Bookmark

# File Performance

BERCS1102016-V001600

0:00 / 8:53 1.0x

Start of transcript. Skip to the end.

SPEAKER 1: So remember the big picture

is that we start with data in databases, files, logs.

We perform extract, transform, and load

to import that data into data warehouses, which we can then

use to produce data products and for business intelligence

Download video Download transcript .srt

## Binary I/O Performance

(1/1 point)
Binary file I/O performance is typically much better (faster) than text file I/O performance?

- (●) True ✔
- ( ) False

**EXPLANATION**

Binary file I/O requires reading or writing less data and avoides the processing overheads of text file I/O.

## File Compression

(1/1 point)
Which of the following statements are true when using compression for reading and writing binary or text files:

- [x] Writing a compressed file is slower than writing an uncompressed file ✔
- [ ] Reading an uncompressed file is slower than writing a compressed file
- [ ] The time to read a compressed file is greater than the time to write a compressed file
- [x] Some compression methods are faster than others ✔
- [x] Reading and writing compressed files can be as fast as reading and writing uncompressed files ✔

✔

Note: Make sure you select all of the correct options—there may be more than one!

**EXPLANATION**

Unlike uncompressed files where the times to read and write files are identical, writing a compressed file is an expensive operation (more expensive than writing an uncomressed file) compared to reading a compressed file because reading a compressed file is a relatively low overhead operation. Some compression methods perform nearly as fast as uncompressed file access.

## TEXT FILES

Text files typically contain lines of ASCII or Unicode characters and may use a human-readable format. Because they are human-readable, text files usually have low entropy which means that they take up more space than necessary to represent the information they contain. Text files can contain binary data, but that data must be encoded in textual form. Examples of text files are Comma Separated Value files,  Python files, and even iPython notebook files.

## BINARY FILES

Binary files are files that are not text files. They consist of sequences of bytes and may contain parts that can be interpreted as text, and may contain headers, or blocks of metadata that can used by a computer program to interpret the data in the file. Examples of binary files are images, video and sound files, and computer programs.

## COMPRESSION

Compression is used to store a file's contents in a binary format that more information dense, and thus typically takes up less space than the original contents. There are two types of compression:
lossy and lossless. When compression is applied to a text file, the resulting information must be stored in a binary file.

Lossy compression by its name, means compression where information is lost or discarded. Examples are image, sound, and video compression. There a various lossy compression techniques that take advantage of the way that we see or hear content to discard extra information without being our noticing the differences (in most cases) between the original and the compressed content. Examples of lossy compression are JPEG for images, MP3 for sound files, and MPEG-4 for video files.

Lossless compression is a set of techniques that reduces the amount of space that data takes without discarding any of the data, and it is used in cases where it is important that the original and the decompressed data be identical. A lossless compression algorithm works in two steps:  first, it generates a statistical model for the input data, and second, it uses this model to map input data to bit sequences in such a way that "probable" (e.g., frequently encountered) data will produce shorter output than "improbable" data. Two examples of lossless compression algorithms are Gzip and LZ4. Each of these algorithms has tradeoffs as discussed in the lecture segment video. Examples of lossless compression are PNG for images and Dolby TrueHD for sound files.

**Note:** The processor (CPU) is used when compressing and decompressing data. As a result, if you have a limited amount of processing capability, then it may be faster to read/write uncompressed data, even though it means reading/writing more data.