A State of Art: Survey for Concurrent Computation and Clustering of Parallel Computing for Distributed Systems

Abstract

In this paper, several works have been presented related to clustering parallel computing for distributed systems. The trend of the paper is to focus on the strengths of previous works in this field towards enhancing the performance of distributed systems. This concentration was conducted by presenting several techniques, each of which has weak and strong features. The most challenging points for all techniques vary from increasing the performance of the system to time-responding to overcome overhead running of the system. For a more specific approach to addressing concurrent computation besides parallel computing classifications for distributed systems, this paper relies on a comprehensive feature study and comparison between SYNC and ASYNC modes.

Keywords: Distributed Computing, Distributed Systems, Clustering System, Parallel Systems.

I. INTRODUCTION

Nowadays, accessing to the Internet services continuously is important and vital for the most of people [1]–[3]. The distributed systems are separated design that makes the worst-case assumptions [4]–[6]. It is the combination of geographically separated and heterogeneous nodes that perform the applications [7], [8] . The challenging issues that facing a wide range of computing areas such as the social computation, web search and others require a big momentum to reach the convergence condition. This is due to the complex and diversity of the data structures and their sizes which needs a more computation iteratively [9]. Hence, many approaches have been proposed to handle with the computation of the large dataset. For instance, the machine learning techniques known Tensor Flow has been proposed [10]. This system operates in large scale in heterogeneous conditions and it is inherent or extending from the Distbelief system which has been utilised by Google since 2011 [11]. The feature of this system is using the parameter server architecture which has some limitation and it leads to propose the Tensor Flow system which uses dataflow graphs to reproduce the computation, shared state and the operations state. It connect several dataflow graph nodes in a cluster considering into account the several machines within the multiple computational machines such as the devices including the multicore unit processes. The algorithms used in this system contain iterative and conditional control as a result of using them within advanced machine learning systems for instance using it in a recurrent neutral network (RNN) [12]–[14] and in long short term memory (LSTM) [15], [16]. More detail has been given in Table II.

Considering the philosophy of think as vertex that proposed by Malewicz et al. [17] in which the coding graph computation is represented as a vertex centric programs. This process vertex in parallel form and the communication is obtained along edges. Typically many machine learning and data mining issues usually appear iterative computation nature by refining input data until a convergence condition is reached [18], [19]. The development of some iterative and convergences lead to introduce two execution modes called synchronous and synchronously, see section 2 for more detail. Increasing the number of potential workers within the parallel computing systems is considered as a recent advanced technique within the computing architectures networking which results to increase also the masses [20]. In particular, recently the two of the important technological development are ongoing related to the computing spectrum which are working independent and are different in terms the scale in which as small scale execution units (i.e. cores) at the point within CPU which can be taking into account as a parallel shared memory computer. On the other hands, at the large scale units, there several advantaging with using this scale such as in the Cloud computing paradigm in which the applications can use these large scale units for pay-as-you-go model [21], [22]. However, most of the works used the computer simulation software program to observe the scale and concluding the obstacles that facing the models before it goes in the practical application or in real life of application such as the Discrete Event Simulation (DES) that proposed and explained in detail by Law & Kelton in which the evolution of any considered model occur at the nodes in time by means of simulation events. Furthermore, the implementation of DES is obtained by using some state variables, global clock that represents the ongoing simulation time [23].

Parallel and distributed simulation (PADS) depends on the partition of the simulation models across the several execution units. Each of them is responsible for part of the model on the other words each PADS is dealing with its local event list hence it fully local. However, local generate events could require to be sending to remote execution units and this could lead to minimize of the run time cost which has a good impact in the efficiency of the model. Another positive impact of using this tool is the possibility of integrating simulators in geographically distributed which integrates a set of commercial off as a result of a single simulator of the composition of the different simulation models [24].

Scheduling tasks (algorithms) have an important role in improving the in enhancement of the performance of the distributed system because of minimizing the overall execution time and reducing the overhead problems such as the delaying of communication which is allocated a suitable task to redistributes the processor [25]. Two types of scheduling algorithm are introduced known static and dynamic scheduling. In general the static scheduling does not prefer to be implemented in distributed computing system due to the principle work of it in which the scheduling occurs before the application running which result of uncertainty [26], [27]. Hence, the dynamic scheduling is a good alternative with the distributed computing system. There are different scheduling techniques are employed for task scheduling some of them is presented in section 3.

The Grid Computing concept is used in the distributed system or cluster of workstations to enable the user task to be online at anytime and anywhere. But unfortunately this leads to raising the issue of uncertainty in scheduling process such as Google search service in which a large number of users over the world send their keywords queries to the Google servers and search engines utilise the MapReduce technique to split the requested queries into some specific groups or categories of tasks and then matching these tasks into servers for execution which result three types of uncertainties. The first type of uncertainty obtains due to the number of tasks. Because the time and which kind of search query user, that will be received by servers, are unknown. Second type is related to the duration of processing unit the convergence is obtained here the convergence is focusing to the evaluation of network delay. To solve this problem Markov Decision Process (MDP) is introduced which allocates the receiving task and execution pattern with free of uncertainty [28].

II. SYNC and ASYNC Modes Comparison

These modes are used for transmission synchronization in which the principle work of two modes is different in the performance in execution stages and also across different graph algorithms such as the SYNC mode is able to minimize the communication cost time and I/O bound algorithms via gathering the massages together. Whereas ASYNC mode the convergence needs less time and it favors CPU bound algorithms such as PageRank obtain much better with using ASYNC mode [29]. But Loopy Belief Propagation significantly executes better with ASYNC mode [30] and also it show a good performance in graph colouring in which with SYNC mode it is impossible to obtain the convergence [31]. AYSNC has a good respond with the starting and ending of Signal Source Shortest Path (SSSP) conversely to SYNC mode in which superior performance is happen throughout the middle of execution. This is attributed to convergence rate, computation and communication load in various execution steps. Taking into account the principle idea of graph parallel systems that divided the computation logic scheduling order in which both of them provide different visibility timing of update variables for subsequent of the computation vertex. In figure 1 the execution flow of SYNC is presented in which the execution of vertices in the sample graph are fixed in order with in every iterations and the boundary limitation between the consecutive iterations grantees that all vertex are updated within considering iteration and visible in the next iteration for all workers. Whereas in ASYNC the updating is not in order within the sample graph due to the lack of barrier. Therefore, they have different features as listed in Table I. On the other hand, the configuration such as clusters scale, the size of data and graph partition methods have impact to the efficiency of two modes.

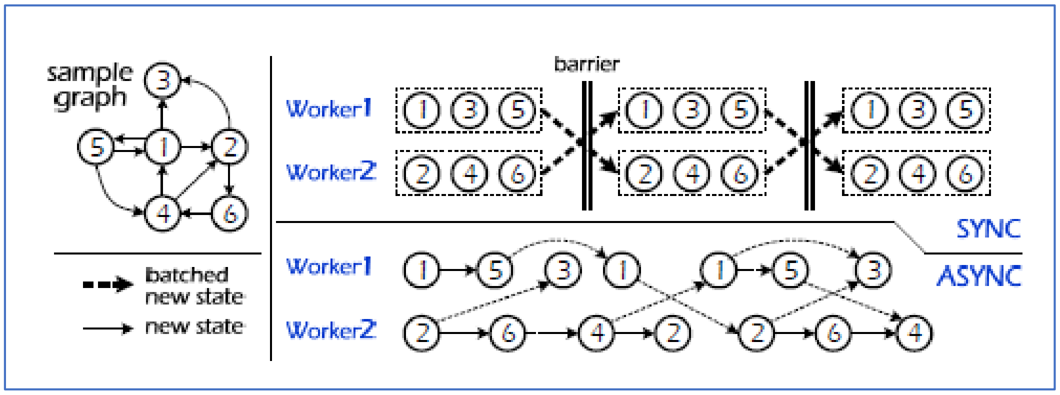

Fig. 1: The execution flows between SYNC and ASYNC modes.

TABLE I: THE COMPARISON BETWEEN SYNC AND ASYNC [5].

	SYNC	ASYNC
Features		
Communication	Regular	Irregular
Convergence	Slow	Fast
Favourites		
Algorithm	I/O intensive	CPU intensive
Execution Stage	High Workload	Low Workload
Scalability	Graph size	Cluster Size

III. SCHEDULING ALGORITHMS TECHNIQUES

The term scheduling can be defined as "A set of tasks such as **T** can be executed on **P** processors by some optimization criteria **C**" [32], [33]. The main purpose behind using the scheduling algorithms is to organize the various tasks to processors with targeting of improving the execution speed, minimising the runtime of tasks and reducing the communication delay and communication cost [34]. In general, the whole tasks in the distributed scheduling can be divided into sub-tasks which assign many processors. Hence they conducted work more quickly as compared with single processor and at the same time the scheduling algorithms of the pre-specified precedence is committed among various tasks [35]. From the above, the best scheduler should be applicable for general purposes such as it should be: a) Efficient (i.e. enhancing the work of the system and reducing overhead problems. b) Fair (i.e. maintaining load and then balancing it when scheduler has many tasks to be executed). Transparent (i.e. it means that the results is not affecting by local or remote site executions). Dynamic (i.e. it means that the scheduler will be classify as good if it responds to local changes and it avail from all resources that is available. The Scheduling techniques can be classified into two types: co-scheduling and local scheduling in which the second type contains the predictive which is easy to adopt new architectures that are capable of sharing the executions proportionally at a uniform rate. While the co-scheduling type has different principles that would be explained in the following section in details with presenting three type of it [35], as illustrated in figure2.

IV. LOCAL SCHEDULING

The local scheduling needs global information for increasing the performance of the system and so far many techniques have been developed such as proportional sharing and predictive schedules [36]. In wireless network, the local scheduling has proven a significant improvement or efficient compared with the traditional routine system in which the topology of wireless breaks down into several sub-graph and also the performing the end to end transmission of varies forwarded is obtained [37]. The important of the proportional sharing scheduling comes into play with incurred problems throughout the traditional priority- based schedulers which needs long time for allocating the processors. In the work of Regehr [38] the technique called Pessimism is introduced in proportional sharing to improve the performance and to remedy the error problem as well as meet the deadline of different real-time applications. An example of this type of scheduling is Stride Scheduling. It illustrates in fair manner the process of the allocation of jobs and how the resources are used up to single processor when many users have to execute their tasks. In this scheduling the hold numbers of tickets are released for all users. These numbers are in the proportion of resources and they have a time interval that is known as stride and they are inversely to allocation of tickets which aids to give a decision about how quick it comes in usable state [39]. Furthermore, the pass is connected with each user and a user holds a minimum pass is scheduled in the time interval with incrementing by job stride. The validation can be performed by two ways: one is implemented by prototypes for Linux Kernel and a other evaluation is performed by using the simulation.

There is an extension to Stride Scheduling which is not only used for I/O and intensive jobs, but is also can be used for CPU-bound jobs. The main idea behind this extension state that it is necessary to improve response time and through put rather than concentrating into resources for competing users. To obtain this, two approaches is proposed: one is working as credits and another is loan and borrow (i.e. many users have the exhausted tickets and if any user wants to withdraw from the system another user will take his place via his tickets, otherwise the ticket will be deactivated). While, in the credits, the system is an approximated and it is easy to be implemented due to not needing any type of overhead. Referring to the figure 2, there is another sub-division of local scheduling called predictive scheduling that supplies the adaptivity, intelligence and proactive kind of scheduling to the system. Consequently, it has the capability of good performing in any kind of condition. The feature of embedded in new kind of architectures is easy to be implemented and it can be categorised into three major components: a) allocator, b) S-Cell, and 3) H-Cell [40].

V. CO-SCHEDULING

In general terms, scheduling is utilized for scheduling the interactive activities for example all execution jobs are happing simultaneously and locally within the workstations [41]. The most well-known work was conducted by Frachtenberg and et al. [42]in which the flexible co-scheduling is proposed that address the issue of external and internal fragmentations. In this kind of scheduling, synchronization among processors plays an important key in its implementation and also in the requirement of load balancing. Proper working of resources and recovery all issues appeared and rising in multi-core system can be obtained via using the co-scheduling such as the work of Schönherr et al. [43]. Several challenging points that facing the co-scheduling algorithms that used for time sharing clusters are presented by Choi [44] in which the privilege of the genetic framework is used for identifications. There are three types of co-scheduling, as shown in figure 2, as follow: a) gang co-scheduling, in this type the jobs are making reference as gang and its member is considered as gang member. They allocate to class in which signal processor is able to sign through one gang member to execute it in parallel. It principle work is based to control all job members and assigns another job to that class when the timestamps is ended. This means the whole work of this co-scheduling is centralized control and this is one of its main drawbacks. This leads to bottleneck when the load is heavy. However, this kind of scheduling is improved its principle work by combining it with backfilling to overcome the aforementioned weak point, see the work of Zhang et al. [45]. b) The second type is called implicit co-scheduling which is also known as time-sharing communication process. It works fully local and the schedules are processed separately and it is making separate decisions instead of centralized policy as in the first type [36]. c) Dynamic co-scheduling is used to do the decisions when the arrival of massages and no need for explicit information to identify the process that requires co-scheduling. This type minimizes the response time up to 20 % compared to type (b) and it is more effective and robust [46].

Mohtajollah and Adibnia [47] proposed an algorithm for parallel job scheduling in cloud computing. They used tentative runs, workload consolidation and two-tier virtual machines architecture. Moreover, the performance improved and parallel jobs starvation was prevented by considering jobs deadline. The experimental results illustrated the introduced algorithm reduced waiting time and it could be utilized as an effective technique for scheduling parallel jobs in the cloud computing.

Xu et al. [48] optimized parallel jobs' scheduling performance in big data systems. They proposed machine learning algorithm based on k-mean clustering in heterogeneous clusters. The main purpose of the introduced method was to integrate various computing, storage, and network resources into

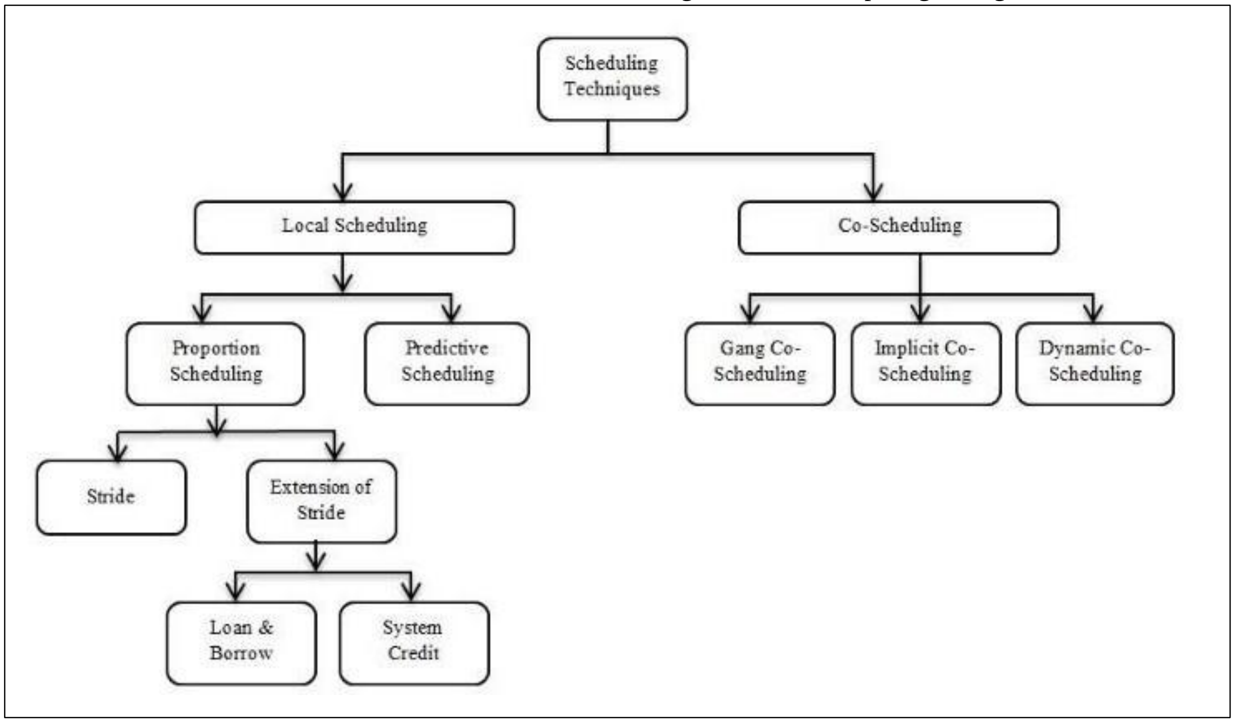

Fig. 2: The scheduling techniques [25].

scheduling strategy. The results showed that the presented algorithm enhanced execution time in single job scheduling and parallel job scheduling by 25% and 30%.

Lepakshi and Prashanth [49] proposed a method for task scheduling in a reliable way in cloud computing systems. The researchers considered earliest finish time and delays to assign a normalized score to the processor for scheduling tasks for a bounded number of heterogeneous virtual machines. The performance of the method improved by 100% in term of the operational availability and 22% for a sample graph in compared with other methods.

TABLE II: THE SUMMERY OF THE SELECTED OF THE UNDERTAKEN PAPERS IN THIS REPORT.

No.	Compared Reference	Strong points	Weak points
1.	Tyagi & Gupta, [16]	• Presenting some techniques of the scheduling that have a good impact of improving the performance of processors	• The single scheduling task is suffering from overhead running and also the execution time is higher than the multi scheduling task
2.	Martin Abdi & et. al, [6]	• Connecting many data flow graph nodes crossing into multiple machines in a cluster or within a single machine a crossing multiples computational devices. These devices can be represented by CPUs, GPU, and TPUs (Tensor Processing Units) which's known as custom designed ASICs. • TensorFlow has widely used in machine learning research and application in which several Google Services use this technique • The unified dataflow graph is used with the TensorFlow for representing both the computation and operations in algorithms	• The lack of default polices that work well for all users and all levels are still need to be determined. Hence more researches should be performed to overcome this gap and automatic optimization should be obtained with this model. • Transparent and efficient distribution of resources is faced this system even when the computation structure was unfold dynamically.
2.	Xie, et al., [5]	• The synchronous (SYNC) and Asynchronous (ASYNC) modes have different performances with different graph algorithms such as in cluster scales, input graphs and partitioning approaches which leads to obtain a hybrid constant gathering execution statistics directly which the prediction of future performance and determining could be profitable.	• There is still the luck of information on SYNC and ASYNC execution properties thus we have to manually select the mode.
4.	D'Angelo & Marzolla, [12]	• This parallel and distributed mechanism is capable to adapt with the application with preserving the same computing architecture most of the current approaches are unable of performing that aforementioned mechanism. • It works well in both setting in multicore process and cloud system. It reduces communication cost via removing the notes from the execution architecture in other words migrating the components of simulation	• The sows some fault tolerance with new type of software layer called GAGA (Generic Adaptive Interaction Architecture) which more details can be found in [37, 38] • The complexity level is obtained by hardware and it cannot be ignored and it should be carefully selected and availed by application level
5.	Tong, et al., [19]	• The proposed method has the capability of adaptivity for the arrival tasks without requirement of the knowing the prior knowledge about the task and it have the feature of auto save (i.e. dynamic robustness) of the task in which the system can respond and execute to the forthcoming tasks	• It shows the fair or average response time compared with some typical heuristic approaches
6.	Mohtajollah and Adibnia [47]	• The authors used several techniques to propose a new algorithm in order to improve job scheduling in cloud computing. • The make-span of the job scheduling was reduced and maximum waiting time.	• The researchers only depended on two metrics for measuring the performance of job scheduling. Also, the average waiting is not improved by the proposed algorithm. The comparison with related works in the research is not performed.
7.	Xu et al. [48]	• Task scheduling algorithm is prepared based on K-mean clustering. The performance of the single and parallel scheduling is enhanced by 25% and 30%. • The performance of the proposed machine learning algorithm is performed in three tests: simulation, virtual machine and real performance computing.	• The performance of the presented method is only based on the job scheduling time parameter. The comparison with previous works is not accomplished.
8.	Lepakshi and Prashanth [49]	• The researchers proposed a heuristic method for task scheduling in cloud computing. Numerous metrics were utilized for the performance of presented algorithm. The performance of the technique is compared with another algorithm.	• The proposed method obtained better results in dynamic cloud computing, However, the authors did not mentioned the algorithm performance real environment.

VI. DISCUSSION

Due to the increasing demand of using the internet in which the computation system area plays an important part of it, there are several issues facing that area such as social computation and web search which needs a huge effort to overcome them and obtaining the convergence condition. Hence, many works have been conducted for that purposes and in the Table II several works have presented illustrating the weak features (e.g. overhead running and the execution time) and strong features (e.g. improving the performance of processors) of each work. In this paper, various works has been presented related to the clustering parallel computing for distributed system. The most challenging points for all techniques vary from increasing the performance of the system to time responding to overcome overhead running of the system.

VII. CONCLUSION

It seems from the Table II that the issues are still the challenging are not overcoming and it is required from the computer community more effort, however, by merging some existing techniques such as machine learning and the development of storage memory and responding to the execution gives an optimistic vision that these challenging will be in minimum level in the future. However, it can be concluded that modern previous works working on connecting many data flow graph nodes crossing into multiple machines in a cluster or within a single machine a crossing multiples computational devices. Adding to that, it can be observed that parallel and distributed mechanism is capable to adapt with the application with preserving the same computing architecture.

REFERENCES

[1] R. R. Zebari, S. R. Zeebaree, and K. Jacksi, "Impact Analysis of HTTP and SYN Flood DDoS Attacks on Apache 2 and IIS 10.0 Web Servers," in 2018 International Conference on Advanced Science and Engineering (ICOASE), 2018, pp. 156–161.

[2] S. R. Zeebaree, K. Jacksi, and R. R. Zebari, "Impact analysis of SYN flood DDoS attack on HAProxy and NLB cluster-based web servers," Indonesian Journal of Electrical Engineering and Computer Science, vol. 19, no. 1, pp. 510–517, 2020.

[3] S. R. Zeebaree, R. R. Zebari, and K. Jacksi, "Performance analysis of IIS10.0 and Apache2 Cluster-based Web Servers under SYN DDoS Attack," TEST Engineering & Management, vol. 83, no. March-April 2020, pp. 5854–5863, 2020.

[4] H. Shukur, S. Zeebaree, R. Zebari, O. Ahmed, L. Haji, and D. Abdulqader, "Cache Coherence Protocols in Distributed Systems," Journal of Applied Science and Technology Trends, vol. 1, no. 3, pp. 92–97, 2020.

[5] L. M. Haji, S. R. Zeebaree, O. M. Ahmed, A. B. Sallow, K. Jacksi, and R. R. Zeabri, "Dynamic Resource Allocation for Distributed Systems and Cloud Computing," TEST Engineering & Management, vol. 83, no. May/June 2020, pp. 22417–22426, 2020.

[6] H. Shukur, S. Zeebaree, R. Zebari, D. Zeebaree, O. Ahmed, and A. Salih, "Cloud Computing Virtualization of Resources Allocation for Distributed Systems," Journal of Applied Science and Technology Trends, vol. 1, no. 3, pp. 98–105, 2020.

[7] H. I. Dino, S. R. Zeebaree, O. M. Ahmad, H. M. Shukur, R. R. Zebari, and L. M. Haji, "Impact of Load Sharing on Performance of Distributed Systems Computations." International Journal of Multidisciplinary Research and Publications (IJMRAP), VOl. 3 no.1, pp. 30-37. 2020.

[8] S. R. M. Zeebaree, H. M. Shukur, L. M. Haji, R. R. Zebari, K. Jacksi, and S. M.Abas, "Characteristics and Analysis of Hadoop Distributed Systems," Technology Reports of Kansai University, vol. 62, no. 4, pp. 1555–1564, Apr. 2020.

[9] C. Xie, R. Chen, H. Guan, B. Zang, and H. Chen, "Sync or async: Time to fuse for distributed graph-parallel computation," ACM SIGPLAN Notices, vol. 50, no. 8, pp. 194–204, 2015.

[10] M. Abadi et al., "Tensorflow: A system for large-scale machine learning," in 12th ${$USENIX$}$ symposium on operating systems design and implementation (${$OSDI$}$ 16), 2016, pp. 265–283.

[11] J. Dean et al., "Large scale distributed deep networks," Advances in neural information processing systems, vol. 25, pp. 1223–1231, 2012.

[12] K. B. Obaid, S. R. Zeebaree, and O. M. Ahmed, "Deep Learning Models Based on Image Classification: A Review," International Journal of Science and Business, vol. 4, no. 11, pp. 75–81, 2020.

[13] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, Deep learning, vol. 1. MIT press Cambridge, 2016.

[14] M. R. Mahmood, M. B. Abdulrazzaq, S. R. Zeebaree, A. K. Ibrahim, R. R. Zebari, and H. I. Dino, "Classification techniques' performance evaluation for facial expression recognition." Indonesian Journal of Electrical Engineering and Computer Science, vol. 21 no.2, pp.176~1184. 2021.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction," Journal of Applied Science and Technology Trends, vol. 1, no. 2, Art. no. 2, May 2020, doi: 10.38094/jastt1224.

[17] G. Malewicz et al., "Pregel: a system for large-scale graph processing," in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, 2010, pp. 135–146.

[18] H. Dino et al., "Facial Expression Recognition based on Hybrid Feature Extraction Techniques with Different Classifiers," TEST Engineering & Management, vol. 83, pp. 22319–22329, 2020.

[19] S. R. Zeebaree, A. B. Sallow, B. K. Hussan, and S. M. Ali, "Design and Simulation of High-Speed Parallel/Sequential Simplified DES Code Breaking Based on FPGA," in 2019 International Conference on Advanced Science and Engineering (ICOASE), 2019, pp. 76–81.

[20] Z. N. Rashid, S. R. Zeebaree, and A. Shengul, "Design and Analysis of Proposed Remote Controlling Distributed Parallel Computing System Over the Cloud," in 2019 International Conference on Advanced Science and Engineering (ICOASE), 2019, pp. 118–123.

[21] G. D'Angelo and M. Marzolla, "New trends in parallel and distributed simulation: From many-cores to cloud computing," Simulation Modelling Practice and Theory, vol. 49, pp. 320–335, 2014.

[22] P. Y. Abdullah, S. R. M. Zeebaree, H. M. Shukur, and K. Jacksi, "HRM System using Cloud Computing for Small and Medium Enterprises (SMEs)," Technology Reports of Kansai University, vol. 62, no. 04, Art. no. 04, Apr. 2020.

[23] A. M. Law, W. D. Kelton, and W. D. Kelton, Simulation modeling and analysis, vol. 3. McGraw-Hill New York, 2000.

[24] M. F. Richard, "Parallel and Distribution Simulation Systems," 1999.

[25] R. Tyagi and S. K. Gupta, "A survey on scheduling algorithms for parallel and distributed systems," in Silicon Photonics & High Performance Computing, Springer, 2018, pp. 51–64.

[26] S. J. Kim, "A general approach to mapping of parallel computations upon multiprocessor architectures," in Proc. International Conference on Parallel Processing, 1988, vol. 3.

[27] Y. Xu, K. Li, L. He, and T. K. Truong, "A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization," Journal of Parallel and Distributed Computing, vol. 73, no. 9, pp. 1306–1322, 2013.

[28] Z. Tong, Z. Xiao, K. Li, and K. Li, "Proactive scheduling in distributed computing—A reinforcement learning approach," Journal of Parallel and Distributed Computing, vol. 74, no. 7, pp. 2662–2672, 2014.

[29] S. Brin and L. Page, "Reprint of: The anatomy of a large-scale hypertextual web search engine," Computer networks, vol. 56, no. 18, pp. 3825–3833, 2012.

[30] J. E. Gonzalez, Y. Low, C. E. Guestrin, and D. O'Hallaron, "Distributed parallel inference on large factor graphs," arXiv preprint arXiv:1205.2645, 2012.

[31] J. Gonzalez, Y. Low, A. Gretton, and C. Guestrin, "Parallel gibbs sampling: From colored fields to thin junction trees," in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 324–332.

[32] S. J. Chapin, "Distributed and multiprocessor scheduling," ACM Computing Surveys (CSUR), vol. 28, no. 1, pp. 233–235, 1996.

[33] Y. S. Jghef and S. R. Zeebaree, "State of Art Survey for Significant Relations between Cloud Computing and Distributed Computing," International Journal of Science and Business, vol. 4, no. 12, pp. 53–61, 2020.

[34] B. Shirazi, A. Hurson, and K. Kavi, "Introduction to scheduling and load balancing," IEEE Computer Society, 1995.

[35] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," in Concurrency: the Works of Leslie Lamport, 2019, pp. 179–196.

[36] H. Nakada et al., “Design and implementation of a local scheduling system with advance reservation for co-allocation on the grid,” in The Sixth IEEE International Conference on Computer and Information Technology (CIT’06), 2006, pp. 65–65.

[37] Y. Li, Y. Liu, L. Li, and P. Luo, “Local scheduling scheme for opportunistic routing,” in 2009 IEEE Wireless Communications and Networking Conference, 2009, pp. 1–6.

[38] J. Regehr, “Some guidelines for proportional share CPU scheduling in general-purpose operating systems,” 2001.

[39] C. A. Waldspurger and E. Weihl W, “Stride scheduling: deterministic proportional-share resource management,” 1995.

[40] R. Koshy, “Scheduling in distributed system: a survey and future perspective,” Int J Adv Technol Eng Sci, 2014.

[41] A. Gupta, A. Tucker, and S. Urushibara, “The impact of operating system scheduling policies and synchronization methods of performance of parallel applications,” in Proceedings of the 1991 ACM SIGMETRICS conference on Measurement and modeling of computer systems, 1991, pp. 120–132.

[42] E. Frachtenberg, G. Feitelson, F. Petrini, and J. Fernandez, “Adaptive parallel job scheduling with flexible coscheduling,” IEEE Transactions on Parallel and Distributed systems, vol. 16, no. 11, pp. 1066–1077, 2005.

[43] G. S. Choi, "Co-ordinated coscheduling in time-sharing clusters through a generic framework," in Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on, 2003, pp. 84-91.

[44] Y. Zhang, H. Franke, J. Moreira, and A. Sivasubramaniam, "Improving parallel job scheduling by combining gang scheduling and backfilling techniques," in Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International, 2000, pp. 133-142.

[45] P. G. Sobalvarro, S. Pakin, W. E. Weihl, and A. A. Chien, "Dynamic coscheduling on workstation clusters," in Workshop on Job Scheduling Strategies for Parallel Processing, 1998, pp. 231-256.

[46] G. D'Angelo and M. Bracuto, "Distributed simulation of large-scale and detailed models," International Journal of Simulation and Process Modelling, vol. 5, pp. 120-131, 2009

[47] Z. Mohtajollah and F. Adibnia, “A Novel Parallel Jobs Scheduling Algorithm in The Cloud Computing,” in 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE), 2019, pp. 243–248.

[48] M. Xu, C. Q. Wu, A. Hou, and Y. Wang, “Intelligent scheduling for parallel jobs in big data processing systems,” in 2019 International Conference on Computing, Networking and Communications (ICNC), 2019, pp. 22–28.

[49] V. A. Lepakshi and C. S. R. Prashanth, “Efficient Resource Allocation with Score for Reliable Task Scheduling in Cloud Computing Systems,” in 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 2020, pp. 6–12.