Defining Parameters for 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

Distributed exascale computing systems are the idea of the HPC systems, that capable to perform one exaflop operations per second in dynamic and interactive nature without central managers. In such environment, each node should manage its own load itself and it should be found the basic rules of load distribution for all nodes because of being able to optimize the load distribution without central managers. In this paper proposed oscillation model for load distribution in fully distributed exascale systems and defined some parameters for this model and mentioned about feature works.

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

1. *Introduction*

High-performance computing (HPC) systems designed based on distributed resources for performing computationally intensive operations (Pereira, E. P., Padoin, E. L., Medina, R. D., & Méhaut, J. F., 2020, July). The HPC systems have a module called a load balancer that serves to distribute the computing load over resources (Khaneghah, E. M., Mollasalehi, F., Aliev, A. R., Ismayilova, N., & Bakhishoff, U., 2018). The aim of this module is to set the resource attributes against the process requirements (Khaneghah, E. M., Mollasalehi, F., Aliev, A. R., Ismayilova, N., & Bakhishoff, U., 2018; Bakhishoff, U., Khaneghah, E. M., Aliev, A. R., & Showkatabadi, A. R., 2020). The HPC process is the smallest parallelizable part of whole application running on HPC system. The purpose of the load balancer is to create an optimal match between process requirements and resource attributes so that no process should be left in the queue and all resources should be loaded to the maximum (Khaneghah, E. M., Mollasalehi, F., Aliev, A. R., Ismayilova, N., & Bakhishoff, U., 2018; Bakhishoff, U., Khaneghah, E. M., Aliev, A. R., & Showkatabadi, A. R., 2020). This rule defined with the following formula:

$$LoadDistribution: \left[Process_{Requirement}\right] \rightarrow \left[Resource_{Space}\right] \tag{1}$$

Therefore

$$Best_{LoadDistribution} = \left[\begin{matrix} \left[\nexists Process \in HPC_{Process} \overset{So\ Means}{\because} \ni HPC_{ProcessScheduling}\right] \\ and \\ \left[Resource_{Activity} = 100\%\right] \end{matrix}\right] \tag{2}$$

Here the resource space is a union of set of resources of the nodes where the resource can be in following types: File, Memory, I/O, Processor. The process requirement is a requirement of HPC process for resources (Shahrabi, S., Mollasalehi, F., Aliev, A. R., & Mousavi, E., 2018). HPC process scheduling is the overall queue of HPC processes. So, the basic goal for load distribution is creating mapping between process requirements and resource attributes, and the best state of load distribution is that any HPC process should not be in HPC process schedule, and each active resource should be loaded 100%.

Traditional distributed computing systems are based on a pre-defined problem-technical accounting (PTA) (Ismayilova, N. T., 2020). In such systems, for achieving the goal defined in the equation (2), the load balancer configures and starts the system according to pre-defined process requirements.

However, it is not possible to prepare a PTA for the issues of the twenty-first century. Thus, dynamic, and interactive events can occur in an unforeseen manner during the runtime of the system. There are three types of dynamic and interactive processes (Khaneghah, E. M., & Sharifi, M., 2014):

One process during execution can create another process that is not planned previously;

Processes may be interconnected in an unforeseen manner. These processes can be any of the ones that were initially defined and created later;

Processes may interact with the system environment in an unforeseen manner. Thus, the resource requirements of the process may be changed.

In all three cases, there can be resource requirements that are not considered at the start of the system, which brings the load balancer to the unknown state. To avoid this problem, dynamic and interactive events should be considered in the equation (1).

$$LoadDistribution(DynamicInteractiveEvent):$$

$$\left[Process_{Requirement}|GlobalActivity_{condition}\right] \rightarrow \left[Resource_{Space}\right] \qquad (3)$$

So, the load distribution process depends on dynamic and interactive events, and it should create a mapping between global activities and resource attributes as well as mapping between process requirements and resource attributes. Here, the global activities are process requirements that locally cannot be fulfilled (Khaneghah, E. M., 2017).

For handling the global activity conditions, it needed to restart the load balancer, considering previously unforeseen resource requirements. However, in this case, it should "try" to change the response structure for unknown resource requirements. Thus, the load balancer should either find a new resource to meet the demand for an unknown resource with activating resource discovery module (Rezaei, S., Khaneghah, E. M., & Aliev, A. R., 2020; Khaneghah, E. M., Aliev, A. R., Bakhishoff, U., & Adibi, E., 2018; Adibi, E., & Khaneghah, E. M., 2018) and migrate the process to relevant resource (Sohrabi, Z., & Khaneghah, E. M., 2020), or replace the unknown resource requirement with known resource requirements (Khaneghah, E. M., Mollasalehi, F., Aliev, A. R., Ismayilova, N., & Bakhishoff, U., 2018; Khaneghah, E. M., & Sharifi, M., 2014).

2. Related works

In (Khaneghah, E. M., & Sharifi, M., 2014), the model based on vector algebra suggested for implementing load distribution in dynamic and interactive environment. For that model, there is a general state vector. Over the time this vector changes. For each executed process, if this vector changes, the load balancer tries to solve two problems: value disorder and direction disorder. Value disorder means the direction of the vector that defines the difference between two states of the system is same as direction of the general state vector, but values are different. If directions are different, it means the direction disorder has occurred. Solving these two problems allows to reconfigure a distributed exascale computing system during dynamic and interactive events in runtime. However, it is difficult to determine the state vector of the system and the new state vector after dynamic and interactive events occur.

In (Ismayilova, N., & Bakhishoff, U., 2018; Bakhishoff, U., Khaneghah, E. M., Aliev, A. R., & Showkatabadi, A. R., 2020) proposed a model based on discrete time Hidden Markov Model. For this model, the system should not consider a dynamic and interactive event itself, but consider system state instead, which the system reached after dynamic and interactive event has occurred. The load distribution function choses the best system state configuration for current state based on learned historical states of the system. This model does not suggest changing resource request, instead it suggests reconfiguring load distribution. Implementing this model, gives opportunity to learn possible states and find best configuration for new state after dynamic and interactive event has occurred. The main problem for implementing this model is that, the model needs to learn the system after each change of system state and need to store many historical data about each system state changes.

In (Bakhishov, U., 2019), the oscillation concept of load distribution is defined. For this concept each global activity should move over the nodes of the system until reaching capable node for that process. But the parameters of oscillation are not defined based on process requirements and resource attributes. Also, the optimization is needed for movement of process over the nodes of the system.

3. Proposed model

As seen in equation (3), global activities should be considered while load distribution as well as local process requirements. Global activities are raised based on a sender-initiated load distribution strategy (Mirtaheri, S. L., & Grandinetti, L., 2017) by nodes that are in imbalanced (Bakhishov, U., 2019). The requirements of the global activities which are assigned to the node and local process requirements should be merged. If this set named as $Local_{ProcessScheduling}$, then the requirements of the global activities raised form current node should be excluded from $Local_{ProcessScheduling}$. If this excluded set named as $Global_{ProcessScheduling}$, it can be given the following definition:

$$HPC_{ProcessScheduling} = Local_{ProcessScheduling} \cup Global_{ProcessScheduling} \qquad (4)$$

In period the processes in global process scheduling should assign back to local machine. In traditional manner, the HPC system starts working with predefined load.

Let's assume that this system has single node which has no available resource. In this state all load in local process scheduling will move into global process scheduling. But, due to lack of another resource all processes in global process scheduling will move back to local process scheduling in a period (Fig. 1).

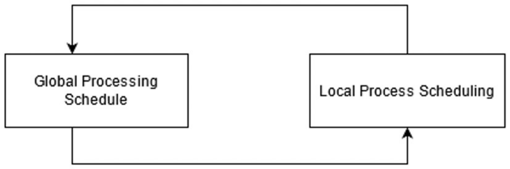

Fig. 1. Load flow process

Figure 1 defines load flow between local and global process scheduling. Because of there is no change in amount of load in period, the process will continue periodically. If the state with an empty local and global process scheduling assumed as initial state, replacement of the workload to opposite sides – local process scheduling and global process scheduling – around initial state, is the oscillation process. And if the amount of the workload does not change over the periods, this is harmonic oscillation process. The harmonic oscillation processes are defined with following formula:

$$\frac{\partial^2 w(t)}{\partial t^2} + \omega_0 w(t) = 0 \tag{5}$$

For the proposed model $w(t)$ is a function of workload of the node which does replacements around the initial state, and ω_0 is an angular frequency of oscillation and t is the time. The workload of the node is the function of the requirements of the processes, and it does not depend on the resource attributes of the node. The angular frequency of the oscillation depends on deadline of the global process scheduling for containing task in it.

$$\frac{\partial^2 w(Process_{Requirements}, t)}{\partial t^2} + \omega_0 w\left(Process_{Requirements}, t\right) = 0 \tag{6}$$

The process with rule of equation (6) will continue infinitely.

If the resources of the node are available then some piece of the workload of the node will reduced while period (Fig. 2), in other words, the oscillation is damped.

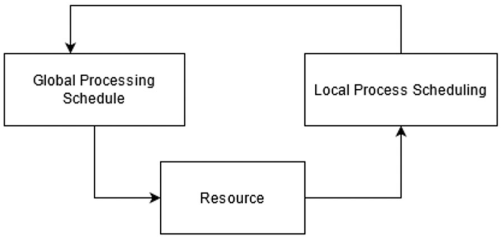

Fig. 2. Process of reducing the workload

The Figure 2. describes workload flow between local and global process scheduling, considering resources of the system. In that case, the resource reduces the workload of the machine. For that reason, it should be considered a damping factor in the equation (6).

$$\frac{\partial^2 w(Process_{requirements}, t)}{\partial t^2} + 2\beta\omega_0 \frac{\partial w(Process_{requirements}, t)}{\partial t} + \omega_0 w\left(Process_{requirements}, t\right) = 0 \tag{7}$$

Here β is a damping factor. The damping factor depends on both resource attributes that causes damping. The process with rule in equation (7) will stop over the time.

If the node has connection with another node, then another piece of load will be reduced by connected node over the time. But it should be considered that, it also can be accepted extra load from connected nodes. In this case the process should turn into a driven oscillation equation that contains another damping factor for reduction of load by the connected node and forcing factor for accepting extra load from the connected node.

In Distributed Exascale HPC systems the Dynamic and Interactive Events (D&I) may occur during runtime. These events create new load internally. But the specifications of these types of loads are their requirements which are completely or partial unknown to the node or not matching to the resource attributes of the node. In this case the node should try to approximate the requirements of the D&I to the resource attributes of the node. If it is possible with acceptable error, it should be accepted approximated loads, original requirements should be kept. These types of loads also should be considered in the forcing factor.

Discussion

The process of load distribution between the resources is a complex oscillation process. But the factors of this process and the dependence of these parameters on the characteristics of the system has not been defined. If these parameters and their dependencies are defined, it would be able to estimate that how long the oscillation process will continue with current load, and it would be possible to optimize oscillation process to finish it as soon as possible. So, if the periodic process will be able to change into an aperiodic process, the system would get into stable state in the minimum time. Because in that case the best resource for handling current global activity may be found with single migration.

References

Adibi, E., & Khaneghah, E. M. (2018). Challenges of resource discovery to support distributed exascale computing environment. *Azerbaijan Journal of High Performance Computing, 1*(2), 168-178.

Bakhishoff, U., Khaneghah, E. M., Aliev, A. R., & Showkatabadi, A. R. (2020). DTHMM ExaLB: discrete-time hidden Markov model for load balancing in distributed exascale computing environment. *Cogent Engineering, 7*(1), 1743404.

Bakhishov, U. (2019) The Oscillation Model of Load Flow of Global Activities in a Fully Distributed Exascale System. *Azerbaijan Journal of High Performance Computing, 2*(2), 178-182.

Ismayilova, N. T. (2020) Challenges of Using Different Mathematical Models for Load Balancing Optimization in Multi-Core Computing Systems. *Azerbaijan Journal of High Performance Computing, 3*(2), 190-195.

Khaneghah, E. M. (2017). *U.S. Patent No. 9,613,312.* Washington, DC: U.S. Patent and Trademark Office.

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

Khaneghah, E. M., Aliev, A. R., Bakhishoff, U., & Adibi, E. (2018). The influence of exascale on resource discovery and defining an indicator. *Azerbaijan Journal of High Performance Computing, 1*(1), 3-19.

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).

Mirtaheri, S. L., & Grandinetti, L. (2017). Dynamic load balancing in distributed exascale computing systems. *Cluster Computing, 20*(4), 3677-3689.

Pereira, E. P., Padoin, E. L., Medina, R. D., & Méhaut, J. F. (2020, July). Increasing the efficiency of Fog Nodes through of Priority-based Load Balancing. In *2020 IEEE Symposium on Computers and Communications (ISCC)* (pp. 1-6). IEEE.

Rezaei, S., Khaneghah, E. M., & Aliev, A. R. (2020) Challenges of Influence Dynamic and Interactive Events on Resource Discovery Functionality outside of Distributed Exascale Systems. *Azerbaijan Journal of High Performance Computing, 3*(2), 164-180.

Shahrabi, S., Mollasalehi, F., Aliev, A. R., & Mousavi, E. (2018) Load Balancing in Distributed Exascale Computing Based on Process Requirements. *Azerbaijan Journal of High Performance Computing, 1*(2), 158-167.

Sohrabi, Z., & Khaneghah, E. M. (2020) Challenges of Using Live Process Migration in Distributed Exascale Systems. *Azerbaijan Journal of High Performance Computing, 3*(2), 151-163.

Submitted: 10.10.2020
Accepted: 31.05.2021