In-Depth Exploration of Single-Snapshot Lossy Compression Techniques for N-Body Simulations

Dingwen Tao (University of California, Riverside)
Sheng Di (Argonne National Laboratory)
Zizhong Chen (University of California, Riverside)
Franck Cappello (Argonne National Laboratory & UIUC)
Outline

• Introduction
• Challenges of lossy compression for particle simulations
• Optimizations for particle simulations
  • Cosmology simulation
  • Molecular dynamics simulation
• Empirical evaluation
• Conclusion
Introduction

- Today’s scientific research is using simulations or instruments and produces extremely large amount of data to process / analyze
- Cosmology Simulation (HACC)
  - 20 PB data: a single 1-trillion-particle simulation
  - Peta-scale system’s File System ~ 20 PB
  - Mira at ANL has 26 PB FS, 20 PB / 26 PB ~ 80%
  - Blue Waters (1TB/s FS), 20 x 10^15 / 10^12 seconds ~ **5h30min** to store the data
  - Data reduction of about a factor of 10 is needed
  - Currently drop 9 snapshots over 10 (decimation in time)
Existing lossless compressors work **not efficiently** on large-scale scientific data (compression ratio up to 2)

**Compression ratios for lossless compressors on large-scale**

<table>
<thead>
<tr>
<th></th>
<th>bzip2</th>
<th>dpcm</th>
<th>fsd</th>
<th>gzip</th>
<th>lzma</th>
<th>p7zip</th>
<th>rar</th>
<th>zzip</th>
</tr>
</thead>
<tbody>
<tr>
<td>aztec</td>
<td>1.15</td>
<td>1.69</td>
<td>1.42</td>
<td>1.22</td>
<td>1.15</td>
<td>1.39</td>
<td>1.26</td>
<td>1.15</td>
</tr>
<tr>
<td>bt</td>
<td>1.10</td>
<td>1.36</td>
<td>1.02</td>
<td>1.13</td>
<td>1.10</td>
<td>1.32</td>
<td>1.15</td>
<td>1.10</td>
</tr>
<tr>
<td>eulag</td>
<td>1.04</td>
<td>1.23</td>
<td>1.06</td>
<td>1.06</td>
<td>1.05</td>
<td>1.15</td>
<td>1.07</td>
<td>1.09</td>
</tr>
<tr>
<td>lu</td>
<td>1.02</td>
<td>1.23</td>
<td>0.99</td>
<td>1.05</td>
<td>1.03</td>
<td>1.22</td>
<td>1.07</td>
<td>1.03</td>
</tr>
<tr>
<td>sp</td>
<td>1.08</td>
<td>1.25</td>
<td>0.95</td>
<td>1.11</td>
<td>1.07</td>
<td>1.31</td>
<td>1.14</td>
<td>1.07</td>
</tr>
<tr>
<td>sppm</td>
<td>6.78</td>
<td>4.16</td>
<td>2.14</td>
<td>6.31</td>
<td>7.94</td>
<td>8.31</td>
<td>7.68</td>
<td>---</td>
</tr>
<tr>
<td>sweep3d</td>
<td>1.06</td>
<td>1.49</td>
<td>1.20</td>
<td>1.09</td>
<td>1.19</td>
<td>1.26</td>
<td>1.30</td>
<td>1.35</td>
</tr>
<tr>
<td>geo_mean</td>
<td>1.40</td>
<td>1.60</td>
<td>1.20</td>
<td>1.42</td>
<td>1.46</td>
<td>1.66</td>
<td>1.52</td>
<td>1.13</td>
</tr>
</tbody>
</table>

Compression ratio (CR) = Original data size / Compressed data size

Existing State-of-The-Art Lossy Compressors

- **SZ** (ANL)
  - Multidimensional / multilayer prediction model
  - Error-controlled quantization
  - Customized Huffman coding
- **ZFP** (LLNL)
  - Customized orthogonal block transform
  - Embedded coding
- **Tucker Decomposition** (SNL)
  - Tensor-based dimensional reduction
- **ISABELA** (NCSU)
  - Sorting preconditioner
  - B-Spline interpolation
Particle Simulation Datasets

**HACC** Cosmology code (Hardware/Hybrid Accelerated Cosmology). N-body problem with domain decomposition, medium/long-range force solver (particle-mesh method), short-range force solver (particle-particle/particle-mesh algorithm).

**AMDF** Molecular Dynamics code (Accelerated Molecular Dynamics Family) Solver only for short-range force interactions

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Source</th>
<th># of Particles</th>
<th># of Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>HACC</td>
<td>Cosmology Simulation</td>
<td>147.3 million</td>
<td>6</td>
</tr>
<tr>
<td>AMDF</td>
<td>Molecular Dynamics</td>
<td>2.8 million</td>
<td>6</td>
</tr>
</tbody>
</table>

- 3 velocity variables and 3 position (coordinate) variables
- Velocity variables – \(v_x, v_y, v_z\), coordinate variables – \(x_x, y_y, z_z\)
- Other quantities can be computed from velocities and coordinates
- \(v_x, v_y, v_z, x_x, y_y, z_z\) are 1D floating-point data
- Storage format: an array of structures or a structure of arrays
Particle Simulation Datasets

HACC

AMDF

(a) Coordinate variables

(b) Velocity variables
Challenges of Lossy Compression for Particle Simulations

- Extremely large-scale n-body simulation only allows **ONE** snapshot to be loaded into the memory → single-snapshot compression
- Trajectory / temporal-coherence based compression methods are not applicable, can only use spatial information
- Spatial information has fairly limited correlation of adjacent elements
- Existing state-of-the-art lossy compressors designed for mesh data have **low** compression ratio on n-body simulation data (especially velocities)

<table>
<thead>
<tr>
<th>Compressor</th>
<th>HACC</th>
<th>AMDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>GZIP</td>
<td>1.2</td>
<td>1.1</td>
</tr>
<tr>
<td>CPC2000</td>
<td>3.5</td>
<td>3.2</td>
</tr>
<tr>
<td>FPZIP</td>
<td>3.1</td>
<td>1.8</td>
</tr>
<tr>
<td>ISABELA</td>
<td>1.4</td>
<td>1.2</td>
</tr>
<tr>
<td>ZFP</td>
<td>2.3</td>
<td>1.9</td>
</tr>
<tr>
<td>SZ</td>
<td>4.6</td>
<td>2.7</td>
</tr>
</tbody>
</table>

Relative error bound = $10^{-4}$

Optimization – Prediction Model

• Good prediction model can provide high prediction accuracy
  • High compression ratio
  • Low compression error

\[ \text{Important to prediction-based lossy compressors} \]

• SZ’s multidimensional / multilayer prediction model
  • 1D: degrades to linear curve-fitting model
  \[ v_{x_i}^{\text{pred}} = 2v_{x_{i-1}} - 2v_{x_{i-2}} \]
  • Not efficient due to high irregularity of data

• Adopt a simple but practical prediction model
  • Last-value model: \[ v_{x_i}^{\text{pred}} = v_{x_{i-1}} \] (1D case in Lorenzo predictor)
Compression Ratio Improved by Optimized Prediction Model

Compression ratios improved by 10+% on average
Optimizations for MD Simulations

- Sorting is a classic method to enhance data continuity
- However, sorting has limitations
  - Time consuming
  - Extra index information must be stored
- Any solutions?
  - Data allow to be reordered without storing index information as long as *locations/indices of elements for same particle remain consistent* across arrays
  - For example:
    
    \[
    \begin{align*}
    vx &= (vx_1, vx_2, vx_3) \\
    vy &= (vy_1, vy_2, vy_3) \\
    vz &= (vz_1, vz_2, vz_3) \\
    xx &= (xx_1, xx_2, xx_3) \\
    yy &= (yy_1, yy_2, yy_3) \\
    zz &= (zz_1, zz_2, zz_3)
    \end{align*}
    \]

    \[
    \begin{align*}
    vx' &= (vx_3, vx_1, vx_2) \\
    vy' &= (vy_3, vy_1, vy_2) \\
    vz' &= (vz_3, vz_1, vz_2) \\
    xx' &= (xx_3, xx_1, xx_2) \\
    yy' &= (yy_3, yy_1, yy_2) \\
    zz' &= (zz_3, zz_1, zz_2)
    \end{align*}
    \]

    Reorder
    No need to store index information
Optimizations for MD Simulations – R-index Based Sorting

• Question: how to sort and make $v_x$, $v_y$, $v_z$, $x$, $y$, $z$ smoother at the same time?

• R-index based sorting proposed by CPC2000

  • Convert coordinate variables from FP values to integer number by dividing them by a user-set error bound

  • Generate R-index by interleaving binary representations of $x$, $y$, $z$

    $\begin{align*}
    x_{x_i} &= 110101111 \\
    y_{y_i} &= 001111100 \\
    z_{z_i} &= 10110001
    \end{align*}$

    $R_i = \begin{bmatrix} 101 & 001 & 110 & 011 & 010 & 011 & 101 \end{bmatrix}$

  • Sort all variables based on R-index value by segmentation
Optimizations for MD Simulations – R-index Based Sorting (cont.)

More **continuous** after R-index based sorting!

- We then apply SZ-LV on the sorted data, called **SZ-LV-RX**
- SZ-LV-RX improves compression ratio from 2.85 to **3.2** (12%)
- How to optimize *time consuming* problem?

<table>
<thead>
<tr>
<th></th>
<th>Segment Size</th>
<th>Compression Ratio</th>
<th>Compression Rate (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SZ-LV</strong></td>
<td>/</td>
<td>2.85</td>
<td>94.4</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>3.03</td>
<td>36.0</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>3.07</td>
<td>33.9</td>
</tr>
<tr>
<td></td>
<td>4096</td>
<td>3.11</td>
<td>34.3</td>
</tr>
<tr>
<td></td>
<td>8192</td>
<td>3.15</td>
<td>33.6</td>
</tr>
<tr>
<td></td>
<td>16384</td>
<td>3.20</td>
<td>35.0</td>
</tr>
</tbody>
</table>
Optimizations for MD Simulations – Partial R-index Based Sorting

- We propose partial R-index based sorting (PRX) scheme
- PRX: sorting started from the last $n$-th 3-bit using radix sorting
- Partial sorting can keep high smoothness and reduce execution time
- For example, performing PRX from the last third 3-bit like

| $xx_i$ | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| $yy_i$ | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| $zz_i$ | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| $R_i$  | 101 | 001 | 110 | 011 | 010 | 011 | 101 |

Radix sorting part | Ignored part

<table>
<thead>
<tr>
<th>Segment Size</th>
<th>Ignored Bits</th>
<th>Compression Ratio</th>
<th>Compression Rate (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SZ-LV</td>
<td>/</td>
<td>2.85</td>
<td>94.4</td>
</tr>
<tr>
<td>SZ-LV-PRX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16384</td>
<td>0</td>
<td>3.20</td>
<td>35.0</td>
</tr>
<tr>
<td>16384</td>
<td>2</td>
<td>3.20</td>
<td>36.6</td>
</tr>
<tr>
<td>16348</td>
<td>4</td>
<td>3.20</td>
<td>39.4</td>
</tr>
<tr>
<td><strong>16384</strong></td>
<td><strong>6</strong></td>
<td><strong>3.20</strong></td>
<td><strong>43.8</strong></td>
</tr>
<tr>
<td>16384</td>
<td>8</td>
<td>3.16</td>
<td>51.1</td>
</tr>
</tbody>
</table>

SZ-LV-PRX improves comp rate from 36 MB/s to **43.8 MB/s** (22%)
Optimizations for MD Simulations – SZ-CPC2000

- Further compression ratio optimization
  - CPC2000 compress sorted integer velocity values by variable-length coding method (differentiate adjacent values in bit-stream)
  - Suffer from high status bit overhead (1 ~ 10 bits per value)
  - Apply SZ-LV DIRECTLY on sorted floating-point velocity values

- Experimental evaluation
  - Further 10% improvement
Optimizations for Cosmology Simulations

- Construction of R-index based on (a) coordinates, (b) velocities, and (c) coordinates + velocities

<table>
<thead>
<tr>
<th></th>
<th>CPC2000</th>
<th>SZ-LV</th>
<th>SZ-LV + Coordinate-based R-index</th>
</tr>
</thead>
<tbody>
<tr>
<td>( xx )</td>
<td>7.1</td>
<td>8.18</td>
<td>8.55</td>
</tr>
<tr>
<td>( yy )</td>
<td>7.1</td>
<td>8.31</td>
<td>6.28</td>
</tr>
<tr>
<td>( zz )</td>
<td>7.1</td>
<td>5.92</td>
<td>6.73</td>
</tr>
<tr>
<td>( vx )</td>
<td>2.3</td>
<td>3.97</td>
<td>3.84</td>
</tr>
<tr>
<td>( vy )</td>
<td>2.3</td>
<td>3.92</td>
<td>3.79</td>
</tr>
<tr>
<td>( vz )</td>
<td>2.3</td>
<td>3.93</td>
<td>3.87</td>
</tr>
<tr>
<td>Overall</td>
<td>3.5</td>
<td>5.12</td>
<td>4.97</td>
</tr>
</tbody>
</table>

- Apply R-index sorting on HACC

Better
Worse
Worse
Optimizations for Cosmology Simulations (cont.)

- SZ-LV plus R-index sorting fail to improve the compression ratio of the whole data sets
- Unlike AMDF, not all variables in HACC are very disordered, e.g., \( yv \) is approximately sorted (in a wide-index range)
- Any attempt of reordering will lead to lower compression ratios
- Best solution for HACC: SZ-LV
Evaluation – Rate Distortion

(a) HACC

(b) AMDF

Table I: Descriptions of N-body simulation data sets used in the assessment

<table>
<thead>
<tr>
<th>Name</th>
<th># of Particles</th>
<th># of Snapshots</th>
<th>Data Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>HACC</td>
<td>147.3 million</td>
<td>500</td>
<td>1.8 TB</td>
</tr>
<tr>
<td>AMDF</td>
<td>2.8 million</td>
<td>500</td>
<td>34 GB</td>
</tr>
</tbody>
</table>
Reduce I/O time with 1,024 processes

- by 80% compared with writing initial data directly
- by 60% compared with second best solution
Conclusion

• We propose three different optimization techniques for molecular dynamics simulation that can improve compression ratio and compression rate
• We identify SZ-LV to be the best lossy compressor for cosmology simulation
• Our methods have the best rate-distortion (higher ratio, lower error) on the tested n-body simulation data compared with state-of-the-art compressors
• Our methods can reduce I/O time for parallel file system
• Future work
  • Evaluate our proposed methods on more particle simulation datasets
  • Propose more powerful method for cosmology datasets
Acknowledgement

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy’s Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation’s exascale computing imperative.
Thank you!

Welcome to use our SZ lossy compressor!
Any questions are welcome!

Contact:
Dingwen Tao (dtao001@cs.ucr.edu)
Sheng Di (sdi1@anl.gov)