NEIS Final Report – 6/30/2013

Understanding Tornadoes and Their Parent Supercells Through Ultra-High Resolution Simulation/Analysis

Robert Wilhelmson, Mark Straka, and Roberto Sisneros
National Center for Supercomputing Applications

Leigh Orf
Central Michigan University

Brian Jewett
Department of Atmospheric Sciences, University of Illinois

George Bryan
National Center for Atmospheric Research

Abstract

CM1 is a parallel three-dimensional, non-hydrostatic, non-linear, time-dependent numerical model designed for idealized studies of atmospheric phenomena. It is being used to simulate storms and tornadoes and is the model used in this proposal that focuses primarily on high performance I/O together with inline- and post- analysis/visualization. VisIt was selected for visualization of three-dimensional spatial data on Blue Waters and Matlab for two-dimensional data. Both of these visualization tools have parallel capabilities. Originally the data to be visualized was going to be moved from the memory of a running CM1 simulation to memory on the GPU side of Blue Waters for in-line visualization using Damaris and VisIt. Some work with the Damaris group demonstrated this capability for CM1 but it was concluded that future availability and maintenance of Damaris on Blue Waters and other systems was unclear. Further, VisIt performance did not significantly benefit from access to the GPUs. Therefore, near real-time visualization became the emphasis with data from a running simulation first written to Blue Waters disk using HDF5 (3D) and pHDF5 (2D). The arrival of new data on disk is monitored and when ready it is read back into Blue Waters for visualization of the 3D data on CPU nodes and sent to another system for generation of 2D visualization and web-based display of both 2 and 3D visualizations.

A VisIt plugin was developed for the near real-time and post analysis/visualization of simulation data. Up-to-date visualizations are made available on web pages by adapting software built for post-analysis of WRF simulation data but that now will update as the simulation proceeds. In addition, work continued with Paul Woodward's team using a simplified version of CM1 for studying the impact of restructuring advection and turbulence calculations to improve computational intensity (number of computations per memory fetch). This paper discusses current accomplishments and progress.

1) INTRODUCTION

Wilhelmson and collaborators are tackling fundamental questions related to the formation and maintenance of tornadoes. Example questions include: 1) What occurs within supercell thunderstorms that leads to tornado formation (tornadogenesis)? 2) What balance of forces exists during a long-lived tornado in a supercell? and 3) Can models be used to better predict when such tornadoes will occur versus when we get a weak tornado or no tornado at all?

The scientific approach for addressing these questions involves use of an idealized cloud model (CM1[1]) designed specifically for parallel architectures using two-dimensional spatial decomposition. It is spatially three-dimensional, non-hydrostatic, non-linear, and time-dependent. Simulated storms in CM1 are initialized in environments known to be conducive to creating supercells that can produce tornadoes. The goal is to simulate tornadogenesis and tornado structure including the balance of forces that exists within long track tornadoes.

The scientific challenge is that only 20-25% of supercells produce tornadoes, and only 5-10% of supercell tornadoes are strong, long-track ones. The computational challenge comes from the fact that a very small time step is required to maintain computational stability if ultra-high resolution is used. Each time step involves many computation and communication cycles, and some global communication when adaptive time stepping is activated.

High temporal frequency of the model state needs to be saved in order to properly capture features of interest (winds are moving fast during tornado genesis and maintenance). Hence, a tremendous amount of I/O (on the order of 1 petabyte) is possible when saving the full model state at high temporal frequency.

VisIt software[2] is being used for visualization of three dimensional model data at a specific time. It was developed at Lawrence Livermore, and supported on Blue Waters. It contains a robust plugin development environment and is designed for massively parallel architectures. The Blue-Waters specific code creation/modification in CM1 involved I/O and visualization; i.e. the addition of a HDF5 output option using a combination of serial (for "3D" files) and parallel (for "2D" files) HDF5 routines.

For serial (3D) files, the scheme assigns one I/O core per node, with intranode communication done to collect data to the I/O core (this is fast, and does not use the network). It does require rank reordering to ensure that the virtual topology is mapped to the hardware topology. 3D model floating point arrays for 50-100 times are buffered in memory before a file per node is written to disk using an HDF5 "core" driver.

A plugin for VisIt was created to access data from these files. The API/low-level file management code written in C "hides" the complexity of the multiple file/multiple time structure. In order to reduce disk operations, a one-time utility is run to create a small HDF5 file that contains data about the CM1 3D file to be used VisIt. Further, only a

[1] http://www.mmm.ucar.edu/people/bryan/cm1/
[2] https://wci.llnl.gov/codes/visit/

subdomain of the model can be requested, hiding full domain data from VisIt. Due to the I/O abstraction, the plugin is not restricted to the size and number of VisIt domains. The plugin can therefore set domain dimensions that work best for a given number of ranks, hardware, etc. The API is used for all 3D access (not just VisIt).

This paper discusses the implementation of HDF5 I/O for CM1, visualization of three dimensional data generated by CM1, web serving of visualization data in near real-time, CM1 performance tests, collaboration with Paul Woodward's group., and a summary of a recent paper on Damaris which used CM1 as a test model.

2) IMPLEMENTATION OF HDF5 I/O FOR CM1

The CM1 model came with three output format options, all of which either result in one file per MPI process per output time or one file per output time. Both of these options present problems for large simulations; one file per process will result in millions of small files, making post-analysis unwieldy. This approach is inefficient, suffering from latency, metadata server overhead, and general overhead associated with concurrently writing such a large number of files to disk on a shared resource. However, writing one file per time becomes prohibitively inefficient for a large number of MPI processes. Blue Waters exhibits the best aggregate I/O throughput when many (but not too many) large files are written concurrently.

These issues motivated the development of new I/O code by Orf for writing out model data for near real-time and later use. An approach was chosen that meets the following objectives: (a) reduces the number of files written to a reasonable number, (b) minimizes the number of times actual I/O to the filesystem is performed, (c) results in big files (each > 1 GB) and (d) makes it easy to do post-processing/analysis and visualization without needing to convert to another format. Objectives (a)-(c) are met by writing only one file per node (a typical simulation has 16 MPI ranks per node) and by buffering data from multiple times in memory before writing an HDF5 file to disk. For planned simulations this reduces the number of files written by a factor of ~800, reduces the number of times I/O is done by a factor of 50, and results in files that are between 1 and 2 Gbytes per node per write. Objective (d) is met by creating an I/O driver that sits on top of HDF5 and that allows any subset of the full model domain to be read into a contiguous buffer. This driver can be called from analysis and visualization software using as input the coordinates of the subdomain and the time of interest. Any 1, 2, or 3D spatial range may be requested for any field. This approach shields the user from having to "stitch together" data that spans multiple files spread out over multiple directories and is coupled with parallel VisIt capability through the VisIt plugin to be discussed.

HDF5 is used as the file format for all output. This format was chosen due to its extreme flexibility and its built-in compression and file buffering capability. Two different approaches are used for I/O: so-called 3D data (what has been described thus far) and 2D data. 3D data is spread out over many files, each file containing 50-100 time levels (typically saved at even intervals, such as every 5 seconds). The multiple time levels are written to a single file using the HDF5 core driver that allows writing to a file in memory. HDF5 groups are created for each time, and within those groups are contained the compressed 3D arrays that span each node. Writing to memory is extremely fast as no actual I/O is done. This approach works very well for our application because CM1 only utilizes about 3% of total available memory per node. Furthermore, gzip data compression is done on data in order to reduce the amount of 3D data written to disk by a factor of about 2. Trial and error has found that writing 50-100 time levels to memory results in files that are on the order of 1-2 GB in size. When it is time to do I/O, all files are written concurrently to disk and memory is freed and the cycle is repeated.

Each 3D file contains metadata that describes the full model I/O geometry (e.g., how many files, the MPI decomposition, the size of the full model domain) as well as the geometry of the individual file. This redundancy allows one to query any one file to read in the necessary parameters that describe how data is spread out on disk. Each file is named using strict conventions, and hence the filename itself contains metadata (for example, each file name contains a node number that represents a specific location in space, as well as a number representing the model time for the first time level in the file). Files are spread out into a few separate directories rather than all being written to a single directory (this has been found to improve performance slightly).

In addition to writing out full model field data at selected times, pHDF5 is used to write unbuffered horizontal slice data to a single file for each selected time. Slices of this two-dimensional floating point data corresponds to selected atmospheric levels (i.e., "weather maps"). This is done for practical purposes, as it is straightforward to make plots of this data from a single file (as opposed to assembling data from hundreds to thousands of 3D files) either while the model is running to monitor its progress or after simulation completion. However, 2D data written using pHDF5 (utilizing all-ranks-to-one-file parallel HDF5 calls) exhibits poor I/O throughput and thus this data is currently written only at infrequent time intervals. Alternative approaches to improve performance will be investigated in the future such that simulation results can be monitored on a frequent basis as a simulation proceeds.

3) VISUALIZATION OF THREE DIMENSIONAL DATA GENERATED BY CM1

There are few visualization and analysis software packages that are available for parallel visualization on large scale computing systems. Furthermore, for our purposes, we require such a package to be tailored to scientific data, be scalable, and offer an adequately diverse suite of data analysis tools and plotting functionality. These requirements practically limit possibilities to two packages, both of which are supported by Blue Waters staff: VisIt and ParaView. Both implement data analysis pipelines that operate on VTK data structures, achieve scalability through data-parallel execution of the pipeline, offer mature support for various data types, and have similar sets of functionalities. Therefore, the choice between these is simply one of personal taste; our choice was VisIt.

The library responsible for the significant increase in CM1's I/O throughput accomplishes the task through I/O aggregation. That is, multiple time steps of the full spatial resolution are combined and written into a single file. Through data readers that open and read in data from disk to fill VTK data structures that VisIt can then operate on, there is already support for many types of data. This includes HDF5. However, a typical reader equates a single file with a single data block. This provides a high level of support for typical file-per-process simulation output, but would eliminate the possibility of parallel analysis in our case.

In response to these issues, additional code was developed to analyze CM1 output data by Sisneros and Orf. First, a routine was added to the existing C library for reading the aggregate files. This routine reads metadata for these files and creates a new header file (a .cm1visit file). This file contains all the needed information for creating necessary VTK data structures: grid size and spacing, variable names, number of time steps, etc. Second, a data reader VisIt plugin was created that is associated with this header file. Using this plugin VisIt has each of its current processors request a block of data. The reader, based on the number of processors allocated in a VisIt session, automatically decomposes the data to feed these processes.

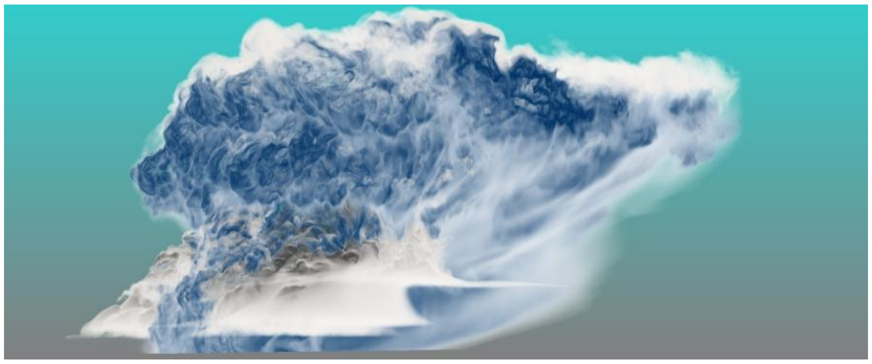
In addition to this decomposition, the data reader also performs the following:

1) Correct "ghost zones" are created for each data block. This is an extra set of cells around all three dimensions of a block that visualization approaches either 1) need to create correct plots or 2) can use to optimize some calculations and allow calculations of horizontal derivatives without creating gaps.

2) Data is decomposed based on the original grid size rather than CM1's original decomposition. Consider an actual run of CM1 made with 3840x3840x400 grid points in x, y, and z using 6400 nodes on Blue Waters. CM1 decomposes data in only the two horizontal dimensions, creating long, thin "pencils" of data with 48x48x400 grid points per node. The VisIt reader creates blocks better suited for visualization algorithms. For some "middle" block, there are exactly 4 other blocks with a shared side. Our reader, if VisIt was run with 6400 processors, would decompose the data into 96x96x100 blocks. A "middle" block now has 6 neighbors. This means whatever communication speedup is gained from ghost zones is magnified by 1.5x.

3) Data is decomposed into "nice" blocks. For instance, running VisIt with 32 or 33 cores will both likely result in splitting a dataset into 32 blocks.

4) Subsets of data can be requested through the header file. If it is known a priori only a subset of the dataset contains interesting features, only this subset will be read in. Then, this subset will automatically be decomposed into the right number of blocks. This has the potential to drastically reduce the resources required for complex pipelines/plots.



A snapshot in time of a supercell thunderstorm simulation done using the CM1 model and rendered with VisIt. The image shows cloud and hydrometeors (rain, snow, hail).

4) WEB SERVING OF VISUALIZATIONS IN NEAR REAL-TIME

The work by Jewett on near real-time visualization of CM1 data being generated during a running simulation on Blue Waters is focused initially on providing 2D (two dimensional) animations of selected fields at selected heights. Near real-time 3D (three dimensional) data will follow.

The near real-time transmission/visualization of tornado simulation data is to be (1) used to monitor the run so it may be halted if the solution appears to be in error, and (2) for ease of examination of selected fields later. Typically 2D horizontal slice data will be saved at higher temporal frequency than the larger 3D (three dimensional) data so rapid tornado formation may be captured and since 2D data sets are comparatively small. During the CM1 model integration, 2D slice data from a single time and selected model levels (X-Y planes and vertical planes near key phenomena) are routinely written to a single file on the Blue Waters disk using the all-to-one capabilities in parallel HDF5 as described in Section 2.

The 2D slice files will include horizontal slices of temperature, pressure, horizontal wind, subgrid turbulent kinetic energy, and water vapor. In addition they will include mass (mixing ratio) and concentration (number per unit volume) of precipitation hydrometeors (rain, hail) at the ground, just above the ground, and near the levels (altitudes) of greatest downdraft (~1-2 km above ground) and largest updraft (~6-9 km above ground).

The temporal interval at which data is saved to disk is chosen based on I/O cost, short-term storage and long-term archival considerations, the latter particularly relevant to 3D fields. For tornado-scale phenomena a full storm/tornado simulation for several hours will take 12-24 hours of computer time on 6400 nodes using a time step on the order of 0.1 seconds. 2D data will be saved every 5 to 10 seconds. Ultimately, 1440 to 720 2D slice data files will be written per simulation. For the example simulation discussed in the last section, a single 2D data file was 2.2 GB.

When the CM1 numerical model is known to have completely written a HDF5 file to disk, the file is transferred over University of Illinois networks (max: 10 gigabits/sec) to a quadcore server in the Atmospheric Sciences Department at the University of Illinois. This machine has RAID6 storage and gigabit ethernet access.

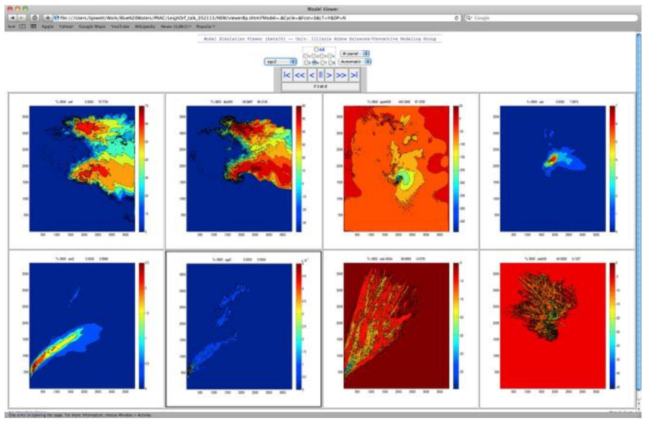
For initial estimates of data transfer speed, RSA-key passwordless transfers were timed from Blue Waters to the server. Data rates reached 48 MB/sec. However, how data transfer is performed is important, particularly for the amount of data under consideration here (e.g. 3600 files @ 2.2 GB = 8 TB). Use of the Blue Waters login nodes for data transfer with *scp, sftp, rsync, globus-url-copy* is strongly discouraged due to the impact on other users. In addition, use of similar tools to transfer large data sets from batch nodes during the simulation are discouraged to avoid causing problems on the Blue Waters torus network. Instead, use of Globus Online is being implemented using the command-line interface to direct transfers that initiate from Blue Waters I/O "mover nodes" (28 dedicated Import/Export servers, with potential throughput of 100 GB/sec). This system offloads the data movement load from login or computational nodes while also dynamically selecting the most lightly loaded I/O servers in the Blue Waters system. A similar facility is available to migrate data to the HPSS subsystem and from there to tape for long-term storage.

The iMac server will also require confirmation when the incoming file write is complete so it may begin post-processing of model data. This is accomplished either by processing one file when the next transfer has begun or by waiting for a short confirmation file to be written following each data file. Data rate tests indicate that it will take 1-2 minutes to transfer 2.2. Gbytes. The elapsed data transfer period effectively sets the limit for the time available for ANY image processing, discussed below, to take place on the server machine.

Several options are available to reduce the time-to-display (from Blue Waters simulation time until images appear on the web server) and the server processing load. These include (a) moving more data processing to Blue Waters compute nodes; (b) sending only a subset of files to the server; (c) processing only a subset of files on the server; (d) optimizing the data transfer, and (e) parallelizing the data processing on the server.

The server machine will serve three purposes: (1) medium-term (1-2 year) storage of selected tornado simulation data, (2) first-cut analysis and image generation from the incoming Blue Waters slice data, and (3) web serving of images, enabling near-real-time display of tornado simulation results. For objective (1), 5 TB are available and more disk is being purchased. For (2), scripts are being developed to run on the server and identify when new data files are ready. Matlab is being used to produce a set of raster images, named with the model data time. Additional 3D visualizations will be carried out using the VisIt software running on Blue Waters. Images from these 3D visualizations would also be available through the web server.

The completed images are made available through the Apache open source software on the server machine. The images are viewed through a Javascript-based web page that allows animation and comparison of different fields or levels through a multi-panel display. The Javascript is being extended to automatically update the display as new data time slices become available. A demonstration of the type of multipanel display is shown here using Blue Waters simulation data:



5) PERFORMANCE TESTS WITH CM1

Three storm simulations on Blue Waters were made by Orf using a grid of 3820x3840x400 in a domain that is 120x120x20 km. The simulations were made using 6400 nodes and ran for two hours of cloud simulation time using 20 m horizontal resolution in the inner 75x75 km region and grid stretching outside this region. Restarts files were written every 300 seconds of model time.

Overall model performance is expressed as a ratio between wallclock time to model time; Morrison microphysics with I/O every 60 model seconds and statistics every 30 seconds resulted in a 4:1 wallclock:model time ratio. Two hours of cloud time therefore took eight wallclock hours in this case. Increasing the write frequency and statistics to every 10 model seconds resulted in a wallclock:model time ratio of 5:1. In both cases, an entire two-hour simulation would fit in a 12 hour time block on blue Waters. A third simulation was run with I/O every 3 seconds and statistics every 5 model seconds, but utilizing the Gilmore/Straka LFO microphysics. This case resulted in an 8:1 wallclock/model ratio, which would require 16 wallclock hours for a 2 hour cloud simulation. Timing statistics revealed that the Gilmore/Straka LFO microphysics took significantly more wallclock time than the Morrison microphysics despite the fact that it is a less sophisticated routine. The reasons for this are being investigated.

All three simulations produced a supercell as expected given the initial atmospheric sounding used. However, only brief and weak tornadoes developed. Further model simulations are being planned.

6) Extension of Woodward's Programming Techniques to CM1

Paul Woodward's group has achieved sustained performance of 1.5 petaflops on Blue Waters with his PPM code. This is due in part to the numerical technique employed and the way blocks of memory (briquettes) are manually mapped to cache, increasing computational intensity (a measure of the number of flops performed divided by the number of off-chip words either read or written). Work is underway to apply this approach to improve the performance of the dynamic calculations in CM1.

A detailed discussion of the status of this work appears in Woodward's NEIS-P2 final report[3]. Several highlights quoted from this report follow:

- Jayaraj has studied a direct application of his CFD Builder code transformation tool [17] to a directionally split variant of the advection scheme used in CM0. He finds that he can accelerate the execution of this algorithm in the same way and to essentially the same degree as the advection scheme in PPM is accelerated in our code.
- Woodward has studied the sound wave step in isolation also.
- Our most recent work with CM0 focuses on the turbulence calculation that forms the beginning of the grid cell update. This portion of the code, which consists of several subroutines and considerable computation, is the most challenging from the perspective of our approach described above. By pipelining all the operations of the turbulence calculation in CM0, which amount to 712 flops per grid cell, we achieve a computational intensity of 17 flops/word. This means that we perform 17 flops with on-chip data for every word of off-chip data that must be fetched or written. This is roughly half the intensity of PPM, but it is much higher than that of the untransformed CM0.

Orf worked directly with Woodward, removing numerous parts of the CM1 code not used in supercell/tornado simulations thus creating CM0 in addition to providing advice and consultation along with Wilhelmson and Bryan.

7) DAMARIS and CM1

CM1 data and VisIt have been used to demonstrate in part the capabilities of the asynchronous framework DAMARIS on Blue Waters. This framework was designed to allow concurrent computation and visualization through sharing of data through memory transfers. A test simulation was carried out in this framework and data interactively visualized as it became available to VisIt. A recent paper is available on the current state of this work that includes some scaling results for VisIt using data from a 20 m simulation on a the grid size noted in Section 5. Roberto Sisneros and Dave Semeraro from NCSA contributed to this work as well as Leigh Orf who provided data and advise on running CM1. The abstract of this paper appears below:

A Nonintrusive, Adaptable and User-Friendly In Situ Visualization Framework[4]

Matthieu Dorier ( ), Roberto R. Sisneros ( ), Tom Peterka ( ), Gabriel Antoniu ( ), Dave B. Semeraro ( )

[3] **Simulating Compressible, Turbulent Mixing in Spherical Geometry on the Blue Waters System at 1.5 Pflop/s with a Discussion of Techniques Used to Achieve High Performance**
Paul R. Woodward, Jagan Jayaraj, Pei-Hung Lin, and Michael Knox
Laboratory for Computational Science & Engineering
University of Minnesota

[4] http://hal.inria.fr/index.php?halsid=khe9eu7157lbe0rl8sua0gik46&view_this_doc=hal-00831265&version=1

INRIA RR-8314 (2013)

> **Abstract:** Reducing the amount of data stored by simulations will be of utmost importance for the next generation of large-scale computing. Accordingly, there is active research to shift analysis and visualization tasks to run in situ, i.e. closer to the simulation via the sharing of some resources. This is beneficial as it can avoid the necessity of storing large amounts of data for post-processing. In this paper, we focus on the specific case of in situ visualization where analysis codes are collocated with the simulation's code and run on the same resources. It is important for an in situ technique to require minimum modifications to existing codes, be adaptable and have a low impact on both run times and resource usage. We accomplish this through the Damaris/Viz framework, which provides in situ visualization support to the Damaris I/O middleware. The use of Damaris as a bridge to existing visualization packages allows us to (1) reduce code modifications to a minimum for existing simulations, (2) gather capabilities of several visualization tools to offer a unified data management interface, (3) use dedicated cores to hide the run time impact of in situ visualization and (4) efficiently use memory through a shared-memory-based communication model. Experiments are conducted on Blue Waters and Grid5000 to visualize the CM1 atmospheric simulation and the Nek5000 CFD solver.

Originally the data to be visualized was going to be moved from the memory of a running CM1 simulation to memory on the GPU side of Blue Waters for in-line visualization using Damaris. The work above did demonstrate the possibility of using Damaris with CM1 and VisIt in this way but it was concluded that future availability and maintenance of Damaris on Blue Waters and other systems was unclear. Further, VisIt performance did not significantly benefit from access to the GPUs. Therefore, near real-time visualization became the emphasis with data from a running simulation first written to Blue Waters disk using HDF5 (3D) and pHDF5 (2D).

8) CONCLUSONS

Significant progress has been made in improving CM1 for use on Blue Waters. As reported, it can now be successfully run and data visualized on Blue Waters. Work will continue over the summer to improve performance including that for parallel 2D I/O as well as visualization and associated web display.

9) ACKNOWLEDGEMENTS

Mark Straka (one of the authors of this report and a Blue Waters applications staff member) provided excellent support throughout this project for which we are very thankful. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. It is supported by the National Science Foundation (award number OCI 07-25070) and the State of Illinois. The reported work was supported by the Blue Waters Project and NSF Grant AGS-0843566.