技术分享

ElasticSearch索引与架构原理分享

王星锦-中间件技术组
2022年1月17日

1 基本概念

简介、用途、相关信息

2 Demo演示

演示ES应用、引出疑问

3 逻辑概念&索引原理

索引、文档等概念，深入索引内部实现

4 物理结构

ES集群架构、分片、选主等逻辑

1

基本概念

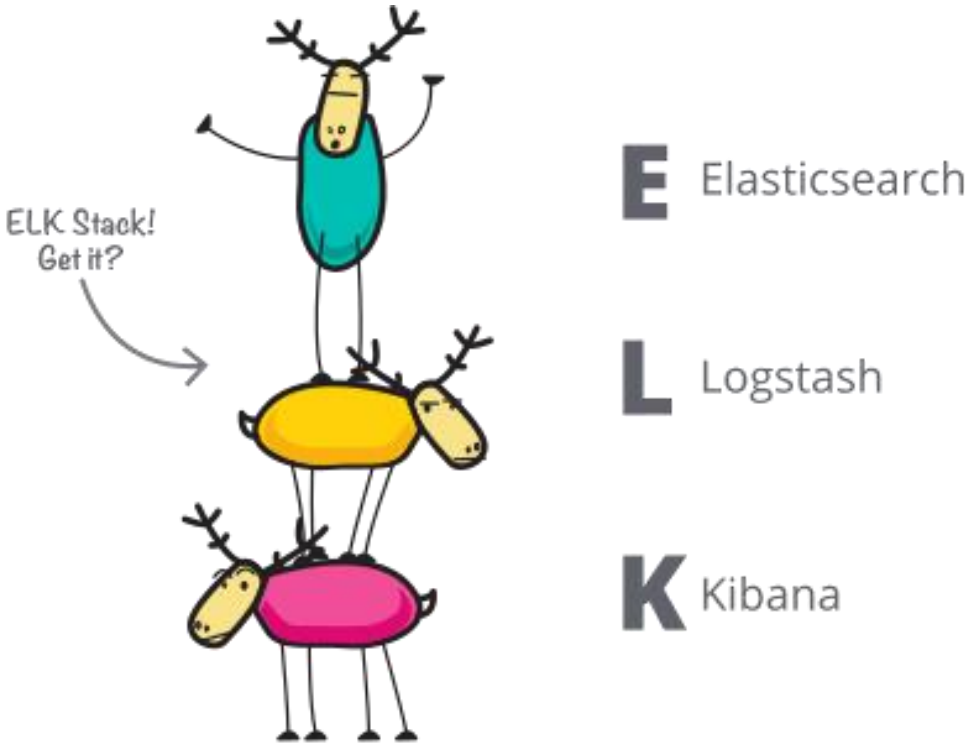
ElasticSearch简介、场景用途、相关信息

1. 基本概念-简介

- ElasticSearch 是一个开源的分布式 RESTful **搜索**和**分析**引擎。
- ElasticSearch基于**Lucene**构建，并对其进行了扩展，使存储、索引、搜索变得更加**灵活、易用**。
- Lucene是一个开源的**搜索引擎工具包**，由Apache软件基金会支持和提供。
- ElasticSearch本质上是基于Lucene增加了分布式高可用和管理功能。

1. 基本概念-用途

- **直接作为后端**：在简单的博客系统、或者独立的搜索推荐模块，前端可**直接对接**ES。
- **日志分析**：ES 拥有一套完整的日志解决方案（ELK），可以快速实现从采集到展示，这也是ES应用最广泛的领域。
- **搜索服务**：使用ES做文档库的全文检索、电商平台的商品搜索等。
- **复杂查询**：关系型DB对复杂查询效率一般不高，可以通过ES实现更高效的查询。



1. 基本概念

以上场景的核心诉求（ES需要能做到的点）

- 海量数据存储与搜索
- 近实时性的查询返回
- 支持多种查询匹配模式
- 易于使用和扩展
- 高可用性与稳定性
- ... ...

2

Demo演示

演示ES简单应用、引入疑问

2. Demo演示

使用ES Suggester API实现搜索自动补全功能

- http://10.246.177.8:18081/jqueryseo/

2. Demo演示-提出疑问

为什么ES基于磁盘存储，并且是海量数据存储，
却仍能实现良好的查询性能？

- 存储时如何压缩空间
- 如何利用内存
- 如何支持复杂的查询

3

逻辑概念&索引原理

索引、文档、段等概念介绍，深入索引内部实现

3. 逻辑概念-基本结构

ES（Lucene）的基础层次结构由**索引、段、文档、域、词**，这五个部分组成。

- 索引（Index）：逻辑上类似于关系型数据库的DB，存储某一类数据和它们的索引
- 段（Segment）：一个Index包含多个段，段之间互相独立，也可看做是子索引
- 文档（Doc）：Lucene会把文档写入段中，每个段中包含多个文档
- 域（Field）：一篇文档会包含多种不同的字段，不同的字段保存在不同的域中
- 词（Term）：Lucene会通过分词器将域中的字符串通过词法分析和语言处理后拆分成词（检索的核心要素）

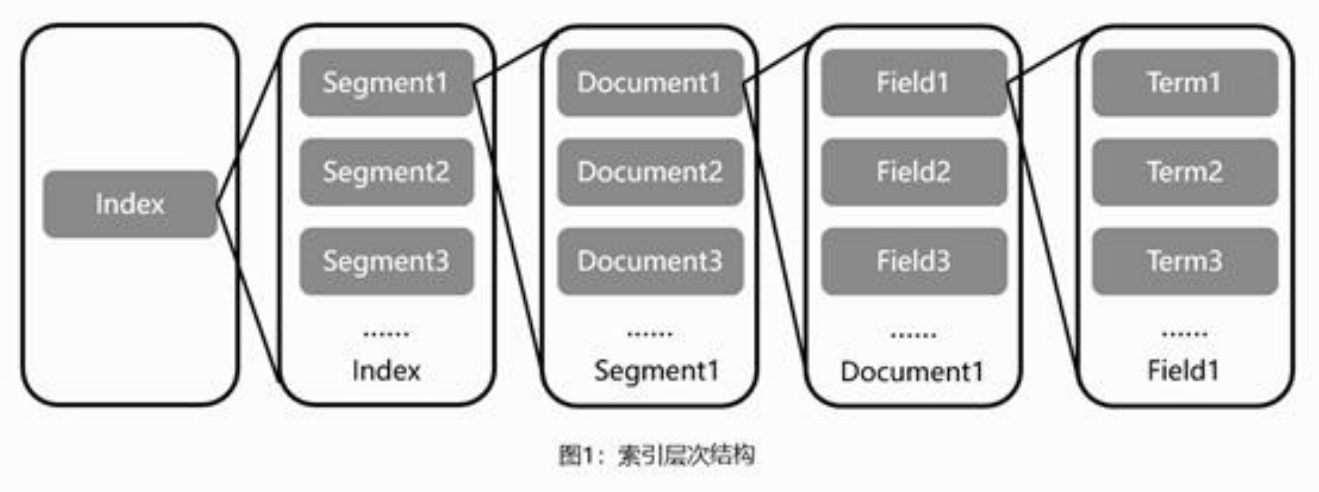


图1：索引层次结构

3. 逻辑概念-索引映射

映射（mappings）是定义一个文档以及其所包含的字段如何被存储和索引的方法。字段有多种数据类型：text、keyword、long、boolean、date等。（5.0之后string类型被拆分为text和keyword）每种类型底层索引构建的方式不同。

- text：text 类型的字段数据会被**分词**，在生成倒排索引以前，字符串会被分词器分成一个一个词项。
- keyword：keyword 类型的字段内容不会被分词，只能被精确搜索。

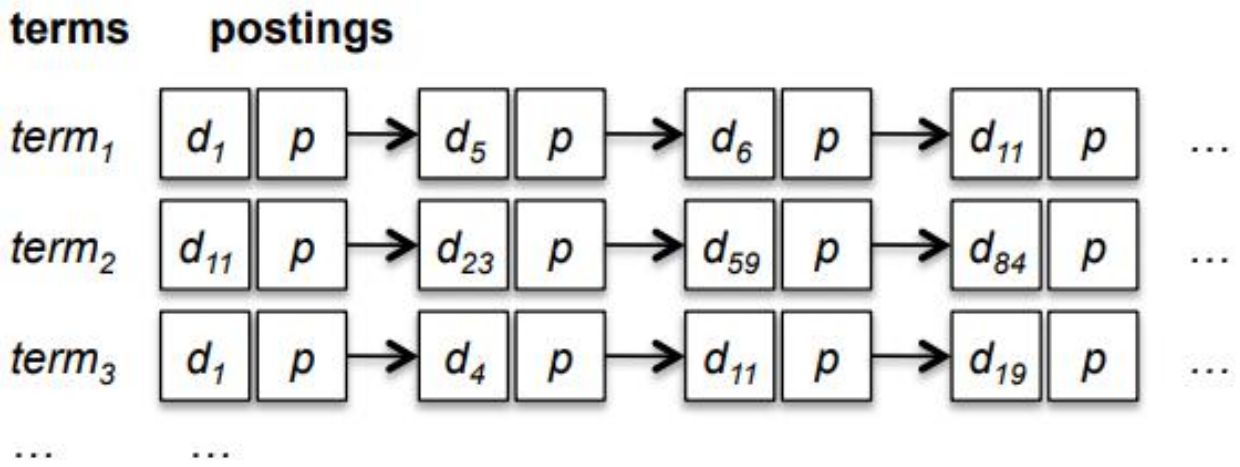
```
{
  "wxjtest" : {
    "mappings" : {
      "properties" : {
        "content" : {
          "type" : "text",
          "fields" : {
            "keyword" : {
              "type" : "keyword",
              "ignore_above" : 256
            }
          }
        },
        "title" : {
          "type" : "text",
          "fields" : {
            "cn-suggest" : {
              "type" : "completion",
              "analyzer" : "simple",
              "preserve_separators" : true,
              "preserve_position_increments" : true,
              "max_input_length" : 50
            },
            "py-suggest" : {
              "type" : "completion",
              "analyzer" : "pinyin_analyzer",
              "preserve_separators" : true,
              "preserve_position_increments" : true,
              "max_input_length" : 50
            }
          },
          "analyzer" : "ik_max_word",
          "search_analyzer" : "ik_smart"
        }
      }
    }
  }
}
```

3. 索引原理-初识倒排

在全文检索领域，各种文档都会提到**倒排索引**，这个概念也是全文检索的根基。

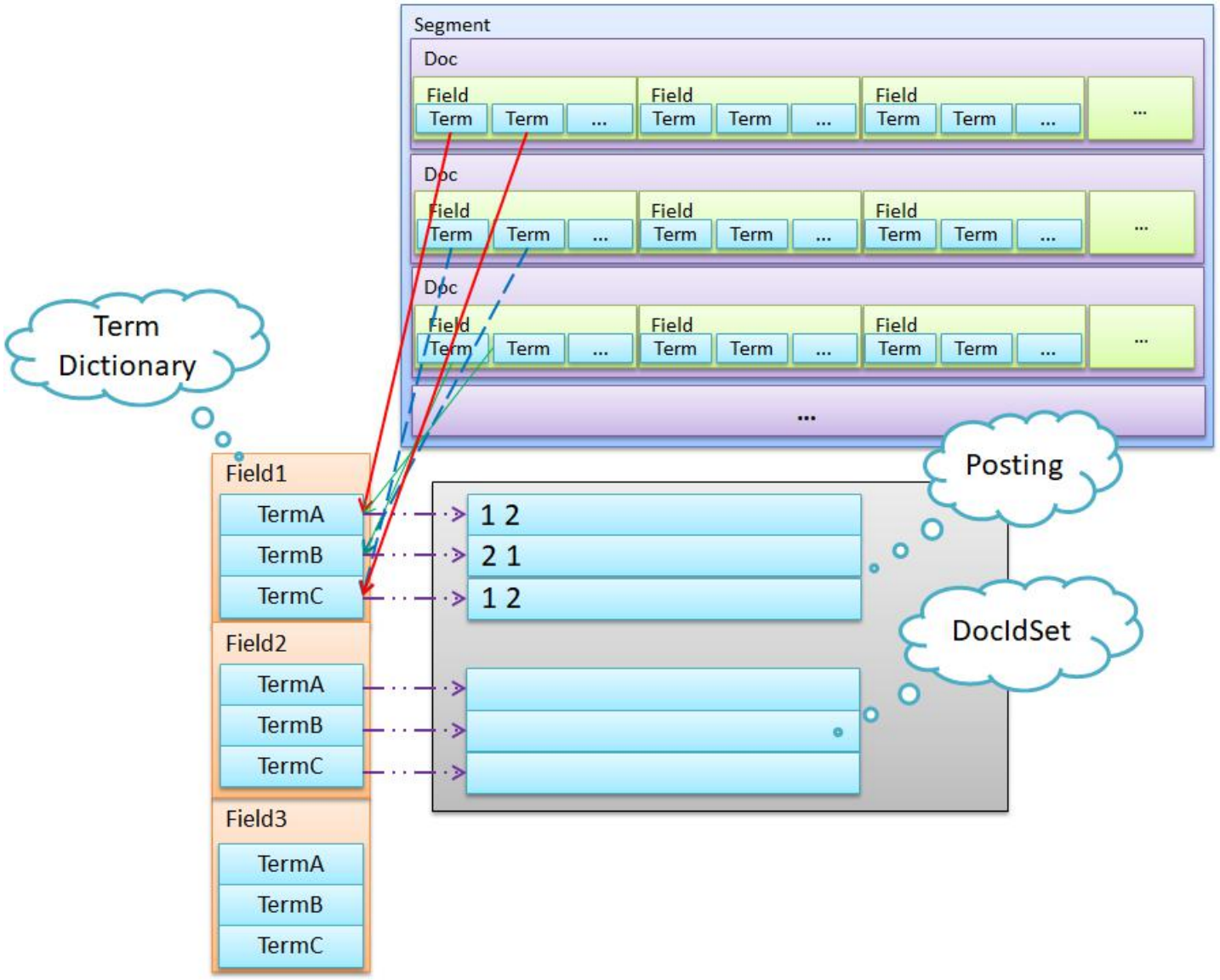
向ES中存储一个文档，也常被称作**索引**一篇文档，因为要根据新增的文档构建倒排索引。

倒排索引可以分为两部分：词典（Term dictionary）和倒排表（Posting list），下图为一个倒排索引的简化示意图。



3. 索引原理-倒排全貌

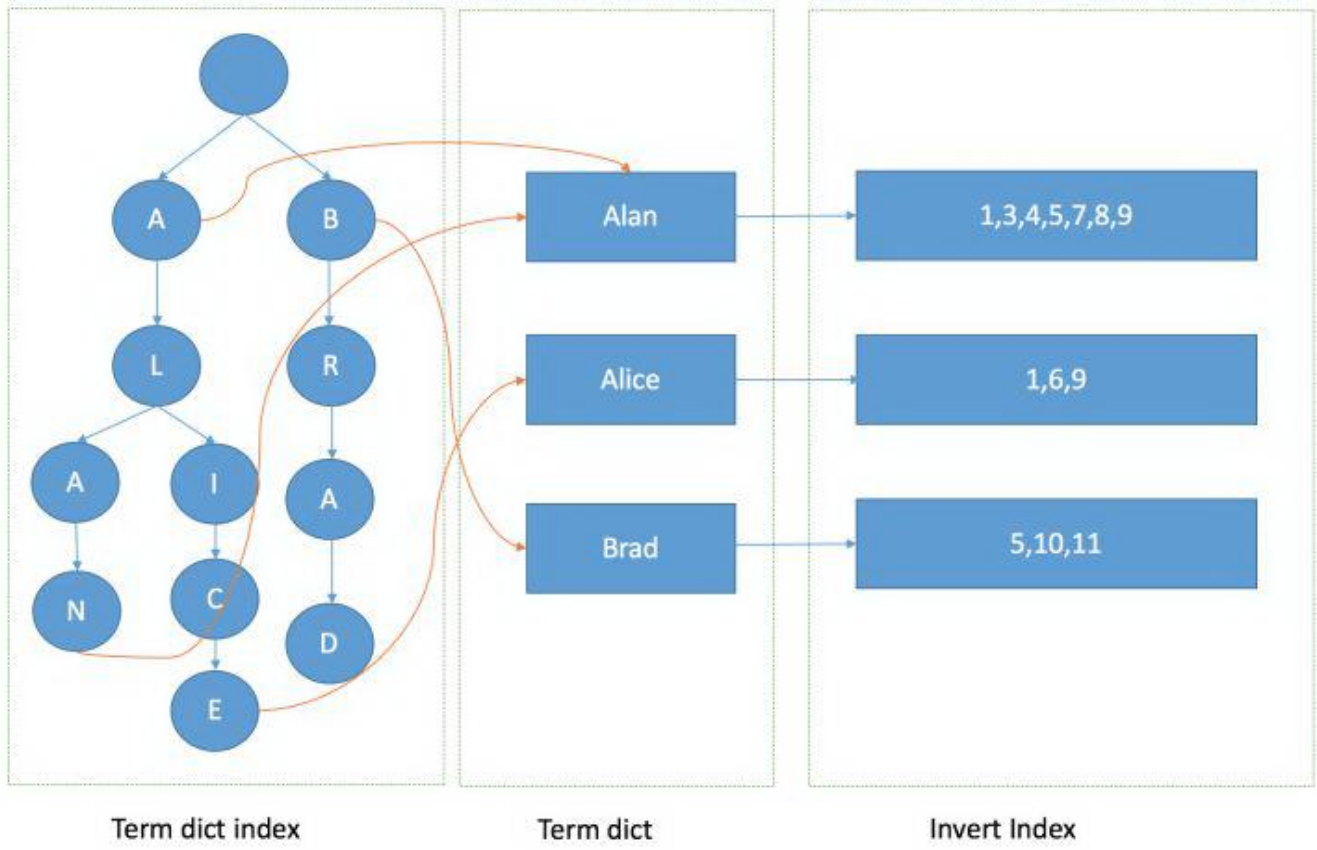
这里是一个更具体的Lucene倒排索引全貌，但实际上各个部分使用了更巧妙的数据结构，将在后面展开。



3. 索引原理-词典

词典（Term dictionary）在Lucene中分为两个部分：词索引（term index）和后缀词块（term dict）

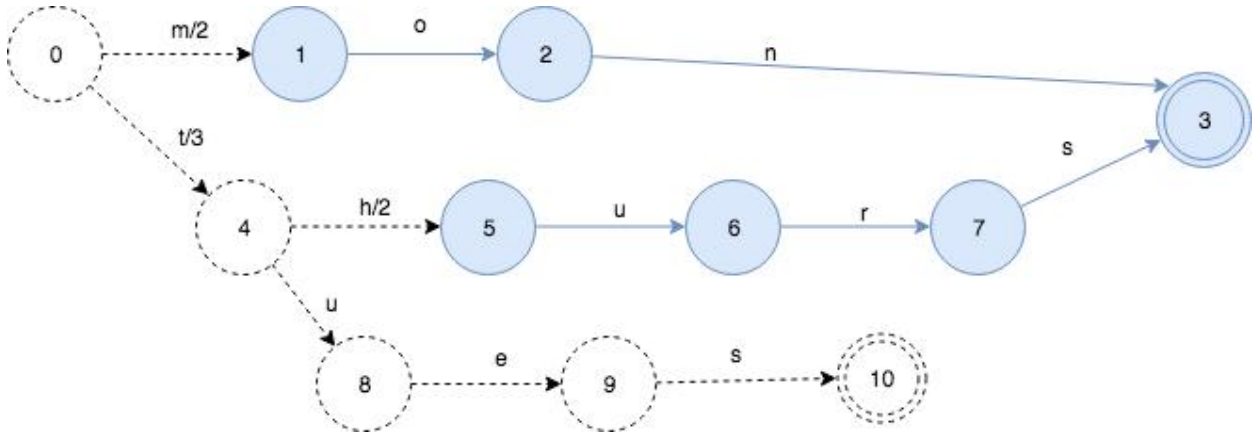
- 词索引：存储term（前缀）的索引文件，以FST的形式加载进内存。
- 后缀词块：是词典和倒排的关系纽带，存储了term和它对应的倒排文件指针。



3. 索引原理-FST

FST（Finite State Transducer，有限状态转换机）

- 构建时要求输入有序
- 内存占用率低，即有较高的压缩率
- 词长度为n，查询复杂度为O(n)
- 能够高效支持前缀、模糊、正则匹配等查询模式。



mon：2

tues：3

thurs：5

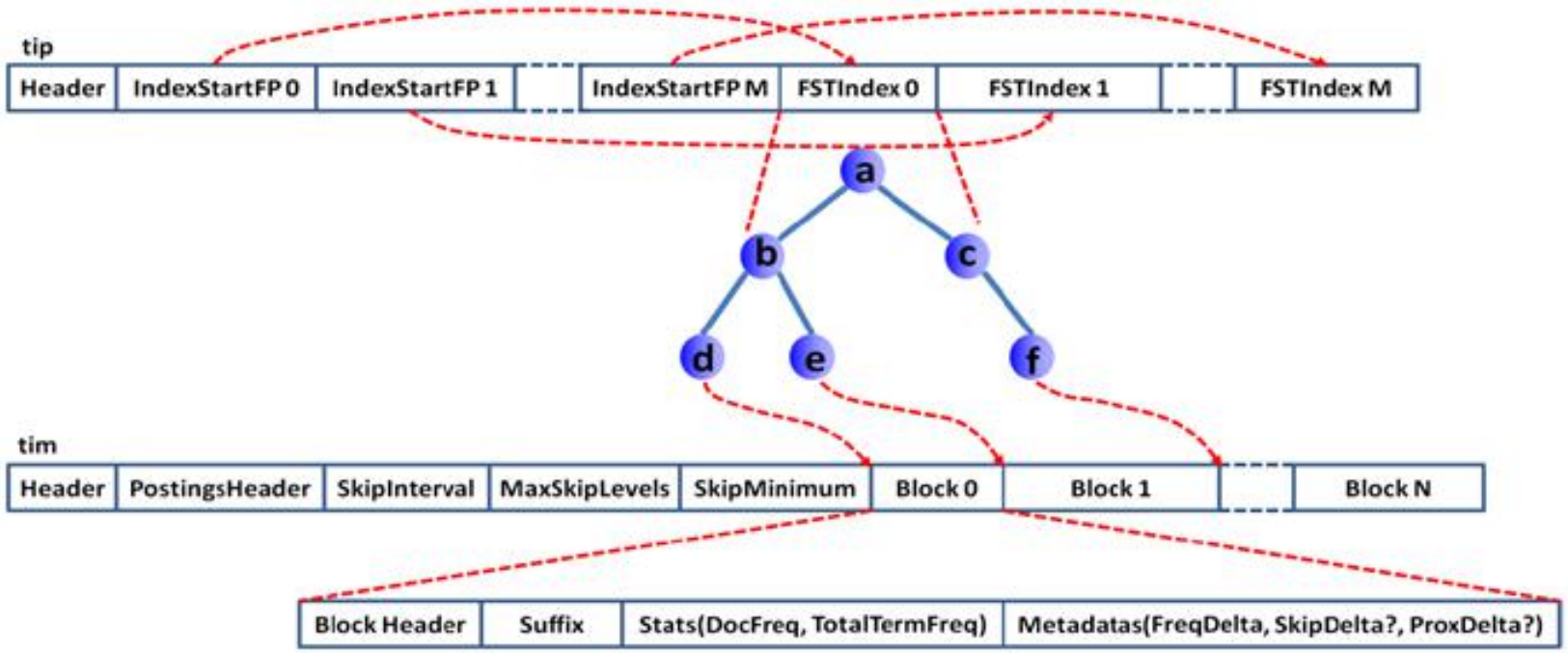
3. 索引原理-词典结构

右图是词典的物理结构示意图：

.tip文件由多个FST构成

Segment上每个字段都有自己的一个FST（FSTIndex）记录在.tip上

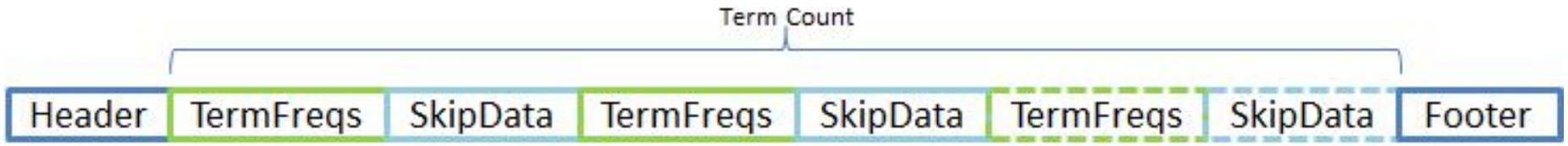
FST个数就是Segment拥有字段的个数



3. 索引原理-倒排表

倒排表（Posting list）保存了**文档ID集合**、term的词频、term在文档中的位置等信息。

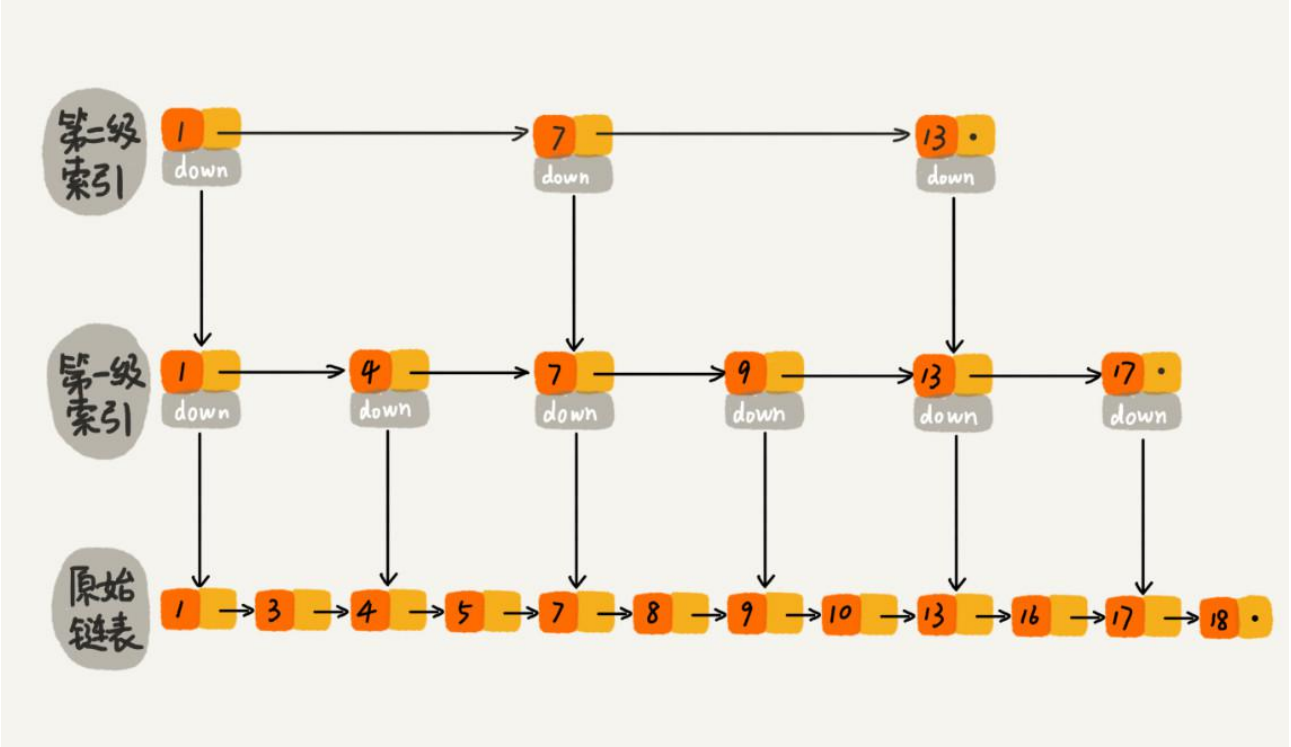
这里我们暂时只关注文档ID集合（DocIds）部分，其中的内容如下图



每个Term都有成对的TermFreqs和SkipData，SkipData是为TermFreqs构建的**跳表**结构

3. 索引原理-跳表

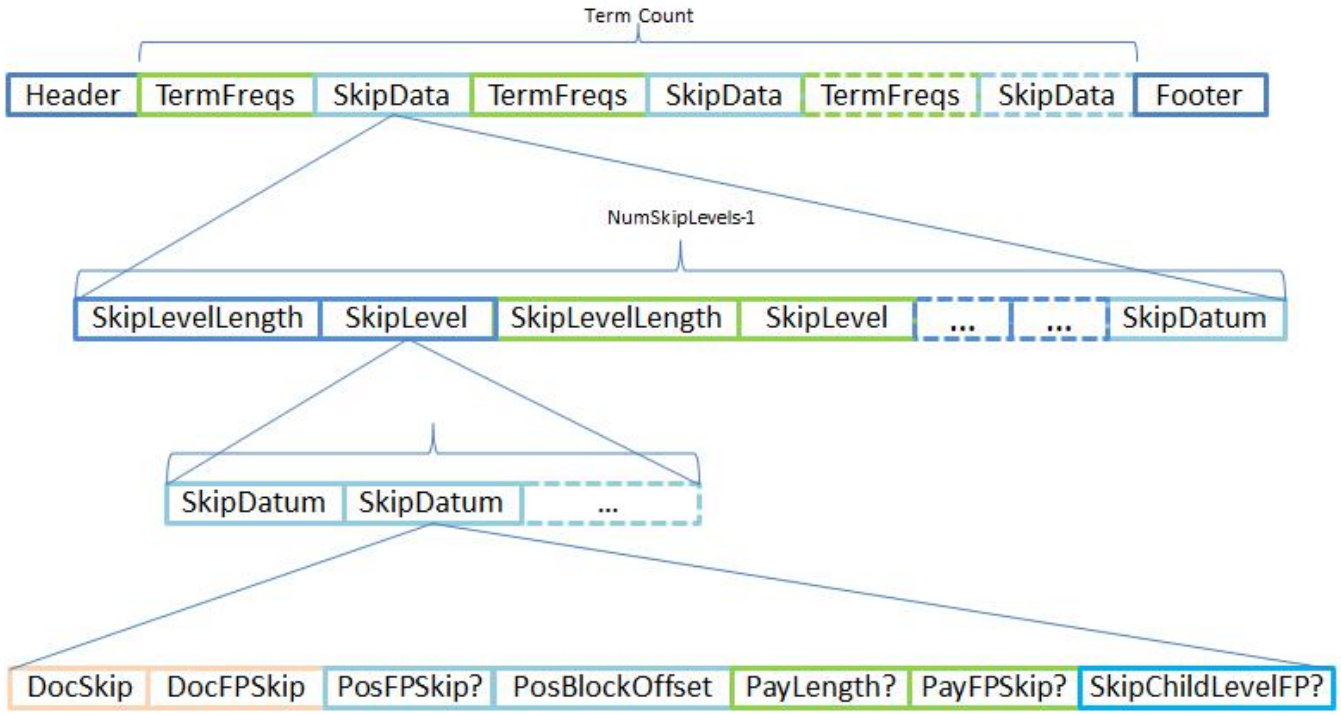
将搜索时耗时转嫁给索引时，这就是Lucene索引的“空间换时间”的基本思想



- 跳表是可以实现二分查找的有序链表
- 最底层包含所有的元素，越往上越稀疏
- 每个元素插入时随机生成它的索引Level
- 查询、插入、删除的时间复杂度为O(log n)

3. 索引原理-SkipData

将搜索时耗时转嫁给索引时，这就是Lucene索引的“空间换时间”的基本思想



索引时构建了SkipList，在Segment中每个Term都有自己唯一的Postings，每个Postings都有需要构建一个SkipList。

引入跳表主要是面向搜索时的优化，提高结果集之前取交集时的效率。

3. 索引原理

Lucene索引一个文档的大致步骤有：

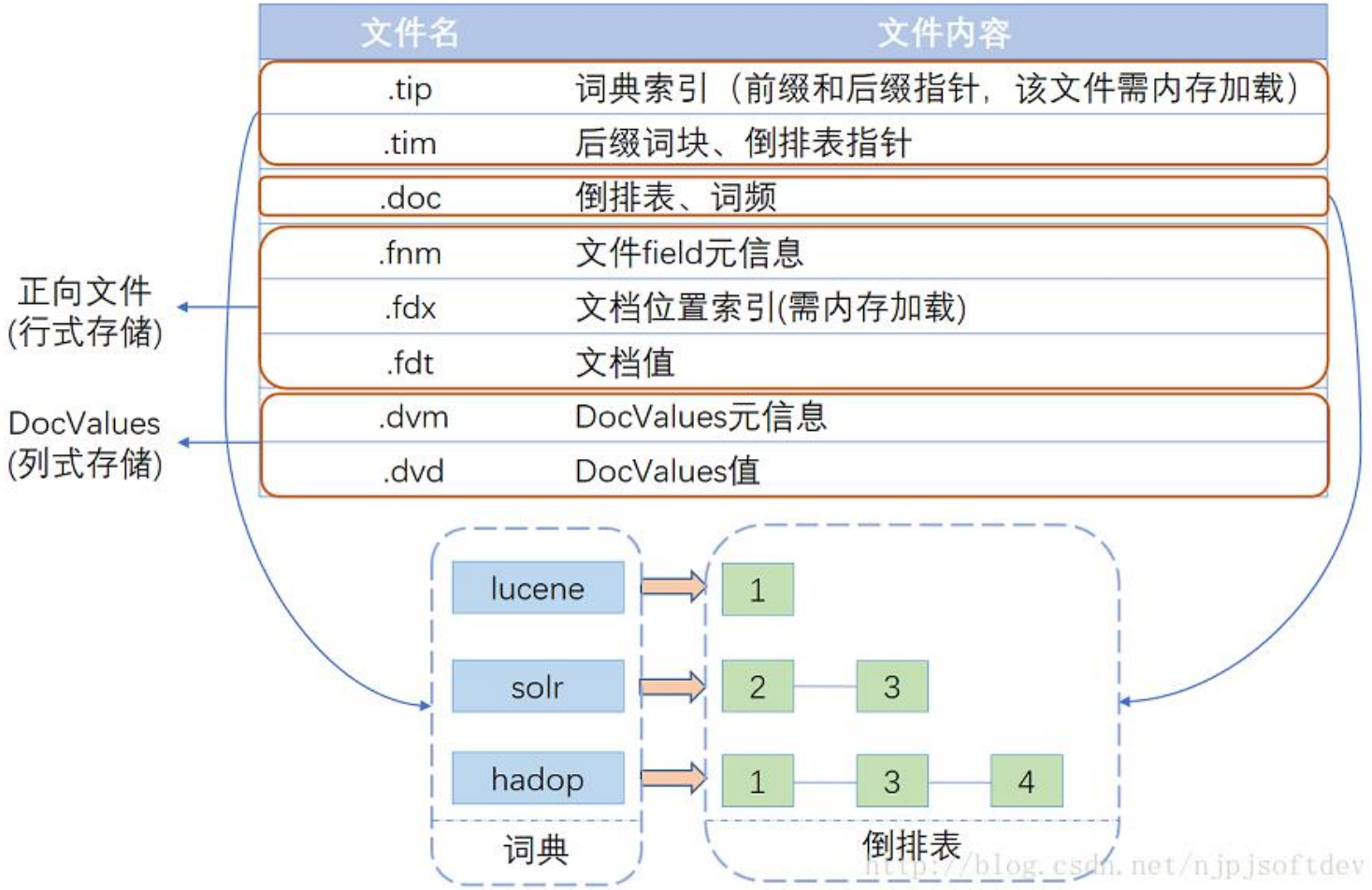
- 存储原始文档，生成正向索引
- 构建词典（Term dict），主要分为**前缀索引FST**（有限状态转换机）和**后缀词块**（Block）
- 构建倒排表（Posting list），由多个成对的**词元信息**和**文档号跳表组成**

词索引（Term index）以FST的结构缓存在内存中，从词索引查到关键词对应的后缀词块位置后，再去磁盘上找term，**内存与磁盘结合，提高了查询速度，并减少了磁盘IO的次数**。

3. 索引原理

Lucene各后缀文件含义

- .tip：词典索引（前缀索引），使用FST。
- .tim：后缀词块+倒排表指针，和.tip一起实现词典的作用。
- .doc：倒排表，运用了跳表。
- .fdx：文档正向索引
- .fdt：文档值，即文档源的存储。



4

物理结构

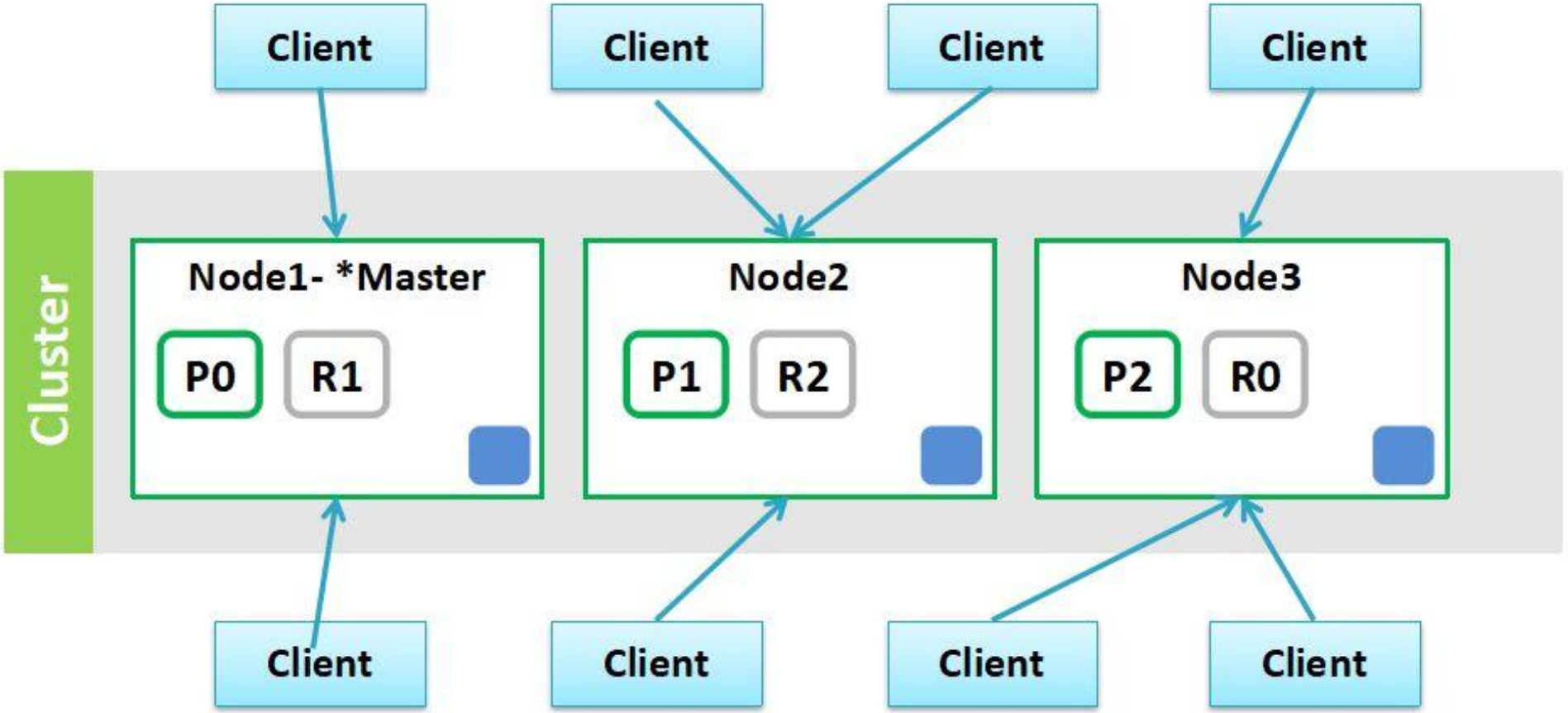
集群架构、分片、选主等原理。

4. 物理结构

在一个ES节点中，有以下概念：

- 节点（node）：集群中的每一个ES实例，即一个ES进程，就是一个节点
- 索引（index）：类似于关系型数据库的DB，一类数据的逻辑汇总。
- 分片（shard）：ES默认创建索引会把它分成5个主分片，**ES的分片底层对应的就是Lucene的索引**，分片也是ES数据迁移的最小单位。
- 副本分片（replica shard）：每一个分片都可以有0到多个副本，而每一个副本也都是分片的完整拷贝，主分片无法访问是副本分片会被提主。

4. 物理结构



4. 物理结构-节点类型

- master：有资格竞选 Master 的节点，负责 es 集群的节点发现和元数据管理，一般需要配奇数个（Zen Dicovery选主）
- data：存储数据的节点，数据以Shard为单元保存，每个Shard就是一个完整的Lucene Index
- client：不配置任何角色的节点，只用于客户端访问、请求分发和结果汇总。
- ... ...

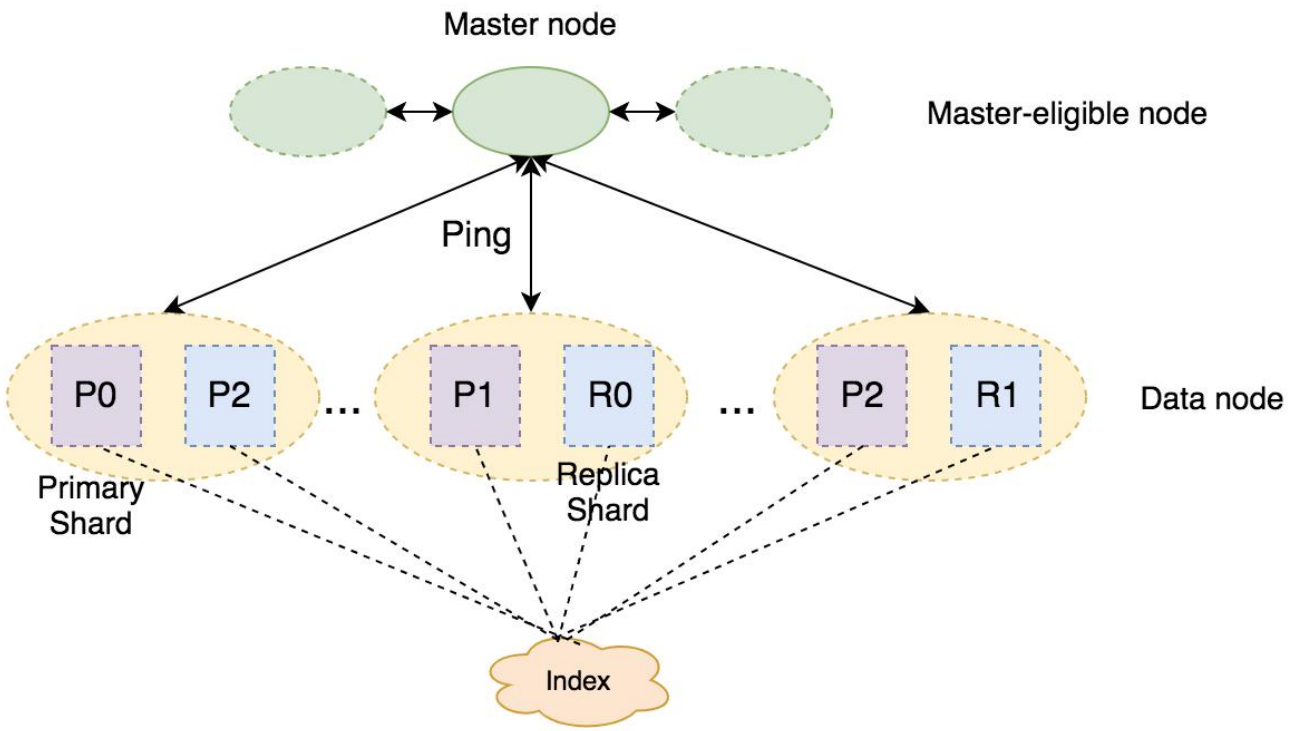
4. 物理结构-选主

7.X之前ES的Master选主默认采用Zen Discovery，是一个相对轻量级的选主模块。

- 使用Gossip进行集群信息收集。
- 基于收集的信息投票

7.X之后ES自行实现了Raft协议进行选主。

4. 物理结构



- master节点会周期性向其它节点发送ping命令，做探活同时获取分片在各个data节点的分布情况，从而不断维护和更新整个集群的元数据信息。
- 其它节点也会ping master来探活，如果master挂了就会重新选举。
- 每个节点都维护了一份路由表，记录Index对应的分片在data节点的分布情况，只有master可以更新路由表，并在更新后publish给其他节点。

补充参考

- FST详解：https://www.shenyanchao.cn/blog/2018/12/04/lucene-fst/
- Lucene倒排索引实现原理探秘：http://www.nosqlnotes.com/technotes/searchengine/lucene-invertedindex/
- 跳表详解：https://www.jianshu.com/p/9d8296562806
- ES Zen Discovery 选主实现原理：https://niceaz.com/2018/11/12/es-zen-discovery-master-election/


THANK YOU
網易 NETEASE

ElasticSearch索引与架构原理

补充参考

- FST详解：https://www.shenyanchao.cn/blog/2018/12/04/lucene-fst/
- Lucene倒排索引实现原理探秘：http://www.nosqlnotes.com/technotes/searchengine/lucene-invertedindex/
- 跳表详解：https://www.jianshu.com/p/9d8296562806
- ES Zen Discovery 选主实现原理：https://niceaz.com/2018/11/12/es-zen-discovery-master-election/