|  |
| --- |
| 应用领域   * Primary storage system * Used to serve primary workloads. * 延迟敏感，因为应用需要通过去重系统读写数据，不能过分影响到应用的性能 * 对内存和CPU的使用需要与其它应用竞争 * 现有方案是针对sequence进行去重的，这样可以减少write path中由于去重引入的开销，也可以减少fragmentation，从而减少read path中额外的开销。 * Secondary storage system * archival or backup purposes * 备份系统需要在一定时间内完成，因此需要保证写数据的速度 * 读取数据时也要保证一定的速率，此时需要限制文件的碎片化   The chunk-lookup disk bottleneck   * 大数据情况下很难将所有的index都存入内存   + a store that contains 10 TB of unique data and uses 4 KB chunks. Then there are  unique chunks. Assuming that every hash entry in the index consumes 40 bytes, we need 100 GB of storage for the full index. * 从硬盘中找index很慢。 * If a seek on average takes 4 ms, this means we can look up only 250 chunks per second for a processing rate of 1 MB/s, which is not acceptable.   Computation bottleneck   * Karp-rabin和对chunk的hash都很费CPU。解决方案有并行计算，GPU替代   Restore bottleneck:   * 由于去重后碎片化的原因，将数据存在CMR硬盘上一定会出现分散存储。由于磁盘随机读取的性能比连续读取差很多，因此去重后的碎片化一定会导致restore的性能下降 * Read amplification: 最极端的情况，一个文件含有N个chunks，每一个chunk都在不同的container中，这意味着要读取N个containers，有着很明显的read amplification   Chunk locality条件：   * the chunks of a backup stream will appear in approximately the same order in **each full backup** with a high probability * 什么时候不满足：incremental backup的时候不满足，备份emails   File Similarity条件：   * 即利用文件之间的相似性。文件A与修改后的文件按A’一般而言很相似。两个基于同一操作系统的虚拟机也有很多相似文件   评价指标   * Deduplication efficiency: 即去重率 * Scalability: the ability to support large amounts of raw storage **with consistent performance** * Throughput: the rate at which data can be transferred in and out of the system. 备份是有窗口期的，一定要保证一定速率   三者不可能同时实现，   * **To achieve high write throughput and deduplication ratio**, all the indexing metadata shall be stored in memory for fingerprint comparison. But as the size of indexing metadata grows, the indexing metadata could no longer be completely stored in memory, which sets a limit to scalability. * **To achieve scalability and high deduplication rate**, disks shall be used to store a part of indices. As a result this will hurt write throughput significantly since I/Os could be introduced during querying the indices * **To achieve high write throughput and scalability**, a sampling strategy(e.g. Sparse Indexing) can be used, where only sampled indices are stored in memory for fast comparison. Since the fingerprint of an incoming chunk is not compared to all the fingerprints existing in the system, not all duplicate chunks can be detected, which hurts the deduplication rate.   Chunk size的影响：   * 越大的chunk size，去重率越低，造成空间节约越小，但是压缩率一般上升。因此可以采用较大的chunk+压缩的方式以达到较好的空间节约效果。 * 越大的chunk size，需要的fingerprints数量越少，可以显著减轻disk bottleneck * Chunk size越小，metadata就越多。小于一个阈值以后，metadata造成的overhead就很有可能抵消去重的影响   Cache:   * Temporal locality比较好的情况下，cache才是有作用的。如果是weak temporal locality的情况，cache很有可能是无用的 * 当cache既要处理workloads of weak temporal locality又要处理workloads of strong/moderate temporal locality时，前者会对后者的命中率有很大负面影响 |