|  |
| --- |
| DupHunter: Flexible High-Performance Deduplication for Docker Registries |
| 背景：   * Docker的container images仍旧存在很多重复数据   + Docker采用share of layers的方式去除重复的layer，但是某layer即使是改变了一点点数据也会被当作不同的layer，无法被share。并且很多layer实际上是共享了很多但不是全部文件的。 * 现有的数据去重技术会显著增大pull of layers的延迟。   + layers有两种存储方式: compressed or uncompressed。compressed的模式下去重几乎没效果。而uncompressed模式下去重一定会引入**碎片化**，增加读延迟   + 在uncompressed的模式下去重以后再去pull layers会由于需要重建docker image的原因导致读延迟增加 * Image access patterns can be predicted reliably   + 用户首先取得docker image的manifest，再根据manifest决定是否Pull a layer。用户可以分为两类，一类为always pull regardless of whether the layer requested already exists locally。另一类为pull once，即如果本地存有该layer，则不再远程去取该layer。   + 由于用户取manifest和取layers存在时间差，去重系统可以采用preconstruction of layers   动机：   * Data redundancy among Docker images * 当docker的layers以compressed modes存放时，deduplication不起作用。这是因为原文件夹中哪怕只是一个byte不同，最后也会导致压缩文件不同。 * 而将layers解压存放到已有的文件去重系统中，则会导致pulling变慢，因此需要专门针对pulling进行优化。   解决方案：   * 一般来说，每一个layer在多个节点上都有replica。该去重系统的每个节点根据去重策略，决定是否将该layer的replica解压并且去重(注：文中**去重的粒度貌似是file**这一级别；多备份的原因是为了保证整个系统的有效) * 去重方案很灵活，可以指定一个layer的多少replicas需要去重，也可以根据layer的popularity程度来决定其replicas的去重比例。一般来说，layers越popular，不去重的replica越多。 * P-server用以存储一个layer完整的tarball，而D-server负责存储该layer的replica去重后的文件 * 根据用户的行为习惯，对P-server，预先取image中的layers到prefetch cache中。对D-sever，预先重建layers到preconstruct cache |
| Available:   * https://github.com/nnzhaocs/DupHunter |