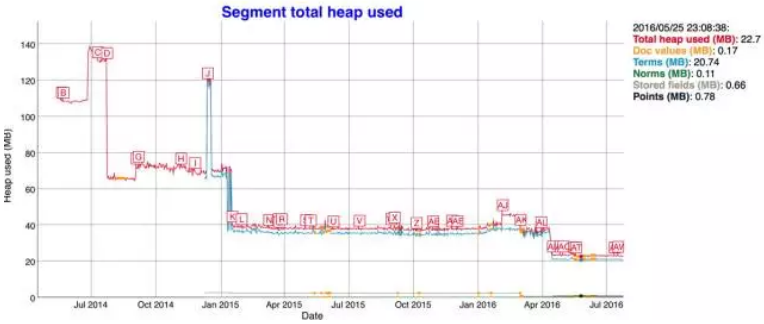


图中从2015/10/32 total bytes飙升是因为es启用了docvalues，我们关注红线，最近的引入新的数据结构之后，红色的索引大小只有原来的一半大小。

索引小了之后，merge的时间也响应的减少了，看下图：



相应的Java堆内存占用只原来的一半：

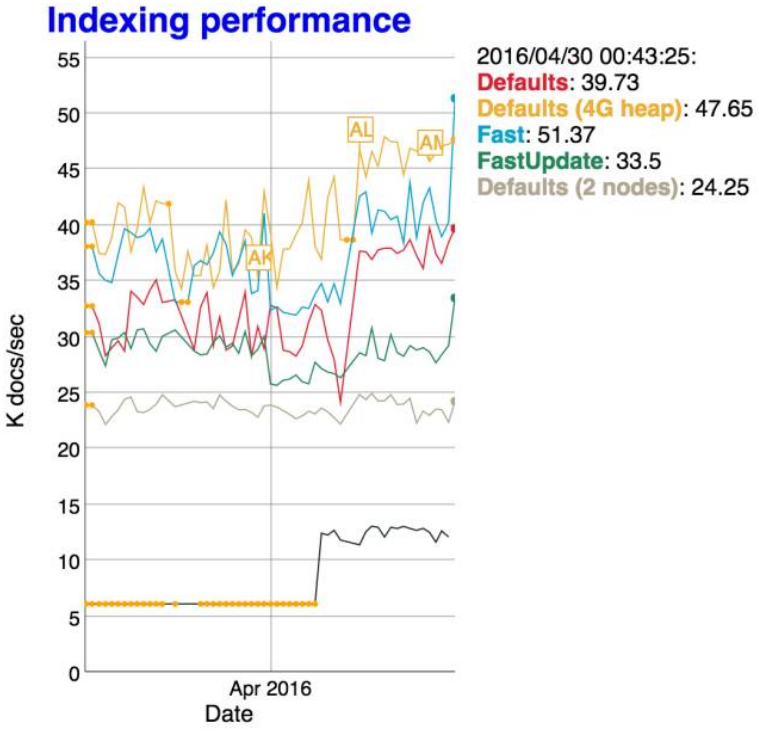


再看看索引的性能，也是飙升：

当然Lucene6里面还有很多优化和改进，这里没有一一列举。

我们再看看索引性能方面的其他优化。

ES5.0在Internal engine级别移除了用于避免同一文档并发更新的竞争锁，带来15%-20%的性能提升 #18060。



以上截图来自ES的每日持续性能监控：https://benchmarks.elastic.co/index.html

另一个和aggregation的改进也是非常大，Instant Aggregations。

Elasticsearch已经在Shard层面提供了Aggregation缓存，如果你的数据没有变化，ES能够直接返回上次的缓存结果，

但是有一个场景比较特殊，就是 date histogram，大家kibana上面的条件是不是经常设置的相对时间，如：

from:now-30d to:now，好吧，now是一个变量，每时每刻都在变，所以query条件一直在变，缓存也就是没有利用起来。

经过一年时间大量的重构，现在可以做到对查询做到灵活的重写：

首先，`now`关键字最终会被重写成具体的值；

其次，每个shard会根据自己的数据的范围来重写查询为 `match_all`或者是`match_none`查询，所以现在的查询能够被有效的缓存，并且只有个别数据有变化的Shard才需要重新计算，大大提升查询速度。

另外再看看和Scroll相关的吧。

现在新增了一个：Sliced Scroll类型

用过Scroll接口吧，很慢？如果你数据量很大，用Scroll遍历数据那确实是接受不了，现在Scroll接口可以并发来进行数据遍历了。

每个Scroll请求，可以分成多个Slice请求，可以理解为切片，各Slice独立并行，利用Scroll重建或者遍历要快很多倍。

看看这个demo

```
GET /twitter/tweet/_search?scroll=1m
{
    "slice": {
        "id": 0,
        "max": 2
    },
    "query": {
        "match" : {
            "title" : "elasticsearch"
        }
    }
}

GET /twitter/tweet/_search?scroll=1m
{
    "slice": {
        "id": 1,
        "max": 2
    },
    "query": {
        "match" : {
            "title" : "elasticsearch"
        }
    }
}
```

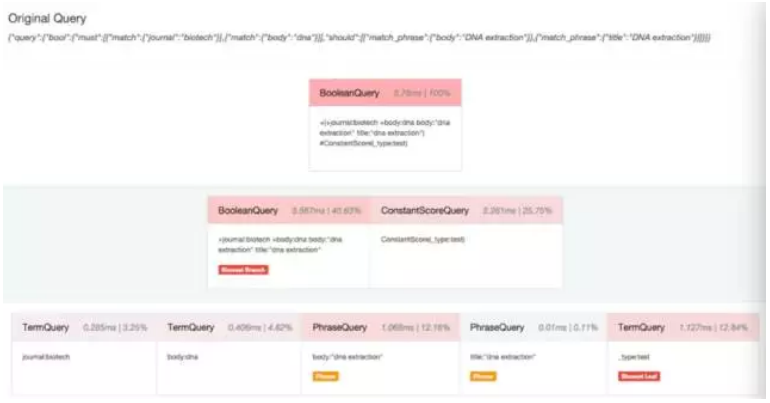
可以看到两个scroll请求，id分别是0和1，max是最大可支持的并行任务，可以各自独立进行数据的遍历获取。

我们再看看es在查询优化这块做的工作。

新增了一个Profile API。

https://www.elastic.co/guide/en/elasticsearch/reference/master/search-profile.html#_usage_3

都说要致富先修路，要调优当然需要先监控啦，elasticsearch在很多层面都提供了stats方便你来监控调优，但是还不够，其实很多情况下查询速度慢很大一部分原因是糟糕的查询引起的，玩过SQL的人都知道，数据库服务的执行计划（execution plan）非常有用，可以看到那些查询走没走索引和执行时间，用来调优，elasticsearch现在提供了Profile API来进行查询的优化，只需要在查询的时候开启profile：true就可以了，一个查询执行过程中的每个组件的性能消耗都能收集到。



那个子查询耗时多少，占比多少，一目了然。

同时支持search和aggregation的profile。

还有一个和翻页相关的问题，就是深度分页，是个老大难的问题，因为需要全局排序（number_of_shards * (from + size)），所以需要消耗大量内存，以前的es没有限制，有些同学翻到几千页发现es直接内存溢出挂了，后面elasticsearch加上了限制，from+size不能超过1w条，并且如果需要深度翻页，建议使用scroll来做。

但是scroll有几个问题，第一个是没有顺序，直接从底层segment进行遍历读取，第二个实时性没法保证，scroll操作有状态，es会维持scroll请求的上下文一段时间，超时后才释放，另外你在scroll过程中对索引数据进行了修改了，这个时候scroll接口是拿不到的，灵活性较差，现在有一个新的 Search After 机制，其实和scroll类似，也是游标的机制，它的原理是对文档按照多个字段进行排序，然后利用上一个结果的最后一个文档作为起始值，拿size个文档，一般我们建议使用_uid这个字段，它的值是唯一的id。

#（Search After

https://github.com/elastic/elasticsearch/blob/148f9af5857f287666aead37f249f204a870ab39/docs/reference/search/request/search-after.asciidoc）

来看一个Search After 的demo 吧，比较直观的理解一下：

```
POST test/test/172
{
        "foo": "bar",
        "age": 24
}

POST test/test/42
{
        "foo": "bar",
        "age": 18
}

POST /test/_search
{
  "query": {"match": {
    "foo": "bar"
  }},"sort": [
    {
      "age": {
        "order": "desc"
      },"_uid":{
        "order": "desc"
      }
    }
  ],"search_after": [24, "test#172"]
}
```

上面的demo，search_after后面带的两个参数，就是sort的两个结果。

根据你的排序条件来的，三个排序条件，就传三个参数。

再看看跟索引与分片管理相关的新功能吧。

新增了一个Shrink API

https://www.elastic.co/guide/en/elasticsearch/reference/master/indices-shrink-index.html#_shrinking_an_index

相信大家都知道elasticsearch索引的shard数是固定的，设置好了之后不能修改，如果发现shard太多或者太少的问题，之前如果要设置Elasticsearch的分片数，只能在创建索引的时候设置好，并且数据进来了之后就不能进行修改，如果要修改，只能重建索引。

现在有了Shrink接口，它可将分片数进行收缩成它的因数，如之前你是15个分片，你可以收缩成5个或者3个又或者1个，那么我们就可以想象成这样一种场景，在写入压力非常大的收集阶段，设置足够多的索引，充分利用shard的并行写能力，索引写完之后收缩成更少的shard，提高查询性能。

这里是一个API调用的例子

```
POST my_source_index/_shrink/my_target_index
{
  "settings": {
    "index.number_of_replicas": 1,
    "index.number_of_shards": 1,
    "index.codec": "best_compression"
  },
  "aliases": {
    "my_search_indices": {}
  }
}
```

上面的例子对my_source_index伸缩成一个分片的my_targe_index,使用了最佳压缩。

有人肯定会问慢不慢？非常快！ Shrink的过程会借助操作系统的Hardlink进行索引文件的链接，这个操作是非常快的，毫秒级Shrink就可收缩完成，当然windows不支持hard link，需要拷贝文件，可能就会很慢了。

再来看另外一个比较有意思的新特性，除了有意思，当然还很强大。

新增了一个Rollover API。

https://www.elastic.co/guide/en/elasticsearch/reference/master/indices-rollover-index.html#indices-rollover-index

前面说的这种场景对于日志类的数据非常有用，一般我们按天来对索引进行分割（数据量更大还能进一步拆分），我们以前是在程序里设置一个自动生成索引的模板，大家用过logstash应该就记得有这么一个模板logstash-[YYYY-MM-DD]这样的模板，现在es5.0里面提供了一个更加简单的方式：Rollover API

API调用方式如下：

```
PUT /logs-0001
{
  "aliases": {
    "logs_write": {}
  }
}

POST logs_write/_rollover
{
  "conditions": {
    "max_age":   "7d",
    "max_docs":  1000
  },
"settings": {
    "index.number_of_shards": 2
  }
}
```

从上面可以看到，首先创建一个logs-0001的索引，它有一个别名是logs_write,然后我们给这个logs_write创建了一个rollover规则，即这个索引文档不超过1000个或者最多保存7天的数据，超过会自动切换别名到logs-0002,你也可以设置索引的setting、mapping等参数,剩下的es会自动帮你处理。这个特性对于存放日志数据的场景是极为友好的。

新增：Reindex。

另外关于索引数据，大家之前经常重建，数据源在各种场景，重建起来很是头痛，那就不得不说说现在新加的Reindex接口了，Reindex可以直接在Elasticsearch集群里面对数据进行重建，如果你的mapping因为修改而需要重建，又或者索引设置修改需要重建的时候，借助Reindex可以很方便的异步进行重建，并且支持跨集群间的数据迁移。

比如按天创建的索引可以定期重建合并到以月为单位的索引里面去。

当然索引里面要启用_source。

来看看这个demo吧，重建过程中，还能对数据就行加工。

```
POST _reindex?pretty
{
  "source": {
    "index": "test"
  },
  "dest": {
    "index": "test2"
  },
  "script": {
    "inline": "ctx._source.tag = ctx._source.remove(\"flag\")"
  }
}
```

再看看跟Java开发者最相关的吧，就是 RestClient了

5.0里面提供了第一个Java原生的REST客户端SDK，相比之前的TransportClient，版本依赖绑定，集群升级麻烦，不支持跨Java版本的调用等问题，新的基于HTTP协议的客户端对Elasticsearch的依赖解耦，没有jar包冲突，提供了集群节点自动发现、日志处理、节点请求失败自动进行请求轮询，充分发挥Elasticsearch的高可用能力，并且性能不相上下。 #19055。

然后我们再看看其他的特性吧：

新增了一个Wait for refresh功能。

简单来说相当于是提供了文档级别的Refresh： https://www.elastic.co/guide/en/elasticsearch/reference/master/docs-refresh.html。

索引操作新增refresh参数，大家知道elasticsearch可以设置refresh时间来保证数据的实时性，refresh时间过于频繁会造成很大的开销，太小会造成数据的延时，之前提供了索引层面的_refresh接口，但是这个接口工作在索引层面，我们不建议频繁去调用，如果你有需要修改了某个文档，需要客户端实时可见怎么办？

在 5.0中，Index、Bulk、Delete、Update这些数据新增和修改的接口能够在单个文档层面进行refresh控制了，有两种方案可选，一种是创建一个很小的段，然后进行刷新保证可见和消耗一定的开销，另外一种是请求等待es的定期refresh之后再返回。

调用例子：

```
PUT /test/test/4?refresh=wait_for
{"test": "test"}
```

新增：Ingest Node

https://www.elastic.co/guide/en/elasticsearch/reference/master/ingest.html

再一个比较重要的特性就是IngestNode了，大家之前如果需要对数据进行加工，都是在索引之前进行处理，比如logstash可以对日志进行结构化和转换，现在直接在es就可以处理了，目前es提供了一些常用的诸如convert、grok之类的处理器，在使用的时候，先定义一个pipeline管道，里面设置文档的加工逻辑，在建索引的时候指定pipeline名称，那么这个索引就会按照预先定义好的pipeline来处理了；

Demo again：

```
PUT _ingest/pipeline/my-pipeline-id
{
  "description" : "describe pipeline",
  "processors" : [
    {
      "set" : {
        "field": "foo",
        "value": "bar"
      }
    }
  ]
}
PUT my-index/my-type/my-id?pipeline=my_pipeline_id
{
  "foo": "bar"
}
```

上图首先创建了一个名为my-pipeline-id的处理管道，然后接下来的索引操作就可以直接使用这个管道来对foo字段进行操作了，上面的例子是设置foo字段为bar值。

上面的还不太酷，我们再来看另外一个例子，现在有这么一条原始的日志，内容如下：

{

"message": "55.3.244.1 GET /index.html 15824 0.043"

}

google之后得知其Grok的pattern如下：）

那么我们使用Ingest就可以这么定义一个pipeline：

```
PUT _ingest/pipeline/my-pipeline2
{
  "description" : "...",
  "processors": [
    {
      "grok": {
        "field": "message",
        "patterns": ["%{IP:client} %{WORD:method} %{URIPATHPARAM
:request} %{NUMBER:bytes} %{NUMBER:duration}"]
      }
    }
  ]
}

POST log/type/id?pipeline=my-pipeline2
{
  "message": "55.3.244.1 GET /index.html 15824 0.043"
}

GET log/type/id
```

那么通过我们的pipeline处理之后的文档长什么样呢，我们获取这个文档的内容看看：

```
{
  "_index": "log",
  "_type": "type",
  "_id": "id",
  "_version": 1,
  "found": true,
  "_source": {
    "duration": "0.043",
    "request": "/index.html",
    "method": "GET",
    "bytes": "15824",
    "client": "55.3.244.1",
    "message": "55.3.244.1 GET /index.html 15824 0.043"
  }
}
```


很明显，原始字段message被拆分成了更加结构化的对象了。

再看看脚本方面的改变

新增Painless Scripting

还记得Groove脚本的漏洞吧，Groove脚本开启之后，如果被人误用可能带来的漏洞，为什么呢，主要是这些外部的脚本引擎太过于强大，什么都能做，用不好或者设置不当就会引起安全风险，基于安全和性能方面，我们自己开发了一个新的脚本引擎，名字就叫Painless，顾名思义，简单安全，无痛使用，和Groove的沙盒机制不一样，Painless使用白名单来限制函数与字段的访问，针对es的场景来进行优化，只做es数据的操作，更加轻量级，速度要快好几倍，并且支持Java静态类型，语法保持Groove类似，还支持Java的lambda表达式。

我们对比一下性能，看下图

Groovy是弱弱的绿色的那根。

再看看如何使用：

```
def first = input.doc.first_name.0;

def last   = input.doc.last_name.0;

return first + " " + last;
```

是不是和之前的写法差不多

或者还可以是强类型（10倍速度于上面的动态类型）

```
String first = (String)((List)((Map)input.get("doc")).get("first_name")).get(0);

String last   = (String)((List)((Map)input.get("doc")).get("last_name")).get(0);

return first + " " + last;
```

脚本可以在很多地方使用，比如搜索自定义评分；更新时对字段进行加工等

如：

```
GET hockey/_search
{
  "query": {
    "function_score": {
      "script_score": {
        "script": {
          "lang": "painless",
          "inline": "int total = 0; for (int i = 0; i < doc['goals'].length; ++i) { total += doc['goals'][i]; } return total;"
        }
      }
    }
  }
}
```

再来看看基础架构方面的变化

新增：Task Manager

这个是5.0 引入任务调度管理机制，用来做离线任务的管理，比如长时间运行的reindex和update_by_query等都是运行在TaskManager机制之上的，并且任务是可管理的，你可以随时cancel掉，并且任务状态持久化，支持故障恢复；

还新增一个：Depreated logging

大家在用ES的时候，其实有些接口可能以及打上了Depreated标签，即废弃了，在将来的某个版本中就会移除，你当前能用是因为一般废弃的接口都不会立即移除，给足够的时间迁移，但是也是需要知道哪些不能用了，要改应用代码了，所以现在有了Depreated日志，当打开这个日志之后，你调用的接口如果已经是废弃的接口，就会记录下日志，那么接下来的事情你就知道你应该怎么做了。

新增: Cluster allocation explain API

『谁能给我一个shard不能分配的理由』，现在有了，大家如果之前遇到过分片不能正常分配的问题，但是不知道是什么原因，只能尝试手动路由或者重启节点，但是不一定能解决，其实里面有很多原因，现在提供的这个explain接口就是告诉你目前为什么不能正常分配的原因，方便你去解决。

另外在数据结构这块，新增: half_float 类型

https://www.elastic.co/guide/en/elasticsearch/reference/master/number.html

只使用 16 位 足够满足大部分存储监控数值类型的场景，支持范围：2负24次方 到 65504，但是只占用float一半的存储空间。

Aggregation新增: Matrix Stats Aggregation #18300

金融领域非常有用的，可计算多个向量元素协方差矩阵、相关系数矩阵等等

另外一个重要的特性：为索引写操作添加顺序号 #10708

大家知道es是在primary上写完然后同步写副本，这些请求都是并发的，虽然可以通过version来控制冲突，

但是没法保证其他副本的操作顺序，通过写的时候产生顺序号，并且在本地也写入checkpoint来记录操作点，

这样在副本恢复的时候也可以知道当前副本的数据位置，而只需要从指定的数据开始恢复就行了，而不是像以前的粗暴的做完整的文件同步，另外这些顺序号也是持久化的，重启后也可以快速恢复副本信息，想想以前的大量无用拷贝吧和来回倒腾数据吧。

Elasticsearch5.0其他方面的改进

我们再看看mapping这块的改进吧。

引入新的字段类型Text/Keyword 来替换 String

以前的string类型被分成Text和Keyword两种类型，keyword类型的数据只能完全匹配，适合那些不需要分词的数据，

对过滤、聚合非常友好，text当然就是全文检索需要分词的字段类型了。将类型分开的好处就是使用起来更加简单清晰，以前需要设置analyzer和index，并且有很多都是自定义的分词器，从名称根本看不出来到底分词没有，用起来很麻烦。

另外string类型暂时还在的，6.0会移除。

还有关于Index Settings 的改进

Elasticsearch的配置实在太多，在以前的版本间，还移除过很多无用的配置，经常弄错有没有？

现在，配置验证更加严格和保证原子性，如果其中一项失败，那个整个都会更新请求都会失败，不会一半成功一半失败。下面主要说两点：

1. 设置可以重设会默认值，只需要设置为 `null`即可
2. 获取设置接口新增参数`?include_defaults`,可以直接返回所有设置和默认值

集群处理的改进: Deleted Index Tombstones

在以前的es版本中，如果你的旧节点包含了部分索引数据，但是这个索引可能后面都已经删掉了，你启动这个节点之后，会把索引重新加到集群中，是不是觉得有点阴魂不散，现在es5.0会在集群状态信息里面保留500个删除的索引信息，所以如果发现这个索引是已经删除过的就会自动清理，不会再重复加进来了。

文档对象的改进：字段名重新支持英文句号，再2.0的时候移除过dot在字段名中的支持，现在问题解决了，又重新支持了。

es会认为下面两个文档的内容一样：

```
PUT my_index/my_type/1
{
  "aaa.bbb.ccc": "some_val",
  "aaa.ddd": "other_val"
}

PUT my_index/my_type/2
{
  "aaa": {
    "bbb": {
      "ccc": "some_val"
    },
    "ddd": "other_val"
  }
}
```

还有其他的一些改进

Cluster state的修改现在会和所有节点进行ack确认。

Shard的一个副本如果失败了，Primary标记失败的时候会和Master节点确认完毕再返回。

使用UUID来作为索引的物理的路径名，有很多好处，避免命名的冲突。

_timestamp 和 _ttl已经移除，需要在Ingest或者程序端处理。

ES可直接用HDFS来进行备份还原（Snapshot/Restore ）了 #15191。

Delete-by-query和Update-by-query重新回到core，以前是插件，现在可以直接使用了，也是构建在Reindex机制之上。

HTTP请求默认支持压缩，当然http调用端需要在header信息里面传对应的支持信息。

创建索引不会再让集群变红了，不会因为这个卡死集群了。

默认使用BM25评分算法，效果更佳，之前是TF/IDF。

快照Snapshots添加UUID解决冲突 #18156。

限制索引请求大小，避免大量并发请求压垮ES #16011。

限制单个请求的shards数量，默认1000个 #17396。

移除 site plugins，就是说head、bigdesk都不能直接装es里面了，不过可以部署独立站点（反正都是静态文件）或开发kibana插件 #16038。

允许现有parent类型新增child类型 #17956。

这个功能对于使用parent-child特性的人应该非常有用。

支持分号（；）来分割url参数，与符号（&）一样 #18175。

比如下面这个例子：

curl http://localhost:9200/_cluster/health?level=indices;pretty=true

好吧，貌似很多，其实上面说的还只是众多特性和改进的一部分，es5.0做了非常非常多工作，本来还打算讲讲bug修复的，但是太多了，时间有限，一些重要的bug在2.x都已经第一时间解决了，大家可以查看下面的链接了解更多更详细的更新日志：

https://www.elastic.co/guide/en/elasticsearch/reference/master/release-notes-5.0.0-alpha1-2x.html

https://www.elastic.co/guide/en/elasticsearch/reference/master/release-notes-5.0.0-alpha1.html

https://www.elastic.co/guide/en/elasticsearch/reference/master/release-notes-5.0.0-alpha2.html

https://www.elastic.co/guide/en/elasticsearch/reference/master/release-notes-5.0.0-alpha3.html

https://www.elastic.co/guide/en/elasticsearch/reference/master/release-notes-5.0.0-alpha4.html

下载体验最新的版本：https://www.elastic.co/v5

升级向导：https://github.com/elastic/elasticsearch-migration/blob/2.x/README.asciidoc

如果有es相关的问题也欢迎前往 Elastic中文社区：http://elasticsearch.cn

进行交流和讨论，可以加我微信单独探讨，也欢迎上elasticsearch.cn发帖讨论，谢谢大家。

Q&A

Q1：是否有用es做hbase的二级索引的

A1：这种案例说实话比较少，因为成本比较高，在两套分布式系统里面做结合，并且要满足足够的性能，有点难度，不建议这样去做。

Q2：批量更新数据会出现少量数据更新不成功

A2：这个首先要看少量失败的原因是什么，es的返回信息里面会包含具体的信息，如果json格式不合法也是会失败的。

Q3：ik插件有没有计划支持同义词，专有名词热更新？对于词库更新比较频繁的应用场景，只能采取全部重新建立索引的方式吗？

A3：同义词有单独的filter，可以和ik结合一起使用的，关于热更新这个确实是需要重建，词库变化之后，分词产生的term不一样了，不重建的话，倒排很可能匹配不上，查询会失败。

Q4：老师，你好，我有个问题想咨询一下，我们原来的商品基本数据，商品评价数据，收藏量这些都在mysql里，但我们现在想上es，我们想把商品的基本数据放es，收藏、评价这些实时数据，还是放mysql，但做排序功能的时候，会参考一个商品的收藏量，评价量，这时候在还涉及数据分页的情况下，怎么结合es和mysql的数据进行排序呢？

A4：这个问题得具体看业务场景，如果更新频繁，但是还在es承受能力范围和业务响应指标内，可以直接放es里面，在es里面做排序，如果太大，建议放外部存储，外部存储和es的结合方式又有很多种，收藏评价是否真的需要那么实时？另外es的评分机制是可以扩展的，在评分阶段使用自定义插件读取外部数据源，进行混合打分也是可行的。

Q5：现在大agg查询可以cancel吗?

A5：现在还不能。

Q6：有考虑提供sql语法查询吗？

A6：目前暂时还没计划。

Q7：128g内存的机器，官方建议机器上放两个es实例，目前也是推荐这样做吗?

A7：这个其实看场景的，单台机器上面的索引比较大的话，建议多留一点给操作系统来做缓存，多个实例可以提供足够吞吐。

Q8：请问用于计算unique count的算法有变化吗？

A8：有的，elasticsearch里面叫cardinality。这里有篇文章：https://www.elastic.co/blog/count-elasticsearch。

Q9：请问在es5中，每个服务器有256G内存，那每个服务器带的存储多少比较合适？是24T，48T还是可以更多？

A9：这个看场景啦，有超过48T的。

Q10：请问下Elastic Stack是只要安装一次这个就行，还是要像原来elk一样，分别安装不同的组件？

A10：安装方式和之前一样的.

Q11：请问es中的如何做到按某个字段去重？具体问题是这样的，我们有一个文章索引，其中有2亿数据量，每次搜索的结果总是存在大量重复的title，我们希望在查询时能根据title进行去重。也就是Field Collapsing特性，官方有一个通过terms aggregation进行去重的方案，但效果不是很理想，仍然会有很多重复，我们希望哪怕是按title严格相等来去重也可以接受。 另外我们有一个通过simhash来去重的思路，就是计算title的simhash，一并存入索引，在搜索阶段通过simhash计算相似性，但这需要全量重新计算，数据量太大。所以还是希望能在不动现有索引的情况下，通过某种技巧，实现这个功能。

A11：直接去重，这个目前没有比较好的方案，不过很多变通的做法，首先你的场景需要确认，title重复是不是不允许的，如果是，那么建索引的时候就可以hash掉作为主键，这样就不会有重复的了，如果你觉得原始数据也要，那么索引阶段产生一个独立的去除title的索引，来做join，当然还是要看你业务的场景具体研究。

Q12：硬件受限的情况下，清理过期数据的策略。

A12：如果你的数据结构固定，结合5.0的Rollover接口，估计能够承载的最大索引量，定期检测删索引就行了。

讲师介绍：曾勇(Medcl)，Elastic开发工程师与技术布道师。

曾勇是Elasticsearch国内首批用户，自2010年起就开始接触Elasticsearch并投入到生产环境中使用，并编写过一系列的中文处理相关的插件，是Elasticsearch中文社区发起人，筹办了一系列线上线下的Elasticsearch技术分享与交流活动，出于对Elasticsearch的喜爱，目前已全职加入Elasticsearch项目背后的Elastic公司。微信号：medcl123。

感谢杜小芳对本文的审校。

给InfoQ中文站投稿或者参与内容翻译工作，请邮件至editors@cn.infoq.com。也欢迎大家通过新浪微博（@InfoQ，@丁晓昀），微信（微信号：InfoQChina）关注我们。

- 领域
- 架构 & 设计
- 语言 & 开发
- 专栏
- 数据库
- 大数据
- Elasticsearch

相关内容

解读2015之大数据篇：大数据的黄金时代

携程基于Storm的实时大数据平台实践

专访链家蔡白银：大数据如何解决房产领域痛点推动行业进步

美团大数据平台架构实践

象SaaS一样用亚马逊Kinesis Analytics做大数据分析

相关厂商内容

携程的推荐及智能化算法及架构体系实践

Autodesk基于Spark自建大数据平台的实践经验

大数据与电商四大核心要素

阿里巴巴数据研发体系的建立和管理之道

苏宁云商数据平台实时化实践

相关赞助商

告诉我们您的想法

请输入主题 信息

允许的HTML标签: a,b,br,blockquote,i,li,pre,u,ul,p

当有人回复此评论时请E-mail通知我

发送信息

社区评论 Watch Thread

图片 by Huang Zhiwei Posted 2016年8月8日 01:38

ES2016年大会 by 白 文辉 Posted 2016年8月15日 09:40

图片 2016年8月8日 01:38 by "Huang Zhiwei"

好多图片看不到...

- 回复
- 回到顶部

ES2016年大会 2016年8月15日 09:40 by "白 文辉"

曾大神，ES2016大会上海这边的今年啥时候召开？

- 回复
- 回到顶部

关闭

by

发布于

- 查看
- 回复
- 回到顶部

关闭

引用原消息

主题 您的回复

允许的HTML标签: a,b,br,blockquote,i,li,pre,u,ul,p

当有人回复此评论时请E-mail通知我
发送信息 取消
关闭

主题 您的回复

允许的HTML标签: a,b,br,blockquote,i,li,pre,u,ul,p

当有人回复此评论时请E-mail通知我

取消
关闭

OK
相关内容

- 郭炜：大数据领域缺的是分析人才 2016年9月18日
- 专访链家蔡白银：大数据如何解决房产领域痛点推动行业进步 2016年9月30日
- 象SaaS一样用亚马逊Kinesis Analytics做大数据分析 2016年9月27日

- 架构师特刊：大数据平台架构 2016年8月24日
- 大数据与机器学习周报 第25期：谷歌开源大规模语言建模库 2016年9月20日

- 携程基于Storm的实时大数据平台实践 2016年10月10日

- 美团大数据平台架构实践 2016年9月30日
- 大数据与机器学习周报 第24期：Facebook开源Zstandard算法及MyRocks存储引擎 2016年9月13日

- YY直播应用大数据决胜安全对抗的实践 2016年9月23日

- 大数据和人工智能在互联网金融上的应用 2016年9月22日

- 大数据下的技术运营（一）——监控系统概览篇 2016年9月20日