BLUE WATERS

SUSTAINED PETASCALE COMPUTING

May 8th, 2014

Unlocking the Full Potential of the Cray XK7 Accelerator

Mark D Klein and John E Stone

NCSA NSF GREAT LAKES CONSORTIUM FOR PETASCALE COMPUTATION CRAY

Outline

- Background
- What we unlocked
- System Modifications
- Direct Benefits to Science
- Future Work
- Conclusion

Background

- Blue Waters does not have a dedicated visualization cluster available for science teams
  - Must move data off-site for processing
- Software rendering was available
  - Outdated API capabilities
  - Slow
  - Not taking advantage of hardware that exists
- Blue Waters has 4224 GPUs sitting in XK7 nodes available to science teams.
  - Why not use them for processing?

What we unlocked

- The ability for our 4224 Kepler GK110 GPUs to be used for graphics.
  - Off screen rendering to pbuffers
    - Fast
    - Current OpenGL API
  - More than just OpenGL acceleration
    - Video Transcoding
    - Computer Vision
    - Interoperability APIs

System Modifications

- What was missing from the Cray?
  - An X11 server
    - Currently required to bootstrap the graphics side of the GPU driver
  - A Graphics Driver
    - The kernel module version was incompatible with publically available graphics drivers
  - Graphics Operation Mode turned on
    - The GPUs have their graphics set to disabled

Adding an X server

- First Major blocking point for X
  - No CONFIG_VT in kernel.
    - Recompiling a custom kernel is just out the question
    - Patch two X initialization functions and recompile

```
void XF86OpenConsole(void){
        return;
        }
void XF86CloseConsole(void){
        return;
        }
```

Adding a Graphics Driver

- Second major blocking point!
  - Kernel module/nvidia driver mismatch, Cray has a custom minor version number.
- Request one on CrayPort
  - Should now be provided by default
    - If you have nvidia_drv.so, you're good.
    - Still need to move and link libraries into proper place for proper environment settings.
      - A script has been created to keep everything up to date
    - xorg.conf: Option "UseDisplayDevice" "none"

Graphics Operation Mode

- Not to be confused with “Compute Mode”
- Three modes
  - 0/ALL_ON
  - 1/COMPUTE ← This is the Cray default
  - 2/LOW_DP

Graphics Operation Mode

- Settable with nvidia-smi –gom
  - Currently requires a reboot to change
    - Possible this might change
  - Not a mode supported by Cray
    - May need to revert if a problem ever comes up
  - Testing concluded no harmful effects
    - Increases power draw by 1-2W
  - Running since March 2013 with no issues.

How does it work?

- Do not want X running all the time.
  - Many users do not need Xorg
    - Uses some resources to run
  - New Moab generic resource request to start them up on demand (and clean them up on end)
    - Similar to how CCM sets up various services
      - Query for "viz" gres request, if exists
        - populate nid list and feed to pcmd to launch Xorg
        - This is done by the Torque Prologue
        - Cleanup in the Epilogue

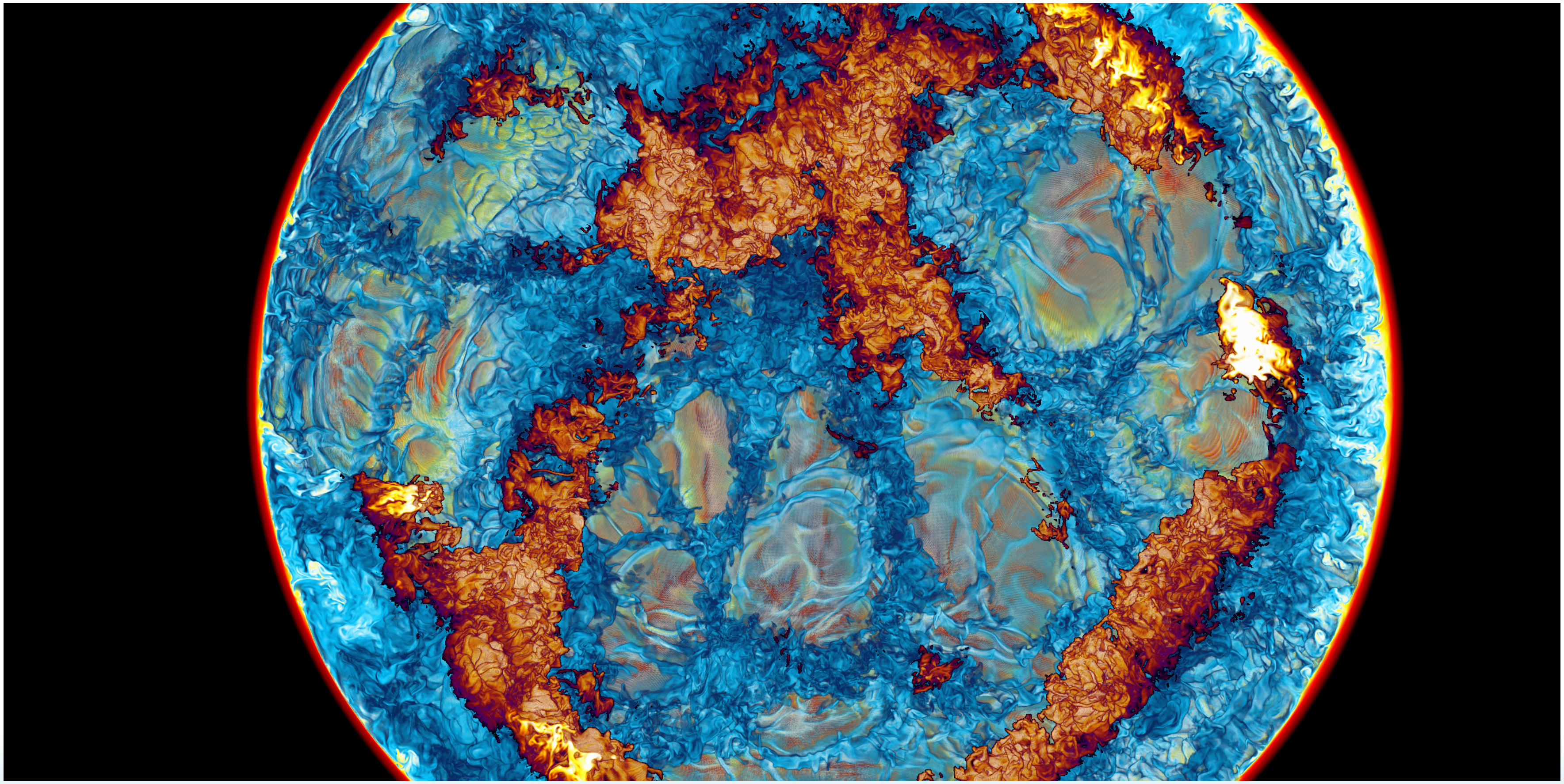
How does it work?

- Had to rebuild many packages to get everything working
  - Followed the Xorg from source wiki
  - An entire X tree in /usr/local/X11R7.7
    - Both shared root and external logins
- Created a loadable module that can be called to set up environment variables
  - Prepends PATH, LDFLAGS, LD_LIBRARY_PATH
  - DISPLAY=:1

Direct Benefits to Science

- Real Improvement in Time to Solution
  - Data transfer is slow
  - Many GPUs are available on the system
  - Ability to process as data is generated

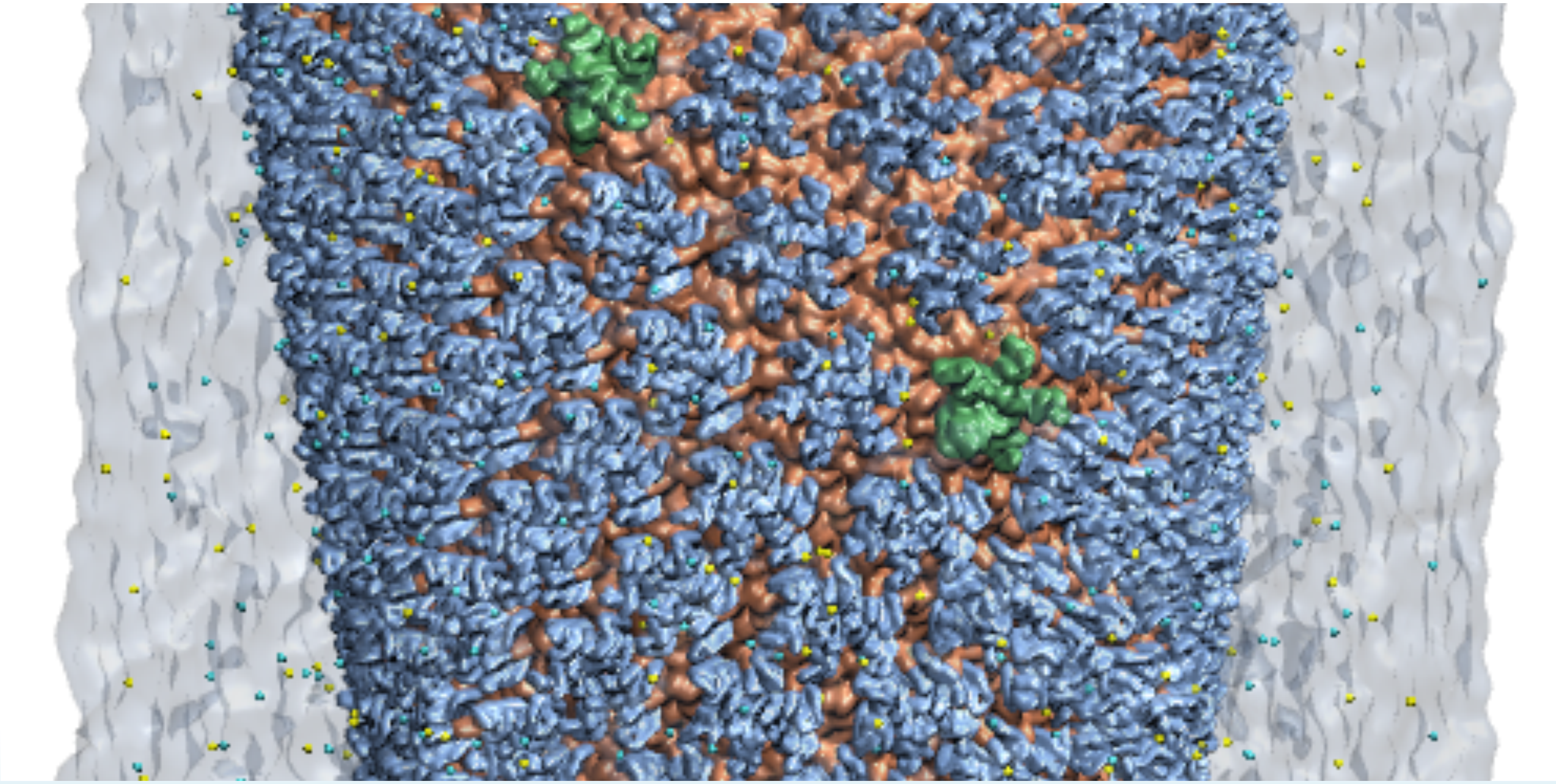
Real Improvement in Time to Solution (PPM)



Real Improvement in Time to Solution (PPM)

- Multifluid PPM stereo movie rendered on Blue Waters
  - Large amount of Data to Process
    - 26TB dataset @ 20MB/s ~= 15 days of transfer
  - Differences in system capabilities
    - Blue Waters has 4224 GPUs
      - 128 were used to render the movie in 24 hours
    - Their visualization cluster only has 6
      - Estimated 33 days to render
  - Resulting 38GB movie much more manageable to move
    - 38GB @ 20MB/s ~= 32 minutes of transfer
  - ~15-40x total speedup, 1 day to solution
    - 15 days saved on the data transfer
    - 32 days saved on the rendering

Real Improvement in Time to Solution (VMD)



Real Improvement in Time to Solution (VMD)

- Rapid movie renderings (they become I/O bound!)
- OpenGL renderings can "piggyback" on other analysis calculations at very little runtime cost and NO additional I/O
- OpenGL used as a very low-cost "fail-safe" for movie renderings where scenes using photorealistic rendering encounter segments with shadowing problems or other issues

Real Improvement in Time to Solution (VMD)

- NVIDIA OpenGL allows zero-copy interoperability with memory buffers used by CUDA, OptiX ray tracing, and the NVENC hardware video encoder found in Kepler-based XK7 Tesla K20X GPU.

Future Work

- Currently investigating ways to remotely display and interact directly
  - VirtualGL
  - Others?

Conclusion

- Enabling Graphics Functionality on the GPU:
  - Is fairly easy and has low impact on the system
  - Allows for a wide range of visualization tools to be used on data directly without the need to copy off of the system.
  - Creates unique opportunities to speed up time to solution

Acknowledgements

- This research is part of the Blue Waters sustained-petascale computing project supported by NSF award OCI 07-25070, and “The Computational Microscope" NSF PRAC award.
- The CUDA Center of Excellence at the University of Illinois, and NIH funding through grants 9P41GM104601 and 5R01GM098243-02.
- Dave Semeraro of NCSA, and Paul Woodward and Mike Knox of the University of Minnesota for their involvement in developing, using, and evaluating the fully GPU-accelerated OpenGL rasterization capabilities implemented on Blue Waters.