An Introduction to the CloudTurbine File Structure

Matthew Miller, Cycronix

6/17/2016

Overview

CloudTurbine (CT) defines a system whereby simple, incremental files are used for all streaming data communication (e.g. from sensors, cameras, Internet-of-Things appliances). This enables utilization of standard, ubiquitous file-handling protocols and networks (e.g. FTP, Dropbox) to move streaming data between sources and viewers. This document describes the attributes of the CT filesystem and its associated Application Program Interface (API).

Figure 1 shows a simple example of files for a CT data source “CTsource”, with a single data channel “image.jpg”, collected once per second. The time folders are named with “seconds-since-epoch” (Unix) timestamps. The channel files in this example are standard, viewable JPEG images. In practice any type of data file can be streamed.

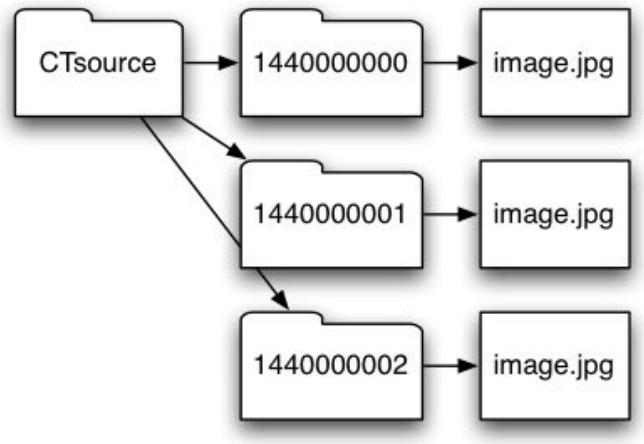

Figure 1. Simple CloudTurbine Example

The CT data file interface consists of nested folders containing time-stamped data in “channel” files according to some simple rules and conventions:

SourceFolder / TimeFolder / ChannelFile

where:

- SourceFolder can be multi-part, e.g. Source/SubSource
- TimeFolder can be multi-part, e.g. Time/SubTime
- ChannelFile can be any type of data file
- Time/Channels can be optionally zipped into time-chunks

Source Folders

Each CloudTurbine “Source” has a top-level folder with a name that identifies the origin of the data. The source name can be anything you want (organization, project, instrument, etc.). Source folders may be nested, for example a test-system folder with multiple test-run subfolders.

Multiple related Source folders typically exist side-by-side under a master CloudTurbine site folder. This facilitates locating them for data review, such as via a web browser HTTP interface such as provided by the “CTweb” application. In Figure 4 that follows, the site folder is “Cycronix1”, and two source folders are “AndroidACL” and “AndroidAV”.

Time Folders

Central to CloudTurbine are its time folders that contain the underlying time-stamped data files. These in turn may be nested as multi-part time-stamps (time/subtime), and may optionally be zipped at intervals to facilitate storage and to control the granularity (number and size) of files to be handled.

The CT API supports the following time stamp notations:

- Absolute seconds since epoch (1/1/1970, e.g. Unix format, denoted by 10 digit integer)
- Absolute milliseconds since epoch (Java/JavaScript time, denoted by 13 digit integer)
- Relative seconds or milliseconds since start of data (denoted by 1 to 9 digit integer)

Top-level time folders are always absolute time notation. Optional sub-time folders may be either absolute or relative notation. Relative sub-time folders must match the units (seconds or milliseconds) of the parent time folder. Nested relative timestamp layers add up to the total absolute time.

Figure 2 (left) shows a simple CT data source, “CTsource”, with several data channels (shown as a stack with example data values), collected once per second. This is the simplest time-notation case, and is a multi-channel variant of Figure 1 above. Figure 2 (right) shows the same case but with nested relative timestamp sub-folders. Relative timestamps are more space-efficient (shorter names), and allow an entire collection of data to be time-shifted by resetting the parent folder.

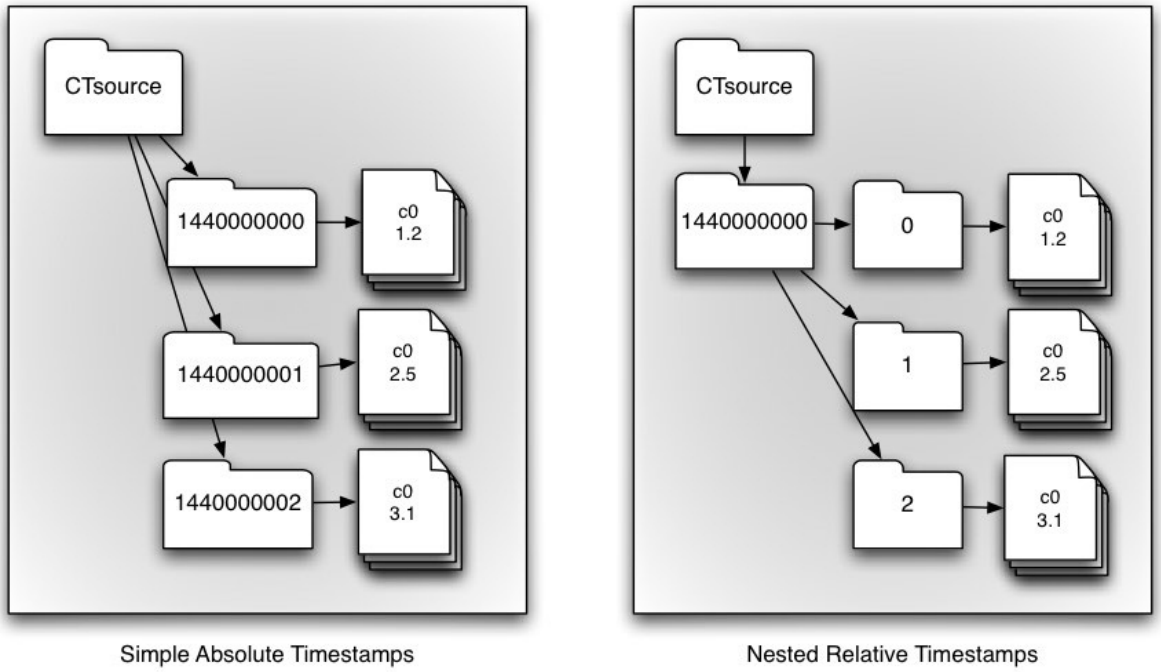

Figure 2. CloudTurbine TimeStamp Examples

Extended Notation

To facilitate efficient handling of large data sets, the CloudTurbine API supports an extended time folder hierarchy of the form:

Source / Session / Segment / Block / Point / Channel

Figure 3 shows the extended case with nested segment, block, and point time layers. This nesting allows great scalability. For example 1000 segments, 1000 blocks per segment, 1000 points per block, captures 1 billion time points with no more than 1000 items per folder. Modern file systems can readily handle 100K+ files per folder, but the file search and retrieval speed tends to bog down for such cases.

The figure also shows data compaction options:

- Zipping folders at the block level consolidates and compresses underlying data. Zipped and unzipped data work interchangeably.
- Packing multiple numerical data per file presumes linearly interpolated times, but further compacts data for efficiency.

“Packed” channel data utilize two layers of time folders to convey start and end times for a block. Timestamps for packed data points are linearly interpolated from the upper folder time to the subfolder time. Channel timestamps are shown at the bottom of Figure 3, with packed data having a time-range per block. Note that times are derived by summing up top-layer absolute time with sub-layer relative times for any given folder.

Data may be zipped and/or packed independently, with packed-and-zipped data together being the most efficient form. Note that compacted data blocks increase the “granularity” of real-time data updates, i.e. updates can only be sent to viewers after each block is compacted. Trading off segment-size with block-size can optimize large data, low latency cases.

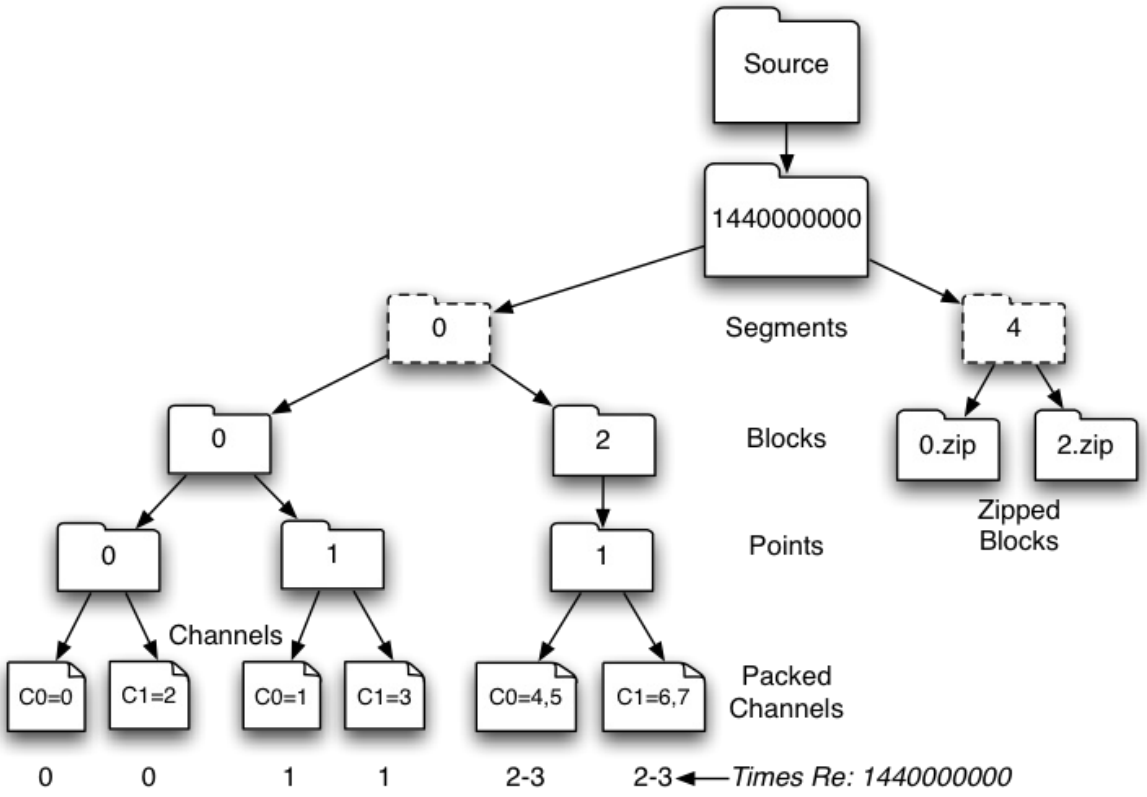

Figure 3. Nested Timestamp Hierarchy with Compaction Options

Channel Files

At the lowest level in the Source / Time / Channel hierarchy are files containing the streaming data. Each file at this level names a data “channel”, for example temperature, aircraft_altitude, or camera_image. Each channel file consists of one or multiple data samples.

CT channel files can be any type of data file with any valid file name. To facilitate recognition by data viewers, it is suggested that commonly accepted names and file suffixes be utilized, e.g. “.jpg” for JPEG images, “.txt” for text data, or “.wav” for WAV audio clips.

Numeric Data Types

CloudTurbine can stream any type of file, which in general are handled as integral byte-arrays. Certain data types, recognized via file-suffix, are parsed as numeric data blocks with linearly interpolated time stamps. The following table summarizes CT data types.

File-Suffix	Data-Type
.f32	32bit floating point
.f64	64bit floating point
.i16	16bit integer
.i32	32bit integer
.i64	64bit integer
.csv (default)	comma separated values
.pcm	audio data (header-less)
.wav	audio data (WAV header)
.txt	text (8bit ASCII)
.jpg	JPEG image

The default format (no file suffix) is ".csv", denoting comma separated text values parsed as double-precision numbers. For example, a numeric CSV file might contain "1,2,3,4,5". This may be more or less efficient than binary formats, but with advantages of human-readability and device portability.

Example Application

Figure 4 shows an example from a working application involving two sources with multiple channels each. One source has three accelerometer channels (".f32"), the other source records audio (".wav") and video (".jpg"). Data are zipped, with the accelerometer and audio data packed into efficient multi-point blocks. Data files are transparent to the user, e.g. you can click on the image.jpg or audio.wav files in your file explorer and view them directly using standard applications.

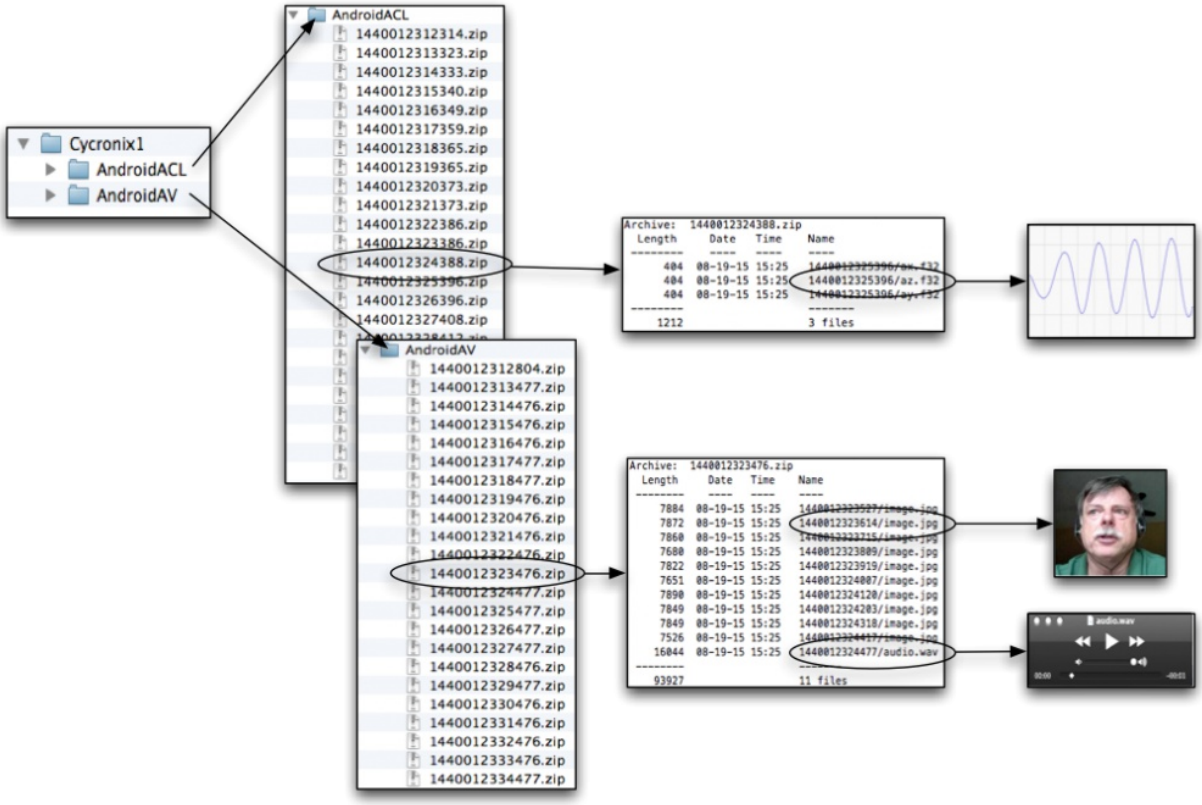

Figure 4. CloudTurbine Multi-Source File Structure Example