



Lab 8: MPI on Stampede3

Due: 9:00 AM, Wednesday, October 21, 2026

In this lab, you will learn how to run interactive and batch jobs with MPI on the Stampede3 production nodes.

You must work individually on this lab, so everyone learns how to use Stampede3.

Learning goals:

1. To learn how to use Stampede3 to run MPI programs on interactive nodes.
2. To learn how to use Stampede3 to run MPI programs on batch nodes.

Much of what we'll be doing is based on the Stampede3 User Guide.

Getting Set Up

In Canvas, you will find a link to the starter GitHub repository for this lab, and you should fork that repository into your own GitHub account and make sure it is shared with your instructor. Only one member of the group should fork the repository on GitHub, then others should request a link to be granted write access.

You may answer the lab questions right in the `README.md` file of your repository, or use the `README.md` to provide a link to a Google document that has been shared with your instructor or the name of a PDF of your responses that you would upload to your repository.

Using a Stampede3 Login Node

Using the procedure from the earlier lab, log into the Stampede3 at the Texas Advanced Computing Center (TACC).

Question 1: What is the hostname of the node to which you are connected on Stampede3? (Hint: this is the output of the `hostname` command) (1 point)

You can now clone a copy of this lab's repository if you haven't already done so, and your repository from Programming Project 4: Collective Communication [HTML] [PDF] onto Stampede3.

The default compiler and MPI configurations should be sufficient for most or all of our purposes. Compile with `mpicc` and run programs with `mpirun`.

We can compile but not run MPI programs on the login nodes. They are not intended for that purpose. But let's try anyway.

Question 2: Compile the `mpihello.c` program on a Stampede3 login node and try to run it with 2 processes. What is your output? (2 points)

Parallel computations must be run on Stampede3's compute nodes, not the login nodes. These are managed by a queueing system called Slurm, which grants access to subsets of the computer to various users. When you are allocated a set of compute nodes, no other users can run jobs on them. But, it counts against our service allocation, so we only want to do this when you really need to.

You can see current summaries of the jobs in the queueing system with the commands `sinfo` and `squeue`.

You can see the types of nodes available through the queueing system with the command

```
sinfo -o "%18P %8a %16F"
```

This will show a compact summary of the status of the queues. See [here](#) to interpret the output.

Question 3: When you executed the above command, what was the output? How many `skx` nodes are in the Active state? (2 points)

Before moving on, also compile your programs from Programming Project 4: Collective Communication [HTML] [PDF] on your login node.

Using a Compute Node Interactively

A user can and should only log into compute nodes allocated to that user at a given time. We can gain exclusive access to one with the `idev` command.

Run `idev`. After a (hopefully) short time, you should get a command prompt on a compute node.

Question 4: What is the host name of the login node you were allocated? (1 point)

Run the `mpihello` program and the `mpirds-reduce` program with `N=1073741824` on 64 processes on a compute node, redirecting your output to files `mpihello-out64.txt` and `mpirds-out64.txt`, respectively.

Output Capture: `mpihello-out64.txt` for 2 point(s)

Output Capture: `mpirds-out64.txt` for 2 point(s)

Make sure both of these are in your repository for this lab and are committed and pushed to GitHub.

The proper way to run MPI programs on the Stampede3 compute nodes is with a different command: `ibrun`. Run `mpihello` on your compute node.

```
ibrun ./mpihello
```

Question 5: How many processes were created? (1 point)

Question 6: Why did it choose that number? (1 point)

Please log out from the compute node as soon as you are finished with the tasks above. Since you have exclusive access to the node while logged in, the time the node is allocated to you is charged against our class allocation, regardless of whether you are actively using the CPUs.

Submitting Batch Jobs to the Compute Nodes

The most common way to use the compute nodes of this or any supercomputer is by submitting *batch jobs*. The idea is that you get your program ready to run, then submit it to be executed when resources to do so come available.

To set up a batch job, you first need a batch script that can be configured to run your program. A script that will run the `mpihello` program with 32 processes one of Stampede3's "SKX" (Skylake) production nodes is provided in your repository in the file `hellotest.mpi.slurm`. Examine this file and make sure you understand each of the lines that start with `#SBATCH`. These lines define the parameters of your batch submission. You might also use this page to find more information about Slurm batch files.

Modify this file to replace the `--mail-user` option with your own email address.

Submit the job to the queueing system with the command

```
sbatch hellotest.mpi.slurm
```

Question 7: What output do you get at your terminal before you get your prompt back? (1 point)

Question 8: You should have received email when your program began executing, and again when it finished. What are the subject lines of those emails? (2 points)

Question 9: What file contains your program's output? How was it specified? Place this file in your repository for this lab submission (don't forget to add, commit and push it so it's on GitHub). (3 points)

Question 10: According your program's output, what was the host name on which your program executed? (1 point)

Next, let's run with more nodes and processes. Modify the Slurm script to request 4 nodes, and run 48 processes per node (so the `-n` value in your script should be 192).

Question 11: What were the host names allocated to your processes on this run? (1 point)

Question 12: What are the ranks of the processes that were assigned to each node? Hint: the `grep` command might be helpful here. (3 points)

Include the last batch script (3 points) and the last output file (4 points) in your repository.

Now let's do the same for the `mpirds-reduce` program. Create another Slurm script with appropriate values to run `mpirds-reduce` for `N=1536000000` on 8 nodes and a total of 384 processes. Include this batch script and the output file in your repository for this lab (not the previous one where you wrote `mpirds-reduce`). (5 points)

Submission

Commit and push!

Grading

This assignment will be graded out of 35 points.

Feature	Value	Score
Question 1	1	
Question 2	2	
Question 3	2	
Question 4	1	
mpihello-out64.txt	2	
mpirds-out64.txt	2	
Question 5	1	
Question 6	1	
Question 7	1	
Question 8	2	
Question 9	3	
Question 10	1	
Question 11	1	
Question 12	3	
mpihello batch script	3	
mpihello batch output file	4	
mpirds batch script and output file	5	
Total	35	