# Elasticsearch

## 介绍1

ElasticSearch是另一个开源的分布式的基于Lucene的Java全文搜索引

擎，用来索引和搜索文本数据。Elasticsearch实际上就是基于Lucene的Java库上的一个架构[Elasticsearch, 2012]o Elasticsearch中关于匹配文本和存储优化好的索引的算法都是由Lucene来实现的。Elasticsearch本身在Lucene的搜索实现上提供一个更加可用的、更加简洁的API、它是高度可扩展的，设计在云计算环境中使用，根据使用者的需求，它允许动态地添加和删除节点。它具有几个特点，使得它易于使用。具体来说，它支持在同一网络中自动发现其他Elasticsearch节点，自动根据可用节点个数来划分和分配分片与副本。用户可以使用REST、native, thrift来搜索被索引好的数据。

Elasticsearch的所有的命令都通过HTTP请求来发送。Elasticsearch的一

个主要强项就是schema-free的模型、multi-tenancy、分布式计算中的可扩展性和近乎实时的搜索。除非显式地定义Schema , Elasticsearch是Schema-free的，所有的JSON文件都可以被添加到Elasticsearch中。这带来了极大的灵活性和零配置。Elasticsearch还提供所谓的“Multi-Tenancy"，这可以让document被分为不同的“indexes"。一个索引就相当于关系型数据库中的一个数据库。Elasticsearch提供分片机制，索引可以被分割为不同的分片。索引和分片的副本配置使得在分布式环境中不同的节点之间备份和复制数据成为可能。ES的另一个设计特点就是在多节点、分布式的环境下近乎实时的搜索。

它提供一个RESTfuI的服务接口，使用JSON的文件格式。在一个分布式的Elasticsearch部署中，索引数据也被划分为许多个分片，被分配给不同的数据节点。同时，没有节点充当master的角色，所有的节点都是平等的数据节点，每个节点都可以接受从client发送的请求，寻找合适的节点来处理请求，最终把结果返回给client

Elasticsearch可以作为一个单进程来运行，它的优点就是它的P2P特性，这使得它可以在同一个网络中的多个节点上动态地添加和删除节点。因此，可以根据当前的需求来提供资源，自动适应当前日志数据量的大小，不会造成资源紧张或是浪费。不仅如此，根据节点的配置，每个节点实例还可以扮演一个或多个角色(client, master, data >，这可以用来配置分层架构的集群。然而，当前的接口还不支持封装或是验证，所以，每个可以访问集群的人都可以获得集群中的数据。为了强调这个限制，最好保证Elasticsearch节点集群是私人的，只对获得授权的用户或者服务开放。在云中建立多个Elasticsearch节点的集群和启动每个Elasticsearch节点一样的简单。通过Elasticsearch的自动发现和自动分片功能，节点会在拓扑中自动地发布自己的信息给其他节点，同时做好接收数据来进行索引和回应查询的准备。不需要对节点进行额外的配置，尤其不需要额外的数据schema。只需要插入数据，Elasticsearch会正确地处理它们。

## 介绍2

Elasticsearch诞生于2010年2月，是一款基于Apache Lucene构建的开源的全文搜索服务器，它采用Java语言编写并使用Lucene构建索引，提供搜索功能。Elasticsearch的目标是让全文搜索变得简单，开发者可以通过它简单明了的RESTFul AP工轻松地实现搜索功能，而不用去面对Lucene的复杂性。Elasticsearch能够轻松地进行大规模的横向扩展，以支撑PB级的结构化和非结构化的海量数据的处理。

要明白Elasticsearch的工作机制和原理，首先需要了解Elasticsearch技术中的相关核心概念。

Document: Document即文档，与Lucene中的文档类似，能够被搜索到的数据都是以Documents的形式存储的。Document有一个或者多个Field组成，一个Field即一个键值对，它有一个名字和对应的值。Document以JSON的形式存在，例如JSON串”{"name":“小明”，"age":20}”可以表示一个Document，它有2个域，分别为name和age，并具有各自的值:“小明”、20。

Type:每一个Document都是属于一种type的，它表明了Document的类型，例如前面的例子表示的Document我们可以定义其Type为Person。

Mapping: mapping用来说明type的具体属性，指明document中每个field是何种类型，是否对该Field的值进行存储、建立索引、分词等等。

Index:工ndex是相当于关系型数据库中的数据库，它可以包含有多个type,type则相当于关系型数据库中的表。Document相当于一条具体的记录。

上面的4个概念说明了Elasticsearch中数据的表示形式，存储逻辑。

Elasticsearch还有几个重要的概念。

Node:表示一台Elasticsearch服务器实例。多台Node都成Elasticsearch

集群。

Cluster:即Elasticsearch集群。多台单机节点(Node)通过运行相同名字

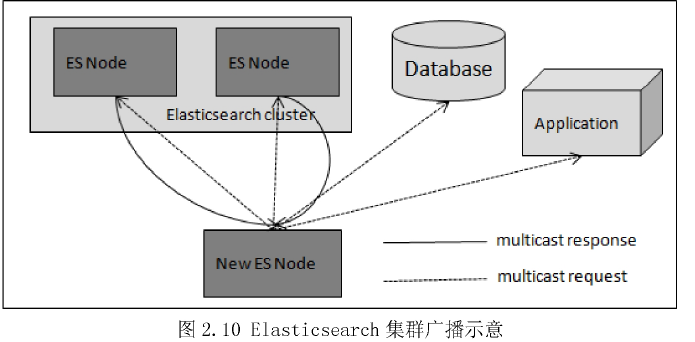
的Elasticsearch进程并互相保持通信，在逻辑上构成一个Elasticsearch集群。Elasticsearch集群内部使用Zookeeper进行分布式协调服务，会自动选举出一个节点作为主节点，用来分配作业，保存集群运行状态，但是对于集群外部来说，用户与Elasticsearch集群任意节点通信都是与Elasticsearch集群整体在通信。

Shards:代表索引分片，Elasticsearch把Lucene的索引文件做了进一步处理，分成了若干个更小的片段，这样做有很多好处，首先Elasticsearch进行数据冗余存储是针对shard进行冗余，具有更高的容错性，而且，分片之后方便分布式存储和分布式搜索。索引文件分片的参数是可以通过配置文件指定的，默认为5。

### 分布式原理

当一个节点上启动Elasticsearch进程之后，该节点向集群中其他的节点发送广播，配置以相同的集群名称的节点相互连接组成集群。

如图2. 10，只有集群中配以相同集群名的运行着Elasticsearch服务进程的节点会返回信息。构成集群之后会自动选举产生master节点，master节点负责监控集群状态，当集群之中有节点出现故障无法正常工作，master节点能及时得到反馈，master节点自身出现故障，剩下的节点自动选举产生新的master节点。



### 与elasticsearch集群交互

Elasticsearch提供了简单易用，功能强大大AP工与其进行交互，能够支持java, javascript, pyhon, c#, php等等多种主流语言。其中Elasticsearch集群内部各节点之间即用java语言API进行通信。Elasticsearch的RESTFul API支持直接以http请求的方式进行索引、搜索、删除等操作，索引及搜索过程中文档均以JSON形式表示，格式化查询请求也以JSON格式的字符串表示

## 问答类

### 相关知识

Elasticsearch是一个开源的分布式实时搜索与分析引擎，支持云服务。它是基于Apache Lucene搜索引擎的类库创建的，提供了全文搜索能力、多语言支持、专门的查询语言、支持地理位置服务、基于上下文的搜索建议、自动完成以及搜索片段（snippet）的能力。 Elasticsearch支持RESTful的API，可以使用JSON通过HTTP调用它的各种功能，包括搜索、分析与监控。此外，它还为Java、PHP、Perl、Python以及Ruby等各种语言提供了原生的客户端类库。

Elasticsearch是一个高伸缩、高可用、基于Apache Lucene的开源搜索与分析引擎。通过它你可以很方便地对数据进行深入挖掘，可以随时放大与缩小搜索与分析的区间，并且这一切都是实时的。

举例来说，当你使用它搜索数据的时候，可以使用传统的查询（‘查找满足条件Y的所有项X’）进行过滤（在Elasticsearch术语中称为“视图”），高亮显示搜索片段，为每条结果提供相应的上下文。也可以使用地理位置（‘查找在Z里之内的所有项’），或是为用户提供搜索关键字建议，并且提供了强大的聚合（即Elasticsearch中的“分面”（facet））能力，例如时间分布图或者统计图。

Elasticsearch既可以搜索、也可以保存数据。它提供了一种半结构化、不依赖schema并且基于JSON的模型，你可以直接传入原始的JSON文档，Elasticsearch会自动地检测出你的数据类型，并对文档进行索引。你也可以对schema映射进行定制，以实现你的目的，例如对单独的字段或文档进行boost映射，或者是定制全文搜索的分析方式等等。

Elasticsearch运行在JVM之上，它使用JSON格式，通过RESTful HTTP接口的方式访问，因此任何一种客户端或语言都能够与其交互。目前已经有了大量的客户端和框架的整合方案，包括对多种编程语言的支持，通过这些原生的API与专门的DSL将不一致的地方最小化，并实现性能最大化。

Elasticsearch非常适合于大数据的场合，它的高伸缩性与分布式架构的本质使得对大量信息的搜索与存储都可以在近乎实时的情况下完成。通过Elasticsearch-Hadoop这个项目，我们使Hadoop使用者（这里也包括Hive、Pig和Cascading）能够用一个成熟的搜索引擎来增强他们的工作流

### 搜索差异

Elasticsearch是用于实时全文搜索的，实时的全文搜索与传统的数据搜索的差异如下：

按照通俗的说法，传统的搜索其实就是全文搜索的一个子集。

大多数数据存储系统都是基于元数据，或是原始数据的某些部分实现搜索功能的。出于效率方面的原因，那些我们认为相关的数据子集（例如对象的id、name等等）会被索引，而其余部分则被忽略。与原始数据大小相比，所生成的索引数据量比较小，但它没有完全覆盖到所有的数据集。全文搜索解决这个问题的方式是对整个数据资料进行索引和搜索，代价就是对存储空间的需求增加。

传统的搜索通常是与结构化数据相关的，因为对用户来说结构化数据更容易理解哪些数据是相关的，而哪些是不相关的。但当前的业务中，大多数数据都是非结构化的。你会将所有数据一次性保存起来，在需要的时候再去获取它的各种不同格式与结构的值，在这种情况下就必需用到全文搜索的方式，因为你已经无法忽略任何数据了。

Elasticsearch同时支持结构化数据搜索与全文搜索，它提供了大量的查询选项，包括关键字、布尔查询、过滤器以及模糊查询等方式，所有这些都选项都可以通过一个丰富的查询语言来完成。

请注意，Elasticsearch与简单的全文搜索相比，还提供了以下一些额外的特性：

地理位置查询：根据数据的地理位置查找结果。

聚合/分面：在查询的时候进行数据聚合，例如查找在某一天之内，访问过你的网站中某篇特定文章，或者是某些标签的用户来自哪些国家。由于聚合是实时计算的，因此当查询变化时，聚合也会相应的变化。换句话说，就是你总能获取数据集的即时反馈。

### 设计的考虑

数据是王道，因此要专注于对它的处理。为了让Elasticsearch能够按照你设想的方式处理数据，它必须理解你的“需求”。虽然它能够尽最大努力去猜测你要的数据，但你的领域知识对于进行正确的配置以支持你的需求来说才是最有价值的部分。这些都可以归结为数据的粒度或者是组织方式。让我们拿日志这个常见的需求来举个例子：与其把所有的日志记录都放在一个巨大的索引里，更好的方法是将日志记录按照时间进行分解，因此每月、每周或者每天的数据都可以对应一条索引。这种分解方式能够在你处理数据巨量增长、以及删除或者归档旧数据时带来许多便利

### 支持的设计和架构模式

一个索引包括了多个分片，每个分片本身就是一个迷你的搜索引擎。一个索引本质上就是对应着一系列分片的虚拟命名空间。多分片的方式能够简化横向扩展任务，只需添加新的节点即可。而创建分发分片对每个主分片的数据进行复制能够实现高可用性，并增加读取时的吞吐能力。

查询某个索引是一种分布式操作，这意味着Elasticsearch不得不对索引中的每个分片的数据复制进行查询，并将结果整理到一个单一的结果集中。而对多个索引进行查询只是将相同的步骤进一步扩大了而已。在设计你的数据存储架构时，这种方式能够带来极大的灵活性。

而如果用户了解他的应用程序特定的一些领域知识，那就可以对查询进行优化，使查询只触及相关的分片。在同样的硬件条件下，这种方式可以支持更大的负载

### 如何支持数据的可伸缩性

为了支持高可用性与高伸缩性，Elasticsearch本身就是分布式设计的。从顶层的角度来说，Elasticsearch在索引（或者集合）中保存文档（或者数据记录），每个集合又分解为多个小块，称为分片。索引越大，所需要分配的分片越多（不必担心会创建过多的分片，它的开销很小）。取决于Elasticsearch的设置和规模，分片会在集群中均匀地平均分布，有两个原因：

出于冗余方面的原因：默认情况下，Elasticsearch为每个分片都准备了一份拷贝，一旦某个节点停机了，备份的分片就能接替它的位置。

出于性能方面的原因：每个查询都发生在某个索引上，并且会在多个分片中并行运行，这种工作流方式是改善性能的关系所在。如果感觉运行速度缓慢，只需简单地在集群中加入新的机器，Elasticsearch就会自动地将分片与查询进行分布到新添加的机器上。

这种方式让使用Elasticsearch的组织可以自由选择进行纵向扩展（如果节点运行缓慢就升级硬件）或者横向扩展（如果集群整体速度慢就加入更多的节点）。

### 挑战

最大的挑战是用户从SQL的世界来到了这个基于上下文的搜索世界时所面对的新理念，对于获取独立的数据项（传统的数据获取）来说，方法基本还是一样的，只要指定id然后取回数据就可以了。但对于数据探索来说，所使用的各种构造就有所不同了，例如采用不同类型的分析方式，或是使用哪种类型的搜索或匹配算法（如模糊查询等等）。

### 与Hadoop技术结合的优势

Hadoop在设计上就是一个分布式的，面向批处理的平台，用以处理大数据集。虽然它是一个非常强大的工具，但它的批处理的本质意味着在处理结果时需要花费一定时间。此外，用户必需重新为各种操作编写代码。Hive或者Pig这样的类库能起到一定作用，但不能完全解决问题，想象一下在Map/Reduce中重新实现地理位置查询的难度吧。

而使用Elasticsearch，你就可以将搜索工作交给搜索引擎去完成，而专注于其它方面的工作，例如数据转换。Elasticsearch-Hadoop项目为Hadoop提供了直接的整合功能，因此用户使用起来没有任何障碍，我们为vanilla Map/Reduce提供了专门的InputFormat与OutputFormat，为Cascading的数据读写提供了Taps，并为Pig和Hive提供了Storages。这样你就可以以HDFS一样方式的访问Elasticsearch的数据了。

通常来说，与Hadoop进行整合的数据存储系统都可能会成为系统的瓶颈，这是由于每个job在集群中的各种任务会造成大量的请求。而Map/Reduce模型的分布式特性用于Elasticsearch上会配合得非常良好，因为我们能够将某个特定查询所生产的Map/Reduce任务数量与Elasticsearch的分片数量相关联。这样每次有查询运行时，系统就会动态地按照Hadoop的划分生成一个数值，该数值与Elasticsearch的可用分片数量成正比，这样各个job就能够并行地运行了。你可以按照Elasticsearch的数量对Hadoop集群进行扩展，或者是反过来也可以。

此外，这种整合将分片的信息暴露给Hadoop，以此可以实现协同定位。Job的任务会在每个Elasticsearch分片所在的同一台机器上运行，通过实现数据本地化消除了网络的开销，并改善了性能。出于这个原因，我们建议你在相同的机器上运行Elasticsearch和Hadoop集群，尤其是他们能够互相平衡资源的使用情况（I/O与CPU）。

最后但也是很重要的一点是，Elasticsearch能够提供近乎实时的响应速度（毫秒等级），这极大的改善了Hadoop job的执行速度以及执行的各种开销，在类似于Amazon EMR这种“租用的资源”上运行时的改善尤其明显。