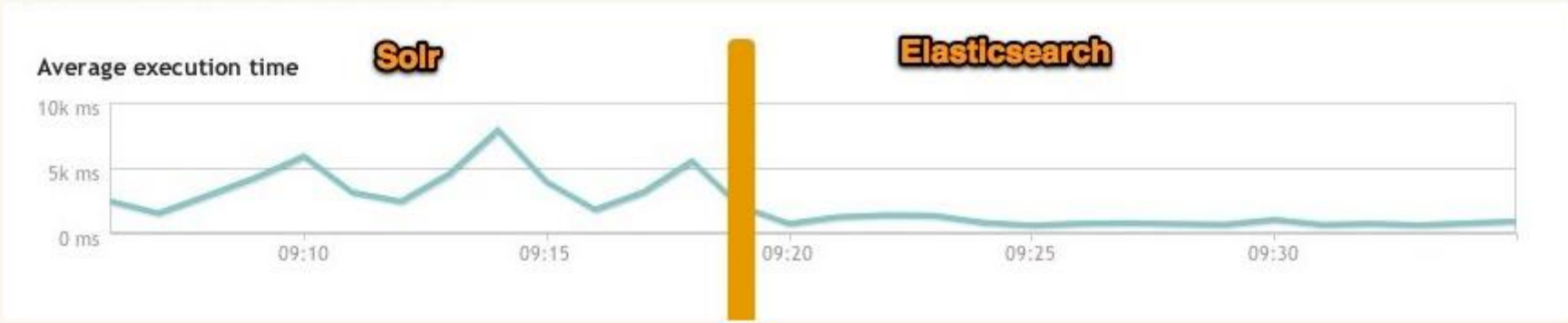
Elasticsearch(二)　20:30开始

讲师：君临天下

- 课程大纲
  - 介绍elasticsearch
  - 架构原理
  - 核心概念
  - CURL使用

- **Solr和elasticsearch的性能对比**

下图为将搜索引擎从Solr转到Elasticsearch以后的平均查询速度有了50倍的提升。



- **Lucene的核心理论——倒排索引**
  - **根据属性的值**来查找记录。这种索引表中的每一项都包括**一个属性值和具有该属性值的各记录的地址**。由于不是由记录来确定属性值，而是由属性值来确定记录的位置，因而称为倒排索引(invertedindex)
  - **单词——文档矩阵**

单词-文档矩阵					
	文档1	文档2	文档3	文档4	文档5
词汇1	✓			✓	
词汇2		✓	✓		
词汇3				✓	
词汇4	✓				✓
词汇5		✓			
词汇6			✓		

- **Lucene索引中的概念**
- document
  用户提供的源是一条条记录，它们可以是文本文件、字符串或者数据库表的一条记录等等。一条记录经过索引之后，就是以一个Document的形式存储在索引文件中的。用户进行搜索，也是以Document列表的形式返回。
- field
  一个Document可以包含多个信息域，例如一篇文章可以包含“标题”、“正文”、“最后修改时间”等信息域，这些信息域就是通过Field在Document中存储的。
- segment
  添加索引时并不是每个document都马上添加到同一个索引文件，它们首先被写入到不同的小文件，然后再合并成一个大索引文件，这里每个小文件都是一个segment。

- Elasticsearch中的核心概念
- cluster
  - 代表一个集群，集群中有多个节点，其中有一个为主节点，这个主节点是可以通过选举产生的，主从节点是对于集群内部来说的。es的一个概念就是去中心化，字面上理解就是无中心节点，这是对于集群外部来说的，因为从外部来看es集群，在逻辑上是个整体，你与任何一个节点的通信和与整个es集群通信是等价的。
  - 主节点的职责是负责管理集群状态，包括管理分片的状态和副本的状态，以及节点的发现和删除。
- 只需要在同一个网段之内启动多个es节点，就可以自动组成一个集群。
- 默认情况下es会自动发现同一网段内的节点，自动组成集群。
- 集群状态查看
  - http://192.168.57.4:9200/_cluster/health?pretty

- Elasticsearch中的master选举
- 源码中ZenDiscovery的findMaster对各个节点通信
- 如果ping的结果显示其他节点已选举出了master

  在这些节点所选举的master列表中选取id值最小的一个作为当前节点的master
- 如果ping结果显示其他节点还没有选举出master
  则在当前节点所能访问的master备选集合中
  选取id值最小的一个作为master

- Elasticsearch中的核心概念
- shards
  - 代表索引分片，es可以把一个完整的索引分成多个分片，这样的好处是可以把一个大的索引拆分成多个，分布到不同的节点上。构成分布式搜索。分片的数量只能在索引创建前指定，并且索引创建后不能更改。
- 可以在创建索引库的时候指定
  - curl -XPUT 'localhost:9200/test1/' -d'{"settings":{"number_of_shards":3}}'
- 默认是一个索引库有5个分片
  - index.number_of_shards: 5

- Elasticsearch中的核心概念
- replicas
  - 代表索引副本，es可以给索引设置副本，副本的作用一是提高系统的容错性，当某个节点某个分片损坏或丢失时可以从副本中恢复。二是提高es的查询效率，es会自动对搜索请求进行负载均衡。
- 可以在创建索引库的时候指定
  - curl -XPUT 'localhost:9200/test2/' -d'{"settings":{"number_of_replicas":2}}'
- 默认是一个分片有1个副本
  - index.number_of_replicas: 1

- Elasticsearch中的核心概念
- recovery
  - 代表数据恢复或叫数据重新分布，es在有节点加入或退出时会根据机器的负载对索引分片进行重新分配，挂掉的节点重新启动时也会进行数据恢复。
  - 设置复苏时的吞吐量，默认情况下是无限的：indices.recovery.max_size_per_sec: 0
    设置从对等节点恢复片段时打开的流的数量上限：indices.recovery.concurrent_streams: 5

- Elasticsearch中的核心概念
- gateway
  - 代表es索引的持久化存储方式，es默认是先把索引存放到内存中，当内存满了时再持久化到硬盘。当这个es集群关闭再重新启动时就会从gateway中读取索引数据。es支持多种类型的gateway，有本地文件系统（默认），分布式文件系统，Hadoop的HDFS和amazon的s3云存储服务。

- Elasticsearch中的核心概念
- gateway
  - 如果需要将数据落地到hadoop的hdfs需要先安装插件elasticsearch/elasticsearch-hadoop，然后再elasticsearch.yml配置
  - gateway:
    type: hdfs
    gateway:
    hdfs:
    uri: hdfs://localhost:9000

- Elasticsearch中的核心概念
- discovery.zen
  - 代表es的自动发现节点机制，es是一个基于p2p的系统，它先通过广播寻找存在的节点，再通过多播协议来进行节点之间的通信，同时也支持点对点的交互。
- 如果是不同网段的节点如何组成es集群
  - 禁用自动发现机制
  - discovery.zen.ping.multicast.enabled: false
  - 设置新节点被启动时能够发现的主节点列表
  - discovery.zen.ping.unicast.hosts: [“192.168.57.4", "192.168.57.5"]

- Elasticsearch中脑裂问题
- 正常情况下，集群中的所有的节点，应该对集群中master的选择是一致的，这样获得的状态信息也应该是一致的，不一致的状态信息，说明不同的节点对master节点的选择出现了异常——也就是所谓的脑裂问题。这样的脑裂状态直接让节点失去了集群的正确状态，导致集群不能正常工作。

- Elasticsearch中脑裂产生的原因
- 1. 网络：由于是内网通信，网络通信问题造成某些节点认为master死掉，而另选master的可能性较小
- 2. 节点负载：由于master节点与data节点都是混合在一起的，所以当工作节点的负载较大时，导致对应的ES实例停止响应，而这台服务器如果正充当着master节点的身份，那么一部分节点就会认为这个master节点失效了，故重新选举新的节点，这时就出现了脑裂；同时由于data节点上ES进程占用的内存较大，较大规模的内存回收操作也能造成ES进程失去响应。

- Elasticsearch中脑裂解决
- 主节点
- node.master: true
- node.data: false
- 从节点
- node.master: false
- node.data: true
- 所有节点
- discovery.zen.ping.multicast.enabled: false
- discovery.zen.ping.unicast.hosts: [“master”，“slave1”，“slave2"]

- Elasticsearch中的核心概念
- Transport
  - 代表es内部节点或集群与客户端的交互方式，默认内部是使用tcp协议进行交互，同时它支持http协议（json格式）、thrift、servlet、memcached、zeroMQ等的传输协议（通过插件方式集成）。

- CURL命令
  - 简单认为是可以在命令行下访问url的一个工具
  - curl是利用URL语法在命令行方式下工作的开源文件传输工具，使用curl可以简单实现常见的get/post请求。
  - curl
  - -x 指定http请求的方法
  - HEAD GET POST PUT DELETE
  - -d 指定要传输的数据

- CURL建立索引库
- curl -XPUT 'http://localhost:9200/index_name/'
- PUT/POST都可以

- CURL创建索引
- #创建索引

```
curl -XPOST http://localhost:9200/bjsxt/employee/1 -d
    '{
"first_name" : "John",
"last_name" : "Smith",
"age" : 25,
"about" : "I love to go rock climbing",
"interests": [ "sports", "music" ]
}'
```

- PUT 和POST用法
- PUT是幂等方法，POST不是。所以PUT用于更新、POST用于新增比较合适。
  - PUT，DELETE操作是幂等的。所谓幂等是指不管进行多少次操作，结果都一样。比如我用PUT修改一篇文章，然后在做同样的操作，每次操作后的结果并没有不同，DELETE也是一样。
  - POST操作不是幂等的，比如常见的POST重复加载问题：当我们多次发出同样的POST请求后，其结果是创建出了若干的资源。
  - 还有一点需要注意的就是，创建操作可以使用POST，也可以使用PUT，区别在于POST是作用在一个集合资源之上的（/articles），而PUT操作是作用在一个具体资源之上的（/articles/123），比如说很多资源使用数据库自增主键作为标识信息，而创建的资源的标识信息到底是什么只能由服务端提供，这个时候就必须使用POST。

- 创建索引注意事项

- **索引库名称必须要全部小写，不能以下划线开头，也不能包含逗号**
- 如果没有明确指定索引数据的ID，那么es会自动生成一个随机的ID,需要使用POST参数

  curl -XPOST http://localhost:9200/bjsxt/emp/ -d '{"first_name" : "John"}'
- 如果想要确定我们创建的都是全新的内容
  - 1：使用自增ID
  - 2：在url后面添加参数
    - curl -XPUT http://localhost:9200/bjsxt/emp/2?op_type=create -d '{"name": “zs","age":25}'
    - curl -XPUT http://localhost:9200/bjsxt/emp/2/_create -d '{"name": “laoxiao","age":25}'
  - 如果成功创建了新的文档，Elasticsearch将会返回常见的元数据以及201 Created的HTTP反馈码。而如果存在同名文件，Elasticsearch将会返回一个409 Conflict的HTTP反馈码

- Elasticsearch中的settings和mappings
- settings修改索引库默认配置
  - 例如：分片数量，副本数量
  - 查看：curl -XGET http://localhost:9200/bjsxt/_settings?pretty
  - curl -XPUT 'localhost:9200/bjsxt/' -d'{"settings":{"number_of_shards":3,"number_of_replicas":2}}'
- Mapping,就是对索引库中索引的字段名称及其数据类型进行定义，类似于关系数据库中表建立时要定义字段名及其数据类型那样，(和solr中的schme类似)不过es的mapping比数据库灵活很多，它可以动态添加字段。一般不需要要指定mapping都可以，因为es会自动根据数据格式定义它的类型，如果你需要对某些字段添加特殊属性（如：定义使用其它分词器、是否分词、是否存储等），就必须手动添加mapping
- 查询索引库的mapping信息
  - curl -XGET http://localhost:9200/bjsxt/emp/_mapping?pretty
- mappings修改字段相关属性
  - 例如：字段类型，使用哪种分词工具

- GET查询索引
  - 根据员工id查询
    - curl -XGET http://localhost:9200/bjsxt/employee/1?pretty
  - 在任意的查询字符串中添加pretty参数，es可以得到易于识别的json结果。
  - curl后添加-i 参数，这样你就能得到反馈头文件
    - curl -i 'http://192.168.1.170:9200/bjsxt/emp/1?pretty'

- GET查询索引
  - 检索文档中的一部分，如果只需要显示指定字段，
    - curl -XGET http://localhost:9200/bjsxt/employee/1?_source=name,age&pretty
  - 如果只需要source的数据
    - curl –XGET http://localhost:9200/bjsxt/employee/1/_source
  - 查询所有
    - curl -XGET http://localhost:9200/bjsxt/employee/_search
  - 你可以再返回的 hits 中发现我们录入的文档。搜索会默认返回最前的10个数值。
  - 根据条件进行查询
    - curl -XGET http://localhost:9200/elasticsearch/employee/_search?q=last_name:Smith

- DSL查询
- Domain Specific Language
  - 领域特定语言
- curl -XGET http://localhost:9200/bjsxt/employee/_search -d
  - '{"query":
    - {"match":
      - {"last_name":"Smith"}
    - }
  - }'

- MGET查询
- 使用mget API获取多个文档
  - curl -XGET http://localhost:9200/_mget?pretty -d '{"docs":[{"_index":"elasticsearch","_type":"emp","_id":1,"_source":"name"},{"_index":"website","_type":"blog","_id":2}]}'
- 如果你需要的文档在同一个_index或者同一个_type中，你就可以在URL中指定一个默认的/_index或者/_index/_type
  - curl -XGET http://localhost:9200/bjsxt/emp/_mget?pretty -d '{"docs":[{"_id":1},{"_type":"blog","_id":2}]}'
- 如果所有的文档拥有相同的_index以及 _type，直接在请求中添加ids的数组即可
  - curl -XGET http://localhost:9200/bjsxt/emp/_mget?pretty -d '{"ids":["1","2"]}'
- 注意：如果请求的某一个文档不存在，不会影响其他文档的获取结果。HTTP返回状态码依然是200，这是因为mget这个请求本身已经成功完成。要确定独立的文档是否被成功找到，需要检查found标识

- HEAD使用
- 如果只想检查一下文档是否存在，你可以使用HEAD来替代GET方法，这样就只会返回HTTP头文件
  - curl -i -XHEAD http://localhost:9200/bjsxt/emp/1

- Elasticsearch的更新
- ES可以使用PUT或者POST对文档进行更新，如果指定ID的文档已经存在，则执行更新操作
- 注意:执行更新操作的时候
  - ES首先将旧的文档标记为删除状态
  - 然后添加新的文档
  - 旧的文档不会立即消失，但是你也无法访问
  - ES会在你继续添加更多数据的时候在后台清理已经标记为删除状态的文档

- Elasticsearch的更新
- 局部更新，可以添加新字段或者更新已有字段（必须使用POST）
  - curl -XPOST http://localhost:9200/bjsxt/emp/1/_update -d '{"doc":{"city":"beijing","car":"BMW"}}'

- Elasticsearch的删除
- curl -XDELETE http://localhost:9200/bjsxt/emp/4/
- 如果文档存在，es会返回200 ok的状态码，found属性值为true，_version属性的值+1
- 如果文档不存在，es会返回404 Not Found的状态码，found属性值为false，但是_version属性的值依然会+1，这个就是内部管理的一部分，它保证了我们在多个节点间的不同操作的顺序都被正确标记了
- 注意：删除一个文档也不会立即生效，它只是被标记成已删除。Elasticsearch将会在你之后添加更多索引的时候才会在后台进行删除内容的清理

- Elasticsearch的删除
- 通过查询API删除指定索引库下指定类型下的数据

```
curl -XDELETE 'http://localhost:9200/bjsxt/emp/_query?q=user:kimchy'

curl -XDELETE 'http://localhost:9200/bjsxt/emp/_query' -d
'{
    "query" : {
        "term" : { "user" : "kimchy" }
    }
}'
```

- Elasticsearch的删除
- 通过查询API删除指定索引库下多种类型下的数据

curl -XDELETE 'http://localhost:9200/bjsxt/emp,user/_query?q=user:kimchy ‘

- 通过查询API删除多个索引库下多种类型下的数据

curl -XDELETE 'http://localhost:9200/bjsxt,index/emp,user/_query?q=user:kimchy'

或者删除所有索引库中的匹配的数据

curl -XDELETE 'http://localhost:9200/_all/_query?q=tag:wow'

- Elasticsearch的批量操作bulk
- 与mget类似，bulk API可以帮助我们同时执行多个请求
- 格式：
  - action：index/create/update/delete
  - metadata：_index,_type,_id
  - request body：_source(删除操作不需要)

```
{ action: { metadata }}\n
{ request body        }\n
{ action: { metadata }}\n
{ request body        }\n
```

- 使用curl -XPOST -d 时注意，不能直接在json字符串中添加\n字符，应该按回车
- create 和index的区别
  - 如果数据存在，使用create操作失败，会提示文档已经存在，使用index则可以成功执行。
- 使用文件的方式
  - vi requests
  - curl -XPOST/PUT localhost:9200/_bulk --data-binary @request;
- bulk请求可以在URL中声明/_index 或者/_index/_type
- bulk一次最大处理多少数据量
  - bulk会把将要处理的数据载入内存中，所以数据量是有限制的
  - 最佳的数据量不是一个确定的数值，它取决于你的硬件，你的文档大小以及复杂性，你的索引以及搜索的负载
  - 一般建议是1000-5000个文档，如果你的文档很大，可以适当减少队列，大小建议是5-15MB，默认不能超过100M，可以在es的配置文件中修改这个值http.max_content_length: 100mb

- Elasticsearch的版本控制
- 普通关系型数据库使用的是（悲观并发控制（PCC））
  - 当我们在读取一个数据前先锁定这一行，然后确保只有读取到数据的这个线程可以修改这一行数据
- ES使用的是（乐观并发控制（OCC））
  - ES不会阻止某一数据的访问，然而，如果基础数据在我们读取和写入的间隔中发生了变化，更新就会失败，这时候就由程序来决定如何处理这个冲突。它可以重新读取新数据来进行更新，又或者将这一情况直接反馈给用户。
- ES如何实现版本控制(使用es内部版本号)
  - 1：首先得到需要修改的文档，获取版本(_version)号
    - curl -XGET http://localhost:9200/elasticsearch/emp/1
  - 2：在执行更新操作的时候把版本号传过去
    - curl -XPUT http://localhost:9200/elasticsearch/emp/1?version=1 -d '{"name": “zs","age":25}'(覆盖)
    - curl -XPOST http://localhost:9200/elasticsearch/emp/1/_update?version=1 -d '{"doc":{"city":"beijing","car":"BMW"}}'(部分更新)
  - 3：如果传递的版本号和待更新的文档的版本号不一致，则会更新失败

- Elasticsearch的版本控制
- ES如何实现版本控制(使用外部版本号)
  - 如果你的数据库已经存在了版本号，或者是可以代表版本的时间戳。这时就可以在es的查询url后面添加version_type=external来使用这些号码。
  - 注意：版本号码必须要是大于0小于9223372036854775807（Java中long的最大正值）的整数。
  - es在处理外部版本号的时候，它不再检查_version是否与请求中指定的数值是否相等，而是检查当前的_version是否比指定的数值小，如果小，则请求成功。
  - example：
    - curl -XPUT 'http://localhost:9200/bjsxt/emp/20?version=10&version_type=external' -d '{"name":"laoxiao"}'
    - 注意：此处url前后的引号不能省略，否则执行的时候会报错

- Elasticsearch的插件
- 站点插件（以网页形式展现）
  - BigDesk Plugin (作者 Lukáš Vlček)
  - 简介：监控es状态的插件，推荐！
  - Elasticsearch Head Plugin (作者 Ben Birch)
  - 简介：很方便对es进行各种操作的客户端。
  - Paramedic Plugin (作者 Karel Minařík)
  - 简介：es监控插件
  - SegmentSpy Plugin (作者 Zachary Tong)
  - 简介：查看es索引segment状态的插件
  - Inquisitor Plugin (作者 Zachary Tong)
  - 简介：这个插件主要用来调试你的查询。

- Elasticsearch的插件
- 这个主要提供的是节点的实时状态监控，包括jvm的情况，linux的情况，elasticsearch的情况
  - 安装bin/plugin -install lukas-vlcek/bigdesk
  - 删除bin/plugin --remove bigdesk
- http://192.168.57.4:9200/_plugin/bigdesk/
- http://192.168.57.4:9200/_cluster/state?pretty

- Elasticsearch的插件
- 这个主要提供的是健康状态查询
  - 安装bin/plugin -install mobz/elasticsearch-head
- http://192.168.57.4:9200/_plugin/head/
- http://192.168.57.4:9200/_cluster/health?pretty