

Debugging HPC applications with mdb

Tom Meltzer

Introduction

\$> whoami



- Principal RSE @ ICCS
- PhD in Theoretical Phys
- HPC enthusiast
- Professional Debugger

\$> whoami



- Principal RSE @ ICCS
- PhD in Theoretical Phys
- HPC enthusiast
- Professional Debugger

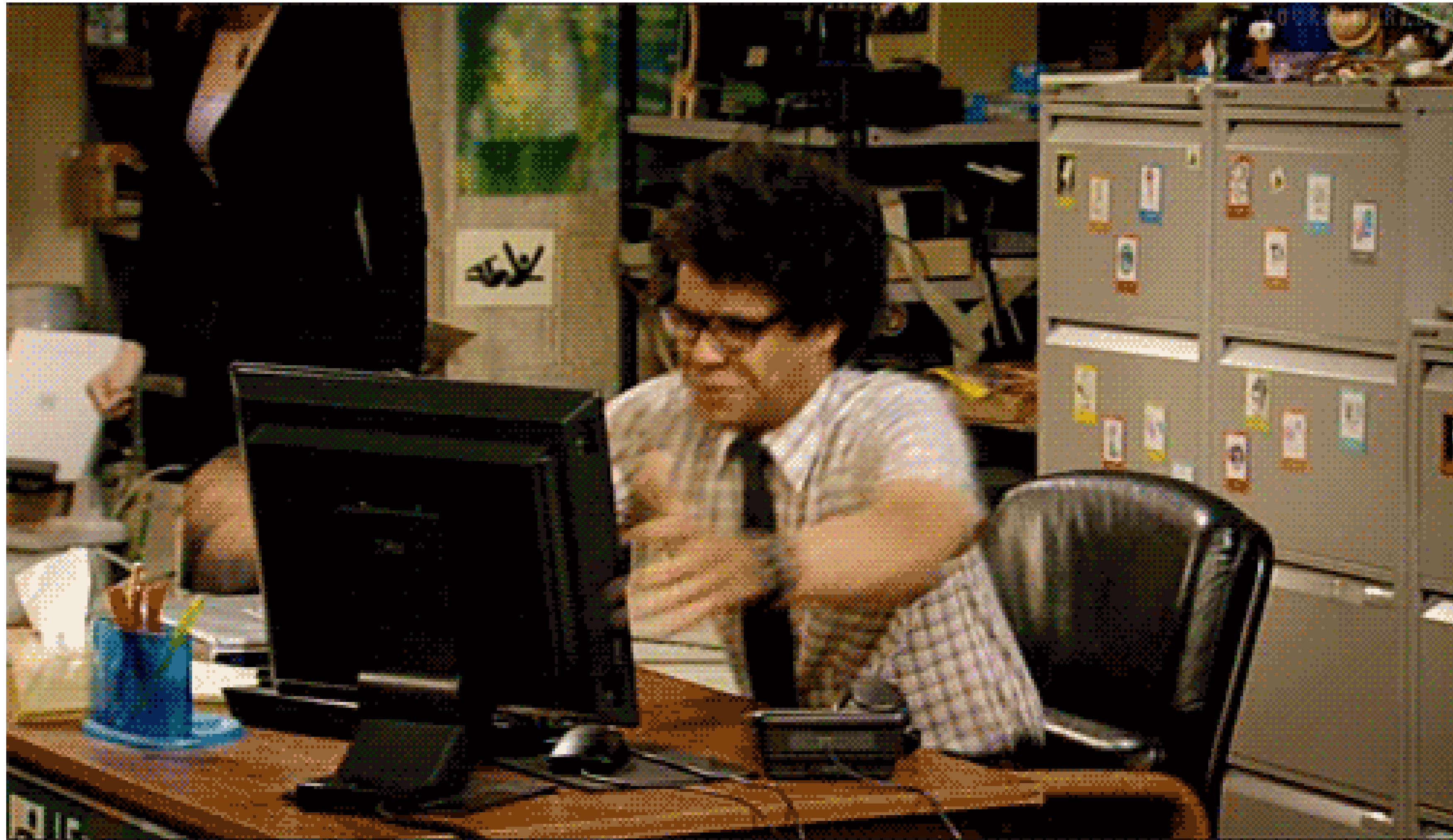
⚠ Not a Rustacean 🦀

Debugging 101

What are my options?

1. Write bug-free code...
2. Use `printf()`
3. Ignore it 🙄 (Ostrich Method)
4. Use a debugger (e.g., `gdb`, `lldb`, ...)

Write bug-free code



source: giphy.com/gifs/M11UVCRrc0LUk

Use `printf()`

Use `printf()`

- Valid if code cannot run without optimization

Use `printf()`

- Valid if code cannot run without optimization
- Requires recompiling code

Use `printf()`

- Valid if code cannot run without optimization
- Requires recompiling code
- Could introduce new bugs

Use `printf()`

- Valid if code cannot run without optimization
- Requires recompiling code
- Could introduce new bugs
- Want to see a new variable?

Use `printf()`

- Valid if code cannot run without optimization
- Requires recompiling code
- Could introduce new bugs
- Want to see a new variable?
- Recompile again 😓

Use `printf()`

- Valid if code cannot run without optimization
- Requires recompiling code
- Could introduce new bugs
- Want to see a new variable?
- Recompile again 😓
- Cannot interactively alter runtime

Ignore it (Ostrich Method)

- Valid if cost outweighs solution

Use a debugger

Use a debugger

- Interact at runtime – *change the flow of execution*

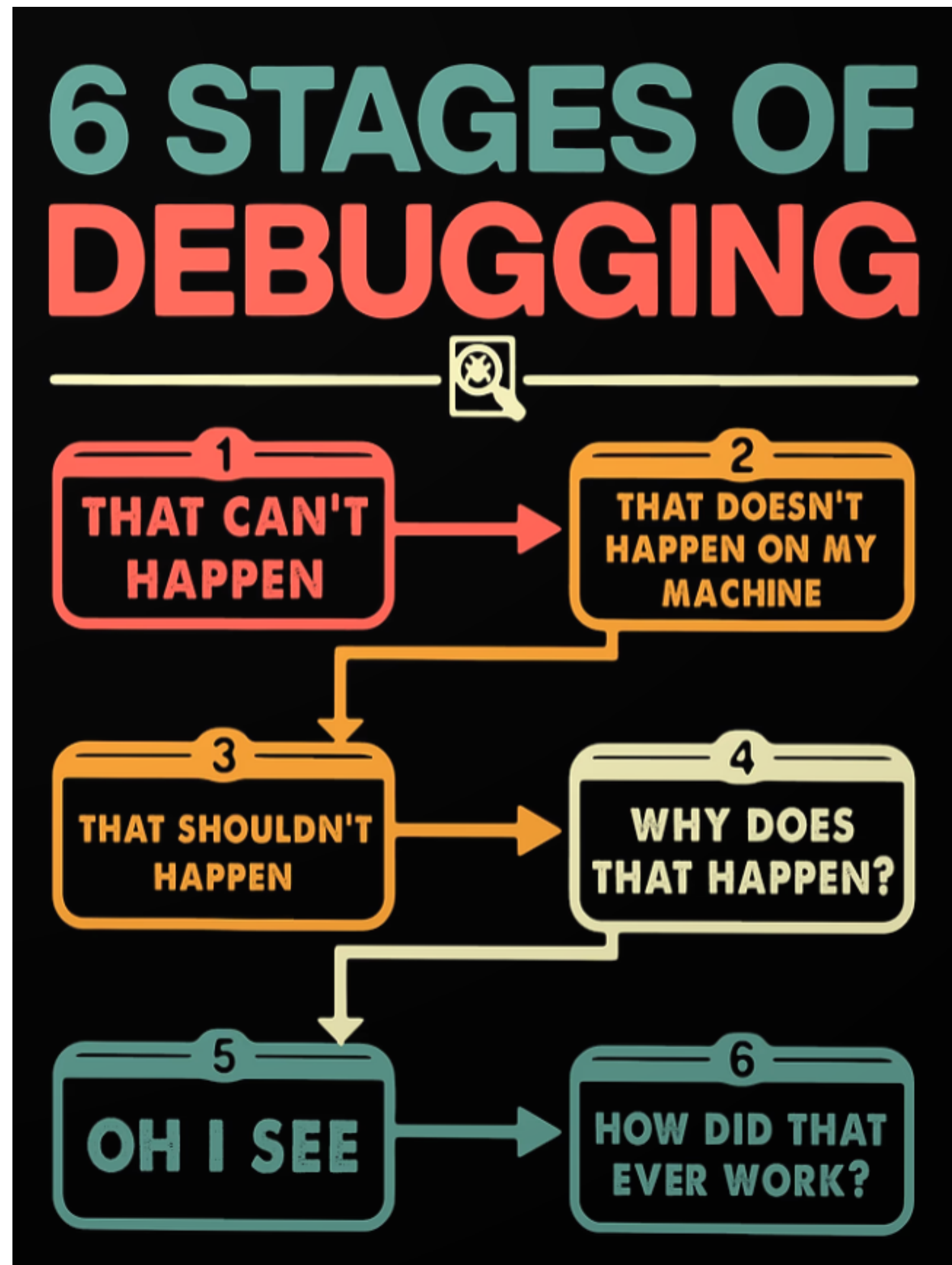
Use a debugger

- Interact at runtime – *change the flow of execution*
- Inspect everything – *private variables... who cares!*

Use a debugger

- Interact at runtime – *change the flow of execution*
- Inspect everything – *private variables... who cares!*
- Use for code exploration – *what actually happens*

What is a debugger?



source: displate.com/displate/4420099

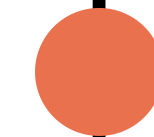
A **program** used to test another “**target**” program. The target program is run in a controlled environment that allows the user to inspect resources, interrupt program flow and even modify code execution.

Serial Debugging Landscape

- `gdb`, `lldb` (and derivatives e.g., `rust-gdb`)
- IDE-integrated (vscode, eclipse, jetbrains, vim, emacs etc.)
- GUI-wrappers: (gdbgui, seer, etc.)

MPI Debugging Landscape

Features &
Support



*mdb**



Intel gdb



mpidb / tmpi / mpigdb



*Linaro ddt
TotalView*

Money

** my possibly biased opinion* 🙄

Key Features of **mdb**

- **Terminal-based** – *debug local to code*
- Supports multiple backends (**rocgdb**, **rust-gdb**)
- Uses Python's **asyncio** library
- Holistic commands (**plot**, **dump**)
- Scriptable & Extensible – *unicode plots*
- **Ctrl-C** is handled correctly (👁👁 Intel gdb)
- Free and Open-Source – **pip/spack/uv**-installable

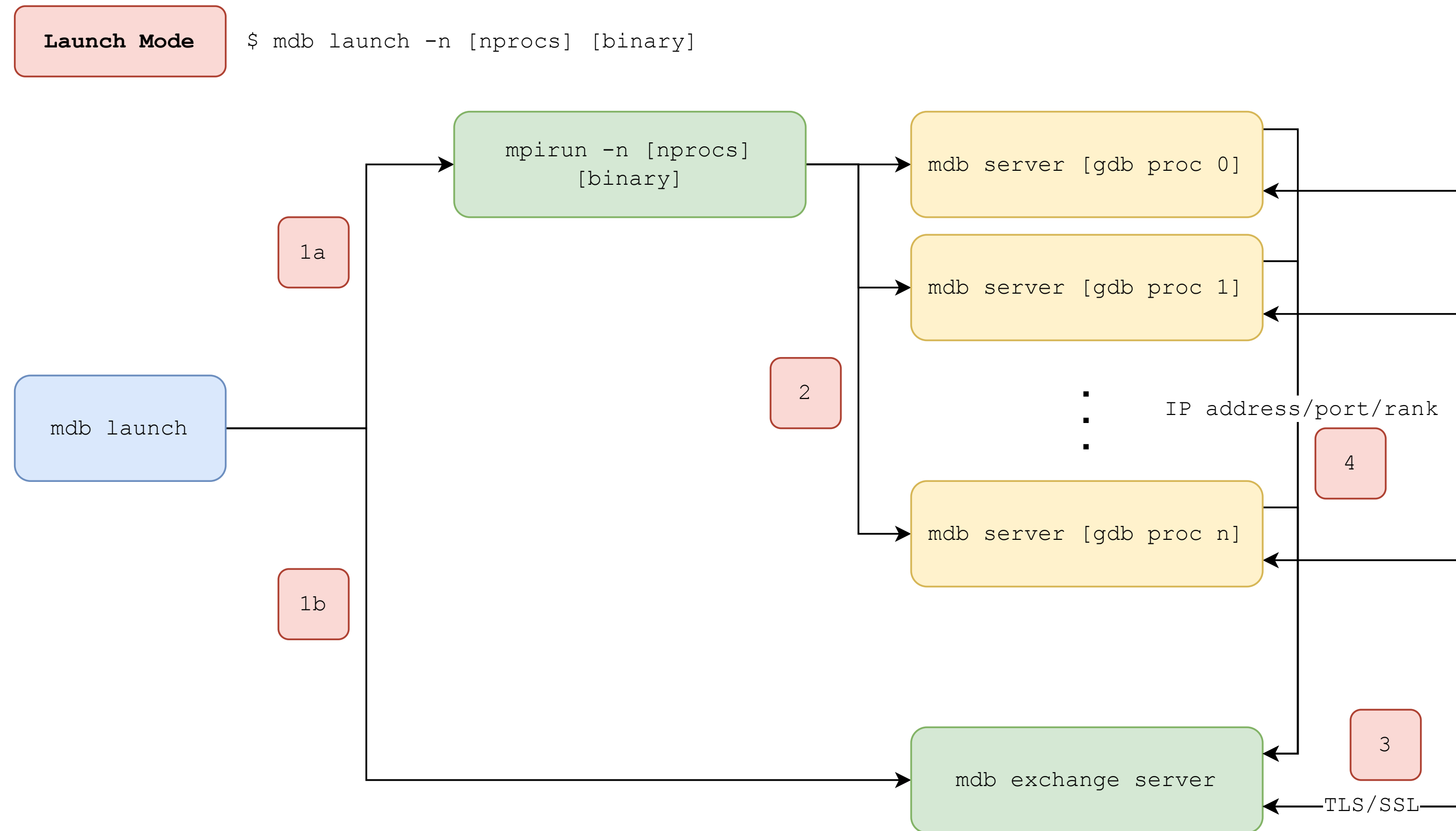
Why terminal?



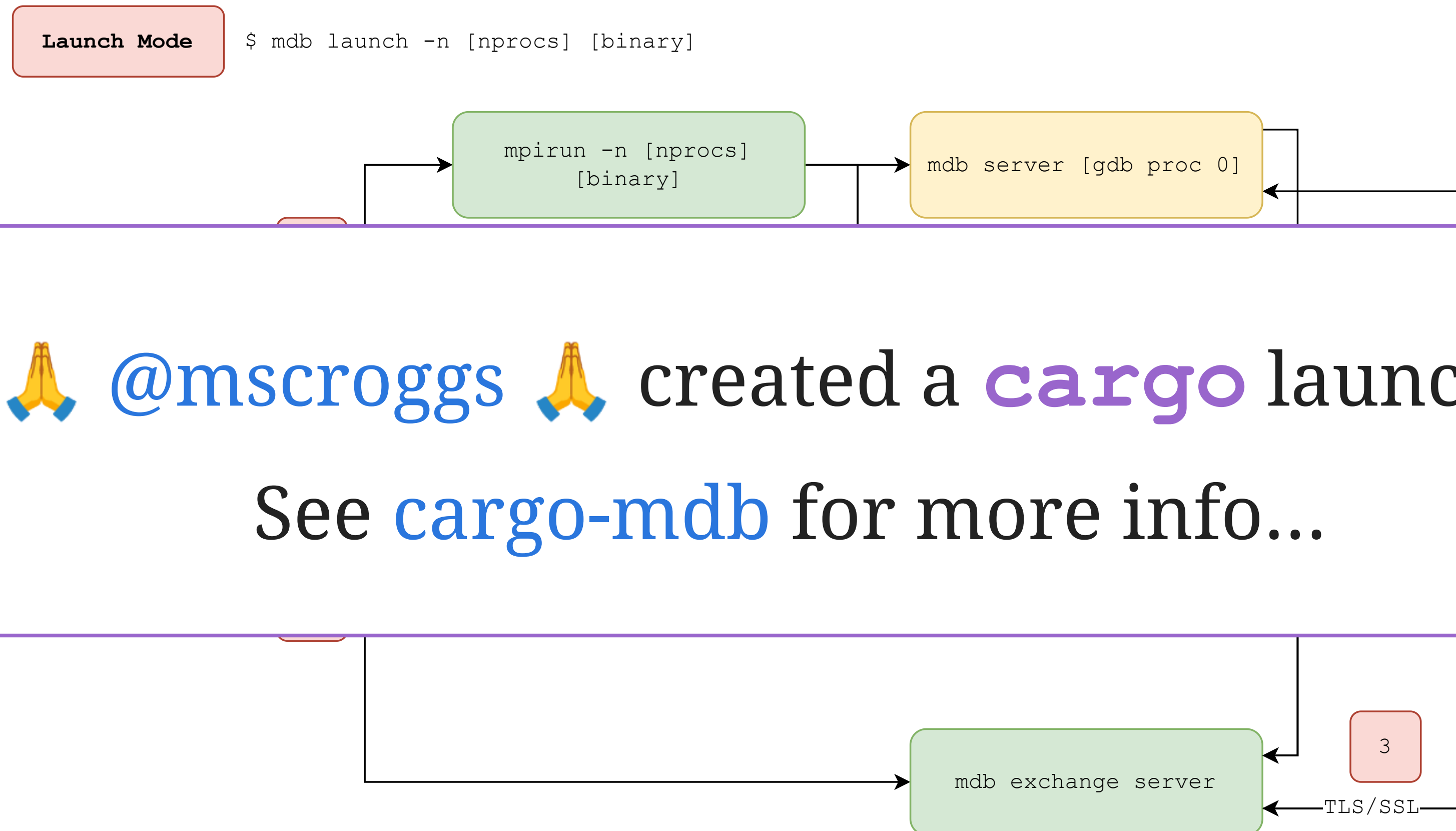
The
Terminal

- Unix philosophy
- Available
- Extensible
- Scriptable

mdb launch



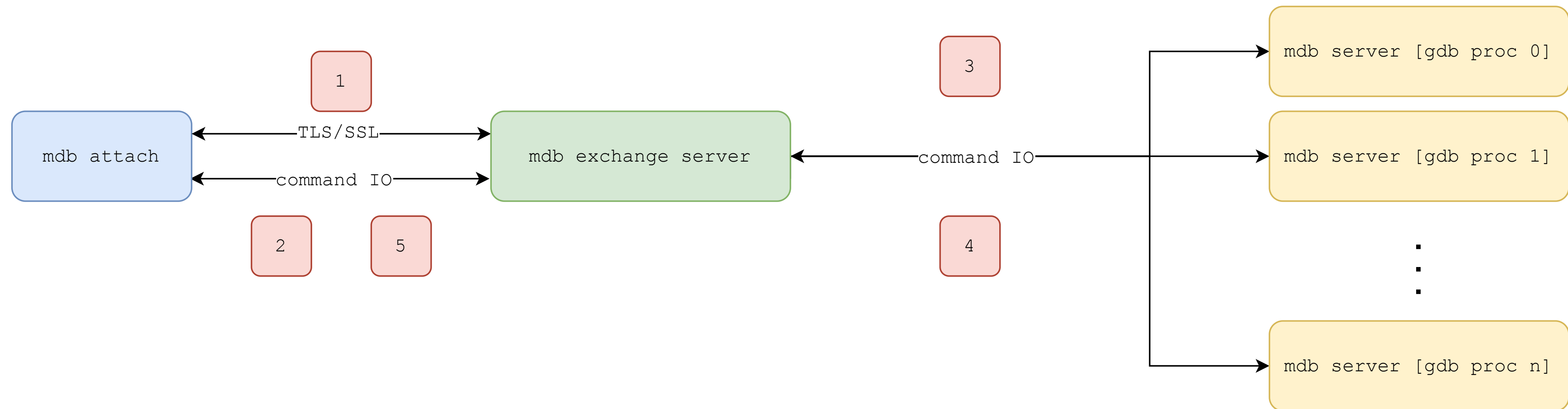
mdb launch



mdb attach

Attach Mode

```
$ mdb attach --host [hostname]
```



Demo time



github.com/TomMelt/mdb

- where to get help (**man**/rtd/github/?)
- **broadcast**
- **combine**-ing output
- scripting **mdb**
- **gdb** plugins

@**1kleonk**'s **rust-mpi-particle-simulator**

Thanks

- @mscroggs - Matthew Scroggs - Invitation
- @1k1eonk - Leon Koch - Rust Demo
- All mdb contributors, especially...
 - @fjebaker - Fergus Baker
 - @joewallwork - Joe Wallwork

Thanks for Listening