Nydus copy 实现不落盘的能力

一、背景

在容器镜像分发和加速场景之中，`nydusify copy` 指令用于将源镜像层的各个 blob 层复制到目标仓库。当前实现中，所有的 blob 层会被完整地拉取到本地磁盘，再从本地磁盘推送到目标仓库。这种“先落盘再上传”的方式带来了明显的性能瓶颈：

- 磁盘 IO 压力大，所有的 blob 必须先复制到本地磁盘，造成了大量的磁盘 IO。
- 磁盘空间占用高，大镜像会大量占用本地磁盘，容易导致磁盘空间不足，尤其是和大模型有关的镜像。
- 传输效率低，数据需要经过 源仓库→本地→目标仓库这一个流程，需要两次传输，整体耗时长。

为此，我们希望能够为 Nydus copy 添加不落盘的能力，直接将源镜像拷贝到目标仓库。

二、现有实现逻辑

拉取阶段

首先需要解析源镜像，获取所有 blob 层的列表，然后依次将每一个 blob 层下载到本地。

推送阶段

读取本地临时目录之中的 blob 文件，通过 containerd 的 pusher 接口，将 blob上传到目标镜像网站上去。

并发与分块

目前的实现之中，支持一定的并发，可以将源镜像的 blob 并发从本地上传上去；对于大文件，支持分块上传，但是目前分块的逻辑仍然基于本地文件。

manifest更新

所有 blob 上传完成之后，更新目标镜像的 manifest信息。

三、优化目标

消除不必要的本地落盘，实现“源仓库→目标仓库”的直接流式传输。降低磁盘空间和 IO 占用，提升整体复制速度。

四、优化方案

首先解析源镜像，获取所有 blob 层的标识。对每个 blob，直接从源仓库获取数据流（Reader），通过 pusher 接口，将数据流实时推送到目标仓库（Writer），整个过程数据仅在内存中流转，不落盘。继续支持多 blob 并发复制，提升带宽利用率。继续支持多 blob 并发复制，提升带宽利用率。

我们需要为 backend 层增加流式读取接口，允许直接获取 blob 的数据流，而不是文件路径：

```Bash
// trueStreamCopy implements true stream copy, pulling and pushing at the same time
func StreamCopy(ctx context.Context, srcReader io.Reader, targetWriter content.Writer, size int64, expectedDigest digest.Digest) error {
  // for small files, use simplified handling
  if size <= 128 {
    return handleSmallFileTransfer(ctx, srcReader, targetWriter, size, expectedDigest)
  }

  // use 16MB buffer for efficient transfer
  return handleLargeFileTransfer(ctx, srcReader, targetWriter, size, expectedDigest)
}

```

通过这样的方式，可以实现不落盘，所有读取均在内存之中。以加快 copy 速度。

五、时间规划

七月：完成拷贝从磁盘到内存的替换，使得镜像的下载不依赖磁盘。

八月：完成流式传输，进一步增加执行效率。

九月：打磨 PR，使其合并。