

# A GPU Accelerated cuSZ Error-Bounded Lossy Compressor

DEVELOPMENT GROUP

JIANAN TIAN, DINGWEN TAO, SHENG DI, FRANCK CAPPELLO

CONTRIBUTORS (ALPHABETIC)

CODY RIVERA, JON CALHOUN, KAI ZHAO, MEGAN HICKMAN, ROBERT UNDERWOOD, XIN LIANG

CREATED ON MAY 7, 2020, LAST COMPILED ON MAY 9, 2020

## Contents

<b>1 introduction</b>	<b>1</b>	3.1.2 tuning . . . . .	2
1.1 workflow . . . . .	1	3.1.3 extension and use scenarios . . . . .	3
1.2 about compression ratio . . . . .	1	3.2 cuSZ as an analytical tool . . . . .	3
<b>2 set up</b>	<b>1</b>	3.3 example . . . . .	3
2.1 requirements . . . . .	1	<b>4 project management</b>	<b>3</b>
2.2 download . . . . .	1	4.1 TODO List . . . . .	3
2.3 compile . . . . .	2	4.2 changelog . . . . .	4
<b>3 run</b>	<b>2</b>	<b>5 reference</b>	<b>4</b>
3.1 cuSZ as a compressor . . . . .	2	<b>6 acknowledgement</b>	<b>4</b>
3.1.1 basic use . . . . .	2	<b>7 license</b>	<b>4</b>

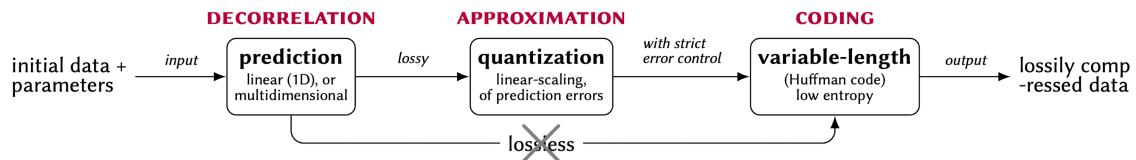


Figure 1: Error-control workflow.

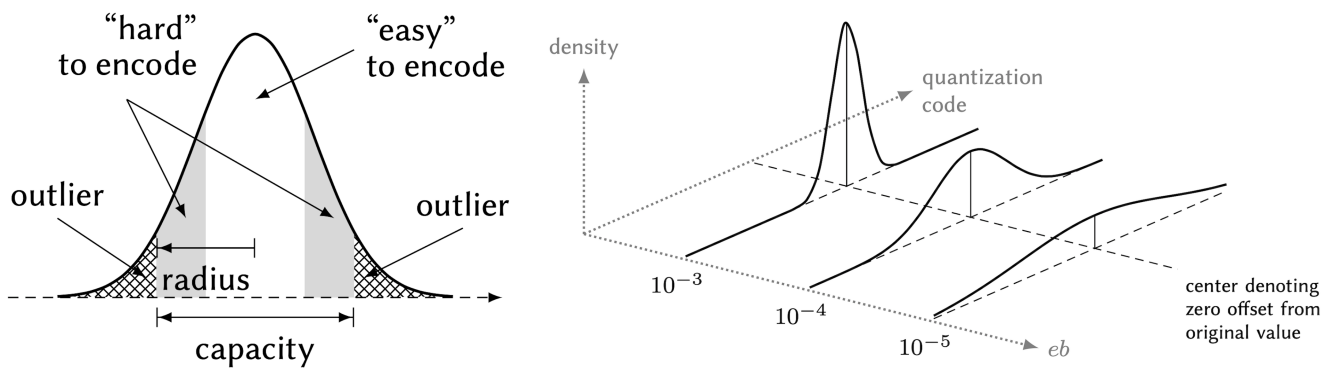


Figure 2: (left) classification in prediction error distribution, (right) smaller error bound linear-scales the prediction error distribution.

## 1 introduction

### 1.1 workflow

### 1.2 about compression ratio

## 2 set up

### 2.1 requirements

- NVIDIA GPU with Kepler, Maxwell, Pascal, Volta, or Turing microarchitectures
- CUDA 9.1+ and GCC 7+ (recommended: CUDA 10.1 + GCC 8)
- CMake 3.11+

### 2.2 download

```
git clone git@github.com:hipdac-lab/cuSZ.git
```

### 2.3 compile

```
cd cuSZ
cmake CMakeLists.txt      # using cmake to compile cuSZ for {1,2,3}-D, with Huffman codec
make
```

## 3 run

Commands `cusz` or `cusz -h` are for instant instructions.

### 3.1 cuSZ as a compressor

#### 3.1.1 basic use

The basic use `cuSZ` is given below.

```
./cusz -f32 -m r2r -e 1.23e-4.56 -i ./data/sample-cesm-CLDHGH -D cesm -z -x
      ^      ^~~~~~ ^~~~~~ ^~~~~~ ^~~~~~ ^~~~~~ ^~~~~~ ^~~~~~ ^~~~~~ ^~~~~~
      |      mode   error bound      input datum file      demo | |
      dtype                                datum  zip  unzip
```

`-D cesm` specifies preset dataset for demonstration. In this case, it is CESM-ATM, whose dimension is 1800-by-3600, following  $y$ - $x$  order. To otherwise specify datum file and input dimensions arbitrarily, we use `-2 3600 1800`, then it becomes

```
./cusz -f32 -m r2r -e 1.23e-4.56 -i ./data/sample-cesm-CLDHGH -2 3600 1800 -z -x
```

To conduct compression, several input arguments are **necessary**,

- `-z` or `--zip` to compress
- `-x` or `--unzip` to decompress
- `-m` or `--mode` to specify compression mode. Options include `abs` (absolute value) and `r2r` (relative to value range).
- `-e` or `--eb` to specify error bound
- `-i` to specify input datum file
- `-D` to specify demo dataset name or `-{1,2,3}` to input dimensions

### 3.1.2 tuning

There are also internal a) quant. code representation, b) Huffman codeword representation, and c) chunk size for Huffman coding exposed. Each can be specified with argument options.

- `-Q` or `--quant-rep` or `--bcode-bitwidth` `<8|16|32>` to specify bincod/quant. code representation. Options 8, 16, 32 are for `uint8_t`, `uint16_t`, `uint32_t`, respectively. (Manually specifying this may not result in optimal memory footprint.)
- `-H` or `--huffman-rep` or `--hcode-bitwidth` `<32|64>` to specify Huffman codeword representation. Options 32, 64 are for `uint32_t`, `uint64_t`, respectively. (Manually specifying this may not result in optimal memory footprint.)
- `-C` or `--huffman-chunk` or `--hcode-chunk` `[256|512|1024|...]` to specify chunk size for Huffman codec. Should be a power-of-2 that is sufficiently large. (This affects Huffman decoding performance *significantly*.)

### 3.1.3 extension and use scenarios

**preprocess** Some application such as EXAFEL preprocesses with binning<sup>1</sup> in addition to skipping Huffman codec.

**disabling modules** Also according to EXAFEL, given binning and `uint8_t` have already result in a compression ratio of up to 16, Huffman codec may not be expected in a real-world use scenario. In such circumstances, `--skip huffman` can be used.

Other module skipping for use scenarios are in development.

## 3.2 cuSZ as an analytical tool

`--dry-run` or `-r` in place of `-a` and/or `-x` enables dry-run mode to get PSNR. This employs the feature of dual-quantization that the decompressed data is guaranteed the same with prequantized data.

## 3.3 example

1. run a 2D CESM demo at  $1e-4$  relative to value range

```
./cusz -f32 -m r2r -e 1e-4 -i ./data/sample-cesm-CLDHGH -D cesm -z -x
```

2. alternatively, to use full option name,

```
./cusz -f32 --mode r2r --eb 1e-4 --input ./data/sample-cesm-CLDHGH \
--demo cesm --zip --unzip
```

3. run a 3D Hurricane Isabel demo at  $1e-4$  relative to value range

```
./cusz -f32 -m r2r -e 1e-4 -i ./data/sample-hurr-CLOUDf48 -D hurricanne -z -x
```

4. run CESM demo with 1) `uint8_t`, 2) 256 quant. bins,

```
./cusz -f32 -m r2r -e 1e-4 -i ./data/sample-cesm-CLDHGH -D cesm -z -x \
-d 256 -Q 8
```

5. in addition to the previous command, if skipping Huffman codec,

<sup>1</sup>A current binning setting is to downsample a 4-by-4 cell to 1 point.

```
./cusz -f32 -m r2r -e 1e-4 -i ./data/sample-cesm-CLDHGH -D cesm -z -x \
-d 256 -Q 8 --skip huffman # or '-X/-S huffman'
```

6. some application such as EXAFEL preprocesses with binning<sup>2</sup> in addition to skipping Huffman codec

```
./cusz -f32 -m r2r -e 1e-4 -i ./data/sample-cesm-CLDHGH -D cesm -z -x \
-d 256 -Q 8 --pre binning --skip huffman # or '-p binning'
```

7. dry-run to get PSNR and to skip real compression or decompression; -r also works alternatively to --dry-run

```
./cusz -f32 -m r2r -e 1e-4 -i ./data/sample-cesm-CLDHGH -D cesm --dry-run # or '-r'
```

## 4 project management

### 4.1 TODO List

Please refer to *Project Management page*.

### 4.2 changelog

May, 2020

**perf** [need review] decrease memory footprint by merging data and outlier during compression

**feature** add --skip huffman and --verify huffman options

**feature** add binning as preprocessing, --pre binning or -p binning

**prototype** use cuSparse to transform outlier to dense format

**feature** add argparse to check and parse argument inputs

**refactor** add CUDA wrappers (e.g., mem::CreateCUDASpace)

April, 2020

**feature** add concise and detailed help doc

**deploy** sm\_61 (e.g., P1000) and sm\_70 (e.g., V100) binary

**feature** add dry-run mode

**refactor** merge cuSZ and Huffman codec in driver program

**perf** 1D PdQ (and reverse PdQ) blockDim set to 32, throughput changed from 2.7 GBps to 16.8 GBps

**deploy** histograming, 2013 algorithm supersedes naive 2007 algorithm by default

**feature** add communication of equivalence calculation

**feature** use cooperative groups (CUDA 9 required) for canonical Huffman codebook

**perf** faster initializing shared memory for PdQ, from 150 GBps to 200 GBps

**feature** add Huffman inflating/decoding

**refactor** merge {1,2,3}-D cuSZ

**feature** set 32- and 64-bit as internal Huffman codeword representation

**feature** now use arbitrary multiple-of-8-bit for quant. code

**feature** switch to canonical Huffman code for decoding

March, 2020

**perf** tuning thread number for Huffman deflating and inflating

**feature** change freely to 32bit intermediate Huffman code representation

**demo** add EXAFEL demo

**feature** switch to faster histograming

February, 2020

**demo** SDRB suite metadata in SDRB.hh

**feature** visualize histogram (pSZ)

**prototype** add CPU pSZ, p for prototyping

**milestone** PdQ for compression, Huffman encoding and deflating

<sup>2</sup>A current binning setting is to downsample a 4-by-4 cell to 1 point.

## 5 reference

- [1] Gómez-Luna, Juan, José María González-Linares, José Ignacio Benavides, and Nicolás Guil. "An optimized approach to histogram computation on GPU." *Machine Vision and Applications* 24, no. 5 (2013): 899-908.
- [2] Barnett, Mark L. "Canonical Huffman encoded data decompression algorithm." U.S. Patent 6,657,569, issued December 2, 2003.

## 6 acknowledgement

This R&D was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations – the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem. This repository was based upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357, and also supported by the National Science Foundation under Grant No. 1948447.

## 7 license

BSD 3-Clause License

Copyright (c) 2020, Jiannan Tian, Dingwen Tao, Sheng Di, Franck Cappello  
All rights reserved.

cuSZ - A GPU Accelerated Error-Bounded Lossy Compressor for Scientific Data  
[Version 0.1]

Redistribution and use in source and binary forms, with or without  
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contact: Dingwen Tao (dingwen.tao@ieee.org), Jiannan Tian(tian.jn09@gmail.com)