

Rendering simulations using VMD MPI on Stampede2 Supercomputer

The following tutorial introduces the use of high-performance massively parallel computing to render high quality figures, from molecular dynamics (MD) trajectories, using VMD MPI. The capability of VMD to render scenes enabling shadows and ambient occlusion techniques is provided by the Tachyon ray tracer. As a result, figures display better description of shape and depth of molecular systems.

The example corresponds to the scene rendering of 1 μ s of MD trajectory using VMD MPI. The use of parallel computing, in combination with fast multiprocessor ray tracer, enhances performance of large-scale trajectories rendering.

1 Setting up VMD display settings

The initial step to render any trajectory is setting up the preferred view and representation of the system. Since VMD builds on supercomputers contain default display settings, visual representations need to be adjusted according to the system of study.

1.1 Loading system on local machine

- Type on the *vmd* to open VMD on your local machine
- Go to Extensions → Tk Console

On the Tk console load the topology and trajectory file

- ```
> Main < (test mpi stampede 2) 1 % set molid [mol new coordinates.pdb] 0
> Main < (test mpi stampede 2) 2 % animate delete all
> Main < (test mpi stampede 2) 3 % mol addfile production.dcd wait for all
```

The command `mol new` assigns and ID to the loaded structure. The variable *molid* contains the molecule ID, which in turns allows the user to specify the target molecule.

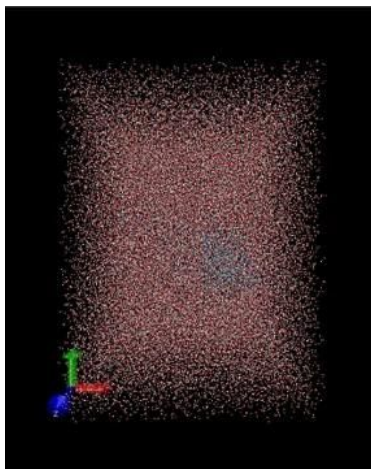


Figure 1: Default visualization on VMD.

## 1.2 Setting visualization commands on console

The additional changes on the display settings are going to be performed through the GUI of VMD. In order to keep track of the commands to add to the final rendering script, the Tcl commands are printed in the Tk console.

- Go to File → Log Tcl commands to console
- Set Display → Orthographic
- Set Display → Axes → Off
- Set Graphics → Colors → Categories: Display, Names: Background, Color: White
- Graphics-> Representations. . . -> Coloring method: Secondary Structure, Drawing Method: New Cartoon, Material: AOChalky, Resolution: 50

As a result, commands are printed on the Tk console.

```
Info) Logging commands to 'console'.
Info) # VMD for LINUXAMD64, version 1.9.4a7 (July 12, 2017)
Info) # Log file 'console', created by user fabiog
Info) display projection Orthographic
Info) axes location Off
Info) menu color off
Info) menu color on
Info) color Display Background white
Info) mol modselect 0 3 protein
Info) display resetview
Info) mol modcolor 0 3 Structure
Info) mol modstyle 0 3 NewCartoon 0.300000 10.000000 4.100000 0
Info) mol modmaterial 0 3 AOShiny
Info) mol modstyle 0 3 NewCartoon 0.300000 15.000000 4.100000 0
>Main< (test_mpi_stampede2) 14 % |
```

Figure 2: Snapshot of commands printed on the console.

In addition, the orientation and view of the system be adjusted using the commands *rotate* and *scale*. A visual inspection of the molecule view is recommended.

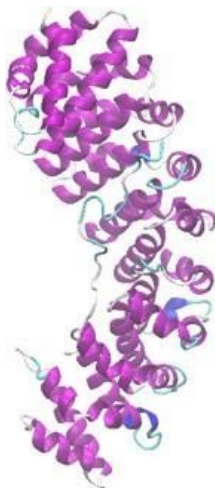


Figure 3: Visual inspection of the system.

## 2 Supercomputer assisted visualization

Login to Stampede2 from the Linux command console via *ssh*. In addition, copy the local directory to the work space using *scp*.

```
localhost $ ssh myusername@stampede2.tacc.utexas.edu
```

```
localhost $ scp -r render_mpi_stampede2.tar.gz myusername@stampede2.tacc.utexas.edu
```

After merging the files, the visualization settings are loaded to the submission script *render\_figures\_mpi.tcl*, under the *mpianalyze* procedure.

```
proc mpianalyze { framecount trajectoryfile } {
 set noderank [parallel noderank]
 set nodecount [parallel nodecount]
 set block [blockdecompose $framecount]
 set start [lindex $block 0]
 set end [lindex $block 1]
 set len [expr $end - $start + 1]
 parallel barrier; # wait for all nodes to reach this point
 puts "Node $noderank, frame range: $start to $end, $len frames total"
 parallel barrier; # wait for all nodes to reach this point
 mol addfile $trajectoryfile first $start last $end step 1 waitfor all
 parallel barrier; # wait for all nodes to reach this point
 puts "Node $noderank, loaded [molinfo top get numframes]"
 parallel barrier; # wait for all nodes to reach this point

 source my.ss.colors.tcl
 display projection Orthographic
 display resize 400 800
 display shadows on
 display ambientocclusion on
 axes location Off
 color Display Background white
 mol modelselect 0 0 protein
 mol modcolor 0 0 Structure
 mol modstyle 0 0 NewCartoon 0.300000 50.000000 4.100000 0
 mol modmaterial 0 0 AOChalky
 mol smoothrep 1 0 2
 display resetview
 rotate y by -340
 rotate z by -40
 scale by 1

 set dir /work/0220/fabiogon/stampede2/test_mpi_stampede2/figs
 set tachyondir /work/0220/fabiogon/stampede2/vmd-1.9.4a43/lib/tachyon
 # save the visualization state to the submission script
 for {set frame 0} { $frame <= [expr $len - 1] } {incr frame 1} {
 set ts [expr $frame + $start]
 set frameinput $(dir)/figure.$(ts).dat
 take picture reset
 take picture format $(dir)/figure.$(ts).dat
 take picture method Tachyon
 animate goto $frame
 puts "Frame: $frame"
 take picture
 exec $(tachyondir)/tachyon_LINUXAMD64 -aasamples 12 -format TARGA $(dir)/figure.$(ts).dat -o $(dir)/figure.$(ts).dat.tga
 }
 parallel barrier; # wait for all nodes to reach this point
}
```

```

VMD TkConsole
Ew Console Edit Help Dev Help
info display projection Orthographic
main (render_mpi_stampede2) 1 % display resize 400 800
info menu display off
info menu display on
info display shadows on
info display ambientocclusion on
info axes location Off
info menu color off
info menu color on
info color display background white
info mol modelselect 0 0 protein
info mol modcolor 0 0 Structure
info mol modstyle 0 0 NewCartoon 0.300000 50.000000 4.100000 0
info mol modmaterial 0 0 AOChalky
info display resetview
Waiter (render_mpi_stampede2) 6 %

```

Figure 4: Addition of display settings to submission script.

After modifying the submission script, the job is then launched using the Slurm Workload Manager using the provided `runbatch.sh` script. The job status can be monitored using the `watch` command.

```
localhost . /runbatch_mpi.sh
localhost watch -n 20 squeue -u username
```



Figure 5: Rendered scene with Tachyon