ADA: An Application-Conscious Data Acquirer for Visual Molecular Dynamics

Tongliang Deng¹, Hanpei Wu², Yanliang Zou², Si Chen³, Tao Xie⁴, Shu Yin²,⁵,⁶,⁷

¹SenseTime Research; ²ShanghaiTech University; ³West Chester University; ⁴San Diego State University; ⁵Shanghai Engineering Research Center of Intelligent Vision & Imaging; ⁶CAS, Shanghai Institute of Microsystem & Information Technology; ⁷State Key Lab of High Performance Computing
Outline

• Background
• Motivation
• Design
• Evaluation
• Conclusion
Background: Molecular Dynamics (MD)

• MD is a computer simulation technique that simulate motion of molecular systems, widely used in biology, physics, chemistry and materials domains.

• For serving different needs, there are many MD applications, including NAMD, GROMACS, BIOVIA, MATERIALS STUDIO...
Background: Molecular Dynamics (MD)

- MD applications accept initial state and environment from input metadata, calculate position and speed for next moment using force field in last moment, write the simulation result of each moment into a trajectory data.
- Trajectory data records space coordinates of each particle in each frame. Large trajectory data often be compressed for storage and transmission.
- Scientist use MD result to observe changes in the microsystem, but they cannot directly observe or analysis from coordinates numbers which may even be compressed.
Background: Visualization for MD Result

- Visualization application for MD decompress and render coordinates to 3D animations, help scientists observe the trend or the key time point of the changes.
- Workflow, Visualization App and animation example:

```
metadata

MD application

trajectory data

visualize application
```

- PyMOL by Schrödinger
- VMD: Visual Molecular Dynamics
- UCSF CHIMERA: an Extensible Molecular Modeling System
Background: GPCR Research

• G-protein coupled receptors (GPCR) is a group of proteins on cell membrane. GPCR research reveal the drug target and physiological mechanism.
• G-protein coupled receptors (GPCR) is the target of about 40% modern medicine.
• So GPCR become a frontier focus of biological research. ShanghaiTech iHuman Institute published many important breakthroughs about GPCR on Science, Cell and Nature.
Outline

• Background
• Motivation
• Design
• Evaluation
• Conclusion
GPCR MD

• To research principles of drug action on GPCR, researchers use GROMACS to simulate, use VMD to render.

• GPCR is on cell membrane, both sides of the membrane are water. MD need to reappear GPCR environment for simulation, so metadata and trajectory both contain water.

• VMD render received trajectory, so VMD also load and render water.
GPCR MD

- Water trajectory is not the focus of observe.
- We define trajectory data into 2 types:
  - Water
  - Non-water (protein, ligand, phospholipid, ...)
- We only need to visualize non-water data, non-water data only accounts for less than 50%.
  - Load water data requests more memory.

<table>
<thead>
<tr>
<th>Number of frames</th>
<th>Compressed file size (MB)</th>
<th>Non-water data fraction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Complete data</td>
<td>Non-water data</td>
</tr>
<tr>
<td>626</td>
<td>100</td>
<td>44</td>
</tr>
<tr>
<td>1251</td>
<td>200</td>
<td>98</td>
</tr>
<tr>
<td>5006</td>
<td>800</td>
<td>348</td>
</tr>
</tbody>
</table>

Non-water

water
Motivation: VMD dataflow

- VMD need Data Pre-processing before Data Rendering.
- Time-consuming redundant effort over repeatedly study in Data Pre-processing. Redundant time accounted for over 50% CPU burst time.
Outline

• Background
• Motivation
• **Design**
• Evaluation
• Conclusion
Design: Overview

• Design goal
• ADA Architecture
  – Data Pre-Processor
  – I/O Determinator
• ADA workflow
  – Data Writing
  – Data Retrieving
Design Goal

• ADA is a middleware between compute node and storage node
• ADA should perform Data pre-processing before data retrieving – avoid redundant work
• ADA only provide the needed data – domain scientists focus on

(a) Traditional VMD (b) ADA-assisted VMD
Computational Biology Applications
(e.g. visual molecular dynamics)

User API Layer
- High-Level I/O Libraries
- MPI-IO
- POSIX

ADA: Application-Conscious Data Acquirer
- Data Pre-Processor
- Data Categorizer
- Label Manager
- Data Decompressor
- Labeler
- Indexer
- I/O Determinator
- I/O Dispatcher
- I/O Retriever

Underlying File System(s)
(e.g. xfs, ext4, Lustre, PVFS, etc.)
ADA Module Function

- **Data Decompressor**: Decompress data into raw data.
- **Data Categorizer**: Categorize data into subsets.
- **Labeler**: Assign a tag to a data subset.
- **Indexer**: Map the tag with data subsets, get path from it.
- **I/O Dispatcher**: Dispatch data subset to underlying File System via tag.
- **I/O Retriever**: Retrieve data subset via tag.
ADA Dataflow - Writing

Data

Data Decompressor
RAW Data
Data Categorizer

Data Subset 1
Data Subset i
Data Subset m
Data Subset n

Labeler

Data Subset 1
Tag 1 (Path to FS1)
Data Subset m
Tag m (Path to FS1)
Data Subset i
Tag i (Path i to FS2)
Data Subset n
Tag n (Path n to FS2)

I/O Dispatcher

ADA

I/O Retriever

Data Requests
Data Queries

Tags

Paths

FS1
Data Subset 1
Data Subset m
FS2
Data Subset 1
Data Subset m

Requested Data
Data Subset 1
Data Subset n

ShanghaiTech University

16
ADA Dataflow - Retrieving

• Example commandline: `mol addfile /mnt/bar.xtc tag 1, ..., i`

![Diagram of ADA Dataflow](image-url)

- Data Decompressor
- Data Categorizer
- Labeler
- Indexer
- I/O Dispatcher
- I/O Retriever

Data Flow:
- RAW Data → Data Decompressor → Data Categorizer → Labeler → Indexer → I/O Dispatcher → I/O Retriever → Requested Data

- Tags: Tag 1 (Path to FS1), Tag m (Path to FS1), Tag i (Path to FS2), Tag n (Path to FS2)

- Paths: (Path to FS1), (Path i to FS2), (Path n to FS2)

Example Commandline:
```
mol addfile /mnt/bar.xtc tag 1, ..., i
```
Outline

• Background
• Motivation
• Design
• **Evaluation**
• Conclusion
Evaluation: Experimental Setup

- **Hardware**
  - Single-node server:
    - Intel Xeon CPU E5-2603 v4 @1.70GHz, 16GB DRAM
    - Two 256GB NVMe SSD
    - CentOS release 6.10 (Final) with ext4
  - 9-node hybrid PVFS cluster:
    - 3*compute node: 6-core Intel Xeon CPU E5-2603 v4 @1.70GHz, 16GB DRAM
    - 3*1TB HDD+3*256GB NVMe SSD
    - CentOS release 6.10 (Final) with PVFS2(OrangeFS 2.8.2)
  - Fat Node:
    - Intel Xeon CPU E7-4802 v3 @1.90GHz, 1007GB DRAM
    - 10*1TB WD HHDs(RAID 0)
    - The CentOS release 7.3 with 3.10 kernel with XFS
Evaluation: Experimental Setup

- Data set for single-node server and PVFS cluster
  - Real world GPCR data, collected from ShanghaiTech iHuman
  - Up to 5006 frames. Compressed file up to 800MB, which become 2612MB after decompress, 1108MB of them are decompressed non-water data.

<table>
<thead>
<tr>
<th>Number of Frames</th>
<th>Loaded Data (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PVFS (compressed)</td>
</tr>
<tr>
<td>626</td>
<td>100</td>
</tr>
<tr>
<td>1251</td>
<td>200</td>
</tr>
<tr>
<td>1877</td>
<td>300</td>
</tr>
<tr>
<td>2503</td>
<td>400</td>
</tr>
<tr>
<td>3129</td>
<td>500</td>
</tr>
<tr>
<td>3754</td>
<td>600</td>
</tr>
<tr>
<td>4380</td>
<td>700</td>
</tr>
<tr>
<td>5006</td>
<td>800</td>
</tr>
</tbody>
</table>
Evaluation: Experimental Setup

- Data set for Fat Node
  - Real world GPCR data, collected from ShanghaiTech iHuman
  - Much larger Data set
    - More frames, larger data size (10x than previous),
    - Up to 5,004,800 frames. Compressed file up to 800GB, which become 2612.8GB after decompress, 1108.8GB of them are decompressed non-water data.

<table>
<thead>
<tr>
<th>Number of Frames</th>
<th>XFS (compressed)</th>
<th>ADA (all)</th>
<th>ADA (non-water)</th>
</tr>
</thead>
<tbody>
<tr>
<td>62,560</td>
<td>10</td>
<td>32.7</td>
<td>13.9</td>
</tr>
<tr>
<td>187,680</td>
<td>30</td>
<td>98</td>
<td>41.6</td>
</tr>
<tr>
<td>312,800</td>
<td>50</td>
<td>163.3</td>
<td>69.3</td>
</tr>
<tr>
<td>437,920</td>
<td>70</td>
<td>228.6</td>
<td>97</td>
</tr>
<tr>
<td>625,600</td>
<td>100</td>
<td>326.6</td>
<td>138.6</td>
</tr>
<tr>
<td>938,400</td>
<td>150</td>
<td>489.9</td>
<td>207.9</td>
</tr>
<tr>
<td>1,251,200</td>
<td>200</td>
<td>653.2</td>
<td>277.2</td>
</tr>
<tr>
<td>1,564,000</td>
<td>250</td>
<td>816.5</td>
<td>346.5</td>
</tr>
<tr>
<td>1,876,800</td>
<td>300</td>
<td>979.8</td>
<td>415.8</td>
</tr>
<tr>
<td>2,502,400</td>
<td>400</td>
<td>1306.4</td>
<td>554.4</td>
</tr>
<tr>
<td>3,440,800</td>
<td>550</td>
<td>1796.3</td>
<td>762.3</td>
</tr>
<tr>
<td>4,379,200</td>
<td>700</td>
<td>2286.2</td>
<td>970.2</td>
</tr>
<tr>
<td>5,004,800</td>
<td>800</td>
<td>2612.8</td>
<td>1108.8</td>
</tr>
</tbody>
</table>
Evaluation: a Single-Node Server

- Evaluation on a single-node SSD Server
  - 14x faster, save 2.35x memory
Evaluation: CPU Burst Time

- CPU burst time comparison.
- Decompression weight more than 50%, which can be eliminated in ADA.
Evaluation: a 9-Node Hybrid PVFS Cluster

- Using the same data set as single-node server used.
- Evaluation on a 9-node hybrid PVFS cluster
  - 10x faster, save 2.35x memory
Evaluation: Fat Node

• Much larger data set to reach the memory bottleneck.
• Evaluation on Fat Node
  – 5.2x faster, save 2.28x memory, render 2.3x frames
Evaluation: Energy consumption

- Real world calculated by A PDU, test in Fat Node.
- ADA consume only 1780 KJ energy while ordinary VMD consume 9283 KJ, saved up to 5.21x energy (80.8%).
  - Device limited energy consumption is 12378 KJ (XFS) and 4252 KJ (ADA), ADA saved 2.91x energy.
Results Summary

• Evaluation on a Single-Node Server
  – 14x faster, save 2.35x memory

• Evaluation on a Small Cluster
  – 10x faster, save 2.35x memory

• Evaluation on a Fat Node
  – 5.2x faster, save 2.28x memory
  – Save 2.91x-5.21x energy
  – Render 2.3x more frames
Outline

• Background
• Motivation
• Design
• Evaluation
• Conclusion
Conclusion

• Identify a data processing challenge faced by many molecular dynamics applications for visualization and analysis.

• A lightweight file system called ADA.
  – A Dedicated *middleware* for VMD.
  – ADA saves up to 14x time, up to 2.35x memory, 2.91x to 5.21x energy.
  – ADA can render much larger (2.3x) trajectory data.