分布式全文搜索引擎Search

范 颖 捷 | 2018年7月

TRANSWARP
星 环 科 技

目录》
CONTENTS

Search简介

- ✓ 什么是Search
- ✓ 适用场景

1.1 什么是Search

➢ 概念

- 一种基于ElasticSearch的分布式全文搜索与分析引擎
- 对ElasticSearch进行了多种功能升级和优化，并通过Esdrive实现了SQL的全方位支持

➢ 特点

- 分布式实时文档存储
  - 独创的分层存储数据结构，支持内存/SSD/SATA自动分层存储
  - 堆外内存管理，单节点存储能力从6TB提升至15TB
  - 支持结构化、半结构化和非结构化数据
  - PB级数据规模
- 分布式实时搜索分析
  - 冷热数据分级存储，毫秒级实时关键字检索
  - 计算紧贴数据和索引，亚秒级在线分析

1.1 什么是Search

- 特点
  - SQL引擎与搜索引擎相融合
    - 通过Inceptor和Esdrive，实现以SQL方式创建索引和全文搜索
  - 高扩展
    - 千个节点的横向线性扩展
- 在TDH平台中，Search扮演两种角色
  - 作为Hyperbase全文检索的底层实现
  - 作为一个独立的服务，既是一个分布式文件存储系统，又是一个强大的全文搜索引擎

- 文档数据库
  - 存储半结构化、非结构化数据
  - 功能和性能均优于Mongodb
- 日志分析与监控
  - 统计和日志类时间序列数据的存储和分析
- 舆情分析
  - 高聚合率的统计分析，如：热词跟踪
- 搜索引擎
  - 多条件模糊查询
  - 不指定列的全文搜索

2
chapter

Search原理

- ✓ 数据模型
- ✓ 分词与索引
- ✓ 系统架构

- Index（索引）
  - Search以Index为单位组织数据（Document），一个Index中的数据通常具有相似的特征
    - 例如：为员工信息创建一个Index，或者为商品信息创建一个Index
  - 与HBase中的索引（全局索引）不是一个概念，这里是指Search的数据对象
- Type（分类）
  - Type是Index的逻辑分类，如何分类由用户决定，一个Index可定义一个或多个Type
    - 例如：员工信息Index可按部门分类，包括财务部Type、销售部Type、研发部Type等
- Document（文档）
  - Search的最基础数据单元，以JSON格式存储
    - 例如：员工的基本信息{"name": "zhangsan", "age": 30, "on_board_data": "2016-10-01", ...}可作为一个Document，保存到员工信息Index中
- Field（字段）
  - Document中的数据存储在Field中

➢ Search数据对象与传统二维表的映射关系

- Type是Index的逻辑分类，不映射为传统二维表中的数据对象

Search	传统二维表
Index（索引）	Table（表）
Document（文档）	Row（行）
Field（字段）	Column（列）

➢ 分词

- 基本过程
  - 将文档拆分成一组单独的词（term）
  - 将词转换为标准形式，以提高查全率，如：电脑 → 计算机
- 分词器
  - 英文分词器：standard、English
  - 中文分词器：ik、mmseg
  - 不同的分词器会产生不同的分词结果，产生不同的索引，所以相同的查询条件会产生不同的结果

➢ 倒排索引

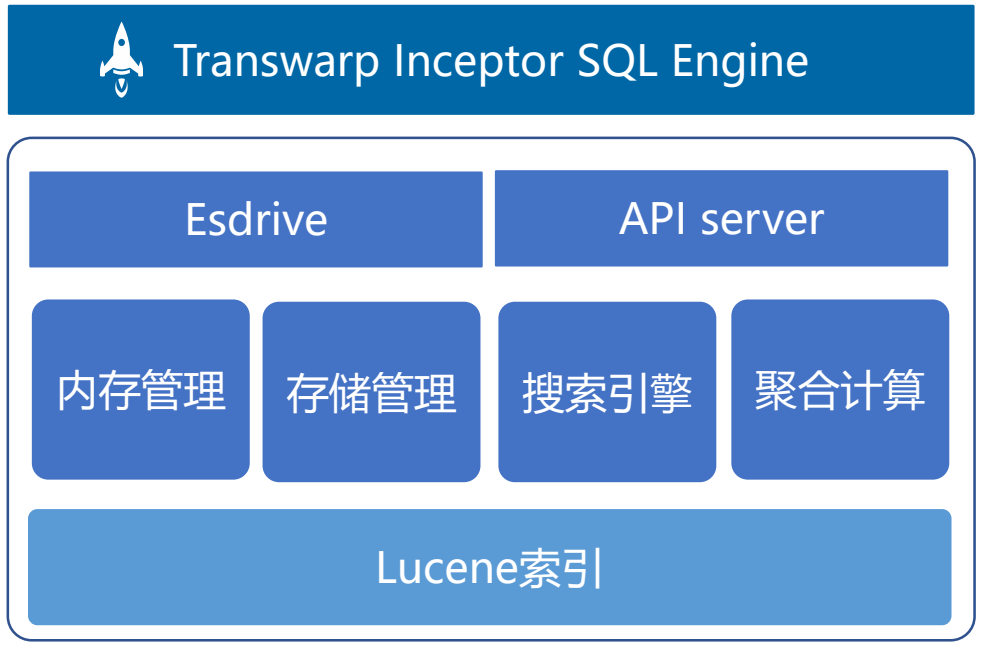
- 假设有两篇文档
  - 文档1的内容：Tom lives in Shanghai, I live in Shanghai too.
  - 文档2的内容：He once lived in Beijing.

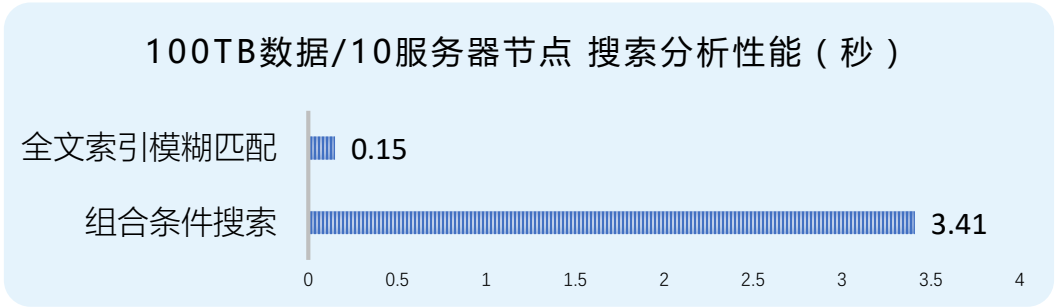
➢ 倒排索引

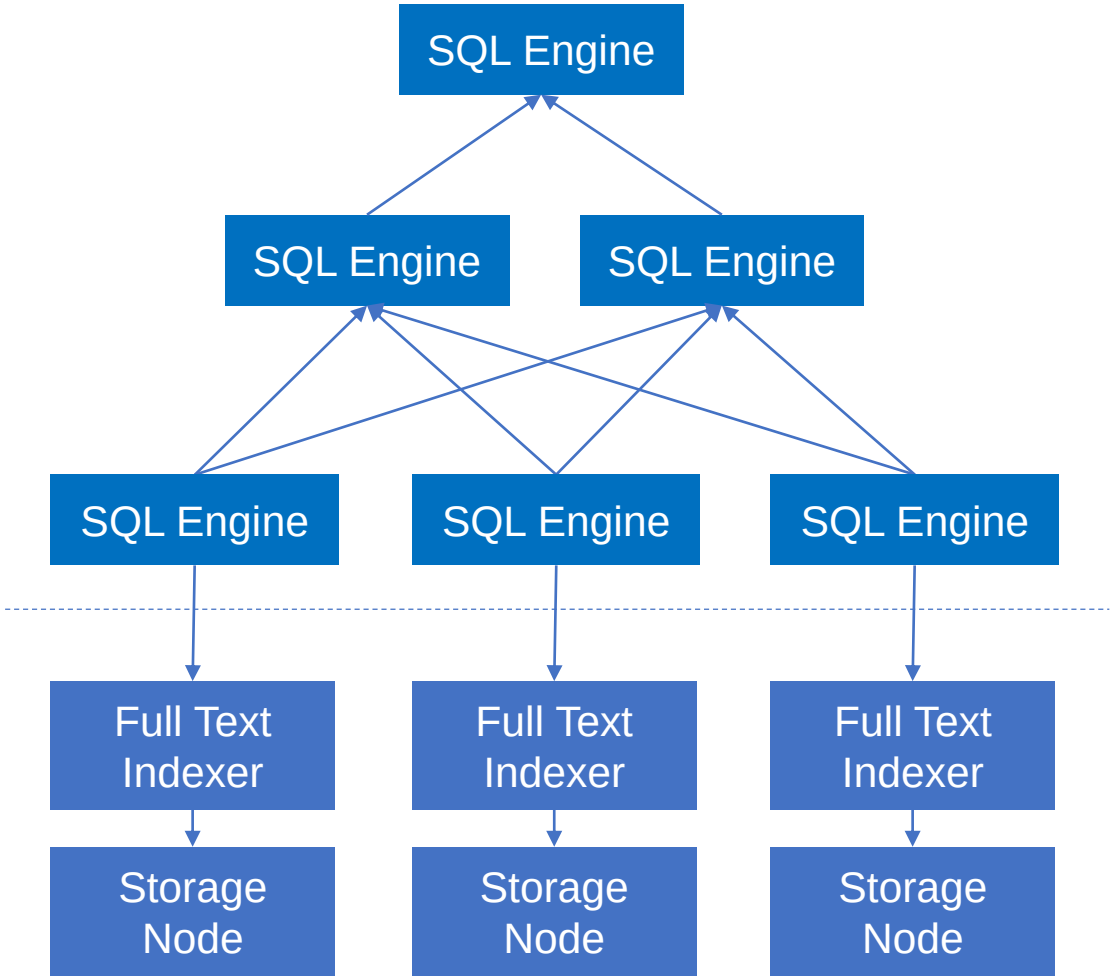
- 对文档进行分词
  - 文档1的分词结果：[Tom] [live] [Shanghai] [i] [live] [Shanghai]
  - 文档2的分词结果：[he] [live] [Beijing]
- 根据分词结果，构建倒排索引

词	文档号[出现频率]	出现位置
shanghai	1[2]	3, 6
he	2[1]	1
I	1[1]	4
live	1[2], 2[1]	2, 5, 2
beijing	2[1]	3
tom	1[1]	1

2.3 系统架构







TRANSWARP 星环科技

3
chapter

Search安装与配置

- ✓ 系统安装
- ✓ 系统配置与管理

1.选择服务　2.分配角色　3.配置服务　4.配置安全　5.服务总览　6.安装

选择想要添加的服务，可能存在某些由于依赖服务未添加而无法添加的服务，请先添加其依赖的服务。

- [ ] **ZOOKEEPER**
  ZooKeeper用于协调同步其他服务
- [ ] **HDFS**
  HDFS是Hadoop应用的基本存储系统
- [ ] **YARN**
  YARN是资源管理框架
- [x] **SEARCH**
  Search是一个分布式的搜索分析引擎
- [ ] **HYPERBASE**
  Hyperbase是实时在线事务处理引擎
- [ ] **TXSQL**
  TxSQL是一个分布式的关系型数据库
- [ ] **NOTIFICATION**
  通知组件是一个提供相关组件中产生的消息的存储，查询，记录组件的工作状态的服务
- [ ] **INCEPTOR**
  Inceptor是基于内存的交互式SQL分析引擎
- [ ] **TRANSPORTER**
  TDT是数据整合工具
- [ ] **TRANSPEDIA**
  Transpedia是TDH产品的文档检索服务
- [ ] **OOZIE**
  Oozie是Hadoop的一个工作流调度系统
- [ ] **SQOOP**
  Sqoop 用于在Hadoop和其他结构化数据存储中传送数据
- [ ] **SOPHON**
  Sophon是一个深度学习交互式分析工具
- [ ] **SLIPSTREAM**
  Slipstream是基于Spark的实时流处理引擎
- [ ] **DISCOVER**
  Discover是基于内存的数据挖掘引擎
- [ ] **KAFKA**
- [ ] **WORKFLOW**
  Workflow是图形化的工作流设计、调试、调度和分析的
- [ ] **RUBIK**

取消　下一步


1.选择服务
2.分配角色
3.配置服务
4.配置安全
5.服务总览
6.安装
本页面为选中的服务分配角色， 默认已按推荐方法分配，如果想要修改分配，先点选左侧菜单的一个角色，再在中间面板上选中/取消从而为该角色分配节点
Search2
Search Server
搜索主机...
3 个节点在 /default-rack
全选
全不选
全选
全不选
上一步
下一步

1.选择服务 2.分配角色 3.配置服务 4.配置安全 5.服务总览 6.安装

配置选中的服务

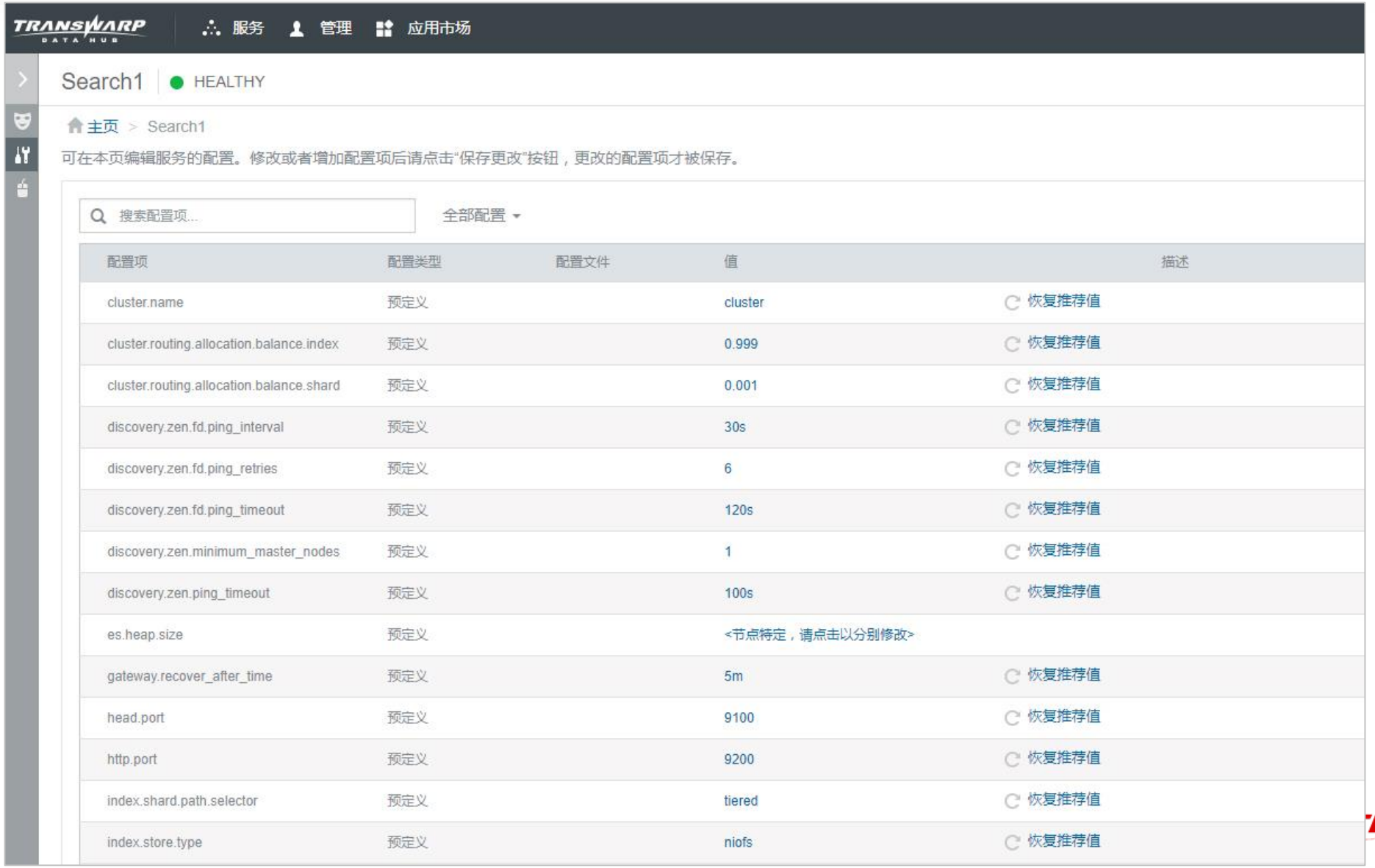
Search2

属性 基础参数 自定义参数

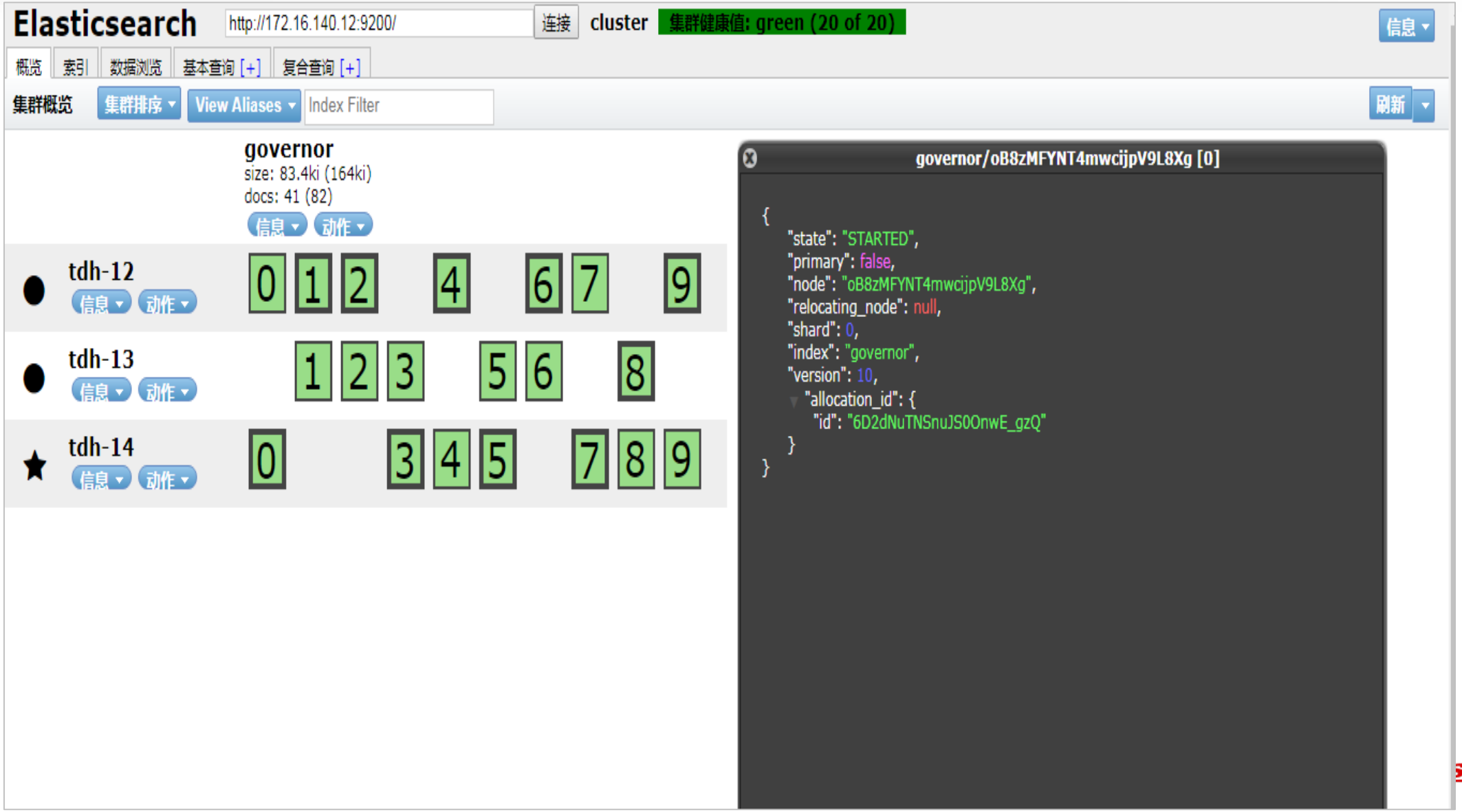
基础参数	值
path.data	<节点特定，请点击以分别修改>
node.master	<节点特定，请点击以分别修改>
node.data	<节点特定，请点击以分别修改>
http.port	9200
transport.tcp.port	9300
network.bind_host	<节点特定，请点击以分别修改>
network.publish_host	<节点特定，请点击以分别修改>
es.heap.size	<节点特定，请点击以分别修改>
search.container.limits.memory	

上一步 下一步

3.2 系统配置与管理



配置项	配置类型	配置文件	值		描述
cluster.name	预定义		cluster	恢复推荐值	
cluster.routing.allocation.balance.index	预定义		0.999	恢复推荐值	
cluster.routing.allocation.balance.shard	预定义		0.001	恢复推荐值	
discovery.zen.fd.ping_interval	预定义		30s	恢复推荐值	
discovery.zen.fd.ping_retries	预定义		6	恢复推荐值	
discovery.zen.fd.ping_timeout	预定义		120s	恢复推荐值	
discovery.zen.minimum_master_nodes	预定义		1	恢复推荐值	
discovery.zen.ping_timeout	预定义		100s	恢复推荐值	
es.heap.size	预定义		<节点特定，请点击以分别修改>		
gateway.recover_after_time	预定义		5m	恢复推荐值	
head.port	预定义		9100	恢复推荐值	
http.port	预定义		9200	恢复推荐值	
index.shard.path.selector	预定义		tiered	恢复推荐值	
index.store.type	预定义		niofs	恢复推荐值	


Elasticsearch
http://172.16.140.12:9200/
连接
cluster
集群健康值: green (20 of 20)
信息
概览
索引
数据浏览
基本查询 [+]
复合查询 [+]
集群概览
集群排序
View Aliases
Index Filter
刷新
governor
size: 83.4ki (164ki)
docs: 41 (82)
信息
动作
tdh-12
信息
动作
0 1 2 4 6 7 9
tdh-13
信息
动作
1 2 3 5 6 8
tdh-14
信息
动作
0 3 4 5 7 8 9
governor/oB8zMFYNT4mwcijpV9L8Xg [0]
{
"state": "STARTED",
"primary": false,
"node": "oB8zMFYNT4mwcijpV9L8Xg",
"relocating_node": null,
"shard": 0,
"index": "governor",
"version": 10,
"allocation_id": {
"id": "6D2dNuTNSnuJS0OnwE_gzQ"
}
}

4
chapter

Search使用

- ✓ REST API
- ✓ Esdrive SQL

访问方法

- 在命令行中通过curl命令访问REST API
- 访问端口默认为9200

示例

- 任务：搭建一个员工信息Index（Index名为employee）
  - Index中的每个Document对应一名员工信息
  - Index的Type按部门分类：dev（研发部）、finance（财务部）和sales（销售部）
- 第1步：创建名为employee的Index

```
/* 创建Index employee， pretty表示以JSON格式返回结果 */
curl -XPUT 'localhost:9200/employee/?pretty'
/* 返回结果：true表示Index创建成功 */
{
  "acknowledged" : true
}
```

➢ 示例

- 第2步：编入Document，即向Index中新增一条员工信息

```
/* 向Index employee的Type dev下编入Document */
curl -XPUT 'localhost:9200/employee/dev/1?pretty' -d '{
   "firstname": "San",
   "lastname": "Zhang",
   "age": 26,
   "on_board_date": "2015-10-31",
   "hometown": "Beijing",
   "school": "Nanjing University",
   "married": false,
   "about": "I love Beijing Opera"
}'
```

```
/* 返回结果：新增Document的元数据（JSON格式）*/
{
   "_index": "employee",  // index名
   "_type": "dev",        // type名
   "_id": "1",            // id
   "_version": 1,         // 版本号
   "_shards": {           // Shard信息
      "total": 2,
      "successful": 2,
      "failed": 0
   },
   "created": true        // 创建成功
}
```

➢ 示例

- 第3步：查看Document是否存在

```
/* 查看/employee/dev/1下是否存在Document，-i表示打印HTTP header */
curl -i -XHEAD 'localhost:9200/employee/dev/1'
```

- 第4步：获取Document

```
/* 获取Index employee的Type dev下的Document */
curl -XGET 'localhost:9200/employee/dev/1?pretty'
/* 返回结果：Document对象（JSON格式）*/
{
   "_index": "employee", "_type": "dev", "_id": "1", "_version": 1, "found": true,
   "_source": {
      "firstname": "San", "lastname": "Zhang", "age": 26, "on_board_date": "2015-10-31",
      "hometown": "Beijing", "school": "Nanjing University", "married": false, "about": "I love Beijing Opera"
   }
}
```

➢ 示例

- 第5步：更新Document

```
/* 将Document中的age改为30 */
curl -XPUT 'localhost:9200/employee/dev/1?pretty' -d '{
   "firstname": "San",
   "lastname": "Zhang",
   "age": 30,
   "on_board_date": "2015-10-31",
   "hometown": "Beijing",
   "school": "Nanjing University",
   "married": false,
   "about": "I love Beijing Opera"
}'
```

```
/* 返回结果：更新后的Document元数据，版本
   号加1 */
{
   "_index": "employee",  // index名
   "_type": "dev",        // type名
   "_id": "1",            // id
   "_version": 2,         // 版本号
   "_shards": {           // Shard信息
      "total": 2,
      "successful": 2,
      "failed": 0
   },
   "created": true        // 创建成功
}
```

➢ 示例

- 第6步：删除Document

```
/* 删除/employee/dev/1下的Document */
curl -XDELETE 'localhost:9200/employee/dev/1?pretty'
```

- 第7步：删除Index

```
/* 删除Index employee */
curl -XDELETE 'localhost:9200/employee/?pretty'
/* 返回结果：true表示Index删除成功 */
{
   "acknowledged": true
}
```

4.2 Esdrive SQL：DDL

- 数据类型

Esdrive SQL	ElasticSearch
String	String
Int	Integer
Boolean	Boolean
Tinyint	Byte
Smallint	Short
Bigint	Long
Float	Float
Double	Double
Date	Date
Timestamp	Long

➢ 创建Esdrive内表

- 在Search中创建Index，同时在Inceptor中创建对应的映射表（Esdrive内表）

```
CREATE TABLE <tableName> (<id> STRING, <col_name1> <data_type1>, <col_name2> <data_type2>, ...) ①
  STORED AS ES
  [WITH SHARD NUMBER <m>] ②
  [REPLICATION <n>] ③
  [TBLPROPERTIES('elasticsearch.tablename'='<dbName.tableName1>')]; ④
```

① 第一列<id>映射为Index的_id，必须是String类型
② 可选项，指定Index的分片（Shard）数，默认值为10，建议每个Shard不超过25G，建表后不可改
③ 可选项，指定Shard的副本数，默认值为1，建表后可改
④ 可选项，<dbName.tableName1>指定新建表在Search中映射的Index名，Index必须是不存在的

➢ 创建Esdrive内表

- 示例：创建了一张名为esdrive_inner_table的Esdrive内表，字段包含所有的数据类型，Shard数为10，副本数为1，均采用默认值

```
create table esdrive_inner_table (key1 string, sv0 int, sv1 boolean, sv2 tinyint, sv3 smallint, sv4 bigint,
  sv5 float, sv6 double, sv7 string, sv8 date, sv9 timestamp) stored as ES;
```

4.2 Esdrive SQL：DDL

➢ 创建Esdrive外表

- 在Inceptor中创建映射表（Esdrive外表），与已存在的Index建立映射关系
- 创建Esdrive外表时，不可指定Shard和副本数

```
CREATE EXTERNAL TABLE <tableName> (
   <id> STRING, <col_name1> <data_type1>, <col_name2> <data_type2>, ...
   )
   STORED BY 'io.transwarp.esdrive.ElasticSearchStorageHandler' ①
   [WITH SERDEPROPERTIES('elasticsearch.columns.mapping'='_id,<cl1>,<cl2>, ...')] ②
   TBLPROPERTIES('elasticsearch.tablename'='<dbName.tableName1>','elasticsearch.indextype'='default') ③
);
```

① 指定Storage Handler，STORED AS ES的简写
② 可选项，指定外表的列和Index字段的映射关系
③ 必选项，建外表时必须指定Search中一个已存在的Index<dbName.tableName1>，支持指定Index Type

➢ 创建Esdrive外表

- 示例

```
CREATE EXTERNAL TABLE esdrive_external_table (
   key1 string, ex0 int, ex1 bigint, ex2 double, ex3 string
   )
   STORED BY 'io.transwarp.esdrive.ElasticSearchStorageHandler'
   WITH SERDEPROPERTIES ('elasticsearch.columns.mapping'='_id,sv0,sv2,sv5,sv7')
   TBLPROPERTIES ('elasticsearch.tablename'='default.esdrive_inner_table');
```

Esdrive外表中的列	Key1(string)	ex0(int)	ex1(bigint)	ex2(double)	ex3(string)
ES索引的字段	_id(string)	sv0(integer)	sv2(byte)	sv5(float)	sv7(string)

➢ 分词器（Analyzer）

- 创建Esdrive内表时，直接用SQL语句对列指定分词器
  - 只用于创建内表
  - 只用于内表中String类型的列，且不可以是内表的首列
- 不同语言类型，选择的分词器不同

```
/* 对中文列指定分词器 */
<col_name> STRING <WITH|APPEND> ANALYZER 'ZH' <ANALYZER_NAME>
① 'ZH'：声明语言类型为中文，必须有单引号
② <WITH|APPEND>：指定分词列的关键字。当该列只需要检索语义时，用WITH关键字；当该列在检索语义的基础上还需要精确查询时，用APPEND关键字
③ <ANALYZER_NAME>：可选的中文分词器有两个，ik和mmseg

/* 对英文列指定分词器 */
<col_name> STRING <WITH|APPEND> ANALYZER 'EN' <ANALYZER_NAME>
① 'EN'：声明语言类型为英文，必须有单引号
② <ANALYZER_NAME>：可选的英文分词器有两个，standard和english
```

4.2 Esdrive SQL：DDL

➢ 为Esdrive表增加列

- 只用于Esdrive内表，不能用于外表

```
ALTER TABLE <tableName> ADD COLUMNS (
   <col_name1> <data_type1>, <col_name2> <data_type2>, ...
);
```

➢ 清空Esdrive表

- 只能清空Esdrive内表，不能用于外表

```
TRUNCATE TABLE <tableName>;
```

➢ 删除Esdrive表

- 对于内表，删除ES中的Index（含数据），以及Inceptor中的Esdrive表（Metastore中的元数据）
- 对于外表，只删除Inceptor中的Esdrive表（Metastore中的元数据），不删除ES中的Index

```
DROP TABLE <tableName>;
```

➢ 插入数据

- 单条插入

```
INSERT INTO esdrive_inner_table (key1,sv0,sv1,sv2,sv3,sv4,sv5,sv6,sv7,sv8,sv9)
  VALUES ('k1',2,true,3,4,5,6.0,7.33,'s7ad','2010-06-30 08:54:42',1477881007946);
```

- 批量插入

```
BATCHINSERT INTO esdrive_inner_table (key1,sv0,sv1,sv2,sv3,sv4,sv5,sv6,sv7,sv8,sv9)
  BATCHVALUES ( VALUES ('k2',3,true,3,4,5,6.1,7.33,'dsfx','2010-06-30 10:07:48',1377832007946),
    VALUES ('k3',4,true,3,4,5,6.2,7.22,'ngfh','2010-06-30 06:08:17',1277833007946),
    VALUES ('k4',5,true,3,4,5,6.3,7.11,'ewwd','2010-06-30 18:52:37',1477844007946)
  );
```

- 插入时首列需唯一
  - 首列key1映射为Index的_id，首列相同的数据只会保留最后插入的数据

```
INSERT INTO TABLE esdrive_start(key1, content, tint, tbool) VALUES ('1', 'hello ES', 1 , true);
INSERT INTO TABLE esdrive_start(key1, content, tint, tbool) VALUES ('2', 'hello search', 2, false);
```

4.2 Esdrive SQL：DML

- 更新数据

```
UPDATE <tableName> SET <col_name> = <value> WHERE <filter_conditions>;
eg: update esdrive_inner_table set sv1=false where key1='k1';
```

- 删除数据

```
DELETE FROM <tableName> WHERE <filter_conditions>;
eg: DELETE FROM esdrive_inner_table WHERE key1='k1';
```

- 对于Select语句，Esdrive SQL与Inceptor SQL的用法完全相同
  - 包括Where、Group By、Join、集合运算等
- Esdrive SQL检索语义的实现
  - 第1步：对被查询文本进行分词，生成倒排索引
  - 第2步：对查询条件进行分词，利用倒排索引实现全文检索
- Esdrive SQL检索语义的优势
  - 语义更丰富
    - 标准SQL：like %word%模糊查询只有一种语义，无法描述复杂语义，如：查询指定间隔内的两个词
    - Esdrive SQL：定义了多种模糊查询的语法，用于描述不同场景下的检索语义
  - 查询性能更优
    - 标准SQL：like %word%模糊查询的算法复杂度为O(n)
    - Esdrive SQL：基于倒排索引的全文检索的算法复杂度为O(log(n))

➢ Contains函数

- 通过Contains函数对查询条件进行分词，实现分词检索

```
CONTAINS(
   <col_name>, '<text_query>'
)
```

- Near操作符
  - 限制所查单词的间隔，提高查询结果的相关性，包含三个参数
    ① token1~n：表示检索词（n≥2），token应该是倒排索引中存在的单词，否则查询无意义
    ② slop：表示token1与token2之间允许间隔的最大token数，该值是一个上限
    ③ in_order：可选项，表示是否按顺序依次匹配token，Boolean类型。默认值为false，不按顺序匹配，token1可以出现在token2之后；若为true，则按顺序匹配，token1必须出现在token2之前

```
CONTAINS(
   <col_name>, 'NEAR((token1, token2[, token3, ...]), slop[, in_order])'
);
```

➢ Contains函数

- Near操作符

```
Select * from news_analyze_zh where contains(content, 'near((京东,阿里), 1, false)');
```

key1	content
7	阿里、京东、小米争相抢滩，“新零售”到底是什么

- Fuzzy操作符
  - 在Contains分词检索的基础上，查询语义相似的短语，包含2个参数

    ① phrase：表示需要查询的短语。先经过分词得到多个token，查询结果必须包含分词后的所有token

    ② fuzziness：表示最大编辑距离（Levenshtein距离），用于表达短语之间的语义相似度，中文较复杂

```
CONTAINS(
  <col_name>, 'FUZZY(phrase, fuzziness)'
);
```

➢ Contains函数

- Fuzzy操作符

```
Select * from news_analyze_zh where contains(content, 'fuzzy(阿里金融, 5)');
```

key1	content
2	重庆市携手阿里金融，共建新型智慧城市
4	马云剖析阿里未来三个业务：金融、平台和数据

- Contains函数语法举例

```
select dt from inceptor where contains(hphm, "term '鲁D528E8'") order by dt  limit 100;
select dt from inceptor where contains(hphm, "prefix '鲁D528'") order by dt  limit 100;
select dt from inceptor where contains(hphm, "wildcard '鲁D*28E8'") order by dt  limit 100;
select dt from inceptor where contains(dt, "term '92233706363649793807'") and
   contains(hphm, "term '鲁P56729'")  order by dt limit 100;
select dt from inceptor where contains(dt, "range'[92233706477354748 07,92233706477354788 07]'");
```

Q&A

TRANSWARP
星 环 科 技