Context Switching In Clouds: Analysis And Enhancements

Abhinav Mishra
Department of Computer Science & Engineering,
Guru Nanak Dev University,
Amritsar, Punjab, 143001, India

Anil Kumar
Department of Computer Science & Engineering,
Guru Nanak Dev University,
Amritsar, Punjab, 143001, India

Abstract— This paper presents an methodology to improve the context switch overhead in cloud computing. As migration of jobs in the cloud makes context switching overhead a challenging task which increases the communication costs of cloud and results in the low utilisation. To succeed this objective, dynamic context switching parallel algorithm is purposed in this paper. In order to enhance the utilisation and computing speeds of cloud an improved strategy is used which uses dynamic context switching parallel algorithm. Purposed methodology makes changes in the decision making process of the context switch parallel algorithm dynamically considerably reducing the context switch overheads. For execution of purposed approach, a cloud simulation environment is established which significantly reduces the waiting time of jobs during execution.

Index Terms—Multiprogramming, Load balancing, Context switching, Job migration, Optimal switching.

I. INTRODUCTION

Cloud computing refers to the model of computing as a utility, just like water and electricity, where users can have access to vast computational resources based on their requirements. In fact Clouds offer the ability to utilize any type of software, either existing or custom-made. The cloud computing paradigm promises a cost-effective solution for running business applications through the use of virtualization technologies, highly scalable distributed computing and data management techniques as well as a Pay-As-You-Go (PAYG) pricing model. In recent years, it also offers high performance computing capacity for applications to solve complex problems.

Improving resource utilization is essential for achieving cost-effectiveness. Low utilization has long been an issue in datacentres. For a datacentre, or a subset of servers in a datacentre that mainly handles applications with high performance computing needs and runs parallel jobs most of the time, the problem can be significant. Parallel applications of certain nature often show a decreasing utilization of CPU resources as parallelism grows, mainly because of the communication and synchronization among parallel processes. It is challenging but important for a datacentre to achieve a certain level of utilization of its nodes while maintaining the level of responsiveness of parallel jobs. There are various performance issues also which degrade the utilization of such kind of environments. Various factors that contribute towards the low utilization as well as high costs like context switch in job migration, scheduling overhead, network driver overhead and may more which lead to increase in the communication costs.

The importance of cloud computing lies in the opportunity for small companies and organizations to have access to computing infrastructure without the need for prior investment. Therefore ideas that would have required major capital investment can be implemented on the cloud with minimal costs and reduced risk. It quickly becomes obvious that Cloud computing can also be applied in the area of HPC (High Performance Computing). Small institutions and individual scientific teams can now have access to large computational resources not only through the Grid, with all its restrictions and limitations, but also through the Cloud which provides an infrastructure platform with virtually infinite resources at a fraction of the cost of maintaining a private cluster, and on a flexible pay-per-use model. Cloud computing providers offer their services according to three fundamental models: Infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SAAS).The flexibility and scalability of commercial cloud infrastructures make them an attractive application migration target; however, due to the associated complexity, it is difficult to make newly migrated applications run efficiently in clouds. Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet) [1]. The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation.

Infrastructure as a service (IaaS)

In this most basic cloud service model, providers offer computers, as physical or more often as virtual machines, and other resources. The virtual machines are run as guests by a hypervisor, such as Xen or KVM. Management of pools of hypervisors by the cloud operational support system leads to the ability to scale to support a large number of virtual machines. Other resources in IaaS clouds include images in a virtual machine image library, raw (block) and file-based storage, firewalls, load balancers, IP addresses, virtual local area networks (VLANs), and software bundles. Examples: Amazon cloud formation, Rackspace cloud, Google compute engine.

Platform as a service (PaaS)

In the PaaS model, cloud providers deliver a computing platform typically including operating system, programming language execution environment, database, and web server [1]. Application developers can develop and run their

software solutions on a cloud platform without the cost and complexity of buying and managing the underlying hardware and software layers. Example: Cloud foundry, Google App. Engine.

Software as a service (SaaS)

In this model, cloud providers install and operate application software in the cloud and cloud users access the software from cloud clients. The cloud users do not manage the cloud infrastructure and platform on which the application is running. This eliminates the need to install and run the application on the cloud user's own computers simplifying maintenance and support. Examples: Microsoft office 365, Onlive.

II. PROBLEM DEFINITION

There are various factors that contribute towards the low utilization as well as high costs like context switch overhead, scheduling overhead, network driver overhead and may more which lead to increase in the communication costs. Main emphasis of this research is to use the approaches which can reduce context switching overhead using dynamic context switching parallel algorithm.All over the world scientists and researchers are trying to increase the utilization of CPU as the parallelism grows. They are trying to improve these overheads that are decreasing the speed as well as the utilization CPU and various resources that affect the computing speeds of clouds. In random context switching during the job migration in cloud reduces the utilisation of CPU and increase communication costs. To overcome this problem, in this research work dynamic context switching strategy has been assimilated with parallel algorithms.

III. LITERATURE REVIEW

The application of Gang scheduling on a Cloud Computing model, based on the architecture of the Amazon Elastic Compute Cloud (EC2). The study takes into consideration both performance and cost while integrating mechanisms for job migration and handling of job starvation [3]. The number of Virtual Machines (VMs) available at any moment is dynamic and scales according to the demands of the jobs being serviced. The aforementioned model is studied through simulation in order to analyse the performance and overall cost of Gang Scheduling with migrations and starvation handling. Results highlight that this scheduling strategy can be effectively deployed on Clouds, and that cloud platforms can be viable for HPC or high performance enterprise applications. Use of gang scheduling in which jobs consist of tasks that must be scheduled to run simultaneously and concurrently since they are in frequent communication with each other avoids possible bottlenecks or deadlocks caused by tasks waiting for an input from other tasks that are not running. This paper integrates both a migration mechanism and a starvation handling system in the model and compares the overall performance and cost efficiency of the model. Migration system along with starvation handling works very efficiently and maintains low response time under low to medium arrival rates. LJFS is more cost efficient for higher arrival rates under all job size coefficients when compared to AFCFS. In this paper we also found out some gaps like the context switching in gang scheduling involves high overhead, therefore the jobs always execute to completion, and cannot be pre-empted. Migration algorithms and starvation handling in low to medium parallelism jobs shows a slowdown i.e. the delay suffered by a job relative to its service time. The migration process involves transferring of tasks from queues of busy VM's to the heads of queues of idle VM's which are available. It causes large overheads in the system.In a large multiprocessor server platform, using multicore chips, the scheduler often migrates a scheduling entity, i.e. a thread or process or virtual machine, in order to achieve better load balancing or ensure fairness. The migration impact is likely to be more severe in virtualized environments, where high over-subscription of logical CPUs is very common for server consolidation workloads or virtual desktop infrastructure deployment. This paper demonstrates the performance benefit of saving and restoring cached data during migration. A lot of research has been done in understanding the overhead of context switching in a computer system. Context switching introduces two types of overheads in computation. Direct overhead is the cost involved in saving and restoring processor registers, flushing processor pipelines and TLBs, and executing the operating system scheduler to deschedule the current task, select the next task to run, and update relevant task structures so the actual computation begins again [6]. Secondary overhead or performance impact of context switching comes from increased cache misses (both data and instruction) and TLB misses (both data and instruction), as well as inaccuracies introduced into branch predictions and prefetches. The migration of a process/thread from one CPU to a different CPU further magnifies the secondary overhead incurred in a context switch. They observed that the indirect cost of context switch varies with different workloads with different memory access behaviour and for different architectures. Though the overhead of moving migration messages and extra unused prefetches might limit the performance gain somewhat and can limit such effect by using enhanced tags with process/VM identification info, which is left as future work. The future work will also extend the current implementation to deal with the case when migration occurs after a process/VM has been descheduled for a while. For such migrations, tags need to be stored first and

prefetched later when the process/VM is rescheduled. This may also be applicable in reducing non-migratory context switch misses.

IV. LIFE CYCLE OF PARALLEL JOBS.

Fig. 1 is showing the life cycle of parallel jobs in distributed environment. Firstly clients submits their jobs for execution, ifno subcloud is free then jobs are queued after this phase node filtering will be done that will detect the currently active nodes by checking the status of all nodes. After node filtering load balancing algorithm will come into action to balance the load of thegiven jobs among active nodes. It also shows that the main source of parallelism is provided by the load balancer, whose role is to split each job into multiple subsets that can be executed on multiple nodes in parallel. After successful execution of threads, each node send its final results back to the sub cloud, then subcloud conquer the results of threads and the output will be passed to the client.

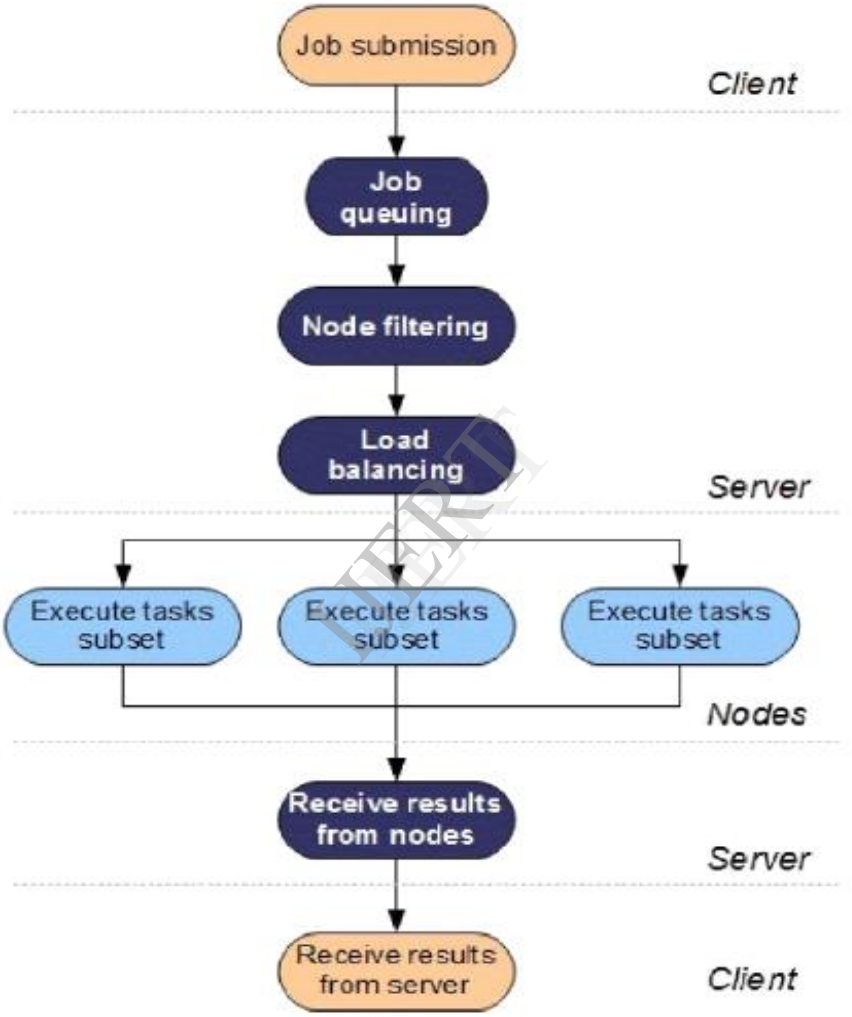


Fig.1. Life Cycle of Parallel Jobs.

V. PURPOSED DYNAMIC CONTEXT SWITCH STRATEGY

In order to reduce the context switch overhead for cloud using dynamic context switch approach an algorithm has been proposed in this research work. Dynamic context switch is defined as the feasible allocation or scheduling of the jobs in such a way that context switches are reduced and so that the waiting time can be reduced. This section discusses the procedure that how dynamic context switch algorithm works.

Proposed dynamic context switch algorithm

Proposed dynamic context switch algorithm is developed considering main characteristics like reliability, performance, throughput and resource utilization. However to fulfil these requirements of reducing context switch overheads, in fig 2 dynamic context switch algorithm is shown.

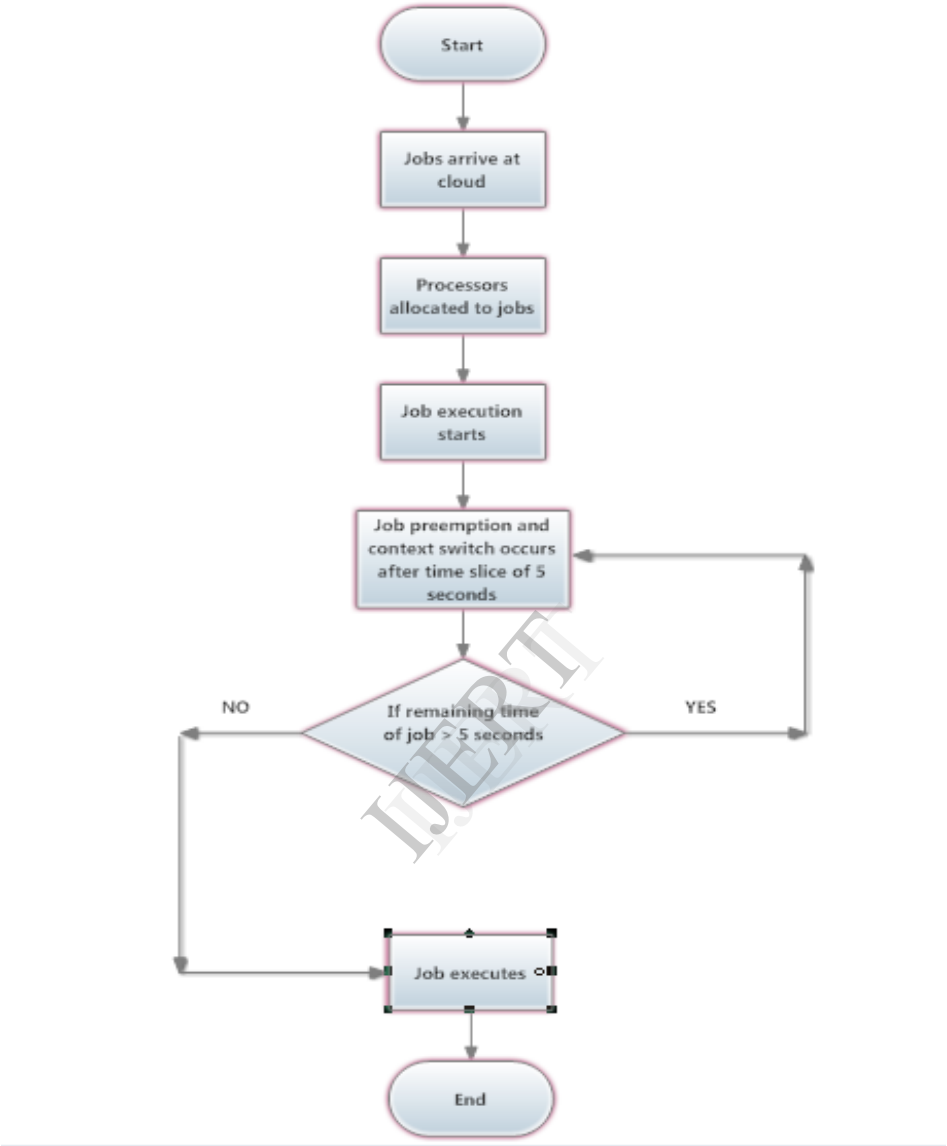


Fig 2. Dynamic context switch algorithm flowchart

The flowchart of dynamic context switch algorithm is shown in Fig 2.that shows how algorithm will work? It will take the following steps to assign the cloud to the requests of the clients:

Step 1: Firstly the jobs are submitted at the cloud by the clients.

Step 2: Cloud will allocate the jobs to the processors on the basis of first come first serve basis.

Step 3: The execution of jobs start and the jobs are prompted after a time slice of 5 seconds and context switch occurs as the job is allocated to different processor.

Step 4: By checking the remaining time of job cloud will check the remaining time of job. If remaining time of job is greater than 5 seconds the job prempts and again context switch occurs and another job is allocated to the processor.

Step 5: If remaining time of job is less than equal to 5 sec then no context switch occurs and the same job is executed by the processor, this reduces the context switch and average waiting time is reduced and throughput increases.

VII. SIMULATION RESULTS

Table 1 gives the inputs that are given to the simulator. In Table 1 various jobs are given with their burst time. The burst time means the execution time required for a job to complete. The jobs will run in a parallel fashion in the simulator and context switching will occur when jobs are subjected to run on the simulator.

Job Name	Burst Time
J1	21
J2	22
J3	20
J4	23
…	…
J100	45

Table 1. Inputs given to simulator

Designed simulator makes all the jobs to get ready for execution in the cloud. Fig 3.giving the detail of which job is going to run and also the other relevant information like burst time, remaining time, status of the job, context switch frequency etc. It allocates the jobs to the cloud in FCFS fashion. When the network is activated it firstly executes the jobs using the classical multiprogramming approach.

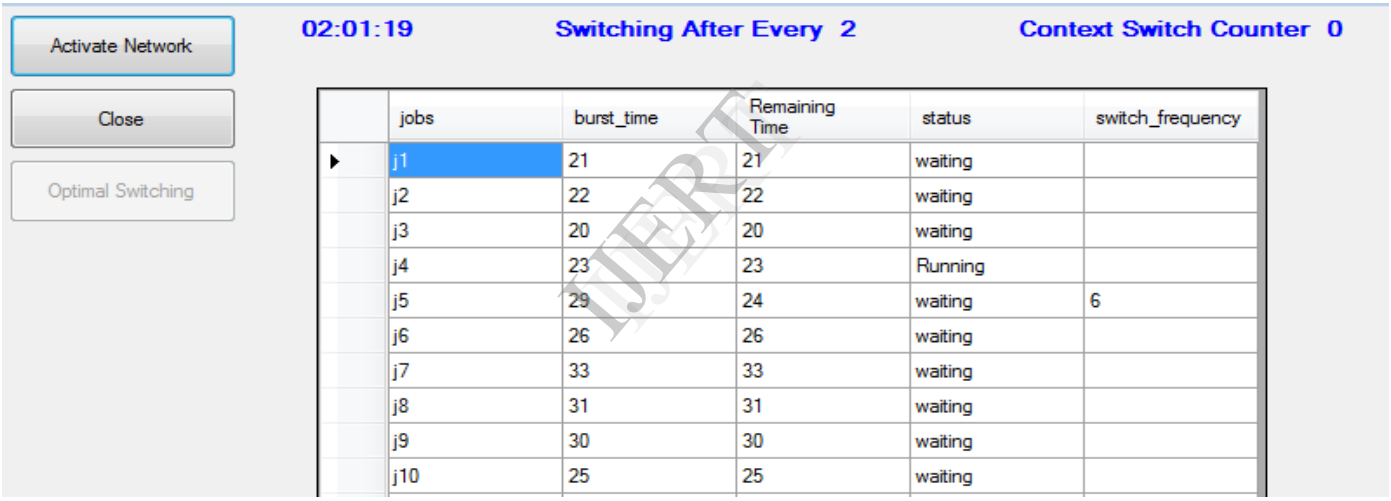


Fig 3. Classical approach

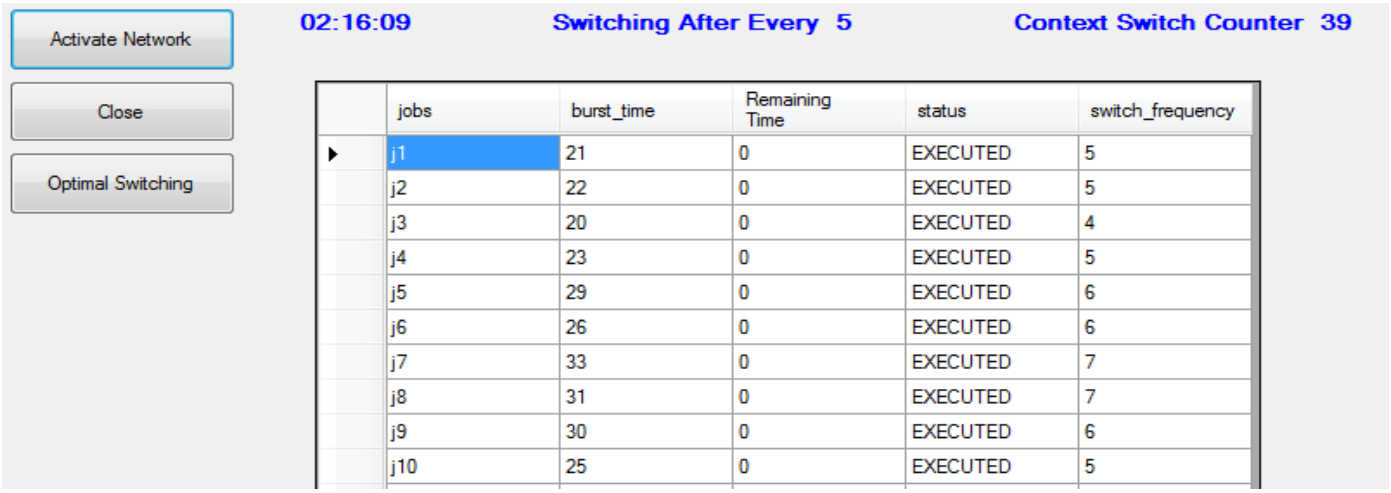
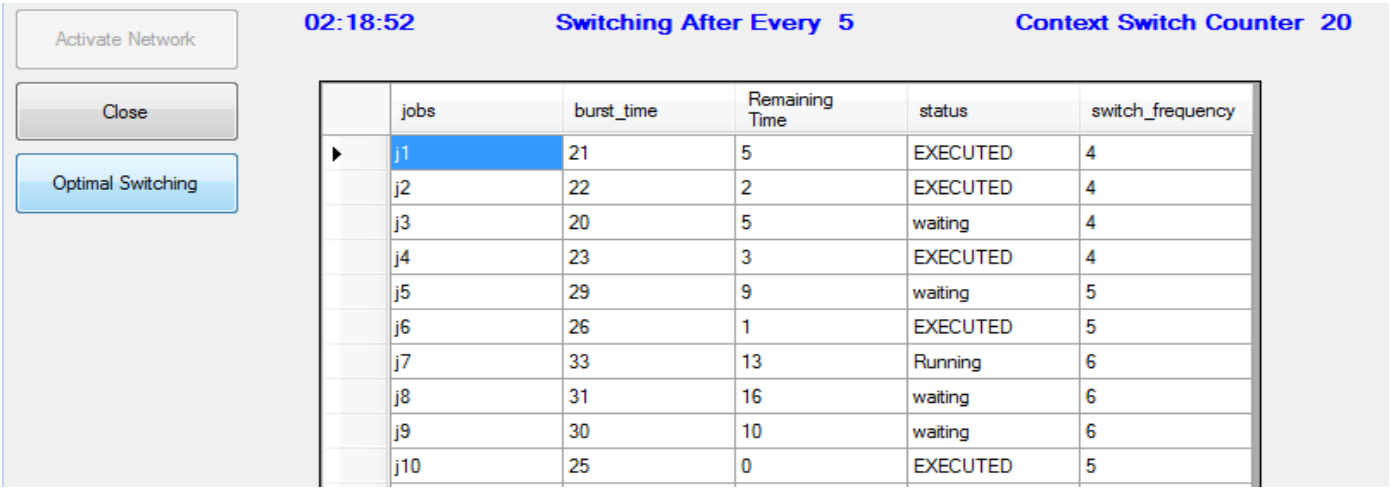


Fig 4.Jobs executed using Classical approach

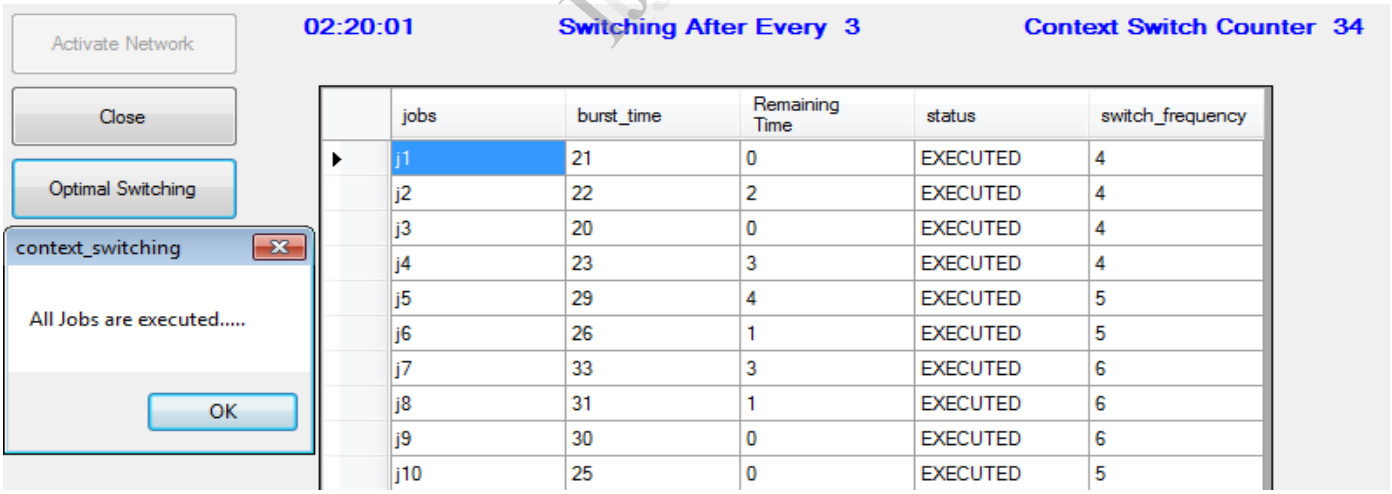
Fig 4 showing all the jobs get executed using the classical multiprogramming approach. Fig 4 giving the detail about the results obtained using this approach. It gives details about the context switching frequency done to execute all the jobs successfully.



jobs	burst_time	Remaining Time	status	switch_frequency
j1	21	5	EXECUTED	4
j2	22	2	EXECUTED	4
j3	20	5	waiting	4
j4	23	3	EXECUTED	4
j5	29	9	waiting	5
j6	26	1	EXECUTED	5
j7	33	13	Running	6
j8	31	16	waiting	6
j9	30	10	waiting	6
j10	25	0	EXECUTED	5

Fig 5. Optimal Switching using Dynamic context switch parallel algorithm

Fig 5 shows the optimal switching using dynamic context switch parallel algorithm. The simulator first obtains the results by using the classical approach then by activating the optimal switching button on the simulator again the execution of jobs starts using our purposed dynamic context switch parallel algorithm. Fig 5 gives details about the jobs in waiting state, running state and the jobs being executed. It also gives details about various parameters calculated during job execution.



jobs	burst_time	Remaining Time	status	switch_frequency
j1	21	0	EXECUTED	4
j2	22	2	EXECUTED	4
j3	20	0	EXECUTED	4
j4	23	3	EXECUTED	4
j5	29	4	EXECUTED	5
j6	26	1	EXECUTED	5
j7	33	3	EXECUTED	6
j8	31	1	EXECUTED	6
j9	30	0	EXECUTED	6
j10	25	0	EXECUTED	5

Fig 6. Jobs executed using dynamic context switch parallel algorithm

Fig 6 shows all the jobs successfully completed execution using our purposed dynamic context switch parallel algorithm. After executing all jobs a message is being displayed. As seen from the figure the context switch frequency is reduced when compared with the fig 4.which uses the classical multiprogramming approach. Using these results context switch frequency, throughput and average waiting time is been calculated and compared with the classical approaches in the next section.

VIII. PERFORMANCE ANALYSIS

In order to do performance analysis a comparison table has been made for this research work. This section first give the performance comparison of developed simulator with existing methods and later on comparison of different approaches is made using different performance metrics.

Type of Metric	Classical Approach	Integrated Optimal Switching
Average Context Switch Frequency	33	28
Average Waiting Time	4.83	3.87
Throughput(THP)(at context switch counter 18)	49	66

Table 2. Metric's comparison of different approaches

Table 2 is showing the performance comparison of different approaches. These approaches are the classical multiprogramming approach and dynamic context switch parallel algorithm (Purposed technique). It has been clearly shown in Table II that purposed method gives better results than other methods. As in purposed technique we get better results in terms of average context switch frequency, average waiting time and throughput.

Fig 7.illustrates the graph of Average Waiting (AWT). Therefore it is clearly shown that the purposed method gives better results than existing methods as AWT of integrated approach always stay lower than the other existing methods lines.

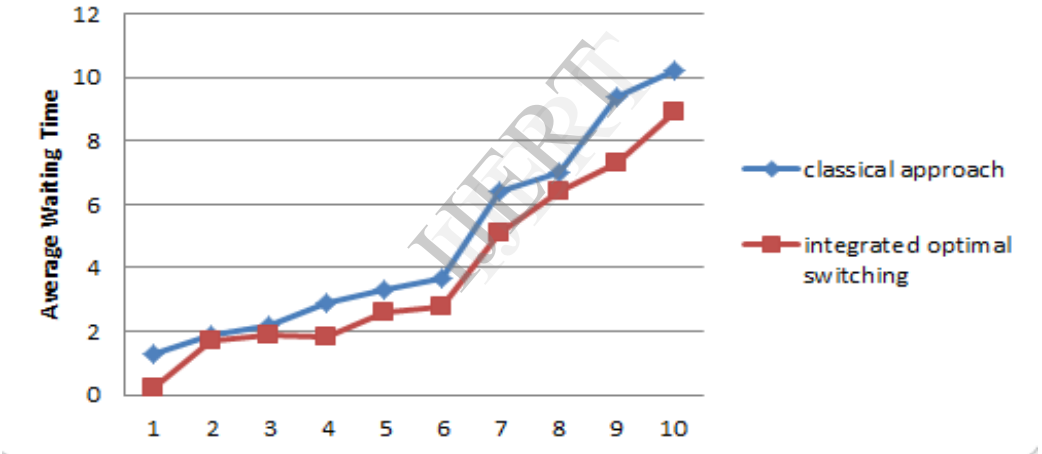


Fig 7.Average Waiting Time comparison with existing method.

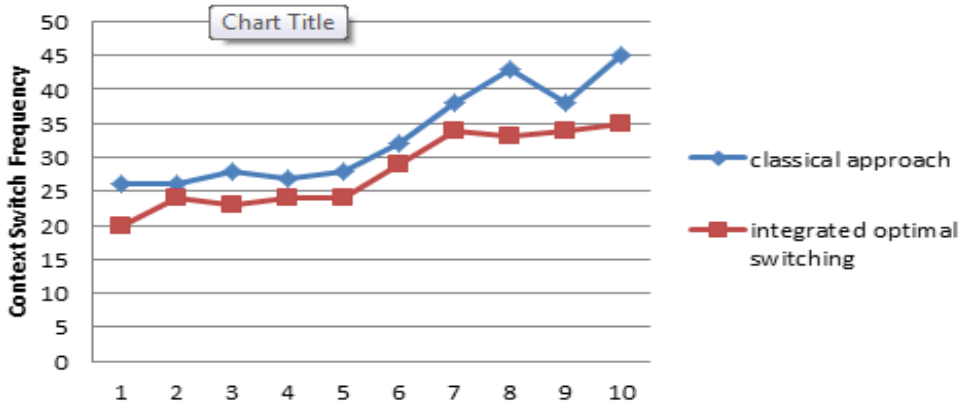


Fig 8.Average Context Switch Frequency comparison with existing method.

Fig 8.demonstrates the diagram of Average Context Switch Frequency (ACSF) metric. Consequently it is undoubtedly revealed that the purposed technique contributes improved fallouts than prevailing approaches as ACSF in incorporated optimal context switching methodologies line continuously vacation subordinate than the supplementary techniques lines.

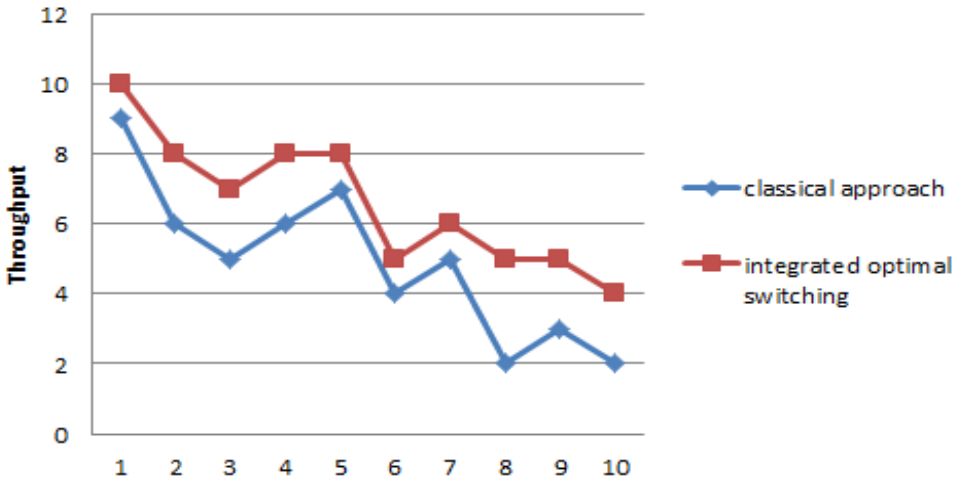


Fig 9.Throughput comparison with existing methods

Fig 9.exhibits the diagram of Throughput (THP) metric.Accordingly it is unquestionably exposed that the purposed technique donates better-quality fallouts than predominant methodologies as THP in incorporated optimal context switching methodologies line continuously vacation subordinate than the supplementary techniques lines.

IX. CONCLUSION AND FUTURE DIRECTIONS

This paper proposes a smart optimal switching strategy for cloud computing using integrated optimal switching algorithms, which include the support of dynamic context switch parallel algorithms. A simulator environment has been developed that implement the purposed method. Performance comparison of existing methods has been made with the purposed method. It has been concluded with the help of performance metric's comparison that the proposed optimal switching strategy gives good results than existing methods.
In this paper context switch overhead paradigm has been considered for simulation environment, in future work we will work on minimising the other communication and job migration overheads for better results.

REFERENCES

[1] Performance and Cost evaluation of Gang Scheduling in a Cloud Computing System with Job Migrations and Starvation Handling(IEEE 2011); Authors: Ioannis A. Moschakis and Helen D. Karatza.
[2] Priority-based Consolidation of Parallel Workloads in the Cloud (IEEE 2012); Authors: Xiaocheng Liu, Chen Wang, Bing Bing Zhou, Junliang Chen, Ting Yang, and Albert Y. Zomaya.
[3] Variations in Performance and Scalability when Migrating n-Tier Applications to Different Clouds (CLOUDS 2011); Authors: DeepalJayasinghe, Simon Malkowski, Qingyang Wang, Jack Li, PengchengXiong, and CaltonPu.
[4] Reducing Migration-induced Cache Misses (IEEE 2012); Authors: Sajjid Reza and Gregory T. Byrd.
[5] Reese, G., "Cloud Application Architectures: Building Applications and Infrastructure in the cloud (Theory in Practice)", O,,Reilly Media, 1st Ed., 2009.
[6] K. StanoevskaSlabeva, T. W. S. Ristol, "Grid and cloud Computing and Applications, A Business Perspective on Technology," 1st Ed., 2004.
[7] Y. J. Wen, S. D. Wang, "Minimizing Migration on Grid Environments: An Experience on Sun Grid Engine," National Taiwan University, Taipei, Taiwan Journal of Information Technology and Applications, March, 2007.
[8] Evaluating Overheads of Integrated Multilevel Checkpointing Algorithms in Cloud Computing Environment
Dilbag Singh, Jaswinder Singh, AmitChhabra. I. J. Computer Network and Information Security, 2012, 5, 29-38, June 2012.
[9] An Adaptive Algorithm For Dynamic Priority Based Virtual Machine Scheduling In Cloud, Subramanian S, Nitish Krishna G, Kiran Kumar M, Sreesh P and G R Karpagam. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 6, No 2, November 2012, ISSN (Online): 1694-0814.

[10] Failures in Cloud Computing Data Centers in 3-tier Cloud Architecture, DilbagSingh,JaswinderSingh,Amitchhabra, I.J. Information Engineering and Electronic Business, 2012, 3, 1-8 Published Online July 2012 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijieeb.2012.03.01.

[11] An Optimistic Differentiated Service Job Scheduling System for Cloud Computing Service Users and Providers, Luqun Li, 2009 IEEE, DOI 10.1109/MUE.2009.58.

[12] Understanding Performance Interference of I/O Workload in Virtualized Cloud Environments, Xing Pu, Ling Liu, YiduoMei, SankaranSivathanu, YounggyunKoh, CaltonPu, 2010 IEEE DOI 10.1109/CLOUD.2010.65.