CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Rust programming language in high-performance computing

Final Presentation

Michal Sudwoj

25.08.2020

Rust programming language in high-performance computing
Final Presentation
Michal Sudwoj
25.08.2020

Introduction

- Fortran, C++ established languages for HPC
- Rust offers safety (borrow checker)
- Does it have (the features) that it needs?
  - see *Introduction to Rust* presentation
- **Is it fast enough?**

Introduction
- Fortran, C++ established languages for HPC
- Rust offers safety (borrow checker)
- Does it have (the features) that it needs?
  - see *Introduction to Rust* presentation
- **Is it fast enough?**

Problem

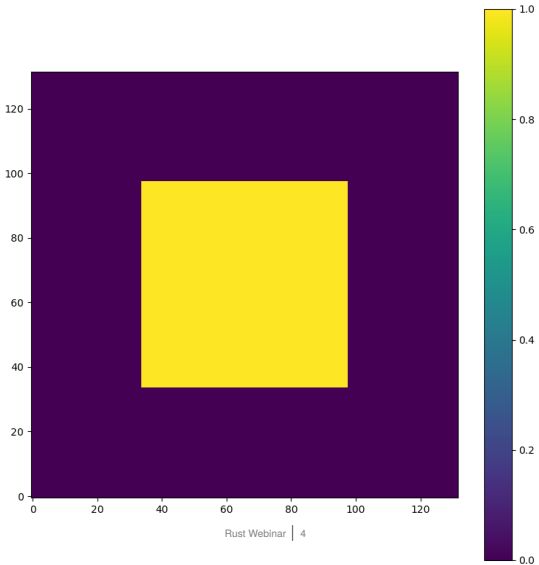
- fourth-order numerical diffusion in xy-plane

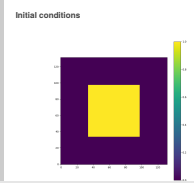
$$\frac{\partial \phi}{\partial t} = -\alpha_4 \nabla^4_{xy} \phi = \underbrace{-\alpha_4 \Delta^2_{xy} \phi}_{\texttt{inline}} = \underbrace{-\alpha_4 \Delta_{xy} (\Delta_{xy} \phi)}_{\texttt{laplap}}$$

- on unit cube
- boundary conditions: $\partial_\Omega = 0$

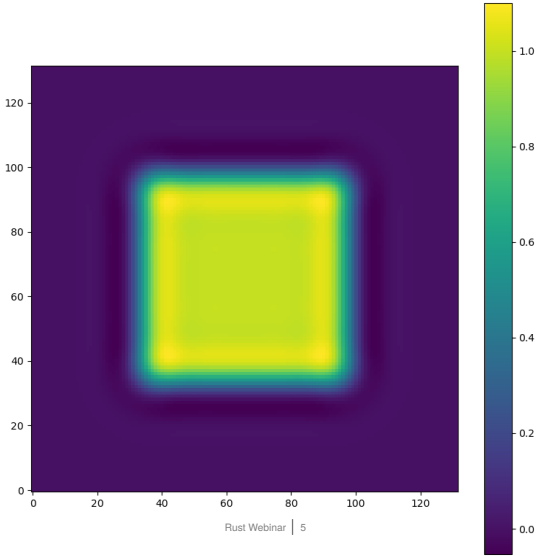
Problem
- fourth-order numerical diffusion in xy-plane
$\frac{\partial \phi}{\partial t} = -\alpha_4 \nabla^4_{xy} \phi = \underbrace{-\alpha_4 \Delta^2_{xy} \phi}_{\texttt{inline}} = \underbrace{-\alpha_4 \Delta_{xy} (\Delta_{xy} \phi)}_{\texttt{laplap}}$
- on unit cube
- boundary conditions: $\partial_\Omega = 0$

Initial conditions

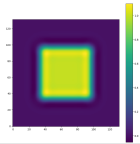




Result



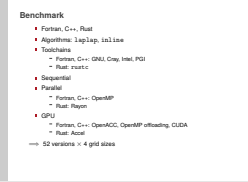
Result



Benchmark

- Fortran, C++, Rust
- Algorithms: `laplap`, `inline`
- Toolchains
  - Fortran, C++: GNU, Cray, Intel, PGI
  - Rust: `rustc`
- Sequential
- Parallel
  - Fortran, C++: OpenMP
  - Rust: Rayon
- GPU
  - Fortran, C++: OpenACC, OpenMP offloading, CUDA
  - Rust: Accel

$\Longrightarrow$ 52 versions $\times$ 4 grid sizes



1. MPI works, but not enough time to debug the partitioner

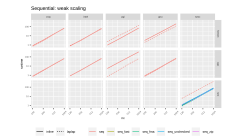
- arrays in column-major order
- −O3 or equivalent
- no LTO
- optimization reports
- shared libraries
- C interface

```
void diffuse(
  float * in_field,
  float * out_field,
  size_t  nx,
  size_t  ny,
  size_t  nz,
  size_t  num_halo,
  float   alpha,
  size_t  num_iter
)
```

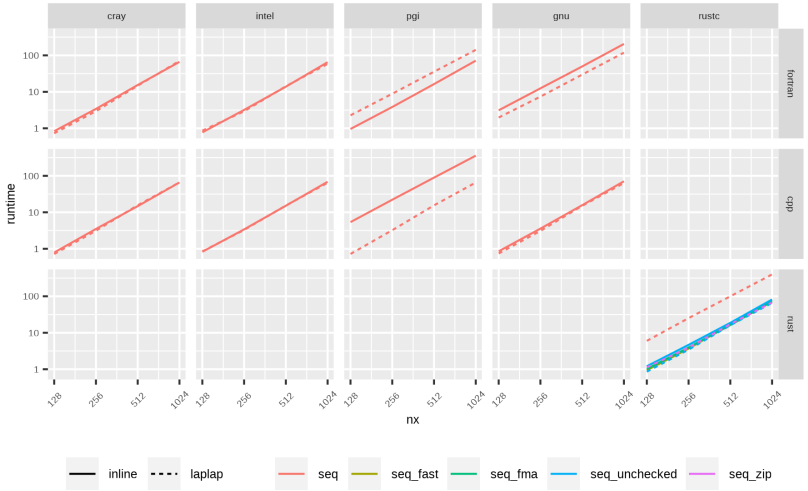

- arrays in column-major order
- −O3 or equivalent
- no LTO
- optimization reports
- shared libraries
- C interface

```
void diffuse(
  float * in_field,
  float * out_field,
  size_t  nx,
  size_t  ny,
  size_t  nz,
  size_t  num_halo,
  float   alpha,
  size_t  num_iter
)
```

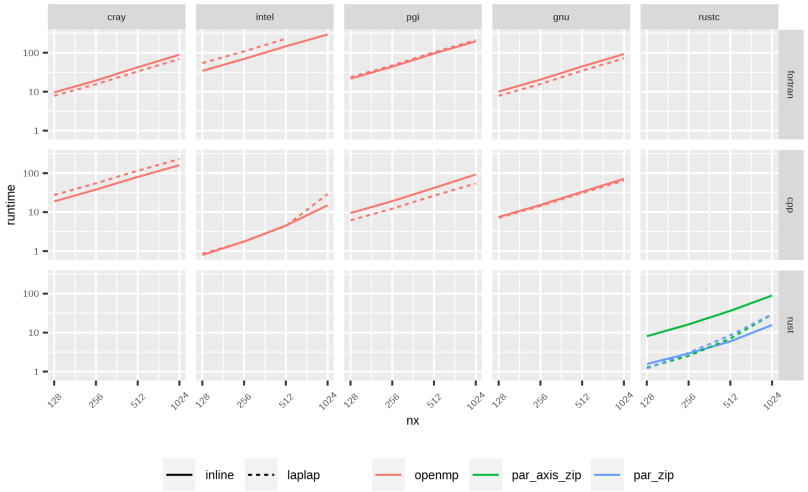
1. Show code on gitlab

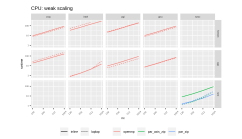


Sequential: weak scaling

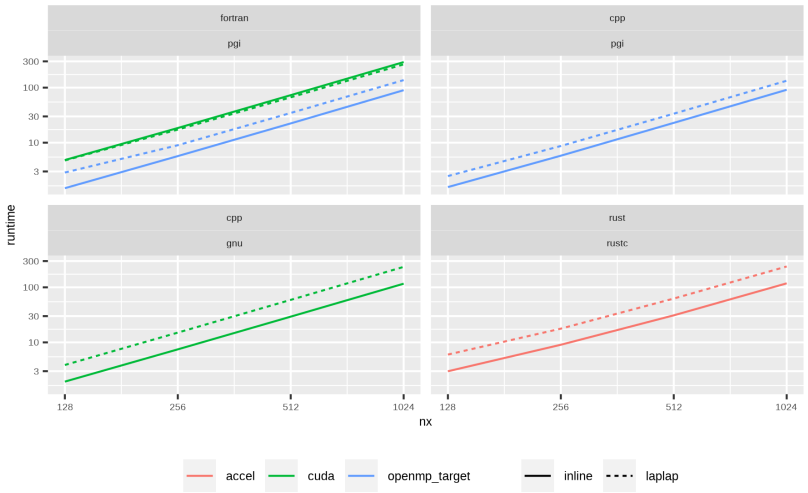


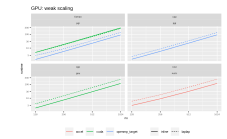
CPU: weak scaling





GPU: weak scaling





Conclusions

- \+ Rust fast enough for scientific software
- \+ Safer code
- \+ Clearer error messages
- − Support for certain features is lacking
- ∼ All languages sometimes lead to lots of boilerplate

Conclusions
+ Rust fast enough for scientific software
+ Safer code
+ Clearer error messages
− Support for certain features is lacking
∼ All languages sometimes lead to lots of boilerplate

1. Rust compiler does good job of vectorizing code
2. Rust: clear what you get
3. Fortran/C++: different support, many bugs
4. GNU doesn't warn about lack of offloading
5. Cray sometimes silently generates invalid PTX code

Reccomendations

- Today
  - Rust for frontend/driver
- Tomorrow (some effort required)
  - GPU support
  - Evolve `ndarray`
  - ScaLAPACK bindings, …
- $\Longrightarrow$ continue to pursue Rust in HPC
  - it has potential

See:

- `https://www.arewelearningyet.com/scientific-computing/`
- `https://www.arewelearningyet.com/gpu-computing/`
- `https://git.cscs.ch/msudwoj/rust-in-hpc`

Reccomendations
- Today
  - Rust for frontend/driver
- Tomorrow (some effort required)
  - GPU support
  - Evolve ndarray
  - ScaLAPACK bindings, …
- ⟹ continue to pursue Rust in HPC
  - it has potential

See:
- https://www.arewelearningyet.com/scientific-computing/
- https://www.arewelearningyet.com/gpu-computing/
- https://git.cscs.ch/msudwoj/rust-in-hpc

1. With Cray and Intel moving to LLVM, cross-language LTO soon?

Start using Rust today!

```
> curl https://sh.rustup.rs -sSf | sh
> rustup toolchain install nightly
> rustup target add nvptx64-nvidia-cuda
> # On Piz Daint
> export
>   CARGO_TARGET_X86_64_UNKNOWN_LINUX_GNU_RUSTFLAGS="
>     -C target-cpu=haswell
>     -C relocation-model=dynamic-no-pic
>   "
>   CARGO_TARGET_NVPTX64_NVIDIA_CUDA_RUSTFLAGS="
>     -C target-cpu=sm_60
>     -C target-feature=+sm_60,+ptx60
>     -C relocation-model=dynamic-no-pic
>   "
>   MPICC=cc
> cargo install ptx-linker
```


Start using Rust today!
> curl https://sh.rustup.rs -sSf | sh
> rustup toolchain install nightly
> rustup target add nvptx64-nvidia-cuda
> # On Piz Daint
> export
> CARGO_TARGET_X86_64_UNKNOWN_LINUX_GNU_RUSTFLAGS="
> -C target-cpu=haswell
> -C relocation-model=dynamic-no-pic
> "
> CARGO_TARGET_NVPTX64_NVIDIA_CUDA_RUSTFLAGS="
> -C target-cpu=sm_60
> -C target-feature=+sm_60,+ptx60
> -C relocation-model=dynamic-no-pic
> "
> MPICC=cc
> cargo install ptx-linker