Elastice search 的使用

海量数据，是个挑战，必须考虑充分。

Elasticsearch stack is much focused on logs aggregation and recently also metrics

（ I am using tiny documents ）

<https://github.com/elastic/elasticsearch/issues/18069>

# 性能和需求

|  |
| --- |
| 总索引数据量(G):  每天新增索引数据量(G):  每天数据更新量(G):  总的文档数量(个):  从生产数据到搜索结果数据实时性要求(秒)：  查询复杂度：  读qps  写qps |

依据参考。

|  |
| --- |
| 单物理机，单个shard:  写的吞吐量，保守按10M/秒 左右估计。  单个Shard 数据量大概控制在1亿个document 之内。  读可以靠增加replication 来实现，后期扩容也方便 |

# 硬件

3台服务器或者以上。(可以按公司mysql机器申请)

|  |
| --- |
| Cpu , 32vCore  Mem, 256G  Disk, SSD ,多块 |

2台服务器将降低ha 级别(1台挂了，集群就停止工作了)。

# 接入数据

## MQ

业务数据写入jdq, 可以分为多个queue,提高吞吐量。

部署ES集群 吞吐量比业务系统高3倍以上，以避免mq 积压。

## 从mysql定期取数据

写个worker ,定期取数据，插入es

## 直接写

对于数据要求不是非常严格的，如果写失败，重试后，暂时不写，事后再补。

## Binlog & Kafka接入

目前大数据部负责kafka的运维。Kafka 的数据，写入es, 需要自己写代码。

## 保证数据完整

现在大部分业务表有流水号，可以定期检查es exist (流水号)。乃至于文档日期。

# 部署

## 版本选择

Tbd

## Linux 准备

内核特性，/etc/sysctl.conf

|  |
| --- |
| sysctl -w net.core.rmem\_max=16777216  sysctl -w net.core.wmem\_max=16777216  sysctl-w net.ipv4.tcp\_rmem="4096 65536 16777216"  sysctl -w net.ipv4.tcp\_wmem="4096 65536 16777216" |

修改系统限制

|  |
| --- |
| echo "\* - nproc 32768" | sudo tee -a /etc/security/limits.d/90-nproc.conf  echo "vm.max\_map\_count = 400000" | sudo tee -a /etc/sysctl.conf  echo 1 | sudo tee /proc/sys/vm/overcommit\_memory  echo "\* - nofile 100000" | sudo tee -a /etc/security/limits.conf  echo "\* - memlock unlimited" | sudo tee -a /etc/security/limits.conf  echo "\* - as unlimited" | sudo tee -a /etc/security/limits.conf  echo "\* - nproc32768" | sudo tee -a /etc/security/limits.conf  sudo sysctl–p  重新登陆 |

关闭swap

|  |
| --- |
| sudoswapoff –a |

Jdk用 1.7 u71+ ，或者1.8 u60+

## 配置

安装配置

Elasticsearch.in.sh

|  |
| --- |
| ES\_HEAP\_SIZE=10g |

Elasticsearch.yml

|  |
| --- |
| cluster.name:  node.name：起个名字  Path.data:，分割  Path.logs  Network.host:  discovery.zen.minimum\_master\_nodes: 2 //可用的master node 的一半+1  discovery.zen.ping.multicast.enabled: false discovery.zen.ping.unicast.hosts:[]  gateway.recover\_after\_nodes: 2  gateway.expected\_nodes: 2 |

elasticSearch.yml, 影响性能的参数

|  |
| --- |
| #indices.memory.index\_buffer\_size: 40%  #indices.store.throttle.type: merge  #indices.store.throttle.max\_bytes\_per\_sec: 50mb  #index.refresh\_interval: 2s  #index.fielddata.cache: soft  #index.store.type: mmapfs  #index.fielddata.cache.size: 20% |

config/logging.yml

|  |
| --- |
|  |

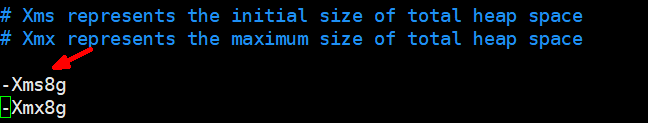
启动：

Bin/elsticsearch -d

<http://192.168.178.56:9200/_plugin/head/>

修改elastic的内存：

修改 [es目录]/config/jvm.options 的如下内容,数字就是使用内存数



# 建模

切记：Es 的doc 更新，是删除+insert，这个是重点关注的。不更新最好。

第二点：es 的索引是bitset ，不同于mysql 的b+ tree。

第三点：一般情况冗余的代价比更新的低，如果用冗余能实现的，考虑冗余

<http://exploringelasticsearch.com/modeling_data.html#ch-modeling-data>

<https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-put-mapping.html>

|  |
| --- |
|  |

创建索引：(相当于库)

curl -XPUT 'http://192.168.178.56:9200/warehouse' ，这个用了缺省的5个shard 。最好自定义一下。

|  |
| --- |
| curl -XPUT 'http://o56:9200/orderbatching/' -d '{  "settings" : {  "number\_of\_shards": 1,  "number\_of\_replicas" : 1  }  }' |

建立mapping (type, 相当于表)

|  |
| --- |
| curl -XPUT 'http://192.168.178.56:9200/twitter/\_mapping/tweet' -d '{"properties":{"code":{"type":"long"},"postDate":{"type":"date","format":"dateOptionalTime"},"user":{"type":"string","index":"not\_analyzed"}}}' |

分析型的应用，基本不需要全文，一定要把字段搞为“not\_analyzed” .

5.0 改为

"type" "keyword", "index": true

<https://www.elastic.co/blog/strings-are-dead-long-live-strings>

可以做几个优化

|  |
| --- |
| 避免存储\_all字段  curl -XPUT 'http://192.168.178.56:9200/twitter/\_mapping/tweet' -d '{"tweet":{"\_all":{"enabled":false}}}’ |

主键处理

|  |
| --- |
| <http://www.elastic.co/guide/en/elasticsearch/reference/1.4/mapping-id-field.html>  但是这种方式有代价，建议不用这种方式。用自己的业务主键。Put /{id} –d {} |

|  |
| --- |
| 需要sort/aggregation 的字段，推荐使用doc\_values类型。   "fielddata": {     "format": "doc\_values"   } |

2.0 以后缺省了

时间类型

|  |
| --- |
| "type":"date","format":"dateOptionalTime"}  <https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-date-format.html>  epoch\_millis  <https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-date-format.html>  public static final TimeUnit *TIME\_UNIT* = TimeUnit.*MILLISECONDS*; public static final boolean *ROUND\_CEIL* = true;  精度是毫秒。  Round\_ceil = true 向顶端靠近。  // Rounding up a date here has the following meaning: If a date is not // defined with full precision, for example, no milliseconds given, the date // will be filled up to the next larger date with that precision. // Example: An upper bound given as "2000-01-01", will be converted to // "2000-01-01T23.59.59.999"  如果需要，可以用 date.round\_ceil 指定为flase . |

## 数据类型

缺省的text 类型。

*FIELD\_TYPE*.setIndexed(true);  
*FIELD\_TYPE*.setTokenized(true);  
*FIELD\_TYPE*.setStored(false);  
*FIELD\_TYPE*.setStoreTermVectors(false);  
*FIELD\_TYPE*.setOmitNorms(false);  
*FIELD\_TYPE*.setIndexOptions(IndexOptions.*DOCS\_AND\_FREQS\_AND\_POSITIONS*);  
*FIELD\_TYPE*.freeze();

数值型

*FIELD\_TYPE*.setTokenized(false);  
 *FIELD\_TYPE*.setOmitNorms(true);  
 *FIELD\_TYPE*.setIndexOptions(IndexOptions.*DOCS\_ONLY*);  
 *FIELD\_TYPE*.setStoreTermVectors(false);

数值型，占用更少的磁盘，建议使用。

可以设定 precision\_step ， 以优化range 查询的效率。

|  |
| --- |
| The default for all data types is 4, which is used, when no precisionStep is given.  Ideal value in most cases for 64 bit data types (long, double) is 6 or 8.  Ideal value in most cases for 32 bit data types (int, float) is 4.  For low cardinality fields larger precision steps are good. If the cardinality is < 100, it is fair to use Integer.MAX\_VALUE (see below).  http://stackoverflow.com/questions/21556095/use-of-precision-step-in-elasticsearch-mappings  <https://www.elastic.co/guide/en/elasticsearch/reference/current/number.html> |

## Index 配置

缺省值，可以在elasticsearch.yml中配置

|  |
| --- |
| 'index.mapping.ignore\_malformed': False,  'index.mapping.coerce': False,  'index.query.parse.allow\_unmapped\_fields': False, |

### 创建索引，指定Shard 数量：

|  |
| --- |
| curl -XPUT 'http://localhost:9200/website/' -d '{  "index" : {  "number\_of\_shards" : 1,  "number\_of\_replicas" : 1  }  }' |

Shard 不能临时改变，最好做一点预留.

5个shard \*1个复制，可以把请求压力分发负载到10台机器上。对于请求压力不大的，就用缺省的5个shard 就好了。写压力大的，可以增加。数据少的，可以减少shard 数量。

每个shard 的文档数量，文档数量小于5000万。

1-2个复制，一般就够了，多了影响写性能。这个可以在线改。

## Index 模板

<https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-templates.html>

|  |
| --- |
| curl -XPUT http://o56:9200/\_template/template\_1 -d '{  "template": "te\*",  "settings": {  "number\_of\_shards": 1  },  "mappings": {  "\_default\_": {  "\_all": {  "enabled": false  }  }  }  }' |

使用模板,创建索引，

|  |
| --- |
| curl -XPUT <http://o56:9200/te_test1> ， te 开头的会用到  curl -XPUT 'http://192.168.178.56:9200/te\_test1/\_mapping/tweet' -d '{"tweet":{"properties":{"code":{"type":"long"},"postDate":{"type":"date","format":"dateOptionalTime"},"user":{"type":"string","index":"not\_analyzed"}}}}}' |

## 建模

<http://www.elastic.co/guide/en/elasticsearch/guide/master/modeling-your-data.html>

<https://www.elastic.co/blog/managing-relations-inside-elasticsearch>

Design document relations in this order: denormalization, nested docs, parent/child, manage relations yourself.

一般情况下采用冗余。或者在代码中join 查询结果。

## 采用冗余方式

<https://www.elastic.co/guide/en/elasticsearch/guide/current/denormalization.html>

## Inner object

适合1对1 的。

否则会把多个值，搞为多值字段，不能满足查询要求。譬如

|  |
| --- |
| {  "name" : "Zach",  "car.make" : ["Saturn", "Subaru"]  "car.model" : ["SL", "Imprezza"]  } |

## 采用nested ，读优化的

能较好的保持对象和子对象的关系。不更新的可以用。

<https://www.elastic.co/blog/managing-relations-inside-elasticsearch>

nested ,只能用nested query ,不利于更新。

<http://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-nested-type.html>

## Parent-child , 更新优化的

you'll run into situations where sorting or scoring are, frankly, very difficult.

占用内存。

## 在程序中join

<https://www.elastic.co/guide/en/elasticsearch/guide/current/application-joins.html>

数次查询，把结果合并

# 架构

## Indice or type

<https://www.elastic.co/blog/index-vs-type>

# 写代码

Api ,<http://www.elasticsearch.org/guide/en/elasticsearch/client/java-api/current/client.html>

简单demo , <http://source.jd.com/app/kili/tree/master/kilies>

## 建索引 （库） ，建type(表) ,删除索引

建表

<https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-put-mapping.html>

删除索引：

curl -XDELETE http://localhost:9200/index

type 相当于表。

|  |
| --- |
| curl -XPUT http://o56:9200/orderbatching/\_mapping/coherentorder/ -d '  {  "properties": {  "volume": {  "type": "integer"  },  "promise\_time": {  "type": "long"  },  "location": {  "type": "integer"  },  "owner": {  "type": "integer"  }  }  }  ' |

删除表：

不能。

## Insert doc ，删除

以json 作为输入输出格式的。

$ curl -XDELETE 'http://localhost:9200/twitter/tweet/1'

<https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/java-docs-index.html>

|  |
| --- |
| IndexResponse response = client.prepareIndex("twitter", "tweet", "1")  .setSource(jsonBuilder()  .startObject()  .field("user", "kimchy")  .field("postDate", new Date())  .field("message", "trying out Elasticsearch")  .endObject()  )  .get(); |

|  |
| --- |
| esClient.prepareDelete(*index*,*iType*,order.id.toString) |

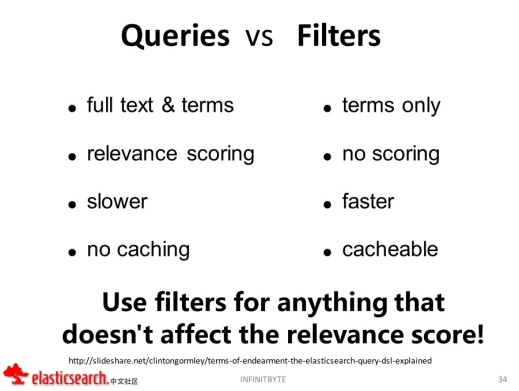
## 查询

为何会有这么多种查询，就是为了查询的结果符合预期。主要是打分方式的不同。

<https://www.compose.com/articles/elasticsearch-query-time-strategies-and-techniques-for-relevance-part-ii/>

这个文档有调优的过程。

除非需要评分，否则总是使用filter



<https://blogs.endjin.com/2015/01/elasticsearch-gotchas-and-tips-part-2/>

一些提高性能技巧：

|  |
| --- |
| * + Prefer filters where possible to queries as filters can be cached, which improves performance. Filters in a filtered query apply to the whole data set and narrow the set that the query needs to run on. Bear in mind that scoring doesn’t apply to filters, only to queries.   + “Bool” filter should be used for BitSet-based filters (i.e. those based on the inverted index). And/Or/Not filter should be used for dynamically calculated filters like geo\_\* and scripted. This is important as it will dramatically alter performance.   + As well as using And/Or/Not for geo filter, prefer to place it as the last filter, as it is expensive so you want it to operate on the most filtered data set possible.   + Avoid faceting on fields that are analysed or have many possible values.   + Use multi fields to index a single field in multiple ways (e.g. a single field “myField” could be tokenized as normal, and used for full text searching, but have a sub property “myField.raw” or “myField.orig” that isn’t analysed for exact matches).   + Use post filters to show different results to those used for aggregations. Data can be thought of as multiple streams. For example you could run a query, perform some aggregations on that data set to get overall numbers, then further filter that data set to whatever  full documents you want to see (perhaps for paging, but with stats across all results).   + Value\_count (distinct count) metric aggregation is accurate but expensive, cardinality aggregation is not totally accurate but fast. Do you need exact counts, or is a reasonable approximation enough?   + You can return a subset of fields in a document. This is useful if you have large documents but only actually need to send a small amount of data back over-the-wire for a particular request. |

推荐使用boolFilter ,性能改进很多

<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-bool-filter.html>

|  |
| --- |
| BoolFilterBuilder fb = FilterBuilders.*boolFilter*(); fb.must(FilterBuilders.*termFilter*("user","u3262"),FilterBuilders.*termFilter*("code",144));  QueryBuilderqb = QueryBuilders.*constantScoreQuery*(fb); SearchResponsesr = client.prepareSearch("twitter").setTypes("tweet")  .setQuery(qb).setFrom(0).setSize(6).  execute().actionGet(); |

分页：

翻页不要大于 20 。如果需要可以加入Query滤掉。

rangeQuery 性能较差

## 复杂查询的例子

在es 源代码测试目录，index/query 下边有很多json .

## Fuzzy 查询， 有编辑距离的查询，最大距离2

|  |
| --- |
| Directory directory = *newDirectory*(); RandomIndexWriter writer = new RandomIndexWriter(*random*(), directory); addDoc("aaaaa", writer); addDoc("aaaab", writer); addDoc("aaabb", writer); addDoc("aabbb", writer); addDoc("abbbb", writer); addDoc("bbbbb", writer); addDoc("ddddd", writer);  IndexReader reader = writer.getReader(); IndexSearcher searcher = *newSearcher*(reader); writer.close();  //距离最大是2， 前缀是0 FuzzyQuery query = new FuzzyQuery(new Term("field", "aaaaa"), FuzzyQuery.*defaultMaxEdits*, 0);  ScoreDoc[] hits = searcher.search(query, null, 1000).scoreDocs; *assertEquals*(3, hits.length);  // same with prefix query = new FuzzyQuery(new Term("field", "aaaaa"), FuzzyQuery.*defaultMaxEdits*, 1);  hits = searcher.search(query, null, 1000).scoreDocs; *assertEquals*(3, hits.length); query = new FuzzyQuery(new Term("field", "aaaaa"), FuzzyQuery.*defaultMaxEdits*, 2);  hits = searcher.search(query, null, 1000).scoreDocs; *assertEquals*(3, hits.length); query = new FuzzyQuery(new Term("field", "aaaaa"), FuzzyQuery.*defaultMaxEdits*, 3);  hits = searcher.search(query, null, 1000).scoreDocs; *assertEquals*(3, hits.length); query = new FuzzyQuery(new Term("field", "aaaaa"), FuzzyQuery.*defaultMaxEdits*, 4);  hits = searcher.search(query, null, 1000).scoreDocs; *assertEquals*(2, hits.length); |

## preFixQuery , 严格的前缀

参见TestPrefixQuery.java

|  |
| --- |
| String[] categories = new String[] {"/Computers",  "/Computers/Mac",  "/Computers/Windows"}; RandomIndexWriter writer = new RandomIndexWriter(*random*(), directory); for (int i = 0; i < categories.length; i++) {  Document doc = new Document();  doc.add(*newStringField*("category", categories[i], Field.Store.*YES*));  writer.addDocument(doc); } IndexReader reader = writer.getReader();  PrefixQuery query = new PrefixQuery(new Term("category", "/Computers")); IndexSearcher searcher = *newSearcher*(reader); ScoreDoc[] hits = searcher.search(query, null, 1000).scoreDocs; *assertEquals*("All documents in /Computers category and below", 3, hits.length);  query = new PrefixQuery(new Term("category", "/Computers/Mac")); hits = searcher.search(query, null, 1000).scoreDocs; |

## MultiPhrasePrefixQuery ， 分词后，保持顺序的，前缀

参见：

MultiPhrasePrefixQueryTests.java

|  |
| --- |
| IndexWriter writer = new IndexWriter(new RAMDirectory(), new IndexWriterConfig(Lucene.*VERSION*, Lucene.*STANDARD\_ANALYZER*)); Document doc = new Document(); doc.add(new Field("field", "aaa bbb ccc ddd", TextField.*TYPE\_NOT\_STORED*)); writer.addDocument(doc); IndexReader reader = DirectoryReader.*open*(writer, true); IndexSearcher searcher = new IndexSearcher(reader);  MultiPhrasePrefixQuery query = new MultiPhrasePrefixQuery(); query.add(new Term("field", "aa")); *assertThat*(Lucene.*count*(searcher, query), *equalTo*(1l));  query = new MultiPhrasePrefixQuery(); query.add(new Term("field", "aaa")); query.add(new Term("field", "bb")); *assertThat*(Lucene.*count*(searcher, query), *equalTo*(1l));  MultiPhrasePrefixQuery query = new MultiPhrasePrefixQuery(); query.add(new Term("field", "aa")); *assertThat*(Lucene.*count*(searcher, query), *equalTo*(1l));  query = new MultiPhrasePrefixQuery(); query.add(new Term("field", "aaa")); query.add(new Term("field", "bb")); *assertThat*(Lucene.*count*(searcher, query), *equalTo*(1l));  query = new MultiPhrasePrefixQuery(); query.setSlop(1); //隔一个单词 ，也可以 query.add(new Term("field", "aaa")); query.add(new Term("field", "cc")); |

## PhaseQuery ， 短语 ， 有顺序的

/\*\* A Query that matches documents containing a particular sequence of terms.  
 \* A PhraseQuery is built by QueryParser for input like <code>"new york"</code>.  
 \*   
 \* <p>This query may be combined with other terms or queries with a {@link BooleanQuery}.  
 \*/  
public class PhraseQuery

## spanQuery ，词距查询

### spanNearQuery

|  |
| --- |
| 例如我们来搜索包含有“共青团中央下发实施意见”关键字的文章。  1.首先我们把”共青团中央下发实施意见”进行分词为：”共青团中央”,”下发”,”实施意见”。 2.设置slop为0.inOrder为true 代码如下：  SpanNearQueryBuilder span=QueryBuilders.spanNearQuery(); span.clause(QueryBuilders.spanTermQuery("content","共青团中央") ); span.clause(QueryBuilders.spanTermQuery("content","实施意见") ); span.inOrder(true).slop(0); client.prepareSearch("test").setQuery(span).execute().actionGet(); |

## moreLikethisQuery ， 多个field 的联合查询，或者与某个文档的相似的

相对 multiMatchQuery 有更精确的相似度控制能力

<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-mlt-query.html>

|  |
| --- |
| {  "more\_like\_this" : {  "fields" : ["title", "description"],  "like" : "Once upon a time",  "min\_term\_freq" : 1,  "max\_query\_terms" : 12  }  } |

## Multi-value ，多值字段

天生支持

<https://www.elastic.co/guide/en/elasticsearch/reference/current/array.html>

<https://www.elastic.co/guide/en/elasticsearch/guide/current/_multivalue_fields_2.html>

PUT /my\_index/groups/1

{

"names": [ "John Abraham", "Lincoln Smith"]

}

{

"filtered": {

"query": {

"match": { "title": "hello world" }

},

"filter": {

"terms": {

"tag": [ "c", "d" ],

"minimum\_should\_match": 1

}

}

}

}

<http://stackoverflow.com/questions/28001632/filter-items-which-array-contains-any-of-given-values>

{

"filtered": {

"query": {

"match": { "title": "hello world" }

},

"filter": {

"terms": {

"tags": ["c", "d"]

}

}

}

}

## 需要分词的 ， match query

match queries accept text/numerics/dates, analyzes them, and constructs a query.

<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-match-query.html>

## 不分词的， term query

https://www.elastic.co/guide/en/elasticsearch/reference/current/term-level-queries.html

## dis\_max , 多个field ，挑一个最好的分数 ， 或

Dismax handler比standard handler多如下功能：

1.以不同的权值来搜索多个field。   
2.限制查询语法为一个小的集合并且用无语法错误。该特性是强制的并是不可配置的   
3.整个搜索查询的自动的短语boosting   
4.便利的查询boosting参数，通常同函数查询一块使用   
5.能指定单词匹配的最少个数，这取决于查询串中的单词数

disjunction 或 , 逻辑或

[conjunction 与](https://www.google.com.hk/search?safe=strict&espv=2&biw=1280&bih=633&q=disjunction+%E6%88%96+conjunction+%E4%B8%8E&spell=1&sa=X&ved=0ahUKEwjz3pnluL7QAhUEKJQKHdhUAnMQvwUIFigA)

This is useful when searching for a word in multiple fields with different boost factors

A query that generates the union of documents produced by its subqueries, and that scores each document with the maximum score for that document as produced by any subquery, plus a tie breaking increment for any additional matching subqueries.

相对于 bool query, 打分方式不同，取最好的打分。

选择那种query ,取决于 打分的希望。

## Bool query， 多个field ，分数合起来

The bool query takes a more-matches-is-better approach, so the score from each matching must or should clause will be added together to provide the final \_score for each document.

## 返回结果 ， 转化为对象 （一切都是json,都是string）

searchHits 从 sourcejson 转化为对象，或者从 fields .都可以。

final Map<String, Object> fields = hit.getSource();

|  |
| --- |
| public interface SearchHit extends Streamable, ToXContent, Iterable<SearchHitField> {   /\*\*  \* The score.  \*/  float score();   /\*\*  \* The score.  \*/  float getScore();   /\*\*  \* The index of the hit.  \*/  String index();   /\*\*  \* The index of the hit.  \*/  String getIndex();   /\*\*  \* The id of the document.  \*/  String id();   /\*\*  \* The id of the document.  \*/  String getId();   /\*\*  \* The type of the document.  \*/  String type();   /\*\*  \* The type of the document.  \*/  String getType();   /\*\*  \* The version of the hit.  \*/  long version();   /\*\*  \* The version of the hit.  \*/  long getVersion();   /\*\*  \* Returns bytes reference, also un compress the source if needed.  \*/  BytesReference sourceRef();   /\*\*  \* Returns bytes reference, also un compress the source if needed.  \*/  BytesReference getSourceRef();   /\*\*  \* The source of the document (can be <tt>null</tt>). Note, its a copy of the source  \* into a byte array, consider using {@link #sourceRef()} so there won't be a need to copy.  \*/  byte[] source();   /\*\*  \* Is the source empty (not available) or not.  \*/  boolean isSourceEmpty();   /\*\*  \* The source of the document as a map (can be <tt>null</tt>).  \*/  Map<String, Object> getSource();   /\*\*  \* The source of the document as string (can be <tt>null</tt>).  \*/  String sourceAsString();   /\*\*  \* The source of the document as string (can be <tt>null</tt>).  \*/  String getSourceAsString();   /\*\*  \* The source of the document as a map (can be <tt>null</tt>).  \*/  Map<String, Object> sourceAsMap() throws ElasticsearchParseException;   /\*\*  \* If enabled, the explanation of the search hit.  \*/  Explanation explanation();   /\*\*  \* If enabled, the explanation of the search hit.  \*/  Explanation getExplanation();   /\*\*  \* The hit field matching the given field name.  \*/  public SearchHitField field(String fieldName);   /\*\*  \* A map of hit fields (from field name to hit fields) if additional fields  \* were required to be loaded.  \*/  Map<String, SearchHitField> fields();   /\*\*  \* A map of hit fields (from field name to hit fields) if additional fields  \* were required to be loaded.  \*/  Map<String, SearchHitField> getFields();   /\*\*  \* A map of highlighted fields.  \*/  Map<String, HighlightField> highlightFields();   /\*\*  \* A map of highlighted fields.  \*/  Map<String, HighlightField> getHighlightFields();   /\*\*  \* An array of the sort values used.  \*/  Object[] sortValues();   /\*\*  \* An array of the sort values used.  \*/  Object[] getSortValues();   /\*\*  \* The set of query and filter names the query matched with. Mainly makes sense for compound filters and queries.  \*/  String[] matchedQueries();   /\*\*  \* The set of query and filter names the query matched with. Mainly makes sense for compound filters and queries.  \*/  String[] getMatchedQueries();   /\*\*  \* The shard of the search hit.  \*/  SearchShardTarget shard();   /\*\*  \* The shard of the search hit.  \*/  SearchShardTarget getShard(); } |

|  |
| --- |
| public interface SearchHitField extends Streamable, Iterable<Object> {   /\*\*  \* The name of the field.  \*/  String name();   /\*\*  \* The name of the field.  \*/  String getName();   /\*\*  \* The first value of the hit.  \*/  <V> V value();   /\*\*  \* The first value of the hit.  \*/  <V> V getValue();   /\*\*  \* The field values.  \*/  List<Object> values();   /\*\*  \* The field values.  \*/  List<Object> getValues();   /\*\*  \* @return The field is a metadata field  \*/  boolean isMetadataField(); } |

## 测试性能

并发：50个。

读写比：1/10 。

工具：

ab

查看slow log

# 统计

# 工具

## 文档

官方

<http://www.elasticsearch.org/guide/en/elasticsearch/guide/current/index.html>

中文

<http://looly.gitbooks.io/elasticsearch-the-definitive-guide-cn/>

## 中文分词

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

# 技巧

## 按时间片，滚动索引

<https://www.elastic.co/guide/en/elasticsearch/guide/master/index-templates.html>

创建模板

|  |
| --- |
| PUT /\_template/order  {  "template": "order-\*",  "order": 1,  "settings": {  "number\_of\_shards": 1  },  "mappings": {  "\_default\_": {  "\_all": {  "enabled": false  }  }  },  "aliases": {  "last\_3\_months": {}  }  } |

创建索引

curl -XPUT <http://192.168.178.56:9200/order>-201501

添加mapping

|  |
| --- |
| curl -XPUT 'http://192.168.178.56:9200/order-201501/\_mapping/order' -d '{"order":{"properties":{"code":{"type":"long"},"postDate":{"type":"date","format":"dateOptionalTime"},"user":{"type":"string","index":"not\_analyzed"}}}}}' |

添加1行数据

|  |
| --- |
| curl -XPUT http://192.168.178.56:9200/order-201501/order/1 -d '{  "user" : "kimchy",  "post\_date" : "2009-11-15T14:12:12",  "code" : 22}' |

查询alias

|  |
| --- |
| curl -XGET <http://192.168.178.56:9200/>last\_3\_months/\_search  curl -XGET <http://192.168.178.56:9200/>last\_3\_months/order/1 |

再添加1周。

curl -XPUT <http://192.168.178.56:9200/order>-201502

curl -XPUT 'http://192.168.178.56:9200/order-201502/\_mapping/order' -d '{"order":{"properties":{"code":{"type":"long"},"postDate":{"type":"date","format":"dateOptionalTime"},"user":{"type":"string","index":"not\_analyzed"}}}}}'

curl -XPUT http://192.168.178.56:9200/order-201502/order/21 -d '{

"user" : "kimchy",

"post\_date" : "2009-11-15T14:12:12",

"code" : 22}'

再查询

curl -XGET <http://192.168.178.56:9200/>last\_3\_months/\_search

从alias 拿掉第1周。

|  |
| --- |
| curl -XPOST 'http://192.168.178.56:9200/\_aliases' -d '  {  "actions" : [  { "remove" : { "index" : "order-201501", "alias" : "last\_3\_months" } }  ]  }' |

再查询。

curl -XGET <http://192.168.178.56:9200/>last\_3\_months/\_search

如果按时间建立index, 可以每晚运行optimize, 把昨天的index optimize 为一个。

## 应对大量update

可以按功能切分，避免update

仅仅在必要情况考虑采用 parent/child 建模。

## Query 优化

尽量减少复杂计算的doc 参与的数量，就是用 andFilter ,先滤掉不用的doc.

## 全文索引的精确度，召回率调优

去掉 stop word , (或者叫common word)

## n-gram indexing technique

# 运维

## Optimize

把不再更新的index, merge 为一个segment.

## 监控

1. Head 插件

<http://blog.csdn.net/july_2/article/details/24481935>

建立elasticsearch-1.0.0\plugins\head\\_site文件

安装：

http://localhost:9200/\_plugin/head/

http://192.168.178.56:9200/\_plugin/head/

1. 审计es收到的query
2. bigDesk插件
3. 统一日志，接入 slow query log .

## 报警

1. 接入ump

public static void businessAlarm(String key, long time, String detail) 报警 。

在query 请求加上timeout ，捕获异常，进行报警 。

1. url存活监控。

|  |
| --- |
| <http://host:9200/_cluster/health>  <http://host:9200/index/_stats/>  <http://192.168.178.56:9200/_nodes/stats/thread_pool> |

## Refactor mapping

The Great Mapping Refactoring

<https://www.elastic.co/blog/great-mapping-refactoring>

## 升级

<http://www.elastic.co/guide/en/elasticsearch/guide/current/_rolling_restarts.html>

## 重建索引

<http://www.elastic.co/guide/en/elasticsearch/guide/current/reindex.html>

可以从es, scan & roll 出来。可以不需要再从数据库搞。

## 备份恢复

定期做snapshot。目前支持Hadoophdfs .

(TODO：联系公司的大数据部门，申请一点hdfs空间)

## 删除旧数据

3种方案：

1. 设立ttl ， 5.0 没有了。
2. 用命令删除
3. 按照时间片建立索引，使用alias

## 其他故障

调整log 级别。查看slow log.

重建最近的索引。

# 手册

<http://www.elastic.co/guide/en/elasticsearch/guide/master/index.html>