2. LITERATURE SURVEY

2.1 INTRODUCTION

This chapter presents the existing research works related to the energy-efficient MCC framework. It describes the prior works in the cloud computing and the mobile cloud environment. Also, it reviews the conventional techniques in the process of application partitioning, offloading, and remote execution methods. It reviews the contemporary and historical work on MCC, focusing on the SLA based service provisioning. This survey seeks to explore the alternative approaches to the various applications in MCC and to identify the issues in the prior works which are resolved by the subsequent chapters in this thesis. Section 2.2 presents the existing cloud computing works involving resource allocation, task scheduling, load balancing, optimization, and workflow management techniques. Section 2.3 reviews the contemporary works related to the resource allocation and task scheduling approaches in MCC. Section 2.4 and 2.5 briefly discuss the dynamic load balancing and optimization approaches based on the energy constraint. Finally, Section 2.6 presents the computing techniques involved in the workflow based IoT application execution in the mobile cloud.

2.2 ADAPTING CLOUD COMPUTING TECHNIQUES FOR MOBILE CLOUD ENVIRONMENT

Mobile cloud environment differs from the cloud environment due to the numerous identical requests, local execution dependent remote execution, and dynamic context-aware analysis of the mobile device based resource allocation and task scheduling [32]. To maximize the resource utilization, the profit of the provider, and enhance the user experience, adopting the cloud computing techniques is crucial to a mobile cloud environment. While adopting the cloud computing techniques to a mobile cloud environment, the system needs to concentrate on several factors, including the demand of the mobile users, the limited capabilities of the mobile devices, battery constraint, communication fault-tolerant, network latency, and unreliable connectivity [33].
In recent years, IoT has become an increasingly growing and influential technology. The IoT is one of the most popular buzzwords in both the academic and industry research. It enables the people and things to establish the connection at anyplace, anytime, with anyone and anything, using any service over the Internet. It offers the benefits to various sectors such as manufacturing, transportation, and remote health care systems. Due to the massive storage of the real-time streaming data from the IoT devices, the mobile devices are incapable of storing the entire data itself due to the resource-scarcity. Also, the workflow of each IoT application is likely to vary according to the application domain, dynamic requirements, and processing capability of the mobile devices. Hence, the IoT system requires the adaptive MCC techniques to deal with the workflow based application execution and resolve the constraints in IoT systems. The design and implementation of IoT applications in the form of layered architectures, including three layers such as a bottom layer, middleware, and an application layer. The bottom layer contains the deployed IoT devices; middleware layer presents the deployed IoT infrastructure in a unified view, and an application layer executes the application logic.

2.2.1 Resource allocation techniques

The existing resource allocation techniques for cloud computing is discussed as follows:

Resource allocation and scheduling strategy maximize the cloud provider's profit and satisfy SLA using dynamic rank based resource allocation and First In First Out (FIFO) scheduling was discussed by Boloor at al., [56]. It considers the context-aware application requests in distributed cloud. In a multi-tier cloud computing environment, the force directed searching method based algorithm developed by Goudarzi Hadi and Massoud Pedram [57] allocates the resources to maximize the total profit while meeting SLAs. It provides processing, memory and communication resources to the user based on the SLA. Adaptive resource allocation proposed by Li J et al., [58] employs task execution time and preemptable scheduling to overcome the resource contention problem and increase the
resource utilization. The cloud users enable the users to set up and boot the resources based on their requirements and then, users can pay merely for the utilized resources from the cloud. It facilitates the users to dynamically add or delete the VM instances based on the VM load [59]. However, the effective resource allocation technique requires more and deeper knowledge for allocating and managing the cloud resources precisely.

2.2.2 Task scheduling techniques

In cloud computing, task scheduling is the process of scheduling the tasks into the cloud resources, involves three steps such as discovering and filtering the resources according to the input request task, selecting the appropriate resource, scheduling a specific task on the target resource. However, in MCC, the task scheduling requires the offloading decision based on the device characteristics. Then, the scheduling technique partitions the application into mobile tasks and cloud tasks before scheduling the application on the cloud server. The scheduling process of the current task depends on the preceding task of an application. The task scheduling techniques involved in cloud computing are First-Come-First-Served (FCFS), Min-Min, Max-Min, and Round-Robin scheduling algorithms. Also, meta-heuristic algorithms are simulated annealing [60], Tabu search, GA [61], Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO) algorithms [62, 63]. The conventional task scheduling techniques in cloud computing are the following: The QoS-driven task scheduling algorithm (TS-QoS) developed by Xiaonian Wu et al., [64] prioritizes the task depends on the special attributes of the tasks and achieves load balancing in cloud computing. The prioritization in TS-QoS changes dynamically to solve “starvation” issue and adopts FCFS strategy using expected completion time of the task. However, this task scheduling algorithm considers only the execution completion time for the current task scheduling and allocation. It increases the overall application completion time of mobile application request, thus, it degrades the performance of the entire system. Baomin Xu et al., were developed the Berger model [65] that has designed an algorithm which introduces the dual fairness constraint. The first constraint categories
user tasks based on QoS preferences and launch the general expectation function. The second constraint defines the resource fairness, justice function according to user's request. Energy-aware workload placement model proposed by Feller et al., [66] solves the Multidimensional Bin-Packing problem (MDBP) using the nature-inspired algorithm. The dynamic placement of the workload depends on the current workload and ACO meta-heuristics. However, this algorithm do not support hardware heterogeneity. In addition to that, virtualization technology resource isolation properties like colocation of workload with same characteristics on the same physical machine can lead to performance degradation. Therefore, the task scheduling techniques need to consider the workload characteristics in MCC environment.

2.2.3 Load balancing techniques

Load balancing is one of main challenging task due to the submission of dynamic input requests onto the cloud. The load represents the memory, network, and CPU capacity. Load balancing is the distribution of the dynamic workload across the multiple cloud resources by averting the overloaded and underloaded resources. The load balancing is being used to improve the resource utilization and performance of the system by preventing the interruption of task execution. The load balancing techniques in the cloud computing environment are:

A VM resource scheduling strategy based on genetic algorithm proposed by J. GU et al., [67] balances the load based on the genetic algorithm in a cloud environment. It considers the historical information and current load of the system while scheduling the resource. ACO based load balancing in a cloud environment suggested by Mishra et al., [68] considers the routing packets as the ants in the cloud environment. It replaces the routing tables with a probability value of pheromone tables which contains the information of pheromone value and incremental pheromone update. Sotomayor B et al., have discussed Round Robin technique [69] and it is a static load balancing technique which divides the workloads into all available processors in which the allocation order is independent of the remote
processor execution. This technique lacks in analyzing the dynamic workload. Hence it leads the heavily loaded resources on the remote server. The task scheduling framework developed by Huanqai Chen et al., [70] ensures the satisfaction of customer’s demand with high QoS and load balancing, minimizes the makespan, and maximizes the resource utilization using User-priority aware - Load balance improved min-min scheduling algorithm (PA-LBIMM). However, this algorithm considers only independent tasks, but mobile applications have a precedence relationship in a real-time scenario. A scheduling strategy employs a genetic algorithm [71] to achieve load balancing and reduces dynamic migration. Moreover, it minimizes the migration cost and increases the resource utilization while task scheduling in cloud computing.

2.2.4 Optimization techniques

In cloud computing, optimization focuses on the several factors such as resource provisioning, task scheduling, cost, and cloud service optimization. It aims at resolving over-provisioning and under-provisioning problems, improving the service quality, and reducing cost and response time. The optimization involves in both the processes, including scheduling the tasks and allocating the cloud resources. Thus, several cloud optimization approaches are described as follows:

A macroscopic job scheduling model based on Multi-Objective Genetic Algorithm (MO-GA) developed by Jing Liu et al., [72] involves encoding rules, genetic algorithm, the crossover operation, and optimal selection in Pareto archive. Meta-scheduler employs the MO-GA for minimizing the energy consumption with QoS provisioning in cloud infrastructure. The workload placement problem and operator cost occur highly during the MO-GA scheduling in the cloud environment. Hybrid cloud-optimized cost scheduling (HCOC) algorithm proposed by Luiz Fernando Bittencourt and Edmundo Roberto Mauro Madeira [73] that reduces makespan, monetary costs and minimizes task computation time before its deadline. However, this algorithm does not consider multiple workflows between private resources for improving scheduling decisions. Beloglazov et al., proposed an energy-
efficient resource management method [74] and it exploits live VM migration during reallocation of VMs in cloud data centers. Live VM migration minimizes the power consumption and provides quality of service. A PSO scheduling method analyzed by Wu Z et al., [75] only concentrates on the workflow execution and minimizes the execution cost, improves the overall efficiency and convergence rate and realizes task-resource mapping using a variation in communication cost, and it balances the load on resources using task distribution. A Pareto-based genetic algorithm suggested by Kessaci et al., [76] schedules the high-performance computing applications using a Pareto approach to identify the optimal solution.

Energy-efficient management work optimizes the data center resources to reduce power consumption of data center and SLA violations in a cloud environment. Energy-aware resource allocation and provisioning algorithm proposed by Buuya Rajkumar et al., [77] allocates the VM resources to cloud resources in an energy-efficient manner. Energy consumption optimized for cloud computing based on the task tolerance (ECCT) approach developed by Hao et al., [78] increases the resource utilization, minimizes power consumption using a resource scheduling optimization algorithm. Here, the optimization depends on the growing task tolerance of resource execution in cloud computing. The resource allocation framework developed by Chrysa Papagianni et al., [79] solves the optimization problem of Mixed Integer Programming (MIP) problem in a cloud environment. Resource mapping procedure offers QoS-aware resource provisioning and costs adequate computing resources. Resource allocation optimization ensures high resource utilization and the right balance between SLA violations and resource utilization in a cloud environment using bin packing approach based algorithm. Resource allocation considers the VM resource allocation of PM resources depends on the demand and capability of resources on the cloud environment [29]. Beloglazov Anton and Rajkumar Buyya developed an Energy-efficient resource provisioning system [80] optimizes the VM resources in cloud data centers. Here, VM placement and VM reallocation depend on the CPU, RAM, and bandwidth utilization. Resource management system minimizes the energy consumption with the
consideration of QoS requirements during resource allocation. Off-the-Cloud service optimization (OCSO) concept developed by Daren Fang et al. [81] discusses the heuristic-based off-the-cloud to optimize the resource consumption using an intelligent resource scaling algorithm. The OCSO enables IaaS and PaaS users' resource utilization using VM up/down and VM in/out respectively in dynamic workload cloud environment. However, this OCSO concept lacks on providing more service provider support to achieve service consumption efficiency.

2.2.5 Workflow management techniques

In a dynamic cloud environment, the Bi-criteria strategy proposed by Bessai et al. [82] tackles the allocation and scheduling shortcomings in the workflow tasks by three bi-criteria approaches with the objective of minimizing the execution cost and overall completion time, and optimally selecting the non-dominated solutions. Deadline and cost based workflow scheduling model proposed by Chopra et al. [83] divides the workflow tasks into the number of levels and then schedules the workflow tasks based on the levels in private and public cloud. In each level, the tasks are the independent tasks which are to be executed in a parallel manner. The work in [84] developed by Caron et al., presents the two resource allocation strategies based on the budget constraints, which decomposes the scheduling problem of non-deterministic workflow into a set of deterministic sub-workflows. The complex workflows comprise the recursively parallel sub-flows of the tasks, which leads the difficulty in deploying such workflows in the heterogeneous cloud computing nodes. To overcome this, Jung et al., [85] proposed an approach and it presents the workflow scheduling algorithm maintaining the trade-off between the computing time, execution cost, and the transfer delay between the sub-tasks. A Fault Tolerant Workflow Scheduling (FTWS) method developed by Jayadivya S.K et al., [86] employs the task replication and re-submission procedure based on the task priority. It maintains the trade-off between the replication and re-submission factor with the deadline satisfaction in which it performs the task replication process in the scheduling phase and re-submission process in the execution phase.
Singh et al., discusses a genetic algorithm [87] and schedules the workflow applications within the user budget, which reduces the makespan and the failure rate of the workflow application while scheduling in the unreliable cloud environment. The workflow task dependency based scheduling methods in cloud environment also applicable to a few types of workflow scheduling applications in the mobile cloud environment.

### 2.3 RESOURCE ALLOCATION AND TASK SCHEDULING APPROACHES IN MCC

Resource allocation and task scheduling play a key role in the cloud environment. In a mobile cloud environment, multiple mobile users access the same applications, but they can only differ in offloading part of an application according to the mobile characteristics. Hence, the numerous growing requests of identical application increases the complexity of the resource allocation and task scheduling processes. To provide the efficient cloud services to the mobile users, extending the cloud computing resource allocation and task scheduling techniques is necessary based on the device application characteristics, which can mutually improve the satisfaction of the mobile user and efficient resource management in the cloud [34].

#### 2.3.1 Dynamic resource allocation approaches

The resource allocation [88] can be enriched to improve the mobile user experience by integrating the cloud environment with the mobile devices via context-aware resource allocation. In a mobile cloud environment, multiple mobile users request the similar mobile applications at the same time, hence accomplishing the high performance of allocating the cloud resources is a challenging task. The research works of mobile cloud based dynamic resource allocation are discussed as follows:

A nested two-stage game-based optimization framework (NTGO) developed by Wang Hinayana et al., [89] takes effective offloading decision for minimizing device energy and improving the response performance in the first stage. Cloud computing controller allocates the resources for increasing
provider's profit in the second phase. This framework considers ACO with brute-force searching method, and this approach gradually decreases profit of the cloud service provider based on varying number of requests. An agent-based optimization framework addressed by Angin Pelin and Bharat Bhargava [90] minimizes the requirements for cloud server using the mobile-agent-based partition. The mobile agents partition the application based on the device energy for taking an offloading decision. However, this optimization framework uses static performance profiler to capture the performance for every possible application partition and it hard to specify all possible problem size for explicit application. An adaptive computing resource allocation concept proposed by Liang et al., [91] maximizes the resource utilization and reward function of the system. The Semi-Markov decision process (SMDP) based resource allocation, achieves the objective function during consideration of both mobile device and cloud. Cloud-assisted motion estimation (CAME) approach suggested by Zhao et al., [92] discusses the resource allocation of the cloud resources for mobile video compression and estimation. It saves the mobile device energy while processing the compute-intensive applications of mobile video streaming. However, this approach does not consider optimization problems between cloud service provider and mobile devices. This degrades the transmission overhead and lower energy-efficiency. Rahmi. M et al., suggested an MAPCloud [93] resolves the Non-deterministic Polynomial-time hard (NP-hard) optimization problem using simulated annealing based heuristic. It allocates the cloud resources for mobile applications based on the 2-Tier cloud architecture. Auction based mobile cloud approach developed by Zhang et al., [94] allocates the bundle of cloud resources to the mobile user and analyzes the incentive factors using auction mechanism. In [95], Liu et al., presented a Dynamic programming based offloading algorithm (DPOA) which solves the optimization problem during offloading based on the optimal partitioning.
2.3.2 Energy-efficient task scheduling approaches

Mobile cloud task scheduling requires the dynamic contextual information about the local execution tasks in the mobile device and resource availability in the mobile device to perform the task scheduling of a mobile application on the remote server. Several mobile cloud task scheduling approaches schedule the mobile application based on the local execution, task dependency, and dynamic resource availability. An Energy-Efficient, Cooperative Offloading Model (E2COM) developed by Song et al., [96] schedules the tasks with the minimization of mobile user energy consumption and internet data traffic redundancy in a mobile cloud environment. The task scheduling algorithm depends on the pricing mechanism and Lyapunov optimization to reduce energy consumption in Wireless Local Area Network (WLAN). However, this model does not consider the implications of the QoS constraints of different type applications. Energy-optimal MCC framework developed by Zhang et al., [97] solves scheduling problems of constrained optimization problem to minimize device energy. It focuses on both execution and transmission of mobile applications on the mobile device or cloud using the Lagrangian multiplier method. An effective distributed parallel scheduling model develops a simulator for analyzing the bottleneck in terms of device energy and quality of the network in MCC. Collaborative execution concept proposed by Weiwen Zhang et al., [99] depends on energy-efficient scheduling policy to minimize the energy consumption in the mobile device and the cloud using Lagrangian Relaxation Based Aggregated Cost (LARAC) algorithm and collaborative execution of power consumption reduction. Greenspot algorithm addressed by Namboodiri et al., [100] saves the mobile device energy while executing the mobile applications on either cloud or mobile device. It preserves the energy-efficient battery level of the mobile device for various cloud-based mobile applications. The weakness of the algorithm is that it does not group the tasks for execution and does not better estimates of relative processing capability between mobile devices and cloud server. MobSched developed by Suraj Sindia et al., [101] ensures the MapReduce job performance with QoS that minimizes the power consumption and latency using the customizable scheduler for mobile cloud
computing. The customizable scheduler considers different factors on mobile devices and the MapReduce framework considers the nature of the wireless mobile ad-hoc network. Incremental Cost-Based Scheduling (iCBS) approach presented by Yun Chi et al., [102] discusses the CBS performance under piecewise linear SLAs and minimizes the computation time of tasks while the online scheduling process using cost-aware scheduling algorithm.

2.4 DYNAMIC LOAD-BALANCING APPROACHES IN MCC

Unlike cloud load balancing, the mobile cloud load balancing requires both the load balancing among the resources in the cloud environment and load balancing among the tasks of an application. Most of the multimedia mobile applications often need the adaptive workload balancing method, since the multimedia applications have the nature of dynamically changing workload conditions. The MCC system leverages the load balancing concept on the tasks of an application to provide elastic and scalable computation and storage benefits to the mobile cloud users while ensuring the optimal profit of the service provider. It leads the system to complete the application execution within a short-time period. Scalability enables the VM migration across the heterogeneous cloud resources within a data center or multiple data centers. The main objective of VM migration is especially achieving live VM migration in the workload balancing over the runtime environment. The mobile cloud load balancing techniques enriches the cloud load balancing techniques that include the following:

An Energy and Performance-Aware (EAPA) task scheduling work proposed by Xue Lin et al., [103] reduces the energy consumption and total completion time of the tasks using an initial task scheduling algorithm with the number of cores, and it migrates the tasks for energy minimization using linear-time rescheduling algorithm in a mobile cloud environment. However, the overall computation complexity is affected by the overall completion time constraint when dynamic mobile application requests are processed in a mobile cloud environment. A prognostic load balancing strategy analyzed by Ahmad et al., [104] minimizes the energy consumption of the mobile device and improves the response time and scalability. It reduces the latency of the
system using the amplified-ESBWLC algorithm. Lei Ying et al., [105] proposes the stochastic model that achieves load balancing and VM scheduling using a routing algorithm (Join-the-Shortest-Queue) and Myopic Max Weight scheduling algorithm in cloud computing. Hybrid ant colony algorithm based application scheduling (HACAS) algorithm proposed by Xianglin Wei et al., [106] ensures load balancing with satisfactory QoS requirements and achieves high profit and low energy consumption. However, this algorithm does not consider the dynamic resource requirements of the mobile applications. It considers only static resource requirements while scheduling the multiple application requests in a dynamic mobile cloud environment. The consideration of current task execution time and the availability of static resources increases the load deviation value, in turn, it increases the application completion time and energy level of the mobile devices.

2.5 EXECUTION OPTIMIZATION APPROACHES IN MCC

The mobile cloud optimization approaches additionally focus on the device energy, application-specific optimization, offloading optimization, and resource optimization. The cloud optimization techniques are extended by the additional mobile and application characteristics to improve the overall performance of the mobile cloud execution in which optimization includes both the mobile device and cloud server. Mobile data offloading research by Lee K et al., [107] and Han et al., [108] discusses the process of data transfer on wireless local area networks and cellular data networks with QoS. A novel framework introduces the joint allocation approach developed by Barbarossa et al., [109] that optimizes the communication and computation resource capabilities in a mobile cloud environment. The Joint optimization considers the multiuser application transmit power minimization, latency reduction, and overall energy consumption of mobile devices. eTime approach purposed by Shu et al., [110] ensures the energy efficient data transmission between the mobile and cloud devices. Lyapunov optimization minimizes the energy consumption of mobile devices during application transmission through the wireless network in a mobile cloud environment. The optimal and fair service
allocation in MCC developed by Rahimi et al., [111] enable tiered mobile cloud approach for ensuring QoS requirements and improves scalability and performance of the mobile devices. Here, the allocation depends on the Location-Time Workflow (LTW) with the consideration of power consumption, delay, and cost of mobile application execution.

2.5.1 Energy-aware execution approaches

Energy-aware Dynamic Task Scheduling (EDTS) algorithm proposed by Y, Chen M et al., [112] utilizes the Dynamic Voltage Scaling (DVS) technique on the runtime environment to mitigate the energy consumption of the smartphones while satisfying the probability constraint and stringent time constraint of the mobile applications. This algorithm works in some kind of mobile devices and needs an improvement by combining more energy-aware execution MCC techniques. Energy-aware Mobile Application Consolidation and Scheduling (E-MACS) algorithm [113] developed by Shakkeera et al., migrates the idle applications to other active cloud servers. It consolidates the application using load balancing techniques to improve the QoS and minimizes the overall energy consumption, cost of application migration, response latency, and maximizes resource utilization. However, this algorithm considers only less intensive applications having independent tasks. But, the dynamic energy-aware mobile cloud environment considers the dependent tasks in mobile applications. A randomized algorithm discussed by Zarei MH et al., [114] determines the optimal solution for QoS-aware task allocation by selecting the appropriate cloud provider for each application execution. However, these techniques are lacking in taking into account the response time and energy as the vital metrics in dynamic task scheduling and resource allocation for streaming applications. An Energy Efficient Computation Offloading Framework (EECOF) developed by Shiraz et al., [115] extends the work in [116] with the provision of more energy efficient architecture. It takes advantage of both the SaaS and IaaS model to configure the intensive tasks on the cloud resources and to adapt the offloading process of mobile applications respectively. It intends to reduce the energy consumption of the mobile device by mitigating the migration cost.
of application binary code and active cloud server state. The Virtualized screen model presented by Lu et al., [117] virtualizes only a part of the screen of the smartphone in the cloud in which collection of data from the mobile screen involves the text contents, display images, keyboard input, touching and pen input on smartphones, and audio and video. Accordingly, the energy-intensive computing is offloaded to the cloud, and light power consumption applications or computing is performed in the local mobile device itself to reduce the interaction delay and the overall power consumption.

2.5.2 Workflow-aware execution approaches

Most of the conventional heuristic algorithms focus on the two most important objectives such as minimizing the overall completion time of an application [118, 119] and minimizing the total energy consumption of the mobile device [120]. Heterogeneous Earliest Finish Time (HEFT) algorithm developed by Topcuoglu H et al., [121] schedules the tasks of an application based on the task precedence relationship on heterogeneous resources with the aim of achieving the minimum application completion time. To minimize the makespan of mobile interactive perceptual applications, an incremental greedy strategy proposed by Ra MR et al., [122] adaptively takes the offloading and parallel execution decisions. Lin X et al., proposed task scheduling approach [123] for energy minimization and schedules the tasks of application along with the consideration of providing the minimal delay scheduling solution and migrates the tasks among the local cores and the cloud using Dynamic Voltage and Frequency Scaling (DVFS) method. Deadline-constrained workflow scheduling algorithm developed by Abrishami S et al., [124] employs the partial critical path of workflow, considering the VM heterogeneity, time interval pricing model, and pay-as-you-go to reduce the execution cost of the tasks. An approach [125] developed by Phuoc Hung P et al., takes into the account of both the thin clients and thick customers with the aim of improving the cloud access and presents a genetic algorithm to schedule the tasks by focusing on the minimization of processing time and the cost. It considers the task execution in the DAG structure. The author Lin
et al., have reviewed Scalable HEFT (SHEFT) scheduling algorithm [126] assists to elastically increase and decrease the number of resources during execution of scientific workflow applications, which provides the high scalability and optimizes the workflow execution time. Multiple QoS constrained scheduling strategy developed by Xu et al., [127] schedules the multiple workflows which are started at different instants, which dynamically schedule the workflow tasks with minimized execution time and cost and substantially increases the success rate. Nowadays, the cloud workflow application plays a crucial role in the field of IoT-based smart environments such as smart government, smart city [128], smart home, smart medical service, and smart tourism. For instance, the integration of Cloud and IoT addresses the shortcomings in the smart healthcare services such as resolving the weak support of medical platform, risks of medical safety, and low-level emergency medical services [129].

2.6 WORKFLOW BASED COMPUTING TECHNIQUES FOR IOT IN MOBILE CLOUD

In a short span, the recent evolution of IoT and the volume of the IoT devices has created radical changes in business fields involving innovative concepts and services. Accordingly, the emerging domains are e-governance, transportation, utilities, and health care. At MCC, it instead of supporting the workflow execution, an IoT application-specific cloud integration framework is essential due to the rapid generation of streaming data and the need of adopting on the cloud infrastructures. The IoT application specific mobile cloud framework has the capability to provide the support for reading data streams from the sensing devices, easily understanding the data analysis logic, and visualizing the event related data streams from the collection of streams. MCC involves several steps in processing the mobile applications on the remote server. It consists of application structuring, application partitioning and computation offloading, remote execution of partitioned tasks including task scheduling, resource allocation, load balancing, and optimization, and transmit the executed results of an application to the corresponding mobile device over the network.
The prior MCC research in each of the steps of MCC processing is discussed in the following:

2.6.1 Application structuring related techniques

Offloading method [38] is an excellent solution to augment the capabilities of the mobile devices by migrating the computation from resource-poor mobile devices to the resource-rich cloud server. To perform the computation offloading, offloading decision making is necessary. Application structuring facilitates the system to make the offloading decision based on the complexity of the application and device characteristics. The offloading decision is made by analyzing several parameters including network bandwidth, server speed, server load, available memory, and the amount of data to be exchanged between a mobile device and a server. The solution of offloading decision is attained by applying, the application partitioning method and analyzing the application behavior as well as the runtime execution environment [130]. If the cloud server does not contain the application data and the data size of an application is large, the local execution is likely to be more favorable rather than the computation offloading. The system may fail in attaining the benefits of computation offloading when the communication bandwidth consumes higher energy and higher turnaround time due to a large number of data transmissions.

2.6.2 Application partitioning and computation offloading techniques

The static offloading method addressed by Huerta-Canepa G and Lee D [131] offloads the application based on the execution history of the applications, which is adapted to the dynamic variation of the mobile device and the cloud environment. The dynamic offloading method analyzed by Nimmagadda et al., [132], and Yang K et al., [133] reduces the computation time in the mobile robot by adaptively migrating the compute-intensive tasks of an application to the remote server. However, the consumption of the communication time is higher when migrating a large size of input data. Hence, it focuses on the computation as well as the communication while selecting the off-loadable tasks of an application. An approach developed by
Wolski et al., [134] estimates both the communication and computation cost before offloading the computation to the remote server, which is accomplished by a Bayesian approach. Energy-aware Mobile Service Overlays (MSOs) considered by Seshasayee B et al., [135] manages the energy consumption of the mobile device by dynamically monitoring the resource and energy characteristics of the mobile device. The adaptive computation offloading method proposed by Xian C, et al., [136] initially executes the application on the mobile device and then, offloads the remaining tasks of an application to the cloud. It offloads the application from the mobile device after completion of the fixed timeout.

In MCC, the earlier works comprehensively illustrate the different frameworks and architectures for effective code offloading [137, 138] and optimal execution of the mobile applications. A framework in [139] addresses the partitioning problem in mobile data stream applications and enhances the performance of the MCC by exploiting the centralized heuristic genetic algorithm. The offloading decision relies on the application characteristics, the efficiency of the cloud server, CPU speed of the mobile device, and network performance. Evidence-based Mobile Cloud Offloading (EMCO) approach developed by Flores. H et al., [140] exploit the fuzzy sets to formulate the mobile code offloading decision by considering the processing potential of the mobile and the cloud. It also performs the dynamic allocation of the VMs based on the cloud infrastructure. However, a lot of earlier attempts in application partitioning and computation offloading method, offering optimal execution in a mobile cloud environment is still at its initial stage.

2.6.3 An efficient remote execution based approaches

In MCC, remote execution of the mobile applications includes several processes such as task scheduling, resource allocation, load balancing, and optimization. The remote execution of the mobile cloud application targets at enhancing the application performance, executing the applications when there is an insufficient resource on a mobile device, and while achieving the energy efficiency of the mobile device. Due to the massive storage of the
sensing information from the IoT devices, IoT systems adopt the remote execution for manipulating the data storage and computing capabilities of the IoT applications. The mobile cloud application model is classified into four different categories such as performance, energy, constraint, and multi-objective-based application model.

CloneCloud developed by B. Chun et al., [141] is a performance based application model, which employs the augmented execution technique to offload the part of a request to the remote server or smartphone clone. The Smartphone clone stores the data of an application, which is beneficial to the mobile user when the data is lost. Weblet model proposed by X.W.Zhang et al., [37] partitions, each elastic mobile application into multiple components regarding Weblet, which is an independent functional unit and employs an elastic application technique. It performs several processing methods, including computation, storing, and communication and focuses on three factors such as energy consumption, data privacy, and application performance.

A μ Cloud application model developed by V. March et al., [142] supports reusability, reconfigurability, and flexibility of the mobile applications by combining the application of heterogeneous components, which is an energy based application model. It executes the application based on the directed graph structure in which the output of the preceding task of an application is given as the input to its consequent task of an application.

Cloudlet [55] is a constraint-based application model proposed by M. Satyanarayanan et al., which is based on the augmented execution technique. It exploits the VM concept in the form of Cloudlet which comprises the cluster of resource-rich computers. It performs the parallel execution when the Cloudlet has multiple execution cores. Here, the offloading method depends on the two approaches, namely VM migration and VM synthesis. Extensible Cloud (eXCloud) developed by R.K. Ma et al., [143] enables VM instance level offloading method and migrates the top stack frames to the cloud by exploiting Stack-On-Demand (SOD) located on top of VM systems. It employs SOD Execution Engine (SODEE) layer which is transparent to the
applications. It offloads the compute-intensive tasks to nearby cloud infrastructure when a smartphone is overloaded, or smartphone has insufficient resources to execute the tasks on the smartphone.

MAUI [54], ThinkAir [144, 145], and Cuckoo [146] mobile application models are to accomplish multiple objectives such as performance and energy efficiency together. These models are more effectual as they support multiple objectives than the singular objective model. MAUI model makes the offloading decision at runtime to conserve the device energy by the optimization engine. It offloads the code of an application based on the energy constraint on the mobile device. It mainly focuses on the mobile application and mobile characteristics, especially the battery level while partitioning the application and making an offloading decision. ThinkAir mobile application model dynamically allocates the resources using an execution controller of VM instances in the cloud. It provides the resources in an on-demand manner, and the execution controller takes the optimal decision according to the execution time, cost, and device energy. Cuckoo model supports local as well as remote execution, which is specially designed for Android platform. It significantly reduces the programmer's effort by exploiting the integration of familiar development tools to the developers, which exploits the partial offloading method. The proposed research contribution uses MAUI, ThinkAir and Cloudlet mobile application models for load balancing, optimization, and workflow management approaches to execute mobile applications and mobile cloud IoT-based real-time applications respectively.

2.6.3.1 Remote execution based approaches for IoT applications

Several conventional research works on cloud-based IoT management schemes [147, 148] investigate the outsourcing, task scheduling, and resource allocation issue to tackle the shortcomings in mobile cloud IoT framework. An effective approach in [149] proposed by Yau SS and Buduru AB exploits a learning technique and Markov decision process to assess the mobile IoT applications dynamically and enhance the efficiency of the IoT device respectively using the cloud computing
technology based intelligent planning method. QoS-based optimization scheme for resource allocation scheme proposed by Li Chunlin et al., [150] which uses hybrid cloud architecture across the local and public cloud to execute resource-constrained mobile applications. The scheme uses a Lagrange method for resource allocation in the local cloud and remote public cloud by satisfying service level agreements. Aura is a localized IoT-based cloud infrastructure developed by Hasan. R et al., [151] which enables the users to create the ad-hoc clouds based on the IoT devices appearing in the nearby physical environment. The rtGovOps is a software-defined IoT cloud system based framework for governance applications developed by Nastic.S et al., [152] that includes two tasks, such as dynamically providing on-demand governance capabilities, and remotely invoking such capabilities in IoT cloud resources through dynamic APIs. Nested Game based offloading method for MCIoT system (NG-MCIoT) proposed by Kim.S et al., [46] employs the Rubinstein game approach to decide the offloading portion of the computation and dynamically allocates the computing resources to the offloaded tasks on the remote server. It facilitates streaming of the big data when numerous IoT devices connect to the gateway. An IoT-oriented data storage framework proposed by Jiang. L et al., [153] in the cloud platform enables the massive, efficient storage of IoT data and also integrates the structured and unstructured data. However, the effective integration and management of various IoT devices during cloud resource allocation is a quite complicated task. Therefore, developing an adaptive system is necessary to manage the dynamic data changes and to ensure seamless application execution in the mobile cloud IoT environment.

2.7 SUMMARY

This chapter reviews the contemporary research works involving in dynamic resource allocation, energy-efficient task scheduling, load balancing, optimization, and workflow management techniques in both the cloud computing and mobile cloud computing environment. Then, it surveys the dynamic load balancing and execution optimization based approaches in the MCC environment in which execution optimization briefly discusses the
existing energy and workflow based execution approaches. Furthermore, it presents the conventional techniques involved in the workflow based IoT mobile applications with the support of cloud environment.

After analyzing all the merits and demerits of the existing approaches, an energy-efficient adaptive optimization framework is required to ensure an energy-efficiency in mobile devices and remote cloud server by satisfying the QoS and SLA-based performance metrics. The proposed load balancing, optimization, and workflow management approaches in mobile cloud framework have been discussed in the consequent chapters.

The next chapter discusses QoS-aware load balancing approach with task scheduling and resource allocation techniques that can be applied to resource-rich mobile applications.