Center for
HIGH PERFORMANCE COMPUTING
THE UNIVERSITY OF UTAH

Newsletter

Fall 2018

Enabling Innovation in Numerical Prediction of High-Impact Weather Systems

Research Highlight · Zhaoxia Pu, Department of Atmospheric Sciences

Along with the rapid advancement in computer technology, numerical weather prediction (NWP) has become a central component of modern weather forecasting. In the United States, daily weather forecasting begins with a supercomputer at the National Oceanic and Atmospheric Administration (NOAA) in Washington, DC. Around the world, most countries use NWP as key guidance for their operational weather prediction.

The basic concept of NWP is to solve a set of partial differential equations (PDEs) that govern atmospheric motion and evolution. This set of PDEs describes basic conservation laws, including the conservation of momentum, mass, energy, and water vapor. In order to predict the atmospheric state in the future, we must integrate this set of equations forward, starting from the current atmospheric state. Therefore, NWP is an initial value problem: given the current atmospheric conditions (initial conditions), we integrate the set of PDEs to obtain future atmospheric states. However, the analytical solution for the set of PDEs is impossible. The equations must be solved using the discrete form with numerical methods. Therefore, finite difference equations (FDEs) are used to find solutions to the PDEs with computational methods. The FDEs are solved at grid points in either global or regional domains; thus NWP model domains are divided by a gridded mesh. The fine-gridded (high resolution) mesh helps provide better representation of atmospheric states and achieves more accurate NWP, but it also requires a lot of computational storage and fast CPUs to guarantee efficiency. In addition, to achieve a successful NWP, accurate initial conditions and good understanding of physical processes (such as interactions among the atmosphere, earth surface, solar radiation, clouds and precipitation, etc.) are essential. Although significant progress has been made toward improvements in both research and application, we are limited by our incomplete observing system and our knowledge of how to accurately represent physical processes in NWP. Uncertainties in initial conditions and physical parameterizations always limit predictability. To deal with these uncertainties, ensemble forecasting has been used in modern operational practice. Ensemble forecasting produces a set of multiple predictions from different initial conditions or with various credible versions of models. Ensemble averaging not only improves forecasts but also provides information about forecast uncertainty. However, ensemble forecasting demands massive computer resources, including storage capacity and large numbers of CPUs. Therefore, modern NWP has been highly associated with and heavily reliant on computational resources. Around the world, many of the most powerful supercomputers are used in NWP.

Professor Zhaoxia Pu
Department of Atmospheric Sciences

At the University of Utah, high-performance computing has wholly or partially supported essential research projects on NWP with innovative science and technology advancements. Dr. Zhaoxia Pu, professor of the Department of Atmospheric Sciences, and her research group (*http://www.inscc.utah.edu/~pu/myhome/research_page.html*) devote studies to improving numerical prediction and understanding of high-impact weather systems, including tropical cyclones, hurricanes, mesoscale convective systems, mountainous fog, and flows over complex terrain. For most of the research, hundreds and sometimes thousands of CPU processors are used for a single set of numerical experiments. Over the past ten years, the group has

Diagram of the procedure of NWP: executing a numerical model on a supercomputer to produce a daily forecast.

obtained significant federal research funding support from NSF, NOAA, NASA, the Office of Naval Research (ONR), DOD, and DOE, leading to about 70 peer-reviewed journal publications (*http://www.inscc.utah.edu/~pu/myhome/publication.html*) and the graduation of eight Ph.D. students and nine M.S. students.

Data Assimilation

Data assimilation is an advanced research area that focuses on making the best use of observations to form initial conditions in NWP. Prof. Pu's research emphasizes developing new methods to utilize satellite and radar data in NWP, not only for improved prediction and numerical simulations but also for better understanding the mechanisms of high-impact weather systems, such as the major factors controlling the rapid strengthening of tropical cyclones. With NSF and ONR support, she and her student have made significant progress in understanding tropical cyclone intensity changes and the evolution of landfalling hurricanes with satellite and airborne Doppler radar data assimilation. The group also joined the NOAA Hurricane Forecast Improvement Program (HFIP) and has contributed to the improved representation of hurricane vortex initial conditions in NWP.

Notably, with NASA support, Prof. Pu and her students supported a recent NASA satellite mission, the Cyclone Global Navigation Satellite System (CYGNSS; *https://www.nasa.gov/cygnss*). They conducted both prelaunch and postlaunch data assimilation studies to demonstrate the impacts of CYGNSS-measured ocean surface wind speed on numerical simulations and prediction of tropical cyclones with a research version of NOAA's operational hurricane model. In addition, Prof. Pu and her team have also been involved in NOAA's research efforts to develop data assimilation methods for the nation's next-generation global prediction system.

Mountain Terrain Atmospheric Modeling and Observations

With support from the Department of Defense Multi-University Research Initiative program, Dr. Pu and several other professors and scientists in the Department of Atmospheric Sciences have been involved in studying the predictability of flows over mountainous terrain at the mesoscale (error growth—the sensitivity to initial conditions at various lead times—in particular), and developing meaningful measures of skill relative to appropriate conditional climatologies (i.e., the skill of capturing specific phenomena when they are supposed to appear; e.g., turbulence generation when a Richardson number criterion is satisfied). With this project, Prof. Pu's research has emphasized investigating the sensitivity of model forecasts to input properties (initial conditions and model parameters) and boundary conditions, data assimilation, and comparison of different techniques in order to improve analyses and forecasts over regions of complex terrain. During the field program of the project, Prof. Pu's team also conducted a real-time forecasting on CHPC Linux clusters to support field planning and activities (*http://www.inscc.utah.edu/~pu/slc/index.html*). Upon completion of the project, Prof. Pu's research has continued to expand in this area, with a focus on data assimilation over complex terrain and the predictability of wintertime fog and severe winter storms.

Tropical Cyclone Formation and Rapid Intensification

The dynamical and physical processes that control tropical cyclone (TC) genesis and strengthening have not been well understood, limiting our ability to predict tropical cyclones and hurricanes. Dr. Pu and her team have participated in several ONR and NASA-funded field programs dedicated to improving our understanding of TC formation and rapid intensification. The objectives of their studies are to investigate large-scale environmental conditions, mesoscale phenomena, and small-scale convective bursts as well as their interactions that are responsible for TC formation and intensity changes. Specific areas include:

- characterizing the intensity of convection over the western Pacific and Atlantic Oceans from radar, aircraft and satellite data

- deriving an accurate mesoscale environment of convective systems through the assimilation of satellite, radar, lidar and in-situ data
- evaluating the quality of the global forecast systems, such as NOAA and U.S. Navy global ensemble forecasting, for accurate TC analyses and forecasts
- understanding the environmental factors that determine tropical cyclone formation and rapid intensification.

Interaction Between Landfalling Hurricanes and the Atmospheric Boundary Layer

Accurate forecasts of the intensity and structure of a hurricane at landfall can save lives and mitigate social impacts. However, among recent efforts in hurricane forecast improvements, few studies have focused on landfalling hurricanes. This reflects the complexity of predicting hurricane landfalls and the uncertainties in representing the atmospheric boundary layer conditions in NWP models. The aim of Dr. Pu's research is to study the interaction between landfalling hurricanes and the atmospheric boundary layer using ensemble-based data assimilation by incorporating Doppler radar observations with other available in-situ and satellite data into an ensemble data assimilation system with the community mesoscale weather research and forecasting (WRF) model and NOAA hurricane WRF (HWRF) model to address the related science questions. If computing resources are available, the group also conducts real-time hurricane forecasting during the hurricane season (*http://www.inscc.utah.edu/~pu/real_hfip/index.html*).

Four-Dimensional Atmospheric Boundary Layer Structure for Cloud Lifecycles

The principal objective of Dr. Pu's research on this topic is to create realistic estimates of high-resolution (1 km-by-1 km horizontal grids) atmospheric boundary layer structure and characteristics of precipitating convection, including updraft and downdraft cumulus mass fluxes and cold pool properties over a region the size of a global general circulation model grid column. This is done from analyses that assimilate the surface Mesonet observations of wind, temperature, and water vapor mixing ratio and available profiling data from single or multiple surface stations using advanced data assimilation methods.

Future Direction

Looking ahead, Dr. Pu's group anticipates continuing their innovative research efforts with high-performance computing. Considering the fast growth of observational data and the demands of conducting many new studies with high-resolution NWP models, they are also looking forward to developing big-data science and applying future increases in computer power.

Recent Updates

Allocation Policy Change

We are making a change to the allocation policy, effective the next allocation cycle (requests due December 1 for the quarter starting January 1). Currently, all allocation requests—regardless of size—use the same allocation request form and are all reviewed by the CHPC allocation committee. Moving forward, there will be two categories of requests: those with 20,000 or fewer core hours per quarter and those with more. The form and review process for requests of more than 20,000 core hours per quarter will not change; however, the requests for fewer core hours will have a simplified request form. Smaller requests will require only the project title, PI name, core hours requested, abstract, sources of funding, and a list of publications and products based on CHPC resources. All complete requests will be awarded without review by the allocation committee.

The next announcement for allocation requests, sent in early November, will provide links to both forms.

Memory Added to Node Features

Node features can be used to limit the nodes targeted by a Slurm job (with the Slurm constraint directive `#SBATCH --constraint` or `#SBATCH -C`). The node features include the owner group, core count of the node, and GPU type (if applicable). We have recently added a nodes' memory to the available features. The format for the feature is the letter "m" followed by the amount of memory (in GB). For example, to limit requested nodes to those with 64 GB of memory, the constraint `m64` can be used.

Using this memory constraint differs from using the memory Slurm directive (`#SBATCH --mem`), which limits nodes available for a job to those with *at least* this amount of memory (and limits the job to the defined amount of memory, even if the assigned node has more). Using the new memory constraints will instead limit the nodes considered for the job to those with the exact specified quantity of memory.

Updated Message of the Day

We have recently updated the Message of the Day (MOTD), which is displayed when users log in to most Linux resources. In the past, this message has only been updated to announce downtimes and major changes to the systems (such as updates to OS versions) and has quickly become out-of-date. Now, the MOTD will also include additional information such as a tip about using various resources (selected from a collection assembled by CHPC staff) and will provide more time-sensitive information.

Keeping Interactive Nodes Responsive: A New Version of Arbiter

Technology · Dylan Gardner, Robben Migacz, and Paul Fischer

In a high-performance computing environment, shared resources are often the most capricious and unreliable. The performance of each process is affected by the behavior of all users. If a user is behaving poorly on a shared resource (running a parallel job or consuming a large fraction of the total memory), processes using that same resource can slow to a crawl, run out of memory, or spend unreasonable amounts of time on network transactions and input and output operations.

Arbiter is a daemon intended to warn the user about such behaviors and help limit resource usage that affects the work of others on interactive nodes, which should only be used for small, lightweight tasks such as scripting, compiling, and transferring data (described in *CHPC Policy 2.1.1*). An old version of the script has been in place for several years, vigilantly keeping watch and sending reprimanding emails to users, though not without flaws and with rather limited information on usage. Many users are likely familiar with the stern emails imploring them to change usage patterns on interactive nodes.

The new version of Arbiter provides extended functionality, including the ability to enforce dynamic limits on the resources available to a given user and to collect detailed historical usage information for plots and email messages. The improved version of the script provides more detailed information about acceptable usage in emails (to better educate users who are not familiar with policies) and reduces the need for administrator action and supervision by nondestructively and automatically limiting the resources available to a given user.

Evaluating Users

Both the original and new Arbiter scripts calculate and store a user's "score," which is a metric indicating the extent of resource usage not suited for interactive nodes over time, to determine when to notify administrators and the responsible user. As a user runs large computational tasks, his or her score increases with time. If he or she stops using extensive resources (and returns to a workload typical on interactive nodes), this score will instead decrease, albeit more slowly.

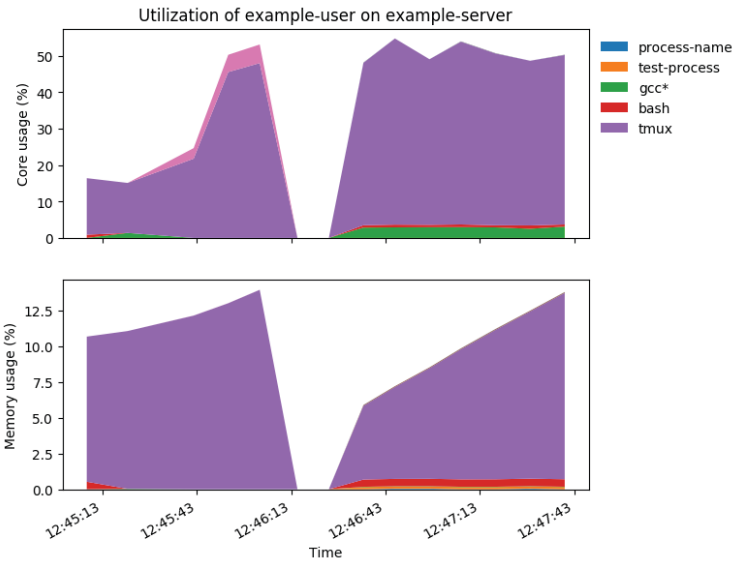
The old Arbiter script sends notification emails when users' usage might impact others on the node (when the score determined by usage metrics reaches a threshold) but is incapable of limiting or stopping errant processes without administrator intervention. It also compares usage values to a simple threshold while the new script introduces scaling: consuming more resources increases the score more quickly. In both scripts, a whitelist of acceptable processes, such as compilers, ensures the effect on users who are using interactive nodes as intended is minimal.

Interpreting Usage Metrics

The original Arbiter collects instantaneous usage information when it sends notification emails (to identify the processes that might impact other users), but this information can be deceptive; there may not be one single unacceptable process or the processes that caused a user's score to accumulate may have stopped by the time the email is sent. The workload information described in the message, therefore, might not necessarily reflect the true workload that affected other users on the node and cannot help the user correct his or her behavior. With the new Arbiter, we are hopeful that easy-to-digest usage information in emails, including a plot of usage over time and a table of high-impact processes, will help users identify and correct behavior that could adversely affect other users on interactive nodes. A visual representation of system load will also help users identify trends and quickly recognize tasks that would be better suited on other resources.

Implementing Arbiter

The new version of Arbiter, written in Python 3, makes use of *cgroups*, a feature of the Linux kernel that provides resource accounting features, to dynamically monitor and limit resource utilization. Users who run processes requiring extensive computational resources—currently memory and CPU, with other metrics possible at a later date—on interactive nodes might see processes throttled temporar-



An example of a plot of resource utilization generated by the new Arbiter. The CPU and memory usage of each process is shown over time to help users better understand their effect on shared resources.

ily to reduce their impact on other users connected to the same node. After a short time, the limitations are removed. If a user has rectified his or her usage, no further steps are taken. However, if the user's usage is still not acceptable, he or she will be limited further and for a longer period of time. This way, processes consuming excessive resources need not be killed but will suffer degraded performance that should dissuade users from abusing interactive nodes.

We expect to deploy the new version of Arbiter on interactive nodes, after testing and adjusting features, in the coming weeks.

Introducing Juno: CHPC's New JupyterHub Service

Technology · Luan Truong

CHPC's long-term JupyterHub cluster is here! This is a service that will be open freely to all of campus and attached to each student or staff member's CIS credentials. The initial deployment, based on Kubernetes, is composed of a virtual machine backed with three servers with a total of 2.5 TB of memory and infinite scalability. For more details on the architecture, please see our whitepaper, which can be found at *https://www.chpc.utah.edu/documentation/white_papers/index.php*.

The JupyterHub deployment is based on Kubernetes to provide flexibility and scalability.

We set the cull time—the time before a server is automatically shut down—to be two minutes after the user closes his or her browser to prevent the wasteful locking of resources. In addition, Juno is not geared for tasks that need large resources or long run-times; in these cases, we recommend you explore the use of the Open OnDemand web portal, described at *https://www.chpc.utah.edu/documentation/software/ondemand.php*.

General Instances

The general instances are open to all users who have a University of Utah uNID. Every user, upon logging in, will be allocated 1 core, 2 GB of RAM, and 10 GB of disk space. There are multiple base general instances with different functionalities:

- juno.chpc.utah.edu—Python 3 with common scientific Python packages
  - Users can add additional Python packages via `conda` or `pip install` with the `--user` flag
- ds-juno.chpc.utah.edu—Julia and R
  - Additional R packages can be added using the same procedure used the clusters

You can pull repositories and files into notebooks with `wget` or `git` from the terminal. To access the instances, you need only navigate to the appropriate URL and enter your university credentials. Your notebook is operational as long as you are interacting with it.

Customized Instances

Customized instances can be created upon request. We envision these being used to meet the specific needs of our researchers, for hands-on presentations on Linux and Python, and for classes. There are a number of custom environments for scientific research and teaching that can be readily installed. A list of predefined containers can be found at *https://hub.docker.com/r/jupyter/*.

On custom instances, administrative privileges can be granted (to multiple people if desired). Administrative users have the ability to access others' notebooks, see their actions, stop and restart users' servers, and turn off and on the entire instance. This is a great mechanism for teaching classes or removing unwanted users from your customized instance.

If there is a specific instance or environment you would like to work in or alternative kernels you would like for your notebook (such as R or Julia), please open a ticket with the CHPC help desk with the following information:

- The name of the base image (from *https://hub.docker.com/r/jupyter*)
- Additional modifications to the image (such as software like "scp," "zgrep," or Python packages)
- The purpose of the custom instance
- The estimated number of users
- Per-user sizing (resource allocation)
- The desired URL (for example, chemeng3200-juno.chpc.utah.edu or physics5220-juno.chpc.utah.edu)
- An expiration date
- A list of administrative users

Change to Slurm Policy on Clusters

CHPC recently made a change to certain Slurm "Quality of Services" to cap the number of jobs a given user can submit to a given account and partition combination to 1,000. This change was applied to the general, freecycle, GPU, and guest partitions of all clusters (e.g., kingspeak, kingspeak-freecycle, notchpeak-gpu, notchpeak-gpu-guest, etc.) to alleviate issues observed when many jobs are in the queue. We will monitor the effect of this limit and the value chosen to determine whether we need to adjust the setting.

Running Multiple Independent Tasks in Parallel on CHPC Clusters

Usage Tip · Brett Milash and Wim Cardoen

Some data analysis projects require running a large number of independent tasks, preferably in parallel. Some analysis programs don't take advantage of the multi-core processing on the compute nodes in our clusters. However, due to an overhead of job scheduling, it is more efficient to submit one Slurm job that can run many of these tasks rather than submitting each independent task in a Slurm job of its own.

Writing one Slurm script to execute multiple tasks can be challenging. There are several good strategies for writing an efficient Slurm script that can process many independent tasks in parallel. Previously, CHPC documented several options to accomplish this for the case of multiple serial jobs on a single node (*https://www.chpc.utah.edu/documentation/software/serial-jobs.php*).

Recently, we have become aware of two additional strategies: running multiple jobs on a single node with `xargs` and running across multiple nodes with the Launcher utility (*https://www.tacc.utexas.edu/research-development/tacc-software/the-launcher*) developed by the Texas Advanced Computing Center (TACC).

Running Multiple Jobs on a Single Node

If you intend to use a single node to run a set of independent tasks in parallel, the Linux shell command `xargs` is a great tool. In its simplest form, the `xargs` statement reads values from its standard input and applies a command to each value that was read. For example, to compress all the text files in the current directory:

```
$ ls *.txt | xargs gzip
```

A more interesting example would be to compress all the text files in the current directory and its subdirectories:

```
$ find . -name "*.txt" -print
  | xargs gzip
```

To run in parallel, simply use the `-P` option, specifying the number of processes to run in parallel:

```
$ find . -name "*.txt" -print
  | xargs -P 5 gzip
```

This works nicely with Slurm, since the `SLURM_NTASKS` environment variable is set automatically when a job starts. The variable can be used to define the number of cores to use for the task:

```
$ find . -name "*.txt" -print
  | xargs -P $SLURM_NTASKS gzip
```

In this example, `xargs` will run `SLURM_NTASKS gzip` processes in parallel, starting a new process when one is completed. The command passed to xargs can even be a shell function, assuming you are using `bash` and the function has been exported with `export -f`. An alternative to reading the command arguments from the standard input is to specify a file using the `-a` or `--arg-file` options. Consult the manual page on `xargs` for more details.

Running Multiple Jobs Across Multiple Nodes

If your project requires multiple nodes, the Launcher utility from TACC simplifies the task of spawning multiple tasks in parallel across multiple nodes. (Thanks to CHPC user Brian Lohman for bringing Launcher to our attention.)

To use Launcher, you must enter all of your parallel commands into a single file with one command per line. Then, create a Slurm script to start the launcher:

```
#!/bin/bash

# Simple Slurm script for submitting
# multiple serial jobs (e.g. parametric
# studies) using Launcher.

#SBATCH -J your_jobs_name
# Number of nodes to use
#SBATCH -N 4
# Total number of concurrent tasks
# (across all nodes)
#SBATCH -n 8
# Name of file for standard output
#SBATCH -o Parametric.o%j
# Total run-time
#SBATCH -t 01:00:00
# Account and partition
#SBATCH --account=cluster_account
#SBATCH --partition=cluster_partition

module load launcher

# Set the command file name (required)
export LAUNCHER_JOB_FILE=command_file

# Run the tasks
paramrun
```

Finally, submit your Slurm script with the `sbatch` command. In this example, the job will use 4 nodes and will run your commands in parallel, 2 on each node, for a total of 8 concurrent processes. The `paramrun` command will execute the contents of `LAUNCHER_JOB_FILE`.

Avoiding Preemption with Slurm Constraints

Technology · Robben Migacz

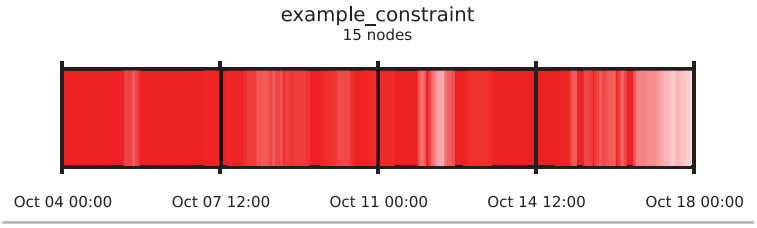
When running jobs as a guest (generally with the "owner-guest" account and partitions ending in "-guest"), jobs are vulnerable to preemption if a user from the group that owns the node runs a job requiring the same resources. Users can make use of the owner group name (defined as a node feature in Slurm) to target a subset of owner nodes with *constraints* in the same manner as requesting nodes with a certain core count or amount of memory. This can help reduce the likelihood of preemption; owner nodes with lower owner utilization are more likely to have long-running guest jobs. To aid the targeting of nodes, CHPC now publishes utilization information per owner group for the past two weeks (at *https://www.chpc.utah.edu/usage/constraints/*).

The suggestions are made by evaluating historical usage in Slurm logs. Every job in the log is associated with the nodes it ran on, which are then evaluated in aggregate for each constraint. The size of each constraint (the number of nodes it contains) and the average utilization and frequency of new job submissions to nodes associated with a constraint are considered for recommendations, which are presented visually to help users evaluate options quickly.

Previous constraint utilization information can be used to help guide the selection of node features in the future. Keep in mind, however, that past usage is not indicative of future usage and should only be used to help gauge the activity patterns of groups (doing so may help reduce preemption, but this is by no means guaranteed). You can also specify multiple constraints to increase the pool of available resources. To specify a set of nodes from multiple owner groups, you can use the logical operator "|"; for example, use

```
#SBATCH -C "feature1|feature2|feature3"
```

to target multiple node features (such as group names) simultaneously. The features of a given node can be acquired with the `si` or `si2` alias (providing additional parameters to `sinfo`) on the CHPC Slurm documentation.



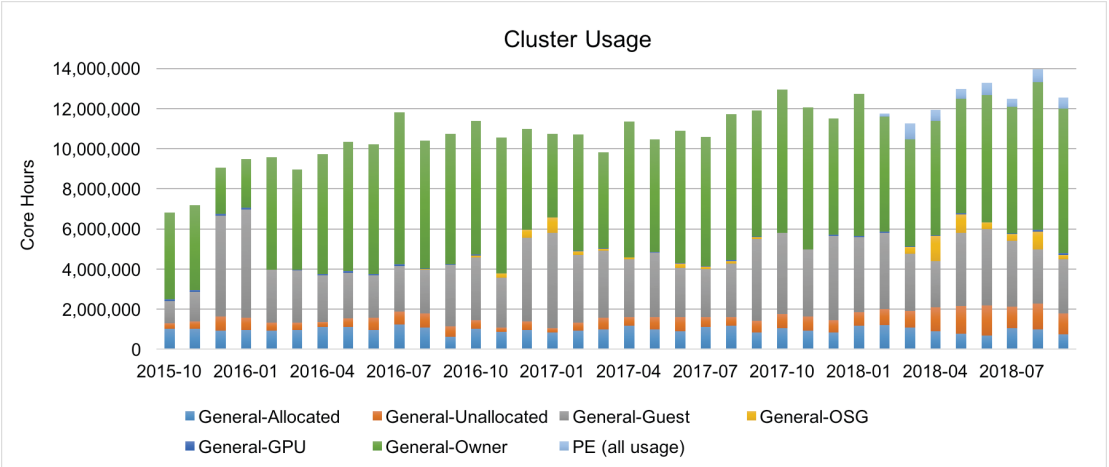
An example of the visual representation of constraint utilization. A white background represents 0% owner usage—no owner jobs running on any nodes—while a red background represents 100%.

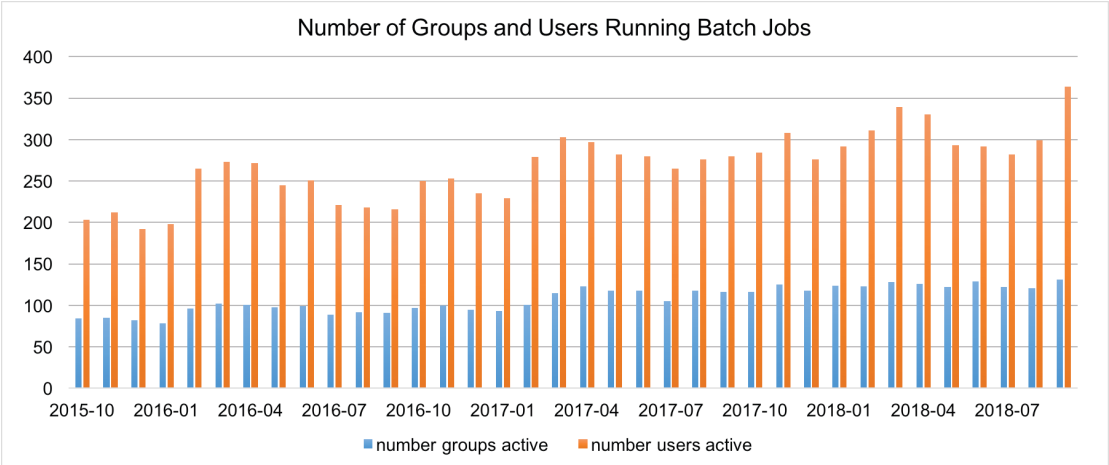
Continued Growth in Usage

The utilization of CHPC resources has been steadily growing over the past few years.

Over 113 million core hours have been used by more than 650 users year-to-date (September 30). The number of core hours is up 15% from the same time period in 2017.

The number of active users, too, has increased to a record 364 in the month of September 2018 (an active user is one who runs at least one batch job in a given time period).





Center for
HIGH PERFORMANCE COMPUTING
THE UNIVERSITY OF UTAH

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 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