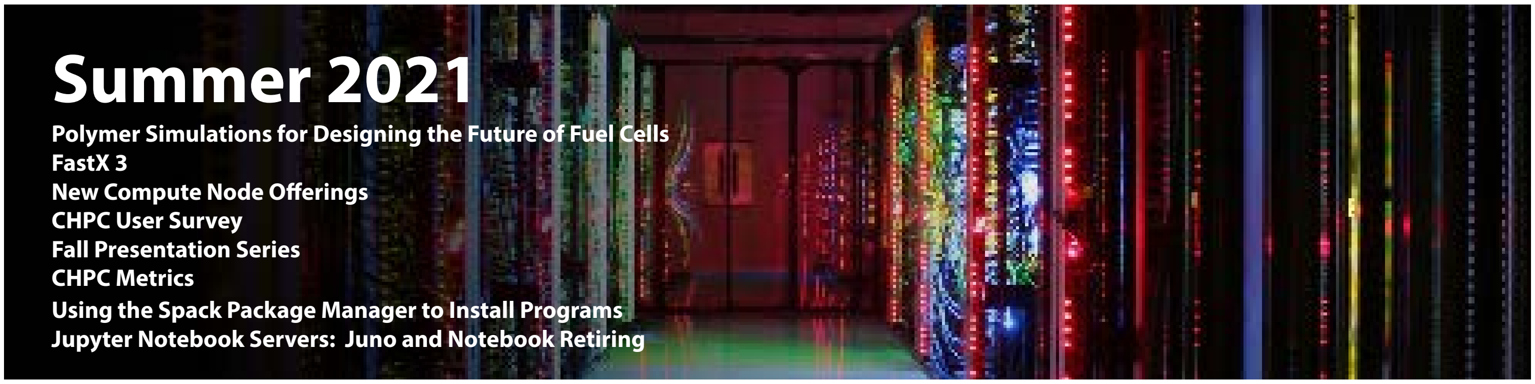
Center for
HIGH PERFORMANCE COMPUTING
THE UNIVERSITY OF UTAH



Summer 2021

Polymer Simulations for Designing the Future of Fuel Cells

Adam Barnett, Molinero Research Group, Department of Chemistry

The world's power needs have more than doubled in the past 30 years, and are projected to increase more than 50% more in the next 30. This tremendous increase in energy needs, combined with the growing climate crisis created by the usage of fossil fuels to meet such demands, has driven the search for more efficient and renewable energy sources. One rapidly growing area of interest is fuel cells, which have been investigated since their inception in the early 1960's and subsequent usage on the Apollo 11 mission to provide power to the command module as well as clean drinking water for astronauts.

Fuel cells are being investigated as one of many solutions to the growing energy needs of our planet due to their high efficiency, portability, and lack of greenhouse gas by-products. By harnessing the direct flow of electrons during the chemical reaction between hydrogen and oxygen, fuel cells offer the maximum efficiency of converting chemical fuel to electrical power.

Modern fuel cells rely on an ion permeable polymer membrane to act as both a chemical and physical separator between electrodes. These membranes must provide both the mechanical rigidity of a solid and the chemical transport properties of a liquid, while being resistant to the corrosive chemical reaction that powers the fuel cell. These properties are achieved by using polymer electrolytes that have chemically stable and rigid backbones with attached ions that serve to both absorb water and aid the transport of reactants. Due to the hydrophobicity of the polymer and the hydrophilicity of the ions that are chemically bonded together, polymer electrolytes have water channels contained within a polymer web, giving it both the structural rigidity and chemical transport needed. The most commonly used polymer electrolyte shuttles protons between the fuel cell electrodes to complete the reaction. However, the extremely high acidity of these protons requires the electrodes to be made of rare corrosion resistant metals such as platinum and palladium. The high cost of these precious metals are one of the reasons fuel cells are not a mainstream source of power production.

By reversing the flow of ions in the cell, it is possible to pass hydroxide ions through the membrane instead of protons. This creates a high pH environment in which more common metal catalysts such as nickel can be used, lowering the cost of the fuel cell and making them more appealing for the consumer market. Unfortunately, anion exchange membranes suffer from lower conductivities than their proton exchange counterparts, an issue that must be overcome before their widespread usage.

Computer Simulations to Aid Material Design

The rapid advance in computing capabilities over the past few decades has led to a greater interest in using simulations to understand and predict materials properties from their molecular structure and interactions, and using the results to guide materials design principles for more a rapid prototyping and development process. In the Molinero research group at the Department of Chemistry of the University of Utah, we use molecular simulations to study how the chemical structure of the polymers used for anion exchange membranes can be designed to solve their current limitations. A computationally efficient model that faithfully reproduces the physical properties of the membrane is paramount to connect these properties – such as conductivity, electro-osmotic drag, water uptake, chemical degradation, and mechanical stiffness – to the molecular design and interactions of the polymer. Molecular dynamics (MD) is a methodology that is akin to an experiment, all performed in a computer: the user sets how molecules interact at the atomic level, then evolves the positions of the molecules

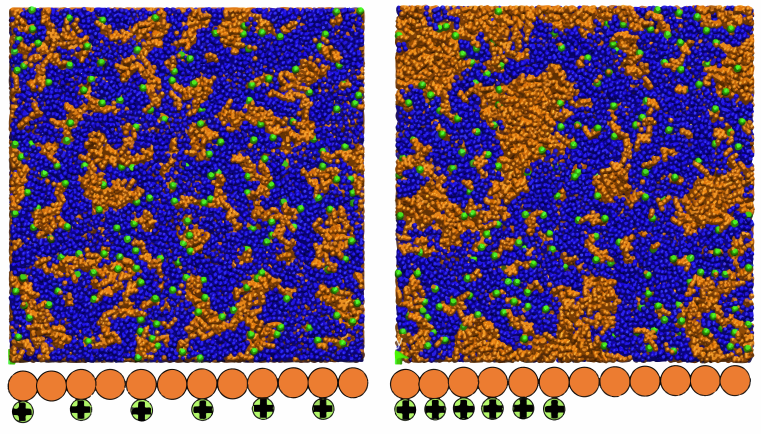


Figure 1. Two different membrane structures that result from having the ions spread evenly across the polymer (left), and having the ions clustered together (right). Polymer is shown in orange, cations in green, and water in blue. Note the large polymer regions in the clustered polymer that occur due to the increased segregation of the ions along the polymer chain.

in time according to Newton's laws. The results are simulation trajectories that sample the states of the material with molecular resolution that provide insights that are complementary yet inaccessible to experiments.

Two challenges in the modeling of fuel cell membranes are their complex heterogeneous structure, which requires simulation cells with hundreds of thousands of atoms, and the wide range of relaxation times of the different components – from the fast mobility of water, intermediate of the ions, and extremely slow of the polymer. The Molinero group has addressed these challenges by developing a model, in which the hydrogens of the chemicals are merged with the heavier atoms, cutting the number of particles simulated by approximately a third. Additionally, the interactions of the system are short ranged such that the computational burden of computing the forces between each pair of atoms is dramatically reduced. The combination of our efforts to lessen the computation time of the system has resulted in an approximately 100 times speedup over traditional, all-atom simulations, allowing for simulation of larger systems over longer times, to reveal many of the structural features that cause differences that have been previously inaccessible.

Even with our model's computational improvements, each of these simulations still take tens of thousands of core-hours and would not be possible without the resources of CHPC and its generous grants of computing time.

Membrane Simulation and Design

The most important structural features that characterize the membranes used in fuel cells are a web-like polymer, providing structural rigidity of the membrane, and interconnected water channels that allow ions to freely move throughout it. Using our model, we have uncovered how varying the order and structure of the monomers in the polymer chain tunes the structure and segregation of these two domains.

Exploring a wide range of polymers, we find that the ionic conductivity is controlled primarily by the amount of water in the membrane. For example, we have simulated the differences obtained when arranging the ions in groups on one end of the polymer chain versus having the ions spread across the polymer (Figure 1). Despite the differences in structure of the polymer, the ion conductivity is nearly identical in the two membranes, due to the similarity in size of the water domains.

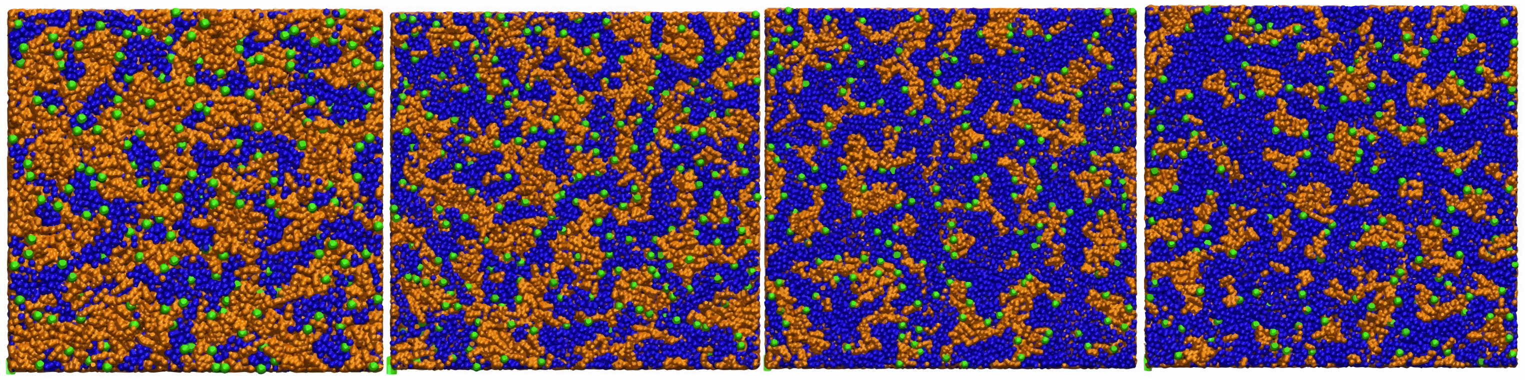


Figure 2. Water sorption within the PPO-TMA ionomer membrane. Membranes with water contents of 5, 10, 15, and 20 (left to right) water molecules per ionic group are shown.

The water content in ionomer membranes is determined by the competing forces of the ions that want to absorb water and the hydrophobic polymers that need to stretch like a rubber band to contain the absorbed water (Figure 2). If the ionic content is too high, the polymer will absorb water uncontrollably and, in some cases, dissolve entirely. However, if the ionic content is too low, the water uptake is low and the scarcity of water and low density of ions results in conductivity that is too low for fuel cells. This leads to a Goldilocks compromise in the design of membranes: they should adsorb not too much water, and not too little.

Despite a wealth of research, very little is known about the root causes of the of water absorption in membranes and how polymer design can affect it. The most common way of controlling the water uptake is to change the ionic content of the polymer membrane. By decreasing the number of ions, the forces causing water to intrude into the membrane lessen, resulting in lower water uptake. Our simulations have shown that this occurs not only due to the lower ionic content of the polymer leading to lower water absorption, but also due to stronger elastic forces of the polymer caused by a more robust polymer web. Our simulations also show that polymers having clustered ions form thick domains and have high water uptakes, whereas polymers with ions evenly distributed, which form tight webs and therefore have lower water uptake. The thicker domains act like fewer, thicker rubber bands, allowing water to form larger channels that are not as well bound. These two results indicate that developmental efforts should focus less on trying to decrease water clustering by increasing the cohesion of the polymer, and instead focus on creating more web-like membranes.

The computational cost of the simulation methods used to obtain the results presented have hindered the use of simulations to study water absorption in the past. Even with our extremely efficient model, the series of simulations carried out to elucidate the results outlined above took about half a million core-hours, equivalent to the entire compute capacity of CHPC's Kingspeak supercomputer for approximately 2 months. Given our 100x speed-up over conventional atomistic models, this problem is reduced from being intractable, to merely costly.

Looking into the future

It has long been a goal of computer modeling and simulation to partner with laboratory experiments to accelerate the design of bespoke materials. The incredible growth of modern computing power, models, and algorithms is making this promise a reality. The insights and design rules derived from the computer simulations have promise to produce membranes that have the high conductivity, mechanical resistance and chemical durability required for the production of efficient, sturdy fuel cells. The advances ushered by our molecular simulations could not have been achieved without the help and resources provided to us by CHPC. To find out more about our work in simulating these membranes and other self-assembling systems visit our group page found at: *http://molinero.hec.utah.edu/*.

FastX 3

Anita Orendt, CHPC Scientific Consultant

CHPC now has FastX 3 installed on all CHPC cluster interactive nodes in the general environment, including the frisco nodes. At this time only FastX 2 is available in the protected environment and on any stand-alone installations. We will maintain both FastX 3 and FastX 2 for a period of time to allow users to transition between the two versions. Unless there are user reported issues with FastX 3, we will remove FastX 2 sometime in late August or early September.

With FastX 3 the mechanism for downloading the desktop client has changed. This is now done via a link provided as a notification – see the bell icon near the top of a FastX 3 session – which will take you to the StarNet site with the downloads. If you do install the desktop client, please note that you will need to redefine the servers – this information will not be migrated from an existing FastX 2 installation.

See *https://www.chpc.utah.edu/documentation/software/fastx2.php* for additional details on the setup and use of FastX 3. Until we discontinue support for FastX 2, information on the usage of both versions will be provided. We will send an announcement when FastX 3 has been installed in the protected environment.

New Compute Node Offerings

Anita Orendt, CHPC Scientific Consultant

In spring 2021 both AMD and Intel released new generations of their processor lines. AMD announced the new 3rd generation EPYC processors, Milan, in mid-March. Intel followed with an early April announcement of their 3rd generation Intel Xeon Scalable processor line, code named Ice Lake.

With the Ice Lake processors, Intel moved to eight channels of DDR4-3200 memory per socket and up to 64 lanes of PCIe Gen4 per socket, compared to six channels of DDR4-2933 and up to 48 lanes of PCI Gen3 per socket for the previous generation. This changes the memory options from 192, 384, or 768 GB of memory to 256, 512, or 1024 GB of memory for a dual socket system. The new processor replacement for the Intel Xeon Gold 6230 and 6230R we have been using is the 6330. The 6330 increases the core count per processor to 28 from the 26 in the 6230R processor. Intel has reported a 20% raw performance boost with the new generation of process.

With the AMD new generation, there are no major changes outside of the actual processor. Reports state that there is a 10-20% performance boost between the 2nd and 3rd generation AMD processors. Please note that the actual performance boost is dependent on your application.

For CHPC's standard HPC compute node configurations, we have obtained refreshed quotes of both the AMD and Intel based servers with processors from these new processor lines. Along with the updated processors, and increased memory channels for the Intel processor-based servers, we are also moving to a 4 TB local drive, up from 2 TB, and to a HDR 200G Mellanox Infiniband (IB) card. The move to the HDR 200 GbE IB cards was due to Mellanox discontinuing the EDR cards we have previously used, as well as to allow for the higher bandwidth between nodes on the same IB switch, as we had previously moved to using HDR IB switches as the top of rack switches in both redwood and notchpeak. These top of rack IB switches still link back to EDR spine switches, so communication between nodes on different IB top of rack switches is still limited to the 100 Gb/s EDR speeds. The servers come with a 7-year warranty at time of purchase.

The new pricing from the vendor is shown in the table below. As before, CHPC adds an additional $1,000 cost per node. This addon cost covers the cost of network ports, cables and power infrastructure required to add the server to the HPC cluster.

We also have quotes of different GPU nodes with the new generation processors. If you are intersted in purchasing a GPU node, or a different configuration of a CPU node, please let us know and we will work with the vendor to obtain pricing.

CHPC User Survey

Brett Milash, CHPC Scientific Consultant

CHPC's first ever User Survey will be sent to all users in early August. Please take the time to complete the survey as it is important that we gather input from as many users as possible. We will leave the survey open for responses for a month.

While we have had annual PI surveys for a number of years, this year we decided to supplement the input from the PIs with information about experiences with CHPC resources and services from the user's perspective. Along with gathering information about your experiences with current CHPC offerings, we are also looking for your input on ways we can improve our offerings to better meet your research computing and data needs.

Fall Presentation Series

Anita Orendt, CHPC Scientific Consultant

The CHPC Fall 2021 Presentation Schedule is now posted at *https://www.chpc.utah.edu/presentations/fall-2021chpcpresentationschedule.php*.

In the fall we will be returning to holding the presentations in person in the INSCC Auditorium as well as via zoom. The XSEDE HPC series will remain available via zoom only. As in other fall semesters, we will be doing presentations on Monday, Wednesday, and Friday with the first presentation being held Monday, September 30. The presentations are free of charge, and there is no registration required.

Dell R6515 PowerEdge server with AMD, single 7713P (64 cores, 2.0 GHz base clock speed) processor	
256 GB memory	*$6,033.52*
512 GB memory	*$7,886.37*
1024 GB memory	*$10,460.05*
Dell R650 PowerEdge Server with Intel, dual 6330 (2 x 28 cores, 2.0 GHz base clock speed) processors	
256 GB memory	*$6,526.79*
512 GB memory	*$8,635.56*
1024 GB memory	*$11,668.34*

CHPC Metrics

Anita Orendt, CHPC Scientific Consultant

One useful metric for the users of the HPC resources is the typical wait time for jobs. While jobs sometimes start as soon as they are submitted to the batch queue, but other times jobs will wait in the queue hours or even days before the job starts.

On the *http://xdmod.chpc.utah.edu/* site the wait time listed is the wait time across ALL partitions on all of the general CHPC clusters and includes the wait time of owner jobs on the owner nodes as well as guest jobs on the owner nodes and freecycle jobs on notchpeak. For the protected environment (PE), the xdmod site is *http://pe-xdmod.chpc.utah.edu/*. Note that you must be on campus or using the campus VPN to access the pe-xdmod site.

However, breaking down the wait time by partition is much more useful. This can be done by moving to the Usage tab on the xdmod site and selecting the Wait Hours per job under the jobs by queue. The results can then be filtered for the queues (partitions). In the figure below the general partitions given are notchpeak (requires allocation) along with the queues open to all users: lonepeak, kingspeak, notchpeak-shared-short, and the general gpu partitions of notchpeak-gpu and kingspeak-gpu which are available upon request to all users who need access to the gpus.

As can be seen in the graphs in Figures 1, the average wait time in the batch queue per partition over a month is typically less than 12 hours in the general environment, though there are spikes in usage for some of the partitions; in the PE the wait time in the queue is typically less than 2 hours.

If you explore a shorter time window, say since the start of 2021, you can see the variation in wait time on a daily basis, which is a more useful timescale for looking at the variations when deciding on which partitions to use. This is shown for the general environment in Figure 2 below.

Groups with owner nodes can select the owner partitions to obtain similar information for those partitions.

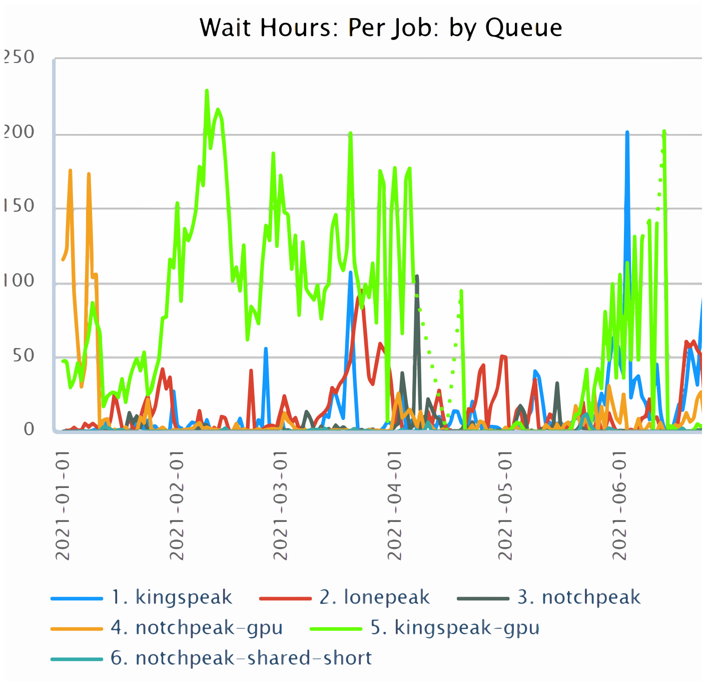
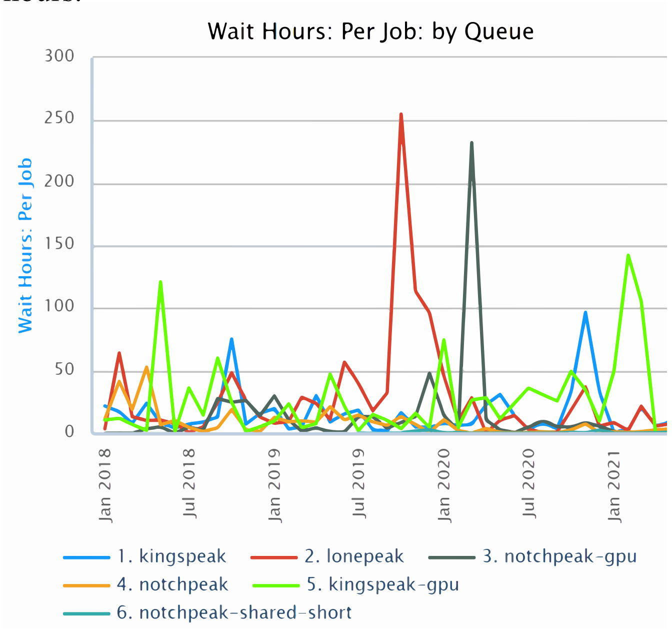


Figure 2: The daily average wait time of each of the general cluster partitions for the time period of Jan 1-Jun 30, 2021. See the caption of Figure 1 for additional information



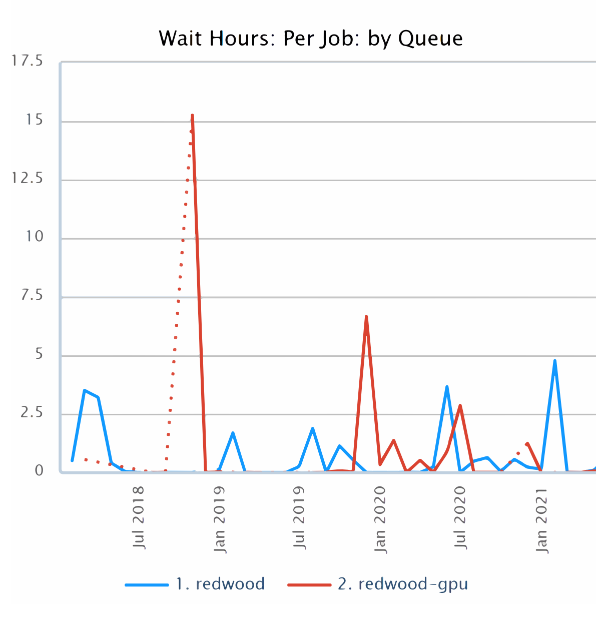


Figure 1: The monthly average wait time of each general cluster partition in the general (left) and Protected Environment (right). Access to the notchpeak and redwood partitions requires that the user's group has an active allocation. Users need to request access to the gpu partitions; this partition should only be used for jobs that make use of the GPUs.

Using the Spack Package Manager to Install Programs

Martin Cuma, CHPC Scientific Consultant

CHPC staff are in the process of deploying the Spack package manager (*https://spack.readthedocs.io/en/latest/*) for building and installing applications on our Linux systems. The major advantages of using a package manager like Spack, as compared to manual installation,are simplicity and reproducibility. Spack provides a more robust approach than our past hand-built applications. Each application known to Spack has a "recipe", a specification file how to build that application, which reduces the need to learn details on how to install the program.

The original plan was to do a complete deployment as we update our clusters' operation system, however due to recent changes in the Linux OS landscape that postponed the OS updates, the decision was made for a phased in roll in of Spack into the current CHPC applications structure. Currently we have some applications built with Spack using a default `gcc/8.3.0` compiler, which users can see by loading the `gcc/8.3.0` module and running `module avail.` Users will see a new section, like that shown at the bottom of this page.

Spack has many design goals, one of which is to be usable both by systems administrators and by users. Users can thus use CHPC installed Spack to build their own programs, stored in their own user space (e.g., a home directory). Spack installs all the applications that it built in a repository. In this article we first describe how users can take advantage of the CHPC Spack installation for application installations in their own space.

Setting up Spack in the user space

Our users can use the CHPC Spack repository, where they don't have write permissions, as an "upstream" to tell Spack to use the applications, thus reducing the storage need for their own repository in their own user space, and also save time in not having to build the dependencies that we have already built, by following the steps outlined below.

First create the user setting files and the user Spack directories, by running the following script:

```
/uufs/chpc.utah.edu/sys/installdir/spack/user/set-
up-user-spack.sh
```

This needs to be done only once. The steps taken in this script are detailed in *https://github.com/CHPC-UofU/spack-config/blob/master/readme-user.md.*

Once this is done, whenever you want to use Spack load the module by:

```
module load spack
```

Building a program with Spack in the user space

Most programs have dependencies – programs or libraries that are necessary to build the program. Because we have set up Spack to use the CHPC repository as an upstream, many of the common dependencies may have already been installed. The `spack spec` command, which asks Spack to generate a list of dependencies needed to build a program, can show what dependencies are available and which ones are missing. For example, for the program Octave – an open source replacement for Matlab- we can:

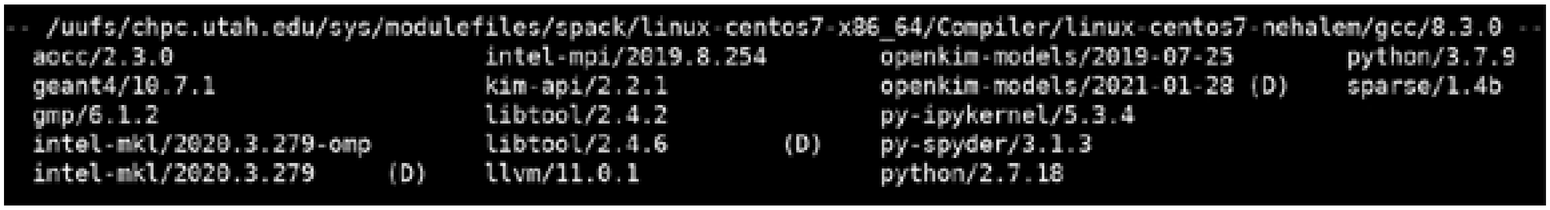
```
$ spack spec -I octave target=nehalem
Input spec
--------------------------------
 -   octave arch=linux-None-nehalem

Concretized
--------------------------------
[-]   octave@6.2.0%gcc@8.3.0~arpack~curl~fftw~fltk~-
fontconfig~freetype~gl2ps~glpk~gnuplot~hdf5~jdk~ll-
vm~magick~opengl~qhull~qrupdate~qscintilla~qt+read-
line~suitesparse~zlib arch=linux-centos7-nehalem
[^]    ^intel-mkl@2020.3.279%gcc@8.3.0~ilp64+shared
threads=none arch=linux-centos7-nehalem
[^]            ^cpio@2.13%gcc@8.3.0 arch=linux-cen-
tos7-nehalem
[^]          ^pcre@8.44%gcc@8.3.0~jit+multibyte+utf
arch=linux-centos7-nehalem
[^]        ^pkgconf@1.7.3%gcc@8.3.0 arch=linux-cen-
tos7-nehalem
[^]         ^readline@8.0%gcc@8.3.0 arch=linux-cen-
tos7-nehalem
[^]          ^ncurses@6.2%gcc@8.3.0~symlinks+termlib
arch=linux-centos7-nehalem
```

The [^] denotes packages used from upstream, [-] packages that are missing in the upstream, [+] denotes a package installed in the user repository.

Now we can build the new program which will then store the build in `$HOME/spack/local/builds:`

```
spack install octave target=nehalem
```



```
-- /uufs/chpc.utah.edu/sys/modulefiles/spack/linux-centos7-x86_64/Compiler/linux-centos7-nehalem/gcc/8.3.0 --
   aocc/2.3.0                        intel-mpi/2019.8.254        openkim-models/2019-07-25       python/3.7.9
   geant4/10.7.1                     kim-api/2.2.1               openkim-models/2021-01-28 (D)   sparse/1.4b
   gmp/6.1.2                         libtool/2.4.2               py-ipykernel/5.3.4
   intel-mkl/2020.3.279-omp          libtool/2.4.6        (D)    py-spyder/3.1.3
   intel-mkl/2020.3.279      (D)     llvm/11.0.1                 python/2.7.18
```

Notice that we told Spack to build the program for the CPU target `nehalem`. That is the lowest common denominator for all the CPUs that CHPC clusters and workstations use. We will describe building optimized executables for specific CPUs in a future article.

We have set up Spack to not generate environment module files for programs automatically, as that would result in a large number of modules for all the dependencies; therefore we need to generate the module after installation. The first step is to find the right Octave version among all the available builds – there will be some from the CHPC repository as well:

```
$ spack find -pl octave
...
ahjj6xh  octave@6.2.0    /uufs/chpc.utah.edu/common/
home/u0123456/spack/local/builds/linux-centos7-ne-
halem/gcc-8.3.0/octave-6.2.0-ahjj6xhs46mk4zycd-
q7irpnwhyojxp46
...
```

Look for the one with the path in your home. Overall, the `spack find` command is good for querying what programs are installed with Spack. Notice the unique string, `ahjj6xh` in this case, at the start of the line, called the hash. The hash uniquely identifies the installed package. Another unique identification would be through the complete specification of the build, though this can get long. You can see it by adding `-dv` options to the `spack find` command. Using the hash to build the module file:

```
$ spack module lmod refresh /ahjj6xh
```

Once approval to proceed has been given, the module file gets generated.

To make the new Octave module active, we need to first load the `gcc/8.3.0` compiler module that was used for the installation, and then also add the module path with the user built programs:

```
module load gcc/8.3.0
module   use   $HOME/spack/local/modules/linux-cen-
tos7-x86_64/Compiler/linux-centos7-nehalem/
gcc/8.3.0
```

Now we should be able to see the new Octave module with:

```
$ module avail

...
    /uufs/chpc.utah.edu/common/home/u0123456/spack/
local/modules/linux-centos7-x86_64/Compiler/
linux-centos7-nehalem/gcc/8.3.0
   octave/6.2.0
…
```

Next steps

Be aware that the Spack built packages can take a large amount of disk space, especially if you have the default 50 GB disk quota. You can use the command `mydiskquota` to display current usage. To see how much is used by spack, run:

```
du -hs $HOME/spack/local
```

Watch for CHPC announcements regarding Spack availability and new features. There are many aspects of Spack that we are exploring, including CPU optimized builds, management of Python or R libraries, build automation, mirrors or containers to plug into the cloud infrastructure, etc.

If you run into any problems with Spack, contact our help desk.

Jupyter Notebook Servers: Juno and Notebook Retiring

Brett Milash, CHPC Scientific Consultant

CHPC will retire the *juno.chpc.utah.edu* and *notebook.chpc.utah.edu* Jupyter Notebook servers on September 3, 2021. Juno and notebook, hosted on virtual machines, do not offer scalable access to hardware, require additional system administration, and in the case of juno does not offer home directory or shared filesystem access.

Notebook has been made redundant by our Open OnDemand service *https://ondemand.chpc.utah.edu* and we encourage users to migrate to this service. Open OnDemand, described at *https://www.chpc.utah.edu/documentation/software/ondemand.php*, is a web portal that provides access to Jupyter Notebook, Jupyter Lab, as well as a variety of other software packages, and is available for use on both the interactive and compute nodes of all CHPC's clusters.

As mentioned above, *juno.chpc.utah.edu* does not require a CHPC account and does not offer home directory or shared filesystem access and has most commonly been used for teaching purposes. Other Jupyter notebook hosting alternatives are *https://mybinder.org*, which we are using for CHPC's Introduction to Python training series, and *https://colab.research.google.com*, which has been used by faculty in the Department of Engineering.

For more information on jupyter notebook options at CHPC see *https://www.chpc.utah.edu/documentation/software/jupyterhub.php*. If you have any questions on the retirement of juno and notebook, need any assistance moving your usage to the other options, or have any questions on the use of jupyter notebooks please reach out to us via *helpdesk@chpc.utah.edu*.

The University of Utah
University Information Technology
Center for High Performance Computing
155 South 1452 East, Room 405
SALT LAKE CITY, UT 84112–0190

Thank you for using CHPC resources!

Welcome to CHPC News!

If you would like to be added to our mailing list, please provide the following information and via the contact methods described below.

Name:
Phone:
Email:

Department
or Affiliation:

Address:
(campus
or U.S. mail)

Please acknowledge the use of CHPC resources!

If you use CHPC computer time or staff resources, we request that you acknowledge this in technical reports, publications, and dissertations. An example of what we ask you to include in your acknowledgments is:

"A grant of computer time from the Center for High Performance Computing is gratefully acknowledged."

If you make use of the CHPC Protected Environment, please also acknowledge the NIH shared instrumentation grant:

"The computational resources used were partially funded by the NIH Shared Instrumentation Grant 1S10OD021644-01A1."

Please submit copies or citations of dissertations, reports, pre-prints, and reprints in which CHPC is acknowledged in one of the following ways:

Electronic responses

By email: *helpdesk@chpc.utah.edu*
By fax: (801) 585–5366

Paper responses
By U.S. mail: 155 South 1452 East, Rm 405
Salt Lake City, UT 84112–0190

By campus mail: INSCC 405