Verifying and Validating CISM: The LIVVkit Experience

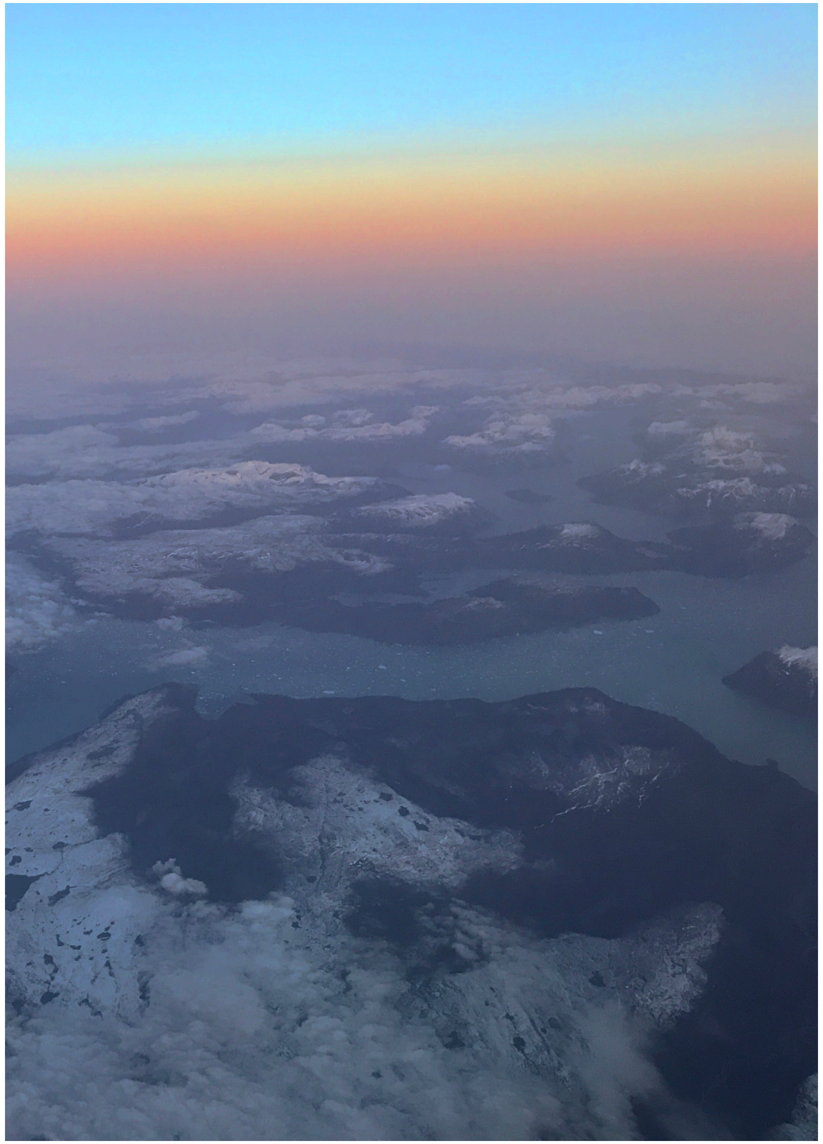
Michael Kelleher
Katherine Evans
Joseph Kennedy
Gunter Leguy
Bill Lipscomb

ORNL is managed by UT-Battelle LLC for the US Department of Energy

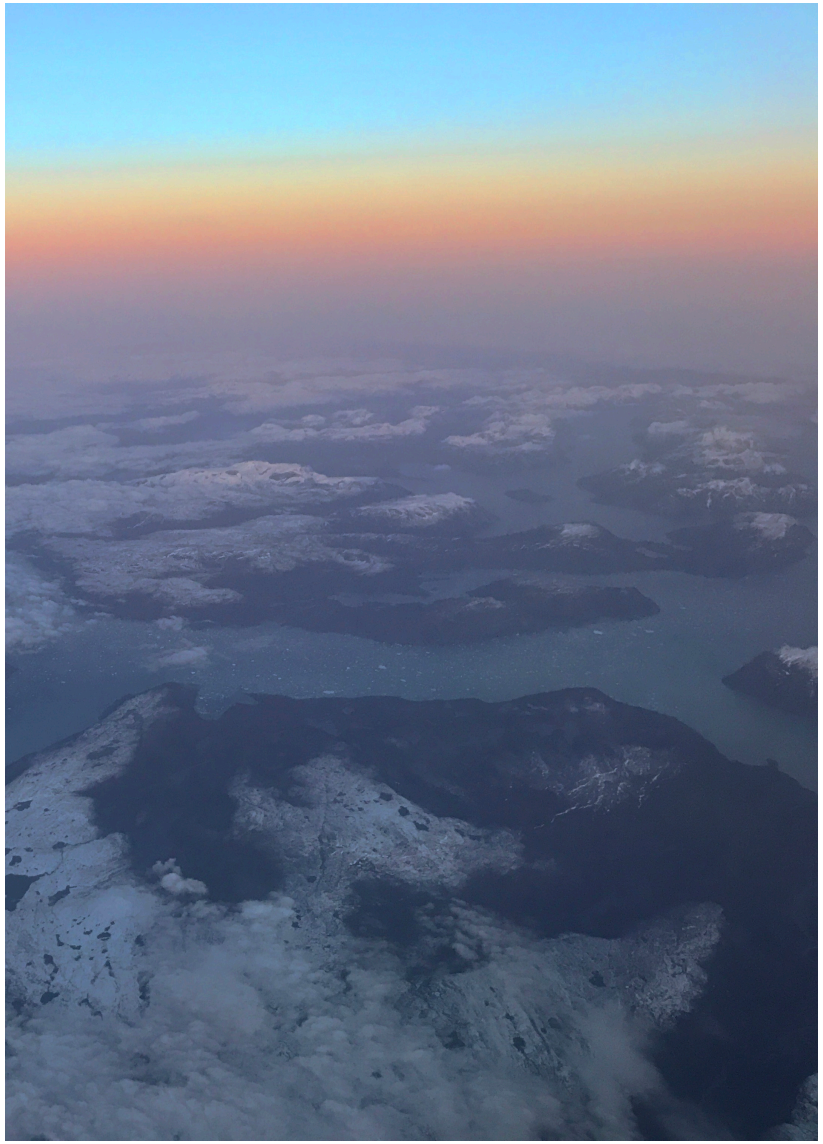
Introduction: What is validation / verification?

- Numerical models are inherently imperfect
- Need a way to ensure the best possible representation of real physics
- **Numerical** verification – “Are we solving the equations correctly?”
- **Code** verification – “did we build what we wanted?”
- **Physical** validation – “Are we using the right physics?”
- **Performance** validation – “did we build what the users wanted?”

[From LIVVkit Documentation]



Introduction: What LIVVkit?



- **L**and **I**ce **V**alidation & **V**erification tool**kit**
- Python based, open source (BSD 3-clause), validation and verification toolkit for land ice numerical models
- https://github.com/LIVVkit/LIVVkit
- Developed through the PISCEES[1] DOE SciDAC[2] project
- Process output from two model runs, performing a variety of checks
  - Easiest to run the ”regression suite” in CISM, which runs tests, and organizes output into a structure which LIVVkit can understand

[1] Predicting Ice Sheet and Climate Evolution at Extreme Scales

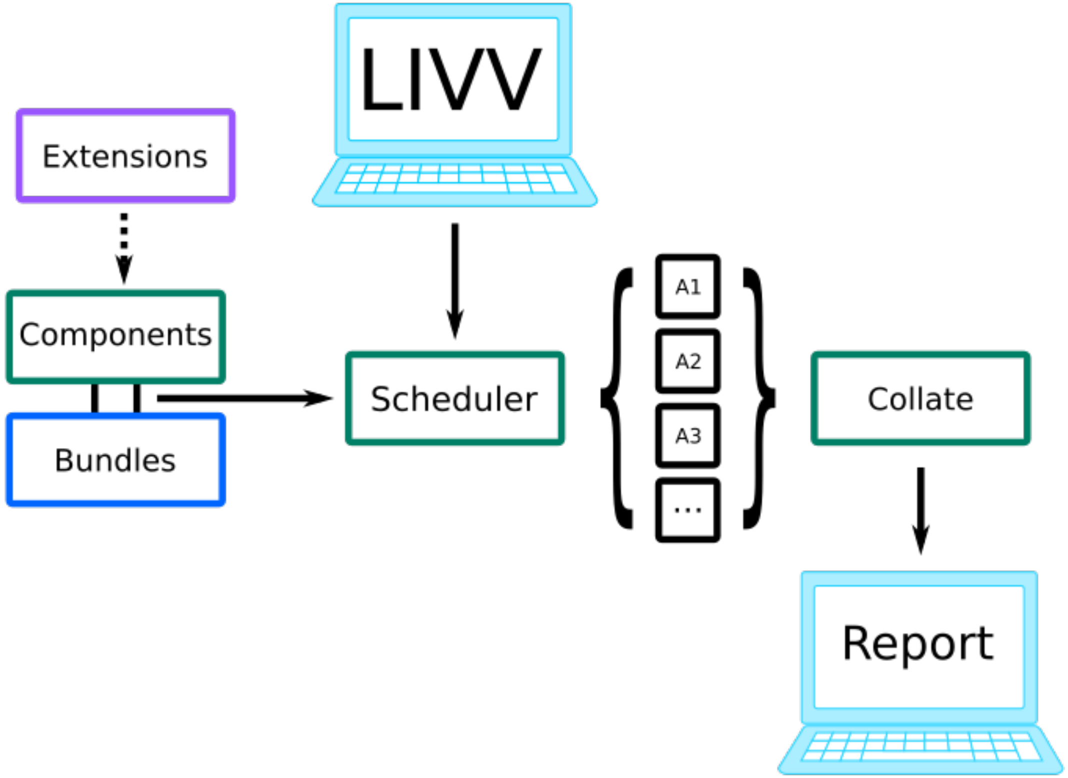
[2] Scientific Discovery through Advanced Computing

What does LIVVkit need?

- Model data! (Of course…but what else?)
- Easiest way for LIVVkit to be installed is through an Anaconda / (or Mamba) environment
  - LIVVkit is on conda-forge and pypi, so can be installed with either
    - `pip install livvkit`
    - `mamba install -c conda-forge livvkit`
- LIVVkit has few dependencies as part of the design philosophy
  - numpy, scipy, pandas, matplotlib, netCDF4
  - jinja2
  - json_tricks
  - pybtex
- Instructions for installation are on the Github repository and documentation website

What does LIVVkit need?

- LIVVkit uses JSON configuration files
  - Specify the variables to examine
  - Location of the data
  - Importantly – description of the tests, which populates the output reports
- API has four primary modules:
  - Bundles
  - Components
  - Scheduler
  - Utilities



(Kennedy, et al., 2017)

What is BATS and how does it run?

- The **B**uild **A**nd **T**est **S**uite – like other test cases – it's a Python script

- A few scripts are available to set up the HPC environment

- BATS can be invoked using the script "build_and_test.py" in the CISM regression test directory (`cism/tests/regression`)
  - Specifying the platform (e.g., "cheyenne-intel")
  - And additional arguments enabling the performance suite, specifying build and output directories

- The Python script generates one or more batch scripts, which presently must be submitted manually

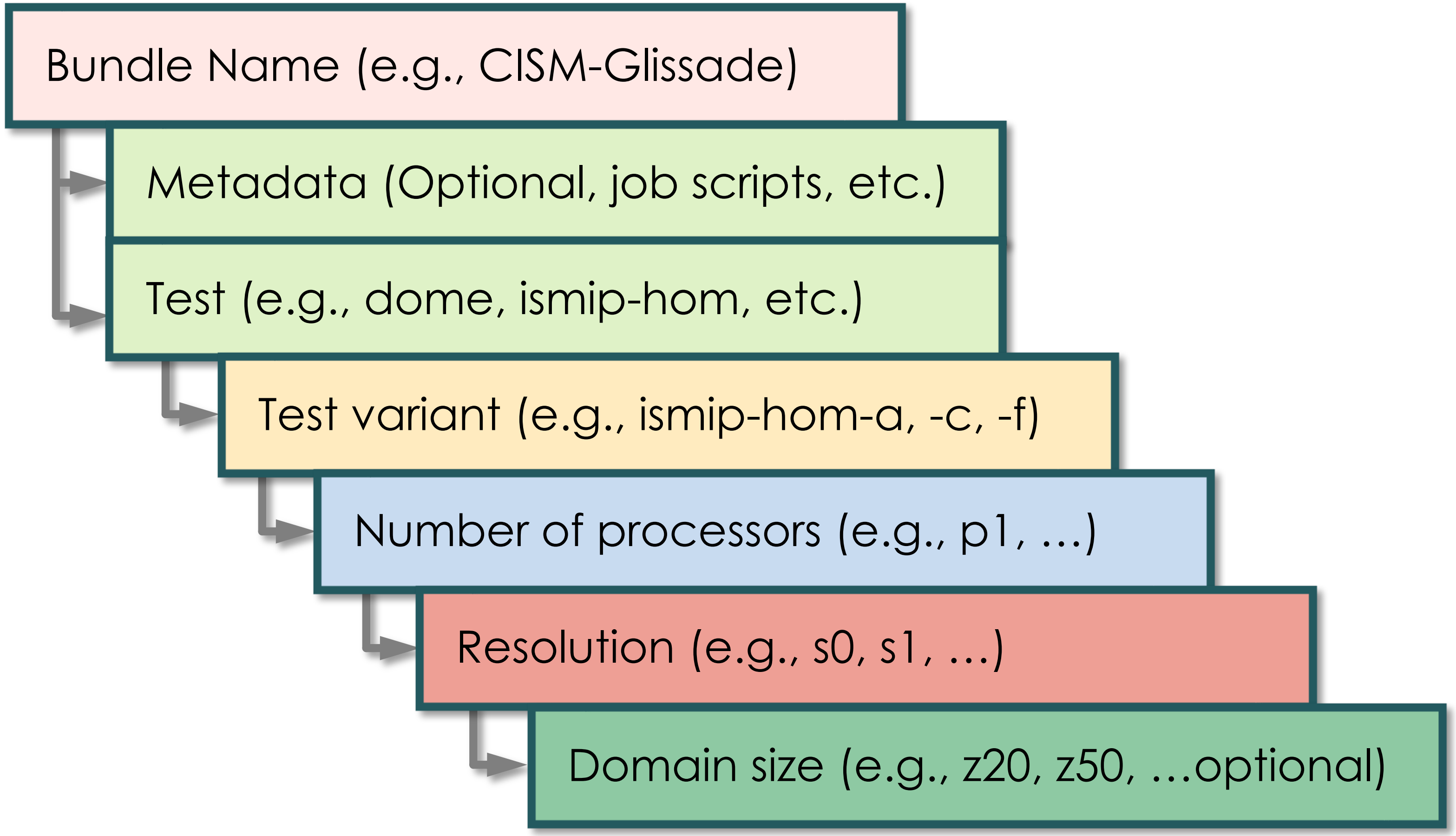
What tests are run in the “BATS”?

- BATS by default on an HPC (Cheyenne) runs:
  - ISMIP-HOM cases
    - **A** and **C** at 20 km and 80 km scale
    - **F** at 100km scale, 0 slip ratio
  - Idealized
    - Dome, varying scales and processor counts for performance
    - Circular and Confined shelf
    - Stream

		Number of processors						
	Dome	1	2	4	8	16	64	256
Domain size	31^2	✓	✓	✓	✓			
	62^2	✓		✓				
	124^2	✓				✓		✓
	248^2						✓	✓
	496^2							✓

How is LIVVkit run on the output of BATS?

- LIVVkit expects a particular directory structure, which BATS will create, and organize model output
- LIVVkit's main script "livv" takes command line arguments
- `livv --verify $TEST $REF -o OUTPUT_DIR`
  - $TEST and $REF point to the "Bundle Name" directory of the test and reference output directories



Bundle Name (e.g., CISM-Glissade)

Metadata (Optional, job scripts, etc.)

Test (e.g., dome, ismip-hom, etc.)

Test variant (e.g., ismip-hom-a, -c, -f)

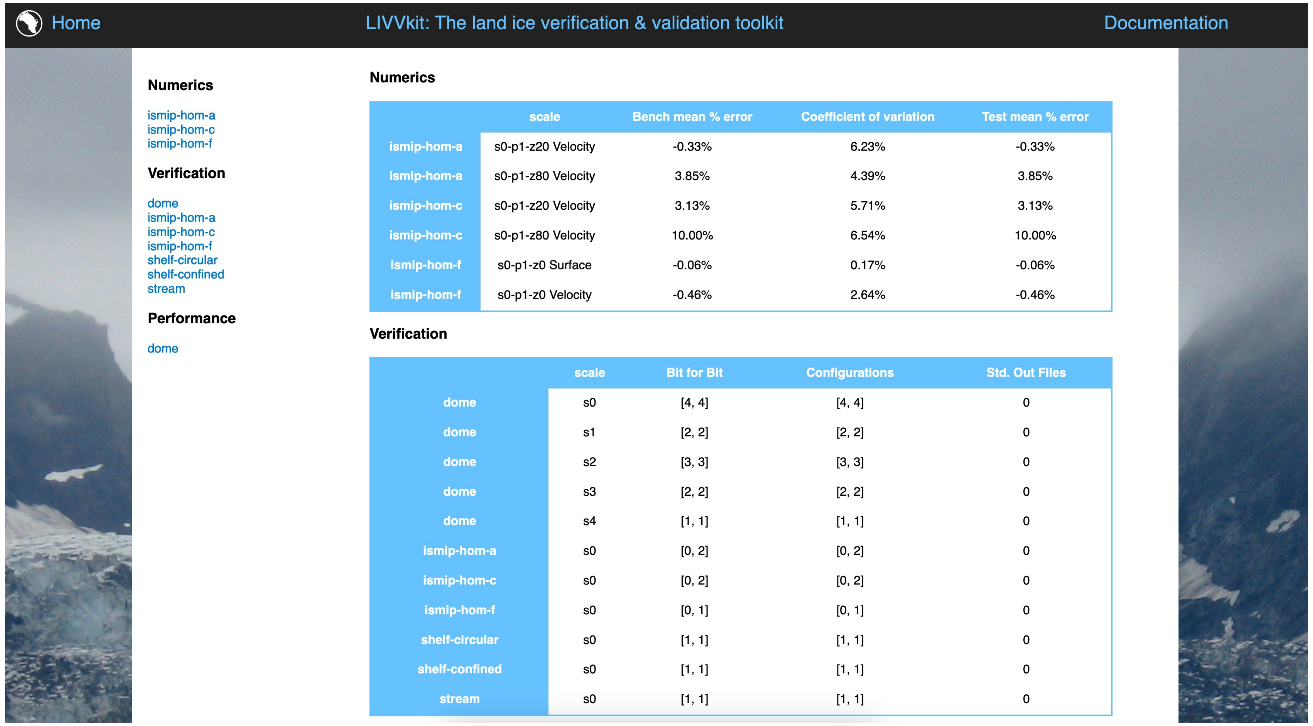
Number of processors (e.g., p1, …)

Resolution (e.g., s0, s1, …)

Domain size (e.g., z20, z50, …optional)

Let’s do a comparison!

- Comparison of CISM “main” to “leguy/update_toward_CISM3”
  - Minor code modifications to allow GPTL timers to be used
  - Infrastructure changes to Cheyenne scripts so queue submission and modules load correctly
- LIVVkit generates an output webpage
- Can be locally served with:
  - “`livv -s`”
- This output was uploaded to Github pages: https://livvkit.github.io/results



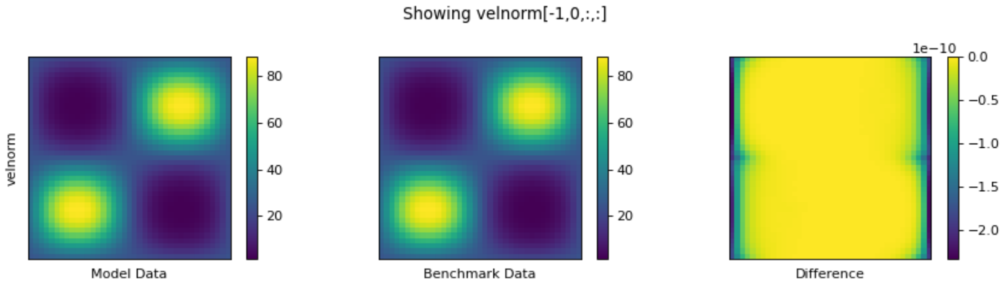
Numerics

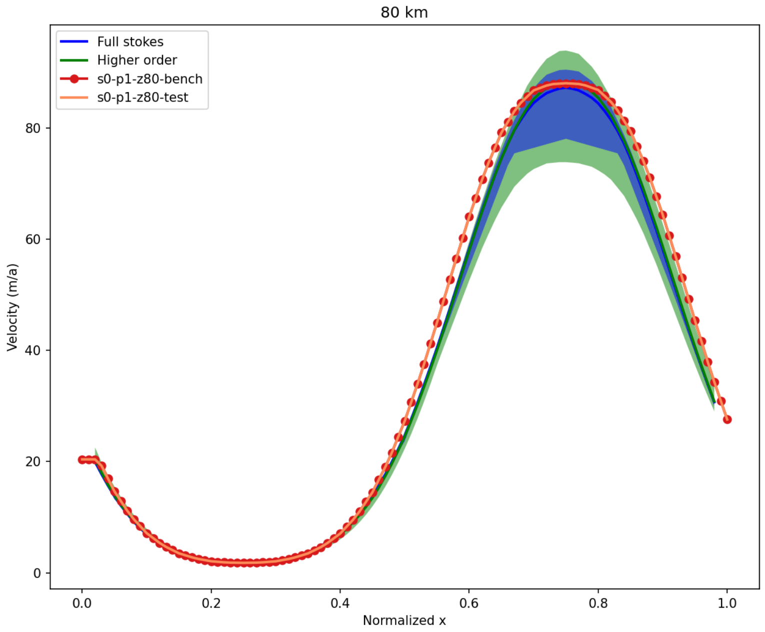
	scale	Bench mean % error	Coefficient of variation	Test mean % error
ismip-hom-a	s0-p1-z20 Velocity	-0.33%	6.23%	-0.33%
ismip-hom-a	s0-p1-z80 Velocity	3.85%	4.39%	3.85%
ismip-hom-c	s0-p1-z20 Velocity	3.13%	5.71%	3.13%
ismip-hom-c	s0-p1-z80 Velocity	10.00%	6.54%	10.00%
ismip-hom-f	s0-p1-z0 Surface	-0.06%	0.17%	-0.06%
ismip-hom-f	s0-p1-z0 Velocity	-0.46%	2.64%	-0.46%

Verification

	scale	Bit for Bit	Configurations	Std. Out Files
dome	s0	[4, 4]	[4, 4]	0
dome	s1	[2, 2]	[2, 2]	0
dome	s2	[3, 3]	[3, 3]	0
dome	s3	[2, 2]	[2, 2]	0
dome	s4	[1, 1]	[1, 1]	0
ismip-hom-a	s0	[0, 2]	[0, 2]	0
ismip-hom-c	s0	[0, 2]	[0, 2]	0
ismip-hom-f	s0	[0, 1]	[0, 1]	0
shelf-circular	s0	[1, 1]	[1, 1]	0
shelf-confined	s0	[1, 1]	[1, 1]	0
stream	s0	[1, 1]	[1, 1]	0

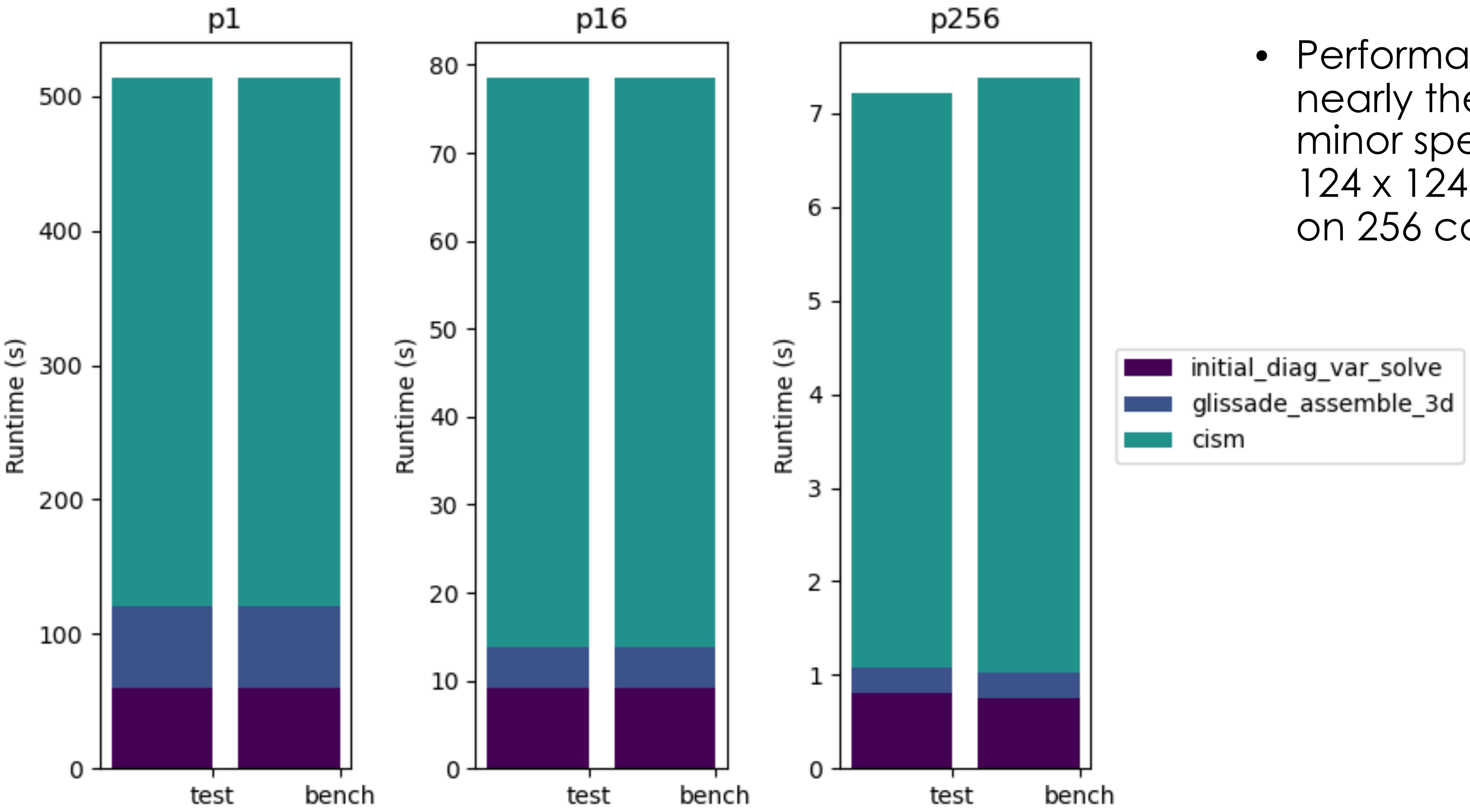
Let’s do a comparison!





- Most tests are bit-for-bit
- Minor (~10^{-11}) differences in the velocity field of ISMIP-HOM-A

Let’s do a comparison!



- Performance is nearly the same, minor speedup for 124 x 124 domain size on 256 cores

OAK RIDGE National Laboratory

Open slide master to edit

Where do we go from here?

- Improvements to LIVVkit
  - Some tweaks to make the landing page more informative
  - Backend coding improvements
  - Handle user requests
- Improvements to BATS
  - Overall structure can be tweaked so a suite of regression tests can be easily modified
  - Building is tied to CISM repository build, perhaps this could be separate so that the version of BATS is not tied to the model version

The LIVVkit Family of Software

- **LIVVkit**: https://github.com/LIVVkit/LIVVkit
  - Validation & Verification
  - LIVVkit's flagship, used to run most other family members
- **LEX** (LIVVkit Extensions): https://code.ornl.gov/LIVVkit/lex
  - Extends LIVVkit to validation against observations / other models
- **Dashboard**: https://github.com/LIVVkit/dashboard
  - Reports nightly test suite of ice sheet models (currently MALI and BISICLES) to Cdash
- **EVV4ESM** (Extended Validation and Verification for Earth System Models): https://github.com/LIVVkit/evv4esm
  - Statistical testing of Earth system models, currently E3SM's EAM and MPAS-O
  - Acts as a LIVVkit extension