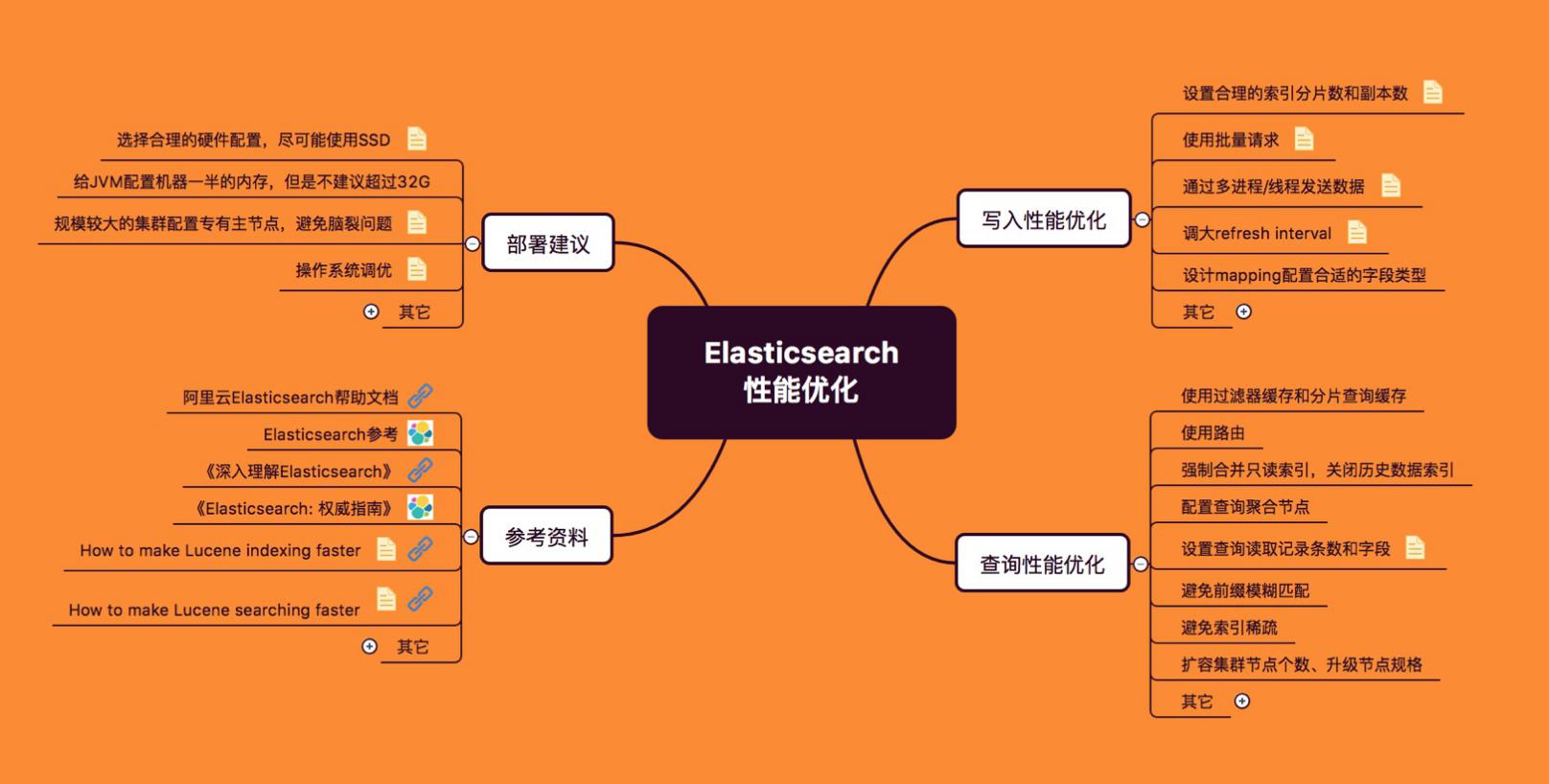
# 阿里云Elasticsearch性能优化实践

阿里巴巴搜索事业部 欧阳楚才

Elasticsearch是一款流行的分布式开源搜索和数据分析引擎，具备高性能、易扩展、容错性强等特点。它强化了Apache Lucene的搜索能力，把掌控海量数据索引和查询的方式提升到一个新的层次。本文结合开源社区和阿里云平台的实践经验，探讨如何调优Elasticsearch的性能提高索引和查询吞吐量。



* **一. Elasticsearch部署建议**

1. 选择合理的硬件配置，尽可能使用SSD

Elasticsearch最大的瓶颈往往是磁盘读写性能，尤其是随机读取性能。使用SSD（PCI-E接口SSD卡/SATA接口SSD盘）通常比机械硬盘（SATA盘/SAS盘）查询速度快5~10倍，写入性能提升不明显。

对于文档检索类查询性能要求较高的场景，建议考虑SSD作为存储，同时按照1:10的比例配置内存和硬盘。对于日志分析类查询并发要求较低的场景，可以考虑采用机械硬盘作为存储，同时按照1:50的比例配置内存和硬盘。单节点存储数据建议在2TB以内，最大不要超过5TB，避免查询速度慢、系统不稳定。

在单机存储1TB数据场景下，SATA盘和SSD盘的检索性能对比（测试环境：Elasticsearch5.5.3，10亿条人口户籍登记信息，单机16核CPU、64GB内存，12块6TB SATA盘，2块1.5 TB SSD盘）：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **磁盘类型** | **并发数** | **QPS** | **平均检索响应时间** | **50%请求响应时间** | **90%请求响应时间** | **IOPS** |
| SATA盘 | 10并发 | 17 | 563ms | 478ms | 994ms | 1200 |
| SATA盘 | 50并发 | 64 | 773ms | 711ms | 1155ms | 1800 |
| SATA盘 | 100并发 | 110 | 902ms | 841ms | 1225ms | 2040 |
| SATA盘 | 200并发 | 84 | 2369ms | 2335ms | 2909ms | 2400 |
| SSD盘 | 10并发 | 94 | 105ms | 90ms | 200ms | 25400 |
| SSD盘 | 50并发 | 144 | 346ms | 341ms | 411ms | 66000 |
| SSD盘 | 100并发 | 152 | 654ms | 689ms | 791ms | 60000 |
| SSD盘 | 200并发 | 210 | 950ms | 1179ms | 1369ms | 60000 |

1. 给JVM配置机器一半的内存，但是不建议超过32G

修改conf/jvm.options配置，-Xms和-Xmx设置为相同的值，推荐设置为机器内存的一半左右，剩余一半留给操作系统缓存使用。jvm内存建议不要低于2G，否则有可能因为内存不足导致ES无法正常启动或内存溢出，jvm建议不要超过32G，否则jvm会禁用内存对象指针压缩技术，造成内存浪费。机器内存大于64G内存时，推荐配置-Xms30g -Xmx30g 。

1. 规模较大的集群配置专有主节点，避免脑裂问题

Elasticsearch主节点（master节点）负责集群元信息管理、index的增删操作、节点的加入剔除，定期将最新的集群状态广播至各个节点。在集群规模较大时，建议配置专有主节点只负责集群管理，不存储数据，不承担数据读写压力。

# 专有主节点配置(conf/elasticsearch.yml)：

node.master:true

node.data: false

node.ingest:false

# 数据节点配置(conf/elasticsearch.yml)：

node.master:false

node.data:true

node.ingest:true

Elasticsearch默认每个节点既是候选主节点，又是数据节点。最小主节点数量参数minimum\_master\_nodes推荐配置为候选主节点数量一半以上，该配置告诉Elasticsearch当没有足够的master候选节点的时候，不进行master节点选举，等master节点足够了才进行选举。

例如对于3节点集群，最小主节点数量从默认值1改为2。

# 最小主节点数量配置(conf/elasticsearch.yml)：

discovery.zen.minimum\_master\_nodes: 2

1. Linux操作系统调优

关闭交换分区，防止内存置换降低性能。 将/etc/fstab 文件中包含swap的行注释掉  
 sed -i '/swap/s/^/#/' /etc/fstab

swapoff -a

单用户可以打开的最大文件数量，可以设置为官方推荐的65536或更大些  
 echo "\* - nofile 655360" >> /etc/security/limits.conf

单用户线程数调大  
echo "\* - nproc 131072" >> /etc/security/limits.conf

单进程可以使用的最大map内存区域数量  
echo "vm.max\_map\_count = 655360" >> /etc/sysctl.conf

参数修改立即生效  
sysctl -p

* **二. 索引性能调优建议**

1. 设置合理的索引分片数和副本数

索引分片数建议设置为集群节点的整数倍，初始数据导入时副本数设置为0，生产环境副本数建议设置为1（设置1个副本，集群任意1个节点宕机数据不会丢失；设置更多副本会占用更多存储空间，操作系统缓存命中率会下降，检索性能不一定提升）。单节点索引分片数建议不要超过3个，每个索引分片推荐10-40GB大小。索引分片数设置后不可以修改，副本数设置后可以修改。Elasticsearch6.X及之前的版本默认索引分片数为5、副本数为1，从Elasticsearch7.0开始调整为默认索引分片数为1、副本数为1。

不同分片数对写入性能的影响（测试环境：7节点Elasticsearch6.3集群，写入30G新闻数据，单节点56核CPU、380G内存、3TB SSD卡，0副本，20线程写入，每批次提交10M左右数据。）：

|  |  |  |
| --- | --- | --- |
| **集群索引分片数** | **单节点索引分片数** | **写入耗时** |
| 2 | 0/1 | 600s |
| 7 | 1 | 327s |
| 14 | 2 | 258s |
| 21 | 3 | 211s |
| 28 | 4 | 211s |
| 56 | 8 | 214s |

索引设置：

curl -XPUT http://localhost:9200/fulltext001?pretty -H 'Content-Type: application/json' -d '

{

"settings" : {

"refresh\_interval": "30s",

"merge.policy.max\_merged\_segment": "1000mb",

"translog.durability": "async",

"translog.flush\_threshold\_size": "2gb",

"translog.sync\_interval": "100s",

"index" : {

"number\_of\_shards" : "21",

"number\_of\_replicas" : "0"

}

}

}

'

mapping设置：

curl -XPOST http://localhost:9200/fulltext001/doc/\_mapping?pretty -H 'Content-Type: application/json' -d '

{

"doc" : {

"\_all" : {

"enabled" : false

},

"properties" : {

"content" : {

"type" : "text",

"analyzer":"ik\_max\_word"

},

"id" : {

"type" : "keyword"

}

}

}

}

'

写入数据示例：

curl -XPUT 'http://localhost:9200/fulltext001/doc/1?pretty' -H 'Content-Type: application/json' -d '

{

"id": "https://www.huxiu.com/article/215169.html",

"content": "“娃娃机，迷你KTV，VR体验馆，堪称商场三大标配‘神器’。”一家地处商业中心的大型综合体负责人告诉懂懂笔记，在过去的这几个月里，几乎所有的综合体都“标配”了这三种“设备”…"

}'

        修改副本数示例：

curl -XPUT "http://localhost:9200/fulltext001/\_settings" -H 'Content-Type: application/json' -d'

{

"number\_of\_replicas": 1

}'

1. 使用批量请求

使用批量请求将产生比单文档索引请求好得多的性能。写入数据时调用批量提交接口，推荐每批量提交5~15MB数据。例如单条记录1KB大小，每批次提交10000条左右记录写入性能较优；单条记录5KB大小，每批次提交2000条左右记录写入性能较优。

批量请求接口API：

curl -XPOST "http://localhost:9200/\_bulk" -H 'Content-Type: application/json' -d'

{ "index" : { "\_index" : "test", "\_type" : "\_doc", "\_id" : "1" } }

{ "field1" : "value1" }

{ "delete" : { "\_index" : "test", "\_type" : "\_doc", "\_id" : "2" } }

{ "create" : { "\_index" : "test", "\_type" : "\_doc", "\_id" : "3" } }

{ "field1" : "value3" }

{ "update" : {"\_id" : "1", "\_type" : "\_doc", "\_index" : "test"} }

{ "doc" : {"field2" : "value2"} }

'

1. 通过多进程/线程发送数据

单线程批量写入数据往往不能充分利用服务器CPU资源，可以尝试调整写入线程数或者在多个客户端上同时向Elasticsearch服务器提交写入请求。与批量调整大小请求类似，只有测试才能确定最佳的worker数量。 可以通过逐渐增加工作任务数量来测试，直到集群上的I / O或CPU饱和。

1. 调大refresh interval

在 Elasticsearch 中，写入和打开一个新段的轻量的过程叫做 *refresh* 。 默认情况下每个分片会每秒自动刷新一次。这就是为什么我们说 Elasticsearch 是 *近* 实时搜索: 文档的变化并不是立即对搜索可见，但会在一秒之内变为可见。  
 并不是所有的情况都需要每秒刷新。可能你正在使用 Elasticsearch 索引大量的日志文件，你可能想优化索引速度而不是近实时搜索，可以通过设置 refresh\_interval，降低每个索引的刷新频率。

设置refresh interval API:

curl -XPUT "http://localhost:9200/index" -H 'Content-Type: application/json' -d'

{

"settings" : {

"refresh\_interval": "30s"

}

}'

refresh\_interval 可以在既存索引上进行动态更新。 在生产环境中，当你正在建立一个大的新索引时，可以先关闭自动刷新，待开始使用该索引时，再把它们调回来：

curl -XPUT "http://localhost:9200/index/\_settings" -H 'Content-Type: application/json' -d'

{ "refresh\_interval": -1 }'

curl -XPUT "http://localhost:9200/index/\_settings" -H 'Content-Type: application/json' -d'

{ "refresh\_interval": "1s" }'

1. 设计mapping配置合适的字段类型

Elasticsearch在写入文档时，如果请求中指定的索引名不存在，会自动创建新索引，并根据文档内容猜测可能的字段类型。但这往往不是最高效的，我们可以根据应用场景来设计合理的字段类型。

例如写入一条记录：

curl -XPUT "http://localhost:9200/twitter/doc/1?pretty" -H 'Content-Type: application/json' -d'

{

"user": "kimchy",

"post\_date": "2009-11-15T13:12:00",

"message": "Trying out Elasticsearch, so far so good?"

}'

查询Elasticsearch自动创建的索引mapping，会发现将post\_date字段自动识别为date类型，但是message和user字段被设置为text、keyword冗余字段，造成写入速度降低、占用更多磁盘空间。

{

"twitter": {

"mappings": {

"doc": {

"properties": {

"message": {

"type": "text",

"fields": {

"keyword": {

"type": "keyword",

"ignore\_above": 256

}

}

},

"post\_date": {

"type": "date"

},

"user": {

"type": "text",

"fields": {

"keyword": {

"type": "keyword",

"ignore\_above": 256

}

}

}

}

}

},

"settings": {

"index": {

"number\_of\_shards": "5",

"number\_of\_replicas": "1",

}

}

}

}

根据业务场景设计索引配置合理的分片数、副本数，设置字段类型、分词器。如果不需要合并全部字段，禁用\_all字段，通过copy\_to来合并字段。

curl -XPUT "http://localhost:9200/twitter?pretty" -H 'Content-Type: application/json' -d'

{

"settings" : {

"index" : {

"number\_of\_shards" : "20",

"number\_of\_replicas" : "0"

}

}

}'

curl -XPOST "http://localhost:9200/twitter/doc/\_mapping?pretty" -H 'Content-Type: application/json' -d'

{

"doc" : {

"\_all" : {

"enabled" : false

},

"properties" : {

"user" : {

"type" : "keyword"

},

"post\_date" : {

"type" : "date"

},

"message" : {

"type" : "text",

"analyzer" : "cjk"

}

}

}

}'

* **三. 查询性能调优建议**

1. 使用过滤器缓存和分片查询缓存

默认情况下，Elasticsearch的查询会计算返回的每条数据与查询语句的相关度，但对于非全文索引的使用场景，用户并不关心查询结果与查询条件的相关度，只是想精确的查找目标数据。此时，可以通过filter来让Elasticsearch不计算评分，并且尽可能的缓存filter的结果集，供后续包含相同filter的查询使用，提高查询效率。

普通查询：

curl -XGET "http://localhost:9200/twitter/\_search" -H 'Content-Type: application/json' -d'

{

"query": {

"match": {

"user": "kimchy"

}

}

}'

过滤器(filter)查询：

curl -XGET "http://localhost:9200/twitter/\_search" -H 'Content-Type: application/json' -d'

{

"query": {

"bool": {

"filter": {

"match": {

"user": "kimchy"

}

}

}

}

}'

分片查询缓存的目的是缓存聚合、提示词结果和命中数（它不会缓存返回的文档，因此，它只在search\_type=count时起作用）。

通过下面的参数我们可以设置分片缓存的大小，默认情况下是JVM堆的1%大小，当然我们也可以手动设置在config/elasticsearch.yml文件里:

indices.requests.cache.size: 1%

1. 使用路由routing

Elasticsearch写入文档时，文档会通过一个公式路由到一个索引中的一个分片上。默认的公式如下：

shard\_num = hash(\_routing) % num\_primary\_shards

\_routing字段的取值，默认是\_id字段，可以根据业务场景设置经常查询的字段作为路由字段。例如可以考虑将用户id、地区作为路由字段，查询时可以过滤不必要的分片，加快查询速度。

写入时指定路由：

curl -XPUT "http://localhost:9200/my\_index/my\_type/1?routing=user1" -H 'Content-Type: application/json' -d'

{

"title": "This is a document",

"author": "user1"

}'

查询时不指定路由，需要查询所有分片：

curl -XGET "http://localhost:9200/my\_index/\_search" -H 'Content-Type: application/json' -d'

{

"query": {

"match": {

"title": "document"

}

}

}'

返回结果：

{

"took": 2,

"timed\_out": false,

"\_shards": {

"total": **5**,

"successful": **5**,

"skipped": 0,

"failed": 0

}

......

}

查询时指定路由，只需要查询1个分片：

curl -XGET "http://localhost:9200/my\_index/\_search?routing=user1" -H 'Content-Type: application/json' -d'

{

"query": {

"match": {

"title": "document"

}

}

}'

返回结果：

{

"took": 1,

"timed\_out": false,

"\_shards": {

"total": 1,

"successful": 1,

"skipped": 0,

"failed": 0

}

......

}

1. 强制合并只读索引，关闭历史数据索引

只读的索引可以从合并成一个单独的大segment中收益，减少索引碎片，减少JVM堆常驻内存。例如某公安行业客户每天建立一个索引写入2TB数据，72个节点的集群数据量达到PB级，集群查询速度较慢、运行不稳定，通过对历史数据forcemerge、关闭1年前的索引后集群恢复正常。

索引forcemerge API:

curl -XPOST "http://localhost:9200/wechat20180923/\_forcemerge"

索引关闭API:

curl -XPOST "http://localhost:9200/wechat2017\*/\_close"

1. 配置查询聚合节点

查询聚合节点可以发送粒子查询请求到其他节点，收集和合并结果，以及响应发出查询的客户端。通过给查询聚合节点配置更高规格的CPU和内存，可以加快查询运算速度、提升缓存命中率。某保险行业客户自建ELasticsearch集群，使用25台8核CPU32G内存节点，查询QPS在4000左右。通过增加6台16核CPU32G内存节点，观察服务器CPU、JVM堆内存使用情况，并调整缓存、分片、副本参数，查询QPS达到12000。

# 查询聚合节点配置(conf/elasticsearch.yml)：

node.master: false

node.data: false

node.ingest:false

1. 设置查询读取记录条数和字段

默认的查询请求通常返回排序后的前10条记录，最多一次读取10000条记录，通过from和size参数控制读取记录范围，避免一次读取过多的记录。通过\_source参数可以控制返回字段信息，尽量避免读取大字段。

查询请求示例：

curl -XGET http://localhost:9200/fulltext001/\_search?pretty -H 'Content-Type: application/json' -d '

{

"from": 0,

"size": 10,

"\_source": "id",

"query": {

"bool": {

"must": [

{"match": {"content":"虎嗅"}}

]

}

},

"sort": [

{

"id": {

"order": "asc"

}

}

]

}

'

1. 避免前缀模糊匹配

Elasticsearch默认支持通过\*?正则表达式来做模糊匹配，如果在一个数据量超过10亿条的索引上执行模糊匹配，尤其是前缀模糊匹配，通常耗时会比较长，甚至可能导致内存溢出。尽量避免在高并发查询请求的生产环境执行这类操作。

某交通行业客户需要对车牌号进行模糊查询，通过查询请求"**车牌号:\*A8848\*"**查询时，往往导致整个集群负载较高。通过对数据预处理，增加冗余字段"车牌号.keyword"，并事先将所有车牌号按照1元、2元、3元...7元分词后存储至该6字段，字段存储内容示例:沪,A,8,4,沪A,A8,88,84,48,沪A8...沪A88488。通过查询"**车牌号.keyword:A8848"**即可解决原来的性能问题。

1. 避免索引稀疏

Elasticsearch6.X之前的版本默认允许在一个index下面创建多个type，Elasticsearch6.X及之后的版本只允许创建一个type。在一个type下面创建多个字段不一样的type，或者将几百个字段不一样的索引合并到一个索引中，会导致索引稀疏问题。

建议每个索引下只创建一个type，字段不一样的数据分别独立创建index，不要合并成一个大索引。每个查询请求根据需要去读取相应的索引，避免查询大索引扫描全部记录，加快查询速度。

1. 扩容集群节点个数、升级节点规格

通常服务器节点数越多，服务器硬件配置规格越高，Elasticsearch集群的处理能力越强。

在不同节点规模下的查询性能测试（测试环境：Elasticsearch5.5.3集群，单节点16核CPU、64G内存、2T SSD盘，10亿条人口户籍数据，数据大小1TB。）：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **集群节点数** | **副本数** | **10并发检索平均响应时间** | **50并发检索平均响应时间** | **100并发检索平均响应时间** | **200并发检索平均响应时间** | **200并发QPS** | **200并发CPU使用率** | **200并发CPU IO等待** |
| 1 | 0 | 77ms | 459ms | 438ms | 1001ms | 200 | 16% | 52% |
| 3 | 0 | 38ms | 103ms | 162ms | 298ms | 669 | 45% | 34% |
| 3 | 2 | 271ms | 356ms | 577ms | 818ms | 244 | 19% | 54% |
| 10 | 0 | 21ms | 36ms | 48ms | 81ms | 2467 | 40% | 10% |

不同集群节点规模写入性能测试（测试环境：Elasticsearch6.3.2集群，单节点16核CPU、64G内存、2T SSD盘，10亿条人口户籍数据，单条记录1KB，数据集大小1TB，20个并发写入线程。）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **集群节点数** | **副本数** | **写入TPS** | **耗时** | **集群CPU使用率** |
| 10 | 0 | 88945 | 11242s | 50% |
| 50 | 0 | 180638 | 5535s | 20% |

在条件允许的情况下，还是希望您可以通过实际的数据和使用场景测试出适合自己的最佳实践。得益于阿里云Elasticsearch提供的弹性扩容功能，您可以在实际使用时根据情况随时增加磁盘大小、扩容节点个数、升级节点规格。

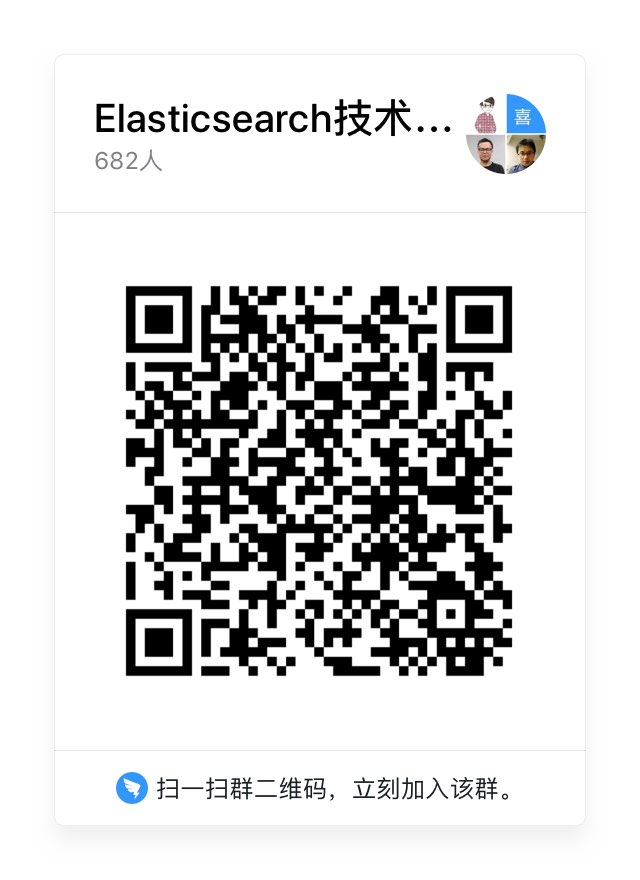
参考文档：

1. 阿里云Elasticsearch帮助文档<https://help.aliyun.com/product/57736.html>
2. Elasticsearch参考<https://www.elastic.co/guide/en/elasticsearch/reference/current/how-to.html>
3. 《深入理解Elasticsearch》<https://detail.tmall.com/item.htm?id=551001166567>
4. 《Elasticsearch: 权威指南》<https://www.elastic.co/guide/cn/elasticsearch/guide/current/indexing-performance.html>
5. How to make Lucene searching faster <https://wiki.apache.org/lucene-java/ImproveSearchingSpeed>
6. How to make Lucene indexing faster <https://wiki.apache.org/lucene-java/ImproveIndexingSpeed>

阿里ATA文档链接地址：

《阿里云Elasticsearch性能优化实践》

<https://www.atatech.org/articles/119815>



Elasticsearch技术交流钉钉群