Azerbaijan Journal of High Performance Computing, Vol 1, Issue 2, 2018, pp. 214-218
https://doi.org/10.32010/26166127.2018.1.2.214.218

Applying Multiple Multidimensional Knapsack Problem to Dynamic Load Balancing in Distributed Exascale computing environment

Ulphat Bakhishoff

Department of General and Applied Mathematics, Azerbaijan State Oil and Industry University, Baku, Azerbaijan, ulfat.baxıshov@asoiu.edu.az

*Correspondence: Ulphat Bakhishoff, Department of General and Applied Mathematics, Azerbaijan State Oil and Industry University, Baku, Azerbaijan, ulfat.baxıshov@asoiu.edu.az

Abstract

Dynamic and Interactive nature of the processes in the Distributed Exascale computing system requires the system to be able to make Load Balancing in runtime. In this paper proposed applying Multiple Multidimensional Knapsack Problem for overcome imbalance at time of occurrence of the dynamic and interactive event at Distributed Exascale computing environment.

Keywords: distributed exascale computing, load balancing, dynamic and interactive event, multiple knapsack problem.

1. Introduction

Exascale computing systems are the type of distributed systems which can perform at least one exaflop operation per second [1] in dynamic and interactive nature [2]. In the dynamic and interactive nature, both processes requirements and resource attributes are dynamically changeable [2]. In this case, it is impossible to map the workload of the processes to the resources at design time. On the other hand, it can occur dynamic and interactive events which can be handled only runtime. That's why Load balancer of the system have to be run continuously and should handle imbalances at runtime. In the distributed system, each node has their operating system, and they manage their resources themselves. However, in Exascale systems when dynamic and interactive events occurred, it is possible the local machine cannot process some actions. This results in an imbalance in system, and these actions are marked as global activities [3, 4]. The load balancer of the system should assign these activities to another machine can do remain part. In centralized systems, central machine collects information from other machines and makes load balancing [5, 6]. However, in such systems, there is an excellent dependence from a central machine. Considering dynamicity of resources in Exascale systems, if resources of the central machine are changed or central machine is stopped completely, then it needed to get information about the current state of the system, creating new central machine and migrating central processes to the new central machine.

However, in P2P systems, there is no dependence from any central machine. In this case, each machine should be informed enough about other machines for deciding the new configurations.

2. Related works

In [7], the systems state vector is used to determine if the system is in a stable state or not. If the current state vector of the system is different from the stable state vector, HPCS

manager tries to bring the system to the old known stable state or waits for the system to reach to the new stable state. It is activated while the length or angle difference occurred between the current state vector and the known stable state vector of the system.

In [9], proposed load balancing model based on supply and demand model for four types of resources – File, Memory, I/O, and CPU. For this model, each machine independently tries solving imbalance, working in supply or demand mode. However, deciding on choosing the mode, it needed to gain information about all active nodes [9] in the system.

In [2], machines in the system, are negative and positive loaded where positively loaded machines are machines which getting loaded for this model, the state of each machine evaluated from perspectives of other machines. The positively loaded machines use status of other machines from the perspective of itself for distributing extra loads. It supplies opportunity making load balancing in each machine in a fully distributed system without being dependent any central machine. However, this model requires calculated load value. Considering each process may require multiple types of resource and additionally in Exascale system resource attributes are dynamically changeable, it needed to make a rule for calculating the overall load of a system based on a load of each type of the resource of the system.

3. The load balancing model

Let us consider that there has N machine in the distributed system. Each machine in the distributed system has C_i capability where $i = \overline{1,N}$ is the index of the machine. On the other hand, consider that there M process in the queue and each process has R_j requirement and P_j profit, where $j = \overline{1,M}$ is the index of the process. At this time considered one-time-policy for processes, so, each process can be executed only one time [2]. However, in dynamic load balancing techniques, it is possible reassignment [8]. Taking into account that, for the paper the task is finalizing work in minimal time, using maximally available resource [2], the profit of the process can be the execution time (i.e., CPU call count) saving in respect of the maximal process execution time, and it can be calculated as following:

$$P_j = \max_{1 \le k \le M} T_k - T_j, \qquad j = \overline{1,M} \tag{1}$$

Here T_k is the execution time of k^{th} process. So, which process finishes earlier it is more valuable.

Considering this the main task can be given as following:

$$maximize\ z = \sum_{i=1}^{N} \sum_{j=1}^{M} P_j X_{ij} \tag{2}$$

$$subject\ to \sum_{j=1}^{M} R_j X_{ij} \le C_i, \qquad i = \overline{1,N} \tag{3}$$

$$\sum_{i=1}^{N} X_{ij} \le 1, \qquad j = \overline{1,M} \tag{4}$$

$$X_{ij} = \begin{cases} 1, & if\ process\ j\ assigned\ to\ machine\ i \\ 0, & otherwise \end{cases} \tag{5}$$

Here z is a gained total profit and X_{ij} is the process assignment configuration that indicates which process assigned to which machine. The load balancer should solve this process to assign processes to resources. The task is the Dynamic Programming problem named 0-1 Multiple Knapsack Problem (0-1MKP) [9]. There are several polynomial time solutions [10, 11] for this problem. Considering that process requirement may be not only for one resource but also for multiple resources, the parameter R should be defined as multidimensional. In this case, the problem become 0-1 Multiple Multidimensional Knapsack Problem (0-1MMKP) [12].

This model applies to traditional systems. However, in Exascale systems, the load balancer should handle dynamic and interactive events too [2]. That's why it should be activated based on dynamic and interactive events.

3.1. Handling unintentional communications of processes with the system.

The processes defined here, are not processes, which can be executed in the local machine in given response time. So, in the paper, it's looked at the processes which cannot be executed in the local machine within given response time [3]. In this case occurs global activity [2, 3]. Therefore, the processes mentioned above are the total collections of global activities. In this case, the capability of each machine for global activities is dependent on the total capability of the machine and total requirements of internal processes.

$$C_i = C_i^{total} - \sum_{j \in internal} R_j\,, \qquad i = \overline{1,N} \tag{6}$$

So, the capabilities of the machines are dynamically changing. On the other hand, in Exascale systems, process requirements and resource attributes are dynamically changeable [7]. For this reason, the parameters C_i, C_i^{total}, and R_j from the model which described above, should be defined as $C_i(t)$, $C_i^{total}(t)$, and $R_j(t)$ properly. If the current effective resources [7] are not enough to finish the task in the given time limit, new resources have to be discovered [13]. It means that, after resource discovery, the resource count also changed.

For this reason, the machine count parameter N should also be defined a time-dependent, $N(t)$. Here t denotes the current time moment. The load balancer should be activated at this time moment and should calculate current capabilities of the machine with following instead of equation 6.

$$C_i(t) = C_i^{total}(t) - \sum_{j \in internal} R_j(t)\,, \qquad i = \overline{1,N(t)} \tag{7}$$

In a distributed system each machine can calculate its current capability with equation 7, and share results with others.

3.2. Handling unintentional forks or communications between processes

At the same time, considering a process being able to fork new processor able to communicate with another process in Exascale systems [14], the process count parameter M which described the above model, should also be defined as dependent from current time moment – $M(t)$. The new processes have their execution times. Considering this at equation 1:

$$P_j(t) = \max_{1 \le k \le M(t)} T_k - T_j, \qquad j = \overline{1, M(t)} \tag{8}$$

3.3. 0-1Multiple knapsack model for Exascale system.

Considering these conditions, the model described above, turn to the following form.

$$maximize\ z(t) = \sum_{i=1}^{N(t)} \sum_{j=1}^{M(t)} P_j(t) X_{ij} \tag{9}$$

$$subject\ to \sum_{j=1}^{M(t)} R_j(t) X_{ij} \le C_i(t), \qquad i = \overline{1, N(t)} \tag{10}$$

$$\sum_{i=1}^{N(t)} X_{ij} \le 1, \qquad j = \overline{1, M(t)} \tag{11}$$

$$X_{ij} = \begin{cases} 1, & if\ process\ j\ assigned\ to\ machine\ i \\ 0, & otherwise \end{cases} \tag{12}$$

In this model, the parameters $N(t), M(t), P_j(t), R_j(t), C_i(t), z(t)$ are time-dependent, but there is not any functional dependency between these parameters and current time moment. These parameters denote values at observation time.

Conclusion

The proposed model is applicable with state of the system at time t. This time moment is the moment of occurrence of the dynamic and interactive event of the Exascale system. This model solves challenge described in [4]. In distributed Exascale computing system equation 6 and equation 7 can be calculated in each machine and shared with others. However, equation 9 should be calculated in each loaded machine and should result from the same configuration X. The main problem is collecting information about all powerful machines and all global activities which occurred in the machine which isn't directly connected with the current machine. If the value of each calculated X configurations is same, it means all global activities until time moment t are handled, in other words, all dynamic and interactive events of Distributed Exascale system occurred until time moment t are handled. The ratio of the count of matching values of all configurations to the count of elements of the configuration matrix denotes success rate of the load balancing.

References

[1] Shalf, J., Dosanjh, S., & Morrison, J. (2010, June). Exascale computing technology challenges. In *International Conference on High Performance Computing for Computational Science* (pp. 1-25). Springer, Berlin, Heidelberg.

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

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

[4] Khaneghah, E. M., Mollasalehi, F., Aliev, A. R., Ismayilova, N., Bakhishoff, U.

(2018, July-August) Challenges of Load Balancing to Support Distributed Exascale Computing Environment. In *the 24th International Conference on Parallel and Distributed Processing Techniques & Applications, Parallel and Distributed Processing + HPC and Data Science* (pp. 100-106), Las Vegas, Nevada.

[5] Sinha, P., & Zoltners, A. A. (1979). The multiple-choice knapsack problem. *Operations Research, 27*(3), 503-515.

[6] Balachandar, S. R., & Kannan, K. (2008). A new polynomial time algorithm for 0–1 multiple knapsack problem based on dominant principles. *Applied Mathematics and Computation, 202*(1), 71-77.

[7] Chekuri, C., & Khanna, S. (2005). A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing, 35*(3), 713-728.

[8] Raja Balachandar, S., & Kannan, K. (2011). A Heuristic Algorithm for Resource Allocation/Reallocation Problem. *Journal of Applied Mathematics, 2011.*

[9] Sharifi, M., Mirtaheri, S. L., & Khaneghah, E. M. (2010). A dynamic framework for integrated management of all types of resources in P2P systems. *The Journal of Supercomputing, 52*(2), 149-170.

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

[11] 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.

[12] Khaneghah, E. M., ShowkatAbad, A. R., & Ghahroodi, R. N. (2018, February). Challenges of Process Migration to Support Distributed Exascale Computing Environment. In *Proceedings of the 2018 7th International Conference on Software and Computer Applications* (pp. 20-24). ACM.

Submitted 11.07.2018
Accepted 31.10.2018