**I. Многопоточная схема**

Основная идея такая: прочитать файл по кусочкам фиксированного размера и каждый кусочек сжать\распаковать в отдельном потоке. Одновременно создавать только фиксированное количество потоков. Реализуется это таким образом (для упаковки и распаковки одинаково):

1. В основном потоке читаем последовательно N кусочков из файла.
2. Для каждого кусочка создаем новые потоки помощники.
3. Из основного потока запускаем потоки помощники.
4. Пока потоки работают, записываем сконвертированные данные в общий файл. Данные берутся с предыдущей итерации, которые запоминаются на шаге 5
5. Ждем завершение потоков при помощи Join
6. Берем из потоков сконвертированные данные и складываем их в массив. Складываются только ссылки, память копировать не нужно. Если 8 потоков, то 8 ссылок.
7. Если данных нет, то значит файл сконвертирован и можно завершать работу.
8. Начинаем новую итерацию

Отклонения в реализациях для упаковки и распаковки, скрыто за интерфейсом IWorkerIterator.

Реализация в коде: ThreadSchema.

**II. Реализация упаковки**

Размер кусочка фиксирован - 8192 байт.

Значение взято исходя из рекомендаций, описанных в этой статье <https://code.logos.com/blog/2012/06/always-wrap-gzipstream-with-bufferedstream.html>

Реализация в коде: GzipCompressor, GzipCompressWorker

**III. Реализация распаковки**

Размеры кусочков(блоков) разные, для нахождения блоков необходимо знать внутренне устройство файла gzip. Начало каждого блока определяется условиями на 10-ть символ. По мере надобности, последовательно находим подходящие 10 символов и таким образом разбиваем файл на части.

Реализация в коде: GzipDecompressor, GzipDecompressWorker

**VI. Высвобождение ресурсов**

Для объектов, которые в полях держат ссылку на FileStream, реализован интерфейс IDisposable

**V. Тестирование**

В проект добавлены UnitTest (Microsoft.VisualStudio.TestTools.UnitTesting)

Нагрузка во время сжатия и распаковки файла размером в 2.5 гб

* на ядра: 80%
* на память: 1-2%
* на диск 5-10%

См. соответствующие иллюстрации.



