Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 8, August 2016, pg.155 – 161

An Improved Shortest Job First Scheduling Algorithm to Decrease Starvation in Cloud Computing

[1]Sumit Kumar Nager, [2]Nasib Singh Gill

[1]Department of Computer Science & Applications, Maharshi Dayanand University, Rohtak, Haryana, India
[2]Department of Computer Science & Applications, Maharshi Dayanand University, Rohtak, Haryana, India
sumitnager18@gmail.com, nasibsgill@gmail.com

Abstract: The introduced work is a change in the non-preemptive existing scheduling algorithm called SJF (Shortest Job First) in a cloud computing. The cloud computing is steadily retrieved by miscellaneous users. On the other hand as the amount over the server increments, in such instance to achieve the operational account time some scheduling algorithm is mandatory. Also, there is the prerequisite of an operational schedule that will designate the process for implementation from the queue. One of the operational scheduling techniques is SJF, but this tactic surges the starvation for small priority processes or the processes with enormous execution time. The introduced work is about to resolve all these difficulties and to decrease the average waiting time.

Keywords: SJF, Scheduling, Cloud, resource managing, waiting time, starvation, LIFO, FIFO

I. INTRODUCTION

Cloud computing is at this time one of the fastest developing and supreme popular areas of Computer Science and Information Technology. Contrasted with the conventional computing model, the cloud offers numerous advantages as far as execution, adaptability, consistency, profitability and independence is concerned. To the extent that the end client is concerned, cloud computing does not necessitate him to deal with the structure at all but involves a service providing possessions such as storage, processing units, networks, and applications. The delivered possessions can be personalized to his requirements and can also be distributed to other clients, thus make the best use of productivity. Apart from the technical community, cloud computing is also a very solid instrument for financial, accounting, management and numerous other applications. Also, entertainment is progressively emerging cloud features, with content uploaded and prepared to be shared from corner to corner on numerous diverse devices. In maximum circumstances, particularly in the commercial ones, the delivered classes of facilities are very serious.

Cloud computing offers resources to clients through virtualization technology. It condenses the coupling among the software and the hardware, and significantly increase the deployment of the resources [5]. Clouds practice virtualization technology in scattered data centers to assign resources to clients as the require them [6]. In Cloud platform Virtualization technology is capable of achieving remapping among virtual machines (VM) and physical resources according to the load alteration so as to attain the load equilibrium of the entire system in a dynamic

method [7]. Load balancing has two implications: first, it places a great number of parallel accesses or data traffic to numerous nodes respectively to decrease the time clients waiting for the reply; second, it places the calculation from a solitary weighty load to the numerous nodes to expand the resource deployment of all node [5]. Load Balancing is accomplished by the prioritizing list of data centers and client trust. Once Load balancing is started, the listing of trusted and un-trusted data centers nodes is done. Trusted list contains nodes having trust value larger than the threshold value in diminishing order i.e. the first node of a list has the maximum trust value. Similarly, untrusted node list contains the node with trust value fewer than the threshold value in diminishing order [12].

The fundamental purpose of using a Cloud system is to enhance the performance of important application programs, great scalability, and fault tolerance. In this way, it is fundamental to keep the preserve the cost of making and handling parallelism as little as allowed. As for scheduling in a Cloud system, there are three objectives to take down the cost:

- Good processor utilization: all processors have work in their queues at all times. All processors, which have tasks allocated to them from the identical program finish execution at the same time, therefore the client acquires the predicted speedup. Processors occupy the maximum of their time performing valuable work rather than managing the division of work.
- Good synchronization effectiveness: Tasks are scheduled in a manner that interrelating tasks diagonally with fine-grained communication should be running at the same time.
- Low communication/memory-access cost: Tasks are scheduled in a manner that communication time, either message passing or shared memory latency is accounted for, and minimized. Scheduling data configurations should be organized so that they won't be any argument.

Scheduling mechanism is the highest significant component of a computer system. Scheduling is the policy by which the system chooses which task should be accomplished at any certain time. There is dissimilarity between real-time scheduling and multiprogramming timesharing scheduling. It is because of the role of timing restrictions in the assessment of the system performance.

II. WRITINGS EXAMINATION

In In Year 2013, Lichen Weng worked on," Scheduling Optimization in Multicore Multithreaded Microprocessors through Dynamic Modelling". This paper portrays those outline through three steps: in the initial step, Author changes over a static scheduling arrangement into a dynamic one, which assesses the thread mapping design at runtime. In the second step, Author utilizes relapse models to guarantee that the scheduling arrangement is fit for reacting to the changing practices of thread amid execution. In the last stride, Author confines the overhead of the proposed arrangement by embracing a heuristic methodology, hence guarantee the adaptability with the exponential development of core and thread tallies [3].

In Year 2013, Vishakha Gupta performed a work," Kinship: Efficient Resource Management for Performance and Functionally Asymmetric Platforms". The author suggests that systems software must advance to dynamically accommodate heterogeneity and to automatically elect task-to-resource mappings to best utilize these features. Author define the kinship method for mapping workloads to heterogeneous cores. A hypervisor-level realization of the method on a diversity of experimental heterogeneous platforms determines the common applicability and usefulness of kinship-based scheduling, matching dynamic workloads to accessible resources as well as scaling with the number of processes and with dissimilar types/configurations of computing resources [4].

In Year 2011, Viswanath Krishnamurthy performed a work," A Novel Thread Scheduler Design for Polymorphic Embedded Systems". In this paper, Author addresses the challenge of coming up with a design for an effective scheduler for Multiple Application and Multi-threaded polymorphic embedded system with clients satisfaction as its objective function. Randomly generated application graphs aid as standards to estimate the performance of the proposed polymorphic scheduler computing. The author also talks over the impression of Presented scheduler on the client's satisfaction of a multimedia application as a real world case study [10].

In Year 2012, Hiroshi Sasaki performed a work," Scalability-Based Manycore Partitioning". This development brings us a state where a number of parallel applications concurrently being executed on a single system. Since multi-threaded applications try to maximize its throughput by utilizing the whole system, every one of them usually produces equal or larger number of threads compared to underlying logical core counts. Author develop a sophisticated scheduler that (1) dynamically predicts the scalability of programs through the use of hardware performance monitoring elements, (2) chooses the ideal number of cores to be assigned to every program, and (3) assigns the cores to programs while maximizing the system utilization to attain fair and maximum performance[9].

In Year 2012, Aamer Jaleel performed a work," CRUISE: Cache Replacement and Utility-aware Scheduling". Author find that smart cache replacement decreases the load on software to deliver intelligent scheduling results. However, underneath smart cache replacement, there is still room to increase performance from better application

co-scheduling. Author find that co-scheduling decisions are a function of the fundamental LLC replacement policy. The author suggests Cache Replacement and Utility-aware Scheduling (CRUISE)—a hardware/software co-designed approach for shared cache management. For 4-core and 8-core CMPs, Author finds that CRUISE approaches the performance of an ideal job co-scheduling policy under different LLC replacement policies [8].
In Year 2010, Jonathan A. Winter performed a work," Scalable Thread Scheduling and Global Power Management for Heterogeneous Many-Core Architectures". This paper presents a range of scheduling and power management algorithms and performs a detailed evaluation of their effectiveness and scalability on heterogeneous many-core architectures with up to 256 cores. The author also conducts a limit study on the potential benefits of coordinating scheduling and power management and demonstrate that coordination yields little benefit. Author highlight the scalability limitations of previously proposed thread scheduling algorithms that were designed for small-scale chip multiprocessors and propose a Hierarchical Hungarian Scheduling Algorithm that dramatically reduces the scheduling overhead without loss of accuracy[1].
In Year 2012, Jun Liu performed a work," A Compiler Framework for Extracting Superword Level Parallelism". In this paper, Author proposes a novel automated compiler framework for improving super word level parallelism exploitation. The key part of Presented framework consists of two stages: super word statement generation and data layout optimization. The first stage is Presented the main contribution and has two phases, statement grouping and statement scheduling, of which the primary goals are to increase SIMD parallelism and, more importantly, capture more super word reuses among the super word statements through global data access and reuse pattern analysis [2].

III. PROPOSED WORK

The presented work is an advancement over the nonpreemptive existing scheduling algorithm called SJF (Shortest Job First) in a cloud environment. The cloud environment is the most common distributed system, in which numerous clients are associated with the system and the parallel requests are accomplished on the system. In such circumstance, there is the necessity of an effective scheduler that will designate processes for accomplishment from the queue. One of the effective scheduling practice is SJF, but this method rises the starvation for low priority processes or the processes with higher execution time. The presented work is to resolve all these complications and to diminish the waiting time processes.

A. Problem Definition

In distributed systems, numerous clients are associated with the system and send the several requests to the system for the resource allotment. A standout amongst such distributed system is the cloud environment. A cloud environment can be an incorporated or circulated design with numerous servers and the CPUs. In such events when the numerous requests are being made on such systems, there is the necessity to accomplish these processes or the requirements to assign the essential resources or the facilities to the clients as per requisite. Now, the problem arises in an encumbered system, where the quantity of requirements is tremendously greater than the quantity of existing CPUs on the server side. In such events, the question arises, which process will be offered first to the CPU for the execution. There are various prevailing scheduling methodologies to answer the question. A standout amongst such methodology is Shortest Job First. But in a cloud environment, this scheduling arrangement grieves from various shortcomings such as the waiting time and the starvation problem. The presented work is to deliver a resolution of this problem by giving an enhanced SJF algorithm to provide effective scheduling in the cloud environment.

B. Goals

The introduced research work is covering the accompanying examination destinations

- The fundamental goal of the work is to outline a powerful procedure scheduling algorithm for cloud environment so that the waiting time of the procedures gets lessened and the starvation is stays away from.
- The goal of the work is to outline an easy to use cloud environment to acknowledge the client demands with particular parameters in a successful way.
- The goal of the work is to characterize a parametric investigation of proposed algorithm with existing SJF regarding waiting time.
- The primary goal of work is to diminish the waiting time of the processes and reduction the odds of starvation.

C. Impact of Work

The presented work is an enhancement on the scheduling algorithm in cloud System environment. As the process will be scheduled in the more effective method it will provide a number of aids. The foremost advantage of the work, the decrease in the wait time of processes. Each process, demand or the client need its execution rapidly

without any delay. But in events of overburdening situations, the processes of greater priority get accomplished first and the small importance processes halt in a queue. The presented work is to diminish the wait time. The wait time decrease will also rise the robustness of the system along with the advancement of the dependability of the system.

D. Research Methodology

The most important challenge in distributed Cloud environment is the process arrangement. When the numerous demands are accomplished to the main server, there are probabilities of overburdening. In such event, the choice of the job is compulsory that will be executed first. This job assortment process is demarcated as Scheduling process. There are numeral of prevailing job scheduling methodologies such as FIFO (First In First Out), LIFO (Last In First Out), SJF (Shortest Job First) etc. The presented work is an enhancement above the current shortest job first algorithm in Cloud environment. The presented algorithmic method is a fusion model that composed of three main methodologies revealed here.

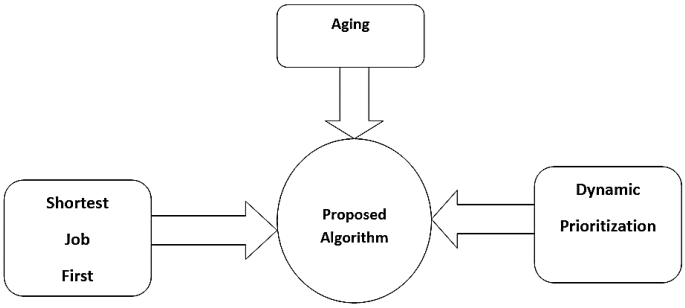

Figure 1: Dimensions of Proposed Approach

The presented work is to state multiprocessor system with the purpose of effective scheduling will be accomplished. We have to structure a new scheduling algorithm such that it will eliminate the shortcomings of earlier SJF algorithm and provide the augmentation in expressions of acquired result. The work comprises the conception to amend the precedence of the process vigorously with the aim of the starvation condition can be committed. To evade this, parametric measures will be established with the intention of resolution concerning the priority alteration will be taken. The proposed work is presented in the flow chart shown below

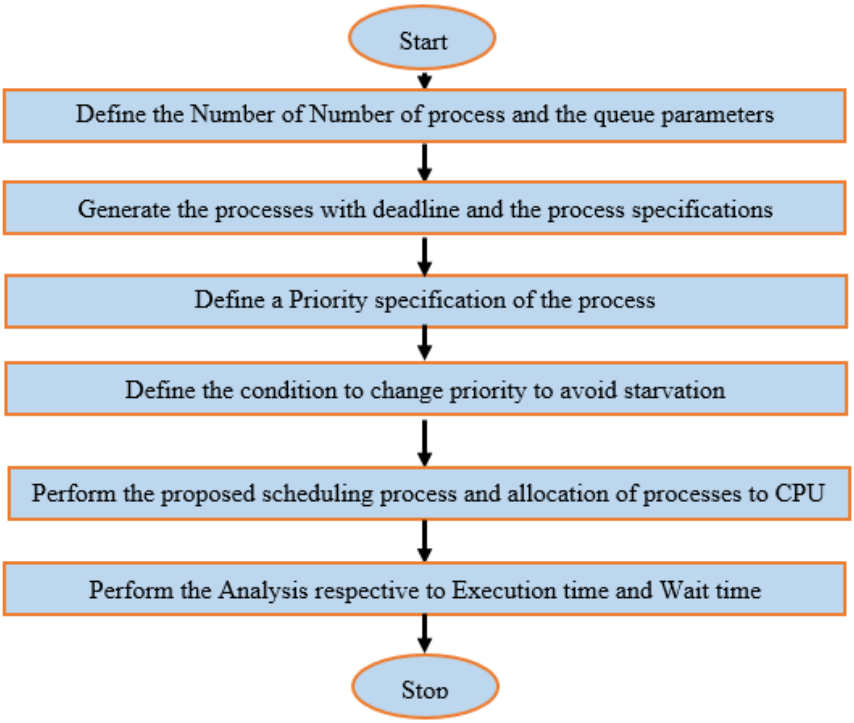

Figure 2: Flow Chart

E. Resource Request

The client's request are accomplished in the arrangement of execution of clients process on the cloud server. The restriction comprised in the request chiefly contains the IO Request and the CPU Request. The request is also distinct with a boundary description for the concentrated amount of demanded resources. As the request is accomplished certain supplementary restrictions are engendered or analyzed by the cloud server. These restrictions comprise the arrival time, process time. These restrictions are essentially appraised by the server centered on the request examination.

IV. RESULTS

The presented work is a simulation-based work in which scheduling is been performed on multiple user requests. Here to present the algorithmic work in an effective way, we have defined all the user parameters manually. The parameters taken here are the arrival time, process time, Number of CPU required and the Number of IO required. As all the input are passed, the next work is to process on these input under existing and proposed approach. Here the restriction is also given for the maximum number of IO and CPUs requested by a user. As the input is given by the user, it is processes for the scheduling task under the proposed approach. The proposed approach is dynamic prioritization and aging improved shortest job approach. Here, we have taken the input for 10 users in a grid network shown the results driven from it. Here figure 3, is showing the sequence of the processes, in which order the processes will be executed in the proposed work and figure 4 is showing the same in the existing approach. With each process the start time, finish time, turnaround time, wait time and the priority is specified.

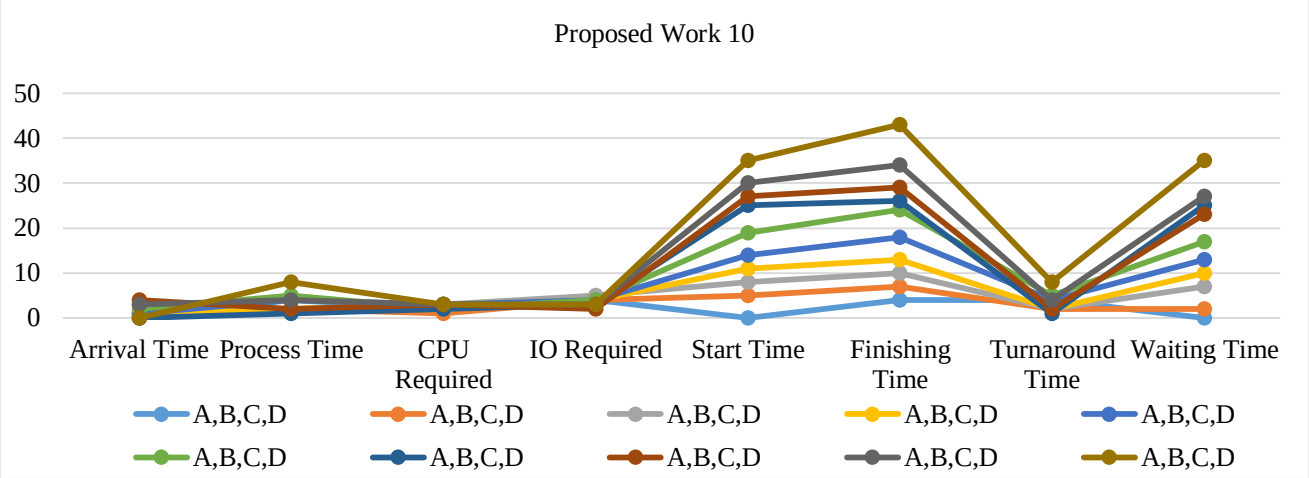

Figure 3: Proposed work

The graph shows the details of how the various process sequenced and how they are executed by the system. After the execution the results are obtained as the average wait time obtained is 15.9 sec and the average turnaround time 3.4 second. The CPU utilization is 12.0 and the throughput is 40.0 processes per hundred cycles.

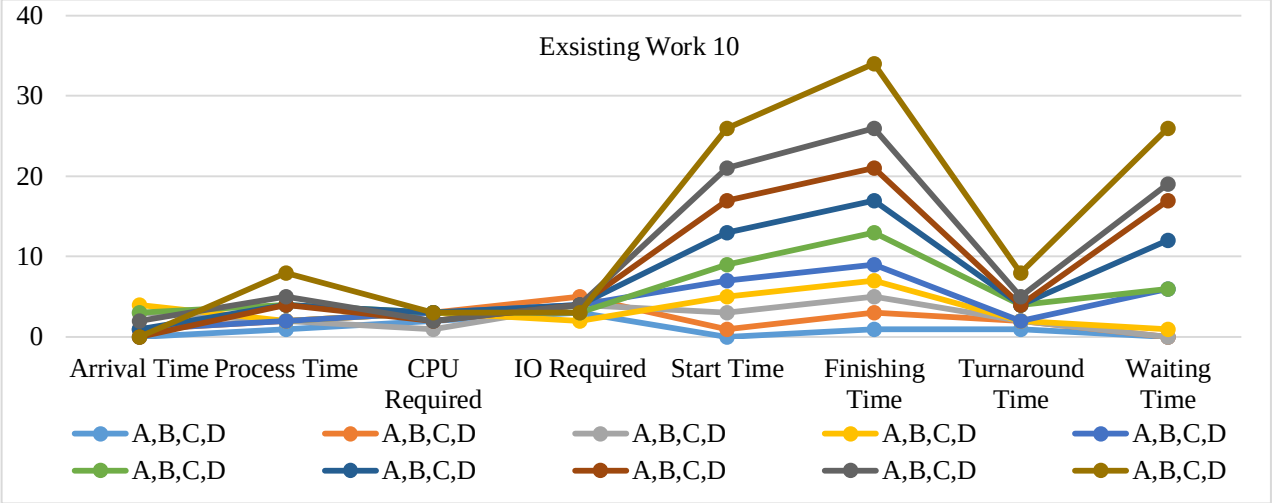

Figure 4: Existing approach

The average wait time obtained is 28.9 sec and the average turnaround time 7.6 second. The CPU utilization is 13 and the throughput is 41.66 processes per hundred cycles.

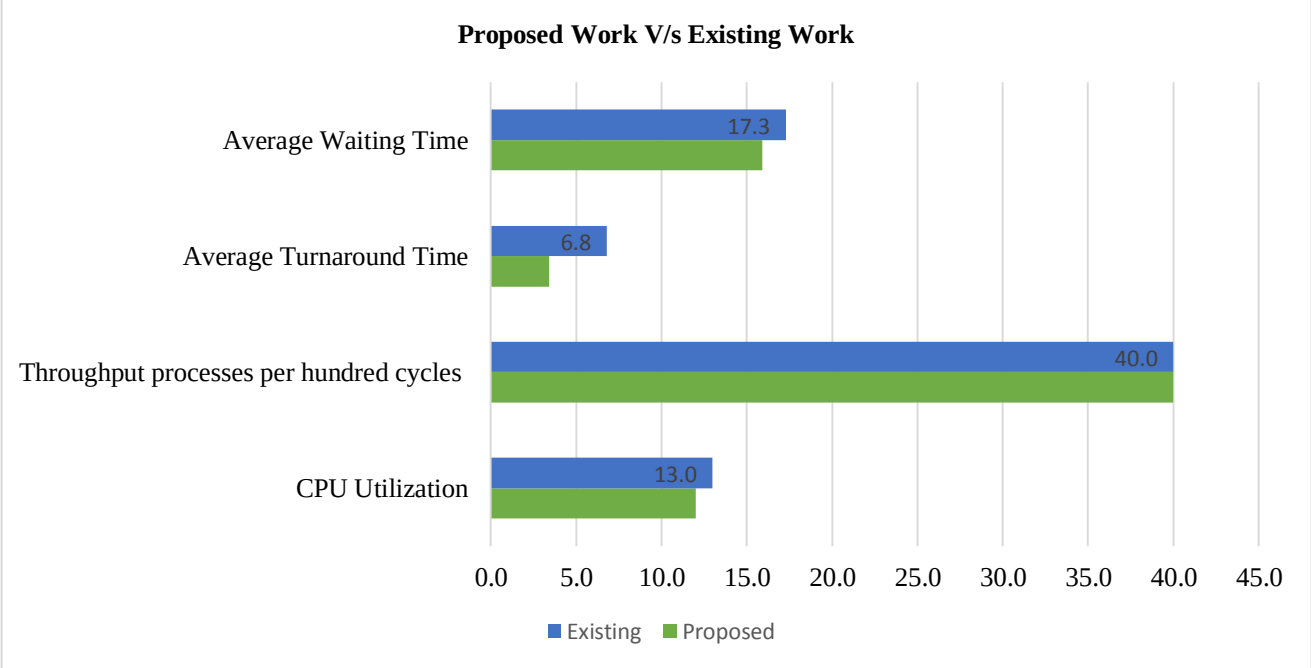

Figure 5: Result Analysis (Proposed Vs. Existing)

As we can see, the order of execution of processes. The average wait time obtained is 17.3 sec and the average turnaround time 6.8 second. The CPU utilization is 13 and the throughput is 41.66 processes per hundred cycles. In figure 5 we can see, the presented approach has reduced the average wait time and turnaround time of the processes as well as the starvation problem is resolved here.

V. CONCLUSIONS

In this presented work, an enhancement over the Shortest Job First scheduling is done in a cloud environment. As the request is accomplished, the request restrictions are composed to accomplish the analysis. These restrictions comprise the process time, request time, CPU requirement and the IO requirement of every process. As the request

acquiesces to the processor, the first work is the scheduler to discover the structure of the process execution. The presented work is committed on the scheduler that is heightened at this point by comprising the supplementary paths. The base algorithm used by the scheduler is the Shortest Job first. As the request attained, the scheduler position the processes in order of process arrival, shortest job and the CPU request. It also allocates the priorities to the processes based IO Request and the CPU requirements. The priority shift mechanism at this point defined by the accomplishment of the frequency computation. It means if the same priority processes surpass the definite boundary the priority of the process will be altered. This prioritization modification diminishes the waiting time of the processes since the intensification of same priority processes. The concluding vector comprised here is the aging to circumvent the starvation. As the waiting time of the process surpasses the definite boundary, the priority of the process will be altered. The proportional examination of the processes acquired outcomes demonstrates that the work has condensed the waiting time of the processes and circumvented the starvation condition. The work demarcated in this research is concentrated on the waiting time and the starvation problem in a cloud system. The work presented an enhanced SJF first algorithm to diminish the waiting time of processes.

REFERENCES

[1] Jonathan A. Winter," Global Power Management for Heterogeneous Many-Core Architectures", PACT'10, September 11–15, 2010, Vienna, Austria. ACM 978-1-4503-0178-7/10/09

[2] Jun Liu," A Compiler Framework for Extracting Superword Level Parallelism", PLDI'12, June 11–16, 2012, Beijing, China. ACM 978-1-4503-1205-9/12/06

[3] Lichen Weng," Scheduling Optimization in Multicore Multithreaded Microprocessors through Dynamic Modeling", CF'13, May 14–16, 2013, Ischia, Italy. ACM 978-1-4503-2053-5

[4] Vishakha Gupta," Kinship: Efficient Resource Management for Performance and Functionally Asymmetric Platforms", CF'13, May 14–16, 2013, Ischia, Italy. ACM 978-1-4503-2053-5

[5] Haozheng Ren, Yihua Lan and Chao Yin, "The Load Balancing Algorithm in Cloud Computing Environment", 2nd International Conference on Computer Science and Network Technology, 2012 IEEE

[6] Jeffrey Galloway, Karl Smith and Jeffrey Carver, "An Empirical Study of Power Aware Load Balancing in Local Cloud Architectures", Ninth International Conference on Information Technology- New Generations, 2012

[7] Jinhua Hu, Jianhua Gu, Guofei Sun and Tianhai Zhao, " A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment", 3rd International Symposium on Parallel Architectures, Algorithms and Programming, 2010 IEEE

[8] Aamer Jaleel," CRUISE: Cache Replacement and Utility-aware Scheduling", ASPLOS'12, March 3–7, 2012, London, England, UK. ACM 978-1-4503-0759-8/12/03

[9] Hiroshi Sasaki," Scalability-Based Manycore Partitioning", PACT'12, September 19–23, 2012, Minneapolis, Minnesota, USA. ACM 978-1-4503-1182-3/12/09.

[10] Viswanath Krishnamurthy," A Novel Thread Scheduler Design for Polymorphic Embedded Systems", CASES'11, October 9–14, 2011, Taipei, Taiwan. ACM 978-1-4503-0713-0/11/10.

[11] Punit Gupta, Mayank Kumar Goyal and Prakash Kumar, "Trust and Reliability based Load Balancing Algorithm for Cloud IaaS", 2012 IEEEM.

[12] I. Chen, and J.K. Lin, "Dynamic Priority Ceilings: A Concurrency Control Protocolfor Real-Time Systems", Journal of Real-Time Systems 2, (1990).

