Ascent: Flyweight In Situ Visualization and Analysis for HPC Simulations

ATPESC 2023

Monday August 7th, 2023

Cyrus Harrison (LLNL), Nicole Marsaglia (LLNL)
Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy’s Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation’s exascale computing imperative.
ATPESC 2023: Exploring Visualization with Jupyter Notebooks

Tutorial Plan

- Short Ascent Overview (~10 min)
- Follow along Ascent Jupyter Tutorial using cloud hosted Jupyter Notebooks (~30 min)
ATPESC 2023: Exploring Visualization with Jupyter Notebooks

- Ascent is an in situ visualization and analysis library for HPC Codes
- Instead of a traditional UI, we use Jupyter Notebooks for some workflows and our Ascent tutorials
- Ascent’s tutorial materials provide examples of using Jupyter for scientific visualization
- For those interested in more details about how we use Jupyter:
  - Source for our Jupyter Widgets:
    * https://github.com/Alpine-DAV/ascent/blob/develop/src/ascent/python/ascent_module/py_src/jupyter.py
  - Dockerfile for the Jupyter Container we use for the tutorial:
    * https://github.com/Alpine-DAV/ascent/blob/develop/src/examples/docker/ubuntu/Dockerfile
  - Research about connecting simulation codes to Jupyter using Ascent (not demonstrated today)
    * Interactive in situ visualization and analysis using Ascent and Jupyter
    * https://dl.acm.org/doi/10.1145/3364228.3364232
Important links and contact info:

Ascent Resources:
- Github: https://github.com/alpine-dav/ascent
- Docs: http://ascent-dav.org/
- Tutorial Landing Page: https://www.ascent-dav.org/tutorial/

Contact Info:
Cyrus Harrison: cyrush@llnl.gov
Nicole Marsaglia: marsaglia1@llnl.gov
Ascent is an easy-to-use flyweight in situ visualization and analysis library for HPC simulations

- Easy to use in-memory visualization and analysis
  - Use cases: *Making Pictures, Transforming Data*, and *Capturing Data*
  - Young effort, yet already supports most common visualization operations
  - Provides a simple infrastructure to integrate custom analysis
  - Provides C++, C, Python, and Fortran APIs

- Uses a flyweight design targeted at next-generation HPC platforms
  - Efficient distributed-memory (MPI) and many-core (CUDA, HIP, OpenMP) execution
    - Demonstrated scaling: In situ filtering and ray tracing across **16,384 GPUs** on LLNL’s Sierra Cluster
  - Has lower memory requirements than current tools
  - Requires less dependencies than current tools (ex: no OpenGL)
    - Builds with Spack [https://spack.io/](https://spack.io/)

Visualizations created using Ascent

Extracts supported by Ascent

[http://ascent-dav.org](http://ascent-dav.org)
[https://github.com/Alpine-DAV/ascent](https://github.com/Alpine-DAV/ascent)

Website and GitHub Repo
Ascent supports common visualization use cases

- Clips
- Iso-Volume
- Threshold
- Slice
- Contour

Rendering
- Pseudocolor
- Volume
- Mesh

[powered by]

VTK
RAJA
mfem
Devil Ray
Ascent supports common analysis use cases

expression:
\[
\begin{align*}
\text{du} &= \text{gradient}(\text{field}('velocity','u')) \\
\text{dv} &= \text{gradient}(\text{field}('velocity','v')) \\
\text{dw} &= \text{gradient}(\text{field}('velocity','w')) \\
w_x &= dw.y - dv.z \\
w_y &= dw.z - dv.x \\
w_z &= dw.x - dv.y \\
\text{vector}(w_x, w_y, w_z) \\
\text{name: vorticity}
\end{align*}
\]

condition:
\[
\text{entropy} - \text{history(entropy, relative_index = 1)} > 0.5
\]

Derived Fields  |  Time Histories  |  Lineouts and Spatial Binning
---|---|---

Triggers

Extracts

Scalar Images  |  HDF5 Files  |  Cinema Databases

python  |  jupyter
Science Enabling Results: Shock Front Tracking (VISAR)

Velocity interferometer system for any reflector (VISAR)

Shock position tracked in Ascent
Science Enabling Results: Simulation Validation

Simulated

Experimental

Radiographs
Science Enabling Results: WarpX Workflow Tools (Jupyter Lab)
Science Enabling Results: Rendering At Scale (2018)

- The **97.8 billion** element simulation ran across **16,384 GPUs** on **4,096 Nodes**

- The simulation application used CUDA via RAJA to run on the GPUs

- Time-varying evolution of the mixing was visualized in-situ using Ascent, also leveraging 16,384 GPUs

- Ascent leveraged VTK-m to run visualization algorithms on the GPUs

Visualization of an idealized Inertial Confinement Fusion (ICF) simulation of Rayleigh-Taylor instability with two fluids mixing in a spherical geometry.
You will learn:

- How to use Conduit, the foundation of Ascent’s API
- How to get your simulation data into Ascent
- How to tell Ascent what pictures to render and what analysis to execute
Ascent tutorial examples are outlined in our documentation and included ready to run in Ascent installs

http://ascent-dav.org
Ascent tutorial examples are outlined in our documentation and included ready to run in Ascent installs

- [http://ascent-dav.org](http://ascent-dav.org)

- Click on “Tutorial”
Ascent’s interface provides five top-level functions

- **open() / close()**
  - Initialize and finalize an Ascent instance

- **publish()**
  - Pass your simulation data to Ascent

- **execute()**
  - Tell Ascent what to do

- **info()**
  - Ask for details about Ascent’s last operation

```c
// Run Ascent
Ascent ascent;
ascent.open();
ascent.publish(data);
ascent.execute(actions);
ascent.info(details);
ascent.close();
```

The `publish()`, `execute()`, and `info()` methods take a Conduit tree as an argument. What is a Conduit tree?
Conduit provides intuitive APIs for in-memory data description and exchange

- **Provides an intuitive API for in-memory data description**
  - Enables *human-friendly* hierarchical data organization
  - Can describe in-memory arrays without copying
  - Provides C++, C, Python, and Fortran APIs

- **Provides common conventions for exchanging complex data**
  - Shared conventions for passing complex data (e.g. *Simulation Meshes*) enable modular interfaces across software libraries and simulation applications

- **Provides easy to use I/O interfaces for moving and storing data**
  - Enables use cases like binary checkpoint restart
  - Supports moving complex data with MPI (serialization)

Website and GitHub Repo

http://software.llnl.gov/conduit
http://github.com/llnl/conduit
Conduit underpins Ascent’s support for C++, C, Python, and Fortran interfaces

Conduit also enables using YAML to specify Ascent actions

Conduit’s zero-copy features help couple existing simulation data structures

Conduit Blueprint provides a standard for how to present simulation meshes

Learning Ascent equates to learning how to construct and pass Conduit trees that encode your data and your expectations.
Ascent’s interface provides five composable building blocks

- **Scenes** (Render Pictures)
- **Pipelines** (Transform Data)
- **Extracts** (Capture Data)
- **Queries** (Ask Questions)
- **Triggers** (Adapt Actions)

The tutorial provides examples for all of these.
For the reminder of the tutorial, we will run the Ascent Tutorial examples using Jupyter Notebooks.
You can run our tutorial examples using cloud hosted Jupyter Lab servers

Start here:

https://www.ascent-dav.org/tutorial/
Thanks!

Ascent Resources:

- Github: https://github.com/alpine-dav/ascent
- Docs: http://ascent-dav.org/
- Tutorial Landing Page: https://www.ascent-dav.org/tutorial/

Contact Info:

Cyrus Harrison: cyrush@llnl.gov
Nicole Marsaglia: marsaglia1@llnl.gov
This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy’s Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation’s exascale computing imperative.