International Journal of Advances in Applied Sciences (IJAAS)
Vol. 8, No. 3, September 2019, pp. 184~194
ISSN: 2252-8814, DOI: 10.11591/ijaas.v8.i3.pp184-194

Cache optimization cloud scheduling (COCS) algorithm based on last level caches

K. Vinod Kumar, Ranvijay
Department of Computer Science and Engineering, Motilal Nehru National Institute of Technology, India

Article Info

Article history:

Received Apr 23, 2019
Revised Jun 18, 2019
Accepted Jul 3, 2019

Keywords:

Cloud computing
Dynamic voltage and frequency scaling
Energy consumption
Last level cache

ABSTRACT

Recently, the utilization of cloud services like storage, various software, networking resources has extremely enhanced due to widespread demand of these cloud services all over the world. On the other hand, it requires huge amount of storage and resource management to accurately cope up with ever-increasing demand. The high demand of these cloud services can lead to high amount of energy consumption in these cloud centers. Therefore, to eliminate these drawbacks and improve energy consumption and storage enhancement in real time for cloud computing devices, we have presented Cache Optimization Cloud Scheduling (COCS) Algorithm Based on Last Level Caches to ensure high cache memory Optimization and to enhance the processing speed of I/O subsystem in a cloud computing environment which rely upon Dynamic Voltage and Frequency Scaling (DVFS). The proposed COCS technique helps to reduce last level cache failures and the latencies of average memory in cloud computing multi-processor devices. This proposed COCS technique provides an efficient mathematical modelling to minimize energy consumption. We have tested our experiment on Cybershake scientific dataset and the experimental results are compared with different conventional techniques in terms of time taken to accomplish task, power consumed in the VMs and average power required to handle tasks.

Corresponding Author:

K. Vinod Kumar,
Department of Computer Science and Engineering,
Motilal Nehru National Institute of Technology,
Allahabad, 211004-India.
Email: vinodkgpt@gmail.com

1. INTRODUCTION

In the field of next upcoming generation of computing platform, the application of big data is most emerging application because there is a huge increment in the creation of data and the storage space. As per the research done in 2012, the ever-lasting increment of data has converted a particular dataset of few terabytes data to the several petabytes [1]. The several features like large capacity, bulky distributed datasets and high velocity is mainly considered in the applications of Big Data which requires the diverse processing schemes such that optimization strategies and an accurate decision making can be enabled [2]. In actual world a huge amount of data is generated in many fields like medical, surfing on internet, telecommunication, pharmaceutical, technology of information and the business.

Moreover, in recent time, cloud computing applications has taken immense rise in real world due to its various facilities provided by the cloud providers like 'pay-as-you-go' scheme, massive promotions, easy to use, large connectivity with massive number of subscribers. An application of the cloud computing is distributed type application which allows the user to provide the services as per there demand through the internet [3]. The main reason for the drastic growth of cloud computing is because its application saves huge amount of computational time and capacity required for storing the data and also for accessing the several resources. The different resources which are provided by the cloud providers (i.e. amazon, Microsoft etc.) are

available in the form of VMs (virtual machines) within Infrastructure-as-a-Service (IaaS) model [4]. In a cloud computing environment, the time required to execute any of the operations on the virtual machines is clearly relied upon the number of instructions handled by VM which is in some millions and processing power which is in million instructions per second per core. However, the execution time is also dependent on the criticality levels of various functions as it can be varied for every function. Thus, an efficient scheduling is highly appreciated to execute millions of instructions at a time in a cloud environment. Therefore, an efficient task scheduling can be utilized to reduce the processing time in numerous functions of cloud computing environment.

Furthermore, various adaptive control methods like dynamic voltage and frequency scaling ($DVFS$), dynamic speed scaling (DSS) and dynamic frequency scaling (DFS) are introduced by different researchers in recent time to conserve energy and shield environment. Dynamic voltage and frequency scaling ($DVFS$) is one of the most promising technology which can adaptively optimize power by scaling frequency and voltage. Other energy optimization technologies which are available in the market are like DFS, DSS and DSP. All these techniques provide energy-conserving scheduling by diminishing the supply frequency and voltage of the cloud computing environment when numerous jobs are processing adaptively [5-8]. Currently, various chip manufacturing companies provide built-in-processors in integration with $DVFS$ technologies to speed up performance of their network and reduce power consumption in various cloud computing systems like Intel utilizes Intel SpeedStep processor [9], ARM utilizes intelligent energy manager [10] and AMD company uses Power Now processors. The processors which works on higher frequencies can provide decent performance. However, at the same time, they consume immense amount of energy in cloud computing environment. Thus, to ensure high performance and lower energy consumption, $DVFS$ is the most suitable technique currently. The energy can be conserved in the cloud computing devices in two ways such as by scaling voltage and frequency either at task slack period or while processing external peripherals. Therefore, a huge amount of energy can be conserved using $DVFS$ technologies.

Thus, numerous researchers are concerned about the lack of energy conservation in cloud computing environment and hence numerous power efficient technologies are introduced by different researchers to protect environment by enormous amount of power dissipation and enhance the performance of the environment. However, these power efficient technologies require enormous amount of interaction cost between inter-processors. Moreover, these technologies provide insufficient results and energy consumption of cloud computing devices cannot be reduced more due to enormous amount of memory utilization. Furthermore, cache memory optimization is an essential factor to reduce further energy consumption in the cloud computing environment. In recent years, the speed of cloud computing processors and density of main memory has taken immense growth as well as the utilization of I/O sub-systems has extremely enhanced. Specifically, the growth of I/O subsystems in the applications like multimedia and networking has further enhanced the demand of storage capacity. Even though the processing speed of I/O memory sub-systems is highly enhanced, it cannot fulfill the demand of computer sub-systems. Therefore, storage sub-systems are the one of the performances limiting factor and even can be a reason for high energy consumption in cloud computing environment. Therefore, to reduce energy consumption and enhance performance of I/O sub-systems in cloud computing environment, the optimization of cache is an essential factor.

Furthermore, the storage devices in which cache memory is utilized are termed as fast storage sub-systems. However, the capacity of these fast storage sub-systems is limited and hence, replace methods should be utilized to enhance the efficiency of storage devices and cache memory. Moreover, some of the well-known cache replace algorithms utilizes $FIFO, LFU\ and\ LRU$ and their relevant technologies. The cache hit ratio is utilized to enhance the performance of cloud computing environment which rely upon the information reference confinedity. The performance of I/O subsystem is clearly depending on the processing speed of storage sub-systems which need to be similar in all the storage devices. However, this often does not happen as the processing speed of storage devices varies system to system. Therefore, there is a need of a technique, which maintains the cache hit ratio and enhance the accessing speed of the network by optimizing cache memory. Moreover, energy consumption and higher memory utilization can be minimized by reducing the traffic on shared cache memories and memory handlers. The additional energy consumption and slower processing in the cloud computing environment is due to missing of last level caches (LLC) on shared storage sub-systems.

Therefore, these problems need to be focused soon so that the performance and memory capacity of cloud computing devices get enhanced. Thus, here, we have adopted a Cache Optimization Cloud Scheduling ($COCS$) Algorithm Based on Last Level Caches to ensure high cache memory Optimization and to enhance the processing speed of I/O subsystem in a cloud computing environment which rely upon Dynamic Voltage and Frequency Scaling ($DVFS$). The proposed cache Optimization technique helps to minimize the mismanagement of last level caches and to identify the behaviors of cache patterns. This technique decreases congestion on shared cache memories and relocates memory sub-systems dynamically. The memory capacity

of cloud computing environment can be enhanced by sharing caches between cores dynamically. The experimental outcomes verify that the proposed Cache Optimization Cloud Scheduling ($COCS$) Algorithm can provide higher performance in terms of energy consumption, memory capacity and efficiency in cloud computing environment even in the worst-case scenarios.

This paper is presented in following sections which are as follows. In section 2, we describe about the related work to cache memory and their importance and drawbacks in existing techniques and in section 3, we described our proposed Cache Optimization Cloud Scheduling ($COCS$) methodology. In section 4, experimental results and evaluation shown and section 5 concludes our paper.

2. RELATED WORK

In recent years, the concerns related to enormous memory utilization and performance in cloud computing environment have taken immense growth. Thus, performance enhancement and proper management of storage sub-systems is highly critical need. Moreover, to enhance the storage in cloud computing environment, the optimization of cache memory. Therefore, a widespread literature survey is introduced on proper utilization of storage sub-systems and energy aware scheduling algorithms and their link with $DVFS$ in a multi-core heterogeneous cloud computing environment.

In [11], an algorithm for the efficient management of shared caches and their effective partitioning is presented to reduce the accessing of main memory in cloud computing environment. This technique helps to minimize the arithmetic and addressing operations. The architecture of last level cache is introduced to reduce loop tiling problems. However, the balancing of energy consumption and performance is a challenging task. In [12], a cache energy Optimization technique is introduced by interchanging from high speed LI cache to L1 low speed cache using the DVFS application. This model provides a cache hierarchy to model a low-power cache algorithm. This method helps to reduce congestion and also decrease extra latency of main memory. However, a proper cache switching modelling is required to utilize this technique in real time. In [13], an efficient cache architecture is presented for DVFS-enabled devices top reduce the cache overhead in the cloud computing environment. This technique is capable of handling faults by changing associativity adaptively in the network. This technique also helps to eliminate redundant information present in the network. In [14], an efficient caching technique is introduced to compare performance of the network with various state-of-art-techniques. The static cache and redis cache technique help to reduce congestion in the cloud computing network and to provide proper resource utilization. In [15], an accurate memory frequency scaling strategy is introduced using Graphics Processing Units ($GPUs$) based on dynamic voltage and frequency scaling ($DVFS$) technique to reduce energy consumption and enhance performance of the network. This strategy optimizes L2 cache, shared memory and L1 cache memory. In [16], various cache bypassing methods are introduced based on CPU and GPU cores and compared with each other. This technique helps to enhance the capacity of cache memory and decrease energy consumption in larger caches. However, the problems like high congestion in the network and miss rate can be enhanced while using cache by-passing techniques. In [17], a cache by-passing technique is presented for various mobile SOCs which are clock domain and $DVFS$ enabled. This technique helps to enhance performance of the network and energy conservation in $DVFS$-enabled CPUs. Here, last level caches are directly accessed to enhance energy conservation and performance of the system. In [18], a novel cache Optimization technique is presented based on Dynamic voltage and frequency scaling to enhance the reliability of the cloud computing network. This technique helps to enhance performance and capacity of the system. However, overhead of the network is very high using this technique.

Various researchers have introduced different cache memory Optimization techniques in above literatures. However, very few methods can be utilized in real-time due to various problems like high overhead, high energy consumption, slower performance and unable to reduce cache memory [11, 12, 14, 17-19]. Thus, we have adopted a Cache Optimization Cloud Scheduling ($COCS$) Algorithm Based on Last Level Caches to ensure high cache memory Optimization and to enhance the processing speed of I/O subsystem in a cloud computing environment based on Dynamic voltage and Frequency Scaling ($DVFS$) technique.

3. PROPOSED CACHE OPTIMIZATION CLOUD SCHEDULING (COCS) ALGORITHM MODELLING

This section provides detailed modelling for the proposed Cache Optimization Cloud Scheduling ($COCS$) technique. Here, we introduce a Cache Optimization Cloud Scheduling ($COCS$) Algorithm Based on Last Level Caches to ensure high cache memory Optimization and to enhance the processing speed of I/O subsystem in a cloud computing environment based on Dynamic voltage and Frequency Scaling ($DVFS$) technique. The architecture of the proposed Cache Optimization Cloud Scheduling ($COCS$) technique is

presented in Figure 1. Here, we provide efficient modeling to reduce energy consumption and enhance capacity and efficiency of the cloud computing network. The proposed cache Optimization technique migrates cloud virtual machines to reduce the final Last Level Cache (LLC) failures in the cloud computing network. This technique contains confined and generalized scheduling stages. Here, Figure 1 demonstrates the proposed cache Optimization technique which identifies the behavior of various caches in every VM from different working nodes and gathers all the VMs to decrease the final cache failures and the latencies of average memory in the cloud computing network. Figure 1 defines the complete architecture of the proposed $COCS$ model. In each working node, LLC failures are checked to measure the performance of the system. The monitoring system measures the LLC failures per VM and transfer it to the cloud computing scheduler. The generalized scheduling is depending on the status of VMs from each node. The proposed $COCS$ Algorithm utilizes the measured information of cloud computing VMs. Initially, the $COCS$ technique locates VMs on working nodes which takes CPU and memory for every node to enhance the behavior of cache memories.

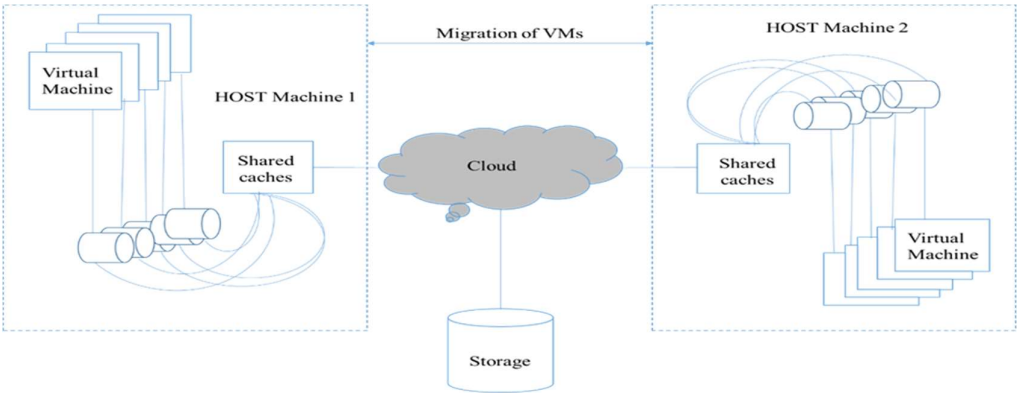

Figure 1. Architecture diagram of proposed Cache Optimization Cloud Scheduling (COCS) model

3.1. Modeling of cache-aware cloud scheduling

This section provides a detailed modeling for proposed cache-aware cloud scheduling technique. Initially, the proposed cache Optimization techniques takes conflicts of shared cache memories. Then, migrates cloud virtual machines to reduce the final Last Level Cache (LLC) failures in the cloud computing network. This technique contains confined and generalized scheduling stages. All the VMs are gathered together to at each working node and planned to domains of mutual caches. The optimization of VMs is presented to enhance the network bandwidth and network capacity. The cloud simulator scheduler is utilized to reallocate VMs at the nodes where LLC failures in a computing network of generalized stage. Here, Algorithm 1 is demonstrated in Table 1 which describes the proposed $COCS$ Algorithm based on Last level caches. This cache aware scheduling algorithm distributed into two stages like one is confined stage and other one is generalized stage. All the VMs at every working nodes are classified by LLC failures and then gathered together according to their missed last level caches in the domains of shared caches. The virtual machines with the most LLC failures fit in group 1, next VM with most LLC in group 2. Similarly, the virtual machines with the least LLC failures fits in group 1 and next VM with least LLC in group 2. This scheme helps to allocate all the VMs in either of the group.

The cloud scheduler realizes two types of nodes present in the cloud computing network such as node with highest LLC failures and node with least LLC failures in the generalized phase. If the difference between LLC failures is more than threshold value then both the VMs are swapped using cache optimization technique. The cloud scheduler performs two-stage scheduling progressively by decreasing the final LLC failures in a cloud computing network after a regular interval of time.

Algorithm 1 Cache Optimization Cloud Scheduling (COCS)Algorithm
Step 1: N_L= <n_x1,….,n_x2> // LLC miss of every computing node
Step 2: W_L= <w_x1,….,w_x2> // LLC miss of VMs in each working nodes
// confined stage
Step 3: for every working node *j* from 1 to *y* do // identify LLC failures in for every VM in working node *j*
Step 4: $nx_j \Leftarrow colllects\ (j)$

Step 5: $W_{\mathbb{L}} \Leftarrow sort\left(nx_j\right)$ // disperse VMs with LLC failures
Step 6: disperse $(W_{\mathbb{L}})$
Step 7: end for
// Generalized stage
// identify working nodes with largest and least LLC failures
Step 8: *largeNode* $\Leftarrow search\ largeNode(N_{\mathbb{L}})$
Step 9: *leastNode* $\Leftarrow search\ leastNode(N_{\mathbb{L}})$
// search VMs which contains maximum and minimum LLC failures
Step 10: *largeVM* $\Leftarrow search\ largestVM(largeNode)$
Step 11: *leastVM* $\Leftarrow search\ leastVM\ (leastnode)$
Step 12: if $T\ <\ largeNode_{\mathbb{LLC}} -\ leastNode_{\mathbb{LLC}}\ then$
Step 13: interchange $(largeVM, leastVM\ )$
Step 14: end if

3.2. Modelling for minimizing energy consumption using COCS

In this section efficient mathematical modelling is presented to minimize energy consumption in multicore cloud computing systems using proposed cache optimization cloud scheduling technique. This technique helps to minimize energy consumption in cloud devices using cache minimization concept by eliminating dynamic constrained minimization problem. The energy consumption $P(t)$ occurs in any multi-core cloud computing processor is expressed as

$$P(t) = e(t)M_a, \tag{1}$$

Where, $P(t)$ represents the energy consumption in a cloud computing processor in the t^{th} managing period. Here, $e(t)$ represents power consumption of a cloud computing multi-core processor which is linked with both frequencies of the core B_k and size of the current $L2$ cache which remains a constant for the frequencies of the core B_k, size of the current L2 cache and task-load of the system and remain unchanged throughout in every managing period whereas M_a is managing period to release multiple illustrations of every task during t^{th} managing period. Here, $v_k(t)$ represents the core (B_k) utilization in a t^{th} managing period which depends on the statistics produced by an operating system. Here, the cloud computing processor consists of two cache levels L1 and L2 which are shared with homogenous cores in multi-core shared architecture. In the proposed model each cloud computing processors support DVFS-enabled cores which conserve high amount of energy. The cache memory is divided into various tasks. The level 2 cache memory divider can be denoted as $a_k(t)$ for a core size B_k. The maximum core frequency for a core size B_k can be expressed as $\mathbb{f}_{k_\uparrow}(t)$. Then, energy consumption $P(t)$ of the cloud computing multi-core processor can be expressed as

$$\min_{a_k(t)|1\ \le k\le i, \mathbb{f}_k(t)|1\le k\le i} \sum_{k=1}^{i}[V_k -\ v_k(t)]^2, \tag{2}$$

$$\min_{a_k(t)|1\ \le k\le i, \mathbb{f}_k(t)|1\le k\le i} P(t) \tag{3}$$

$$R_{\downarrow,k} \le\ \mathbb{f}_k(t)\ \le\ R_{\uparrow,k}\ where, (1 \le k \le i) \tag{4}$$

$$\sum_{k=1}^{i} a_k(t)\ \le A \tag{5}$$

Where, V_k is a Utilization points of sets in which $V = [V_1, \dots\dots, V_i]^T$ for a frequency range of $\left[R_{\uparrow,k,}, R_{\downarrow,k}\right]$ for every core B_k and overall size of L2 caches cloud computing processors is denoted by A whereas $\{a_k(t)|1\ \le k \le i\ \}$ represents the size of cache memory partition and $\{\mathbb{f}_k(t)|1 \le k \le i\}$ denotes the frequency of cores in a t^{th} managing period to reduce the difference between core utilization $v_k(t)$ and Utilization points of sets (V_k). Here, the (2) represents the least energy consumption in any cloud computing multicore processor due to constant power generation $e(t)$ for the t^{th} managing period. Here, the (3) represents shows that the frequency of CPU usually lies in the acceptable range for every core suing the proposed $COCS$ technique. The change in the frequency of any cloud computing device depends on the processors utilized. The (4) represents summation of each divided cache memory which is equivalent to the overall cache memory of the cloud computing processor. For every core of the processor, the difference between core utilization $v_k(t)$ and Utilization point of sets (V_k) is minimized using the proposed cache

optimization technique by changing size of cache partition and frequency of cores. However, these processes decrease the speed of the cloud computing processors. Therefore, to enhance the speed and performance of the system a dynamic model is introduced to maintain a link between managing $v_k(t)$ and manipulated $a_k(t)$ factors and core frequency $\mathbb{f}_k(t)$ in the t^{th} managing period. Initially, for a core B_k, the proposed dynamic model provides a link between $b_{kp}(t)$, task operating time M_{kp} and both manipulated factors $\mathbb{f}_k(t)$ in the t^{th} managing period and $a_k(t)$. Then, $b_{kp}(t)$ can be of operates in various manners like frequency dependent and independent segments and can be represented as

$$b_{kp}(t) = i_{kp}.\left(\mathbb{f}_k(t)\right)^{-1} + s_{kp}(t), \tag{6}$$

Where, $i_{kp}.\left(\mathbb{f}_k(t)\right)^{-1}$ is a segment which rely upon frequency whereas $s_{kp}(t)$ represents the a segment which is independent of frequency for an operating time M_{kp} due to operating speed of I/O devices does not rely upon the frequencies of core. These I/O devices does not participate at the time of executing task. Then, the reserved cache memory for a task operating time M_{kp} is denoted as $s_{kp}(t)$ which defines a strong link between size of cache memory and the number of cache failures. Then, a link between $s_{kp}(t)$ and $a_{kp}(t)$ and allocated cache size for a multi-core architecture B_k is expressed as

$$s_{kp}(t) = \begin{cases} D_{kp}a_{kp}(t) + H_{kp} \;\; 0 \le a_{kp}(t) \le X_{kp} \\ \quad Constant\; a_{kp}(t) \ge X_{kp} \end{cases} \tag{7}$$

Where, X_{kp} is the size of operating set within a task operating time M_{kp} whereas D_{kp} and H_{kp} are the specified task factors. Here, the (7) describes that using the proposed COCS technique, whenever size of operating set X_{kp} is larger than $a_{kp}(t)$, and then the size of cache memory enhances and can lead to a minimum operating time. Similarly, whenever size of operating set X_{kp} is smaller than $a_{kp}(t)$, then cache failure rate becomes high and cannot be handled by assigning an extra cache memory. Then, to manage real-time task, the link between overall independent frequency and operating time of every task for multi-core processor B_k and size of overall cache $a_k(t)$ allocated to core B_k can be expressed as

$$s_k(t) = \begin{cases} \sum_p D_{kp}{}' a_k(t) + \sum_p H_{kp} \;\; 0 \le a_k(t) \le X_k \\ \quad Constant\; a_k(t) \ge X_k \end{cases} \tag{8}$$

Where, $D_{kp}{}' = D_{kp}a_{kp}(t).\left(a_k(t)\right)^{-1}$ and $X_k = \sum_p X_{kp}$. Here, the (8) represents the summation of (7) for each task on a multi-core processor B_k. Then, the proposed COCS technique helps to minimize interference between shared caches resources of various cores and can be expressed as

$$h_k(t) = \sum_p i_{kp}\, q_{kp}\,.\left(\mathbb{f}_k(t)\right)^{-1} + \sum_p D_{kp}{}' q_{kp} a_k(t) + \sum_p H_{kp}\, q_{kp} \tag{9}$$

Where, $h_k(t)$ represents the predicted core utilization and q_{kp} represents rate of the task within operating time M_{kp} for a multi-processor core B_k. From (9), it is verified that $h_k(t)$ is inversely proportional to the frequency of core $\mathbb{f}_k(t)$. The predicted change in utilization $\Delta h_k(t)$ for a multi-core architecture B_k can be expressed as

$$\Delta h_k(t) = l_k(t) \sum_p i_{kp}\, q_{kp} + \Delta a_k(t) \sum_p D_{kp}{}' q_{kp} \tag{10}$$

Where, $l_k(t) = \left(\mathbb{f}_k(t)\right)^{-1} - \left(\mathbb{f}_k(t-1)\right)^{-1}$ and $\Delta a_k(t) = a_k(t) - a_k(t-1)$. Here, $\Delta h_k(t)$ can be termed as the linear function of $l_k(t)$ and $\Delta a_k(t)$. Here, the (10) replaces the direct utilization of frequency of core $\mathbb{f}_k(t)$ to $l_k(t)$. The (10) shows that $\Delta h_k(t)$ is directly proportional to i_{kp} and $D_{kp}{}'$. Then, the cost function of cloud computing system can be reduced with the help of controller for a multi-core B_k as

$$Z_k(t) = \sum_{c=1}^{E} \| v_k(t+c-1|t) - \beta \mathbb{f}_k(t+c-1|k) \|^2 + \| u_k(t|t) - u_k(t-1|t) \|^2 \tag{11}$$

Where,

$$R_{\downarrow,k} \le \mathbb{f}_k(t) \le R_{\uparrow,k} \tag{12}$$

$$a_k(t) \leq a_{quota,k} \tag{13}$$

Where, $u_k(t) = \begin{bmatrix} l_k(t) \\ \Delta a_k(t) \end{bmatrix}$ and E represents estimated horizon to estimate the behavior of the device in E managing periods. Here, $\beta \mathbb{f}_k(t+1|t)$ represents the trajectory with respect to the utilization factor $v_k(t+c-1|t)$ must change from the present utilization factor $v_k(t)$ to utilization point of sets V_k. The size of cache $a_k(t)$ for a multi-core system B_k is limited by $a_{quota,k}$ to satisfy (5). From this above modelling, the optimization of cache memory can be easily achieved and optimization and least square problems are also minimized. The optimized power consumption can be presented using our proposed COCS technique as

$$e_k(t) = S_k \mathbb{f}_k(t)^3 + Y_k a_k(t) + C_k \tag{14}$$

Where,

$$R_{\downarrow,k} \leq \mathbb{f}_k(t) \leq R_{\uparrow,k} \tag{15}$$

$$a_k(t) \leq a_{quota,k} \tag{16}$$

Where, $S_k, Y_k \; and \; C_k$ are the power factors of the cloud computing multi-core processor. The power consumption of cloud computing multi-core processor is represented as the sum of the power consumption of cores and shared caches. The total power consumption is directly depending on the dynamic components of power $S_k \mathbb{f}_k(t)^3$ and leakage power C_k. Therefore, the power consumption in cache can be optimized using the proposed $COCS$ technique. In this way, energy consumption of a cloud computing model is minimized and performance of the system can be enhanced.

4. PERFORMANCE EVALUATION

In this modern era, the utilization of cloud computing embedded devices in daily life has tremendously increased due to abundant utilization of various portable gadgets, information systems and digital appliances etc. However, these embedded devices require high amount of power to operate properly, Thus, the performance of this multi-core cloud computing systems and embedded devices must be very high due to satisfy the ample demand of the market. However, in recent time, the performance of various cloud computing centers has degraded due to large amount of energy consumption in these cloud computing centers. Another major problem while using the cloud computing systems is the storage of the system. The amount of data present in these cloud computing systems is extremely high and requires ample amount of storage to handle these large data. Therefore, to maintain a balancing between high amount of energy consumption and storage enhancement of cloud computing systems, we have introduced a Cache Optimization Cloud Scheduling ($COCS$) Algorithm Based on Last Level Caches to ensure high cache memory Optimization and to enhance the processing speed of I/O subsystem in a cloud computing environment which rely upon Dynamic Voltage and Frequency Scaling ($DVFS$). The proposed cache Optimization technique helps to minimize the mismanagement of last level caches and to identify the behaviors of cache patterns. Here, we have tested our model on $Cybershake$ scientific dataset and various sizes of tasks are considered like 30, 50, 100, and 1000 tasks to calculate time taken to accomplish tasks. $Cybershake$ Workflow is produced with the help of four type of jobs like 30, 50, 100 and 1000. It is used for the Southern California Earthquake center to identify earthquake dangers in a specified place [20]. It requires large amount of storage and CPU resources. The experimental outcomes are presented in terms of time taken to execute task of different size, power consumed in the VMs and average power required to handle tasks against number of tasks considered. Different factors are considered to calculate operating time and energy consumption which is demonstrated in the following section I Table 1. Our proposed $COCS$ model carried out on 64-bit windows 10 OS with 16 GB RAM which contains an $INTEL\ (R)\ core\ (TM)\ i5-4460$ processor. It consists of 3.20 GHz CPU. This project is simulated using $EclipseWS$ Neon.3 editor and code is written in $JAVA$.

4.1. Comparative analysis

Recently, the modern high-tech cloud computing systems are attempting to match high-end device performance to fulfill the extreme demand of market. However, achieving high-end system performance can head to the extreme energy consumption in these high-tech cloud centers. This is due to this device are mainly battery-oriented and it is very critical to accurately control high amount of energy consumption. If not, then the lifespan of these devices becomes limited due smaller battery life and it may frustrate cloud

subscribers. Therefore, efficient task scheduling and storage capacity enhancement by removing unwanted data is the best way to keep up with the high performance and minimum energy consumption. Therefore, here, we have introduced a Cache Optimization Cloud Scheduling ($COCS$) Algorithm Based on Last Level Caches to ensure high cache memory Optimization and to enhance the processing speed of I/O subsystem in a cloud computing environment which rely upon Dynamic Voltage and Frequency Scaling ($DVFS$). The proposed technique helps to maintain balancing between high perform and minimum energy consumption. Therefore, here, we have tested our experiment on $Cybershake$ scientific dataset and the experimental results are compared with different conventional techniques in terms of time taken to accomplish task, power consumed in the VMs and average power required to handle tasks. These $Cybershake$ scientific dataset is widely utilized in evaluating the performance of various scheduling techniques and $Cybershake$ workflow is utilized in our experiment. Here, Table 1 demonstrates the performance comparison of our proposed COCS technique in terms of task completion time, power sum, average power and power consumption occurs in VMs using $Cybershake$ scientific dataset of different size like 30, 50,100 and 1000. Power consumption using the proposed $COCS$ technique for $Cybershake$ 30 is 128.4845 Watts, $Cybershake$ 50 is 151.5101 Watts$Cybershake$100 is 258.672 Watts and $Cybershake$ 1000 is 1696.490 Watts demonstrated in Table 1 which is extremely low in contrast to conventional standard technique. The proposed $COCS$ technique compares Average task completion time using the proposed $COCS$ technique is compared with other conventional techniques utilizing $Cybershake$ scientific dataset of different size like 30, 50, 100 and 1000 as demonstrated in Table 1. The Average Task Completion Time with the proposed $COCS$ technique for scientific dataset $Cybershake$ 30 is 0.368431 sec, $Cybershake$ 50 is 0.193506 sec, $Cybershake$ 100 is 0.140568 sec and $Cybershake$ 1000 is 0.058476 sec and compared with various state-of-art-techniques. The average task completion time is much smaller than any other state-of-art techniques like EMS-C, SPEA2, MODE, NSPSO, $\in -FUZZY$ PSO, MOHEFT [20] using our proposed $COCS$ technique demonstrated in Table 2. The average Power required is much smaller than any other state-of-art techniques like EMS-C, SPEA2, MODE, NSPSO, $\in -FUZZY$ PSO, MOHEFT [20] using our proposed $COCS$ technique demonstrated in Table 2. The VMs types with description used in our experiment is shown at Table 3. The machine configuration used to compute these results are demonstrated in Table 4. The Power Consumption required is much smaller than any other state-of-art techniques like EMS-C, SPEA2, MODE, NSPSO, $\in -FUZZY$ PSO, MOHEFT [20] using our proposed $COCS$ technique demonstrated in Table 2. The machine configuration used to compute these results are is demonstrated in Table 5. Figure 2 demonstrates the internal architecture of $Cybershake$ workflow.

Table 1. Various parameters comparison for proposed COCS technique vs DVFS using scientific model Cybershake

Parameters			Total Execution Time (s)	Power Sum (W)	Average Power (W)	Power Consumption (Wh)
DVFS	Cybershake 30	VM=20	6359.41	12175922.64	19.146320	3495.42
	Cybershake 50	VM=30	14448.90	29068552.89	20.118183	8518.39
	Cybershake 100	VM=50	30124.41	61177338.40	20.308229	18966.33
	Cybershake 1000	VM=30	74543.57	149968122.08	20.118184	236303.28
$COCS$	Cybershake 30	VM=20	262.73	415469.6734	15.8135	128.4845
	Cybershake 50	VM=30	283.68	451080.8167	15.9010	151.5101
	Cybershake 100	VM=50	443.21	704751.2215	15.9010	258.672
	Cybershake 1000	VM=30	1328.05	2111745.021	15.9010	1696.490

Table 2. Runtime ratios of the peer algorithms against the proposed EMS-C on the real-world workflows (i.e., runtime (peer algorithm) =runtime (COCS))

DAGs	Number of nodes	Average Execution time (s)							
		EMS-C	DVFS	SPEA2	MODE	NSPSO	$\in -fuzzy$ PSO	MOHFET	$COCS$
$Cybershake$ 30	30	23.77	211.98	1.62	1.18	21.25	15.04	10.39	0.368431
$Cybershake$ 50	50	29.32	288.978	1.28	1.37	30.17	26.81	38.16	0.193506
$Cybershake$ 100	100	31.53	301.244	0.92	1.31	44.75	42.22	110.42	0.140568
$Cybershake$ 1000	1000	22.71	74.54	0.32	0.97	70.72	69.56	--	0.058476

Table 3. VMs types with description used in our experiment

Type	Memory (GB)	Core Speed (ECU)	Cores
m1. small	1.7	1	1
m1. large	7.5	4	2
m1. xlarge	15	8	4

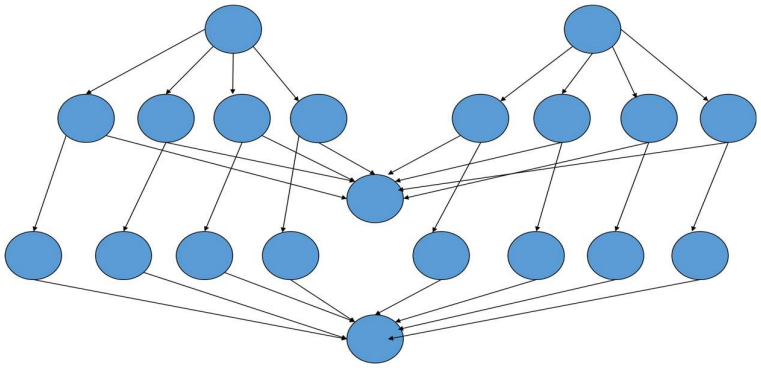

Figure 2. Internal architecture of Cybershake workflow

Table 4. Runtime ratios of the peer algorithms against the proposed EMS-C on the real-world workflows (i.e., average power (peer algorithm) = average power (COCS))

DAGs	Number of nodes	Average power						
		EMS-C	DVFS	SPEA2	MODE	NSPSO	$\in -fuzzy\ PSO$	$COCS$
$Cybershake$ 30	30	34.0079	141.02	2.317	1.688	30.402	21.51782	0.5271166
$Cybershake$ 50	50	48.186	156.72	2.103	2.251	49.583	44.06124	0.31802
$Cybershake$ 100	100	35.666	151.92	1.040	1.481	50.621	47.75910	0.15901
$Cybershake$ 1000	1000	6.175	52.191	0.087	0.263	19.230	18.91500	0.015901

Table 5. Runtime ratios of the peer algorithms against the proposed EMS-C on the real-world workflows (i.e., power consumption (peer algorithm) = power consumption $COCS$)

DAGs	Number of nodes	Power consumption						
		EMS-C	DVFS	SPEA2	MODE	NSPSO	$\in -fuzzy\ PSO$	$COCS$
$Cybershake$ 30	30	276.31	1145.8	18.83	13.71	247.02	174.8320	4.2828166
$Cybershake$ 50	50	459.13	1493.2	20.04	21.45	472.44	419.8304	3.030202
$Cybershake$ 100	100	580.21	2471.4	16.92	24.10	823.48	776.9287	2.58672
$Cybershake$ 1000	1000	658.85	5568.3	9.283	28.14	2051.7	2018.056	1.69649

4.2. Graphical representation

This section describes about the graphical demonstration of experimental outcomes using proposed $COCS$ technique. Here, Figure 3 demonstrates task completion time Comparison of proposed $COCS$ technique with conventional $DVFS$ technique with the help of scientific dataset $Cybershake$ for different task sizes as 30, 50,100 and 1000. Here, Figure 4 demonstrates Power Sum Comparison of proposed $COCS$ technique with conventional $DVFS$ technique with the help of scientific dataset $Cybershake$ for different task sizes as 30, 50,100 and 1000. Here, Figure 5 demonstrates Average Power Comparison of proposed $COCS$ technique with conventional $DVFS$ technique with the help of scientific dataset $Cybershake$ for different task sizes as 30, 50, 100 and 1000. Similarly, Figure 6 demonstrates Power Consumption Comparison of proposed $COCS$ technique with conventional $DVFS$ technique with the help of scientific dataset $Cybershake$ for different task sizes as 30, 50, 100 and 1000. Furthermore, Figure 7 demonstrates average task completion time Comparison of proposed $COCS$ technique with various state-of-art-techniques using scientific dataset $Cybershake$ for different task sizes as 30, 50, 100.

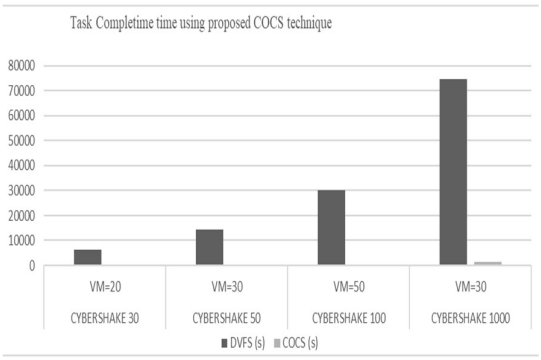

Figure 3. Task completion time comparison of proposed COCS technique vs DVFS using scientific workload $Cybershake$

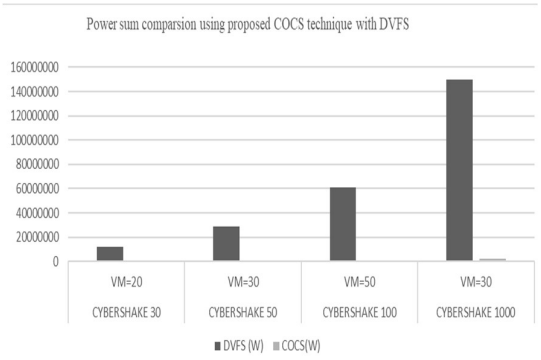

Figure 4. Power sum comparison of proposed COCS technique vs DVFS using scientific workload $Cybershake$

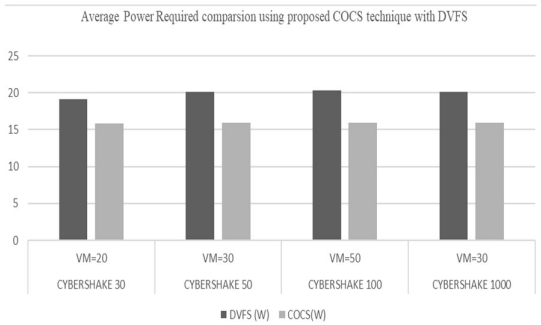

Figure 5. Average power comparison of proposed COCS technique vs DVFS using scientific workload $Cybershake$

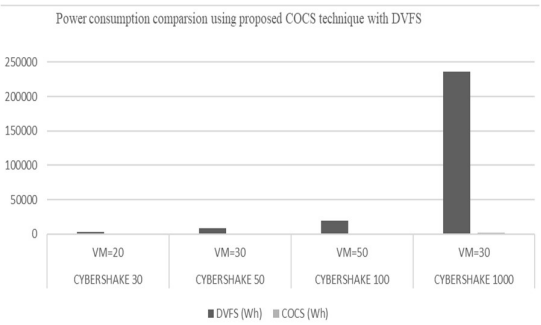

Figure 6. Power consumption comparison of proposed COCS technique vs DVFS using scientific workload $Cybershake$

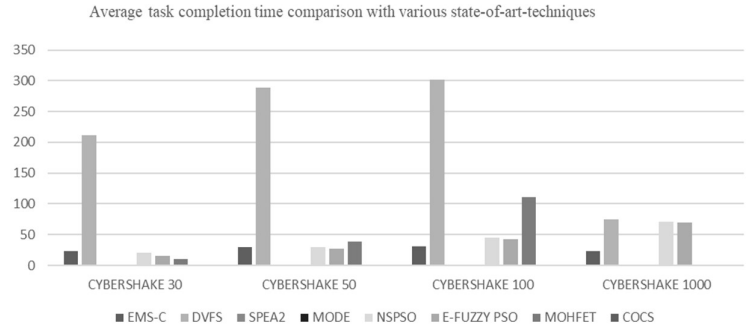

Figure 7. Average task completion time comparison of proposed COCS technique with various state-of-art-techniques using scientific workload $Cybershake$

5. CONCLUSION

An efficient task scheduling in each cloud computing VMs and reduction of high energy consumption in these devices is become a primary priority. Various researchers have provided different techniques to decrease energy consumption in multi-core cloud computing processors. However, still it remains an unsolved issue. Therefore, one method to decrease storage and energy consumption in multi-core processors is cache minimization. Therefore, we have adopted Cache Optimization Cloud Scheduling ($COCS$) Algorithm Based on Last Level Caches to ensure high cache memory Optimization and to enhance the processing speed of I/O subsystem in a cloud computing environment which rely upon Dynamic Voltage

and Frequency Scaling ($DVFS$). This technique helps to reduce last level caches and identifies the behavior of different caches in every VM from different working nodes and gathers all the VMs to decrease the final cache failures. A detailed modelling is presented to minimize the energy consumption and storage in cloud computing multi-core processors by reducing cache memory. We have tested our experiment on $Cybershake$ scientific dataset and the experimental results are compared with different conventional techniques in terms of time taken to accomplish task, power consumed in the VMs and average power required to handle tasks. The Average Task Completion Time with the proposed $COCS$ technique for scientific dataset $Cybershake$ 30 is 8.7576 sec, $Cybershake$ 50 is 5.6736 sec, $Cybershake$ 100 is 4.4321 sec and $Cybershake$ 1000 is 1.328 sec. Power consumption using the proposed COCS technique for $Cybershake$ 30 is 128.4845 Watts, $Cybershake$ 50 is 151.5101 Watts $Cybershake$100 is 258.672 Watts and $Cybershake$ 1000 is 1696.490 Watts which is very less compare to any other state-of-art-techniques. Experimental results verify superiority of our proposed $COCS$ technique in terms of task completion time, average power required and energy consumption.

REFERENCES

[1] Wikipedia, Big data, 2014a. [Online]. Available: http://en.wikipedia.org/wiki/Big_data.

[2] M. A. Beyer and D. Laney, *The Importance of 'big data': A Definition*. Gartner, Stamford, CT, 2012.

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *National Institute of Standards and Technology,* 2009.

[4] H. Topcuoglu, S. Hariri, and M.y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13(3), pp. 260-274, 2002.

[5] K. Li, "Energy-efficient task scheduling on multiple heterogeneous computers: Algorithms, analysis, and performance evaluation," *IEEE Trans. Sustain. Comput*., vol. 1(1), pp. 7–19, 2017.

[6] K. Li, "Power and performance management for parallel computations in clouds and data centers," *J. Comput. Syst. Sci*., vol. 82(2), pp. 174–190, 2016.

[7] X. Xiao, G. Xie, R. Li, and K. Li, "Minimizing schedule length of energy consumption constrained parallel applications on heterogeneous distributed systems," *Proc. 14th IEEE Int. Symp. on Parallel and Distributed Processing with Applications, IEEE Computer Society*, pp. 1471-1476, 2016.

[8] G. Xie, X. Xiao, R. Li, and K. Li, "Schedule length minimization of parallel applications with energy consumption constraints using heuristics on heterogeneous distributed systems," *Concurrency Comput. Parctice Experience*, vol. 29(16), 2016.

[9] "Enhanced intel speedstep technology for the intel pentium m processor," March 2014. [Online]. Available: http://downloa10] "Amd powernow!? technology informational white paper," November 2000. [Online]. Available: http://www.amd-k6.com/wpcontent/ uploads/2012/07/24404a.pdf

[10] K. Flautner, D. Flynn, and M. Rives, "A combined hardware software approach for low-power socs: Applying adaptive voltage scaling and intelligent energy management software," 2003. [Online]. Available: intel.com/design/network/papers/30117401.pdf

[11] V. Kelefouras, G. Keramidas, and N. Voros, "Cache Partitioning + Loop Tiling: A Methodology for Effective Shared Cache Management," *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Bochum, pp. 477-482, 2017.

[12] K. Saito, R. Kobayashi, and H. Shimada, "Reduction of cache energy by switching between L1 high speed and low speed cache under application of DVFS," *2016 International Conference on Advanced Informatics: Concepts, Theory and Application (ICAICTA)*, pp. 1-6, 2016.

[13] M. Mavropoulos, G. Keramidas, and D. Nikolos, "A defect-aware reconfigurable cache architecture for low-Vccmin DVFS-enabled systems," *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 417-422, 2015.

[14] M. Kusuma, Widyawan, and R. Ferdiana, "Performance comparison of caching strategy on wordpress multisite," *2017 3rd International Conference on Science and Technology - Computer (ICST)*, pp. 176-181, 2017.

[15] Q. Wang and X. Chu, "GPGPU performance estimation with core and memory frequency scaling," *arXiv preprint arXiv*:1701.05308, 2017.

[16] S. Mittal, "A Survey of Cache Bypassing Techniques," *MDPI J. Low Power Electron. Appl.*, vol. 6(5), 2016.

[17] Joonho Kong and Kwangho Leeb, "A DVFS-aware cache bypassing technique for multiple clock domain mobile SoCs," *IEICE Electronics Express*, vol. 14(11), pp. 1-12, 2017.

[18] Y. H. Chen, Y. L. Tang, Y. Y. Liu, A. C. H. Wu, and T. Hwang, "A Novel Cache-Utilization-Based Dynamic Voltage-Frequency Scaling Mechanism for Reliability Enhancements," *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*, vol. 25(3), pp. 820-832, 2017.

[19] Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary Multi-Objective Workflow Scheduling in Cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27(5), pp. 1344-1357, 2016.

[20] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," *Future Generat. Comput. Syst.,* vol. 25(5), pp. 528-540, 2009.