Introduction to Parallel Processing

Lecture 8 : Debugging MPI Applications

Professor Amanda Bienz

Debugging MPI Programs

- **MPI Programs are difficult to debug!**
- No indication which process error occurs on
- Sometimes, no error until running on 1000s of processes
- Often, error occurs during MPI communication
- Serial methods of debugging don’t always work (e.g. print statements)

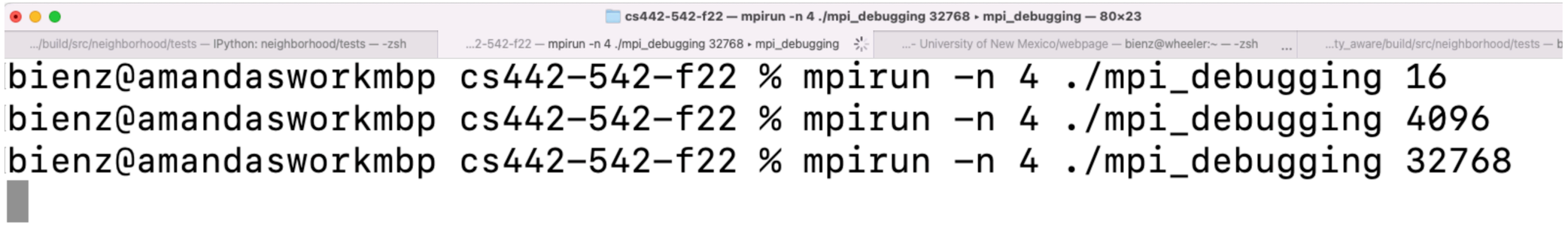
Some Common Errors

- **"Invalid rank" :**
  - Some MPI communication has a rank that is < 0 or >= num_procs

```
bienz@amandasworkmbp cs442-542-f22 % mpicxx -o mpi_debugging mpi_debugging.cpp
bienz@amandasworkmbp cs442-542-f22 % mpirun -n 4 ./mpi_debugging 32768
Abort(737823238) on node 3 (rank 3 in comm 0): Fatal error in internal_Send: Inv
alid rank, error stack:
internal_Send(120): MPI_Send(buf=0x7f8775784000, count=32768, MPI_DOUBLE, 4, 0,
MPI_COMM_WORLD) failed
internal_Send(78).: Invalid rank has value 4 but must be nonnegative and less th
an 4
bienz@amandasworkmbp cs442-542-f22 %
```

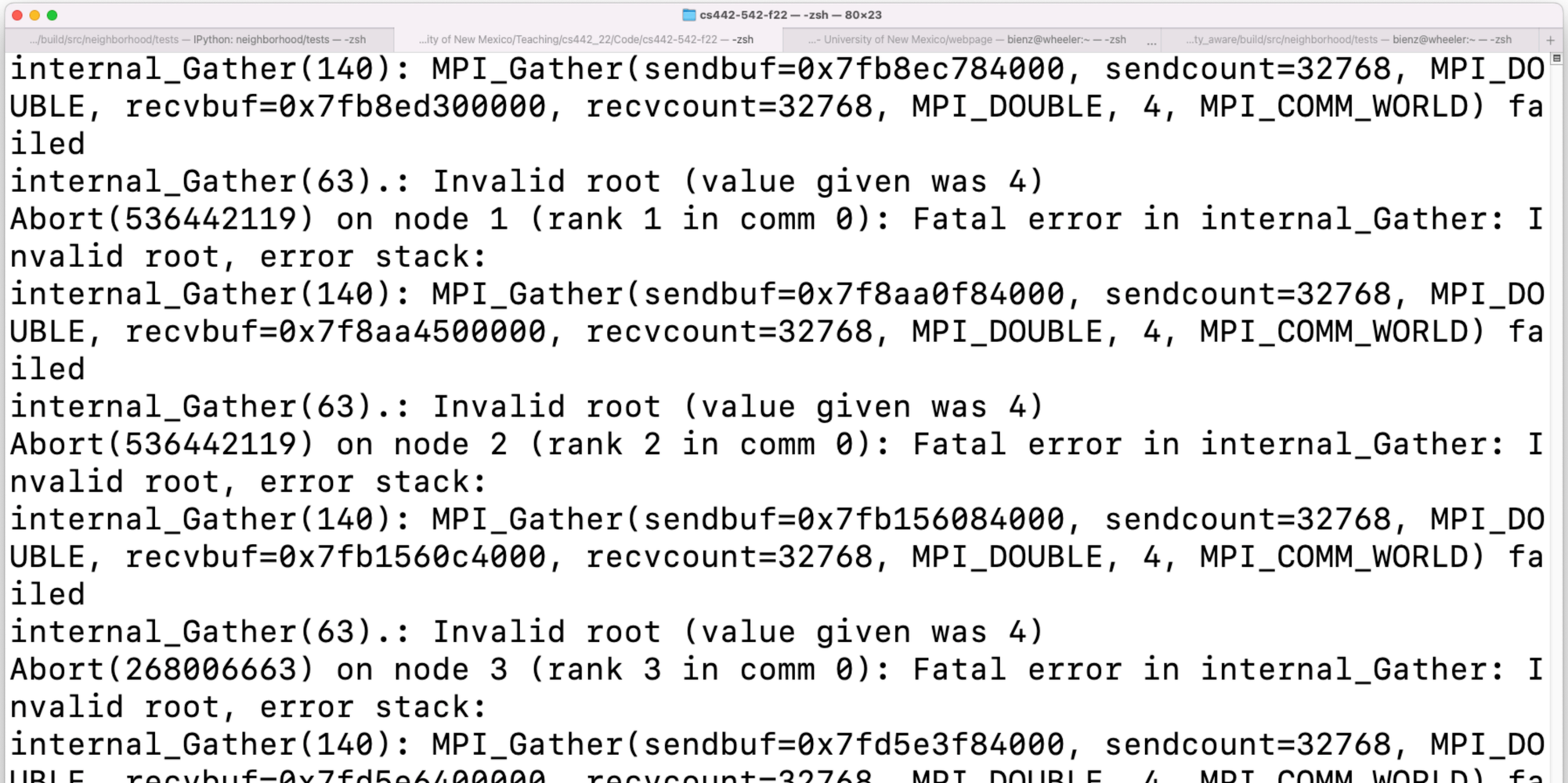
Some Common Errors

- **If your program hangs indefinitely:**
  - Almost always error in communication
  - One way to debug : print out info about communication, may sometimes need to comment out communication
  - If only for large message sizes, likely MPI_Send with no matching MPI_Recv



Some Common Errors

- **"Invalid root": root < 0 or >= num_procs**



MPI Error Handling

- You can add custom error handling
- MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN)
  - Default error flag : MPI_ERRORS_ARE_FATAL
  - MPI_ERRORS_RETURN : error codes are returned and program no longer crashes with error
- Can print out and handle errors on your own
  - MPI_Error_class(error_code, &error_class)
- Error codes : https://www.mpi-forum.org/docs/mpi-2.2/mpi22-report/node192.htm

Some Common Errors

- **Segfaults : accessed unallocated memory**

cs442-542-f22 — -zsh — 80×23

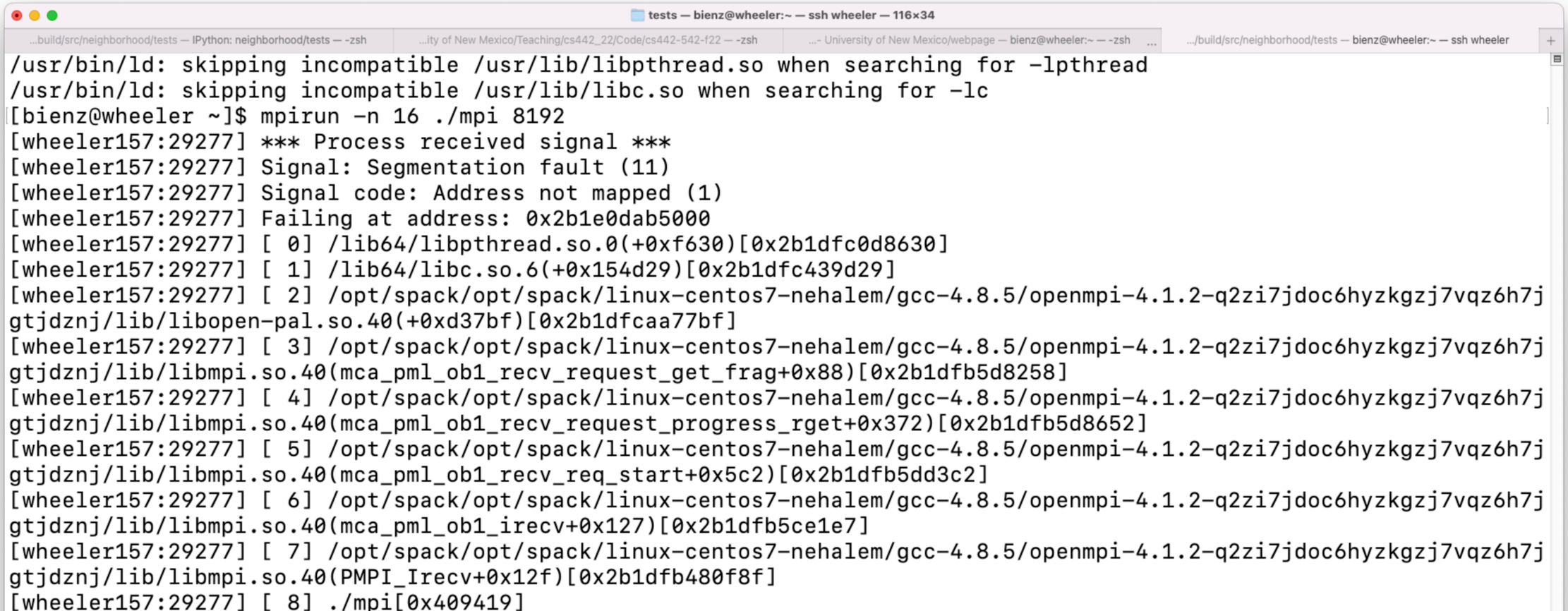
.../build/src/neighborhood/tests — IPython: neighborhood/tests — -zsh | ...ity of New Mexico/Teaching/cs442_22/Code/cs442-542-f22 — -zsh | ...- University of New Mexico/webpage — bienz@wheeler:~ — -zsh | ...ty_aware/build/src/neighborhood/tests — bienz@wheeler:~ — -zsh

```
bienz@amandasworkmbp cs442-542-f22 % mpicxx -o mpi_debugging mpi_debugging.cpp
bienz@amandasworkmbp cs442-542-f22 % mpirun -n 4 ./mpi_debugging 32768

===================================================================================
=   BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES
=   PID 31771 RUNNING AT amandasworkmbp.unm.edu
=   EXIT CODE: 11
=   CLEANING UP REMAINING PROCESSES
=   YOU CAN IGNORE THE BELOW CLEANUP MESSAGES
===================================================================================
YOUR APPLICATION TERMINATED WITH THE EXIT STRING: Segmentation fault: 11 (signal
 11)
This typically refers to a problem with your application.
```

Some Common Errors

- **Segfaults : accessed unallocated memory**
  - Memory issue can occur within MPI
  - Example : changing send_buffer in MPI_Isend



Logical Bugs

- Probably the hardest to figure out
- Program doesn’t crash, but you get an incorrect result
- Some ways to solve :
  - Run a bunch of unit tests
  - Check solution along the way
  - Check values before and after communication

Memory Leaks

- Valgrind!
  - This works on Wheeler
- On Mac : leaks (type ‘man leaks’ to learn more)

Some Debugging Methods

- Sophisticated tools, like Totalview
- Definitely use this if available!
  - Very expensive, not on most machines

Some Debugging Methods

- GDB : The GNU Project Debugger
  - https://www.sourceware.org/gdb/
- Attach GDB to individual processes after they are running
- Use mpirun to launch separate instances of serial debuggers
  - mpirun -n 4 xterm -e gdb <program_to_debug>
  - Great for local machines, usually not a great option on supercomputers

Some Debugging Methods

- Commenting out code
  - Print statements aren’t always as useful in parallel
  - If you comment out code, you know can find where the error is
- Printing information