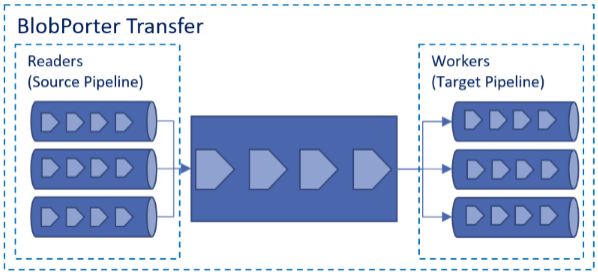
BlobPorter Documentation

Jesus Aguilar

May 14, 2019

Contents:

BlobPorter is a data transfer tool for Azure Blob Storage that maximizes throughput through concurrent reads and writes that can scale up and down independently.



CHAPTER 1

Getting Started

1.1 Linux

Download, extract and set permissions

```
wget -O bp_linux.tar.gz https://github.com/Azure/blobporter/releases/download/v0.6.20/
↪bp_linux.tar.gz
tar -xvf bp_linux.tar.gz linux_amd64/blobporter
chmod +x ~/linux_amd64/blobporter
cd ~/linux_amd64
```

Set environment variables:

```
export ACCOUNT_NAME=<STORAGE_ACCOUNT_NAME>
export ACCOUNT_KEY=<STORAGE_ACCOUNT_KEY>
```

Note: You can also set these values via options

1.2 Windows

Download BlobPorter.exe

Set environment variables (if using the command prompt):

```
set ACCOUNT_NAME=<STORAGE_ACCOUNT_NAME>
set ACCOUNT_KEY=<STORAGE_ACCOUNT_KEY>
```

Set environment variables (if using PowerShell):

```
$env:ACCOUNT_NAME="<STORAGE_ACCOUNT_NAME>"
$env:ACCOUNT_KEY="<STORAGE_ACCOUNT_KEY>"
```

1.3 Command Options

-f, --source_file (string) URL, Azure Blob or S3 Endpoint, file or files (e.g. /data/*.gz) to upload.

-c, --container_name (string) Container name (e.g. mycontainer).

-n, --blob_name (string) Blob name (e.g. myblob.txt) or prefix for download scenarios.

-g, --concurrent_workers (int) Number of go-routines for parallel upload.

-r, --concurrent_readers (int) Number of go-routines for parallel reading of the input.

-b, --block_size (string) Desired size of each blob block.

Can be specified as an integer byte count or integer suffixed with B, KB or MB.

-a, --account_name (string) Storage account name (e.g. mystorage).

Can also be specified via the ACCOUNT_NAME environment variable.

-k, --account_key (string) Storage account key string.

Can also be specified via the ACCOUNT_KEY environment variable.

-s, --http_timeout (int) HTTP client timeout in seconds. Default value is 600s.

-t, --transfer_type (string) Defines the source and target of the transfer.

Must be one of

```
file-blockblob, file-pageblob, http-blockblob,
http-pageblob, blob-file, pageblock-file (alias␣
↪of blob-file),
blockblob-file (alias of blob-file), http-file,
blob-pageblob, blob-blockblob, s3-pageblob and s3-
↪blockblob.
```

-m, --compute_blockmd5 **(bool) If set, block level MD5 has will be computed and included** as a header when the block is sent to blob storage.

Default is false.

-q, --quiet_mode (bool) If set, the progress indicator is not displayed.

The files to transfer, errors, warnings and transfer completion summary is still displayed.

-x, --files_per_transfer (int) Number of files in a batch transfer. Default is 500.

-h, --handles_per_file (int) Number of open handles for concurrent reads and writes per file. Default is 2.

-i, --remove_directories (bool) If set blobs are downloaded or uploaded without keeping the directory structure of the source.

Not applicable when the source is a HTTP endpoint.

-o, --read_token_exp (int) Expiration in minutes of the read-only access token that will be generated to read from S3 or Azure Blob sources.

Default value: 360.

-l, --transfer_status (string) Transfer status file location. If set, blobporter will use this file to track the status of the transfer.

By referencing the same status file after a failure, the transfer will skip files already transferred.

If the transfer is successful a summary will be appended.

-u, --endpoint_suffix Endpoint suffix to be used for blob sources or targets.

Default is blob.core.windows.net

CHAPTER 2

Examples

2.1 Upload to Azure Block Blob Storage

Single file upload:

```
./blobporter -f /datadrive/myfile.tar -c mycontainer
```

Note: BlobPorter will create the container if it doesn't exist.

Upload all files that match the pattern:

```
./blobporter -f "/datadrive/*.tar" -c mycontainer
```

You can also specify a list of files or patterns explicitly:

```
./blobporter -f "/datadrive/*.tar" -f "/datadrive/readme.md" -f "/datadrive/log" -c␣
↪mycontainer
```

If you want to rename the target file name, you can use the -n option:

```
./blobporter -f /datadrive/f1.tar  -n newname.tar -c mycontainer
```

2.2 Upload to Azure Page Blob Storage

Same as uploading to block blob storage, but with the transfer definiton (-t option) set to `file-pageblob`. For example, a single file upload to page blob:

```
./blobporter -f /datadrive/mydisk.vhd -c mycontainer -n mydisk.vhd -t file-pageblob
```

Note: The file size and block size must be a multiple of 512 (bytes). The maximum block size is 4MB.

2.3 Transfer data from S3 to Azure Storage

You can transfer data from an S3 compatible endpoint.

First you must specify the access and secret keys via environment variables.

```
export S3_ACCESS_KEY=<YOUR ACCESS KEY>
export S3_SECRET_KEY=<YOUR_SECRET_KEY>
```

Then you can specify an S3 URI, with the following format:

```
[HOST]/[BUCKET]/[PREFIX]
```

For example:

```
./blobporter -f s3://mys3api.com/mybucket/mydata -c froms3 -t s3-blockblob
```

Note: BlobPorter will upload the data as it downloads it from the source. The performance of the transfer will be constraint by the bandwidth of the host running BlobPorter. Consider running this type of transfer from a Virtual Machine running in the same Azure region as the target or the source.

2.4 Transfer data between Azure Storage accounts, containers and blob types

First, you must set the account key of the source storage account.

```
export SOURCE_ACCOUNT_KEY=<YOUR  KEY>
```

Then you can specify the URI of the source. The source could be a page, block or append blob. Prefixes are supported.

```
./blobporter -f "https://mysourceaccount.blob.core.windows.net/container/myblob" -c␣
↪mycontainer -t blob-blockblob
```

Note: BlobPorter won't use the bandwidth of the host for this scenario. BlobPorter uses the Put Block From URL API.

2.5 Transfer from an HTTP/HTTPS source to Azure Blob Storage

To block blob storage:

```
./blobporter -f "http://mysource/file.bam" -c mycontainer -n file.bam -t http-
↪blockblob
```

To page blob storage:

```
./blobporter -f "http://mysource/my.vhd" -c mycontainer -n my.vhd -t http-pageblob
```

Note: BlobPorter will upload the data as it downloads it from the source. The performance of the transfer will be constraint by the bandwidth of the host running BlobPorter. Consider running this type of transfer from a Virtual Machine running in the same Azure region as the target or the source.

2.6 Download from Azure Blob Storage

For download scenarios, the source can be a page, append or block blob:

```
./blobporter -c mycontainer -n file.bam -t blob-file
```

You can use the -n option to specify a prefix. All blobs that match the prefix will be downloaded.

The following will download all blobs in the container that start with f:

```
./blobporter -c mycontainer -n f -t blob-file
```

Without the -n option all files in the container will be downloaded.

```
./blobporter -c mycontainer -t blob-file
```

By default files are downloaded keeping the same directory structure as the remote source.

If you want download to the same directory where you are running blobporter set -i option.

```
./blobporter -c mycontainer -t blob-file -i
```

For scenarios where blob endpoint is from a soverign cloud (e.g. China and Germany), Azure Gov or Azure Stack, you can specify the fully qualified domain name:

```
./blobporter -f "https://[ACCOUNT_NAME].[BASE_URL]/[CONTAINER_NAME]/[PREFIX]" -t blob-
↪file
```

And the source account key, must be set via an environment variable.

```
export SOURCE_ACCOUNT_KEY=<YOUR  KEY>
```

2.7 Download a file from a HTTP source

```
./blobporter -f "http://mysource/file.bam" -n /datadrive/file.bam -t http-file
```

Note: The ACCOUNT_NAME and ACCOUNT_KEY environment variables are not required.

CHAPTER 3

Resumable Transfers

BlobPorter supports resumable transfers. This feature is enabled when the -l option is set with the path where the transfer status file will be created. In case of failure, if the same status file is specified, BlobPorter will skip files that were already transferred.

```
blobporter -f "manyfiles/*" -c many -l mylog
```

For each file in the transfer two entries will be created in the status file. One when file is queued (Started) and another when the file is successfully transferred (Completed).

The log entries are created with the following tab-delimited format:

```
[Timestamp] [Filename] [Status (1:Started,2:Completed,3:Ignored)] [Size] [Transfer ID␣
↪]
```

The following output from a transfer status file shows that three files were included in the transfer: **file10** , **file11** and **file15** . However, only **file10** and **file11** were successfully transferred. For **file15** the output indicates that it was queued but there's no second entry confirming that it was transferred successfully (status = 2).

```
2018-03-05T03:31:13.034245807Z  file10  1       104857600       938520246_mylog
2018-03-05T03:31:13.034390509Z  file11  1       104857600       938520246_mylog
2018-03-05T03:31:13.034437109Z  file15  1       104857600       938520246_mylog
2018-03-05T03:31:25.232572306Z  file10  2       104857600       938520246_mylog
2018-03-05T03:31:25.591239355Z  file11  2       104857600       938520246_mylog
```

Consider the previous scenario and assume that the transfer was executed again. In this case, the status file shows two new entries for **file15** in a new transfer (the transfer ID is different) which is an indication that this time the file was transferred successfully.

```
2018-03-05T03:31:13.034245807Z  file10  1       104857600       938520246_mylog
2018-03-05T03:31:13.034390509Z  file11  1       104857600       938520246_mylog
2018-03-05T03:31:13.034437109Z  file15  1       104857600       938520246_mylog
2018-03-05T03:31:25.232572306Z  file10  2       104857600       938520246_mylog
2018-03-05T03:31:25.591239355Z  file11  2       104857600       938520246_mylog
```

(continues on next page)

(continued from previous page)

```
2018-03-05T03:54:33.660161772Z  file15  1       104857600       495675852_mylog
2018-03-05T03:54:34.579295059Z  file15  2       104857600       495675852_mylog
```

Finally, since the process completed successfully, a summary is appended to the transfer status file.

```
-------------------------------------------------------
Transfer Completed-----------------------------------------
Start Summary----------------------------------------------
Last Transfer ID:495675852_mylog
Date:Mon Mar  5 03:54:34 UTC 2018
File:file15     Size:104857600  TID:495675852_mylog
File:file10     Size:104857600  TID:938520246_mylog
File:file11     Size:104857600  TID:938520246_mylog
Transferred Files:3     Total Size:314572800
End Summary------------------------------------------------
```

CHAPTER 4

Performance Considerations

4.1 Best Practices

- By default, BlobPorter creates 5 readers and 8 workers for each core on the computer. You can overwrite these values by using the options -r (number of readers) and -g (number of workers). When overriding these options there are few considerations:
  - If during the transfer the buffer level is constant at 000%, workers could be waiting for data. Consider increasing the number of readers. If the level is 100% the opposite applies; increasing the number of workers could help.
  - Each reader or worker correlates to one goroutine. Goroutines are lightweight and a Go program can create a high number of goroutines, however, there's a point where the overhead of context switching impacts overall performance. Increase these values in small increments, e.g. 5.
- For transfers from fast disks (SSD) or HTTP sources reducing the number readers or workers could provide better performance than the default values. Reduce these values if you want to minimize resource utilization. Lowering these numbers reduces contention and the likelihood of experiencing throttling conditions.
- Transfers can be batched. Each batch transfer will concurrently read and transfer up to 500 files (default value) from the source. The batch size can be modified using the -x option.
- Blobs smaller than the block size are transferred in a single operation. With relatively small files (<32MB) performance may be better if you set a block size equal to the size of the files. Setting the number of workers and readers to the number of files could yield performance gains.
- The block size can have a significant memory impact if set to a large value (e.g. 100MB). For large files, use a block size that is close to the minimum required for the transfer and reduce the number of workers (g option).

  The following table list the maximum file size for typical block sizes.

Block Size (MB)	Max File Size (GB)
8	400
16	800
32	1600
64	3200

4.2 Performance Measurement Mode

BlobPorter has a performance measurement mode that uploads random data generated in memory and measures the performance of the operation without the impact of disk i/o. The performance mode for uploads could help you identify the potential upper limit of the data throughput that your environment can provide.

For example, the following command will upload 10 x 10GB files to a storage account.

```
blobporter -f "1GB:10" -c perft -t perf-blockblob
```

You can also use this mode to see if increasing (or decreasing) the number of workers/writers (-g option) will have a potential impact.

```
blobporter -f "1GB:10" -c perft -t perf-blockblob -g 20
```

Similarly, for downloads, you can simulate downloading data from a storage account without writing to disk. This mode could also help you fine-tune the number of readers (-r option) and get an idea of the maximum download throughput.

The following command downloads the data previously uploaded.

```
export SRC_ACCOUNT_KEY=$ACCOUNT_KEY
```

```
blobporter -f "https://myaccount.blob.core.windows.net/perft" -t blob-perf``
```

Then you can download the file to disk.

```
blobporter -c perft -t blob-file
```

The performance difference will provide with a base-line of the impact of disk i/o.