

Alternate Means of “Data Compression”

Part II: Initial Data Streams

By Thomas O’Hare
Thomas@RedTile.Com

Copyright© 2004, Thomas O’Hare – All Rights Reserved.

Overview

All data is generated in some sort of “data stream”. The frequency and amount of data in that stream is dependent on the source generating the data and the mechanism used to capture the data stream. Since this series of papers deals with alternate types of data compression we will assume that at least a fair amount of data is being generated at some sort of regular intervals.

In this paper we will be dealing only with the “raw data” or what we will term the “Initial Data Stream”. We will show how this “raw data stream” can best be used and it’s interaction with alternate means of data compression.

The Concept

As with any type of data compression, you first need a set of data or a “data set” to work with. In this paper we are dealing more with “data streams” or data in a more “dynamic” fashion as opposed to “static data” or data we know will not change. Since the data is dynamic we can count on it changing at any time so we must keep this fact in the fore front of our schema at all times.

If you use this “stateless” concept as a key element in all data handling operations, you will be able to handle data whenever it changes and realize true data integrity.

The Initial Handling

Since data is in a raw streaming state we will first need to initially “capture” or “buffer” the data. There will be two types of buffers we will talk about here but the second type is the one we will focus on.

The initial buffer is raw data as it is arrives. We first need to capture this data until we know we have a complete data set and feel reasonably certain that the data is not corrupted in any way. Data must remain in the initial buffer until we are reasonably sure we have data integrity. Data must be checked here for any obvious data corruption that may have occurred in the capture process. These mechanisms are local and specific to the apparatus doing the initial data capture or buffering.

Once we are reasonably sure data is complete and intact, we can then pass this data to a secondary buffer. The secondary buffer is a means of semi-permanent storage, e.g.:

Copyright© 2004, Thomas O’Hare – All Rights Reserved.

written to a file on a computer disk drive. Once the data is stored it is now accessible for manipulation at any point in time. Data can be read immediately from this secondary type of buffer, or stored until an external mechanism is ready to use this data.

The Mechanism

The best kind of secondary or semi-permanent data storage for this type of data is what is commonly referred to as a “data base”. A database is a file that contains data that can be referenced easily by an external entity. A “flat file” database, such as a pure text file, is good storage but not very efficient. Other types of data storage utilize advanced mechanisms such as “data indexing”. When data is indexed in a file, it is much like the index at the beginning of a book that you read. In the beginning of a book there is a recognizable name, the way by which something is classified, and the location or page number where the information can be found. Indexing of a database is much like the same apparatus used in a book. We just merely make it easier and faster to find particular subject matter, or the exact location of the data we want that is stored in the file.

With this way of referencing or indexing data, we can easily do other operations such as sorting, check for duplicate values or find a particular piece of data in a very large file extremely fast.

If we now use a database as a secondary or “main buffer”, we will have data written to a semi-permanent storage area that can be accessed very quickly. In stateless or dynamic environments, it is only this data in the main storage buffer that we can be relatively sure is complete, has integrity, and is somewhat static in nature.

Conclusion

By using a database as a secondary or main buffer we have a very fast, relatively safe and very efficient means of storing data for any period of time. By using this database as a buffer for our alternate means of data compression, we have a very stable working environment for future operations.

Since the data is buffered for any period of time we deem necessary, we now also have a means by which we can perform data “auditing”. In other words, we can use data from this buffer in future operations and still be able to reference this data to be sure other operations have completed to our satisfaction.

Once data has been successfully used in other operations it can easily be “deleted”, or the data space can be “recycled” to make room for any new incoming data.

If we constantly recycle existing data space in a continuous read/write cycle we have achieved true “data buffering” in a very safe and efficient means. This database, or main data buffer, is now the cornerstone of our means of alternate data compression.

Change Log Part II:

Original Release: March 15, 2004.