**Deduplicate & Restore：**

备份最关键的部分是指纹索引，恢复最关键的部分是数据块存储。重删的底层存储介质一般是磁盘，而重删天然地会打乱数据块的逻辑顺序，导致数据流不是连续存储。

大部分重复数据删除研究集中在备份的指纹索引上，大家都认为这是最关键的问题。的确，不解决磁盘瓶颈问题，备份窗口过大将导致重删不实用。重删系统的恢复性能一直处于被忽略的状态，原因是相对备份，恢复是小概率时间，而且体系结构最重要的Amdahl定律讲的就是重视大概率事件。但是我们也要明白一点，备份就是为了恢复的，如果备份的工作做得很充足，到了关键时刻不能快速恢复，一切都是白费力气了。或者说，备份就是一个不断投入的过程，而一次恢复就是赎回成本的过程，我们总是在研究如何减少备份的投入，而忽视了如何赎回更多的成本：恢复的越快，赎回的成本就越高！

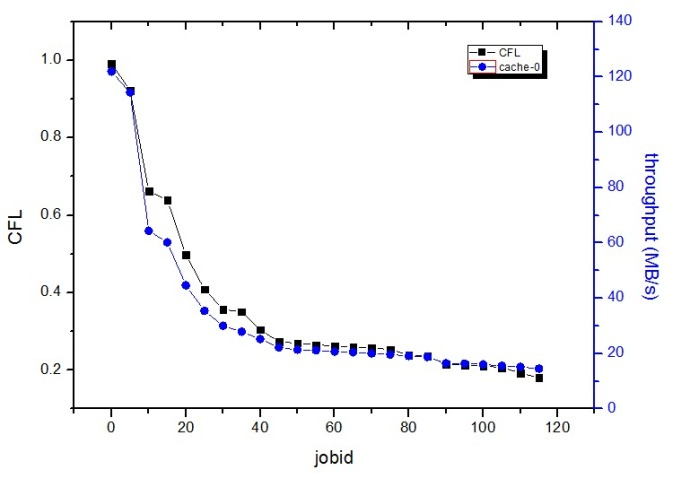
**Fragmentation:**

借用文件系统的两个名词，但是意思并不一样。  
  
内部碎片：同一个备份作业或者同一个文件内部的重复数据导致的碎片，比如vmdk文件，通常自身有20%的重复数据。“内部”的范围指通常的一次读操作的范围，如果每次读一个文件，则内部指文件内部；如果是恢复一个备份作业，则指作业内部。直觉上，LRU缓存对这种碎片有较好效果。  
  
外部碎片：不同备份作业或者不同文件之间的重复数据导致的碎片，比如linux源码归档，两个版本的源码通常有很多重复数据。“外部”也由一次读操作的范围而定，如果读的单元为文件，则外部指文件之间；如读的单元为作业，则外部指作业之间。直觉上，LRU缓存对外部碎片无能为力。  
  
我们还需要借用一个概念：Chunk Fragment Level（CFL）[1]。CFL=OCF/CCF，其中OCF值最佳情况需要使用的container数量，CCF为实际使用的container数量。当CFL=1代表没有碎片情况。需要说明的是，CFL只能衡量外部碎片，在使用时，OCF需要去掉内部重复数据。

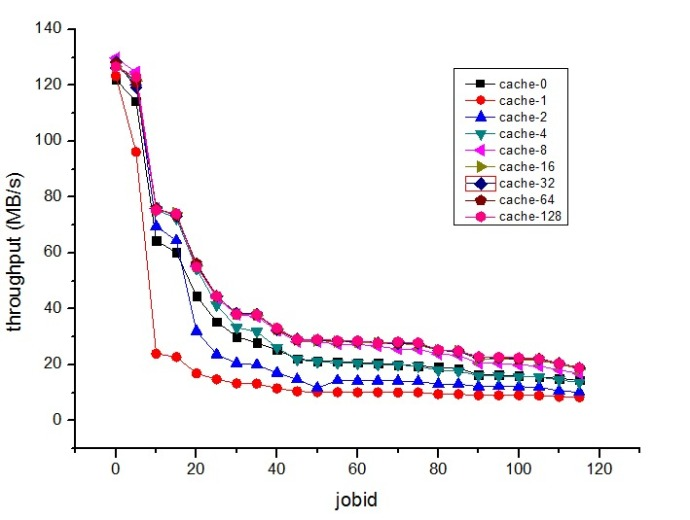
**Experimentation:**

存储这一层使用了Data Domain的设计，container是固定大小的、自描述的单元。备份开始后，数据块被不断添加到container buffer中，直到contianer满了，再一次性写入磁盘。这有几个好处：1.避免的磁盘的小写；2.保留了数据块的空间局部性。Data Domain利用这种设计，配合Bloom Filter，缓解了备份时的磁盘瓶颈问题。  
  
使用的数据集是115个linux版本的tar文件，以及8个版本的虚拟机镜像。这两种数据集代表了两种情况：linux tar只有外部碎片；虚拟机兼有内部碎片和外部碎片。  
  
**Linux tar**

linux tar只有外部碎片，没有内部碎片。一个版本的内核源码本身没有什么重复数据，但是多个版本之间存在大量重复数据，而且数据的变化主要是更新（随机写，而不是append）。采用的是container-based cache 而不是chunk-based cache 因为数据块不存在时间局部性（无内部碎片），所以以数据块为单元的缓存的价值不高。



如图所示，横坐标是作业id，黑色是CFL，蓝色是不使用缓存策略获得的恢复性能，两条曲线惊人匹配，不断下降，到最后只剩下10MB/s作业的吞吐率了。由于CFL不考虑内部碎片，因此两条曲线的吻合说明在这里外部碎片是主要瓶颈。因此，这种数据集的症状是外部碎片越来越严重，早期作业和后期作业恢复性能差距明显。这幅图还反映了一个问题，后期作业占用的container数量越来越多，简单的垃圾回收（比如等到container中不存在任何有效数据块时，才回收）并不能有效回收空间，大多数情况是，后期作业倾向于覆盖所有的container。再来看看LRU缓存对恢复性能能起到什么作用。

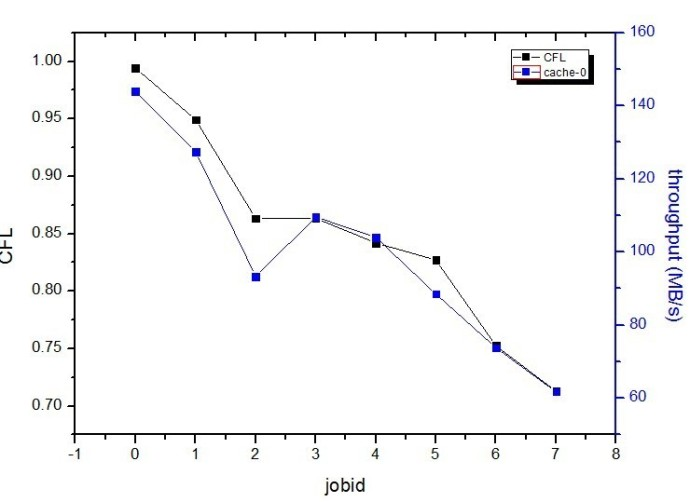


图中cache-N表示，缓存大小为N个container，N=0表示没有缓存。可以看出当N很小时，性能还不如N=0，使用较大的缓存可以小幅度提升恢复性能。jobid还小的时候，数据块尚有较好的空间局部性，所以预取的效果比较明显，而随着不断备份，外部碎片化严重后，以容器为单元的预取效果不佳，必须通过为预取设置更大缓存（来利用容器的时间局部性）才可获得性能提升，而且不明显。通过图可以发现，随着备份继续，大缓存的恢复性能会慢慢被cache-0超越（因为块的空间局部性越来越差，导致容器的时间局部性越来越差），必须使用更大的缓存来弥补预取的失效。这验证了LRU缓存对外部碎片无效的直觉。

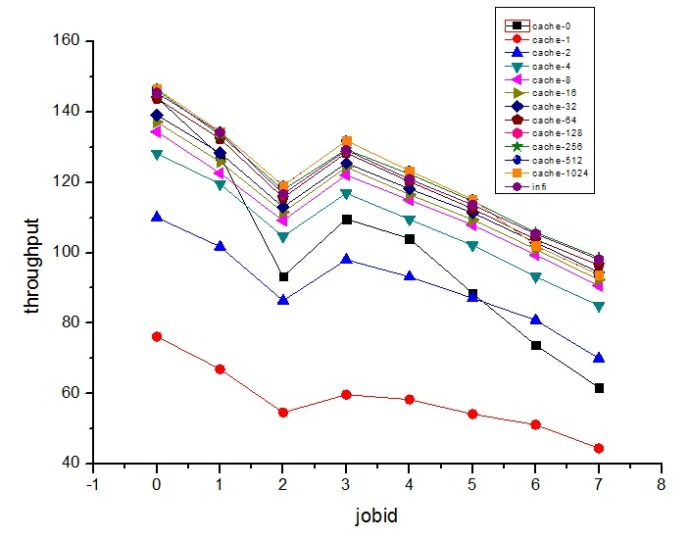
这里解释一下chunk的空间局部性和container的时间局部性，随着备份的不断进行，在deduplication过程中chunk会不断的被打乱其逻辑顺序，物理上分散到各个位置，所以其空间局部性会越来越差（体现在取出的一个container中也许只有百分之一的chunk是将要被使用的其余都是浪费cache空间）。而chunk的空间局部性就等于container的时间局部性，因为高的chunk空间局部性会使得一个container被反复利用，对这一个container来说其时间局部性就因此大大增加。

**虚拟机**

虚拟机镜像既有内部碎片也有外部碎片，单独备份一个作业，大概有20%的重复数据，这导致了内部碎片；多个版本的vmdk之间也存在大量重复数据，而且既有append也有随机更新操作，因此产生外部碎片。先来看看CFL。



两条线的总体趋势还是比较匹配的，只是在job2到job3的时候，CFL没有变化，而恢复性能却提升了，原因可能是内部碎片增加了，而外部碎片没有变化。再来看看LRU缓存的效果。

这种数据集用预取和缓存可解决部分问题，但仍有优化的空间。

可以看出LRU缓存可以对内部碎片起到一定的作用，但是对于外部碎片作用不大。

内部碎片和外部碎片其本质区别在于：内部碎片肯定会在一次restore过程中被触发，数据块的时间局部性较好，其reuse distance是cache可以接受的范围，因此能利用缓存有效缓解，而外部碎片本身就无法保证时间局部性（文件A和B的恢复时不一定前后脚的被恢复）其空间局部性又因为deduplicate的天然性质而无法保证，所以无法利用缓存。往往出现的情况是，在恢复文件的后期，数据被打散在各个容器中，为了读取一个数据块就预取一整个 container是得不偿失的。