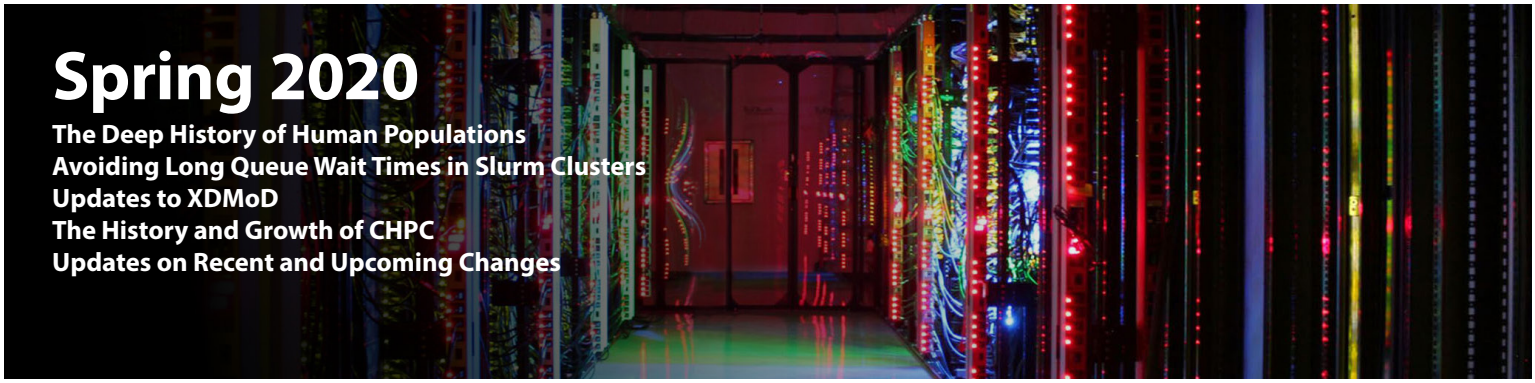
Center for
HIGH PERFORMANCE COMPUTING
THE UNIVERSITY OF UTAH

Newsletter



Spring 2020

The Deep History of Human Populations

Research Highlight · Alan R. Rogers, Departments of Anthropology and Biology

Our lab has developed a new statistical method, called "Legofit," which uses genetic data to estimate the history of population size, subdivision, and gene flow [1]. Our recent publications have used it to study human evolution over the past 2 million years [2, 3, 4].

Legofit studies the frequencies of "nucleotide site patterns," which are illustrated in Fig. 1. The solid black lines and arrows represent a network of populations. The dashed and colored lines show one of many possible gene genealogies that might occur at different nucleotide sites within the genome. Upper-case letters refer to populations. *X* represents an African population (the Yorubans), *Y* a European population, *A* Altai Neanderthals, and *D* Denisovans. *S* is an unsampled "superarchaic" population that is distantly related to other humans. Lowercase letters at the bottom of Fig. 1 label nucleotide site patterns. A nucleotide site exhibits its pattern *xya* if random nucleotides sampled from *X*, *Y*, and *A* carry the derived allele, but those sampled from other populations are ancestral. Site pattern probabilities can be calculated from models of population history, and their frequencies can be estimated from data. Legofit estimates parameters by fitting models to these relative frequencies.

Nucleotide site patterns contain only a portion of the information available in genome sequence data. This portion, however, is of particular relevance to the study of deep population history. Site pattern frequencies are unaffected by recent population history because they ignore the within-population component of variation [1]. This reduces the number of parameters we must estimate and allows us to focus on the distant past.

Figure 2 shows a set of site pattern frequencies. The largest effects in these data are the least interesting: singleton site patterns (*x*, *y*, *a*, and *d*) are common because each of the populations has a long history of independent evolution, *xy* is common because Africans (*X*) and Europeans (*Y*) are both populations of modern humans and therefore share ancestors; *ad* is common for the same reason.

The interesting effects are more subtle. Notice that *ya* is more common than *xa*, *xd*, or *yd*. This is because Europeans have some Neanderthal ancestors [8]. Notice also that *d* is more common than the other singleton site patterns and that *xya* is more common than the other tripletons. This is the signature of superarchaic admixture into Denisovans, as explained in Fig. 1: mutations on the red branch inflate the frequency of *d* and those on the blue branch inflate *xya*.

In fitting models to site pattern frequencies, we use a numerical algorithm—differential evolution [DE, 9]—to maximize a composite likelihood function. DE maintains a swarm of points, each representing a set of parameter val-

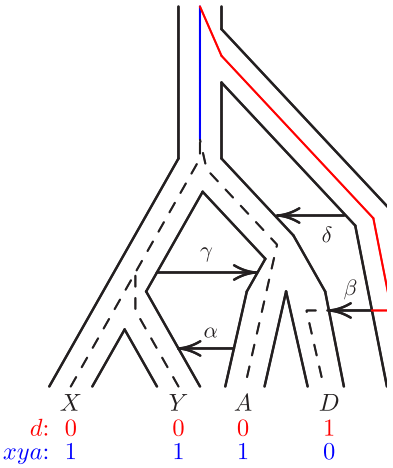


Figure 1: **A population network including four episodes of gene flow, with an embedded gene genealogy.** *Uppercase letters (*X, Y, A, D, *and* S*) represent populations (Africa, Europe, Altai Neanderthal, Denisovan, and superarchaic). Greek letters label episodes of admixture.* d *and* xya *illustrate two nucleotide site patterns, in which 0 and 1 represent ancestral and derived alleles. A mutation on the red branch would generate site pattern* d. *One on the blue branch would generate* xya.

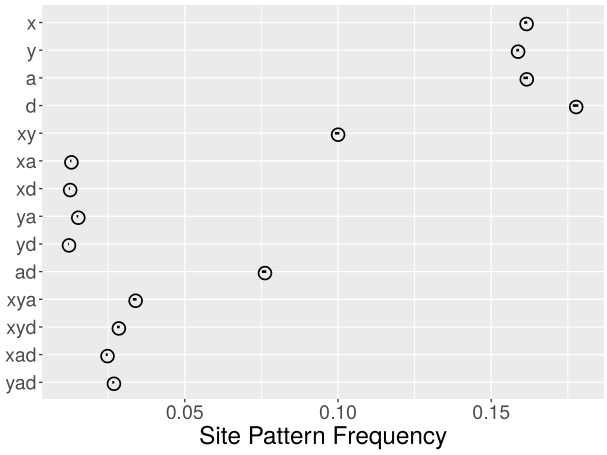


Figure 2: **Observed site pattern frequencies.** *Horizontal axis shows the relative frequency of each site pattern in random samples consisting of a single haploid genome from each of* X, Y , A, *and* D, *representing Africa, Europe, Altai Neanderthal, Denisovan. Horizontal lines (which look like dots) are 95% confidence intervals estimated by a moving blocks bootstrap [10]. Data: Simons Genome Diversity Project [12] and Max Planck Institute for Evolutionary Anthropology [18].*

ues. The number of points is 10 times the number of free parameters. In recent work, our models involve nearly 20 parameters, so there are nearly 200 points in the swarm. In each DE iteration, the composite likelihood of each point is estimated by computer simulation. These simulations run in separate threads of execution on a single CHPC node.

To estimate uncertainty, we use a moving blocks bootstrap [5], which resamples blocks of nucleotides. Each bootstrap replicate runs on a separate CHPC node, administered by a Slurm array. By parallelizing both across nodes and across threads within nodes, we can potentially accelerate these calculations by about 2,000-fold, depending on the availability of nodes and the number of cores per node.

To choose among models, we use the bootstrap estimate of predictive error [bepe, 10, 11]. Bepe is analogous to cross-validation, but uses bootstrap replicates instead of partitions of the data. We also use bootstrap model averaging [booma, 12], which assigns weights to each model based on the fraction of replicates (including the real data and 50 bootstrap replicates) in which that model "wins"—that is, has the lowest value of bepe. Booma deals with problems of statistical identifiability by broadening confidence intervals to include uncertainty about the model itself.

We first used Legofit in 2017 to argue that Neanderthals and Denisovans separated early, that their neandersovan ancestors endured a bottleneck of population size, and that the post-separation Neanderthal population was large [2]. That analysis omitted "singleton" site patterns—those in which the derived (or mutant) allele is present only in the sample from one population. Mafessoni and Prüfer [13] pointed out that introducing singletons led to different results. In response Rogers et al. [3] agreed, but also observed that the with-singleton analysis implied that the Denisovan fossil was only 4,000 years old—a result that is plainly wrong. Furthermore, a residual analysis showed that neither of the models under discussion in 2017 fit the data very well [3]. Something was apparently missing from both models—but what? Our latest paper [4] provides an answer to that question.

There were suggestions in the literature about what might be missing. We had already included gene flow from Neanderthals into modern Europeans [α, 8]. In addition, there was evidence for gene flow from early moderns into Neanderthals [γ, 14], and into Denisovans from a mysterious "superarchaic" population, which had separated from other hominins early in the Pleistocene [β, 7, 14, 15, 16, 17]. Adding these episodes of admixture improved things, but the fit was still not satisfactory.

The archaeology of the early middle Pleistocene provided an additional clue. At this time, the "neandersovan" ancestors of Neanderthals and Denisovans separated from the ancestors of modern humans. Modern humans seem to have evolved in Africa, so it seemed plausible that neandersovans separated from an African population and emigrated to Eurasia. Had they done so, they would have encountered the previous "superarchaic" inhabitants of Eurasia, who had been there since about 1.85 million years ago [18]. This suggested a fourth episode of admixture, labeled δ in Fig. 1, from superarchaics into neandersovans.

We studied eight models, all of which included episode α, and including all combinations of the other three episodes of admixture. We labeled models by concatenating greek letters to indicate which episodes of admixture were included. In evaluating these models, we used an expanded data set that includes the high-coverage Vindija Neanderthal genome [7]. In the site pattern labels in Fig. 3, the letter

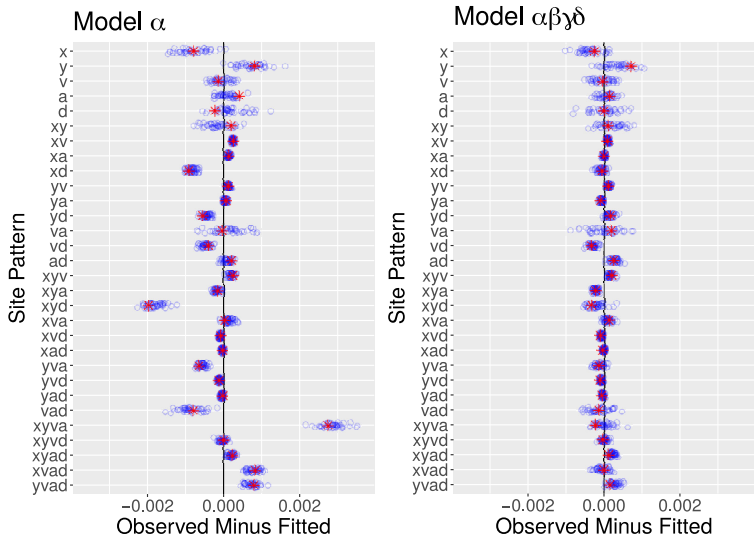


Figure 3: **Residuals from models** *α* **and** *αβγδ***.** *Key: red asterisks, real data; blue circles, 50 bootstrap replicates.*

Model	bepe	weight
α	1.16×10^{-6}	0
$\alpha\delta$	0.87×10^{-6}	0
$\alpha\gamma$	0.62×10^{-6}	0
$\alpha\gamma\delta$	0.44×10^{-6}	0
$\alpha\beta$	0.18×10^{-6}	0
$\alpha\beta\gamma$	0.17×10^{-6}	0
$\alpha\beta\delta$	0.15×10^{-6}	0.16
$\alpha\beta\gamma\delta$	0.13×10^{-6}	0.84

Table 1: Bootstrap estimate of predictive error (bepe) values and bootstrap model average (booma) weights.

"v" refers to this genome. Fig. 3 shows the residual errors from the simplest and the most complex models. Note that for model α several of the residuals are large compared with their uncertainties, as indicated by the scatter of blue circles. This shows that model α fits relatively poorly. Model $\alpha\beta\gamma\delta$, on the other hand, provides a much better fit.

Table 1 shows the bepe values and booma weights of the various models. Only two models have positive weights. To understand what this means, recall that bootstrap replicates approximate repeated sampling from the process that generated the data. The models with zero weight lose in all replicates, implying that their disadvantage is large compared with variation in repeated sampling. We therefore restrict attention to the two models with nonzero booma weights.

Fig. 4 shows model-averaged parameter estimates. Parameter m_δ measures the fraction of neandersovan DNA derived from admixture with superarchaics. It has a wide confidence interval, but even the lower bound implies substantial admixture. Parameter T_{XYNDS} is the time at which superarchaic populations separated from other hominins. Our estimate—over two million years ago—may be inflated, because it assumes that the age of male puberty has been constant over the past two million years. If the average value of this parameter were halfway between the values of modern humans and chimpanzees, our estimate of T_{XYNDS} would drop to about 1.9 million years ago—roughly coincident with the earliest dates of *Homo erectus* and of the earliest expansion of hominins out of Africa [18].

The effective size of the superarchaic population (parameter N_S) is surprisingly large: even the lower bound is 20,000. This does not necessarily mean that there were large numbers of superarchaic humans, because effective size can be inflated by geographic population structure [19]. Our large estimate may mean that neandersovans and Denisovans received gene flow from two different superarchaic populations.

According to our estimates, the Neanderthal and Denisovan populations separated early in the Middle Pleistocene (T_{ND} = 737 thousand years), and their neandersovan ancestors had a very small population. This is consistent with our previous estimates [3]. Our new results, however, contradict our previous findings about Neanderthal population size. In 2017, we argued that the Neanderthal population was larger than others had estimated. However, we now estimate that the Neanderthal population was initially large (parameter N_{N0}) but then declined in size (parameter N_{N1}) [4]. The difference does not result from our new and more elaborate model. It was including the Vindija Neanderthal genome that made the difference. Without this genome, we still get a large estimate ($N_{N1} \approx 11{,}000$), even using model $\alpha\beta\gamma\delta$. This implies that the Neanderthals who contributed DNA to modern Europeans were more similar to the Vindija Neanderthal than to the Altai Neanderthal, as others have also shown [7].

These results document the earliest known episode of interbreeding between hominin populations. Furthermore, these interbreeding populations had been separate far longer than any pair of hominin populations previously known to interbreed. They also confirm our previous finding of a bottleneck in population size among the neandersovan ancestors of Neanderthals and Denisovans.

References

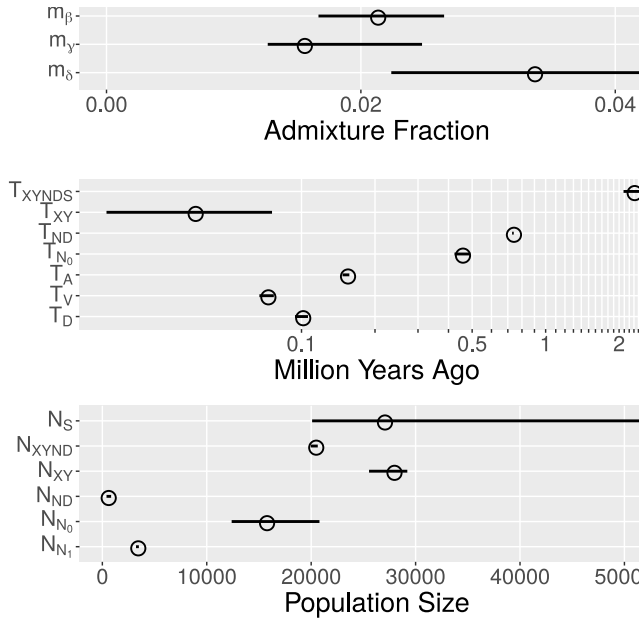


Figure 4: **Model-averaged parameter estimates with 95% confidence intervals estimated by moving-blocks bootstrap [10].** *Key:* m_α, *fraction of* Y *introgressed from* N; m_β, *fraction of* D *introgressed from Neanderthals;* m_γ, *fraction of Neanderthal DNA introgressed from* XY; m_δ, *fraction of neandersovan DNA introgressed from* S; T_{XYNDS}, *superarchaic separation time;* T_{XY}, *separation time of* X *and* Y; T_{ND}, *separation time of Neanderthals and Denisovans;* T_{N0}, *end of early epoch of Neanderthal history;* T_A, *age of Altai Neanderthal fossil;* T_V, *age of Vindija Neanderthal fossil;* T_D, *age of Denisovan fossil;* N_S, *size of superarchaic population;* N_{XYND}, *size of populations* XYND *and* XYNDS; N_{XY}, *size of population* XY; N_{ND}, *size of population* ND; N_{N0}, *size of early Neanderthal population;* N_{N1}, *size of late Neanderthal population. Parameters that exist in only one model are not averaged.*

[1] Rogers, A. R. (2019). Legofit: Estimating population history from genetic data. BMC Bioinformatics *20*, 526.

[2] Rogers, A. R., Bohlender, R. J., and Huff, C. D. (2017). Early history of Neanderthals and Denisovans. Proceedings of the National Academy of Sciences, USA *114*, 9859–9863.

[3] Rogers, A. R., Bohlender, R. J., and Huff, C. D. (2017). Reply to Mafessoni and Prüfer: Inferences with and without singleton site patterns. Proceedings of the National Academy of Sciences, USA *114*, E10258–E10260.

[4] Rogers, A. R., Harris, N. S., and Achenbach, A. A. (2020). Neanderthal-Denisovan ancestors interbred with a distantly-related hominin. Science Advances *6*.

[5] Liu, R. Y. and Singh, K. (1992). Moving blocks jacknife and bootstrap capture weak dependence. In LePage, R. and Billard, L., eds., Exploring the "Limits" of the Bootstrap In LePage, R. and Billard, L., eds., Exploring the "Limits" of the Bootstrap. (New York: Wiley).

[6] Mallick, S. et al. (2016). The Simons Genome Diversity Project: 300 genomes from 142 diverse populations. Nature *538*, 201–206.

[7] Prüfer, K. et al. (2017). A high-coverage Neandertal genome from Vindija Cave in Croatia. Science *358*, 655–658.

[8] Green, R. E. et al. (2010). A draft sequence of the Neandertal genome. Science *328*, 710–722.

[9] Price, K., Storn, R. M., and Lampinen, J. A. (2006). Differential Evolution: A Practical Approach to Global Optimization. (Berlin: Springer Science and Business Media).

[10] Efron, B. (1983). Estimating the error rate of a prediction rule: Improvement on crossvalidation. Journal of the American Statistical Association *78*, 316–331.

[11] Efron, B. and Tibshirani, R. J. (1993). An Introduction to the Bootstrap. (New York: Chapman and Hall).

[12] Buckland, S. T., Burnham, K. P., and Augustin, N. H. (1997). Model selection: an integral part of inference. Biometrics *53*, 603–618.

[13] Mafessoni, F. and Prüfer, K. (2017). Better support for a small effective size of Neandertals and a long shared history of Neandertals and Denisovans. Proceedings of the National Academy of Sciences, USA *114*, E10256–E10257.

[14] Kuhlwilm, M. et al. (2016). Ancient gene flow from early modern humans into Eastern Neanderthals. Nature *530*, 429–433.

[15] Waddell, P. J., Ramos, J., and Tan, X. (2011). Homo denisova, correspondence spectral analysis, finite sites reticulate hierarchical coalescent models and the Ron Jeremy hypothesis. ArXiv *1112.6424*.

[16] Waddell, P. J. (2013). Happy New Year *Homo erectus*? More Evidence for Interbreeding with Archaics Predating the Modern Human/Neanderthal Split. ArXiv *1312.7749*.

[17] Prüfer, K. ``et al. (2014). The complete genome sequence of a Neanderthal from the Altai Mountains. Nature *505*, 43–49.

[18] Ferring, R., Oms, O., Agustí, J., Berna, F., Nioradze, M., Shelia, T., Tappen, M., Vekua, A., Zhvania, D., and Lordkipanidze, D. (2011). Earliest human occupations at Dmanisi (Georgian Caucasus) dated to 1.85–1.78 Ma. Proceedings of the National Academy of Sciences, USA *108*, 10432–10436.

[19] Nei, M. and Takahata, N. (1993). Effective population size, genetic diversity, and coalescence time in subdivided populations. Journal of Molecular Evolution *37*, 240–244.

Avoiding Long Queue Wait Times in Slurm Clusters

Brett Milash, CHPC Scientific Consultant

"Why isn't my job running?", you ask. Often, when you submit a job to a Slurm cluster, it sits waiting in the queue in the pending (PD) state. Your job is sitting in line, waiting for one or more compute nodes to become available. How can you avoid this? That depends upon the compute resources available to you, and in some cases how many jobs you have submitted. This article will describe some tools to help you identify less busy partitions, and strategies to target those resources.

First, it is important to know what resources are available to you. Use the `myallocation` command to list all the account and partition combinations on the various clusters. Here is the output when I run `myallocation`:

```
$ myallocation
    You have a general allocation on kingspeak. Account: chpc, Partition: kingspeak
    You have a general allocation on kingspeak. Account: chpc, Partition: kingspeak-shared
    You can use preemptable mode on kingspeak. Account: owner-guest, Partition: kingspeak-guest
    You have a general allocation on notchpeak. Account: chpc, Partition: notchpeak
    You have a general allocation on notchpeak. Account: chpc, Partition: notchpeak-shared
    You have a general allocation on notchpeak. Account: notchpeak-shared-short, Partition: no…
    You have an owner allocation on notchpeak. Account: jones-np, Partition: jones-np
    You have an owner allocation on notchpeak. Account: jones-np, Partition: jones-shared-np
    You can use preemptable mode on notchpeak. Account: owner-guest, Partition: notchpeak-guest
    You have a general allocation on lonepeak. Account: chpc, Partition: lonepeak
    You have a general allocation on lonepeak. Account: chpc, Partition: lonepeak-shared
    You can use preemptable mode on lonepeak. Account: owner-guest, Partition: lonepeak-guest
    You can use preemptable mode on ash. Account: smithp-guest, Partition: ash-guest
```

You can see I have access to several clusters (lonepeak, kingspeak, notchpeak, and ash), and my access is either "general" (non-preemptable) on the regular or shared partitions on those clusters, or "preemptable" on the guest partitions. If I want to submit a job to one of the general partitions, I can check the number of pending jobs on each cluster:

```
$ squeue -M kingspeak -p kingspeak,kingspeak-shared -t PD | wc -l
481
$ squeue -M lonepeak -p lonepeak,lonepeak-shared -t PD | wc -l
4
$ squeue -M notchpeak -p notchpeak,notchpeak-shared -t PD | wc -l
2
```

We can see there are 481 pending jobs on kingspeak, but only a few jobs pending on the other clusters. Note that I can query the queue on any cluster using the `-M clustername` option—there is no need to log in to an interactive node on a different cluster to do this. Also, notice that I check **both the regular and shared partitions for a cluster** (e.g., notchpeak and notchpeak-shared). Since both of those partitions include the same nodes, it is important to consider both of them to accurately gauge how busy that cluster is.

To target one of those less-busy clusters I can either edit my Slurm script giving it the desired partition, or more simply I can specify the cluster name and partition on the command line:

```
$ sbatch -M lonepeak -p lonepeak testjob.slurm
Submitted batch job 1342570 on cluster lonepeak
```

Done this way, I can be more agile in how I target compute resources, rather than using the same partition all the time.

An alternative to `squeue -p` is to look at the Cluster Utilization graphs on the CHPC home page (*www.chpc.utah.edu*), or click through those graphs to get a more detailed listing. These graphs show, at a glance, how busy the clusters are. An example is shown on the next page. The orange bars represent CPU usage at a given time relative to the total potential given with the black line. You can see that kingspeak is at capacity, while lonepeak and notchpeak are less busy. Note that the image was captured during Spring Break and is not representative of typical usage.

Another technique for avoiding long wait times in the general partitions pertains to smaller jobs that need less than an entire node or are expected to end quickly. If you need fewer than all the cores in a node, target a shared partition such as notchpeak-shared or kingspeak-shared, specifying the number of cores needed using `-n` or `--ntasks`. Your job may start sooner by sharing a node with another job. For more information on node sharing see *https://www.chpc.utah.edu/documentation/software/node-sharing.php*. If your job will finish quickly, provide an estimate of your job's run time to Slurm. For example, `#SBATCH --time=01:00:00` requests resources for one hour.

The Slurm scheduler can frequently backfill gaps in the cluster schedule with small jobs like these. Make sure, however, to overestimate the run time, since jobs will be terminated when the requested wall time has elapsed.

Yet another strategy is to run jobs in guest mode. This type of access is available to you even if you have access to your own owner nodes or have an allocation on the general nodes. Your jobs in guest mode are preemptable. However, you can target nodes owned by groups with low utilization by following the instructions on this page: *https://www.chpc.utah.edu/usage/constraints/*.

Sometimes there is no alternative to waiting in the queue. In some cases, Slurm can provide the estimated start time of your job while it waits in the pending state. Given the job ID, you can get this information with `scontrol`:

```
$ scontrol -M kingspeak show job 7879081
JobId=7879081 JobName=testjob.slurm
   UserId=u0424091(500779) GroupId=chpc(1722) MCS_label=N/A
   Priority=101862 Nice=0 Account=chpc QOS=kingspeak
   JobState=PENDING Reason=Priority Dependency=(null)
   Requeue=0 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=00:05:00 TimeMin=N/A
   SubmitTime=2020-03-08T22:24:49 EligibleTime=2020-03-08T22:24:49
   AccrueTime=2020-03-09T08:31:14
   StartTime=2020-03-11T18:48:31 EndTime=2020-03-14T18:48:31 Deadline=N/A
```

Here we can see a job StartTime for a job that is still in the PENDING state—this is an estimated start time, based on the number of jobs in the queue, the number of available nodes, the estimated run-time of queued jobs, and so on.

The lengths of the queues on the CHPC clusters are very dynamic, with ever-changing numbers of jobs and wait times. By using these strategies to find and target less-busy resources, you may find you can cut your jobs' wait times significantly.

Updates to XDMoD

Anita Orendt, Assistant Director for Research Consulting

CHPC has been using the open source tool, Open XDMoD—XSEDE Metrics on Demand—to track usage of the HPC clusters for several years. This tool was originally developed to provide a wide range of usage metrics on the XSEDE resources and was subsequently made available to track usage on any HPC environment. CHPC has two XDMoD instances:

- *xdmod.chpc.utah.edu*, which tracks usage of the compute nodes of the general environment clusters
- *pe-xdmod.chpc.utah.edu*, which tracks the usage of the compute nodes of the redwood cluster in the protected environment

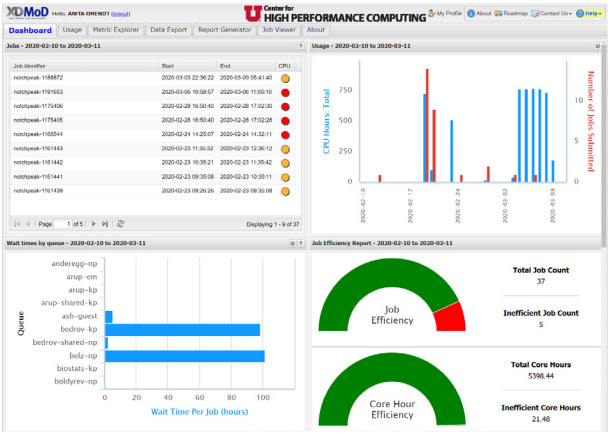
The page you land on when you go to either of these sites provides an overview of the usage of the clusters via a few different metrics, including Total CPU Hours (these are the core hours) and a breakdown of the usage by Slurm account. While these are useful metrics, they do not provide the level of detail that an individual user would be interested in. From the "Usage" tab, one can do queries at the resource, queue, user or Slurm account level; however, you would need to repeat these queries each time you want an update. We show an example of how to do this at a group level on our XDMoD page, *https://www.chpc.utah.edu/documentation/software/xdmod.php*.

We have recently updated our XDMoD installation, adding several features which we believe increase the value of these XDMoD instances to our HPC user base; two of these features are highlighted in this article.

Customized Dashboard

We have enabled login to the two CHPC XDMoD instances, which allows you to access a more personalized dashboard that you can customize. The first step is to sign in using your university login credentials. By default, your role is that of a user, which gives you the current standard user dashboard, which includes a list of your recent jobs (last month), a chart of the usage of these jobs (both number of jobs and core hours used), the wait time of each queue, and a job efficiency meter, as shown in the image on the following page.

By default, the time period is the last 30 days. If you move the cursor over the different sections, you will see additional information. Note that there is a different metric used in the CPU efficiency in the job listing—a scale of red, orange, yellow, green, or N/A—whereas in the "Job Efficiency" report, jobs are marked "inefficient"/red if they have less than 10% User CPU usage ***and*** they are using less than 50% of available memory. Also note the way hyper-



threading is treated in XDMoD: an efficiency of 50% is all cores assigned to the job being used.

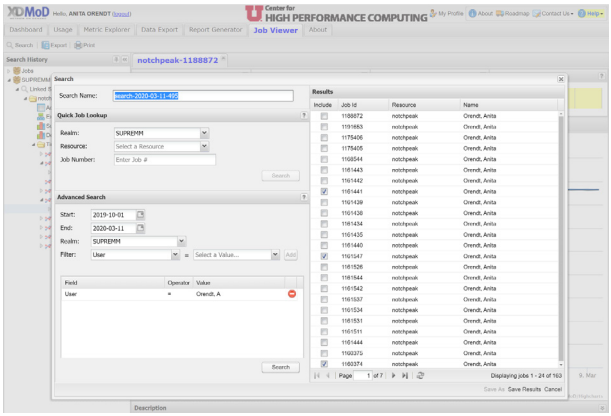
If you go to your profile, you will see your current role. As mentioned above, the default role assignment is user. There is, however, a PI role that will show not only your own usage, but the usage of all users of your Slurm account. In order to be given access to this, you can send a request to *helpdesk@chpc.utah.edu*. Note that CHPC will require anyone given this role to be a CHPC PI or have the PI's permission.

If an existing chart in the Dashboard view has the "Edit in Metric Explorer" icon in the title bar you can use this icon to change or customize the chart. Alternately, you can add additional charts to your personal dashboard by selecting the "Metric Explorer" tab. On the CHPC XDMoD page, *https://www.chpc.utah.edu/documentation/software/xdmod.php*, we give examples of both. The XDMoD User Manual, *https://xdmod.ccr.buffalo.edu/user_manual/*, provides additional, detailed information about the options.

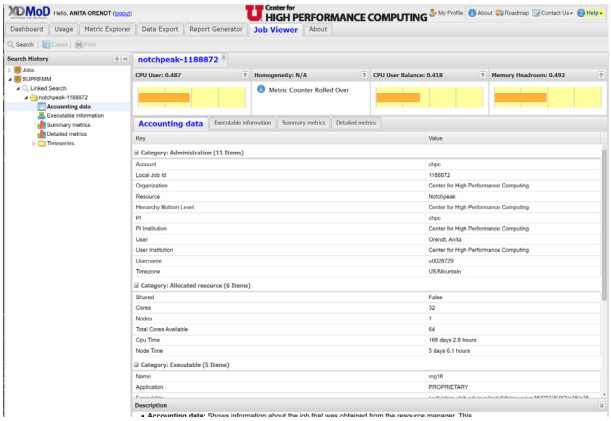
Addition of the SUPReMM Module

The SUPReMM (Systems Usage and Performance of Resources Monitoring and Modeling) module is also referred to as the Job Performance Module. The information about a job is pulled from Slurm and the Performance Co-Pilot (PCP) monitoring running on all compute nodes. We have the Slurm data for jobs back to when we started using Slurm and the PCP data since June 2019.

There are two main ways to access this information about a given job. The first is to select the job from the jobs listing on the personalized dashboard mentioned above; this will pull the job information and open the "Job Viewer" tab. The second way is to select the "Job Viewer" tab, and use the search tab to choose the job or jobs you want to analyze, either via the quick lookup if you have the cluster and job number, or via the advanced search which will give you a list of jobs. Note that in the advanced search you can choose any date range; when you choose the filter, it will populate a pull-down of options available in the "Select a Value" field. Once this is populated, click the "Add" button, and then the "Search" button near the bottom. If there are any jobs that match the selected criteria, they will appear in the "Results" window. From the jobs that appear, you can select one or more by checking them in the "Include" column, then "Save Results" at the bottom. This is shown in the image below.



When you do either of these, you will see the selected job(s) in the "Job Viewer" tab, as shown in the image below.



In the "Job Viewer" window, there are four tabs:

- Accounting data: information about the job, such as its run time and the executable used
- Executable information: includes the node(s) and the cores assigned to the job
- Summary metrics: information about CPU, memory, file and network I/O, and energy usage
- Detailed metrics: increased granularity of data in Summary metrics tab

There is also the option to get plots of timeseries data on a number of the metrics over the course of the job by opening the Timeseries folder found on the left hand side of this window.

If you would like assistance in interpreting this data for any of your jobs, refer to the XDMoD user manual or reach out to CHPC via *helpdesk@chpc.utah.edu*.

The History and Growth of CHPC

Tom Cheatham, Director of CHPC, and Anita Orendt, Assistant Director for Research Consulting

The University of Utah has a long history in computing excellence with the formation of its computer science department in 1965 and its early role in the internet as the 4th node on the Arpanet in 1969.

The Utah Supercomputing Institute (USI) was formed in 1989 with a donation of an IBM 3090 worth about $22M at a time when the total higher education budget in Utah was about $28M. The Scientific Computing and Imaging Institute (SCI) formed in 1994. USI transitioned into the Center for High Performance Computing (CHPC) in 1995 based on the Detar report that added IT research and development, distributed computing, security, and advanced networking, and expanded the mission to enhance computational science research at the university.

CHPC's *first and relatively early deployment of a protected environment (PE)* for compute and restricted data occurred in 2009. CHPC and the University of Utah also deployed one of the first Science DMZs in 2011. In 2014, CHPC shifted focus to being a service provider rather than a research institute, breaking down previous barriers to collaborations with centers and institutes who felt that CHPC was competing instead of providing. Grant efforts in CHPC are now focused on "things that will broadly help campus," such as CC* and equipment awards. In 2017, CHPC was awarded an NIH S10 to replace and upgrade the PE.

Year after year, growth of CHPC resource and usage has been astounding. Diversity of usage along with the number of disciplines is rising; we definitely are no longer simply an HPC center. We continue to have rapid growth in the HPC usage by several different measurements. The increase in core hours used in 2019 was 10.6% relative to 2018. Note that this usage does not include non-HPC usage—Windows server, specialized compute instruments (pets), or the usage of virtual machines (VMs)—for computational purposes.

YEAR	CORE HOURS USED (MILLIONS)	NUMBER OF GROUPS RUNNING AT LEAST ONE BATCH JOB	NUMBER OF USERS RUNNING AT LEAST ONE BATCH JOB
2015	83.8	111*	344*
2016	124.2	174	556
2017	134.7	207	636
2018	160.6**	235	727
2019	177.7	314	891

*The number of users and groups is for the last six months of 2015 (after XDMoD was in production)
**Starting in 2018, this includes usage of the protected environment cluster (redwood)

We also continue to see growth in CHPC account requests, allocation requests, and groups purchasing their own nodes.

YEAR	GROUPS OWNING NODES*	GROUPS WITH ALLOCATION IN FALL QUARTER	NEW PIS PROVISIONED	USER ACCOUNTS PROVISIONED	NEW NODES ADDED
2015	28	65	62	459	
2016	37	75	85	546	
2017	48	76	95	562	117 (plus 88 repurposed**)
2018	55	75	93	767	72
2019	70	82	110	683	120 (plus 100 repurposed**)

*In some cases, these are departments or coalitions of several groups
**Repurposed nodes were servers being retired from use elsewhere

Finally, we have seen rapid growth in both the number of training sessions we offer as part of our CHPC Presentation Series and the number of attendees. Over the last several years, CHPC has moved to do more hands-on training, added training during the summer semester, and enabled remote participation. During 2019, excluding the XSEDE-run workshops and the Build a Cluster workshop, there were over 50 presentations, encompassing 80 hours and attended by over 900 users.

Updates on Recent and Upcoming Changes

In CHPC's Fall 2019 Newsletter, we outlined a number of upcoming changes. Several were completed in the last few months. The major changes completed were the addition of 32 new AMD nodes to notchpeak, the retirement of ember, and the move to make notchpeak the only cluster that requires allocation to run on the general nodes without jobs being subject to preemption. In addition, lonepeak was expanded with donated hardware, including four additional donated nodes with 1 TB of memory each.

There are still two upcoming changes. The first of the two upcoming changes involves providing priority access to the lonepeak general nodes for "cancer-related research." CHPC will implement this change effective April 1, 2020.

The second upcoming change is related to the previously announced changes to local scratch. Now that the initial step of software encryption of the local hard drives on compute and interactive nodes has been completed, several changes will be implemented in the coming months:

- The permissions of the top-level /scratch/local will be set such that users cannot create directories in the space
- The job prolog (before the job starts) will make a job-level directory /scratch/local/$USER/$SLURM_JOB_ID, to which only the user will have access
- The job epilog will the remove this job-level directory after the job has ended, regardless of the exit state

We will make this change on a per-cluster basis to allow users time to adapt their scripts to write to /scratch/local/$USER/$SLURM_JOB_ID instead of creating a job-level directory under /scratch/local.

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