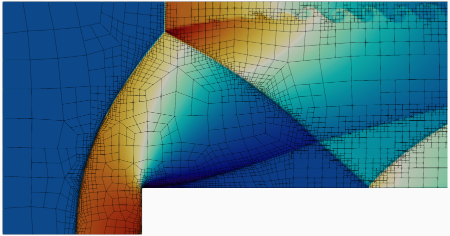
Running Julia code in parallel with MPI

Lessons learned

Lars Christmann[*†], Niklas Neher[*],
Michael Schlottke-Lakemper[*]

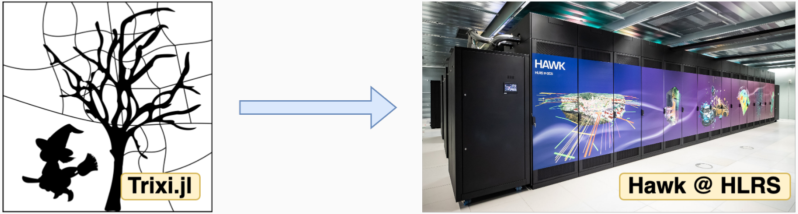
JuliaCon 2022, 26th July 2022

[*] High-Performance Computing Center Stuttgart (HLRS),
University of Stuttgart, Germany
[†] University of Cologne, Germany



Does it scale?

- Julia code for computational fluid dynamics: ✓ (Trixi.jl)
- Parallelized with MPI: ✓
- Supercomputer available: ✓ (Hawk @ HLRS)



©Trixi.jl authors, Ben Derzian, HLRS

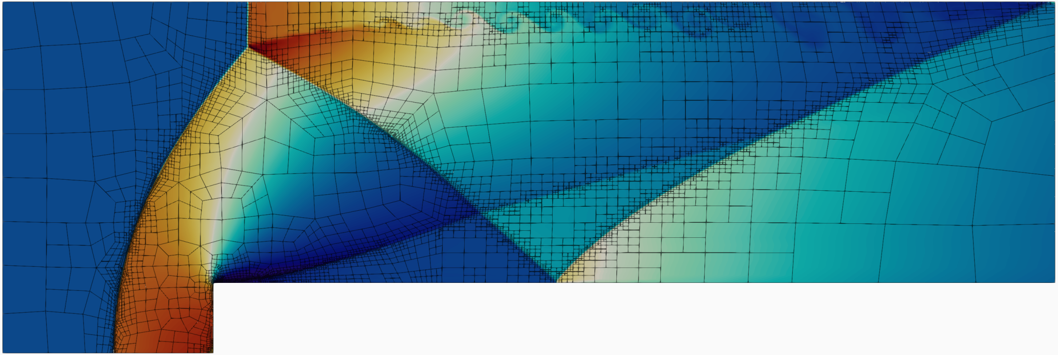
What is Trixi.jl?

Trixi.jl

- Adaptive numerical simulation framework
- Focus on high-order methods for computational fluid dynamics
- Hybrid parallelization with MPI and multithreading (Polyester.jl)

https://github.com/trixi-framework/Trixi.jl

Trixi.jl simulation of front-facing step in wind tunnel at Mach 3



Credit: Andrew R. Winters, https://www.youtube.com/watch?v=glAug1aIxio

Systems at HLRS

Hawk and Vulcan

- High-Performance Computing Center Stuttgart (HLRS)
- Hawk → capability computing (AMD EPYC Rome, ~720k cores)
- Vulcan → industry computing (various Intel Xeon, 10k-20k cores)
- Julia 1.7.2 available as modules

https://www.hlrs.de/solutions/systems

HLRS Hawk

HLRS Vulcan

Where to put the Julia folder

- `JULIA_DEPOT_PATH` envvar specifies location for package files, precompilation files...
- Put it in home directory (NFS)?
  - Julia crashed when using >2000 MPI ranks
- Put it on parallel file system (Lustre)?
  - >10,000 small files
  - Issues: slow on `Pkg` update, quota limits

Lessons learned

Figure out depot location before doing parallel runs.

Observations for MPI and multithreading

- Generally high memory usage of Julia runtime: Trixi.jl + OrdinaryDiffEq.jl
  - 360 MiB after loading
  - 510 MiB after compilation
- Hybrid parallelization (MPI + multithreading) support may be system dependent
- Non-standard MPI implementations might require additional efforts

Lessons learned

Test which MPI implementation to use.
Verify that multithreading works as expected.

Using Julia with a system MPI may require user action

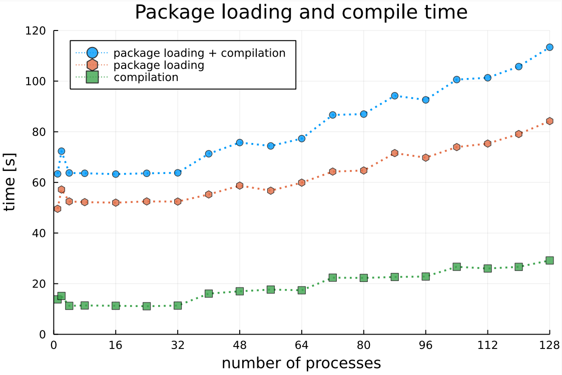
- Need to replace MPI-enabled JLL-provided binaries by system binaries (e.g., MPI.jl, HDF5.jl)
- Need to regenerate C bindings for libraries due to MPI ABI change (e.g., P4est.jl)

Lessons learned

Document setup procedures for your users and yourself.
Have a look at MPItrampoline (`https://github.com/eschnett/MPItrampoline`).

Runtime overhead due to code loading/compilation

- Code loading times increase due to file system pressure
- Also code compilation can be slower in parallel
- General rule: non-parallel operations increase computational cost

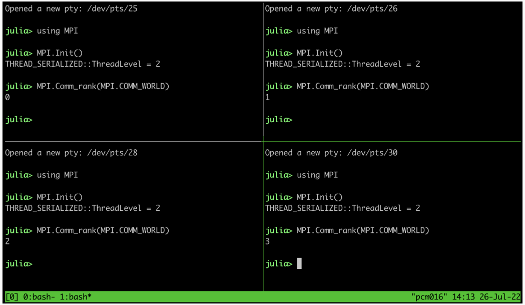


Lessons learned

Use PackageCompiler.jl to create custom Julia sysimage (`https://github.com/JuliaLang/PackageCompiler.jl`).
Try to use HPC/Spindle to avoid I/O bottleneck (`https://github.com/hpc/Spindle`).

Restrictions for interactive computing with MPI

- No inherent MPI support for the REPL
- Long load/compilation times
- Parallel development is sub-optimal (debugging is even less fun)



```
Opened a new pty: /dev/pts/25

julia> using MPI

julia> MPI.Init()
THREAD_SERIALIZED::ThreadLevel = 2

julia> MPI.Comm_rank(MPI.COMM_WORLD)
0

julia>
```

```
Opened a new pty: /dev/pts/26

julia> using MPI

julia> MPI.Init()
THREAD_SERIALIZED::ThreadLevel = 2

julia> MPI.Comm_rank(MPI.COMM_WORLD)
1

julia>
```

```
Opened a new pty: /dev/pts/28

julia> using MPI

julia> MPI.Init()
THREAD_SERIALIZED::ThreadLevel = 2

julia> MPI.Comm_rank(MPI.COMM_WORLD)
2

julia>
```

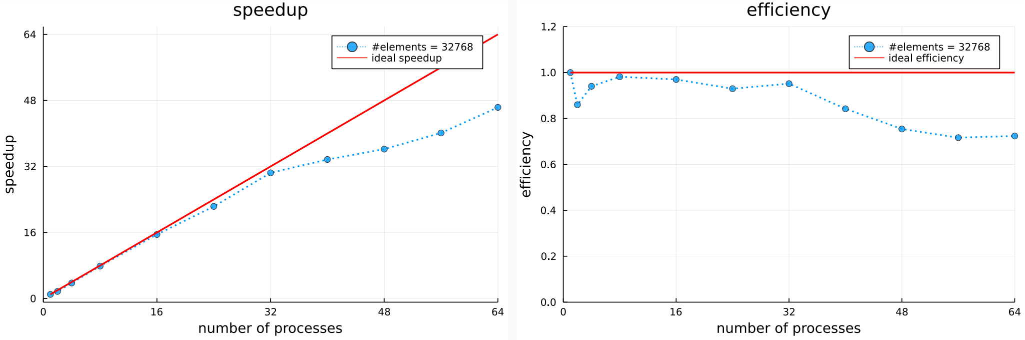
```
Opened a new pty: /dev/pts/30

julia> using MPI

julia> MPI.Init()
THREAD_SERIALIZED::ThreadLevel = 2

julia> MPI.Comm_rank(MPI.COMM_WORLD)
3

julia> ▌
```

```
[0] 0:bash- 1:bash*                                      "pcm016" 14:13 26-Jul-22
```

Lessons learned

Run MPI in `tmux` via `tmpi` as a workaround (`https://github.com/Azrael3000/tmpi`). Be patient.

Strong scaling results for Trixi.jl



Good scalability from 1 up to 32 MPI ranks, acceptable scalability until 64 MPI ranks

Lessons learned

Parallel performance of Julia with MPI behaves similarly to C++/Fortran codes.

Conclusions

- Running Julia code in parallel with MPI works!
- Additional challenges compared to traditional HPC languages
- Parallel development toolchain not as mature yet
- Is Julia with MPI the right tool for your HPC needs: it depends

Lessons learned

Interact with the great Julia for HPC community!

Thank you for your interest!