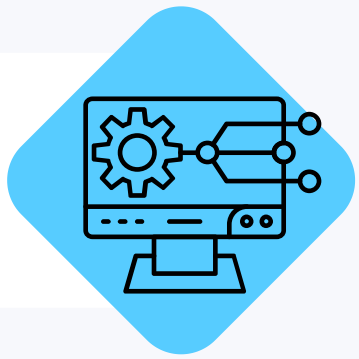
DEBUGGING PARALLEL PROGRAMS IN HPC

Debugging and Troubleshooting in MPI and OpenMP

Race Condition: Two threads/process write to same variable

Fix Strategy: Use #pragma omp critical or atomic operations



Deadlock: Process wait indefinitely for each other

Fix Strategy: Check MPI send/recv order; add timeouts

Load Imbalance: Uneven workload distribution

Fix Strategy: Use dynamic scheduling or load balancing

False Sharing: Threads share same cache line

Fix Strategy: Add padding or align memory

Memory Leaks: Unreleased allocated memory

Fix Strategy: Monitor outcomes and make adjustments when necessary.

Debugging and Troubleshooting in MPI and OpenMP

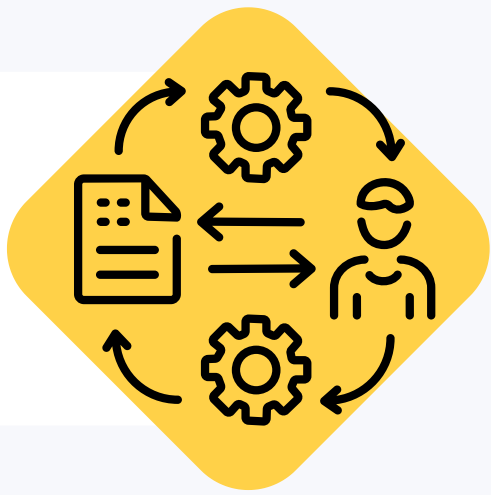
Debugging Tools

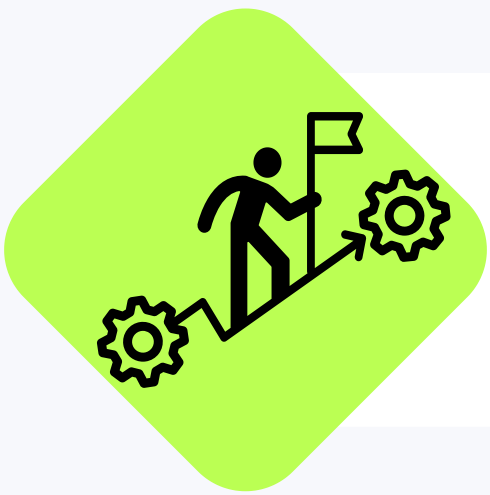
- gdb - Step Through program execution.
- valgrind - Detect memory leaks or invalid accesses.
- ThreadSanitizer - Find race conditions in OpenMP.
- mpiP - Profile and debug MPI communication.
- TotalView/ Arm DDT - Visual Debuggers for HPC cluster.

Debugging Workflow

1. Reproduce the Error: Run with smaller input.
2. Isolate the Section: Use logging or conditional compilation (**#ifdef DEBUG**).
3. Check for Thread Safety: Identify shared variables.
4. Run Under Debugger:

```
mpirun -np 4 xterm -e gdb ./app
```





Best Practices

- Add varbose logs (printf or MPI_Comm_rank to identify
- Always check return value of MPI calls
- Use small datasets for testing
- Compile with debug symbols: -g -00
- After debugging, recompile with optimizations

“DEBUGGING IS TWICE AS HARD AS WRITING THE CODE. SO IF YOU WRITE THE CODE AS CLEVERLY AS POSSIBLE, YOU ARE, BY DEFINITION, NOT SMART ENOUGH TO DEBUG IT.”

— BRIAN KERNIGHAN