On Effectiveness of Compressed File Transfers to/from the Cloud: An Experimental Evaluation

Armen Dzhagaryan and Aleksandar Milenković
Electrical and Computer Engineering Department, The University of Alabama in Huntsville, Huntsville, AL, U.S.A.

Keywords: Cloud Computing, Lossless Compression, Energy Efficiency, Performance Evaluation, Cost Efficiency.

Abstract: An increasing reliance on cloud and distributed processing of scientific and big data in commercial, academic, and government institutions necessitate new approaches to optimize file transfers. Lossless data compression and decompression is essential in improving the overall effectiveness of file transfers between edge devices and the cloud by increasing communication throughput, reducing connection latency, making effective use of cloud storage, and reducing costs. This paper experimentally evaluates effectiveness of common and emerging general-purpose compression utilities for file transfers between edge devices and the cloud. The utilities are evaluated in terms of throughput and costs during representative file transfers between a workstation and the cloud, while varying LAN network conditions. The results show that the optimal compressed transfer modes improve both upload and download throughputs. For uploads, the peak improvements range from 5.16 to 25.6 times relative to uncompressed file uploads, and from 1.33 to 17.4 times relative to the default compressed uploads. For downloads, the peak improvements range from 3.82 to 19.57 times relative to uncompressed downloads, and from 1.8 to 13.8 times relative to the default compressed downloads. In addition, the best performing compressed transfer modes reduce the costs related to cloud computing.

1 INTRODUCTION

Cloud and distributed computing are used by commercial, academic, and government institutions to process and analyze big data generated from multiple sources (Chen et al., 2014). For example, big data applications may analyze web or users online activities generated from public web and social media to make predictions in areas such as, product evaluation and market characterization. An increased collaboration and data exchange in scientific communities result in an increased communication between edge devices and the cloud. Finally, emerging machine learning applications require transfer of a large amount of training data to and from the cloud.

The cloud computing poses new challenges associated with costs, security, and storage (Abadi, 2009; Abu-Libdeh et al., 2010; Sboner et al., 2011). Whereas the use of cloud services opens up new opportunities in computing and reduces the costs of owning, operating, and maintaining hardware, the costs associated with the use of cloud services can eventually become prohibitively high for small research and industry organizations. Providers of cloud platforms charge utilization fees for using computing resources and data transfer fees for data transfers either to or from the cloud (Microsoft, 2017; Amazon, 2017; Google, 2017a). The specific cloud instance configuration (disk space, tenancy type, network priority, and computational power) and location of the cloud instance determine the final utilization and transfer fees associated with each instance type. Consequently, optimizing data transfers between edge devices and the cloud is very important for a range of cloud computing applications.

The importance of lossless compression in optimizing throughputs and costs in cloud computing has been recognized by both industry and academia. Lossless data compression is currently being used to reduce the required bandwidth during file downloads (e.g., software repositories, distributed storage) and to speed up web page loads in browsers. Several studies showed that lossless compressed transfers between edge devices and the cloud can increase throughput and reduce overall cloud costs during transfers (Sboner et al., 2011; Nicolae, 2010; Bonfield and Mahoney, 2013; Bicer et al., 2013). Another study is focused on tradeoffs of compression in reducing time and energy of IO in Hadoop (Chen et al., 2010). Studies focusing on transfers over WLAN and cellular inter-

Dzhagaryan, A. and Milenković, A.
On Effectiveness of Compressed File Transfers to/from the Cloud: An Experimental Evaluation.
DOI: 10.5220/0006905800350046
In *Proceedings of the 8th International Joint Conference on Pervasive and Embedded Computing and Communication Systems (PECCS 2018)*, pages 35-46
ISBN: 978-989-758-322-3

faces on mobile platforms showed that not a single combination of a compression utility and a compression level performs the best for all file transfers and network conditions (Barr and Asanović, 2003; Barr and Asanović, 2006; Milenković et al., 2013; Dzhagaryan et al., 2013; Dzhagaryan and Milenković, 2015; Dzhagaryan et al., 2015). Using measurement-based observations, a couple of studies introduce run-time techniques for deciding whether to use compressed transfer or not (Krintz and Sucu, 2006; Nicolae, 2010; Harnik et al., 2013). Our previous works on embedded and mobile devices introduced analytical models for estimating effective throughput and energy efficiency for uncompressed and compressed file transfers (Dzhagaryan and Milenković, 2016), as well as a run-time technique using those models (Dzhagaryan and Milenković, 2017). The importance of lossless data compression is further underscored with the recent development of new compression algorithms, such as Apples *lzfse* (Apple, 2017), Facebooks *zstd* (Facebook, 2017), and Googles *brotli* (**?**). All three are designed to replace the existing compression algorithms, such as *zlib*, by employing newer encoding methods (e.g., FSE (Cyan, 2013) in *zstd* and *lzfse*) and focusing on optimizations to improve performance and energy efficiency. Three compression utilities were designed specifically for mobile devices and application repositories (*lzfse*), data centers (*zstd*), and web compression (*brotli*).

In this paper, we perform a comparative, measurement-based study of general-purpose compression utilities with the focus on performance and cost effectiveness in file transfers between a workstation and the cloud. We perform compressed file transfers to/from the cloud instances residing in North Virginia and Tokyo for a range of input files and network conditions. Measured throughputs and data transfer cloud costs of each selected utility are compared to the throughputs and costs of uncompressed file transfers and the default compressed file transfers that use *gzip* with *-6* compression level.

We find that the compressed file transfers significantly improve effective throughputs and reduce the cloud costs. The level of reduction depends on (i) characteristics of an input file; (ii) network connection; (iii) cloud pricing for data transfers and selected instance type and location, and (iv) performance characteristics of the workstation that initiates the transfer and the cloud instance performing compression. Depending on the file type, selecting optimal compressed uploads improves effective throughputs up to 5.16 25.6 times relative to raw uploads, and 1.33 17.4 times relative to the default compressed uploads with *gzip -6*. Optimal compressed downloads improve download throughputs up to 3.82 29.57 times relative to raw downloads, and 1.8 13.8 times relative to the default downloads with *gzip -6*.

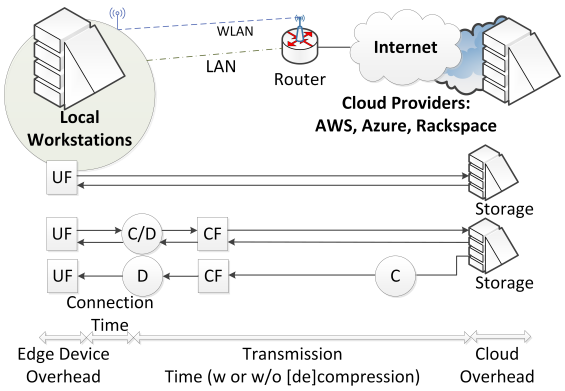

Figure 1: File uploads and downloads in cloud computing.

We find that optimal compressed file upload modes reduce the costs associated with data transfers up to 83 - 99.6% relative to uncompressed uploads and up to 25.4 - 90.3% relative to the default compressed uploads. Optimal compressed file download modes lower the download costs relative to uncompressed downloads up to 86.1 - 93%, and up to 42.9 - 83.7% relative to the default compressed downloads.

Optimal data uploads that improve effective throughputs and costs are achieved using *pzstd/zstd* on connections with higher network throughputs, and using *lbzip2/pbzip2* on connections with lower network throughputs. Optimal data downloads that improve effective throughputs are achieved using upper levels of *pzstd/zstd* on connections with higher network throughputs, and using upper levels of *lbzip2/pbzip2* and *pxz/xz* on connections with lower network throughputs. The cost-effective data downloads are achieved using *lbzip2/pbzip2* and *pixz/pxz*, as well as using *bzip2* and *xz* in case of low network throughput.

The rest of the paper is organized as follows. Section 2 presents background and motivation. Section 3 describes experimental evaluation. Section 4 presents the results of the evaluation. Section 5 surveys related work. Finally, Section 6 draws conclusions and summarizes our findings.

2 BACKGROUND

2.1 Transfers in the Cloud Computing

Figure 1 illustrates main steps in file transfers initiated from a workstation. A data file can be transferred to or from the cloud uncompressed or compressed. For

uncompressed transfers, an uncompressed file *(UF)* is uploaded or downloaded over a network directly. For compressed uploads, the uncompressed file is first compressed locally, and then a compressed file *(CF)* is uploaded to the cloud over the network. For compressed downloads, a compressed version of the requested file is downloaded from the cloud, and then the compressed file is decompressed locally. When a compressed version of the requested file is not available in the cloud, the file is compressed in the cloud, and then the compressed file is downloaded and decompressed on the workstation. The file transfer and local (de)compression tasks on the workstation and the cloud are often overlapped. Compressed transfers utilize one of available compression utilities. Each compression utility supports a range of compression levels that allow us to trade off speed for compression ratio: lower levels favor speed, whereas higher levels favor compression ratio.

We consider six common compression utilities, as well as their parallel implementations, listed in Table 1. We have selected relatively fast *gzip*, *lzop*, *lz4*, and *zstd* utilities, as well as *bzip2* and *xz*, which provide high compression ratios. As modern workstations include multicore processors, we also consider *pigz*, *pbzip2*, *lbzip2*, *pxz*, *pixz*, and *pzstd*, which are parallel versions of *gzip*, *bzip2*, *xz*, and *zstd*, respectively. For each utility, we consider at least three compression levels: low (1), medium (6), and high (9, 12, 19).

To evaluate the performance of individual compression (utility, level) pairs, we measure the total time and the total costs of doing the file transfer to and from the cloud. The total time, in general, includes the following components: (i) workstation overhead time, (ii) network connection setup time, (iii) file transmission time, and (iv) cloud overhead time. Instead of reporting execution time, the effective throughputs, *Th.CUP [Th.CDW]*, are defined as the ratio between the uncompressed file size in megabytes and the total time needed to complete the file transfer. The total costs depend on the utilization fees and transfer fees charged by the cloud provider and heavily depend on the effective throughput and achievable compression ratio.

Table 1: Lossless compression utilities.

Utility	Levels	Version	Notes
gzip	1-9 (6)	1.6	DEFLATE (Ziv-Lempel, Huffman)
lzop	1-9 (6)	1.03	LZO (Lempel-Ziv-Oberhumer)
lz4	1-9 (6)	r128	LZ4
bzip2	1-9 (6)	1.0.6	RLE+BWT+MTF+Huffman
xz	0-9 (6)	5.1.0	LZMA2
zstd	1-19 (3)	1.1.4	Huff0+FSE+LZ77
pigz	1-9 (6)	2.3.3	Parallel implementation of gzip
pbzip2	1-9 (9)	1.1.12	Parallel implementation of bzip2
lbzip2	1-9 (9)	1.1.12	Parallel implementation of bzip2
pxz	0-9 (9)	5.1.0	Parallel implementation of xz
pixz	0-9 (9)	1.0.6	Parallel implementation of xz
pzstd	1-19 (3)	1.1.4	Parallel implementation of zstd

2.2 AWS Cloud Computing Platform

To facilitate cloud computing, we use Amazons Elastic Cloud Computing (EC2) that provides computational and storage resources across a number of global locations. Cloud instances in North Virginia and Tokyo (Table 2) are created using a compute and memory optimized *m4.xlarge* Linux instance type. The selected locations of the clouds have a direct impact on the network throughput and time to set up the network connection from the local workstation. For the geographically close instance in North Virginia, the observed network throughput is 18.4 MB/s for uploads and 80.6 MB/s for downloads, and the time to set up a secure connection (ssh) is less than 1 s. For the cloud instance in Tokyo, the observed network throughput is 8.4 MB/s for uploads and 10.5 MB/s for downloads, and the time to set up a connection is 5 s. In both cases, the network throughputs can be further capped by a high-intensity network traffic and a low network priority.

Per AWS EC2 on-demand pricing model, the user is charged the utilization fee (per hour) based on selected instance type and location, and the download-out fee charged per GB of data transfer. For example, the utilization fees for the selected North Virginia and Tokyo clouds are $0.239 and $0.348 per hour, respectively. The download-out fees are $0.09 and $0.14 per GB, respectively. File transfers to and between the cloud instances in the same region are free.

The use of the cloud can be divided into two categories, a service-based and compute-based on-demand usage. A service-based usage refers to always on cloud instance, with primary examples such as web and file servers. In this case, the utilization cost becomes an invariable constant. A compute-based usage refers to on-demand cloud instance, switched on only for servicing computational tasks submitted by the workstation. In this case, the total utilization cost can be optimized by using AWS scripts to start an instance, upload inputs, execute job on the cloud, download results, and finally shutdown the instance.

Table 2: AWS EC2 Cloud Instances.

ID	Instance Type	Location	Distance (miles)	Utilization & Download Cost
NV	m4.xlarge	North Virginia	600	$0.239 / $0.09
TK	m4.xlarge	Tokyo	7,000	$0.348 / $0.14

With on-demand cloud usage, the total cost of uploading a file to the cloud *(TC.UUP [TC.CUP])* depends on the file size, US, utilization fee, $Util_{FEE}$ (expressed in \$/s), and the effective upload throughput, as shown in (1). Because AWS does not charge for data uploads, the optimal costs are achieved by transfer modes with the highest upload throughputs. The total costs of downloading a file from the cloud *(TC.UDW [TC.CDW])* depend on the file size, utilization fee, download-out fee, $Dout_{FEE}$ (expressed in \$/MB), effective download throughput, and the compression ratio, CR, as shown in (2). To compare efficiency of cloud utilization by each (utility, level) pair, we also defined cost efficiency for upload *(CE.UUP [CE.CUP])*, as shown in (3), and cost efficiency for download *(CE.UDW [CE.CDW])*, as shown in (4). Cost efficiency is expressed in megabytes per dollar (MB/\$).

$$TC.UUP[TC.CUP] = US \cdot (\frac{Util_{FEE}/3600}{Th.UUP[Th.CUP]}) \quad (1)$$

$$TC.UDW[TC.CDW] = US \cdot (\frac{Util_{FEE}/3600}{1 + Th.DW \cdot \frac{T.SC}{US}} + \frac{Dout_{FEE}/1024}{CR}) \quad (2)$$

$$CE.UUP[CE.CUP] = \frac{US}{TC.UUP[TC.CUP]} \quad (3)$$

$$CE.UDW[CE.CDW] = \frac{US}{TC.UDW[TC.CDW]} \quad (4)$$

3 EXPERIMENTAL EVALUATION

Experimental evaluation involves performing compressed file transfers initiated from a workstation to and from the cloud instances in North Virginia (NV) and Tokyo (TK). The *m4.xlarge* cloud instance type used in both cases features 2.3 GHz Intel Xeon E5-2686 with 4 virtual cores, 16 GB of RAM memory and 750 Mbps of dedicated bandwidth. The workstation initiating transfers is a Dell PowerEdge T110 II featuring quad-core Intel Xeon E3-1240 v2 and 8 GB of RAM memory.

Each transfer mode is augmented to measure the total transfer times and thus the effective throughputs. The total costs for using the cloud are calculated using measured effective throughputs, local compression ratios of each (utility, level) pair, and cloud fees. To measure effectiveness of each utility in terms of speedup, the effective throughputs achieved by the (utility, level) pair, *Th.CUP [Th.CDW]*, are compared to the effective throughputs of uncompressed transfers, *Th.UUP [Th.UDW]*, and the default compressed transfers with *gzip -6*, *Th.CUP(gzip -6) [Th.CDW(gzip -6)]*. To measure cost savings, the percentage of saved cost is calculated relative to the cost of doing uncompressed transfers as shown in (3) and relative to the cost of doing default compression transfers as shown in (4). To capture effect of network, the experiments are repeated using uncapped, 5 MB/s, and 2 MB/s LAN network connections.

$$\frac{TC.UUP - TC.CUP]}{TC.UUP}[\frac{TC.UDW - TC.CUP}{TC.UDW}] \quad (5)$$

$$\frac{TC.CUP(gzip-6) - TC.CUP]}{TC.CUP(gzip-6)} [\frac{TC.CDW(gzip-6) - TC.CUP}{TC.CDW(gzip-6)}] \quad (6)$$

Table 3: Datasets to characterize local compression.

ID	Dataset	Type	Size GB	Notes
D0	netcdf	bin	7.2	Relief data in NetCDF format
D1	seq.all	txt	6.7	DNA sequence data in text
D2	wikipages	xml	8.1	Wikipedia archived pages

Datasets. A diverse collection of files representative of data transfers to and from the cloud is compiled as shown in Table 3. The datasets consist of text files containing DNA sequence data from the UniGene (NCBI, 2017) project, binary files containing Earths surface relief data collected by the National Oceanic and Atmospheric Administration (NOAA) (Amante, C. and B. W. Eakins, 2009), and XML files containing web data with human-readable English text collected from archived pages on Wikipedia (Wikipedia, 2017). Each dataset includes 20 files varying in size. Files range in size from 1.61 MB to 1.87 GB with the average and median being 592.36 MB and 333.29 MB for *wikipages* files, 263.77 MB and 157.38 MB for *netcdf* files, and 216.5 MB and 121.76 MB for *seq.all* files.

4 RESULTS

4.1 Local Compression Ratios and Throughputs

Table 4 shows the compression ratios for all considered sequential compression utilities and levels for the input datasets. The compression ratios are averaged from 20 points to 2 points that encompass files smaller than 100 MB and larger than 100 MB in size. The parallel utilities achieve identical or very similar compression ratios as their sequential counterparts. *lzop* has the lowest compression ratios, followed by

Table 4: Compression ratios.

	gzip			lzop			lz4			bzip2			xz			zstd		
ID	-1	-6	-9	-1	-6	-9	-1	-6	-9	-1	-6	-9	-1	-6	-9	-1	-6	-19
Less than 100 MB																		
D0	2.50	2.43	2.59	1.40	1.40	2.26	1.69	2.13	2.13	4.31	6.11	6.47	4.59	4.21	6.06	2.56	3.39	4.60
D1	2.25	2.17	2.30	1.30	1.30	2.01	1.53	1.90	1.90	3.80	5.27	5.57	3.97	3.66	5.16	2.26	2.95	3.94
D2	2.28	2.20	2.34	1.31	1.31	2.04	1.55	1.93	1.93	3.88	5.40	5.71	4.06	3.74	5.29	2.30	3.01	4.03
More than 100 MB																		
D0	2.40	2.32	2.49	1.41	1.42	2.25	1.61	2.09	2.09	3.83	4.77	4.93	3.95	3.55	4.59	2.38	2.91	3.88
D1	2.39	2.31	2.49	1.41	1.41	2.25	1.60	2.09	2.09	3.81	4.74	4.90	3.93	3.53	4.55	2.37	2.89	3.86
D2	2.42	2.40	2.58	1.44	1.45	2.32	1.64	2.15	2.16	3.86	4.79	4.94	3.97	3.62	4.67	2.45	2.98	3.96

Table 5: Local compression throughputs.

	pigz			lzop			lz4			lbzip2			pixz			pzstd		
ID	-1	-6	-9	-1	-6	-9	-1	-6	-9	-1	-6	-9	-1	-6	-9	-1	-6	-19
Less than 100 MB																		
D0	188	45.7	17.9	148	142	3.22	128	31.9	30.0	71.4	81.7	78.8	43.3	3.60	1.10	468	100	4.10
D1	254	67.5	20.9	212	212	2.04	190	22.6	6.03	61.1	59.2	57.5	67.0	3.30	1.16	633	195	3.63
D2	209	85.9	2.09	113	111	6.86	110	26.5	21.1	55.4	52.0	48.6	42.0	3.76	1.20	399	96.4	3.70
More than 100 MB																		
D0	168	32.1	14.5	157	141	2.55	99.5	28.3	27.7	68.9	81.2	80.4	36.9	4.96	2.95	457	99.6	7.12
D1	262	72.5	26.0	211	211	2.57	188	24.0	7.10	64.3	63.7	62.3	61.7	5.92	3.04	928	254	7.75
D2	239	93.1	2.20	123	121	7.30	122	28.5	22.5	63.6	60.7	58.0	45.7	6.68	3.70	587	116	7.79

Table 6: Local decompression throughputs.

	pigz			lzop			lz4			lbzip2			pixz			pzstd		
ID	-1	-6	-9	-1	-6	-9	-1	-6	-9	-1	-6	-9	-1	-6	-9	-1	-6	-19
Less than 100 MB																		
D0	83.0	93.8	95.2	120	122	155	143	153	151	196	109	90.2	43.3	3.60	1.10	188	233	287
D1	95.2	137	135	150	150	239	198	235	269	189	118	107	45.2	68.7	85.6	424	429	463
D2	78.6	89.4	86.3	95.4	94.2	123	138	168	160	165	88.6	78.2	31.6	38.2	47.5	183	258	279
More than 100 MB																		
D0	76.6	91.0	99.6	122	123	169	138	183	182	221	98.5	80.5	36.9	4.96	2.95	193	243	300
D1	107	157	161	166	168	325	246	349	372	240	133	116	43.0	70.4	93.5	583	631	809
D2	101	108	109	121	119	157	188	244	243	204	113	96.7	33.9	40.1	50.1	267	332	393

lz4, *zstd*, and *gzip*. The highest compression ratios are achieved by *bzip2* and *xz*. For all utilities, higher compression levels result in higher compression ratios. Compression ratios range from 1 (*lzop -1*) to 10.12 (*bzip2 -9*) for *netcdf*, 2.26 (*lzop -1*) to 14.39 (*xz -9*) for *seq.all*, and 1.19 (*lzop -9*) to 7.73 (*xz -9*) for *wikipages*.

Table 5 and Table 6 show the local compression and decompression throughputs. The throughputs of the parallel utilities are shown in place of the sequential counterparts because they achieve higher throughputs. Both *lbzip2* and *pixz* outperform *pbzip2* and *pxz* through better parallel implementation. The highest local compression throughputs are achieved by *pzstd -1* reaching up to 746.05, 1557.58, and 1092.03 MB/s for *netcdf*, *seq.all*, and *wikipages* respectively. The lowest compression throughputs are achieved by *lbzip2*, *pbzip2*, *pixz*, *pxz*, and their sequential counterparts. The highest local decompression throughputs are achieved by *pzstd* and *zstd* with high compression levels, reaching up to 609.75, 1530.51, and 810.02 MB/s for *netcdf*, *seq.all*, and *wikipages*, respectively.

After comparing the compression ratios and taking into account better compression and decompression throughputs, we find that *zstd* and *pzstd* can effectively replace both *gzip* and *pigz* for all upload and download transfers considered in this evaluation.

4.2 Compressed Upload/Download Throughputs

Uploads. Figure 2 shows the effective throughput for compressed uploads of all selected files *(netcdf, seq.all, wikipages)* to the NV instance with the uncapped network connection. The plots show that the effective throughput saturates for the larger files, approaching the product of the compression ratio, *CR*, and the network connection upload throughput, *Th.UUP*. For top performing utilities and levels, the effective throughput increases with increase of input file size, ranging from 5.02 to 171.43 MB/s (*lbzip2, zstd, pzstd*). Availability of compute resources on the workstation (quad-core processor and 8 GB of RAM memory) allows parallel utilities to significantly increase the effective throughput of compressed uploads when using higher compression levels over their

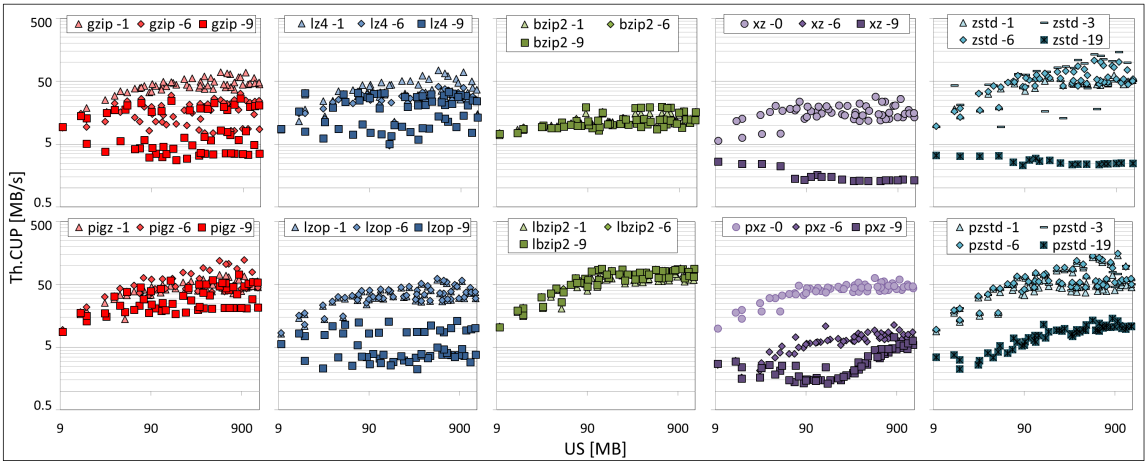

Figure 2: Effective throughput of compressed uploads (Uncapped, North Virginia).

Table 7: Maximum upload throughput speedups and optimal modes (North Virginia & Tokyo).

	Uncapped				5 MB/s				2 MB/s			
Clouds	NV		TK		NV		TK		NV		TK	
TH.CUP/TH.UUP												
D0	8.84	lbzip2	9.56	lbzip2	7.38	lbzip2	9.07	lbzip2/pbzip2	8.93	lbzip2	5.69	lbzip2/pbzip2
D1	5.28	pzstd	10.2	lbzip2/pbzip2	10.3	pzstd2/zstd	8.74	pzstd/zstd	10.5	zstd	9.92	pzstd/zstd
D2	9.44	pzstd/lbzip2	25.6	lbzip2	6.63	pbzip2/zstd	22.5	pzstd/zstd	5.92	pbzip2	5.16	pzstd/zstd
TH.CUP/TH.CUP(gzip -6)												
D0	10.3	lbzip2	4.69	lbzip2	4.76	lbzip2	7.41	lbzip2/pbzip2	4.26	lbzip2	2.1	lbzip2/pbzip2
D1	7.73	pzstd	3.08	lbzip2/pbzip2	2.19	pzstd2/zstd	2.03	pzstd/zstd	1.35	zstd	1.33	pzstd/zstd
D2	8.84	pzstd/lbzip2	1.47	lbzip2	3.74	pbzip2/zstd	17.4	pzstd/zstd	2.11	pbzip2	1.39	pzstd/zstd

sequential counterparts. In some cases (e.g., *lbzip2* and *pzstd*), higher compression levels outperform lower compression levels. For example, the maximum throughput achieved by the best performing *zstd* pair (*-3*) is 145.34 MB/s, while the maximum throughput achieved by the best performing *pzstd* pair (*-9*) is 171.43 MB/s. This observation is deviation from a similar study done on the mobile devices (Milenković et al., 2013; Dzhagaryan et al., 2013; Dzhagaryan et al., 2015), where lower levels consistently performed better on upload. For compressed uploads of all selected files to the TK instance with the uncapped network connection, the effective throughput ranges from 0.80 to 58.60 MB/s, similarly approaching the product of the compression ratio and the network connection upload throughput.

For uploads to the NV instance with the uncapped network connection, the optimal (utility, level) pairs are *lbzip2 -9* for *netcdf*, *pzstd -9* for *seq.all*, and *pzstd -9* for small and *lbzip2 -9* for large files in *wikipages*. High network throughput, low connection time, and abundance of computer power in the workstation favor utilities with solid throughputs and compression ratios. Files with higher compressibility (e.g., *netcdf*, and *wikipages*) and larger sizes favor *lbzip2*. For transfers with the 5 MB/s connection, the optimal pair for *netcdf* remains unchanged, whereas *seq.all* and *wikipages* achieve high throughputs with *zstd* and *pbzip2* for larger files. For transfers with the 2 MB/s connection, the optimal pairs are *lbzip2 -9* for *netcdf*, *zstd -9* for *seq.all*, and *pbzip2 -9* for *wikipages*.

For transfers to the TK instance that has lower network throughput and a larger connection time than the NV instance, the highest effective throughput is achieved by utilities with high compression ratios (e.g., *pbzip2*, *lbzip2*, *pxz*). For transfers with uncapped network connection, the optimal modes are *lbzip2 -9* for *netcdf*, upper levels of *pbzip2* and *lbzip2* for *seq.all*, and *lbzip2 -6* and *-9* for *wikipages*. For transfers with 5 MB/s an d 2 MB/s network, a slower *pbzip2 -9* is selected more frequently over *lbzip2 -9* for netcdf, while for *seq.all* and *wikipages*, optimal throughputs are achieved with upper levels of *zstd* and *pzstd* (*-9, -12*).

Table 7 shows optimal (utility, level) pairs and the maximum throughput speedups relative to the uncompressed and default compressed uploads to the NV and TK instances. In both cases, speedups are greater at higher network throughputs. Relative to uncompressed uploads to the NV instance, the maximum speedups range from 5.28 (*netcdf*) to 9.44 (*wikipages*) on uncapped network, from 6.64 (*wikipages*) to 10.3 (*seq.all*) on 5 MB/s network, and from 5.92 (*wikipages*) to 10.5 (*seq.all*) on 2 MB/s network. Rela-

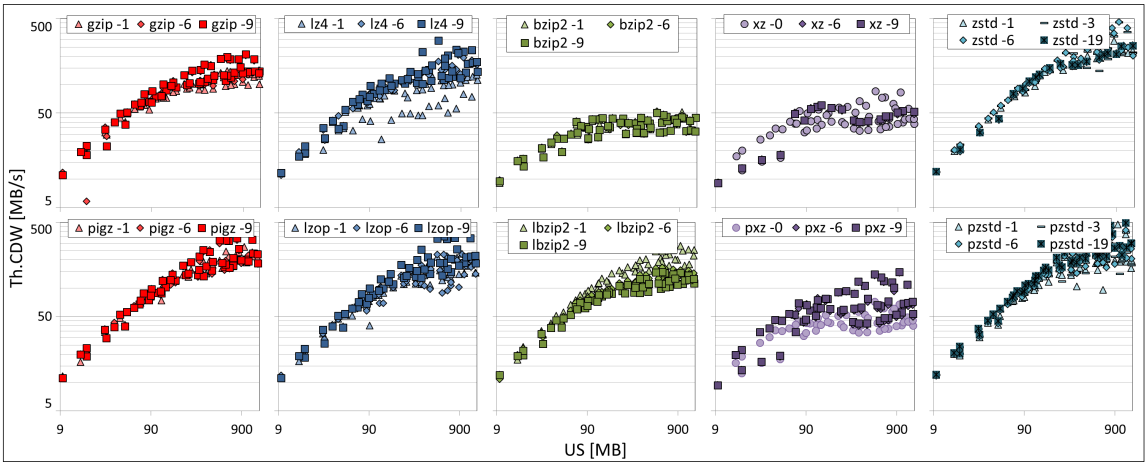

Figure 3: Effective throughput of compressed downloads (Uncapped, North Virginia).

Table 8: Maximum download throughput speedups and optimal modes (North Virginia & Tokyo).

	Uncapped				5 MB/s				2 MB/s			
Clouds	NV		TK		NV		TK		NV		TK	
TH.CDW/TH.UDW												
D0	3.82	pzstd	4.64	pbzip2	7.65	lbzip2	4.74	bzip2/pbzip2	8.98	lbzip2	6.14	bzip2/pbzip2
D1	7.87	pzstd	9.37	xz	14.01	xz	19.57	xz	14.23	xz	13.2	xz
D2	5.31	zstd/pzstd	5.76	xz	7.28	xz	6.26	xz	7.43	xz	6.48	xz
TH.CDW/TH.CDW(gzip -6)												
D0	3.61	pzstd	6.78	pbzip2	2.68	lbzip2	4.43	bzip2/pbzip2	2.85	lbzip2	13.0	bzip2/pbzip2
D1	2.94	pzstd	4.66	xz	1.8	xz	13.8	xz	1.82	xz	2.52	xz
D2	2.66	zstd/pzstd	2.91	xz	1.84	xz	2.27	xz	1.87	xz	5.6	xz

tive to uncompressed uploads to the TK instance, the maximum speedups range from 9.56 (*netcdf*) to 25.6 (*wikipages*), from 8.74 (*seq.all*) to 22.5 (*wikipages*), and from 5.16 (*wikipages*) to 9.92 (*seq.all*) on uncapped, 5 MB/s, and 2 MB/s network respectively. Relative to default compressed uploads to the NV instance, the maximum speedups range from 8.84 to 10.3 (uncapped), from 2.19 to 4.76 (5 MB/s), and from 2.11 to 4.26 (2 MB/s) respectively. Relative to the default compressed uploads to the TK instance, the maximum speedups range from 1.47 to 4.9 (uncapped), 2.03 to 17.4 (5 MB/s), and from 1.33 to 2.1 (2 MB/s).

Downloads. Figure 3 shows the effective throughput for compressed downloads of all selected files *(netcdf, seq.all, wikipages)* from the NV instance with the uncapped network connection. Due to high local decompression throughputs, the effective throughputs reach higher levels than for uploads and saturate similarly across most compression levels within each utility. For top performing utilities and levels, the effective throughput increases with increase of input file size, ranging from 5.66 MB/s to 579.58 MB/s (*zstd, pzstd*). For compressed downloads of all selected files to the TK instance with the uncapped network connection, the effective throughput ranges from 0.86 to 93.54 MB/s. Limiting the network throughput on both NV and TK instance similarly limits maximum throughput to the product of the compression ratio and the network connection download throughput.

For downloads from the NV instance with uncapped network connection, the optimal (utility, level) pairs are *pzstd -19* for *netcdf*, *pzstd -19* for *seq.all*, and *pzstd -19* for small and *zstd -19* for large files in *wikipages*. With lower network throughputs (5 MB/s and 2 MB/s), favor shifts toward slower or sequential (utility, level) pairs with stronger compression. For downloads with 5 MB/s and 2 MB/s network, the optimal pair is *lbzip2 -9* for *netcdf*, and *xz -9* for *seq.all* and *wikipages*. Similarly, for downloads from Tokyo instance slower or sequential (utility, level) pairs with stronger compression are selected (*pbzip2*, *bzip2*, *xz*). For downloads with uncapped network connection, the optimal modes are *pbzip2 -9* for *netcdf*, *xz -9* for *seq.all* and *wikipages*. For downloads with 5 MB/s and 2 MB/s networks, *pbzip2 -9* for smaller and *bzip2 -9* for larger files are selected for *netcdf*, whereas the optimal pairs for *seq.all* and *wikipages* remain unchanged.

Table 8 shows the maximum throughput speedups and optimal (utility, level) pairs when being compared to throughputs achieved with uncompressed and default compressed downloads from the NV and TK instances. In both cases, speedups are greater at the lower network throughput. Relative to uncompres-

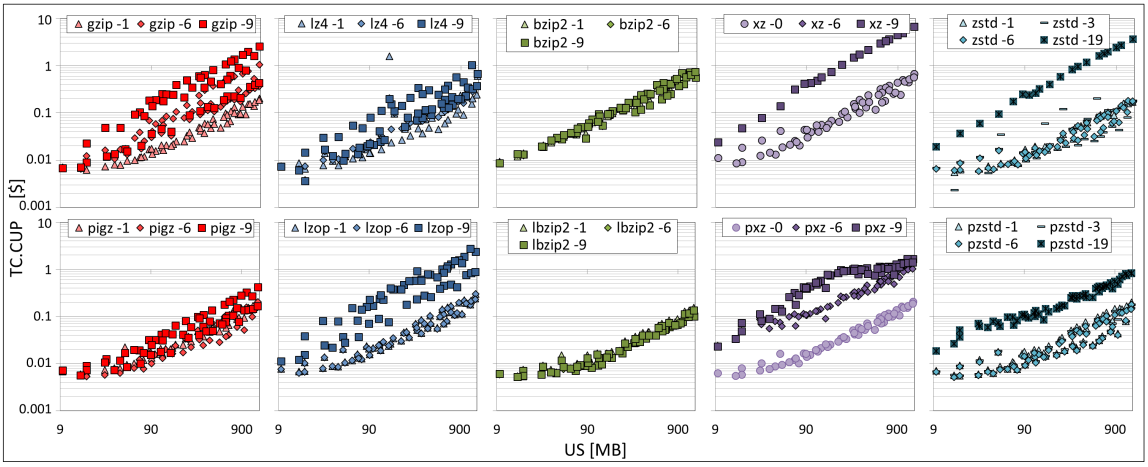

Figure 4: Total cloud cost of compressed uploads (Uncapped, North Virginia).

Table 9: Maximum upload cost savings and optimal modes (North Virginia & Tokyo).

	Uncapped				5 MB/s				2 MB/s			
Clouds	NV		TK		NV		TK		NV		TK	
(%)	Relative to TC.UUP											
D0	99.2	lbzip2	86.6	lbzip2	89.6	lbzip2	86.9	lbzip2/pbzip2	89.8	lbzip2	87.9	lbzip2/pbzip2
D1	99.6	pzstd	89.9	lbzip2/pbzip2	90.5	pzstd2/zstd	90.3	pzstd/zstd	90.5	zstd	90.4	pzstd/zstd
D2	99.2	pzstd/lbzip2	85.1	lbzip2	83.7	pbzip2/zstd	89.3	pzstd/zstd	83.3	pbzip2	83.0	pzstd/zstd
Relative to TC.CUP(gzip -6)												
D0	90.3	lbzip2	65.2	lbzip2	67.8	lbzip2	65.5	lbzip2/pbzip2	67.1	lbzip2	62.9	lbzip2/pbzip2
D1	87.1	pzstd	31.0	lbzip2/pbzip2	30.3	pzstd2/zstd	29.5	pzstd/zstd	25.8	zstd	25.4	pzstd/zstd
D2	88.7	pzstd/lbzip2	71.2	lbzip2	48.2	pbzip2/zstd	78.9	pzstd/zstd	40.3	pbzip2	33.1	pzstd/zstd

sed downloads from the NV instance, the maximum speedups range from 3.82 (*netcdf*) to 5.31 (*wikipages*) on uncapped network, from 7.28 (*wikipages*) to 14.01 (*seq.all*) on 5 MB/s network, and from 7.43 (*wikipages*) to 14.23 (*seq.all*) on 2 MB/s network. Relative to uncompressed downloads from the TK instance, the maximum speedups range from 4.64 (*netcdf*) to 9.37 (*seq.all*), from 4.74 (*netcdf*) to 19.57 (*seq.all*), and from 6.14 (*netcdf*) to 13.2 (*seq.all*) on uncapped, 5 MB/s, and 2 MB/s network. Relative to default compressed downloads from the NV instance, the maximum speedups range from 2.66 to 3.61 (uncapped), from 1.8 to 2.68 (5 MB/s), and from 1.82 to 2.85 (2 MB/s) respectively. Relative to default compressed downloads from the TK instance, the maximum speedups range from 2.97 to 6.78 (uncapped), 2.27 to 13.8 (5 MB/s), and from 2.52 to 13.0 (2 MB/s).

4.3 Compressed Upload/Download Costs

Uploads. Figure 4 shows the total cloud cost for compressed uploads of all selected files *(netcdf, seq.all, wikipages)* to the NV instance with the uncapped network connection. The plots show that the total cloud cost increases with increase of input file size, ranging from $0.0005 to $6.7119. With the cloud cost depending only on the utilization fees ($Util_{FEE}$), the derived cost efficiency has similar amount of saturation and ranking among different (utility, level) pairs as those of the effective throughput. For top performing utilities and levels, the cloud cost efficiency ranges from 93.5 to 25,821 MB/$ (*lbzip2, zstd, pzstd*). For compressed uploads of all selected files to the TK instance with the uncapped network connection, the total cloud cost ranges from $0.002 to $9.80 and the cloud cost efficiency ranges from 35.08 to 6061.95 MB/$. Limiting the network throughput on both NV and TK instance similarly increases total cost and reduces cost efficiency.

Table 9 shows the maximum cost savings when being compared to the total costs of uncompressed and to the default compressed uploads to the NV and TK instances. With no charge for uploading files to AWS instances, the (utility, level) pairs that achieve the highest throughputs achieve the highest cost savings. The maximum cost savings relative to the cost of doing uncompressed file uploads to the NV instance range from 83.3 - 90.5% (2 MB/s LAN) to 99.2 - 99.6% (uncapped). The maximum cost savings achieved with optimal compressed uploads to the TK instance are similar to those of the NV instance for

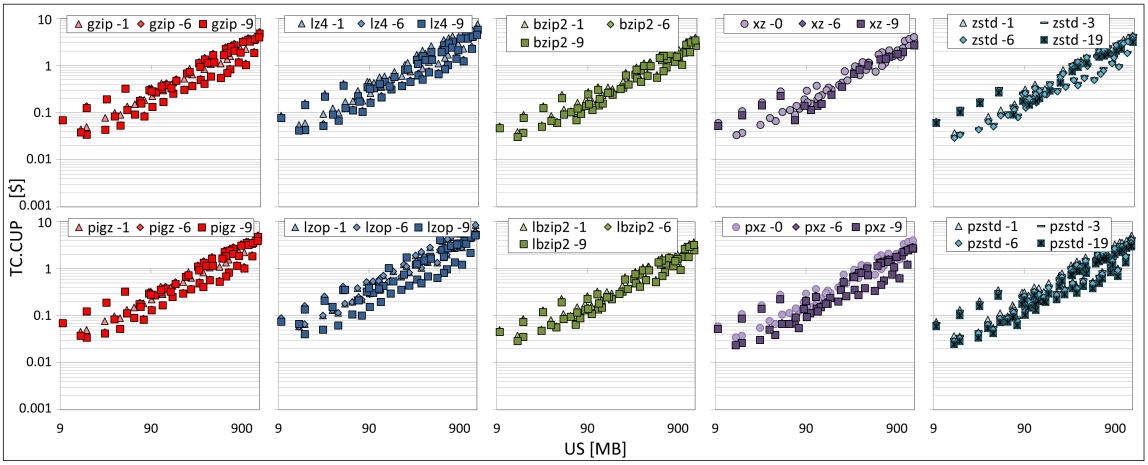

Figure 5: Total cloud cost of compressed downloads (Uncapped, North Virginia).

Table 10: Maximum download cost savings and optimal modes (North Virginia & Tokyo).

	Uncapped				5 MB/s				2 MB/s			
Clouds	NV		TK		NV		TK		NV		TK	
(%)	Relative to TC.UDW											
D0	89.1	lbzip2	86.8	pbzip2	89.7	lbzip2	86.8	pbzip2	89.8	lbzip2	88.2	pbzip2
D1	92.6	xz	92.8	xz	93.0	xz	92.8	xz	93.0	xz	92.9	xz
D2	86.3	xz	86.1	xz	86.9	xz	86.1	xz	86.9	xz	86.1	xz
Relative to TC.CDW(gzip -6)												
D0	65.6	lbzip2	68.5	pbzip2	66.9	lbzip2	68.5	pbzip2	67.1	lbzip2	83.7	pbzip2
D1	42.9	xz	50.9	xz	45.2	xz	50.9	xz	45.2	xz	67.7	xz
D2	45.7	xz	48.6	xz	47.4	xz	48.6	xz	47.5	xz	75.9	xz

2 MB/s and 5 MB/s network throughputs, but are lower with the uncapped network due to lower network throughput (85.1 - 89.9%).

The maximum cost savings relative to the cost of doing the default compressed uploads to the NV instance range from 25.8 - 65.5% (2 MB/s LAN) to 88.7 - 90.7% (uncapped). The maximum cost savings relative to the cost of the default compressed uploads to the Tokyo cloud are lower, ranging from 25.4 - 62.9% (2 MB/s LAN) to 31.0 - 71.2% (uncapped).

Downloads. Figure 5 shows the total cloud cost for compressed downloads of all selected files *(netcdf, seq.all, wikipages)* from the TK instance with the uncapped network connection. The total cloud cost ranges from $0.0089 to $8.5198. For top performing utilities and levels, the derived cloud cost efficiency ranges from 107.25 to 1522.78 MB/$ (*lbzip2, pxz*). For compressed downloads of all selected files to the TK instance with the uncapped network connection, the total cloud cost ranges from $0.0387 to $14.1553 and the cloud cost efficiency ranges from 49.99 to 949.18 MB/$. Limiting the network throughput on both NV and TK instance similarly increases total cost and reduces cost efficiency.

Table 10 shows the maximum cost savings when being compared to the total costs of uncompressed and default compression downloads from the NV and TK instances. The most cost-effective (utility, level) pairs are the ones that achieve a balance between good compression ratio and high throughput. For downloads from the NV instance, the most cost-effective (utility, level) pairs are *lbzip2 -9* for *netcdf* files, and *xz -9* for *seq.all* and *wikipages* files. For downloads from the TK instance the most cost-effective (utility, level) pairs are *pbzip2 -9* for *netcdf* files, and *xz -9* for *seq.all* and *wikipages* files, regardless of network throughput.

The maximum cost savings relative to the cost of doing uncompressed file downloads from the NV instance range from 86.9 - 93.0% (2 MB/s LAN) to 86.3 - 92.6% (uncapped). Similar cost savings are achieved by the most cost-effective compressed downloads from the TK instance. The maximum cost savings relative to the cost of doing default compressed downloads are greater at the lower network throughput in both cases. The maximum cost savings relative to the default compressed downloads from the NV instance range from 42.9 - 65.6% (uncapped) to 45.2 - 67.1% (2 MB/s LAN). With lower network throughputs and longer network setup times, higher cost savings are achieved by the most cost-effective compressed downloads from the TK instance, ranging from 48.6 - 68.5% (uncapped) to 67.7 - 83.7% (2 MB/s LAN).

Cloud cost for download depends on both utiliza-

tion fee and download-out fee. Due to fast decompression and Due to additional download-out cost, total cloud cost of download increases much more exponentially with increasing file size.

5 RELATED WORK

We are aware of several studies that investigate use of lossless data compression and decompression in improving the overall effectiveness of file transfers between edge devices and the cloud. This includes measurement-based studies that evaluated common and emerging general-purpose compression utilities, and studies that introduced either analytical models or run-time techniques for deciding whether to use compressed transfer or not. Studies have been conducted on workstations, as in this paper, and on embedded and mobile devices.

The first set of studies, most closely related to own work, focused on optimizing data transfers to and from the cloud while being initiated from a workstation. Optimizing data transfers calls for increasing communication throughput, reducing connection latency, making effective use of cloud storage, and reducing cloud costs (Abadi, 2009; Abu-Libdeh et al., 2010; Sboner et al., 2011). Measurement based study performed by Nicolae (Nicolae, 2010) compares uncompressed transfers with *lzo* and *bzip2* compressed transfers and proposes precompression of header to select either uncompressed or optimally compressed transfer. Work by Bonfield and Mahoney (Bonfield and Mahoney, 2013) compares several general purpose compression (*gzip, bzip2*) and genome specific compression utilities using DNA sequencing data. Bicer and others (Bicer et al., 2013) develop new compression utility and compare it to common compression utilities (*gzip, bzip2, LZO, LZMA*) for upload and download of scientific data. Over all, measurement-based studies showed that lossless compressed transfers can increase throughput and reduce overall cloud costs during transfers. Similar to our work, datasets with larger representative files are considered for investigations. Compared to their work, we have covered a wider range of sequential and parallel lossless compression utilities, including *pzstd/zstd* developed specifically for use in data centers (Facebook, 2017). Our experiments were conducted on two AWS EC2 Cloud instances in geographically different locations, allowing us to extract not only effective throughput, but also effective cost for a range of input files and network conditions.

The second set of studies focused on optimizing data transfers initiated from battery-powered embedded and mobile devices. There, energy-efficiency becomes a priority followed by similar goals of increasing communication throughput and reducing connection latency. The datasets in those studies are selected to better represent data transfers initiated to and from mobile devices, which included raw images, physiological data (csv), code, binary, and text. Study conducted almost a decade ago by Barr and Asanović (Barr and Asanović, 2003; Barr and Asanović, 2006) investigated energy efficiency of lossless data compression on a wireless handheld devices. Several studies introduced run-time techniques for deciding whether to use compressed transfer or not (Krintz and Sucu, 2006; Nicolae, 2010; Harnik et al., 2013). Study by Krintz and others (Krintz and Sucu, 2006) introduced technique that analyzes streaming data in-transit and makes decisions about use of compression. Works by Nicolae and Harnik (Nicolae, 2010; Harnik et al., 2013) propose techniques which rely on data pre-compression to estimated compression ratios and use it as the deciding factor in selecting optimal transfer modes.

Our past work on embedded and modern mobile devices performed extensive measurement-based evaluation of common general-purpose compression utilities (Milenković et al., 2013; Dzhagaryan et al., 2013; Dzhagaryan and Milenković, 2015; Dzhagaryan et al., 2015), introduced analytical models describing effective throughput and energy efficiency for uncompressed as well as for compressed file transfers (Dzhagaryan and Milenković, 2016), and introduced a framework for selecting an optimal transfer mode (Dzhagaryan and Milenković, 2017). Those studies have also been supported by development of environment and tools for automated measurement of energy consumption on mobile and embedded devices (Milosevic et al., 2013; Dzhagaryan et al., 2016a; Dzhagaryan et al., 2016b).

6 CONCLUSIONS

This paper presents an experimental evaluation of general-purpose compression utilities for a range of data files transferred between an edge workstation and the cloud. The goal of this evaluation is to quantify effectiveness of individual compression utilities and to provide guidelines for achieving the optimal throughput or the lowest costs. *The results of our study demonstrate that reliance on a single (utility, level) pair is not an answer for achieving optimal throughput or costs when transferring files to or from the cloud.*

Selecting the optimal compression (utility, level) pair for specific file size and type, network con-

nection, cloud fees, and workstation performance can significantly improve the effective throughput and significantly reduce cloud cost when compared to uncompressed and the default compressed file transfers. Throughput-effective upload and download modes favor utilities that offer tradeoffs between compression ratio and throughput, such as *pzstd/zstd* and *lbzip2/pbzip2*, whereas cost-effective download modes favor utilities with higher compression ratio, such as *lbzip2/pbzip2* and *pxz/xz*. These findings may guide throughput and cost optimizations of big data transfers in the cloud and encourage the development of data transfer frameworks conscientious of the existing parameters for real-time selection of optimal transfer modes.

ACKNOWLEDGEMENTS

This work has been supported by AWS Cloud Credits for Research and in part by National Science Foundation grants CNS-1205439 and CNS-1217470.

REFERENCES

Abadi, D. J. (2009). Data management in the cloud: limitations and opportunities. *IEEE Data Eng. Bull.*, 32(1):3–12.

Abu-Libdeh, H., Princehouse, L., and Weatherspoon, H. (2010). RACS: A case for cloud storage diversity. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 229–240. ACM.

Amante, C. and B. W. Eakins (2009). ETOPO1 1 arc-minute global relief model: Procedures, data sources and analysis.

Amazon (2017). Amazon web services (AWS) - cloud computing services. https://aws.amazon.com/.

Apple (2017). Data compression | apple developer documentation. https://tinyurl.com/ybwc77su.

Barr, K. and Asanović, K. (2003). Energy aware lossless data compression. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys'03)*, pages 231–244. ACM Press.

Barr, K. C. and Asanović, K. (2006). Energy-aware lossless data compression. *ACM Transactions on Computer Systems*, 24(3):250–291.

Bicer, T., Yin, J., Chiu, D., Agrawal, G., and Schuchardt, K. (2013). Integrating online compression to accelerate large-scale data analytics applications. In *2013 IEEE 27th International Symposium on Parallel Distributed Processing (IPDPS)*, pages 1205–1216.

Bonfield, J. K. and Mahoney, M. V. (2013). Compression of FASTQ and SAM format sequencing data. *PLoS ONE*, 8(3):1–10.

Chen, M., Mao, S., and Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209.

Chen, Y., Ganapathi, A., and Katz, R. H. (2010). To compress or not to compress - compute vs. IO tradeoffs for mapreduce energy efficiency. In *Proceedings of the First ACM SIGCOMM Workshop on Green Networking*, Green Networking '10, pages 23–28. ACM.

Cyan (2013). RealTime data compression: Finite state entropy - a new breed of entropy coder. https://tinyurl.com/p5ehc54.

Dzhagaryan, A. and Milenković, A. (2015). On effectiveness of lossless compression in transferring mhealth data files. In *2015 17th International Conference on E-health Networking, Application Services (HealthCom)*, pages 665–668.

Dzhagaryan, A. and Milenković, A. (2016). Models for evaluating effective throughputs for file transfers in mobile computing. In *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9.

Dzhagaryan, A. and Milenković, A. (2017). A framework for optimizing file transfers between mobile devices and the cloud. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–7.

Dzhagaryan, A., Milenković, A., and Burtscher, M. (2013). Energy efficiency of lossless data compression on a mobile device: An experimental evaluation. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 126–127.

Dzhagaryan, A., Milenković, A., and Burtscher, M. (2015). Quantifying benefits of lossless compression utilities on modern smartphones. In *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9.

Dzhagaryan, A., Milenković, A., Milosevic, M., and Jovanov, E. (2016a). An environment for automated measurement of energy consumed by mobile and embedded computing devices. *Measurement*, 94:103–118.

Dzhagaryan, A., Milenković, A., Milosevic, M., and Jovanov, E. (2016b). An environment for automated measuring of energy consumed by android mobile devices. In *Proceedings of the 6th International Joint Conference on Pervasive and Embedded Computing and Communication Systems - Volume 1: PEC,*, pages 28–39.

Facebook (2017). Zstandard - real-time data compression algorithm. https://tinyurl.com/zf4vfz6.

Google (2017a). Google cloud computing, hosting services & APIs. https://cloud.google.com/.

Google (2017b). google/brotli. https://github.com/google/brotli.

Harnik, D., Kat, R., Margalit, O., Sotnikov, D., and Traeger, A. (2013). To zip or not to zip: Effective resource usage for real-time compression. In *Proceedings of the 11th USENIX Conference on File and Storage Technologies*, FAST'13, pages 229–241. USENIX.

Krintz, C. and Sucu, S. (2006). Adaptive on-the-fly com-

pression. *IEEE Transactions on Parallel and Distributed Systems*, 17(1):15–24.

Microsoft (2017). Microsoft azure: Cloud computing platform & services. https://tinyurl.com/nmxlnae.

Milenković, A., Dzhagaryan, A., and Burtscher, M. (2013). Performance and energy consumption of lossless compression/decompression utilities on mobile computing platforms. In *2013 IEEE 21st International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 254–263.

Milosevic, M., Dzhagaryan, A., Jovanov, E., and Milenković, A. (2013). An environment for automated power measurements on mobile computing platforms. In *Proceedings of the 51st ACM Southeast Conference*, ACMSE '13, page 6. ACM.

NCBI (2017). UniGene - NCBI. http://tinyurl.com/yjnsmok.

Nicolae, B. (2010). High throughput data-compression for cloud storage. In Hameurlain, A., Morvan, F., and Tjoa, A. M., editors, *Data Management in Grid and Peer-to-Peer Systems*, number 6265 in Lecture Notes in Computer Science, pages 1–12. Springer Berlin Heidelberg.

Sboner, A., Mu, X. J., Greenbaum, D., Auerbach, R. K., and Gerstein, M. B. (2011). The real cost of sequencing: higher than you think! *Genome Biology*, 12:125.

Wikipedia (2017). enwiki dump. http://tinyurl.com/zqffp3l.