Scalable Software Infrastructure for Integrating Supercomputing with Volunteer Computing and Cloud Computing

Ritu Arora[1], Carlos Redondo[2], and Gerald Joshua[3]

[1] Texas Advanced Computing Center, University of Texas at Austin, TX, USA
rauta@tacc.utexas.edu
[2] University of Texas at Austin, TX, USA
carlos.red@utexass.edu
[3] University of Texas at Austin, TX, USA
gerald.joshua153@utexass.edu

Abstract.
Volunteer Computing (VC) is a computing model that uses donated computing cycles on the devices such as laptops, desktops, and tablets to do scientific computing. BOINC is the most popular software framework for VC and it helps in connecting the projects needing computing cycles with the volunteers interested in donating the computing cycles on their resources. It has already enabled projects with high societal impact to harness several PetaFLOPs of donated computing cycles. Given its potential in elastically augmenting the capacity of existing supercomputing resources for running High-Throughput Computing (HTC) jobs, we have extended the BOINC software infrastructure and have made it amenable for integration with the supercomputing and cloud computing environments. We have named the extension of the BOINC software infrastructure as BOINC@TACC, and are using it to route *qualified* HTC jobs from the supercomputers at the Texas Advanced Computing Center (TACC) to not only the typically volunteered devices but also to the cloud computing resources such as Jetstream and Chameleon. BOINC@TACC can be extremely useful for those researchers/scholars who are running low on allocations of compute-cycles on the supercomputers, or are interested in reducing the turnaround time of their HTC jobs when the supercomputers are over-subscribed. We have also developed a web-application for TACC users so that, through the convenience of their web-browser, they can submit their HTC jobs for running on the resources volunteered by the community. An overview of the BOINC@TACC project is presented in this paper. The BOINC@TACC software infrastructure is open-source and can be easily adapted for use by other supercomputing centers that are interested in building their volunteer community and connecting them with the researchers needing multi-petascale (and even exascale) computing power for their HTC jobs.

1. INTRODUCTION

Due to the constantly increasing need for running large-scale applications, the supercomputing resources at open-science data centers can be over-subscribed at times, and when this happens, the turnaround time of small High-Throughput Computing (HTC) jobs can be longer than expected. To reduce the turnaround time of the small HTC jobs in such situations, these jobs can be routed to external computing resources. Such routing depends upon the users' consent to take advantage of the external resources, and the characteristics of their jobs, such as, the 1) anticipated job completion time, 2) amount of data to be ingested or produced during the job run, 3) amount of memory needed during run-time, and the 4) type of hardware resources needed (i.e., CPU or GPU).

The researchers may also be interested in using the external computing resources when they are running low on the compute-time granted to them through a competitive resource allocation process. Typically, it is hard to get 100% of the requested allocation of compute-time on the supercomputers that are in high-demand. Therefore, users with unsatisfied computational needs have to find additional resources to supplement their allocations. The additional computing resources can be an agglomeration of laptops, desktops, tablets, and the VMs in the cloud, and the computing cycles on these resources can be donated by the volunteers in the community, thereby, making the Volunteer Computing (VC) model relevant to the supercomputing user community.

We formally define VC as a computing model that uses donated computing cycles on devices such as laptops, desktops, and tablets to do scientific computing. BOINC [1] is the most popular software framework for VC and helps in connecting the projects needing computing cycles with the volunteers interested in donating the computing cycles on their resources. It has a client-server architecture, and has already enabled projects with high societal impact to harness several PetaFLOPs of donated computing cycles.

Given its potential in elastically augmenting the capacity of existing supercomputing resources for running HTC jobs, we have extended the BOINC software infrastructure and have made it amenable for integration with the supercomputing and cloud computing environments. We have named the extension of the BOINC software infrastructure as BOINC@TACC [2], and are using it to route *qualified* HTC jobs from the supercomputers at the Texas Advanced Computing Center (TACC) to not only the typically volunteered devices but also to the cloud computing resources such as Jetstream [3] and Chameleon [4]. ***We have developed a decision-support system for helping users determine whether or not their jobs are qualified for running through BOINC@TACC.*** Depending upon the hardware requirements of the BOINC jobs, the routing scripts also determine if these jobs should be run in the cloud or on other volunteered resources.

A high-level overview of BOINC@TACC software infrastructure is shown in Figure 1. As can be be noticed from this Figure, running jobs through the BOINC@TACC software infrastructure involves Docker [5], which is a commonly used tool for the containerization of applications. Both community code and users' home-grown applications can be containerized and made portable across different hardware resources and environments by using Docker. As a new functionality, ***we have developed a framework for automatically creating Docker images of user's home-grown code***, thereby, freeing them from the burden of climbing the learning curve for Docker and creating the Docker images of their applications. A complete overview of this system is available in Figure 2 below.

Additionally, we have developed the software components for running BOINC jobs on the VMs in the cloud. These component can also be useful for cloud bursting [6] and routing not just the HTC jobs but also the High Performance Computing (HPC) jobs from the oversubscribed resources to the relatively underutilized systems at the supercomputing centers. In order ***to support the cloud bursting mechanism, we have developed a new client software component for interacting with the BOINC server, and also a protocol for routing jobs and information between the new client and the BOINC server.***

The ***BOINC@TACC software infrastructure has been made General Data Protection Regulation (GDPR) [7] compliant.*** To ensure data privacy and security, our instance of the BOINC server is run on a private cloud computing infrastructure. For enabling TACC users in accessing BOINC@TACC through the project website, ***we have also integrated the TACC user database with the BOINC@TACC software infrastructure.***

We are iteratively refining the BOINC@TACC infrastructure on the basis of the feedback from the researchers, volunteers, and the developers in the community. In the rest of this paper, we describe the details of the BOINC@TACC infrastructure. We also describe the process of submitting jobs to the BOINC server through the convenience of the web-browser or through the command-line interface. We include a description of the evaluation metrics, and future work. The BOINC@TACC infrastructure can be adapted by different supercomputing centers in their pursuit of multi-petascale (or even exascale computing).

2. SOFTWARE AND IMPLEMENTATION

The key software components in the BOINC@TACC infrastructure are its front-end (command-line and web-based), BOINC client-server, special client for supporting cloud bursting, framework for automatically creating Docker images of users' custom-code, Docker images of the popular community applications, scripts, APIs, databases, job taxonomy, accounting system, and email notifica-

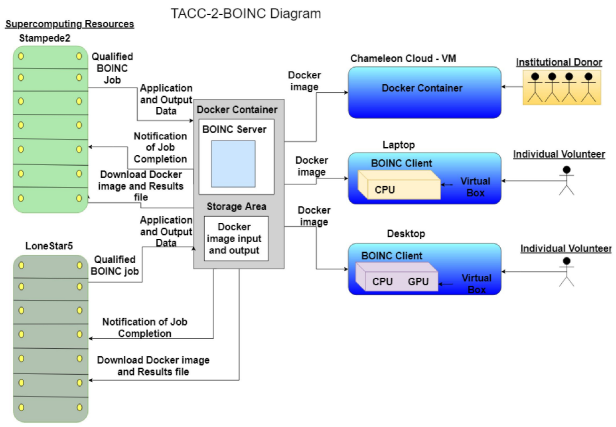

Fig. 1: *BOINC@TACC project overview*

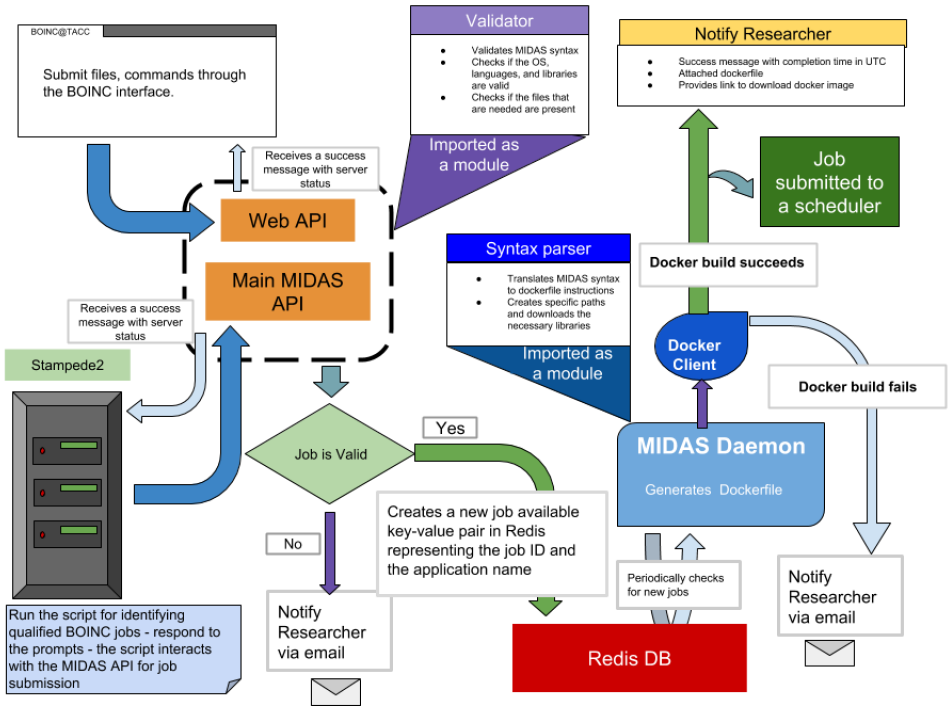

Fig. 2: *MIDAS: Automatic dockerization via image, the system can be accessed both through the Stampede2 scripts or the web interface.*

tion system. The majority of the implementation work required for this project was done using PHP/Javascript/HTML/CSS, Python, Bash scripts, Redis and MySQL database management systems.

We describe each of the aforementioned components in this section and provide an overview of the key software components/extensions that we have developed for integrating VC with supercomputing.

1. Front-End: The BOINC@TACC software infrastructure can be used for submitting jobs via two interfaces that were developed in the project: command-line and web-based. The command-line interface can be used on TACC supercomputers, and is basically a decision-support system that helps the users in determining whether their jobs are qualified for running through the BOINC@TACC infrastructure or not. If, as a part of the decision-making, it is determined that a job is not suitable for being routed to the BOINC server (perhaps because it is data-intensive or memory-intensive), then it is run directly on the TACC supercomputers. However, if a job is qualified for being routed to the BOINC server, there is further automatic decision-making involved to determine the type of resource on which the job should be scheduled, such as, a cloud computing resource or another volunteered resource that may be a laptop, desktop, or a tablet. Supporting cloud computing resources is especially important to meet the diverse hardware needs of the current TACC user-community. There is a high-demand for GPUs and some demand for FPGAs. The cloud computing resources can help in elastically supplementing the volunteered resources and provide new capabilities.

   The web interface supports job submissions through the convenience of the web-browser, and mimics the functionality of the command-line interface. However, before a researcher can use the web interface, they are required to run a registration script from a TACC supercomputer (viz., Stampede2 or Lonestar5) and get validated. The registration scripts are organization specific and can only be run from TACC systems. The web interface can also be used to track the status of a submitted job or to access any other feature provided by BOINC.

2. Databases: There are two types of database management systems used in the BOINC@TACC software implementation: MySQL and Redis. BOINC provides a MySQL database by default. This database stores the information related to the volunteers' account, in addition to the job statistics, data related to the messageboards, volunteered devices, and the status of the jobs. We have made several changes to the default MySQL database to anonymize user information while maintaining the compatibility of the data with the existing software implementation. All usernames have been anonymized in compliance with GDPR and are stored as such in the database. No one other than the user and the administrator has access to the information related

to user accounts. The users can choose to delete all the data, and we can guarantee its complete erasure from our systems.

Redis is an in-memory, key-value based database. It is used for data-caching and saving basic job information, such as job IDs and the image tags, while the complete information is stored in the MySQL database. Redis is also used for automatic Dockerfile construction, setting triggers that alert the internal daemon to begin the Docker build, and select the files to use. Redis is also used to store job classification tags for each job run.

3. BOINC client-server: The entire BOINC server architecture is composed of an Apache server, a MySQL database, and back-end programs/tools, and all of these components run as separate Docker containers [8] but are made to start/stop together using Docker compose [9]. The project's state is made to persist using shared Docker volumes [10]. All the default features of BOINC (such as messageboards, job scheduler, and job statistics) have been maintained in the BOINC@TACC project. The default BOINC client, when installed on a volunteered device, automatically polls the server to check for available jobs, and runs them if the volunteered device is underutilized. The BOINC client also automatically returns the job results to the BOINC server. The client notifies the BOINC server if there are any errors while processing a job. Any job that fails to run due to the error on the volunteered hardware or software is considered as incomplete and is resubmitted.

   One of the default requirements for running the Docker-based BOINC applications is the availability of VirtualBox on the volunteered resources. When the Docker containers are run inside the VirtualBox installed on the VMs in the cloud, they are unable to access the GPUs due to the hypervisor settings. While a Peripheral Component Interconnect (PCI) passthrough can be used to overcome this limitation, it is still in experimental stage, is difficult to set-up, and has strict hardware requirements that would make it difficult to uniformly use it on different volunteered resources. Moreover, on the Linux based resources/VMs, Docker containers can be run directly without requiring VirtualBox as long as Docker is installed. Therefore, in order to support Dockerized BOINC applications that need to access the GPUs, and to circumvent the requirement of installing VirtualBox on Linux-based resources (especially in the cloud), ***we have developed the Automatic Docker Task Distribution Protocol (ADTD-P).***

ADTD-P relies on the availability of Docker on the VMs for running CPU jobs, and requires Nvidia Docker 2 package [11] for running GPU jobs that are submitted through the BOINC server. Using ADTD-P does not require any modification to the applications themselves. The functionality of ADTD-P is further described as follows:

1. On the server side, ADTD-P saves Docker images of the applications to be run as a BOINC job in a *tar.gz file. It then creates a JSON file with the information about the job, such as, whether or not it requires GPU support. The Docker image and the JSON file are compressed and saved together as a packet to be shipped as a BOINC job.

2. It should also be noted that ADTD-P is not exclusive for BOINC usage and can be used for submitting other types of volunteer jobs as long as they are packaged using docker images. Clients can also be made to avoid GPU jobs if the Nvidia Docker 2 package is not installed or CUDA jobs are not desired. We provide an automatic installer for ubuntu/debian systems.

3. When a job has completed - successfully or with a failure - the client will return a *tar.gz file containing the job information logs in a JSON format and the output files to the server. Both these files are then forwarded to the users. In case a job fails due to the issues not related to BOINC, the error logs are also sent to the users. The Docker container and the image used are then deleted from the client.

4. ADTD-P clients also maintain a database to locally track the jobs that they process. This information is stored in the Redis database connected to the ADTD-P server and contains processing times, commands run, job status, and job IDs. The job statistics can be directly retrieved from Redis, but ADTD-P also provides a command-line interface [12].

5. Docker images: We maintain Docker images of multiple community applications such as, Autodock-vina, GROMACS, and OpenSees in Docker Hub. Users can either choose to run these applications that are maintained by us, or provide Docker images created by them or someone else. We have also developed a framework for supporting automatic creation of Docker images from source code and this framework is known as MIDAS (the Multiple Input Docker Automation System). Users can use MIDAS either in the command-line mode or through the web interface to generate Docker images from their source code and specifications. The user may select the OS to be utilized and and can also provide configuration options as they deem fit. Users are provided root access to the Docker image. If the image is built successfully, the researchers are notified via email and sent a copy of both the Dockerfile used to build the Docker image of their code, and a hyperlink to download the image. However, if the build fails, the researchers are notified and their job is not processed. To enable running the Docker images on volunteered resources, we use the boinc2docker tool [13]. This tool helps in transforming Docker images and their respective commands into a VirtualBox application. A complete diagram of MIDAS functioning can be seen in Figure 3 below.

6. Routing jobs from the BOINC server to the BOINC clients or directly to the VMs in the cloud: Jobs that the server processes using the boinc2docker tool are distributed to the volunteer devices using BOINC's default scheduler, which will select an appropriate volunteer host based on the job requirements (e.g., memory needs and the computation time required). Jobs submitted through the ADTD-P protocol, however, are processed on the First Come First Served (FCFS) basis, and are processed by the first ADTD-P client requesting a job. However, ADTD-P clients can have their own requirements, such as restricting GPU jobs.

7. Scripts for gathering results from the volunteered resources: BOINC jobs processed through BOINC's standard scheduler return the results using BOINC's default APIs through the BOINC client. These results are then added to a directory on the BOINC server. ADTD-P clients also return the results and a complete log of all executed commands to this directory.

8. Email notifications, job-tracking, and job statistics: We also support a job history page through which the researchers can track the status of the jobs submitted through the web interface. BOINC@TACC is also integrated with an email server for notifying the researchers about their jobs, results, and account information.

9. Job Classifier (Tagging): All jobs run using BOINC@TACC can be tagged - classified with none, one, or multiple science fields and subfields. Information about these tags is stored in a Redis database.

10. Accounting System: This system can be used by the organizations to check and limit the allocation of compute time and storage space for the researchers using volunteered resources. When a user signs up, allocation gets automatically assigned according to the parameters set by the organization. More than one organization can be served by a BOINC server instance and each one may provide different storage requirements and permissions to its users. In order for an individual to use the BOINC@TACC functionality, they must belong to an allowed organization which is TACC.

11. Information and System Security: Only researchers with both valid TACC credentials and an active TACC allocation are allowed to submit jobs through BOINC@TACC. Once a user has registered for BOINC@TACC project by running a script (provided by us), the system will automatically generate a token for him/her. This token is used to internally map the users and the jobs submitted by them.

    The BOINC@TACC project is GDPR compliant. To comply with GDPR, as a default setting, it was important to anonymize the volunteers' data presented on the leaderboard accessible through the web interface. If users wish to, and give their explicit consent, we can easily display their chosen screen-

names instead of the anonymized names. GDPR also mandates presenting the terms and conditions for joining the project in the simplest possible manner. The project team keeps track of all the places where the volunteers' data is kept so that in the event the volunteers need to delete their accounts, all their information can be deleted with certainty. This feature however may need to be rethought/reworked once the project is integrated with the Science United web application [14] in the future.

The default BOINC APIs sanitize all files submitted to ensure that there are no hidden commands within the file-names. The BOINC@TACC server is deployed on a private VMWare cloud computing system instead of Jetstream for reliability, security, and to ensure access to TACC's internal LDAP server. By default, the usage of bots to automatically run BOINC jobs is permitted and encouraged for users who have large computing systems and wish to do so. All users wishing to become volunteers and allow BOINC to compute on their devices will be required to pass a CAPTCHA test when signing up for the very first time.

3. JOB SUBMISSION WORKFLOW

Any researcher or scholar wishing to submit jobs using the BOINC@TACC framework must possess TACC credentials and a valid allocation. All prospective users are required to execute the registration script through either Lonestar5 or Stampede2 before being able to use the web interface. Researcher login is integrated with TACC's LDAP server in order to ensure appropriate access to computer resources.

We maintain a set of Docker images of popular community code (e.g., Autodock-Vina, GROMACS, NAMD, and OpenSees). A researcher/scholar can choose to run any of these with their input files/commands on volunteered resources while interacting with the BOINC@TACC infrastructure through the command-line interface or through the web interface. A screen-shot of the web interface is shown in Figure 3, and the scripts for using BOINC@TACC from the command-line interface have been made available through a Github repository [15].

Additionally, instead of choosing to run the applications whose Docker images are maintained by us, researchers/scholars can provide any public Docker Hub image along with the input files/commands to run.

BOINC@TACC also provides an automated process of creating custom Docker images from the source code for users who do not have existing Docker images of their custom-written/home-grown code. This process involves using the MIDAS software component. We support building Docker images of applications written in C, C++, Fortran, Python, R, and bash. These new images can contain any number of data files as well. The jobs submitted through the command-line inter-

face on Stampede2/Lonestar5, but not qualified to run on volunteered resources, are automatically submitted to the SLURM scheduler on Stampede2/Lonestar5.

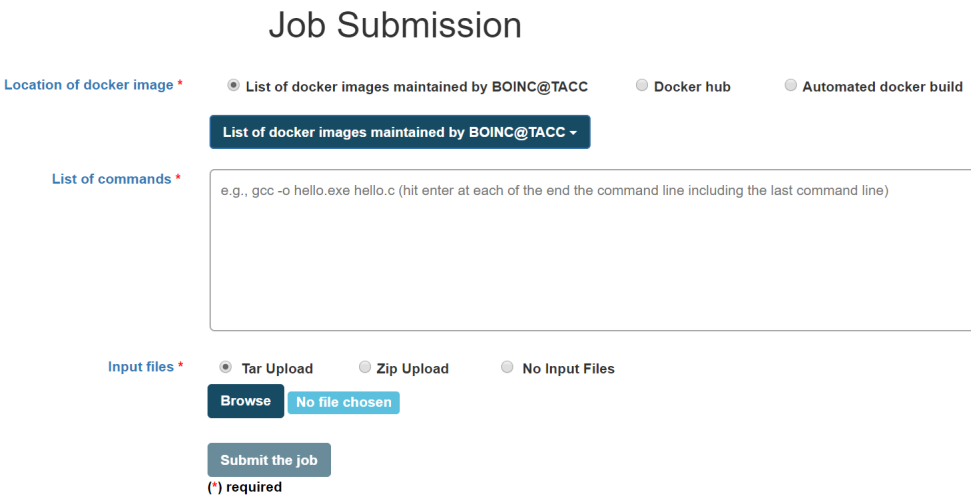

Fig. 3: *Web interface for BOINC job submission.*

4. BACKEND WORKFLOW

After job submission, the job information (user information, image used, commands, and input files) is relayed to the BOINC server where any required pre-processing is done (such as downloading data from a third-party server or converting the source-code to a Docker image).

The server automatically accesses the images supported by us without requiring any detailed user input. If a user selects to run a third-party image from Docker Hub, the BOINC@TACC system will prompt the user to specify appropriate tags for their jobs. MIDAS jobs - or those jobs for which the user needs help in generating Docker images - must be specified using a particular syntax. To the best of our knowledge, no other BOINC project supports the feature of automatically creating Docker images from source-code and user-specifications.

Most BOINC@TACC jobs are processed using the boinc2docker tool and are made available to the BOINC scheduler directly. The BOINC scheduler selects an available volunteered device based on the job characteristics, such as the memory and I/O requirements. The devices running BOINC clients are expected

to have VirtualBox installed. The Docker image runs inside the Docker container, which in turn runs in the VirtualBox connected via BOINC client.

BOINC@TACC is also integrated with a secondary job submission framework: the Automated Docker Task Distribution Protocol (ADTD-P). ADTD-P does not require VirtualBox and is only dependant on Docker for CPU-only jobs. It requires CUDA drivers and the Nvidia Docker 2 package for CUDA GPU jobs. When an ADTD-P job is available, a client running on Jetstream or Chameleon downloads the Docker image, loads it, and executes the commands inside a Docker container. If the job is successful, it collects the results and uploads them to the BOINC server. It also logs all the job processing details. A pictorial overview of the ADTD-P framework is shown in Figure 4.

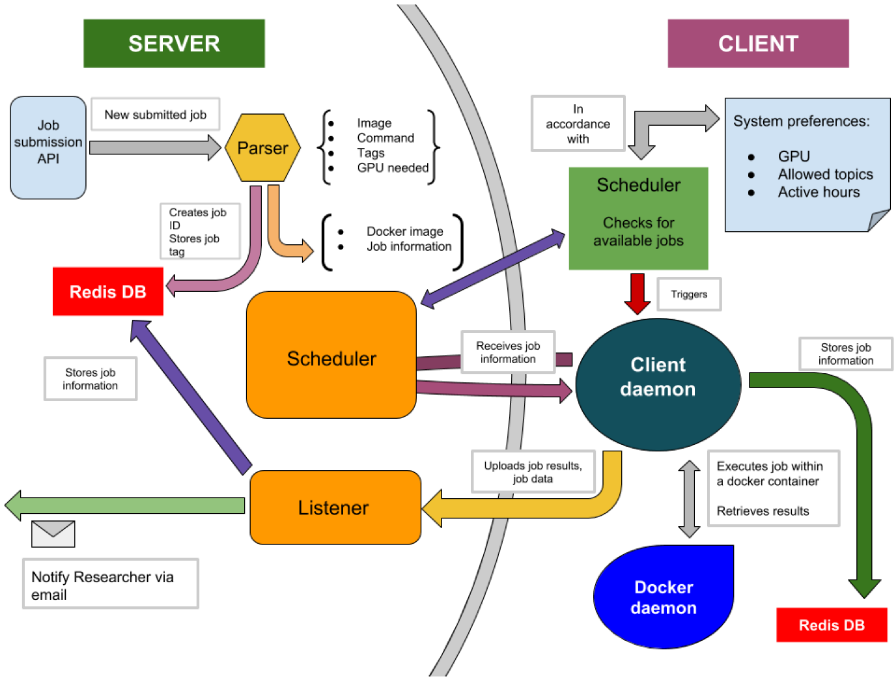

Fig. 4: *MIDAS: ADTD-P system as implemented in BOINC@TACC.*

When a job is completed, the BOINC client notifies the BOINC server, which then sends an email message to the researcher/scholar along with the link to download the package containing the output files. The researchers/scholars can also monitor the status of their job through the BOINC@TACC project website: http://boinc.tacc.utexas.edu/ (login as a researcher is required). All jobs submitted through BOINC@TACC are tagged, and the data related to the tags will be used in future to integrate the project with the Science United project

so that the volunteers can choose the projects that they wish to support more easily.

5. SCALABILITY

The BOINC@TACC system can elastically scale-up to support HTC jobs by harnessing the VMs running in the cloud. In comparison to a regular server or container clusters, the number of jobs run through BOINC@TACC can become virtually infinite as long as volunteers provide devices to execute the computation.

All user data and results are stored on a Docker volume mounted on the host for data persistence. This container can also be run on a different server in order to reduce the disk space requirements of the main server. Information credentials for passing information between the BOINC server and the storage server (named as Reef) are not shared directly with the users for security purposes. Furthermore, all communication and files passed to storage must go through the BOINC server.

6. CHALLENGES AND LESSONS LEARNT

- Data anonymization: For GDPR Compliance, we initially added a new column to the MySQL database for storing the anonymized volunteer names. This created issues related to inconsistent display of volunteer information across the project website. BOINC recognizes only the default structure of the MySQL database, which does not contain anonymized names. To fix this issue, we created a new table that only stores the anonymized user name for each user's screen name.

- Emails: Users are sent results attached to an email and a link for future downloads is provided. Gmail was used in the early stages but it was discontinued in favor of GNU Mailman because Gmail has storage limits and Google can shut-down Gmail accounts that send automated mails to discourage any spam-related activity.

- VirtualBox installation: It was not possible to use VirtualBox on the VMs available to us due to a communication issue between the BOINC client and VirtualBox for running GPU jobs. ADTD-P was created to run BOINC jobs directly in the VMs without using VirtualBox.

- Nvidia GPU access through Docker: Regular Docker containers do not have access to an Nvidia GPU even when the necessary drivers and CUDA services have been installed. As a result, ADTD-P clients were built using the Nvidia Docker 2 package. This allows containers to run CUDA jobs without any extra setup, while retaining the same functionality and syntax of a regular Docker job.

7. LIMITATIONS

One of the main limitations of VC is the volatility in the availability of the volunteered resources, and their performance. While the users are guaranteed a certain level of performance when running applications on supercomputers, they may experience unpredictable performance when running on volunteered resources. A lot of the supercomputer users run memory-intensive and IO-intensive applications which are not a good-fit for the VC environment. A lot of the jobs run at the supercomputing centers are based on distributed-memory paradigm, and such jobs are also not a right fit for the VC model.

8. EVALUATION

We are in the process of evaluating the BOINC@TACC project in terms of the job turnaround time, the number of jobs submitted, the number of jobs that run successfully, and the number of volunteers and researchers that are interested in using the project.

As the project is in its early stages of community building, currently, we have only about 65 volunteers and researchers who are being serviced through the BOINC@TACC project, and this number will likely increase with future community building efforts. We have not opened the system to unknown researchers.

We selected popular community code for our testing and evaluation: AutoDock Vina [16] from the computational biology domain, Opensees [17] from the earthquake simulation engineering domain, and three applications from the molecular dynamics domain - NAMD [18], LAMMPS [19], GROMACS [20]. These applications were submitted through the BOINC@TACC command-line interface available on the Stampede2 supercomputer. We also ran these applications directly on the Skylake nodes on the Stampede2 supercomputer at TACC. The Skylake nodes are high-end as compared to the processors on the volunteered resources, and as can be noticed from the data in Table 2, the performance of the applications on Stampede2 was better than their performance on the volunteered resources. As we did not experience any queue wait time at the time of running these test jobs, the job turn-around time on the volunteered resources is longer than the time on Stampede2. However, the queue wait time on Stampede2 is a highly unpredictable characteristic that depends upon the other users who are simultaneously using the system. When Stampede2 is down for maintenance or has a large back-log of pending jobs, the turn-around time from the volunteered resources could be shorter than that of Stampede2.

Table 1: Comparing job turn-around time (email sending time not included)

Application Name	**Image Size**	**Stampede2** Skylake node -4 cores requested	**BOINC@TACC**	
			Boinc2docker (VirtualBox in in 4-core volunteer)	ADTD-P (Jestream cloud server, 6 cores, 4 used)
Autodock-Vina	697 MB	30 s	6 min 32 s	8 min 30 s
OpenSees	1.331 GB	5 s	3 min 19 s	10 min 18 s
NAMD	288 MB	5 s	15 min 47 s	3 min 22 s
LAMMPS	1.47 GB	3 s	3 h 14 min 19 s	12 min 13 s
GROMACS	1.27 GB	8 s	3 h 8 min 34 s	11 min 58 s

Table 2: Comparing the computation time spent

Application Name	**Image Size**	**Stampede2** Skylake node -4 cores requested	**BOINC@TACC**	
			Boinc2docker (VirtualBox in in 4-core volunteer)	ADTD-P (Jestream cloud server, 6 cores, 4 used)
Autodock-Vina	697 MB	28 s	5 min 21 s	1 min 9 s
OpenSees	1.331 GB	3 s	2 min 44 s	<1 s
NAMD	288 MB	<1 s	23 s	2 s
LAMMPS	1.47 GB	3 s	1 min 45 s	<1 s
GROMACS	1.27 GB	<1 s	1 min 38 s	<1 s

9. RELATED WORK

There are multiple VC projects in the community that are using BOINC for harnessing the power of the donated compute-cycles. These projects are typically deployed by the research groups for running specific applications to solve very specific research problems that are related to their interest. Contrary to the objective of other VC projects in the community that run niche applications, the BOINC@TACC project is designed to cater to the computing needs of a wide range of TACC users from different research domains, and hence, supports running arbitrary applications in Docker containers.

10. FUTURE WORK

All current BOINC logs about submitted jobs are stored in MySQL by the scheduler, and in an additional Redis database for internal server mapping. However, as mentioned above, Redis is an in-memory database and is not designed for large job logs. Instead, InfluxDB is a better choice for handling large amounts of time-series data, and will be used in the next iteration of the BOINC@TACC software release. The Grafana interface [21] will also be added to facilitate administrative overview of job statistics. Currently, the main BOINC server automatically generates the Docker image using the MIDAS client locally. It also packages and archives all ADTD-P information locally. This could become a problem when

an extremely large set of jobs is submitted, since it becomes a bottleneck to job processing. In the future software releases, we will improve the scalability of our software.

11. CONCLUSION

In the paper we presented an introduction to the BOINC@TACC project, and discussed the advantages of unifying supercomputing with volunteer computing and cloud computing. We explained that the BOINC@TACC project can potentially help users in supplementing their allocation of compute-time on the TACC supercomputers, especially because there are roughly 500,000 devices in the community that are already actively participating in the VC projects. Some of these devices are equipped with modern GPUs. More than 100 volunteers have already signed-up for the BOINC@TACC project.

We also discussed the need for harnessing the available computing power through the cloud computing systems hosted by TACC so that if there is a spike in the demand for the computing power, or need for special hardware that is not available on the other volunteered resources, we can still service the jobs submitted through BOINC@TACC with a reasonable guarantee for the quality of service. The software infrastructure for the BOINC@TACC project is being iteratively refined and released to the public. Other supercomputing and cloud computing service providers can conveniently adapt and adopt the BOINC@TACC software infrastructure for their environments.

12. ACKNOWLEDGEMENT

The BOINC@TACC project is funded through National Science Foundation (NSF) award # 1664022. We are grateful to XSEDE, TACC, and the Science Gateway Community Institute for providing the resources required for implementing this project. We are grateful to David Anderson, Thomas Johnson, and Anubhaw Nand for contributing to the BOINC@TACC codebase and their contribution in preparing this paper. Figure 1 was prepared by Thomas Johnson. Several results presented in this paper were obtained using the Chameleon testbed supported by the NSF and we are grateful to NSF for the same.

13. REFERENCES

[1] Anderson DP (2004) "BOINC: A System for Public-Resource Computing and Storage". Fifth IEEE/ACM International Workshop on Grid Computing

[2] BOINC@TACC, website accessed on October 26, `http://boinc.tacc.utexas.edu/`

[3] Jetstream, website accessed on October 26, https://use.jetstream-cloud.org/

[4] Chameleon, website accessed on October 26, https://www.chameleoncloud.org/

[5] Docker, website accessed on October 26, https://www.docker.com/

[6] Cloud Bursting, website accessed on October 26, https://azure.microsoft.com/en-us/overview/what-is-cloud-bursting/

[7] GDPR Compliance, website accessed on October 26, https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and\discretionary{-}{}{}organisations_en

[8] Docker Containers, website accessed on October 26, https://www.docker.com/resources/what-container

[9] Docker Compose, website accessed on October 26, https://docs.docker.com/compose/overview/

[10] Docker Volumes, website accessed on October 26, https://docs.docker.com/storage/volumes/

[11] NVIDIA Docker Github Repo, website accessed on October 26, https://github.com/NVIDIA/nvidia-docker

[12] ADTD-P Protocol Github Repo, website accessed on October 26, https://github.com/noderod/adtd-protocol/blob/master/history.py

[13] BOINC2Docker Github repo, website accessed on October 26, https://github.com/marius311/boinc2docker

[14] Science United, website accessed on October 26, https://scienceunited.org/

[15] BOINCatTACC Github Repo, website accessed on October 26, https://github.com/ritua2/BOINCatTACC

[16] Trott O, Olson AJ (2010) "AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading". Journal of Computational Chemistry, 31(2): 455–461.

[17] Opensees, website accessed on October 26, http://opensees.berkeley.edu/

[18] Phillips JC, Braun R, Wang W, Gumbart J, Tajkhorshid E, Villa E, Chipot C, Skeel RD, Kale L, and Schulten K (2005) Scalable molecular dynamics with NAMD. Journal of Computational Chemistry, 26:1781-1802.

[19] Plimpton S (1995) Fast Parallel Algorithms for Short-Range Molecular Dynamics. Journal of Computing Physics, 117:1-19, http://lammps.sandia.gov

[20] Berendsen HJC, Drunen RV, Spoel DVD (1995) GROMACS: A message-passing parallel molecular dynamics implementation. Computer Physics Communications, 91:43-56

[21] GRAFANA, website accessed on October 26, http://docs.grafana.org/features/datasources/influxdb/