Azerbaijan Journal of High Performance Computing, Vol. 2, Issue 2, 2019, pp. 178-182
https://doi.org/10.32010/26166127.2019.2.2.178.182

The Oscillation Model of Load Flow of Global Activities in a Fully Distributed Exascale System

Ulphat Bakhishov

Azerbaijan State Oil and Industry University, Baku, Azerbaijan, ulfet_bakhishoff@hotmil.com

*Correspondence: Ulphat Bakhishov, Azerbaijan State Oil and Industry University, Baku, Azerbaijan, ulfet_bakhishoff@hotmil.com

Abstract

Exascale systems are the concept of HPC systems that able to perform one exaflop (10^{18} floating-point operations) per second in a dynamic and interactive environment. As the traditional HPC systems, the major challenge of this system is load balancing. Providing load balancing in dynamic and interactive nature requires a model which handles dynamic and interactive events and allows to manage load distribution over the system. It strongly depends on the distribution degree of the system. This paper defined a new model of load flow while imbalance occurred in the node of a fully distributed Exascale system.

Keyword: Distributed Exascale Computing system, Load Balancing, Dynamic and Interactive Nature, Load distribution model

1. Introduction

The main difference between Exascale systems from traditional HPC systems is handling natural events in a dynamic and interactive environment (Mirtaheri & Grandinetti, 2017). The dynamic and interactive environment is an environment of Exascale applications such as human brain simulation, space weather simulation, computational fluid engineering etc. (Eicker), where while designing the system, it is impossible to design a model for all possible events and their requirements.

On the other hand, as said in (Dongarra et al., 2011), “The first challenge at such a large scale is to provide efficient, scalable, resilient, and transparent access to the external (concerning the data center) and distributed (from a geographical point of view) data repositories.” That is why the system should manage resources in runtime while changing resource attributes and process requirements. For resource management architecture, the system can be centralized, hierarchical or decentralized (Wang, Brandstatter, & Raicu, 2013). A centralized workload management architecture is inefficient for large-scaled and geographically distributed systems (Khaneghah et al., 2018; (Wang et al., 2013). On the other hand, collecting or sharing the actual state of the system and particular nodes is needed for fully distributed systems as there are not any central repository for this type of information (Sharifi, Mirtaheri, & Khaneghah, 2010).

For overcoming imbalance that occurred in a fully distributed Exascale system, either each node should be informed about the current status of the system, or each node should resolve imbalance that occurred in it in such a way that this operation should not create an extra imbalance in another node.

In the first case, for informing nodes with actual information about the current status of the system, there is a necessity for additional resources and time. Taking into account that the dynamic and interactive event may have occurred in Exascale while this time, the information shared around the nodes loses its actuality. According to this, it is necessary to re-share the information about the current status of the system is needed. Solving this problem requires an additional resource, which is practically impossible to perform.

In the second case, it should be considered that the system may be built from geographically dispersed nodes that are connected by the network with different topologies and containing resources in a different architecture. In this situation, if each node resolves only is own imbalance with a custom way without considering others, it may increase the dynamicity of events in the Exascale system. However, if each node resolves imbalance that occurred in it, in the same way as others, it may be possible to model the load flow of global activities and optimize this model for minimizing response time.

2. Related Works

2.1. Load distribution strategies

For the different load balancing policies, a load of the global activity may be assigned to a single node time – one-time policy or it may be reassigned (Mirtaheri & Grandinetti, 2017; Bakhishoff, 2018; Sharma et al., 2008). In a one-time policy, the task should be solved only by the node which is assigned. However, if reassignment is available, a process may be reassigned to another node until it executed. As a process, it is considered the smallest collection of commands which cannot be parallelized.

Load distribution strategies are categorized into sender-initiated and receiver-initiated strategies (Mirtaheri & Grandinetti, 2017). In a sender-initiated strategy, imbalance occurred node raises global activities (Balasangameshwara & Raju, 2013). However, in receiver-initiated strategy lightly loaded nodes look for overloaded nodes for receiving the load to process (Fairuzullah et al., 2019; Domanal & Reddy, 2015, November; (Kumar & Kumar, 2019). In this strategy, nodes should be informed about other nodes' current status or it should check them one by one if they are overloaded. For this reason, this strategy considered significantly cost in huge distributed systems (Balasangameshwara & Raju, 2013; Fairuzullah et al., 2019).

2.2. Resource management models

For the resource management model for P2P systems proposed by Wang et al. (2013), the system does not need any exact centroids. However, it needs dynamic controllers and regions created at runtime, which provide hierarchical workload management. This model based on the supply and demand model, so the same node can be in role client and server at the same time from different aspects. As the server, this machine stores and shares the current state of the system. For this purpose, they use resources which specially allocated for this reason, called Oasis. The main shortcoming of this model is sharing a large amount of metadata over the nodes which need to be kept actual.

Another approach proposed diffusive load balancing algorithms by (Lieber, 2016, September) - these algorithms based on a graph model. The main advantage of these models versus the model proposed by Wang et al. (2013) is that they require communication between neighbor nodes only. It applies to scalable systems. However, there need rules to recalculate coefficients while dynamic and interactive events occurred in Exascale systems.

3. Proposed model

This paper proposed a sender initiated and reassignment available load flow model for a fully distributed Exascale system. It is considered that each node has information only about its neighbors. If node overloaded, it distributes the extra load between its neighbors equally. In a simple case, it is considered a system that has two nodes. If one node overloaded, all extra load of it will be transferred to its neighbor. At the same time, if the neighbor node would not be able to process these global activities, it will return all extra loads. In that situation, a harmonic oscillation occurs. The time of returning sent load to back is the period of this oscillation. This period depends on network bandwidth and a load of global activities. If processing power is considered, the current load of the node will decrease over time and it will be able to process a piece of the extra load that is sent to it. In this case, the extra load will disappear over time and this situation is damped oscillation. The damping coefficient of this oscillation depends on the processing powers of nodes in communication. If this rule considered in complex structured systems, each node would distribute global activities currently in it between neighbors equally. At the same time, this process can happen in each neighbor if it is overloaded. However, global activities assigned to the underloaded machine will be processed and will not be reassigned. As a result of this, total global activities will be reduced. In other words, the oscillation will stop over time. This model indicates that flow of all global activities is directed to underloaded machines.

3.1. Considering dynamic and interactive events in the proposed model

In Exascale systems can be occurred following types of dynamic and interactive events (Eicker, N; Khaneghah & Sharifi, 2014):

• The process can fork a new process
• The process can communicate with another process
• The process can interact with the system environment

When a fork occurs, if the requirements of the new process can not be mapped to resource attributes, then it is marked as global activity and is assigned to one of the neighbors of the current node while distributing all global activities on the current node. This assignment starts oscillation. This oscillation can be critically damped if the assigned node can process it, or can continue several periods otherwise. However, over time, as the result of executions of processes in nodes, this oscillation will stop.

When one process communicates with another one, these are not able to executed parallelly. Taking into account that, a distribution made only for parallel processes, these communicated processes should consider as one process. In this case, if this process creates an extra load, it should be marked as a new global activity and should be resolved as global activities created during the fork.

When one process interacts with the environment, its requirements change unintentionally. In this case, it can be considered as a new process with new requirements that are forked, and the old process is finished. In this case, creation and handling global activities is as same as it occurred during the fork.

4. Conclusion

This model provides the flow of extra load directed to capable machines, without considering where they are created. However, the load may be transferred to the capable machine without an optimal direction. For example, if one node has five neighbors and only one of them is capable, each time load distributed between neighbors, a portion of it will reduce and another portion of it will return. This load will be distributed again and again until fully reduced. However, the total load is not assigned to the capable machine directly. That is why it should be controlled the direction of the flow by optimizing the model. At the same time, it should be calculated parameters of oscillation dependent on resource attributes and process requirements.

Another problem is fault tolerance. So, if when some machine goes down or moves from the system, the processes assigned to this machine and are currently executing in it should be marked as global activities and should not be lost.

References

Bakhishoff, U. (2018). Applying Multiple Multidimensional Knapsack Problem to Dynamic Load Balancing in Distributed Exascale computing environment. *Azerbaijan Journal of High Performance Computing, 1*(2), 214-218.

Balasangameshwara, J., & Raju, N. (2013). Performance-Driven Load Balancing with a Primary-Backup Approach for Computational Grids with Low Communication Cost and Replication Cost. *Ieee Transactions on Computers, 62*(5), 990-1003. doi:10.1109/tc.2012.44

Dharmik, R. C., & Sathe, S. R. (2018). A Sender Initiated Dynamic and Decentralized Load Balancing algorithm for Computational Grid Environment Using Variable CPU Usage. *International Journal of Applied Engineering Research, 13*(1), 189-194.

Domanal, S. G., & Reddy, G. R. M. (2015, November). Load balancing in cloud environment using a novel hybrid scheduling algorithm. In *2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)* (pp. 37-42). IEEE.

Dongarra, J., Beckman, P., Moore, T., Aerts, P., Aloisio, G., Andre, J. C., Yelick, K. (2011). The International Exascale Software Project roadmap. *International Journal of High Performance Computing Applications, 25*(1), 3-60. doi:10.1177/1094342010391989

Eicker, N. The DEEP Project.

Fairuzullah, A., Noraziah, A., Arshah, R. A., & Herawan, T. (2019). Optimize Performance Load Balancing Techniques Using Binary Vote Assignment Grid Quorum (BVAGQ): A Systematic Review. In *Proceedings of the International Conference on Data Engineering 2015 (DaEng-2015)* (pp. 31-39). Springer, Singapore.

Khaneghah, E. M., Mollasalehi, F., Aliev, A. R., Ismayilova, N., & Bakhishoff, U. (2018). Challenges of Load Balancing to Support Distributed Exascale Computing Environment. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)* (pp. 100-106). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Khaneghah, E. M., & Sharifi, M. (2014). AMRC: an algebraic model for reconfiguration of high performance cluster computing systems at runtime. *Journal of Supercomputing, 67*(1), 1-30. doi:10.1007/s11227-013-0982-z

Kumar, P., & Kumar, R. (2019). Issues and Challenges of Load Balancing Techniques in Cloud Computing: A Survey. *Acm Computing Surveys, 51*(6), 35. doi:10.1145/3281010

Lieber, M., Goebner, K., & Nagel, W. E. (2016, September). The potential of diffusive load balancing at large scale. In *Proceedings of the 23rd European MPI Users' Group Meeting* (pp. 154-157). ACM.

Mirtaheri, S. L., & Grandinetti, L. (2017). Dynamic load balancing in distributed exascale computing systems. *Cluster Computing-the Journal of Networks Software Tools and Applications,* 20(4), 3677-3689. doi:10.1007/s10586-017-0902-8

Rathore, N., & Chana, I. (2013, September). A sender initiate based hierarchical load balancing technique for grid using variable threshold value. In *2013 IEEE International Conference on Signal Processing, Computing and Control (ISPCC)* (pp. 1-6). IEEE.

Sharifi, M., Mirtaheri, S. L., & Khaneghah, E. M. (2010). A dynamic framework for integrated management of all types of resources in P2P systems. *Journal of Supercomputing, 52*(2), 149-170. doi:10.1007/s11227-009-0281-x

Sharma, S., Singh, S., & Sharma, M. (2008). Performance analysis of load balancing algorithms. *World Academy of Science, Engineering and Technology, 38*(3), 269-272.

Wang, K., Brandstatter, K., & Raicu, I. (2013). SimMatrix: SIMulator for MAny-Task computing execution fabRIc at eXascale. *High Performance Computing Symposium 2013 (Hpc 2013) - 2013 Spring Simulation Multi-Conference (Springsim'13), 45*(6), 66-74.

Submitted 07.07.2019
Accepted 30.11.2019