DYNAMIC ENERGY AWARE CLOUDLET MODEL WITH EFFICIENT OFFLOADING ALGORITHM IN CLOUD COMPUTING
CHAPTER-3
DYNAMIC ENERGY-AWARE CLOUDLET MODEL WITH
EFFICIENT OFFLOADING ALGORITHM IN CLOUD
COMPUTING

Based totally on the present study of the literature of mobile cloud computing we've got observed the software development models primarily based on both the energy and performance moreover every software version has its own advantages and drawbacks.

The simple consideration for cell gadgets is energy performance while the cloud computing has the possibility to store cell client strength however the savings from offloading the computation fee and time is want to exceed the energy cost of the extra conversation.

Many packages on smart phones might also once in a while bring about negative performance and less life of the battery because of its limited sources. So it results in plenty of task to growth the electricity efficiency and the improvement of useful resource-limited gadgets.

Cellphone at the moment are capable of helping a wide variety of programs, lots of which demand an ever growing computational strength. There ought to be a utility improvement model which should shorten the reaction time and store battery as well. An offloading framework that objectives to shorten reaction time and reduce battery existence as well is the need of the hour.

3.1 Research Methodology

The main aim of this research is used to improve the energy efficiency and resolve the waste energy in the mobile cloud computing. It is used to resolve the additionally energy consumption in the wireless communication, by Here by proposing the energy
aware model in the dynamic cloudlet the additional energy consumption can be resolved and it reduces the latency delay.

To minimize the energy consumption in the mobile devices, we propose the offloading algorithm with the energy aware model for resolving additional unwanted energy in the MCC.

In proposed offloading algorithm, we incorporate the data caching mechanism and improve task management strategies with dynamic energy scheduling algorithm with time constraints for improving the energy optimization while performing the task in the dynamic cloudlet, which ultimately improves the overall performance of the mobile cloud computing.

It rejects the unused or unwanted task from the mobile environment for reducing the traffic overload occurring between the mobile devices and cloud servers. The proposed method can obtain energy optimization and removes the waste energy form the mobile devices, it reduces the latency delay and traffic overload in the Mobile cloud computing.

Figure 3.1 Proposed Architecture Diagram

In proposed model, by introducing the advancement deployment model, Figure 3 explains the dynamic energy optimization in mobile cloudlet with an energy model in the offloading strategy, it provides a novel approach that it would avoids the wastage of
energy when a mobile users are tolerating with complicated and unstable networking surroundings.

This methodology concentrates on providing best communication between the cloud servers and mobile users.

Figure 3.2 Monolithic Mobile Application Running on a Mobile Device

Figure 3.2 demonstrates the energy efficient offloading model. It demonstrates that, with calculation offloading, a circulated application execution will be apportioned between the mobile devices which should contain no less than one execution module.

For example, the client interface and one or more servers which can be utilized for calculation offloading as a part of request to enhance the execution or diminish energy consumption for the mobile device. Regularly, figuring out which divides of a calculation to offload is given a role as a chart parceling issue.
Our proposed Energy-Efficient Offloading (EEO) models the project to be parceled as a Weighted Object Relation (WOR) Graph, with hubs speaking to the calculation module (a run time object of the application), and edges speaking to the collaboration between modules (e.g., summons between one item and another). In a WOR, the heaviness of an edge demonstrates correspondence costs (in force) of the connection between two modules, while the heaviness of a hub speaks to the calculation power consumption of the item module. The objective of this paper is to minimize the energy/power consumption by calculation offloading.

The aggregate expenses of the considering so as to parcel can be ascertained both the weights of edges for correspondence and the weights of hubs for calculation to get the best exchange off. The ideal apportioning scheme implies the ideal decision of modules to offload.

3.2. Graph Construction

Concerning the previously stated weights of hubs and edges of WOR, they can be evaluated by either static investigation or profiling of the project. EEO first apply builds the beginning ORG (Object Relation Graph) of the application by utilizing the investigation system to perform the static focuses to examination. And after that disconnected from the net profiling is performed to appoint weights to the hubs and edges of the ORG to build the WOR. Figure 5 demonstrates a WOR which we develop by both the static examination and disconnected from the net profiling strategies for an application.
3.3 Problem Formulation

A graph partition $WORG=(V,E)$ with vertices set and edges set $E \in V \times V$, and a set of $K+1$ partitions denoted as $P=\{P_0, P_1, ..., P_K\}$ ($P_0$ represents the mobile device, and $P_1, ..., P_K$ represent the offloading sites, is the number of offloading sites). As shown in Figure 5, the weight of the vertex $v$ is described as a 2-tuple $(t_c(v), t_s(v))$, where $t_c(V)$ indicates the CPU execution time for each object running on the client, and $t_s(V)$ is that for each object running on the server. $t_s(V)$ can be calculated by $t_c(i)/k$, where $k$ indicates that the server is $k$ times faster than the mobile device.

Each edge $e_{V_1,V_2}$ is associated with a weight $(s(V_1,V_2))$ indicating the amount of the total data that need to be transmitted between two nodes. EEO collects $(t_c(V), t_s(V))$ and $(s(V_1,V_2))$ metrics by offline Profiling [7] during the WOR graph. We can define the multi-way dividing issue as the 0-1 ILP issue. Our goal is to minimize the energy consumption, that is, the value of the following objective function:

**Figure 3.3 Weighted Object Relation Graph of an Application**
\[ \text{Energy}(WOR) = \sum_{v \in V} (E(v_1).x_l + E(v_s).x_s) + \sum_{v_1 \in V, v_2 \in V} |x_l - x_s|.E(e(v_1, v_2)) \]  

(1)

where \( x_l \) and \( x_s \) indicate the assignment of each node: \( x_l = 1, x_s = 0 \) if vertex \( V_l \) is assigned to the client and \( V_s \) is assigned to servers, \( x_l = 0, x_s = 1 \) otherwise. Equation (1) is subject to the following constraint:

\[ \forall V \in V: x_l + x_s = 1. \]  

(2)

\((V_l)\) and \((V_s)\) are the energy consumption of vertex \( V \) running on the client and the server, respectively. They can be computed through the following (3):

\[
(V) = P_c x (V), \ V \in P_0,
\]

\[
E (V_s) = P_s x (V), V \in P_1 \ldots P_k,
\]

(3)

where \((V)\) and \(ts(V)\) are the weights of vertex \( V \) when running on the customer and on the servers, separately. \(P_c\) and \(P_s\) are the force CPU of the customer or the servers. \((eV_1, V_2)\) is the vitality utilization for information transmission between vertex \( V_1 \) and vertex \( V_2 \) when they are not running on the same site, for instance, one running on the customer and the other on the servers. \((eV_1, V_2)\) is computed by (4):

\[
E (eV_1, V_2) = s (V_1, V_2) b x P_{wi-fi},
\]

(4)

where \((V_1, V_2)\) is the weight of the edge between vertex \( V_1 \) and \( V_2 \). \( b \) demonstrates the system data transfer capacity and \( P \) Wi-Fi is the force of the remote Wi-Fi system interface.

To minimize the value of (1), the key is to determine the value of \( x_l \) and \( x_s \), that is, 0 or 1. As remote servers generally executed much speedier than cell phones with
capable design, it can spare vitality and enhance execution to offload a portion of calculation to servers. In any case, when vertexes are appointed to various locales, the cooperation between them prompts correspondence cost.

In this way, our issue definition goes for the ideal task of vertexes for diagram parcelling and calculation offloading by exchanging off calculation expenses and correspondence costs.

3.4 Partitioning Steps

Input: WOR graph= (V, E), B, b, NL, a

Output: Xmin—the optimal partitioning scheme, MinEnergy—the minimal energy consumption

1. Compute the minimum energy consumption when bandwidth = b using the Stoer-Wagner algorithm, noted as minE

2. \( \text{minE} = \text{minE}(1 + a) \)

3. For Vi in V

4. If Vi in NL

5. \([i] = 1; // vertexes running on the client\)

6. Else

7. \([i] = -1; // vertexes to be partitioned\)

8. End if

9. End for

10. DF Search(1, minE, WORG, X, Xmin, MinEnergy)

11. Return \{Xmin, MinEnergy\}
3.5 Offloading Algorithm

We perform a multi-way graph partitioning based algorithm to solve the ILP problem. First, we transform the WOR to a directed Acyclic Graph (DAG) and perform the topologic sort. At that point, we utilize the profundity first inquiry to navigate the pursuit tree and register the $\text{Ene}(G)\text{Band} \ Energy(G)b$ for each encountered nodes, where $B$ is the current bandwidth, and $b$ is the basic data transmission that meets $P\{B \geq b\} > Pc$.

$Pc$ is the guaranteed probability, and $(G)$ represents the energy consumption of the particular partitioning scheme when bandwidth is $b$. During the search, if $(G)$ does not fulfill the constraints or $Energy(G)B$ is larger than the current minimal energy ($\text{MinEnergy}$), that is, $Energy(G) B > \text{MinEnergy}$, the sub-tree of the hub will be decreased to the guardian hub to keep seeking. In the wake of navigating the entire tree of the DAG, we will get the ideal dividing that satisfies the given imperatives.

The apportioning calculation is appeared as Algorithm 1, where $NL$ is the hub accumulation which runs locally on the customer, and $d$ is an exact steady to set the limitation of the minimal energy.

3.6 Performance Analysis

The assessment measurements are energy consumption and execution time. We perform the examinations on three irregular diagrams produced by specific schemes to mimic this present reality situation including a customer device and two remote servers. The dataset utilized as a part of examinations are recorded in Table 1 as takes after. We assume that the application is at first situated on a mobile device and the data transmission changes between the estimation of 10 kb/s and 100 kb/s.
Table 3.1 Random Graph for NAP, SAP, EEO

<table>
<thead>
<tr>
<th>Random Graph</th>
<th>Number of Nodes</th>
<th>Number of Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph 1</td>
<td>15</td>
<td>60</td>
</tr>
<tr>
<td>Graph 2</td>
<td>30</td>
<td>350</td>
</tr>
<tr>
<td>Graph 3</td>
<td>100</td>
<td>3238</td>
</tr>
</tbody>
</table>

3.6.1 Energy Consumption Evaluation

From table 1, we can see that the communication cost (i.e., \((eV1,V2)\)) is critical to partitioning decision, and it is directly related with the network bandwidth. However, in most cases the system data transfer capacity changes progressively, particularly in remote systems of cell phones. The data transmission is considered as a variable to enhance the element of parceling in our EEO. To assess the adaption of EEO to data transmission transforms, we look at the vitality utilization of three algorithms with bandwidth changes. The experimental results with bandwidth varying in steps of 10 kb/s are presented in Figure 6. As shown in Figure 6, from Random Graph 1 with fewer nodes and edges to Random Graph 3 with the most nodes and edges, the energy consumptions of all three approaches increase, because the computation become larger and more complex as the nodes and edges grow.
Figure 3.4 Energy (Power) Consumption Comparisons with Varied Bandwidth

NAP consumes the constant energy with increasing bandwidth because the whole application keeps running on the mobile device without offloading and without energy costs of communication. As the bandwidth changes between 10 kb/s and 100 kb/s, the energy consumption of SAP varies more severely than that of EEO. When the bandwidth becomes lower, SAP still maintains the former partitioning scheme,
resulting in great increase of communication costs. However, our EEO algorithm can find the better partitioning assignment when network bandwidth changes.

Particularly, when bandwidth >20 kb/s, EEO algorithm saves about 25% energy compared to SAP. Meanwhile, when bandwidth >20 kb/s, our proposed EEO algorithm also outperforms NAP due to partitioning approaches. The results demonstrate that EEO is effective and beneficial to perform the partitioning for mobile devices.

3.6.2 Execution Time Evaluation

To further evaluate the performance of our EEO algorithm, we estimate the total execution time of various algorithms as another evaluation metric to perform the partitioning in Figure 3.5. If it takes lower time to execute a mobile application, it is useful for energy conversation of mobile devices and also improves the user experience with high-efficiency execution.
Figure 3.5 Total Execution Time Comparisons of Different Algorithms
As shown in Figure 3.5, we can see that our EEO executes the computation offloading much faster than NAP without code offloading and faster than SAP with static partitioning, which demonstrates that, compared to NAP and SAP, our proposed EEO can altogether enhance execution time and diminish vitality utilization for asset limited cell phones. Furthermore, the outcome that execution time for Random Graph 3 is much bigger than the other two (Random Graph 1 and Random Graph 2) additionally lives up to our desires due to its more hubs and edges contrasted and the other two.

As a conclusion, as shown in Figure 3.4 on energy consumption and Figure 3.5 on total execution time, it is obvious that our proposed EEO algorithm effectively saves the most energy and takes the least execution time to perform the partitioning, which is essentially advantageous for energy conservation of mobile devices.