Light-Weight Parallel Python Tools for Earth System Modeling Workflows

Kevin Paul
Sheri Mickelson
Haiying Xu
John M. Dennis
David Brown

The National Center for Atmospheric Research
Boulder, CO

NCAR

NSF

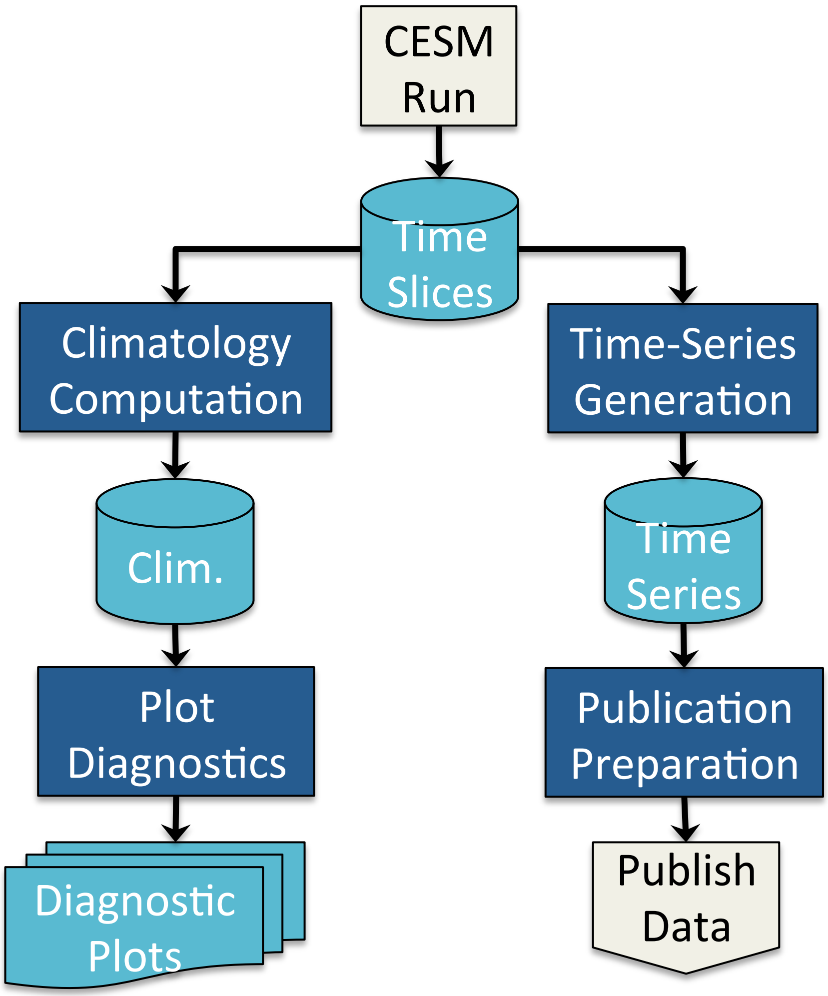
The Problem

Big Data in Earth System Modeling

- NCAR's Community Earth System Model:
  - Massively parallel (MPI-based)
  - Higher resolution simulations
    - … "Big Data"!
- Coupled Model Intercomparison Project:
  - CMIP5 (2010-2013):
    - 20 different institutions from around the world!
    - CESM: 2.5 PB generated ➔ 175 TB published
    - Ran out of time before completing publication!
  - CMIP6 (2016-2020):
    - **EXPECT**: 12 PB generated ➔ 6 PB published

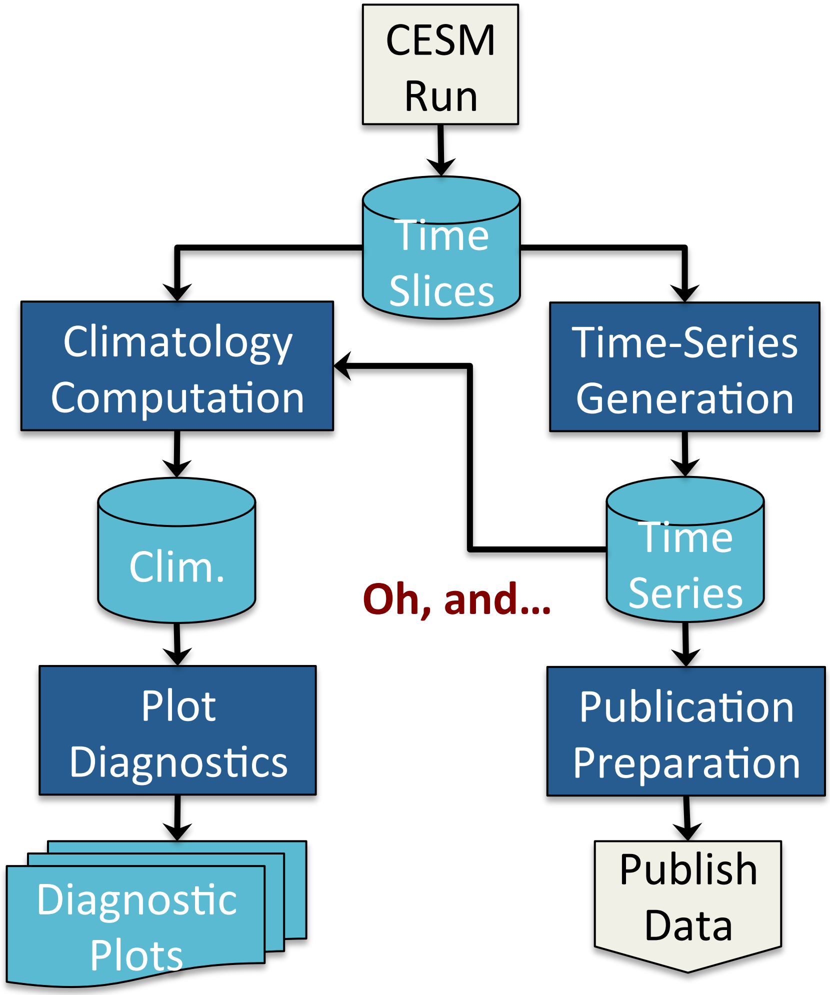
Post-Processing: Why so slow?

- All post-processing steps are serial scripts
  - ***Parallelize***
- Required human intervention between steps
  - ***Automate***
- Error prone workflow
  - ***Thorough testing***



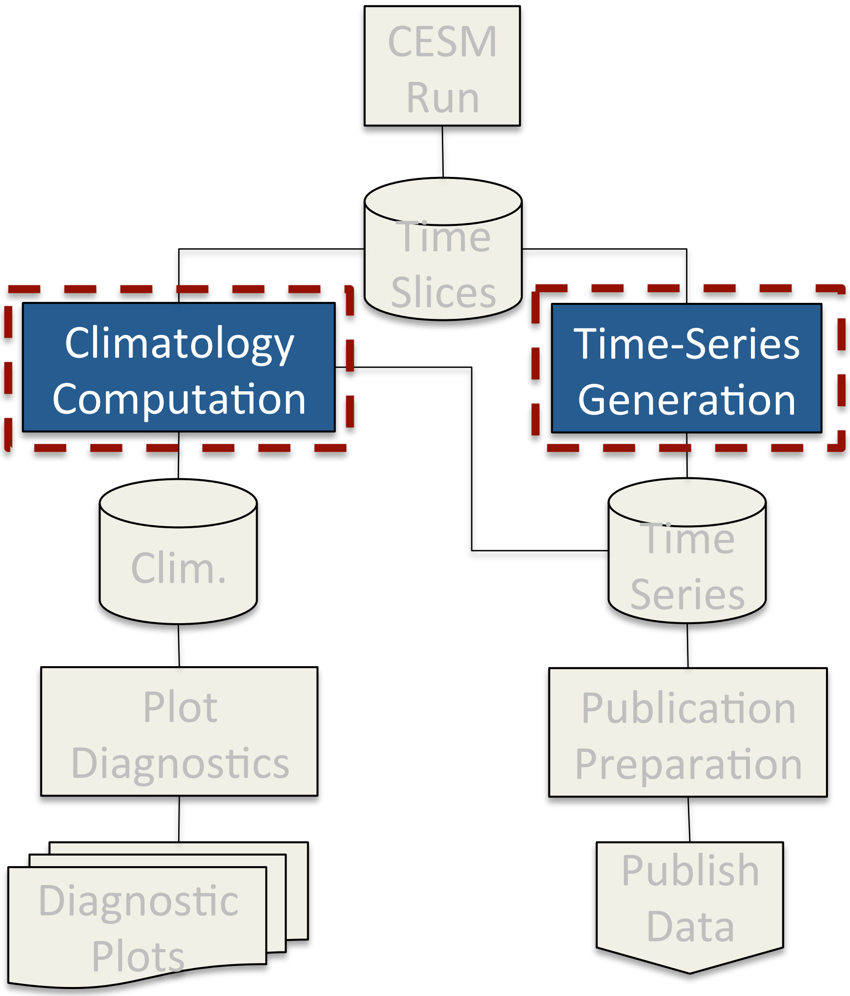
Post-Processing: Why so slow?

- All post-processing steps are serial scripts
  - ***Parallelize***
- Required human intervention between steps
  - ***Automate***
- Error prone workflow
  - ***Thorough testing***



Post-Processing: Why so slow?

- All post-processing steps are serial scripts
  - ***Parallelize***
- Required human intervention between steps
  - ***Automate***
- Error prone workflow
  - ***Thorough testing***



Approach & Design

Principle of Least Astonishment

- Don't change anything that already works!
  - Existing workflow is fine for "little data"
- "Minimally Transformative"
  - Fewest changes seen by the user
  - Best way to achieve "buy-in" from users
  - Target & replace the "bottleneck" scripts
- Requires the least development
  - Fastest to solution
  - Easiest to maintain
- Biggest "bang for the buck"

Why Python?

- Rapid prototyping
- CESM Workflow is already script-driven
- Easily extensible
- Modular
- Existing Module Functionality:
  - numpy
    - fast array-based data manipulation
  - mpi4py
    - Fits into CESM MPI-based workflow
    - No new knowledge to run on supercomputers
  - PyNIO
    - NCAR's multi-format (netCDF, grib, etc) I/O library

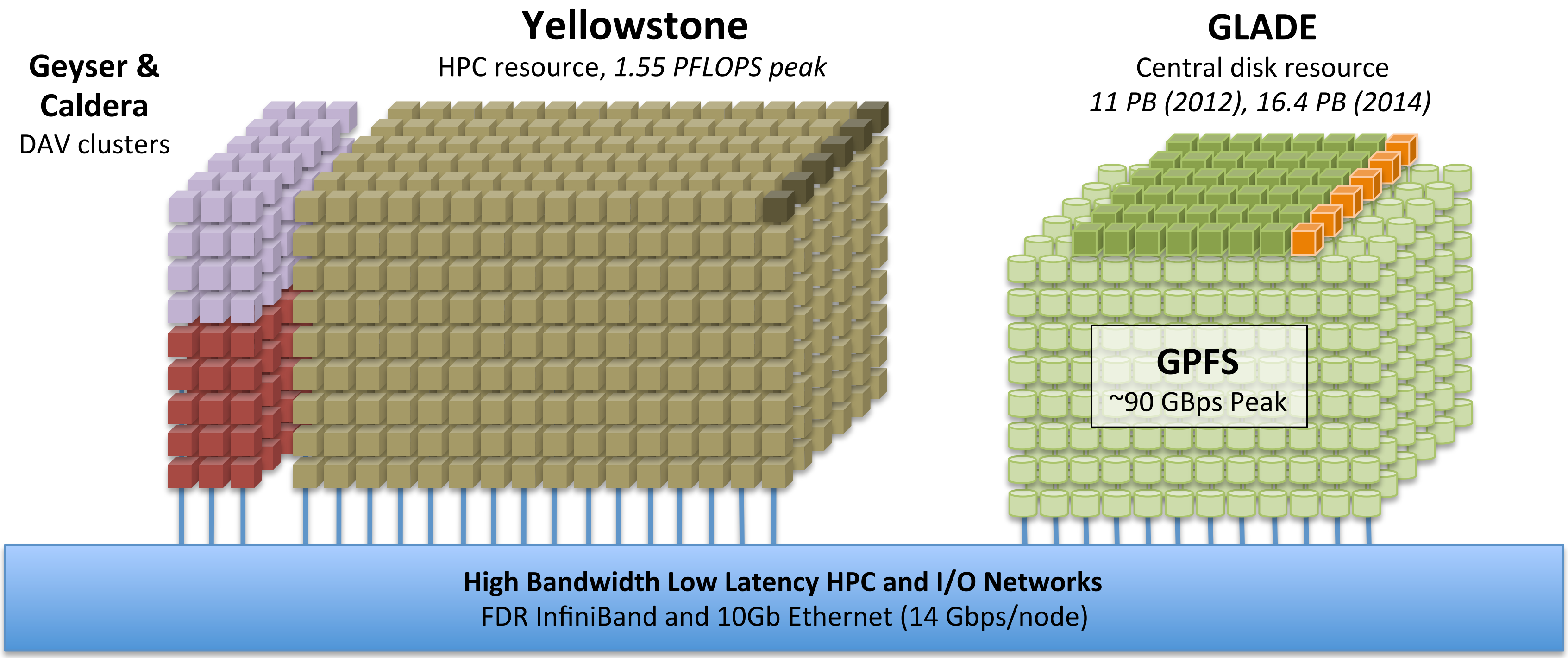
Testing

Testing Datasets

Component Model Name	Resolution	Total Size (GB)	Number of Time-Series Variables
Ice	1 degree	8.2	117
	0.1 degree	556	112
Ocean	1 degree	190	114
	0.1 degree	3100	34
Atmosphere	1 degree	30	132
	¼ degree	1000	198
Land	1 degree	8.7	297
	¼ degree	88	150

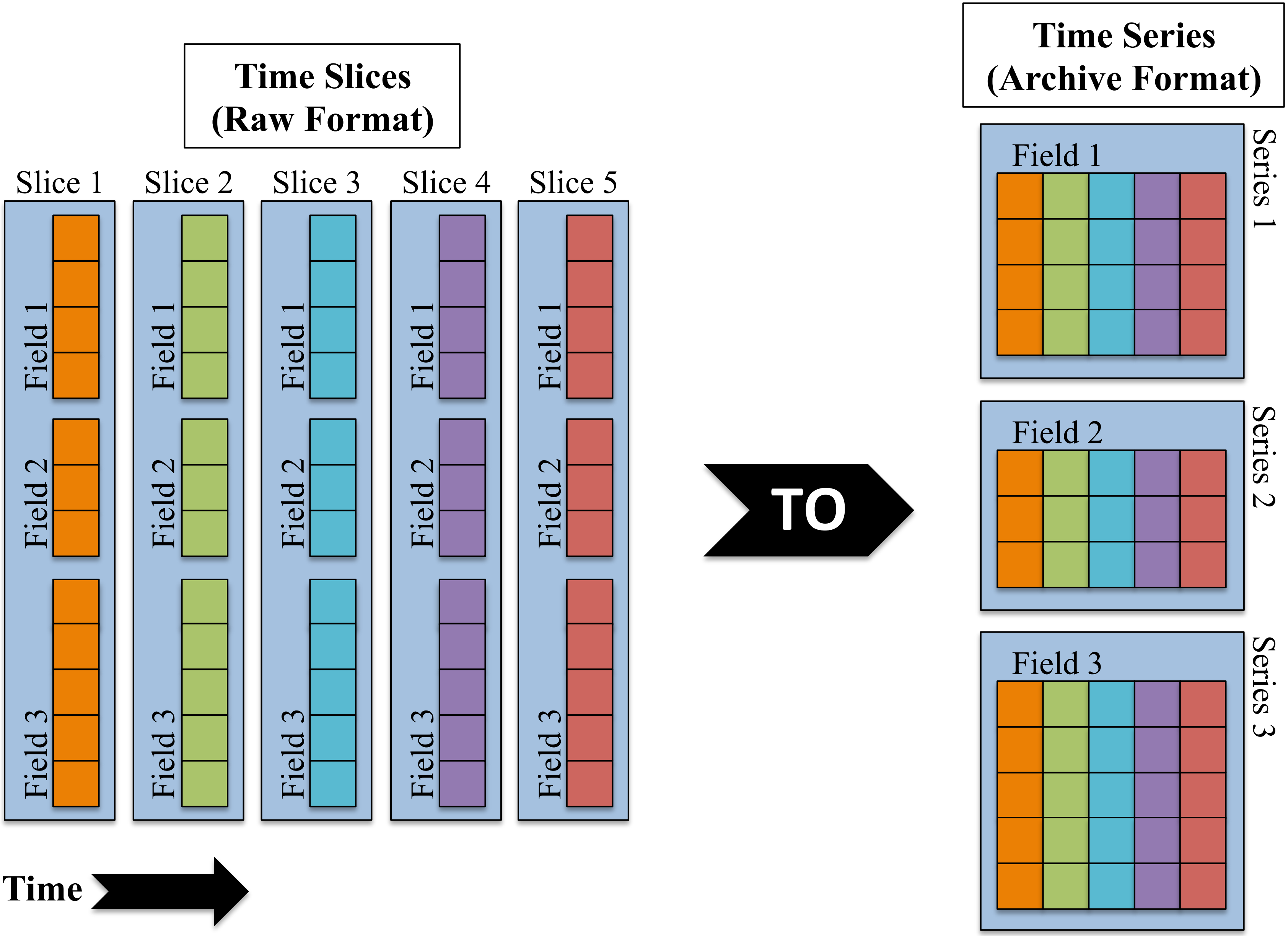
- Datasets span 10 years of monthly data

Testing Platform

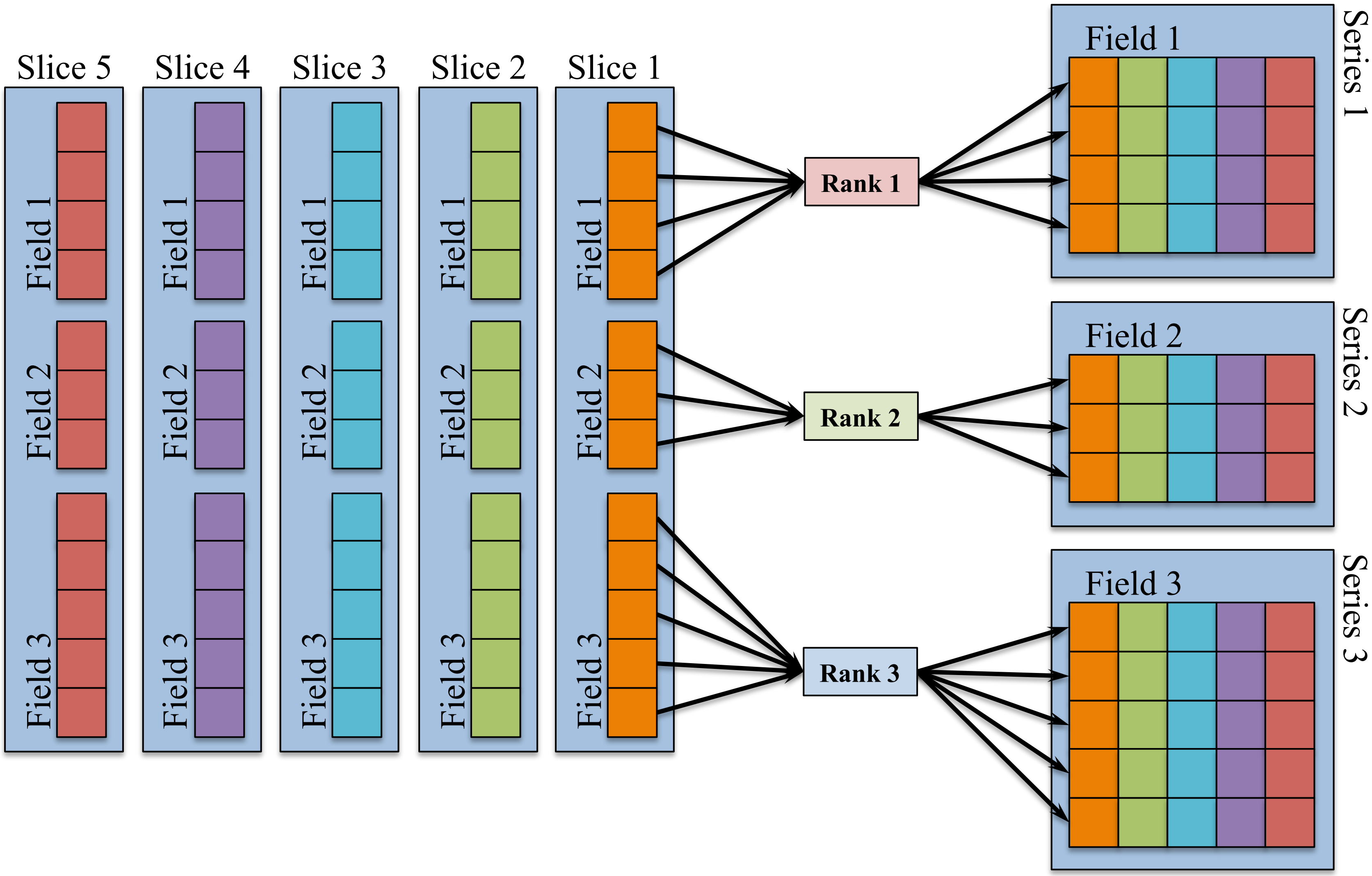


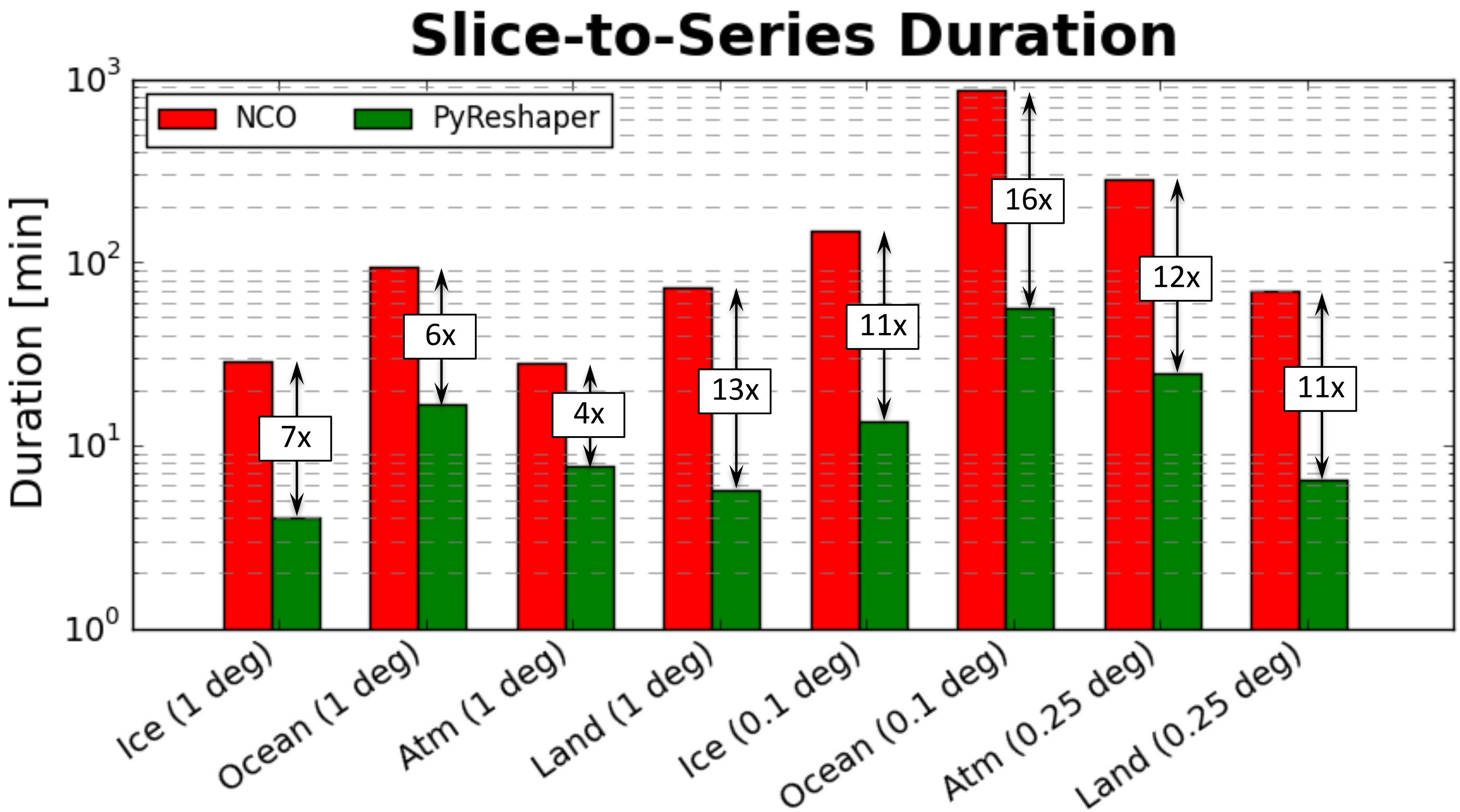
- NCAR's Yellowstone, GLADE & GPFS:
  - ~90 GB/s peak from GPFS
  - ~1.5 GB/s from each compute node

The PyReshaper: "Time-Series Generation"

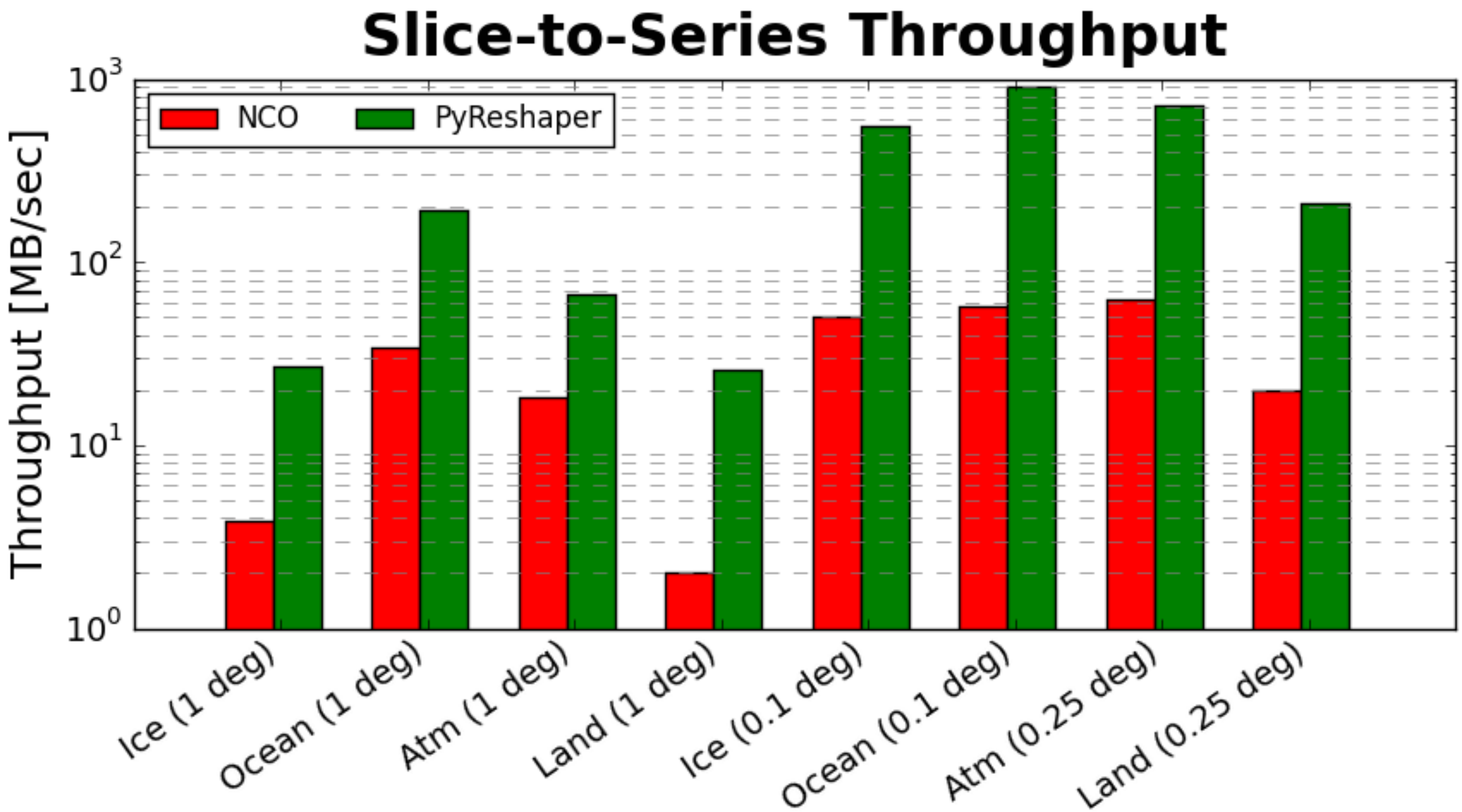

Time Slices
(Raw Format)
Slice 1
Slice 2
Slice 3
Slice 4
Slice 5
Field 1
Field 2
Field 3
Time
TO
Time Series
(Archive Format)
Field 1
Series 1
Field 2
Series 2
Field 3
Series 3

Task Parallelism



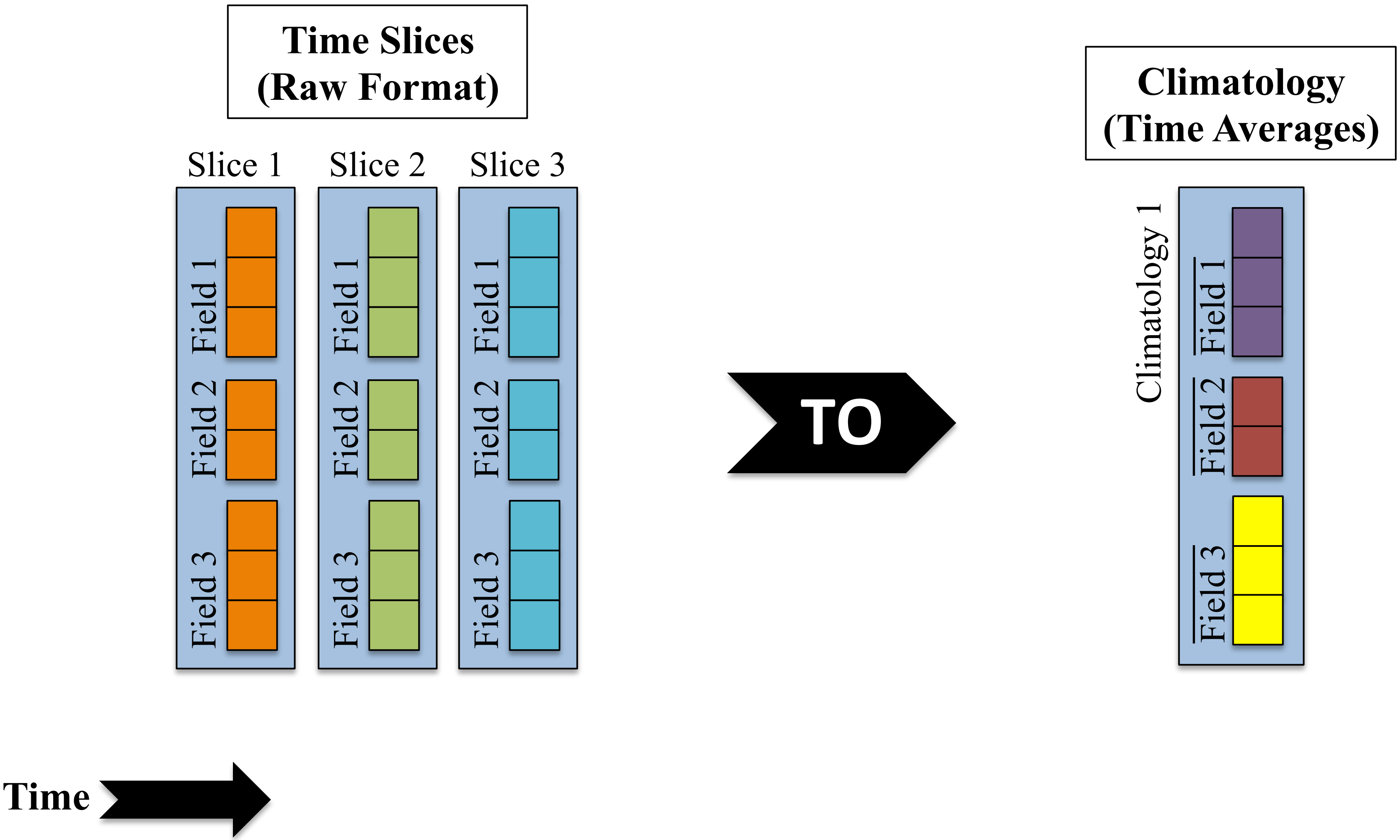


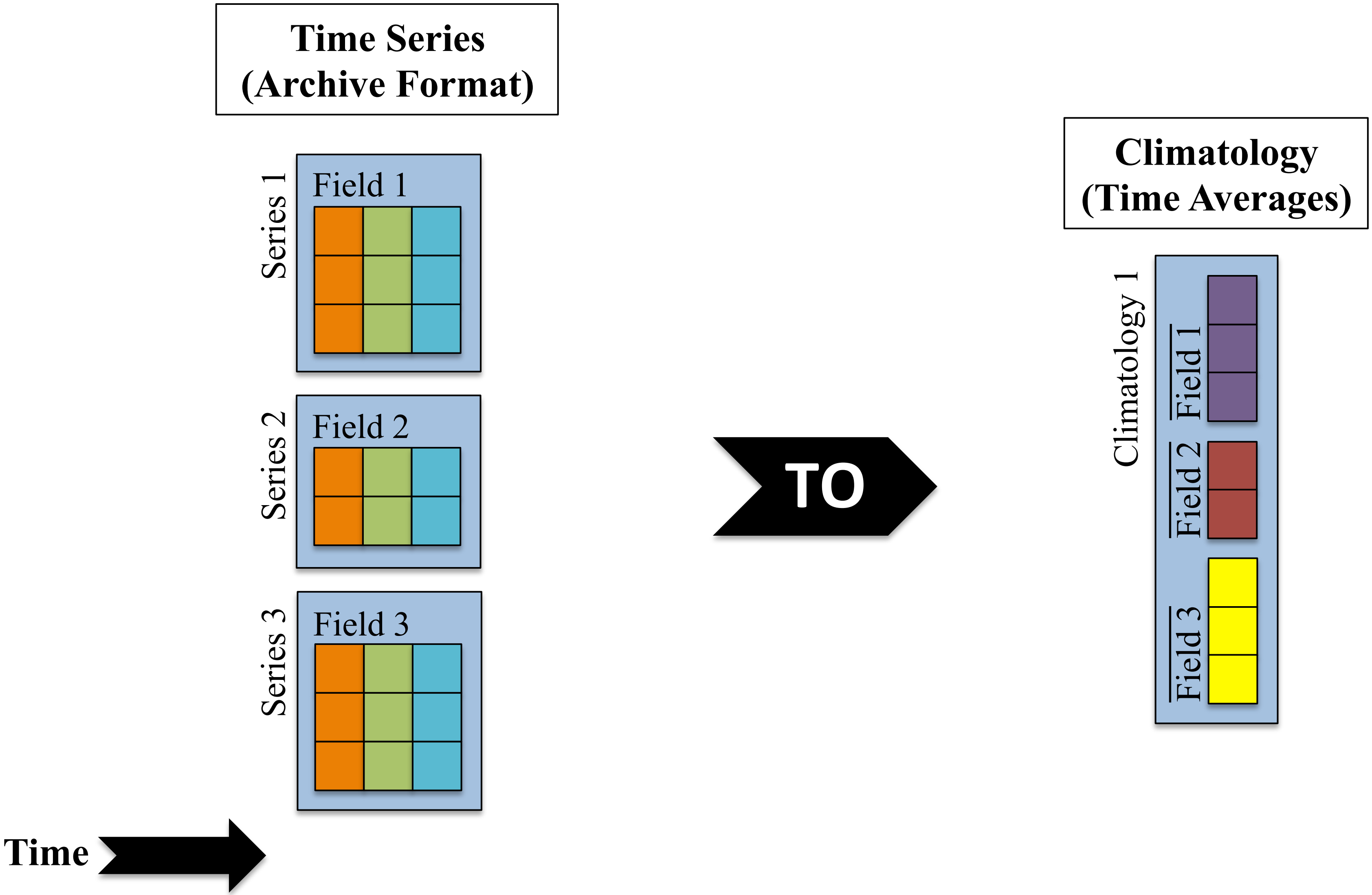
- Run with 4 nodes / 4 processors per node
  - Greater parallelism available!
- Overall 12x speedup ("sum of all times")
  - 7x for Low-Resolution / 14x for High-Resolution

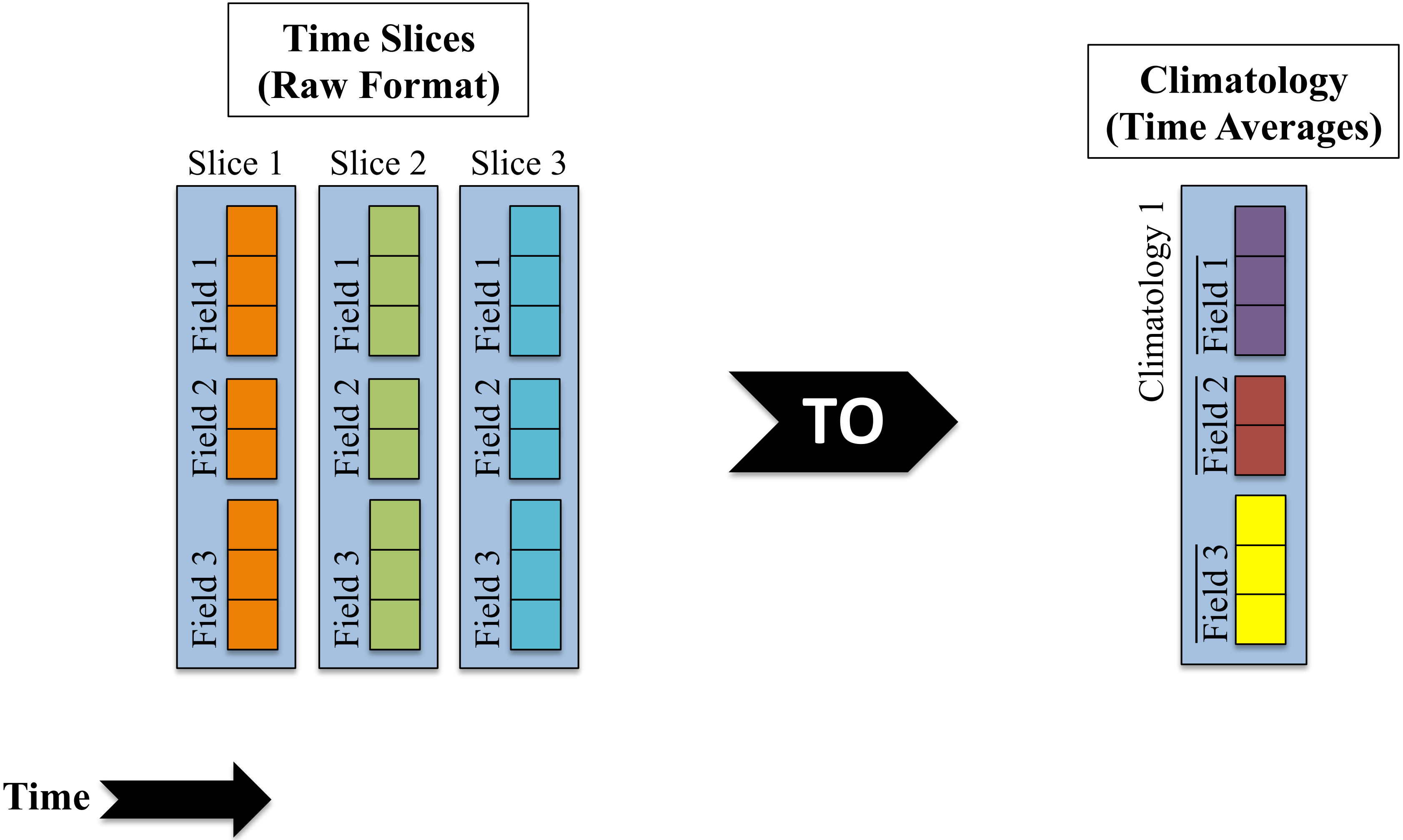


- Run with 4 nodes / 4 processors per node
  - Greater parallelism available!
- Overall 12x speedup ("sum of all times")
  - 7x for Low-Resolution / 14x for High-Resolution

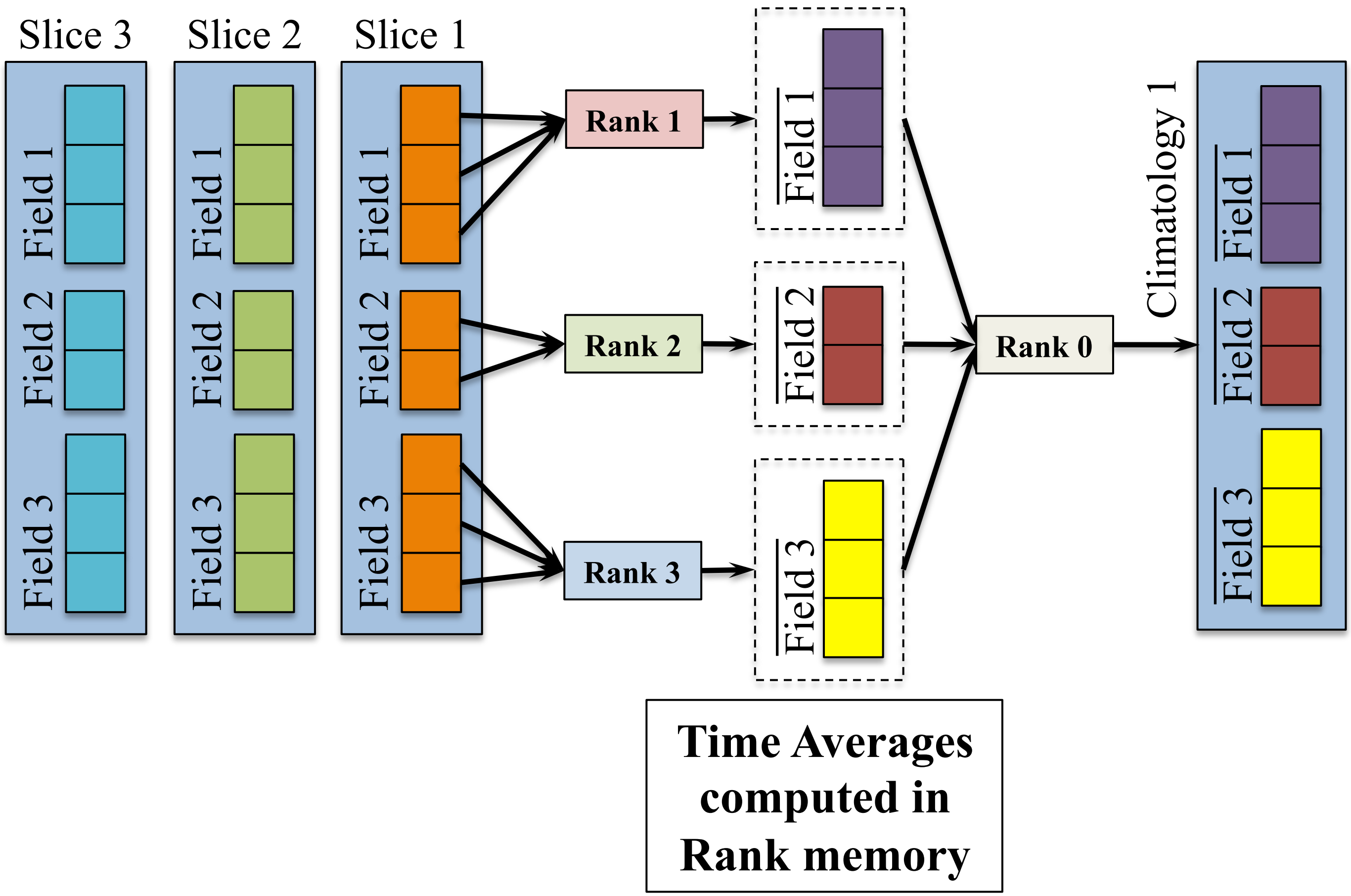
The PyAverager:
“Climatology Computation”


Time Slices
(Raw Format)
Slice 1
Slice 2
Slice 3
Field 1
Field 2
Field 3
TO
Climatology
(Time Averages)
Climatology 1
Field 1
Field 2
Field 3
Time

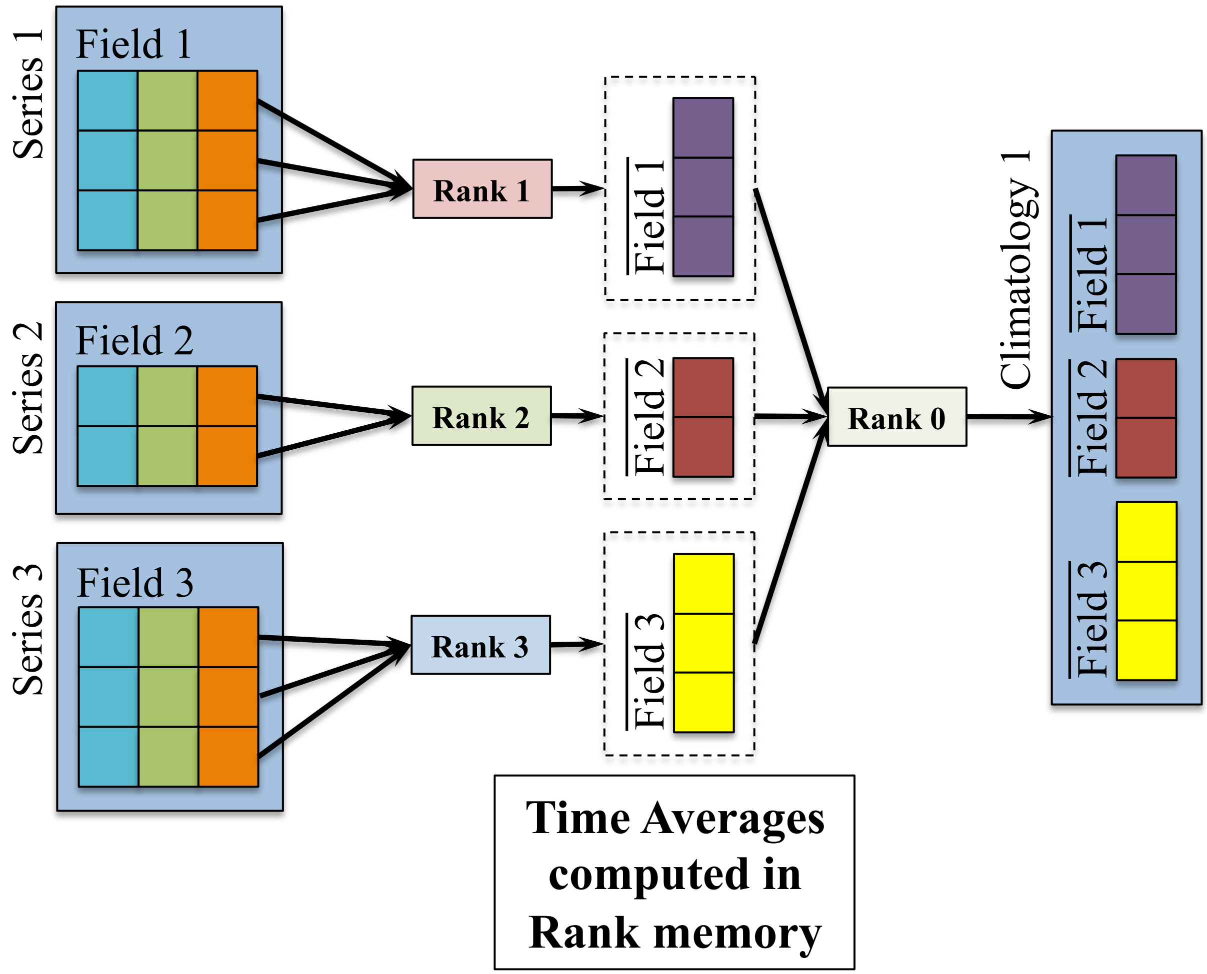

Time Series
(Archive Format)
Series 1
Field 1
Series 2
Field 2
Series 3
Field 3
Time
TO
Climatology
(Time Averages)
Climatology 1
Field 1
Field 2
Field 3


Time Slices
(Raw Format)
Slice 1
Slice 2
Slice 3
Field 1
Field 2
Field 3
Field 1
Field 2
Field 3
Field 1
Field 2
Field 3
TO
Climatology
(Time Averages)
Climatology 1
Field 1
Field 2
Field 3
Time

Task Parallelism

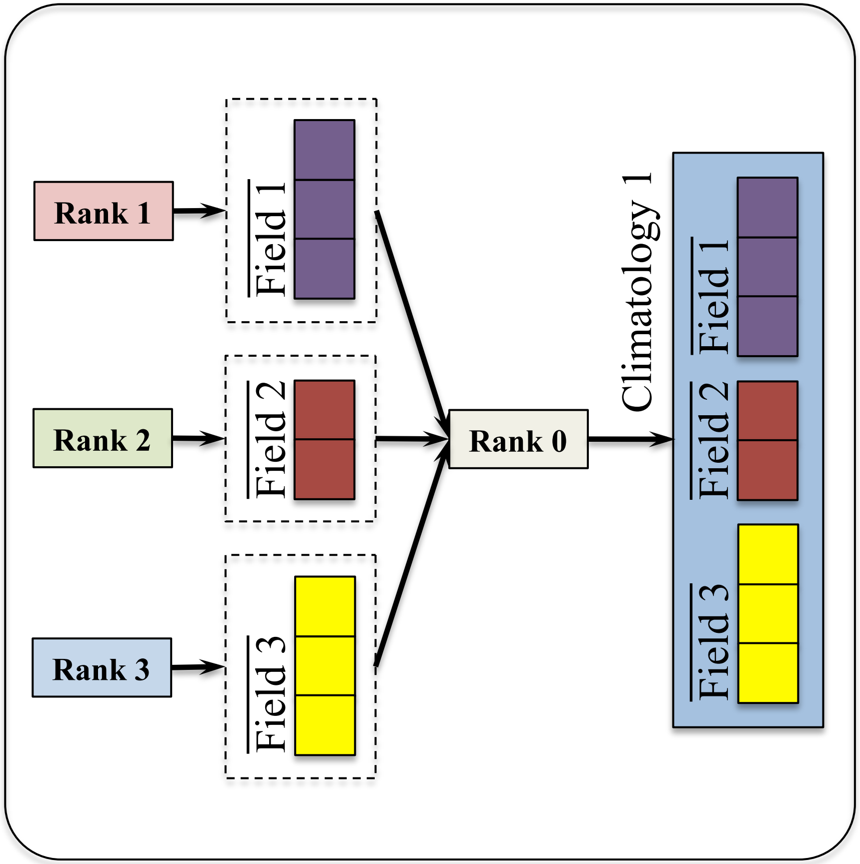


Task Parallelism

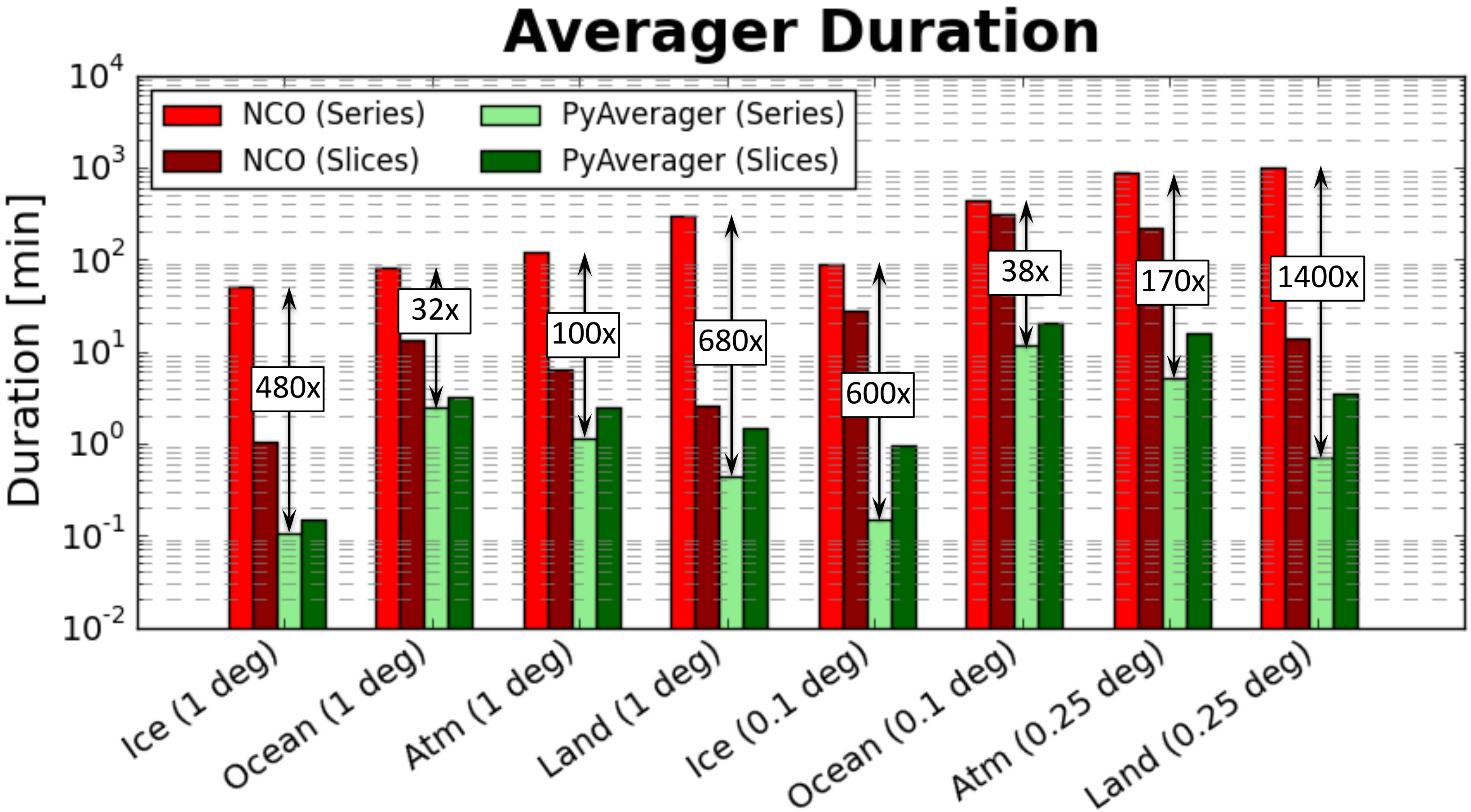


Task Parallelism

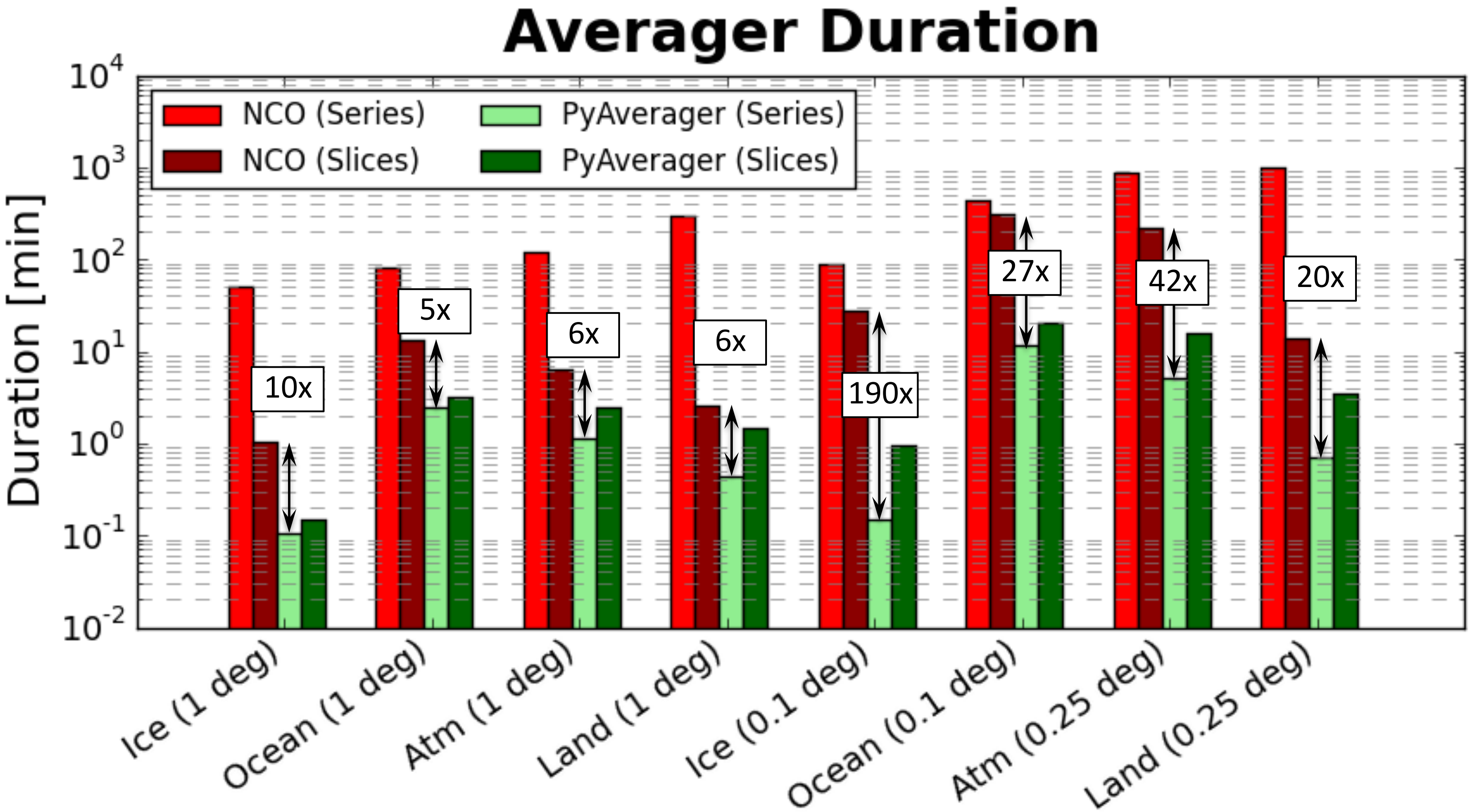
- Additional Parallelism over Climatologies
  - Seasons
  - Years
  - Months
- Climatologies Ordered:
  - Mos ➔ Seasons ➔ Yrs
- Each Climatology given its own MPI subcommunicator



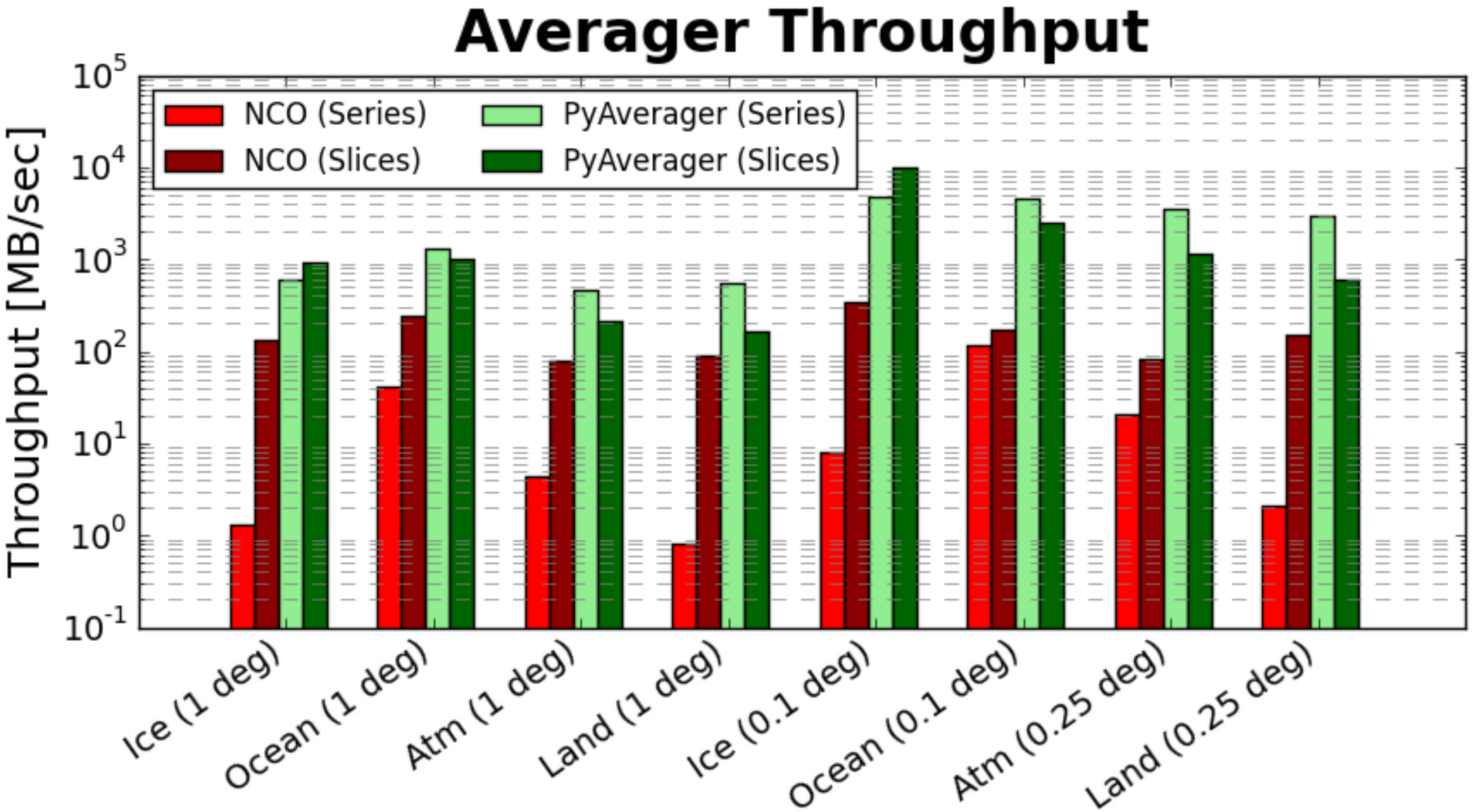
MPI Intercommunicator 1



- Incredible!
  - Overall 136x speedup!
  - 130x for Low-Resolution / 138x for High-Resolution



- Overall 27x speedup ("sum of all times")
  - 6x for Low-Resolution / 32x for High-Resolution



- Overall 27x speedup ("sum of all times")
  - 6x for Low-Resolution / 32x for High-Resolution

Conclusions & Future Work

Done!

- Began a new development program:
  - Minimally transformation / Maximal benefit
  - "Principle of Least Astonishment"
- New tools:
  - PyReshaper (Overall 12x Speedup)
  - PyAverager (Overall 27x Speedup)
  - Common Dependency: *ASAP Python Toolbox*
  - Available on GitHub:
    - https://github.com/NCAR-CISL-ASAP/

Yet to be done…

- Data Parallelism
  - Python-based NetCDF Parallel Write
  - Should improve scalability of tools
- New Parallel Publication Preparation Tool
  - CMIP Formatting Conversion ("CMOR")
  - In development

Thanks!

Thanks to the NSF, and special thanks to NCAR's CESM Development Team for all there help with testing and design.