Scaling Resiliency via machine learning and compression

Alok Choudhary

Henry and Isabel Dever Professor
EECS and Kellogg School of Management
Northwestern University
choudhar@eecs.northwestern.edu

Founder, chairman and
Chief Scientist
4Cinsights Inc: A Big
Data Science Company
+1 312 515 2562
alok@4Cinsights.com
Motivation

• Scientific simulations
  • Generate large amount of data.
  • Data feature: high-entropy, spatial-temporal

• Exascale Requirements*
  • Scalable System Software: Developing scalable system software that is power and resilience aware.
  • Resilience and correctness: Ensuring correct scientific computation in face of faults, reproducibility, and algorithm verification challenges.

• NUMARCK (NU Machine learning Algorithm for Resiliency and ChecKpointing)
  • Learn temporal relative change and its distribution and bound point-wise user defined error.

* From Advanced Scientific Computing Advisory Committee Top Ten Technical Approaches for Exascale
Checkpointing and NUMARACK

- Traditional checkpointing systems store raw (and uncompressed) data
  - cost prohibitive: the storage space and time
  - threatens to overwhelm the simulation and the post-simulation data analysis
- I/O accesses have become a limiting factor to key scientific discoveries.

Solution? NUMARCK
What if a Resilience and Checkpointing Solution Provided

- Improved Resilience via more frequent yet relevant checkpoints, while
- Reducing the amount of data to be stored by an order of magnitude, and
- Guaranteeing user-specified tolerable maximum error rate for each data point, and
- an order of magnitude smaller mean error for each data set, and
- reduced I/O time by an order of magnitude, while
- Providing data for effective analysis and visualization
Motivation: “Incompressible” with Lossless Encoding

Shannon’s Information Theory:

\[
H(X) = \sum_{i=1}^{n} p(x_i) \log p(x_i)
\]

Compressible Exponent.
Low Entropy.

Incompressible mantissa.
Less predictable.
High Entropy.

Probability distribution of more common bit value

Bit position of double precision rlds data
Motivation: Still “Incompressible” with Lossy Encoding

- Highly random
- Extreme events missed

~0.35 correlation!
Observations:

• Variable Values – distribution
• Change in Variable Value – distribution
• Relative Change in Variable Value - distribution

Hypothesis: The relative changes in variable values can be represented in a much smaller state space.

• A1(t) = 100, A1(t+1) = 110 => change = 10, rel change = 10%
• A2(t) = 5, A2(t+1) = 5.5 => change = .5, rel change = 10%
Sneak Preview: Relative Change is more predictable

Iteration 1 and 2 on climate CMIP5 rlus data

Relative Change between iteration 1 and 2 on climate CMIP5 rlus data
Challenges

• How to learn patterns and distributions of relative change at scale?
• How to represent distributions at scale?
• How to bound errors?

• System Issues
  • data movement
  • I/O
  • Scalable software
  • Reconstruction when needed
NUMARCK Overview

Traditional checkpointing:
- Full checkpoint in each checkpoint

Machine learning based checkpointing:
- Forward Predictive Coding
  - Transform the data by computing relative changes in ratio from one iteration to the next
- Data Approximation
  - Learn the distribution of relative change $r$ using machine learning algorithms and store approximated values

F: Full checkpoint
C: change ratios
NUMARCK: Overview

Forward coding

~0.99 correlation!
0.001 RMSE

Distribution Learning
E.g., Distribution Learning Strategies

- Equal-width Bins (Linear)
- Log-scale Bins (Exponential)
- Machine Learning – Dynamic clustering

Number of bins or clusters depends on the bits designated for storing indices and error tolerance examples

- index length (B): 8 bits
- tolerable error per point (E): 0.1%

the number of clusters
the width of each cluster
Equal-width distribution

In each iteration, partition value into 255 bins of equal-width. Each value is assigned to a corresponding bin ID (represented by the center of bin). If the difference between the original value and the approximated one is larger than user-specified value (0.1%), it is stored as it is (i.e., incompressible).

Pros: Easy to Implement

Cons: (1) Can only cover range of $2^E(2^B-1)$; (2) Bin width: $2^E$
In each iteration, partition the ratio distribution into 255 bins of log-scale width.

Pros: cover larger range and more finer (narrower) bins
Cons: may not perform well for highly irregularly distributed data

dens: iteration 32 to 33
In each iteration, partition the ratio data into 255 clusters using (e.g., K-means) clustering, followed by approximated values based on corresponding cluster’s centroid value.
## Methodology Summary

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initialization</strong></td>
<td>• this is the model, initial condition and metadata</td>
</tr>
<tr>
<td><strong>Calculation</strong></td>
<td>• Calculate the relative change</td>
</tr>
<tr>
<td><strong>Learning Distributions</strong></td>
<td>• Bin the relative change into N bins</td>
</tr>
<tr>
<td></td>
<td>• Index and Store bin IDs</td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td>• Store index, compress index</td>
</tr>
<tr>
<td></td>
<td>• Store exact values for change outside error bounds</td>
</tr>
<tr>
<td><strong>Reconstruction</strong></td>
<td>• Read last available complete checkpoint</td>
</tr>
<tr>
<td></td>
<td>• Reconstruct data values for each data point, can report the error bounds.</td>
</tr>
</tbody>
</table>
NUMARCK Algorithm

• Change ratio calculation
  – Calculate element-wise change ratios
• Bin histogram construction
  – Assign change ratios within an error bound into bins
• Indexing
  – Each data element is indexed by its bin ID
• Select top-K bins with most elements
  – Data in top-K bins are represented by their bin IDs
  – Data out of top-K bins are stored as is
• (optional) Apply lossless GNU ZLIB compression on the index table
  – Further reduce data size
• (optional) File I/O
  – Data is saved in self-describing netCDF/HDF5 file
Experimental Results: Datasets

• FLASH code is a modular, parallel multi-physics simulation code: developed at the FLASH center of University of Chicago
  – It is a parallel adaptive-mesh refinement (AMR) code with block-oriented structure
  – A block is the unit of computation
  – The grid is composed of blocks
  – Blocks consists of cells: guard and interior cells
  – Cells contains variable values

• CMIP - supported by World Climate Research Program: (1) Decadal Hindcasts and predictions simulations; (2) Long-term simulations; (3) atmosphere-only simulations.

var 0, 1, 2, ..., 23 (e.g., density, pressure and temperature)
Evaluation metrics

• Incompressible ratio
  • % of data that need to be stored as exact values because it would be out of error bound if approximated

• Mean error rate
  • Average difference between the approximated change ratio and the real change ratio for all data

• Compression ratio
  • Assuming data D of size |D| is reduced to size |D'|, it is defined as:

\[
\frac{|D| - |D'|}{|D|} \times 100
\]
Incompressible Ratio: Equal-width Binning

FLASH dataset, 0.1% error rate
Incompressible Ratio: Log-scale Binning

FLASH dataset, 0.1% error rate
Incompressible Ratio: Clustering-based Binning

FLASH dataset, 0.1% error rate
Mean Error Rate: Clustering-based

FLASH dataset, 0.1% error rate
Increasing Index Size:
Incompressible Ratio

% of data needed to be stored as exact values (i.e., uncompressible)

- rlds-8
- rlds-9
- rlds-10

Increasing bin sizes (8-bit to 10-bit) reduces % of incompressible significantly.

Note: rlds is the most difficult to compress with 8-bit
Increasing bin sizes (8-bit to 10-bit) increases compression ratio significantly.
Different Tolerable Error Rates: Incompressible Ratio (0.1% - 0.5%)
Scaling - Experimental Settings

<table>
<thead>
<tr>
<th>Name of data set</th>
<th>Application</th>
<th>Domain</th>
<th>Size per iteration</th>
<th>Variable dimension</th>
<th>Variable size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sedov</td>
<td>FLASH</td>
<td>Astrophysics</td>
<td>15MB</td>
<td>165<em>32</em>32*1</td>
<td>1.3MB</td>
</tr>
<tr>
<td>Stir-1</td>
<td>FLASH</td>
<td>Astrophysics</td>
<td>3.7GB</td>
<td>64<em>157</em>157*157</td>
<td>945MB</td>
</tr>
<tr>
<td>Stir-2</td>
<td>FLASH</td>
<td>Astrophysics</td>
<td>296GB</td>
<td>1024<em>314</em>314*157</td>
<td>59GB</td>
</tr>
<tr>
<td>Stir-3</td>
<td>FLASH</td>
<td>Astrophysics</td>
<td>2.4TB</td>
<td>8192<em>314</em>314*157</td>
<td>472GB</td>
</tr>
<tr>
<td>ASR</td>
<td>ASR</td>
<td>Climate</td>
<td>103MB</td>
<td>29<em>320</em>320</td>
<td>11MB</td>
</tr>
<tr>
<td>CMIP</td>
<td>CMIP3</td>
<td>Climate</td>
<td>19GB</td>
<td>42<em>2400</em>3600</td>
<td>1.4GB</td>
</tr>
</tbody>
</table>

- Data sets and environment:
  - FLASH datasets
    - SuperMUC at Leibniz Supercomputing Centre, Germany, a parallel computer consists of 9216 nodes (16 cores per node)
    - We used up to 12,800 cores in our experiments
  - Others
    - A Linux machine, 2 quad-core CPUs (32 GB memory)
Compression ratios

- Compared with lossy compression algorithms: ZFP (LLNL), ISABELLA (NCSU)

![Compression ratios graph showing iterations and compression ratios for NUMARCK, ISABELLA, and ZFP algorithms. The graph indicates that NUMARCK has a consistently high compression ratio across iterations, while ISABELLA and ZFP have lower ratios with some variability. The X-axis represents iterations while the Y-axis represents compression ratios. CMIP (1.4 GB) is noted at the bottom right of the graph.]
Scalability Experiments

FLASH datasets (turbulence stirring test)

- **Stir-2 (59GB) data**
  - Numbers of cores: 1600
  - Speed-up: 1404
  - Time: 2.655 sec
  - Original I/O time: 13.2 sec/iteration

- **Stir-3 (472GB) dataset**
  - Number of cores: 12800
  - Speed-up: 8788
  - Time: 3.610 sec
  - Original I/O time: 18.0 sec
Open Problems and Challenges

• Optimize and/or create new machine learning algorithms
  – for higher compressions and more accurate representation
  – Scalable implementation
  – Learning from historical results to optimize the “learning step”
    for minimizing data movement and power
  – Adaptation for anomaly detection (for resilience and analysis)
• Use of memory hierarchy and local SSDs
• Incorporation into pNetCDF etc and libraries
• I/O optimizations
THANK YOU!

Alok Choudhary
Henry and Isabel Dever Professor
EECS and Kellogg School of Management
Northwestern University
choudhar@eeecs.northwestern.edu