Canopus: Enabling Extreme-Scale Data Analytics on Big HPC Storage via Progressive Refactoring

Tao Lu*, Eric Suchyta, Jong Choi, Norbert Podhorszki, and Scott Klasky, Qing Liu*, Dave Pugmire and Matt Wolf, and Mark Ainsworth+

* New Jersey Institute of Technology
Oak Ridge National Laboratory
† Brown University
Overview

• Background and related work
• Progressive data refactoring
• Conclusion
HPC systems

• Mission: illuminate phenomena that are often impossible to study in a laboratory [Oak Ridge National Lab, 2013]
  • Climate impacts of energy use
  • Fusion in a reactor not yet built
  • Galaxy formation

• Methodology: Modeling and simulation, along with data exploration
  • Data generation
  • Storage
  • Analysis
  • Visualization
HPC systems (cont’d)

• The big data management challenge [Shalf et. al., 2014]
  • Worsening I/O bottleneck for exascale systems
  • Exponentially increasing multi-scale, multi-physics data
Data compression in exascale (next-generation HPC) systems

• Goal: 10x to 100x data reduction ratio [Ian Foster et. al., 2017]
  • Reduce data by at least 90%

• Data features
  • Temporal and spatial
  • High-dimensional
  • Floating-point
Data compressors

• Lossless compression
  • Deduplication
  • GZIP
  • FPC [Burtscher et. al., 2009]

• Lossy compression
  • ZFP [Lindstrom et. al., 2014]
  • ISABELA [Lakshminarasimhan et.al., 2011]
  • SZ [Shen et. al., 2016]
Lossy compression

Original Data

1. Array linearization

2. Data transformation

3. Curve fitting

4. Optimize the unpredictable

Compressed data

Workflow of curve fitting based compression (e.g. ISABELA and SZ)

Floating-point Block:
0.368178
-1.269298
-0.904911
-0.242216

1. Align to a common exponent

2a. Encode exponent

2b. Convert mantissa to integer

Integers:
848960939002912256
-3157387122695243776
-2086581739634989568
-558511819984848000

3. Orthogonal transform, reorder, and convert to unsigned integer

Integers:
373945919058534944
395169966790744
44579205367707264
5803990578546072

4. Encode coefficients

Compressed high bits

Compression low bits

Workflow of quantization and transformation based compression (e.g. ZFP)
Can floating-point compressors achieve a near 100x compression ratio?
Performance of compressors

Relative error bound 0.000001

Relative error bound 0.001
Can floating-point compressors achieve a near 100x compression ratio?

Yes. If
Dataset contains a lot of identical values;
Or, data values highly skew with moderate compression error bounds.

No. For most datasets.

Can floating-point compressors achieve a 10x compression ratio?

Yes. For a lot of datasets with moderate compression error bounds.
What if the compression ratio is rushed to 100x by loosening error bounds?
Visualization and blob detection on compressed Dpot data.
Limitations of data compression by reducing floating-point accuracy

• Near 100x compression ratio is hardly achievable

• Lost data accuracy cannot be restored
Overview

• Background and related work

• Progressive data refactoring

• Conclusion
We propose *Canopus*

- Compressing HPC data in another dimension (resolution)
- Enabling progressive data refactoring
- User transparent implementation
<xml version="1.0"?>
<adios-config host-language="C">
  <adios-group name="field" stats="On">
    <var name="NX" type="integer"/>
    <var name="GX" type="integer"/>
    <var name="OX" type="integer"/>
    <var name="MX" type="integer"/>
    <var name="MY" type="integer"/>
    <global-bounds dimensions="GX" offsets="OX">
      <var name="dPot" gwrite="t" type="double" dimensions="NX" transform="none"/>
    </global-bounds>
    <var name="R" gwrite="g" type="double" dimensions="NX" transform="none"/>
    <var name="Z" gwrite="g" type="double" dimensions="NX" transform="none"/>
  </adios-group>
  <method group="field" method="CANOPUS">
    decimation-ratio=4;
    save-delta=1;
    compress-delta=0;
    compression-tolerance=0.001;
    thresh_type=absolute;
    thresh=40;
    method=MP1; path=data0; parameters;
    method=MP1; path=data1; parameters;
  </method>
  <buffer max-size-MB="10"/>
</adios-config>

Canopus I/O configuration
**Canopus: basic idea**

- Refactor the simulation results (via decimation) into a base dataset along with a series of deltas
- Base dataset is saved in fast devices, deltas in slow devices
- Base dataset can be used separately (at a lower resolution) for analysis
  - Selected subset of deltas to be retrieved to restore data to a target accuracy

---

**Canopus in HPC Systems**

- Simulation
  - ADIOS Write API
- Data Analytics
  - ADIOS Query API
- Canopus
  - (I/O, refactoring, compression, placement, retrieval, restoration)
  - ADIOS Kernel
    - (buffering, metadata, scheduling, etc.)
  - I/O Transport
    - MPI, MPI_AGGRGEATE, POSIX, Dataspaces, FLEXPATH, MPI_LUSTRE
- Storage Tiers
  - Node-local Storage (NVRAM, SSDs)
  - Burst Buffer
  - Remote Parallel File System
  - Campaign Storage

---

**Canopus in HPC Systems**
Canopus workflow

HPC Simulations (high accuracy)

Storage Hierarchy

Base
Delta_{2x}
Delta_{full}
ST_1
ST_2
ST_3

Refactoring (decimation, compression)

Retrieving & Reconstruction

Analytics Pipeline 1 (low accuracy)

base = L_{4x}

Analytics Pipeline 2 (medium accuracy)

base + delta_{2x}

Analytics Pipeline n (high accuracy)

base + delta_{2x} + delta_{full}
Data refactoring

1. Mesh decimation
2. Delta calculation
3. Floating-point compression
Mesh decimation

\[ V_{i}^{l+1} = \frac{1}{2} (V_{i}^{l} + V_{j}^{l}) \]

Delta Calculation

- For mesh data, it’s common that each vertex corresponds to a value (floating-point)
- After triangular mesh decimation:

\[ \delta_{n}^{l} = F(V_{n}^{l}) - \frac{1}{3} F(V_{i}^{l+1}) - \frac{1}{3} F(V_{j}^{l+1}) - \frac{1}{3} F(V_{k}^{l+1}) \]
Compression

• The floating-point values corresponding to vertexes are compressed using ZFP compressor

• A potential optimization to our framework is supporting adaptive compressors based on dataset features
Progressive data exploration (reverse the data refactoring procedures)

• I/O (read the base dataset and deltas)
• Decompression
• Restoration
Performance gain of Canopus for data analytics

(a) End-to-end time of the analytics pipeline.

(b) Restoring full accuracy data from the base dataset and deltas.
Impact on Data Analytics

Original

2x reduction

4x reduction

8x reduction

16x reduction

32x reduction
A quantitative evaluation of blob detection

(a) Number of blobs detected

(b) Avg. blob diameters

(c) Blob area

(d) Blob overlapping ratio
Overview

• Storage stacks of HPC systems
• Progressive data refactoring
• Conclusion and future work
Conclusion

• Lossy compression may devastate the usefulness of data to achieve high compression ratio (such as 100x)

• It is critical to compress data in multiple orthogonal dimensions such as accuracy and resolution

• Canopus combines mesh compression and floating-point compression, possibly delivering a high compression ratio without devastate the usefulness of data
Future work

• Investigate the impact of lossy compression on analytical applications other than visualization
  • Original data $A == B$, compressed data $A' == B'$?
  • $F(D) == F(D')$? $F$ is a function
References

• Oak Ridge National Lab, *Solving Big Problems: Science and Technology at Oak Ridge National Laboratory*, 2013


Thanks & Questions