AMRIC: A Novel In Situ Lossy Compression Framework for Adaptive Mesh Refinement Applications Use HDF5

Daoce Wang⁴, Jesus Pulido², Pascal Grosset², Jiannan Tian¹, Sian Jin¹, Houjun Tang³, Jean Sexton³, Sheng Di⁴
Zarija Lukić³, Kai Zhao⁵, Bo Fang⁶, Franck Cappello⁴, James Ahrens², Dingwen Tao¹

¹Indiana University, ²Los Alamos National Lab
³Lawrence Berkeley National Lab, ⁴Argonne National Lab
⁵Florida State University, ⁶Pacific Northwest National Lab,
AMRIC, In situ AMR Compression: Background of AMR

Introduction to AMR

• Each mesh represents a value of an area.
  • Smaller mesh $\rightarrow$ higher resolution
• Change the mesh (spatial resolution) based on the level of refinement needed by the simulation, use finer mesh in “more important” region
  • Achieve the desired accuracy as well as increase computational and storage savings.
• Result in hierarchical data with different resolutions
• One of the most widely used frameworks for HPC apps

https://www.cttc.upc.edu/?q=node/165
Why HDF5?

Streamlined (de)compression
- Data can be (de)compressed using a (de)compression filter during write/read operations
  - For compression: set the filter and call H5Dwrite
  - For decompression: call H5Dread

Better usability, especially for the AMR data
- AMR data has a hierarchical nature which aligns well with HDF5
  - Contains different lvl & dataset, which can be easily managed using H5
  - Contains lots of metadata which can be easily accessed & manage
    - h5dump -A
AMRIC: HDF5 Compression Filter Modification

1. Compression-oriented preprocessing workflow for AMR data
2. Optimize the state-of-the-art SZ lossy compressor’s efficiency for AMR data
3. Overcome the gap between the HDF5 and AMR applications → bigger chunk
   • Modifying the AMR data layout
   • Modifying the HDF5 compression filter mechanism

HDF5 need chunked data for compression filters → What is the best chunk size?
   • We want a large chunk size in terms of compression
     • Small chunk → too many of data blocks → low compression ratio & I/O perf
   • HDF5 may not prefer too large chunk
     • I/O load imbalance
     • Cache size issue
     • Memory overhead
   • Compression perf vs HDF5 Perf?

Figure 1: Data array is logically split into equally sized chunks each of which is stored separately in the file.
AMRIC: Evaluation on Compression perf

Boost compression perf for AMR applications

Original, CR = 19.0
small chunk size

Our AMRIC, CR = 23.9
Large chunk size
AMRIC: Evaluation on I/O Time

Up to $10.5 \times$ I/O performance improvement over the non-compression solution. Up to $39 \times$ over the previous compression solution (w/ small chunk)

Figure 17: Writing time of WarpX runs with different scales (in a weak scaling study). Log scale is used here for better comparison.

Figure 18: Writing time of Nyx runs with different scales. Log scale is used for better comparison.