Parallel Debugging

Jonathan Schmalfuß

Chair of Scientific Computing
University of Bayreuth

March 19, 2025

Parallel debugging

> “Sequential programming is really hard, and parallel programming is a step beyond that.” - Andrew S. Tanenbaum, professor at Vrije Universiteit Amsterdam

> “Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.” - Brian Kernighan, professor at Princeton University.

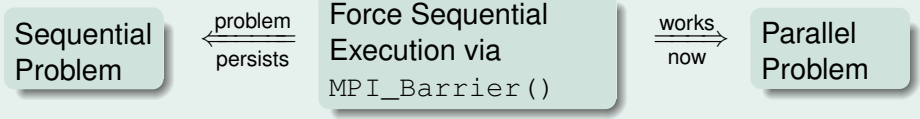
Techniques in general

> “Most bugs also appear in the sequential version of the code” - me

Techniques in general

"Most bugs also appear in the sequential version of the code" - me

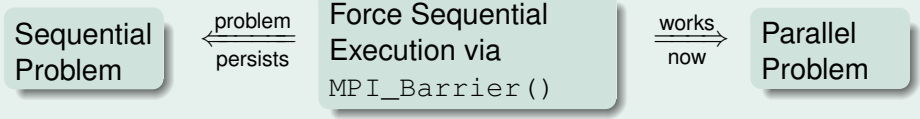
Distinguish Problem



Techniques in general

"Most bugs also appear in the sequential version of the code" - me

Distinguish Problem



The experts opinion: Anthony Williams - author/ coauthorof the thread library in C++

1. Reviewing code to locate potential bugs
2. Locating concurrency-related bugs by testing / Designing for testability

Questions while reviewing multiprocess code

- Are there any ordering requirements between the operations done in this process and those done in another? How are those requirements enforced?
- Which data needs to be protected from concurrent access? How do you ensure that the data is protected?
- Where in the code could other processes be at this time?
- Is the data loaded by this process still valid?
- If you assume that another process could be modifying the data, what would that mean and how could you ensure that this never happens?

Parallel Debugging: How to?

the easy way

- use a debugger and or fronted made for debugging MPI code
- Industry Standard: ddt[1], TotalView[1]
- Open Source Projects / Free: mdb, oneAPI with mpigdb or a shell script tmpi

should-always-work way

- mininmal requirements: a debugger (gdb) + a way of finding the running processes (ps/top)
- `mpirun` creates multiple processes –> attach to relevant processes or all –> debugg each of them sequentially
- limits: low number of processes, requires duplicating input for each process

[1] temporary free / student license available

Debug Deadlocks: Attach to the process [Live-session]

Situation: you have a deadlock, i.e. your executable is stuck

1. Compile with debug flags: `mpic++ -g -Wall <file> -o <name>`
2. Wait until stuck
3. Figure out process id's via `top` or `ps -a | grep <name>`
4. Attach to the process and see where you are stuck –> figure out what the problem is

MPI debugging

Debug Deadlocks: Attach to the process [Live-session]

Situation: you have a deadlock, i.e. your executable is stuck

1. Compile with debug flags: `mpic++ -g -Wall <file> -o <name>`
2. Wait until stuck
3. Figure out process id's via `top` or `ps -a | grep <name>`
4. Attach to the process and see where you are stuck –> figure out what the problem is

Attaching debugger to serval instances of the executable

- Use mpirun to launch separate instances of serial debuggers
- Drawback: many process, usually problematic

OpenMPI: FAQ: Debugging applications in parallel

MPI debugging: Memchecker

- Requires: Open MPI 1.3 or later, and Valgrind 3.2.0 or later
- Otherwise: works, but with many false positives
- Needs to be enable at compilation state of OpenMPI, unfortunately often is not
- to enable locally, see How can I use Memchecker
- ```
  mpirun -np 2 valgrind
  ↪  --suppressions=$PREFIX/share/openmpi/openmpi-valgrind.supp
  ↪  <executable name>
  ```
  ```

The easy way

ddt [Live-session]

- same deadlock problem

Exercises

Brought you some programs, which you can check out. Find the errors.

1. Is it a sequential problem or parallel for `mpi_reusing_a_buffer.cpp`