# CESM Load Balancing Development: Python Scripts for Workflow

*Thomas Johnson III (Elizabeth City State University) and Soudeh Kamali (University of Wyoming) +Sheri Mickelson, Brian Dobbins, John Dennis*

July 30th, 2020

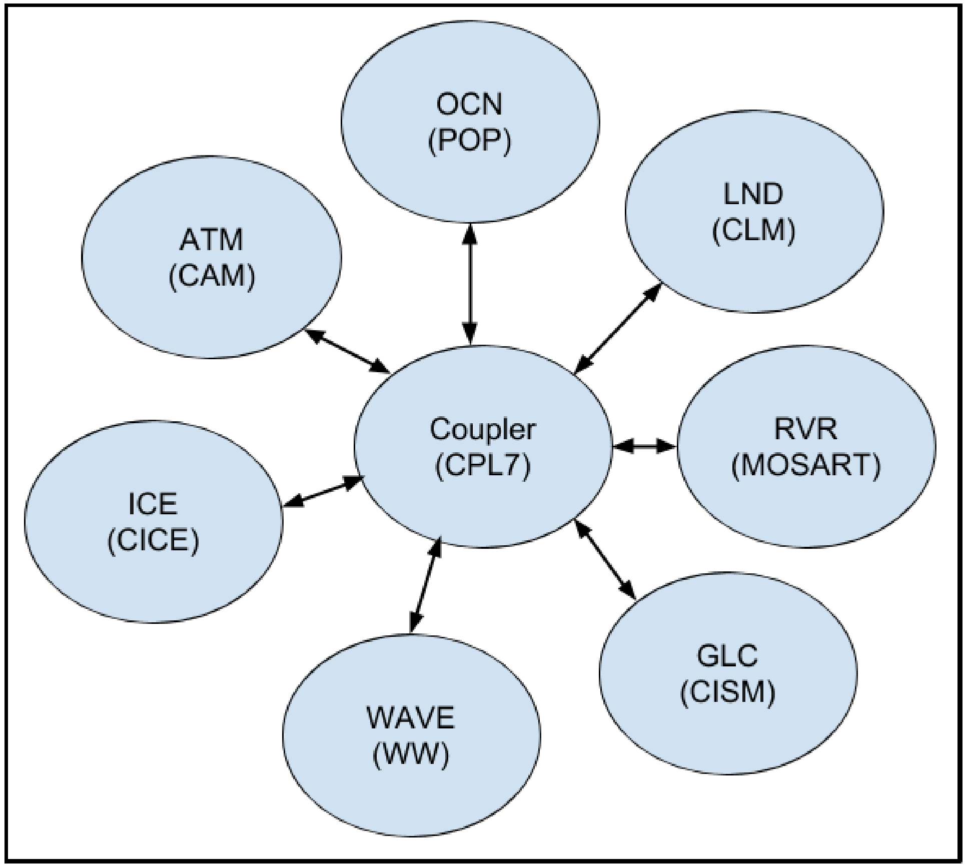
This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

## What is Load Balancing?

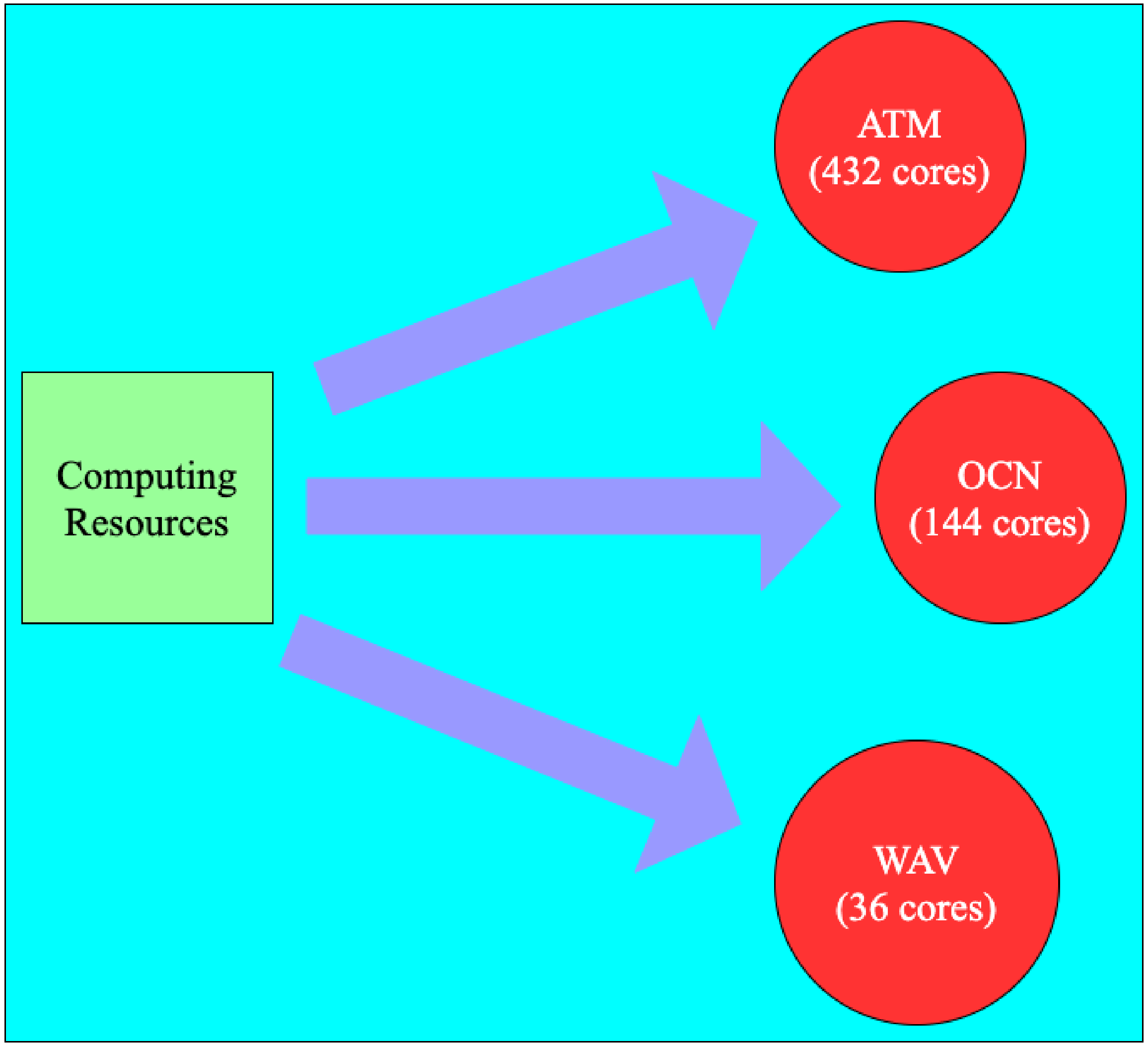
- Load balancing is a process of actively managing resources.
- Applied by determining which tasks should receive a given amount of finite resources.

## What is CESM?

- Stands for Community Earth System Model.
- Software for simulating the weather and climate systems of the Earth.
- Critical tool for climate studies.



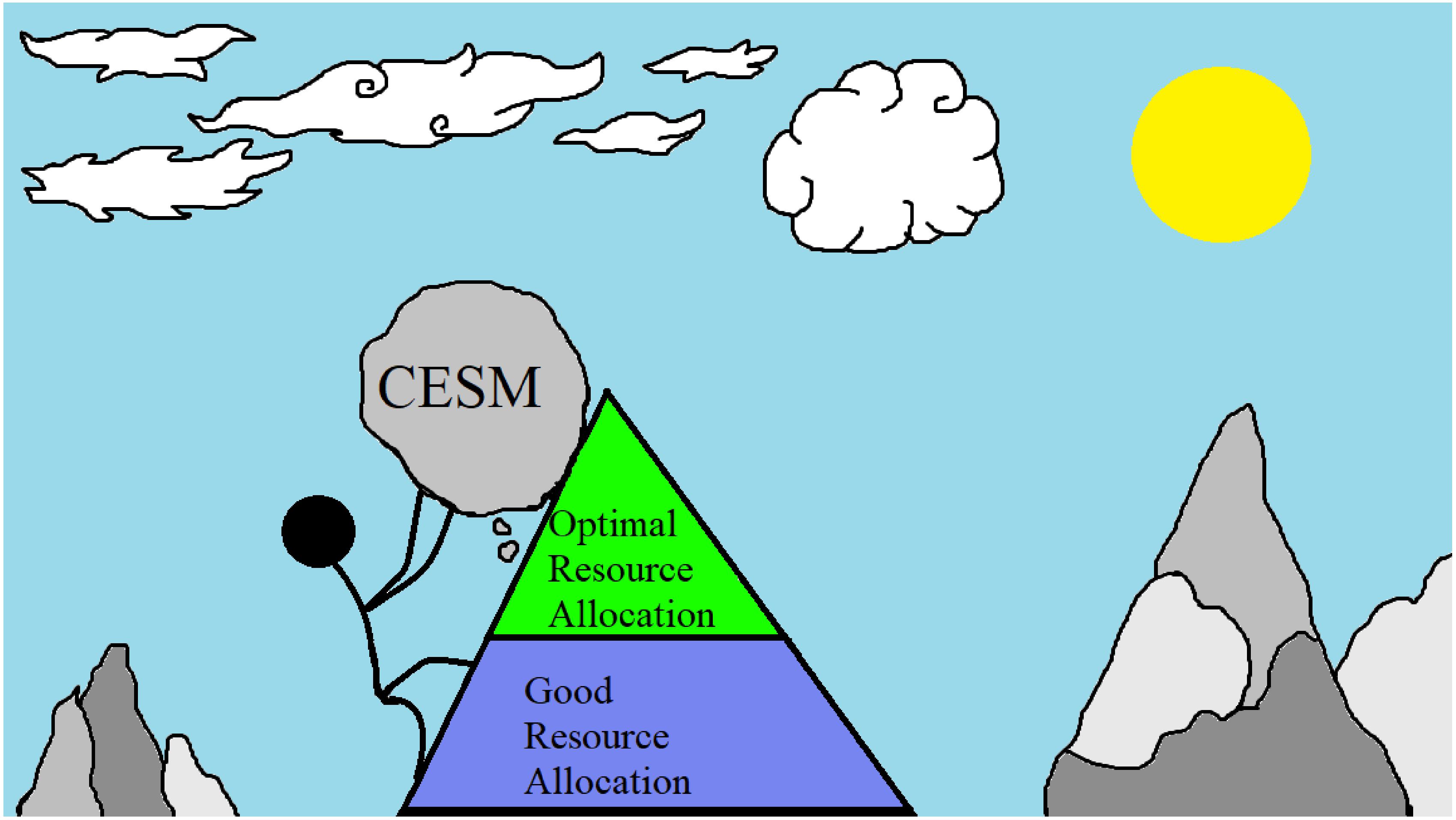
# Why Load Balancing for CESM?



- Building and running CESM models can be resource expensive.
- Provide guidance for running CESM for newcomers.
- Enable CESM to be run across multiple environments effectively.

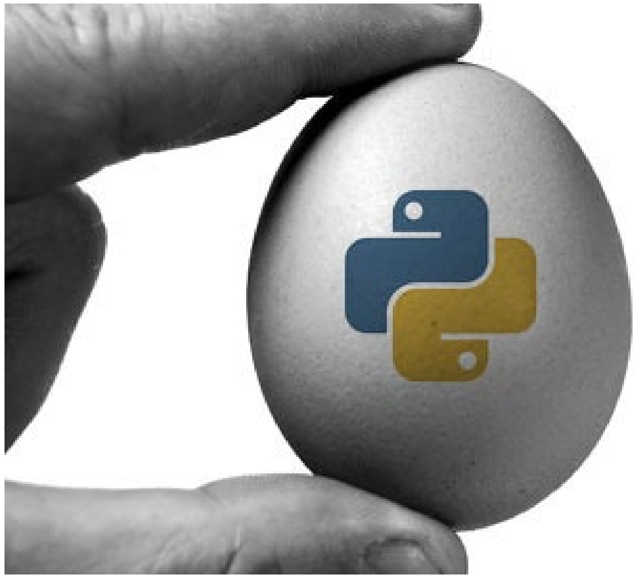
## Current Status for CESM

- Manual application of load balancing and running CESM models.
- Users must guess the best scheme for allocating resources.



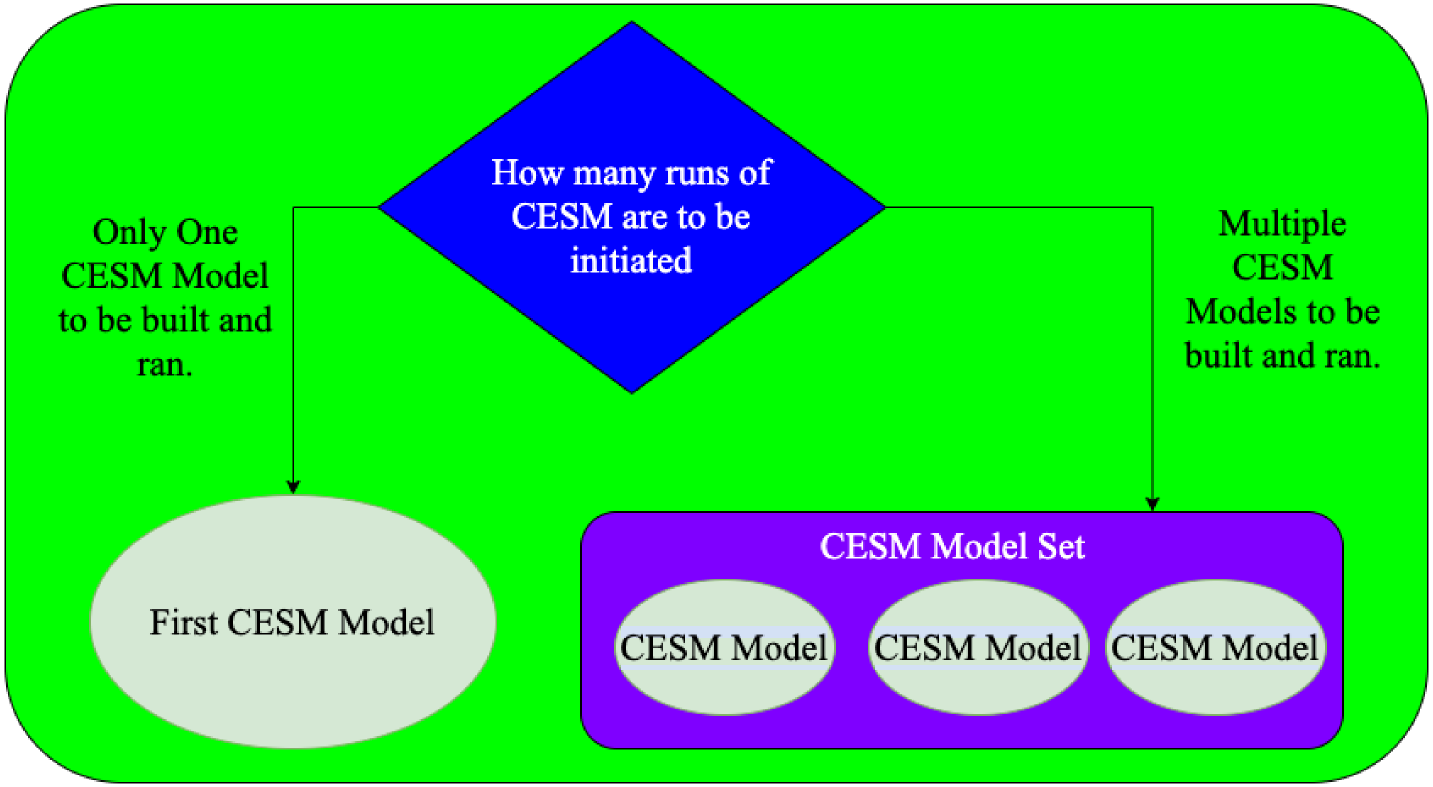
## Resources Utilized

- Software components: Python, Bash, CESM
- Hardware Components: Cheyenne



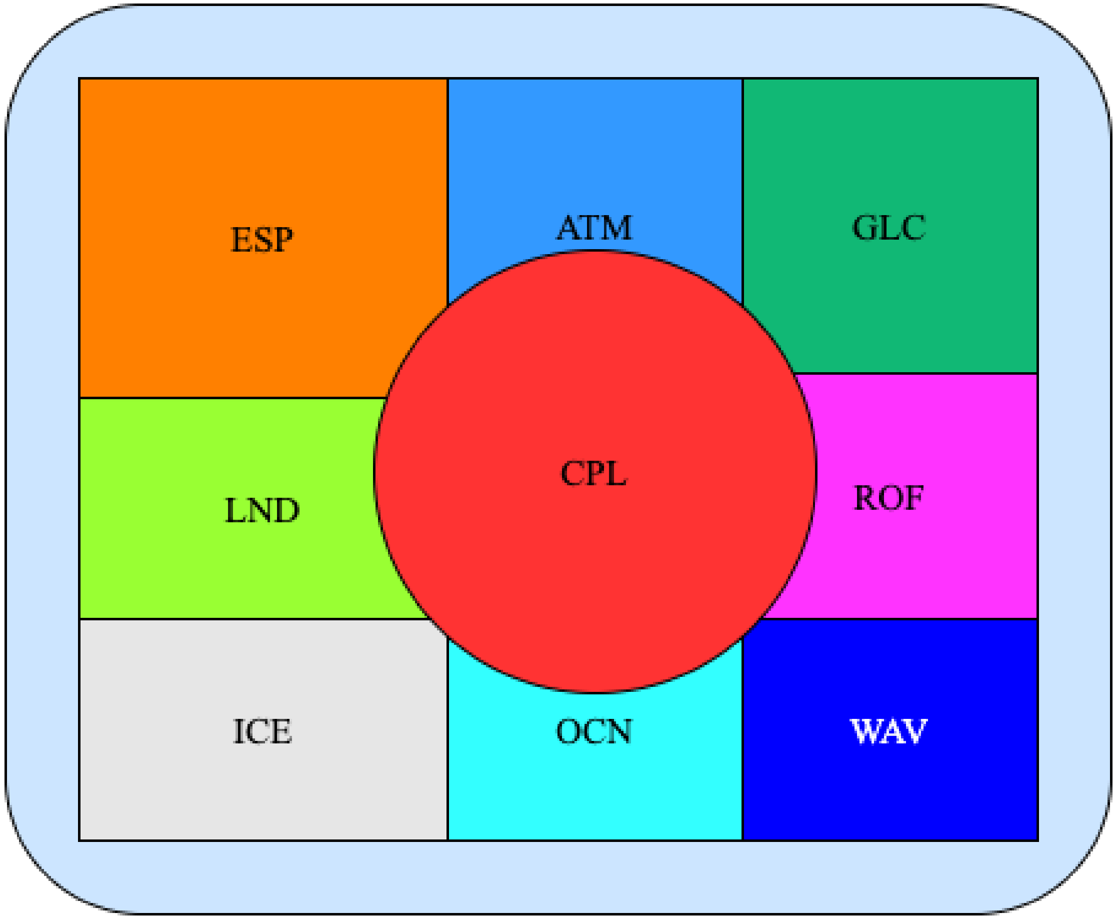
# Initiating

- Specify the max number of tasks in the command line.
- Specify the number of CESM model runs.


How many runs of CESM are to be initiated
Only One CESM Model to be built and ran.
Multiple CESM Models to be built and ran.
First CESM Model
CESM Model Set
CESM Model
CESM Model
CESM Model

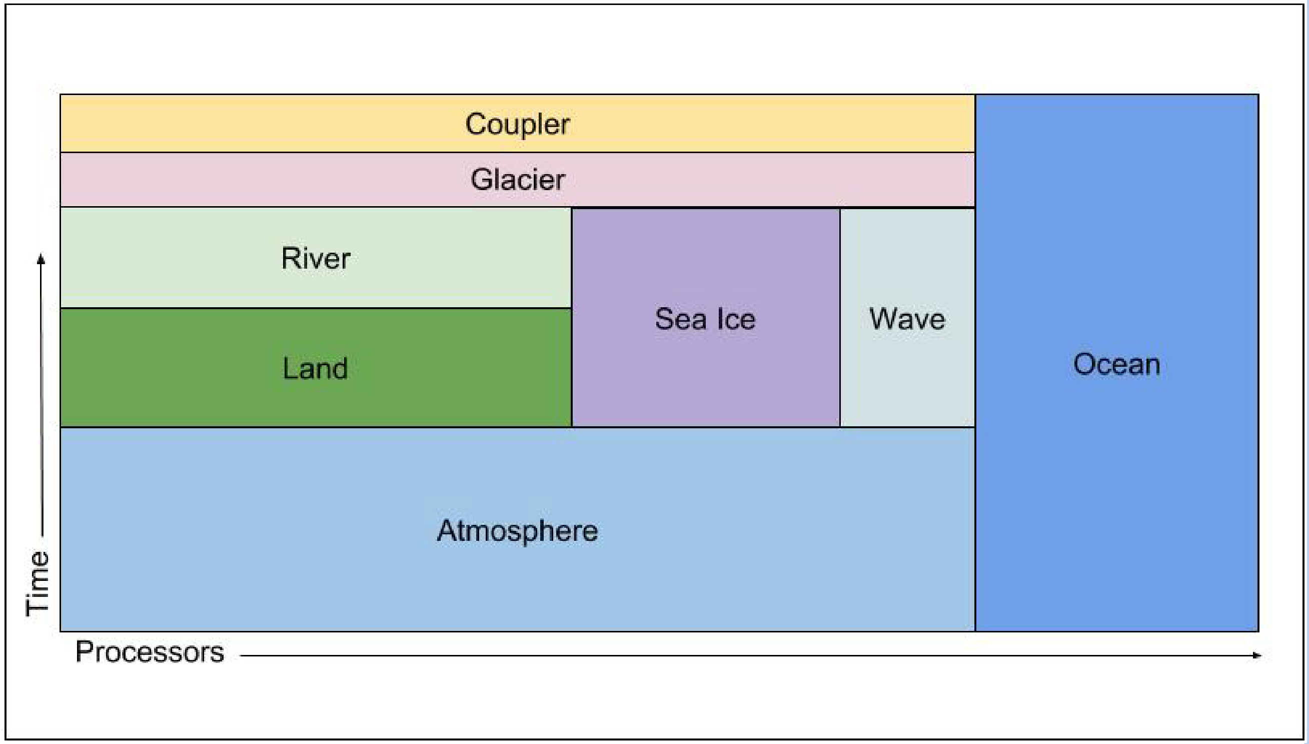
## Core Allocation

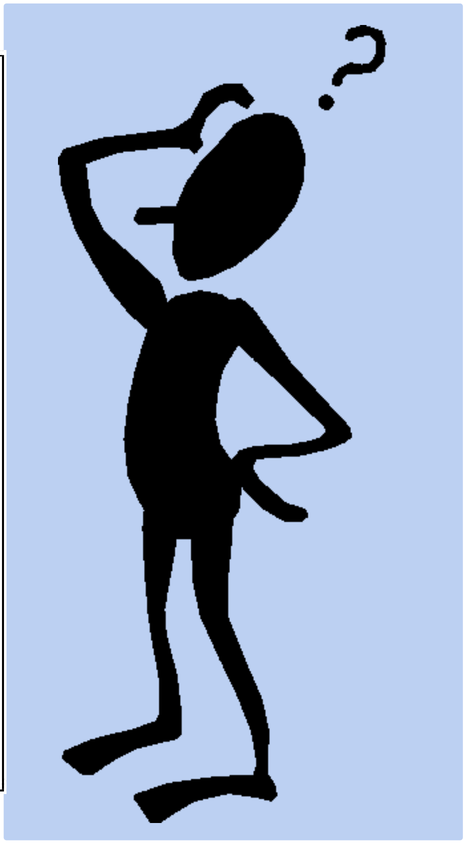
- Cores are allocated for each component, with WAV capped to preserve efficiency.
- CESM models are built and run.



# Load Balancer Called

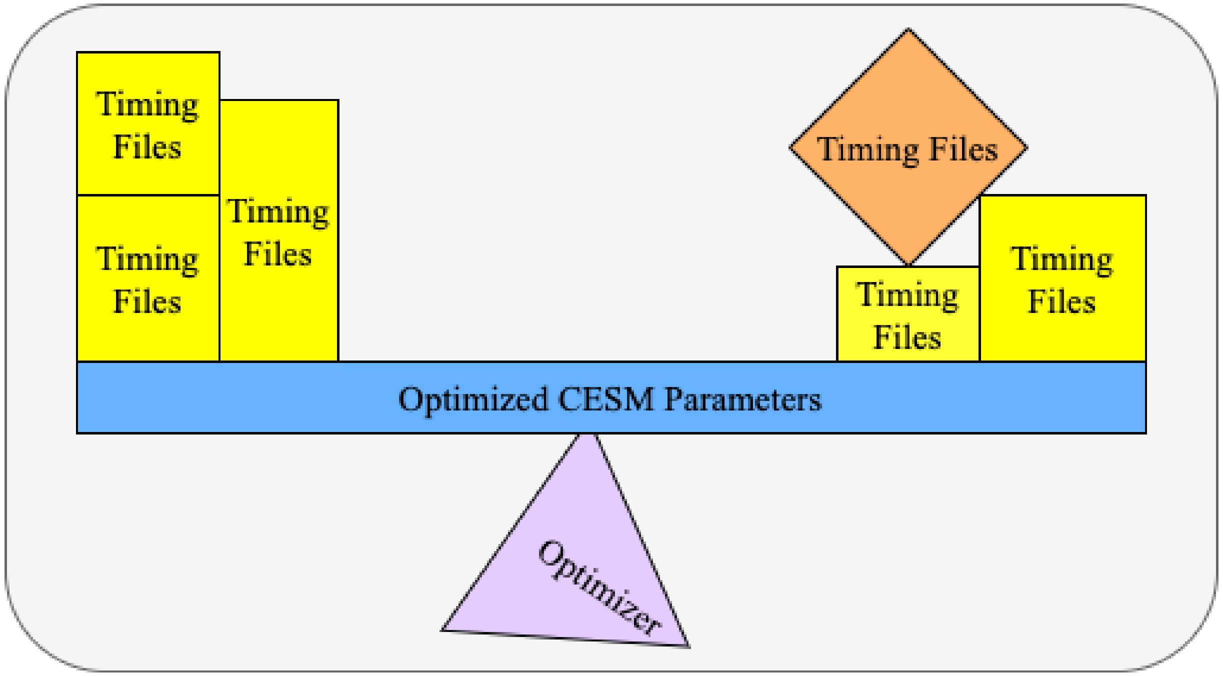
- User decides to run load balancing software on outputted timing files.
- Load balancing software processes the timing files.





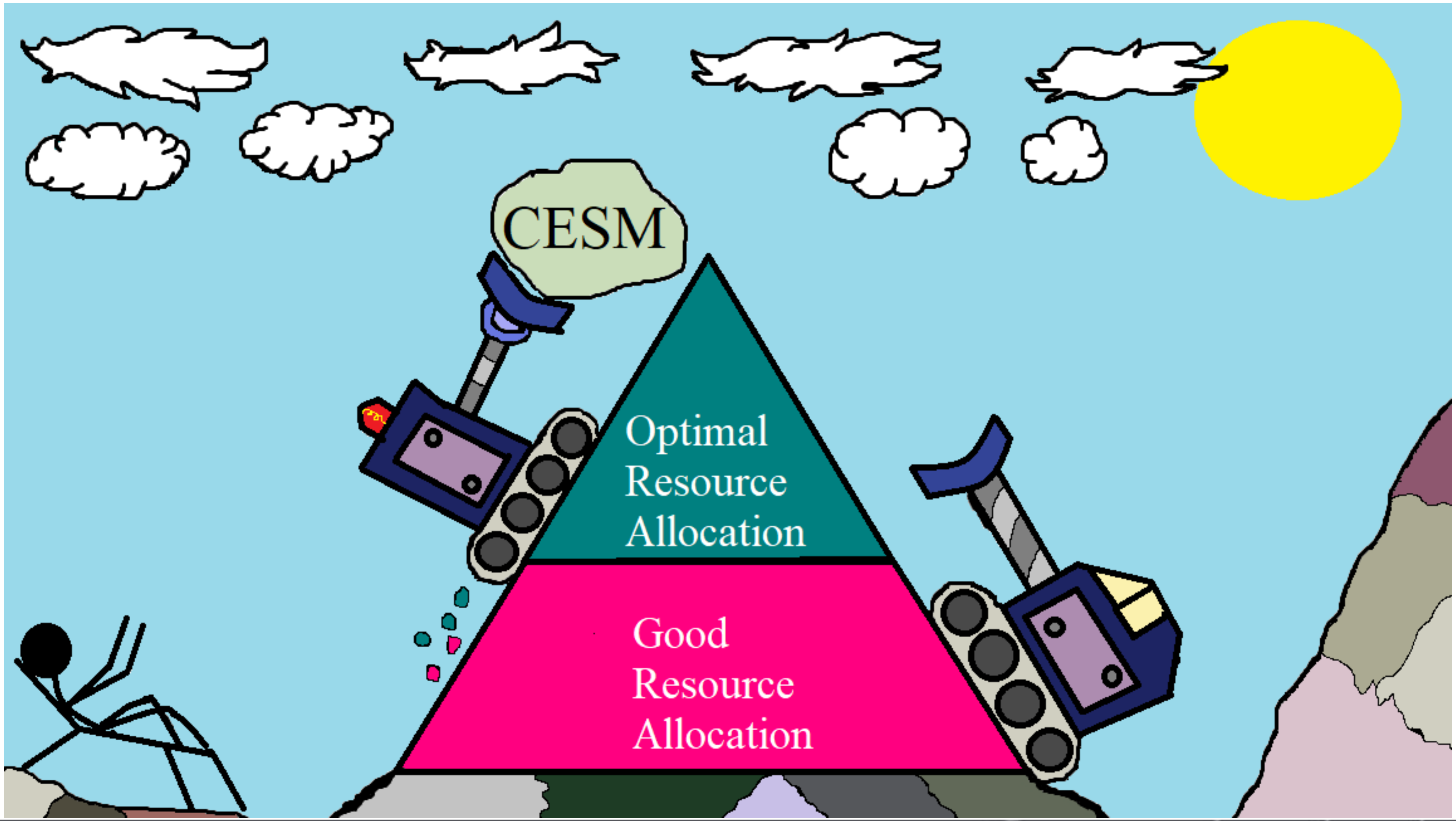
# Load Balancer Process

- Load balancing software produces optimized values based on provided timing files.
- Said optimized values are stored into a JSON file.
- JSON files can be turned into a python dictionary to be processed by CESM.



## Automated Load Balancing Continued

- The user now has optimized values to build and run future CESM models with.
- Start to finish in one process.



## Command Line Input

- An Example of Running the CESM Load Balancing Code:
  - % python cesm_allocation.py 288 --compset_designation B1850 --sim_time_designation 2 --sim_time_unit ndays

## ATM Component Relative Seconds Per Model Day Statistics

- B1850 2 degree for ATM and 1 degree for OCN run Statistics.

| Average Times of Runs On Different Number of Days Simulation Settings of ATM Component | | | | |
|---|---|---|---|---|
| Number of Days Simulated | 2 Days | 5 Days | 7 Days | 12 Days |
| Relative Difference of Seconds Per Model Day to 2 Simulated Days | 0% | 0.61% | 0.70% | 0.78% |
| Absolute Seconds Per Model Day | 40.43 | 40.18 | 40.15 | 40.12 |

## OCN Component Relative Seconds Per Model Day Statistics

- B1850 2 degree for ATM and 1 degree for OCN run Statistics.

| Average Times of Runs On Different Number of Days Simulation Settings of OCN Component | | | | |
|---|---|---|---|---|
| Number of Days Simulated | 2 Days | 5 Days | 7 Days | 12 Days |
| Relative Difference of Seconds Per Model Day to 2 Simulated Days | 0% | 3.9% | 4.5% | 5.7% |
| Absolute Seconds Per Model Day | 55.06 | 52.91 | 52.59 | 51.90 |

# ATM Component Relative Seconds Per Model Day Statistics for B1850 1 Degree ATM 1 Degree OCN

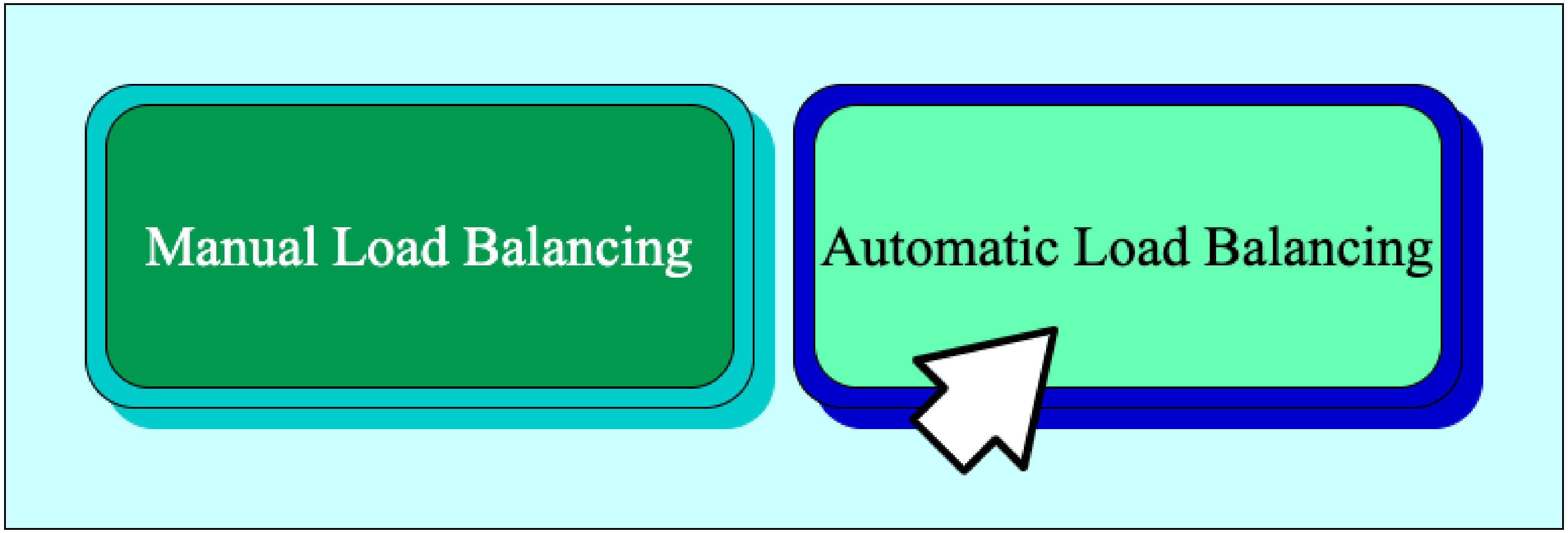
- B1850 1 degree for ATM and 1 degree for OCN run Statistics.

| Time Statistics of Runs On 5 Day Simulation Settings of ATM Component | |
|---|---|
| Average Seconds Per Model Day | 178.41 |
| Maximum Seconds Per Model Day | 178.50 |
| Minimum Seconds Per Model Day | 178.34 |

| Time Statistics of Runs On 5 Day Simulation Settings of OCN Component | |
|---|---|
| Average Seconds Per Model Day | 52.93 |
| Maximum Seconds Per Model Day | 53.23 |
| Minimum Seconds Per Model Day | 52.60 |

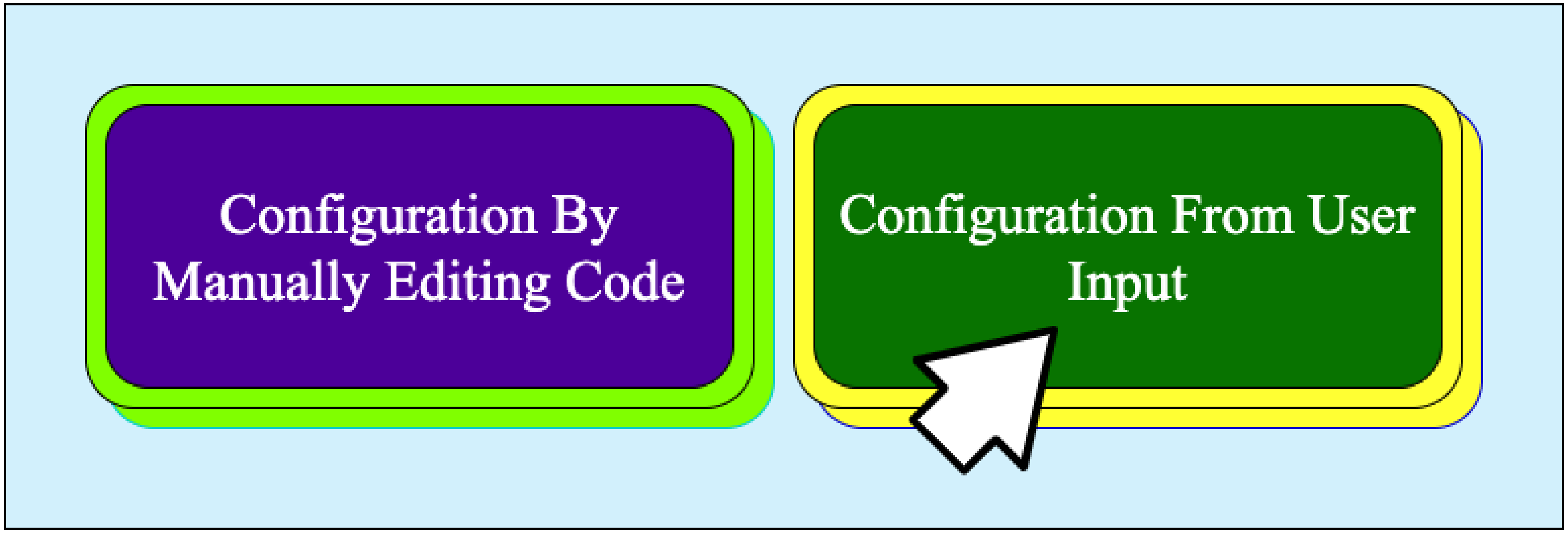
## Implemented Features

- Build CESM models and utilize load balancer in one process.
- Ability to select compsets.
- Automatically scales out components.



## Added Features

- JSON files for storing and loading CESM parameters.
- Restriction for the WAV component to prevent inefficiency from excess core usage.
- Specify the number of CESM models to be built and ran concurrently.


Configuration By Manually Editing Code
Configuration From User Input

## Future Work

- Implementing more options for scaling.
- Exploring more options for configuration of building CESM models.

## Acknowledgements

- Thanks to the SIParCS Program and NCAR for providing this opportunity.
- Thanks to Soudeh Kamali for being an excellent SIParCS partner and supportive colleague and extracting data as well as acquiring summary statistics.
- Thanks to Sheri Mickelson, Brian Dobbins, and John Dennis for being incredible SIParCS mentors offering guidance and knowledge.

## References

- Load Balancing Software by Jim Edwards et al. DOI:http://dx.doi.org/10.5065/WE0D-9K91. URL: https://github.com/ESMCI/cime.
- CIME Load Balancing Tool by Sheri Mickelson, updated by Yuri Alekseev (2017): https://github.com/ESMCI/cime/tree/master/tools/load_balancing_tool
- Image citations within the alt text.

## Questions

- Thomas Hilton Johnson III
- Email: Thomash.johnson261@gmail.com