Towards Petascale Simulation and Visualization of Devastating Tornadic Supercell Thunderstorms

Leigh Orf[1] Robert Wilhelmson[2,3] Roberto Sisneros[3]
Louis Wicker[4]

[1]Central Michigan University
[2]University of Illinois
[3]NCSA
[3]National Severe Storms Laboratory

May 12, 2014

Outline

Understanding the most devastating tornadoes

- The strongest, longest-lived tornadoes (long-track EF5) are the least common, but produce the most devastating damage and significant loss of life
- There is a lack of published research involving the simulation of such tornadoes embedded within their parent storm

Understanding the most devastating tornadoes

- The strongest, longest-lived tornadoes (long-track EF5) are the least common, but produce the most devastating damage and significant loss of life
- There is a lack of published research involving the simulation of such tornadoes embedded within their parent storm
- We aim to simulate supercells where the entire life cycle (genesis, maintenance, decay) of many long-track EF5 tornadoes occurs

Understanding the most devastating tornadoes

- The strongest, longest-lived tornadoes (long-track EF5) are the least common, but produce the most devastating damage and significant loss of life
- There is a lack of published research involving the simulation of such tornadoes embedded within their parent storm
- We aim to simulate supercells where the entire life cycle (genesis, maintenance, decay) of many long-track EF5 tornadoes occurs
- This is the first step in providing better forecasts of such events

Understanding the most devastating tornadoes

- The strongest, longest-lived tornadoes (long-track EF5) are the least common, but produce the most devastating damage and significant loss of life
- There is a lack of published research involving the simulation of such tornadoes embedded within their parent storm
- We aim to simulate supercells where the entire life cycle (genesis, maintenance, decay) of many long-track EF5 tornadoes occurs
- This is the first step in providing better forecasts of such events

The scientific approach

- CM1 Cloud model (written by George Bryan at NCAR) chosen as model due to its ability to scale well to massively parallel architectures
- Getting the "storm we want" to form proved difficult.

The scientific approach

- CM1 Cloud model (written by George Bryan at NCAR) chosen as model due to its ability to scale well to massively parallel architectures
- Getting the "storm we want" to form proved difficult.
  Experimented with different:
  - Atmospheric environments
  - Mesh geometries/resolution
  - Surface treatments
  - Microphysics parameterizations
  - Turbulence parameterizations
  - Cloud initializations approaches

The scientific approach

- CM1 Cloud model (written by George Bryan at NCAR) chosen as model due to its ability to scale well to massively parallel architectures
- Getting the "storm we want" to form proved difficult. Experimented with different:
  - Atmospheric environments
  - Mesh geometries/resolution
  - Surface treatments
  - Microphysics parameterizations
  - Turbulence parameterizations
  - Cloud initializations approaches
- As with the real atmosphere, most simulated supercells do not produce long-track EF5 tornadoes

The scientific approach

- CM1 Cloud model (written by George Bryan at NCAR) chosen as model due to its ability to scale well to massively parallel architectures
- Getting the "storm we want" to form proved difficult. Experimented with different:
  - Atmospheric environments
  - Mesh geometries/resolution
  - Surface treatments
  - Microphysics parameterizations
  - Turbulence parameterizations
  - Cloud initializations approaches
- As with the real atmosphere, most simulated supercells do not produce long-track EF5 tornadoes

The computational challenge

- CM1 "out of the box" did not contain an I/O option appropriate for the scale of our problem, nor a pathway to doing full-scale visualization
- Much time and effort was put into creating an I/O and visualization framework for a "Blue Waters sized" problem

The computational challenge

- CM1 "out of the box" did not contain an I/O option appropriate for the scale of our problem, nor a pathway to doing full-scale visualization
- Much time and effort was put into creating an I/O and visualization framework for a "Blue Waters sized" problem
- Wrote HDF5 output code for CM1 and a VisIt plugin to operate on the model output format

The computational challenge

- CM1 “out of the box” did not contain an I/O option appropriate for the scale of our problem, nor a pathway to doing full-scale visualization
- Much time and effort was put into creating an I/O and visualization framework for a “Blue Waters sized” problem
- Wrote HDF5 output code for CM1 and a VisIt plugin to operate on the model output format

HDF5 output and interface to model data

- 3D floating point arrays written on a per-node basis using serial HDF5 and buffering to memory before writing to disk
  - C API created to interface with native CM1 HDF5 output, allows easy conversion to other formats (e.g., netCDF)
  - VisIt plugin flexible enough to operate on full domain or any subset, and does its own domain decomposition unrelated to file geometry
- Domain-wide 2D floating point arrays of selected fields written utilizing pHDF5
  - 2D data is easy to visualize quickly and tells much of the storm's story without necessitating a full 3D visualization session

HDF5 output and interface to model data

- 3D floating point arrays written on a per-node basis using serial HDF5 and buffering to memory before writing to disk
  - C API created to interface with native CM1 HDF5 output, allows easy conversion to other formats (e.g., netCDF)
  - VisIt plugin flexible enough to operate on full domain or any subset, and does its own domain decomposition unrelated to file geometry
- Domain-wide 2D floating point arrays of selected fields written utilizing pHDF5
  - 2D data is easy to visualize quickly and tells much of the storm's story without necessitating a full 3D visualization session

Success. . . or not?

- End of PRAC with time running out, a simulation produces a long-track EF5 (using 625 nodes, 10,000 cores)
  - U of Illinois proposal written/granted to further this work
  - An issue with the environmental conditions in EF5 simulation identified, potentially nullifying results
  - Reproducing this run with fixed environment is successful

Success. . . or not?

- End of PRAC with time running out, a simulation produces a long-track EF5 (using 625 nodes, 10,000 cores)
  - U of Illinois proposal written/granted to further this work
  - An issue with the environmental conditions in EF5 simulation identified, potentially nullifying results
  - Reproducing this run with fixed environment is successful

The May 24, 2011 El Reno, OK, EF5 tornado

- On May 24, 2011, a supercell thunderstorm produced a long-track (65 mile long path) EF5 (peak winds estimated to exceed 210 mph) tornado outside Oklahoma City
- Our successful simulation used as its environment the conditions surrounding the El Reno supercell

The May 24, 2011 El Reno, OK, EF5 tornado

- On May 24, 2011, a supercell thunderstorm produced a long-track (65 mile long path) EF5 (peak winds estimated to exceed 210 mph) tornado outside Oklahoma City
- Our successful simulation used as its environment the conditions surrounding the El Reno supercell
- Simulated tornado is on the ground for over 65 miles, and produces surface winds exceeding 300 mph

The May 24, 2011 El Reno, OK, EF5 tornado

- On May 24, 2011, a supercell thunderstorm produced a long-track (65 mile long path) EF5 (peak winds estimated to exceed 210 mph) tornado outside Oklahoma City
- Our successful simulation used as its environment the conditions surrounding the El Reno supercell
- Simulated tornado is on the ground for over 65 miles, and produces surface winds exceeding 300 mph
- Winds this strong have been observed by Doppler radar, but our tornado is strong even for a "typical" EF5
    - First simulation was free slip (no friction) which may contribute to these unusually strong winds
    - Run with friction being analyzed; still produces EF5 tornado

The May 24, 2011 El Reno, OK, EF5 tornado

- On May 24, 2011, a supercell thunderstorm produced a long-track (65 mile long path) EF5 (peak winds estimated to exceed 210 mph) tornado outside Oklahoma City
- Our successful simulation used as its environment the conditions surrounding the El Reno supercell
- Simulated tornado is on the ground for over 65 miles, and produces surface winds exceeding 300 mph
- Winds this strong have been observed by Doppler radar, but our tornado is strong even for a "typical" EF5
  - First simulation was free slip (no friction) which may contribute to these unusually strong winds
  - Run with friction being analyzed; still produces EF5 tornado

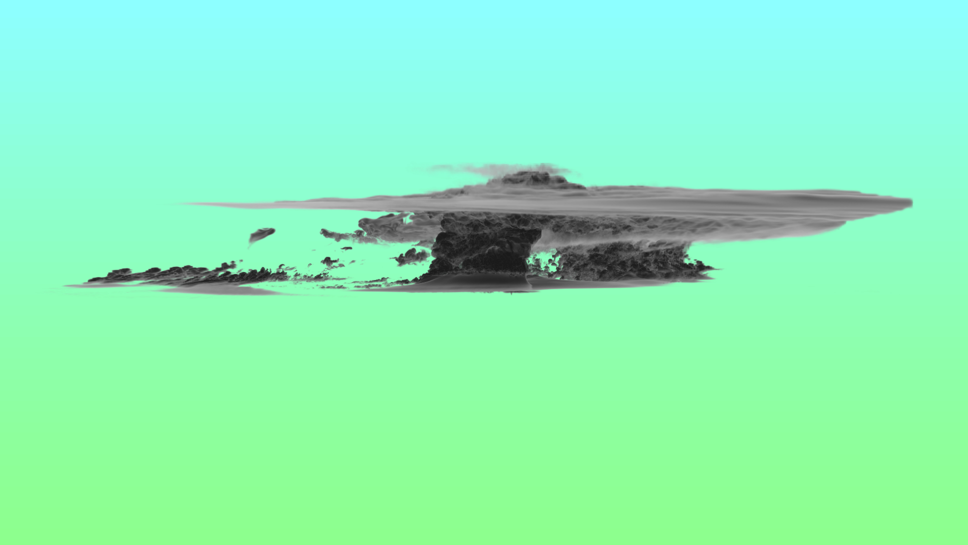
Visualizing the storm

- Following the successful simulation, we went back from restart files and saved data in 2 second intervals for visualization and analysis
  - This approach has been found to be practical, since the vast majority of our simulations do not produce the storms we wish to explore
  - I/O load was high and did increase wallclock time but not inordinately
- Volume rendering (VisIt raycasting) was used to visualize the cloud and rain fields, as well as the vorticity fields

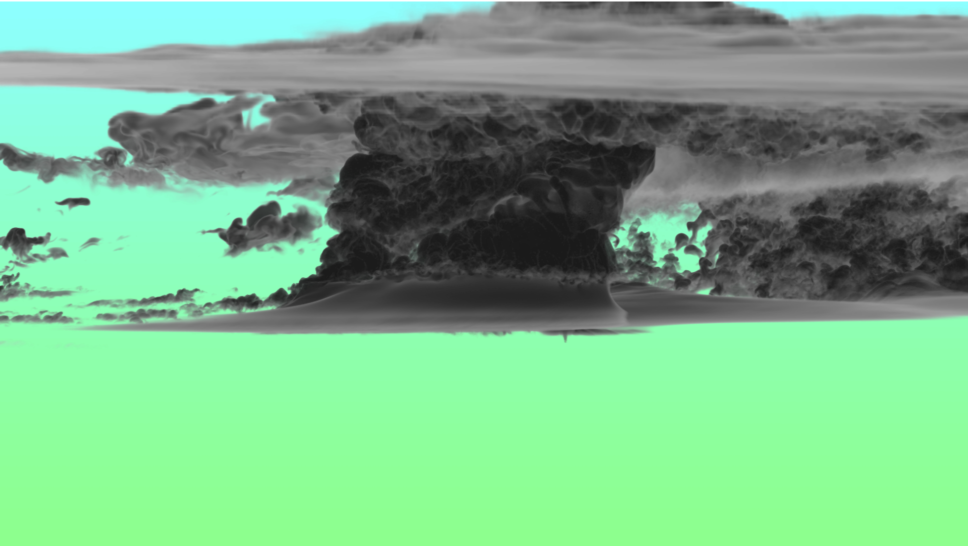
Visualizing the storm

- Following the successful simulation, we went back from restart files and saved data in 2 second intervals for visualization and analysis
  - This approach has been found to be practical, since the vast majority of our simulations do not produce the storms we wish to explore
  - I/O load was high and did increase wallclock time but not inordinately
- Volume rendering (VisIt raycasting) was used to visualize the cloud and rain fields, as well as the vorticity fields

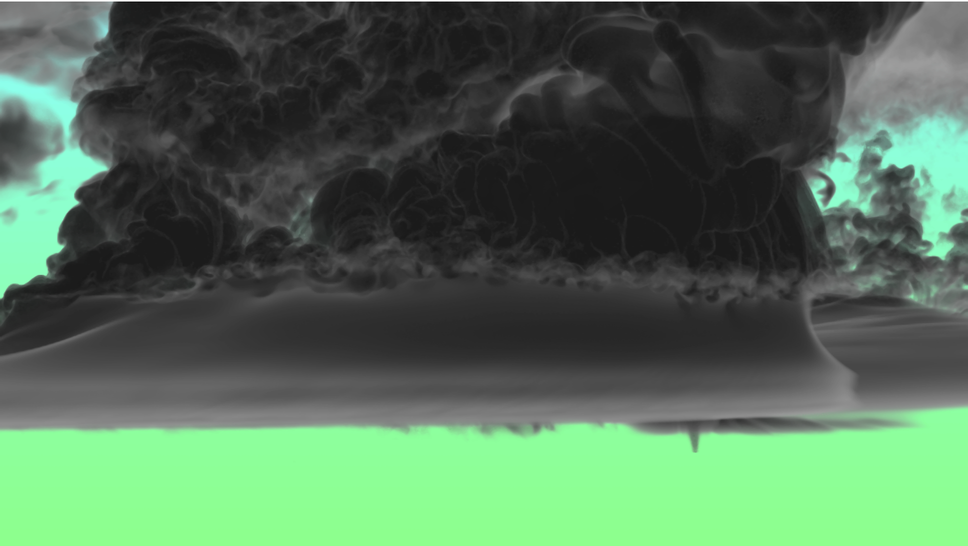
Far view of cloud field



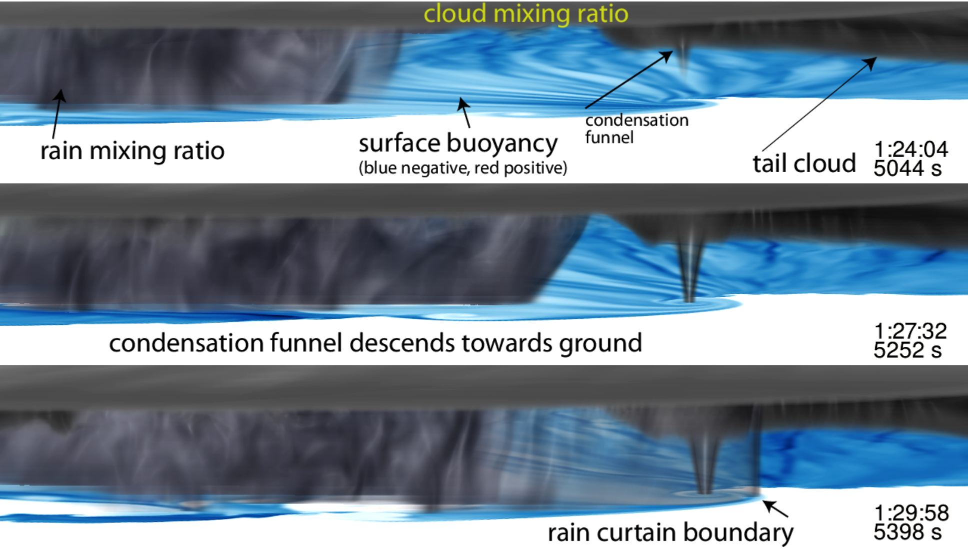
Far view of cloud field



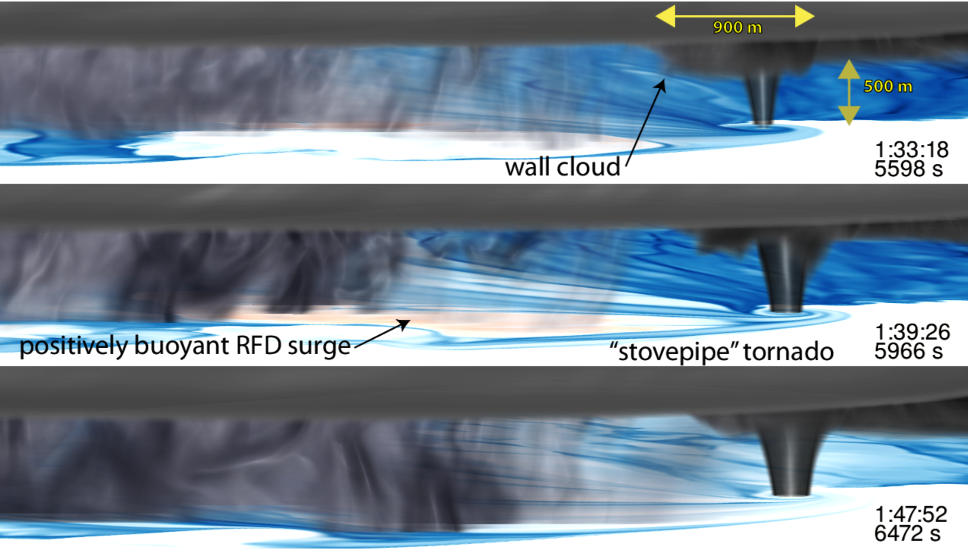
Far view of cloud field



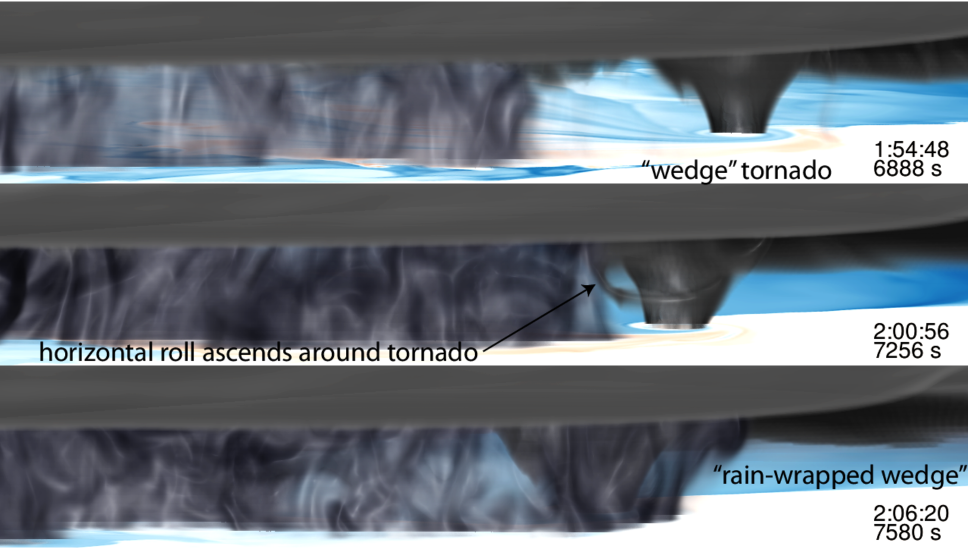
Near view of cloud/rain fields



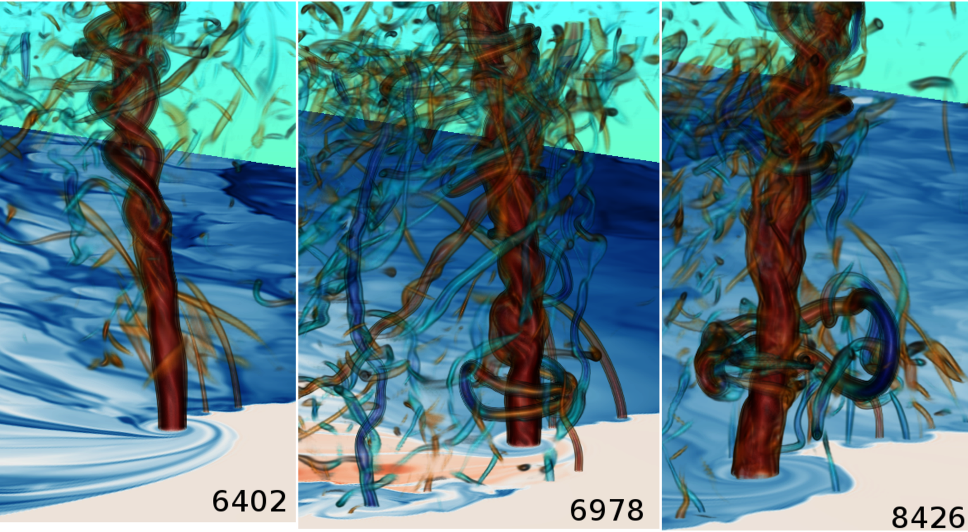
Near view of cloud/rain fields



Near view of cloud/rain fields



Near view of vorticity field



Issues with VisIt raycasting

- In order to achieve high quality imagery without artifacts, the "number of samples per ray" must be increased to 2000 or more
- Executing VisIt in parallel on Blue Waters with this configuration will result in the job failing
  - Memory utilization skyrockets with more samples per ray, and this is not alleviated by throwing more cores at the problem

Issues with VisIt raycasting

- In order to achieve high quality imagery without artifacts, the "number of samples per ray" must be increased to 2000 or more
- Executing VisIt in parallel on Blue Waters with this configuration will result in the job failing
  - Memory utilization skyrockets with more samples per ray, and this is not alleviated by throwing more cores at the problem
- Only way to consistently achieve high quality raycasting was to run *one (serial) VisIt engine per shared-memory node (leaving all other cores idling)*
  - Parallelization was therefore done in time, running dozens of instances concurrently

Issues with VisIt raycasting

- In order to achieve high quality imagery without artifacts, the "number of samples per ray" must be increased to 2000 or more
- Executing VisIt in parallel on Blue Waters with this configuration will result in the job failing
  - Memory utilization skyrockets with more samples per ray, and this is not alleviated by throwing more cores at the problem
- Only way to consistently achieve high quality raycasting was to run *one (serial) VisIt engine per shared-memory node (leaving all other cores idling)*
  - Parallelization was therefore done in time, running dozens of instances concurrently

Issues with VisIt raycasting

- Each 1920x1080 frame took roughly 20-25 minutes to render, utilizing about 30 GB of memory
- Automation was achieved using the VisIt python interface
- Master python script created a bunch of python scripts, each of which was executed using
  `visit -nowin -cli -s [scriptXXX.py]`
- Shell script fired off 90 jobs, each of which rendered 40 frames
- Due to the way VisIt works, each job stays attached to the login node from which the job is executed. Many files are opened; ran into `ulimit -n` issues (too many files open)
- End result: Took less than 24 wallclock hours to render 3600 frames at 1080p (producing 2m30s of video at 24 fps)

Issues with VisIt raycasting

- Each 1920x1080 frame took roughly 20-25 minutes to render, utilizing about 30 GB of memory
- Automation was achieved using the VisIt python interface
- Master python script created a bunch of python scripts, each of which was executed using `visit -nowin -cli -s [scriptXXX.py]`
- Shell script fired off 90 jobs, each of which rendered 40 frames
- Due to the way VisIt works, each job stays attached to the login node from which the job is executed. Many files are opened; ran into `ulimit -n` issues (too many files open)
- End result: Took less than 24 wallclock hours to render 3600 frames at 1080p (producing 2m30s of video at 24 fps)

Animations

- Movies can be found at http://orf5.com/bw2014

More simulations using May 12 environment

- Existing simulation is exciting but:
  - Simulation is free slip (no friction)
  - Rain gets caught in tornado due to lack of hydrometeor centrifuging
- Adding friction is straightforward; however simulation is extremely sensitive to initial conditions and going back to t=0 will not work

More simulations using May 12 environment

- Existing simulation is exciting but:
  - Simulation is free slip (no friction)
  - Rain gets caught in tornado due to lack of hydrometeor centrifuging
- Adding friction is straightforward; however simulation is extremely sensitive to initial conditions and going back to t=0 will not work
- Use of isotropic grid (which seems important) limits vertical resolution near ground, complicating adding surface friction

More simulations using May 12 environment

- Existing simulation is exciting but:
  - Simulation is free slip (no friction)
  - Rain gets caught in tornado due to lack of hydrometeor centrifuging
- Adding friction is straightforward; however simulation is extremely sensitive to initial conditions and going back to t=0 will not work
- Use of isotropic grid (which seems important) limits vertical resolution near ground, complicating adding surface friction

More simulations using May 12 environment

- Have already achieved some success running from restarts and having the storm move over a region of increasingly rough surface, using same mesh
- Using a different mesh but running from existing restarts will require some work
  - Will explore using meshes beyond what CM1 "allows"
  - We need to increase vertical resolution near the surface in order properly include the effects of surface roughness while preserving a realistic near-surface flow field

More simulations using May 12 environment

- Have already achieved some success running from restarts and having the storm move over a region of increasingly rough surface, using same mesh
- Using a different mesh but running from existing restarts will require some work
  - Will explore using meshes beyond what CM1 “allows”
  - We need to increase vertical resolution near the surface in order properly include the effects of surface roughness while preserving a realistic near-surface flow field

Long view

- Existing simulation is on the low side of desired resolution; doubling resolution will require around 16x more SU's per simulation
- Achieving the most useful science results will require a suite of simulations (one simulation is not enough; wish to avoid the "single hero simulation" problem)

Long view

- Existing simulation is on the low side of desired resolution; doubling resolution will require around 16x more SU's per simulation
- Achieving the most useful science results will require a suite of simulations (one simulation is not enough; wish to avoid the "single hero simulation" problem)
- Looking forward to moving beyond the technical challenges and moving directly into achieving meaningful science results

Long view

- Existing simulation is on the low side of desired resolution; doubling resolution will require around 16x more SU's per simulation
- Achieving the most useful science results will require a suite of simulations (one simulation is not enough; wish to avoid the "single hero simulation" problem)
- Looking forward to moving beyond the technical challenges and moving directly into achieving meaningful science results

Closing remarks

- A breakthrough simulation has been achieved - to the best of our knowledge, this is the first time a supercell producing long-track EF5 has been simulated
- The Blue Waters environment was crucial to the development, execution, data management, visualization, and post-processing (“end-to-end”)
- Existing tools are solid and now the real fun begins