



LEADERSHIP
COMPUTING
FACILITY

September 30, 2026 | Monthly User Call

Announcing Riker, OLCF's New Data Analysis Cluster

Bill Renaud & Michael Sandoval

ORNL/NCCS



U.S. DEPARTMENT
of **ENERGY**

ORNL IS MANAGED BY UT-BATTELLE LLC
FOR THE US DEPARTMENT OF ENERGY

FRONTIER



Agenda

- Riker Overview
 - Access
 - Hardware Specs
 - Node Sharing Examples
- Visualization Tool Changes
 - VisIt
 - ParaView
- What's Next

Warning: Andes is set to be decommissioned.

- All current Andes users gained access to Riker on September 10th.
- Riker and Andes will operate in parallel **until the end of October** to provide users time to transition their workflows.
- Users are encouraged to begin their transition to Riker as soon as possible to help identify issues with porting before Andes enters decommission.
- At the end of this transition period, Andes will be decommissioned.

What is Riker?

Riker is a commodity-type Linux cluster with the primary purpose of being used as a conduit for large-scale scientific discovery via pre/post processing, analysis, and visualization of simulation data generated on Frontier.

Riker is a 136-node system with 128 AMD EPYC CPU-only nodes and 8 hybrid AMD EPYC CPU + NVIDIA L40S GPU nodes.

The new system will provide OLCF users with upgraded capabilities for data analysis and visualization workflows.

Keeping with Andes scheduling policies and the implementation of node-sharing, Riker has been positioned to be as available as possible to smaller and longer running workloads.

Accessing Riker

All projects with an active allocation to Andes have received access to Riker and an allocation of 1,000 Riker Node-Hours to start.

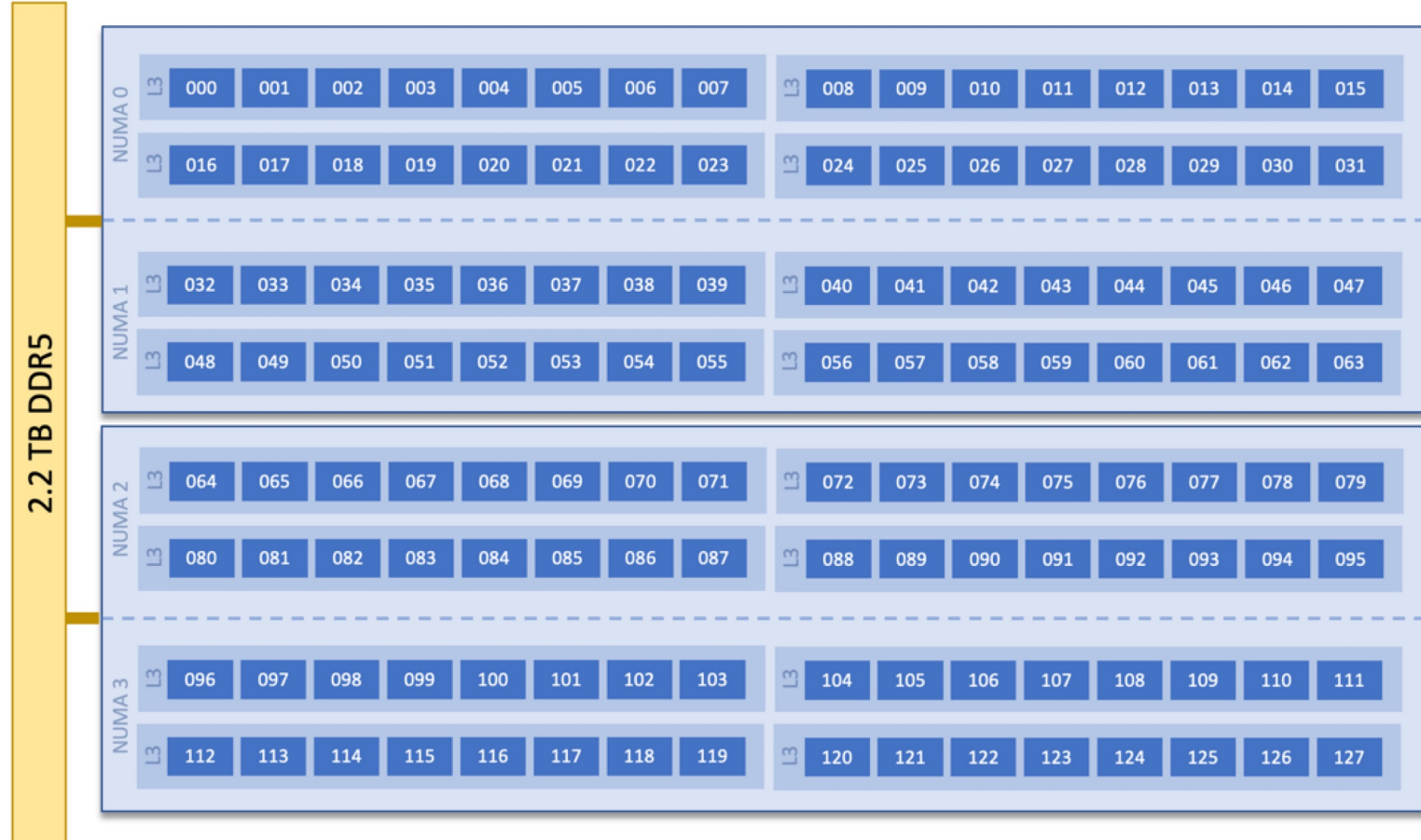
To connect to Riker, Users will ssh to `riker.olcf.ornl.gov` using your OLCF Username and Passcode (PIN+Tokencode)

```
$ ssh username@riker.olcf.ornl.gov
```

There are 4 login nodes which provide an environment for editing, compiling, and launching codes onto the compute nodes. Like other OLCF systems, production work should not be conducted on login nodes.

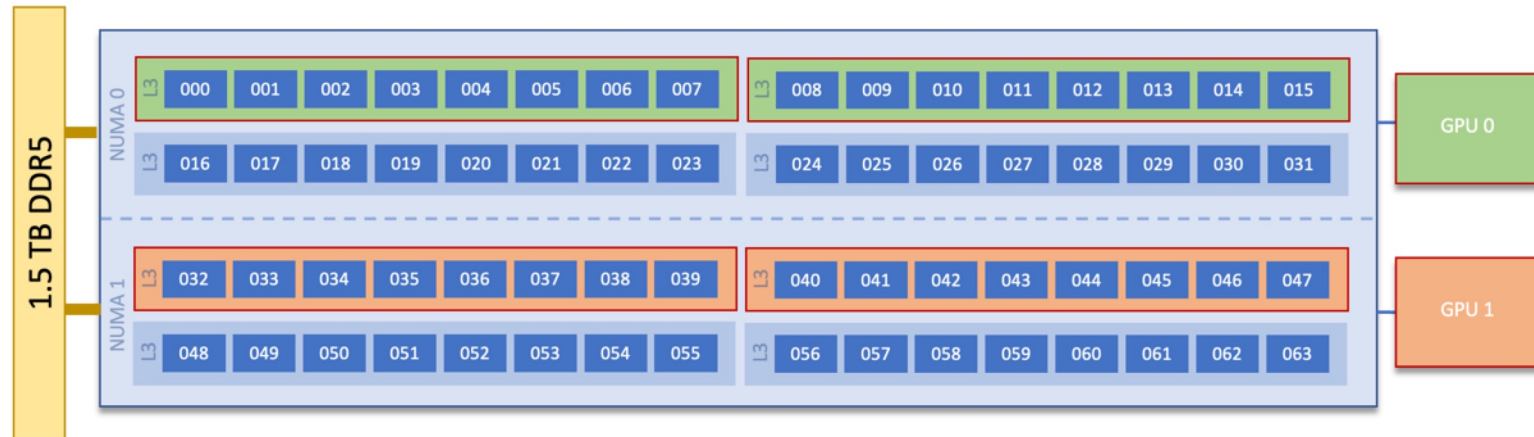
Overview of Riker Hardware: CPU Nodes

- 128 CPU Compute nodes
- [2x] 64-Core AMD EPYC 9534 CPUs
- 2.2TB of DDR5 Memory



Overview of Riker Hardware: GPU Nodes

- 8 GPU Compute nodes
- [1x] 64-Core AMD EPYC 9575F CPU
- 1.5TB of DDR5 Memory
- Two 48GB NVIDIA L40S GPUs



Filesystems available on Riker

NFS Network File System

- User home: /ccs/home/[userid]
- Project home: /ccs/proj/[projid]
- Long-term storage for your small data
- Read/write from Riker compute nodes; however, users should use Orion to launch/run jobs.
- Not purged
- Quota of 50GB (may request increase in well justified cases)
- There is an automated backup

Orion

- Parallel filesystem based on Lustre (/lustre/orion/).
- Member Work: Short-term user data for fast, batch-job access that is not shared with other project members.
 - /lustre/orion/[projid]/scratch/[userid]
- Project Work: Short-term project data for fast, batch-job access that's shared with other project members.
 - /lustre/orion/[projid]/proj-shared
- World Work: Short-term project data for fast, batch-job access that's shared with users outside your project.
 - /lustre/orion/[projid]/world-shared
- Data that has not been accessed in 90-days are eligible for purge.

Programming Environment on Riker

The programming environment on Riker is managed using LMOD Lua modules.

All available software packages and scientific libraires can be found using the LMOD `Module avail` command.

Similar to Andes, Riker utilizes the Slurm Batch Scheduler for job scheduling.

The software stack itself is an OLCF-maintained stack consisting of various compilers, libraries and applications

adios2/2.12.1-mpi	boost/1.88.0-mpi	fftw/3.3.11-mpi-omp	hdf5/1.14.6-mpi	nco/5.3.9-omp	netcdf-cxx4/4.3.1	osu-micro-benchmarks/7.5.2-mpi	scorep/9.4-mpi	vtk/9.5.1-mpi
amdfw/5.1-mpi-omp	cdo/2.6.2-omp	globalarrays/5.8.2-mpi	mpifileutils/0.12-mpi	ncview/2.1.9	netcdf-fortran/4.6.2	parallel-netcdf/1.14.1-mpi	valgrind/3.27.1-mpi	
amdscalapack/5.1-mpi	ferret/7.6.0	gromacs/2026.1-mpi-omp	mpip/3.5-mpi	netcdf-c/4.10.0-mpi	netlib-scalapack/2.2.3-mpi	paraview/6.1.1-mpi	vtk-m/2.3.0-mpi-omp	
----- [gcc/14.2.0, mpich/5.0.1] -----								
amdblis/5.1-omp	amdlbflame/5.1-omp	aocl-compression/5.1	aocl-libmem/5.1	fftw/3.3.11-omp (D)	hdf5/1.14.6 (D)	mpich/5.0.1 (L)	openblas/0.3.33-omp	
amdfw/5.1-omp (D)	amdlbm/5.1	aocl-crypto/5.1	aocl-sparse/5.1-omp	gmp/6.3.0	hwloc/2.13.0	netlib-lapack/3.12.1	ucx/1.20.1 (L)	
----- [gcc/14.2.0] -----								
automake/1.18.1	ccache/4.12.1	exuberant-ctags/5.8	git/2.53.0	gnuplot/6.0.0	htop/3.4.1	libtool/2.5.4	makedepend/1.0.9	ninja/1.13.2
bazel/7.7.1	cmake/3.31.11	gdb/17.1	gmake/4.4.1	go/1.26.4	imagemagick/7.1.1-39	libtree/3.1.1	nano/8.7	papi/7.2.0
bzip2/1.0.8	emacs/30.2	ghostscript/10.06.0	gnupg/2.5.20	gsl/2.8	jq/1.8.1	libyaml/0.2.5	ncl/6.6.2-omp	numactl/2.0.19
----- [Core/26.01] -----								
qt-base/6.11.1	screen/5.0.1	tcl/8.6.17	visit/3.5.0	vmd/1.9.3-cpu				
qt/5.15.18	socat/1.7.4.4	texlive/20260301						
r/4.5.3	subversion/1.14.5	tmux/3.6a (L)						
----- [Base Modules] -----								
Core/26.01 (L)	DefApps (L)	aocc/5.1.0	cuda/12.9.1	cuda/13.3.0 (D)	gcc/14.2.0 (L)	llvm/21.1.8	matlab/R2026a	miniforge3/25.3.1-0
								nvhpc/26.3
								oneapi/2025.2.1
								paraview/6.1.1-gpu-mpi
								vampir/10.8.1
								visit/3.5.0

Node-sharing on Riker Batch / CPU nodes

Users are required to specify a number of cores or an amount of memory when allocating a job on Riker.

Cores and Memory are a locked ratio or chunk. Increase Cores to get more memory. The smallest request is 1 Core : 17240M of Memory.

Slurm will round your request up to the nearest chunk.

```
[loganwgillum@riker-login3 ~]$ salloc -A stf007uanofn -N1 -t 10 -p batch
salloc: error: 1 ERROR FOUND:
salloc: error:
salloc: error: -----
salloc: error: You must specify a cpu selection with '-c'.
salloc: error:
salloc: error: or
salloc: error:
salloc: error: You must specify a memory selection with '--mem'.
salloc: error: -----
salloc: error:
salloc: error:
salloc: error: Job not submitted, please update and resubmit batch job.
salloc: error: Job submit/allocate failed: Unspecified error
```

Example: You are a current Andes user and are wanting to move to Riker.

Your job current runs across 4 nodes using all 32 cores per node. What would you need to do to transition to Riker?

Method 1: Add the cores request to your job submission script and scale out like normal.

Andes

```
#!/bin/bash
#SBATCH -A STF007
#SBATCH -J hello
#SBATCH -o %x-%j.out
#SBATCH -t 0:10:00
#SBATCH -p batch
#SBATCH -N 4

srun -N 4 --ntasks-per-node=32 -c 1 ./run
```



Riker

```
#!/bin/bash
#SBATCH -A GEN107
#SBATCH -J hello
#SBATCH -o %x-%j.out
#SBATCH -t 0:10:00
#SBATCH -p batch
#SBATCH -N 4
#SBATCH -c 32

srun -N 4 --ntasks-per-node=32 -c 1 ./run
```

Method 2: Utilize all 128 cores on a node using tasking instead of scaling out across multiple nodes.

Andes

```
#!/bin/bash
#SBATCH -A stf007uanofn
#SBATCH -J hello
#SBATCH -o %x-%j.out
#SBATCH -t 0:10:00
#SBATCH -p batch
#SBATCH -N 4

srun -N 4 --ntasks-per-node=1 -c 32 ./run
```



Riker

```
#!/bin/bash
#SBATCH -A stf007uanofn
#SBATCH -J hello
#SBATCH -o %x-%j.out
#SBATCH -t 0:10:00
#SBATCH -p batch
#SBATCH -N 1
#SBATCH -c 128

srun -N 1 --ntasks-per-node=4 -c 32 ./run
```

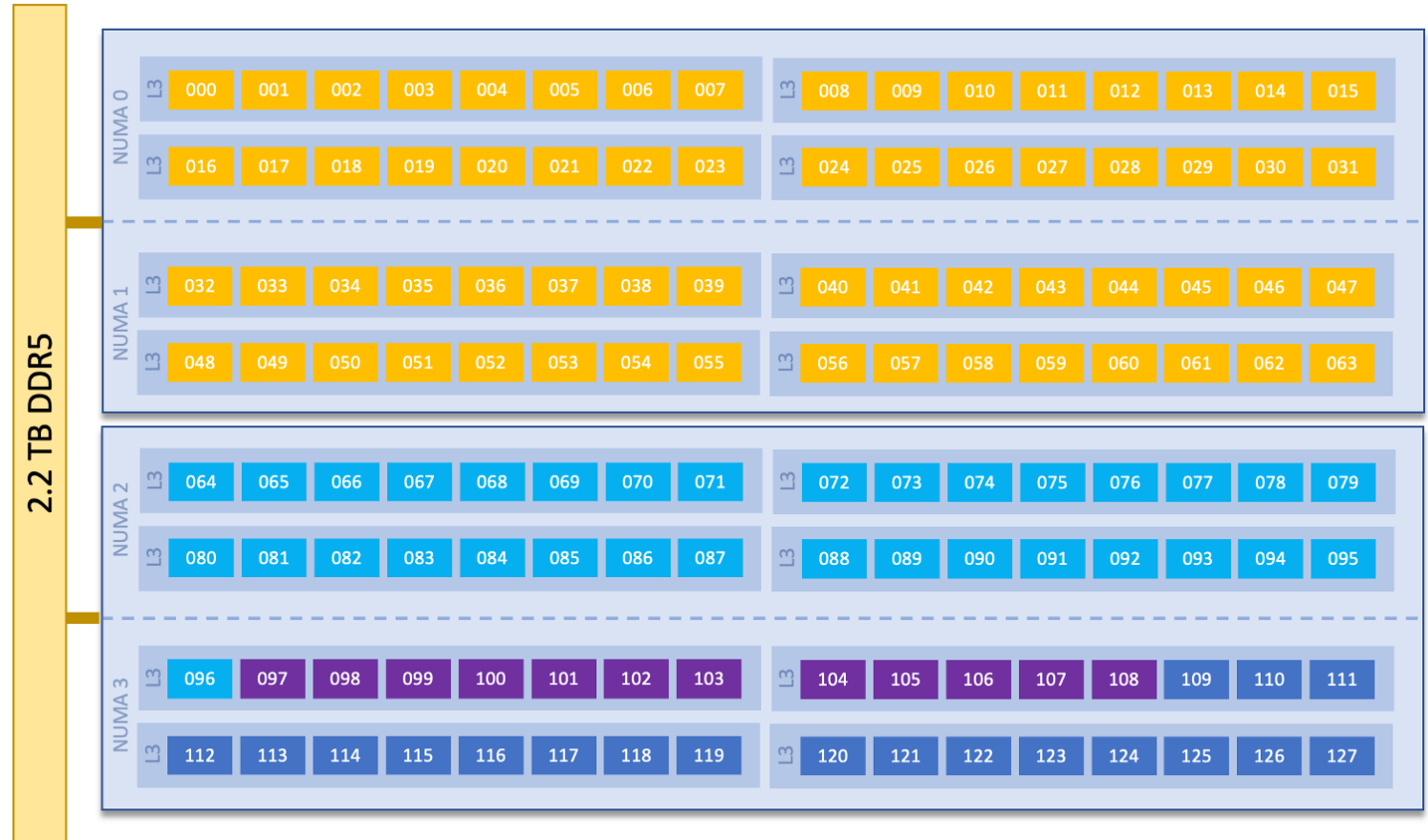
Example: Let us assume there are two users already running on a batch Riker node.

- Yellow User has 64 cores allocated.
- Light Blue User has 33 cores allocated.
- Purple User wants to slot into that same node filling in more of the unused cores proportional to their memory request.

Purple would append the following header:
`#SBATCH --mem=200G`

Slurm would figure out how many cores to allocate to the job and fill in the corresponding number of cores to satisfy the memory request.

$200000M / 17240M = 11.6$ cores of memory. Slurm would allocate 12 cores to cover that memory request.

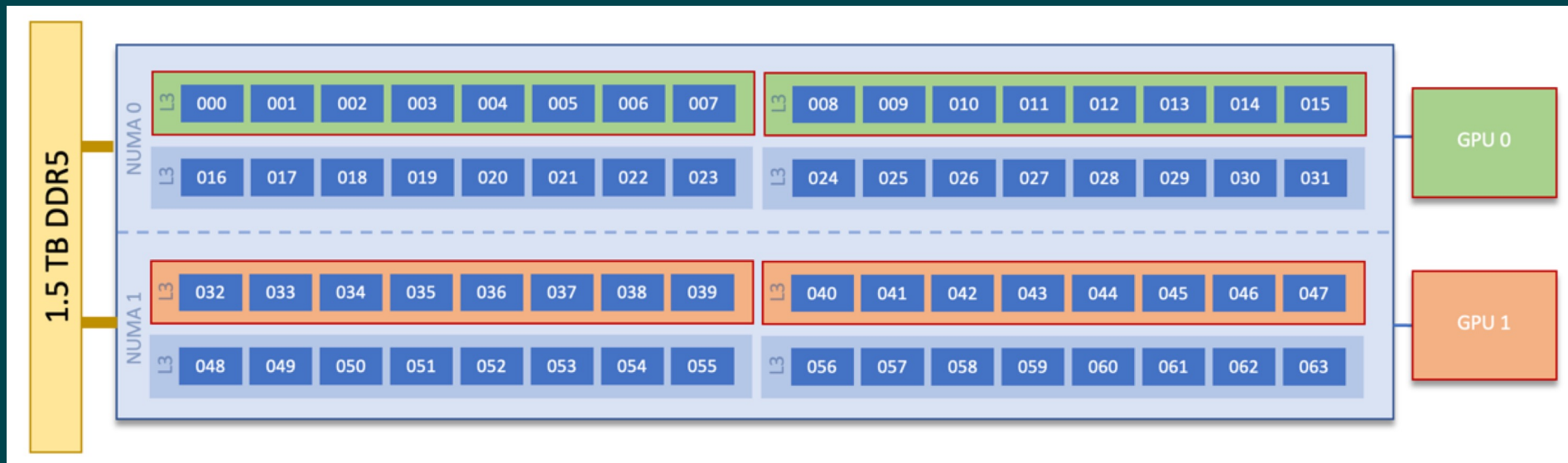


Preface for GPU Nodes

The Riker GPU nodes take a unique approach to node-sharing. Similar to Batch nodes, each node can be allocated on a 1-Core basis and come with an equal share of memory. (1Core : 23444M)

Where it differs is the Reserved Cores mechanism that we use to protect GPU utilization. 32 Cores (16 per GPU) are reserved for the GPUs and can only be allocated if you also allocate the bound GPU. This is reflected in the Green/Orange cores below. These cores & memory shares are automatically allocated when allocating a GPU.

The cores that are not bound to a GPU are referred to as “Flex/Shared” cores. This essentially means that they can be allocated by GPU-Enabled workloads and CPU-Only workloads.



Node-sharing on Riker GPU nodes

Riker GPU nodes have two modes for scheduling resources: GPU-Enabled and CPU-Only.

GPU-Enabled Workloads must use the flag: `--gpus` or one of its derivatives. **This is the total amount of GPUs**

Use of the `--gpus=1` flag is sufficient for getting a default allocation with 1 GPU and 16 Cores/Shares of Memory.

Additionally, when using GPUs, you must use `--cpus-per-gpu` or `--mem-per-gpu` flags when asking for resources above defaults with GPUs (Flex/Shared Cores).

```
[loganwgillum@riker-login4 ~]$ salloc -A gen107 -t10 -N1 -p gpu --gpus=1
salloc: GPU-Enabled Job
salloc: Granted job allocation 73408
salloc: Waiting for resource configuration
salloc: Nodes riker-gpu1 are ready for job
bash-5.1$
```

CPU-Only Workloads can use the same flags as Batch nodes.

Users are required to specify a number of cores or an amount of memory when allocating a CPU-Only job on Riker GPU nodes.

Utilizes the Flex/Shared Cores on each node.

```
[loganwgillum@riker-login4 ~]$ salloc -A gen107 -t10 -N1 -p gpu
salloc: error: 2 ERRORS FOUND:
salloc: error:
salloc: error: -----
salloc: error: You must specify a cpu selection with '-c'.
salloc: error:
salloc: error: or
salloc: error:
salloc: error: You must specify a memory selection with '--mem'.
salloc: error: -----
salloc: error:
salloc: error: -----
salloc: error: For GPU-Enabled jobs, you must specify a gpu selection with '--gpus=',
salloc: error: -----
```

Example: You job uses all 4 GPUs on an Andes GPU node; how would you change your job to transition to Riker?

Riker GPU nodes have 2 GPUs per node, so you would need 2 nodes. To request the GPUs on each node, you would need to add the `--gpus` flag or one of its derivatives.

Andes

```
#!/bin/bash
## ALLOCATION TIME RESOURCE REQUESTS ##
#SBATCH -A stf007uanofn
#SBATCH -J tst_job
#SBATCH -o %x-%j.out
#SBATCH -t 00:05:00
#SBATCH -p gpu
#SBATCH -N 1

#<Load Modules>

## RUNTIME RESOURCE DELEGATION ##
srun -n4 --gpus-per-task=1 hello_jobstep
```

```
MPI 000 - OMP 000 - HWT 000 - Node andes-gpu1 - RT_GPU_ID 0 - GPU_ID 0 - Bus_ID 0000:06:00.0
MPI 001 - OMP 000 - HWT 001 - Node andes-gpu1 - RT_GPU_ID 0 - GPU_ID 1 - Bus_ID 0000:07:00.0
MPI 002 - OMP 000 - HWT 028 - Node andes-gpu1 - RT_GPU_ID 0 - GPU_ID 2 - Bus_ID 0000:84:00.0
MPI 003 - OMP 000 - HWT 029 - Node andes-gpu1 - RT_GPU_ID 0 - GPU_ID 3 - Bus_ID 0000:85:00.0
```

Riker

```
#!/bin/bash
## ALLOCATION TIME RESOURCE REQUESTS ##
#SBATCH -A stf007uanofn
#SBATCH -J tst_job
#SBATCH -o %x-%j.out
#SBATCH -t 00:05:00
#SBATCH -p gpu
#SBATCH -N 2
#SBATCH --gpus=4

module load cuda

## RUNTIME RESOURCE DELEGATION ##
srun -n4 --gpus-per-task=1 hello_jobstep
```

```
MPI 002 - OMP 000 - HWT 000 - Node riker-gpu3 - RT_GPU_ID 0 - GPU_ID 0 - Bus_ID 0000:81:00.0
MPI 003 - OMP 000 - HWT 032 - Node riker-gpu3 - RT_GPU_ID 0 - GPU_ID 0 - Bus_ID 0001:C1:00.0
MPI 000 - OMP 000 - HWT 000 - Node riker-gpu2 - RT_GPU_ID 0 - GPU_ID 0 - Bus_ID 0000:81:00.0
MPI 001 - OMP 000 - HWT 032 - Node riker-gpu2 - RT_GPU_ID 0 - GPU_ID 0 - Bus_ID 0001:C1:00.0
```

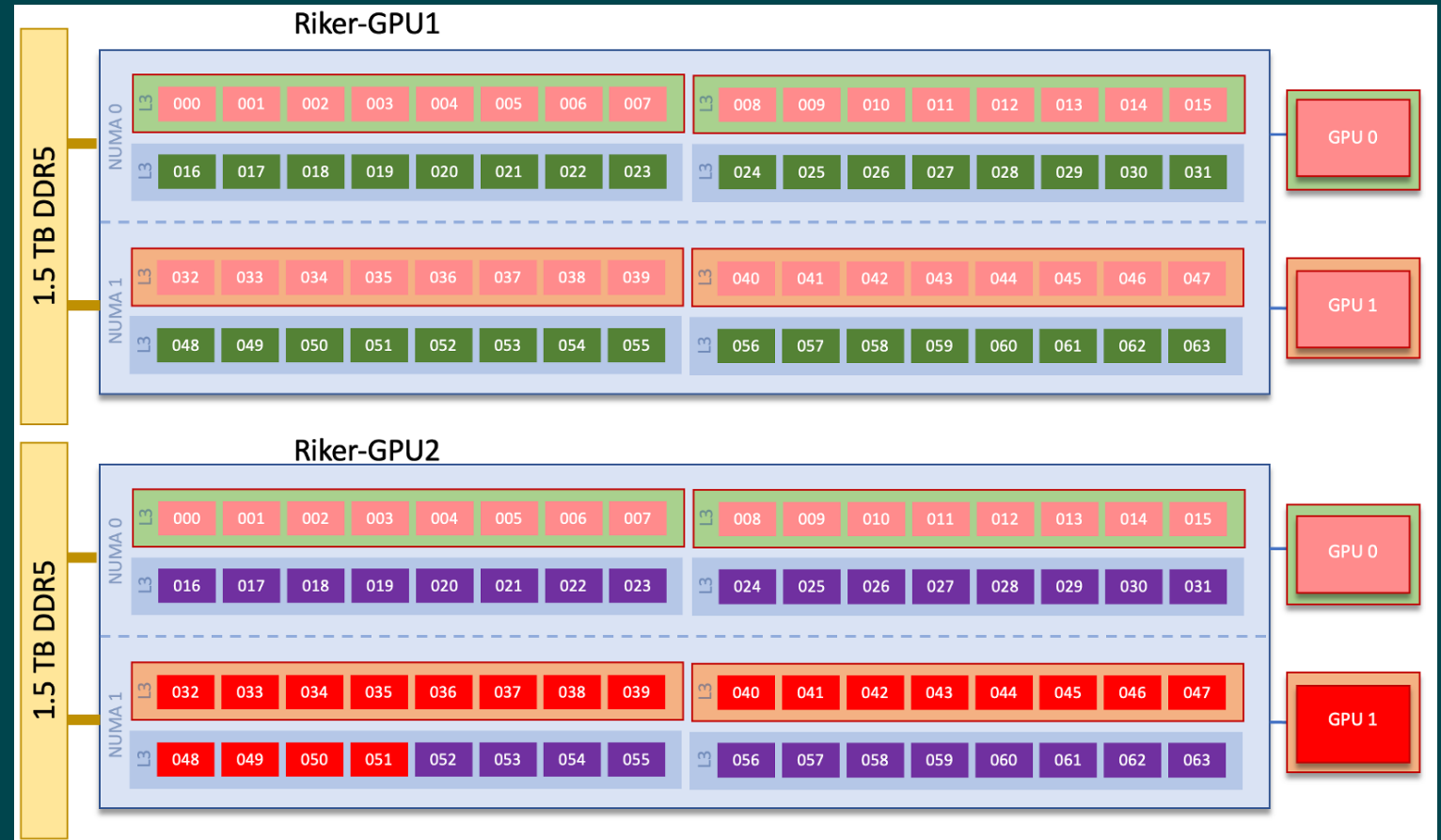
Example: Let us assume there are three users already running on two riker-gpu nodes.

- Pink User has 3 GPUs allocated across riker-gpu1 and riker-gpu2,
- Green User allocated the remaining 32 cores on riker-gpu1 for a CPU-Only workload, and
- Purple User allocated 28 cores on riker-gpu2 for a CPU-Only workload.

Red user wants to slot into the second GPU on riker-gpu2, flexing to 20 cores instead of the default 16 cores to consume the rest of the resources on riker-gpu2.

```
#!/bin/bash
## ALLOCATION TIME RESOURCE REQUESTS ##
#SBATCH -A <project_id>
#SBATCH -J <job_name>
#SBATCH -o %x-%j.out
#SBATCH -t 00:05:00
#SBATCH -p gpu
#SBATCH -N 1
#SBATCH --gpus=1
#SBATCH --cpus-per-gpu=20

## RUNTIME RESOURCE DELEGATION ##
srun -n1 --cpus-per-task=20 --gpus-per-task=1 ./a.out
```



Exclusive Mode

Need the entire node, or just do not want to node-share?

Riker supports Slurm Exclusive mode access to all the nodes.

If a user wants to own the entire node, they can use the `--exclusive` flag in their batch job SBATCH Headers or their interactive Salloc submit line.

Batch Job Usage

```
#!/bin/bash
## ALLOCATION TIME RESOURCE REQUESTS ##
#SBATCH -A stf007uanofn
#SBATCH -J tst_job
#SBATCH -o %x-%j.out
#SBATCH -t 00:05:00
#SBATCH -p gpu
#SBATCH -N 2
#SBATCH --exclusive

module load cuda

## RUNTIME RESOURCE DELEGATION ##
srun -n4 --gpus-per-task=1 -c1 hello_jobstep
```

Interactive Job Usage

```
$ salloc -A stf007 -N1 -t 10 -p gpu --exclusive
```

Job Accounting on Riker

Jobs on Riker are scheduled in partial-node increments. The OLCF charges based on what a job makes *unavailable* to other users, so users are encouraged to only use what their job requires. Allocations on Riker are separate from those on Frontier and other OLCF resources.

The *node-hour* charge for each job will be calculated as follows:

$$\text{node-hours} = (\{\text{weight for resource}\} * \{\text{Resource used}\}) * (\text{batch job endtime} - \text{batch job starttime})$$

Where we take a weighted percentage of the node resources used and multiply it by the number of hours the resources were unavailable to other users.

Resources are weighted differently depending on the partition/queue; however, the weights are overall configured to be X percentage of a node (as opposed to charging A for a core on the batch partition and B for a core on the GPU partition).

The exact weights can be found in the documentation.

Queue Policies

There are two compute node partitions: “batch” and “gpu”. Each has different queue policies for the number of jobs that can be run and wall times per number of nodes requested.

Batch Partition Policies:

Bin	Node Count	Duration	Policy
A	1 - 8 Nodes	0 - 48 hr	max 4 jobs running and 4 jobs eligible per user in bins A, B, and C
B	9 - 64 Nodes	0 - 36 hr	
C	65 - 128 Nodes	0 - 3 hr	

GPU Partition Policies:

Bin	Node Count	Duration	Policy
A	1 - 2 Nodes	0 - 48 hrs	max 1 job running per user
B	3 - 8 Nodes	0 - 6 hrs	

We want Riker to be a useful OLCF resource and will work with you providing exceptions or even changing the queue structure if necessary.

VisIt

- Due to complex nature of node sharing, there is now a downloadable Host Profile on the docs
 - https://docs.olcf.ornl.gov/software/viz_tools/visit.html#creating-ornl-olcf-host-profiles
- Other than host profile changes: familiar usage, mainly better hardware and rendering support
- NEW: NVIDIA VisRTX Backend now supported
 - <https://github.com/NVIDIA/VisRTX/>
 - [VisIt docs on ANARI Rendering](#)
 - For 3.5.0 build and future builds (not included in our 3.4.2 build)
 - Current known issue: can't be used in remote server/client mode due to a [VisIt Bug](#)
 - must be used via X-forwarding or over VNC
 - Fix coming in VisIt 3.6.0
- **See Riker launch constraints for both client/server or command-line usage here:**
https://docs.olcf.ornl.gov/software/viz_tools/visit.html

VisIt: Host Profile

VisIt won't prompt for these fields at job launch time, but feel free to vary beforehand

Host SettingsLaunch Profiles

☒ batch-shared

☒ gpu-shared

☐ batch-exclusive

☐ gpu-exclusive

New Profile

Delete Profile

Copy Profile

Make Default

SettingsParallelGPU Acceleration

☒ Launch parallel engine

LaunchAdvanced

Load balancingStatic

☐ Use VisIt script to set up parallel environment

☒ Launcher arguments

☐ Sublauncher arguments

☐ Sublauncher pre-mpi command

☐ Sublauncher post-mpi command

--gpus=1 --cpus-per-gpu=16

Total number of GPUs in the job

Range of 16-32
(technically to 48 if
only using 1 GPU)

Host SettingsLaunch Profiles

☒ batch-shared

☐ gpu-shared

☐ batch-exclusive

☐ gpu-exclusive

New Profile

Delete Profile

Copy Profile

Make Default

SettingsParallelGPU Acceleration

☒ Launch parallel engine

LaunchAdvanced

Load balancingStatic

☐ Use VisIt script to set up parallel environment

☒ Launcher arguments

☐ Sublauncher arguments

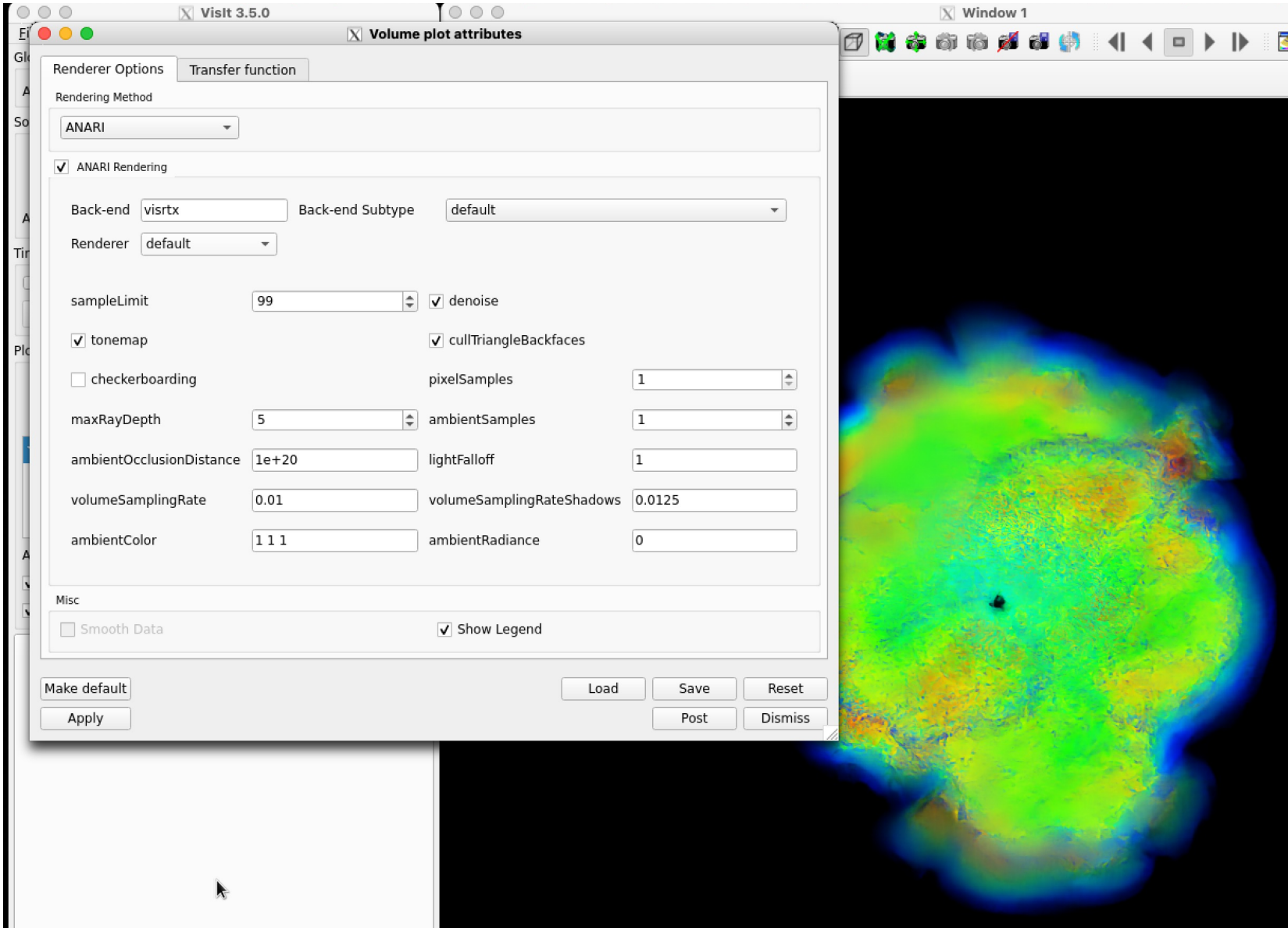
☐ Sublauncher pre-mpi command

☐ Sublauncher post-mpi command

-c 1

Cores per task. VisIt appends `--ntasks` to
its sbatch so enough resources are allocated

VisIt: ANARI VisRTX rendering



ParaView

- Similar to VisIt, host profile is downloadable on the docs: https://docs.olcf.ornl.gov/software/viz_tools/paraview.html#installing-and-setting-up-paraview
 - Some fields are ignored behind the scenes if you're launching an exclusive job or not (see next slide)
 - Cores per task forced to 1 in shared jobs
- [NVIDIA IndeX](#) plugin now included
 - Must disable IceT rendering mode when using > 1 MPI task (see next slide)
- Viskores (formerly VTK-m) auto accelerated filters on GPUs (similar to Frontier)
 - If things hang when using these (e.g., Threshold, Slice, Contour), might be hitting OOM – try reducing MPI ranks
 - ParaView will also *eventually* fallback to non-GPU filters in the job
 - You can turn this feature off by unchecking “Use Accelerated Filters” in the ParaView settings under “General”
- Two modules `paraview/6.1.1-mpi` (intended for batch partition) and `paraview/6.1.1-gpu-mpi` (intended for gpu partition)
 - If need a large number of MPI tasks: use batch partition
 - If interested in GPU acceleration and volume rendering: use gpu partition
 - CPU-only workloads still work with the GPU module/partition by setting total number of GPUs to “0”

ParaView: Host Profile

Connection Options for "ORNL riker (OLCF Official)"

Accelerated Compute: CUDA

Server username: username

Server port: 29229

Number of compute nodes: 1

Total number of MPI tasks: 2

Exclusive node allocation?: True

Number of cores per MPI task (exclusive only): 32

Total number of GPUs (non-exclusive only): 1

Number of cores per GPU (non-exclusive only): 16

Project to charge: ABC123

Number of minutes to reserve: 30

Cancel OK

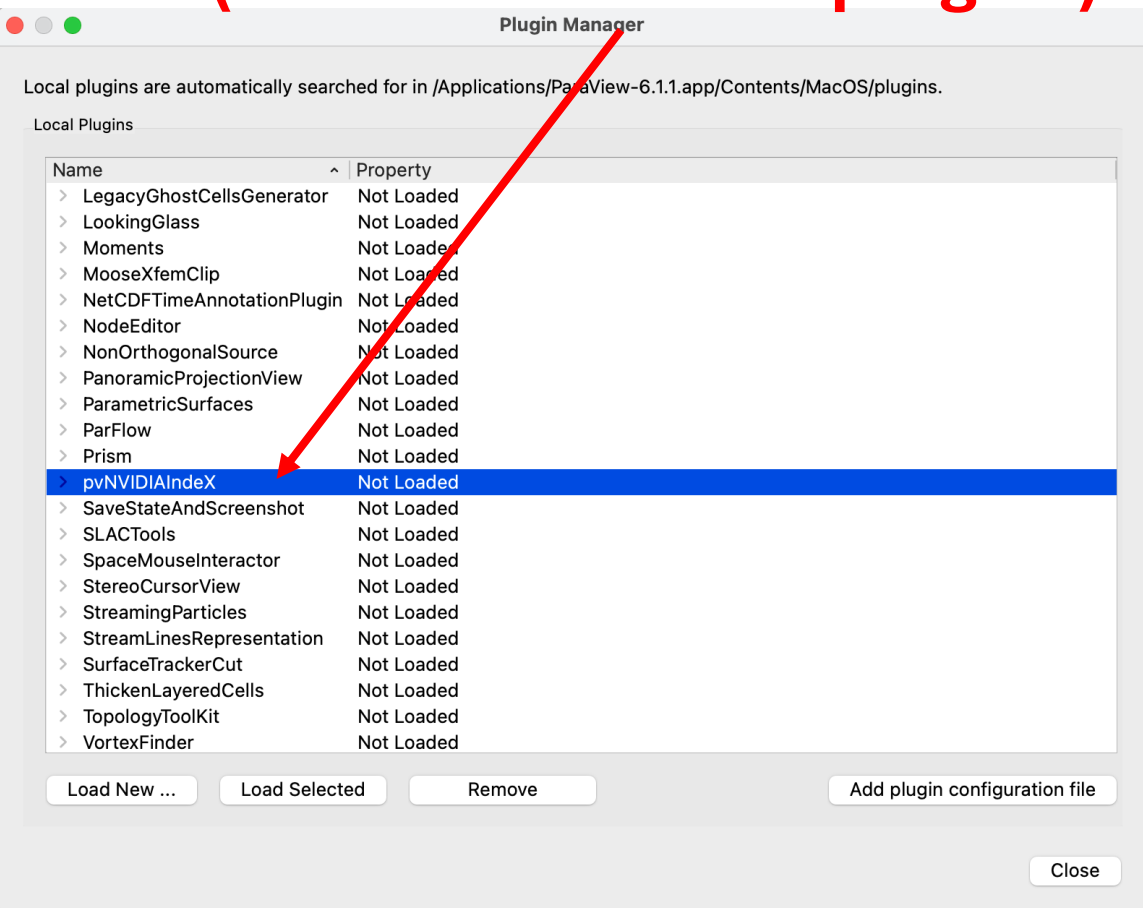
Controls which module is loaded (and which partition is targeted)

Defaults to `-c 1` in non-exclusive jobs

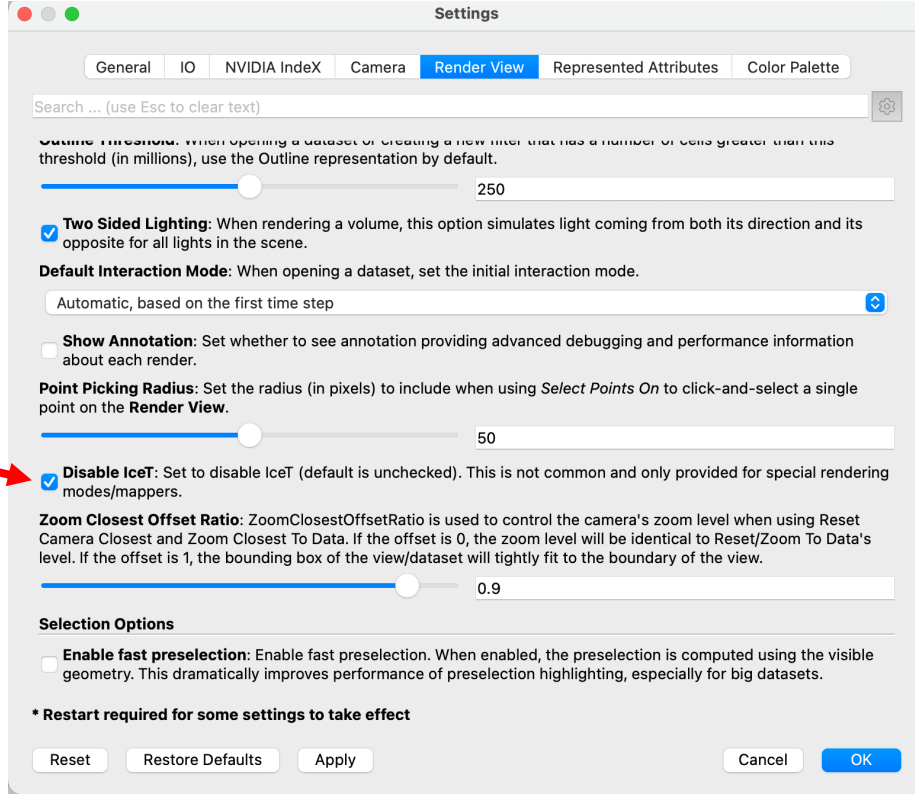
In a "CUDA" non-exclusive job, can be switched to 0 for a CPU only job on the GPU partition

ParaView: Enabling NVIDIA IndexX

1. (load it as a “local plugin”)



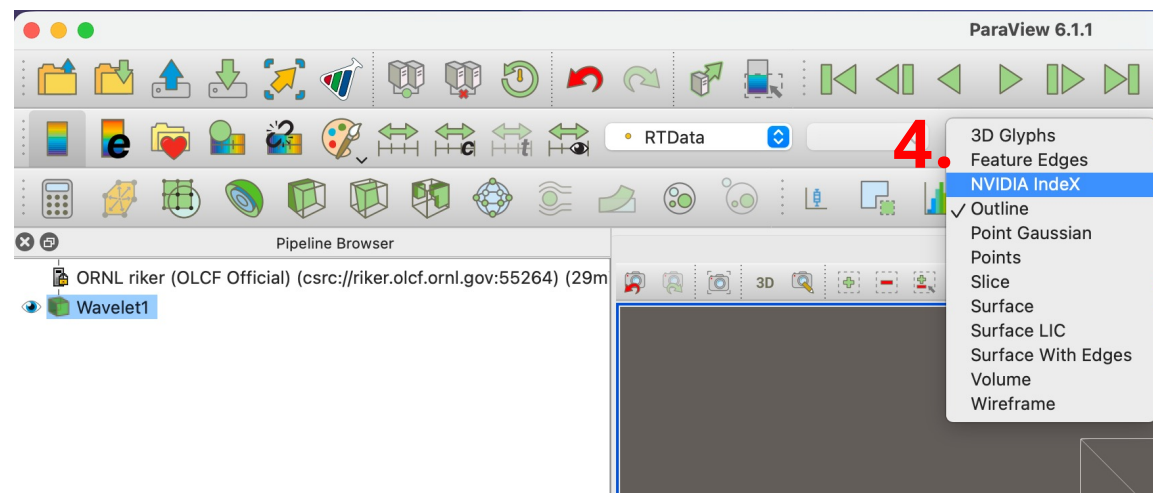
2.



3.



4.



`nvindex: 0.0 PVPLN main warn : IceT compositing must be disabled when using the NVIDIA IndexX plugin with MPI, otherwise nothing will be rendered. Please open 'Edit | Settings', go to the 'Render View' tab, activate 'Advanced Properties' (gear icon) and check 'Disable IceT'. Then restart ParaView for the setting to take effect.`

Looking Down the Road...

- Planned to have Riker added to Job Script Generator: <https://my.olcf.ornl.gov/script-generator>
- DCV (formerly “Nice DCV”) capability
 - We also migrated the remote desktop tools (e.g., VNC, DCV, etc.) to be a separate page: https://docs.olcf.ornl.gov/software/viz_tools/remote_desktop.html
- VMD usage
- Hopefully more viz tutorials in general
- More formal instructions on the new VisIt/ParaView features
- Quantum capability
 - CUDA-Q
- If there’s anything missing from the VisIt/ParaView builds (especially GPU or ray tracing functionality), let us know / please ask

Questions?

Full Documentation can be found here: https://docs.olcf.ornl.gov/systems/riker_user_guide.html

If you have problems or need helping running on Riker, please submit a ticket by emailing help@olcf.ornl.gov