Comparative Analysis of a Containerized Compilation Process Versus Bare-metal Compilation Runs

Dana Singh
Thomas Robinson
Geophysical Fluid Dynamics Lab (GFDL)

CANOPIE Workshop
November 17th, 2024

SC24
Atlanta, GA | hpc creates.

GFDL Climate Models Utilize FRE

- FRE: **F**lexible Modeling System **R**untime **E**nvironment
  - Toolset that automates steps and manages experiments to run GFDL climate models
  - Internal lab use

- Checkout model source code
- Compile model

- Configure and run model

- Post-process model output
- Launch analysis scripts

- Modernization of FRE Bronx to FRE Canopy
  - FRE-CLI
  - Users use FRE Canopy to build and run models at GFDL
    - fre [tool] [subtool] [options]
    - Ex.: fre make create-checkout [options]

Model Container: Motivation

1

Containerize compilation process:

- Create model container
- Portability, flexibility
- Easy to write and maintain yaml configuration files
- Support both bare-metal and container builds

2

Integrate model container into FMS Runtime Environment (FRE) Ecosystem

- FRE 2024 integration
- Users don't need to have technical knowledge of building dockerfiles and containers → workflow handles that for them

3

Effectively use container to meet bare-metal performance for compilation process

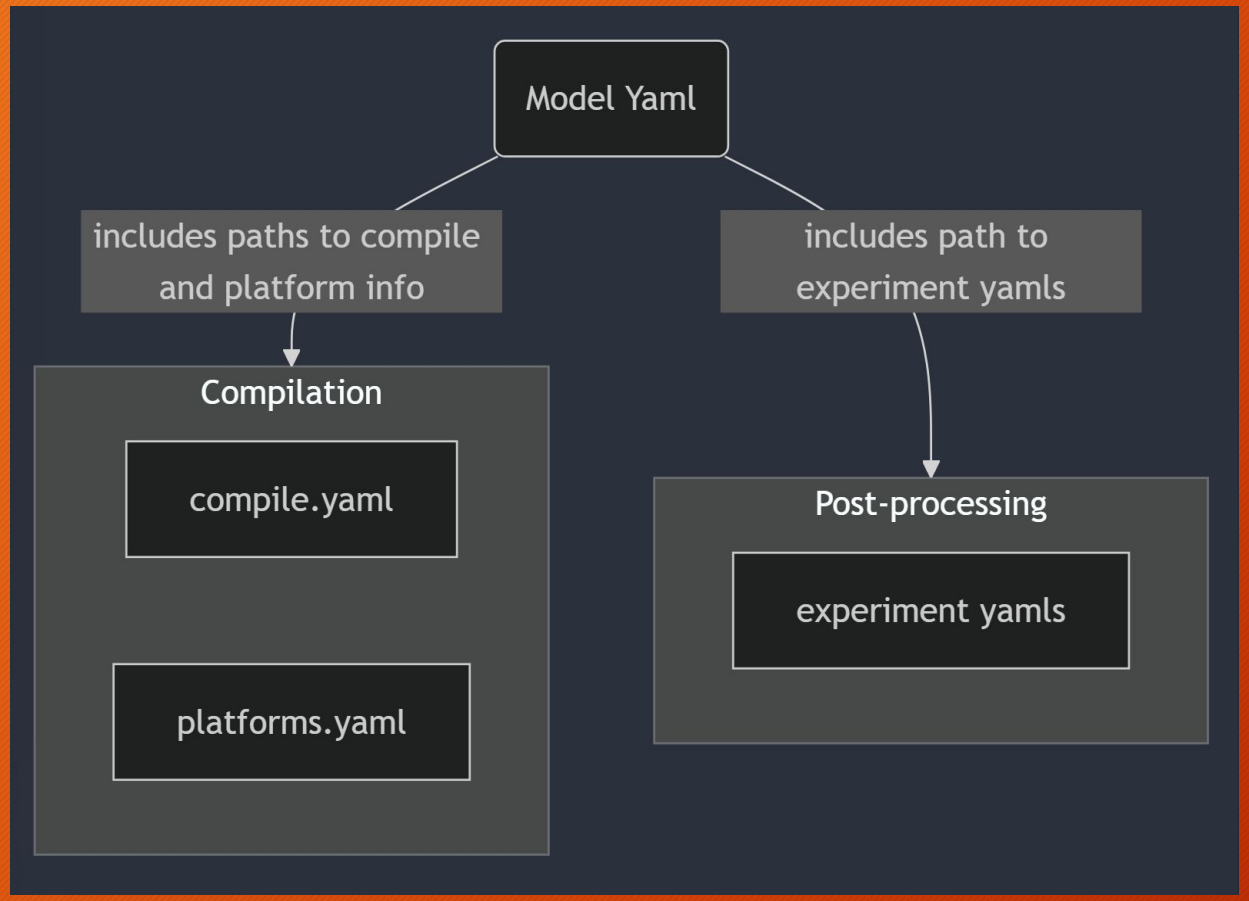
- Comparative analysis on runtime performance

Model Container Image

- Base image
  - intel 2024.01 with spack stack
    - Useful for production level modeling
    - Distributed by ParaTools/E4S
    - Integrated into FRE
- Built with podman
- Run with apptainer
  - Interchangeable with singularity

"Spack-stack provides a practical framework for setting up and managing software libraries…" (https://github.com/JCSDA/spack-stack)

Yaml Framework



- Yamls configuration files
  - Platforms.yaml, compile.yaml, model yaml
  - Easily validated through json schemas
  - Clearly conveys compile information
  - Easily points to other yamls
- Yamls give ability …
  - to specify additional libraries to look for in container to set up spack environment
  - Easily set up and parse information needed for container

Model Container Development

YAML framework for model information

Rather than GFDL XMLs, we use model, compile, and platform yamls

FRE make Canopy

Rewritten from perl to python. Integrated in updated FRE-CLI (modularized)

Dockerfile Creation

The Dockerfile, generated by FRE, pulls from an e4s base image and is used to build the model container.

- Dockerfile
  - Spack-stack based image with intel compilers
  - Image layers
  - use podman/docker (to build) and apptainer/singularity (to run)

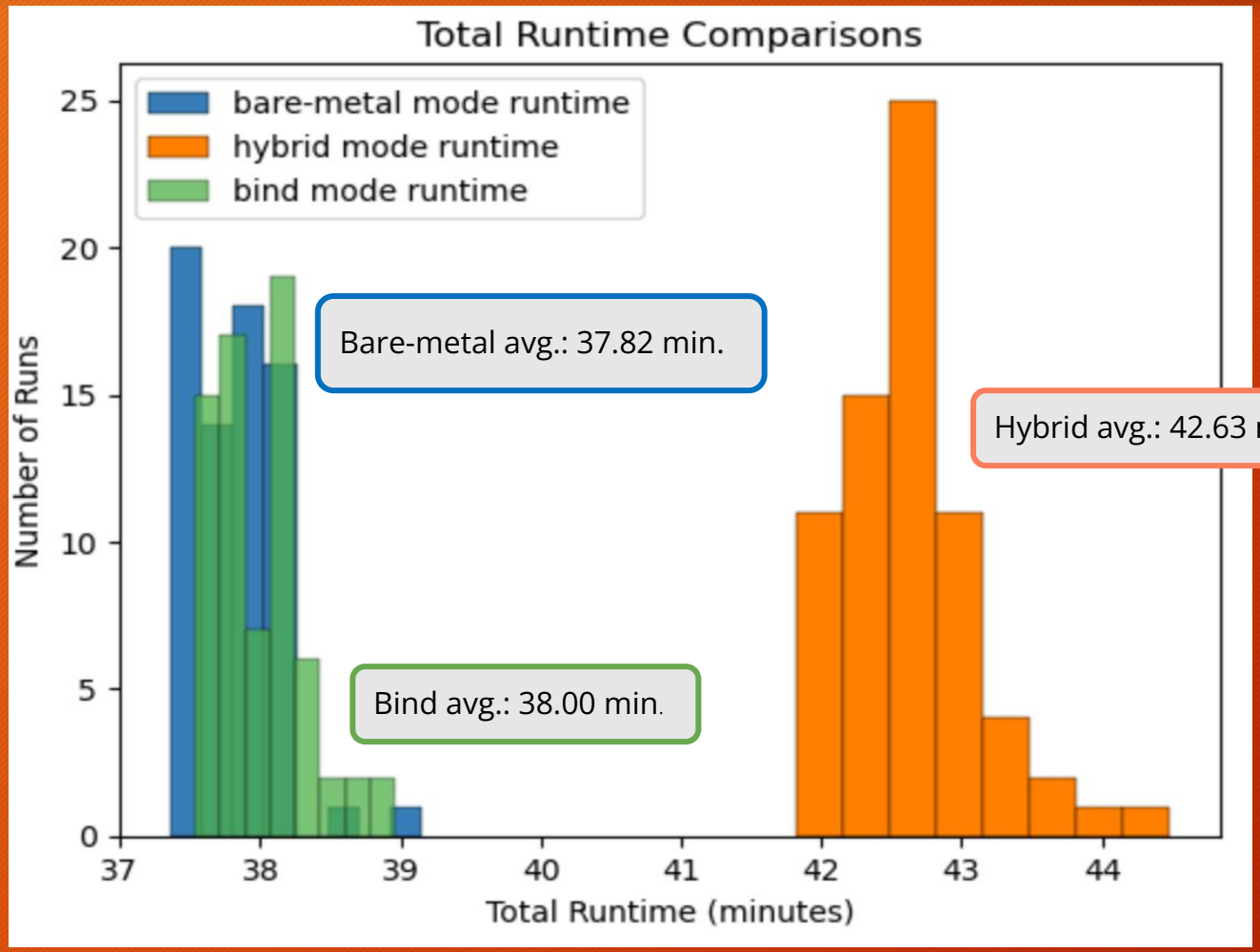
Model Container: MPI BINDING

- MPI binding:
  - Files/directories on host machine are mounted into the container
  - Without binding in host MPI - model cannot effectively and efficiently communicate with the container

- 3 comparative tests done

Bare-metal:	Hybrid:	Bind:
• Running model on-prem system (no container)	• Using container • Using container's MPI	• Using container • Using host system's MPI

Bind mode allows for a bare-metal performance



Distribution of data for bare-metal, hybrid, and bind mode compilation approaches. Each approach was run 70 times and bin size covers 15 seconds.

Bare-metal and bind modes demonstrated similar average runtimes, while hybrid mode exhibited a 4-5 minute slow down.

In development

- Comparative analysis for scaled experiments!
- Integrate model container in frerun step
  - portability, flexibility
  - possibly help maintain answers
- Reduce size of container
  - container will hold only necessary packages and libraries
- Multi-stage build
  - strip out intel compiler

Thank you!