Lightweight Compression Methods Achieving 120GBps and More

Piotr Przymus

Laboratoire d’Informatique Fondamentale de Marseille
Aix-Marseille University, France

GPU Technology Conference
Silicon Valley May 2017

- A lightweight compression library for GPU.
- [github.com/mis-wut/feathergpu](https://github.com/mis-wut/feathergpu)
- MIT-licensed.
- This project was partly funded by National Science Centre, decision DEC-2012/07/D/ST6/02483.

**Team**

- Krzysztof Kaczmarski
  - Warsaw University of Technology, Poland
- Piotr Przymus
  - Aix-Marseille University, France
  - Nicolaus Copernicus University in Toruń, Poland.
Lightweight compression algorithms favours compression and decompression speed over compression ratio.

- Improved data transfer:
  - Disk ↔ RAM ↔ GPU.
  - GPU ↔ GPU:
    - exchange of already compressed data,
    - compress → transfer → decompress.

- Lower memory footprint:
  - Less disk space used.
  - Less RAM used.
  - Less GPU memory used.

- Improved internal memory access:
  - In some cases improved internal GPU memory access.
Lightweight compression on GPU – motivation

Lightweight compression algorithms favours compression and decompression speed over compression ratio.

- Improved data transfer:
  - Disk ↔ RAM ↔ GPU.
  - GPU ↔ GPU:
    - exchange of already compressed data,
    - compress → transfer → decompress.

- Lower memory footprint:
  - Less disk space used.
  - Less RAM used.
  - Less GPU memory used.

- Improved internal memory access:
  - In some cases improved internal GPU memory access.
Lightweight compression on GPU – motivation

Lightweight compression algorithms favours compression and decompression speed over compression ratio.

- Improved data transfer:
  - Disk ↔ RAM ↔ GPU.
  - GPU ↔ GPU:
    - exchange of already compressed data,
    - compress → transfer → decompress.

- Lower memory footprint:
  - Less disk space used.
  - Less RAM used.
  - Less GPU memory used.

- Improved internal memory access:
  - In some cases improved internal GPU memory access.
Lightweight compression algorithms favour compression and decompression speed over compression ratio.

- **Improved data transfer:**
  - Disk ↔ RAM ↔ GPU.
  - GPU ↔ GPU:
    - exchange of already compressed data,
    - compress → transfer → decompress.

- **Lower memory footprint:**
  - Less disk space used.
  - Less RAM used.
  - Less GPU memory used.

- **Improved internal memory access:**
  - In some cases improved internal GPU memory access.
Fixed length compression

Fixed length (FL) – is a simple well known compression scheme where fixed number of bits is suppressed. Suppressed bits should be equal to 0.

Figure: Original data, only 4 bits are used in each byte.
Fixed length compression

Fixed length (FL) – is a simple well known compression scheme where fixed number of bits is suppressed. Suppressed bits should be equal to 0.

Figure: Original data, only 4 bits are used in each byte.
Fixed length compression

Fixed length (FL) – is a simple well known compression scheme where fixed number of bits is suppressed. Suppressed bits should be equal to 0.

Figure: Original data, only 4 bits are used in each byte.

Figure: Compressed data (each byte encodes two words of length 4 bits.)
Fixed length compression

Fixed length (FL) – is a simple well known compression scheme where fixed number of bits is suppressed. Suppressed bits should be equal to 0.

Figure: Original data, only 4 bits are used in each byte.

Figure: Compressed data (each byte encodes two words of length 4 bits.)

compression ratio (CR) = \( \frac{\text{Uncompressed size}}{\text{Compressed size}} = 2 \),
Fixed length compression

**Fixed length (FL) compression:**
- easy to implement,
- easy to achieve high data throughput.

**Many applications:**
- Database compression: Columns, Indexes,
- Timeseries compression,
- Graph compression,
- etc.

**Many variants:**
- Patched FL, Adaptive FL, DELTA-*
Performance over flexibility (Fang et al. 2010)
- High performance but highly simplified version of algorithm.
  - Words are mapped to full bytes e.g. 4 bits word will be mapped to 1 byte.
- Uses map primitive.
- Coalesced reads and writes: YES.
- Direct memory access: YES.

Flexibility over performance
(Nvbio and Kaczmarski, Przymus 2012-2017)
- No simplifications at the cost of lower performance.
- Supports all possible bit encodings.
- Uses allgather or gather primitive.
- Coalesced reads and writes: NO.
- Direct memory access: YES.
Fixed length compression on GPU

Figure: Read pattern: GPU version of FL algorithm

Figure: Write pattern: GPU version of FL algorithm
Fixed length on GPU (C+D, GTX Titan Black)
Fixed length on GPU (1 GB of data, GTX Titan Black)
Can we do better?

Aligned Fixed Length (AFL) algorithm.

- The FL algorithm is optimized for CPU memory access scheme.
- We can do better with GPU friendly memory organisation scheme.

Features

- No simplifications, high performance on GPU.
- Still works quite well on CPU, but loses some cache hits benefits.
- Supports all possible bit encodings.
- Uses allgather or gather primitive.
- Coalesced reads and writes: YES.
- Direct memory access: YES.
**Aligned FL on GPU**

![Read pattern: GPU version of Aligned FL algorithm](image1)

**Figure:** Read pattern: GPU version of Aligned FL algorithm

![Write pattern: GPU version of Aligned FL algorithm](image2)

**Figure:** Write pattern: GPU version of Aligned FL algorithm

---

Lightweight Compression Methods Achieving 120GBps and More 11/25
Aligned FL on GPU (C+D, GTX Titan Black)
Aligned FL on GPU (1 GB of data, GTX Titan Black)

![Graph showing Compr./GB/s and Decompr./GB/s for different Bit Encoding values (8, 16, 24, 32, 40, 48, 56, 63) comparing int and long data types.]
Direct memory access (Random access):
- single value access with decompression on-the-fly,
- no need for explicit decompression step,
- simple integration with existing algorithms.

Example applications of direct access FL:
- Bioinformatics – NVBIO,
- Databases – Fang et al. 2010,

Aligned FL supports direct access!
Direct access AFL on GPU (C+D, GTX Titan Black)

![Graph showing bandwidth vs data size for different data types and formats. The graph displays the performance of int and long data types with both max and min bandwidths, highlighting the efficiency of direct access on GPUs.](image-url)
Initial transformations

A initial data preparation is often used in order to minimize bits usage.

**Frame of reference (FOR):**
- Subtracts a given value from all values
  \[ \{v_0, v_1, \ldots, v_n\} \rightarrow \{v_0 - f, v_1 - f, \ldots, v_n - f\} \].
- Straightforward and simple integration with **AFL**.

**DELTA:**
- Transforms data into the differences between successive values
  \[ \{v_0, v_1, \ldots, v_n\} \rightarrow \{v_1 - v_0, v_2 - v_1, \ldots, v_n - v_{n-1}\} \].
- Integration with **AFL** algorithm is not that simple.
  - Coalesced memory reads and writes require different scheme of data reads.
  - Threads operate on data subsequence \( \{a_k, a_{k+32}, a_{k+64}, \ldots\} \).
  - **Solution:** Interthread communication within warp using **shuffle** instruction available starting from the **Kepler**.
Dealing with the data outliers

**FL** and **AFL** algorithms are prone to data outliers.

**Example:**

- Input data: \(\{1, 2, 3, 2, 2, 3, 1, 1, 3, 2, 3, 1, 1\}\)
  - 2 bits **FL** (or **AFL**) encoding may be used.
- Input data: \(\{1, 2, 3, 2, 2, 3, 1, 1, 64, 2, 3, 1, 1\}\)
  - 6 bits **FL** (or **AFL**) encoding may be used.

**Solution:** Outlier aware algorithms

- Patched FL and Patched Aligned FL
- Adaptive FL and Adaptive Aligned FL
Patched Aligned FL

- Patched FL – **Zukowski et al. 2006** for CPU.
- Patched FL (new memory organisation) – **Yan et al. 2009** for CPU.
- Patched FL on GPU – **Kaczmarski and Przymus 2012, 2013**.

**Structure:**

- compressed data, exceptions positions, exceptions values.

**Patched Aligned FL:**

- **One step compression:**
  - threads gather outliers in local memory buffers,
  - buffer overflow is managed per warp (voting of threads).

- **Two step decompression:**
  - step 1: AFL decompression,
  - step 2: exceptions extraction.
Patched Aligned FL on GPU (C+D, GTX Titan Black)

[Graph showing bandwidth in GB/s vs. data size in MB for different data types (int/optim., int/pessim., long/optim., long/pessim.)]
Adaptive FL

First introduced by Delbru et al. 2010 for CPU:

- processes data in chunks,
- sets different bit length encoding for each chunk.

Adaptive FL does not fit GPU memory model very well.

**Adaptive Aligned FL:**

- Aligned FL + new organisation of compressed chunks.
- Bit length encoding is established per warp.
- Compressed chunks are order according to warps completion time.

Note that warps may finish in different order then data order.
Figure: Adaptive aligned memory organisation: main compression array.

Figure: Adaptive aligned memory organisation: warp offset index.
Adaptive Aligned FL on GPU (C+D, GTX Titan Black)

The diagram shows the relationship between data size (MB) and bandwidth (GB/s) for different data types and optimizations.

- **Data Size MB:** The x-axis represents the data size in megabytes, ranging from 0 to 1000 MB.
- **Bandwidth GB/s:** The y-axis represents the bandwidth in gigabytes per second, ranging from 0 to 120 GB/s.

The chart includes the following data types and optimizations:
- **int optim.**
- **int pessim.**
- **long optim.**
- **long pessim.**

Color codes for different data types and optimizations are indicated in the legend at the bottom of the chart.
<table>
<thead>
<tr>
<th>Column</th>
<th>Org. / used</th>
<th>Size MB</th>
<th>Sort.</th>
<th>Alg.</th>
<th>CR</th>
<th>Bandwidth [GB/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compress.</td>
</tr>
<tr>
<td>l_discount</td>
<td>dec. / int(4)</td>
<td>N 239</td>
<td></td>
<td>FL</td>
<td>8.00</td>
<td>52.7137</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AFL</td>
<td>8.00</td>
<td>211.8846</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PAFL</td>
<td>8.00</td>
<td>200.6869</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AAFL</td>
<td>30.79</td>
<td>101.4848</td>
</tr>
<tr>
<td>l_quantity</td>
<td>int(4) / int(4)</td>
<td>N 239</td>
<td></td>
<td>FL</td>
<td>2.46</td>
<td>45.4952</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AFL</td>
<td>2.46</td>
<td>154.2067</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PAFL</td>
<td>2.46</td>
<td>153.2897</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AAFL</td>
<td>9.72</td>
<td>81.5276</td>
</tr>
<tr>
<td>l_partkey</td>
<td>id / int(4)</td>
<td>N 239</td>
<td></td>
<td>FL</td>
<td>1.52</td>
<td>37.7608</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AFL</td>
<td>1.52</td>
<td>125.7666</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PAFL</td>
<td>1.52</td>
<td>125.0484</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AAFL</td>
<td>6.05</td>
<td>72.4169</td>
</tr>
<tr>
<td>l_shipdate</td>
<td>date / int(4)</td>
<td>Y 239</td>
<td></td>
<td>FL</td>
<td>1.06</td>
<td>28.2662</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DELTA-AFL</td>
<td>1.88</td>
<td>136.6975</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DELTA-PAFL</td>
<td>31.91</td>
<td>205.3076</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DELTA-AAFL</td>
<td>126.31</td>
<td>210.5169</td>
</tr>
</tbody>
</table>
FeatherGPU

- A lightweight compression library for GPU.
- [github.com/mis-wut/feathergpu](https://github.com/mis-wut/feathergpu)
- MIT-licensed.

**Supported algorithms:**
- FL, FOR, Adaptive FL, Patched AFL
- Aligned FL, Aligned FOR, Adaptive Aligned FL, Patched Aligned AFL

**GTX Titan Black:**
- Compression **up to** 250GB/s
- Decompression **up to** 250GB/s

**Tesla P100 16 GB (New!):**
- Compression **up to** 550GB/s
- Decompression **up to** 450GB/s

Appendix
Aligned FL on GPU (C+D, Tesla P100 16 GB)

![Graph showing bandwidth vs. data size for different data types (int max, int min, long max, long min, int, long)]

- **X-axis**: Data Size MB
- **Y-axis**: Bandwidth GB/s

Legend:
- **int max**: Dark yellow
- **int min**: Light yellow dotted line
- **long max**: Dark blue
- **long min**: Light blue dotted line
- **int**: Orange
- **long**: Light blue

P. Przymus
Lightweight Compression Methods Achieving 120GBps and More

2/9
Aligned FL on GPU (1 GB of data, Tesla P100 16 GB)
DELTA Aligned FL on GPU

![Graph showing Compressed and Decompressed GB/s for different bit encodings (int and long types).](image)
Patched Aligned FL on GPU (compression)
Adaptive Aligned FL on GPU (compression)

Bandwidth GB/s vs Data Size MB

- **int optim.**
- **int pessim.**
- **long optim.**
- **long pessim.**

**P. Przymus**

Lightweight Compression Methods Achieving 120GBps and More