Center for
HIGH PERFORMANCE COMPUTING
THE UNIVERSITY OF UTAH

Summer 2022

Turning Weather and Climate Research into Actionable Science

Jon Meyer, Utah Climate Center, Department of Plants, Soils, and Climate, Utah State University

The Utah Climate Center, hosted by the College of Agriculture and Applied Sciences at Utah State University, serves a mission of weather and climate 'research-to-operations' (R2O). Within the backdrop of changing climate, the R2O initiative is meant to help facilitate academic endeavors that are focused on actionable science products. R2O is well suited to address the dynamic and ever-changing suite of climate service needs at the state and federal level.

The University of Utah's Center for High Performance Computing (CHPC) serves a critical role in supporting the Utah Climate Center's R2O mission initiatives by providing the bedrock of the computational and system administration expertise upon which many of the climate service platforms are built. CHPC offers the Utah Climate Center reliable and dedicated computational resources with the benefit of minimal system downtime over long periods of time. Such resources with limited downtime are critical for the automated, real-time software systems to be able to provide reliable and on-time information stakeholders seek. Compute nodes on the NOTCHPEAK partition are employed by the Utah Climate Center to handle the collection of automated software platforms designed to perform external data ingestion and processing as well as for running of computationally intensive real-time operational forecast modeling.

In the face of the recent extreme drought, the Utah Climate Center has focused on building a comprehensive Utah Drought Dashboard to better monitor and assess drought condition. This dashboard (viewable at *https://climate.usu.edu/service/index.php*) integrates numerous internal and external sources of weather and climate information into a 'one stop shop' website. CHPC resources have shouldered a great deal of the computational backend needed to monitor drought conditions, with the recent implementation of real-time soil moisture mapping being a major accomplishment. Each day, station data is downloaded and quality controlled. Final form data points are directed to the Utah Climate Center's servers, where daily maps of soil moisture conditions are hosted. Figure 1 shows an example of the color-coded daily soil moisture observations.

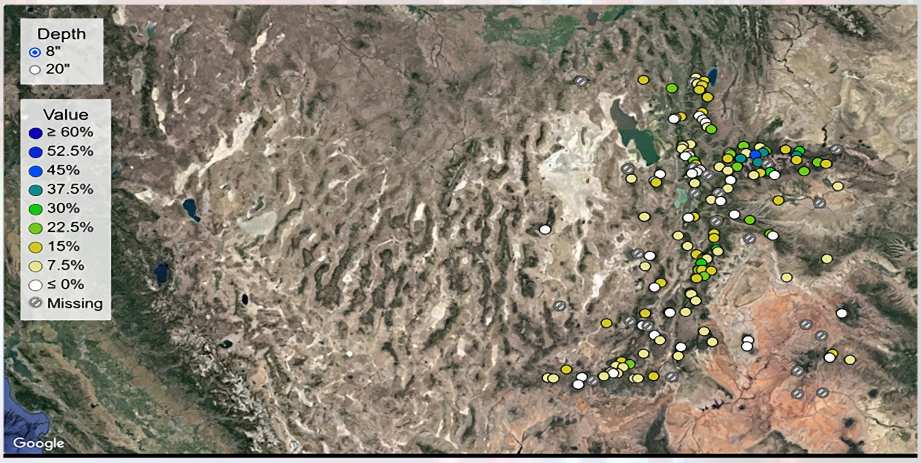


Figure 1: Example of a map of daily soil moisture conditions color coded to visualize each station's percentage of saturation. Maps are updated daily at the Utah Climate Center's drought dashboard website.

Behind this daily soil moisture map is a process that involves CHPC software that tabulates hourly soil moisture observations from approximately 225 surface weather station locations. In addition to daily considerations, these hourly soil moisture conditions are also passed

Figure 2: (left) Coverage of soil moisture stations within Utah and (right) an interpolated map of high-resolution soil moisture conditions demonstrating the blending and overlaying of the observation-based Utah map with the lower-resolution NCEP modeled soil moisture conditions.

through a routine that extracts 6-hourly conditions across the state and performs a statistical interpolation and blending between modeled soil moisture conditions to create high-resolution maps covering the region. Figure 2 illustrates the distribution of surface soil moisture observations ingested by the Utah Climate Center (left), with an example of the high-resolution 2-km interpolated map of soil moisture conditions being overlaid and blended with soil moisture conditions modeled by the National Center for Environmental Prediction (NCEP) North American Mesoscale (NAM) forecast model (right). The added performance afforded by CHPC allows the Utah Climate Center the capacity to perform more intricate interpolation and blending methodologies that yields a superior hybrid of soil moisture. The goal of these routines is to preserve the direct observations from point measurements, while accounting for changes in soil moisture conditions where direct measurements are unavailable.

In addition to data mining and processing, the Utah Climate Center also conducts real-time operational forecast modeling. While numerous forecast models are operated by NCEP, Utah's complex terrain and intricate climate processes limit a great deal of forecast fidelity by the national models. CHPC resources allow the Utah Climate Center's in-house modeling platforms to more closely focus on Utah's highly nuanced weather patterns through a methodology called dynamic downscaling. Dynamic downscaling uses forecasted conditions from a coarse-resolution 'parent' forecast model to supply initial and boundary conditions for a higher resolution forecast domain placed inside the parent domain's coverage. With complex terrain, the high resolution is especially important for Utah and leads to a much improved representation of weather and climate patterns.

Currently, the Utah Climate Center operates both a short range (3.5 day forecast) and a long range (30 day forecast) modeling platform based on the Weather Research and Forecasting (WRF) model. The short-range forecast model dynamically downscales the North American Mesoscale (NAM) model, which provides a 3.5-day forecast 4-times each day using a 12-km domain over the majority of North America. The dynamic downscaling approach allows the Utah Climate Center's forecast model to replicate the NAM model's 4x-daily, 3.5-day forecast, but at a much-improved 2-km resolution focused on Utah and neighboring states. The higher grid resolution greatly improves the mountain-valley gradients of temperature, wind, and precipitation. Figure 3 highlights the difference the higher resolution makes when forecasting localized convective thunderstorms driven by the summer monsoon. The left panel is based on the NCEP 12-km NAM output, while the right panel is from the Utah Climate Center's 2-km forecast domain. Not only are spatial patterns of precipitation location affected by the downscaling, but so is the forecast intensity of thunderstorms. The short-range forecast model is also used to help Utah's wintertime cloudseeding operations. Across the state, over 25 mountain cross-sections visualizing vertical profiles of temperatures, wind, cloud conditions, and humidity aids in the optimization of cloud seeding resource deployment.

In addition to the short-range forecast model, the Utah Climate Center operates a longer-range, sub-seasonal forecast model. Run twice a day over a 1-month forecast, this model platform dynamically downscales the Climate Forecast System (CFS) version 2 model with a western U.S. domain at 12-km, and a nested Utah-specif-

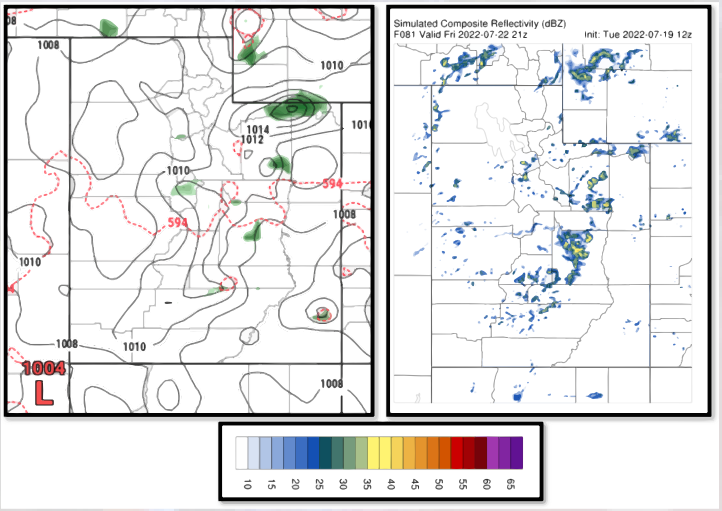


Figure 3: Comparison of forecasted monsoonal thunderstorms as they would appear on a hypothetical weather radar using the 12-km NCEP NAM model (left) and the 2-km Utah Climate Center model (right).

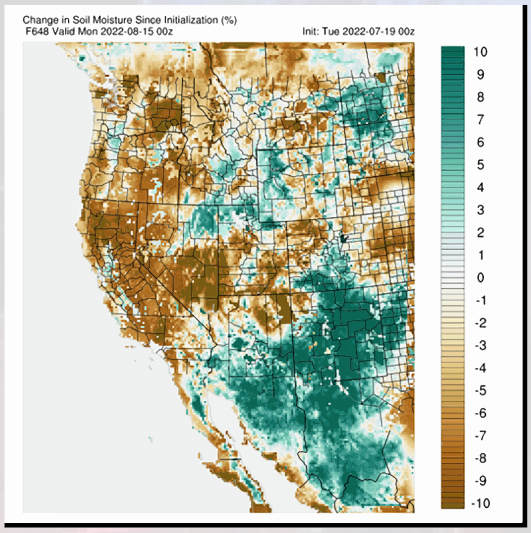


Figure 4: Example of a long-range one-month forecast of surface soil moisture changes for the 12-km western U.S. domain. Values represent a change in percentage of saturation from the initial model state and indicate regions of drying (brown) or wetting (green).

ic domain at 3-km. By going from CFSv2's ~56-km grid resolution to these much-higher resolutions, the Utah Climate Center is able to extract more reliable guidance into seasonal climate applications. Such applications are growing in number as the platform's utility becomes apparent, but initial applications focus on advanced snow melt timing and duration forecasts during the spring months to aid in streamflow and water resource decision making and summer soil moisture outlooks during the growing season. Figure 4 shows predicted 1-month soil moisture changes from the 12-km western U.S. domain. Areas of drying or wetting give guidance to drought monitoring and impact assessments while helping agriculture improve water use strategy during the growing season.

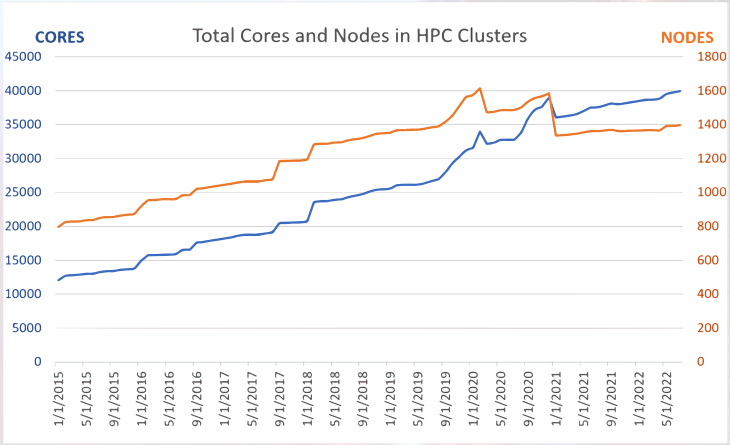
While still early in its development, the in-house modeling and data processing afforded to the Utah Climate Center by the CHPC has yielded early dividends while showcasing the power of pairing high-performance computing with an R2O initiative. The early testbed successes discussed here will no doubt serve as a springboard to help secure additional funding support to continue future application development and model evaluation. Without the ongoing support of the CHPC, progress in both academic knowledge and actionable science products addressing the immediate needs of Utah's stakeholders would be far more difficult to accomplish. Lastly, the inter-collegiate HPC collaboration between Utah State and the University of Utah that has allowed the Utah Climate Center access to such immense computational resources should be heralded for its overall return on investment to the state and its stakeholders.

CHPC Metrics

Anita Orendt, CHPC Scientific Consultant

As part of an ongoing project to track the growth of CHPC resources and usage we are collecting a number of metrics. The graph to the right shows the growth in number of nodes and cores in the HPC clusters over the last seven years, including both the compute and interactive nodes.

Among the noticeable changes in nodes and cores, we can identify the growth of lonepeak in September 2017 and again in early 2108. We can also identify the start of notchpeak in January 2018, along with the subsequent the addition of 32 general nodes, each 64 cores, to notchpeak in January 2020 followed by the retirement of ember in retired in February 2020. During December 2020, we see the impact of the retirement of about 3000 cores on ash.



Operation System (OS) Change on Linux Clusters and Servers

Martin Cuma, CHPC Scientific Consultant

In recent months CHPC updated the operation system (OS) on the Linux systems, including the general environment clusters, to Rocky Linux 8. This was a major OS update that happens roughly every 5 years and as such it came with some notable changes, that every user of our systems needs to be aware of. We summarize these changes in this article. For further information, please see the *online documentation*.

As of this writing only the Redwood protected environment cluster and a handful of custom Linux systems remain on the old CentOS 7 operating system. CHPC is working on a time table to complete the OS update on the remaining systems.

Access

The host names and access modes remain the same, however, we now only support *FastX 3*. Also, the list of supported terminals and desktops is different since the MATE desktop is not available in Rocky Linux 8.

Software

The most notable change is that we have fully deployed the Spack package manager for most software installations. Since Spack has different names for some programs and libraries, that we used to have, everyone needs to use and get used to these new names. In the table below we list the programs and libraries which changed names.

New Name	Old Name	Note
intel-oneapi-compilers	intel	Branding name change
intel-oneapi-mkl	mkl	Branding name change
intel-oneapi-mpi	impi	Branding name change
nvhpc	pgi	Manufacturer acquisition
netcdf-fortran	netcdf-f	
parallel-netcdf	pnetcdf	
quantum-espresso	qe	

Note also that users can use the Spack packages installed by the CHPC as a base for their own program installations using Spack. We have described this at our *Spack help webpage*.

The base OS compiler is now **gcc/8.5.0**. Other compilers include **intel-oneapi-compilers/2021.4.0** supplying the Intel compilers, and **nvhpc/21.5** supplying the Nvidia, formerly PGI, compilers.

Most packages and libraries retained the same module names, and many of the commonly used ones have been newly built with the newer compilers. From our experience so far, there is a good chance that a program built on the old CentOS 7 OS will work, but some do not. If you encounter an error running a program that has worked before, please first check if newer version exists and try the newer version. If there is no newer version, report this to CHPC help desk.

The network driver support that underlies the parallel programming support, especially the MPI libraries has also changed, which means that most old MPIs don't work. We have installed recent versions of Intel MPI (**intel-oneapi-mpi/2021.1.1**), OpenMPI (**openmpi/4.1.3**) and MVAPICH2 (**mvapich2/3.2.6**), which have been verified to work. Users are highly encouraged to re-build their MPI programs using these new MPI libraries. Note that the recent **intel-oneapi-mpi/2021.2.0** and higher have a deadlock problem on the notchpeak AMD nodes, therefore the 2021.1.1 should be used instead.

Python built natively on CentOS 7 has dependencies incompatible with Rocky Linux 8, therefore more recent **python/3.10.3** has been built. Also note that the new OS does not provide the **python** command as a part of the system, use **python2** or **python3**, depending on what Python version you need. Any Python libraries that have been installed by the user using the older Python versions will need to be re-installed for use with the new Python versions. To allow for the use of existing script that use the **python** command with the system Python 2 or Python 3 installations, CHPC has created two new modules. For use of the system Python 2 the module is **python/2.7.18** and for the system Python 3 it is **python/3.6.8**. These modules link the python command to the OS supplied **python2**, and **python3**, respectively.

A similar situation exists with the R builds. The R built natively on Centos 7 no longer runs on the new OS. We have therefore installed a new version, **R/4.1.3**. For R, in addition to this new build, the containerized **R/4.1.2-basic**, **R/4.1.2-bioconductor**, and **R/4.1.2-geospatial** can still be used. As with Python, any user installed R libraries using the old R versions will need to be reinstalled for use with the new R versions.

While we have updated many packages, both before and after the update, there is a chance that a specific program may not work. If that happens, please, contact our helpdesk, *helpdesk@chpc.utah.edu*.

Fall 2022 Presentation Schedule

Date	Presentation Title	Presenter
Mon, Aug 22, 2022	*Overview of CHPC*	Anita Orendt
Weds, Aug 31, 2022	*Module Basics*	Anita Orendt
Fri, Sept 2, 2022	*Slurm and Slurm Batch Scripts*	Anita Orendt
Weds, Sept 7, 2022	*Hands on Introduction to Linux, part 1**	Anita Orendt & Brett Milash
Fri, Sept 9, 2022	*Hands on Introduction to Linux, part 2**	Anita Orendt & Brett Milash
Mon, Sept 12, 2022	*Hands on Introduction to Linux, part 3**	Brett Milash & Wim Cardoen
Wed, Sept 14, 2022	*Hands on Introduction to Linux, part 4**	Wim Cardoen & Brett Milash
Fri, Sept 16, 2022	*Hands on Introduction to Open OnDemand**	Martin Cuma
Mon, Sept 19, 2022	*Introduction to Parallel Computing**	Martin Cuma
Weds, Sept 21, 2022	*Introduction to Programming with MPI*	Martin Cuma
Fri, Sept 23, 2022	*Introduction to Programming with OpenMP*	Martin Cuma
Mon, Sept 26, 2022	*Hybrid MPI-OpenMP Programming*	Martin Cuma
Weds, Sept 28, 2022	*Hands on Introduction to Python, part 1**	Brett Milash & Wim Cardoen
Fri, Sept 30, 2022	*Hands on Introduction to Python, part 2**	Brett Milash & Wim Cardoen
Mon, Oct 3, 2022	*Hands on Introduction to Python, part 3**	Brett Milash & Wim Cardoen
Weds, Oct 5, 2022	*Numpy, part 1 (Hands on Introduction to Python part 4)**	Wim Cardoen & Brett Milash
Fri, Oct 7, 2022	*Numpy, part 1 (Hands on Introduction to Python part 5)**	Wim Cardoen & Brett Milash
Oct 10-14, 2022	Fall Break	Fall Break
Mon, Oct 17, 2022	*National and Regional Compute Resources*	Anita Orendt
Weds, Oct 19, 2022	*Open Science Grid (OSG)*	Wim Cardoen
Fri, Oct 21, 2022	*Workflows Using Snakemake**	Brett Milash
Mon, Oct 24, 2022	*Nextflow**	Brett Milash
Weds, Oct 26, 2022	*GPU Programming*	Wim Cardoen
Fri, Oct 28, 2022	*Introduction to I/O at CHPC*	Martin Cuma
Mon, Oct 31, 2022	*Introduction to Profiling*	Martin Cuma
Weds, Nov 2, 2022	*Introduction to Debugging*	Martin Cuma
Fri, Nov 4, 2022	*Introduction to Containers**	Martin Cuma
Mon, Nov 7, 2022	*Overview of the Protected Environment*	Anita Orendt

Presentations are 1-2 pm unless labeled with *, then they are hands on, 1-3 pm. For online schedule, see *https://www.chpc.utah.edu/presentations/fall2022chpcpresentationschedule.php*.

Python Tips #2

Wim R.M. Cardoen, CHPC Scientific Consultant

In the following paragraphs we discuss some recent innovations in Python. All code snippets were tested using Python 3.11.0b3. The examples can be downloaded from *https://github.com/wcardoen/python-reflections*.

Match-operator: Most programming languages have selection control mechanisms beyond the `if, elif, else` constructs. Its C counterpart is the `switch` construct.

```
# Traditional if/elif/else statements
def find_capital(country):
    if country == 'France':
        return 'Paris'
    elif country == 'Netherlands':
        return 'Amsterdam'
    elif country == 'Belgium':
        return 'Brussels'
    else:
        return 'SORRY!'
for country in ['Belgium', 'Poland']:
    print(f" Country:{country:>15s} -> Capital:{find_capital(country):>10s}
    ")
```

The aforementioned code block results in the following output:

```
Country:        Belgium -> Capital:  Brussels
Country:        Poland -> Capital:    Sorry!
```

In Python 3.10, the **match** construct was introduced.

```
def find_capital2(country):
# match/case construct (Python >= 3.10)
    match country:
        case 'France':
            return 'Paris'
        case 'Netherlands':
            return 'Amsterdam'
        case 'Belgium':
            return 'Brussels'
        case _:
            return 'Sorry!'
for country in ['Belgium', 'Denmark']:
    print(f" Country:{country:>15s} -> Capital:{find_capital2(country):>10s
}")
```

The aforementioned block results in:

```
 Country:        Belgium -> Capital:  Brussels
 Country:        Denmark -> Capital:    Sorry!
```

You can combine several patterns using the | (i.e., ∪ operator).

```
def find_continent(country):
    match country:
        case 'Belgium'|'France'|'Germany'|'Netherlands':
            return 'Europe'
        case 'China'|'India'|'Japan':
            return 'Asia'
        case _:
            return 'Sorry!'
for country in ['France', 'China', 'USA']:
    print(f" Country:{country:>15s} -> Continent:{find_continent(country) :>10s}")
```

The above code block produces the following output:

```
Country:      France -> Continent: Europe
Country:      China -> Continent: Asia
```

```
    Country:      USA -> Continent: Sorry!
```

Patterns can also be verified by unpacking:

```
def find_location(point):
    match point:
        case (0,0,0):
            return "Origin"
        case (x,0,0):
            return "Pt. on x-axis"
        case (0,y,0):
            return "Pt. on y-axis"
        case (0,0,z):
            return "Pt. on z-axis"
        case (x,y,0):
            return "Pt. in the xy-plane"
        case (x,0,z):
            return "Pt. in the xz-plane"
        case (0,y,z):
            return "Pt. in the yz-plane"
        case (x,y,z):
            return "Reg. pt."
        case _:
            return "NOT a 3D-point"
for item in [(3,4,5), [2,0,0], (0,0,0), (0,3,2), (3,4,5,6)]:
    print(f" Pt.:{str(item):>15s} Type:{find_location(item)}")
```

This results in:

```
    Pt.:   (3, 4, 5)     Type:Reg. pt.
    Pt.:   [2, 0, 0]     Type:Pt. on x-axis
    Pt.:   (0, 0, 0)     Type:Origin
    Pt.:   (0, 3, 2)     Type:Pt. in the yz-plane
    Pt.:   (3, 4, 5, 6)  Type:NOT a 3D-point
```

The match pattern construct is an extensive topic. Three PEPS (*PEP-622, PEP-634, PEP-636*) were written to address it.

Merging of dictionaries: The merging and update of Python dictionaries has been improved in Python 3.9 by introducing the | and |= operators. The details are discussed in *PEP-0584*

```
capitals1 = {'france':'paris', 'germany':'berlin'}
capitals2 = {'france':'paris', 'belgium':'brussels'}
# Merging: Creation of a new dict object
capitals3 = capitals1 | capitals2
print(f" capitals3:\n{capitals3}")
# Update in-place operation
capitals1 |= capitals2
print(f" capitals1:\n{capitals1}")
```

This results in:

```
    capitals3:
  {'france': 'paris', 'germany': 'berlin', 'belgium': 'brussels'}
    capitals1:
  {'france': 'paris', 'germany': 'berlin', 'belgium': 'brussels'}
```

Removing the prefixes/suffixes of strings: Python 3.9 introduced some handy methods to remove suffixes and prefixes (*PEP-0616*).

```
city="Witwatersrand"
STR1, STR2 ="Wit", "rand"
print(f"String:'{city}'")
print(f" remove the prefix '{STR1}' -> '{city.removeprefix(STR1)}'")
print(f" remove the sufffix '{STR2}' -> '{city.removesuffix(STR2)}'")
```

Results in:

```
String:'Witwatersrand'
    remove the prefix 'Wit'   ->  'watersrand'
    remove the suffix 'rand' ->  'Witwaters
```

Math module: The math module was extended with some interesting methods, among them:
– math.isqrt : returns the integer part of the square root
– math.gcd : returns the Greatest Common Divisor
– math.lcm : returns the Least Common Multiple
– math.prod : calculates the products of the elements in a iterable
– math.comb : calculates the number of combinations
– math.perm : calculates the number of permutations
– math.dist : calculates the euclidean distance between two points

```
import math
print(f" math.isqrt(26) :{math.isqrt(26)}")
print(f" math.gcd(24,12,36):{math.gcd(24,12,36)}")
print(f" math.lcm(2,4,6,8) :{math.lcm(2,4,6,8)}")
print(f" math.prod([2,3,4,5,6)], start=10) :{math.prod ( [2,3,4,5,6] , start=10}")
print(f" math.comb(5,2):{math.comb(5,2)}")
print(f" math.perm(5,2):{math.perm(5,2)}")
p = range (1 ,10)
q = range (2 ,11)
print(f" math.dist(p,q):{math.dist(p,q)}")
```

Which results in:

```
math.isqrt(26):5
math.gcd(24,12,36):12
math.lcm(2,4,6,8) :24
math.prod([2,3,4,5,6],start=10):7200
math.comb(5,2):10
math.perm(5,2):20
math.dist(p,q):3.0
```

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 acknowledgement 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 by sending to *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

Center for
HIGH PERFORMANCE COMPUTING
THE UNIVERSITY OF UTAH