Volunteer Computing: A Road to Efficient and Low-cost Scientific Research
Sukanya Devi Ramachandran
S 3140964

ABSTRACT
Grid computing is the process of connecting numerous computing resources that are loosely coupled, heterogeneous and geographically dispersed. In general a middleware determines how scalable and robust a grid can be. Volunteer computing is the act of using the unused computing resources from household computers as a part of the grid. Volunteer computing significantly helps several real time projects related to Climate study, Artificial Intelligence, Cryptography, space science, Chemistry and Biology. Volunteer computing is perceived by many research organizations that have less funding to buy computing resources. Volunteer computing demands a robust middleware that could provide a scalable communication framework and security. One of the most popular middleware for volunteer computing is BOINC (Berkeley Open Infrastructure for Network Computing). There are two major challenges in designing a robust volunteer computing architecture; One, the challenge of enhancing the communication framework and two, the challenge of saving the grid from attackers. This paper proposes efficient communication framework that will dynamically form smaller clusters of closely located computing nodes. In addition, the proposed communication framework is application specific; that is the communication framework switches its clustering algorithm specifically with respect to different applications. This paper aims at enhancing a commercially popular volunteer computing framework to eventually outstrip propriety grid computing frameworks.

Keywords
Grid Computing, Volunteer Computing, BOINC.

1. INTRODUCTION
1.1 What is Volunteer computing?
Volunteer computing [6] is a distributed computing arrangement in which internet-connected computer owners donate their computing resources to one or more “projects” (shown in Figure 1). The computing resources are such as processing power and storage. While the owner is not using the computer's entire power, volunteer computing software uses the leftover computer power to solve calculations, perform simulations and otherwise contribute to some amazing projects. Projects are typically academic and do scientific research. There are a few non-academic projects as well [7].

1.2 Volunteer computing Vs. Grid computing Vs. Peer to Peer computing
Volunteer computing is neither grid computing nor peer to peer computing. Grid computing combines computers from multiple administrative domains or organizations to reach a common goal, to solve a single task, and may then disappear just as quickly. The organizations are mutually accountable. If one organization misbehaves, the others can respond by suing them or refusing to share resources with them. This is different from volunteer computing. Volunteer computing is different from peer to peer computing in the fact that it uses central servers. Peer-to-peer computing benefits the participants there is no notion of a ‘project’ to which resources are donated.

Figure 1. Volunteer computing architecture

1.3 Why Volunteer computing?
Traditional supercomputers are extremely expensive, and are available only for applications that can afford them. But with volunteer computing, a research project with limited funding but large public appeal can get huge computing power. Volunteer computing can supply more computing power to science than any other type of computing considering the large number of Personal computers that are available worldwide. The popularity of PCs and game consoles is never fading. The count of such consumer products keeps increasing every day. The number of Internet-connected PCs is growing rapidly, and is projected to reach 1 billion by 2015[2]. Volunteer computing encourages public interest in science, and provides the public with voice in determining the directions of scientific research.

1.4 Volunteers
Volunteers are not linked to a real-world identity and are effectively anonymous. Because of their anonymity, volunteers are not accountable to projects. It is left to the server to handle problems such as malfunctioning and incorrect results returned by the clients. The servers use a common approach called “replicated computing” to overcome these problems. The job is performed on at least two volunteer computers and the results are accepted only if they agree sufficiently.
Costs for volunteer computing participants: A volunteer participating in a project is forced to leave the PC switched on or disable power-saving features like suspend. This results in increased power consumption. In addition proper cooling has to be provided to prevent overheating due to this constant load on the volunteer's CPU. The volunteer computing application causes increase contention of CPU cache, disk I/O and network I/O if it attempts to run while the computer is in use. This can degrade the performance of the computer. Even if these effects are noticeable, the volunteer might choose to continue participating. Some of the volunteer computing clients like BOINC get details of the processor usage percent to take care of this increased power consumption. The volunteer trusts the project to follow proper security practices, so that hackers cannot use the project as a vehicle for malicious activities.

2. MIDDLEWARE
The early volunteer computing client softwares had monolithic architecture that was inflexible. It consisted dependent scientific computation and distributed computing infrastructure. Lately volunteer computing has moved to middleware systems that provide an autonomous distributed computing infrastructure. Some of the commonly used middlewares are: The Berkeley Open Infrastructure for Network Computing (BOINC) [8] which is the most widely-used middleware system. Other middlewares are XtremWeb, Xgrid, Grid MP.

These middleware systems have a client program running on the volunteer's computer. Since many volunteer computers are behind firewalls they use the “pull” model. The volunteer periodically contact the project-operated servers over the Internet, requesting jobs and reporting the results of completed jobs. The system keeps track of each user's "credit", a numerical measure of how much work that user's computers have done for the project.

The first volunteer computing project was GIMPS (Great Internet Mersenne Prime Search), which started in 1995. Other early projects include distributed.net, SETI@home, and Folding@home. Today there are over 50 active projects. The details about these projects are given in [3].

This paper focusses on the enhancement of the performance of volunteer computing. The existing techniques that improve the volunteer computing performance are pertaining to load balancing [4] and increasing the robustness of the middleware [5]. In this paper, it proposes two different algorithms to enhance the performance of volunteer computing. The proposed algorithms are devised for overcoming practical challenges. Since, there is a high demand on utilization of volunteer computing by several clients around the world, it is a crucial requirement to enhance its performance by utilizing the behaviors of the growing utilization.

3. PROPOSED ALGORITHMS
The two algorithms proposed in this paper are based on location proximity and application specific. In addition to these two algorithms, this paper will explore the pros and cons of both the aspects and finds a third optimistic algorithm which optimizes both proposed algorithm.

3.1 Proximity based algorithm
In simple terms, the proximity based algorithm tries to find the best nodes to share the resources for a particular client with respect to proximity. The proximity algorithm will be added as a component to the existing middleware available for supporting volunteer computing. The proposed algorithm will first detect the location coordinates of the client and will detect the closest sub server to which the client can get connected. Since, the clients are going to be stationary the volunteers are going to be dynamic. The proposed algorithm thrives among existing extensions to efficiently support the clients by inducing opportunistic clustering. We modify the volunteer computing architecture (shown Figure 2) by adding sub servers at hot spots. The hot spots are close location coordinates near the clients. Each sub servers act exactly like the main server, but mainly involves in clustering the volunteers as and when the volunteers join the computing grid.

- All the sub servers will be in the learning phase. During the learning phase it detects its closest clients and computes a distance metric with which it will initiate the clustering.
- In the next stage the sub servers keeps collecting the volunteers that are at close proximity (decided with the help of the distance metric). After detecting the volunteers it will create preliminary clusters. A threshold of retries decides when the sub server should enter into normal phase from learning phase. Whenever a new client leases the grid resources the sub server goes through the cycle of entering into learning phase. The sub server is also recommended to switch to the learning phase at regular intervals to refresh unwanted data structure elements. Typically the retries in the proposed algorithm will be until the sub server finds few nodes that can suffice at least 80% of the computing resources required by the client.
- In the third stage the sub server will be able to track the behavior of the dynamic volunteers. Now it goes to the normal phase with a primary cluster and a secondary cluster of volunteers to serve the client. The primary cluster is given more priority since this cluster will have volunteers who are permanent as compared with the secondary cluster of volunteers.
- The final stage is the normal phase. In this stage, whenever a client with particular distance metric requests for resources it will serve with the primary cluster rather than any volunteer as it was before.
- To elucidate the advantages of proximity based volunteer computing algorithm over normal methods let us consider, a client from United States of America. When the proposed algorithm is not used then this client would non-deterministically be served by volunteers from Canada, Europe and even Asia. The client will be served with all the resources the client needs. However the communication model used for communicating between different volunteers around the
work will be very expensive when the wrong volunteer is chosen. Instead, the proposed proximity-based algorithm carefully chooses the cluster to serve and promises at least 80% of the computing power to serve the client.

### 3.2 Application specific algorithm

As the name states, this is the second proposed algorithm. Although there are some existing proximity-based clustering algorithms [4] in the field of peer-to-peer streaming, wireless networks, etc., the algorithm proposed here is a novel introduction to the volunteer computing domain. Similarly, the application-specific algorithm is another novel approach to enhance the performance of volunteer computing. This concept can also be extended to grid computing. However, we propose for enhancing the performance of volunteer computing.

Being a volunteer is a simple process. A user has to download the middleware and will be asked series of questions about the email address and other login details. Then, the user has to agree to provide certain amounts of resources fixed by the user himself. The resources set to share can be some part of the hard disk space, RAM memory, GPU power, and even internet bandwidth. Once the user agrees this until the user declines the agreement the user will be a volunteer by sharing the resources.

Any middleware after getting all the information from the user, the computer will have a thorough check for all the peripherals before accepting the agreement. It will report to the central server database about the addition of the resources. It should be noted that the increase in overall resources will be reflected as a whole. For instance, when a volunteer agrees to share 100 GB of hard disk space and 2 GB of RAM, the middleware will not require intricate details and will just add the overall memory space available as x + 100 GB and adds the RAM as y + 2 GB RAM. This approach might work well in all situations, but this paper claims that the middleware should have more intelligence to distinguish the resources rather than just adding them up.

For instance, let us consider a client from NASA who is using the volunteer grid for the sake of operating with satellite readings and other high input data sets. Similarly, consider another application which is used for Genetic purposes. It is not always, that both the applications require the same type of resources like, the computation of numerical values and data sets will require comparatively more processor power than the genetic problem. On the other hand, the genetic algorithm which relies on parallel processing depends on GPU power more than the former example. In addition to that there are still some applications that require specific clock speed of the processor, new generation RAMs etc. These differences are not captured by the current middleware. In fact, the current middleware doesn’t even know this intricate requirement, mainly because the client’s agreement is successful when it gets the required amount of electricity.

This paper proposes an algorithm that detects the application requirements of the client. It also finds the right volunteer group cluster to serve the client. This algorithm should not be confused with the proximity-based clustering algorithm. The proposed application-specific algorithm adds more intelligence to the middleware. Similar to the proximity-based algorithm, this algorithm has multiple stages,

- In the first stage the volunteer’s machine is completely tested and an image of the machine is sent to the closest sub-server of the client, same as the previous algorithm. When the sub-server gets this image it will add the resources list as usual but will map all the intricate detail in the resource from the image it received about.
- During the second stage the sub-server has to fetch the volunteers related to the requirements of the client. But, this straightforward method will not work since not all the users might have the right configurations. In addition, if we only focus on the accurate application-specific requirement then we will not be able to satisfy even 80% of the total agreed power.
- To overcome that challenge, the clustering is done dynamically and will not involve any learning phase as in the previous algorithm. During dynamic allocation, polling is made between volunteers’ images and the most optimal application-resource match will be elected to participate in the cluster. Recall that the main concern of volunteer computing is not to involve volunteers in any algorithms and that is the reason for having the images of the volunteer’s system.
- The poll results derive the application-specific metric. Each volunteer’s image will be designated with the specific metric.
- Thus all the volunteers failing inside the metric threshold will be clustered and served to the client. Now, the client’s computational capacity obviously increases, since the client is going to get what it exactly wants.
- This algorithm can also help in places when the volunteer don’t obey their agreements resource patterns. For instance, the volunteer agrees to give the most of its GPU power and eventually runs a high graphical pixel shader program which consumes the GPU power. In that case the volunteer cannot be blamed.

![Figure 2. Modified volunteer computing architecture](image-url)
but the sub server will get an opportunity to detect that and give chance to the next available volunteer to participate in the sharing.

### 3.3 Optimized solution

From the algorithms we derived earlier we find a common disadvantage, when the client requests require an optimistic solution from both the algorithms rather than the single metric. This leads us to find an optimistic solution. This solution uses both the metrics derived from the aforementioned algorithms. Then an optimization problem is solved to find the optimistic solution. The equation is

\[ D_p + A_m \geq Q_p \]

where, \( D_p \) is the distance metric, \( A_m \) is the application specific metric and \( Q_p \) is the optimistic value.

### 4. CONCLUSION

This paper focuses on volunteer computing which significantly helps several real time projects related to Climate study, Artificial Intelligence, Cryptography, space science, Chemistry and Biology. Volunteer computing is perceived by many research organizations that have less funding to buy computing resources. Volunteer computing demands a robust middleware that could provide a scalable communication framework and security. However, the existing extensions of volunteer computing are related to load balancing and improving robustness. This fact motivated to propose two algorithms that clusters volunteers intelligently based on proximity and application specific. Finally this paper consolidates both the proposed algorithm and finds an optimistic solution for enhancing the performance of volunteer computing. Thus this paper claims that volunteer computing will lead to efficient and cost-effective scientific research [1].

### 5. REFERENCES


