An Extensible Ecosystem of Tools Providing User Friendly HPC Access and Supporting Jupyter Notebooks

Ben Glick and Jens Mache

Lewis & Clark College, Portland, OR

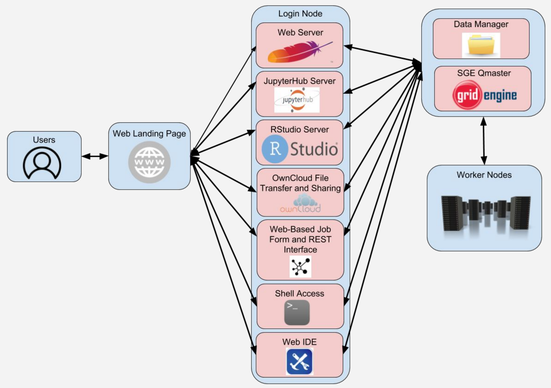
{glick, jmache}@lclark.edu

Abstract

- **High performance computing systems can be hard to use, and can require advanced knowledge of the command line, and resource managers.**
- **In this project, we describe our extensible ecosystem of tools which has allowed students and researchers with minimal HPC backgrounds to use our system with virtually no training.**
- **Among the tools we support are**
  - **(1) a web-based job submission and management system,**
  - **(2) a GUI-based file transfer and sharing tool, and**
  - **(3) a Jupyter notebook environment.**
- **Together, these tools and the interfaces around them allow users to run complete HPC workflows without any use of the command line. Performance measurements show that our ecosystem incurs marginal overhead.**

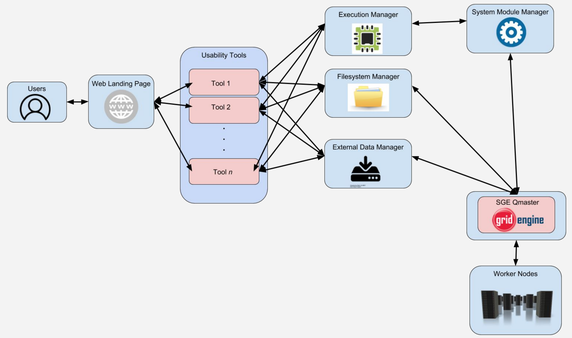
Introduction and Motivation

High Performance Computing (HPC) is becoming more and more important in many domains. However, HPC can be complex for people from domains other than Computer Science.

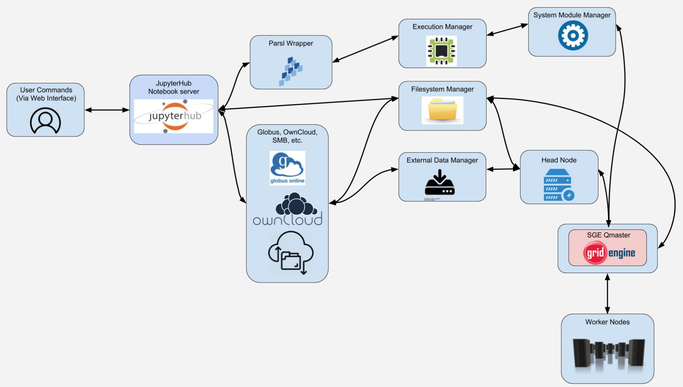


Our response: Create a “one stop shop” so that users only need to interact with one resource for all needs.

Architecture and Design

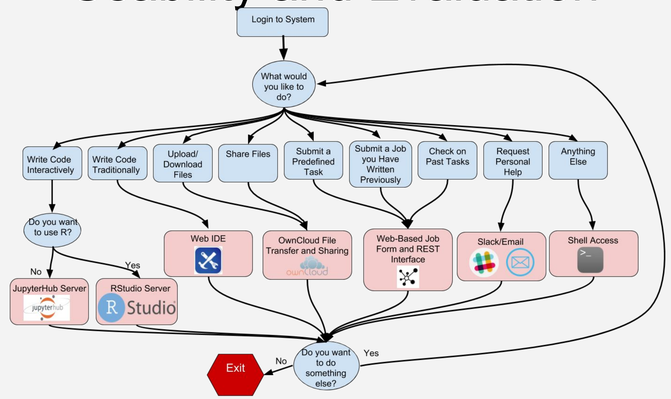


- **The three component modules are the execution manager, the filesystem manager, and the external data manager.**
- **The interface ensures that all tools which are to become modules in our system are able to interact with each one of these components.**
- **Before a tool is added to the system, a "wrapper" layer needs to be written for it.**
- **This allows the tool to interact with the rest of the ecosystem and ensures that it follows the interfaces provided by the ecosystem.**

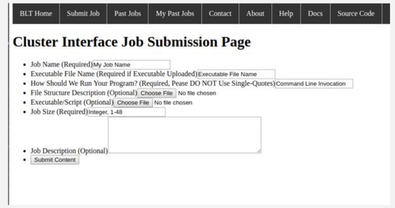


- **The wrapper we have designed for the Jupyter environment allows Jupyter to interact with the execution manager, and the filesystem and external data managers.**
- **Conforms to the filesystem manager interface we designed, which allows it to read and write files on the HPC system's filesystem.**
- **The Jupyter environment is also able to interact with the external data manager, which allows it to request files or other data from the external Internet.**

Usability and Evaluation



- **Our ecosystem has already been used for multiple computational science projects**
- **Our users, both professional researchers and students, have found our ecosystem to be easier to use than they expected.**
- **Users have a simple, intuitive workflow which allows them to design, implement, execute, analyze, and share their HPC tasks easily.**



The system has a web-based GUI form as well as a RESTful interface for automated job submission.

Our Jupyter notebook installation is simple and easy to use. As an example, this is a code sample which calculates pi via monte carlo simulation through our Jupyter notebook.

```
In [ ]:
# Workflow defined here:

@App('python', dfk)
def pi(total):
    # App functions have to import modules they will use.
    import random
    # Set the size of the box (edge length) in which we drop random points
    edge_length = 10000
    center = edge_length / 2
    c2 = center ** 2
    count = 0

    for i in range(total):
        # Drop a random point in the box.
        x, y = random.randint(1, edge_length), random.randint(1, edge_length)
        # Count points within the circle
        if (x - center)**2 + (y - center)**2 < c2:
            count += 1

    return (count * 4 / total)


@App('python', dfk)
def avg_n(inputs=[]):
    return sum(inputs) / len(inputs)
```

```
In [ ]:
# Call the workflows
sims = [pi(10**6) for i in range(10)]
avg_pi = avg_n([task.result() for task in sims])

In [ ]:
# Print the results
print("Average: {0:.63f}".format(avg_pi.result()))
```

Method of Launch	Min Performance (GFlops)	Median Performance (GFlops)	Max Performance (GFlops)
qsub	2,490.618	2,921.180	3,352.980
Jupyter notebook through our ecosystem	2,203.119	2,738.812	3,274.505
Percentage Difference	12.25%	6.44%	2.36%

Summary of benchmarking results showing that running HPC jobs through our ecosystem only imparts a slight performance penalty, done using repeated matrix multiplication submitted to our cluster both traditionally and through our ecosystem's Jupyter Notebook server.