十倍性价比蜕变:
腾讯云 ES 全新架构助力日志场景降本增效

2024/07/27

陈曦

一 综述

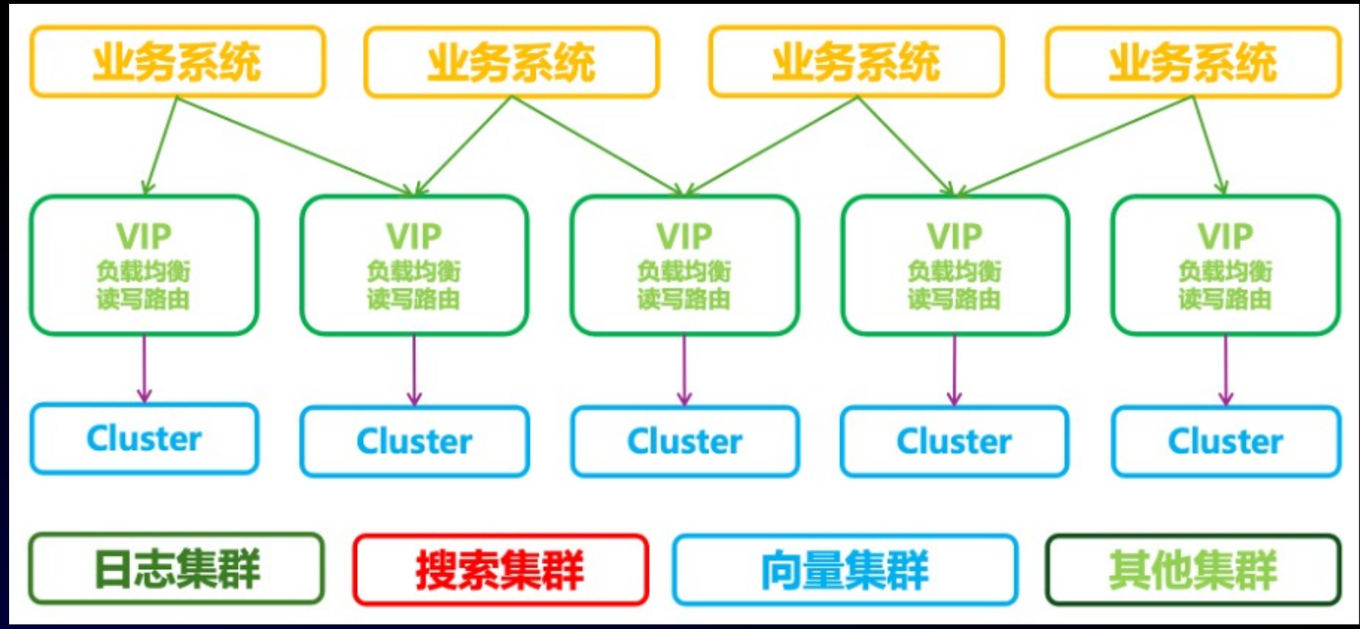
ES 技术栈一直是日志、安全、搜索场景的开源首选方案。但存储成本大、写入查询慢等成为普遍且尖锐的挑战，客户降本增效的诉求也越来越高。

本次分享主要解析腾讯云 ES 全新架构助力客户日志场景实现十倍降本效果。将围绕 **存算分离、读写分离、IO 并行化、查询裁剪** 等几个关键自研技术点进行深入分析。

二 架构演进之路

2.1 原生架构

客户一般会申请一个或多个独立的 ES 集群提供服务，此时独立集群的性能和成本在客户可接受的范围内。



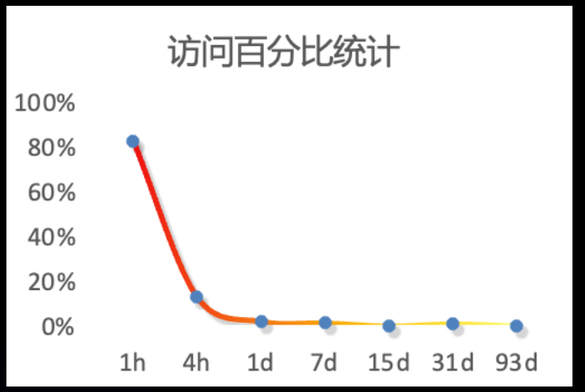
原生架构弊端

1）集群相互独立，资源利用率低。
2）副本冗余写入，冗余计算开销。
3）副本冗余存储，叠加云盘底层 3 副本，存储成本放大严重。
4）无法弹性扩缩容，数据迁移成本大。
5）存储与计算耦合，资源无法独立弹性扩缩容。
6）分片长尾效应，写入时各分片并发写入，一个慢分片拖累整体写入吞吐。

二 架构演进之路

2.2 冷热分离架构

根据日志场景的冷热特性，频繁读写的热数据放到 SSD 磁盘上，冷数据查询频率低数据量大，迁移到价格更低的 HDD 磁盘上进行降本。

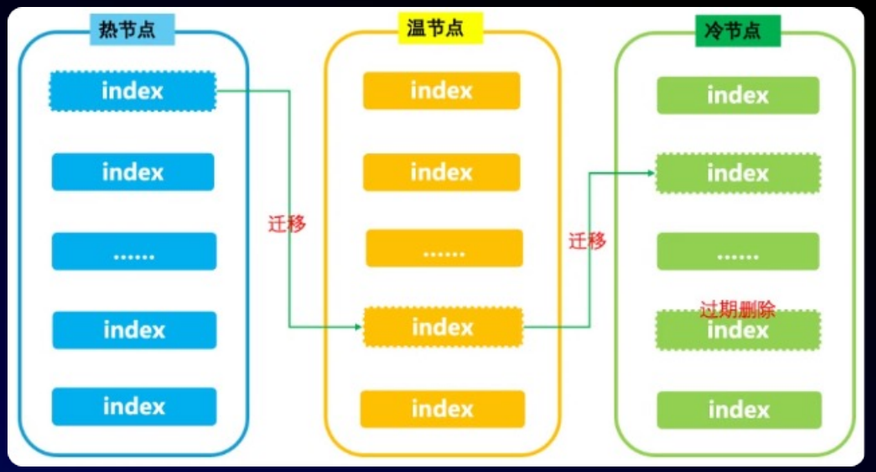


冷热分离架构的弊端：

1）原生架构的缺点都存在。

2）需要做大量的数据迁移，消耗大量资源，影响集群稳定性。

3）只有热节点提供数据的写入能力，波峰波谷的场景会存在比较大的资源浪费。



二 架构演进之路

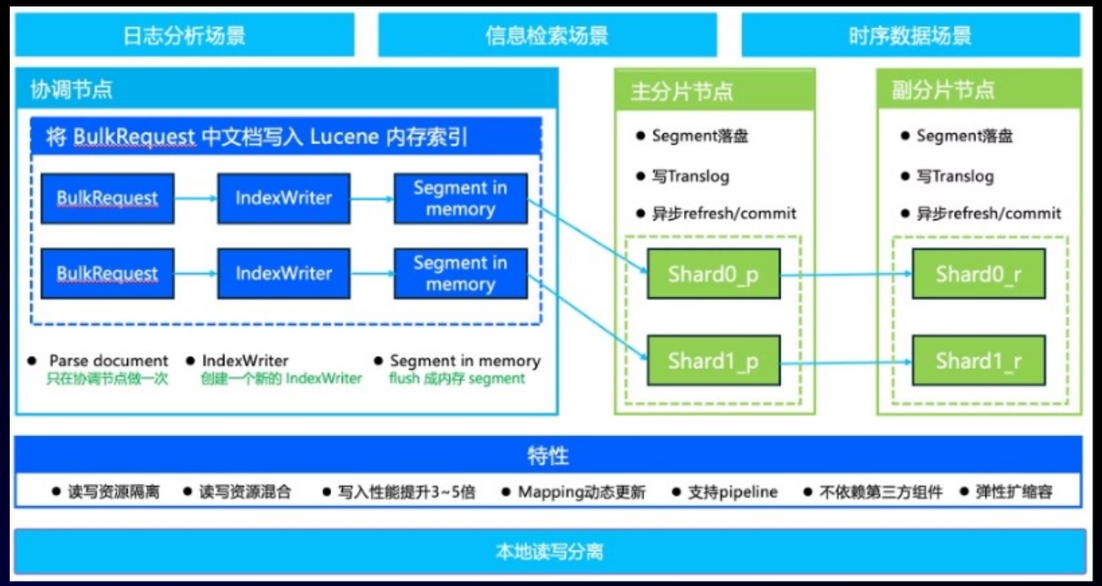
2.3 读写分离架构

在外部系统(Flink、Spark、独立 ES 集群等) 提前通过 Lucene 的 API 构建好 Segment，然后转发给具体的分片，分片收到内存 Segment 后定时追加到 Lucene 中。

读写分离分为两种架构模式

本地读写分离：

特性：本地读写资源隔离、写入性能提升3~5倍、不依赖第三方组件、弹性扩缩容。适合读写资源隔离、提升写入性能场景。



二 架构演进之路

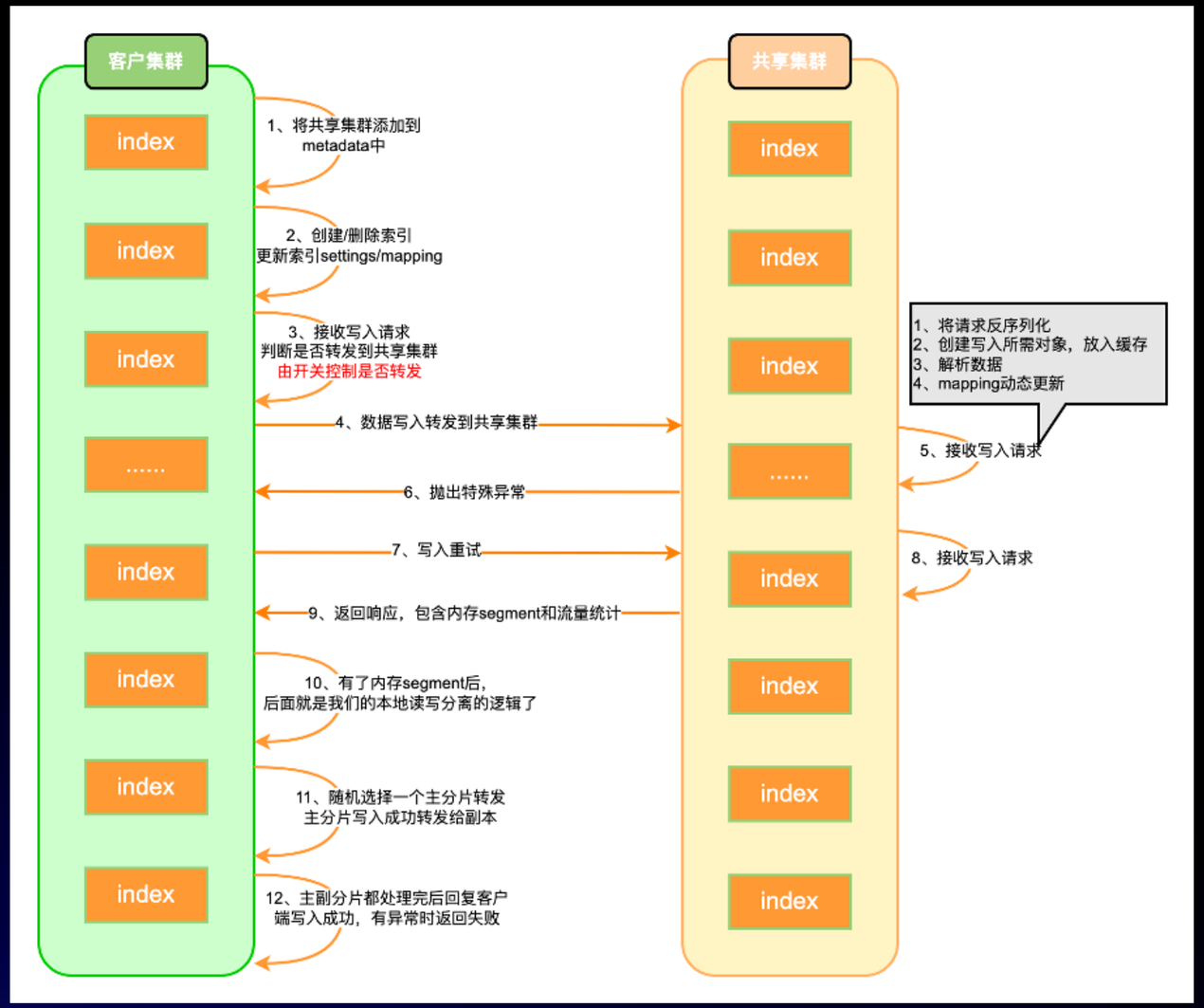
2.3 读写分离架构

共享读写分离：

特性：读写资源隔离、无状态共享资源池、写入性能提升 5~20 倍、不依赖第三方组件、弹性扩缩容。适合超高吞吐、波峰波谷、容灾切量等场景。

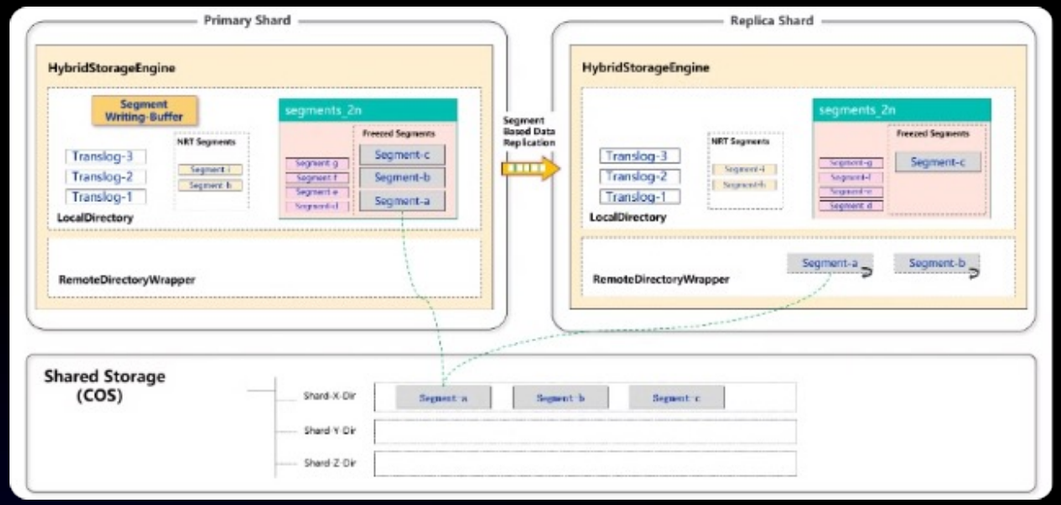
读写分离架构的弊端：

该方案解决了副本冗余写入的问题，提升了写入性能，但没有解决存储成本的问题。



二 架构演进之路

2.4 存算分离架构：实现冷热一体混合搜索能力



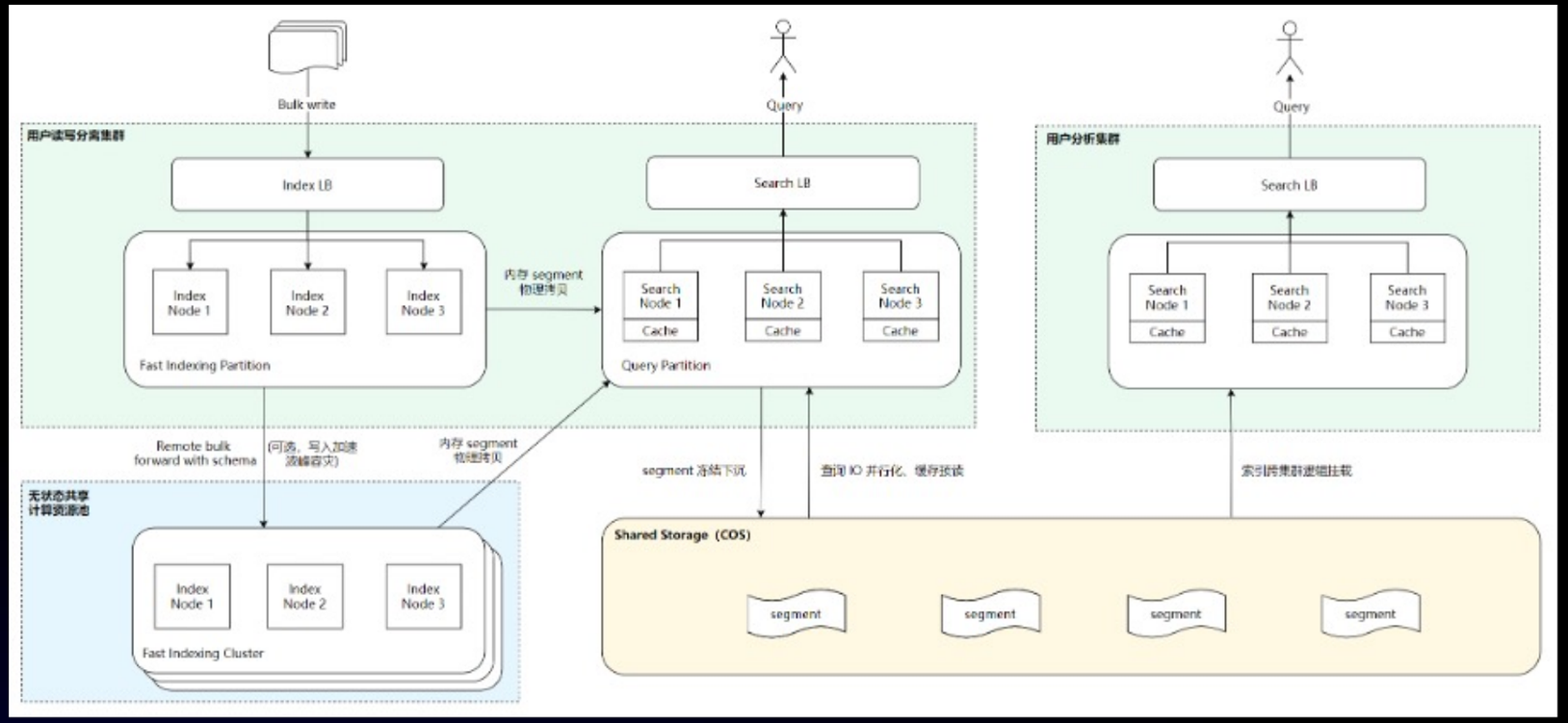
自研存算分离的核心思路：

- **物理复制**：消除副本冗余计算并确保主从副本分片完全一致
- **Delta + Base 架构：**
  - 本地 SSD ： 并发写入+承载 Merge 计算开销
  - 对象存储 ： 实现高可用，同时大幅降低存储成本
- **缓存模块：**对高频访问数据进行缓存，降低对象存储的访问频次。针对对象存储和本地磁盘访问性能差异，采用IO并行化技术结合多级缓存实现冷热一体混合搜索能力。

2.4 读写分离架构

• 读写分离、物理复制

1. 超高吞吐写入，灵活的共享计算资源池调度，以实现高性能、高可用。
2. 读写资源的物理隔离，避免大查询、突发流量的相互影响。



• 基于对象存储的存算分离架构

1. 热数据实时下沉、按需卸载，降低存储成本
2. 共享存储实现逻辑副本、弹性伸缩。
3. 索引实现跨节点、跨集群挂载，实现一份数据应对检索过滤、分析等不同的使用场景。

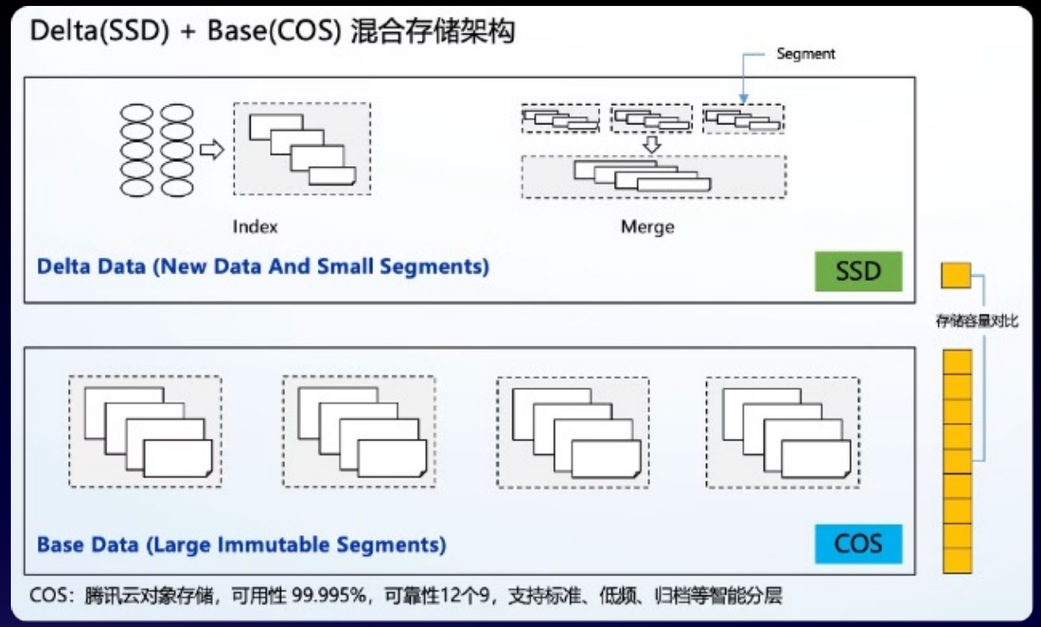
• 查询/IO并行化

1. 充分利用空闲资源达到性能倍数提升
2. 利用 IO 并行化技术弥补对象存储访问延时

• 自治索引

1. 分片、索引多级智能分区托管
2. 支持索引、分片、Segment 维度多级裁剪。

3 核心技术剖析

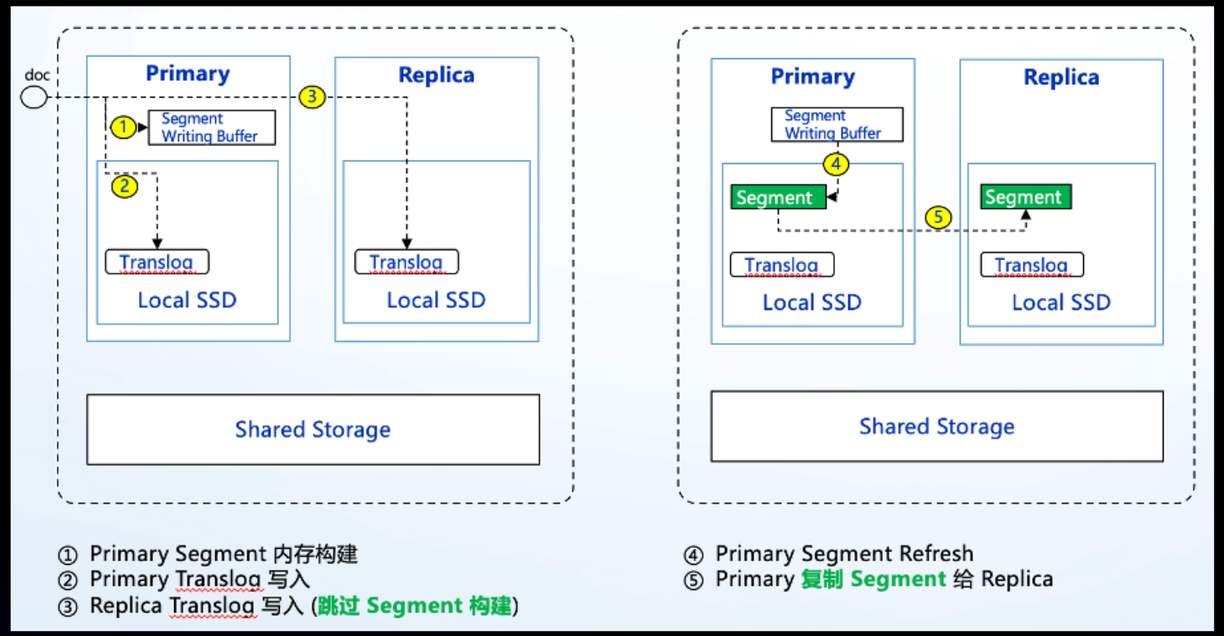

Delta(SSD) + Base(COS) 混合存储架构
Segment
Index
Merge
Delta Data (New Data And Small Segments)
SSD
存储容量对比
Base Data (Large Immutable Segments)
COS
COS：腾讯云对象存储，可用性 99.995%，可靠性12个9，支持标准、低频、归档等智能分层

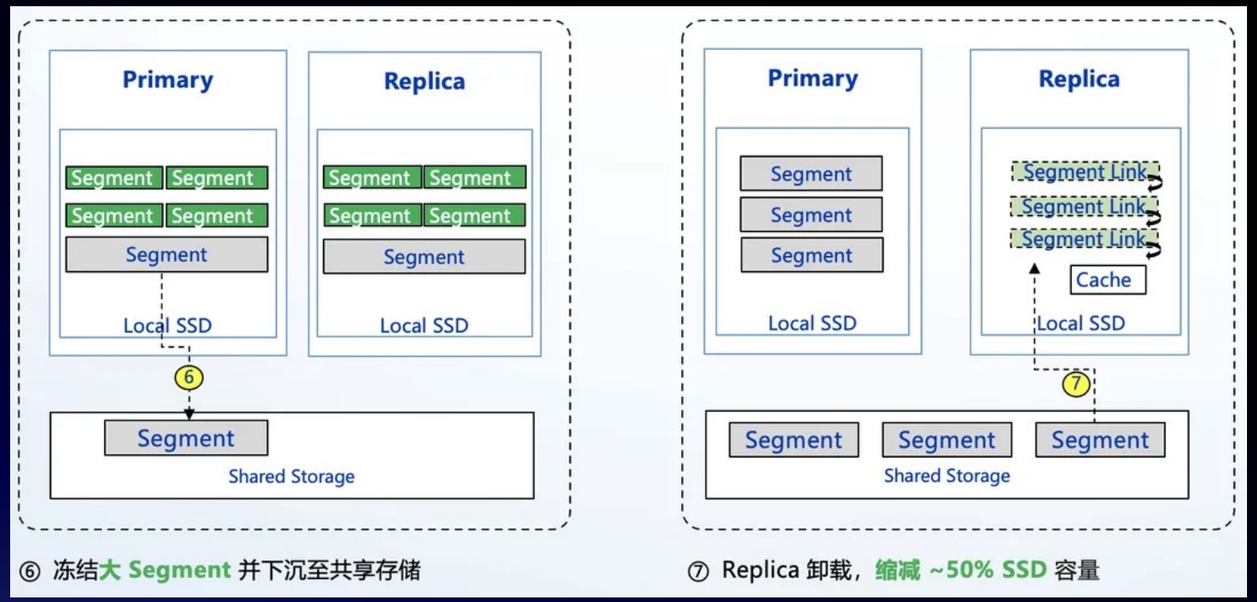
3.1 存算分离流程

前五个阶段，主要是物理复制的流程，提升写入性能，确保主从完全一致，为后续的数据共享提供基础。

第六个阶段，逐层 merge 产生了较大的 segment，下沉至底层共享存储。注意这个阶段是从热数据就开始的，避免数据降温后整体下沉的排队拥塞。此时，查询会集中在本地，本地 primary 和 replica 也能很好的抗住读写压力。

第七个阶段，数据的查询频次有所降低。一般情况，本地的 primary 即可满足绝大部分查询性能需求。此时 replica 会从本地卸载，读取会走远端共享存储，同时本地会有缓存机制保存用户常用查询数据提升性能。此时的查询会优先打到 primary。通过卸载本地 replica，我们可以缩减约 50% 的 SSD 容量。



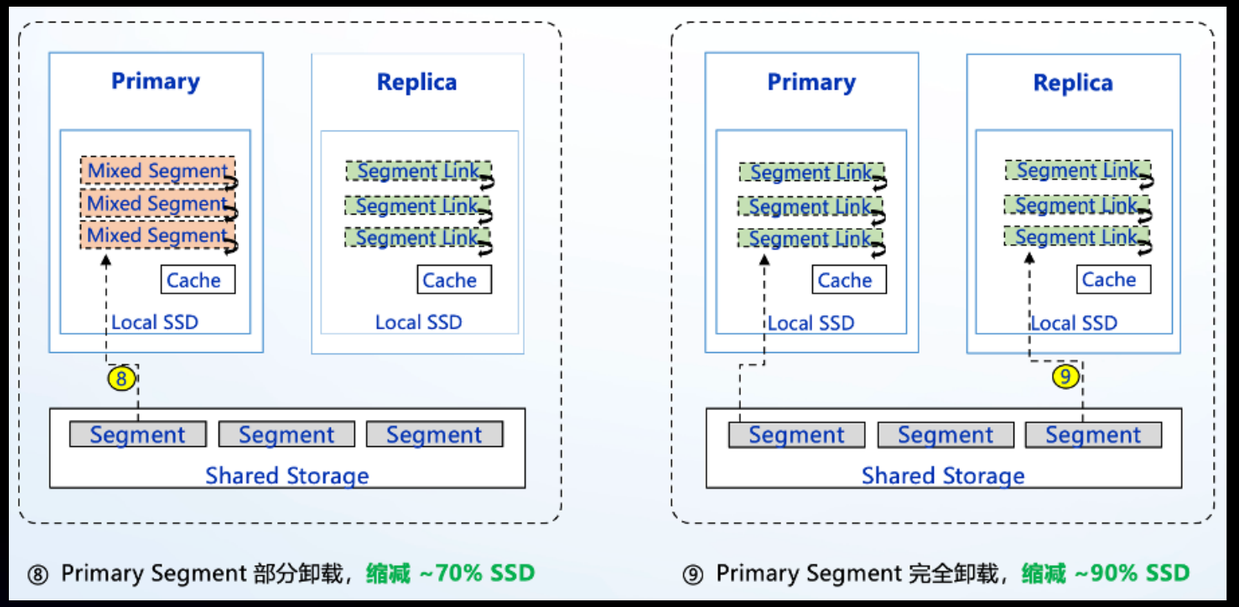


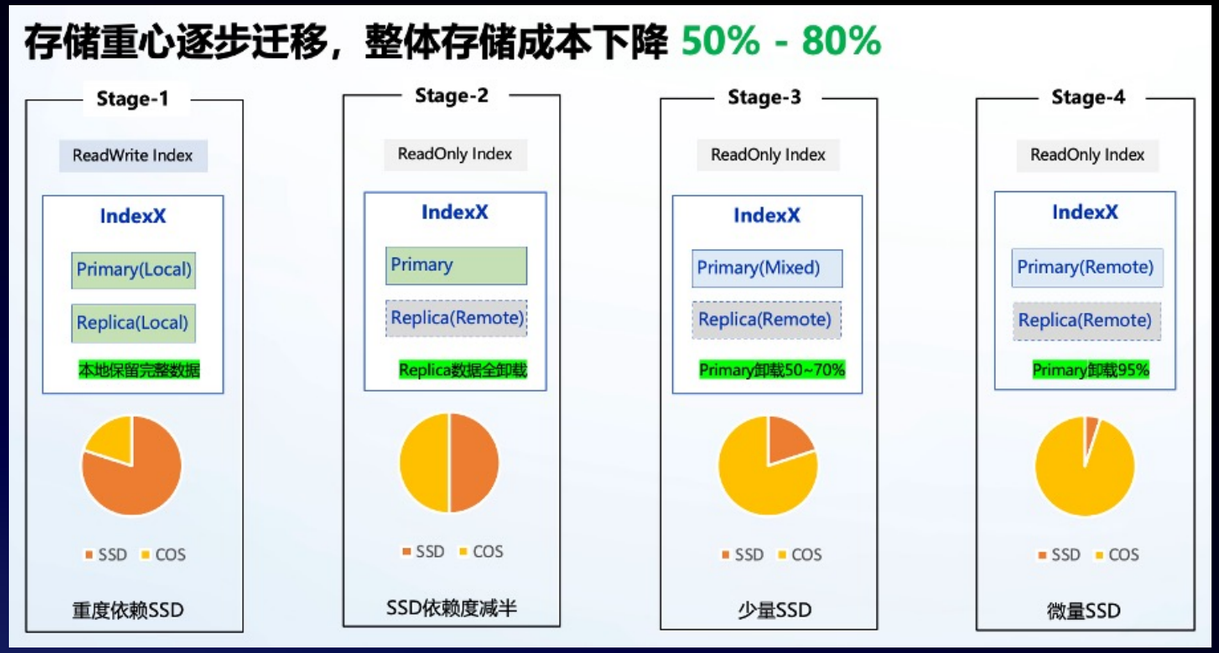
3.1存算分离流程

第八个阶段，查询频率大幅缩减。Primary 上只有部分数据或部分 Segment 需要被查询，此时 primary 上的部分文件或 Segment 会先被卸载。同时本地构建缓存体系加速查询。本阶段 SSD 的缩减达到 70% 左右，但仍然能满足业务的查询需求。

第九个阶段，查询几乎没有了，数据处于归档状态。本地的 primary 彻底实现卸载，依靠本地的缓存加速满足极少量的查询需求。本阶段 SSD 的缩减到达 90% 左右。

上面是完整的索引生命周期中数据的演变过程，存储重心随着数据逐渐降温过程逐步从 SSD 到对象存储迁移，且用户无明显感知，**最终整体存储成本下降 50% - 80%。**

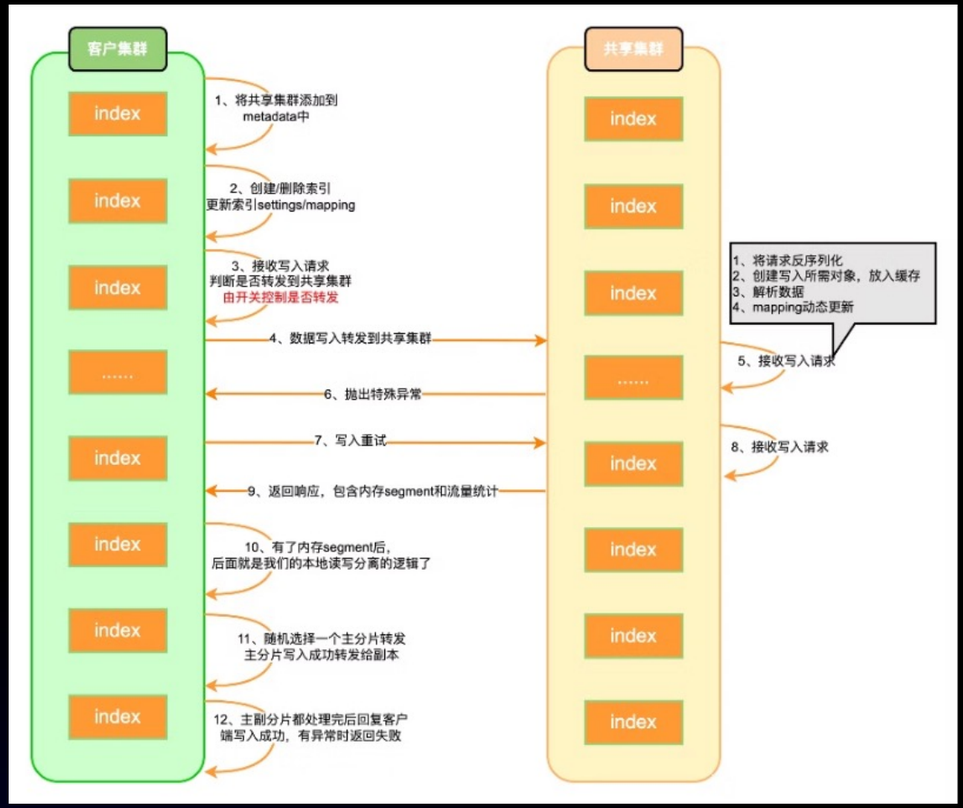




3.2 读写分离

3.2.1 设计思想

共享集群创建好 Segment 发送给 ES，ES 接收到内存 Segment 后转发到索引分片对应的数据节点中，索引分片直接追加到 Lucene 中即可。



3.2.2 赋能性能提升

1）定向路由：一批数据只发送给一个分片，每个分片单位时间内处理的数据会增多，提升吞吐量。减少 cpu 的占用。

2）数据在共享集群构建：

没有了锁的争抢，数据解析只有一次

3）内存segment：基于内存传递，共享集群不需要落盘重复读取；主分片和副本也是异步 IO 落盘，落盘即返回响应。

4）磁盘硬链接：向 lucene 中追加 segments 时使用了硬链接，避免了磁盘数据重复 copy。

	cpu 数量	单节点分片数	每分片分配核数
客户集群	32C*5	20*2/5=8	32/8=4
共享集群	32C*20	20/20=1	32/1=32

默认写入性能提升 32/4=8 倍

加上本地读写分离的特性加持，性能在此基础上还会再乘以3~5。

3.3 查询/IO并行化

1 默认查询模型

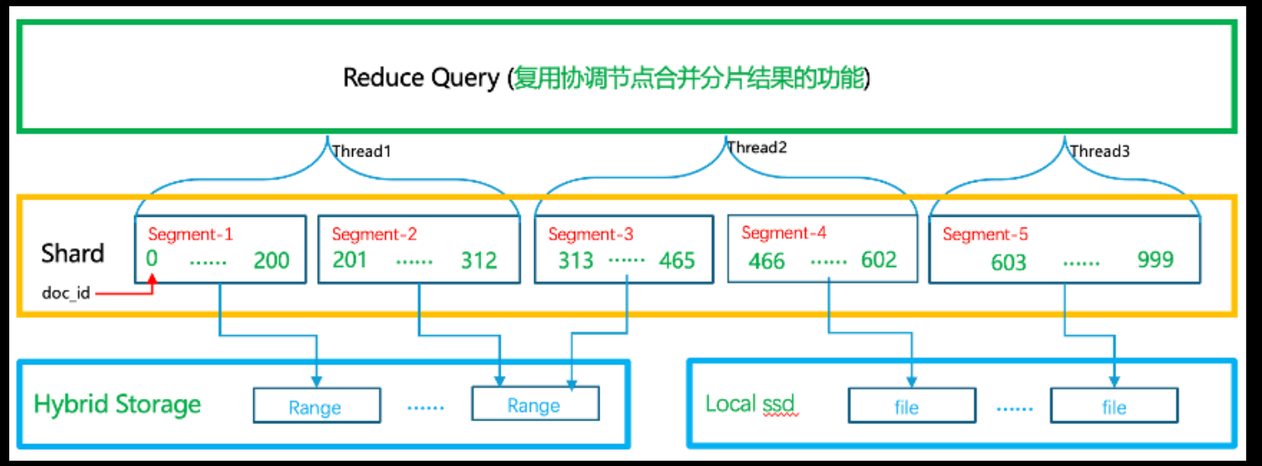
分片级查询子请求转发给各个分片执行，最后在协调节点合并各个分片的结果。在每个分片内部有多个 Segment，默认情况下 ES 执行分片级查询时是单线程串行处理每个 Segment 的。

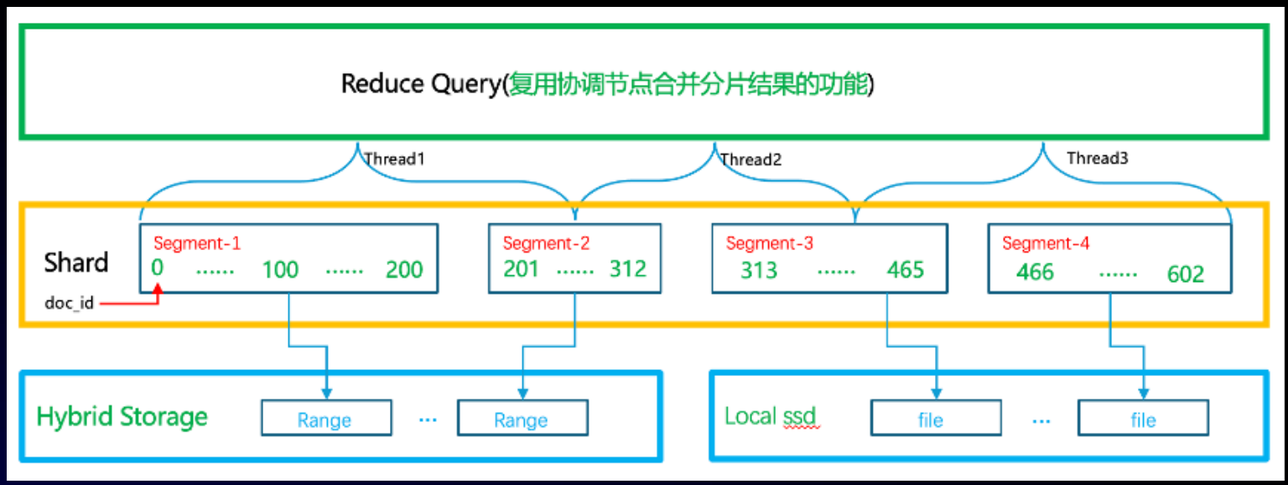
2 查询/IO并行化

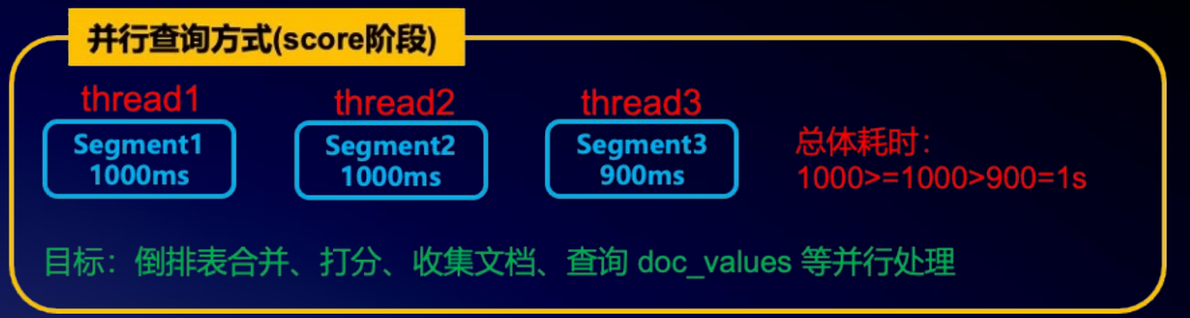
该方案意在压榨空闲 cpu 资源，将ES的单个分片级请求拆分成3~5个子请求并行处理该分片下的 Segment 或者 docs，每个线程只处理一部分 docs 或者 Segment，在数据节点合并每个线程的结果后再返回给协调节点，从而达到性能倍数级的提升。

3 性能提升

假设让3个线程并行处理这 3 个 Segment，由于是 3 个线程同时处理，因此整体耗时等于耗时最长的那个线程的执行时间，score 阶段整体耗时 =1000>=1000>900 = 1s，理论上比 2.9s 快 3 倍。







4.1 性能测评 - 存算分离

压测环境：3 台标准型 SA2 16 核 64G, 1500GB SSD 云硬盘 x 1。

压测数据：http_logs。

压测工具：esrally。

存算分离与可搜索快照查询性能对比

自研云原生存算分离架构查询性能大幅领先可搜索快照。

Min Throughput	terms_enum	22.771	49.5729	26.8019	ops/s	+117.70%
Mean Throughput	terms_enum	22.8012	49.5904	26.7892	ops/s	+117.49%
Median Throughput	terms_enum	22.8013	49.5904	26.7892	ops/s	+117.49%
Max Throughput	terms_enum	22.8312	49.6079	26.7767	ops/s	+117.28%
50th percentile latency	terms_enum	13043	4.52512	-13038.5	ms	-99.97%
90th percentile latency	terms_enum	13958.7	4.93294	-13953.8	ms	-99.96%
99th percentile latency	terms_enum	14164.1	5.40945	-14158.7	ms	-99.96%
100th percentile latency	terms_enum	14189.4	5.45118	-14183.9	ms	-99.96%
50th percentile service time	terms_enum	42.9265	3.74585	-39.1807	ms	-91.27%
90th percentile service time	terms_enum	43.5481	4.06804	-39.48	ms	-90.66%
99th percentile service time	terms_enum	45.3471	4.37935	-40.9678	ms	-90.34%
100th percentile service time	terms_enum	47.3879	4.46216	-42.9257	ms	-90.58%
error rate	terms_enum	0	0	0	%	0.00%
Min Throughput	range	11.9893	20.6366	8.64737	ops/s	+72.13%
Mean Throughput	range	13.3427	21.4274	8.08467	ops/s	+60.59%
Median Throughput	range	13.3427	21.4924	8.14962	ops/s	+61.08%
Max Throughput	range	14.6962	22.0883	7.39206	ops/s	+50.30%
50th percentile latency	range	1277.55	16.7846	-1260.76	ms	-98.69%
90th percentile latency	range	1909.13	17.7898	-1891.34	ms	-99.07%
99th percentile latency	range	2056.05	24.0536	-2032	ms	-98.83%
100th percentile latency	range	2072.16	24.6383	-2047.52	ms	-98.81%
50th percentile service time	range	23.5334	16.1196	-7.41381	ms	-31.50%
90th percentile service time	range	25.3944	16.8315	-8.56291	ms	-33.72%
99th percentile service time	range	28.1915	23.4036	-4.78789	ms	-16.98%
100th percentile service time	range	32.0388	23.7164	-8.32232	ms	-25.98%
error rate	range	0	0	0	%	0.00%

4.1 性能测评 – 存算分离 VS 本地盘

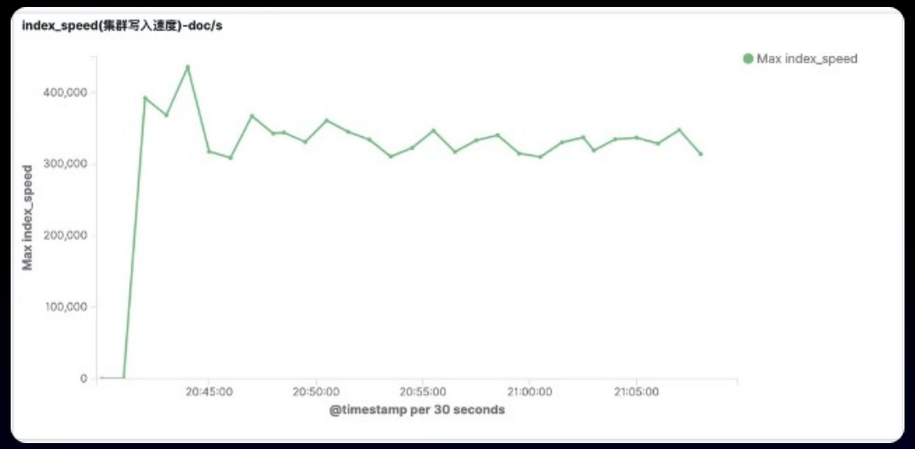
查询类型	本地(SSD云盘)	存算分离	存算分离+并行化	可搜索快照
match_all（全部匹配）	3	3	4	59
term（单值精确匹配）	6	16	7	71
terms（多值精确匹配）	5	4	5	45
range（范围查询）	12	23	9	28
aggs（聚合查询）	1544	2020	580	2575
desc_sort_timestamp（按时间字段降序）	65	81	33	187
asc_sort_timestamp（按时间字段升序）	71	54	8	256
desc_sort_with_after_timestamp（降序排序中增加search_after）	1140	1968	440	1863
asc_sort_with_after_timestamp（升序排序中增加search_after）	936	1389	395	1692

总结

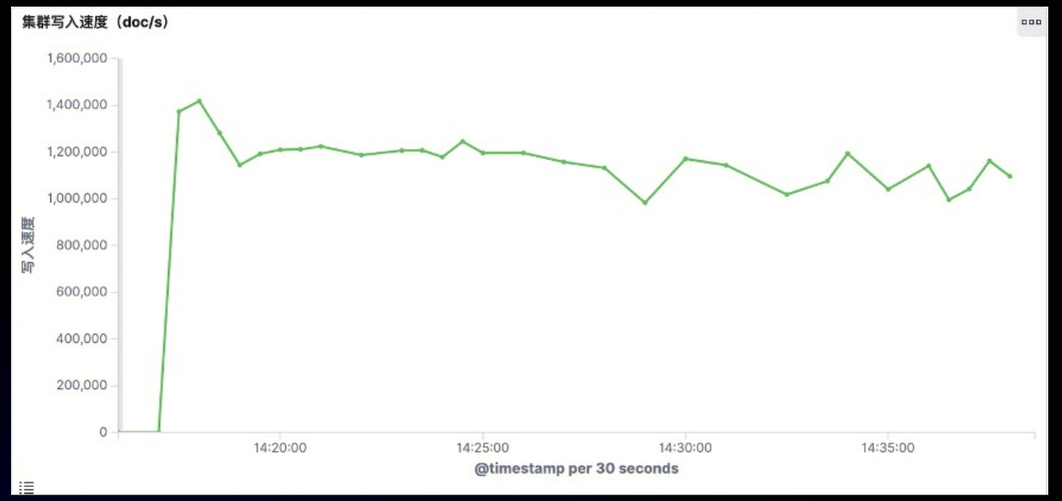
从和本地盘的查询性能损耗可以看出，可搜索快照的查询性能损失太大，自研云原生存算分离相较于本地盘的查询性能损耗，仍在可接受的范围内。增加并行化压测后大部分场景比本地还要快 2~3 倍。（查询延时单位：ms）

4.2 性能测评 -读写分离

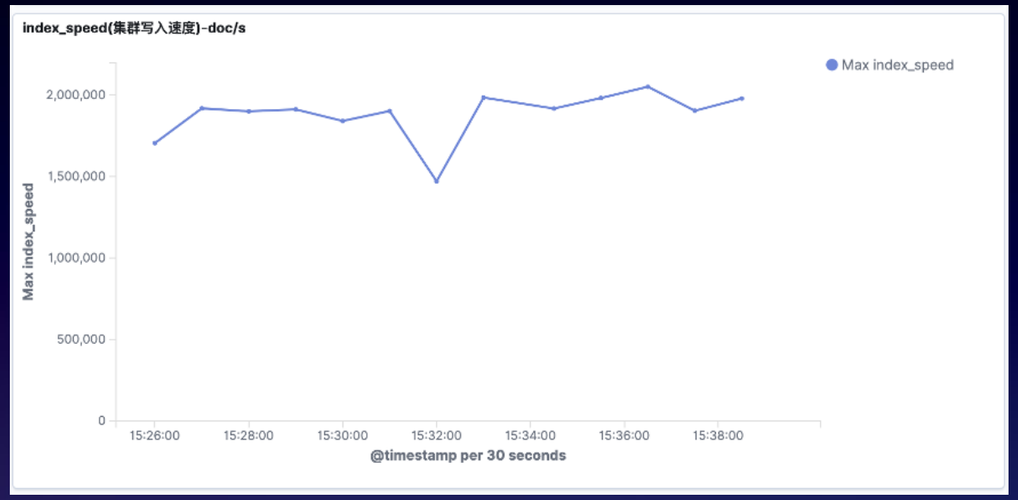
压测环境：3台标准型 SA2 16 核 64G, 1500GB SSD 云硬盘 x 1
压测数据：随机生成
压测工具：编写代码消费 kafka



ES默认写入性能



本地读写分离写入性能



共享读写分离写入性能

写入类型	副本	批次大小	写入性能	说明
ES 默认	1	5000	31w/s	默认 ES 的写入性能
本地读写分离	1	5000	119w/s	是 ES 的 3.8 倍
共享读写分离	1	5000	200w/s	是 ES 的 6.5 倍

4.3 性能测评 – IO 并行化

压测环境：3台标准型SA2 16 核 64G, 1500GB SSD 云硬盘 x 1

压测数据：geonames

压测工具：esrally

压测背景：此处 IO 并行化并发度设置为 3

4.3.1 压测明细

Min Throughput	phrase	95.776	110.015	14.2387	ops/s	+14.87%
Mean Throughput	phrase	106.572	110.029	3.45703	ops/s	+3.24%
Median Throughput	phrase	109.88	110.029	0.14886	ops/s	+0.14%
Max Throughput	phrase	109.94	110.048	0.108	ops/s	+0.10%
50th percentile latency	phrase	3.30563	3.02988	-0.27574	ms	-8.34%
90th percentile latency	phrase	1703.91	3.44456	-1700.47	ms	-99.80%
99th percentile latency	phrase	1953.4	3.70481	-1949.7	ms	-99.81%
99.9th percentile latency	phrase	1973.03	5.98688	-1967.04	ms	-99.70%
100th percentile latency	phrase	1979.58	8.59021	-1970.99	ms	-99.57%
50th percentile service time	phrase	2.47474	2.32926	-0.14548	ms	-5.88%
90th percentile service time	phrase	2.98445	2.5114	-0.47304	ms	-15.85%
99th percentile service time	phrase	43.5048	2.69534	-40.8095	ms	-93.80%
99.9th percentile service time	phrase	43.7923	5.08675	-38.7055	ms	-88.38%
100th percentile service time	phrase	44.0776	8.26443	-35.8131	ms	-81.25%
error rate	phrase	0	0	0	%	0.00%

4.3.2 总结

IO并行化并发度设置为3，性能普遍提升3倍左右，经过压测对比发现，P50，P90，P99耗时普遍减少5~10倍，查询抖动更少更稳定。

如果cpu核数更多，拆分的子请求可以更多，性能会更好，如果将并发度设置为5，理论上性能会提升5倍左右。

类别	短语查询(ms)	聚合查询(ms)	脚本查询(ms)	自定义打分(ms)	排序(ms)
关闭并行化	43	43	374	408	143
打开并行化	5	4	132	125	44

总结

降本增效

存算分离技术可以有效降低成本，**成本降低 50%~80%。**

物理复制技术可以消除冗余副本写入计算开销，**写入性能提升 50%。**

性能提升

读写分离技术可以提升可用性，并通过内存 Segment 构建与拷贝，**将写入性能提升 3-5 倍。**

无状态共享计算资源池可以换取额外性能，**将写入性能提升 5-20 倍。**

查询优化

查询性能优化通过 IO 并行化和查询裁剪，**实现了冷热一体搜索。**

智能分层技术可以根据数据查询频率智能下沉和卸载，**提升易用性并实现无感知降本。**