Exploring Cinema with the Cinema Virtual Machine

David H. Rogers*, James Ahrens*, John Patchett* and David DeMarle**

*Los Alamos National Laboratory
**Kitware, Inc.

May 27, 2015
CVM v1.1

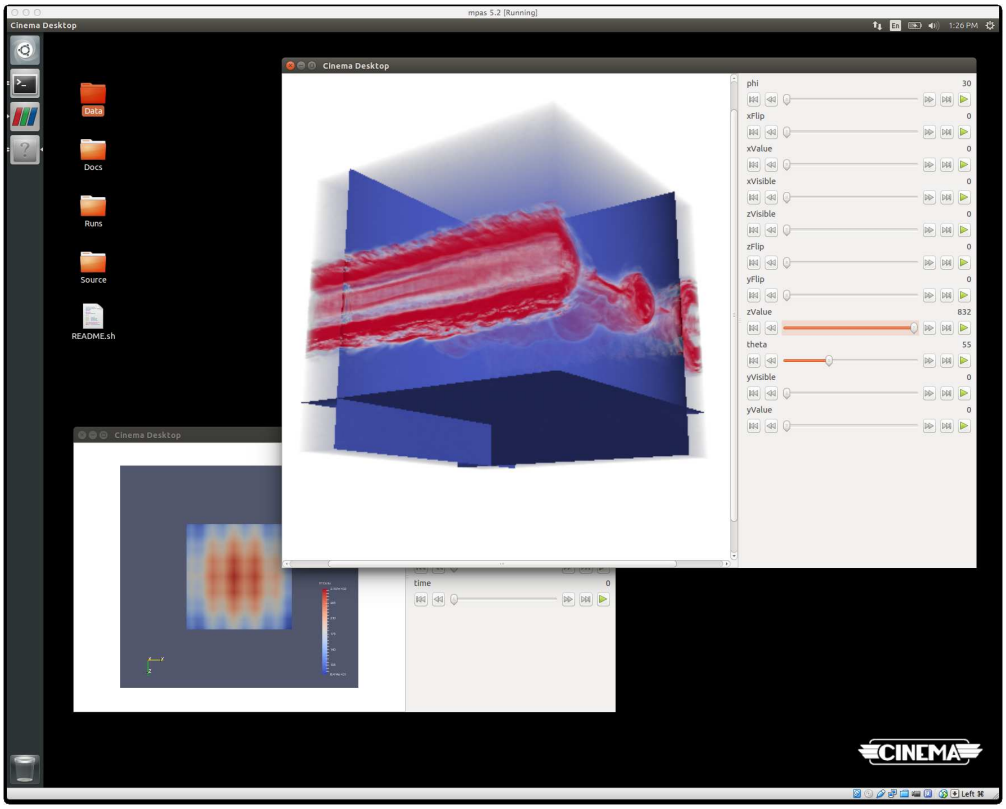

Figure 1: Screenshot of the MPAS Virtual Machine, showing two Cinema databases being explored with a desktop viewer. Cinema is workflow for interacting with large scale data.

1 Introduction

Extreme scale scientific simulations are pushing the limits of scientific computation, and are stressing the limits of the data that we can store, explore, and understand. Options for extreme scale data analysis are often presented as a stark contrast: save massive data files to disk for interactive, exploratory visualization, or perform *in situ* analysis to save detailed data about phenomena a scientist knows about in advance. We propose that there is an alternative approach

—a highly interactive, image-based approach that promotes exploration of simulation results, and is easily accessed through extensions to widely used open source tools. This new approach supports interactve exploration of a wide range of results, while still significantly reducing data movement and storage.

We call this new approach **Cinema**. This tutorial assumes that you are somewhat familiar with the technical motivations that drove the creation of Cinema, but further technical detail can be found in [1].

We have created the Cinema Virtual Machine (CVM) to provide a hands-on way of understanding some of the concepts behind Cinema, along with the deeper technical ideas. With the CVM, you can explore several small Cinema databases, explore concrete examples of the Cinema file specification, and get hands-on experience creating *in situ* export scripts with an MPAS-coupled analysis workflow.

1.1 Who Is This Document For?

For brevity, this document assumes you know a bit about Cinema, and would like to dig into some examples. If you've heard a talk about Cinema, or have read a technical paper about it, this CVM is a good place to learn more. If you're just starting to learn about Cinema, the project website below or the mailing list are a good way to get started.

This document is a companion guide to the Cinema Virtual Machine, but if you don't have that, you can find all Cinema-related information on the Cinema home page:

`http://datascience.lanl.gov/Cinema.html`

You can also send questions to the Cinema information email:

`cinema-info@lanl.gov`

2 What is Cinema?

Cinema represents a new data analytics workflow for massive data that differs significantly from a traditional post-processing workflow. In summary, the workflow focuses on automated data reduction on data as it is generated (such as, in situ from a simulation or from an experimental stream) into a flexible data representation that provides custom interactive exploration options.

Cinema is an image-based approach to exploring large scale data. Designed as a added capability to extreme-scale data analysis workflows, Cinema can be used to create compact, explorable data products from extreme scale simulations, in cases where saving and visualizing the entire database would be impracticale or impossible due to storage or networking requirments.

Thus, a cinema workflow is a way of creating images and other data that sample the analysis and visualization space that one would normally explore through interactive tools, if the entire simulation results were available. In the extreme data sizes that we normally encounter in high performance computing (HPC), we to not have the storage or the bandwidth available to save that entire set of results. For further information about the technical and analytic drivers behind Cinema, please see [1].

2.1 Cinema Is Tool Agnostic

Though this virtual machine includes ParaView-based tools that implement the Cinema workflow, any tool can be instrumented to create Cinema databases. Using the information in the Cinema file specificaion [2]. The CVM includes data created by direct coupling with another simulation. An example of this is included in the `CinemaStores/volume_render` directory.

This cinema database was generated by Greg Abram at The University of Texas, at Austin, using a prototype in-situ visualization tool based in the Intel Ospray raytracing framework. To minimize impact on the simulation a custom ray tracer was developed that provides ray cast volume rendering of simulation data directly from simulation data without generating intermediate data. It incorporates parametric clipping planes that may be either visible (as opaque surfaces) or transparent. The data for this simulation is used courtesy of Bill Daugthon (LANL) and Berk Geveci (Kitware, Inc.) [3]

3 Getting Started

Taking a hands-on approach, we are providing you with a sandbox of working examples that show the different parts of cinema. Again, this is only a primer on Cinema, but is intended to show how the various parts work together.

We note that a virtual machine has performance limitations, and that the examples included in the CVM are, as a result, small. The three prototype workflows included in this VM give a feel for how the analytics, workflow and interactive analysis work.

This document provides instructions on how to use the Cinema Virtual Machine (CVM) v1.1, to explore and understand Cinema. The VM has everything you need in order to run a simulation code, coupled with *in situ* analysis capabilites that create a Cineam database. The database can then be explored through the simple viewers that are included in the VM. It includes the following:

Coupled MPAS/Cinema simulation code MPAS Ocean is a multiscale simuation of the ocean, intended for the study of anthropogenic climate changes, as the ocean component of climate system models. [4] This code is coupled with Cinema *in situ* components so that data products can be exported during the execution of the simulation.

Cinema viewers These are simple applications that can be used to view a valid cinema database. There are both browser-based and python-based implementations of viewers.

Cinema documentation This includes a Cinema file specification and other documents that give more detail on Cinema.

Cinema source code There are several examples of Cinema python scripts, viewers, and other cinema code that you can explore.

CVM is designed to let you dig right in. We recommend using VirtualBox `http://www.virtualbox.org` to run the CVM, but other solutions should work as well.

There are three workflows to explore, but first let's take a look at the overall organization of the CVM.

```
~/Desktop
    /Data           contains several example Cinema databases
    /Docs           contains all CVM-specific documents
    /Runs           MPAS run directories, for in-situ runs
    /Source         source files for you to explore
    README.sh       a 'getting started' link. Click and start
```

3.1 Activity I: Explore an Existing Cinema Database

To get started, it will be useful to see some small existing Cinema databases. To do this, open a shell from the Ubuntu Launcher. The type one of the following commands in the shell. This will launch a simple viewer on a specific dataset. See Figure 2

```
cview ~/Desktop/Data/CinemaStores/two_views/image_0/info.json
cview ~/Desktop/Data/CinemaStores/two_views/image_1/info.json
cview ~/Desktop/Data/CinemaStores/magnitude-volume-halos-time/info.json
cview ~/Desktop/Data/CinemaStores/volume-render/info.json
```

A cinema database is a set of images that can be explored in many different ways. The `cview` application shows one of those ways—an interactive visualization app. This application lets you explore the various dimensions of the images. One simple interaction is rotating the images, as if you are rotating around the object. This is done through mouse interaction. Other variables, such as cut planes, opaticy and time, can be explored through interacting with the sliders on the right hand side of the application. Again, this is a simple interaction with the database, but it affords a well-understood interaction with the database. The interaction with this data can be made as smooth as you'd like—it's simply a matter of increasing the number of samples in the dimension. It's a tradeoff between interaction and compute/render time.

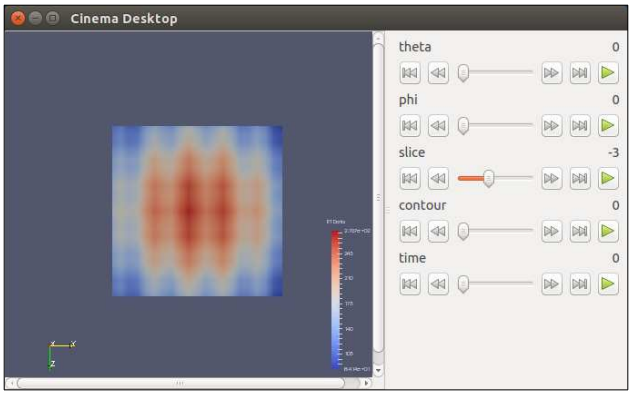

Figure 2: Screenshot of the desktop viewer, showing a sample dataset included with this Virtual Machine. Note the variables on the right, which can be controlled using the ui elements, and in the case of theta and phi angles, through interactive dragging in the view window.

3.2 Activity II: Create a New Cinema Database, and Explore It

There is an MPAS run directory that will run a low resolution MPAS Ocean simulation coupled with *in situ* analysis components that will write out a Cinema datbase.

1. Run the MPAS/Cinema coupled simuation by typing these commands in a shell. Note that while the *in situ* code is running, you will see a window on the screen that renders all the views that will be captured in the Cinema database.

   ```
   cd ~/Desktop/Runs/Run2
   ./runmpas.sh
   ```

2. After the simulation has completed, explore the Cinema database:

   ```
   cview cinema/image/info.json
   ```

3.3 Activity III: Create an Export Script, Create a New Cinema Database, and Explore It

The CVM includes tools that help you do this, even for a complex simulation such as MPAS. In the previous figure, we described how an *in situ* analysis pipeline can be constructed to execute analysis code during the simulation. There is a flexible architecture for this using ParaView Catalyst components to write out Cinema databases. The diagram below shows how this works. PV-¿script-¿cinema database. We can create a wide range of visualizations with ParaView, and MPAS will execute a ParaView pipeline *in situ*, creating elements that populate a Cinema database. So, our next step is to create a python script that MPAS will use to write out new data.

The following instructions can be used to export any visualization created in ParaVew with this data, so these serve as a guide to the process that you can use on many different *in situ* visualizations. To export a script:

1. Create geometry that you will use to design a visualization. This is rather large, so we didn't include it in the CVM. Runnin MPAS creates this *in situ* with a simple script:

   ```
   cd ~/Desktop/Runs/Run1
   ./runmpas.sh
   ```

2. Open `paraview` by clicking on its icon in the Ubuntu Launcher.

3. From the top menu, select File ->Recent Files ->(pick the first file on the list). ParaView will open, and you will see the `vtu` file in the Pipeline Browser. Click the green **Apply** button at the lower left hand side of the UI, and you will see the MPAS simulation geometry rendered in grey in the RenderView.

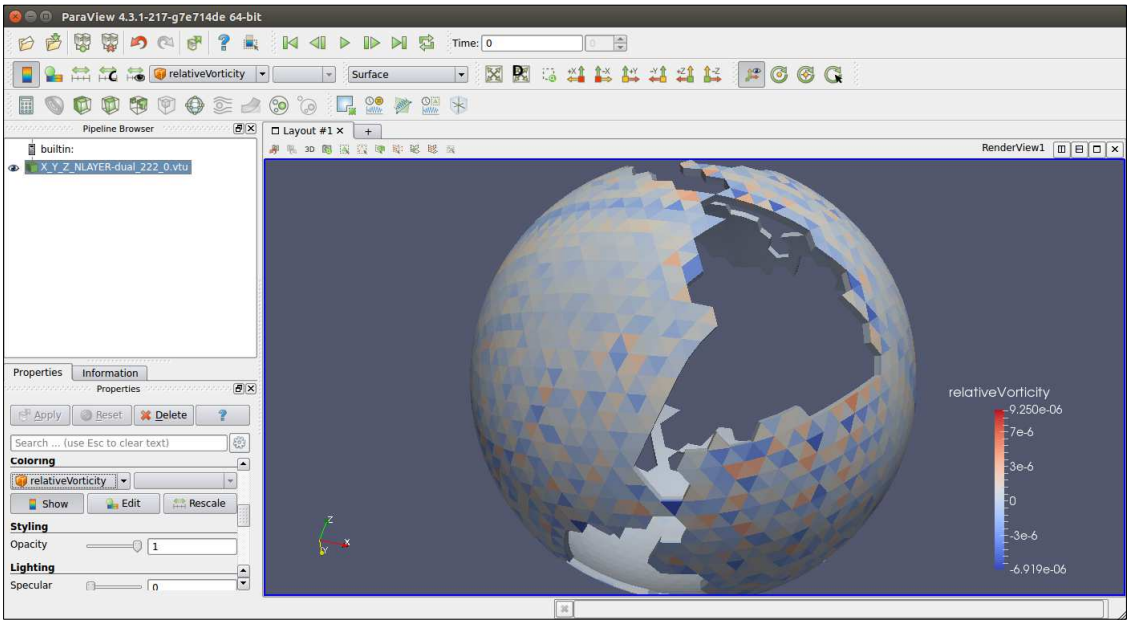

Figure 3: Screenshot of ParaView, showing the sample geometric output from the MPAS simulation, colored by a specific variable. Note that since this is the MPAS Ocean simulation, the area represented by geometry is the ocean, and the land masses are devoid of geometry. In this view, we see South America as a void. The simulation grid is quite coarse, simply because we will run the simulation within the CVM, so we want to constrain compute and space requirements.

4. Now you can set up a visualization that you'd like to save in a Cinema database. For this simple example, you can simply choose a variable to render, by picking a variable from the toolbar at the top of the application. A good one to choose is `relativeVorticity`, but you can choose any of the ones from the pull down menu (see Figure 3).

5. Save out an export script, using the export script wizard.

   (a) Click on `CoProcessing->ExportState` from the top menu. This will bring up the **Export Co-Processing State** dialogue.

   (b) Click **Next** in the first dialogue box, which will take you to the **Select Simulation Inputs** dialogue box.

   (c) Click on the top item in the list on the left of the dialogue (it will be a `vtu` file), then click the **Add** button near the bottom of the dialogue. The item will transfer to the right list.

   (d) Click **Next**. The **Name Simulation Inputs** dialogue will appear.

   (e) Double Click on the item in the **Simulation Name** column, and edit the text to be `X_Y_Z_NLAYER-dual`. NOTE: this is a specific edit because of the contents of the MPAS data structures, and is not necessary for exporting a script for most other simulations.

   (f) Click **Next**. The **Configuration** dialogue will appear.

   (g) Click on the **Output to Cinema** checkbox, then in the **Cinema Export** pulldown menu at the bottom of the first set of controls, select **Spherical**. This means that you will be creating a Spherical Cinema Camera with the script. See Figure 4

   (h) Click **Finish**, and save the script in `~/Desktop/Runs/Run2`, as `myscript.py` or another name you'd like.

   (i) Quit `paraview`.

6. Open a shell by clicking on the icon in the appropriate icon in the Ubuntu Launcher.

7. `cd ~/Desktop/Runs/Run2`

8. MPAS is architected to look for a script named `mpas.py` while running, and execute the contents of that script as *in situ* analysis. We therefore link our new script to mpas.py:

```
rm -f mpas.py
ln -s myscript.py mpas.py
```

9. Now, run MPAS. This will create a cinema database.

```
./runmpas.sh
```

10. After the run finshes, there will be a cinema database in `cinema` directory. You can view it by typing:

```
cview cinema/image/info.json
```

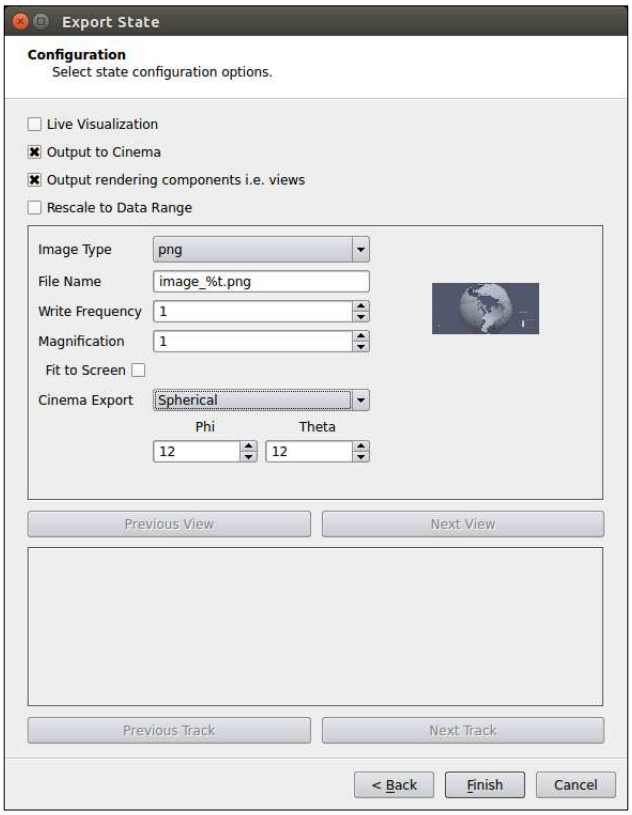

Figure 4: Screen shot of the **Export State Configuration** dialogue box, showing the a Cinema export with a spherical camera. The `Phi` and `Theta` options show how dense the camera sphere will be sampled during *in situ* rendering.

4 How Do I Couple Cinema Code to My Simulation?

First, as was noted above, one simple way to create Cinema databases from your code is to implement native methods for creating data products that follow the Cinema specification [2]. If you already are familiar with the architecture of your code, this could be the best way forward.

If you'd like to take advantage of the components that have already been authored for ParaView and VTK, you start by creating an *in situ* adapter that couples the data structures in your code to data structures in VTK. This makes it

possible for the entire set of VTK filters to execute on the simulation data without writing it to disk. There is extensive documentation on how to develop components to couple your code with Catalyst and Cinema, found in *The ParaView Catalyst User's Guide* [5].

If you're interested in this level of integration of Cinema, please contact us at `cinema-info@lanl.gov`. The Cinema components for `paraview` are in early development, but we are looking for partners to help develop and test these components.

5 What's Next?

The CVM is a quick way to get familiar with the concepts, workflows, and tecnical directions of Cinema. If you'd like more information, visit the Cinema home page to check out the latest R&D, releases and documents. You can also query the cinema mailing list, and we'll try to answer any questions you may have.

`http://datascience.lanl.gov/Cinema.html`
`cinema-info@lanl.gov`

6 Acknowledgements

This work was funded by Dr. Lucy Nowell, Program Manager for the Advanced Scientific Computing Research (ASCR) program office in the Department of Energy's (DOE) Office of Science. The authors would like to thank Dr. Mark Petersen for his enthusiastic support of *in situ* analysis for MPAS, and Dr. Andy Bauer of Kitware, Inc. for his help in developing the MPAS *in situ* capability.

References

[1] James Ahrens, Sébastien Jourdain, Patrick O'Leary, John Patchett, David H. Rogers, and Mark Petersen. An image-based approach to extreme scale in situ visualization and analysis. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 424–434, Piscataway, NJ, USA, 2014. IEEE Press.

[2] David Rogers, James Ahrens, and John Patchett. Cinema simple database specification. Technical Report LA-UR-15-20572, Los Alamos National Laboratory, January 2015.

[3] Fan Guo, Hui Li, William Daughton, and Yi-Hsin Liu. Formation of hard power laws in the energetic particle spectra resulting from relativistic magnetic reconnection. *Phys. Rev. Lett.*, 113:155005, Oct 2014.

[4] Todd Ringler, Mark Petersen, Robert L. Higdon, Doug Jacobsen, Philip W. Jones, and Mathew Maltrud. A multi-resolution approach to global ocean modeling. *Ocean Modelling*, 69(0):211 – 232, 2013.

[5] Andrew C. Bauer, Berk Geveci, and Will Schroeder. The paraview catalyst user's guide. Technical report, Kitware, Inc., 2013.