UNIVERSITY OF BATH

HPCBYTES

RESEARCH COMPUTING NEWSLETTER

ISSUE 6 JULY 2024

In This Edition

nimbus_splash – A Simpler and Quicker Way to Use ORCA on Nimbus

Jon Kragskow (Research Associate, Department of Chemistry)
Email: jgck20@bath.ac.uk

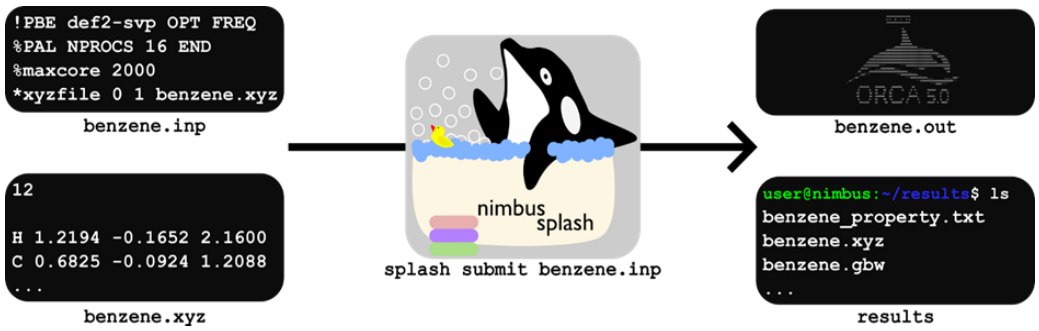


Figure 1: An example of using nimbus_splash to submit an ORCA job to Nimbus.

Jon is a research associate in the Department of Chemistry. His work uses complex electronic structure calculations to understand the structure and properties of magnetic molecules.

The field of computational chemistry uses complex mathematical theories such as quantum mechanics to describe the structure and properties of a diverse range of chemical systems including single molecules, solids, or even large biological systems such as proteins. Unfortunately, there is no single method which gives a perfectly accurate description of every chemical system in finite time, and so a computational chemist must select a model which balances chemical accuracy with compute time.

Jon works with relatively small molecules, each containing fewer than 200 atoms. He is particularly interested in understanding how their magnetic properties are influenced by their chemical structures. This intricate investigation necessitates the use of computationally expensive and highly accurate methods. For Jon's research, these sophisticated computational techniques are best executed using ORCA [1]. ORCA is a versatile, efficient, and user-friendly open-source software package tailored for quantum chemistry, with a specific focus on the spectroscopic properties of open-shell molecules.

"By leveraging splash, Jon significantly streamlines the ORCA job submission process, minimizes errors, and ultimately enhancing the productivity and reliability of his research work using Nimbus computational resources."

As is common with most High Performance Computing (HPC) compatible software, setting up an ORCA simulation on Nimbus requires an initial configuration. This involves creating an input file that contains a comprehensive set of instructions, including the specific model to be used along with resource information (number of cores and per-core memory). Once Jon has meticulously prepared this input file, the next step is to submit it as a batch job on Nimbus. Upon completion of the simulation, one or more output files are generated.
However, this entire process occurs on the compute node of Nimbus. If there are any errors in the input file, the job fails, necessitating a resubmission and resulting in a loss of valuable computational time. Additionally, since Jon frequently uses Nimbus' highly sought-after spot-fsv2 instances, he often encounters eviction notices for his jobs. These interruptions result in the inefficient use of HPC time and money.

To optimize his workflow, Jon has developed nimbus_splash [2] (or splash for short), a Python package that simplifies ORCA job submission on Nimbus. After a quick and easy installation process, splash allows users to submit ORCA jobs with a single command (Figure 1). Put simply, splash is a submission script generator with the added ability to parse ORCA input files. Before submitting any jobs to the queue, splash searches for unmet dependencies, checks the input file formatting, and verifies that the specified configuration is compatible with the selected compute instance. Moreover, the submission script generated by splash also makes sure that, in the event of an eviction notice, the simulation data is copied back from the compute node before it's released back to the scheduler. Finally, splash enforces a clean directory structure for all output files, thus avoiding a muddled collection of old and new data. In Jon's own words:

"While splash is not ground-breaking, I believe that it is a useful tool for saving time when using ORCA on Nimbus. I encourage interested users to look at the splash documentation, and I am open to feature requests and feedback."

References

1. *Neese F. Software update: The ORCA program system–Version 5.0. WIREs Comput Mol Sci. 2022.*
2. *https://www.kragskow.dev/nimbus_splash/*

Running Your Julia Workload with Jupyter Notebook

As discussed in issue 4 of HPCBYTES, Jupyter Notebook is a useful tool for shortening the feedback loop between coding and results visualization in an interactive manner. This article demonstrates how to run Julia programs using Jupyter Notebook on Nimbus. The steps are similar to those used for running Python programs with Jupyter. However, for Julia, a one-time setup process is required in the Nimbus user account before submitting Julia batch jobs and accessing them on the user's local web browser via Jupyter Notebook.

Steps to Run Julia Interactively with Jupyter Notebook

Step-1: Adding IJulia package in user account

```
# Login to your nimbus account and load the Julia module
user@nimbus-1-login-2 ~ $ module load Julia
$ julia
julia> using Pkg
julia> Pkg.add("IJulia")
julia> exit()
```

Step-2: Submit the Jupyter Notebook Job

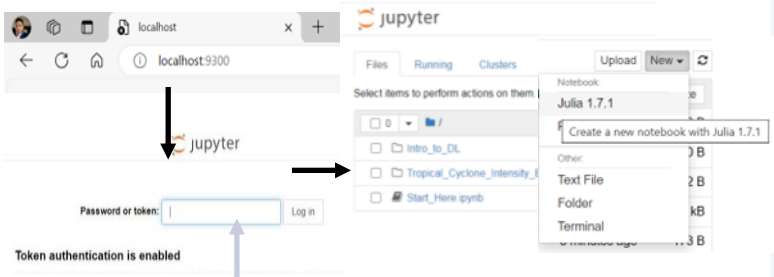
```
# The job script to launch Jupyter with Julia.
#!/bin/bash
#SBATCH --job-name=july-test
#SBATCH --output=july-test.out
#SBATCH --error=july-test.err
#SBATCH --account=CA-CS1HGN-0XX
#SBATCH --qos=paygo-fsv2-1
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --partition=paygo-fsv2-1
#SBATCH --time=01:00:00

NOTEBOOK_LOGFILE=jupyterlog.out
# get port forwarding info into a file
node=$(hostname -s)
user=$(whoami)
cluster="nimbus.hpc.bath.ac.uk"
port=9300

echo -e "Run the following command from your local machine terminal with local machine port YYYY:
$ ssh -N -f -L YYYY:${node}:${port} ${user}@${cluster}" > port_forwarding.txt

module purge
source /apps/build/easy_build/scripts/id_instance.sh
source /apps/build/easy_build/scripts/setup_modules.sh
module load Julia/1.7.1-linux-x86_64
module load Anaconda3/2022.10

# Run the Jupyter notebook
jupyter notebook --no-browser --ip=${node} --port=${port} > ${NOTEBOOK_LOGFILE} 2>&1
```

Step-3: Once the job is running, do the port forwarding on your local machine

Check the "port_forwarding.txt" file made after the job started running on Nimbus and copy paste in your local terminal (change YYYY to your local port, anything >9000):

```
[@localmachinexxx]$ ssh -N -f -L 9300:nimbus-1-paygo-xxxxxxxx:9300 user@nimbus.hpc.bath.ac.uk
```

Step-4: Open web browser in local machine and type localhost:YYYY



- Get the token from jupyterlog.out file and paste

```
[user@nimbus-1-login-1~]$ tailf jupyterlog.out
http://nimbus-1-xxxxxxx:9300/
token=93dd55ca40b2f463fxxxxxxxxxxxxx
```

HPC News and Updates

Isambard 3 Update

- Isambard 3 early users access is expected by August 2024 once the GW4 agreement is finalized.
- **Users must move important data off Isambard 2.**

Jupyter on Nimbus Training Recording

- Training recording would be available on website shortly. In meantime, if you need to access the recording, please email research-computing@bath.ac.uk.

Tip of the Month

To get a list of available instances/ partitions accessible to you

```
# An instance/partition is a set of compute nodes grouped logically. To get the information available partitions and resources
[user@login1 ~]$ sinfo -l
# Display instances which the user can access
[user@login1 ~]$ sacctmgr show User $USER --associations -P
```

Acknowledgements

The Research Computing team would like to thank all contributors for the current issue of ***HPCBytes***.

- If you would like to contribute a case study or article to be featured in ***HPCBytes***, please get in touch with the Research Computing team.
- If you would like to hear more, please subscribe to the Research Computing mailing list here: https://forms.office.com/e/rF8rLWbakA

Contact us

Research Computing Team
Digital, Data and Technology
Email: it-hpc@bath.ac.uk