International Journal of Computer Techniques -– Volume 9 Issue 2, April 2022

RESEARCH ARTICLE OPEN ACCESS

IMPROVE WORKFLOW SCHEDULING TECHNIQUE USING SEMO IN CLOUD COMPUTING

Ms. K.E. Eswari M.C.A., M.Phil., M.E., SET.,[1], U. Naveenchandar[2],

[1]Associate Professor, Department of Computer Applications, Nandha Engineering College (Autonomous), Erode, Tamilnadu, India.

[2]Final MCA, Department of Computer Applications, Nandha Engineering College (Autonomous), Erode, Tamilnadu, India.

Email: [1]eswarisaravanan2001@gmail.com, [2]naveen.chandar2018bsc@gmail.com

Abstract. In the cloud environment, the workflows have been frequently used to model large-scale problems in areas such as bioinformatics, astronomy, physics and arithmetic process. Such a resource obtains a task from the cloud providers that has ever-growing data and computing requirements and therefore demand a high-performance computing environment in order to be executed in a reasonable amount of time. These workflows are commonly modeled as a set of tasks interconnected via data or computing dependencies.Cloud computing is the latest distributed computing paradigm and it offers tremendous opportunities to solve large-scale problems. However, it presents various challenges that need to be addressed in order to be efficiently utilized for workflow applications. Although the workflow scheduling problem has been widely studied, there are very few initiatives tailored for cloud environments. Furthermore, the existing works fail to either meet the user's Quality of Service (QoS) requirements or to incorporate some basic principles of cloud computing such as the elasticity and heterogeneity of the computing resources. This project proposes a resource provisioning and scheduling strategy for scientific workflows on Infrastructure as a Service (IaaS) and Platform as services clouds (PaaS). This project presents an algorithm based on the Superior Element Multitude Optimization (SEMO), which aims to minimize the overall workflow execution cost while meeting deadline constraints. The main scope of the project is used to analyze best available resource in the cloud environment depend upon the total execution time and total execution cost which is compare between one process to another process. If the provider satisfies the time least time, then the process becomes to termination.

Keywords: Cloud Computing, Resource Provisioning, Particle Swarm Optimization.

I. INTRODUCTION

Cloud computing is internet- grounded computing in which large groups of remote waiters are networked to allow sharing of data-processing tasks, centralized data storehouse, and online access to computer services or coffers. Shadows can be classified as public, private or mongrel. Cloud computing is a type of calculating that relies on participating computing coffers rather than having original waiters or particular bias to handle operations.

Virtualization is the main processing in Cloud computing. Virtualization software allows aphysical computing device to be electronically separated into one or further"virtual" bias, each of which can befluently used and managed to perform calculating tasks. Cloud computing adopts generalities from Service acquainted Architecture (SOA) that can help the stoner break these problems into services that can be integrated to give a result.

Cloud computing provides all of its coffers as services, and makes use of the well- established norms and stylish practices gained in the sphere of SOA to allow global and very easier access for cloud services in a standardized way.Cloudcomputing is a kind of grid computing; it has evolved by addressing the QoS (quality of service) and trustability problems. Cloud computing gives the tools as well as technologies to make data/ cipher intensive parallel operations with affordable prices when compared with traditional resemblant computing ways.

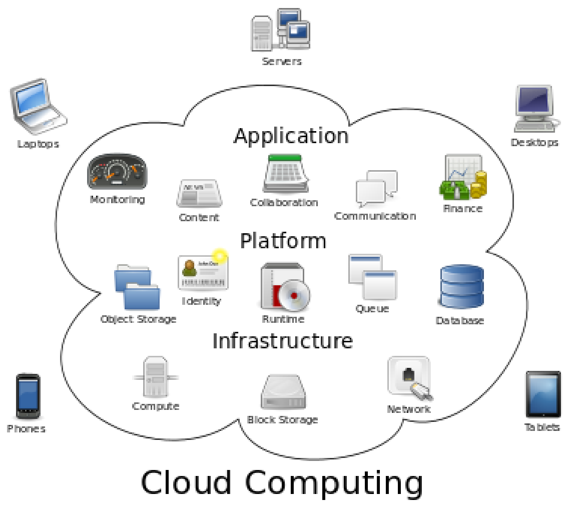

FIGURE 1.1 CLOUD COMPUTING

II. LITERATURE REVIEW

According to authors in this paper (1) load-balancing problems arise in numerous operations, but, most importantly, they play a special part in the operation of resemblant and distributed calculating systems. load-balancing deals with partitioning a program into lower tasks that can be executed coincidently and mapping each of these tasks to a computational resource such a processor (e.g., in a multiprocessor system) or a computer (e.g., in a computer network). By developing strategies that can collude these tasks to processors in a way that balances out the load, the total processing time will be reduced with bettered processor application.

Utmost of the exploration on load- balancing concentrated on static scripts that, in utmost of the cases, employ heuristic styles. Still, inheritable algorithms have gained immense fashionability over the last many times as a robust and fluently adaptable hunt fashion. The work proposed then investigates how a inheritable algorithm can be employed to break the dynamic load- balancing problem. A dynamic load- balancing algorithm is developed whereby optimal or near-optimal task allocations can ªevolveº during the operation of the resemblant computing system.

The algorithm considers other load- balancing issues similar as threshold programs, information exchange criteria, andinter-processor communication. The goods of these and other issues on the success of the inheritable-grounded load- balancing algorithm as compared with the first-fit heuristic are outlined.

Load-BALANCING algorithms are designed basically to inversely spread the load on processors and maximize their application while minimizing the total task prosecution time (11), (12). In order to achieve these pretensions, the load- balancing medium should be “ fair” in distributing the load across the processors. This implies that the difference between the heaviest- loaded and the lightest- loaded processors should be minimized.

Thus, the load information on each processor must be streamlined constantly so that the load- balancing medium can be more effective. Also, the prosecution of the dynamic load- balancing algorithm shouldn't take long to arrive at a decision to make rapid-fire task assignments (12). In general, load- balancing algorithms can be astronomically distributed as centralized or decentralized, dynamic or stationary, periodic ornon-periodic, and those with thresholds or without thresholds (13).

In a centralized load- balancing algorithm, the global load information is collected at a single processor, called the central scheduler. This scheduler will make all the load- balancing opinions grounded on the information that's transferred from other processors. In decentralized load-balancing, each processor in the system will broadcast its load information to the rest of the processors so that locally maintained load information tables can be streamlined. As every processor in the system keeps track of the global load information, load- balancing opinions can be made on any processor.

A centralized algorithm can support a larger system as it imposes smaller charges on the system than the decentralized (distributed) algorithm. Still, a centralized algorithm has lower trustability since the failure of the central scheduler will affect in the dysfunction of the load-balancing policy. Despite its capability to support lower systems, a decentralized algorithm is still easier to apply.

Also, for stationary load-balancing problems, all information governing load- balancing opinions is known in advance. Tasks will be allocated during collect time according to a priori knowledge and won't be affected by the state of the system at the time. On the other hand, a dynamic load- balancing medium has to allocate tasks to the processors stoutly as they arrive. A near-optimal schedule must be determined “ on the cover” similar that the tasks listed can be completed in the shortest time possible. As redivision of tasks has to take place during runtime, dynamic load- balancing mechanisms are generally harder to apply. Still, they tend have better performance in comparison to stationary bones.

Moment, load sharing and task migration are some of the extensively delved issues in dynamic load- balancing algorithms (12). In a situation whereby recently created tasks arrive aimlessly into the system, processors can come heavily loaded while others are idle or smoothly loaded.

Thus, the main ideal of load sharing is to develop task assignment algorithms to transfer or resettle tasks from heavily to smoothly loaded processors so that no processors are idle while there are other tasks staying to be reused. In general, a dynamic load- balancing algorithm consists of four major factors the load dimension rule, the information exchange rule, the inauguration rule, and the load- balancing operation (14).

The authors concluded that the proposed dynamic load- balancing medium developed using inheritable algorithms has been veritably effective, especially in the case of a large number of tasks. In fact, a GA- grounded scheme works more when the number of tasks is large and where we observe harmonious performance while other heuristics fail. The use of a central scheduler was also effective as it can handle all load- balancing opinions with minimuminter-processor communication. The threshold policy used also handed better performance in comparison to the first fit algorithm that doesn't have such a medium. Therefore, the GA- grounded algorithm worked rather well in terms of achieving the pretensions of minimal total completion time and maximum processor application (15).

The authors in this paper (2) stated that Fair queuing is a fashion that allows each inflow passing through a network device to have a fair share of network coffers. Former schemes for fair queuing that achieved nearly perfect fairness were precious to apply specifically, the work needed to reuse a packet in these schemes was O (log (n)), where n is the number of active overflows. This is precious at high pets. On the other hand, cheaper approximations of fair queuing that have been reported in the literature exhibition illegal geste. In this paper, the authors described a new approximation of fair queuing, that they called Deficit Round Robin. Their scheme achieves nearly perfect fairness in terms of outturn, requires only O (1) work to reuse a packet, and is simple enough to apply in tackle. Deficiency Round Robin is also applicable to other scheduling problems where servicing can not be broken up into lower units, and to distributed ranges.

When there's contention for coffers, it's important for coffers to be allocated or listed fairly. They need fire walk between contending druggies, so that the “ fair” allocation is followed rigorously. For illustration, in an operating system, CPU scheduling of stoner processes controls the use of CPU coffers by processes, and insulates well- conducted druggies from ill- conducted druggies. Unfortunately, in utmost computer net works there are no similar firewalls; most networks are susceptible to poorly-carrying sources. A guileful source that sends at an unbridled rate can seize a large bit of the buffers at an intermediate router; this can affect in dropped packets for other sources transferring at further moderate rates! A result to this problem is demanded to insulate the goods of bad geste to druggies that are carrying poorly.

An insulation medium called Fair Queuing (DKS89) has been proposed, and has been proved (GM90) to have nearly perfect insulation and fairness. Unfortunately, Fair Queuing (FQ) appears to be precious to apply. Specifically, FQ requires O (log (n)) work per packet to apply fair queuing, where n is the number of packet aqueducts that are coincidently active at the gateway or router. With a large number of active packet aqueducts, FQ is hard to apply 1 at high pets. Some attempts have been made to ameliorate the effectiveness of FQ; still similar attempts either don't avoid the O (iog (n)) tailback or are illegal.

In this paper they defined an insulation mechanising that achieves nearly perfect fairness (in terms of through put), and which takes O (1) processing work per packet. Their scheme is simple (and thus affordable) to apply at high pets at a router or gateway. Further they handed logical results that don't depend on hypotheticals about business distributions; we do so by furnishing worst-case results across sequences of inputs. Similar amortized (CLR90) and competitive (ST85) analyses have been a major influence in the analysis of successional algorithms because they finesse the need to make hypotheticals about probability distributions of inputs.

They described a new scheme, Deficit Round Robin (DRR), that provides near-perfect insulation at veritably low perpetration cost. As far as we know, this is the first fair queuing result that provides near-perfect outturn fairness with O (1) packet processing. DRR should be seductive to use while enforcing Fair Queuing at gateways and routers.

They've described theorems that describe the geste of DRR in backlogged business scripts. They've not fully understood its geste in non – backlogged cases, though they've conjectured that its outturn differs from the geste of bit-by- bit round robin by at most a constant cumulative factor.

Their simulations support this guess and indicate that DRR works as well in non – backlogged cases. The Quantum size needed for keeping the work O (1) is high (at least equal to Maz). We feel that while Fair Queuing using DRR is general enough for any kind of network, it's stylish suited for datagram networks. In ATM networks, packets are fixed size cells; thus Nagle’s result (simple round robin) will work as well as DRR. Still, if connections in an ATM network bear weighted fair queuing with arbitrary weights, DRR will be useful.

DRR can be combined with other FQ algorithms similar that DRR is used to service only the best- trouble business. They described a trivial combination algorithm called DRR that offers good quiescence bounds to Quiescence Critical overflows as long as they meet their contracts. Still, indeed if the source meets the contract, the contract may be violated due to " bunching" goods at intermediate routers. Therefore other combinations need to be delved. Recall that DRR requires having the amount size be at least a maximum size packet in order for the packet processing work to be low; this does affect detention bounds.

They believed that DRR should be easy to apply using being technology. It only requires a many instructions beyond the simplest queuing algorithm (FCFS), and this addition should be a small chance of the instructions demanded for routing packets. The memory requirements are also modest; 6K size memory should give a small number of collisions for about 100 concurrent overflows. This is a small quantum of redundant memory compared to the buffer memory used in numerous routers. Note that the buffer size conditions should be identical to the softening for FCFS because in DRR buffers are participated between ranges using McKenney's buffer stealing algorithm.

III. PROPOSED METHODOLOGY

The existing system develops a static cost-minimization, deadline-constrained heuristic for scheduling a workflow application in a cloud environment. The approach considers fundamental features of IaaS providers such as the dynamic provisioning and heterogeneity of unlimited computing resources. To achieve this, both resource provisioning and scheduling are merged and modeled as an optimization problem. Particle Swarm Optimization is then used to solve problem and then produce a schedule defining not only the tasksfor resource mapping, but also number of nodes to be utilized/assigned. In the thesis the process referred in the single cloud provider which is used to compute the consumption time and execution cost for running the process in the environment. The scheduling process is done in the basis of set of resources, number of task which are defined to that resource in the environment. The result of total consumption cost and total execution time using PSO logic are computed.

- Adaptable only in situations where same initial set of resource availability.
- Suitable only where single cloud service provider is available.
- Data transfer cost is not considered between different cloud data centers.

The dissertation presented the algorithm named SEMO (Superior Element Multitude Optimization) which is compare the total execution time and total execution cost between one processes to another process. In addition, it extends the resource model to consider the data transfer cost between data in cloud environment so that nodes can be deployed on different regions.

Also, it assigns different options for the selection of the initial resource pool. For example, for the given task, the different set of initial resource requirements is assigned. In addition, data transfer cost between data environment are also calculated so as to minimize the cost of execution in multi-cloud service provider environment.

In existing system, Dominant firefly behavior is applied on cloud load balancing methods, which is termed the dominant firefly algorithm. In a fireflies group, there are several dominant fireflies with many submissive fireflies. The method is being assumed that dominant fireflies denote cloud servers and submissive fireflies denote users. Whenever the cloud servers are filled with a lot of load (i.e., user requests), this needs to be regularly and equally balanced in such a manner that queries / requests are migrated to some other cloud server for completing the task.

Based on this firefly behavior, it is represented that if dominant firefly is already occupied with many other submissive fireflies during searching, then load is balanced by utilizing/passing on excess submissive fireflies to next available dominant firefly. According to this algorithm, when Cloud user requests are increased to a particular Cloud server, then users are automatically transferred to the next (dominant) Cloud server. Also, the path of submissive fireflies towards dominant firefly represents nearby cloud servers which provide load balancing dynamicity.

Along with present system implementation, deadline resource provisioning algorithm to execute Job is also carried out. The dissertation presented the algorithm which is named as SEMO (Superior Element Multitude Optimization) and then compares total execution time total execution cost mong various one processes. Moreover, it extends the new resource model for considering the data transfer cost among data in cloud environment so that nodes are deployed on different regions. In addition, it reassigns different options for selection of the initial resource pools. For example, for the given new task, different set of initial resource requirements are assigned. Moreover, data transfer cost between data environment is also calculated to reduce execution cost in multi-cloud service provider environment.

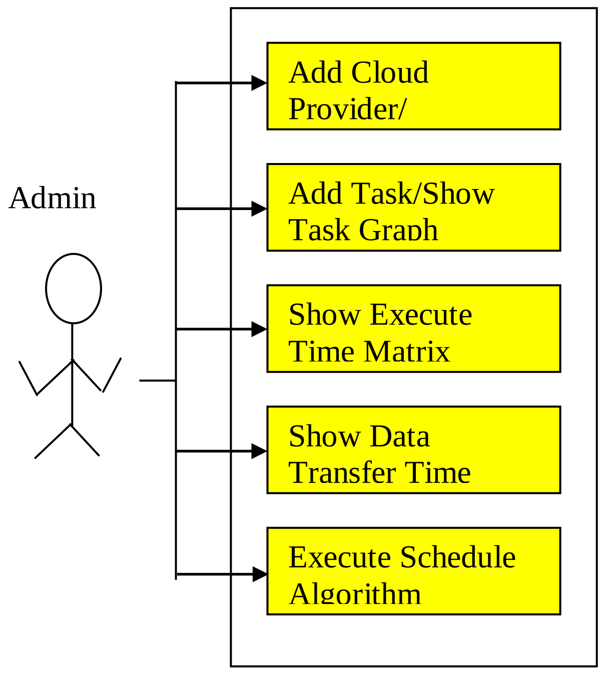

FIGURE 3.1 PROCESS FLOW

IV. FINDINGS

1.It is seen that improvements in load balancing of tasks in the Cloud computing environment is happened.
2. It is found that the load of job requests from Cloud end-users submitted to CSVMs is optimally balanced to increase the efficiency of the Cloud server.
3. An improvement in energy consumption among Cloud servers is found out.
4.Enhancements in m-learning environments could be made by finding many relational models to avoid the highest energy consuming server throughout the world.
5. This work reveals many challenges in m-learning using Cloud computing technologies.
6. It is seen that, there are many opportunities in the field of m-learning, green computing, and in Cloud-based organizations are available.
7. After applying the load balancing algorithm, response time was drastically decreased, which improved the m-learning system's overall performance.
8. PSO algorithm helps to select the best resources among the resource pools with the ownership of multiple cloud providers.
9. Execution cost and time could be minimized to a greater extent.

V. CONCLUSION

The thesis presented the SEMO(Superior Element Multitude Optimization) algorithm which is used to predict the least time computation in the cloud provider area. In addition, the thesis compared the time evaluation work between one dynamic resource flow to another process flow of dynamic resource in the cloud environment. In addition, it extends the resource model to consider the data transfer cost between data centers so that nodes can be deployed on different regions. Extending the algorithm to include heuristics that ensure a task is assigned to a node with sufficient memory to execute it will be included in the algorithm. Also, it assigns different options for the selection of the initial resource pool. For example, for the given task, the different set of initial resource requirements is assigned. In addition, data transfer cost between data centers are also calculated so as to minimize the cost of execution in multi-cloud service provider environment.The main contribution of thesis, the following problem solve in the existing system, they contribution are, Adaptable in situations where multiple initial set of resource availability.Suitable for multiple cloud service provider environments.Data transfer cost is reduced between different cloud data centers.

REFERENCES

[1] A. Y. Zomaya and Y.-H. Teh, ''Observations on using genetic algorithms for dynamic load-balancing,'' IEEE Trans. Parallel Distrib. Syst., vol. 12, no. 9, pp. 899–911, Sep. 2001.
[2] M.ShreedharandG.Varghese,''Efficientfairqueueingusingdeficitroundrobin,'' IEEE/ACM Trans. Netw., vol. 4, no. 3, pp. 375–385, Jun. 1996
[3] D. Eyers, R. Routray, R. Zhang, D. Willcocks, and P. Pietzuch, "Towards a middleware forconfiguring large-scale storage infrastructures" 'in Proc. 7th Int. Workshop Middleware Grids, Clouds e-Sci., 2009, p. 3.
[4] C.Fehling,T.Ewald,F.Leymann,M.Pauly,J.Rütschlin,andD. Schumm, ''Capturing cloud computing knowledge and experience in patterns,'' in Proc. IEEE 5th Int. Conf. Cloud Comput. (CLOUD), Jun. 2012, pp. 726–733.
[5] X.-S. Yang, ''Firefly algorithms for multimodal optimization,'' in Proc. Int. Symp. Stochastic Algorithms. Berlin, Germany: Springer, 2009, pp. 169–178.

[6] L. D. D. Babu and P. V. Krishna, ‘‘Honey bee behavior inspired load balancingoftasksincloudcomputingenvironments,’’Appl.SoftComput., vol. 13, no. 5, pp. 2292–2303, May 2013.
[7] B.Mondal,K.Dasgupta,andP.Dutta, “LoadbalancinginCloudcomputing using stochastic hill climbing-a soft computing approach,” Procedia Technol., vol. 4, pp. 783–789, Jun. 2012.
[8] T. R.Armstrong, D.Hensgen, The relative performance of various mapping algorithms is independent of sizable variances in runtime predictions, in: 7th IEEE Heterogeneous Computing Workshop (HCW ’98), 1998, pp. 79–87.
[9] A. Vouk, Cloud computing- issues, research and implementations, in: Information Technology Interfaces, 2008, pp. 31–40.
[10] B.Wickremasinghe, R.N.Calheiros, R. Buyya, Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computingenvironments and applications, in: Proceedings of the 24th International Conference on Advanced Information Networking and Applications (AINA 2010), Perth, Australia,, 2010.
[11] S.H. Bokhari, “On the Mapping Problem,” IEEE Trans. Computers, vol. 30, no. 3, pp. 550-557, Mar. 1981.
[12] S. Salleh and A.Y. Zomaya, Scheduling in Parallel Computing Systems: Fuzzy and Annealing Techniques. Kluwer Academic, 1999.
[13] F. Bonomi and A. Kumar, ªAdaptive Optimal Load-Balancing in a Heterogeneous Multiserver System with a Central Job Scheduler,º IEEE Trans. Computers, vol. 39, no. 10, pp. 1232-1250, Oct. 1990.
[14] C. Xu and F. Lau, Load-Balancing in Parallel Computers - Theory and Practice. Kluwer Academic, 1997.
[15] A.Y. Zomaya, F. Ercal, and S. Olariu, Solutions to Parallel and Distributed Computing Problems: Lessons from Biological Sciences. New York: Wiley, 2001.
[16] M. Armbrust, A. Fox, R. Griffith, et al. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.
[17] E. Anderson, S. Spence, R. Swaminathan, et al. Quickly finding near-optimal storage designs. ACM Transactions on Computer Systems, 23(4):337–374, 2005.
[18] P. Sarkar, R. Routray, E. Butler, et al. SPIKE: best practice generation for storage area networks. In SYSML’07: Proceedings of the 2nd USENIX workshop on tackling computer systems problems with machine learning techniques, pages 1–6, Berkeley, CA, USA, 2007. USENIX Association.
[19] F. Chong, G. Carraro, “Architecture Strategies for Catching the Long Tail”, Microsoft Whitepaper, 2006.
[20] J. Varia: “Cloud Architectures,” Technical Report, Amazon, 2010.
[21] C. Fehling, F. Leymann, R. Retter, D. Schumm, W. Schupeck, “An Architectural Pattern Language of Cloud-based Applications,” Proceedings of the Conference on Pattern Languages of Programs (PLoP), 2011.
[22] Bonabeau E., Dorigo M., Theraulaz G., Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, (1999)
[23] R.Buyya, C. Yeo, S.Venugopal, J.Broberg, I.Brandic, Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility, in: Future Generation Computer Systems, vo1.25, 2009, pp. 599–616.