zfit: Compression of FIT files with Delta Coding

Mark Rages
August 2014

Introduction

FIT is a extensible binary format for arbitrary data developed by Dynastream and used by Garmin and others as a recording format for exercise and GPS data.

In a resource-constrained headunit or watch, it is desirable to compress the FIT data to fit longer recordings in limited storage. But standard data-compression algorithms require more memory than is commonly available in small, battery-powered devices.

zfit combines a delta coding preprocessor with an off-the-shelf embedded compression engine to achieve lossless compression ratios approaching 50%, requiring less than 2048 bytes of memory while doing so.

FIT Overview

Inside the FIT file format is a small header, then a series of data records, comprised of definition messages and data messages. The definition messages provide a mapping between the data messages and an external, global profile. The data messages contain the data itself. Each data message refers to a previous definition message for the details of its contents. The definition messages do not have to be repeated, allowing the storage density to approach a plain binary dump.

Each data message has a header describing which data definition it follows, which is usually a single byte followed by the data.

Heatshrink Library

The main compression is handled by the 'heatshrink' embedded compression library, obtained from `https://github.com/atomicobject/heatshrink`.

This library can be configured to encode using small amounts of memory, and without dynamic memory allocation, which makes it well suited to compression within embedded systems.

'heatshrink' is an implementation of LZSS which can be configured to run in under 100 bytes of memory. (There is a memory use vs. compression tradeoff.)

Delta Encoding

Many times, subsequent data messages do not vary much. For example, timestamps increment a second at a time. And geographical locations do not change suddenly. So a simple approach to condition the data is to subtract each field from the previous one. Then repeated values become (highly compressible) zeroes.

Delta coding doesn't change the file size in itself, but improves compression performance by 20~30%. The better the compression algorithm, the more delta encoding helps:

zip algorithm	mem	size	zip	delt+zip	ratio	deltratio
'gzip'	?	6126698	3728252	2595275	0.61	0.42
'bzip2'	?	6126698	3964347	2194608	0.65	0.36
'heatshrink/heatshrink -w 6 -l 3'	144	6126698	4821560	3598916	0.79	0.59
'heatshrink/heatshrink -w 6 -l 4'	144	6126698	4825146	3620196	0.79	0.59
'heatshrink/heatshrink -w 6 -l 5'	144	6126698	4853454	3693858	0.79	0.60
'heatshrink/heatshrink -w 7 -l 3'	272	6126698	4773710	3471387	0.78	0.57
'heatshrink/heatshrink -w 7 -l 4'	272	6126698	4771920	3478001	0.78	0.57
'heatshrink/heatshrink -w 7 -l 5'	272	6126698	4799624	3551310	0.78	0.58
'heatshrink/heatshrink -w 7 -l 6'	272	6126698	4866119	3644768	0.79	0.59
'heatshrink/heatshrink -w 9 -l 4'	1040	6126698	4678868	3286301	0.76	0.54

Heatshrink Parameter Choice

heatshrink has two parameters, window size and lookahead size. I ran the algorithm through a corpus of Garmin-encoded FIT files to choose the best parameters:

hs param	l=3	l=4	l=5	l=6	l=7	l=8	l=9	l=10
w=6	0.59	0.59	0.60	--	--	--	--	--
w=7	0.57	0.57	0.58	0.59	--	--	--	--
w=8	0.55	0.55	0.56	0.58	0.59	--	--	--
w=9	0.54	0.54	0.55	0.56	0.58	0.60	--	--
w=10	0.53	0.53	0.54	0.55	0.57	0.59	0.60	--
w=11	0.53	0.52	0.53	0.54	0.56	0.58	0.60	0.62

Based on this, w=9 and l=4 seems like a reasonable choice. (The choice of w affects the amount of memory needed for encoding.)

Delta Encoding Parameter

The delta encoder has a small buffer of memory to hold the last value of each local FIT message definition. This buffer is adjustable in size. FIT definitions beyond the buffer size will not be delta encoded, so the best choice of buffer size will be just enough to cover all message definitions. The parameter supplied is log2() of the actual buffer size.

Delta Encoding Implementation

The file 'fit_delta_encode.c' contains a basic FIT parser and delta encoder / decoder. Instead of tracking the widths of each FIT field for the subtraction, the algorithm simply subtracts individual bytes, modulo 256. This acheives about the same compression benefit while reducing external knowledge required in the parser.

Heatshrink Implementation

The subdirectories 'dynamic/' and 'static/' contain dynamically allocated zfit encoder / decoders, and a statically allocated encoder for embedded use.

The heatshrink file format does not encode the heatshrink parameters, so a three byte header is prepended to the zipped file to allow unzipping without supplying them. This gives flexibility to the encoder to adjust parameters as needed.

```
zfit header
+-----+--------+---+
| 'Z' | [w][l] | b |
+-----+--------+---+
w and l are window and lookahead parameters.
w in 4 most significant bits.
l in 4 least significant bits.
b is the buffer size parameter
```