Wan Accelerators: Optimizing Network Traffic with Compression

Bartosz Agas, Marvin Germar &
Christopher Tran

Introduction

A WAN accelerator is an appliance that can maximize the services of a point-to-point(PTP) connection. It provides high quality performance using minimal resources. There are various methods that a WAN accelerator uses to optimize resources such as caching repetitive data, compression, and latency optimization. The basic approach to wan optimization lies in lossless data compression.

Data compression involves encoding data into a compact size without losing the original information. Given a set of strings, a new string can be created using data compression whose length is smaller, but contains the same information. This application is important in transferring and storing data. However, implementing lossless data compression requires time. It uses a technique that learns redundant data after initial data transmission to improve performance. Data compression begins with an encoder that represents data more concisely. The original data is compressed and will be decompressed with a decoder. As the encoder and decoder channels data between connections, a dictionary is being formed. The dictionary stores initial data, but as data transfer increases the amount of redundant data also increases. Therefore the size and time of the transmission will be smaller since the information is already stored in the dictionary. Lossless data compression analyzes data to construct a dictionary that minimizes file transfer size.

These transfer sizes are decreased because redundant data is eliminated during data transmission. One method of reduction is deduplication which removes identical information in a data packet. Instead, a reference is sent from the dictionary that represents the identical information rather than sending the actual data itself. The references are smaller so it shrinks the transfer size. There are other techniques that optimize network traffic. Caching reduces the transfer size by storing data that is repeatedly accessed in the local cache. Since transfer size is reduced the cost and space required is also reduced. As a result data transmission is more efficient.

There are different ways to implement lossless data compression. A popular algorithm is the Lempel-Ziv compression. It uses an encoder and decoder to build a dictionary. Another algorithm is Huffman coding which also uses an encoder and decoder to build a dictionary. Huffman coding will be focused to study WAN accelerator and lossless data compression.

With such foregoing means, a WAN accelerator can analyze data packets being transferred over a PTP connection and only send the critical or compressed differentials. Consider two local networks where network_1 is connected to network_2 via a PTP connection and the networks frequently send each other similar data packets such as IP headers, emails, and documents. A lot of these data packets contain repetitive data segments that do not need to utilize the resources of the PTP connection. Via the aforementioned methods, a WAN accelerator significantly improves the transfer process and minimizes monetary expenses. Companies can save time and money

required for higher PTP connections if they can use a T1 connection to do what used to require a T3 connection. This is the basis of a WAN accelerator causing interest in investigating this technology.

Literature Review

When choosing a service provider, several companies are faced with the decision as to what type of bandwidth to select for their network. These choices include a T3 line, multiple T1s, a single T1, or a fractional T1. Organizations with a large number of sites will tend to vary the speeds they offer each office based on their size and usage. By implementing a WAN accelerator, the amount of bandwidth required for the various locations goes down dramatically. This can save an organization a significant amount of money as the different services vary greatly in price. T1s typically cost between $350 and $600 with the mean price being about $500 per month. A T3 is comprised of 28 T1s and can therefore reach a couple thousand dollars per month.

Being able to make a downgrade in bandwidth without actually having any production being slowed down can save a great deal of money down the line. Consider a scenario with
Company XYZ who wishes to implement a WAN accelerator offered by Cisco (Cisco WAAS). It is an organization comprised of 50 offices of varying sizes with at least 10 users for small offices and about 200 for three large offices. Their current contracts offer bandwidths of 256kbps for small offices and 1.5mbps (T1) for the three large offices. On average, their current monthly cost, per office, is about $1,700 per month or $20,400.

After implementing Cisco WAAS their bandwidth requirements are reduced by 55%. There current average cost per month for each office is now $765 or $9,180. The total average annual cost for company XYZ prior to implementing Cisco WAAS was about $1,020,000 (20,400 * 50). After implementation it is reduced to $459,000. As you can see the drop is dramatic as the savings just keep increasing as time goes by. There are initial costs for setting up the service as well annual costs for the service itself. As the table below shows for company XYZ, the costs for the WAN optimization service cannot even compare to the savings that can be achieved over a three year span. The total three year bandwidth savings total up to about 1.4 million dollars.

As mentioned earlier, WAN optimizers also offer a speed boost as the actual data being sent out is reduced. The table below shows the drastic improvements of opening a variety of files with different sizes and how long the process would take on the initial run and the ones following it.

Total costs	Initial	Year 1	Year 2	Year 3	Total
Cisco WAAS hardware and enterprise licenses *	$498,875	$0	$0	$0	$498,875
Cisco professional services fees for planning, training, and implementation *	$3,000	$0	$0	$0	$3,000
Cisco annual support costs*	$0	$32,435	$32,435	$32,435	$97,305
Internal preparation and planning labor	$14,400	$0	$0	$0	$14,400
Total costs	$516,275	$32,435	$32,435	$32,435	$613,580

Figure 1: The table above shows the costs of implementing Cisco's WAAS solution

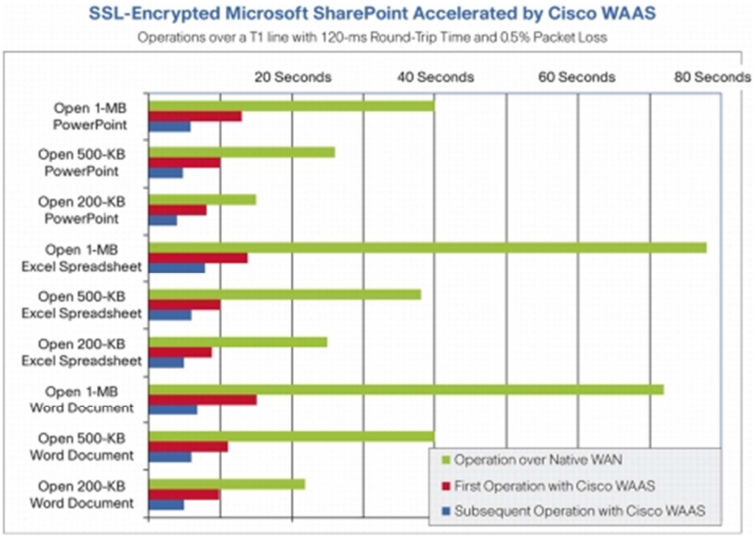


Figure 2: This is an example of how much faster files can be sent over a network with a WAN accelerator

Approach and Methodology

A WAN accelerator is a network optimizing solution that requires hardware and software to be deployed at all branches of a company. The accelerator needs to be installed in a way that it can encode data right before it leaves a network and decode it as soon as it enters a network. The WAN accelerator adapts and learns data redundancy of a network by updating a dictionary to be referenced when sending data differentials between point-to-point connections. The frequency in which dictionaries are updated can be configured depending on what the user wants. This is advantageous because customers can choose to train a dictionary for their most common files without worrying about the occasional batch of rare data streams distorting the dictionary. Below is an overview of how a WAN accelerator would operate.

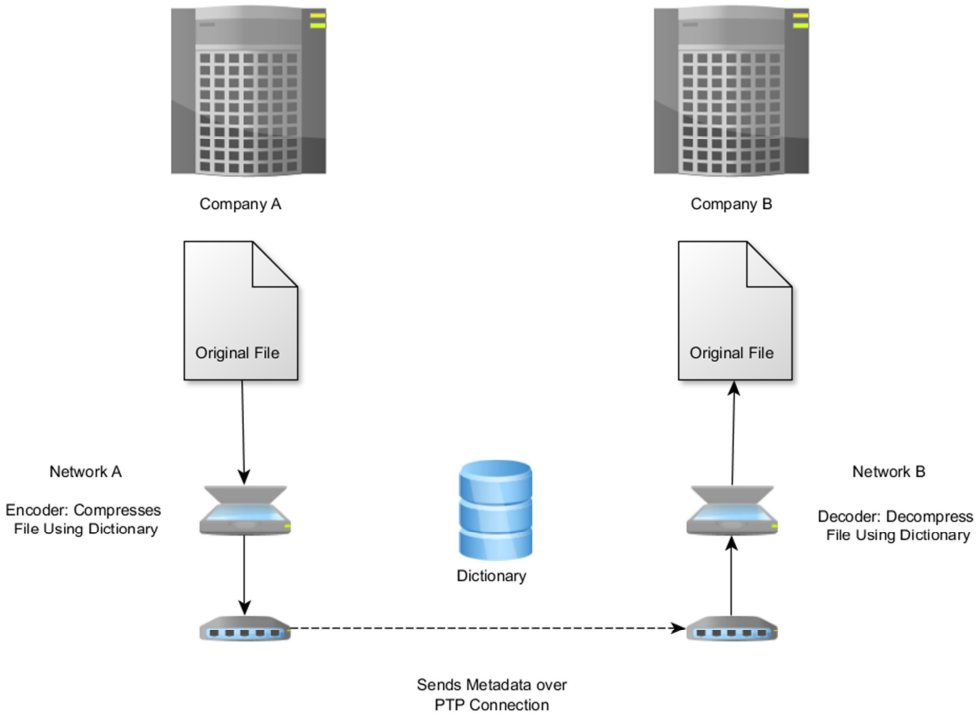


Figure 3: Visual representation of where compression takes place between companies' networks

The software driving the hardware uses a compression algorithm known as Huffman coding, a lossless data compression algorithm that creates a binary tree of nodes to reduce the number of bits to represent information during data transfer. The set of data strings that occur frequently are represented with shorter codes than those strings that are less frequent. The dictionary stores the references and rankings of what data streams show up most often for the encoder and decoder to use.

Encoder

In a general way, the encoder takes an input file, scans it, and builds a Huffman tree. Once it has the tree, it puts the tree at the front of the 'header'. It then attaches an EOF (end of file) marker at the end of the header and then writes it to the beginning of the encoded version of the file. After this, the program encodes the file using the built Huffman tree and writes it to the end of the file (starting after the EOF marker). The encoder also updates the dictionary to reflect the new probabilities of the more frequent string of data.

Decoder

The decoder is straightforward as well. The decoder scans the header portion of the file and knows where the encoded data begins through the EOF marker. Once it scans in the header, it builds its own tree and then decompresses the data based on that tree. The decoder reads the file and decompresses it using the trained dictionary from the encoder.

Dictionary

The modified Huffman code creates a binary tree with nodes that contain the symbols to be encoded. It develops an ongoing dictionary that a WAN accelerator can use. An ongoing dictionary adapts to the data being transferred by stressing the priority of data that is sent most frequently. A WAN accelerator's dictionary learns from all data transfer and is updated unlike normal compression methods that deletes it dictionary after it outputs a file. By using an updateable dictionary, a company can use a WAN accelerator to optimize their network because the dictionary is trained specifically for the company's common files. Huffman coding produces a dictionary that persist over numerous data transmissions.

Example: Given a 6 symbol alphabet with the following symbol probabilities:

A = 1, B = 2, C = 4, D = 8, E = 16, F = 32

Step 1. Combine A and B into AB with a probability of 3.
Step 2. Combine AB and C into ABC with a probability of 7.
Step 3. Combine ABC and D into ABCD with a probability of 15.
Step 4. Combine ABCD and E into ABCDE with a probability of 31.
Step 5. Combine ABCDE and F into ABCDEF with a probability of 63.
The Following tree results:

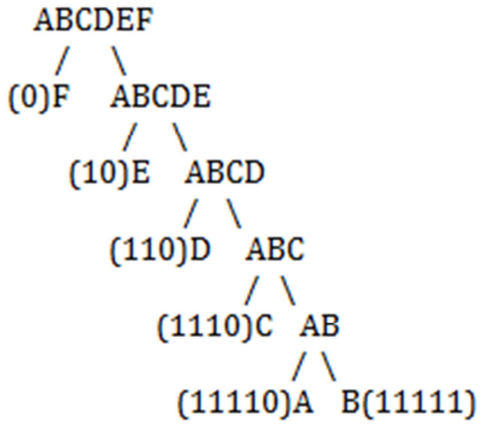


Figure 1: Huffman Tree Example

To summarize this example, because 'F' has the highest probability from the letters in this alphabet, we want to represent it with the shortest bit string possible, in this case just '0'.

For the WAN accelerator used in this project, it was coded in such a way where the user would manually choose when the dictionary should be updated or trained with a new batch of files. This would involve the user feeding the program a set of common files that the dictionary should be optimized around.

How the Huffman Algorithm reduces costs

Companies use mass amounts of bandwidth to keep their services going and fast and reliable point-to-point (PTP) connections costs a lot of money. Sending less data over the PTP pipe would be a great way to reduce costs. The Huffman Algorithm implemented in this particular WAN accelerator takes in a batch of common files supplied by the company using it, and trains the dictionary. The dictionary is updated by making a Huffman Tree as explained earlier. The strings of data that appeared most frequently now only have to be represented by smaller byte strings such as '0', '10', and '110'. The output of the encoder is sent over the network and is much smaller than the raw, original file. The decoder will read the compressed string of bytes and references the dictionary every time it sees one of these combinations and decrypt the file accordingly. It is easy to understand how using a system such as a WAN accelerator could allow companies to purchase slower PTP connection pipes and ultimately making more money.

Obstacles with Compression and WAN accelerators

A WAN accelerator can be implemented using any one of numerous compression algorithms and there is a reason that Huffman is used for this particular case. Compression applications typically read in one file, compress it, and then decompress it. A WAN accelerator on the other hand must constantly read in files and adapt by building its dictionary synchronously between the encoder and decoder. If an encoder and decoder use different dictionaries, then the output of the decoder will not produce the original file. Initially, Lempel-Ziv-Welch (LZW) was the algorithm to be used because of its high efficiency, but it would take too much time to overcome the obstacles that came with it. LZW source codes, such as the one made by James Schumacher III, are very complex and particular about how it compresses redundant byte streams. Modifying source codes like this would require so much effort that it might make sense just to implement LZW from scratch. However, coding LZW from the ground up also takes a significant amount of time due to the nature of its complex yet efficient design.

To overcome the problems with using LZW, Huffman coding was used instead as it is much easier to understand and code in a shorter amount of time. The issues that Huffman coded presented dealt with extracting its dictionary to be "ongoing" so that the WAN accelerator could adapt when it needed to. The Huffman source code used as a foundation for this project sent the probabilities and references of reoccurring data streams to the header of the encoder's output. This code was modified to take the encoder's header and store it in a common dictionary that could be used by the decoder as well. As a result of not needing to store the header, less bandwidth is needed to send the data from point-A to point-B.

Results

```
C:\Capstone\Huffman\Release>dir original
 Volume in drive C has no label.
 Volume Serial Number is ACA5-B15F

 Directory of C:\Capstone\Huffman\Release\original

04/28/2012  14:51    <DIR>          .
04/28/2012  14:51    <DIR>          ..
02/24/2004  22:22         4,397,206 bible.txt
01/20/2012  12:45        11,902,610 book.pdf
03/06/2012  04:38         1,105,920 darknet5.doc
04/22/2012  13:32           494,989 darknet5.docx
03/06/2012  03:53           758,784 NumAccSet.xls
04/22/2012  13:31           213,281 NumAccSet.xlsx
04/22/2012  13:35         2,548,503 world.jpg
03/06/2012  04:33         3,136,664 world.png
               8 File(s)     24,557,957 bytes
               2 Dir(s)  36,431,273,984 bytes free

C:\Capstone\Huffman\Release>dir compressed
 Volume in drive C has no label.
 Volume Serial Number is ACA5-B15F

 Directory of C:\Capstone\Huffman\Release\compressed

04/28/2012  14:52    <DIR>          .
04/28/2012  14:52    <DIR>          ..
04/22/2012  12:48         2,507,333 bibletxt_comp
04/22/2012  13:41        11,907,105 bookpdf_comp
04/22/2012  13:44           494,987 darknetdocx_comp
04/22/2012  13:44           837,309 darknetdoc_comp
04/22/2012  13:48           205,716 Numxlsx_comp
04/22/2012  13:47           500,500 Numxls_comp
04/22/2012  13:49         2,488,826 worldjpg_comp
04/22/2012  13:50         3,125,054 worldpng_comp
               8 File(s)     22,066,830 bytes
               2 Dir(s)  36,431,273,984 bytes free
```

Figure 4: Original file sizes with compressed file sizes.

```
C:\Capstone\Huffman\Release>dir original
 Volume in drive C has no label.
 Volume Serial Number is ACA5-B15F

 Directory of C:\Capstone\Huffman\Release\original

04/28/2012  14:51    <DIR>          .
04/28/2012  14:51    <DIR>          ..
02/24/2004  22:22         4,397,206 bible.txt
01/20/2012  12:45        11,902,610 book.pdf
03/06/2012  04:38         1,105,920 darknet5.doc
04/22/2012  13:32           494,989 darknet5.docx
03/06/2012  03:53           758,784 NumAccSet.xls
04/22/2012  13:31           213,281 NumAccSet.xlsx
04/22/2012  13:35         2,548,503 world.jpg
03/06/2012  04:33         3,136,664 world.png
               8 File(s)     24,557,957 bytes
               2 Dir(s)  36,431,269,888 bytes free

C:\Capstone\Huffman\Release>dir decompressed
 Volume in drive C has no label.
 Volume Serial Number is ACA5-B15F

 Directory of C:\Capstone\Huffman\Release\decompressed

04/28/2012  14:52    <DIR>          .
04/28/2012  14:52    <DIR>          ..
04/22/2012  12:49         4,397,206 bible_decomp.txt
04/22/2012  13:41        11,902,610 book_decomp.pdf
04/22/2012  13:44         1,105,920 darknet5_decomp.doc
04/22/2012  13:44           494,989 darknet5_decomp.docx
04/22/2012  13:47           758,784 NumAccSet_decomp.xls
04/22/2012  13:48           213,281 NumAccSet_decomp.xlsx
04/22/2012  13:49         2,548,503 world_decomp.jpg
04/22/2012  13:50         3,136,664 world_decomp.png
               8 File(s)     24,557,957 bytes
               2 Dir(s)  36,431,269,888 bytes free
```

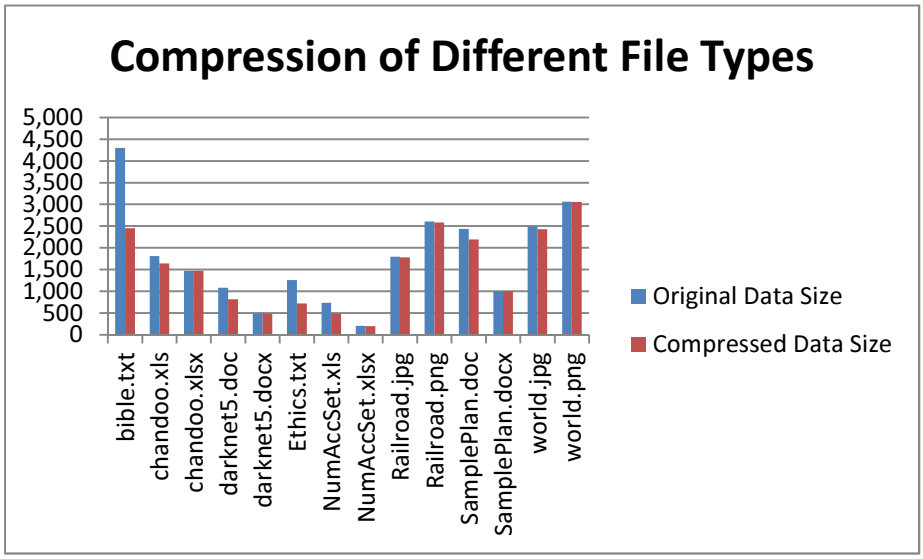
Figure 5: Original file sizes with decompressed file sizes.

The WAN accelerator implemented with the Huffman algorithm uses Michael Dipperstein's single file compression source code as a basis. After coding the encoder and decoder, preliminary test are run to determine lossless data compression efficiency. To do so, a new dictionary is built for each file type. Figure 4 and Figure 5 show implementation that uses a dictionary which optimizes each file. Overall, the compressed file sizes are smaller than the original data. However, there are compressed file sizes that increased. More tests are run to study which file types are optimized by Huffman coding. Despite the mixture of file sizes, the original file sizes are the same as the decompressed file sizes as Figure 4 illustrates. The encoder and decoder work, but not all file types are encoded concisely.

Further tests are run with an individual dictionary for each file type. Table 1 displays the efficiency of the different file types. Similar results ensued where certain files are better compressed than others. Efficiency shows the potential that each file type possess after using a Wan accelerator.

Original Data	Size	Compressed Data Size	Decompressed Data Size	Efficiency
bible.txt	4,295 KB	2,449 KB	4,295 KB	+42.9%
chandoo.xls	1,809	1,639	1,809 KB	+9.4%
chandoo.xlsx	1,473	1,472	1,473 KB	+0.1%
darknet5.doc	1,080 KB	818 KB	1,080 KB	+24.3%
darknet5.docx	484 KB	484 KB	484 KB	0%
Ethics.txt	1,261 KB	726 KB	1,261 KB	+42.4%
NumAccSet.xls	741 KB	489 KB	741 KB	+34.0%
NumAccSet.xlsx	209 KB	201 KB	209 KB	+3.8%
Railroad.jpg	1,794 KB	1,779 KB	1,794 KB	+0.8%
Railroad.png	2,606 KB	2,582 KB	2,606 KB	+0.9%
SampleBusinessPlan.doc	2,435 KB	2,198 KB	2,435 KB	+9.7%
SampleBusinessPlan.doc	1,008 KB	1,008 KB	1,008 KB	0%
world.jpg	2,489 KB	2,431 KB	2,489 KB	+2.3%
world.png	3,064 KB	3,052 KB	3,064 KB	+0.4%

Table 1: Efficiency of Different File Types

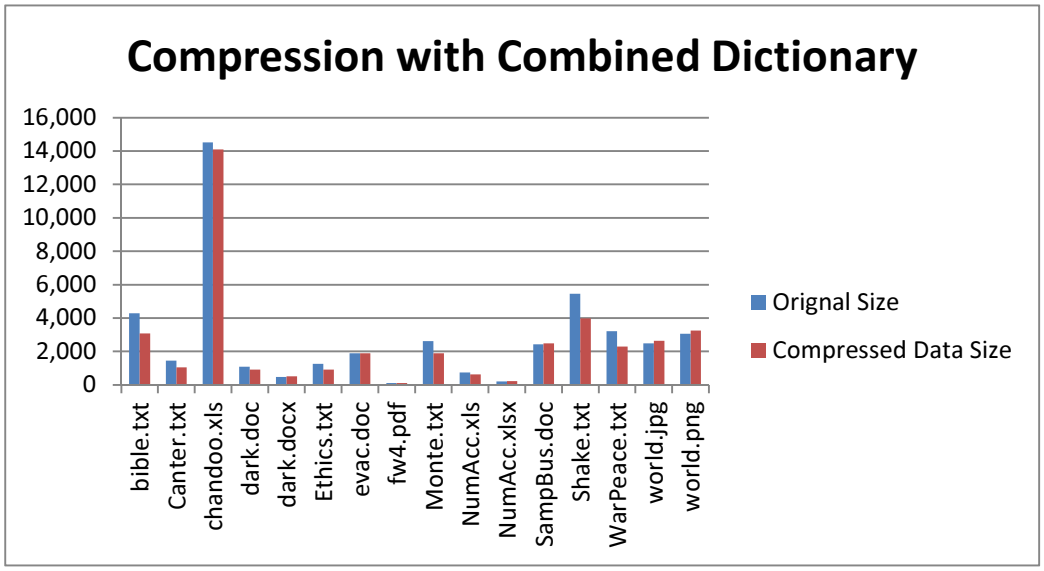


Graph 1: Compression of Different File Types

The graph above is illustrates which file types are optimized using Huffman coding. It is clear that certain file types are better compressed than others. Thus, better compression depends on the data that is being compressed. The files that are better compressed are .txt, .doc, and .xls. Other file types such as .pdf, .docx, .xlsx, .jpg, and. png, did not compress as well because they are already in compressed format. Compressing a compressed file does not compress it further.

Original Data	Size	Compressed Data Size	Efficiency
bible.txt	4,295 KB	3,071 KB	+28.5%
Canterbury.txt	1,459 KB	1,043 KB	+28.5%
chandoo.xls	14,519 KB	14,088 KB	+2.9%
darknet5.doc	1,080 KB	914 KB	+15.4%
darknet5.docx	484 KB	519 KB	-7.2%
Ethics.txt	1,261 KB	910 KB	+27.8%
evacuagationguide.doc	1,893 KB	1,885 KB	+0.4%
fw4.pdf	107 KB	105 KB	+1.9%
Monte.txt	2,624 KB	1,890 KB	+27.9%
NumAccSet.xls	741 KB	627 KB	+15.4%
NumAccSet.xlsx	209 KB	224 KB	-7.2%
SampleBusinessPlan.doc	2,435 KB	2,480 KB	-1.8%
Shakespeare.txt	5,460 KB	3,974 KB	+27.2%
WarandPeace.txt	3,212 KB	2,303 KB	+28.3%
world.jpg	2,489 KB	2,640 KB	-6.1%
world.png	3,064 KB	3,257 KB	-6.3%

Table 2: Efficiency of Different File Types

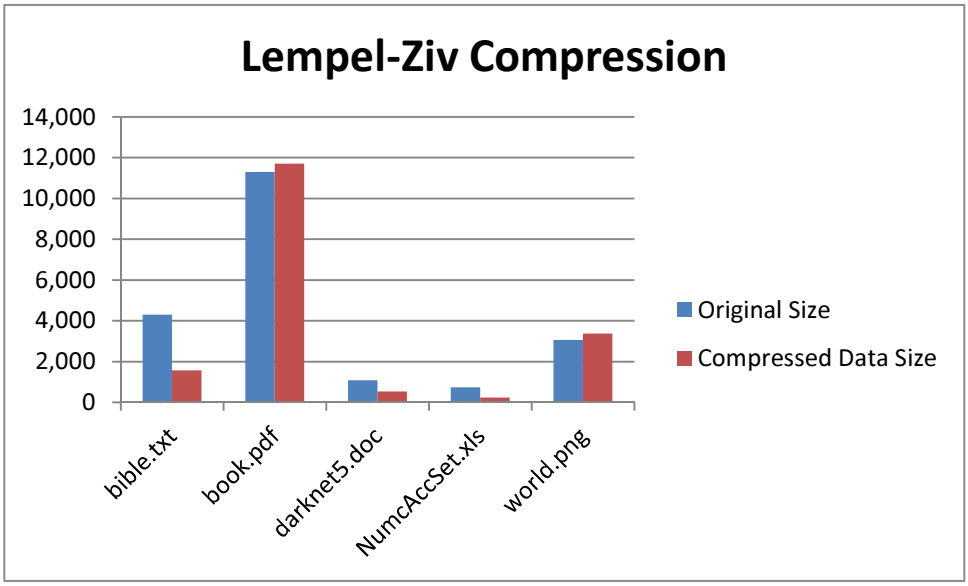


Graph 2: Compression with Combined Dictionary

Using a WAN accelerator takes times before benefits can be seen. Table 2 implements a combined dictionary for all the data types. As noted before, compressed file types do not compress effectively. However, as companies send redundant data packets with highly compressible files, a WAN accelerator can be advantageous. Overall efficiency increases drastically as Table 2 illustrate. A WAN accelerator proves to be valuable for data transmission.

Original Data	Size	Compressed Data Size	Efficiency
bible.txt	4,295 KB	1570 KB	+63.4%
book.pdf	11,300 KB	11,708 KB	-3.6%
darknet5.doc	1080 KB	527 KB	+51.2%
NumcAccSet.xls	741 KB	235 KB	68.3%
world.png	3064 KB	3379 KB	-10.3%

Table 3: Lempel-Ziv Compression



Graph 3: Lempel-Ziv Compression

Lempel-Ziv is another lossless data compression algorithm. Much like Huffman coding, documents and text files are better compressed. Other file types that are already compressed do not compress further. When using Lempel-Ziv, these file types become bigger. In comparison to Huffman coding, Lempel-Ziv does a better job compressing uncompressed data types. Nonetheless, both algorithms do not compress highly compressed data types such as .pdf and .png.

Conclusion

Optimizing a WAN accelerator is a relatively new field of study that has high potential in data storage and transmission. Based on the findings, its ability to compress files has significant impact on network traffic. With implementation, its application can expedite data transfers with reduced file sizes. This not only gives faster transmission, but it also decreases monetary expenses since less space is used. Furthermore, by minimizing the file sizes, it widens the capacity of the medium. Thus, using lossless data compression is beneficial because it optimizes available resources.

Huffman coding is one algorithm that uses lossless data compression. It uses an encoder and decoder to build a binary tree of node as a dictionary. The dictionary analyzes data transfer and adapts to redundant data. Strings that appear more frequently are referenced with shorter code than those that are less frequent. Therefore, repetitive data are accessed with smaller bits leading to smaller transmission size. Similar to Huffman coding, Lempel-Ziv compression also minimizes the data being transfer. In the study, documents and text files benefit the most from a WAN accelerator because they are highly compressible. Already compressed file types such as .pdf and .png, are not compressed further. Huffman coding and Lempel-Ziv are two out of many lossless data compression algorithm that produce efficient throughput in data storage and transmission. As results show, a WAN accelerator optimizes network traffic with compression.

References

“Cisco Wide Are Application Services SSl Acceleration Technical Overview.” *Cisco Public Information.* June 2012. Web. 11 January 2012. <http://www.cisco.com/en/US/prod/collateral/ps6712/ps6718/solution_overview_c22-532534.pdf>.

Dipperstein, Michael. "Huffman Code Discussion and Implementation."*Michael.dipperstein.com*. Michael Dipperstein. Web. 2 Apr. 2012. <http://michael.dipperstein.com>.

"Free EBooks by Project Gutenberg." *Project Gutenberg*. Web. 02 April 2012. <http://www.gutenberg.org/>.

“Huffman Coding.” *Wikipedia, The Free Encyclopedia*. Wikimedia Foundation, Inc. 04 April 2012. Web. 11 January 2012. <http://en.wikipedia.org/wiki/Huffman_coding>.

“Lempel-Ziv-Welch.” *Wikipedia, The Free Encyclopedia*. Wikimedia Foundation, Inc. 03 May 2012. Web. 11 January 2012. <http://en.wikipedia.org/wiki/Lempel-Ziv-Welch>.

“Lossless Data Compression.” *Wikipedia, The Free Encyclopedia*. Wikimedia Foundation, Inc. 30 March 2012. Web. 11 January 2012. <http://en.wikipedia.org/wiki/Lossless_data_compression>.

Nelson, Mark and Jean-Loup Gailly. *The Data Compresssion Book, Second Edition.* Cambridge, MA: IDG Books Worldwide, Inc. 1995.

Schumacher III, James. "Lempel Ziv Compression." *PlanetSourceCode*. Exhedra Solutions, Inc., 18 Apr. 2010. Web. 30 Mar. 2012.< http://www.planet-source-code.com>.

“Wan Optimization.” *Wikipedia, The Free Encyclopedia*. Wikimedia Foundation, Inc. 23 April 2012. Web. 11 January 2012. < http://en.wikipedia.org/wiki/WAN_optimization>.

“Wide Area Application Services.” Cisco. Web. 11 January 2012. <http://www.cisco.com/en/US/products/ps5680/Products_Sub_Category_Home.html#~feat-prod>.