Reservation Algorithm for Consistent Performance of Distributed Systems

Muhammed Suhail T. S.
SE Tata Consultancy Services
Pooranam House
Nellikkunnu Kasaragod

ABSTRACT
In this paper, a study on how to stabilize the performance of a distributed grid computing system is discussed. The key performance characteristics such as response time, throughput and scalability are vital to the operation of grid computing systems. Moreover, it is of utmost importance to have a computing network which is stable and less volatile. That is, when the performance factors of grid increase, it is prone to instability. Instability can arise due to rapid connection and disconnection of systems in the grid. In volunteer computing networks this is a huge problem that needs to be addressed. It is impractical to implement a complex algorithm to stabilize the grid as it would again require computing overheads and delays in response, which will make the system slow. Instead by using the data captured by the grid, the system needs to employ an algorithm which does not utilize huge computational power, at the same time, can be used as a versatile reservation algorithm.

Keywords
GRA – Grid Reservation Algorithm

1. INTRODUCTION
Distributed systems are geographically separate systems, connected through a network, working to accomplish a common goal or task. In such a connected networking system, the most important element is the systems itself which are connected to the grid. As the performance of the grid is completely dependent on the systems which are present in the grid, the entry and exit of systems to and from the grid environment will have an impact on its performance.

For any system to be operationally efficient, its performance needs to be steady and reliable. If a computing environment is highly volatile and unstable, the usability of the system is decreased drastically. For a distributed computing network, it is important to have a stable and reliable computation capacity at all time, whereby it could be made usable.

2. CONCEPT
The Grid Reservation algorithm is the concept using which a certain number of systems can be made as reserves dynamically based on their availability in the grid, whereby a reliable computing capacity is always retained by the grid. The concept can be further explained using the following case study below.

Case: Consider a grid system with 5000 systems connected to it. Every system is performing its own immutable tasks and provides output to the server for further calculation based on the results obtained from other systems. If a certain number of systems, say 300, suddenly drops out of the grid the entire performance of the grid is impacted. The scheduler needs to reroute the tasks which were currently handled by those 300 systems to the existing systems in the grid once they have completed their tasks. Consider the performance graph in such a scenario.

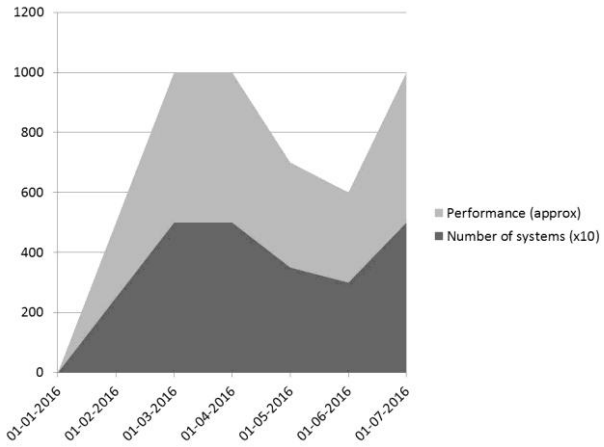

Figure 1.Relation between performance & number of systems

For this example, the performance factor is calculated as double that of the number of systems. In reality, the throughput is much higher than what is depicted in the graph. When the entire set of systems getting connected to the grid is taken for computation at that moment itself, the performance becomes highly volatile. That is, the continuous addition and drop of systems in the grid makes the grid unstable.

The inference from the graph is that: when the number of systems are either added up or removed at a particular point of time, the performance of the grid is varied drastically. This problem could be effectively solved by implementing the grid reservation algorithm. However there are considerable amount of challenges that needs to be addressed.

Let's have a look at some of the important challenges that is faced while solving the reliability problem.

2.1 Implementation Challenges
Berkeley Open Infrastructure for network computing viz. BOINC is an extensively used distributed computing environment. The working of this algorithm can be tested effectively if employed in such a well-established network. Any other distributed computing environment with large amount of computational resources can also be considered. But, before proceeding there are some initial challenges that need to be solved.

Minimum Calculation overhead

The purpose of distributed computing systems is to connect and leverage the computational capacities of geographically disconnected computing systems, to achieve a common goal. If any new computation is introduced to the existing system, the primary challenge is to reduce amount of calculation required. That is, more computation capacity cannot be spend on making the system work. The output is rather required to accomplish many other complex tasks. Hence only minimum computation should be used to obtain a dynamic reservation output.

Synchronization/Availability of systems

The next important measure under consideration is the availability of the systems. Every distributed computing system should be connected to the network. However, in some cases it is possible to download the part which needs to be executed by the system so that the can be done offline and the result can be updated when network becomes available. But, if a dynamic reservation algorithm is run on the system then a dynamic quantifiable value to uniquely identify the system is required, to be synched with the host so that the whole dynamic execution could work. If offline computing systems are present then the algorithm may not consider them for reservation.

Dynamic re shuffling of reserved systems

Similar to what is mentioned above, a dynamic re-shuffling of reserved system will happen as time progresses. This is based on a calculation which the algorithm would execute. However this might include many unforeseen factors which need to be considered before execution. Hence the complexity might increase making the dynamic reservation algorithm obsolete. The dynamic re-shuffling of systems, i.e. the reservation queue will not be having the same systems over a period of time. The systems will continuously enter and exit the queue based on a threshold.

Reservation factor calculation

The dynamic reservation algorithm mainly works based on a particular value. It is obtained by dividing the amount of time (in hours) the system was running on the grid and the total grid uptime. It requires least calculation and is stored in the central system where the recalculation happens every 15 minutes. It is easier to assign and release the systems in the reservation queue by using this technique. This is another important aspect of implementing this algorithm.

Once the above mentioned challenges are addressed and resolved, the reservation should work properly. The next question that needs to be addressed is "what is the advantage of implementing such a system?"

2.2 Advantages

For any distributed computing system, performance is the key factor. The implementation of a reservation algorithm enhances the reliability and the overall performance of a distributed network.

1. Performance factor increase

The performance of a distributed computing environment is mainly monitored using three aspects viz. throughput, scalability and reliability. By implementing a reservation algorithm in the existing grid environment, the reliability factor is enhanced considerably. This in turn increases the overall performance of the system.

As the reservation algorithm comes into play, the grid performance graph no longer shows varying spikes. Instead a considerably stable performance output would be obtained.

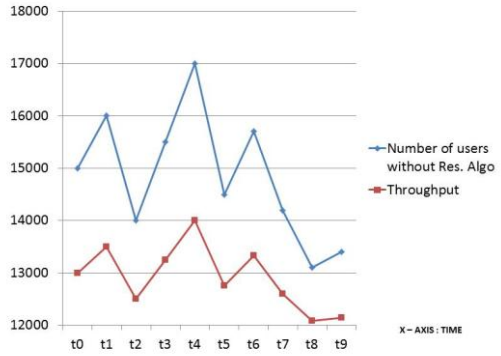

Figure 2. Throughput spikes with change in number of users

Note: The values used in the graph is absolute

The above figure represents how a normal distributed computing network performs. As the number of users in the network shows variation, the corresponding variation is seen in the throughput of the system. In turn, the overall network seems to be highly unstable.

However, once the reservation algorithm is employed in the grid, it automatically queues the system into the reservation in such a way that more active systems form the buffer and this buffer is always dynamically scheduled. This dynamic scheduling ensures that no system stays in the reservation pool for more than a particular amount of time. Either the system exit the reservation queue when the required threshold is reached or when the system goes below the current reservation factor. The below graph depicts the distributed system after the reservation algorithm is employed.

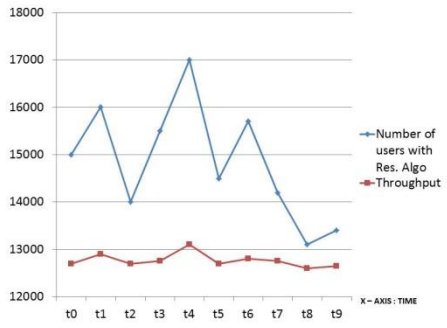

Figure 3. Stable throughput with change in number of users

Note: The values used in the graph is absolute

From the graph the important inference is, whenever there is a large amount of user drop out or user drop in from/to the distributed network there is only minimal change in the throughput. Thus exponential increase or decrease of throughput, with change in number of users is avoided. This

increases the reliability of the grid which ultimately improves the grid performance.

2. Reliability Enhancement

The reliability of the whole distributed network could be enhanced once the reservation algorithm comes into play. The overall performance of the grid is never put into its full power. Always, a minimum percentage of systems are kept in standby, which is shuffled dynamically, and could be used as immediate replacement for a number of systems exiting the grid. The result is a stable throughput which is hard to achieve in a distributed computing environment. Reliability is one of the major factors affecting a distributed networking system, If the reliability factor of the system could be enhanced it will improve the overall performance of the distributed system.

The following graph depicts the reliability enhancement which could be accomplished if the grid reservation algorithm is deployed.

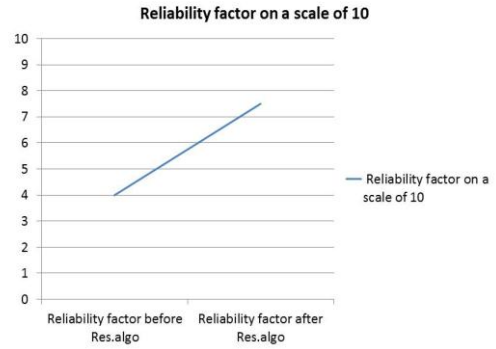

Figure 4. Reliability factor change - on a scale of 10

3. Stable throughput

Throughput is vital for all systems. It is a measure of how many operations could the system perform with in particular amount of time. By implementing the reservation algorithm in distributed systems, a stable throughput is achieved. The system is never powered to its complete extent and the performance doesn't show huge variations with drop out of a limited number of systems from the grid. The stable throughput factor is achieved at the expense of never completely utilizing the maximum computational potential of the grid. However where stability matters over complete utilization the reservation algorithm could be used. Figure 3 represents how reservation algorithm provided a stable throughput to the distributed systems when there is a substantial amount of change in the total number of systems connected to the grid.

It should be noted that, if a huge change in the number of systems occurs at a particular point in time, then the reservation algorithm becomes obsolete.

4. Ensures continuity of operations if a region outage occurs

Assume that a distributed network consists of 15000 nodes connected to it. Now, based on geographical region, 3000 nodes present in the network are from Asia. 5000 nodes are from United States, another 4000 from Europe, 1000 from Africa and the rest distributed across the globe. Suppose a regional outage occurs in the African region of the distributed network. The result would be a sudden drop of 1000 systems in the grid which in turn results in a rapid drop in the computational power of the systems. Most of the tasks being performed by these systems would be incomplete and the intermediate checkpoints results would be available in the central system.

Such a case could be effectively handled by the reservation algorithm if deployed in a carefully monitored scenario. The reservation algorithm always keeps on shuffling the systems in reserve whereby it never sets a system completely to reserve over a long period of time. If the reservation algorithm is used based on region and the systems in the US region is set to be considered for reservation, then the drop out of systems in the African geography could be substituted by systems from US region coming online from the reserve. However, if a drop of Asian region occurs, the case would be different. Since the number of systems supporting the distributed network from Asia is high, the reserve system cannot completely nullify the effect of a huge dropout.

If employed based on insight into the network the reservation algorithm could make the grid a continuous high performing network.

3. ALGORITHM

The grid reservation algorithm is a novel approach whereby the overall performance of a distributed computing environment could be increased and made stable. The algorithm is defined below.

Algorithm:

1. Central System monitors whether the specified number of systems are available in the grid (Say variable s). The variable s varies in a range of integer values.
2. Once the central system gets confirmation that the grid has crossed the threshold of variable s, the system checks for a predefined reservation percentage for the project viz. variable Tp, which varies from 0 to 25 and is a percentage value.
3. Based on Tp value, calculate the number of systems to be considered for reservation, viz. Ns, which is also an integer value. Ns = (Total number of systems in the grid / 100) * Tp
4. Reservation factor, viz. Rf is calculated for all systems connected in the distributed network. Rf is a fractional value obtained on dividing the amount of system uptime to that of the total grid up time. Ie, Rf = Total system uptime in grid (mins/hours) / Total grid uptime (mins/hours).

   Note: Maximum Rf value = 1. It varies as follows 1<= Rf <= 0 . The Rf value is calculated every 15 mins, as a new broadcast should be added to the client side which updates the values to the central system.
5. Reservation counter, Rc, is a counter which is started once the system goes into reservation. No system remains in reserved state for more than 30 mins as once the reservation counter is activated that system has a cool off timer Ct, which prevents the system from going to reserve again, even if it meets the Rf value set by the central system
6. Once all Rf values are calculated the central system can set an Rf threshold, viz. Rft values which varies between 1 & 0. The systems with Rf value above Rft is directly assigned to the reservation pool till

the total number of systems in reservation pool is equal to variable Ns.

7. When a system is in reservation pool the uptime won't be increased as the reservation counter is active. After the reservation counter expires the Rf value is calculated which remains to that of the last known Rf value. The Cool off timer is activated and till its end no more Rf factor updation is done for that system.

8. The Reservation factor threshold, Rft, is also varied based on the average of all Rf values in the grid. Ie, there is a dynamic changing of Rft based on the below calculation, but initially the value could either be set from the central system or it could leverage the below formula from the beginning itself

   Rft = Sum of all Rf values / Total number of systems (s)

The reservation algorithm is thus simple, dynamic and easy to employ in any grid system with least modifications to be done to the code architecture. However the choice of implementation is completely dependent on the application behavior. I.e. a system where continuous availability and reliability is more important over maximum throughput the reservation algorithm could be useful to a great extent.

3.1 Implementation Feasibility, Considering Existing System

Grid Reservation Algorithm could be easily integrated to existing system with minor changes to the existing code base. The primary concern while creating this algorithm was its integration to existing systems. Only a minimum overhead calculation is introduced and very limited additional usage of network. The result is a highly stable output rather than a continuously varying output. The following are the major factors considered while developing the Grid reservation algorithm.

1. Complexity

Providing a new feature at the expense of valuable resources is never advised in a distributed network. Hence the complexity of the algorithm is kept to a minimum. The algorithm only leverages basic operations such as addition, division & multiplication of mainly integer values and some floating point values. However, the complexities of these operations are very minimal and don't affect the overall system performance.

2. Minimal Network Usage

The resources in a distributed network uses internet for calculations, control as well as result updates. So when implementing a new approach in distributed computing space care should be taken to minimize the network usage of the algorithm. The Grid Reservation Algorithm uses very minimal amount of network for its functioning. It is one of the novel features of this algorithm.

3. Minimal Resource Usage

For effectively employing any algorithm to distributed networks another major factor is resource usage. Suppose the total grid computing capacity is 10 FLOPS and the algorithm needs 1 FLOPS to run. Implementing such a system is only an overhead and drastically reduces the performance of the grid. But here the amount of computation and storage required for the algorithm to effectively run is very minimal whereby making it a suitable candidate for adoption.

So the GRA could be introduced and run on the existing distributed network because of its ease of use and very low complexity as well as resource utilization.

3.2 Further Enhancement

The algorithm presented in this paper is least complex and an easy to employ method. But, further enhancements could be made to this algorithm with a goal to increase the throughput and making the algorithm even more agile. As mentioned previously, the GRA is a highly agile algorithm wherein it ensures that no system stays in the reservation pool forever, at the same time provides a considerable reliability enhancement to the system.

However, the implementation of GRA restricts the grid from full utilization of its resources at any point of time. This reduces the throughput of the system. Since the GRA would be employed in projects which require availability over complete utility, it is of less concern. The throughput reduction is an area which could be improved, as it is possible to handle the throughput difference in an effective way, along with GRA feature. Moreover the possible migration of GRA to other distributed networks and the impact it would cause is also a new area of study.

3.3 Implementation Scope In Cloud Systems

Cloud computing is a new method through which users can access a shared pool of computing resources, viz. servers, storage, applications, services etc. through a connected network. Cloud systems are an extension of the distributed computing systems. Many methodologies and performance factors which is key to distributed networks, remain the same in cloud computing. Hence any new concept applicable to distributed systems could be a part of cloud as well.

In cloud systems, unlike volunteer computing grid, large systems and servers are available with the key focus on cheap availability of resources. Primary aim of cloud systems is to bring in a new business model, wherein the users pay as they use (on demand). Previously, what used to exist was a system where the user has to pay a fixed amount for storage space/computation which is allocated to him/her. It doesn't matter whether the user is always using that or not, the payment is done for the whole. But in the cloud paradigm, the users only have to pay for what he/she uses. It is highly scalable and easy to use. The performance factors of cloud systems are of huge importance.

Cloud is derived from distributed computing. Hence, GRA will have a very active role in cloud systems as well. Any distributed computing network is prone to failure due to network issues. In cloud as well the same threat exists. So, if an enterprise requires availability over max throughput, the GRA could be used in such a cloud system, where the end users don't see the switching of resources when something goes wrong. Instead the users have a reliable experience while using the resources through cloud. Availability is a key performance factor of the cloud systems. Since the GRA only enhances the same its application could also be found in cloud systems. As the availability of resources in cloud increases its performance also increases whereby it helps to improve the user satisfaction while using cloud services and helps in user retention for cloud services.

The larger possibilities and challenges when integrating GRA to cloud should be discussed in another paper. Hence only the possibility that GRA could be used in cloud is explored and being discussed in this paper.

4. CONCLUSION

Distributed computing network occupies a large part in our daily life. From search engines to services to applications, everything is using one or the other form of distributed networks. As more and more systems are coming online, the possibilities to tap into the humungous potential of distributed systems remains a challenge. A lot of research is already being conducted in this field. However, its evolution is making it all the more a hot spot for new concepts as well as paradigms.

The performance of distributed systems is a key characteristic which defines its quality. The performance is mainly attributed by throughput, availability & reliability. If it is possible to enhance any one of these three, then a considerable increase in the performance factor could be obtained. GRA is an algorithm which enhances the availability and reliability of the distributed network at a minor expense of throughput. Its application would be mainly in areas where reliability is important over 100% throughput. Time critical applications are a brilliant example where availability matters. Most Enterprise systems as well require availability over throughput. The GRA helps to improve the reliability to a considerable extent. However large fluctuations in the connected systems (likely beyond 25% of the total) cannot be effectively tackled with GRA. However, limited changes could be tackled and a continuous reliable performance could be provided if GRA is used in distributed networks.

The distributed computing space still has a lot to explore and remains a hotspot for researchers around the globe.

5. ACKNOWLEDGMENTS

My thanks to the experts who have contributed towards development of the template.

6. REFERENCES

[1] Fangpeng Dong and Selim G. Akl *Scheduling Algorithms for Grid Computing:State of the Art and Open Problems* School of Computing, Queen's University, Kingston, Ontario, January 2006

[2] Manjeet Singh1, Shourabh Sholliya2, Palak Gupta3, *Scheduling in Grid Computing – a Review*, Department of Computer Engineering, 1NIT Kurukshetra, 2Thapar University Patiala, 3MIET Meerut 1Haryana, 2Punjab, 3U.P. INDIA

[3] Anthony Sulistio and Rajkumar Buyya, A GRID *SIMULATION INFRASTRUCTURE SUPPORTING ADVANCE RESERVATION*, GRIDS Laboratory and NICTA Victoria Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia, ICT Building, 111 Barry Street, Carlton, VIC 3053

[4] Randeep Kaur1 ,Supriya Kinger2, *Analysis of Security Algorithms in Cloud Computing*, 1Student Masters Of Technology, Shri Guru Granth Sahib World University, Fatehgarh Sahib. 2Assistant Professor, Shri Guru Granth Sahib World University, Fatehgarh Sahib.

[5] Marish Kr. Singla, *Task Scheduling Algorithms for Grid Computing with Static Jobs*: A Review, M.E. Scholar, Dept. of CSE, NITTTR, Chandigarh

[6] Atsuko Takefusa1, Hidemoto Nakada1, Tomohiro Kudoh1, and Yoshio Tanaka1, *An Advance Reservation-based Co-Allocation Algorithm for Distributed Computers and Network Bandwidth on QoS-guaranteed Grids*, National Institute of Advanced Industrial Science and Technology (AIST)

[7] N. Krishnamoorthy1 and R. Asokan2, *Optimized Resource Selection to Promote Grid Scheduling Using Hill Climbing Algorithm*, 1Department of Computer Science and Engineering, Kongu Engineering College, Erode, India. 2Kongu Nadu College of Engineering and Technology, Thottiyam, Trichy, India

[8] Daniel Funke, Fabian Brosig and Michael Faber, *Towards Truthful Resource Reservation in Cloud Computing*, Karlsruhe Institute of Technology, Karlsruhe, Germany

[9] Nabeel Zanoon, Ph.D, Nashat Al Bdour, Ph.D, Evon Abu-Taieh, Ph.D, *Survey of Algorithm: Scheduling Systems and Distributed Resource Management in Grid*, 1.Al- Balqa' Applied University (BAU), Department of Applied Science, Aqaba-Jordan, 2.TafilaTechnical University Tafila - Jordan 66110, 3.Faculty of Computer Information Systems, Jordan University -Aqaba, Aqaba-Jordan

[10] J.SRINIVAS1, K.VENKATA SUBBA REDDY2, Dr.A.MOIZ QYSER3, *CLOUD COMPUTING BASICS*, Assistant Professor, Dept. of CSE, M. J College of Engg & Tech, Hyderabad, India, Assistant Professor, Dept. of CSE, M. J College of Engg & Tech, Hyderabad, India, Professor, Dept. of IT, M. J College of Engg & Tech, Hyderabad, India

[11] Charu Sharma, Tanu, *Dynamic Resource Allocation in Grid Computing*, Department of Computer Science & Engineering, Punjab Technical University, Jalandhar, India

[12] Wikipedia, the free encyclopedia, https://en.wikipedia.org/wiki/Distributed_computing

https://en.wikipedia.org/wiki/Grid_computing