elasticsearch7.0 新特性

一、 Release highlights(7.0 发布亮点)

1.Adaptive replica selection enabled by default （自适应索引副本选择 默认启用）

在Elasticsearch 6.x和之前，对同一分片的一系列搜索请求将以循环方式转发到主副本和每个副本。如果一个节点启动一个长的垃圾收集，这可能会有问题 - 搜索请求仍然可以转发到慢节点，无论如何都会对搜索延迟产生影响。

在6.1中，添加了一个实验性自适应副本选择功能。每个节点跟踪并比较搜索请求到其他节点的时间，并以此调整向特定节点上的分片发送请求的频率。在我们的基准测试中，这导致搜索吞吐量的整体提高，并减少了99%的延迟。默认情况下，此选项在整个6.x中被禁用，但我们听到用户发现该设置非常有用的反馈，因此我们默认情况下从Elasticsearch 7.0.0开始启用它。

2.Skip shard refreshes if a shard is "search idle" （如果一个分片处于搜索空闲 ，就跳过分片刷新）

Elasticsearch 6.x和先前刷新的索引会在后台自动刷新，默认每秒一次。这提供了Elasticsearch的“近实时”搜索功能：默认在文档被添加后一秒可用于搜索请求。但是，如果不需要刷新，则此行为会对索引性能产生重大影响（例如，如果Elasticsearch不为任何活动搜索提供服务）。

引入了分片“搜索空闲”的概念后，Elasticsearch 7.0对此行为更加智能。默认，在没有任何搜索30秒后，分片现在转换为搜索空闲状态。一旦分片搜索空闲，将跳过所有定时的刷新，直到搜索请求到来，并触发下一个定时的刷新。我们知道，这将显着提高许多用户的索引吞吐量。仅当没有显式设置刷新间隔（settings：index.refresh_interval 字段）时才应用此方式，因此需要为偏好旧的刷新行为的索引显式设置刷新间隔。

3. Default to one shard （默认为一个分片）

多年来从用户那里看到的最大麻烦之一就是过度分片，这个主要祸源之一来自默认分片原则。在Elasticsearch 6.x和之前，我们默认默认为每个索引五个分片。如果您有十个不同应用程序的每日索引，并且每个应用程序默认有五个分片，那么您每天创建五十个分片，并且不久您就有了数千个分片，即使您只为每个数据库索引几千兆字节数据天。索引生命周期管理是帮助实现这一目标的第一步：提供本机翻转功能，以按大小而不是（仅）按日创建索引，并使用内置缩减功能缩小每个索引的分片数。将索引默认为一个分片是帮助减少过分片的下一步。当然，如果您有另一个首选主分片计数，则可以通过索引设置进行设置。

4. Lucence 8 （使用Lucene8最新版本）

与每个主要版本一样，我们希望支持Lucene的最新主要版本，以及随之而来的所有优点。这包括我们为新Lucene版本做出的所有开发。 Elasticsearch 7.0捆绑了Lucene 8，这是Lucene的最新版本。 Lucene版本8作为Elasticsearch其余部分的许多功能改进的基础，包括改进的top-k查询的搜索性能以及在保持速度的同时为搜索结合相关信号的更好方法

5. Introduce the ability to minimize round-trips in cross-cluster search(跨集群搜索最小化往返次数)

在Elasticsearch 5.3中，我们发布了一项称为跨群集搜索的功能，供用户跨多个群集进行查询。我们已经改进了跨群集搜索框架，添加了最终使用它来弃用和替换部落节点的功能，以此作为联合查询的一种方式。在Elasticsearch 7.0中，我们为跨群集搜索添加了一种新的执行模式：一种在不需要时往返次数较少的模式。跨集群搜索查询跨越高延迟（例如，跨WAN）时，此模式（ccs_minimize_roundtrips）可以导致更快的搜索。

6.New cluster coordination implementation （新的集群协调实现）

从一开始，我们专注于使Elasticsearch易于扩展并适应灾难性故障。为了支持这些要求，我们创建了一个可插拔的集群协调系统，其默认实现为Zen Discovery。 Zen Discovery本来就是毫不费力的，让我们的用户安心（顾名思义）。 Elasticsearch使用量的迅速增长给我们带来了很多好处。例如，Zen的minimum_master_nodes设置经常被错误配置，这使得集群处于脑裂和丢失数据的更大风险。在大型和动态调整大小的集群中维护此设置也很困难。

在Elasticsearch 7.0中，我们已经完全重新思考并重建了集群协调层。新的实现提供了安全的亚秒级主选举时间，Zen可能花了几秒钟选出一个新的主人，这是一个关键任务部署的宝贵时间。删除minimum_master_nodes设置后，增长和缩小群集变得更安全，更容易，并且使系统配置错误的空间更小。最重要的是，新的集群协调层为Elasticsearch的未来提供了强大的构建块，确保我们可以为更高级的用例构建功能

7.Better support for small heaps (the real-memory circuit breaker)（ 更好地支持小堆（真实存储器断路器））

Elasticsearch 7.0添加了一个全新的断路器，可以跟踪JVM使用的总内存，如果保留加实际堆使用率超过95％，则会拒绝请求。我们还将更改聚合的默认最大存储桶（search.max_buckets）返回数到10,000，默认情况下在6.x和之前是无限制的。这两个改进在防止7.x Elasticsearch内存不足很重要，即使面对用户运行大型查询和聚合的时候，也可帮助保持群集的存活

8.Cross-cluster replication is production-ready （跨集群复制CCR 已生产就绪）

我们在Elasticsearch 6.5中引入了跨集群复制作为beta功能。跨集群复制是Elasticsearch要求最多的功能。我们很高兴地宣布跨集群复制现已普遍可用，并可在Elasticsearch 6.7和7.0中进行生产使用！跨集群复制有多种使用案例，包括跨数据中心和跨区域复制，复制数据以更接近应用程序服务器和用户，以及维护从大量较小群集复制的集中报告群集。

跨集群复制不仅晋升到GA功能外，CCR还有6.7和7.0的许多重要技术进步。以前版本的跨群集复制只需要在新索引上启动复制：无法复制现有索引。跨群集复制现在可以开始复制在6.7和7.0中启用了软删除的现有索引，并且新索引默认启用了软删除。我们还推出了新技术，以防止追随者指数远远落后于领先指数。我们在Kibana中添加了一个管理UI，用于配置远程集群，复制索引和自动复制的索引命名模式（例如，用于复制metricbeat- *索引）。我们还添加了一个监控UI，用于深入了解跨群集复制进度和错误警报。查看跨群集复制入门指南，或访问参考文档以了解更多信息。

9.Index lifecycle management is production-ready(索引生命周期管理 已生产就绪)

索引生命周期管理（ILM）作为Elasticsearch 6.6中的beta功能发布。我们已正式将ILM从测试版转移到GA版本中，为生产使用做好准备！ ILM可以轻松管理Elasticsearch中数据的生命周期，包括数据如何在热，暖，冷和删除四个阶段之间进行。有关数据如何在这些阶段中移动的具体规则可以通过Elasticsearch中的API或Kibana中的漂亮管理UI创建。

在Elasticsearch 6.7和7.0中，ILM现在可以管理冻结索引。冻结索引对于Elasticsearch中的长期数据存储很有价值，并且需要少量的与节点管理的数据量相关的内存（堆）。在6.7和7.0中，冻结索引现在可以被冻结作为ILM冷阶段的一部分。此外，ILM现在可以直接使用跨群集复制（CCR），它也是Elasticsearch 6.7和7.0版本中的GA（可生产使用的功能）。可以在文档中找到每个ILM阶段中可用的潜在操作。 ILM可以免费使用，也是Elasticsearch的默认发版的一部分。

10.SQL is production-ready (SQL支持 已生产就绪)

Elasticsearch的SQL接口现在是GA版本。 SQL界面在6.3中作为alpha版本引入，允许熟悉SQL的开发人员和数据科学家使用Elasticsearch的快速、可伸缩性和全文功能，这是其他人所熟知和喜爱的。它还允许使用SQL的BI工具轻松访问Elasticsearch中的数据。除了在Elasticsearch中批准SQL访问作为GA功能外，我们还将JDBC和ODBC驱动程序指定为GA。有四种方法可以访问Elasticsearch SQL：通过Elasticsearch REST终端，Elasticsearch SQL命令行界面，JDBC驱动程序和ODBC驱动程序。

11.High-level REST client is feature-complete（高级rest 客户端，功能齐全）

如果你一直关注我们的博客或我们的GitHub仓库，你可能已经知道我们已经工作了很长一段时间的任务：创建一个用于访问Elasticsearch集群的下一代Java客户端。我们开始研究最常用的功能，如搜索和聚合，并一直在通过管理和监控API。许多使用Java的人已经在使用这个新客户端，但对于那些仍在使用TransportClient的人来说，现在是升级到我们的高级REST客户端或HLRC的好时机。

从7.0.0开始，HLRC现在已经检查了所有API复选框以将其称为“完成”，因此仍然使用TransportClient的那些人应该能够迁移。我们当然会继续开发我们的REST API，并将它们添加到此客户端。有关所有可用API的列表，请查看我们的HLRC文档。要开始使用，请查看我们文档的HLRC开始入门，如果您需要从TransportClient迁移，请查看我们的迁移指南

12.Support nanosecond timestamps （支持纳秒级时间戳）

直到7.0 Elasticsearch只能以毫秒精度存储时间戳。如果您想处理以更高速率发生的事件 - 例如，如果您想在Elasticsearch中存储和分析跟踪或网络数据包数据 - 您可能需要更高的精度。从历史上看，我们使用Joda时间库来处理日期和时间，而Joda缺乏对这种高精度时间戳的支持。

使用JDK 8，引入了官方Java时间API，它还可以处理纳秒精度时间戳，在过去的一年中，我们一直在努力将Joda时间使用迁移到本机Java时间，同时尝试保持向后兼容性。从7.0.0开始，您现在可以通过专用的date_nanos字段mapper使用这些纳秒时间戳。请注意，使用此字段的聚合仍然是以毫秒级处理，以避免出现桶聚合爆炸。

13.Faster retrieval of top hits（更快获取命中的顶端数据）

在搜索方面，查询性能是一项关键功能。对于不需要精确命中计数的情况，我们已经在Elasticsearch 7.0中对搜索性能进行了重大改进，并且足以把结果数量设置一个低的下界。例如，如果您的用户通常只是查看您网站上的第一页结果，而不关心匹配的文档数量，那么您可以向他们展示“超过10,000个命中”，然后为他们提供分页结果。让用户在他们的查询中输入诸如“the”和“a”之类的频繁出现的术语是很常见的，这在历史上迫使Elasticsearch对很多文档进行评分，即使这些频繁的术语不可能对得分增加太多。

在这些条件下，Elasticsearch现在可以跳过计算在早期阶段识别为不会在结果集顶部排名的记录的分数。这可以显着提高查询速度。数据结果得分最高的实际数量是可配置的，但默认值为10,000。具有小于此阈值的结果集的查询的行为将不会改变 - 即结果计数是准确的，但是对于匹配少量文档的查询没有性能改进。由于改进基于跳过低排名记录，因此不适用于聚合。您可以在我们的博客文章中看到有关这一强大的算法开发的更多信息：在Wlasticsearch中对Top Hits的快速检索。

14.Support for TLS 1.3 （TLS1.3(规范化的SSL）协议的支持）

Elasticsearch长期以来一直支持加密通信，但是，我们最近开始支持JDK 11，它为我们提供了新的功能。 JDK 11现在支持TLS1.3，所以从7.0开始，我们现在支持Elasticsearch中的TLSv1.3用于那些运行JDK 11的用户。为了避免新用户无意中以低安全性运行，我们的默认选项中删除了TLSv1 .0。对于那些运行旧版Java的用户，我们有TLSv1.2和TLSv1.1的默认选项。如果您需要入门帮助，请查看我们的TLS设置说明。

15.Bundle JDK in Elasticsearch distribution（Elasticsearch发行版中捆绑了JDK）

我们看到用户遇到的一个更突出的“入门障碍”一直是不知道Elasticsearch是一个Java应用程序，他们需要先安装一个受支持的JDK。有了7.0，我们现在捆绑了一个OpenJDK发行版，以帮助用户更快地开始使用Elasticsearch。我们知道有些用户更喜欢JDK发行版，因此我们也支持自带JDK。如果你想带自己的JDK，你仍然可以在启动Elasticsearch之前设置JAVA_HOME。

16.Rank features（排名功能）

Elasticsearch 7.0有几种新的字段类型可以充分利用您的数据。帮助核心搜索用例的两个是rank_feature和rank_features。这些可用于基于数值或分类值来增强文档评分，同时仍保持新的快速热门命中查询（ fast top hits query）功能的性能。有关这些字段以及如何使用它们的更多信息，请阅读我们的博客文章。

17.JSON logging（json 日志）

现在，除了普通文本日志之外，还在Elasticsearch中启用了JSON日志记录。从7.0开始，您将在日志目录中找到扩展名为.json的新文件。这意味着您现在可以使用jq等过滤工具以更加结构化的方式打印和处理日志。您还可以期望在每个日志行中查找node.id，cluster.uuid，type（和更多）等其他信息。每个JSON日志行的类型字段将允许您在docker上运行时区分日志流。

18.Script score query (aka function score 2.0) （评分脚本查询（又名 评分函数2.0））

在7.0中，我们将介绍下一代评分函数功能。这个新的script_score查询提供了一种新的，更简单，更灵活的方法来生成每条记录的排名分数。 script_score查询由一组函数构成，包括算术和距离函数，用户可以混合和匹配这些函数来构造任意函数得分计算。模块化结构更易于使用，并将为其他用户打开这一重要功能。

二、Breaking changes(重要变更)

> Breaking changes ,是一些查询api、集群、聚合、分词、提词、插件，rest api 等一些细节用法上的调整变动，有些功能平时也没用到，变更比较多，可以选择性关注，了解知道即可，我这里挑选部分变更进行翻译。

1.Aggregations changes

- **search.max_buckets设置 ：** 名为search.max_buckets的动态集群设置现在默认为10,000（而不是以前版本中的无限制）。尝试返回超过限制的请求将失败并出现异常。

- **已删除复合聚合的missing选项** ： 已删除在6.x中弃用的复合聚合的missing选项。应该使用missing_bucket代替。

2. Analysis changes

- **限制由_analyze生成的短语数量**： 为了防止内存不足错误，可以使用_analyze端点生成的短语数量限制为10000.可以使用索引的setting：index.analyze.max_token_count更改特定索引的此默认限制。
- **限制高亮处理时的解析的文本长度 ：** 高亮显示索引的没有偏移量的文本，需要在搜索请求期间实时在内存中分析此文本。对于大文本，这种分析可能需要大量的时间和记忆。为了防止这种情况，将分析的最大字符数限制为1000000.可以使用索引setting：index.highlight.max_analyzed_offset为特定索引更改此默认限制。
- **标准过滤器已被删除 ：** 标准分词过滤器已被删除，因为它不会更改分词流中的任何内容。
- **不推荐使用standard_html_strip分析器 ：** 不推荐使用standard_html_strip分析器，应将其替换为标准tokenizer和html_strip char_filter的组合。使用此分析器创建的索引仍可在elasticsearch 7.0中读取，但无法使用它创建新索引

3.Cluster changes

- **群集名称中不再允许冒号(:)** ：由于跨群集搜索使用":",分隔集群和索引名称，集群名称可能不再包含“：”。
- **群集范围的分片软限制 ：** 现在，群集基于节点数和cluster.max_shards_per_node群集设置对群集中打开的分片总数进行了软限制，以防止意外操作破坏群集的稳定性。可以在该设置的文档中找到更多信息。

4.Discovery changes

- **如果配置了集群发现，则需要群集引导 ：** 第一次启动集群时，必须将cluster.initial_master_nodes设置为执行集群引导。它应包含初始集群中符合主节点资格的节点的名称，并在集群中的每个符合主节点的节点上定义。有关示例，请参阅发现设置摘要，集群引导参考文档更详细地介绍了此设置。在7.x节点上允许但忽略discovery.zen.minimum_master_nodes设置
- **生产环境的集群发现配置是必须的 ：** 现在，Elasticsearch的生产部署至少需要在elasticsearch.yml配置文件中指定以下设置之一：
  - discovery.seed_hosts
  - discovery.seed_providers
  - cluster.initial_master_nodes
  - discovery.zen.ping.unicast.hosts
  - discovery.zen.hosts_provider

  此列表中的前三个设置仅适用于7.0及更高版本。如果您准备从早期版本升级，则必须设置discovery.zen.ping.unicast.hosts或discovery.zen.hosts_provider。
- **减少故障检测的默认超时 ：** 默认情况下，群集故障检测子系统现在认为节点有故障，如果它无法响应3次连续ping，每次ping都会在10秒后超时。因此，长时间超过30秒的无响应节点可能会从群集中删除。以前，每次ping的默认超时为30秒，因此无响应的节点可能会在群集中保留超过90秒。

5.Indices changes

- **索引创建不再默认为五个分片 ：** 以前版本的Elasticsearch默认为每个索引创建五个分片。从7.0.0开始，默认为每个索引一个分片。
- **索引名字不在包含冒号：** 由于跨群集搜索使用：分隔群集和索引名称，索引名称可能不再包含：。
- **文档分发更改 ：** 使用7.0.0及更高版本创建的索引将具有自动index.number_of_routing_shards值集。这可能会更改文档在分片中的分布方式，具体取决于索引具有多少分片。为了保持与7.0.0之前的索引完全相同的分布，必须在索引创建时将index.number_of_routing_shards设置为index.number_of_shards。注意：如果路由分片数等于分片数_split操作不受支持。
- **搜索空闲分片，跳过的后台刷新 ：** **（第一部分解释过）**

6.Mapping changes

- **元字段：*_all*、*_uid*被删除了 ：** uid用于索引由_type和_id组成的复合键。现在索引不能有多种类型，这已被删除，有利于_id。
- **限制嵌套(nested) json对象的数量** ： 为了防止内存不足错误，所有字段中单个文档中嵌套的json对象的数量已限制为10000.可以使用索引设置index.mapping.nested_objects.limit更改此默认限制。
- **不推荐使用geo_shape参数** ： 对于geo_shape字段类型，不推荐使用以下类型参数：tree，precision，tree_levels，distance_error_pct，points_only和strategy。它们将在未来版本中删除。

7.Search and Query DSL changes

- **搜索API为无效请求返回400** ： 在以下无效请求的情况下，Search API返回400 - Bad请求，而之前返回500
- **滚动查询设置**：request_cache:true 已在6.*中弃用，现在将返回400 - Bad请求。滚动查询不是要缓存的。
- **Term Suggesters支持距离算法** ： 以下字符串距离算法在6.2中给出了其他名称，并且不推荐使用它们的现有名称。已弃用的名称现已删除。
  - levenstein - 替换为 levenshtein
  - jarowinkler - 替换为 jaro_winkler
- **限制terms查询请求中可以使用的term数量** ：使用大量terms执行terms查询可能会降低群集性能，因为每个附加term都需要额外的处理和内存。为了防止这种情况，可以在terms查询请求中使用的最大term数限制为65536.可以使用索引setting : index.max_terms_count更改特定索引的此默认最大值。
- **限制可以在Regexp Query请求中使用的正则表达式的长度 ：** 使用长正则表达式字符串执行Regexp查询可能会降低搜索性能。为了防止这种情况，可以在Regexp查询请求中使用的正则表达式的最大长度限制为1000.可以使用索引设置index.max_regex_length更改特定索引的此默认最大值。

- **限制自动扩展字段的数量 ：** 执行使用字段自动扩展的查询（例如query_string，simple_query_string或multi_match）可能会对具有大量字段的索引产生性能问题。为了防止这种情况，已经为使用“所有字段”模式（“default_field”：“”）或其他字段名扩展（例如“foo”）的查询引入了1024个字段的硬限制。
- **max_concurrent_shard_requests的语义已更改 ：** max_concurrent_shard_requests用于限制单个高级别搜索请求可以执行的并发分片请求的总数。在7.0中，这更改为每个节点的最大并发分片请求数。默认值现在为5。
- **不允许负向提升评分 ：** 在此版本中不允许为查询或字段设置负数提升评分，6.x已经不建议使用。要降低特定查询或字段的评分，您可以使用系数0到1之间的boost。
- **hits.total现在是搜索返回中的一个对象** ：现在，与搜索请求匹配的总命中数将作为具有值和关系的对象返回。 value表示匹配的匹配数，关系表示值是准确的（eq）还是下限（gte）：

```
{
    "hits": {
        "total": {
            "value": 1000,
            "relation": "eq"
        },
        ...
    }
}
```

响应中的总对象表示查询恰好匹配1000个文档（“eq”）。当track_total_hits在请求中设置为true时，该值始终准确（“relation”：“eq”）。您还可以通过在搜索请求的请求参数中添加rest_total_hits_as_int = true来检索hits.total作为其余响应中的数字。添加此参数是为了简化向新格式的转换，将在下一个主要版本（8.0）中删除。