Code Scaling

The CIPRES Science Gateway currently provides access to parallel versions of the 15 codes listed in Table 1. These codes have run exclusively on Comet during the current allocation year. We will continue to provide access to all of these codes in the coming allocation year and expect to provide access to additional ones as well.

Table 1. Parallel codes currently supported by the CIPRES Science Gateway

Code	Latest version	Language	Compiler	Computer	Cores charged	MPI library	Threading
BEAST	1.8.4	Java		Comet	2 to 84		Java
BEAST2	2.4.8	Java		Comet	1 to 84		Java
DPPDiv	1.0	C++	GNU	Comet	12		Pthreads
FastTree	2.1.10	C	Intel	Comet	3		OpenMP
GARLI	2.01	C++	Intel	Comet	1 to 24	OpenMPI	
IQ-TREE	1.6.2	C++	GNU	Comet	12 or 24		OpenMP
jModelTest2	2.1.10	Java		Comet	24		Java
MAFFT	7.305	C	Intel	Comet	12		Pthreads
Migrate-N	3.6.11	C	Intel	Comet	1 to 72	MVAPICH2	
MrBayes	3.2.6	C	Intel	Comet	2 to 24	MVAPICH2	
ParallelStruct	2.3.4	C + R	Intel	Comet	12		R
Partition-Finder2	2.1.1	Python	Intel	Comet	8		
PAUP	4.0a151	C	Intel	Comet	1 or 4		Pthreads
PhyloBayes*	1.7b	C++	Intel	Comet	48	MVAPICH2	
RAxML	8.2.10	C	Intel	Comet	12, 24, or 48	OpenMPI	Pthreads

*Use of PhyloBayes is by special request only, since its runs are relatively expensive.

Table 2 shows the relative usage of each code by gateway users in the first three quarters of the current allocation year. The codes with the heaviest usage are BEAST, BEAST2, Migrate-N, MrBayes, and RAxML.

The number of cores charged for each code is listed in Table 1 and is based upon benchmark studies we have done for representative data sets. Scaling plots for the five most heavily used codes follow. Each of the other codes has usage less than 5% of the total, so scaling plots for them are not presented here.

Note that PhyloBayes runs are no longer allowed under the CIPRES community allocation because the SU usage per job is dramatically higher than that of the other CIPRES codes. Thus we allow PhyloBayes runs only by individuals who have a separate XSEDE allocation against which the usage is charged or when a referee has insisted on a PhyloBayes analysis in order to accept publication of a paper.

Table 2. Breakdown of usage by code for July 1 2017 - Mar 31 2018

Code	Version	Number of jobs	Fraction of jobs	Number of SUs*	Fraction of SUs*	SUs/job
BEAST	1.8.x	27,348	0.129	3,672,239	0.214	134
BEAST2	2.3.x + 2.4.x	17,516	0.083	2,108,868	0.123	120
DPPDiv	1.0	24	< 0.001	27	< 0.001	1
FastTree	2.1.x	1,000	0.005	548	< 0.001	<1
GARLI	2.01	3,708	0.018	480,620	0.028	130
IQ-TREE	1.6.x	639	0.003	54,102	0.003	85
jModelTest2	2.1.x	11,421	0.054	258,647	0.015	23
MAFFT	7.xxx	5,580	0.026	29,545	0.002	5
Migrate-N	3.6.11	1,816	0.009	1,130,147	0.066	622
MrBayes	3.2.x	48,940	0.231	4,125,860	0.241	84
ParallelStuct	2.3.4	1,112	0.005	102,486	0.006	92
Partition-Finder2	2.1.1	4,763	0.023	139,625	0.008	29
PAUP	4.a150	4,279	0.020	53,253	0.003	12
PhyloBayes	1.5a	414	0.002	805,724	0.047	1,946
RAxML	8.2.x	82,878	0.392	4,191,704	0.244	51

*GPU consumption is normalized to CPU consumption using a multipler of 14.

BEAST runs can be parallelized in various ways, especially when using the BEAGLE library to evaluate the computationally intensive phylogenetic kernel. The relevant options follow.

1. Runs with multiple partitions can be parallelized with threads across the partitions.
2. Runs with BEAGLE and many patterns per partition can break up the partitions into subpartitions and parallelize across the latter with threads.
3. Runs with BEAGLE that have very many patterns in a partition or that have amino acid data instead of nucleotide data can use one or more GPUs to obtain even greater parallel speedup.

All of these options are used for CIPRES gateway runs on Comet with the specific choice dependent upon the data set based upon an extensive set of benchmark runs that we made. Runs use either 2, 4, 6, or 8 Haswell cores or 1, 2, or 4 P100 GPUs depending upon the speedup achievable.

Figures 1 and 2 show that the speedup of BEAST for single-partition, nucleotide data sets improves as the number of patterns increases. Our rules for running these cases via the gateway are as follows.

1. Runs with < 1,200 patterns, such as the 568-pattern case, use 2 Haswell cores.
2. Runs with ≥ 1,200 but < 4,000 patterns, such as the 2,980-pattern case, use 4 Haswell cores.
3. Runs with ≥ 4,000 but < 7,000 patterns, such as the 5,565-pattern case, use 8 Haswell cores.
4. Runs with ≥ 7,000 but < 20,000 patterns, such as the 10,908-pattern case, use 1 P100 GPU.
5. Runs with ≥ 20,000 but < 40,000 patterns, such as the 25,576-pattern case, use 2 P100 GPUs.
6. Runs with ≥ 40,000 patterns, such as the 83,144-pattern case, use 4 P100 GPUs.

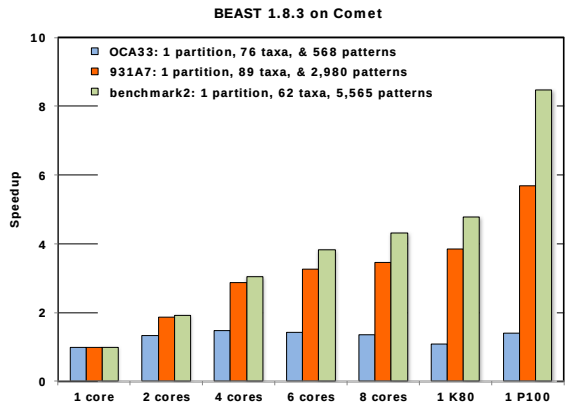

Figure 1. Speedup of BEAST on Comet for single-partition, nucleotide data sets with < 7,000 patterns; gateway runs use 2, 4, or 8 Haswell cores for the cases with 568, 2,980, or 5,565 patterns, respectively.

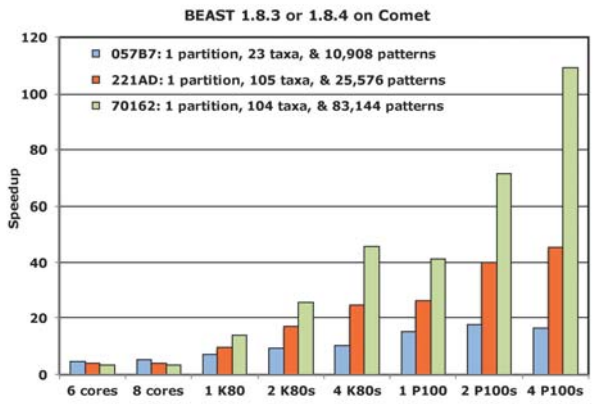

Figure 2. Speedup of BEAST on Comet for single-partition, nucleotide data sets with ≥ 7,000 patterns; gateway runs use 1, 2, or 4 P100 GPUs for cases with 10,908, 25,576, or 83,144 patterns, respectively.

Similar rules apply for nucleotide data sets with multiple partitions, except that more than one P100 GPU is never used. Amino acid data sets entail more work for each evaluation of the phylogenetic kernel, so such runs always use either 1 or 4 P100 GPUs.

The availability of GPUs, especially the newer and faster P100s, has significantly increased the usage of BEAST (and BEAST2) over the past year and allows analyses that were not practical before. For example, analysis of the largest data set in Figure 2 is 109x faster on 4 P100s than on 1 Haswell core and 31x faster than on 8 cores, where the speedup on cores reaches a plateau. This exceptional speedup allows large analyses in days instead of months.

Use of the P100 GPUs is also more cost effective than use of either the Haswell cores or the older K80 GPUs based on the allocation conversion rate of 1 P100 GPU hour = 1.5 K80 GPU hours = 21 Haswell core hours. Analysis of the largest data set in Figure 2 for a given number of steps would cost only 34% as much on 4 P100s as on 8 Haswell cores.

BEAST2 runs can be parallelized in a similar fashion as those for BEAST. Runs use either 1, 3, 4, or 6 Haswell cores or 1, 2, or 4 P100 GPUs. Figure 3 (below) shows the speedup for three single-partition, nucleotide data sets with more than 10,000 patterns. As before, there is an exceptional benefit from using the P100 GPUs.

Migrate-N is parallelized with MPI over the product of two parameters: the number of loci and the number of replicates. The parallelization uses a master-slave model with an extra core dedicated to the master process.

Figure 4 shows the scaling of Migrate-N on Comet for a microsatellite data set with two combinations of loci and replicates. The parallel efficiency drops in half as expected going from one to two cores, but generally increases thereafter. Gateway runs use 21 cores within a single node for the case of 10

loci with 2 replicates and 41 cores in two nodes for the case of 10 loci with 4 replicates. Seven cores are idle in the latter case, but the effective parallel efficiency is still high.

MrBayes is parallelized with MPI over the product of two parameters: nruns and nchains. The default values are 2 and 4, respectively, though 4 and 4 are occasionally used. Thus most runs via the gateway use either 8 or 16 cores.

Figure 5 shows the scaling of MrBayes on Comet for a typical data set with two parameter-pair combinations. Although the parallel efficiency drops below 0.5 for the number of cores used, there is still useful speedup compared to fewer cores.

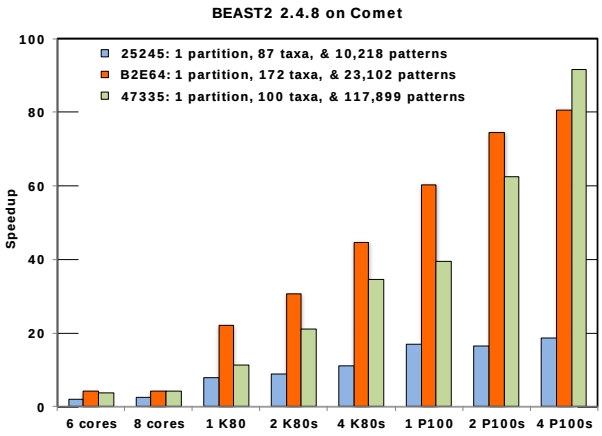

Figure 3. Speedup of BEAST2 on Comet for single-partition, nucleotide data sets with ≥ 10,000 patterns; gateway runs use 1, 2, or 4 P100 GPUs for the cases with 10,218, 23,102, or 117,899 patterns, respectively.

RAxML is most commonly run in a so-called comprehensive or all-in-one analysis, which consists of 100 or more bootstrap searches followed by ~20 regular searches. Such an analysis works well with a hybrid parallel scheme developed at SDSC, in which fine-grain parallelization across patterns is

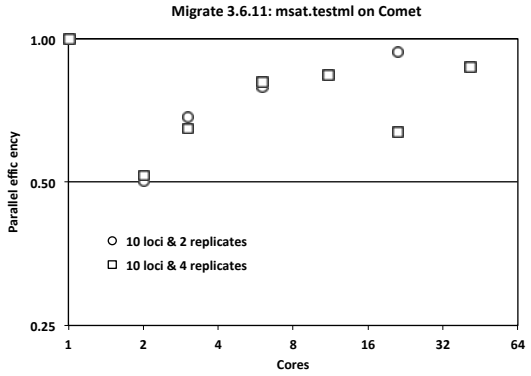

Figure 4. Parallel efficiency of Migrate-N on Comet for two parameter-pair combinations; gateway runs use 21 and 41 cores for the circle and square cases, respectively.

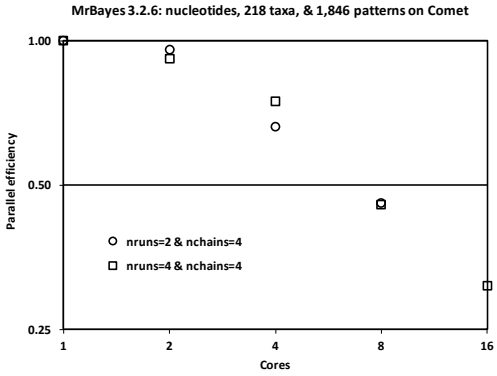

Figure 5. Parallel efficiency of MrBayes on Comet for two parameter-pair combinations; gateway runs use 8 and 16 cores for the circle and square cases, respectively.

done by Pthreads within each search and coarse-grain parallelization across the searches is done by MPI.

Figure 6 shows the scaling of RAxML on Comet for a typical data set with two ways of doing the bootstraps. The circles correspond to specifying a fixed number of bootstraps, which is nominally 100, while the squares correspond to using a statistical test to automatically select the number of bootstraps, which is about 550 for this data set. Automatic selection of the number of bootstraps is

the recommended analysis approach and gives outstanding scaling. In fact, the scaling is superlinear over the range of cores shown because of a change from the serial algorithm for more than 100 bootstraps that reduces the computational work while still ensuring a good solution.

Comprehensive analyses via the gateway that are expected to take more than 2 hours to complete use 48 cores in two nodes of Comet, whereas shorter analyses use only 24 cores in a single node. This is because single-node jobs start more quickly when the queue is busy. If no more than 20 GB of memory is needed per MPI process, then the two-node jobs use 12 MPI processes and 4 Pthreads per process, while the single-node jobs use 6 MPI processes and 4 Pthreads per process. If more memory is needed, then the number of MPI processes decreases, and the number of Pthreads per process increases.

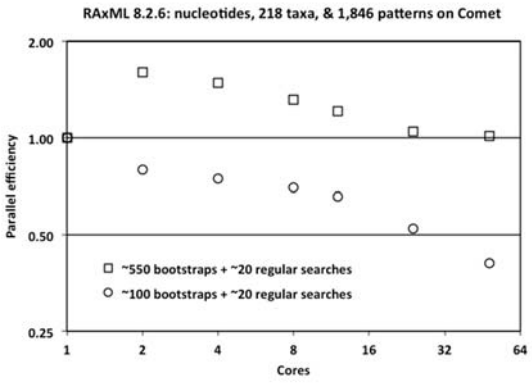

Figure 6. Parallel efficiency of RAxML on Comet for two comprehensive analysis approaches; gateway runs that take more than 2 hours use 48 cores, while shorter runs use 24 cores.

RAxML-NG, a new version of RAxML, is currently under development. When it is released later this year, we expect to make it available along with the existing version and will run more benchmarks to decide how to run the new code efficiently.

IQ-TREE usage has been modest so far, since it was only installed during the past year. Going forward, however, we expect that its usage will grow significantly.

Currently, IQ-TREE jobs use either 12 or 24 cores within a single node of Comet depending upon the data set. Analyses of data sets with no more than 12 partitions and that require no more than 60 GB of memory run on 12 cores, while other analyses run on a full, 24-core node.

Renewal Request for the CIPRES Science Gateway

A. Research Objectives

This request is for a renewal of the XSEDE allocation that sustains the CIPRES Science Gateway (henceforth CIPRES). CIPRES was initially targeted to the evolutionary biology and systematics communities, but its users now extend across virtually all fields of biology. The idea of a public portal that provides community access to phylogenetics codes run on substantial computational resources was initially implemented by the CIPRES (CyberInfrastructure for Phylogenetic RESearch) project, a multi-institutional NSF-funded collaboration that created algorithms, tools, and computational capabilities to help researchers analyze phylogenetic relationships in a manner that scales across the entire tree of life. CIPRES evolved from this implementation and is designed to simplify access to community codes run on high-performance computing (HPC) resources, so domain users can infer evolutionary relationships on data sets that would otherwise be intractable strictly because of their computational requirements.

Inferring evolutionary relationships from protein and DNA sequence data is computationally intensive. The process involves two distinct steps. First, ***sequence alignment*** algorithms are used to predict evolutionary relationships between individual nucleotide or amino acid residues in a collection of homologous DNA or protein sequences. Next, ***phylogenetic trees are inferred*** using an algorithmic model for how a given set of aligned sequences evolved from a common ancestral sequence. Both steps in this process are NP hard [9,10] and community codes for analyzing the data must use heuristics to simplify the calculations. Even with simplifying heuristics, routine phylogenetic analyses require more and more computational power every year because the amount of sequence data available for analysis is increasing exponentially.

The rapid increase in the amount of sequence data has in turn dramatically increased the need for access to HPC resources, even for routine analyses. As a result, investigators who cannot access such resources are at a significant disadvantage relative to those who can. The computations required for even very routine analyses can require weeks or even months on a laptop or desktop machine. The powerful HPC resources supported by XSEDE offer a scalable solution to this problem, but the overhead for access to XSEDE is prohibitive for most individual researchers. CIPRES provides access without requiring each user to submit a convincing resource request, set up a personal environment, learn to use the queuing systems, install and configure the tools, and then learn to run these tools efficiently from the command line. CIPRES provides a point-and-click browser interface that allows users to configure and run community codes on HPC resources with no knowledge of the detailed implementation of these codes. This saves researcher time and effort, and dramatically expands the scope of what many researchers can do in a given time frame. CIPRES has also added programmatic access via a CIPRES RESTful API (CRA), which allows developers of other gateway resources to incorporate CIPRES capabilities into their applications.

CIPRES currently provides the following benefits for its user community:

- The browser interface allows users to run on HPC resources supported by XSEDE without requiring users to obtain an allocation of computer time or deal with the complexities of Linux, MPI, and job schedulers.
- The REST interface allows scripters and developers to incorporate the same services into existing software packages and new scripted workflows.
- Individuals who are resource-limited can complete analyses of larger data sets, so analyses can be tailored to the question at hand, rather than to available resources.
- Large scientific questions can be asked and answered by individuals irrespective of their local cyberinfrastructure.

- New community codes can be deployed for a large audience in a familiar environment. This promotes communication between users and tool creators.
- The interaction between community code developers and staff providing Extended Collaborative Support Services (ECSS) via XSEDE has promoted more efficient use of existing community codes and encourages the development of new parallel codes.

Importantly, access provided by CIPRES speeds up user analyses significantly at a very modest cost. In 2017, the average CIPRES user consumed only about 2,200 Comet SUs per year. CIPRES is used by two types of users for research: 1) evolutionary biologists and systematists who run large and comprehensive analyses over many weeks and 2) biologists in a wide variety of other fields who require an occasional phylogenetic analysis to place their results in the context of an evolutionary history. Both populations are represented in the CIPRES user base, as evidenced by individual use patterns and a sampling of recent publications. In addition to its use by researchers, many students are being trained in the use of phylogenetic tools via the CIPRES, primarily as part of graduate curriculum.

In January, 2015, we launched the CIPRES RESTful API (CRA), which provides a route for programmatic access to all CIPRES capabilities. The CRA was created with an award from the NSF Division of Biological Infrastructure and is based on the Representational State Transfer (REST) architecture [11]. We have now embedded CRA services into six well-established community software environments and are in the process of embedding CRA services into others. The CRA allows anyone to access CIPRES services from within their preferred work environment, whether it is a complex desktop software environment or a simple *ad hoc* script. This further reduces the barrier to access and increases the impact of XSEDE resources on the scientific progress. In addition, our collaboration with Marlon Pierce and Suresh Marru at Indiana University has created a generic set of infrastructure services that can be used by any community to build new gateways. We are presently incorporating the Airavata sharing service into CIPRES, which will make it easier for users to share data and job results with others. The design of these services will take advantage of the tools and software we have created here, and we will incorporate them into the open governance Apache Airavata project.

B. Background

CIPRES was originally released in May 2007 as the CIPRES Portal. It provided simple point-and-click access to community codes run on a modest cluster at no cost to users. The CIPRES Portal provided access to most command-line options for sequence alignment codes (ClustalW, Contralign, Muscle, MAFFT, Probalign, Probcons), tree inference codes (MrBayes, RAxML, GARLI, PAUP*), and a code that combines both sequence alignment and tree inference (POY). Data and results of analyses were stored indefinitely. The architecture used a straightforward XML specification to create interfaces, so new codes could be added as they appeared within the community, and existing interfaces could be adapted as community codes evolved. The CIPRES Portal was converted to the CIPRES Science Gateway in 2009, the name change signaling that the scalable resources available through (then) TeraGrid and (now) XSEDE could be used. As a Gateway, CIPRES was able to offer users a significant wall time savings by running parallel versions of popular community codes.

Over the past five years, CIPRES has evolved significantly. Highlights of this evolution are listed here:

- Created access to TeraGrid/XSEDE resources from the CIPRES web application using GSI-SSH and adapted the infrastructure to use resources at NCSA, TACC, and SDSC.
- Ran benchmarking tests to develop rules for running parallel versions of 15 phylogenetics codes efficiently on TeraGrid/XSEDE resources.

- Developed and deployed interfaces that permit users to launch parallel jobs on TeraGrid/XSEDE where each job is automatically configured according to the aforementioned rules.
- Created documentation and software to help browser and REST users launch their jobs, monitor job progress, and interpret error messages.
- Ran jobs from the command line for selected users whose jobs were too computationally demanding to run via existing interfaces for one reason or another. This engaged the community and allowed us to understand where the CIPRES interfaces break down for high-end users.
- Collaborated with the developer of RAxML to create a hybrid parallel version of that code [12] that was dramatically faster than earlier versions and made the new version generally available.
- Migrated serial codes to the TSCC cluster at SDSC, which is used on a fee-for-service basis.
- Created pluggable job submission and monitoring interfaces and pluggable file system interfaces, allowing us to work with a variety of job schedulers and file transfer protocols, including SGE, TORQUE, LSF, Slurm, GRAM, GridFTP and SFTP.
- Developed reliable and robust mechanisms for results tracking, intermediate result viewing, and final results retrieval.
- Implemented tools to track usage by individual user accounts over each allocation year and tools to disable submissions from accounts that are consuming excessive amounts of resources.
- Implemented tools to notify users of their usage through the user interface and through email.
- Implemented methods to help users request a personal XSEDE allocation and to charge CIPRES usage to that personal account.
- Made it possible to authenticate through the iPlant/CyVerse authentication mechanism, thereby giving iPlant/CyVerse users access to the RAxML-Light code.
- Refactored the underlying web application code so large files are not loaded into memory.
- Improved our database management, including a switch from myiasm to innodb tables and implementation of a new record locking mechanism, improved threadpool error handling, new database connection pool implementation, and a hash system that identifies duplicate datasets and stores just a single copy.
- Increased the stored accounting information for jobs, refined logging major events as a job progresses.
- Developed tools to automatically manage and distribute output files in excess of 2 GB via sFTP.
- Developed tools that limit user data storage, and that automatically delete user data and results in accounts that have been idle for more than one year (after warning the users).
- Improved job submission performance by making job staging and submission asynchronous. Submission retries are now possible if a remote resource is down when the job is initially submitted.
- Developed automated testing of each tool with a variety of parameters through the middle tier of the CIPRES web application.
- Refactored the software package to make it simpler to distribute and re-use, and supported creation of two new gateways based on that software.
- Created programmatic access via the CIPRES RESTful API, with all attendant tools to monitor and control usage through RESTful services.
- Created job configuration helper tools for the REST service.
- Implemented a tool for resumable large file uploads that can be offered as a service.
- Added RESTful services to five web applications and two desktop application.
- Implemented tools for editing input files in the application.
- Deployed BEAST and BEAST2 jobs for large alignments on highly efficient GPUs.
- Created data searching/sharing tools in collaboration with the SciGaP project at Indiana University (in progress).

Public submissions to TeraGrid/XSEDE resources began on 12/1/2009, and the number of users, jobs, and usage has increased roughly linearly since then (see Figures 1 and 2 in the Progress Report). Based on these trends, we project that nearly 19 million Comet SUs and more than 300,000 Comet_GPU SUs will be consumed in the present allocation year. As noted in the Progress Report, the time requested has been put to good use, supporting more than 900 peer-reviewed publications in 2017 and 348 publications in the first quarter of 2018 (users report another 32 *in press*). In eight years, the CIPRES has supported more than 30,000 users, who have reported their analyses in 4,300+ peer-reviewed publications using TeraGrid/XSEDE resources.

Our initial CIPRES TRAC/XRAC awards included support from the ASTA/ECSS project groups, with assistance in software design and implementation (Nancy Wilkins-Diehr, Doru Marcusiu, and Terri Schwartz) and in parallel algorithm development, implementation, and tuning (Wayne Pfeiffer). We are grateful for that support, since without it, CIPRES would not exist.

C. Codes and Methods

Current Codes: At the time of this writing, the CIPRES provides access to parallel versions of 15 widely used phylogenetics codes on XSEDE: MrBayes/BEAGLE (which is MPI-only), RAxML (hybrid MPI/Pthreads), GARLI (MPI), BEAST/BEAGLE (Java threads), BEAST2/BEAGLE (Java threads), FastTree2 (OpenMP), MAFFT (Pthreads), Migrate-N (MPI), PhyloBayes-MPI (MPI), DPPDIV (Pthreads), jModelTest2 (Java threads), PartitionFinder2 (Python threads), PAUP (Pthreads), ParallelStructure (R), and IQ-TREE (OpenMP). Since the retirement of Gordon as an XSEDE resource, all parallel codes run on Comet. The codes used by CIPRES and their parallel implementations are described below:

MrBayes [13] is a Bayesian tree-inference code that is parallelized with MPI. MrBayes (currently 3.2.6) incorporates commands from the related program BEST (Bayesian Estimation of Species Trees) [14]. BEST expands the capabilities of MrBayes by implementing a Bayesian hierarchical model to jointly estimate gene trees and the species tree from multi-locus sequences. MrBayes 3.2.x allows checkpointing, which minimizes resource loss when there is a system outage or when a run does not complete in the time allowed. CIPRES provides an interface for restarting MrBayes jobs. In addition, the BEAGLE library [15] is used to evaluate the phylogenetic likelihood kernel, resulting in better performance than in earlier versions of the code. MrBayes jobs run on up to 24 cores (depending on the input) within a single node of Comet. Runs of less than 24 cores are made in the shared queue. In the first nine months of this allocation year, MrBayes accounted for 24% of the Comet-equivalent SUs consumed by CIPRES (4.1 million).

RAxML [16] is a maximum likelihood phylogenetics code. CIPRES provides the hybrid MPI/Pthreads version (currently 8.2.10), which parallelizes over the number of searches with MPI, so multiple nodes can be used. Benchmarking studies were performed to determine the preferred numbers of MPI processes and Pthreads for Comet. The user interface configures these numbers automatically based on user-entered problem specifications. For the recommended case of a comprehensive analysis with automatic bootstopping (which results in 100-500 bootstraps), two 24-core nodes of Comet are used with 12 MPI processes and 4 threads each. In the first nine months of this allocation year, RAxML accounted for 24% of the SUs consumed by CIPRES (4.2 million).

GARLI [17] is a maximum likelihood phylogenetics code that includes an MPI implementation. GARLI (currently 2.01) runs efficiently on Comet using MPI for the typical case of multiple regular searches or multiple bootstrap searches. One MPI process is assigned to whichever search is limiting, up to a maximum of 24 processes. All runs execute on a single 24-core node of Comet, either in the shared partition (if there are fewer than 24 processes) or in the compute partition otherwise. In the first nine months of this allocation year, GARLI accounted for 2.8% of the SUs consumed by CIPRES.

BEAST [18] is a Bayesian tree inference code. Although BEAST (currently 1.8.4) is written in Java, the computationally intensive kernel can use the BEAGLE library [15], which works well with threading on CPUs and GPUs. All BEAST jobs run on Comet and, depending upon the data set, use either 2, 4, 6, or 8 Haswell cores or 1, 2, or 4 P100 GPUs. In the first nine months of this allocation year, BEAST accounted for 21% of the Comet-equivalent SUs consumed by CIPRES (3.7 million, including 2.3 million on GPUs).

BEAST2 [19] is a follow-on to BEAST that has morphed into its own separate code base with separate developers. Parallelization of the current version (2.4.8) has better performance, than prior versions, and can take advantage of Comet's GPUs. The codes and methods section reports details of the performance on of various data sets using combinations of compute cores and GPUs. In the first nine months of the allocation year, BEAST2 accounted for 12% of the SUs consumed by CIPRES (2.1 million, including 1.2 million on GPUs).

FastTree2 [20] employs a likelihood method, but uses a neighbor-joining algorithm to speed the initial stages of the analysis. The neighbor-joining algorithm is particularly fast, making this code useful for very large data sets and very large trees. CIPRES uses an OpenMP version of FastTree2 (currently 2.1.10) that scales well to only three cores. FastTree2 is currently run using the shared queue on Comet. This code accounted for a negligible fraction of all CIPRES resource consumption.

MAFFT [21] is a multiple sequence alignment code that is parallelized with Pthreads. All runs of MAFFT (currently 7.305, though 7.394 will be available soon) use 12 cores of Comet. In the first half of the current allocation year, MAFFT represented 0.2% of all CIPRES resource consumption.

DPPDIV [22,23] is a code for estimating species divergence times and lineage-specific substitution rates on a fixed topology. All runs of DPPDIV (currently 1.0) are now using 12 cores in a shared queue on Comet and run the Pthreads implementation, using the Intel compiler. In the first nine months of this allocation year, DPPDIV accounted for a negligible fraction of the SUs consumed by CIPRES.

jModelTest2 [24] carries out statistical selection of best-fit models of nucleotide substitution. It accomplishes this using five different model selection strategies: hierarchical and dynamical likelihood ratio tests, Akaike and Bayesian information criteria, and a decision theory method. It also provides estimates of model selection uncertainty, parameter importance, and model-averaged parameters, including model-averaged tree topologies. We benchmarked Java threads version 2.1.10 on Comet and obtained good speedups up to 24 cores, which is the number of cores we currently use in production. jModelTest2 accounted for 1.5 % of CIPRES usage in the first nine months of the current allocation year.

Migrate-N [25] uses either Bayesian inference or maximum likelihood to estimate effective population sizes and past migration rates between populations assuming a migration matrix model with asymmetric migration rates and different subpopulation sizes. Migrate-N (currently 3.6.11) is parallelized with MPI, and individual jobs use between 1 and 72 cores on Comet, depending on the number of loci in the user's dataset. Scaling in this range gave good parallel efficiencies, as noted in the Codes and Methods sections. Migrate-N jobs used 6.6% of all SUs (1.1 million) in the first nine months of the 2016/2017 allocation year.

PAUP [26] is a well-known program for inferring phylogenetic trees using parsimony. It was recently re-released in a form that can take advantage of multiple cores in a processor when the optimality criterion used is likelihood. For these jobs (about 10% of all PAUP jobs), PAUP uses Pthreads and is run on 4 cores in the shared queue of Comet. Other jobs (where parsimony is used as the optimality criterion) are run on a single core. PAUP used 0.3% of the CIPRES SUs in the current allocation year.

ParallelStructure [27] is a parallel implementation of the code Structure in R. It achieves parallelization by dividing up the input data set, and running individual tasks concurrently, Scaling of this code was optimal using 12 cores of Comet in the shared queue. ParallelStructure was released in December, 2016, and these jobs used 0.6% of all SUs in the current allocation year.

PartitionFinder2 [28] PartitionFinder2 is a code used to select best-fit partitioning schemes and models of molecular evolution for tree inference experiments. PartitionFinder2 achieves parallelization by dividing up the data set and running independent calculations across all available cores. Our benchmarking shows 8 cores to be optimal on Comet. CIPRES first offered this code in June of 2017; it has accounted for 0.8% of resource use in the current allocation year.

IQ-Tree: [29] IQ-Tree offers a fast stochastic algorithm to infer phylogenetic trees by maximum likelihood. It is competitive with and sometimes superior to RAxML. IQ-Tree is run on 12 or 24 cores of Comet, depending on the job's memory requirements and the number of partitions in the data set. CIPRES first offered this code in December of 2017, and it has received only modest use, only 0.3% of total SUs.

PhyloBayes-MPI [30] is a Bayesian code for inference of Phylogenetic trees that was implemented in response to user demand. In the first year of release, PhyloBayes-MPI accounted for 12% of the SUs consumed by CIPRES, but was used by fewer than 1% of the users. As a result, we permit its use only by individuals with a personal XSEDE allocation or those needing to reproduce prior work or answer a reviewer's request. Despite the limited access, it accounted for 4.7% of all use in the first nine months of the current allocation.

D. Planned Software Development

In December of 2017 we received a four-year award for $1.86 million from the NIH-NIGMS Software hardening for Big Data program to make the CIPRES software more capable, robust, and user friendly. This new funding offers us an exciting opportunity to increase the positive impact of CIPRES and will allow us to carry out the following software development activities.

Add tools for sharing/collaboration. In a recent survey, more than half of current CIPRES users expressed a need for the ability to share their ongoing work in a collaborative way with users of their choosing, as well as the ability to make the results of their analyses public. To support the first need, we will integrate a service created by the SciGaP project (Science Gateways as a Platform) that supports a data sharing. SciGaP maintains this service for any gateway that wishes to consume it, allowing us to add this feature without adding maintenance cost to the project. SciGaP has funding through 2018 and PI Miller of CIPRES is also a Co-PI of SciGaP. A pop-up window in the browser interface will allow users to specify who may access specific data items and provides a search capability to discover data containing specific text strings. Users will be allowed to access results in read-only fashion but they will be able to import them into their own area to modify as required. This will protect the integrity of the original shared data.

Expose CIPRES services in other applications. The CIPRES REST API (CRA) allows us to expose all capabilities supported by CIPRES in the context of existing applications. We will wrap CRA services and register them in the Galaxy Tool Shed thereby making them available to all Galaxy users. We will also create modules that expose CIPRES tools in the Geneious and BioKepler platforms (adding Geneious was among the most requested features in a recent survey, and we have already made significant headway with the integration). The process for adding CRA services to these applications is well developed and documented. Galaxy, Geneious, and BioKepler users will acquire powerful tools for phylogenomic analysis in the context of an application they are already familiar with (current support for phylogenetic tools is minimal and serial on Galaxy's main site, Geneious users are presently confined to the compute power of

their local resource, and there is no such capability in BioKepler currently). Galaxy, Geneious, and BioKepler users will gain the capability to run jobs concurrently and to run larger jobs than is currently possible in these applications. Meanwhile, CIPRES users will gain the ability to use phylogenomic tools in the context of interfaces that support workflow, offer powerful DNA manipulation tools, and provide access to online DNA databases.

Improve restart capabilities. Although CIPRES allows users to run jobs for up to 168 hours, jobs frequently do not complete even in their configured time frame. The ability to restart a job that terminated abnormally after running for a long time or that reached the end of its maximum run time without completing ensures that the compute time is not wasted. Several CIPRES codes support checkpointing, but they all assume the user can restart their job in the directory where it terminated. The design of CIPRES places each submitted job run in a new, uniquely named directory. Currently, restarts are cumbersome and fragile, because users must transfer files manually to this new directory. In some cases it is impossible to support native restart features (e.g., BEAST2) because of file naming issues. This problem can be solved by refactoring the existing software so users can restart jobs in the existing directory, as intended by the individual codes.

Improve error messaging. Users frequently find it difficult to diagnose problems with their jobs. For example, when a job reaches its maximum run time in the current application, the user must know to look in a particular file (scheduler_stderr.txt) to learn why the job stopped without producing the expected results. Often users request help instead, which can cause a several-hour delay in productivity, and require CIPRES project staff time to resolve. We will create tools that parse output files for common errors and post a message to the user interface so it is obvious to the user that the job did not complete normally. We will also add icons to the interface to indicate the most common types of failure. These changes alone will save hundreds of emails between staff and users in a given year.

Make job submissions/job checking more flexible. The design of user interfaces for job submission is based on the PISE XML standard. PISE makes creating new interfaces and exposing command-line options highly scalable, but in its current form there are two problems: each interface can only submit to a specific cluster, and the PISE XML interpreter (created as a prototype, but still in use) makes extensive use of Perl for complex expressions, which is highly problematic at times. We plan to refactor the XML interfaces, so it is possible for CIPRES to submit a job to any given resource. This will allow us to choose where the job goes based on the status of the resource. It will also simplify cloud-bursting to elastic resources (see below) and simplify movement of codes to new clusters as existing clusters are retired. We will also refactor the XML interpreters so Python can be used in place of Perl.

Make file transfers resumable. The growth in input and output data file sizes has created a need for a resumable file transfer service that can be used within the CIPRES browser and REST interfaces. We created a Java implementation of the http-based resumable file transfer protocol created by TUS.io (https://tus.io/implementations.html). This file transfer is a standalone service, and can be used outside of the CIPRES browser interface. In addition, the Globus on Line file transfer capability has been added to the Workbench Framework by Mona Wong. This service is also resumable and robust for large file transfers. We will integrate one or both of these capabilities into CIPRES, so users will have more robust trasnfers of large files, and can continue their work while their file transfers progress.

E. Using the Comet_GPU Resource Efficiently

In July 2017, Comet was divided into two separate resources: Comet_GPU which contains all nodes that were equipped with K80 or P100 GPU; and Comet which contains all the CPU-only nodes. As a result, allocations and usage on the Comet_GPU resource are now calculated independently of the Comet CPU-only resource. We have found that the Comet_GPU resource is extremely valuable to a subset of CIPRES users, because the P100 GPUs are remarkably efficient for analyzing large data sets using the BEAST and BEAST2 codes. This provides CIPRES with significant opportunities, as well as risks that need to be mitigated.

As an example of the opportunity Comet_GPU provides, the Code Scaling document shows a BEAST analysis of a nucleotide data set with 83k patterns that is 31 times faster on 4 P100s than on 8 Haswell cores (eight is the maximum number of Haswell cores that can be used productively for this analysis). Accordingly, a job run of 1 hour running on 4 P100s is charged the equivalent of 84 core hours on Comet. The same amount of work on 8 Haswell cores would require 31 hours and incur a charge of 31 * 8 = 248 core hours. The charge for doing this work on 4 P100s would therefore be 84/248 = 34% the charge for the same job run on 8 Haswell cores, and the user would receive results in one hour instead of 31 hours. This is a rare win:win where the user gets their results 31x faster, and the resource use goes down by 34% as well. For context, jobs in this class can now be completed within the 7-day time limit allowed for CIPRES that would require 7 months using 8 Haswell cores. As a result, analyses that would not have been possible in the past can now be run routinely on CIPRES.

This benefit comes with a risk, however. Use of 4 P100 cores is charged the equivalent of 84 SUs per hour and CIPRES allows run times of up to 168 hours, so a single job run on 4 P100 GPUs can result in a charge of 14,100 SUs, or more than 25% of a US user's annual CPU allocation. This is the first time it has been possible in CIPRES for a user to consume his or her entire annual allocation with the submission of just four jobs. Although we have tools in place for limiting overuse of CPU hours, we did not implement those tools for GPU hours last year because our funding for staff was very limited at that time.

With new award from NIH, we are proceeding with the following plan to ensure that the GPU resource is efficiently utilized: (1) We began monitoring GPU usage manually last month and have disabled submissions for 10 users who exceeded their annual limit through GPU usage alone; (2) we have begun adapting our existing tools for automatically monitoring and limiting CPU usage so they track and limit GPU usage accurately; (3) we resumed work on developing tools to obtain the information that we need to schedule jobs efficiently on either CPUs or GPUs by parsing input files rather than allowing users to enter that information manually.

Item 2 will require tracking GPU usage in the same way as we currently track CPU usage. Initially we will advise users of their total CPU and GPU usage to date. We will also give them an estimate for the maximum GPU consumption of a job prior to submission, in the same way as when they make CPU submissions. Next we will automatically prevent users from submitting jobs when they have reached their annual GPU limit. We have already created many of these tools for the CIPRES REST application, and these can be adapted to the CIPRES web application fairly easily. The tools we have created include:

1. Submission monitoring tools that reject submissions of more than x jobs by a single application, where x is a number that will be set arbitrarily and be adjusted empirically based on a balance between creating excessive vulnerability and restricting the user's ability to take advantage of the tools we have created.
2. Submission monitoring that prevents any given user from sending more than y jobs to the queue simultaneously, where y is set arbitrarily and adjusted empirically, as described above.

3. Submission monitoring tools that place "reserves" on each user's account. If a user has z jobs running, the application will calculate the maximum SU charge possible for those running jobs. The total will be debited from their account. When projected use of running jobs exceeds the maximum allocation under CIPRES policy, submissions will be halted from that user's account, until running jobs complete, and their consumption is known.

Item 3 involves adding a feature to the web application that parses uploaded files, abstracts key metadata from those files, and attaches these data to job submissions. Currently we rely on user-entered data, which is an error-prone method, and we have seen it produce a number of misconfigured files. The planned feature will also check input file configurations and can be extended to include all of the tools that CIPRES offers, thus improving submissions and efficiency across the entire platform.

F. Management of Resources

Resource consumption by CIPRES users is growing consistently over time as detailed in the companion Progress Report. Over eight years of operation, the increase in CPU resource consumption has been driven mostly by an increase in the number of users. The number of jobs per user has remained essentially unchanged relative to the previous year, and the number of SUs per job has increased only modestly, apart from the GPU-enabled codes.

The high level of resource consumption by CIPRES makes it imperative to monitor usage routinely. Our main concerns are to ensure 1) that CIPRES is democratizing access to XSEDE resources over a broad population and 2) that a few users are not accessing large amounts of resources without peer review. Table 1 shows resource usage for the past three allocation cycles, with users binned according to how many Comet-equivalent SUs (both on CPUs and GPUs) they consumed from the CIPRES community account. Since the 2017-2018 allocation cycle is still in progress, the relevant data for the total number of users in each bin were projected assuming that current usage trends continue.

Table 1. Binned usage of the CIPRES Science Gateway in each allocation year

Usage	Fraction of Total SUs			Total Users			Fraction of Users		
	2015-16	2016-17	2017-18*	2015-16	2016-17	2017-18‡	2015-16	2016-17	2017-18*
> 100 K	0.034	0.044	0.163	2	3	17	0.0002	0.0004	0.0016
50 – 100 K	0.098	0.089	0.071	29	25	23	0.003	0.003	0.002
30 - 50 K	0.144	0.174	0.145	72	88	83	0.009	0.010	0.007
10 - 30 K	0.399	0.371	0.293	439	401	419	0.055	0.045	0.038
1 - 10 K	0.288	0.280	0.284	1,579	1,479	1,668	0.198	0.167	0.150
0.1 - 1K	0.031	0.037	0.039	1,679	1,812	2,137	0.199	0.204	0.193
0 – 100	0.004	0.004	0.005	4,246	5,061	6,756	0.535	0.570	0.608
Total				7,942	8,869	11,102			

* The figures for 2017-2018 were calculated using data from July 1, 2017 – March 31, 2018.
‡ The total numbers of users for 2017-2018 are projections based on trends through March 31, 2018.

The data show a projected increase in the total number of active users (those that submitted one or more jobs to XSEDE) of nearly 20% annually over the years listed. This increase is predominantly from the 90-95% of the users in the bins between 0 and 10K SUs per year who accounted for more than 30% of the usage. The number of users requiring more than 10K SUs is relatively constant, and this group typically

accounts for 65-70% of the usage. Overall, the data in Table 1 show that CIPRES continues to distribute XSEDE resources across a very broad and growing user community.

The number of users in the top two bins is small because we restrict users in the US to 50,000 SUs and those at non-US institutions to 30,000 SUs. Users at US institutions who require more than 50,000 SUs can acquire a personal allocation. This year, no users have requested personal allocations from the XRAC. During the current allocation year, submissions were disabled from 34 user accounts when they reached the allowed limit, and it is likely that more users will reach this limit before the allocation year ends.

The recent increase in the number of users in the upper bin occurred because the CIPRES software was not able to track GPU usage when the GPU nodes of Comet were separated into a distinct resource. As a result, these users were not throttled properly. Going forward we have imposed limits for the GPU resource similar to those for the CPU resource, allowing a maximum of 5,000 GPU SUs per year from the CIPRES community allocation for US users and 3,000 GPU SUs per year for foreign users.

Why are there a few users who consume more than 50,000 SUs?

Each year, a few users consume more than our imposed limit of 50,000 SUs. In the current allocation year, this number is much larger than in prior years because our application did not track GPU usage correctly, as mentioned above. This resulted in a significant number of users overshooting the 50,000 SU limit. Currently, a small number of large GPU jobs can exceed the 50,000 SU limit before we are aware of it. We will adapt to correct this automatically as described elsewhere in the report. Users of the Comet CPU-only resource often overshoot the 50,000-SU limit because of the way user submissions have historically been disabled. The CIPRES application has not automatically disabled user accounts when the threshold of 50,000 SUs limit is passed. Rather, when a user crossed the limit, the CIPRES application generates an email message to the user and the CIPRES PI. The CIPRES PI then sends a personal email message to the user to inquire whether the project is nearly complete. The account is disabled if the user does not reply, but there is a 3-day lag between notification and disabling submissions from the account. Several users in Table 1 who exceeded the limit submitted additional jobs between the time they were contacted and when the accounts were disabled. Several others requested a small amount of additional time and were allowed to complete work that required slightly more than the limit.

The policy for disabling submissions from user accounts through the RESTful interface does not permit this kind of overshoot. Because a user can deploy many jobs simultaneously, it is necessary to prevent a user from deploying) a large number of jobs on CIPRES (accidentally or purposefully) using scripting tools. The public RESTful API controls job use as follows: each time a user submits a job, the CIPRES application calculates the total number of SUs the job will consume if it runs for the maximum configured wall time. It reserves this number of SUs in the user's account. If a user deploys a job that exceeds the remaining (unreserved) time in the community allocation, the job is held and the user is notified. This modification prevents a user from overshooting his or her maximum allowed SUs from the community allocation. It also prevents accidental mishaps that might consume large amounts of XSEDE resources wastefully. This system will be implemented shortly for the GPU usage by the web application as well.

G. Resources Requested and Their Justification

Compute resources. For the coming allocation year, we request a renewal allocation of 22,000,000 Comet SUs and 320,000 Comet GPU SUs. The gateway-friendly policies of Comet make it extremely effective for CIPRES, and it remains our best option within the XSEDE portfolio. Jobs start quickly, they can run for up to a week, and GPU jobs run especially fast on the P100s.

Our projections for usage during the current allocation year are 19,000,000 Comet SUs and 320,000 Comet GPU SUs. Our request would allow for a 15% increase in Comet SUs, whereas we expect that an allocation

of Comet GPU SUs comparable to our recent usage will suffice now that we have imposed limits on GPU usage.

We appreciate that this is a large request. However, when considered in the context of the number of researchers who will benefit from the allocation, the request is actually quite modest. We project that 11,100 unique users will submit jobs to XSEDE resources via CIPRES by the end of the current allocation year. This is a 24% increase from 8,900 in the prior allocation year. For the coming allocation year, we project an increase of nearly 20% to 13,000 unique users via the browser interface and 200 more via the CRA. By these projections, the allocation request amounts to 2,200 Comet CPU-equivalent SUs per user. This is slightly more than 4% of the maximum Comet startup allocation, yet incurs no overhead for user access or for XSEDE personnel in awarding allocations and supporting users. Based on the number of known publications enabled by the CIPRES (4,300+), 21,000 Comet SUs are required for each publication, or less than 50% of the maximum startup allocation request on Comet. This is a highly effective use of computational resources.

The request also appears quite reasonable when considered in the context of the number of CIPRES users as a fraction of all XSEDE users. In the fourth quarter of 2017, 3,953 users accessed XSEDE through CIPRES, which represented 30% of all active gateway users. The CIPRES request will support 20% of active XSEDE users with less than 2% of the total SUs allocable via XSEDE.

Project Data Storage. We are requesting 5 TB of data storage space on Data Oasis. Over the past year, the size of output files produced by the programs BEAST and BEAST2 have increased steadily. We have received approximately 44,000 BEAST jobs during this allocation year, and about 4% of these jobs (1,572) had output files with more than 2 GB of data. These data had to be returned to the user via http, using the SDSC cloud storage system. The requested space will be used for storage of large files that need to be returned to users outside the normal web application. This will replace the current use of the SDSC Cloud system and will provide a mechanism for robust return of user files that are too large for us to store locally. It is important that users be able to retrieve their data as efficiently as possible from the web interface. This is also important when accessing CIPRES via the RESTful interface.

ECSS support. The ECSS support that we received i was instrumental for the success of CIPRES. At the present time, we are not requesting further ECSS support.

H. Availability of Other Computational Resources

CIPRES currently uses TSCC at SDSC to run serial jobs. A Gateway that lacked these codes would be less useful. Supporting serial codes on TSCC allows us to offer commonly used sequence alignment codes and file manipulation tools, while reserving XSEDE resources for jobs that use these resources efficiently. TSCC offers cost-effective fee-for-service access to cycles, and we have sufficient funding to continue support of this resource for our serial codes.

I. Community Support and Management Plan

Community Support Plan: The Community support plan for the web browser version of CIPRES is essentially unchanged. CIPRES allows users to report issues and request features via a commercial bug-tracking software package. Documentation for appropriate use is provided in text format; flash demonstrations of job creation and data management are provided. Browser-based users will be provided with links to XSEDE resource status and maintenance calendars, and warnings are posted when specific resources are unavailable. User support is provided by the PI on a greater than 8/5 and less than 24/7 basis, and it covers all aspects of user issues, from feature requests to file format issues. The addition of two new junior developers to the CIPRES team will allow us to be more responsive to new tool requests.

CRA services will continue to be supported by the CIPRES Team. Support activities will include partnering with the application development teams of eight community projects to expose CRA services within the existing applications, supporting users of these applications in testing, debugging, and using these new services, supporting community developers in incorporating the services into new applications, and supporting individual users in accessing and using these tools through simple or complex scripting tools. The CIPRES team will provide support, training, and documentation for CRA service use.

Management Plan: The Web application infrastructure for CIPRES is stable and flexible. It is designed to support changes in the toolkit with minimal developer effort. The cost of maintaining the Web presence is minimal at this time, with more than 90% of reported issues being attributable to user error or resource unavailability. The PI assumes responsibility for day-to-day operations, user support, and tracking resource availability. Software infrastructure for managing users and all middleware development for deploying will be addressed by Paul Hoover, a senior developer with more than 20 years of experience who is supported at 50% effort. Terri Schwartz, who created both the current CIPRES and the new REST services is now retired, but has agreed to consult on the project at 10% effort. She is collaborating with Tony Chen, another senior developer who is working at 50% effort for CIPRES projects on planning new features. Wayne Pfeiffer continues to provide expert advice on HPC. The team is supported by the current NSF and NIH awards noted below.

The PI will continue to be responsible for overseeing and tracking user consumption of resources and ensuring the CIPRES fair use policy is enforced. He will continue to assist users in requesting individual XSEDE allocations as necessary. He will also oversee the development and implementation of the tools required to track usage, support users in accessing personal allocations, and ensure that technical support is available to such users.

Financial Support: The current CIPRES was created with funding from two awards: the original CIPRES ITR award (NSF EF 03-31648) and the NEW Biology Workbench, NIH 5R01GM073931. Further development of the CIPRES was supported through a sub-contract with the iPlant Collaborative (www.iplantcollaborative.org). Development of RESTful service access to CIPRES capabilities was supported under NSF DBI-1262628 from 5/1/2013 – 4/30/2017. The creation of SciGaP, a set of Generic Software services that can be used to create Science Gateways (including the sharing service noted above), is supported by NSF ACI-1339774 from 10/1/2013 – 9/30/2018. Current development activities are supported by the recently received NIH R01 GM126463-01 award, which will support development of new capabilities from 12/15/2017 – 11/30/2021. We are still awaiting a decision from NSF ABI on a pending award that would provide additional funds to sustain operations. A second pending proposal to NSF will fund the study of the impact of CIPRES and other digital resources on economic growth in the US. We are also investigating opportunities with private foundations to expand the CIPRES mission.

Sustainability: We plan to continue developing additional proposals for NSF, foundation, and corporate sponsorships for the CIPRES software as an open source package for creating new gateways. We will continue to engage the community in development of new portal features and explore alternate funding models that will sustain the web application for the future.

Progress Report

During the past allocation year, use of CIPRES has continued to grow steadily. The number of returning and total users per month has continued to increase (a user is an individual who has submitted at least one job). This trend has been consistent over the past seven years (Figure 1). Over the current allocation cycle, an average of 50 more users submitted jobs to CIPRES than in the previous month. Moreover, in the each of the past 12 months, an average of 588 new users submitted an XSEDE job for the first time (compared with 478 over the previous 12 months).

The black data in Figure 2 show that job submissions are increasing in parallel with user numbers. In March 2018, 2,593 users submitted jobs to XSEDE resources, compared with 2,078 in March 2017.

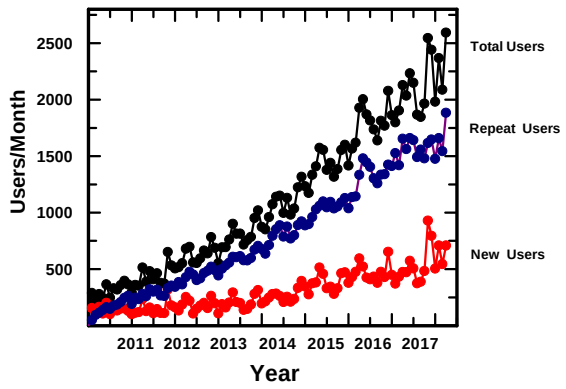

Figure 1. Number of unique users making one or more job submission per month over CIPRES lifetime. Repeat users made a submission in at least one previous month.

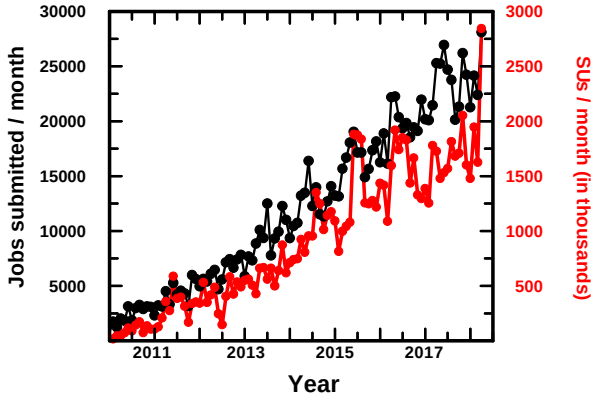

Figure 2. Job Submission (Black) and SU consumption (Red) over the lifetime of CIPRES. SUs are normalized to Comet.

The red data in Figure 2 show total SUs per month. Over its lifetime, CIPRES has run jobs on Comet, Gordon, Trestles, Lonestar2, and Abe, so SUs in Figure 2 are normalized to Comet SUs with appropriate TeraGrid/XSEDE conversion factors. Also included is Comet GPU usage (which has been allocated and accounted for separately from CPU-only usage since July 1, 2017) using the conversion factor: 1 Comet_GPU SU = 14 Comet SUs.

A total of 17.1 million Comet SUs were consumed by CIPRES users in the first nine months of the current allocation cycle, an increase of 22% over the 13.9 million SUs for the same period in the previous cycle. This was due primarily to substantial increases in usage for BEAST, BEAST2 (primarily GPU usage), and several lesser used codes, offset somewhat by a 13% decrease in average SUs per job for RAxML. The dramatic spike in usage seen March 2018 reflects a number of large BEAST jobs deployed on Comet_GPU by a few users. We will consider the opportunities and challenges that GPU nodes present for CIPRES in the main document.

Table I shows the distribution of resources to individual users in the 2017/2018 allocation year (July 1, 2017 – March 31, 2018). Over this time period 9,818 users submitted a job to XSEDE resources (an increase of 25% over the same period (7,800) in the 2016/2017 allocation year). Individual usage covers a broad spectrum: 1,491 users ran a single job that generated no SU charge while others consumed more than 50,000 SUs. During the current allocation cycle, the largest growth was in the 1-100 bin, where the number of users increased by 36%. Growth in the 0 and 100-1000 bins were approximately 16%, while the upper bins grew only by 4%. Users consuming 0 SUs ran jobs that were so short no XSEDE charge was incurred, or their jobs failed at the outset, because of a problem with the dataset. Increases in resource consumption were proportionate to growth in user number in all non-zero bins, with the exception of the top bin.

Disproportionate use was seen in this bin as a result of our present inability to control Comet_GPU usage effectively. More will be said on that in the main document. In the current allocation cycle, submissions were disabled for 24 individual user accounts that had exceeded their quota (compared with 47 at this time in the previous year). No users acquired personal accounts for large analyses this allocation year.

Table I. CIPRES resource consumption in the 2017/2018 allocation year binned according to usage.

SUs Consumed	Number of Users	Percentage	Number of SUs	Percentage
> 30k	92	1.0%	5,853,312	35.8%
10k-30k	314	3.2%	5,430,635	33.2%
1k-10k	1251	12.7%	4,383,931	26.8%
0.1-1k	1601	16.2%	601,036	3.6%
1-100	5087	51.7%	80,245	0.5%
0	1,491	15.2%	0	0

Success Metrics

Publications: Publications enabled by CIPRES are identified by two techniques: first, Google search and the Science Citation index are used to identify publications that cite the founding CIPRES publication [1], and second, users are asked to report publications that were enabled by CIPRES in the annual CIPRES survey. These publications are combined to create a master list of supported publications on CIPRES home page (please see the attached XSEDE-enabled publications document). Because we rely on users to report publications that do not cite the founding CIPRES reference, this number should be viewed as a lower limit. In 2017, at least 900 peer-reviewed publications were supported by CIPRES, and 348 have been supported so far in 2018. This brings the total number of publications supported over CIPRES lifetime to (at least) 4,300. In the same survey, users reported an additional 32 publications *in press*, and 214 works in various stages of preparation/review. These publications are reported in journals that serve many fields, including Systematics, Phylogenetics, Phylogeography, Botany, Ecology, Evolutionary Biology, Bacteriology, Virology, Molecular Biology, Structural Biology, Cell Biology, Biochemistry, Bioinformatics, and Veterinary Science.

New scientific discoveries or results enabled by the codes and resources of the gateway: The publications noted above cannot all be reviewed here, but some highlights include: Evidence that a carnivorous bladderwort plant relied on small-scale, tandem duplications to adapt its metabolism to the carnivorous lifestyle [2]; contrary to expectation, herbivores do not adapt rapidly to plant defense mechanisms, rather they feed on plants whose defenses they can defeat [3]; that the Chlorflexi, thought to be a pre-oxygen phototroph, is not ancient, and acquired its photosynthetic pathway through horizontal gene transfer [4], USA300, a pandemic clonal lineage of hypervirulent methicillin-resistant Staphylococcus aureus (CA-MRSA) originated in central Europe in the mid-19th century [5]; sterol synthesis began at least 0.71 Gyr earlier than previously thought, and that stem eukaryotes shared sterol biosynthesis genes with bacteria via horizontal gene transfer [6]; biosynthetic gene clusters that appear to be silent are actually often actively producing previously undetected metabolites [7]; contrary to prevailing thinking, songbirds are frequently infected by retroviruses, suggesting a role for retroviral transposable elements in speciation among songbirds [8].

User visits/number of unique users: CIPRES ran jobs for 11,093 unique users during 2017 (a 15% increase relative to 2016). The number of unique users running at least one job has increased from 100/month in Dec, 2009 to 2,500+/month in Mar, 2018. Nearly 70% of users return to run jobs in at least one subsequent month.

Number of institutions/countries running jobs on CIPRES: Our database shows that approximately 21% of all jobs submitted via CIPRES are submitted by users at US institutions. In user surveys, an additional 30% of users in other countries report they are working with a US collaborator. Over the lifetime of CIPRES, job submissions have come from all 50 states and the District of Columbia, including 29 of the 31 EPSCOR states/territories. These jobs were run for 184 US research universities (including Harvard, Yale, UC Berkeley, and Stanford) and 68 non-PhD granting colleges (including one all-women's college, two community colleges, four historically black colleges, and 24 Hispanic-serving institutions), and four K-12 school systems. Submissions were also received from 46 non-governmental organizations, including museums (e.g., the Smithsonian Institution, the American Museum of Natural History, and the Field Museum), botanical gardens, (e.g., Chicago, Rancho Santa Ana, and New York), and institutes (e.g., JCVI and Broad). Submissions were made from 21 US governmental entities (representing US Departments of Agriculture, Commerce, Interior, Defense, Energy, and Health and Human Services, as well as NSF and NASA) plus four commercial entities.

CIPRES continues to receive a significant number of submissions from the international community as well. Users from 101 countries on 6 continents have submitted jobs via CIPRES. These submissions came from 603 universities (including Oxford and Cambridge), 161 institutes (including Max Planck Institute and the Institute of Botany, Chinese National Academy), 80 museums, (including the British National Museum, the Royal Botanical Gardens, and the Museum National D'Histoire Naturelle in Paris), and 134 governmental agencies and institutes (including Australia's National Science Agency and the NIH of the UK, Canada, and Macedonia).

Number of classes or courses using the gateway: 85 survey respondents report they used CIPRES to deliver curriculum in 2017.

Number of jobs run overall: In the 2017 calendar year, CIPRES ran 280,161 parallel jobs on XSEDE resources, a 14% increase over the previous year. An additional 9,235 jobs were run serially on non-XSEDE resources, a 9% increase relative to the previous year. Over the lifetime of CIPRES (Dec 1, 2009 through March 31, 2018), more than 1.22 million jobs have been submitted to XSEDE/TeraGrid resources.

Impact on productivity: In a recent survey, 95% of survey respondents stated that CIPRES benefits their research in a tangible way; 82% stated that CIPRES allows them to complete work that would be difficult or impossible otherwise; and 96% stated they were able to accomplish what they wanted to using CIPRES. Many users remarked that the service helped them when they had no access to other computing resources, and 92% of survey respondents said their progress would be slowed or halted if CIPRES Science Gateway became unavailable.

New codes/features added by CIPRES group: IQ-TREE was released as a new code. Existing codes BEAST2, FastTree, MAFFT, and PAUP were updated. Embedded REST services were created to deploy in raxmlGUI and Geneious, which are desktop applications. User can now edit their data in the application. In addition, our REST services support the T-BAS Gateway, run by Ignazio Carbone at North Carolina State University.

Number of user issues and feature requests received and addressed: We resolved 329 individual user cases since July 1, 2017. This included assisting users in submitting jobs properly, dealing with bugs in codes, and responding to feature requests. At this moment there are 4 pending cases, and 92 requests for new programs, features, and capabilities.

Impact on K-12 Education: Four high school students used CIPRES in 2017/2018.

References

1. Miller, M. A., Pfeiffer, W., and Schwartz, T. (2010) Creating the CIPRES Science Gateway for Inference of Large Phylogenetic Trees in *SC10: Workshop on Gateway Computing Environments (GCE10)*, New Orleans
2. Lan, T., Renner, T., Ibarra-Laclette, E., Farr, K. M., Chang, T.-H., Cervantes-Pérez, S. A., Zheng, C., Sankoff, D., Tang, H., Purbojati, R. W., Putra, A., Drautz-Moses, D. I., Schuster, S. C., Herrera-Estrella, L., and Albert, V. A. (2017) Long-read sequencing uncovers the adaptive topography of a carnivorous plant genome. *Proceedings of the National Academy of Sciences* **114**, E4435-E4441
3. Endara, M.-J., Coley, P. D., Ghabash, G., Nicholls, J. A., Dexter, K. G., Donoso, D. A., Stone, G. N., Pennington, R. T., and Kursar, T. A. (2017) Coevolutionary arms race versus host defense chase in a tropical herbivore–plant system. *Proceedings of the National Academy of Sciences* **114**, E7499-E7505
4. Shih, P. M., Ward, L. M., and Fischer, W. W. (2017) Evolution of the 3-hydroxypropionate bicycle and recent transfer of anoxygenic photosynthesis into the Chloroflexi. *Proceedings of the National Academy of Sciences* **114**, 10749-10754
5. Strauss, L., Stegger, M., Akpaka, P. E., Alabi, A., Breurec, S., Coombs, G., Egyir, B., Larsen, A. R., Laurent, F., Monecke, S., Peters, G., Skov, R., Strommenger, B., Vandenesch, F., Schaumburg, F., and Mellmann, A. (2017) Origin, evolution, and global transmission of community-acquired Staphylococcus aureus ST8. *Proc. Natl. Acad. Sci. U. S. A.* **114**, E10596-E10604
6. Gold, D. A., Caron, A., Fournier, G. P., and Summons, R. E. (2017) Paleoproterozoic sterol biosynthesis and the rise of oxygen. *Nature* **543**, 420-423
7. Amos, G. C. A., Awakawa, T., Tuttle, R. N., Letzel, A.-C., Kim, M. C., Kudo, Y., Fenical, W., S. Moore, B., and Jensen, P. R. (2017) Comparative transcriptomics as a guide to natural product discovery and biosynthetic gene cluster functionality. *Proceedings of the National Academy of Sciences*
8. Suh, A., Smeds, L., and Ellegren, H. (2018) Abundant recent activity of retrovirus-like retrotransposons within and among flycatcher species implies a rich source of structural variation in songbird genomes. *Molecular Ecology* **27**, 99-111.
9. Wang, L., and Jiang, T. A. O. (1994) On the Complexity of Multiple Sequence Alignment. Journal of Computational Biology 1, 337-348
10. Chor, B., and Tuller, T. (2005) Maximum likelihood of evolutionary trees: hardness and approximation. Bioinformatics 21, i97-i106
11. Fielding, R. T. (2000) Architectural Styles and the Design of Network-based Software Architectures. in Information and Computer Science, University of California, Irvine. pp
12. Pfeiffer, W., and Stamatakis, A. (2010) Hybrid MPI/Pthreads parallelization of the RAxML phylogenetics code. in Ninth IEEE International Workshop on High Performance Computational Biology (HiCOMB 2010) Atlanta
13. Huelsenbeck, J., and Ronquist, F. (2001) MRBAYES: Bayesian inference of phylogenetic trees. Bioinformatics 17, 754 - 755
14. Liu, L. (2008) BEST: Bayesian estimation of species trees under the coalescent model. Bioinformatics 24, 2542-2543
15. Suchard, M. A., and Rambaut, A. (2009) Many-core algorithms for statistical phylogenetics. Bioinformatics 25, 1370-1376
16. Stamatakis, A. (2014) RAxML Version 8: A tool for Phylogenetic Analysis and Post-Analysis of Large Phylogenies. Bioinformatics

17. Zwickl, D. J. (2006) Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion., The University of Texas at Austin. . pp

18. Drummond, A., and Rambaut, A. (2007) BEAST: Bayesian evolutionary analysis by sampling trees. BMC Evol Biol 7

19. Bouckaert, R., Heled, J., Kühnert, D., Vaughan, T., Wu, C.-H., Xie, D., Suchard, M. A., Rambaut, A., and Drummond, A. J. (2014) BEAST 2: A Software Platform for Bayesian Evolutionary Analysis. PLoS Comput Biol 10, e1003537

20. Price, M. N., Dehal, P. S., and Arkin, A. P. (2010) FastTree 2 – Approximately Maximum-Likelihood Trees for Large Alignments. PLoS ONE 5, e9490

21. Katoh, K., and Standley, D. M. (2013) MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol Biol Evol 30

22. Heath, T. A., Holder, M. T., and Huelsenbeck, J. P. (2012) A Dirichlet Process Prior for Estimating Lineage-Specific Substitution Rates. Molecular Biology and Evolution 29, 939-955

23. Flouris, T., and Stamatakis, A. (2012) An Improvement to DPPDIV Heidelberg Institute for Theoretical Studies

24. Darriba, D., Taboada, G., Doallo, R., and Posada, D. (2012) jModelTest 2: more models, new heuristics and parallel computing. Nat Methods 9, 772 - 772

25. Beerli, P., and Palczewski, B. (2010) Unified Framework to Evaluate Panmixia and Migration Direction Among Multiple Sampling Locations. . Genetics 185, 313-326

26. Swofford, D. L. (2002) PAUP*. Phylogenetic Analysis using Parsimony (* and other Methods). Version 4.0b10, Sinauer Associates, Sunderland

27. Besnier, F., and Glover, K. A. (2013) ParallelStructure: A R Package to Distribute Parallel Runs of the Population Genetics Program STRUCTURE on Multi-Core Computers. PLOS ONE 8, e70651

28. Lanfear, R., Frandsen, P. B., Wright, A. M., Senfeld, T., and Calcott, B. (2016) PartitionFinder 2: New Methods for Selecting Partitioned Models of Evolution for Molecular and Morphological Phylogenetic Analyses. Mol Biol Evol

29. Nguyen, L.-T., Schmidt, H. A., von Haeseler, A., and Minh, B. Q. (2015) IQ-TREE: A Fast and Effective Stochastic Algorithm for Estimating Maximum-Likelihood Phylogenies. Molecular Biology and Evolution 32, 268-274

30. Lartillot, N., Lepage, T., and Blanquart, S. (2009) PhyloBayes 3: a Bayesian software package for phylogenetic reconstruction and molecular dating. Bioinformatics 25, 2286-2288