SURVEY OF ENERGY-EFFICIENT LOAD-BALANCING SCHEDULING ALGORITHMS IN DATA CENTER

Smruti Phadke, K.K. Joshi
Student, Department of Computer Engineering & Information Technology, Veermata Jijabai Technological Institute, Mumbai, Maharashtra, India

Abstract— Nowadays, in order to save operational costs, IT companies are getting their data intensive jobs serviced by a third party platform, i.e. a cloud computing data center. These data centers house huge amounts of equipment to service these jobs, resulting in increased energy consumption. More energy consumption results in an increase in the heat generated by these data centers. In order to minimize these problems, several energy-efficient and power-aware resource allocation algorithms, also known as Green Schedulers, have been devised. This paper surveys some of the most significant Green Schedulers, explores their approaches and analyses their performance. This paper also concludes which Green Scheduler is best suited for different cloud computing data center architectures.

Index Terms— Data center, Energy-efficient, Green Cloud, Load-balancing, Power-aware, Resource-allocation

1. Introduction
The concept of cloud computing involves a data center somewhere in the world, or even multiple data centers scattered around the world. A data center (also known as server farm) is an integrated archive for the storage, administration, and distribution of data and information. These data centers house computer, server and networking systems and components to service hundreds and thousands of user requests [5].

Nowadays, in order to satisfy bulk of user requests, specialized hardware is required in data centers. Data centers with infrastructure from a single vendor are not capable of servicing these user requests. There are two types of cloud computing data center infrastructures, viz. homogeneous and heterogeneous. A homogeneous cloud is one where the complete software bundle, everything from hypervisors (or remote cloud provider), to several central management layers, to the end-user terminal, is supplied by a single vendor. A heterogeneous cloud [9] combines segments by several different providers, either at different levels (a hypervisor from one vendor and its management tool from another) or even at the same level (multiple hypervisors from different vendors, all driven by the same management tool). In order to service these user requests, we need a heterogeneous data center [8].

A huge amount of energy is required for large geographically distributed data centers to work. This energy accounts for a big section of the total working costs for cloud data centers. It is projected that this energy contributes up to 10% of the current data center operational expenses (OPEX), which may rise to 50% in the next few years. However, this energy consumption is not the only part of the OPEX bill. This huge power consumption results in the generation of heat and we need a cooling system that will cost in between $2 to $5 million per year for majority of data centers. If this heat is not controlled, hardware reliability decreases and Service Level Agreement (SLA) with the customer is violated. A huge part (over 70%) of the heat is generated by data center infrastructure. Therefore, the OPEX bill can be drastically reduced by building an optimized data center infrastructure. We need energy efficient algorithms for scheduling user requests and balancing load equally among servers [5]. These energy efficient algorithms, also known as Green Schedulers, should also operate in heterogeneous data centers.

2. Review of literature
A few essential power saving solutions were studied in detail that were concerned with converting the data center hardware components into energy efficient components [5]. Technologies, such as Dynamic Voltage and Frequency Scaling (DVFS), and Dynamic Power Management (DPM) were reviewed and set up. As these techniques relied on power-down and power-off procedures, the efficiency of these methods is narrow. It is found that an idle server consumes around 2/3rd of the maximum load. Computing and communicational resources are many-a-times overprovisioned to contain the expected maximum load as the workload of a data center fluctuates hourly or weekly. In fact, the average load contributes to only 30% of all resources. As a result, we can put rest 70% of the resources into a sleep mode for...
maximum time. But in order to achieve the above condition, we require central coordination and energy-aware workload scheduling techniques. Majority of energy-aware scheduling algorithms try to: (1) intensify the workload on the smallest set of computing resources and (2) augment the resource which can be put into sleep mode. Usually a simple enhancement of the data center architecture and energy-aware scheduling of the workloads may lead to significant energy savings.

Dynamic voltage and frequency scaling (DVFS) is a widely known power management technique where the supply voltage is reduced by decreasing the clock frequency of a processor. This results in less consumption of power leading to significant reduction in the energy needed for a computation. Dynamic voltage and frequency scaling is commonly deployed to conserve power on a wide range of computing systems, including embedded systems, laptops and desktop systems, high-performance server-class systems, etc. Power consumption is minimised by reducing the operating frequency but energy consumption remains same as the computation needs more time to finish. Energy can be reduced majorly by lowering the supply voltage. Therefore, lowering both the supply voltage and operating frequency reduces the power and energy consumption. On reducing the clock frequency by half, processor’s power consumption reduces and still the tasks can complete by deadline, energy consumption being the same. Power consumption can be further reduced by reducing the voltage level by half, without any corresponding increase in execution time. As a result the energy consumption is reduced significantly and still the appropriate performance is achieved.

Dynamic power management (DPM) is the judicious shutdown of system segments that are idle or underutilized. DPM is one of the best techniques for decreasing power dissipation in computer systems. An effective DPM strategy must increase power savings while keeping performance deterioration within acceptable limits [11].

Today, nearly 70% of all communications are performed internally in data centers that house around 100,000 hosts. Due to the peripheral and cost-based limitations of the used networking devices, the traditional hierarchical network infrastructure becomes a hurdle. Therefore, three-tier data center architectures are nowadays commonly used, as shown in figure 1. They include: (1) access layer, (2) aggregation layer, and (3) core layer as shown in figure 1. The availability of the aggregation layer causes an increase in the number of server nodes (to over 10,000 servers) while keeping the Layer-2 (L2) switches in the access network, which provides a loop-free topology [5].

3. ENERGY-EFFICIENT RESOURCE-ALLOCATION ALGORITHMS

Many algorithms have been designed for an energy-efficient power-aware resource allocation in a cloud computing data center. Some of the most significant algorithms have been explained in detail in this section.

![Figure 1: Three-tier Data Center Architecture](image)

**DFS: Distributed Flow Scheduling**

It is a distributed flow scheduling (DFS) system for power-efficient data centers. High utilization of available network bandwidth and efficient elephant flow management is achieved [3]. Flow discrimination is d at the end hosts, and only flows that transfer significant amount of data (elephant flows) will be scheduled. A distributed scheduling algorithm and a mechanism to detect network traffic are also introduced to ensure processing efficiency. Data centers include elephant flows, which account for less than 10% of the total number of flows. This causes network congestion resulting from collisions. DFS effectively manages elephant flows. DFS has following steps: First, elephant flows are detected at the end hosts. Next, all the current available routes are searched for these flows. Finally, suitable routes are figured out to place them. The steps are as follows:

1. **Elephant Flow Detection**

A special field in the hosts OS is included that inspects the TCP buffer. An elephant flow can be accurately detected if the buffer of the flow crosses a predefined threshold. Once an elephant is detected, the average flow rate moving out of the TCP buffer in every packet marking period is detected.

2. **Route Searching**

The scheduler determines the next hop to forward the flow once the first marked packet is received. All available next hops are listed and one of them is chosen using a scheduling strategy. But not all active neighbors are available candidates. The real available candidates are on active routes leading to the target servers.

3. **Elephant Flow Scheduling**

The suitable route to place an elephant flow is selected by determining hop pairs from the sides of two
communicating servers to the intermediate switches. Finally, the core switch whose corresponding combined utilization is the smallest is chosen. Each packet in the corresponding elephant flow is routed according to the flow entry.

DCEERS: Data Center-wide Energy-Efficient Resource Scheduling framework

DCEERS is Data Center-wide Energy-Efficient Resource Scheduling framework (DCEERS) [4]. According to the current workload of the datacenter, data center resources are scheduled. Minimum number of resources to account for the current workload is calculated by the Minimum Cost Multi Commodity Flow (MCMCF) using the Benders decomposition algorithm. The Benders decomposition algorithm can solve the MCMCF problem in linear time for large data centers.

The DCEERS framework has two basic modules: (a) workload calculator, and (b) the resource/ priority scheduler. The workload calculator is used to calculate the core bandwidth based on: (a) the number of user requests measured at the front-end servers, and (b) the required bandwidth per request. The core bandwidth, \( bw_{core-req} \), is calculated from the following equation (1):

\[
bw_{core-req} = \sum_{i=1}^{n} (requests \times bw_{per-req}) \quad \ldots \ldots \ldots (1)
\]

When the required bandwidth is greater than the data center core capacity, then the data center is servicing more users than its capacity and no energy is saved. Service Level Agreements (SLAs) will be affected by powering off data center resources in such a throughput demand scenario. If bandwidth demand is less than the data center core capacity, the resource optimizer will find the minimum number of server and network resources required to service the demand. The remaining resources can be powered off to achieve energy efficiency. The MCMCF optimization will generate a minimum set of resources that are needed for resource optimization. The MCMCF problem is then solved by using Bender’s Algorithm.

DENS: Data Center Energy-Efficient Network-aware Scheduling

The DENS methodology reduces the total energy consumption of a data center by selection of the best-fit computing resources for job execution based on the communication potential and the load level of data center components [1]. For a three-tier architecture, a DENS metric \( M \) is defined as a weighted sum of server-level \( fs \), rack-level \( fr \), and module-level \( fm \) functions, given by equation (2):

\[
M = a \cdot fs + b \cdot fr + y \cdot fm \quad \ldots \ldots \ldots (2)
\]

where \( a, b, \) and \( y \) are weighted coefficients that define the impact of servers, racks and modules on the metric. Higher \( a \) values select highly loaded servers in lightly racks. Higher \( b \) values select computationally loaded racks with low network traffic activity. Higher \( y \) values select fully loaded modules. Since \( a + b + y \) must always equate unity, the values of \( a = 0.7, b = 0.2, \) and \( y = 0.1 \) are selected. Now, the server, rack and module functions are calculated by equations (3), (4) and (5), respectively:

\[
f_s(l, q) = L_s(l) \cdot \frac{\beta}{L_m(l)} \quad \ldots \ldots \ldots (3)
\]

\[
f_r(l, q) = L_r(l) \cdot \frac{\beta_m(q)}{L_m(l)} = \frac{\beta_m(q) \cdot \gamma}{\beta_m} \sum_{i=1}^{n} L_s(I) \quad \ldots \ldots \ldots (4)
\]

\[
f_m(l) = L_m(l) = \frac{1}{\beta} \sum_{i=0}^{k} L_r(I) \quad \ldots \ldots \ldots (5)
\]

Where, \( L_s(I) \) is the server load factor, \( Q_r(q) \) defines the load at the rack uplink, \( \beta_r \) is a bandwidth over-provisioning factor at the rack switch, and \( \phi \) is a coefficient defining the proportion between \( L_s(I) \) and \( Q_r(q) \) in the metric. DENS load factor is defined as a sum of two sigmoid functions:

\[
L_s(I) = \frac{1}{1 + e^{-10(k(l-\frac{1}{2}))}} - \frac{1}{1 + e^{-12 \cdot \frac{1}{4} (l-\frac{1}{2})}} \quad \ldots \ldots \ldots (6)
\]

The first part of equation (6) defines the shape of the main sigmoid, while the second part servers as a penalizing function aimed at the convergence towards the maximum server load value. The parameter \( e \) defines the size and the incline of this falling slope. \( Q(q) \) is defined using inverse Weibull cumulative distribution function, given by equation (7):

\[
Q(q) = e^{-\frac{q}{Q_{max}}} \quad \ldots \ldots \ldots (7)
\]

DENS uses the following algorithm to compute the DENS metric during runtime:

Algorithm: Calculate DENS metric:

**Begin**

1. **Initialization**
   - Step 1 Set weighted coefficient \( a = 0.7, b = 0.2, y = 0.1 \)
   - Step 2 Set proportional coefficient \( \phi = 2 \)
   - Step 3 Get server load \( l \)
   - Step 4 Get queue size at access and aggregate switches \( q \)

**End**
II. Server selection

FOR all servers DO

Step 1 Compute server load $L_s(l)$, rack load $L_r(l)$, and module load $L_m(l)$

Step 2 Compute communications potentials of rack $Q_r(q)$ and module $Q_m(q)$

Step 3 Compute metric factors related to servers $f_s(l,q)$, racks $f_r(l,q)$, and modules $f_m(l)$

Step 4 Compute DENS metric as a weighted sum of $f_s(l,q)$, $f_r(l,q)$, and $f_m(l)$

END FOR

III. Select server with highest DENS metric

Thus, server with the highest DENS metric is selected for resource allocation.

HEROS: Heterogeneous Energy-efficient Resource allocation Optimizing Scheduler

HEROS is a load balancing algorithm for heterogeneous, energy-efficient resource allocation in data centers [6]. HEROS considers the heterogeneity [7],[8] of a system during the decision-making process and uses a complete representation of the system. Therefore, servers with resources of multiple forms (computing, memory, storage and networking) and have different internal structures of their components can be utilized more efficiently. Servers contain multiple parts, which are grouped by resource types. Each part is further presented by a vector of numbers, called capacities, which are quantitative represents their capabilities. Task descriptions are also fully heterogeneous, containing several requirements for the same type of resource. The HEROS methodology is based on DENS and e-STAB, and backward compatible. HEROS tasks are allocated to the server with maximum score. Score is calculated by a decision function, having two main parts: the server selection function and the communication potential function. Performance per Watt (PpW) metric is used to define energy efficiency and can be directly used to select the most energy efficient server. A PpW function is defined for server $s$ as given by equation (8): 

$$PpW_s(l) = \frac{Perf_s(l)}{Ps(l)}$$

where $PpW_s(l)$ is the PpW function, $Perf_s(l)$ is the performance function and $Ps(l)$ is the power function.

HEROS server selection function is defined by equation (9) as:

$$H_s(l) = PpW_s(l) \cdot (1 - \frac{1}{\gamma} e^{-\frac{1}{\gamma} \left( L_m(l) + L_r(l) \right)})$$

where $max l$ is the maximum load. The domain of $l$ is defined as $L_s := [0; max l]$. The second term is a sigmoid scaled to the domain $L_s$ and the range of $PpW_s(l)$ and its aim is to counter the impact of the PpW function form high values of load. The coefficient $\alpha$ defines sharpness of the descending slope, $\beta$ is based on the maximum acceptable load of the server. And $\alpha = 1.10$, $\beta = 0.9$, and $\gamma = 1.2$, to make sure there is a smooth degradation of the selection function starting from 90% of maximum load. The final decision function is obtained by multiplying the server selection function and the communication potential function, given by equation (10):

$$Fs(l,t) = H_s(l) \times Q_s(t)$$

The server having the maximal decision function value is chosen to execute a task. In case of a tie, the server is chosen randomly among the ones having the best value [10].


The energy-efficient scheduler for cloud computing applications with traffic load balancing (e-STAB) optimizes energy consumption of cloud computing data centers [2]. The communicational demands of the jobs are treated equally by the e-STAB scheduler with respect to the computing requirements. e-STAB scheduler aims to:

- (a) balance communication flows produced by the jobs
- (b) consolidate jobs on a minimum amount of the computing servers
- (c) reach the available transmission capacity
- (d) optimize energy consumption of cloud computing data centers

The e-STAB scheduling policy carries the following two steps that are executed for every incoming cloud computing data center workload:

Step 1: A group of servers $S$ is selected connected to the data center network with the highest available bandwidth, given that at least one of the servers in $S$ can serve the computational demands of the scheduled job. Available bandwidth is the unused capacity of the link or a set of links that connect the group of servers $S$ to the remaining data center network.

Step 2: Within the selected group of servers $S$, a computing server with the lowest available computing capacity is selected, but it should be sufficient to satisfy the computational demands of the schedule task.
In order to select the group of servers with the largest available bandwidth, following steps are taken:

**Step 1:** e-STAB selects a module using equation (11):

\[
F_{m_i}(t) = A_{m_i}(t) \cdot Q_{m_i}(t) = \int_{t}^{t+T} \left[ \frac{Q_{m_i}(t) - Q_{m_i}(t-\tau)}{Q_{m_i,\text{max}}} \right] d\tau 
\]

.....(11)

**Step 2:** e-STAB selects a rack using equation (12):

\[
F_{r_j}(t) = A_{r_j}(t) \cdot Q_{r_j}(t) = \int_{t}^{t+T} \left[ \frac{Q_{r_j}(t) - Q_{r_j}(t-\tau)}{Q_{r_j,\text{max}}} \right] d\tau 
\]

.....(12)

where Qmi(t) and Qrj(t) are weights associated with occupancy levels of the queues, qm(t) and qr(t) are sizes of the queues at time t, and Qmi,max and Qrj,max are maximum allowed sizes of the queues at the module i and rack j, respectively. After selecting a proper module and a rack based on the traffic load and congestion state given by the queue occupancy, a computing server is selected for the job execution. The e-STAB scheduler uses a metric for server selection as given by e-STAB selects a module using equation (13):

\[
F_{s_k}(t) = \frac{1}{T} \int_{t}^{t+T} \left[ \frac{Q_{s_k}(t) - Q_{s_k}(t-\tau)}{Q_{s_k,\text{max}}} \right] d\tau
\]

where lk(t) is an instantaneous load of server k at time t and T is an averaging interval. A joint metric calculated by the e-STAB scheduler for every incoming workload is given by equation (14):

\[
G_{i,j,k}(t) = (F_{m_i}(t), F_{r_j}(t), F_{s_k}(t))
\]

.....(14)

### IV. COMPARATIVE STUDY OF ALGORITHMS

The following section gives a comparison of the performances of various data center energy-efficient load balancing algorithms.

**DCEERS:**

As compared to DENS, DCEERS (a) includes both server and network components (b) saves up to 70% of energy in the data center, and (c) DCEERS adapts promptly to workload fluctuations since Benders decomposition can solve the optimization problem in linear time. The DENS methodology selects the best fit data center resources with sufficient communicational capability. DENS methodology runs devices in three states: power on, power off, and power scaled. The power scaled devices can be switched to power on state with minimum downtime as compared to power off devices. But power scaled devices still consume energy. DCEERS does not operate devices in power scaled state, thus, achieves better energy efficiency than the DENS methodology. Although, the running time of DCEERS is more than that of DENS, the energy consumption of DCEERS is less than that of DENS [4].

**e-STAB:**

e-STAB scheduler is different than the DENS scheduler in the following: a) the way network traffic is analyzed, b) possibility to load balance the traffic and c) metrics used for the selection of the data center servers, racks and modules. The Green scheduler often selects multiple servers from the same rack, having performed no analysis of the network traffic. The e-STAB scheduler uses the same number of servers for servicing user demands in total. Servers are selected in bulks which are distributed evenly through the whole of data center infrastructure. It causes equal distribution of traffic load among all the racks in the data center [2].

**HEROS:**

Distributed Flow Scheduling (DFS) is an algorithm for energy-aware data center networks. It does not consider the communication sources and sinks, neither the corresponding computation nor data storage needs. HEROS is more complete. It considers multiple resources that are used during the data centers activities. DCEERS appoints minimum set of resources to the workload by finding the minimum cost of a multi commodity flow using the Benders decomposition. DCEERS shows an improved performance as compared to DENS, but this comes at the cost of increased runtime (approximately 10 times). DCEERS is also restricted to homogeneous data centers. Another limitation of DENS is that it can only be applied to the three-tier architecture, which is commonly used, but can have multiple alternatives (e.g. DCell, BCube, FiConn, DPillar). But, e-STAB is a two step-algorithm requiring online knowledge of the full data center network utilization. Thus, the load among racks in e-STAB is better distributed than that of DENS. Even the server selection functions of these algorithms have different shapes, resulting in opposing behaviors: while DENS uses DNS, e-STAB promotes low utilization and
prohibits consolidation. HEROS combines the best features of DENS and e-STAB and proposes a heterogeneity-aware decision making approach. It accepts any standard network topology, as it operates on the rack level. Still, similarly to e-STAB, the network load is balanced among multiple racks. HEROS has better mean response time and total energy consumption as compared to DENS and Green schedulers in small homogeneous topology. Whereas, for full scale homogeneous topology, DENS is the most energy-efficient algorithm, followed by Green and HEROS. The mean response time is the best for HEROS, closely followed by DENS. For heterogeneous topologies, HEROS gives the best performance in all the parameters [6].

V. SIMULATION RESULTS

Green Cloud Simulator [5] was used to simulate DENS, e-STAB and HEROS algorithms. Three parameters were used to compare these algorithms, i.e. Energy Consumption of the entire setup, Power Consumption of servers and Energy Consumption of servers, as shown in figures 2, 3 and 4. One cloud user and 144 servers were set to perform the simulations. HEROS algorithm was performed in a three-tier heterogeneous data center infrastructure whereas DENS and e-STAB algorithms were performed in three-tier homogeneous data center infrastructure. It can be seen that DENS algorithm gives the best performance in terms of energy consumption in a homogeneous environment whereas HEROS gives the best results in a heterogeneous environment (figure 2). The number of servers needed to perform the tasks is very less in DENS as compared to e-STAB resulting in less energy consumption in DENS algorithm. Also, the number of servers in HEROS algorithm is very less resulting in less energy consumption. The workload is also better distributed in HEROS algorithm (figure 3 and 4).
VI. CONCLUSION

Green cloud computing is a growing trend in the world of cloud computing and data center infrastructure. It is a domain that cannot be neglected. Green computing deals with minimizing the energy consumed by data centers by designing power-aware load-balancing and resource-allocation algorithms. This can be achieved by restraining the number of servers required to service a request to a minimum set and powering off rest of the servers. Some of these algorithms are discussed in this paper.

The algorithms included in this paper are DFS, DENS, DCEERS, e-STAB and HEROS. As described in comparative study, DFS is the least significant approach as it does not consider the communicational links of the data center. DCEERS gives better performance than DENS as it consumes less energy. e-STAB has an opposing nature as to DENS. This is because DENS uses Dynamic Shutdown (DNS) whereas e-STAB promotes low utilization. Finally, HEROS combines the best practices of both DENS and e-STAB and gives the best performance as compared to all the algorithms. HEROS can be used in both homogeneous and heterogeneous data centers to provide energy-efficient load balancing.

4. References


