I/O Optimization for ENZO Cosmology Simulation

Using MPI-IO

Jianwei Li 12/06/2001
Outline Today

- Introduction to ENZO Cosmology Simulation
- I/O Analysis for ENZO Cosmology Simulation
- Introduction to MPI-IO
- MPI-IO Approaches for ENZO
- Conclusion and Future Work
ENZO Cosmology Simulation

ENZO is a parallel, 3D cosmology hydrodynamics code with structured adaptive mesh refinement (AMR), using grid and its evolution to represent the space. Parallelism is achieved by domain decomposition of the root grid of the AMR grid hierarchy and distribution of the grids among many processors.

The Cosmology Simulation is one of the problem types for ENZO code. It simulates the formation of the astrospace by evolving the initial grid and taking “slap-shots” during each cycle of evolutions until it reaches to some end status.
Application Data Structure

Grid Data
- Grid Data on Proc 0
- Grid Data on Proc 1
- Grid Data on Proc 2
- Grid Data on Proc 3
- Grid Data on Proc 4
- Grid Data on Proc 5
- Grid Data on Proc 6
- Grid Data on Proc 7

Level 0
- Topgrids
- Level 0 subgrids

Level 1
- Subgrids

Level 2
- Subgrids
I/O Analysis

• I/O based on Grid – a C++ object consisting of many large datasets
• Datasets and their Partition & Distribution

**Baryon Fields – 3D (B, B, B)**

- **Density Field**
- **VelocityX Field**
- **VelocityY Field**
- **VelocityZ Field**

**Particle Datasets - 1D, Irregular**

- **ParticleNumber**
- **ParticlePosition X**
- **ParticlePosition Y**
- **ParticlePosition Z**
- **ParticleVelocity X**
- **ParticleVelocity Y**
- **ParticleVelocity Z**
- **ParticleMass**
- **ParticleAttributes**
Original I/O Approach

• Serial Read
  – Processor 0 read in all grids
  – Processor 0 partition grids and distribute their datasets
    Partition grids: based on their boundaries (LeftEdges, RightEdges)
    Distribute Baryon Fields: 3D (B, B, B)
    Particle Datasets: 1D, based on particle position falling within each partitioned grid’s boundary

• Write
  – Combine partitioned topgrids on each processor
to a single TopGrid on Processor 0
    Combination: Baryon Fields: 3D (B, B, B)
    Particle Datasets: 1D, based on order of particle number
  – Processor 0 write TopGrid to a single file
  – Each processor write its own subgrids out to separate Files
Introduction to MPI-IO

- MPI-2; ROMIO; MPICH, HP MPI, SGI MPI
- High level interface of a parallel I/O system
- Support partition of file data among procs
- Data partition and access pattern by derived datatypes
- Data-sieving - strided/noncontiguous accesses
- Collective interface – distributed array accesses
- Support asynchronous I/O
Date-sieving and Collective I/O

Reduce number of I/O requests, thus reduce the effect of high I/O latency
MPI-IO Approaches for ENZO

• For Regular Partition,
  Use derived datatypes and make use of the built-in data-sieving and collective interface of MPI-IO

• For Irregular Partition,
  Use simple block-wise MPI I/O and implement the data-sieving and two-phase I/O at the application level.
Implementation (1)

1. New Simulation Read Grid (Datasets)
   Baryon Fields: 3D (Block, Block, Block)
   View the file data as a 3D array, and the requested dataset part as its subarray defined by the offsets and sizes in each dimension; Then use MPI collective read.

   Particle Datasets: 1D Irreg(distributed by ParticlePosition)
   Partition the datasets block-wise, perform contiguous MPI read; Check the ParticlePosition and grid edges to determine which processor each particle belongs to, then perform communication to send particles to their right processors, where right means that particle falls within the grid boundary of the processor.
Implementation (2)

2. Write TopGrid (Datasets)

Baryon Fields: 3D (Block, Block, Block)

View the file data as a 3D array, and the requested dataset part as its subarray defined by the offsets and sizes in each dimension; Then use MPI collective write.

Particle Datasets: 1D Irreg(combined by the order of ParticleNumber)

Perform a parallel sort by the ParticleNumber and redistribute the particle data by checking which processor each particle belongs to so that its ParticleNumber falls between the ParticleNumber bounds of the processor; then perform block-wise MPI write.
Implementation (3)

3. **Write Subgrids (Datasets)**
   All processors write their own subgrids into the same single file.

4. **Restart Read Grids (Datasets)**
   Topgrid is read in the same way as new simulation read;
   Subgrids are read by processors in a round-robin manner;
   Since all subgrids are written in a single file, there can be further improvement with I/O balance, pre-fetching …
Conclusion and Future Work

ENZO I/O can be effectively improved by using MPI-IO

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Pattern</th>
<th>Regular</th>
<th>Irregular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td>Pattern</td>
<td>Strided or contiguous derived datatypes to define data partition.</td>
<td>Block-wise partition plus communication and/or data-sieving.</td>
</tr>
<tr>
<td>Dimension</td>
<td>1-Dimensional</td>
<td>Subarray/other more complicated derived datatype may be used.</td>
<td>Depending on pattern choose simple I/O pattern plus comm.</td>
</tr>
</tbody>
</table>
Future Work

Using MDMS on ENZO application

Useful MetaData:

- Dataset Rank and Dimensions
- Dataset Partition & Access Patterns
- Offsets and Sizes for Each Access of Datasets
- Order, Loop Access of Datasets