Scientific Data Compression with SCIL

Julian M. Kunkel\textsuperscript{1,2}, Nabeeh Jum’ah\textsuperscript{2}, Anastasiia Novikova\textsuperscript{2}

1: Department of Computer Science, University of Reading
2: Scientific Computing, Department of Informatics, University of Hamburg

2019-10-23

\textsuperscript{3}Thanks to Thomas Dubos, Hisashi Yashiro for supporting this work.
Motivation

Large Data Volumes

- Weather and climate data is huge (30+ PByte for CMIP6)
  - Preserving the data is expensive (but necessary)
- Data compression can reduce the burden
  - But there exist many solutions for data compression
  - Users must hardcode the compression algorithm to use

Goals of AIMES:

- Design of domain-specific compression (ratio > 10 : 1)
- Separation of concerns
  - User specifies required behavior, i.e., precision, performance
  - Compression library picks appropriate compression method
Approach

- Defined quantities to define accuracy/performance
- Design of the Scientific Compression Interface Library (SCIL)
  - Implementation of suitable compression schemes
  - Supporting existing compression algorithms
- Integration of SCIL into HDF5/NetCDF
- Evaluation of the performance on various data sets
  - Climate/Weather data including NICAM
- Develop a methodology for identifying the required accuracy
  - Explore the impact of compression on the scientific conclusions
Example for Data Compression

Synthetic data (Simplex, options 206, 2D: 100x100 points)

Right image uses lossy compression (Sigbits 3bits, ratio 11.3:1)
Outline

1. Introduction
2. SCIL
3. Evaluation
4. Summary
Supported Quantities

**Accuracy quantities:**
- **absolute tolerance:** compressed can become true value $\pm$ absolute tolerance
- **relative tolerance:** percentage the compressed value can deviate from true value
- **relative error finest tolerance:** value defining the absolute tolerable error for relative compression for values around 0
- **significant digits:** number of significant decimal digits
- **significant bits:** number of significant decimals in bits

**Performance quantities:**
- **compression speed:** in MiB or GiB, or relative to network or storage speed
- **decompression speed:** in MiB or GiB, or relative to network or storage speed

**Supplementary quantities:**
- **fill value:** a value that scientists use to mark special data point
Overview of SCIL Features

- **API for setting quantities**
  - C, NetCDF, HDF5 compatibility
  - NetCDF variables can also be defined in an external file

- **Metacompressor**
  - Can utilize state-of-the-art lossy and lossless compressors
  - Brings some own compressors
  - Can chain different components of compressors

- **Additional tools**
  - Compression (NetCDF4, NetCDF3, CSV)
  - Benchmarking
  - Pattern creator to create arbitrary sized synthetic data
  - Noise generation/addition
  - (Plotting)
Configuration File for the Quantities

The file in the env. var. `NETCDF_SCIL_HINTS_FILE` is read by NetCDF and applied to the respective variables:

**Variable 1:**
- `relative_tolerance_percent=1`

**Variable 2:**  # Developer comment
- `relative_tolerance_percent=1`
- `relative_err_finest_abs_tolerance=1`
- `absolute_tolerance=1`
- `significant_digits=1`
- `significant_bits=1`
- `lossless_data_range_up_to=1`
- `lossless_data_range_from=1`
- `fill_value=4711`
- `comp_speed=0.5*MiB`
- `decomp_speed=1*NetworkSpeed`
- `force_compression_methods=abstol,lz4`
Architecture of SCIL

**Figure:** SCIL compression path and components (extended)
Compression Chain

- Chaining of compressors, e.g., preconditioner, entropy coder
  - Depends on the datatype: float, integer or bytes
  - Preconditioner: manipulate data to make it better compressible
  - Entropy coder: lossless compression scheme
Supported Compression Schemes

Existing algorithms

- **Lossless**: ZSTD, LZ4, GZIP, huffman
- **Lossy**: FPZIP, ZFP, SZ
  - They support only absolute or the relative precision quantity
- *(Memcopy for benchmarking and to sustain performance)*

Developed algorithms

- **Abstol**: Only absolute tolerance
- **Reltol**: Only relative tolerance
- **Sigbits**: Only significant bits
- **Allquant**: respect all precision quantities
Identifying Appropriate Algorithm: Applying the Quantities

- Significant digits and bits can be converted
  - Take the more precise one
- Relative tolerance can be converted to significant bits
  - Use absolute compression for finest tolerance (for small value)
- Treat fill value explicitly
- Min/Max are used by many methods to change behavior
- Check if compression would be fast enough

Figure: Range for Allquant, the sign is only stored if necessary
Outline

1 Introduction

2 SCIL

3 Evaluation

4 Summary
Analyzing Performance of Lossy Compression

Data

- Single precision (1+8+23 bits)
- Synthetic, generated by SCIL’s pattern lib.
  - e.g., Random, Steps, Sinus, Simplex
- Data of the variables created by ECHAM, Isabel, NICAM

Experiments

- Single thread, 10 repeats
- Lossless (memcpy and lz4)
- Lossy compression with significant bits (zfp, sigbits, sigbits+lz4)
- Lossy compression with absolute tolerance (zfp, sz, abstol, abstol+lz4)
  - Tolerance: 10%, 2%, 1%, 0.2%, 0.1% of the data maximum value
## Comparing Algorithms for the Scientific Files

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Ratio</th>
<th>Compr. MiB/s</th>
<th>Decomp. MiB/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>NICAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abstol</td>
<td>0.206</td>
<td>499</td>
<td>683</td>
</tr>
<tr>
<td>abstol,lz4</td>
<td>0.015</td>
<td>458</td>
<td>643</td>
</tr>
<tr>
<td>sz</td>
<td>0.008</td>
<td>122</td>
<td>313</td>
</tr>
<tr>
<td>zfp-abstol</td>
<td>0.129</td>
<td>302</td>
<td>503</td>
</tr>
<tr>
<td>ECHAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abstol</td>
<td>0.190</td>
<td>260</td>
<td>456</td>
</tr>
<tr>
<td>abstol,lz4</td>
<td>0.062</td>
<td>196</td>
<td>400</td>
</tr>
<tr>
<td>sz</td>
<td>0.078</td>
<td>81</td>
<td>169</td>
</tr>
<tr>
<td>zfp-abstol</td>
<td>0.239</td>
<td>185</td>
<td>301</td>
</tr>
<tr>
<td>Isabel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abstol</td>
<td>0.190</td>
<td>352</td>
<td>403</td>
</tr>
<tr>
<td>abstol,lz4</td>
<td>0.029</td>
<td>279</td>
<td>356</td>
</tr>
<tr>
<td>sz</td>
<td>0.016</td>
<td>70</td>
<td>187</td>
</tr>
<tr>
<td>zfp-abstol</td>
<td>0.039</td>
<td>239</td>
<td>428</td>
</tr>
<tr>
<td>Random</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abstol</td>
<td>0.190</td>
<td>365</td>
<td>382</td>
</tr>
<tr>
<td>abstol,lz4</td>
<td>0.194</td>
<td>356</td>
<td>382</td>
</tr>
<tr>
<td>sz</td>
<td>0.242</td>
<td>54</td>
<td>125</td>
</tr>
<tr>
<td>zfp-abstol</td>
<td>0.355</td>
<td>145</td>
<td>241</td>
</tr>
</tbody>
</table>

### Harmonic mean compression performance for different scientific data sets

(a) 1% absolute tolerance

(b) 9 bits precision

**Table:** Harmonic mean compression performance for different scientific data sets
Results for Absolute Tolerance of ECHAM

- Using NetCDF3
- Abstol: 1% max
- Compress whole file
- Sometimes SZ better, sometimes abstol+lz4
Results for Absolute Tolerance of NICAM

- Using NetCDF4/HDF5
- Abstol: 1% max
- Multiple chunks generated
- Data is smooth: SZ best ratio
  Abstol+LZ4 fastest
Compression in HDF5 and NetCDF

- Testing parallel compression on NetCDF capable DSL code
  - 128 processes on Mistral
  - 268 million grid cells x 64 vertical levels
  - Field values between -10 and +55

<table>
<thead>
<tr>
<th>Compression method</th>
<th>Parameter</th>
<th>Write time</th>
<th>Data size</th>
<th>Throughput*</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCIL</td>
<td></td>
<td>in s</td>
<td>in GB</td>
<td>in MB/s</td>
<td></td>
</tr>
<tr>
<td>No-compression</td>
<td></td>
<td>165.3</td>
<td>71.4</td>
<td>432</td>
<td>1.0</td>
</tr>
<tr>
<td>memcpy</td>
<td></td>
<td>570.1</td>
<td>71.4</td>
<td>125</td>
<td>0.3</td>
</tr>
<tr>
<td>lz4</td>
<td></td>
<td>795.3</td>
<td>71.9</td>
<td>90</td>
<td>0.2</td>
</tr>
<tr>
<td>abstol,lz4</td>
<td>absolute_tolerance=1</td>
<td>12.8</td>
<td>2.7</td>
<td>5578</td>
<td>12.9</td>
</tr>
<tr>
<td></td>
<td>absolute_tolerance=.1</td>
<td>72.6</td>
<td>2.8</td>
<td>983</td>
<td>2.3</td>
</tr>
<tr>
<td>sigbits,lz4</td>
<td>relative_tolerance_percent=1</td>
<td>12.9</td>
<td>2.9</td>
<td>5535</td>
<td>12.8</td>
</tr>
<tr>
<td></td>
<td>relative_tolerance_percent=.1</td>
<td>18.3</td>
<td>3.2</td>
<td>3902</td>
<td>9.0</td>
</tr>
</tbody>
</table>

Table: Results. *Virtual throughput relative to the uncompressed file size.

- Access to shared file is slow, overhead of HDF5 compression chain
- Parallel compression lately added to HDF5
Investigating Compression Error

- Blue shade: conditional mean SST difference between CPL and SMO simulations where statistical test is passed
- Adding noise affects little the overall pattern, but makes it harder to pass statistical test (more white holes in the blue)
- Conclusions unchanged even with quite large noise on SST (0.2°K), probably due to averaging several events
Outline

1. Introduction
2. SCIL
3. Evaluation
4. Summary
Summary

- SCIL addresses compression (lossy and lossless)
  - Allow scientists to define required variable accuracy
  - Exploit this knowledge in the compression scheme
  - Novel schemes sometimes compete with existing algorithms
- Predicting performance/ratio for data is difficult
- Future work: machine learning of
  - Performance, expected ratio for data
  - Best compression algorithms
Synthetic Patterns

- Four scatter plots showing the relationship between different variables and ratios.
- X-axis represents absolute tolerance in % or precision bits.
- Y-axis represents the ratio values.
- Different markers represent different algorithms.

Julian Kunkel
Under noise mimicking compression, reproduce conclusions of Berthou et al. 2016

- Mediterranean region (Spain, France)
- Evaluates how wind, through its action on the ocean, impacts (with some delay) heavy precipitation
- Compares outputs from two simulations (CPL and SMO) and subjects the difference to tests of statistical significance
  - Student t test; 97.5% probability of rejecting a zero difference
- Analyzed fields: wind, rain (convective and non-convective), humidity, sea-surface temperature (SST)
Approach

1. Reproduce published results (Find data/scripts used in paper)
2. Apply statistical model for noise induced by lossy compression
   - Use Gaussian white noise
3. Redo the analysis, check if the conclusions are still supported
4. Increase levels of noise to input data (= model output)
   - One field at a time; then together