# Accessing Distributed Jupyter / Spark in OnDemand

Jeremy W. Nicklas
Ohio Supercomputer Center
Columbus, Ohio
jnicklas@osc.edu

Eric Franz
Ohio Supercomputer Center
Columbus, Ohio
efranz@osc.edu

Alan Chalker
Ohio Supercomputer Center
Columbus, Ohio
alanc@osc.edu

Doug Johnson
Ohio Supercomputer Center
Columbus, Ohio
djohnson@osc.edu

Morgan E. Rodgers
Ohio Supercomputer Center
Columbus, Ohio
mrodgers@osc.edu

David E. Hudak
Ohio Supercomputer Center
Columbus, Ohio
dhudak@osc.edu

***Abstract*—There are a variety of gateway software platforms available, each of which provide their own unique advantages. OnDemand's unique architecture empowers developers and users to easily create and run system-level access applications as well as interactive HPC applications. In this demonstration we show the ease of use through OnDemand to standup a Jupyter / Spark stack and run a distributed workload on an HPC cluster all within a browser.**

***Keywords—Open OnDemand, High Performance Computing, Interactive web platform, Jupyter, Spark, Ohio Supercomputer Center***

## I. Introduction

Open OnDemand, an open source software project that started in June 2015, provides HPC access through a web portal called OnDemand as an alternative to the command line for accessing and utilizing HPC resources [1, 2]. It was based around the Ohio Supercomputer Center's (OSC) original portal [3] that had been in production since January 2013 and was deployed into production at OSC in September 2016. Like science gateways Open OnDemand is about easing access to HPC resources by utilizing the web browser. OnDemand can be used to create science gateways via its application building framework, at its core OnDemand is about general system access: managing files, using shells and managing batch jobs.

A key feature of the OnDemand architecture over a traditional web service is the per-user web server model that ensures all processes are run as the authenticated Linux user. This feature directly leverages the Linux kernel for security and accountability making OnDemand an attractive solution for developers and users to easily create and run system-level access applications that handle file management, job management, as well as a multitude of other services. This enables OnDemand to easily support interactive HPC applications [4], custom plugin applications that build on system-level applications leveraging the cluster's resource manager to schedule and deploy web stacks directly on the cluster as well as allowing the user access to connect back to these running web stacks from their local browser. OSC currently provides Jupyter [5] (optionally with Apache Spark [6]), RStudio Server [7], and COMSOL Server [8] as custom interactive applications to its users running on its HPC clusters. A complete discussion of Open OnDemand's architecture can be found in Hudak et al 2016 [1].

Since OnDemand's public release in August 2017 it has been installed at sixteen different commercial and academic sites, with several dozen other centers showing significant interest. At OSC alone we have seen rapid user adoption with the number of OnDemand users doubling in less than a year since its initial launch as seen in Fig. 1. As of July 2017, OSC has more users accessing supercomputing resources through OnDemand than through traditional SSH alone.

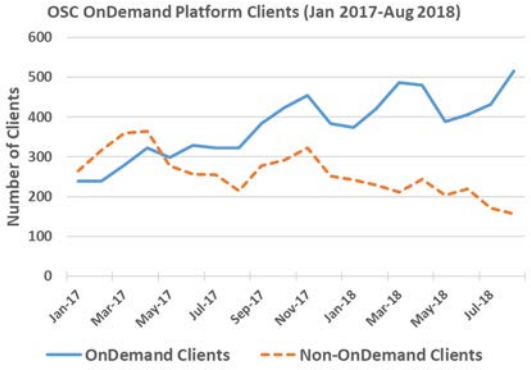


Fig. 1. OSC OnDemand platform usage for Jan 2017 – Aug 2018.

The rapid growth in OnDemand usage at OSC is in large part due to the lower barrier of entry for new users when they are first introduced to the complex HPC working environment. This has led to a reduction in "time to science" for new users


Fig. 2. Median times for first login of new users created in 2017 at OSC.

Presented at Gateways 2018, University of Texas, Austin, TX, September 25–27, 2018.
https://gateways2018.figshare.com/


Fig. 3. Median times for first job of new users created in 2017 at OSC.

using OnDemand. Fig. 2 displays the time it typically takes for a new user to first login and access the OSC clusters whether it was through OnDemand or traditional SSH. Users who first access OSC resources through OnDemand as opposed to traditional SSH login three days sooner. This could be attributed to only needing a browser and URL when connecting to OnDemand as opposed to the need for third-party software when connecting through traditional SSH. Fig. 3 displays the time it takes from first login until the user submits their first job to a cluster. OnDemand users are observed to begin working on their domain science nearly an entire day sooner when compared with traditional SSH users.

Admins will appreciate the RPM-based installation, and a focus of future development will be on simplifying installation of interactive applications.

For this demonstration we will present how easy it is for a new user with little to no Linux experience orchestrate the deployment of a properly configured and secure Jupyter server as well as a standalone Spark cluster across a set of HPC cluster nodes in just a few clicks all from within their browser. After connecting to this Jupyter instance we will demonstrate how simple it is to perform a distributed workload across the running Spark cluster from within a new Jupyter notebook.

## II. OnDemand Overview

When a user first logs into OnDemand they are presented with the Dashboard as seen in Fig. 4. The Dashboard serves as the landing page for the OnDemand portal as well as enabling discoverability of the various OnDemand features. These include the system-level access applications as well as the interactive applications.

OnDemand provides the following core system-level access applications that a user can launch from the Dashboard: File Explorer and File Editor for file management, Active Jobs and Job Composer for job management and monitoring, as well as a Shell App for command line access. The Dashboard can be further extended to include custom interactive applications that a user can access as seen in Fig. 5.

## III. Jupyter Stack

OSC currently provides their users access to Jupyter (optionally with Spark), RStudio Server, and COMSOL Server served through custom interactive applications hosted on their OnDemand portal as well as a host of X11/VNC applications: Xfce Desktop, ANSYS Workbench, Abaqus/CAE, COMSOL Multiphysics, MATLAB, ParaView, and VMD. For this demonstration we will focus our attention on the Jupyter interactive application.

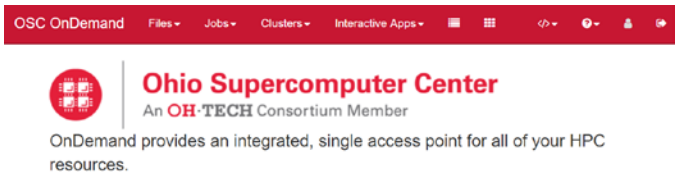


Fig. 4. OSC's OnDemand Dashboard page.

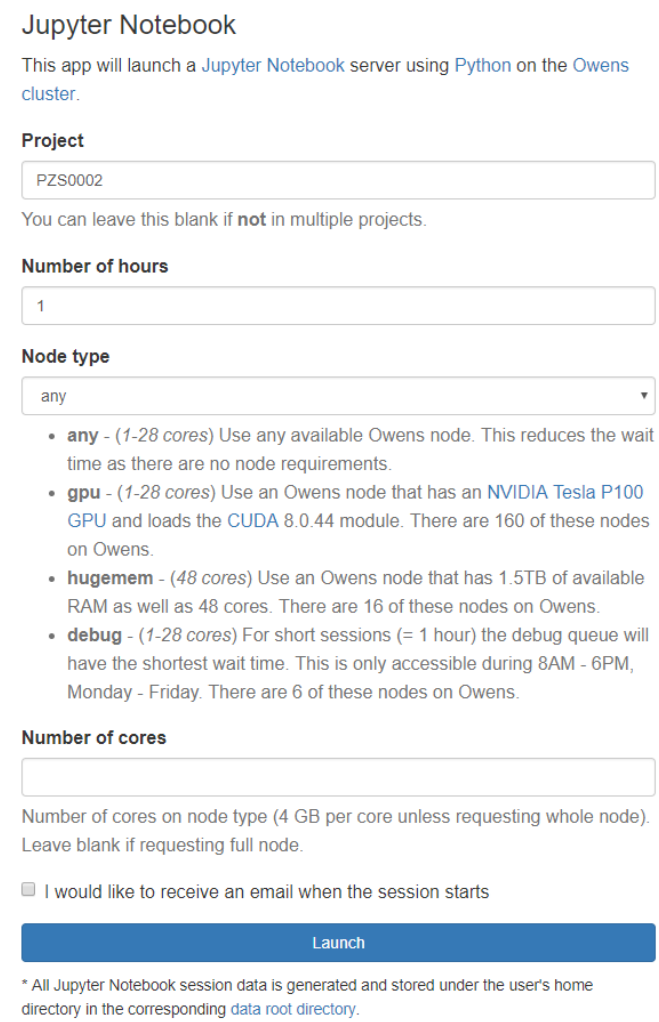


Fig. 6. Web form for launching Jupyter on OSC OnDemand.

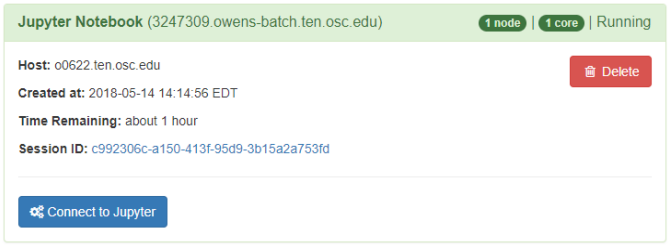


Fig. 7. User can connect to Jupyter stack when job is running.

From the Dashboard the user clicks the link for the Jupyter interactive application. The user is then presented with the web form seen in Fig. 6 which asks for the account the job will be charged against, the wall time limit for the job, the type of node and number of cores to run the job on, and whether or not to email the user when the job starts.

After the user fills out the form and clicks "Launch" a batch job is subsequently generated and submitted to the cluster's resource manager. The user is then presented with a button to connect to Jupyter after the job leaves the queue and begins running as seen in Fig. 7.

The user clicks "Connect to Jupyter" and is connected to their Jupyter server instance running on the allocated node in a new browser tab. The user can now open a new notebook or manage their current notebooks all from within their browser. It should be made clear that the user's browser is now connected to the allocated cluster node without the need for an SSH tunnel.

We now open a new notebook and populate it with the Python function to compute the mathematical constant Pi using Monte Carlo integration as seen in Fig. 8. We benchmark this calculation to later compare against the distributed calculation performed with the Spark cluster. By default, Python runs the calculation on a single thread, but could be made multi-threaded by leveraging third-party libraries.

For the case demonstrated in Fig. 8 it took a wall time of roughly 8 minutes to compute Pi on a single thread using a sample size of 1,000,000,000 points running on an OSC Owens compute node. Even if this calculation was made multi-threaded it would still be limited by the number of cores

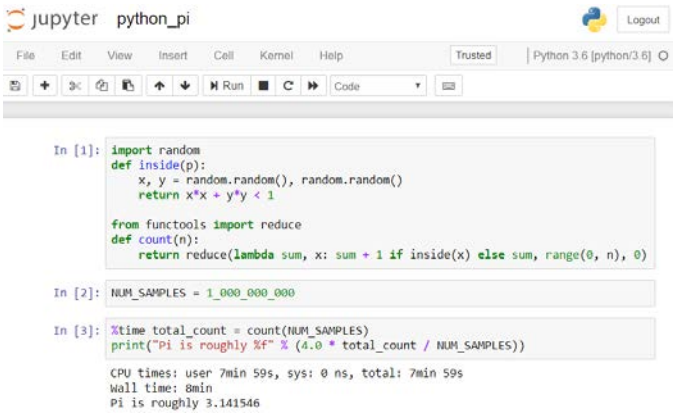


Fig. 8. Single-threaded Pi calculation in Jupyter notebook.

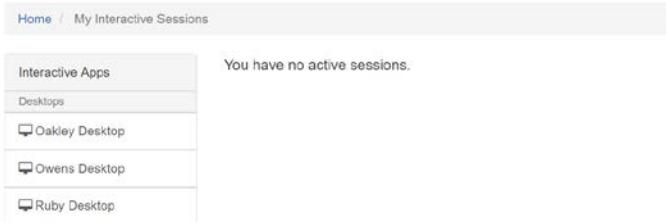


Fig. 5. Dashboard view of available and running interactive apps.

available on the OSC Owens compute node (28 cores).

## IV. Jupyter / Spark Stack

In order to scale this solution further we will need to distribute the calculation over multiple nodes. This can be accomplished with the Apache Spark cluster-computing framework. Apache Spark provides the Python API PySpark used to communicate with a running Spark cluster. PySpark utilizes the cloudpickle Python package to serialize Python constructs not supported by the default pickle module and distribute them across the Spark workers to be executed. This

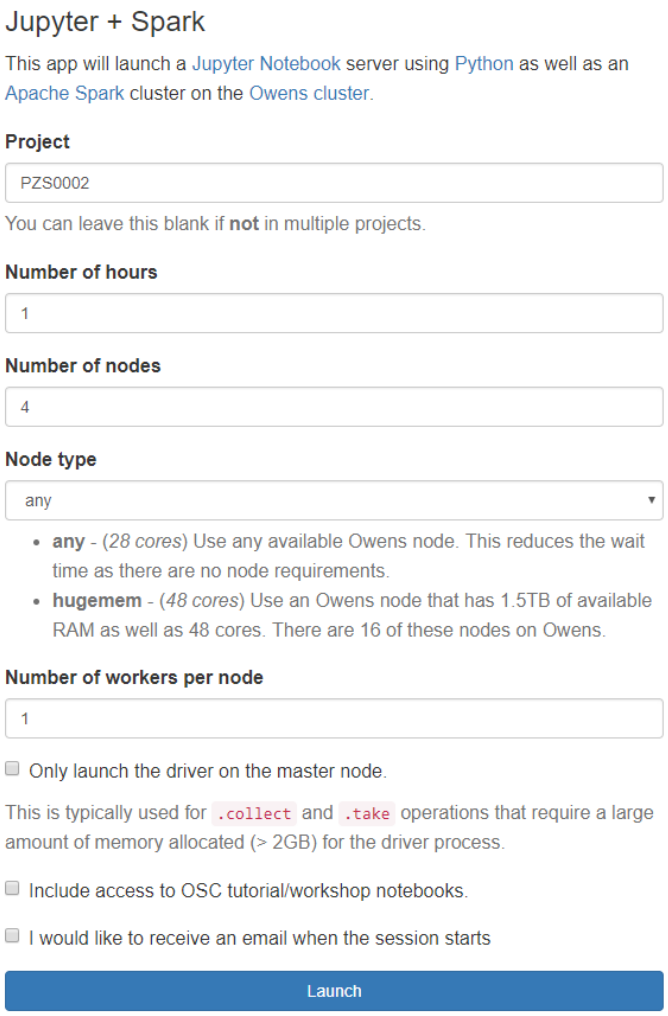


Fig. 9. Web form for launching Jupyter + Spark on OSC OnDemand.

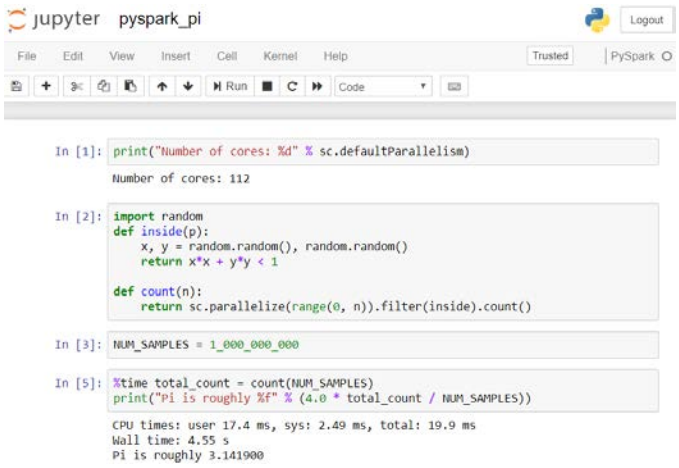


Fig. 10. Distributed Pi calculation in Jupyter notebook using Spark cluster.

allows us to take arbitrary Python functions and distribute their execution across more than one node.

The user now goes back to the OSC OnDemand Dashboard and clicks the link for the Jupyter + Spark interactive application. The user is then presented with the web form seen in Fig. 9 which now asks for number of nodes instead of cores to run the job on as well as configuration options for launching the standalone Spark cluster across the nodes. It also provides an option to copy over helpful tutorials into the user's home directory if requested.

As before, after the user fills out the form and clicks "Launch" a batch job is subsequently generated and submitted to the cluster's scheduler. The user is then presented with a button to connect to Jupyter + Spark after the job leaves the queue and begins running similar to Fig. 7. Behind the scenes the job dynamically configured and launched a standalone Spark cluster across all the allocated nodes before starting the Jupyter server. It also generated a custom Jupyter kernel for connecting to the running Spark cluster when the user launches a Jupyter notebook.

The user now clicks "Connect to Jupyter" and is connected to their Jupyter server instance running on the first node allocated by the cluster's scheduler. The Jupyter interface is the same as before but now all notebooks will be opened using the custom kernel that initiates a Spark Context (given by the `sc` variable in the interpreter) that connects to the Spark cluster. The user can make use of the Spark Context to immediately begin distributed work without any manual initialization.

We now open a new notebook and immediately use this available Spark Context object to display the number of cores allocated in the Spark cluster. We then use the same Monte Carlo integration routine from before but leverage the Spark cluster to distribute the function call across all allocated cores as seen in Fig. 10. We benchmark this calculation to compare against the single-threaded case previously.

For the case demonstrated in Fig. 10 it took a wall time of 4.55 seconds to compute Pi on 112 cores using a sample size of 1,000,000,000 points running on four OSC Owens compute nodes. This distributed example exhibits a speed up of 105 and parallel efficiency of 93.8% relative to the previous single-threaded calculation as expected for the embarrassingly parallel Monte Carlo integration.

Computation and resource usage may be ended at any point by shutting down the Jupyter notebook. Alternatively, OnDemand provides an interface to a user's active and pending interactive sessions. Clicking the Delete button on an active session calls the resource manager's job removal method: `qdel` in the case of Torque. Any other required changes to a pending or running job are handled outside of OnDemand using the hosting site's HPC resource management tools.

## V. Related Work

Other integrations between Jupyter and HPC systems exist; including Yin et al's CyberGIS and Milligan's customizations to JupyterHub. CyberGIS is an application framework itself built on Jupyter, which intends to leverage hybridized HPC stacks including "traditional HPC, scalable databases, cloud environments as well as big data computation frameworks like Hadoop and Spark" [9]. CyberGIS is deployed as a containerized application ready for cloud providers. One of Milligan's key contributions to the Jupyter community has been the creation of the BatchSpawner, which starts per-user instances of Jupyter on HPC batch resources. Milligan further extended MSI's Jupyter platform allowing it to act as a portal to graphical applications [10] using a JavaScript VNC client similar to OnDemand [3].

Development of OnDemand has focused on supporting HPC resource managers such as Torque [11], and Slurm [12] as they are the most common mechanism employed for distributed resource management at HPC centers. Open OnDemand does not support alternative distributed resource management frameworks such as Kubernetes [13] at this time, as it would require centers to dedicate hardware distinct from the HPC clusters to meet resource demands. Future work for OnDemand includes investigations into alternative non-HPC resource management schemes such as Kubernetes.

## VI. Conclusion

OnDemand's architecture lends itself to making web applications that directly interact with the HPC cluster easy to develop and run. This opens up the possibility for running interactive applications that are distributed across a cluster for "science at scale" while also making it easy enough that a user with little to no Linux experience can get started right away.

## Acknowledgment

To learn more about the Open OnDemand project please visit the project's website [12]. The website provides access to the documentation for installing OnDemand as well as tutorials on getting started with interactive application development. It also provides a link to sign up for the mailing list to ask questions, receive updates, and participate in relevant discussions.

This work is supported by the National Science Foundation of the United States under the award NSF SI2-SSE-1534949.

## REFERENCES


[1] D. E. Hudak, D. Johnson, J. Nicklas, E. Franz, B. McMichael, and B. Gohar, "Open OnDemand: transforming computational science through omnidisciplinary software cyberinfrastructure," in *Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale, XSEDE16, Miami, FL, USA, July 17-21, 2016*, ACM, New York, NY, USA, no. 43, 2016, pp. 1-7. [Online]. doi: https://doi.org/10.1145/2949550.2949644

[2] D. Hudak, D. Johnson, A. Chalker, J. Nicklas, E. Franz, T. Dockendorf, and B. L. McMichael, "Open OnDemand: a web-based client portal for HPC centers", *The Journal of Open Source Software*, vol. 3, no. 25, pp. 622-623, 2018. [Online]. doi: https://doi.org/10.21105/joss.00622

[3] D. E. Hudak, T. Bitterman, P. Carey, D. Johnson, E. Franz, S. Brady, and P. Diwan, "OSC OnDemand: a web platform integrating access to HPC systems, web and VNC applications," in *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery, XSEDE '13, San Diego, CA, USA, July 22-25, 2013*, ACM, New York, NY, USA, no. 49, 2013, pp. 1-6. [Online]. doi: https://doi.org/10.1145/2484762.2484780

[4] J. W. Nicklas, D. Johnson, S. Oottikkal, E. Franz, B. McMichael, A. Chalker, and D. E. Hudak, "Supporting parallel, interactive Jupyter and RStudio in a scheduled HPC environment with Spark using Open OnDemand", in *PEARC '18: Practice and Experience in Advanced Research Computing, Pittsburgh, PA, USA, July 22-26, 2018*, ACM, New York, NY, USA, in press.

[5] 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. Abdallan 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. [Online]. doi: https://doi.org/10.3233/978-1-61499-649-1-87

[6] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, "Apache Spark: a unified engine for big data processing", *Commun. ACM*, vol. 59, no. 11, pp. 56-65, Oct. 2016. [Online]. doi: https://doi.org/10.1145/2934664

[7] RStudio Team, "RStudio: integrated development environment for R", RStudio, Inc., Boston, MA, USA, 2015. [Online]. Available: https://www.rstudio.com

[8] COMSOL, "COMSOL Multiphysics Modeling Software", 2018. [Online]. Available: https://www.comsol.com [Accessed: 15- May- 2018].

[9] D. Yin, Y. Liu, A. Padmanabhan, J. Terstriep, J. Rush, S. Wang, "A CyberGIS-Jupyter Framework for Geospatial Analytics at Scale", in *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact* (p. 18). ACM. [Online[. doi: https://doi.org/10.1145/3093338.3093378

[10] M. Milligan, "Jupyter as Common Technology Platform for Interactive HPC Services", in PEARC '18 Proceedings of the Practice and Experience on Advanced Research Computing (article 17). ACM. [Online] Available: https://doi.org/10.1145/3219104.3219162

[11] Torque by Adaptive Computing, "Torque Resource Manager", 2018. [Online]. Available: https://www.adaptivecomputing.com/products/torque/. [Accessed: 4-Sept-2018].

[12] Slurm by SchedMD, "Slurm Workload Manager", 2018. [Online]. Available: https://slurm.schedmd.com. [Accessed 4-Sept-2018].

[13] Kubernetes, "Production-Grade Container Orchestration – Kubernetes", 2018. [Online]. Available: https://kubernetes.io. [Accessed: 4-Sept-2018].

[14] Open OnDemand by OSC, "Open-source project based on the Ohio Supercomputer Center's OnDemand platform", 2018. [Online]. Available: http://openondemand.org. [Accessed: 15- May- 2018].