大讲台
dajiangtai.com

elk课程QQ学习交流群：732021751　加群备注：elk

客服微信/QQ：84985152

助教QQ：484166349

网址：http://www.dajiangtai.com

Elastic search 6.x

讲师：john

ElasticSearch大纲

- elasticsearch 概述
- elasticsearch单节点安装
- elasticsearch rest基本操作
- elasticsearch 核心概念
- elasticsearch Java 客户端
- elasticsearch 索引模块
- elasticsearch 集群安装部署
- elasticsearch 优化

ElasticSearch概述

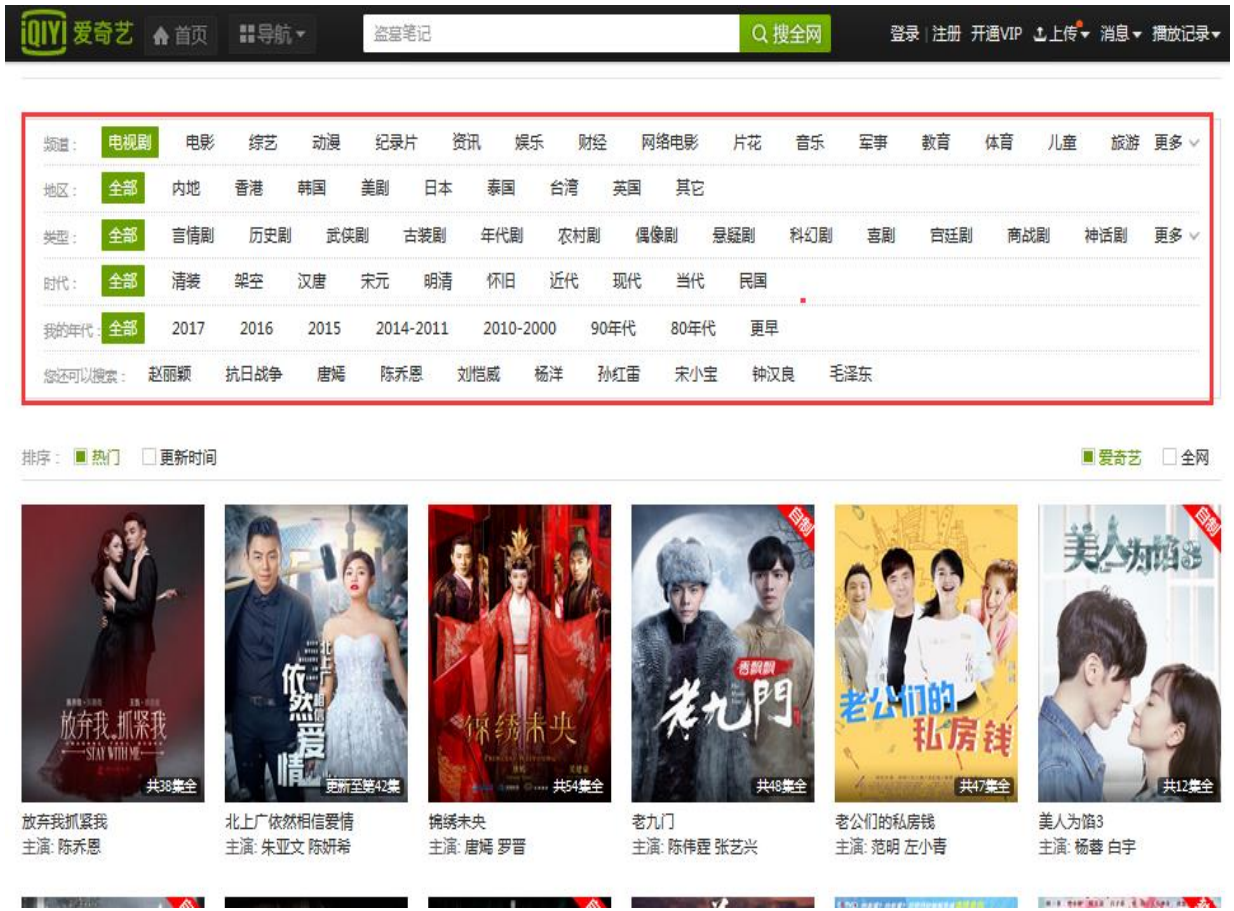
- elasticsearch 产生背景
- elasticsearch 定义
- elasticsearch vs Solr
- elasticsearch vs 关系型数据库
- elasticsearch 架构原理
- elasticsearch 在大数据中的应用
- elasticsearch 应用场景

ES 产生背景

图书馆图书目录手册	
索引书号	书名
A-19/23	java之降龙十八掌
B-28/12	大数据从入门到康复
B-39/80	javaweb之康复指南
C-29/10	php从入门到放弃
X-27/28	mysql从入门到删库
M-87/43	C++从入门到跑路

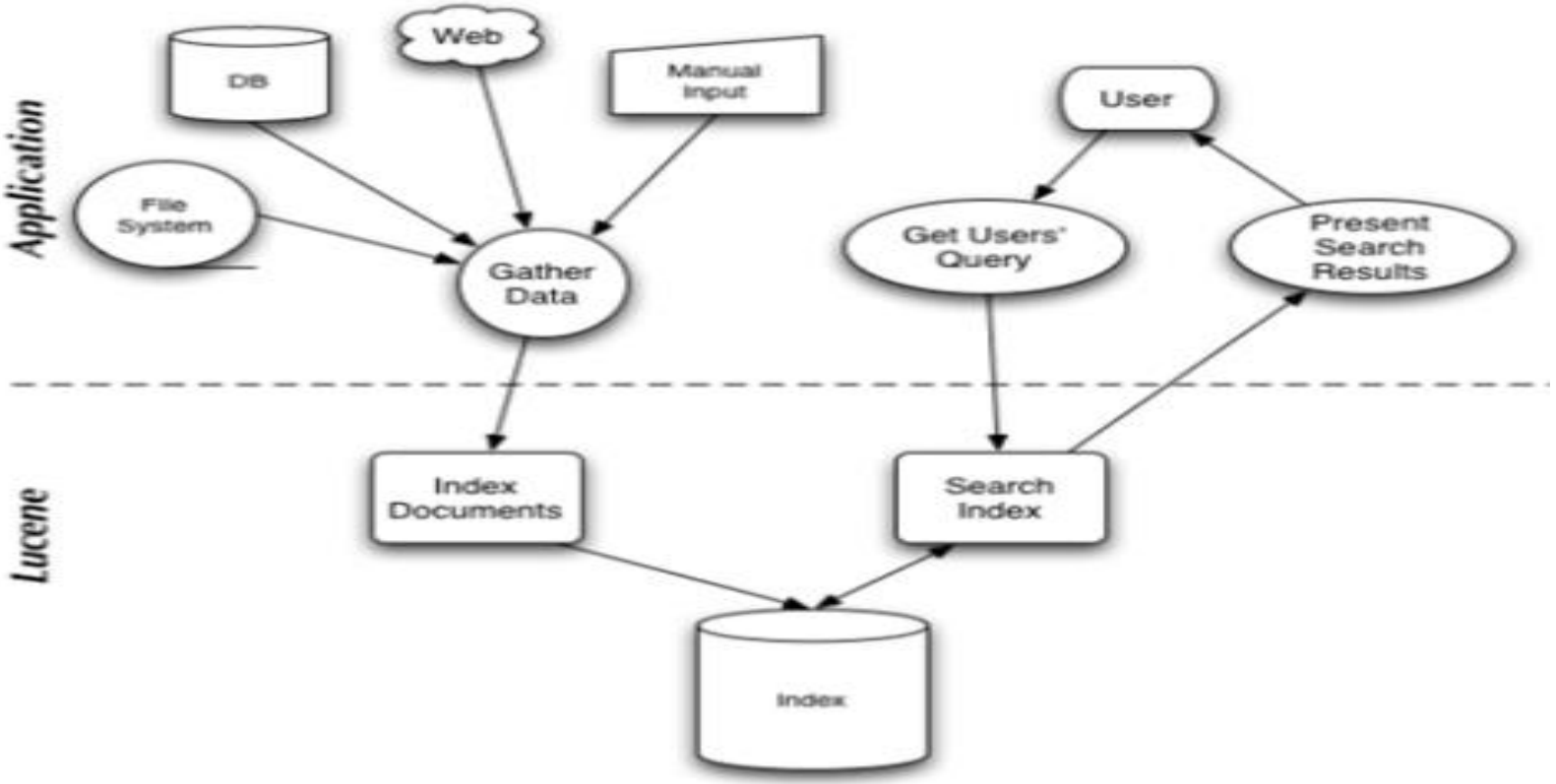
ES 产生背景

- 海量数据组合条件查询
- 毫秒级或者秒级返回数据



Lucene 定义

lucene是一个开放源代码的全文检索引擎工具包，但它不是一个完整的全文检索引擎，而是一个全文检索引擎的架构，提供了完整的查询引擎和索引引擎，部分文本分析引擎。
官网地址： http://lucene.apache.org/



ES 定义

ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java开发的，并作为Apache许可条款下的开放源码发布，是当前流行的企业级搜索引擎。设计用于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。

官网地址：https://www.elastic.co/

ES vs Lucene

- 成品与半成品的关系
- Lucene专注于搜索底层的建设，而ElasticSearch专注于企业应用。

Solr 定义

Solr是Apache 下的一个开源项目，使用Java基于Lucene开发的全文检索服务是一个独立的企业级搜索应用服务器，它对外提供类似于Web-service的API接口。用户可以通过http请求，向搜索引擎服务器提交一定格式的XML文件，生成索引；也可以通过Http Get操作提出查找请求，并得到XML格式的返回结果。

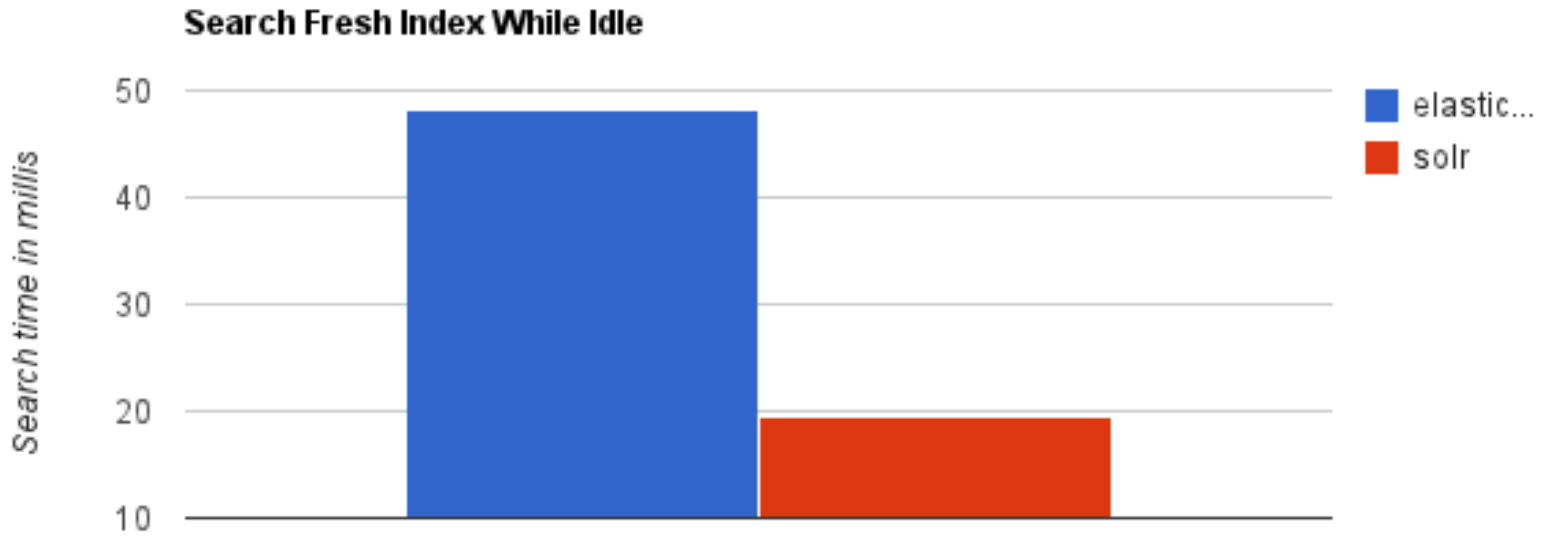
官网地址： https://lucene.apache.org/solr/

ES vs Solr 优缺点

	Solr	ES
优点	1）Solr有一个更大、更成熟的用户、开发和贡献者社区。 2）Solr支持多种数据格式的索引，比如：JSON、XML、CSV等多种数据格式。 3）Solr发展比较成熟、稳定。 4）Solr搜索海量历史数据，速度非常快，毫秒级返回数据。	1）分布式：节点对外表现对等，加入节点自动均衡。 2）Elasticsearch 完全支持 Apache Lucene 的接近实时的搜索。 3）处理多租户（multitenancy）不需要特殊配置，而Solr则需要更多的高级设置。 4）Elasticsearch 采用 Gateway 的概念，使得数据持久化更加简单。 5）各节点组成对等的网络结构，某些节点出现故障时会自动分配其他节点代替其进行工作。
缺点	Solr建立索引时，搜索效率下降，实时索引搜索效率不高。	更改数据格式，比较麻烦。

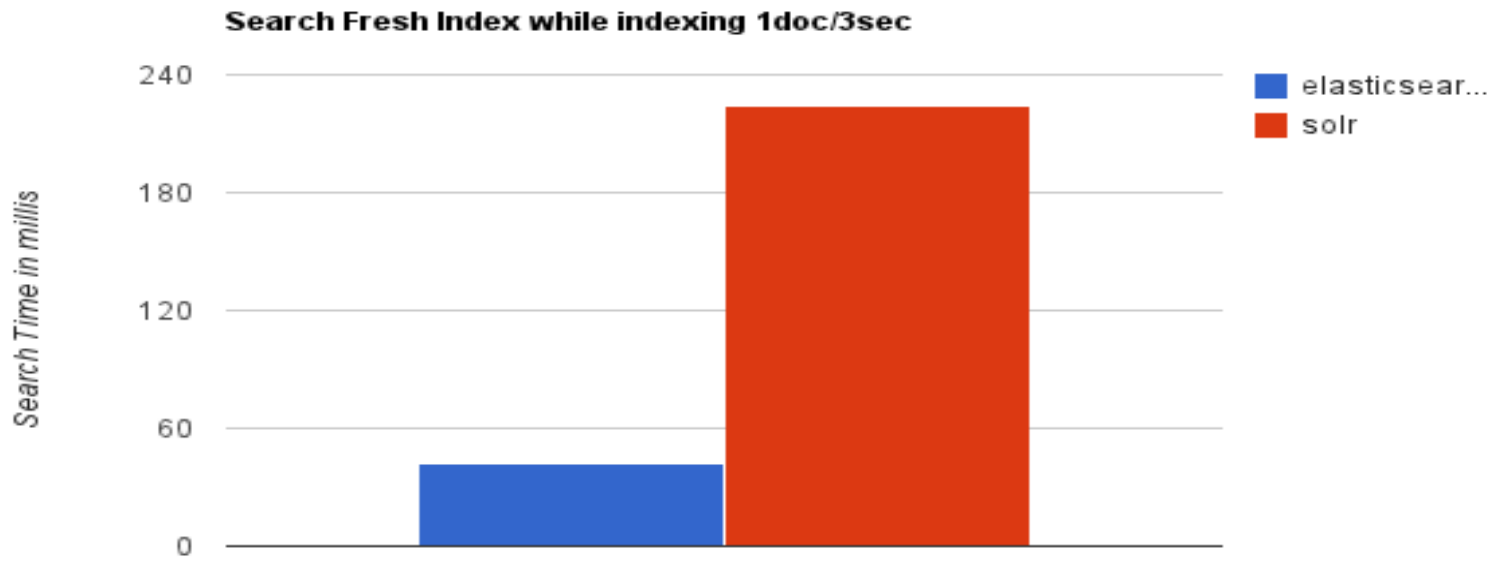
ES vs Solr 检索速度

当单纯的对已有数据进行搜索时，Solr更快。



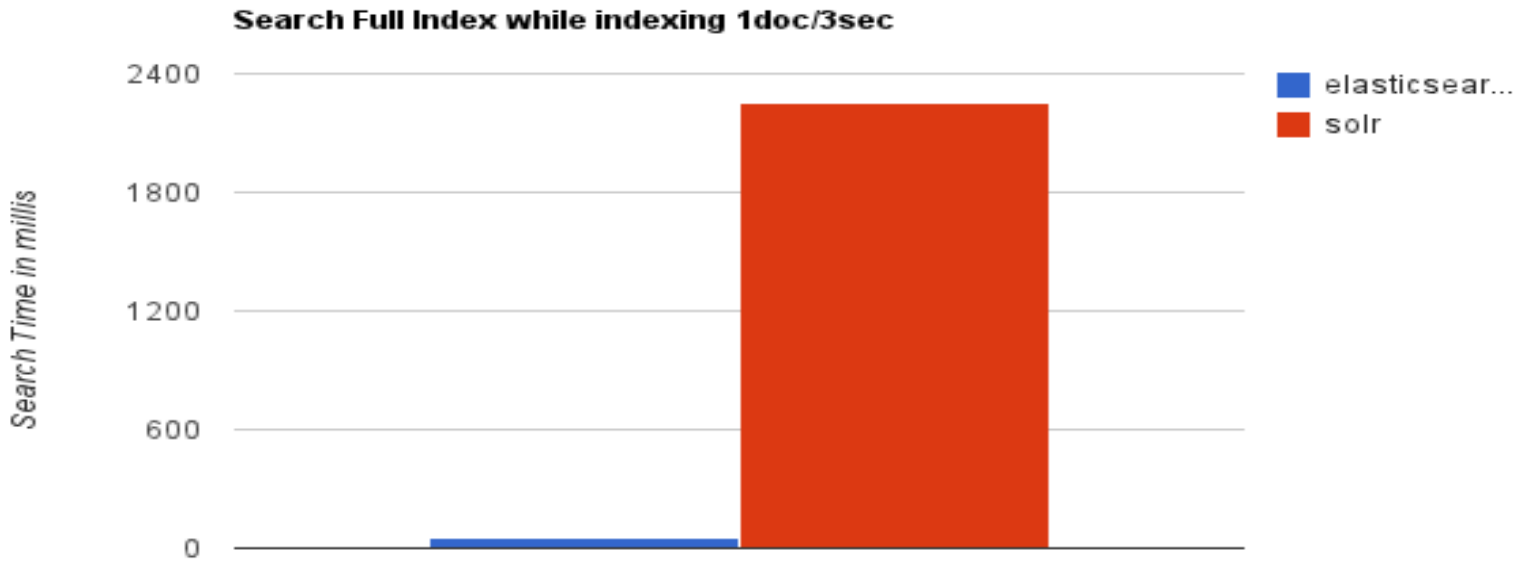
ES vs Solr 检索速度

当实时建立索引时, Solr会产生io阻塞，查询性能较差, Elasticsearch具有明显的优势。



ES vs Solr 检索速度

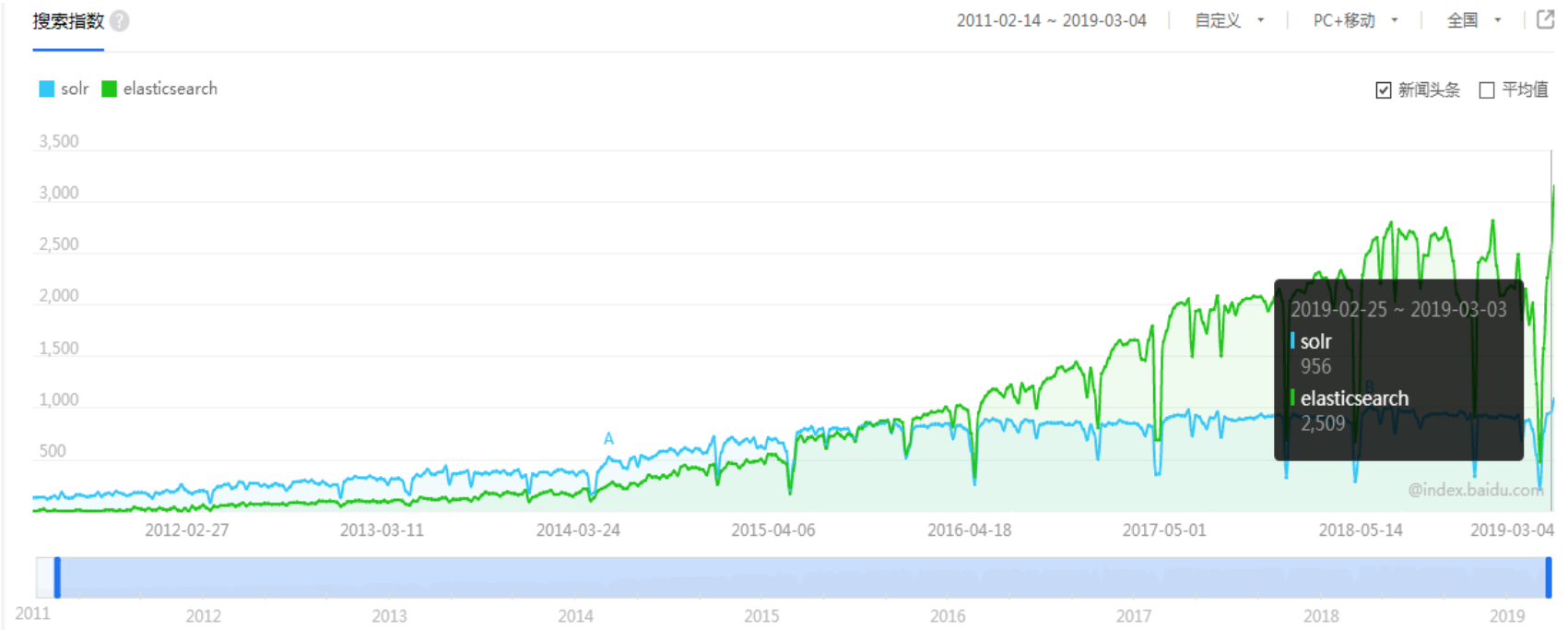
随着数据量的增加，Solr的搜索效率会变得更低，而Elasticsearch却没有明显的变化。



ES vs Solr 检索速度

大型互联网公司，实际生产环境测试，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。

ES vs Solr 热度



搜索指数概览

关键词	整体日均值	移动日均值	整体同比	整体环比	移动同比	移动环比
solr	589	90	-	-	-	-
elasticsearch	879	117	-	-	-	-

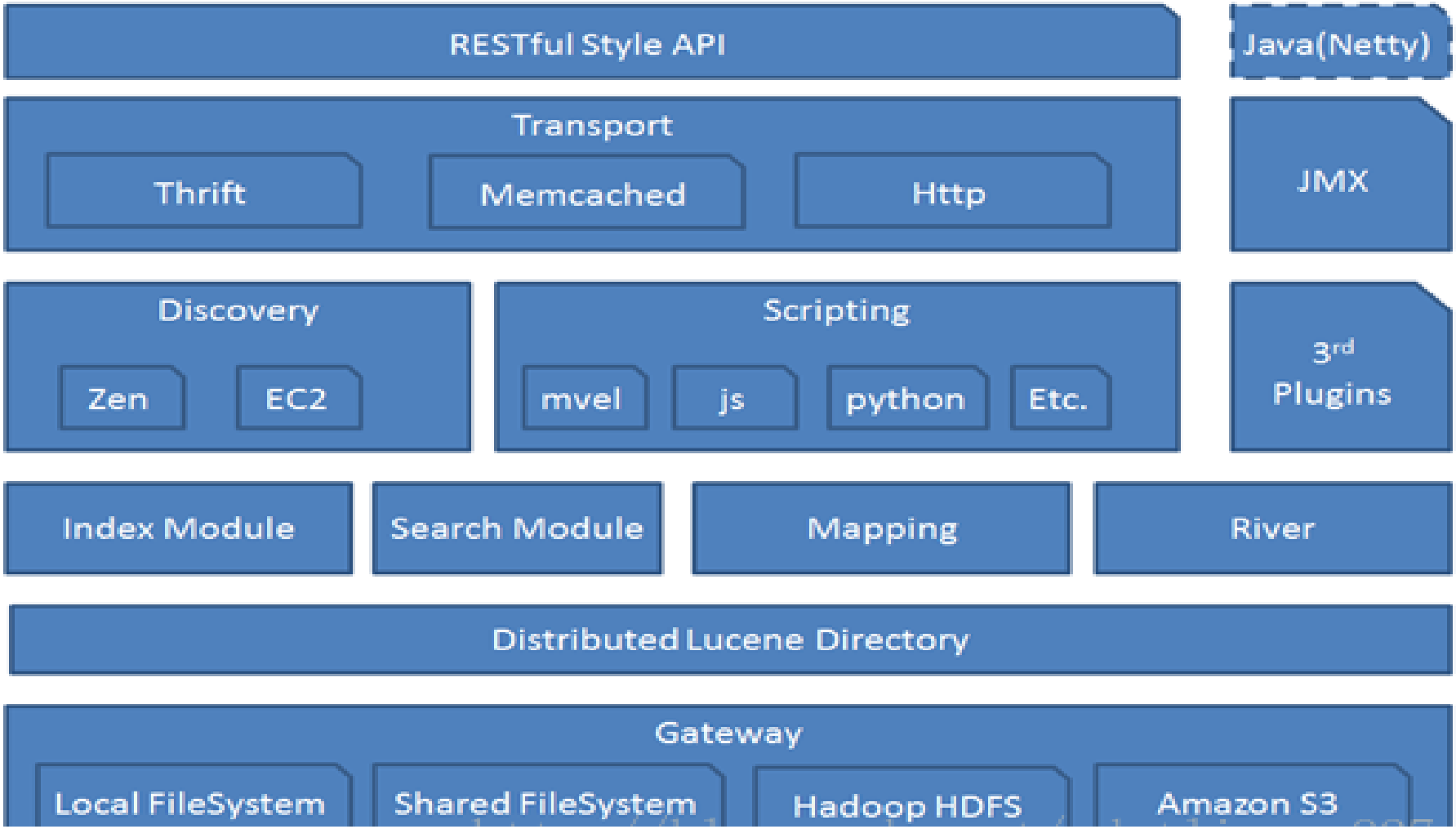
ES vs Solr 总结

- 二者安装都很简单。
- Solr 利用 Zookeeper 进行分布式管理，而 Elasticsearch 自身带有分布式协调管理功能。
- Solr 支持更多格式的数据，比如JSON、XML、CSV，而 Elasticsearch 仅支持json文件格式。
- Solr 官方提供的功能更多，而 Elasticsearch 本身更注重于核心功能，高级功能多有第三方插件提供
- Solr 在传统的搜索应用中表现好于 Elasticsearch，但在处理实时搜索应用时效率明显低于 Elasticsearch。
- Solr 是传统搜索应用的有力解决方案，但 Elasticsearch 更适用于新兴的实时搜索应用。

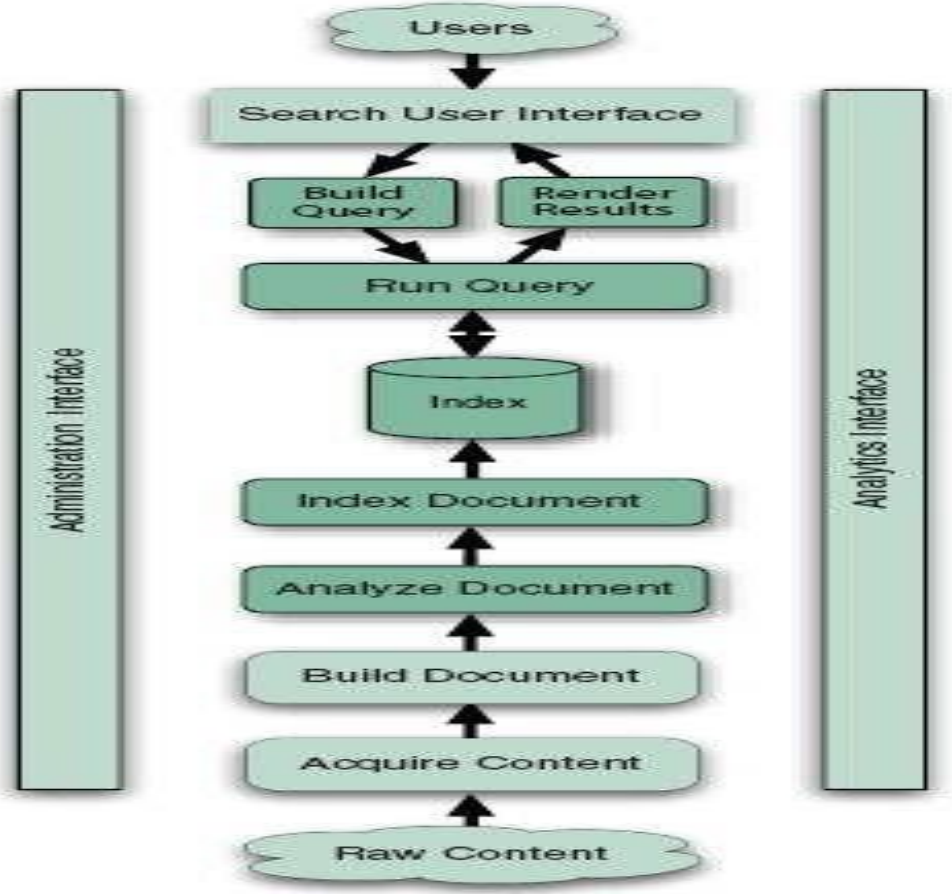
ES vs 关系型数据库

关系型数据库	Database （数据库）	Table （表）	Row （行）	Column （列）
ElasticSearch	Index （索引库）	Type （类型）	Document （文档）	Field （字段）

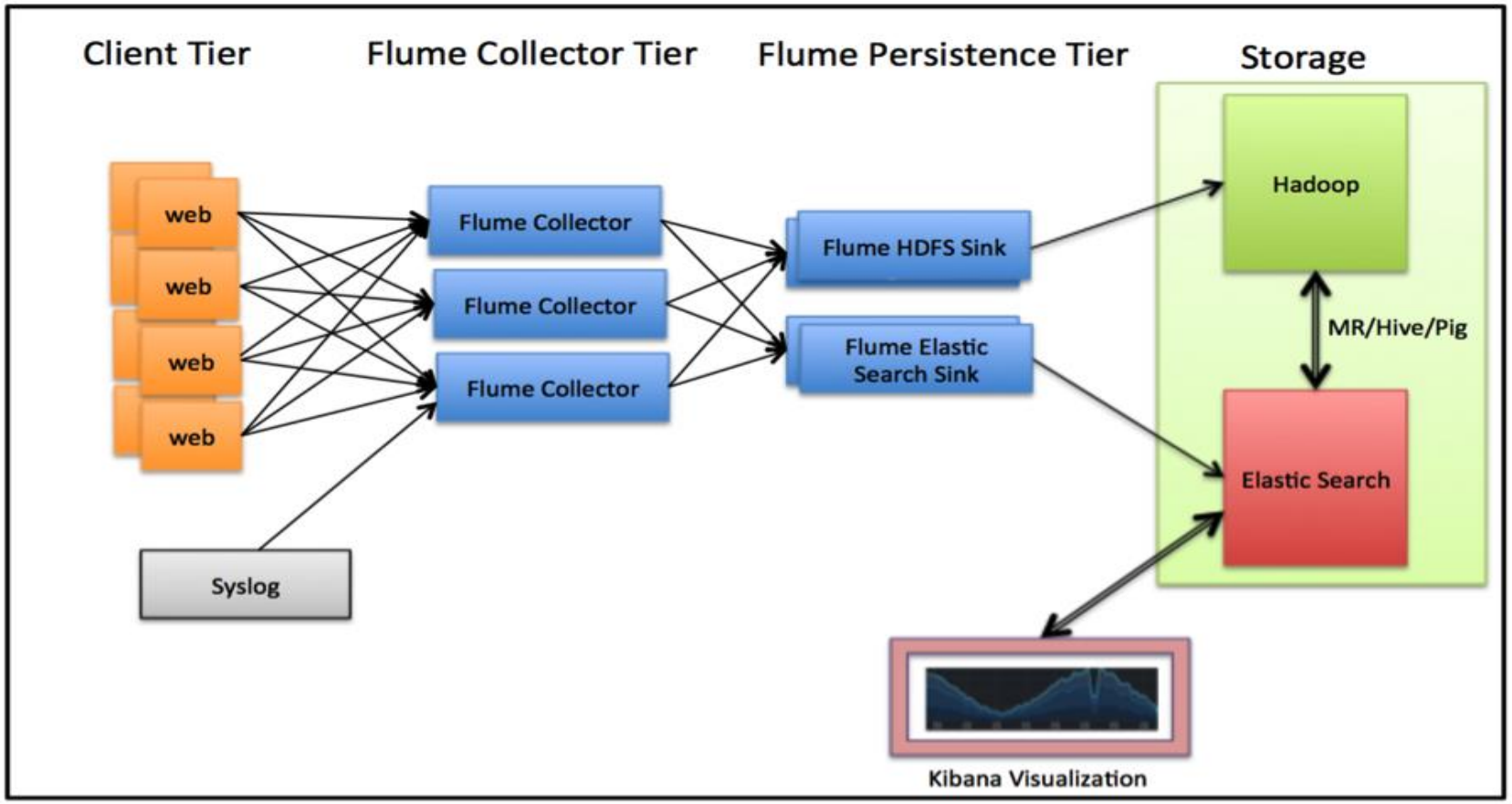
ES 架构


RESTful Style API
Java(Netty)
Transport
Thrift
Memcached
Http
JMX
Discovery
Zen
EC2
Scripting
mvel
js
python
Etc.
3rd Plugins
Index Module
Search Module
Mapping
River
Distributed Lucene Directory
Gateway
Local FileSystem
Shared FileSystem
Hadoop HDFS
Amazon S3

ES 工作原理


Users
Search User Interface
Build Query
Render Results
Run Query
Index
Index Document
Analyze Document
Build Document
Acquire Content
Raw Content
Administration Interface
Analytics Interface

ES 在大数据中的应用


Client Tier
Flume Collector Tier
Flume Persistence Tier
Storage
web
web
web
web
Syslog
Flume Collector
Flume Collector
Flume Collector
Flume HDFS Sink
Flume Elastic Search Sink
Hadoop
MR/Hive/Pig
Elastic Search
Kibana Visualization

ES 应用场景

- 站内搜索：主要和 Solr 竞争，属于后起之秀
- NoSQL Json文档数据库：主要抢占 Mongo 的市场，它在读写性能上优于 Mongo ，同时也支持地理位置查询，还方便地理位置和文本混合查询。
- 监控：统计、日志类时间序的数据存储和分析、可视化，这方面是引领者
- 国外：Wikipedia（维基百科）使用 ES 提供全文搜索并高亮关键字、Stack Overflow（IT问答网站）结合全文搜索与地理位置查询、Github使用Elasticsearch检索1300亿行的代码
- 国内：百度（在云分析、网盟、预测、文库、钱包、风控等业务上都应用了ES，单集群每天导入30TB+数据，总共每天60TB+）、新浪 、阿里巴巴、腾讯等公司均有对ES的使用
- 使用比较广泛的平台ELK(ElasticSearch, Logstash, Kibana)

ElasticSearch单节点安装

- JDK安装
- elasticsearch 下载安装
- elasticsearch 启动运行
- elasticsearch 测试验证
- elasticsearch head插件安装

ES的安装

- 零配置，开箱即用
- java版本要求：
  - 目前官方推荐1.8或以上版本：https://www.elastic.co/guide/en/elasticsearch/reference/current/setup.html
- es下载地址：
  - https://www.elastic.co/downloads/elasticsearch
- 启动
  - cd ...../elasticsearch-6.6.1
  - bin/elasticsearch
  - bin/elasticsearch -d(后台运行)

ES安装验证

- http://your ip:9200
  - 注意：默认启动的时候es绑定的网络ip是本机127.0.0.1，只能通过这个ip访问。
- 两种修改方式
  - 1：修改config/elasticsearch.yml文件

```
network.host: 192.168.20.210
```

  - 2：在启动es的时候指定参数

```
bin/elasticsearch -Dnetwork.host=192.168.80.100
```


```
{
  "name" : "OHu_XZN",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "n_zxSR90TqWmdD_AQfM3Fg",
  "version" : {
    "number" : "6.6.1",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "1fd8f69",
    "build_date" : "2019-02-13T17:10:04.160291Z",
    "build_snapshot" : false,
    "lucene_version" : "7.6.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

ES head插件

- ElasticSearch-head是一个H5编写的ElasticSearch集群操作和管理工具，可以对集群进行傻瓜式操作。
  - 显示集群的拓扑,并且能够执行索引和节点级别操作
  - 搜索接口能够查询集群中原始json或表格格式的检索数据
  - 能够快速访问并显示集群的状态.
  - 有一个输入窗口,允许任意调用RESTful API。
  - 5.0版本之前可以通过plugin脚本来安装，5.0之后可以独立运行。

Head 插件安装

- 安装head插件

因为head是一个用于管理Elasticsearch的web前端插件，该插件在es5版本以后采用独立服务的形式进行安装使用（之前的版本可以直接在es安装目录中直接安装），因为需要安装nodejs、npm。

- 安装nodejs、npm（root用户）

```
yum -y install nodejs npm
```

- 安装git

```
yum -y install git
```

- 下载head

```
git clone git://github.com/mobz/elasticsearch-head.git
```

- 安装head

```
cd elasticsearch-head/
npm install
```

报错： node npm install Error: CERT_UNTRUSTED

ssl验证问题，使用下面的命令取消ssl验证即可解决

```
npm config set strict-ssl false
```

Head 插件安装

- 配置head插件
  - 修改Gruntfile.js配置

增加hostname:‘*‘配置

[es@master elasticsearch-head-master]$ vi Gruntfile.js

```
connect: {
        server: {
                options: {
                        port: 9100,
                        base: '.',
                        keepalive: true,
                        hostname: '*'
                }
        }
}
```

Head 插件安装

- 修改head/_site/app.js文件

修改head连接es的地址（修改localhost为本机的IP地址）

[es@master _site]$ vi app.js

- Es配置

修改elasticsearch.yml，增加跨域的配置(需要重启es才能生效)

vi config/elasticsearch.yml

http.cors.enabled: true

http.cors.allow-origin: "*“

Head 插件安装

- 启动head插件

[es@master elasticsearch-head]$ cd node_modules/grunt/bin/
[es@master bin]$ ./grunt server &
[es@master ~]$ netstat -ntlp

```
Foreign Address          State       PID/Program name
0.0.0.0:*                LISTEN      2250/grunt
0.0.0.0:*                LISTEN      -
0.0.0.0:*                LISTEN      -
:::*                     LISTEN      2118/java
:::*                     LISTEN      2118/java
:::*                     LISTEN      -
:::*                     LISTEN      -
```

- Head查看es集群状态

http://192.168.20.210:9100/

ElasticSearch Rest基本操作

- REST介绍
- 创建索引库
- 查询索引
- DSL查询
- MGET查询
- HEAD使用
- 更新索引
- 删除索引
- 批量操作
- 版本控制

REST 定义

REST (REpresentation State Transfer)描述了一个架构样式的网络系统，比如 web 应用程序。它首次出现在 2000 年 Roy Fielding 的博士论文中，他是 HTTP 规范的主要编写者之一。REST 指的是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就是 RESTful。

REST 定义

Web 应用程序最重要的 REST 原则是，客户端和服务器之间的交互在请求之间是无状态的。从客户端到服务器的每个请求都必须包含理解请求所必需的信息。如果服务器在请求之间的任何时间点重启，客户端不会得到通知。此外，无状态请求可以由任何可用服务器回答，这十分适合云计算之类的环境。客户端可以缓存数据以改进性能。

在服务器端，应用程序状态和功能可以分为各种资源。资源是一个有趣的概念实体，它向客户端公开。资源的例子有：应用程序对象、数据库记录、算法等等。每个资源都使用 URI (Universal Resource Identifier) 得到一个惟一的地址。所有资源都共享统一的界面，以便在客户端和服务器之间传输状态。使用的是标准的 HTTP 方法，比如 GET、PUT、POST 和 DELETE。

REST 资源

资源	GET	PUT	POST	DELETE
一组资源的URL http://example.com/products/	列出URL列表	使用给定的一组资源替换当前组资源	在本资源组中创建或者追加一个新的资源	删除整组资源
单个资源的URL http://example.com/products/4554	获取指定资源的详细信息	替换或者创建指定资源	在资源组下创建或者追加一个新的元素	删除指定的元素

REST基本操作

- GET 获取对象的当前状态
- PUT 改变对象的状态
- POST 创建对象
- DELETE 删除对象
- HEAD 获取头信息

ES 内置的REST接口

URL	说明
/index/_search	搜索指定索引下的数据
/_aliases	获取或者操作索引下的别名
/index/	查看指定索引下的详细信息
/index/type/	创建或者操作类型
/index/mapping	创建或者操作mapping
/index/settings	创建或者操作settings
/index/_open	打开指定索引
/index/_close	关闭指定索引
/index/_refresh	刷新索引（使新增加内容对搜索可见，不保证数据被写入磁盘）
/index/_flush	刷新索引（会触发Lucene提交数据）

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的使用

➢ -X 指定http请求的方法GET POST PUT DELETE

➢ -d 指定要传递的参数

CURL 创建索引库

- curl -XPUT 'http://master:9200/test/'
- PUT/POST都可以
- 示例：

```
curl -H "Content-Type: application/json"  -XPOST
http://master:9200/test/user/1 -d '{"name" : "jack","age" : 28}'
```

PUT和POST的用法区别

- PUT是幂等方法，而POST并不是。所以PUT用于更新操作、POST用于新增操作比较合适。
- 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。

创建索引库注意事项

- 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 -H "Content-Type: application/json" -XPOST http://master:9200/test/user/ -d '{"name" : "john"}'
```

- 创建全新内容的两种方式：
  1）使用自增ID（post）
  2）在url后面添加参数

```
curl -H "Content-Type: application/json" -XPUT http://master:9200/test/user/2?op_type=create -d '{"name":"lucy","age":18}'

curl -H "Content-Type: application/json" -XPUT http://master:9200/test/user/3/_create -d '{"name":"lily","age":28}'
```

查询索引-GET

- 根据员工id查询

```
curl -XGET http://master:9200/test/user/1
```

- 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。

➢ 检索文档中的一部分，如果只需要显示指定字段

```
curl -XGET 'http://master:9200/test/user/1?_source=name&pretty'
```

➢ 查询指定索引库指定类型所有数据

```
curl -XGET http://master:9200/test/user/_search?pretty
```

- 根据条件进行查询

```
curl -XGET
'http://master:9200/test/user/_search?q=name:john&pretty=true‘
```

或者

```
curl -XGET
'http://master:9200/test/user/_search?q=name:john&pretty'
```

DSL查询

- Domain Specific Language领域特定语言
- 新添加一个文档

```
curl -H "Content-Type: application/json"  -XPUT
http://master:9200/test/user/4/_create -d '{"name":"qiqi","age":17}'

curl -H "Content-Type: application/json" -XGET
http://master:9200/test/user/_search -
d'{"query":{"match":{"name":"qiqi"}}}'
```

MGET查询

- 使用mget API获取多个文档

先新建一个库

```
curl -XPUT 'http://master:9200/test2/'
curl -H "Content-Type: application/json" -XPOST
http://master:9200/test2/user/1 -d '{"name" : "marry","age" : 16}'

curl -H "Content-Type: application/json" -XGET
http://master:9200/_mget?pretty -d
'{"docs":[{"_index":"test","_type":"user","_id":2,"_source":"name"},{"_index":"te
st2","_type":"user","_id":1}]}‘
```

- 如果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 -H "Content-Type: application/json" -XGET
http://master:9200/test/user/_mget?pretty -d '{"docs":[{"_id":1},{"_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 -H "Content-Type: application/json" -XGET
http://master:9200/test/user/_mget?pretty -d '{"ids":["1","2"]}'
```

HEAD的使用

- 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://master:9200/test/user/1
```

ES更新(1)

- ES可以使用PUT或者POST对文档进行更新(全部更新)，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

ES更新(2)

- 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 -H "Content-Type: application/json" -XPOST http://master:9200/test/user/1/_update -d '{"doc":{"name":"baby","age":27}}‘
  - curl -XGET http://master:9200/test/user/1?pretty

ES删除

- 删除操作

curl -XDELETE http://master:9200/test/user/1
curl -XGET http://master:9200/test/user/1

  - 如果文档存在，result属性值为deleted，_version属性的值+1
  - 如果文档不存在，result属性值为not_found，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。

ES批量操作-bulk

- 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 }}
{ request body        }
{ action: { metadata }}
{ request body        }
```

- 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

新建一个requests文件
vi requests

```
{"index":{"_index":"test","_type":"user","_id":"6"}}
{"name":"mayun","age":51}
{"update":{"_index":"test","_type":"user","_id":"6"}}
{"doc":{"age":52}}
```

执行批量操作

```
curl -H "Content-Type: application/json"  -XPOST http://master:9200/_bulk --data-binary @requests;

curl -XGET http://master:9200/test/user/6?pretty
```

ES批量操作-bulk

- 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

1）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.

2）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.

3）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.

4）https://www.elastic.co/guide/en/elasticsearch/reference/6.6/modules-http.html

ES版本控制

- 普通关系型数据库使用的是（悲观并发控制（PCC））

当我们在修改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)
  - 首先得到需要修改的文档，获取版本(_version)号

```
curl -XGET http://master:9200/test/user/2
```

  - 在执行更新操作的时候把版本号传过去

```
curl -H "Content-Type: application/json" -XPUT http://master:9200/test/user/2?version=1 -d '{"name":"john","age":29}'
curl -H "Content-Type: application/json" -XPOST http://master:9200/test/user/2/_update?version=2 -d '{"doc":{"age":30}}'
```

  - 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核心概念

- Cluster
- shards
- replicas
- recovery
- gateway
- discovery.zen
- Transport
- Setting
- Mapping

ES中的核心概念

- cluster

1）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。

2）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。

3）注意：主节点不负责对数据的增删改查请求进行处理，只负责维护集群的相关状态信息。

- 集群状态查看

http://192.168.20.210:9200/_cluster/health?pretty

ES中的核心概念

- Shards

1）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。把一个大的索引水平拆分成多个，分布到不同的节点上。构成分布式搜索，提高性能和吞吐量。

2）分片的数量只能在创建索引库时指定，索引库创建后不能更改。

```
curl -H "Content-Type: application/json"  -XPUT 'master:9200/test3/' -d '{"settings":{"number_of_shards":3}}'
```

- 默认是一个索引库有5个分片
- 每个分片中最多存储2,147,483,519条数据

https://www.elastic.co/guide/en/elasticsearch/reference/6.6/getting-started-concepts.html

ES中的核心概念

- Replicas

代表索引副本，es可以给索引分片设置副本，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。

【副本的数量可以随时修改】

- 可以在创建索引库的时候指定

```
curl -H "Content-Type: application/json" -XPUT 'master:9200/test4/' -d '{"settings":{"number_of_replicas":3}}‘
```

- 默认是一个分片有1个副本

```
index.number_of_replicas: 1
```

注意：主分片和副本不会存在一个节点中

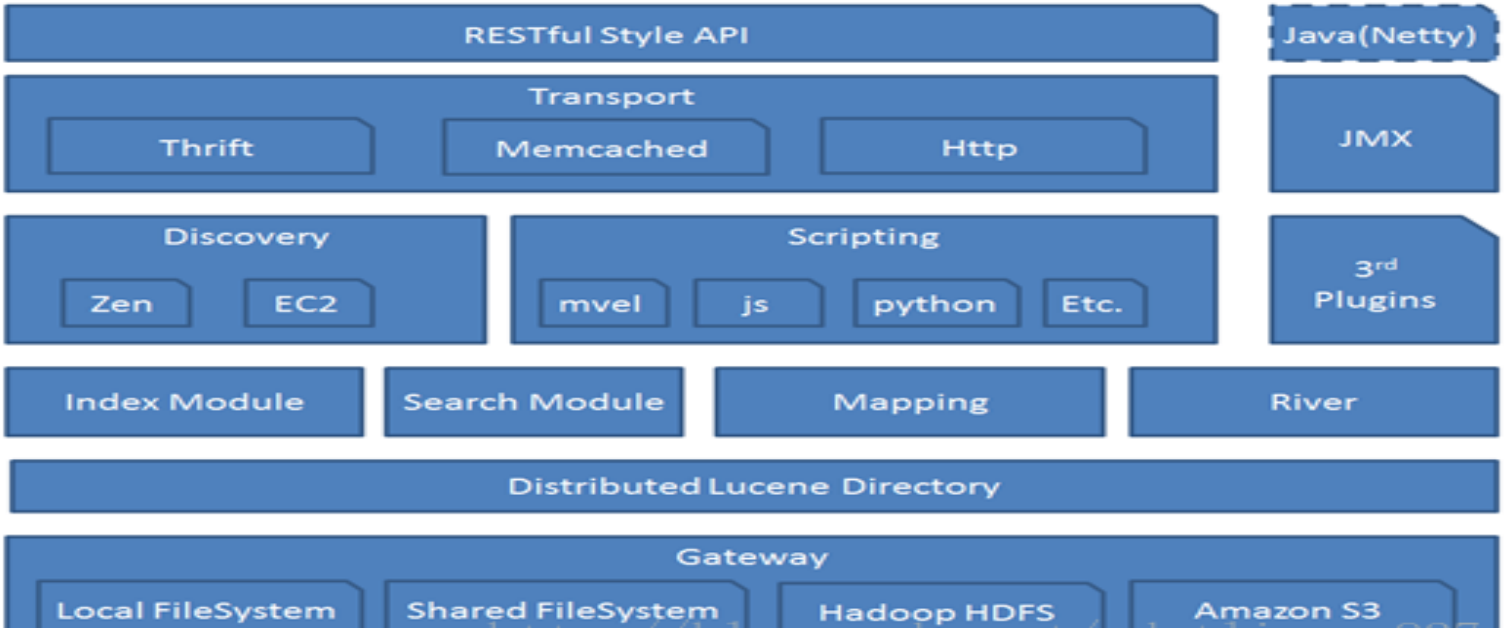
ES中的核心概念

- 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。

ES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。



ES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
```

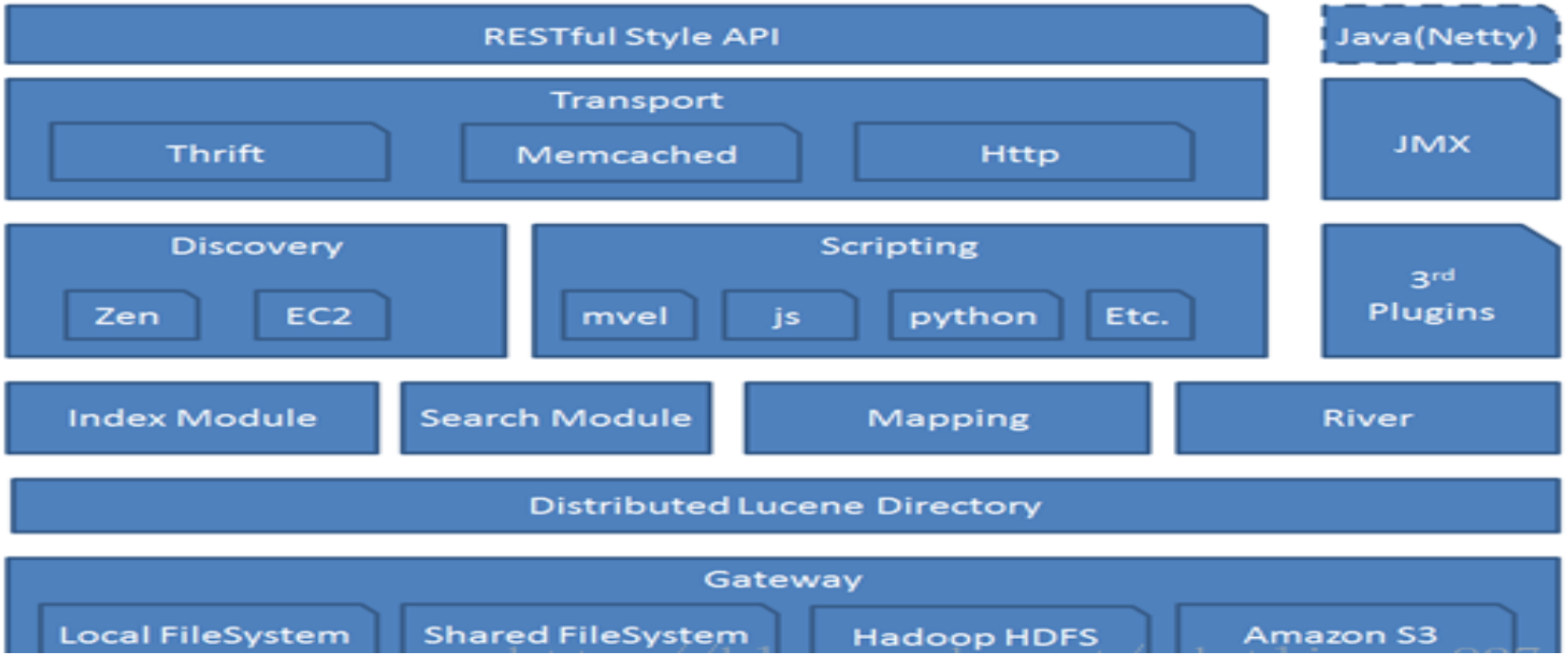
设置新节点被启动时能够发现的主节点列表

```
discovery.zen.ping.unicast.hosts: ["192.168.20.210", "192.168.20.211", "192.168.20.212"]
```

ES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）。



ES中的核心概念

- settings

https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-create-index.html

例如：分片数量，副本数量

- 查看：

```
curl -XGET http://master:9200/test/_settings?pretty
```

- 操作不存在索引(创建)：

```
curl -H "Content-Type: application/json"  -XPUT
'http://master:9200/test5/' -d
'{"settings":{"number_of_shards":3,"number_of_replicas":2}}'
```

- 操作已存在索引（修改）：

```
curl -H "Content-Type: application/json" -XPUT
'http://master:9200/test5/_settings' -d
'{"index":{"number_of_replicas":1}}'
```

ES中的核心概念

Mapping

https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html

- 查询索引库的mapping信息：

```
curl -XGET http://master:9200/test/user/_mapping?pretty
```

- 操作不存在的索引（创建）：

```
curl -H "Content-Type: application/json" -XPUT
'http://master:9200/test6' -d
'{"mappings":{"user":{"properties":{"name":{"type":"text","an
alyzer": "ik_max_word"}}}}}'
```

- 操作已存在的索引（修改）：

```
curl -H "Content-Type: application/json" -XPOST
http://master:9200/test6/user/_mapping -d
'{"properties":{"name":{"type":"text","analyzer":
"ik_max_word"}}}'
```

ElasticSearch Java 客户端

- Java High Level REST Client
- Document API
- SearchType
- 查询详解
- 分页
- 多索引和多类型查询
- 极速查询

ES-JAVA客户端-依赖

- 添加maven依赖

Java api地址：https://www.elastic.co/guide/en/elasticsearch/client/java-api/6.6/index.html

```
<dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>4.10</version>
                        <scope>test</scope>
</dependency>
<dependency>
            <groupId>org.elasticsearch.client</groupId>
            <artifactId>elasticsearch-rest-high-level-client</artifactId>
            <version>6.6.1</version>
</dependency>
```

ES-JAVA客户端-端口

- ES 9200端口与9300端口的区别

es启动监听两个端口：9300和9200。

9300是Tcp协议端口：通过tcp协议通讯，ES集群之间是通过9300进行通讯，java客户端（TransportClient）的方式是也是以tcp协议在9300端口上与集群进行通信。

9200是Http协议端口：主要用于外部通讯，外部使用RESTful接口进行访问。

ES-JAVA客户端-Client

- Java High Level REST Client与Transport Client

ES Java Client地址：
https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/client.html

Transport Client：
https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/transport-client.html

High Level REST Client：
https://www.elastic.co/guide/en/elasticsearch/client/java-rest/6.6/java-rest-high.html

ES-JAVA客户端-Client

- Java High Level REST Client

Java高级别REST客户端（The Java High Level REST Client）以后简称高级客户端，内部仍然是基于低级客户端。它提供了更多的API，接受请求对象作为参数并返回响应对象，由客户端自己处理编码和解码。
https://www.elastic.co/guide/en/elasticsearch/client/java-rest/6.6/index.html

每个API都可以同步或异步调用。 同步方法返回一个响应对象，而异步方法的名称以async后缀结尾，需要一个监听器参数，一旦收到响应或错误，就会被通知（由低级客户端管理的线程池）。

高级客户端依赖于Elasticsearch core项目。它接受与TransportClient相同的请求参数并返回相同的响应对象。

ES-JAVA客户端-Client兼容性

- Java High Level REST Client兼容性

高级客户端需要Java 1.8并依赖于Elasticsearch core项目。 客户端版本需要与Elasticsearch集群版本相同。 它与TransportClient请求的参数和返回响应对象相同。

如果您需要将应用程序从TransportClient迁移到新的REST客户端：https://www.elastic.co/guide/en/elasticsearch/client/java-rest/current/java-rest-high-level-migration.html

客户端要与Elasticsearch集群进行通信，主版本号需要一致，次版本号不必相同，因为它是向前兼容的。客户端次版本号小于等于elasticsearch集群的都可以。这意味着它支持客户端与更高版本的Elasticsearch集群进行通信。

6.0客户端能够与任何6.x Elasticsearch节点通信，而6.1客户端肯定能够与6.1,6.2和任何后来的6.x版本进行通信，但与旧版本的Elasticsearch节点通信时可能会存在不兼容的问题，例如6.1和6.0之间，可能6.1客户端支持elasticsearch 6.0还没出来的API。

ES-JAVA客户端-连接集群

- 连接到es集群

RestHighLevelClient实例需要低级客户端构建器来构建，如下所示：

```
RestHighLevelClient client = new RestHighLevelClient(
                                RestClient.builder(
                                        new HttpHost("192.168.20.210", 9200, "http")));
```

高级客户端将在内部创建低级客户端，用来执行基于提供的构建器的请求，并管理其生命周期。

当不再需要时，需要关闭高级客户端实例，以便它所使用的所有资源以及底层的http客户端实例及其线程得到正确释放。可以通过close方法来完成，该方法将关闭内部的RestClient实例。

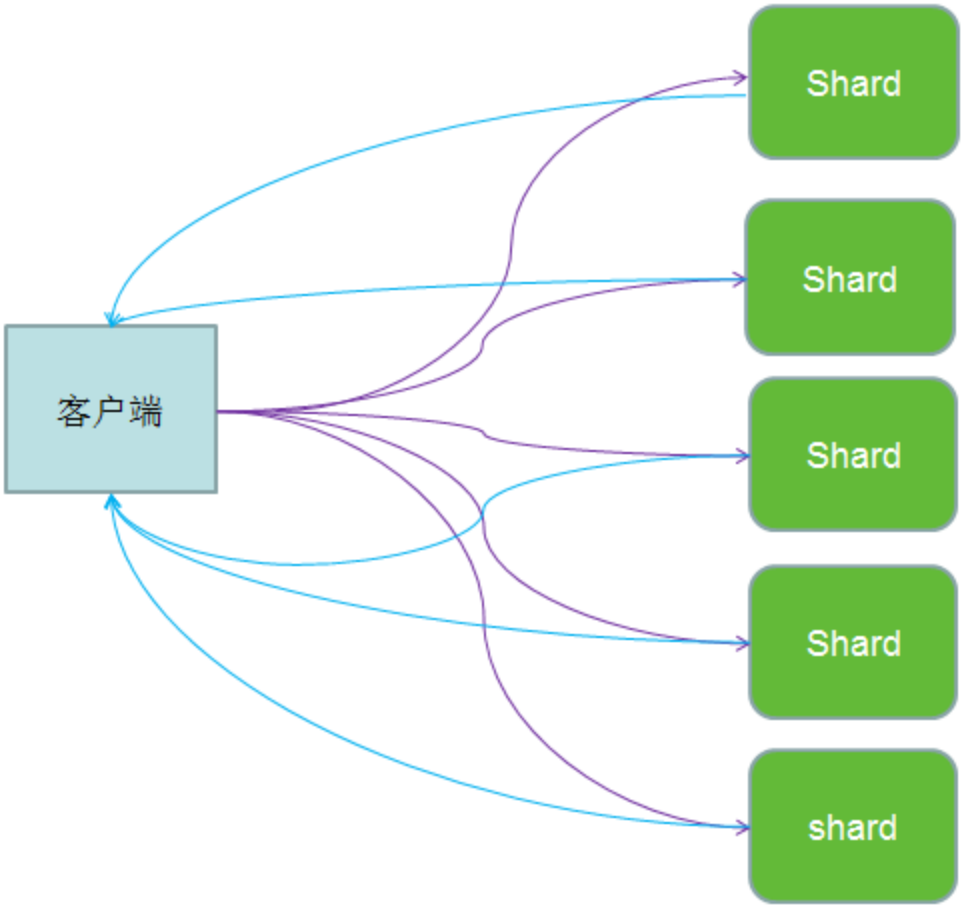
```
client.close();
```

ES-JAVA客户端-Document API

- 索引index（四种json,map,XContentBuilder,object）
  - IndexResponse indexResponse = client.index(indexRequest, RequestOptions.DEFAULT);
- 查询get
  - GetResponse getResponse = client.get(getRequest, RequestOptions.DEFAULT);
- 判断exist
  - boolean exists = client.exists(getRequest, RequestOptions.DEFAULT);
- 更新update
  - UpdateResponse updateResponse = client.update(request, RequestOptions.DEFAULT);
- 删除delete
  - DeleteResponse deleteResponse = client.delete(request, RequestOptions.DEFAULT);
- 批量操作bulk
- 多条数据查询Multi Get

ES-JAVA客户端-SearchType

- 搜索原理



ES-JAVA客户端-SearchType

搜索问题

- 返回数据数量问题

如果数据分散在默认的5个分片上，ES会向5个分片同时发出请求，每个分片都返回10条数据，最终会返回总数据为：5*10=50条数据，远远大于用户请求。

- 返回数据排名问题

每个分片计算符合条件的前10条数据都是基于自己分片的数据进行打分计算的。计算分值（score）使用的词频和文档频率等信息都是基于自己分片的数据进行的，而ES进行整体排名是基于每个分片计算后的分值进行排序的(打分依据就不一致，最终对这些数据统一排名的时候就不准确了)

ES-JAVA客户端-SearchType

- 搜索问题-解决思路

➢ 返回数据数量问题

第一步：先从每个分片汇总查询的数据id，进行排名，取前10条数据。

第二步：根据这10条数据id，到不同分片获取数据。

➢ 返回数据排名问题

将各个分片打分标准统一。

ES-JAVA客户端-SearchType

SearchType类型1-query and fetch

- 实现原理

向索引的所有分片（shard）都发出查询请求，各分片返回的时候把元素文档（document）和计算后的排名信息一起返回。

- 优点

这种搜索方式是最快的。因为相比后面的几种搜索方式，这种查询方法只需要去shard查询一次。

- 缺点

各个shard返回的结果的数量之和可能是用户要求的size的n倍。

https://www.elastic.co/guide/en/elasticsearch/reference/2.4/search-request-search-type.html

ES-JAVA客户端-SearchType

SearchType类型2-query then fetch

- 实现原理

第一步，先向所有的shard发出请求，各分片只返回文档id(注意，不包括文档document)和排名相关的信息(也就是文档对应的分值)，然后按照各分片返回的文档的分数进行重新排序和排名，取前size个文档。

第二步，根据文档id去相关的shard取document。这种方式返回的document数量与用户要求的大小是相等的。

- 优点：

返回的数据量是准确的。

- 缺点：

数据排名不准确且性能一般。

ES-JAVA客户端-SearchType

- SearchType类型3-DFS query and fetch

这种方式比第一种类型多了一个DFS步骤，它可以更精确控制搜索打分和排名。

- 实现原理

第一步：先对所有分片发送请求，把所有分片中的词频和文档频率等打分依据全部汇总到一块。

第二步：然后再执行后面的操作后续操作

- 优点：

数据排名准确。

- 缺点：

搜索性能一般，且返回的数据量不准确，可能返回(N*分片数量)的数据。

ES-JAVA客户端-SearchType

- SearchType类型4-DFS query then fetch

  比第2种方式多了一个DFS步骤。

- 实现原理

  第一步：先对所有分片发送请求，把所有分片中的词频和文档频率等打分依据全部汇总到一块。

  第二步：然后再执行后面的操作后续操作

- 优点：

  返回的数据量是准确的，数据排名也是准确的。

- 缺点：

  性能最差【这个最差只是表示在这四种查询方式中性能最慢，也不至于不能忍受，如果对查询性能要求不是非常高，而对查询准确度要求比较高的时候可以考虑这个】

ES-JAVA客户端-SearchType

- 从性能考虑来说，QUERY_AND_FETCH是最快的，DFS_QUERY_THEN_FETCH是最慢的。

- 从搜索的准确度来说，DFS要比非DFS的准确度更高。

- ES6.x进行了优化只保留了第2、4类型

https://www.elastic.co/guide/en/elasticsearch/reference/6.6/search-request-search-type.html

https://www.elastic.co/guide/en/elasticsearch/reference/1.4/search-request-search-type.html

ES-JAVA客户端-SearchType

- SearchType示例操作
https://www.elastic.co/guide/en/elasticsearch/client/java-rest/current/java-rest-high-search.html

```
@Test
public void test1() throws IOException {
    SearchRequest searchRequest = new SearchRequest();
    SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();
    searchSourceBuilder.query(QueryBuilders.termQuery("field", "baz"));
    searchRequest.indices("posts");
    searchRequest.source(searchSourceBuilder);
    searchRequest.searchType(SearchType.DFS_QUERY_THEN_FETCH);

    SearchResponse searchResponse = client.search(searchRequest, RequestOptions.DE
    SearchHits hits = searchResponse.getHits();
    System.out.println(hits.getTotalHits());
    client.close();
}
```

ES查询详解-query

- 查询：query
  - 分页：from/size
  - 排序：sort
  - 过滤：filter
  - 按查询匹配度排序:Explain
  - 高亮显示：highlight（后面结合实例讲解）

ES查询详解-聚合

- 统计:aggregations
  - 根据字段进行分组统计
  - 根据字段分组，统计其他字段的值

aggregations 统计实例一

- 统计相同年龄的学员个数

姓名	年龄
tom	18
jack	29
jessica	18
dave	19
lilei	18
lili	29

代码一

```
public void test1() throws IOException {
    SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();
    // 按年龄分组聚合统计
    TermsAggregationBuilder aggregation = AggregationBuilders.terms("by_age").field("age");
    searchSourceBuilder.query(QueryBuilders.matchAllQuery())
            .aggregation(aggregation);

    SearchRequest searchRequest = new SearchRequest();
    searchRequest.indices("test6");
    searchRequest.source(searchSourceBuilder);
    SearchResponse searchResponse = client.search(searchRequest, RequestOptions.DEFAULT);
    // 获取分组信息
    Terms terms = searchResponse.getAggregations().get("by_age");
    for (Terms.Bucket entry : terms.getBuckets()) {
        Object key = entry.getKey();
        long docCount = entry.getDocCount();
        System.out.println(key + "@" + docCount);
    }
}
```

aggregations 统计实例二

- 统计每个学员的总成绩

姓名	科目	分数
tom	语文	59
tom	数学	89
jack	语文	78
jack	数学	85
jessica	语文	97
jessica	数学	68

代码二

```
public void test2() throws IOException {
    SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();
    // 按姓名分组聚合统计
    TermsAggregationBuilder nameAggregation = AggregationBuilders
            .terms("by_name")
            .field("name.keyword");
    SumAggregationBuilder scoreAggregation = AggregationBuilders.sum("by_score").field("score");
    nameAggregation.subAggregation(scoreAggregation);
    searchSourceBuilder.query(QueryBuilders.matchAllQuery())
                       .aggregation(nameAggregation);

    SearchRequest searchRequest = new SearchRequest();
    searchRequest.indices("test7");
    searchRequest.source(searchSourceBuilder);
    SearchResponse searchResponse = client.search(searchRequest, RequestOptions.DEFAULT);
    // 获取分组信息
    Terms terms = searchResponse.getAggregations().get("by_name");
    for (Terms.Bucket entry : terms.getBuckets()) {
        Sum sum = entry.getAggregations().get("by_score");
        System.out.println(entry.getKey() + "@" + sum.getValue());
    }
}
```

ES中的分页

- SQL 语句分页：limit m,n

m：从哪条结果开始
n：size：每次返回多少个结果

- Elasticsearch使用的是from以及size两个参数

from：从哪条结果开始，默认值为0
size：每次返回多少个结果，默认值为10

- 假设每页显示5条结果，那么1至3页的请求就是

GET /_search?size=5
GET /_search?size=5&from=5
GET /_search?size=5&from=10

- 注意

不要一次请求过多或者页码过大的结果，这么会对服务器造成很大的压力。因为它们会在返回前排序。一个请求会经过多个分片。每个分片都会生成自己的排序结果。然后再进行集中整理，以确保最终结果的正确性。

ES中的分页


Bai du 百度
ElasticSearch集群安装部署
百度一下
网页 资讯 视频 图片 知道 文库 贴吧 采购 地图 更多»
百度为您找到相关结果约10,900,000个
搜索工具
elasticsearch 集群的安装部署 - LittlePony - 博客园
2019年3月1日 - elasticsearch 是居于lucene的搜索引擎,可以横向集群扩展以及分片,开发者无需关注如何实现了索引的备份,集群同步,分片等,我们很容易通过简单的配置就...
https://www.cnblogs.com/maybo/... - 百度快照
Elasticsearch集群部署 - Aubin - 博客园
2018年9月3日 - 三、部署Elasticsearch集群 1.安装JDK Elasticsearch是基于Java开发是一个Java程序,运行在Jvm中,所以第一步要安装JDK yum install -y java-1.8.0-open...
https://www.cnblogs.com/aubin/... - 百度快照
ElasticSearch5.6.3的安装部署以及集群部署、ElasticSe..._CSDN博客
2017年11月3日 - 二、集群安装、 这里使用三台机器模拟ElasticSearch集群,这三台的ip地址是如下: "192.168.0.153" "192.168.0.154" "192.168.0.155" 上面我已近在192...
CSDN博客号 - 百度快照
ElasticSearch集群部署与head插件安装 - liu123641191的博客 - ...
2018年7月8日 - Elasticsearch部署(使用3个物理节点,分别作为:master、slave1、

ES中的分页


Bai du 百度
ElasticSearch集群安装部署
百度一下
2017年11月5日 - 解压完毕 /usr/local/elasticsearch-5.6.3是我的安装目录 image.png 介绍一下...cluster.name 配置: elasticsearch会通过相同cluster.name组成一个集...
https://www.jianshu.com/p/2e25... - 百度快照
在kubernetes集群中部署efk-ylw6006-51CTO博客
2018年2月20日 - 本文将介绍在kubernetes1.9集群下配置elasticsearch、fluentd、kibana集中收集k8s集群日志信息。俗称EFK,其中elasticsearch负责存储日志。fluentd负责...
https://blog.51cto.com/ylw6006... - 百度快照
搭建Elasticsearch 5.4分布式集群 - ◕‿◕ - CSDN博客
2018年6月4日 - 访问head,一个master一个slave组成集群,界面如下: 四、基于5.4的单机多节点集群配置 如果想要在一台机器上启动多个节点,步骤如下: 复制一份ELasticsea...
CSDN博客号 - 百度快照
ElasticSearch 安装报错整理 - 简书
2018年1月8日 - 单机安装报错 初次启动服务 当使用root账户调用启动命令出现错误信息,错误提示信息如下: 创建新用户 由于Elasticsearch可以接收用户输入的脚本并且执...
https://www.jianshu.com/p/365d... - 百度快照
CentOS7下安装elasticsearch-head插件 - 简书
2018年12月10日 - Head是elasticsearch的集群管理工具,可以用于数据的浏览和查询
(1)elasticsearch-head是一款开源软件,被托管在github上面,所以如果我们要使用它,必须...
https://www.jianshu.com/p/8e2e... - 百度快照
2018-10-25 elasticsearch.yml 配置详解 - 有点意思_3539 - 简书
2018年10月25日 - 下面主要讲解下elasticsearch.yml这个文件中可配置的东西。

ES多索引和多类型查询

```
@Test
public void test3() throws IOException {
    SearchRequest searchRequest = new SearchRequest();
    SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder(
    searchSourceBuilder
                    .query(QueryBuilders.matchAllQuery())
                    .size(100);
    //searchRequest.indices("test");
    //searchRequest.indices("test","test2");//可以指定多个索引库
    searchRequest.indices("test*");//索引可以使用通配符
    searchRequest.types("class","user");//支持多个类型，但不支持通配符
    searchRequest.source(searchSourceBuilder);
    SearchResponse searchResponse = client.search(searchRequest, Reque
    SearchHits hits = searchResponse.getHits();
    SearchHit[] hits2 = hits.getHits();
    for (SearchHit searchHit : hits2) {
        System.out.println(searchHit.getSourceAsString());
    }
```

ES极速查询

- Es将数据存储在不同的分片中，根据文档id通过内部算法得出要将文档存储在哪个分片上，所以在查询时只要指定在对应的分片上进行查询就可以实现基于es的极速查询。

- 知道数据在那个分片上，是解决问题的关键。
  - 实现方式：我们可以通过路由参数来设置数据存储在同一个分片中，setRouting("")

org.elasticsearch.cluster.routing——OperationRouting——indexShards

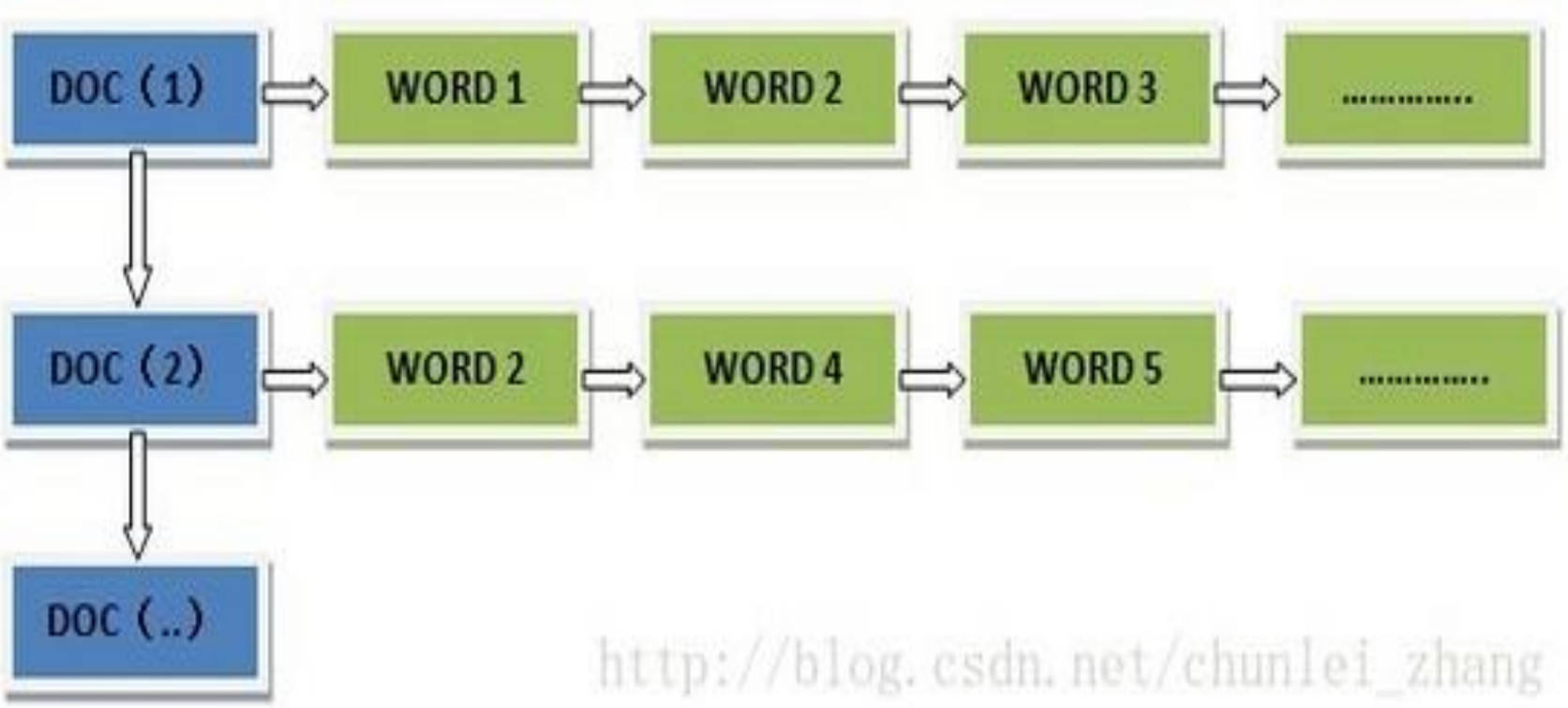
ES极速查询

```
@Test
public void test5() throws IOException {
    SearchRequest searchRequest = new SearchRequest();
    SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();
    searchSourceBuilder
                    .query(QueryBuilders.matchQuery("age", "28"))
                    ;
    searchRequest.indices("test8");
    searchRequest.types("user");
    searchRequest.routing("18903762536".substring(0, 3));
    searchRequest.source(searchSourceBuilder);
    SearchResponse searchResponse = client.search(searchRequest, Request
    SearchHits hits = searchResponse.getHits();
    SearchHit[] hits2 = hits.getHits();
    for (SearchHit searchHit : hits2) {
        System.out.println(searchHit.getSourceAsString());
    }
```

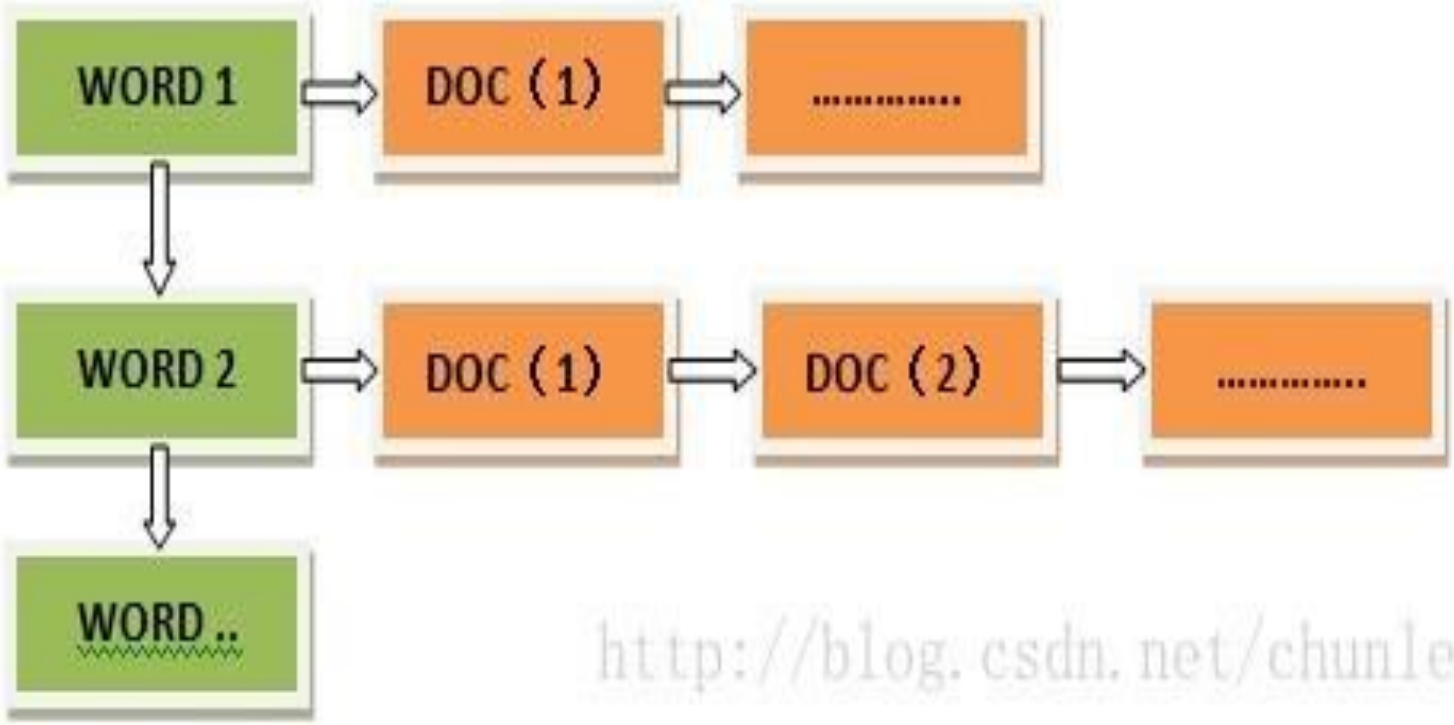
ElasticSearch索引模块

- 正排索引
- 倒排索引
- 索引模块组成部分
- 索引过程
- 停用词
- 中文分词器
- 常见的中文分词器
- 集成IK中文分词插件
- 自定义IK词库
- 热更新IK词库

Es索引模块-正排索引

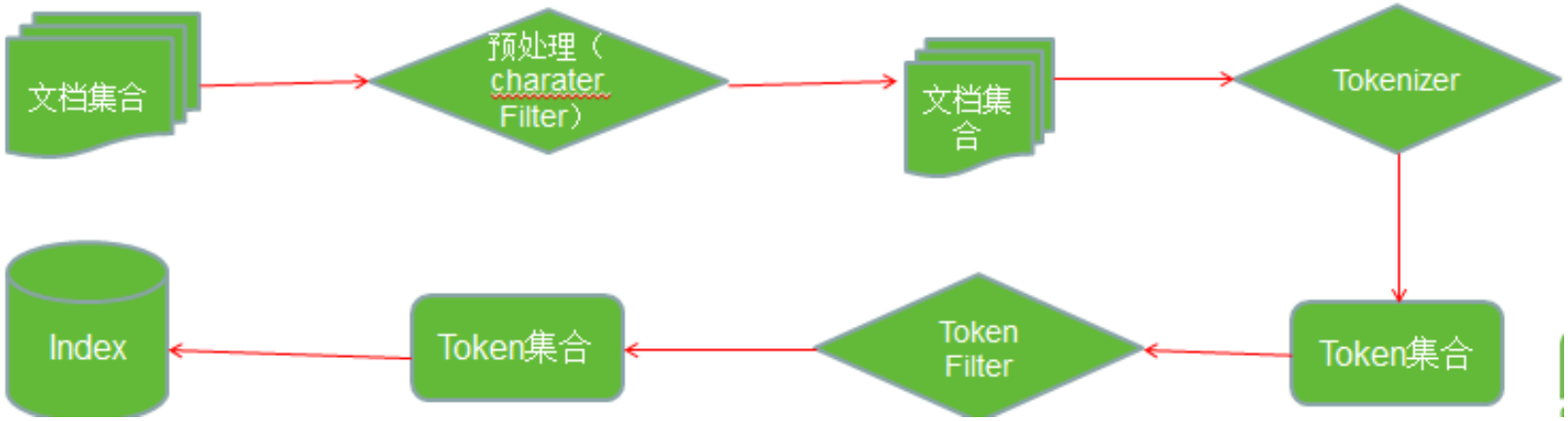


Es索引模块-倒排索引



ES索引模块-重要组成部分

- 索引分析模块Analyzer
  https://www.elastic.co/guide/en/elasticsearch/reference/6.6/analysis.html
  - 分解器(Tokenizer)：
  - 词元过滤器(token filters)
- 索引建立模块Indexer
  - 在建立索引过程中，分析处理过的文档将被加入到索引列表。事实上，Lucene为此仅提供了一个非常简单的API，而后自行内生地完成了此过程的所有功能。



ES索引模块-索引过程（1）

DOCID	DESCRIBE
NO.1	Tom likes eating apple and banana ,tony likes apple too.
NO2	Tom likes apple and orange .

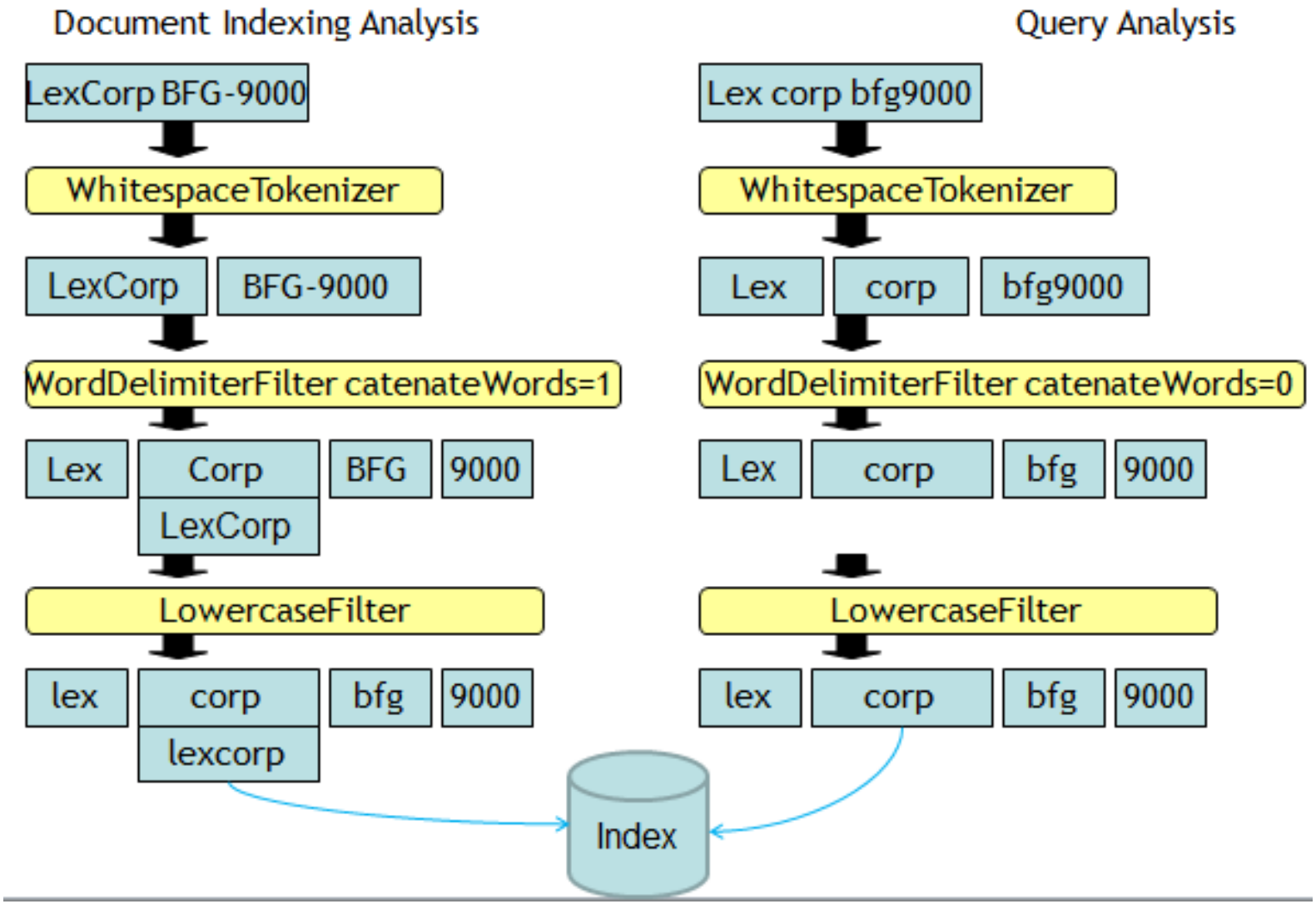
ES索引模块-索引过程（2）

DOCID	DESCRIBE
NO.1	[tom] [like] [eat] [apple] [banana] [tony] [like] [apple]
NO2	[tom] [like] [apple] [orange]

ES索引模块-索引过程（3）

Word	Describe(DOCID[FREQUENCE] , POSITION)
apple	NO.1[2] NO.2[1] , 4 9 3
banana	NO.1[1] , 6
eat	NO.1[1] , 3
like	NO.1[2] NO.2[1] , 2 8 2
orange	NO.2[1] , 5
tom	NO.1[1] NO.2[1] , 1 1

ES索引模块-索引和搜索



ES索引模块-停用词

有些词在文本中出现的频率非常高，但是对文本所携带的信息基本不产生影响，这样的词我们称之为停用词。

- 英文

a、an、the、of and so on

http://www.ranks.nl/stopwords/chinese-stopwords

- 中文

的、了、着、是等标点符号

http://www.ranks.nl/stopwords

- 用法
  - 文本经过分词之后，停用词通常被过滤掉，不会被进行索引。
  - 在检索的时候，用户的查询中如果含有停用词，检索系统也会将其过滤掉
- 优点
  - 排除停用词可以加快建立索引的速度，减小索引库文件的大小。

ES索引模块-中文分词器

- 单字分词：

比如：”我们是中国人”
效果：“我”“们”“是”“中”“国”“人”

- 二分法分词：

比如：”我们是中国人”
效果：“我们”“们是”“是中”“中国”“国人”

- 词库分词：

按某种算法构造词，然后去匹配已建好的词库集合，如果匹配到就切分出来成为词语。通常词库分词被认为是最理想的中文分词算法。

ES索引模块-常见的中文分词器

- StandardAnalyzer(单字分词)

能够根据空格、符号、数字、字母、E-mail地址、IP地址以及中文字符的分析处理分割原始的文本信息。但中文文字没有完成中文分词的功能，只是按照单个的汉字进行了分割。

- CJKAnalyzer(二分法)

专门用于中文文档处理的分析器，可以实现中文的多元切分和停用词过滤。

- IKAnalyzer(词库分词)

当前比较流行中文分词器，对中文支持较好。属于第三方插件，需要安装。

- 测试默认的分词对中文的支持

```
curl -H "Content-Type: application/json" -XGET 'http://master:9200/_analyze?pretty=true' -d '{"text":"我们是中国人"}'
```

ES索引模块-集成IK中文分词插件

- 下载ES的IK插件

https://github.com/medcl/elasticsearch-analysis-ik

- 使用maven对下载的插件进行源码编译（提前安装maven）

```
mvn clean package  -DskipTests
```

- 拷贝和解压release下的文件: target/releases/elasticsearch-analysis-ik-*.zip 到你的 elasticsearch 插件目录：ES_HOME/plugins/ik
- 重启ElasticSearch服务
- 测试分词效果：

```
curl -H 'Content-Type:application/json' -XGET http://master:9200/_analyze?pretty  -d '{"analyzer": "ik_max_word", "text": "我们是中国人"}'
```

```
curl -H 'Content-Type:application/json' -XGET http://master:9200/_analyze?pretty  -d '{"analyzer": "ik_smart", "text": "我们是中国人"}'
```

ES索引模块-自定义IK词库

- 创建自定义词库

测试新词：curl -H 'Content-Type:application/json' -XGET http://master:9200/_analyze?pretty -d '{"analyzer": "ik_max_word", "text": "蓝瘦香菇"}'

首先在ik插件的config/custom目录下创建一个文件test.dic，在文件中添加词语即可，每一个词语一行。

- 修改ik的配置文件IKAnalyzer.cfg.xml

将test.dic添加到ik的配置文件中即可

vi config/IKAnalyzer.cfg.xml

<entry key="ext_dict">custom/test.dic</entry>

- 重启ElasticSearch服务

bin/elasticsearch

- 测试分词效果：

curl -H 'Content-Type:application/json' -XGET http://master:9200/_analyze?pretty -d '{"analyzer": "ik_max_word", "text": "蓝瘦香菇"}'

ES索引模块-热更新IK词库

- 部署http服务，安装tomcat
  - 切换到

/home/hadoop/app/apache-tomcat-7.0.67/webapps/ROOT

  - 新建热词文件

vi hot.dic

么么哒

  - 需正常访问

bin/startup.sh

http://192.168.20.210:8080/hot.dic

- 修改ik插件的配置文件

vi config/IKAnalyzer.cfg.xml

添加如下内容

<entry key="remote_ext_dict">http://192.168.20.210:8080/hot.dic</entry>

分发修改后的配置到其他es节点

- 重启es，可以看到加载热词库

bin/elasticsearch

- 测试动态添加热词

对比添加热词之前和之后的变化

curl -H 'Content-Type:application/json' -XGET http://master:9200/_analyze?pretty -d '{"analyzer": "ik_max_word", "text": "老司机"}'

ElasticSearch集群安装部署

- ES规划
- ES环境配置
- JDK安装
- ES集群安装部署
- X-Pack安装
- Kibana安装

ES规划-集群节点

	master	slave1	slave2
ElasticSearch	是	是	是
Kibana	是		
Head	是		

ES规划-用户

	ElasticSearch	Kibana
master	es用户	es/root
slave1	es用户	
slave2	es用户	

ES规划-软件

软件	版本	位数
Linux	Centos6.5	64位
Jdk	Jdk1.8	64位
ElasticSearch	ElasticSearch6.6.1	较新版本
Kibana	kibana-6.6.1-linux-x86_64	较新版本

ES规划-目录

软件目录	/home/es/app/
数据目录	/home/es/data/es6.6/data
日志目录	/home/es/data/es6.6/datalog
进程pid目录	/home/es/data/es6.6/pid

ES环境配置-时钟同步

- 设置本地时间

cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

- 集群时间日期同步NTP

yum install ntp
ntpdate pool.ntp.org

ES环境配置-Hosts

- 配置hosts文件

vi /etc/hosts

```
192.168.20.210      master
192.168.20.211      slave1
192.168.20.212      slave2
```

ES环境配置-防火墙

- 查看防火墙状态

service iptables status

- 关闭防火墙

chkconfig iptables off　　//永久关闭防火墙
service iptables stop　　//临时关闭防火墙

JDK安装

- 下载

ES集群各个节点统一下载jdk-8u51-linux-x64.tar.gz

- 安装配置

各个节点统一配置jdk环境变量

vi ~/.bashrc

```
JAVA_HOME=/home/es/app/jdk
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
PATH=$JAVA_HOME/bin:$PATH
```

ES集群安装部署

- 下载

官网下载：https://www.elastic.co/downloads/past-releases

- 解压

tar -zxvf elasticsearch-6.6.1.tar.gz

- 配置

修改config/elasticsearch.yml配置文件

```
vi elasticsearch.yml
cluster.name: yunwei-es
node.name: master
path.data: /home/es/data/es6.6/data
path.logs: /home/es/data/es6.6/datalog
network.host: 192.168.20.210
node.master: true
node.data: true
discovery.zen.ping.unicast.hosts: ["192.168.20.210",
"192.168.20.211", "192.168.20.212"]
discovery.zen.ping_timeout: 10s
discovery.zen.minimum_master_nodes: 3
```

ES集群安装部署

ES其他节点配置

使用如下命令将ES安装目录同步其他节点。

scp -r elasticsearch-6.6.1 es@slave1:/home/es/app/
scp -r elasticsearch-6.6.1 es@slave2:/home/es/app/

修改配置

vi elasticsearch.yml
node.name: slave1
network.host: 192.168.20.211

创建规划目录

mkdir -p /home/es/data/es6.6/data
mkdir -p /home/es/data/es6.6/datalog

ES集群安装部署

- 启动

ES集群各个节点分别执行如下命令：
bin/elasticsearch

- 查看ES节点状态

http://192.168.20.210:9200/


```
{
  "name" : "master",
  "cluster_name" : "yunwei-es",
  "cluster_uuid" : "DEoIxfAvQRSMPJUSbS5Iag",
  "version" : {
    "number" : "6.6.1",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "1fd8f69",
    "build_date" : "2019-02-13T17:10:04.160291Z",
    "build_snapshot" : false,
    "lucene_version" : "7.6.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```


- 查看ES集群状态

http://192.168.20.210:9200/_cluster/health?pretty


```
{
  "cluster_name" : "yunwei-es",
  "status" : "green",
  "timed_out" : false,
  "number_of_nodes" : 3,
  "number_of_data_nodes" : 3,
  "active_primary_shards" : 0,
  "active_shards" : 0,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 0,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 100.0
}
```

X-Pack

X-Pack是一个Elastic Stack的扩展，将安全，警报，监视，报告，图形功能和机器学习包含在一个易于安装的软件包中。您可以在 Elastic Stack 中放心地使用这些功能。

在Elasticsearch 5.0.0之前，您必须安装单独的Shield，Watcher和Marvel插件才能获得在X-Pack中所有的功能。

X-Pack 提供以下几个级别保护elastic集群:

- 安全防护功能：不想让别人直接访问你的5601，9200端口。
- 实时监控功能：实时监控集群的CPU、磁盘等负载。
- 生成报告功能：图形化展示你的集群使用情况。
- 高级功能：包含机器学习等。

ES安装X-Pack

在Es的根目录（每个节点）执行如下命令：
bin/elasticsearch-plugin install x-pack
地址：
https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-xpack.html
备注：ES6.6默认已经安装X-Pack，无需再安装。

◘ES启用X-Pack

- 启动ES

bin/elasticsearch

- 启用trial license（30天试用）

地址：
https://www.elastic.co/guide/en/elasticsearch/reference/6.6/start-trial.html
选择其中一个节点执行如下命令：
curl -H "Content-Type:application/json" -XPOST http://master:9200/_xpack/license/start_trial?acknowledge=true

可以看到elasticsearch控制台显示license 已变为trial

ES安装X-Pack

- 配置elasitcsearch.yml

各个ES节点添加一下配置：

vi elasticsearch.yml

xpack.security.enabled: true

然后重新启动ES。

- 设置用户名和密码

bin/elasticsearch-setup-passwords interactive

```
[es@master elasticsearch-6.6.1]$ bin/elasticsearch-setup-passwords interactive
Initiating the setup of passwords for reserved users elastic,apm_system,kibana,
You will be prompted to enter passwords as the process progresses.
Please confirm that you would like to continue [y/N]y


Enter password for [elastic]:
Reenter password for [elastic]:
Enter password for [apm_system]:
Reenter password for [apm_system]:
Enter password for [kibana]:
Reenter password for [kibana]:
Enter password for [logstash_system]:
Reenter password for [logstash_system]:
Enter password for [beats_system]:
Reenter password for [beats_system]:
Enter password for [remote_monitoring_user]:
Reenter password for [remote_monitoring_user]:
Changed password for user [apm_system]
Changed password for user [kibana]
Changed password for user [logstash_system]
Changed password for user [beats_system]
Changed password for user [remote_monitoring_user]
Changed password for user [elastic]
[es@master elasticsearch-6.6.1]$
```

ES安装X-Pack

修改密码方式：
curl -H "Content-Type:application/json" -XPOST -u elastic 'http://192.168.20.210:9200/_xpack/security/user/elastic/_password' -d '{ "password" : "123456" }'

参考地址：
https://www.elastic.co/guide/en/elasticsearch/reference/current/security-api-change-password.html

- Web访问ES
http://192.168.20.210:9200/_cluster/health?pretty

用户名：elastic
密码：123456

Kibana安装

- 下载

下载地址：https://www.elastic.co/downloads/past-releases

- 解压

tar -zxvf kibana-6.6.1-linux-x86_64.tar.gz

- 修改配置

修改kibana.yml配置文件
vi kibana.yml
server.port: 5601
server.host: "0.0.0.0“
elasticsearch.hosts: ["http://192.168.20.210:9200"]
elasticsearch.username: "elastic"
elasticsearch.password: “123456“ #之前设置的elastic的密码

- 安装X-Pack

Kibana6.6默认已经安装过x-pack(bin/kibana-plugin install x-pack)

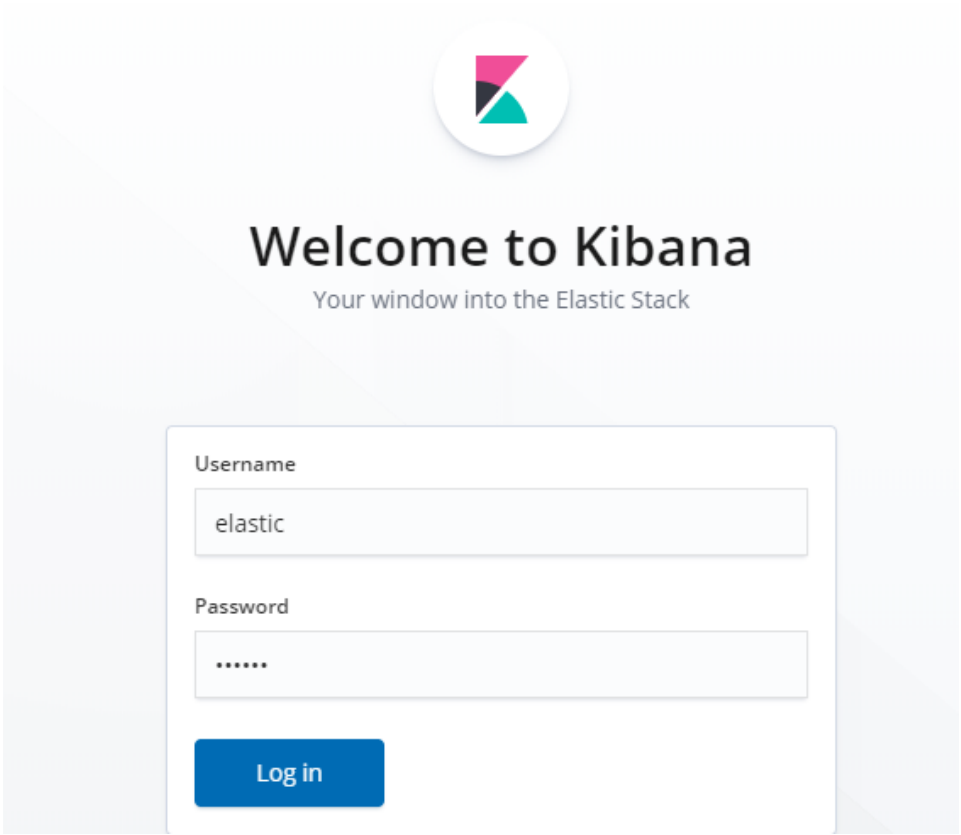
Kibana安装

- ☑ 启动

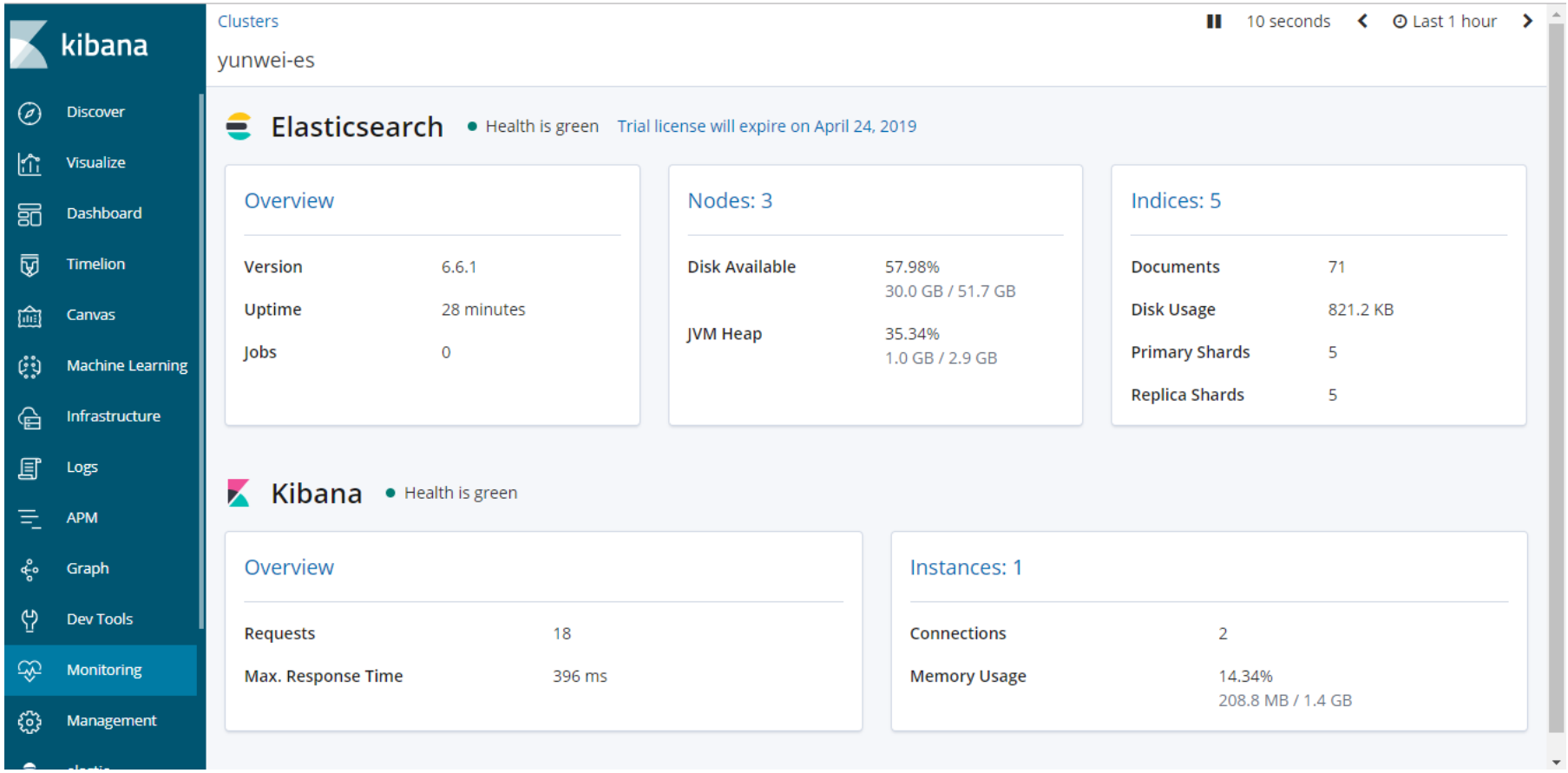
bin/kibana

- ☑ Web访问Kibana

http://192.168.20.210:5601（用户名：elastic 密码：123456）



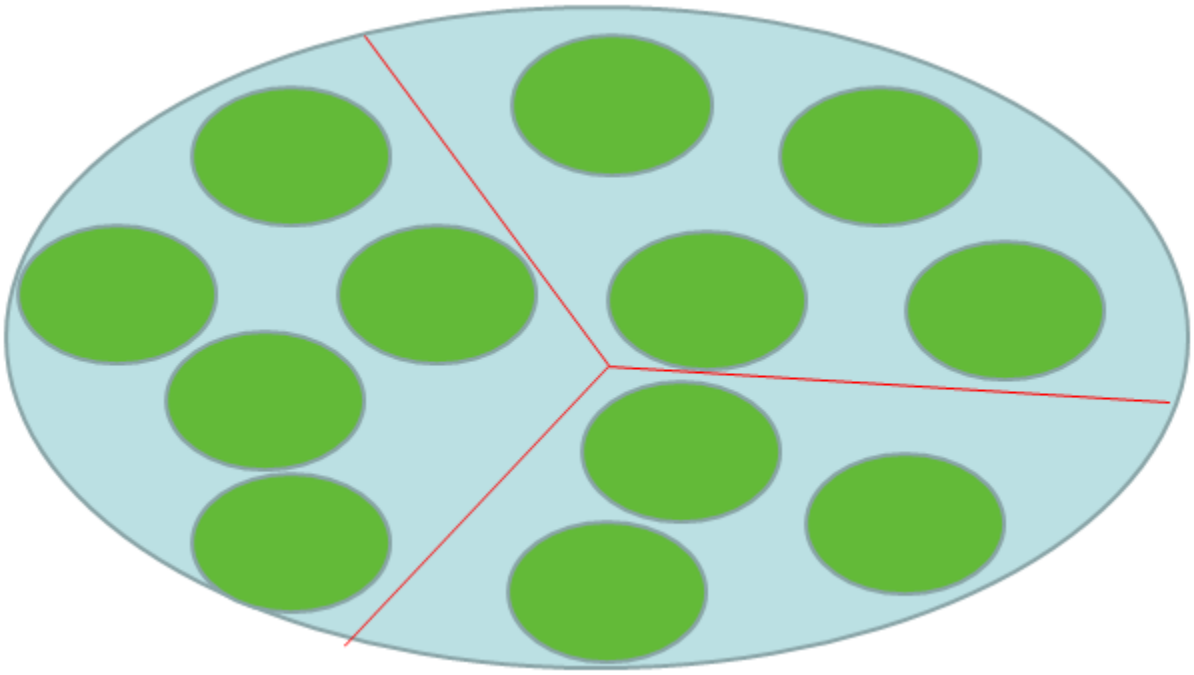
Kibana安装


kibana
Discover
Visualize
Dashboard
Timelion
Canvas
Machine Learning
Infrastructure
Logs
APM
Graph
Dev Tools
Monitoring
Management
Clusters
10 seconds
Last 1 hour
yunwei-es
Elasticsearch
Health is green
Trial license will expire on April 24, 2019
Overview
Version 6.6.1
Uptime 28 minutes
Jobs 0
Nodes: 3
Disk Available 57.98%
30.0 GB / 51.7 GB
JVM Heap 35.34%
1.0 GB / 2.9 GB
Indices: 5
Documents 71
Disk Usage 821.2 KB
Primary Shards 5
Replica Shards 5
Kibana
Health is green
Overview
Requests 18
Max. Response Time 396 ms
Instances: 1
Connections 2
Memory Usage 14.34%
208.8 MB / 1.4 GB

ElasticSearch优化

- 集群脑裂优化设置
- 增大系统打开文件数
- 合理设置JVM内存
- 锁定物理内存
- 合理设置分片
- 合理设置副本数
- 合并索引
- 关闭索引
- 清除删除文档
- 合理数据导入
- 设置索引_all
- 设置索引_source
- 版本一致

ES优化(1)-集群脑裂优化设置



ES优化(2)

◘增大系统打开文件数

调大系统的“最大打开文件数”，建议32K甚至是64K。

ulimit -a （查看）

ulimit -n 32000(设置)

```
[root@master ~]# ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 22609
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 65536
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 10240
cpu time               (seconds, -t) unlimited
max user processes              (-u) 4096
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
[root@master ~]#
```

ES优化(3)

◘合理设置JVM内存

修改配置文件调整ES的JVM内存大小。

修改jvm.options中-Xms和-Xmx的大小，
建议设置一样大，避免频繁的分配内存。根据服务器内存大小，一般分配60%左右(默认1g)

```
################################################################
## IMPORTANT: JVM heap size
################################################################
##
## You should always set the min and max JVM heap
## size to the same value. For example, to set
## the heap to 4 GB, set:
##
## -Xms4g
## -Xmx4g
##
## See https://www.elastic.co/guide/en/elasticsearch/reference/cu
## for more information
##
################################################################

# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms1g
-Xmx1g
```

ES优化(4)

◘锁定物理内存

设置memory_lock来锁定进程的物理内存地址
避免内存交换（swapped）来提高性能

修改文件
vi config/elasticsearch.yml
bootstrap.memory_lock: true

```
#
# ---------------------------------- Memory -----------
#
# Lock the memory on startup:
#
#bootstrap.memory_lock: true
bootstrap.memory_lock: false
bootstrap.system_call_filter: false
#
```

ES优化(5)

◘合理设置分片

适当增大分片，可以提升建立索引的能力，5-20个比较合适。

如果分片数过少或过多，都会导致检索比较慢。

- 分片数过多，会导致检索时打开文件较多，另外也会导致多台服务器之间通讯，影响效率。
- 分片数过少会导至单个分片索引过大，所以检索速度慢。
- 建议单个分片最多存储20G左右的索引数据，通用计算公式：分片数量=数据总量/20G

地址：
https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-create-index.html

ES优化(6)

合理设置副本数

- 增加副本，可以提升搜索的能力。
- 如果副本设置过多，会对服务器造成额外的压力，因为主分片需要给所有副本同步数据。另外，副本过多也会占用磁盘空间。
- 一般建议最多设置2-3个即可

地址：
https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-create-index.html

ES优化(7)

◘合并索引

定时对索引进行合并优化，segment越多，占用的segment memory越多，查询的性能也越差。

1）索引量不大的情况下，可以将segment设置为1。

2）在es2.1.0以前调用_optimize接口，后期改为_forcemerge接口。

```
curl -u elastic:123456 -XPOST 'http://master:9200/test/_forcemerge?max_num_segments=1'
```

ES优化(8)

- 关闭索引

针对不使用的index，建议close，减少内存占用。
只要索引处于open状态，索引库中的segement就会占用内存，close之后就只会占用磁盘空间不会占用内存。

curl -u elastic:123456 -XPOST 'master:9200/test/_close'

ES优化(9)

◘清除删除文档

在Lucene中删除文档，数据不会马上在硬盘上清除，而是在lucene索引中产生一个.del的文件，然而在检索过程中这部分数据也会参与检索，lucene在检索过程会判断是否删除，如果已经删除，再过滤掉，这样也会降低检索效率。

可以执行清除删除文档命令：

```
curl -u elastic:123456 -XPOST 'http://master:9200/test/_optimize?only_expunge_deletes=true'
```

ES优化(10)

◘ 合理数据导入

- 如果在项目开始阶段，需要批量入库大量数据，建议将副本数设置为0。因为es在索引数据的时候，如果副本已经存在，数据会立即同步到副本中，这样会对es增加压力。

- 等到索引完成后，再恢复副本数即可，可以提高索引效率。

```
curl -XGET http://master:9200/test/_settings?pretty

curl -H "Content-Type:application/json"  -XPUT
'http://master:9200/test/_settings' -d
'{"index":{"number_of_replicas":0}}'

curl -H "Content-Type:application/json"  -XPUT
'http://master:9200/test/_settings' -d
'{"index":{"number_of_replicas":1}}'
```

ES优化(11)

- 设置索引_all

去掉mapping中_all域，Index中默认会有_all的域，虽然会给查询带来方便，但是会增加索引时间和索引尺寸。

地址：
https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-all-field.html

ES优化(12)

◘ 设置索引_source

_source字段我在们进行检索时相当重要。

ES默认检索只会返回ID，如果在{"enabled":false}情况下，你需通过根据这个ID去去倒排索引中去取每个Field数据，效率不高。而反之，在{"enabled":true}的情况下可以根据ID直接检索对应source JSON的字段，不用去倒排索引去按Field取数据。

地址：
https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-source-field.html

ES优化(13)

版本

- 使用Java代码操作es集群,要保证本地es的版本和集群上es的版本保持一致。
- 保证集群中每个节点的JDK版本和es配置一致。

Thanks

大讲台
dajiangtai.com

elk课程QQ学习交流群：732021751　加群备注：elk

客服微信/QQ：84985152

助教QQ：484166349

网址：http://www.dajiangtai.com