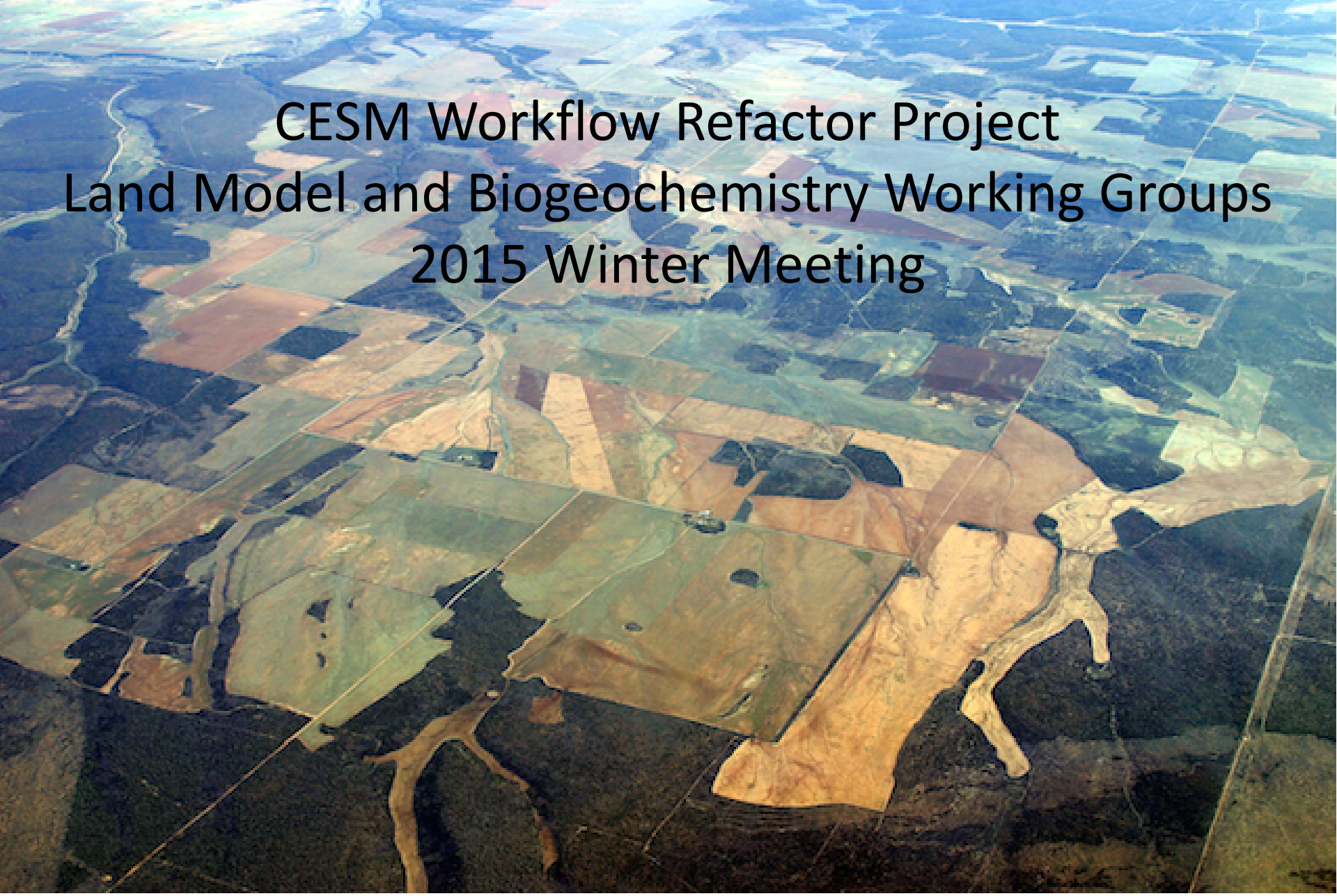


CESM Workflow Refactor Project
Land Model and Biogeochemistry Working Groups
2015 Winter Meeting

Alice Bertini Sheri Mickelson
CSEG & ASAP/CISL

CESM Workflow Refactor Project

Who's involved? Joint project between CSEG, CISL and CCP

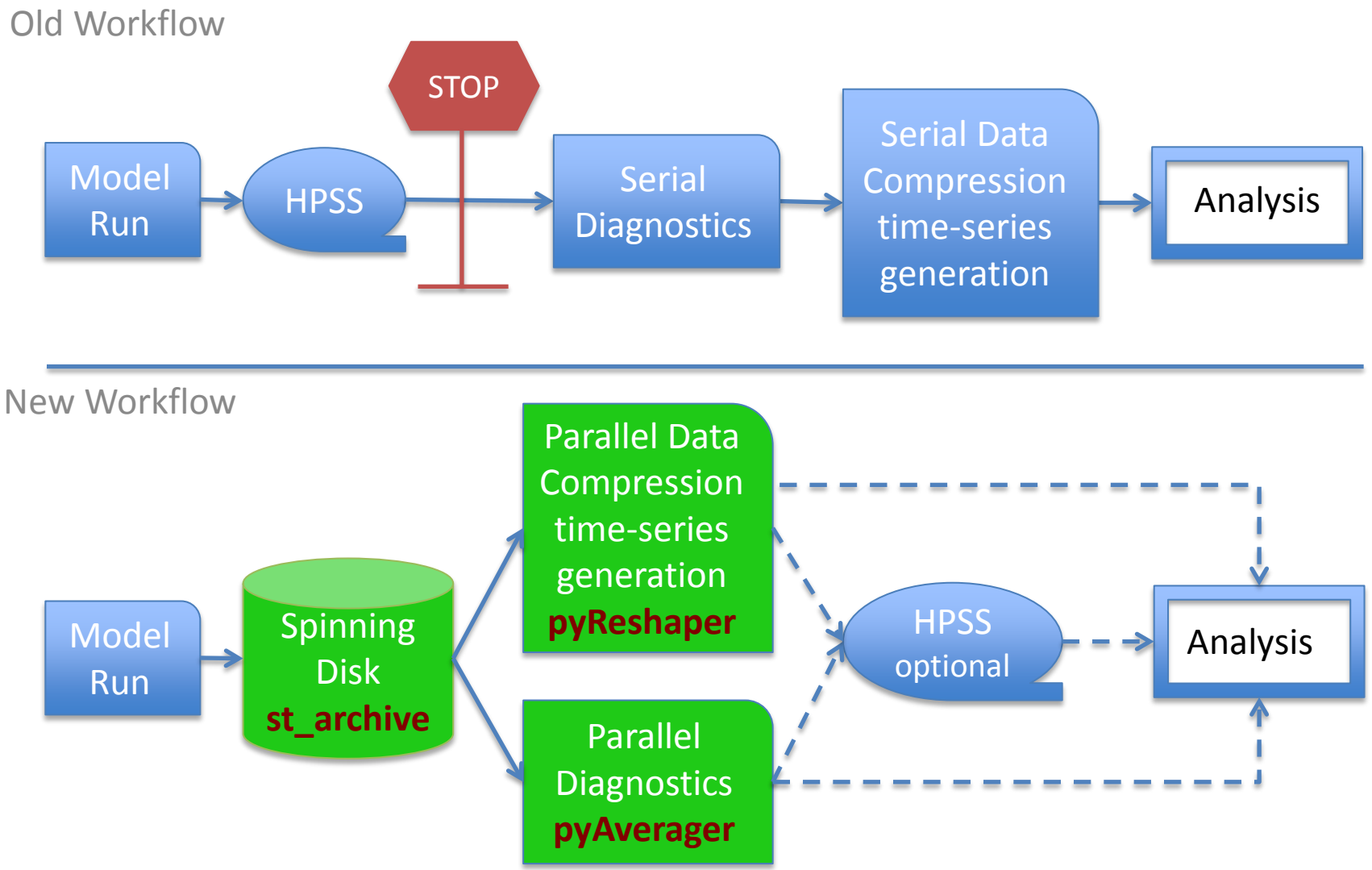
Goals? To create a new end to end workflow that enables scientists to get work done easier and faster

What we're looking to improve? Input data creation, archiving, model variable time-series generation, and post-processing

What is our process? Looking at current workflow functionality and performance and incrementally adding improvements that yield the most "bang for the buck"

CESM Workflow Refactor Project

Uses NCL, Matlib, XML, Python, and CESM scripts



CESM Script Modifications

Problems:

- The current CESM framework can not automate the time-series generation or diagnostic submission
- Existing framework is not flexible and wastes compute cycles

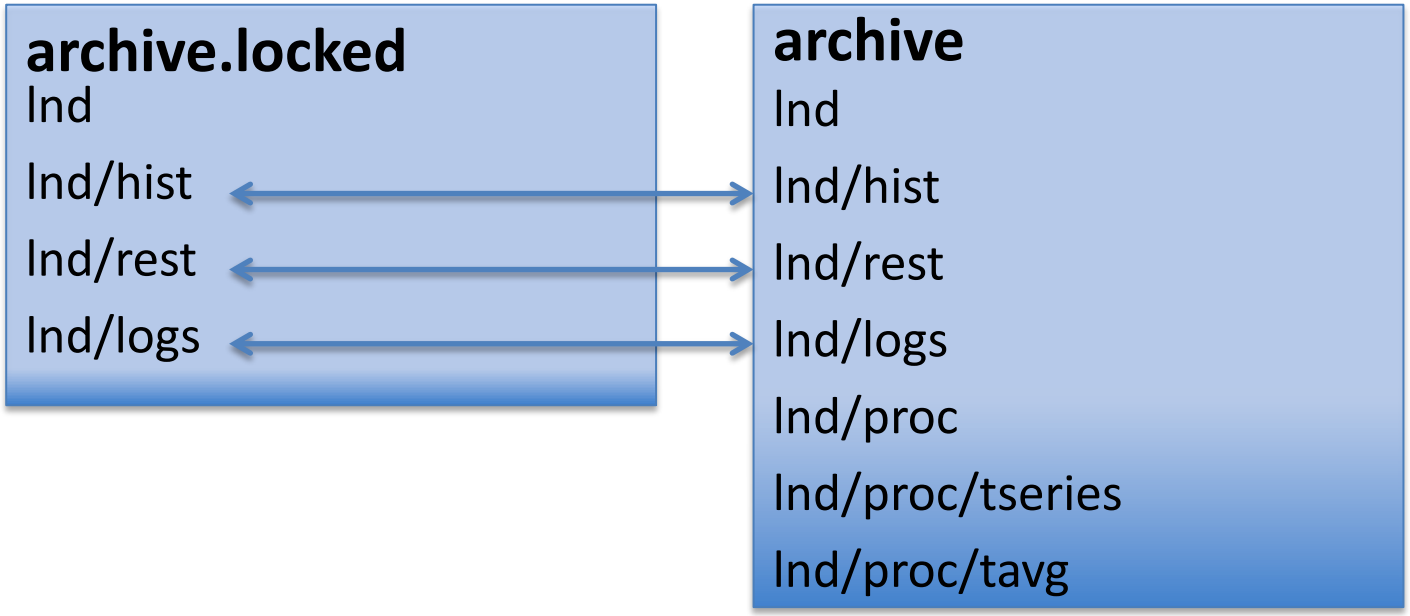
Solution:

- Automate post-processing tasks submitted as separate dependent jobs in the model run script
- Allow for the flexibility to submit these jobs with different node counts and stand-alone
- Refactor the short-term archive script to create a post-processing location on disk to allow for concurrent model run and post-processing tasks

Short-Term Archiver

What it does:

- At model run completion, copies or moves all files from the run directory into the archive directories on disk
- Retains a complete set of restart files in the run directory allowing for a new run job submission
- Controlled by XML
- Follows the CESM output file naming conventions



Data Compression and Time-Series Generation

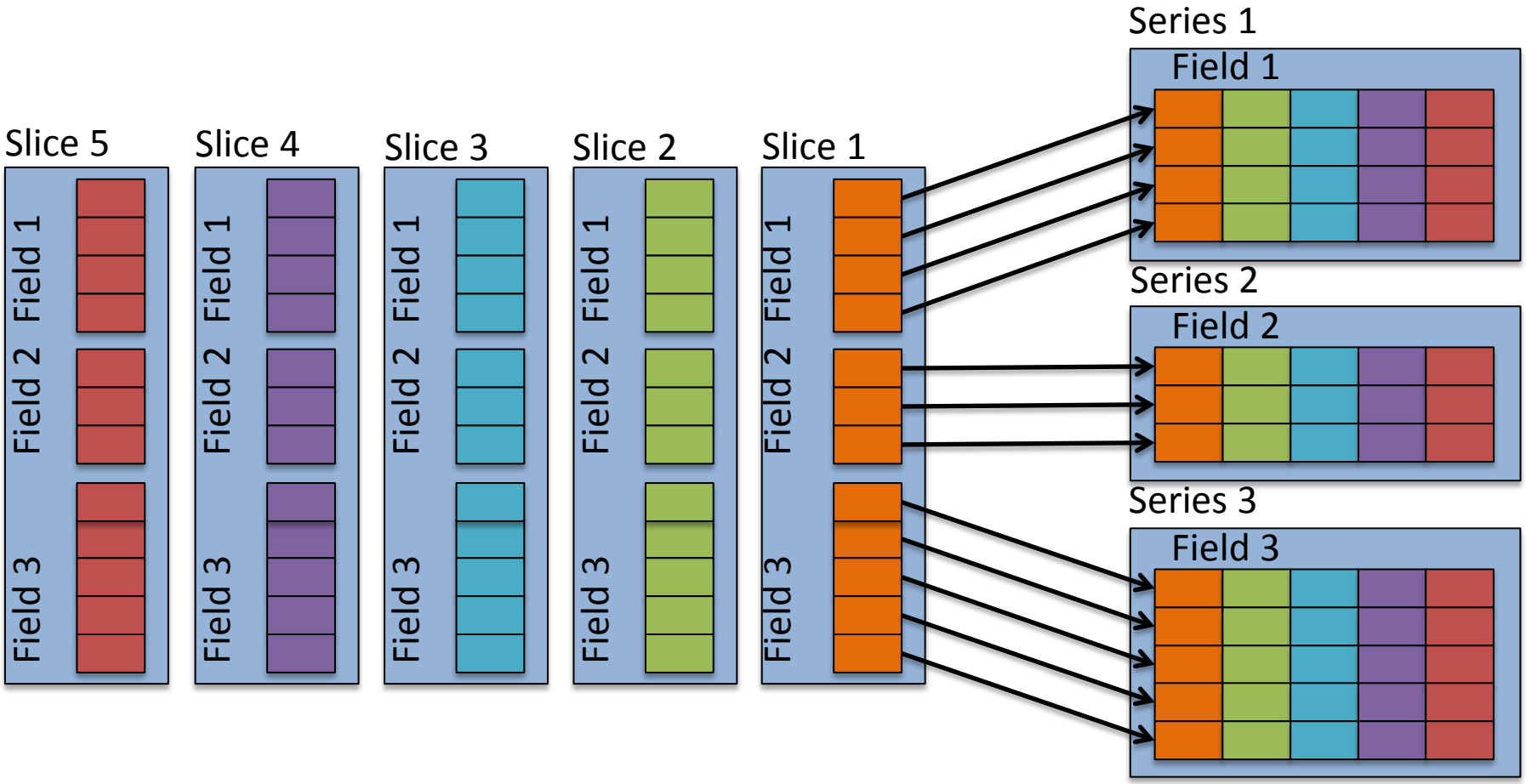
Problems:

- The current post-processing suite works in serial using NCO
- **CMIP5 post-processing required about as much wall-clock time to post-process data as actual model run time**

Solution:

- Light-weight parallel Python tool to do conversion in-line with the CESM run script called pyReshaper
- Works with CESM run environment, short-term archive and XML (cesm_tseries_generator.py)
- Supports NetCDF3, NetCDF4, and NetCDF4C

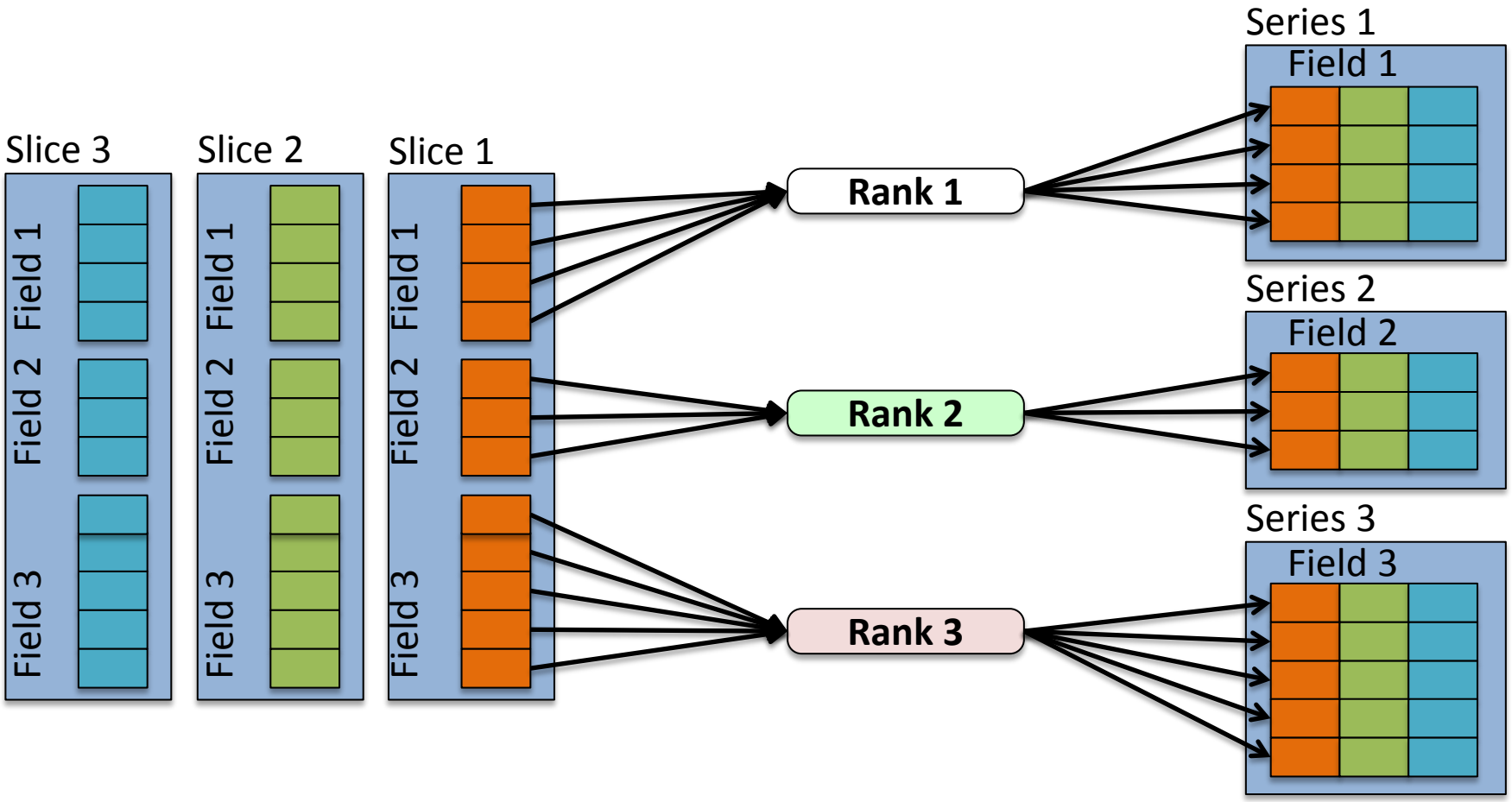
History Time-Slice to Time-Series Converter – Serial NCO



Task Parallelization Strategy

Each rank is responsible for writing one (or more) time-series variables to a file

pyReshaper



Time-Slice to Time-Series Conversion

PyReshaper Timing Statistics

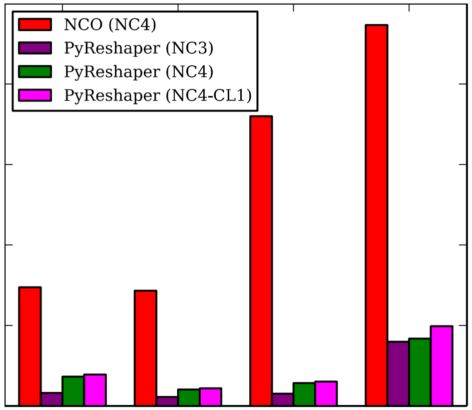
Existing Method (NCO)	Time (per MIP per Year)	Average Throughput (per run)
f09 x g16	225 minutes	1.85 MB/sec
ne120 x g16	478 minutes	4.85 MB/sec

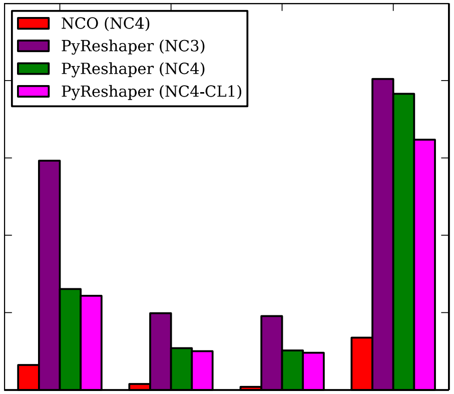
New Method (PyReshaper)	Time (per MIP per Year)	Average Throughput (per run)
f09 x g16	4 minutes	104 MB/sec
ne120 x g16	8 minutes	290 MB/sec

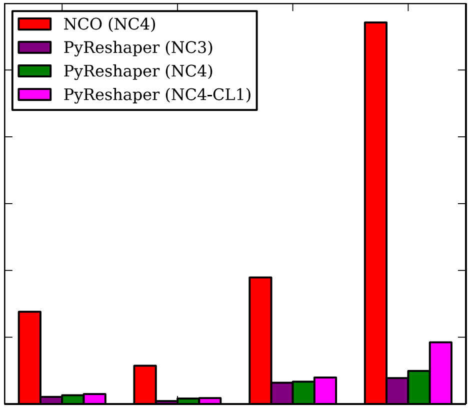
- Times include the approximate full time to convert all component data to NetCDF4
- Conversions were ran on Yellowstone using 4 nodes/4 cores (16 cores total)
- We can expect a 2X increase in throughput if we double core counts for low-resolution data
- We can expect a 3X increase in throughput if we double core counts for high-resolution data

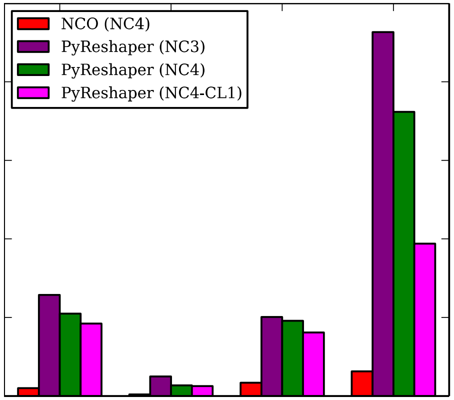
PyReshaper Plots

Time to convert 10 years of CESM data from time slice to time series.









Tasks

Completed and available in the CESM Developer Repository:

- New CESM Short-Term Archiving capability to local disk (st_archive) allows model to continue running concurrently with post-processing
- A Parallel Time-Series File Generator and File Compression (pyReshaper and cesm_tseries_generator.py)

Currently Working On:

- Bringing diagnostics and analysis capabilities into the CESM case environment and run scripts
- Automating the submission of the diagnostic packages
- Modifying diagnostic packages to be more extensible, robust, and scalable. (pyAverager)
- Archiving run metadata to the experiment database directly from the case directory for provenance. (archive_metadata)

Diagnostic Packages

Problems:

- Runs either serially or with limited parallelization
- Not easily integrated into the CESM run environment
- Not easily extensible
- Hard to run with big data
- Only works with history time-slice data

Solutions for the Diagnostic Packages

Reworking each package following these steps :

1. Integrate diagnostics into the CESM end-to-end automated workflow, while still maintaining stand alone capabilities
2. Diagnostic environment defined in XML
3. Creating climatology files with the PyAverager
4. Task parallelizing existing plotting scripts
5. Works with either time-slice or time-series files

Diagnostics Integration

- Brings in the CESM case and diagnostic settings as a Python data structure
- Calls the parallel pyAverager
- Calls NCL plotting scripts in parallel
- Converts ps plots in parallel
- A directory that contains the html file and plots is created

PyAverager Details

A light weight custom Python averaging tool

- Parallelizes over averages and variables
- Works on time slice and time series data

Types of averages it can compute:

- Temporal Averaging
  - Seasonal, Yearly, Annual, Monthly (weighted optional)

Looking to also compute:

- Zonal Averaging
- Variance
- Across ensembles

Time Averaging Options

- NCO (serial)
  - Controlled by a top level csh script that calls NCO operators to calculate averages.
- Swift (limited task parallel)
  - Averages are calculated in parallel calling the NCO operators
- PyAverager (task parallel)
  - New method written in Python that task parallelizes over variables and averages.

Each method was operated on both time slice and time series files

Time Averaging Comparisons

Datasets Used

Component	Res	Size (GB)	# of Vars
CAM FV	1 .0	28	139
CAM SE	1.0	30	148
CAM SE	0.25	1055	214
CICE	1.0	8/4	137
CICE	0.1	556/42	132
CLM	**1.0**	**10**	**310**
CLM	**0.25**	**113**	**163**
POP	1.0	190	170
POP	0.1	3113	87

* All dataset contain 10 years of both monthly time slice and time series files

Types of time averages computed

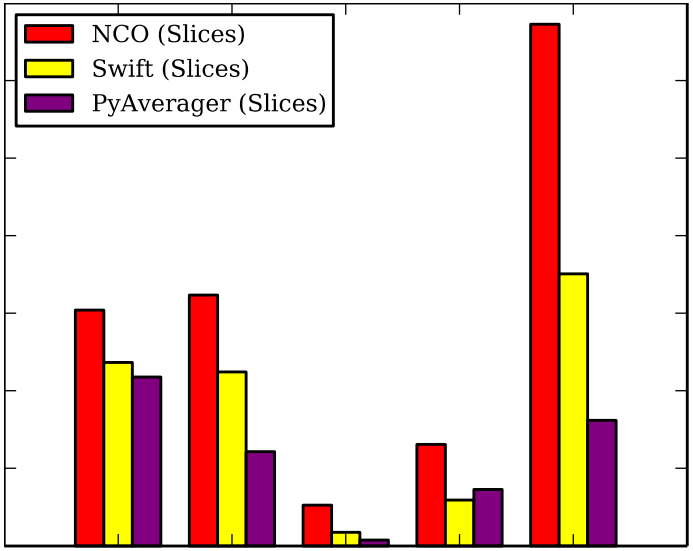
CAM & CLM

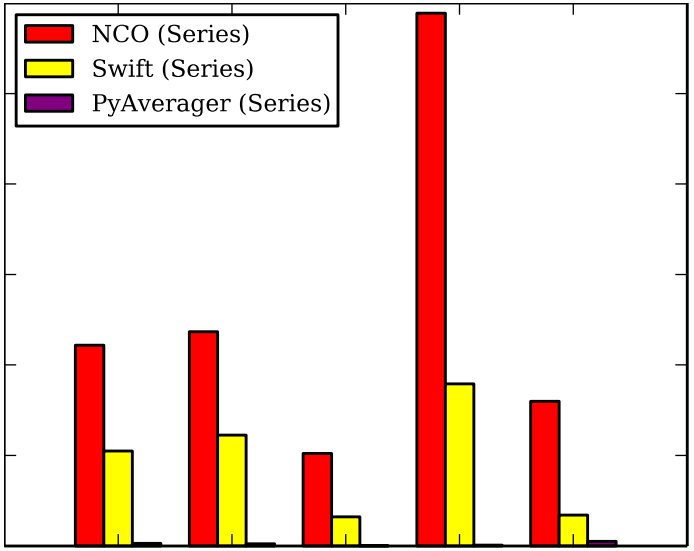
- **Seasonal Averages**
  - **ANN,DJF,MAM,JJA,SON**
- **Monthly Averages**
  - **One average per month**
- **17 Averages Total**

POP & CICE

- Yearly Averages
  - One average per year
- 10 Averages Total

Low Resolution Timings
Original method vs. Swift vs. pyAverager



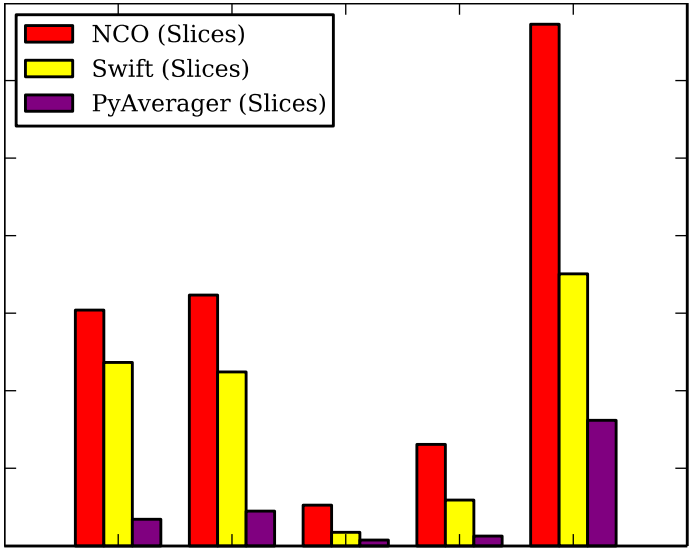


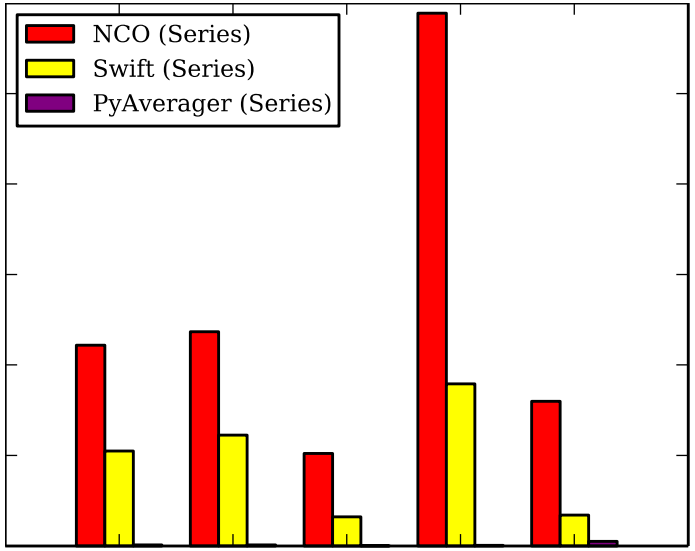
(min)	CAM FV	CAM SE	CICE	**CLM**	POP
NCO	6	7	1	**3**	14
SWIFT	5	5	0.4	**1.2**	7
pyAve	4	3	0.2	**1.5**	3

(min)	CAM FV	CAM SE	CICE	**CLM**	POP
NCO	111	118	51	**295**	80
SWIFT	53	61	16	**90**	17
pyAve	2	1	0.1	**0.4**	3

Low Resolution Timings

(New PyAverager timings for CAM & CLM that use dependency averaging for seasons)

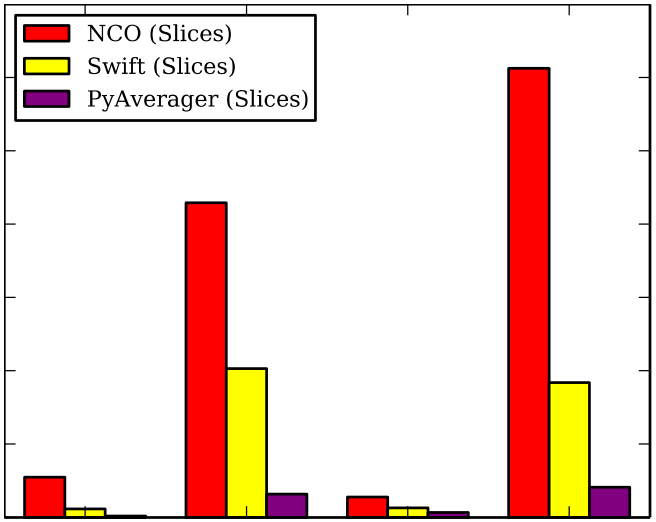


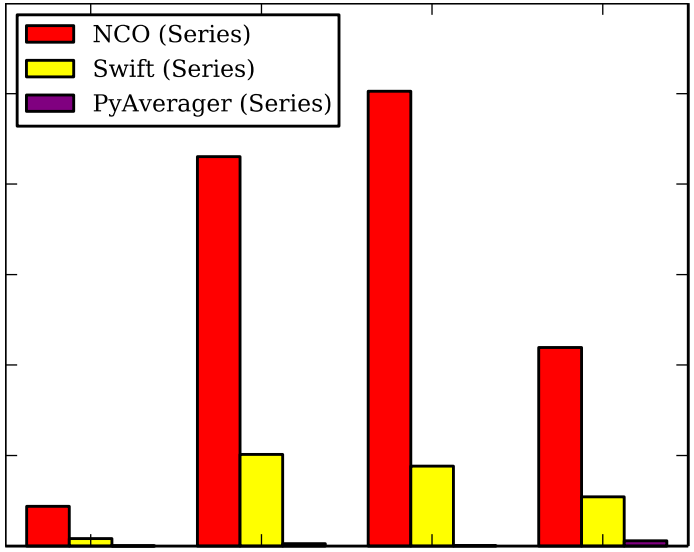


(min)	CAM FV	CAM SE	CICE	**CLM**	POP
NCO	6	7	1	**3**	14
SWIFT	5	5	0.4	**1.2**	7
pyAve	0.7	1	0.2	**0.25**	3

(min)	CAM FV	CAM SE	CICE	**CLM**	POP
NCO	111	118	51	**295**	80
SWIFT	53	61	16	**90**	17
pyAve	0.6	0.6	0.1	**0.3**	3

High Resolution Timings
Original method vs. Swift vs. pyAverager





(min)	CICE	CAM	**CLM**	POP
NCO	27	215	**14**	306
SWIFT	6	102	**7**	92
pyAve	1	16	**4**	21

(min)	CICE	CAM	**CLM**	POP
NCO	88	861	**1005**	439
SWIFT	16	203	**177**	109
pyAve	0.2	5	**0.7**	12

Computational Resources Used For Timing Comparisons

File Type	NCO-Slice	NCO-Series	Swift-Slice	Swift-Series	PyAvg-Slice	PyAvg-Series
POP-1.0	Y/1	Y/1	G/16	G/16	Y/160	Y/160
CLM-1.0	**Y/1**	**Y/1**	**G/16**	**G/16**	**Y/160**	**Y/160**
CICE-1.0	Y/1	Y/1	G/16	G/16	Y/160	Y/160
CAMSE-1.0	Y/1	Y/1	G/16	G/16	Y/160	Y/160
CAMFV-1.0	Y/1	Y/1	G/16	G/16	Y/160	Y/160
POP-0.1	BM/1	Y/1	BM/4	G/16	G/40	G/40
CLM-0.25	**GP/1**	**Y/1**	**G/16**	**G/16**	**Y/160**	**Y/160**
CICE-0.1	GP/1	Y/1	G/16	G/16	Y/160	Y160
CAM-0.25	GP/1	Y/1	G/16	G/16	Y/160	Y/160

Machine/Cores
Y=Yellowstone G=Geyser GP=GPGPU BM=BigMem

PyAverager Ver 0.1.0 available to friendly users

- Download the package from
  https://proxy.subversion.ucar.edu/pubasap/pyAverager/tags/v0.1.0/

- Depends only on NumPy, mpi4py, and PyNIO
  - Dependencies already installed on
    yellowstone, mira, edison, coming soon on goldbach
- Contains README's and Doxygen documentation
- Several examples in the examples directory
- Questions/Feedback can be sent to mickelso@ucar.edu

CSEG Support

- CESM Users Guide updates
- XML modifications via existing tools
- DiscussCESM bulletin board forums
- Coordination with LMWG
  - Diagnostics packages
- Coordination with CISL
  - Parallel Python tools

Continued Work

- Extending the flexibility of the run scripts
- Creating more extensible working environments
- Providing solutions to increase scalability and automation within the workflow
  - Including other CMIP workflow tools
- Adding testing into the current post-process workflow

Questions?

CESM workflow refactor team

- Ben Andre
- Alice Bertini
- John Dennis
- Jim Edwards
- Mary Haley
- Jean-Francois Lamarque
- Michael Levy
- Sheri Mickelson
- Kevin Paul
- Sean Santos
- Jay Shollenberger
- Gary Strand
- Mariana Vertenstein

https://www2.cisl.ucar.edu/tdd/asap/parallel-python-tools-post-processing-climate-data