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

Ben Glick and Jens Mache, Lewis & Clark College, Portland, OR 97219
{glick, jmache}@lclark.edu

***Abstract*— High performance computing systems can be hard to use, and can require advanced knowledge of the command line, resource managers, and other details that scientists without a systems administration background may find distracting. 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 only marginal overhead.**

I. INTRODUCTION

One challenge with HPC is ensuring that researchers without a systems administration background are able to easily and effectively use the HPC system. This paper introduces our ecosystem of tools, which serves as the ”one-stop” interface which allows users to focus on their research without remembering a lot of commands and syntax, or struggling with details like job submission or file systems.

II. TECHNOLOGIES

We call the interaction model we have implemented ”one-stop shopping.” For a user, the one-stop shopping model drastically simplifies the process of interacting with our cluster. Instead of attempting to either run all of their programs over the command line or develop programs on their computers before uploading to the HPC system, users are able to design, develop, and run complete workflows without any context switching. Switching from working on a local machine such as a laptop to working on a high-performance system is one of the most time-consuming parts of computational science. Users need to understand how programs run on their personal computers differ when run on high performance systems, how to upload and download data files and executables to and from the system, and how to make changes to their code on the remote systems. For many computational scientists, these tasks are unintuitive and uninteresting. In our system, users only need to know how to use a web browser in order to design, implement, and run full-scale workflows on a high-performance system.

We have designed an interface comprised of verbs that the underlying layer must be able to carry out upon request from tools which sit above it in the software stack. Because of this interface, it is very easy to add new tools to our list of modules, which makes it easy to incorporate user feedback into the system quickly and adopt new features and tools into the ecosystem. The lowest layer, which the tools sit on top of, is responsible for delegating the tasks launched by users to the worker nodes. This layer, which is comprised of three main components, is responsible for balancing load on the login node, delegating work to worker nodes when possible, and ensuring that user-requested tasks are performed as efficiently as possible.

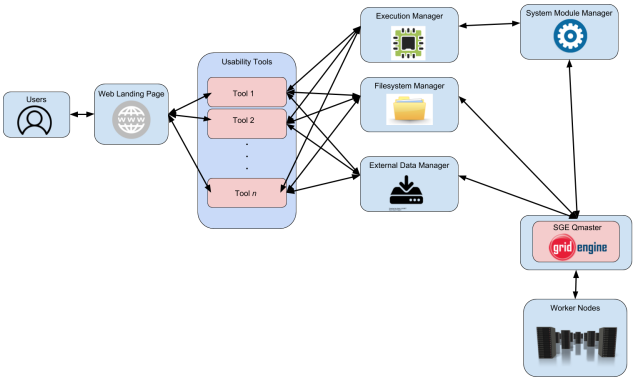

Fig. 1. **Diagram of the makeup of different interaction layers in our ecosystem**

The three aforementioned component modules are the execution manager, the filesystem manager, and the external data manager. The interface, which the tools in layer two must conform to in order to be modules in the ecosystem, ensures that all tools which are to become modules in our system are able to interact with each one of these components. However, some modules may not make use of all of the components. Before a tool is added to the system, a ”wrapper” layer needs to be written for it. This wrapper layer allows the tool to interact with the rest of the ecosystem and ensures that it follows the interfaces provided by the ecosystem.

III. DISCUSSION

When designing and building our ecosystem, we considered a number of factors which had design and usability implications.

A. Security

We have taken a multi-tiered approach to security in our ecosystem. The first tier is to attempt to keep unwanted

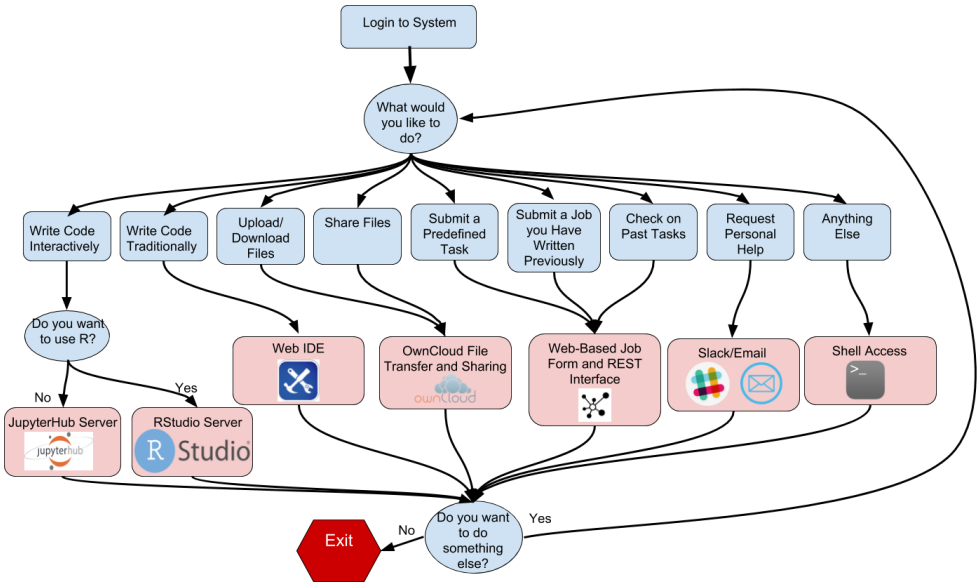

Fig. 2. **Flowchart summarizing the user workflow process**

users out of the ecosystem entirely, and the second tier is to make sure that even if an attacker can access the ecosystem, they will still be locked out of any important functions and tools. The third tier ensures that users only have access to the specific data they own or have been given rights to by its owner. This is important because some research projects require access to sensitive data, and that data must be carefully secured from users not on the project. We support both the traditional command line interface for changing file permissions, as well as a web-based GUI, provided through a wrapper around OwnCloud [1], for sharing files to users and groups, and changing read, write, and execute permissions on the files.

B. Usability and Extensibility

The end goal of this project is to provide a single unified environment from which all of our users' HPC needs are filled, regardless of the scale of the task they are trying to accomplish, their level of experience with HPC systems, or their familiarity with UNIX and command line interfaces. We believe we have taken a step towards this goal by providing users with an ecosystem designed to offer them enough options that they can compose, execute, analyze, and publish their HPC workflows, as well as requesting personalized assistance when necessary, all without ever leaving the web-based front end of our ecosystem. One of the most important and popular tools we have made available to users are Jupyter notebooks. Our Jupyter [2] notebook installation is simple and easy to use.

C. Scalability

Our ecosystem is also highly scalable. Though we do not have a larger cluster on which to install our ecosystem, we are confident that it would be easy to replicate our setup on a larger system with similar performance. If the cluster we were to install our ecosystem on has only one login node, as ours does, it would be essentially the same installation as we have on our cluster, and would be extremely simple to replicate. If the larger system in question has more than one login node, the only major change we would need to make is ensuring that an instance of each user-facing tool is installed and running on each login node, and ensuring that those instances of tools are able to communicate with each other through our interface. Our interface already supports network traffic.

IV. CONCLUSION

While related work about web-based access to HPC exists, not many support Jupyter Notebooks [3], and other works have not presented a single unified, integrated environment in which users can design, implement, execute, analyze, and share their HPC tasks without changes in context [4][5]. We have provided this "one-stop shop" of useful tools and modules to our users with minimal performance overhead.

REFERENCES

[1] A. Patawari, *Getting Started with ownCloud*. Packt Publishing, 2013.

[2] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds. IOS Press, 2016, pp. 87 – 90.

[3] F. Pérez and B. E. Granger, "IPython: a system for interactive scientific computing," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 21–29, May 2007. [Online]. Available: http://ipython.org

[4] A. Prout, W. Arcand, D. Bestor, B. Bergeron, C. Byun, V. Gadepally, M. Hubbell, M. Houle, M. Jones, P. Michaleas, L. Milechin, J. Mullen, A. Rosa, S. Samsi, A. Reuther, and J. Kepner, "MIT supercloud portal workspace: Enabling HPC web application deployment," *CoRR*, vol. abs/1707.05900, 2017. [Online]. Available: http://arxiv.org/abs/1707.05900

[5] K. Benedyczak, B. Schuller, M. P.-E. Sayed, J. Rybicki, and R. Grunzke, "Unicore 7 — middleware services for distributed and federated computing," in *2016 International Conference on High Performance Computing Simulation (HPCS)*, July 2016, pp. 613–620.