**Elasticsearch集群的概念（cluster）**

在一个分布式系统里面,可以通过多个elasticsearch运行实例组成一个集群,这个集群里面有一个节点叫做主节点(master),elasticsearch是去中心化的,所以这里的主节点是动态选举出来的,不存在单点故障。

在同一个子网内，只需要在每个节点上设置相同的集群名,elasticsearch就会自动的把这些集群名相同的节点组成一个集群。节点和节点之间通讯以及节点之间的数据分配和平衡全部由elasticsearch自动管理。

在外部看来elasticsearch就是一个整体。

**Elasticsearch 节点(node)**

每一个运行实例称为一个节点,每一个运行实例既可以在同一机器上,也可以在不同的机器上。

所谓运行实例,就是一个服务器进程，在测试环境中可以在一台服务器上运行多个服务器进程，在生产环境中建议每台服务器运行一个服务器进程。

**Elasticsearch索引(index)**

Elasticsearch里的索引概念是名词而不是动词，在elasticsearch里它支持多个索引。

优点类似于关系数据库里面每一个服务器可以支持多个数据库是一个道理，在每一索引下面又可以支持多种类型，这又类似于关系数据库里面的一个数据库可以有多张表一样。

但是本质上和关系数据库还是有很大的区别，我们这里暂时可以这么理解。

**Elasticsearch 分片(shards)**

Elasticsearch 它会把一个索引分解为多个小的索引，每一个小的索引就叫做分片。

分片之后就可以把各个分片分配到不同的节点中去。

**Elasticsearch副本(replicas)**

Elasticsearch的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，好处是可以用它来增加速度的同时也提高了系统的容错性。

一旦Elasticsearch的某个节点数据损坏或则服务不可用的时候，那么这个时就可以用其他节点来代替坏掉的节点，以达到高考用的目的。

**Elasticsearch 的recovery概念**

Elasticsearch 的recovery代表的是数据恢复或者叫做数据重新分布。

elasticsearch 当有节点加入或退出时时它会根据机器的负载对索引分片进行重新分配，当挂掉的节点再次重新启动的时候也会进行数据恢复。

**Elasticsearch river**

Elasticsearch river 代表的是一个数据源，这也是其它存储方式（比如：数据库）同步数据到 elasticsearch 的一个方法。

它是以插件方式存在的一个 elasticsearch 服务，通过读取 river 中的数据并把它索引到 elasticsearch 当中去，官方的 river 有 couchDB、RabbitMQ、Twitter、Wikipedia。

**Elasticsearch 的 gateway 概念**

gateway 代表 elasticsearch 索引的持久化存储方式，elasticsearch 默认是先把索引存放到内存中去，当内存满了的时候再持久化到硬盘里。

当这个 elasticsearch 集群关闭或者再次重新启动时就会从 gateway 中读取索引数据。

elasticsearch 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。

**Elasticsearch的discovery.zen概念**

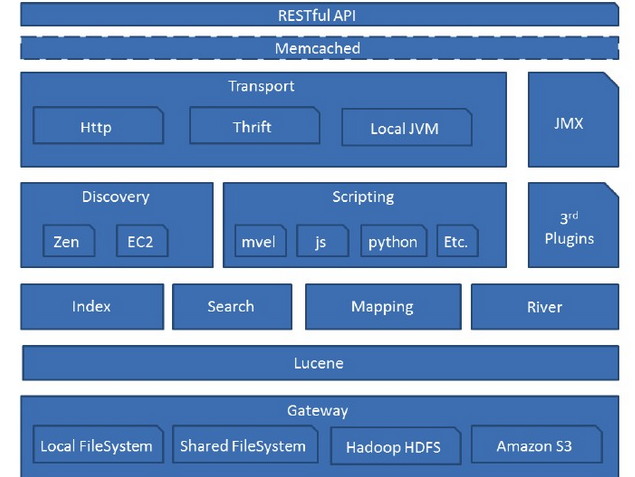
discovery.zen代表 elasticsearch 的自动节点发现机制，而且 elasticsearch还是一个基于 p2p 的系统。

首先它它会通过以广播的方式去寻找存在的节点，然后再通过多播协议来进行节点之间的通信，于此同时也支持点对点的交互操作。

**Elasticsearch里Transport的概念**

Transport代表 elasticsearch 内部的节点或者集群与客户端之间的交互方式。

默认的内部是使用 tcp 协议来进行交互的，同时它支持 http 协议（json格式）、thrift、servlet、memcached、zeroMQ等多种的传输协议（通过插件方式集成）。



es包含以下特性：

1. 分布式索引、搜索。
2. 索引自动分片、负载均衡。
3. 自动发现机器、组建集群。
4. 支持Restful 风格接口。
5. 配置简单等。

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。

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

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。

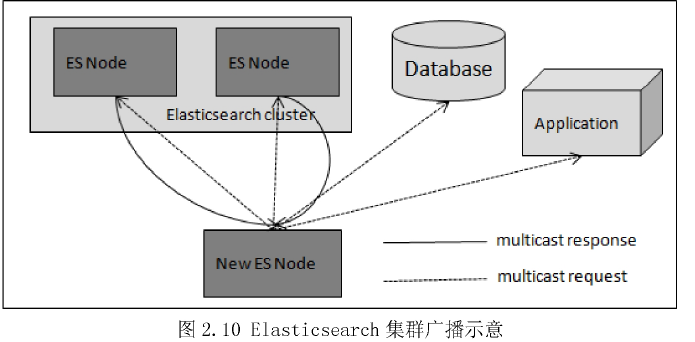
特点：



## 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

## 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。

## 技术结合

Elasticsearch提供了Hadoop Plugin，是Hadoop和elasticsearch的集成插件，可以通过Hadoop的mapreduce算法来并行创建索引，同时支持cascading，hive，pig等框架