Summary

BOINC, the Berkeley Open Infrastructure for Network Computing, is a software system for public-resource computing. Volunteer computing, sometimes called public-resource computing, utilizes the combined computing power of millions of personal computers to do scientific supercomputing. BOINC allows scientists to create applications that will be executed on the personal computers of any volunteers that are participating in the project. The overall goals of the BOINC project are to ease entry to volunteer computing, share resources among autonomous projects, support diverse applications, and to retain participants.

There are several challenges that arise in a volunteer computing environment that are not present in other computing environments. Primarily, projects relying on volunteer computing need to be able to work with vastly diverse resources. Secondly, project operators need to deal with issues arising from not having fine control over the resources. Furthermore, those working in volunteer computing need to not only attract new participants, but also retain existing ones.

One of the most important aspects of BOINC is the scheduler. The scheduler makes local decisions in order to maximize the use of resources while satisfying deadlines. BOINC gives participants a great deal of control over how their machine is used for computing, and all of these preferences must be respected by the scheduler. For example, participants specify how much disk space, memory, network, and processor time the projects are allowed to consume. Additionally, computations can be limited to specific times on specific days of the week. The scheduler must also take into account the resource share assigned to a project by the participant and make it appear that processing time is actually being shared.

Review

The most important difference between volunteer computing frameworks and other computing frameworks such as Apache Hadoop is the vast diversity between resources available to the system. This diversity includes not only the speed, number of processors, memory, and disk space at each resource but also the operating system and hardware installed on the machine. Furthermore, different resources have different levels of availability and reliability, which can change unpredictably over time. This diversity means that the scheduling algorithms used with other computing frameworks are no longer effective since the assumptions about the system no longer hold. For the most part, existing scheduler research in areas such as grid computing cannot be applied in a volunteer computing setting.

Three papers related to scheduling are critiqued in this review. All three papers bring something to the table with regards to how the BOINC scheduler can be improved. One paper, [2], looks at how client policies can be improved to reduce idleness and wasted CPU cycles. Another paper, [3] looks at how the BOINC project parameters can be tuned using an emulator to improve project throughput, latency, and starvation. The third paper, [4], looks at how excluding certain participants can influence a projects valid task rate.

EmBOINC is a trace-driven emulator of BOINC projects. The server interactions are emulated, while the actions of the volunteer hosts are simulated. This hybrid emulation with simulation helps to avoid the maintenance burden associated with a rapid development cycle. It communicates directly with the project's servers, which triggers task generation, distribution, collection, and validation.

An inefficiency of this approach lies in the way EmBOINC interacts with the existing BOINC services. The BOINC daemons utilize wall-time to properly function. The discrete event simulator events operate on simulates time. In order for the BOINC daemons to properly respond to the simulated events, the simulated time must be sent to the BOINC server which sends a signal to the daemons causing them to wake and synchronize with the new time.

The paper provided three case studies using EmBOINC. There are a few weaknesses in the case studies. The simulated hosts used in the three case studies are all based on those of the Docking@Home

project, generated using its database. The studies could have been improved by selecting two projects with different participants and comparing the simulator's performance during each of the studies. The paper needed to discuss how well EmBOINC was able to simulate the systems performance during these case studies.

One of the challenges of operating a project in a volunteer computing environment is that volunteers are not always available or reliable. A once available machine may change its regular uptime, or a volunteer may leave a project altogether. Erroneous results can occasionally be produced by misbehaving or malicious tasks. By default, BOINC handles these situations by replicating work. Using a quorum, a canonical valid result is selected from the submitted results. As long as the required number of of results is submitted, the task succeeds. Paper [4] provides a solution that helps improve the rate at which results are successfully returned to the server and validated. The approach involves tracking a participant's availability and reliability over time. The project servers will only assign work to nodes that have ratings above a threshold.

This approach has the effect of permanently removing poor performing participants from the project. Any eliminated workers are now starved and idle, a result in conflict with the work of [2] which attempted to reduce idleness when assigning tasks to workers. One addition, having the rating of a worker improve over time allows the worker to begin participating again, but does not solve the issue of idleness. An alternative is to assign an additional replica task to the worker. Even though the project would not be relying on the result of the replica, it would have the benefit of potentially improving validation time if it was returned in a timely manner.

Another tactic provided by BOINC to help improve the stability of the system is deferred communications. With potentially millions of participants, the central servers can easily become overloaded. In the case of a failure, exponential backoff of communications helps prevent a cycle of overload failures at the central project servers. Workers usually only contact the project servers once per day or when they run out of work, but this poses a challenge that none of the papers directly tackled. When a client requests work, it receives enough tasks to keep the worker busy for an extended period of time, typically at least half a day. In the case where faster workers have already returned and validated the result, the other workers with a replica are wasting CPU time on a completed task. A solution is for workers to occasionally confirm tasks with the central servers. This will allow the server to cancel tasks that have completed and can even allow the reassignment of tasks that are unlikely to complete successfully by the deadline. On the other hand, this type of querying reopens the issue of overloading the central servers. To alleviate this, the rate at which nodes perform this operation can be tuned based on the number of participants, number of replicas, and the quorum.

A challenge of volunteer computing is retaining participants. BOINC retains participants using two main tactics, graphics and credit. The graphics include screensavers that show the progress of a currently executing task. Credit, on the other hand, is claimed and awarded after a computation completes. The amount of credit awarded is typically the average or minimum of the claims of all replicas of a specific task, where the claims are based on CPU time and benchmark results. The system is designed to prevent cheating, but the amount of credit awarded often ends up being influenced by which workers the replicas end up being assigned to instead of how much effort a worker actually contributed to a project. Awarding fixed credit based on the job's FLOPS estimate fixes this issue.

References

[1] David P. Anderson. 2004. Boinc: A system for public-resource computing and storage. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*. IEEE (2004), 4-10.

[2] David P. Anderson, Derrick Kondo, and John McLeod VII. 2007. Performance Evaluation of Scheduling Policies for Volunteer Computing. *e-Science and Grid Computing, IEEE International Conference on*. IEEE (Dec. 2007), 415-422.

[3] David P. Anderson, Trilce Estrada, and Michela Taufer. 2009. Performance Prediction and Analysis of BOINC Projects: An Empirical Study with EmBOINC. *Journal of Grid Computing* 7, 4 (2009), 537-554.

[4] Trilce Estrada, David A. Flores, Michela Taufer, Patricia J. Teller, Andre Kerstens, and David P. Anderson. 2006. The Effectiveness of Threshold-based Scheduling Policies in BOINC Projects. In *Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing*. IEEE, 2006.