DYNAMIC ASSIGNMENT OF SCIENTIFIC COMPUTING RESOURCES USING CONTAINERS

Manish Kumar Abhishek

Research Scholar, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, India

D. Rajeswara Rao

Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, India

K. Subrahmanyam

Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, India

Abstract

Currently the requirement of data processing using container assisted cluster computing is gaining a high momentum and becoming mature now days. In scientific workloads scenarios, data processing needs high performance computing cluster to execute the applications. High performance computing aka cluster computing is configured with multiple system parallelism equivalents to provide supercomputing. Parallel cluster could build up with high speed interconnected computational systems to enhance the execution speed. The open source or community developed software could be used to build the cluster and to deploy the applications but the allocation and scheduling of computing resources is usually a matter of concern which must be done efficiently. This paper is proposing architecture to assign the computing resources at run time to address the challenge of high performance computing resources such as GPU, Storage, and CPU as cloud deployments. Results are captured as output of different deployment sizes in terms of computing infrastructure and the comparison of load and execution time with different configurations along with deployment of computing resources.

***Keywords*: High Performance Computing, Computing Resources, Parallel Computing, Resource Scheduler, Cluster Computing, Virtualization, Torque, Slurm, X86, GPU.**

1. Introduction

There were 4.66 billion active internet users reported in January 2021 worldwide and generating huge amount of data every day which is estimated around 2.5 quintillion bytes per day. To solve the large computational problems in terms of application execution is always recommended in form of High Performance Computing from last few decades. Generally it uses Message Passing Interface (MPI) library to allocate the cluster computing resources to multiple parallel running job. Allocated infrastructure must be suffice to meet minimum computational requirement in terms of memory, CPU, GPU, network interconnect to hold cluster computing functionalities. A current resource manager for cluster computing like Torque is responsible for mapping of resources to the application execution with the proposed configurations and specified requirement. In case of updating or any changes required in terms of already defined computing resources, the complete job respective to application must reset and resubmit due to static behavior in configuration. To overcome this limitation we need dynamic assignment of resources that will not only help to ensure running application with better performance but also efficient way to handle data exhaustive application and coexist it's with cloud deployment.

Now days cluster computing required by many scientific and consumer centric computational jobs. User mainly interested in computer intensive job which required huge IO and CPU performance. GPU along with CPU provides faster scalar processing through parallelization. Linux is most preferable choice to build such cluster with high speed interconnect which helps to achieve parallelism. In parallel computing submitted job is split using job scheduler and MPI allow to run in parallel across cluster of computing resources connected by high speed network. Usually in HPC job scheduler placed job across cluster to ensure maximum performance but it's not always ensure maximum resource utilization. Minimum resource starvation of any thread can have a huge impact on whole running application. It is needed to ensure resource allocation very carefully with required amount of buffer. In HPC cluster same sets of job executed multiple time with different set of data. Typically with ranking and profiling technique computing resource could assign per set of job which also ensure proper resource utilization.

The other sections in this paper are organized in herewith mentioned manner: Section 2 is describing resource manager for distributed work load, Section 3 describes the proposed framework for manage computing resources and its allocation at run time without impacting job queue performance, Section 4 is capturing the details of evaluation blueprint, output and work done and Section 5 holds the Discussion and conclusion.

2. Resource Allocation Approaches

2.1. *Work Load Schedulers*

High performance computing workload can be govern by workload schedulers which supports Linux MPI efficiency of workload scheduler based on wait time minimization and resource optimization which should also support the resource reservation in terms of CPU, GPU and memory assigned. Job generally not consumes all resources and its allocation is usually based on of computing node granularity. Queue can be established for combination of job characteristics and job partition could be maintained. Jobs can be further divided into five sub categories i.e. evolving, adaptive, rigid, malleable and moldable. Large data sets problem can solve by adaptive job which is dynamic in nature and flexible for resource allocation changes requirements. Fixed job required predefined format of resources before job submission and assigned to long running application types.

2.1.1 TORQUE

Torque (Tera-scale Open-source Resource and Queue manager) is widely used distributed workload resource manager which is used to assign and entry control for job sets between computing nodes for distributed jobs. It acts as scheduling interface which helps to achieve fault tolerance including scalability and usability with extensive logging. It is widely used for supercomputing deployment and control batch jobs.

Commonly used commands:

- Qsub –> To submit job
- Qdel –> To stop job after completion
- Qstat –> Display current status of running jobs

2.1.2 Slurm

Slurm the Simple Linux utility resource management is highly scalable and open source cluster workload management which effectivity managed large and small Linux clusters and there job scheduling additionally. It is not required any kernel modification for deployment. It allocates node exclusive or shared as per job requirement and managed by centralized manager slurmctld to monitor resource and work each computer node having slurmd daemon which enable fault tolerant communication between scheduler and worker.

Commonly used commands:

- Srun –> To initiate job
- Scontrol –> To manage and monitor submitted jobs
- Sview –> Graphically report system

2.1.3 IBM Spectrum LSF

IBM Spectrum LSF widely used by semiconductor industry to manage electronic design and automation work load. LSF manage to use heterogeneous resources tied together in one single work load that deliver faster, more reliable and workload performance at reduced cost. LSF provide resource management which find best resources based on host load and site polices to run the job and monitor their progress. Multiple job slots can be configured per host which can consume job from scheduler queue until job slots for host gets full. All jobs waits in queue until dispatched and scheduled to respective host.

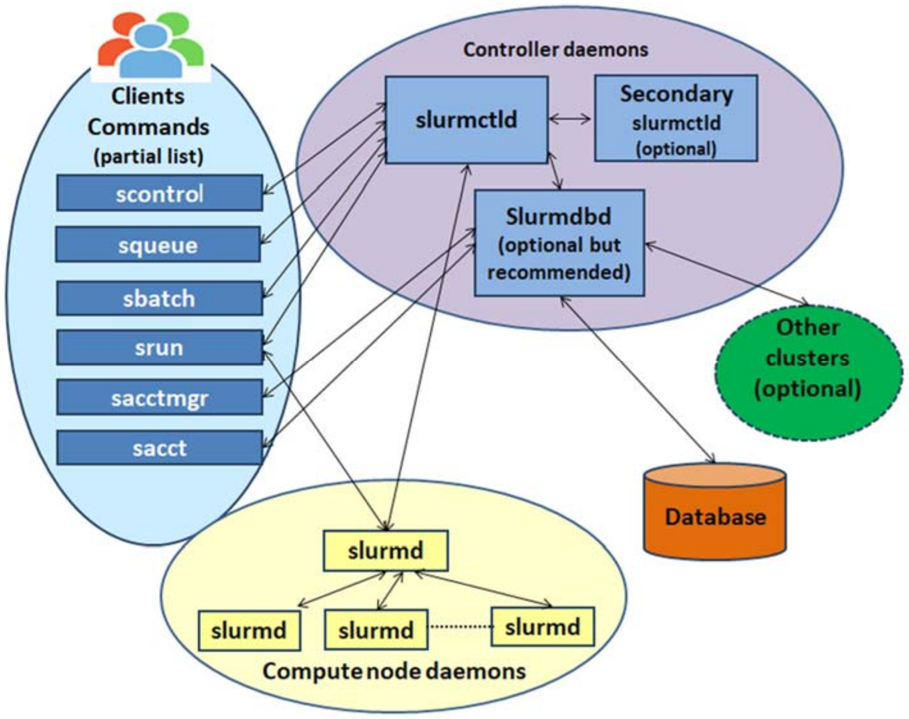

Fig. 1: Typical Resource Management using Slurm

Fig. 1 shows schematic diagram for managing and distributing computing resource in HPC cluster. Using Slurm resource manager, each job having assortment of constraints such as job size, job time limit and user permitted etc. Job getting allocated to node until the resources CPU, memory and IO is within cluster limit getting exhausted. Once job is assigned to node scheduler search for available for available resource and places in queue accordingly. For instance job may be started to utilize all node or multiple jobs may indecently use a portion of allocation, In general HPC cluster recourse allocation done statically to job prior to execution via static assignment resource allocation is always challenging task which distributed job scheduled to run. Technique like Co-Scheduling could be used to make changes in resource utilization which can lead to SLA violation. Job array are efficient mechanism for managing batch jobs µ based on weightages € of resource impact.

2.2. *Challenges in Resource Allocation*

HPC cluster is very complex in nature and consist many vital components. Scheduler plays vital roles in jobs placement and distribution among nodes nowadays. HPC cluster becomes more and more heterogeneous with respect on configuration and architecture however resource manager often not flexible enough to incorporate. It's hard to analyses and incorporate control and resource changes once queue assigned where it should be operated in isolation. It's beneficial to have co-existed job with adequate isolation on kernel level implementation e.g. Docker normally batch schedulers to allocate resources with no time-sharing of nodes. In this paper we have proposed an approach that bridge gap of static scheduling middleware with cloud native schedulers which always prefer better utilization of computational resources this concept provides containers layers over exiting kernel deployment therefor advance implicit reservation possible.

3. Proposed Model for Dynamic Resource Management

Here, we have elucidate our proposed approach in which the infrastructure computing resources can be allocated at run time without impacting job queue performance to achieve fault tolerance including isolation of resources from other job sets executing on same node. Fig. 2 representing the semantic layout of the proposed high level architecture holding the job queue and data set flow to complete the job. In proposed model job set is going to manage by Slurm NDC containers implementation which is greatly relay on exiting kernel features and entry control of jobs using cggroups functionality which allows greater control over application and can interact with job in any given point of time for HPC related workload. Normally it is restricted with mount namespace. Slurm natively supports interacting with unprivileged NDC containers for jobs and steps which ensure that job is not going to overconsume the resources in apropos of CPU, Memory and IO from allocated ones. We have proposed to configure HPC cluster using Linux containers with high speed interconnect. Linux Operating systems have chosen with latest kernel (Rocky Linux 8) for deployment of HPC applications. Containers images also build from same version of Linux with Intel MPI preinstalled Slurm interfaced with pattern replacement of commands issued for each NDC runtime operations. Sites with trusted users can add them and control via Docker directly from job queue which can interface with users and running jobs core component of mentioned services is implemented using following micro services architecture in lieu of monolith to get comprehensive fault tolerance and scalability in Slurm.

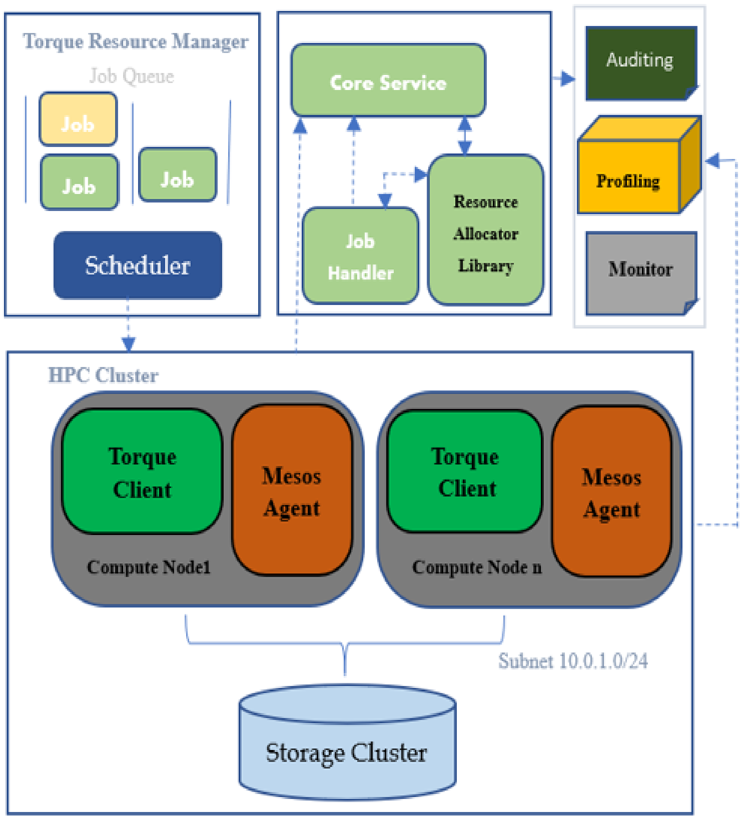

Fig. 2: High level architecture for HPC Cluster deployment.

Fig. 2 shows our proposed resource aware container platform deployment where GPU aware scheduler applications get deployed using core services initialization. Whenever inference engine gets job scheduled from custom job scheduler, the inference engine analyses resources will be used and calculate with resource database. As result inference engine gets motif of resource consumption and modeled prediction architecture for resource placement and tried to profiling once again. It will update to master database and shared with users to get current resource consumptions and can pick out the hotspots for said extent of time e.g. 2 hours. Events will trigger in case to thresholds breaches and consumptions go beyond set limit. Micro services architecture is used for deployment model and standard deviation can be calculated on base of data consumed by mentioned service is very light weight type and having minimum memory utilization.

In this section we have proposed Algorithm 1 which deigned to find appropriate node for resource allocation. Rank has been given to type of services and based on their effect on computing resources. NDC_s is selected node for container placement and there service type is weight vector which ensure weightage of each impacting factor. CPU, GPU, memory, IO and affinity categories are considered as computing resources factor. If overall ranking calculated eight times of value of m that deduce that computational of assignment of n containers further it is computing requirement based on 8 proliferate having the complexity i.e. O(l x n). Container is represented by annotation ct.

Algorithm 1. The pseudocode to assign the resources for deployed containers.

```
Data Input: $Sv_{t,}$  $A_n$, $PC_n$
Data Output: $C_x$
1: $NDC_s = \varphi$
2:  for each ct ∈ $PC_n$ do
3:         instantiate $NDC_s$ with $WN_s$
4:         for each l ∈ $WN_s$ do $A_n$
5:              $Rank_N = \varphi$
6:              find Scorecpu, Nc
7:              find Scoremem, $N_l$
8:              find Scorenet, $N_s$
9:              find GPU, f, t
10:             find Affinity, f, t
11:             Scoretotal, $N_* = Rank_N * Sv_t$
12:             set $NDC_s$ with $Rank_N$
13:         end for
14:         sort NDCs using node_Scoretotal in asc order
15:         map (ct, $NDC_s$ [0]) into $C_p$
16: end for
17: return $C_x$;
```

4. Evaluation

In this section, we have configured the proposed approach based model that implements our scheduling algorithm for dynamic assignment of resources and ensure non-utilized computing resources for non-HPC applications based on queue throughput, profiling and application performance. We have prepared test bed with two applications which is designed for HPC task and one for non HPC task configured with dynamic and static type of use cases. Three cluster size prepared here 512, 256 and 128 vCPU considered time for test application is 60 minute. We have considered execution time and performance impact at test bed cluster configured in university HPC research center where Fig. 3 and Fig. 4 having corresponding result of Slurm with reference of proposed model. The cluster consist of 22 HP ProLiant Gen 8 servers with 2 X Intel cascade lake processors with 32 core each processors (2.8 GHZ), 256 GB DDR4 with 400Gbps InfiniBand Ethernet network with shared cluster filesystem having 1 million IOPS. Proposed cluster running at Rocky Linux 8 and managed by Slurm workload processors.

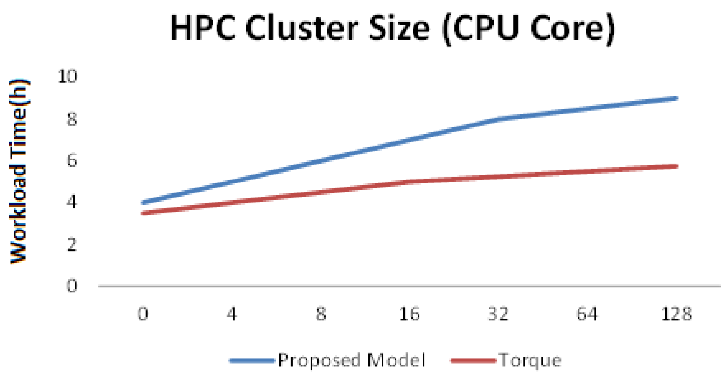

Fig. 3: Average Queue Make span

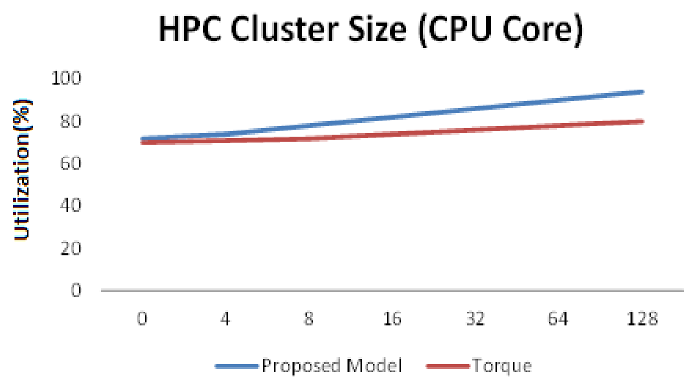

Fig. 4: Resource Usage Average Computation

4.1. *Use Case*

We have executed one application based on non-HPC architecture and two application of HPC architecture using CPU core 32, 16 and 8 for each. Each container execution is controlled using baseline and cgroup static mapping. Dynamic scheduler plugin provide score to each node scheduler will select node chosen based on highest weighted scores and compare result of proposed model with default Slurm resource mapper which is using type of both parallel and random freezer where they are allowing the execution of application once at a time and stopping the application respectively based on fraction of time intervals priorities required from application perspective so that higher priority based always gets resource allocation first and then the lower priority ones. Using Slurm log and Docker metrics to handle every use case which is responsible to hold the mapping of CPU usage, Kube-scheduler can be customized by writing a configuration file and provide scheduling behaviors by implementing it which ensure the every single job running time and other submitted job.

Table 1 is showing the output corresponding to the HPC job running time sets which is ensured using the different configuration sets to handle the high computing workloads based on specified CPU cores i.e. 8, 16 and 32 in number. The results respective to Job execution are promising using our proposed dynamic resource assignment model with reference to Slurm freezers i.e. parallel and random ones.

Cores	Cgroups			Parallel Freezer	Random Freezer	Proposed Model
	95-05	80-20	05-95			
8	1.04	1.03	1.34	1.02	1.11	1.07
16	1.12	1.13	1.45	1.08	1.23	1.11
32	1.33	1.29	1.74	1.08	1.21	1.26

Table 1. Impact of total running time

4.2. *Results*

In our proposed model we have recorded test case results and captured the impacted running time of 2 MPI assisted application via usage of two resource managers. We found that rule based on static configuration shows the promising result for high priority only. Results were not promising for the lower priority base. The proposed dynamic assignment model shows comparatively good results. It is also verified with different core allocation and performance found comparable Slurm with different size of clusters. In case of core allocations, count decreased gradually 512, 256,126 as counter for job which gets impacted with a very small deviation in terms of resource utilization. It is also observed that proposed approach gain the better performance in shorter time span compare then Slurm. It's not equipped with user-driven scheduler to control the process. It's scheduling usually managed by the Operating system. The Devops kind of deployment normally configured with user interface for deployment. Besides scheduling of resources our system presented better and enhance computing resource utilization as well as active monitoring of resource usages during variation of workload and there allocated resources. Proposed task scheduler is no longer restricted for high priority job it can be extended to non-hpc or lower priorities jobs

Table 2 is capturing the comparison of Core utilization, job throughput and job partition management between Torque and our proposed high level architecture.

Queue	Make Span (sec)	Core Utilization (%)	Job throughput (@Job/Time [sec])
Torque	11954	60	4.2 * 10-4
Proposed Model	10260	75	1.2 * 10-3

Table 2: Torque vs proposed architecture

4.3. *Discussion*

We have examined results and deduced that our proposed model with resource manager ensures the utilization of resources consistently and it is enhancing the allocation of resources at run time with significate reduction in performance overhead in form the execution time of tasks, deadlines and waiting clock times. DevOps architecture must ensure basic isolation fundamentals for integrating various resource managers within a single cluster which is not stable way of implementation. System must be designed to resilient for system crash and hardware augmentation or removal from main cluster should not impact on overall stability. We can use any third party tool for profiling, pushing metrics data to time series and map the same for public analytics.

5. Conclusion

Container platform that allows creating expansion of agility and package of software in way of portable and reproducible infrastructure on demand self-adaptive approach introduce multiple hurdle in profiling the computing resources and monitoring of metrics. High performance computing is used widely in research areas which facilitate and fueled advancement of scientific computing. In this paper, we have tried to address the shortfall of default workload manager and allocation of resources at run time using the proposed architecture based on resource utilization pattern which can be also leveraged to find out the available resources across nods for the execution of non-HPC application as well. In future, it can be also ensured to save the metrics collection to time series database and provided user resource mapping for consolidating the workflow in containers which can be used to simplify the distribution and replication of scientific results.

References

[1] S. Chen, B. Mulgrew, and P. M. Grant. High Performance Computing (HPC) on AWS. [http://aws.amazon.com/hpc-applications.]
[2] Torque Resource Manager overview and its advantages. [https://adaptivecomputing.com/cherry-services/torque-resource-manager/]
[3] Slurm Overview, its features and how to use it. [https://slurm.schedmd.com/overview.html]
[4] Mesos Overview, its advantages and how it works. [http://mesos.apache.org/documentation/latest/]
[5] Z. Fang, X. H. Sun, Y. Chen, S. Byna, Core aware Memory Access Scheduling Schemes. In IPDPS23,2009.
[6] A Souza, M Rezaei, E Laure, J Tordsson, “Hybrid Resource Management for HPC and Data Intensive Workloads”,2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)
[7] R. M. Tomasul o. An efficient algorithm for exploiting multiple arithmetic units. In IBM Journal ofResearch and Development, Volume 11, Number 1, Page 25, 1967.
[8] D. T. Wang. Modern DRAM Memory Systems Performance Analysis and a High Performance, Power Constrained DRAM Scheduling Algorithm. Ph. D. Dissertation, Dept. Of ECE, University ofMaryland, 2005.
[9] Wang, Gunawan & Maulana, Lazuardi & Leonardi, Nico & Kaburuan, Emil. (2020). The Use of Big Data in Supporting Customer Profiling. International Journal of Advanced Trends in Computer Science and Engineering.9.1128-1133.10.30534/ijatcse/2020/35922020.
[10] K. Jackson et al. Performance Analysis of High-Performance Computing Applications on the Amazon Web Services Cloud. In CloudCom'10, 2010.
[11] Kim, H., El-Khamra, Y., Jha, S., Parashar, M.: Exploring application and infrastructure adaptation on hybrid grid-cloud infrastructure. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, June 21-25, 2010, pp. 402–412 (2010).
[12] Wook, Muslihah. (2020). Big Data Analytics Application Model Based on Data Quality Dimensions and Big Data Traits in Public Sector. International Journal of Advanced Trends in Computer Science and Engineering. 9. 1247-1256. 10.30534/ijatcse/2020/53922020.
[13] S.P. Bingulac, “On the Compatibility of Adaptive Controllers,” Proc. Fourth Ann. Allerton Conf. Circuits and Systems Theory, pp. 8-16, 1994. (Conference proceedings).
[14] H. Goto, Y. Hasegawa, and M. Tanaka, “Efficient Scheduling Focusing on the Duality of MPL Representation,” Proc. IEEE Symp. Computational Intelligence in Scheduling (SCIS ’07), pp. 57-64, Apr. 2007, doi:10.1109/SCIS.2007.367670. (Conference proceedings)
[15] Nie, L., Xu, Z.: An adaptive scheduling mechanism for elastic grid computing. In: International Conference on Semantics, Knowledge and Grid, pp. 184–191 (2009).
[16] Feitelson D. G.: Parallel workloads archive (PWA), February 2012
[17] [http://www.cs.huji.ac.il/labs/parallel/workload/]
[18] Feitelson D. G., Weil A. M.: Utilization and predictability in scheduling the IBM SP2 with backfilling. [in:] 12th International Parallel Processing Symposium, pages 542–546. IEEE, 1998

[19] Kleban S. D., Clearwater S. H.: Fair share on high performance computing systems: What does fair really mean? [in:] Third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03), pp. 146–153. IEEE Computer Society, 2003.
[20] Klus´a˘cek D., Rudov´a H., Baraglia R., Pasquali M., Capannini G.: Comparison of multi-criteria scheduling techniques. [in:] Grid Computing Achievements and Prospects, pp. 173–184. Springer, 2008
[21] SubW., Jakob W., Quinte A., Stucky K.-U.: GORBA: A global optimising resource broker embedded in a Grid resource management system. [in:] International Conference on Parallel and Distributed Computing Systems, PDCS 2005, pp. 19–24. IASTED/ACTA Press, 2005
[22] Xhafa F., Abraham A.: Metaheuristics for Scheduling in Distributed Computing Environments, volume 146 of Studies in Computational Intelligence. Springer, 2008.
[23] Xhafa F., Abraham A.: Computational models and heuristic methods for Grid scheduling problems. Future Generation Computer Systems, 26(4):608–621, 2010.
[24] Xu M. Q.: Effective metacomputing using LSF multicluster. [in:] CCGRID '01: Proceedings of the 1st International Symposium on Cluster Computing and the Grid, pp. 100–105. IEEE, 2001.
[25] Kurowski K., Oleksiak A., Piatek W., Weglarz J.: Hierarchical scheduling strategies for parallel tasks and advance reservations in grids. Journal of Scheduling, 11(14):1–20, 2011. C. D. Locke, "Best-effort Decision-making for Real-time Scheduling," Ph.D. dissertation, Pittsburgh, PA, USA, 1986, aAI8702895.
[26] P. Li and B. Ravindran, "Fast, Best-Effort Real-Time Scheduling Algorithms," IEEE Trans. Comput., vol. 53, no. 9, pp. 1159–1175, 2004.
[27] Abhishek, Manish. (2020). Dynamic Allocation of High Performance Computing Resources. International Journal of Advanced Trends in Computer Science and Engineering. 9. 3538-3543. 10.30534/ijatcse/2020/159932020.
[28] N. Bansal and K. R. Pruhs, "Server Scheduling to Balance Priorities, Fairness, and Average Quality of Service," SIAM J. Comput., vol. 39, no. 7, pp. 3311–3335, 2010.
[29] S. Aldarmi and A. Burns, "Dynamic value-density for scheduling realtime systems," in Proceedings of the Euromicro Conference on RealTime Systems, 1999, pp. 270–277.

Authors Profile

.

Manish Kumar Abhishek holds a Master's degree in Software System from BITS Pilani and currently pursuing Ph.D. from Koneru Lakshmaiah Education Foundation in High Performance Computing consolidation using cloud computing. He has more than 10 years of experience in the IT industry that involves High Performance Computing, Super Computing and Cloud computing services offerings with OpenStack, Google, AWS and Data Centre infrastructure deployment architecture design and security. In the past, he has worked with Center for Development of Advanced Computing (C-DAC), Tata Consultancy Services(TCS) and currently leading Data Centre Infrastructure and Cloud team as a Senior Manager @ Railtel Corporation of India Ltd (under Ministry of Railway India). He has authored multiple articles in IEEE publications and Scopus index journals.

D. Rajeswara Rao is currently working as a professor in the Department of Computer Science Department at Koneru Lakshmaiah Education Foundation. He received the Ph.D. (Computer Science and Engineering) from Jawaharlal Nehru Technological University, Hyderabad. He has twenty two years plus experience with various research interests including machine learning, soft computer techniques. He has awarded and honoured for his academic profile. He has authored multiple articles in IEEE publications and Scopus index journals. He is a member of ACM, life member of Computer Society of India and Indian Society of Technical Education. He has two published patents.

Dr. K. Subrahmanyam is a Gold Medalist from Andhra University (1992-93). He is currently working as a Professor in Computer Science & Engineering, Principal-College of Sciences and OSD to Chancellor, K L Deemed to be University, Vaddeswaram, Guntur. He is in teaching profession for the past twenty eight years and prior to joining KLEF he worked as Programmer Leader in the School of Engineering, Science & Technology at KDU University, Malaysia for about 10 years. He is the Founder Chairman of ACM Amaravati Chapter and an active member of other professional societies like CSI, IEEE & CSTA. He has obtained several research grants from Government organizations like AICTE, DST-INSPIRE, DST-SERB, DST-SEED and has completed numerous consultancy works for various NGOs.