ISSN No:-2456-2165

Bigdata and HPC Convergence with Locality Based Cuckoo Search Method

Dr. Reshmi B,
Associate Professor,
Department of Computer Science and Engineering,
Jawaharlal College of Engineering and Technology,
Palakkad, Kerala

Dr. P. Poongodi,
Professor and Dean,
Department of ECE,
Kaliagnar Karunanidhi Institute of Technology,
Coimbatore, TamilNadu

Abstract:- Bigdata analytics with High Performance Computing has attained focus of various researchers due to the services that has been provided to the cloud users with user satisfaction. Understanding the evolution of big data systems and HPC systems helps to define the key differences, the goals behind them, and their architectures. There are four broad application classes that driving the requirements of data analytics tools and frameworks. They are the data pipelines, large-scale machine learning including deep learning applications streaming applications, and graph applications. Historically, HPC systems have given less focus to data management and more focus to designing high-performance algorithms. Big data systems have done an excellent job in data management, data queries, and streaming applications. In this Research optimal scheduling of group of tasks would be done by using Locality Aware Scheduling based on Cuckoo Search Algorithm (LS-CSA) and the performance of Bigdata systems can immensely benefit from HPC. This method would schedule the similar tasks that shares the same data in the virtual machine where its corresponding data resides. The overall evaluation of the research work is done in the Cloudsim environment which is implemented and evaluated in terms of various performance metrics. The proposed research method provides optimal results than the existing research methods.

Keywords:- *HPC Systems, ML, Scientific application, Workflow, Big Data Systems.*

I. INTRODUCTION

Big data computing and high-performance computing (HPC) has evolved over the years as separate paradigms. With the explosion of the data and the demand for machine learning algorithms, these two paradigms increasingly embrace each other for data management and algorithms. For example, public clouds such as Microsoft Azure are adding high-performance compute instances with InfiniBand and large-scale deployments of GPUs in HPC clusters, enabling artificial intelligence algorithms on large datasets. In the future, you can expect more applications to explore the benefits of HPC while taking advantage of big data systems. The differences between HPC systems and big data systems, outlining the areas they can benefit from each other.

Parallel operators are one of the key foundational blocks in a distributed computing system. MapReduce is one of the most well-known parallel operators, and there are many more, such as gather, partition, and scatter. These operators help distribute data among parallel tasks and have a consensus when a computation progresses. HPC systems have their own parallel operators with similar semantics as big data systems. There are many possible optimizations for these operators, and they are a huge factor in any distributed application. Advanced hardware such as InfiniBand plays another big role in HPC applications. They provide low-latency, high-throughput networking among a large number of nodes. Such networks are vital to scale applications to thousands of nodes and tens of thousands of CPU cores.

The way iterations are programmed and executed is the other major difference between HPC and big data systems. Iterations are a key component in complex applications and one of the success points behind Spark over Hadoop. There are many ways to handle an iteration in different programming models and systems. For example, in Spark, the iterations are handled in a central place (driver), Flink embeds iterations into the data flow graph and the HPC system distributes iterations to each worker. Each of these choices has different implications for programming models and performance.

II. BIGDATA AND HPC SYSTEMS

Programming APIs and data abstractions are quite different between big data and HPC systems. HPC systems have adopted low-level APIs while big data systems have adopted high-level user-friendly APIs. The performance and usability is a delicate balance in any system, and achieving performance while preserving usability is a challenge. Examples of big data APIs around HPC systems and the integration of HPC techniques to big data systems has been explored.

It utilizes systems and speculations drawn from numerous fields inside the expansive regions of arithmetic, insights, data science, and software engineering, specifically from the subdomains of machine learning, characterization, group examination, vulnerability evaluation, computational science, information mining, databases, and representation. graphics processing units (GPUs) and field- are well equipped to combine the predictive capabilities of simulation with the analytic and optimization capabilities of machine learning. With the recent adoption of deep neural networks for machine learning, data analysis now has computational characteristics of traditional HPC workloads.

HPC's and data analytic systems are adopting the use of accelerators such as GPUs to improve the performance of individual computing nodes, and this trend will continue as a response to the limited gains of scaling.

III. CHALLENGES WITH RESPECT TO BD INTEGRATION WITH HPC SYSTEMS

Significant difficulties with relevancy programming include:

• System plan: At the framework level, there's a essential hole between however HPC frameworks ar planned (firmly coupled assortments of unvaried hubs) versus baccalaureate frameworks (in lightweight of distributed computing server farm structures that comprise of monumental quantities of roughly coupled and doubtless heterogeneous problem solving hubs). These underlying contrasts, thus, have prompted a compound within the product stack that's each innovative and social. The HPC stack depends on instruments created in government analysis centers and also the scholarly community. Conversely, the baccalaureate stack may be a ton larger and a lot of differed and is usually determined by ASCII text file comes, with the first patrons being business substances.

• Edge Computing, or savvy registering at the edge: This was recognized as a quickly arising keyarea—one requiring new reflections, ideas, and instruments, as well as the merchandise design, runtime frameworks, and perhaps even new programming dialects. The arising mix of edge, cloud, and HPC would force programming that produces these conditions less complicated to program, investigate, improve, and interoperate in varied future application regions.

• System the board: no matter whether or not the registering is HPC or baccalaureate, dispatching monumental positions can expect backing to diminish work dispatch immobility, screen occupations more and more, and handle runtime hub and completely different disappointments.

• Common libraries: Most space researchers and baccalaureate purchasers haven't got the ability to influence the intricacies of arising instrumentation. Having a typical arrangement of libraries would allow nonexperts to any or all the a lot of effectively utilize the frameworks, departure the programming of those gadgets to specialists. BD, and ML., a dream arose of an expensive procedure setting comprising of heterogeneous mixes of edge, cloud, and elite process frameworks. This setting would be flexible ANd have the choice to induce data from an assortment of sources like logical and clinical instruments, device organizations, and security and foundation checking systems. It would have edge web of Things gadgets that will free important highlights and convert data into structures cheap for ingesting and putt away within the cloud. monumental scope data investigation would run within the cloud, consolidating eaten data with place away knowledge sets.

HPC frameworks would perform all the a lot of computationally serious styles of examination and sweetening, even as arrived not on time for discerning displaying. a very wealthy registering climate might provide capacities past those of this confined frameworks. For medicine and clinical examination and treatment, it might empower the use of clinical, lab, and astonishingly sub-atomic data for patients and for specialists. data sources might incorporate good upbeat applications wherever patient results ar related to a symptom primarily based registered model, during this manner putt data as a "computerized first" resource within the medical services framework. The registering climate would allow logical and clinical scientists to require care of problems with varied levels of chance in manners that allow data to coach models and reproductions to construct higher models. Accomplishing this vision of an expensive registering biological system would force new skills in instrumentation (processing, organization, and capacity); the executive ways of activity; and programming. Giving real combination among momentum and future registering conditions presents varied specialized and association challenges, but it might provide skills in logical examination, public safety, treatment, and trade past what's conceivable these days.

IV. TASK SCHEDULING OF HPC BIGDATA SYSTEMS USING CUCKOO SEARCH ALGORITHM

The analysis technique here tries to explore the performance of HPC cloud systems with optimum planning technique. Task planning is delineating as tasks or jobs that may be waiting for a collection of computing resources. Computing resources is measured based on availability of virtual machines For instances, given the matter that if there's a collection of n tasks, which require to be regular on m identical machines , whereas attempting to cut back the whole time interval. All the tasks square measure non- preemptive [5] that is the process of 1 task in one machine can't be dead on another and every one the resources will execute all variety of tasks. In [5], a mathematical model it is necessary to balance the load of individual VM whereas attempting to reduce the create span. With the assistance of this model, time interval has been evaluated. Let there be a collection of m virtual machines wherever all the tasks are going to be processed and n variety of tasks pictured as a collection [5]. The time interval varies from one VM to alternative VM supported based on the capability and information measure of the VM. The research method here tries to explore the performance of HPC cloud systems with optimal scheduling method. Task scheduling can be described as tasks or jobs that will be executed by a set of computing resources. Computing resources are known as virtual machines. For instances, given the problem that if there is a set of n tasks, $\{T_1, T_2, T_3, T_4... T_n\}$ which need to be scheduled on m identical machines {VM1, VM2….VMm}, while trying to reduce the total processing time. All the tasks are non-preemptive [5] that is processing of one task in one machine cannot be executed on another and all the resources can execute all type of tasks. In [5], a mathematical model for scheduling has been proposed to balance the load of individual VM while trying to minimize the make span. With the help of this model, processing time has been evaluated. For the mathematical representation of the scheduling problem let there be a set of m virtual machines where all the tasks will be processed and n number of tasks represented as a set [5]. All the tasks in the model are non-preemptive and independent. Processing time varies from one VM to other VM based on capacity and bandwidth of the VM. So Capacity of individual VM,

$C_i = pe_{num,i} \times pe_{mips,i} + vm_{bw,i}$

Where number of processing element of VM is denoted by pe_{num}, pe_{mips} is the million instructions per second of all processors and vm_{bw} is the communication bandwidth ability to a VM.

Capacity of all VMs, $C = \sum_{i=1}^{m} C_i$

Total capacity of all the VM is known as capacity of the datacenter. Load of individual VM can be defined as the total length of tasks that are assigned to the VM.

$$L_{vmi,t} = \frac{N\ (T, t)}{S\ (VM, t)}$$

It is calculated using number of task at time t on a service queue and divided by the service rate of individual VM at time instance t. Load of all VM is calculated as,

$$L = \sum_{i=1}^{m} L_{vmi}$$

Processing time of a VM, $PT_i = \frac{L_{vmi}}{C}$

Processing time of all the VM, $PT = \frac{L}{C}$

This equation defines the total processing time of all the VM. So the problem is to minimize,

$$PT_{max} = \sum_{i=1}^{m} PT_i, \qquad \text{where } i \in VM, i = 1,2,3, \ldots n$$

And load value of a VM must be less than its capacity.

Cuckoo Search Algorithm is a meta-heuristic calculation that models normal conduct of cuckoo species [13], [20]. Each egg in the home addresses an answer. The principle point is to discover better answer for change a not so better arrangement in the home. At a time, each cuckoo lays one egg and dumps this egg in a home that has picked haphazardly. The best home with excellent eggs (arrangement) will be done to the future. The host can find an outsider egg by a likelihood Pa [0, 1]. The host bird either discards the egg or relinquishes the home to construct another home in another area. The calculation is given beneath:

```
Step 1: Objective functions f(x), x=(x1, x2... xd)
Step 2: Generate the population of n host nests xi, (i=1, 2… n)
Step 3: While (t<Maxgeneration)
3.1. Get a cuckoo randomly and generate a new solution by Lévy flights
3.2. Evaluate its quality/fitness, Fi
3.3. Choose a nest among n (say j) randomly
3.4. if (Fi>Fj),
Replace j with a new solution
end
3.5. Abandon a worse nest with probability (Pa)
3.6. Build new ones at new locations via Lévy flights
3.7. Keep the best solutions (with high quality)
3.8. Rank the solutions and find the current global best
End while
Step 4: Post process results and visualization
```

For minimization issues, the fitness or quality function is proportional of target work. An answer is addressed by an egg in the home and the cuckoo egg addresses another arrangement. So principally there is no distinction between an egg, a home and final soution. From x_i^t of it the next generation is calculated by,

$$X_i^{(t+1)} = X_i^{(t)} + \alpha \bigoplus \text{Levy}\ (u)$$

And
Lévy (u) $= t^{-\gamma}$, $1 < \lambda \leq 3$

Where $\alpha > 0$ is the step size, depending on the scale of the given problem, $\oplus$ means entry-wise multiplications. To track down an ideal arrangement of the given issue, an answer is addressed by an egg and new arrangement is addressed by a cuckoo egg. The underlying populace is addressed by a vector of n components, where n addresses the absolute number of assignments to be booked and vector esteem addresses the VM number.

V. EXPERIMENTAL RESULTS

The experimental evaluation of the proposed research method is done in the Cloud simulation environment to prove the effectiveness of the proposed research algorithm and perform locality aware scheduling in HPC cloud systems. The comparison evaluation is made between existing methods namely SLA based model [15].The performance measures of the different algorithms with respect to Profit that are considered in this work to prove the improvement of the proposed research method.

➢ *Profit Comparison*

Profit is expressed as the difference between the total amount that is invested and the total amount which is retrieved as earnings. The profit of the proposed research methodology should be high for its better performance. Profit is calculated using the procedure as given in the following Equation.

$$Prof_{ij}^{new} = B^{new} - Cost_{ijl}^{new}; \forall i \in I, j \in J, l \in N_j$$

The simulation values obtained are shown in the following table 1.

Table 1. Total profit comparison values

Total number of user request	Total profit in $		
	SLA based model	MTCF	LS-CSA MODEL
100	850	990	1500
200	2000	2400	3100
300	2700	3100	3450
400	3700	3900	4160
500	4050	4100	4850

The comparison chart of profit of the proposed research methodologies and the existing research method is given in the following Figure 1.

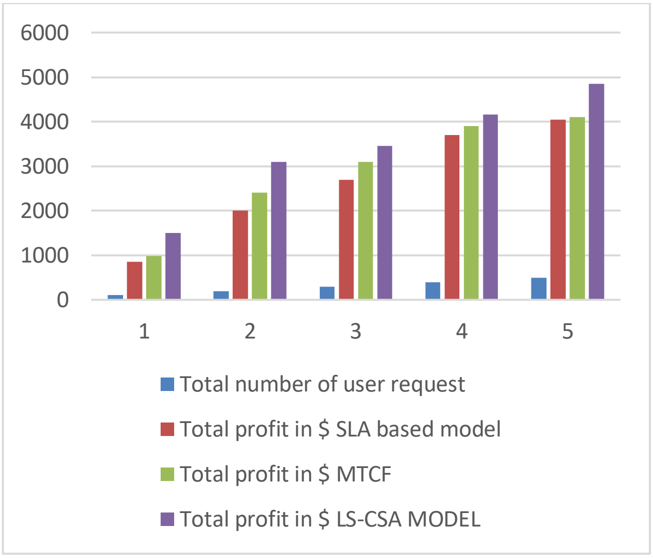

Fig 1. Comparison chart of profit

Figure 1 shows that the LS-CSA Model provides the higher profit 22.5% more than SLA model based and 11% more than MTCF by accepting more users and initiating the least number of VMs (50% less than SLA based model, 33% less than MTCF model when arrival rate increases from 100 to 500. This is because LS-CSA accepts users with existing HPC machines with penalty delay.

VI. CONCLUSION

High performance computing scientific application handling is the most complex task in the real world cloud scenario where it requires more number of computational resources for the execution and allocation. The heterogeneous nature of cloud resources would make it more difficult task that incurs more of computational overhead. These problems are focused in the proposed research method by introducing the novel framework called Similarity aware High Performance Scientific Application Scheduling (SHPSAS). This research methodology assures the optimal scheduling and successful completion of the HPC scientific applications that are submitted into the cloud server. The proposed research method is implemented and evaluated in the cloud sim environment under varying server configurations values. From this evaluation, it can be proved that the proposed research method provides an optimal scheduling rate than the existing research methodologies. Overall evaluation of the research method proves that the proposed research methods can assure the optimal outcome in terms of increased profit with reduced response time. In future work, handling of heterogeneous resources with different characteristics has been concentrated to provide the better outcome. In addition more soft computing techniques can be integrated to improve the scheduling efficiency. Future prediction of resource availability can be concentrated to improve the resource allocation efficiency.

REFERENCES

[1]. Zheng, Z., Wang, R., Zhong, H., & Zhang, X. (2011, March). An approach for cloud resource scheduling based on Parallel Genetic Algorithm. In Computer Research and Development (ICCRD), 2011 3rd International Conference on(Vol. 2, pp. 444-447). IEEE.

[2]. Li, W., Tordsson, J., & Elmroth, E. (2011, November). Modeling for dynamic cloud scheduling via migration of virtual machines. In Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on (pp. 163-171). IEEE.

[3]. Jackson, K. R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., ... & Wright, N. J. (2010, November). Performance analysis of high performance computing applications on the amazon web services cloud. In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on (pp. 159-168). IEEE.

[4]. Wu, L., Garg, S. K., & Buyya, R. (2012). SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments. Journal of Computer and System Sciences, 78(5), 1280-1299.

[5]. Hoang, D. T., Niyato, D., & Wang, P. (2012, April). Optimal admission control policy for mobile cloud computing hotspot with cloudlet. In Wireless Communications and Networking Conference (WCNC), 2012 IEEE (pp. 3145-3149). IEEE.

[6]. Garg, SK, Buyya, R & Siegel, HJ 2009, 'Time and cost trade-off man-agement for scheduling parallel applications on utility grids'. Future Generation Computer System. vol. 26, no. 8, pp. 1344-55

[7]. Dhingra, A & Paul, S 2014, 'Green Cloud: heuristic based BFO tech¬nique to optimize resource allocation', Indian Journal of Science and Technology; vol. 7, no. 5, pp. 685-91.

[8]. Chitra, S & Kalpana, B 2014, 'Optimum session interval based on particle swarm optimization for generating personalized ontology'. Indian Journal of Science and Technology, vol. 7, no. 8, pp. 1137-43.

[9]. Wu, L., Garg, S. K., & Buyya, R. (2012). SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments. Journal of Computer and System Sciences, 78(5), 1280-1299.

[10]. Navendu Jain, Ishai Menache, Joseph (Seffi) Naor & Jonathan Yaniv 2015, 'Near-Optimal Scheduling Mechanisms for Deadline-Sensitive Jobs in Large Computing Clusters', ACM Transactions on Parallel Computing, vol. 2, Issue 1.
[11]. Kiranveer Kaur & Amritpal Kaur 2015, 'Optimal Scheduling and Load Balancing in Cloud using Enhanced Genetic Algorithm', International Journal of Computer Applications (0975 – 8887) vol. 125, no. 11.
[12]. Nasrin Hesabian & Hamid Haj Seyyed Javadi 2015, 'Optimal Scheduling In Cloud Computing Environment Using the Bee Algorithm', International Journal of Computer Networks and Communications Security, vol. 3, no. 6, pp. 253-258
[13]. Amit Agarwal & Saloni Jain 2014, 'Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment', International Journal of Computer Trends and Technology (IJCTT) – vol. 9, no. 7.
[14]. Ruben Van den Bossche, Kurt Vanmechelen & Jan Broeckhove 2010, 'Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads', 2010 IEEE 3rd International Conference on Cloud Computing.
[15]. A Bharathi, RS Mohana, A Ushapriya, Reducing Energy Consumption and Increasing Profit with Task Consolidation in Clouds, IJESIT, Volume 3, Issue 1, January 2014, 200 – 207, ISSN: 2319-5967