Brotli-G Bitstream Format

Introduction

AMD proposes a GPU suitable bitstream format as a variant of a widely used compression format definition with open, implementation-friendly licensing terms, defined by a well-established organization with an excellent track record of supporting open technologies. It also provides a compression ratio that is near the top of the current state-of-the-art, with relatively moderate SIMD execution complexity demands that are expected to substantially improve over time as driver code optimizations and other focused improvements are implemented.

The AMD modifications to the original bitstream format to make it suitable for effective GPU shader implementation are planned to be contributed along the same open contribution licensing agreement that the original compression format has. The format modifications preserve the compression ratio benefits (slightly dependent on meta-block size) while allowing efficient parallelization on the GPU SIMD engines.

General description

This proposal is built upon modifications to the Brotli compression format (also known as RFC7932), a format specification well-established in web content delivery and supported in practically all common web browsers for web assets like bitmaps, JavaScript code, and other content. The use of this format may allow the use of the same mechanisms for accelerated decompression of web-delivered assets like bitmaps and other data in web browsers, either on GPU or multi-threaded CPU implementations in the future and extend the applicability of the definition beyond uses in-game scenarios to content creation, content management, and data visualization. The Brotli-G compression process can be built to be compatible with the standard decompression implementation, allowing for a backward-compatible support path for web content.

The proposal description below assumes familiarity with general compression technologies like Huffman entropy encode/decode, the LZ77 compression algorithm and extensively references information also found in the above-mentioned RFC7932 to describe the bitstream in slightly abridged definition; if the definition in this document is unclear in any way, please refer to the above specification for the exact details.

In Brotli, a compressed data set is built of a header and a variable number of meta blocks, each representing an uncompressed byte sequence up to 1677216 bytes (16 MiB) in size. The final uncompressed output is built by concatenating in sequence the processed output of each of the meta blocks. The header contains the size of a sliding window put across the uncompressed input data during the compression process. During the decompression at least that amount of uncompressed data must be retained before the current position in the stream. The sliding window size is 2^n-16 bytes (10 <= n <= 24), leading to possible window sizes between 1KiB - 16B to 16 MiB - 16B.

Each meta block is compressed using a combination of LZ77 and Huffman coding (in further text referenced as "prefix code"). The prefix codes for each meta block are independent of

previous or subsequent meta blocks. The LZ77 algorithm may use references to duplicated data in previous meta-blocks up to the sliding windows size of uncompressed data; in addition, Brotli allows for references to static prefix dictionary entries.

Each meta block consists of a meta block header that describes the representation of the compressed data part and a compressed data part, consisting of a series of commands.

A command contains information about LZ77 prefix code references. A single command may consist of the following symbolic data types:

- An insert-and-copy length (**IaC**)
- Insert length: number of unmatched literals
- Copy length: copy length of the LZ77 reference
- A series of consecutive literals (for which no match was found) **(L)**
    - These are the set of unmatched literals from the input stream right before a match was found.
    - Brotli, in addition to LZ77, allows for predefined literal references.
- A backward copy distance of the LZ77 reference **(D)**

Format: |<insert-and-copy-length> <set-of-literals> <copy-distance>|

Example: |IaC0 L0 L1 L2 D0|

A block is a set of elements from the command list of the same block category, e.g., insert-and-copy-length, literal, copy distance.

E.g., of a possible set of blocks:

|**IaC**0 **IaC**1 **IaC**2|

|**L**0 **L**1 **L**2 **L**3 **L**4 **L**5|

|**D**0 **D**1 **D**2 **D**3|

BlockSwitch

Format: <block_type>**BlockSwitch**(block_id, n)

A block switch indicates wherein the output of the block part of the context changes.

The switch may consist of prefixed block_type which can be one of the following:

- L : literal block switch
- IaC: insert-and-copy block switch
- D : distance block switch

The following parameters are used for the block switch

- block_id : Each block is assigned a numbered identification to consecutive blocks, which is used in determining the context of the elements of that block.
- n: The number of elements in the new block

Example: L**BlockSwitch**(1, 3), IaC**BlockSwitch**(1,2), D**BlockSwitch**(1, 3)

Brotli-G Bitstream

A Brotli-G compressed stream is built of a stream header, a page index, and a variable number of uncompressed or compressed pages or meta-blocks, each representing an (uncompressed) byte sequence. A meta-block may represent an element of a specific data type, e.g., a texture tile, represented by an uncompressed byte sequence of 0 up to 65536 bytes in size, for texture streaming scenario use. Other default sizes may be considered for other data types, depending on their size and use model and suitability or for overall compression efficiency.

Note: Unlike Brotli, Brotli-G does not allow LZ77 references to reference previous meta-blocks to ensure maximum decompression parallelization. The LZ77 window size may also differ for each meta-block (depending on the meta-block input size).

Each Brotli-G meta-block consists of a meta-block header, a sub-stream size offset table, 3 sets of prefix codes and a sequence of compressed bytes.

Meta-block Header:

Each compressed page or meta-block has a 1-byte header has the following fields.

2 bits:	NPOSTFIX, parameter used in distance encoding-
4 bits:	four most significant bits of NDIRECT, parameter used in distance encoding
1 bit:	IS_DELTA_ENCODED
1 bit:	reserved

Sub-stream Size Offset Table:

Meta-block header is followed by a sub-stream size offset table that stores the base size and the size offsets of the 32 sub-streams in the compressed meta-block. The sub-stream size offset table has the following fields.

NBASESIZE bits:	BASESUBSTREAMSIZE, length of the smallest sub-stream (in bytes) in the compressed meta-block, where NBASESIZE =log2(AVGSUBSTREAMSIZE) + 1. AVGSUBSTREAMSIZE is the average size of the sub-streams in the meta-block.
NDELTASIZE bits:	DETLASIZE, number of bits used to store size offsets of sub-streams, where NDELTASIZE = log2(log2(COMPRESSEDSIZE – 1) + 1) + 1. COMPRESSEDSIZE is the total compressed size of the meta-block.
32*DELTASIZE bits:	size offset of 32 sub-streams

Note: Meta-block header and size offset table are DWORD aligned.

Prefix Codes:

A compressed meta-block has 3 sets of prefix codes follow, one for each element type (insert-and-copy length, literal and copy distance). The prefix codes are also swizzled and distributed among the 32 sub-streams for efficient read and construction of the prefix tables during decompression.

Each prefix code set contains a 5-bit header. The header is always assigned to the sub-stream 0.

Brotli-G uses 3 types of prefix codes (Trivial, Simple, and Complex).

- Trivial prefix code: Contains a single symbol. Code is always 0. A symbol is stored and assigned to sub-stream 0.
- Simple prefix code: Contains less than 5 symbols. Symbols have fixed code and code lengths defined as per RFC7932. Symbols are stored in ascending order of code length, assigned to each sub-stream in order, starting from sub-stream 0.
- Complex prefix code: Contains greater than or equal to 5 symbols. Per RFC7932, first code lengths of code lengths are stored in the Brotli defined order: 1, 2, 3, 4, 0, 5, 17, 6, 16, 7, 8, 9, 10, 11, 12, 13, 14, 15, assigned to each sub-stream in round-robin fashion. Then, code lengths of the actual codes, which themselves are encoded, are assigned to each sub-stream in a similar manner. Since some symbols are used to encode runs of code lengths (e.g., symbols 16 encode runs of zeros), there may be fewer codes than the total number of code lengths.

Note: Unlike Brotli, Brotli-G uses a constant (3) number of prefix codes.

A meta-block may also be uncompressed, in which case meta-block header, offset table and prefix codes are not stored, and the uncompressed data block directly follows.

Compressed Data:

To build the Brotli-G compressed data bitstream the following steps are applied:

A) LZ77 step

LZ77 produces a list of commands: |IaC0 L0 L1 L2 D0| |IaC1 D1| |IaC2 L3 L4 D2| |IaC3 L5 D3| ...

The list of commands is appended with a custom |0,0,0| command which identifies the end of the command list: |IaC0 L0 L1 L2 D0| |IaC1 D1| |IaC2 L3 L4 D2| |IaC3 L5 D3| ... |0,0,0|

B) Swizzle step

Swizzles the groups into different sub-streams.

Bitstream 1: |IaC0 L0 L1 L2 D0| ...

Bitstream 2: |IaC1 D1| ...

Bitstream 2: |IaC2 L3 L4 D2| ...

Bitstream 3: |IaC3 L5 D3| ...

…

Note: Brotli-G uses only one block per block category per meta-block, generating three sets of prefix codes. Block switches and context tables are not required to switch between prefix codes and are therefore omitted from the compressed data stream. Brotli-G compressed stream may in principle be processed via a regular implementation of Brotli for decompression, with a slight modification to the Brotli-G bitstream format.

C) Entropy encoding step

Encodes each sub-stream using the computed prefix codes:

	4 bytes	**4 bytes**	**4 bytes**	**4 bytes**	**...**
Encoded Sub-stream 1:	B1	B2	B3	B4	...
Encoded Sub-stream 2:	B5	B6	B7	B8	...
...	...	...	...	...	

D) Serialization

Stores the header. Stores the sub-stream size offset table. Store each sub-stream end to end in order.

Output Stream: |MB Header| |BASESUBSTREAMSIZE| |DELTASIZE| |BSOFFSET1| |BSOFFSET2| … | B1 | B2 | | B3 | B4 | ... | B5 | B6 | | B7 | B8 | …

References:

1) https://github.com/google/brotli Brotli source code reference repository

2) https://datatracker.ietf.org/doc/html/rfc7932 Official IETF/W3C reference specification of Brotli

3) https://trustee.ietf.org/documents/trust-legal-provisions/ licensing and contributor requirements of IETF/W3C

4) https://datatracker.ietf.org/ipr/2396/ IETF IP disclosure related to Brotli

5) http://www.data-compression.info/Corpora/CalgaryCorpus/