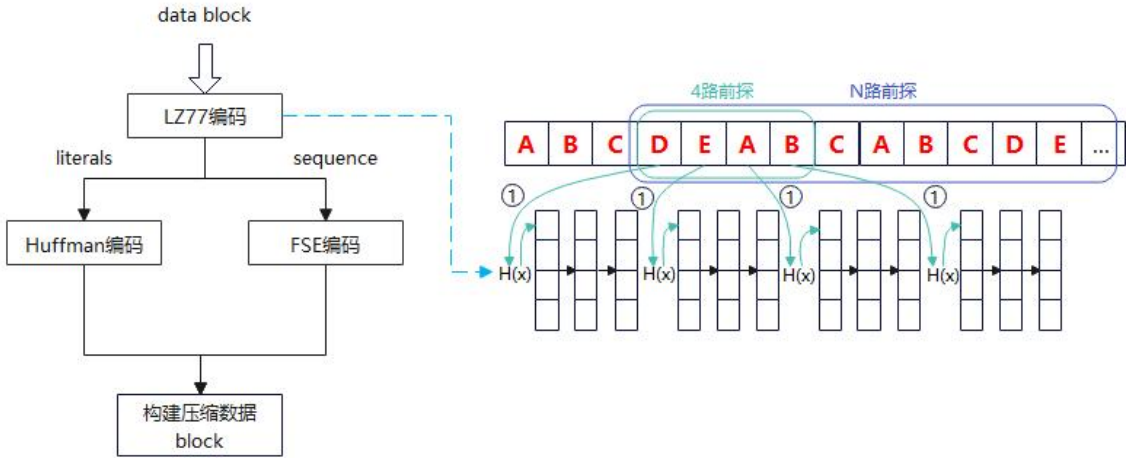
难题1：高带宽无损压缩硬件架构设计

出题组织：数据存储产品线|中央研究院　　接口专家：陈仁海 chenrenhai@huawei.com

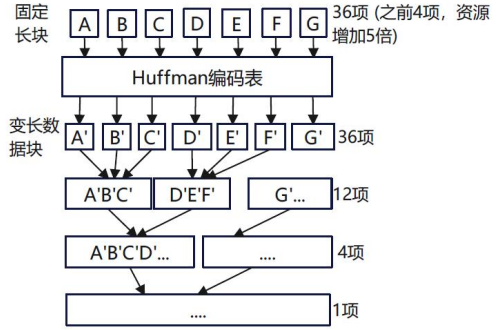
技术背景

- 由于存储和传输的数据量呈指数级增长，数据压缩在过去几十年一直是研究的主题，而使用硬件进行数据压缩，可以极大提高压缩速度。当前LZ77编码、Huffman编码和FSE编码所组成的混合压缩算法有较高的压缩比，然而压缩带宽受限，通过硬件实现可以有效提高压缩速度，但当前硬件加速的设计方案需要较多资源开销。如何保证压缩比和性能的同时，尽可能的减少资源的使用是当前遇到的主要问题。
- LZ77使用多路前探进行并行处理，例如，4路前探是分别以“DEA”、“EAB”、“ABC”和“BCA”为起始字符串去各自的引擎（哈希表结构）中进行匹配，每一路前探都会给出多个匹配候选结果，然后再从多个匹配候选结果中选择一个最优结果。
- Huffman和FSE将对LZ77压缩后的数据进行分段编码，有不错的并行性。



技术挑战

- **LZ77资源开销过大**：使用4路前探需要对访问的哈希表进行复制，从而实现4字节并行处理。N路前探，要达到N字节并行处理，就需要实现哈希表的N份资源复制，会使得硬件资源开销过大。
- **多路前探选择最佳匹配困难**：每一路前探会自身给出一个最长匹配作为最优匹配，但综合多路前探如何选择最优匹配?
- **Huffman编码阶段开销过大**：Huffman会同时进行36个字符编码，但编码后的每个数据块长度不确定，每个字符有11种编码长度可能，导致硬件设计中需要把所有的情况展开，资源暴增，需要在保证带宽和编码效率的同时尽量减少硬件资源开销。



当前结果

- **算法整体压缩比**：当前使用混合压缩算法在典型数据集（数据库、vid、vsi），共计364GB数据上进行测试，LZ77使用四路前探，在8KB、16KB、32KB、64KB和128KB压缩粒度下，相比于ZSTD-9压缩比分别损失6.6%、6.8%、7%、7.5%、8%。
- **LZ77算法资源开销**：哈希表使用1310KB，历史表使用240KB，LZ77共计使用资源1550KB。
- **Huffman算法资源开销**： FPGA硬件环境下Huffman共计使用124000个LUT。

技术诉求

- **基于硬件实现的并行化LZ77算法设计**：在单个时钟周期内至少处理6个字节的情况下，LZ77资源开销应小于1000KB。
- **基于硬件实现的并行化Huffman算法设计**：在单个时钟周期内至少处理36个字节的情况下，FPGA硬件资源LUT的开销降到60000以下。
- **基于硬件实现的整体算法设计**：对于典型数据集（数据库、vid、vsi），要求支持8KB~128KB的压缩粒度，并且压缩比不低于软件ZSTD-9算法。在FPGA上主频应不低于200MHz。

难题2：相似图片关联压缩

出题组织：数据存储产品线|亚太研究院　接口专家：孙赵亿 sun.zhaoyi1@huawei.com

技术背景

- 图片数据在当前各设备存储、网络传输流量中占据很大比重。
- JPEG/JPEG2000是最常见的一些图片格式，在相机连拍、监控抓拍、自动驾驶、医疗等场景，存在大量视觉相似度高、内容重复的图片。
- 当前的主要问题是，给定一些视觉上高度相似的JPEG图片，其二进制数据差异较大，通常的文件去重技术和预测编码，无法实现大比例数据缩减以达到节约存储空间的目的。

问题描述

给定若干张视觉相似度高的JPEG/JPEG2000格式图片（来源：相机连拍；自动驾驶抓拍。JPEG和JPEG2000不混合），Image_0, Image_1, Image_2, … , Image_N, 基于数据关联，对各图片进行高比例无损重压缩。

技术挑战

- **数据已编码**：JPEG/JPEG2000文件是由图片裸数据有损压缩所得到，二进制数据随机性较强，无法直接进行重压缩。现行重压缩方案均需将数据部分解析，如何高效的进行数据部分解析并能保证完美还原是一大挑战。
- **内容相似和数据关联的gap**：图片内容或视觉上的相似，无法直接转换为数据上的相似，尤其是对于变换和编码后的数据。

图1 [1]

当前结果

- 自研:
  - ✓ 对多个JPEG二进制文件，采用块去重、文件去重技术，无数据缩减收益。
- 业界:
  - ✓ 多图关联JPEG重压缩：业界无相应或类似算法。
  - ✓ 单张JPEG重压缩算法：如Brunsli, packJPG, Lepton，空间节约收益只有20%（针对常见手机拍照，4K图片）。

JPG文件　　无损还原JPG文件

图2 [2]

技术诉求

- **高性能完美还原**：JPEG/JPEG2000格式复杂，需实现保证二进制文件相同前提下的高性能解码和重编码（鲲鹏920服务器，解码编码达到60/40 MB/s）。
- **图片相似度量与检测**：视觉相似度无法量化，需定义一种可量化的JPEG/JPEG2000等已编码图像间相似度的度量，并提供在图片池中的相似检测方案，利用相似性设计如预测、重删等压缩方案
- **提高压缩率**：相比已有的单JPEG/JPEG2000图片重压缩算法，利用图片间关联性，需大幅提升整体压缩比，达到2:1以上。
- **性能需求**：为在一些通用架构上部署，要求ARM架构单核单线程重压缩/解压性能不低于20MB/s（鲲鹏920服务器或同等性能服务器）。

参考文献：
[1] https://www.shutterstock.com/zh-Hant/blog/sequence-photography-guide
[2] https://pixabay.com/zh/photos/maremma-sheepdog-dog-pet-5565203/

数据集建议：
1. 自动驾驶连拍：加州理工学院行人检测数据集 （下载链接）
2. 多尺度：以DIV2K高清图片为基础，生成多尺度JPEG

难题3：数据特征识别融合检测算法

出题组织：数据存储产品线　　接口专家：刘帮 liubang9@huawei.com；朱锴 zhukai27@huawei.com

技术背景

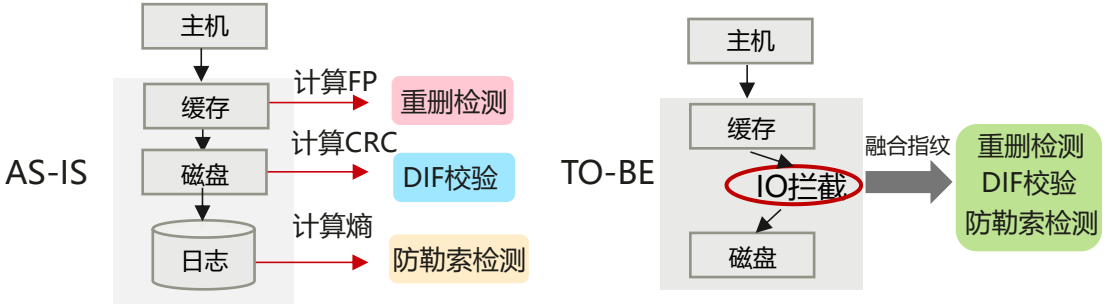
- 重删技术通过卷积数据块的每个byte计算重删指纹(FP)[1]，识别与消除完全相同的数据块。
- 存储系统中的DIF校验技术会计算每个512字节的CRC值以验证数据读写时的完整性。
- 存储系统中的防勒索技术会检测文件熵值[2]的变化幅度判断文件是否被加密[3]。
- 当前的主要问题是:（1）重删技术，DIF校验，防勒索检测都需要计算确定数据byte级变化性（指纹，CRC值，信息熵），导致多次计算，资源开销大。（2) 当下防勒索方案采取分析日志记录的方式实现文件级别的离线检测，无法做到IO级别的秒内实时检测，以快速恢复。

问题描述

对每个数据块计算融合指纹，该指纹能够用于数据去重，或DIF校验；同时融合指纹能够识别数据块的加密性，在数据块写入时实时检测数据块是否被异常加密，实现数据块级别在线防勒索检测。

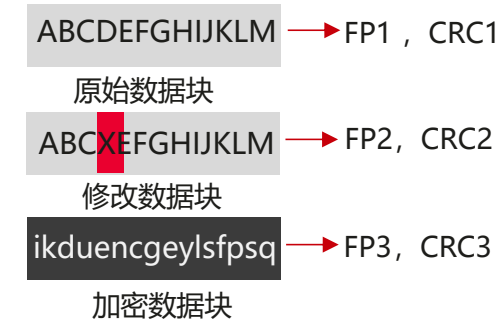
技术挑战

- **多次独立计算数据变化性，算力消耗大：**经典重删指纹采用SHA256等字节级改动敏感算法，DIF校验计算CRC32值，防勒索检测算法计算熵值，三者都是计算密集型任务，CPU消耗大，对系统端到端的性能影响20%+。
- **防勒索检测延迟高：**当下防勒索检测通常基于日志记录的后台异常加密行为检测，端到端最快检测时间需要10秒，延迟高。
- **已编码数据检测准确率低**：对于图片，视频已编码、或已压缩文件等高熵数据，无法准确区分正常的数据更新与防勒索加密，误报率高[4]。

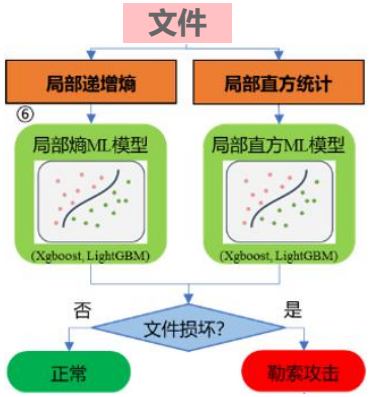


当前结果

- 对于每个数据块（比如8KB），只要数据块更改1byte，重删指纹与CRC值都将会发生变化，无法直接用于防勒索的加密文件检测，误判率高。
- 防勒索技术：基于日志记录，后台分析日志，并针对整个文件级别计算局部熵变化，以及局部直方图分布，送入ML模型中判别是否是被加密，无法在线实时快速报警。
- 业界三种技术相互独立，并无法实现数据块级别的在线加密检测。



重删指纹与CRC值都不相同



后台执行，非实时

技术诉求

- **近实时的块级数据融合特征检测算法**：计算融合特征指纹，计算性能不低于1GBps/核，实现在线防勒索检测， 触发异常检测1秒内报警；同时要求8KB分块重删率无丢失，DIF校验无异常。
- **高精度安全检测：**有效区分正常压缩编码数据与勒索加密行为，降低误判率，误判率小于1%，召回率大于99.999%。

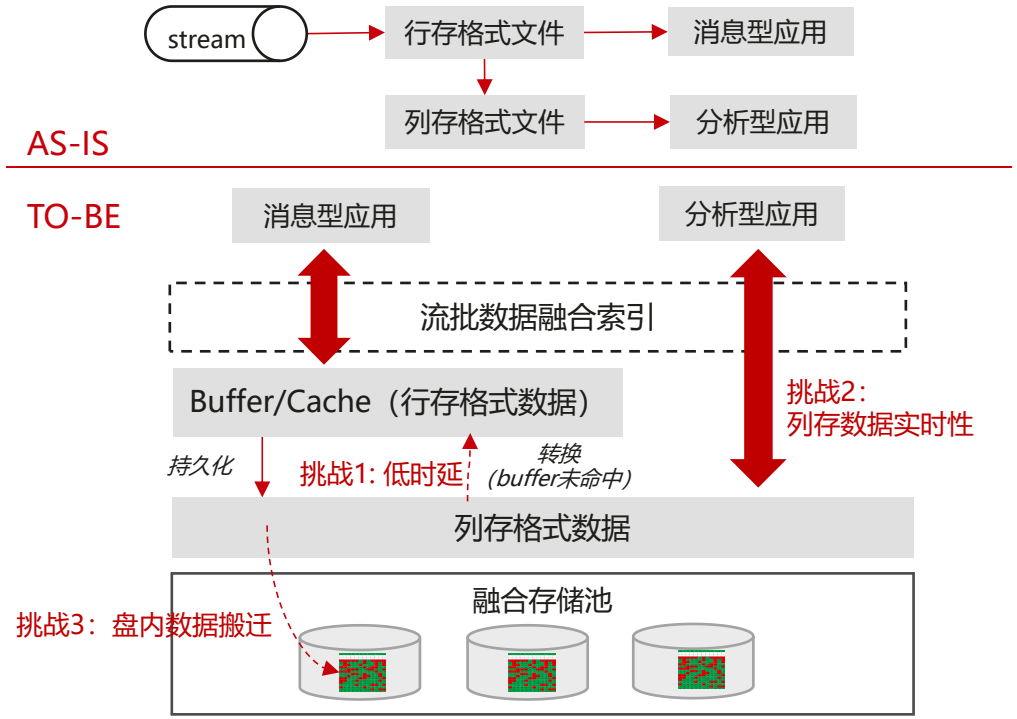
参考文献:
[1] https://en.wikipedia.org/wiki/SHA-2.
[2] https://zh.wikipedia.org/wiki/%E7%86%B5_(%E4%BF%A1%E6%81%AF%E8%AE%BA)
[3] Ransomware: Recent advances, analysis, challenges and future research directions[J]. Computers & Security, 2021, 111: 102490.
[4] Byte Frequency Based Indicators for Crypto-Ransomware Detection from Empirical Analysis[J]. Journal of Computer Science and Technology, 2022, 37(2): 423-442.

难题4：流批数据合一且盘控协同的融合索引技术

出题组织：数据存储产品线　　接口专家：张家全 zhangjiaquan269@huawei.com

技术背景

- 流批一体数据处理是大数据领域最重要的数据分析手段之一。业务数据以流式方式实时写入、通过消息队列通知实时分析系统来消费；同时定期进行格式转换和合并，生成列式数据格式大文件，提供给批处理引擎进行全量分析。
- 主要问题：数据重复存储，在流式引擎和批处理引擎分别以行存和列存的方式存放数据，空间浪费一倍，且两侧数据永远不一致。
- 尝试在存储系统中，流式引擎数据和批处理引擎数据都以列存格式存放，数据生产和消费时：
  - ✓ 流式数据进入存储先存放于行式数据缓冲区，定期转换持久化为列存数据；
  - ✓ 分析型应用直接访问转换完成的列存格式数据。



技术挑战

- **保证流存储稳定低时延访问**：当访问数据在Buffer空间未命中、需要到列式数据中读取时，如何保证稳定百微秒级低时延。
- **保证分析型应用数据实时性**：当流式数据还在Buffer空间，没有完成到列存数据的转换时，如何对分析型应用可见，实现流批访问数据零GAP。
- **数据更新时减少盘内搬迁**：列式数据发生更新时，会产生大量数据块合并、删除等行为，如何优化索引以减少大块数据在盘内搬移的情况发生。

当前结果

- **Lamda架构（流、批分离）**：流存储系统（Kafka）与批处理存储系统（HDFS或S3）互相独立，数据存了2份，且批处理系统的数据一般滞后流处理系统1天。
- **数据更新方法**：列存数据文件不支持in-place更新。有部分数据删除时，将未更新的数据和更新的新数据写入新文件，再把老文件删除，在盘上产生大量的数据搬移。

技术诉求

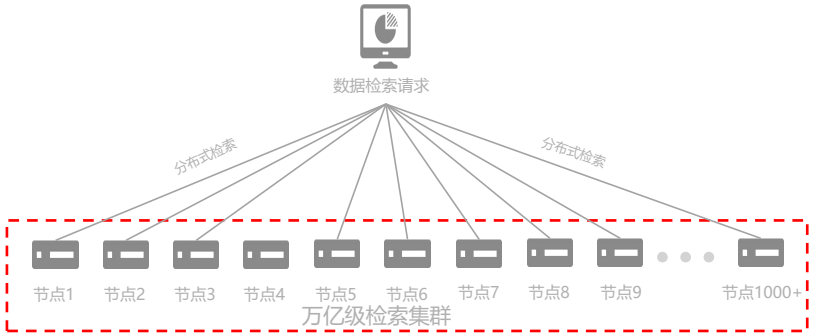
需要建立一套流批数据融合的数据索引机制，同时满足流式处理基于时间的查询，以及批处理基于范围匹配的查询：

- **流式应用查询诉求**：快速返回给定时间窗口（t1, t2）内的所有数据，时间复杂度O(1)，响应时间200us以内（假设介质时延<100us）；
- **分析型应用查询诉求**：对于给定时间窗口（t1, t2）和字段d的取值范围（s, t）,返回所有符合条件的数据，查询时间复杂度为 E * ln（N）/ n，E为常数，N为时间窗口内的总数据量，n为并发工作线程数；
- 充分考虑与SSD层FTL的GC机制结合，在数据更新时使物理介质上发生的数据搬移最小化。

难题5：极致性价比的万亿秒级检索技术

出题组织：数据存储产品线　接口专家：汪慧 wanghui397@huawei.com；王艳 jessica.wangyan@huawei.com

技术背景



万亿级元数据检索需要部署的集群规模达上千节点

- 传统的元数据检索技术主要依赖于资源的堆积和多节点部署，万亿级元数据检索通常需要部署上千个节点，每个节点的硬件配置要求数百G的内存和数十核的CPU，成本过高；而在存储场景，仅有十个管理节点，所需要管理的数据达上万亿，亟需实现有限管理节点下的海量元数据检索。
- 现有存储业务场景中，需要管理万亿条元数据，每条元数据对应着30+字段，支持的字段类型包括文本，时间，整数，浮点数以及布尔值。针对不同字段类型，所采用的Elasticsearch元数据搜索引擎提出了不同的索引加速策略，分别是针对时间和数值型字段的BKD-Tree和针对文本字段的FST+后缀词块+倒排表。在实际检索中，最需要优化的是文本类型的字段。其检索时需要将文本索引加载到内存中实现检索加速，对于内存的需求较大。如果减少节点数量，单节点管理数据规模成倍增长，其导致的内存开销将成为检索技术的主要障碍。

技术挑战

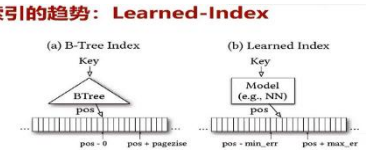
- 检索规模呈百倍增长：有限资源引起的内存瓶颈是万亿秒级检索的核心问题。搜索引擎需要缓存DocValues，复合索引，FST，后缀词块和倒排表等索引以实现加速。现有内存无法满足万亿级缓存需求且调度策略未考虑到多级索引的不同特征，频繁读盘和GC （内存释放）将引起检索性能下降。
- 复杂表达式检索引起检索效率低：复杂的表达式检索主要基于确定性有限自动机实现，每条语句的匹配需要遍历整个自动机，CPU开销大；尤其是数据规模骤增引起的语句匹配数量增加进一步加剧了CPU的占用，导致检索效率极低。

当前结果

- 内存占用大：目前在10节点（单节点 16v 64G）百亿情况下，表达式检索性能为十秒级，检索结果聚合算法性能为秒级，内存占用率达98%；数据规模增长百倍后，相同检索算法对内存的需求理论上百倍提升，在内存不变的情况下检索性能将受到极大的影响。
- 表达式检索匹配状态多：目前表达式检索中，检索语句长度和确定性有限自动机状态数呈指数增长关系，长度为几十的复杂语句通常产生过百甚至上千状态数。

技术诉求

- 检索结果聚合算法：优化索引结构（如学习型索引[1]），设计软硬协同加速算法或改进调度算法，以达成搜索引擎的高效内存利用，实现10节点（单节点16v 64G）万亿规模元数据的统计结果可视化 (并发数10，查询时延<2秒)。



- 表达式检索算法：基于当前的索引结构设计高效的表达式检索算法，减少表达式检索时匹配的复杂度，实现10节点（单节点16v 64G）万亿规模元数据的高效检索(并发数10，查询时延<2秒)

参考文献：
[1] T. Kraska et al., The case for learned index structures. SIGMOD, 489-504 (2018).