CODING ASSIGNMENT - Concurrent DOWNLOADER

Goal

Code a HTTP Download Client in Go or Java that concurrently downloads files from a HTTP server. Also supports concurrent downloading of large files (>10MB) by dividing them into parts and downloading parts simultaneously. Use any local or remote HTTP server to serve a directory of such files. Code unit tests to demonstrate functionality.

Input

1. A directory of `10+` files varying from `10MB` to `1GB`.
2. You may generate the files using `dd` command if you don't have large files ready.
    1. Ex: `dd if=/dev/random of=1gig.bin bs=1G count=1` generates a `1GB` file named `1gig.bin`.
3. You may serve the files using a local/remote HTTP server of your choice.

Detail

1. Code a `DownloadClient` according to the interface specification below. Choose Go or Java.
2. Also write unit tests. Unit test should check that files are downloaded correctly.
3. Clean Code is appreciated.
4. You can use nginx to serve files and throttle bandwidth using `limit_rate` to simulate a slow, remote server. (Otherwise everything can happen very quickly).
5. You can freely reference any documentation on-line

Constraints

1. Only use standard library. No usage of third party download/http-client libraries that already do concurrent downloading :).
2. In case you are using Java, you can use Java 11+ HTTP client for convenience.
3. Optimize by using SINGLE thread pool or go-routine pool. ie keep overall bound on concurrency.

Interfaces

GO Interface You can make your own interface if you are un-happy with the below

```go
package dwnd

// DownloadClient is a simple HTTP Downloader that supports
// concurrent downloading of files.
type DownloadClient interface {
    // Download downloads the given urls into the configured downloadDir using
    // DownloadOptions.NumParallelPartsand DownloadOptions.NumParallelFiles
    // appropriately and returns paths to the locally downloaded files or error.
    Download(fileUrls ...string) (downloadPaths []string, err error)
}

type DownloadOptions struct {
    DownloadDir string
    // NumParallelParts represents max number of go-routines used to download diff parts
    // of a large file simultaneously.
    // Only use when file size > 10MB.
    NumParallelParts int
    // NumParallelFiles represents number of go-routines used to download
    // different files simultaneously.
    NumParallelFiles int
}

func NewDownloadClient(opts DownloadOptions) DownloadClient {
    //TODO: Instantiate impl of DownloadClient
    return nil
}
```

Java Interface You can make your own interface if you are un-happy with the below

```java
// --- DownloadOptions.java
package io.dwnd;

import java.nio.file.Path;

public record DownloadOptions(Path downloadDir, int numParallelParts, int numParallelFiles)
}

// ---- Downloader.java
package io.dwnd;

import java.nio.file.Path;
import java.util.List;

public interface Downloader {
```

```
    List<Path> download(List<String> urls) throws DownloadException;
    default Downloader NewDownloader(DownloadOptions options) {
        return new DownloaderImpl(); // CODE ME
    }
}

// -----  DownloadException

package io.dwnd;

public class DownloadException extends Exception {
    // FILL ME.
}
```