Docker MITgcm-ECCO Project Overview

**Problem Statement:** Currently, researchers, scientists, and other members of the MITgcm climate science community experience complexity with, and time-intensive involvement in configuring access to computing resources, and establishing environments for building, running, and visualizing MITgcm model data. Many users find it inconvenient and time-consuming to run through many hurdles necessary to set up environments that can be used for their research such as determining what software modules and libraries to install, learning new skills such as shell scripting, and navigating Linux environments, and others. Additionally, it can be meddlesome for users to perform the post-processing/data science tasks of writing out, importing, wrangling, and visualizing model output data using numerous analysis routines that exist within the MITgcm-ECCO community. Furthermore, many of these models can take an unreasonable amount of time to build and run-on host architecture and could benefit from a more lightweight execution environment. where these tasks can be run in isolation.

**Need Statement**

Given these limitations and inconveniences with building, running, and interpreting MITgcm models, there exists a need for the development of portable units of software that can be used to build and run models in an isolated environment where minimal prior knowledge of software module, library, and other dependency installation are required. These portable units of software would ideally come equipped with a pre-installed operating system, along with necessary software modules, libraries, and dependencies needed for the end user to perform all tasks in the MITgcm-ECCO workflow. This would effectively reduce the need for the end user to determine what software tools to install, and certain skill sets in computing (like shell scripting) to acquire, along with the need to comprehend analysis routines for post-processing before building, running, and interpreting results from models. These isolated environments would ideally function as a one-stop-shop for all MITgcm-ECCO workflow tasks and as lightweight virtual machines that can run any host machine (with some fundamental limitations) regardless of that machine’s architecture, operating system, etc.

**Project Objectives**

To address this project’s need statement, project objectives have been documented in two categories, namely, **SMART Goals** (concrete goals that are smaller in scale and more achievable) and **Stretch Goals** (ambitious goals that expand on or provide a different functionality than the concrete goals).

**SMART Goals**

1. Develop portable units of software that can be leveraged to build, and run computationally inexpensive MITgcm model problems on compute nodes without mpi libraries
2. Configure support for these units of software on the most frequently used computing platforms, architectures, and operating systems according to CRIOS group users
3. Document the software products in code repository and image registry
4. Present results to MITgcm-ECCO master repository managers

**Stretch Goals**

1. Implement a graphical-user-interface for end users to input quantitative model parameters and specifications along with desired post-processing/data analysis tasks and render output in the interface
2. Introduce support for more computationally expensive jobs to enable execution of more robust MITgcm model problems
3. Implement mpi libraries to add portability, flexibility, and efficiency to jobs
4. Add features for ASTE regional domain model problems
5. Scale up support for product to run on most-widely adopted computing platforms, architectures, and operating systems according to general purpose MITgcm-community use

**Deliverable(s)**

**The deliverables in this project will follow a conventional product design process including:**

* **Project Description and Literature Review**
* **Systems Overview**
* **Requirements Definition**
* **Preliminary Design**
* **Project Proposal**
* **Detailed Design and Analysis Plan**
* **Preliminary Results**
* **Final Presentation**
* **Final Report**

**Product Design Process Phases**

1. **Identification of Problem Phase**

* Problem Statement
* Need Statement
* Project Objectives

1. **Understanding of Problem Phase**

* Definition of design requirements and goals using the following framework
* **Requirements Gathering**: practice of obtaining the requirements of a system from users, customers, and other stakeholders; the requirements are to be split up into **subjective technical requirements (**ex: enough storage, large surface area for screen) and **quantitative requirements (**ex: minimum storage > 32 GB, screen size > 3” x 4”)
* **Requirements Analysis:** determining whether the stated requirements are clear, complete, consistent, and unambiguous
* **Requirements Specification:** clearly stating requirements in a form that can be understood by all stakeholders
* **Requirements Validation:** Checking requirements with stakeholders
* Background Research and Literature Review
* Enumeration of customer needs

1. **Ideation Phase**

* Brainstorm design ideas
* Develop sketches and schematics to communicate flow of information in product in a hardware/software-independent way: Define product as a black-box with inputs and outputs and identify each input/output in the system (product) and the systems they link to
* Translate black-box definition of product into hardware-software-specific product designs **(preliminary designs)**
* Document designs of product using hardware/software specific setups

1. **Evaluation Phase**

* Determine if preliminary designs meet requirements using the following framework:
* **Requirements Gathering**: practice of obtaining the requirements of a system from users, customers, and other stakeholders; the requirements are to be split up into **subjective technical requirements (**ex: enough storage, large surface area for screen) and **quantitative requirements (**ex: minimum storage > 32 GB, screen size > 3” x 4”)
* **Requirements Analysis:** determining whether the stated requirements are clear, complete, consistent, and unambiguous
* **Requirements Specification:** clearly stating requirements in a form that can be understood by all stakeholders
* **Requirements Validation:** Checking requirements with stakeholders
* Develop a requirement compromise schematic and decision matrix using the following templates

Diagram, shape

Description automatically generatedTable

Description automatically generated with low confidence

* Weigh preliminary design strengths/weaknesses and perform down-selection
* Incorporate decision tool to rate down-selected designs in feasibility study
* Select design(s) that are preferrable according to results of feasibility study

1. **Prototype and Testing Phase**

* Document detailed technical drawings of **detailed design(s)** that are optimal according to feasibility studies
* Implement mathematical and computer models for detailed design(s) in applicable system architecture
* Build physical model of detailed design(s)
* Design **Analysis Plan** to specify approaches to testing and analyzing the performance of detailed design(s) including but not limited to the following:

1. Time limitations (compile time, runtime)
2. Computing Resources (RAM, CPU clock time)
3. Accuracy of results (Confidence Intervals, Deviation)
4. Etc.

* Run and document test results from detailed design prototypes according to analysis plan
* **Perform the last step in requirements analysis (Requirements Verification):** Checking a working product against some standards and conditions imposed

1. Determine what the limits of the requirement(s) are by specifying the test results from analysis plan execution with quantitative bounds
2. Figure out input conditions under which requirements will fail
3. Document cases in which product will fail when in the hands of the user

* Using requirements verification results, document requirements that were and were not achieved, anything unexpected during testing phase, and iterate through design process using results

1. **Communicate your Solution Phase**

* Commit all project work to a public code repository and grant MITgcm and ECCO researchers privileges to view content
* Schedule meetings with senior research scientists to communicate results of project and address plans for potentially incorporating final product into MITgcm-ECCO workflow

**Use Cases**

**End Users**

The end users that these deliverables are intended for include scientists, researchers, graduate students, curious members of the ocean climate science community, and others interested in MITgcm.

**Points of Contact**

1. **Ivana Escobar –** PhD Candidate in Computational Science, Engineering, and Mathematics (CSEM) at University of Texas at Austin
2. **Patrick Heimbach –** Professor in Jackson School of Geosciences at University of Texas at Austin; Boss of Computational Research in Ice and Ocean Systems Group (CRIOS)
3. **An Nguyen –** Research Associate with CRIOS
4. **Helen Pillar –** Research Associate with CRIOS
5. **Shreyas Gaikwad -** PhD Candidate in Computational Science, Engineering, and Mathematics (CSEM) at University of Texas at Austin
6. **Ian Fenty -** Scientist with NASA’s Jet Propulsion Laboratory
7. **Ou Wang –** Scientist with NASA’s Jet Propulsion Laboratory