The Journal of Supercomputing (2020) 76:10258–10287
https://doi.org/10.1007/s11227-020-03240-y

EAAM: Energy-aware application management strategy for FPGA-based IoT-Cloud environments

Atanu Majumder[1] · **Sangeet Saha**[2] · **Amlan Chakrabarti**[1]

Published online: 17 March 2020

Abstract

An efficient integration of Internet of Things (IoT) and cloud computing techniques accelerates the evolution of next-generation smart environments (e.g., smart homes, buildings, cities). The advanced modern cloud networking architecture also helps to efficiently host, manage and optimize the IoT services in smart environments. In this paper, we have considered an "IoT-Cloud" environment where servers are composed of Field Programmable Gate Arrays (FPGAs) which are reconfigurable in nature. The energy consumption is considered as a major driving factor for the operational cost of the "IoT-Cloud" platform. We have proposed an "energy-aware application management" strategy for FPGA-based IoT-Cloud environments, which can efficiently handle sensors' data transmission by positioning them into the best possible coordinates and execute the Service Requests requested by the users. We have compared our strategy performances with an existing technique and the results show that our proposed strategy is capable to achieve high resource utilization with low energy consumption over different simulation scenarios.

Keywords FPGAs · Sensors · Servers · SRs scheduling · Energy-efficient allocation

✉ Atanu Majumder
a.majumder007@gmail.com

Sangeet Saha
sangeet.saha@essex.ac.uk

Amlan Chakrabarti
acakcs@caluniv.ac.in

[1] A. K. Choudhury School of Information Technology, University of Calcutta, Kolkata, India

[2] School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK

1 Introduction

The Internet of Things (IoT) represents the network of objects, devices, machines and other physical systems with embedded sensing, computing, and communication capabilities. Such capabilities help the systems to sense and share real-time information with the physical world [3]. By integrating the IoT to the cloud, huge amount of data collected from multiple locations can be processed and analyzed to generate useful information for the end users. Moreover, this "IoT-Cloud" structure alleviates the limitations of lightweight mobile devices (e.g., battery life, processing power, storage capacity) by utilizing the advanced resources of cloud [5]. The advent of this IoT-Cloud model enables a large number of services and applications (e.g., health system monitoring, traffic control, energy management, vehicular networking), which will play their inevitable role in the making of next-generation smart environments (e.g., smart homes, smart buildings, smart cities, smart grids) [26].

In traditional centralized cloud architecture, computing and storage resources remain concentrated in a few large data centers. However, as the number of IoT services and connected devices increases, the traditional centralized cloud architectures lead to excessive network load, end-to-end service latencies, and unbearable energy demands [6]. In order to maintain the stringent deadline requirement associated with real-time IoT applications (to maximize the overall efficiency), cloud architectures now became distributed [10]. In such distributed architecture, the cloud infrastructure can be modeled into hierarchical layers where in each layer, the devices have their different storage and computing capability. Hence, in contrast to the traditional centralized cloud architecture, IoT-Cloud networks provide greater flexibility for allocation of resources to serve IoT services and thus render the capability of meeting their stringent latency and mobility constraints.

Generally, the functionalities of "IoT-Cloud" platforms can be divided into two phases. In the first phase, the mobile sensors sense the data and send them to the cloud for further processing. The Second phase is mainly responsible for providing the services to the end users. The user sends the service request to the cloud in order to access the stored information. However, still today, battery life is one of the prime factors affecting the usability of mobile sensors and the data transmission to the cloud. Hence, there is a dire need of the strategies which will execute both the phases in a faster way (such that the service requests can be finished within the deadline) while maintaining the energy efficiency.

Servers of a data center execute the arrived SRs and store the information from sensors. Nowadays, each server contains one or more FPGA-based processing elements (PEs) for faster application execution [11]. A particular SR is physically mapped into a processing element (FPGA) via virtual machine (VM). More specifically, we can argue that from the user end, SR is mapped in a VM and then it executes. VMs are responsible for serving the SRs on a physical server by providing an abstraction to the users. SRs need to be scheduled on the cloud infrastructure while maximizing the resource utilization.

The main motivation of this work remains to collect data from sensors in an energy-efficient way. Hence, an energy-efficient service request management

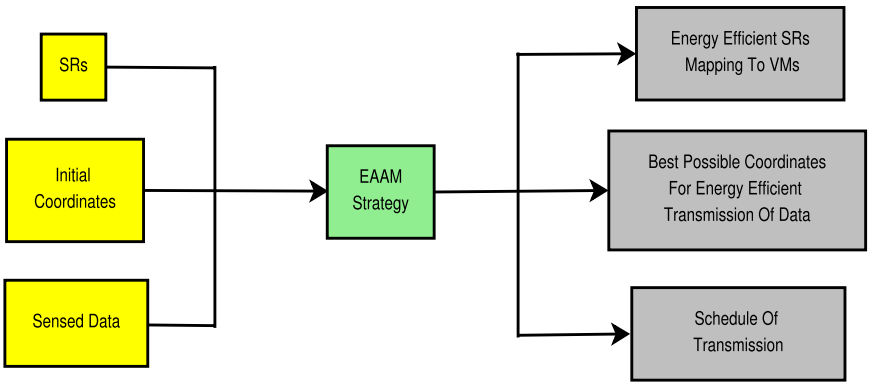


Fig. 1 Block diagram

scheme is required which can satisfy two important concerns, i.e., (1) real-time IoT service management, (2) minimize the overall energy consumption. In this paper, we have coined the strategy named "*EAAM (energy-aware application management)*". This strategy is pictorially shown in Fig. 1.

The main contributions of this work can be summarized as follows:

- We have modeled a hardware-based system architecture to incorporate the sensors or actuators to transfer data to the base station and stored the information in cloud. Proposed architecture integrates the concept of IoT-Cloud, where IoT services are computed and executed by the cloud resources.
- We have proposed a strategy namely *EAAM (energy-aware application management)* which tracks the sensors (wireless) position and organizes the sensors for effective data transmission to cloud servers. EAAM executes users-requested services in such a way so that each SR can meet their deadline requirements.
- This work incorporates the FPGA as a hardware accelerator for SR execution. FPGA minimizes the total execution time of the real-time SRs, ensures high task throughput, less task rejection which enhances system performance.
- An existing scheduling technique known as first fit decreasing (FFD) has been included to compare the performances analysis with EAAM. The comparison with the existing work reveals that the proposed strategy is able to achieve better results in terms of power optimization by 13.8% and 15.7% in terms of overall execution time than the existing technique.

The rest of the paper is organized as follows: Related work is discussed in Sect. 2. FPGA-based IoT-Cloud infrastructure is discussed in Sect. 3. System model and assumptions are briefed in Sect. 4. Problem formulation is stated in Sect. 5. Working principle of EAAM with an illustrative example is discussed in Sect. 6. Performance of the proposed strategy with an existing technique has been described in Sect. 7. Experimental setup and the results are illustrated in Sect. 8. Finally, the paper concludes with future scope in Sect. 9.

2 Related work

Combining IoT with cloud constitutes the IoT-Cloud, which has several mutual advantages that have been derived from their integration, especially in the domain of wireless sensors [1, 21]. As there is a rapid growth of cloud data centers power consumption, in order to reduce the power consumption and promoting the green cloud computing authors in [20], an adaptive task allocation algorithm was proposed to minimize the makespan for the heterogeneous cloud environment and reduce the power consumption. In [19], authors described the architecture for the future IoT network considering the cases of its security challenges. An efficient algorithm proposed for IoT network known as Media-based Surveillance System (EAMSuS) which is actually a combination of two existing algorithms for data routing and security. Research on scheduling algorithm of variant applications on geographically distributed data centers of cloud architecture has drawn considerable interest from the past few years. Authors in [24] present a provably efficient online algorithm for scheduling where tasks are assigned in a batch among the multiple geographically distributed cloud data centers. The proposed algorithm achieved low energy cost by considering the queuing delay. In [30], authors proposed a temperature-aware methodology to manage the workload of the geo-distributed data centers and formulated the problem as a joint optimization for request routing and capacity allocation. Authors in [32] described a strategy named "Fuxi", which makes proper management of the resources in a distributed scenario and schedule jobs to the systems. Authors have also shown three novel techniques which allow Fuxi to tackle the scalability and fault tolerance issues at Internet scale. In [18], authors have presented an approach known as "EnaCloud" for dynamic allocation of the resources which ensures energy efficiency by minimizing the count of active servers. Most of these recent works depicts the advantages of integration in between the cloud and IoT network. In those papers, having a distributed scenario how efficiently applications can be allocated (algorithm) to the cloud resources so that the execution time and power consumption should be minimized have been discussed. But the disadvantage is mass of the existing research works have tackled this allocation problem for the second phase (refer to Introduction section).

In the recent past, researchers have also been engaged to explore this domain of "FPGA-based task or SR scheduling approach". Authors discussed a virtualized framework in [7] for efficient resource management and communication where (FPGA-based) reconfigurable accelerators have been integrated to the servers. FPGA is an ideal choice for integration as an accelerator in the cloud, as it can be reprogrammed as per needs. Moreover, it allows multiple accelerators to share optimized communication infrastructure. In [15], authors have shown that embedded systems and cloud computing infrastructure with dynamic reconfiguration will bring "neuroplasticity" to the interconnected space of IoT embedded systems, enabling high-performance, customized, secure-by-design cloud computing services. In the next section, we will show the benefits of FPGAs utilization on "IoT-Cloud" architecture. FPGAs are nowadays used as accelerators in servers of

cloud data centers. Normally, VM instance resides in a FPGA-accelerated server which could lead to poor resource sharing. To overcome this situation, authors depict an idea of FPGA pooling in [33], which managed all FPGA-accelerated servers as a single resource pool and shared among all VMs. Authors proposed a framework in [8], which supports self-organizing and self-management local strategies for heterogeneous cloud infrastructure. These dynamic resource allocation strategies help to reduce the energy consumption, maximized throughput and computational efficiency. In [14], authors review the Smart surveillance system mainly related to automated audio-video analysis techniques. This review mostly considers the three main aspects of surveillance IoT system is architecture, network and advanced algorithms. A framework was proposed by the authors in [28] for payload distribution of tasks in order to run the tasks efficiently. Fragmentation of payloads and distributing them over multiple clusters in a cloud-like environment is a challenging task. Heterogeneous chip which is a combination of CPUs and FPGAs gives a choice to select most suitable processing platform for a task execution. Authors showed that the existing software-defined framework performance is enhanced by two schedulers: Dynamic and LogFit in [22]. These schedulers help to distribute the tasks to the resources in an optimal manner. Advantages of FPGA as a processing element have been described in the above works. Reconfigurable systems like FPGAs combine the benefits of flexibility of general purpose processors and the performance efficiency of a dedicated hardware. Acceleration in task processing is the major issue in cloud and fog which can be resolved by choosing FPGAs as an accelerating engine. However, "IoT-Cloud" infrastructure with FPGA as processing platform has not been explored too much for low energy computation.

From the above discussion, it is evident that management of the cloud application in an energy-efficient way has drawn a considerable interest. However, majority of the existing research works have tackled this problem in the second phase (refer to Introduction section). Thus, in the work we have tracks the sensor's (wireless) position and organizes the sensors for effective data transmission to cloud servers. Energy-aware strategies have been implemented on IoT-Cloud previously, but most are on software-based platforms. FPGA works as an accelerator engine with low power computation which gave it as a preferable choice for an energy-efficient processing platform. In most of the cases, data from moving sensors are transferred in broadcast way which requires more energy and re-transmission added extra energy burden on it. We implemented the design where data are transferred in unicast way only when mobile sensors are positioned in their best coordinates which ensure distances between sensors and base stations are minimum. Proposed EAAM allocates the resources in the best possible way so that re-transmission occurrence is minimum. Hence, the energy consumption is optimized.

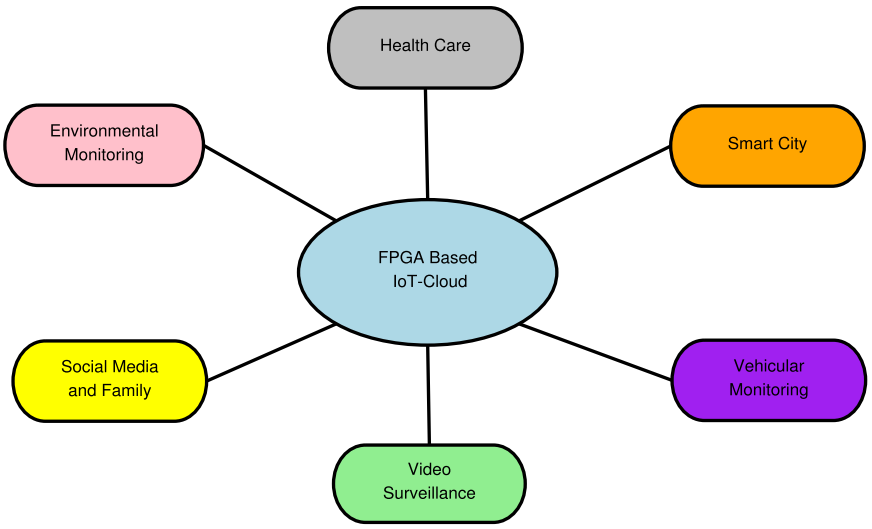


Fig. 2 IoT devices integration with the cloud network

3 FPGA-based IoT-Cloud infrastructure

IoT-Cloud infrastructure is a combination of geographically distributed cloud networks and IoT. Processing resources of the back-end servers are composed of FPGAs. The main prospect of such infrastructure is to utilize the advantage of pervasive sensing capabilities of IoT and computing resources of cloud networks. With this new paradigm of virtualization and distributed network technology, we can dynamically handle the real-time service functionality by allocating them to the proper resources. IoT-Cloud-based sensor networks are not only used to gather data from health care, environment monitoring, vehicle monitoring, video surveillance, social media but it also utilized the cloud infrastructure for storing and computation of data. The various applications of FPGA-based IoT-Cloud platform are shown in Fig. 2.

As an application scenario, we can deploy sensor kits in vehicles (like buses, trams, trains etc) for tracking air pollution, temperature, humidity, etc., in cities. Each of these sensor kits consists of multiple sensors and a communication device equipped with both WiFi and 4G capabilities. Sensors can collect data from environment while moving. When the vehicles enter a certain stoppage point where the base station has been deployed, they start to transmit data. Hence, we can push sensor data to the cloud when vehicles reach nearest to their predefined stoppages. The communication devices fitted in the vehicles will connect to the stoppage point WiFi and push the data collected since the last stoppage. Users or data consumers can track or collect information through a service request.

Main advantages of FPGA-based IoT-Cloud infrastructures can be summarized as follows:

Flexibility IoT-Cloud infrastructure offers the virtualization technique which allows to utilize and share the homogeneous or heterogeneous resource pool among the

multiple service requests. Without any dedicated deployment, it gives access to the virtually unlimited resources.

Scalability Cloud architectures typically give access to a rich pool of resources, where storage and computing resources are enormous and services are provided without any network saturation.

Mobility Considering the wireless sensor network (WSN) the current location of the mobile devices and users are provided by the contextual services on demand.

Operational cost Service functions SRs can be generated from any geographical location. Some require huge computations; some require less. Those services which require large inputs or produce large outputs can be computed through their closest respective sources or destinations which will help to reduce the network load as well as the associated operational cost.

At the back end of IoT-Cloud architecture, it employs virtualization technique to utilize the FPGA resource pools. Hence, this architecture supports the access of virtually unlimited resources without any network saturation. Cloud data centers are located in distributed geographical location, but the allocation strategy helps to reduce the operational cost of services by computing them to their nearest data centers. Smart computing by the edge devices like FPGAs in the modern IoT-Cloud architecture reduces the burden of both the data transmission and computations of the cloud computing devices. Hence, service latency is decreased to a great extent. The proposed working functionality for IoT-Cloud infrastructure is described in the next section.

4 System model and assumptions

The adopted model of this paper is a general "IoT-Cloud" architecture. Our proposed system model is shown in Fig. 3. Model consists of different functional components which are discussed in the next subsection.

4.1 Component

Individual components required to build a system which ensures the desired service are elaborated in detail.

Sensors Wireless sensors are considered, and each sensor monitors as well as records the physical data from the source. Then, sensors send the accumulated data to the static base station's (like bus stop, rail station, house etc) DAC.

DAC The Data Analytical Center (DAC), which acts as a base station (data receiving center), consists of Energy-Efficient Resource Allocator (EERA), a Data Analyzer (DA), and an Application-Dependent Data Converter (ADDC). DA and ADDC

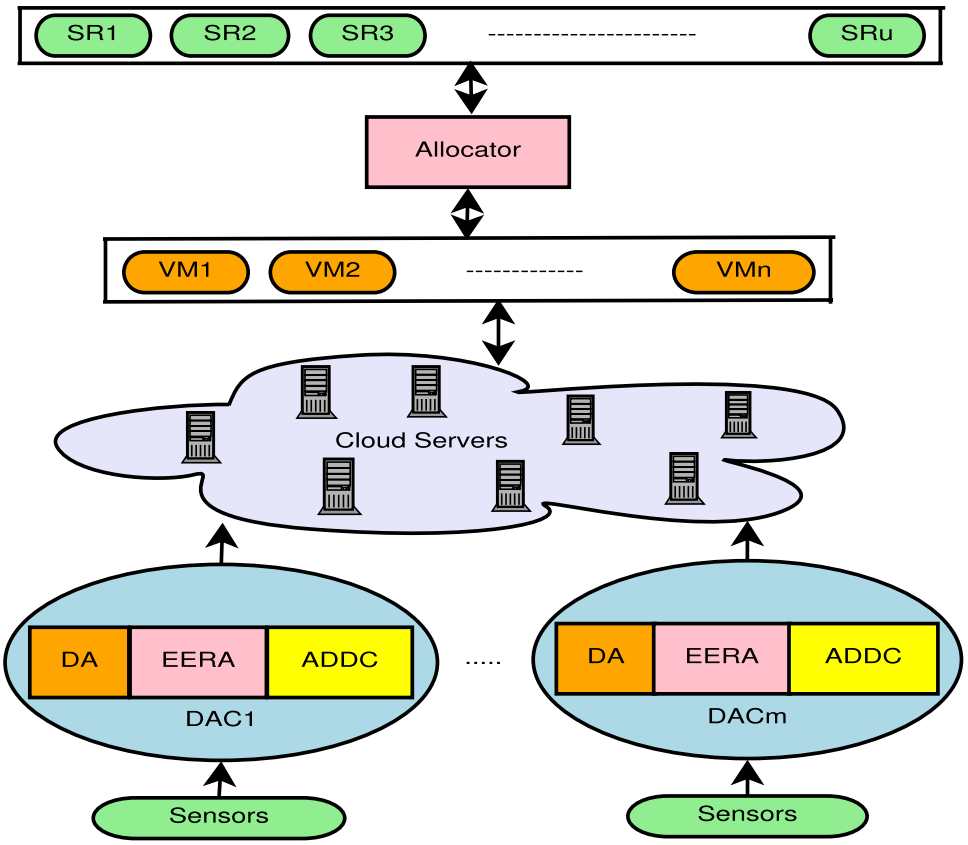


Fig. 3 System model

collect raw data in large volumes which are stored, processed and analyzed to extract meaningful information from it. On the arrival of data, Energy-Efficient Resource Allocator (EERA) will attempt to allocate the data to the appropriate cloud servers, depending upon the energy and performance characteristic.

Servers Each server consists of η number of reconfigurable processing elements (FPGAs). Servers can be located in geographically distributed areas. They are responsible to store the information from DAC and execute efficiently the real-time or non-real-time SRs which are requested by the end users.

SRs Service requests can be initiated by the users. SRs can be periodic or non-periodic in nature. Periodic SRs will appear at a certain time interval, whereas the non-periodic SRs may appear at an arbitrary instant of time. Each SR has some parameters such as e_i, worst case execution time of the task and p_i, the relative deadline of the task.

Allocator Requested SRs need to be satisfactorily resolved by allocating them to proper PEs. Each SR has different criteria hence to satisfy all the requirements or deadline a proper allocation strategy required. Allocator will allocate the SRs to proper PEs so that the overall power consumption can be minimized.

4.2 Functionality of the model

Functionality of our proposed system model can be classified into two subsets: (1) data accumulation from sensors and processed to cloud storage and (2) storage information retrieved from cloud by the users through service requests. Sensors sense data form environment and send them to the base station identified as DAC. DAC is being executed on an FPGA platform which actually utilizes the cloud virtualized storage space through a virtualization technique to store the information from the sensed data. On the other hand, users can initiate SRs to retrieve the data from cloud. It is the responsibility of the allocator to manage the requested SRs satisfactorily. The allocator allocates SRs to appropriate PEs using virtualization technique to reduce energy consumption of the servers. As both the DAC and allocator have to work in an energy-efficient way, we need to develop an energy-efficient scheduling algorithm for DAC as well as for the allocator which are presented in the next subsequent sections.

4.3 Assumptions

However, without loss of generality, we have made the following assumptions:

- Data centers are distributed, i.e., existence of multiple servers in different geographical locations. They could send data like environment conditions, such as temperature, humidity, pressure, wind speed, and pollution level..
- Each sensor has some certain coverage area; If distance between the DAC and sensors is in that coverage range, then only data will be transmitted.
- SR cannot simultaneously execute on two distinct PEs at the same instant of time and will be executed on a single PE until its execution requirements finish.
- All the SRs are independent hence, each PEs can operate in parallel to execute individual service request.

5 Sensors working strategy and problem formulation

Let us assume that a bunch of sensors attempt to send data to a DAC. In the background, each DAC is connected to a server pool in the cloud computing architecture. Mobile sensors cover a certain area for communication, and that area comes under the jurisdiction of a DAC. Sensors send data to DAC through a unicast fashion as broadcast may require more energy and the energy source of a sensor is limited. Sensors are classified and divided into multiple logical groups based on the communication distance between DAC and sensor's current position.

Each logical group is assigned a priority number to maintain the order of execution. Top priority group, which has the minimum average distance between DAC and sensors, is at the center, and others are positioned at right and left of a DAC. Sensors having the lowest individual distance from DAC are selected first for transmitting

data. Longer distance means it will take more time to send data, and hence, energy consumption will be high. Based on the distance from DAC, each sensor has been assigned a penalty value. The longer the distance, the more be the penalty value. We have chosen a threshold, and sensors with less or equal to the threshold penalty value will be able to transmit data.

At a particular time instance, DAC can process or accept a certain amount of data from the sensors. In case accumulated data (accumulated in ascending priority order) exceed the bandwidth of the DAC, then the data transmission will fail for the sensors which have the least priority. If any sensor fails to transmit the data, a re-transmission will be attempted after re-arranging their positions.

Sensors transmission types are periodic in nature which will occur at a certain time interval when it comes under a DAC. At the start of transmission period (TP) τ, data will transmit from the sensors to DAC. Within a TP, sensors will be allowed to send data for a specific time duration and we call such time duration as multiple Bandwidth Allocation Time-gap (BATG), which represents a time quanta for data communication. If sensors fail to transmit, then they will take participate in the next BATG under the τ for transmission. Each sensor's transmission time cannot be more than the BATG; otherwise, transmission will fail due to the exceeded communication bandwidth limit. Our focus will remain to reduce the BATG size,[1] as energy consumption will increase with the BATG size.

Let us assume $\mathcal{P}(P^t_1, P^t_2, P^t_3, ..., P^t_N)$ be the positions of N number of sensors at a particular time instance t and there is M number of DACs present in a distributed cloud environment. DACs are connected with η number of servers. At the end of a BATG, sensors re-position them and the new position becomes: $\mathcal{P}(P^{t'}_1, P^{t'}_2, P^{t'}_3, ..., P^{t'}_N)$. The new position value will be:

$$P^{t'}_i = P^t_i + rand(0,1).[V_s] \times (DAC - P^t_i) \times IP \tag{1}$$

where $[V_s]$ is a unit vector, which determines the displacement of sensor from the starting point, IP is the improvement percentage in the position of a sensor, and $(DAC - P^t_i)$ represents the Euclidean distance between the current position of the sensor and DAC.

If the data transmission for all the sensors is not successful within a BATG, then it demands re-positioning of the unsuccessful sensors. Chances of successful transmission depend on the re-positioning and grouping. Center positioning group has a much higher success rate than others as it has a less average distance from DAC, then right-wing and at last left-wing sensors. Let us assume N number of sensors participating in each BATG that are grouped into G_c, G_r, G_l with N_c, N_r, N_l numbers of sensors respectively. G_c, G_r, G_l represents the group at center, right and left, respectively, and N_c, N_r, N_l represents the number of sensors present in each group, respectively. Grouping of sensors is shown in Fig. 4.

Successful transmission of data to DAC by sensors relies on their positions inside a group. High number of individual success (S_i) indicates that the sensors have converged

[1] This can be viewed as iteration number as in each BATG (iteration) sensors attempt to send data.

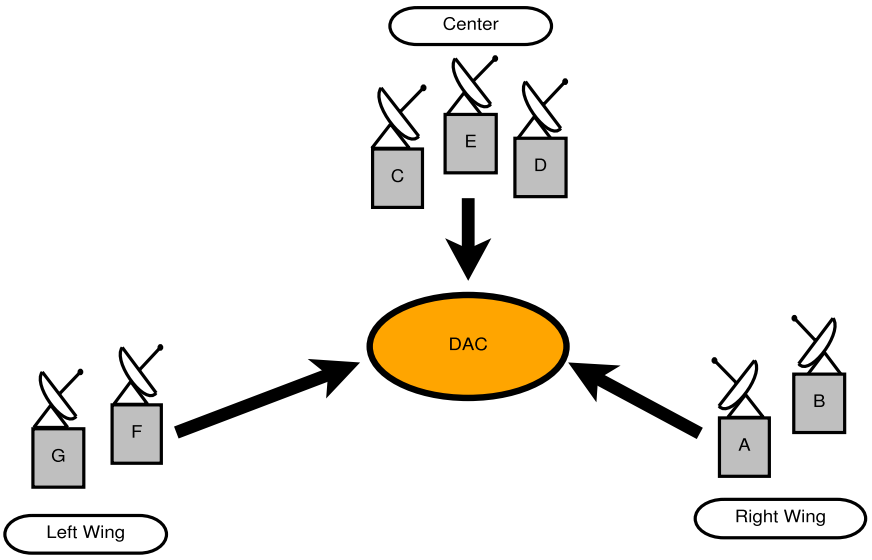


Fig. 4 Sensors grouping model

to a point that has the optimum distance. So this factor can be used as a useful element for the size of BATG. Using the success values, total BATG size $SBATG_k$ for the kth for TP is computed as:

$$SBATG_k \geq \frac{N}{\sum_{i=1}^{N} S_i} \tag{2}$$

Our objective is to increase the number of successful transmission of sensors per BATG specifically, to reduce the size or number of iterations of BATG ($SBATG$) at each transmission period. Size of $SBATG_k$ decreases which indicates the requirements of less number of re-transmission of sensors which eventually helps to reduce the burden of heavy energy consumption.

$$Minimize \sum_{k=1}^{SBATG} \sum_{i=1}^{N} S_i \times E_i \tag{3}$$

where E_i represents the total energy consumption for sensing, transmitting and processing data by a sensor.

6 EAAM strategy

As discussed, the EAAM strategy works in two phases. The first phase deals with the energy minimization for sensors, and the other phase focuses on the energy minimization for SRs. In the next section, we will discuss the first phase.

6.1 Energy optimization strategy for sensors

To optimize the energy consumption for sensor data computation, we need to ensure that the sensors should be positioned in their best possible coordinates so that the data communication distance between source and destination is minimized. In order to optimize energy, we should look into both the cases, i.e., energy optimization on the server side and energy optimization for sensor devices. We have formulated the entire optimization strategy in a mathematical way and the algorithm as follows:

6.1.1 Penalty threshold

Sensors are grouped randomly according to the distance between DAC and sensors. For data transmission, a fixed priority is assigned to each group by providing a priority index (*PI*) value. Sensors within a group transmit data according to the minimum shortest distance between DAC and sensors. A queue is maintained for sequential execution of sensors which are divided into groups as per their assigned priority. Each sensor is positioned at different distance levels from the base station DAC. Hence, time and energy consumption will be different for transmitting data by each of the sensors. Based on the distance, each sensor is assigned a penalty value. Higher priority means sensors with less Euclidean distance value ($DAC - P_i^t$) and thus, penalty value decreases and vice versa. For each sensor, penalty value can be measured as:

$$PV_i = DD_{ijk} \times PI_{G_x} \qquad \forall x \in (c|r|l) \tag{4}$$

where DD_{ijk} represents the distance (meters) between the ith sensor to the jth DAC at kth BATG and PI_{G_x} is the assigned priority index for group G_c, G_r, G_l, respectively. Penalty value for individual sensors should be less or equal to the threshold value; otherwise, the sensors will be discarded from the transmitting queue. Penalty value for each sensor group is estimated as:

$$PV_{G_y} = \frac{\sum^{N_y} DD_{ijk}}{N_y} \times PI_{G_x} \qquad \forall x, \forall y \in (c|r|l) \tag{5}$$

where $\sum^{N_y} DD_{ijk}$ represents the cumulative distances of all sensors from DAC and belongs to a particular group.

Hence, the penalty threshold value for the sensors is defined as:

$$PTV = Max(PV_{G_x}) \quad \forall x \in (c|r|l) \tag{6}$$

Sensor with the shortest distance within a group gets the highest priority for transmission of data. Hence, the success of the ith sensor at any time instant within a BATG is denoted by:

$$S_i = \left\{ \begin{array}{ll} 1, & \text{Penalty Value } \leq \text{ Threshold} \\ 0, & \text{else} \end{array} \right\} \tag{7}$$

6.1.2 Data flow measurement

Sensors sense the data from the source and after accumulating the sensing data they transmit them to the DAC. Data sensed by a sensor at a particular time instance are defined as:

$$TDS_{P_i^t} = DA_{P_i^t} \times T_i \quad \forall P_i^t \in \mathcal{P} \tag{8}$$

where $DA_{P_i^t}$ represents the unit amount of data (bits/sec) sensed by the ith sensor at position P_i^t and T_i represents the sensing time in terms of seconds.

We assume that after penalty threshold verification a few number of sensors may be discarded from sending data to DAC. Hence, the total sensed data by N' number of successful sensors are represented as:

$$\sum_{i=1}^{N'} TDS_{P_i^t} \tag{9}$$

where N' represents the number of valid sensors for transmitting data after checking of the penalty threshold.

Sensed data of a sensor is accumulated and transmitted to the DAC. Data transmitted by a sensor is measured as:

$$TDT_{P_i^t} = DT_{P_i^t} \times TP_i \quad \forall P_i^t \in \mathcal{P} \tag{10}$$

where $DT_{P_i^t}$ represents the unit amount of data (bits/sec) transmitted by the ith sensor at position P_i^t and TP_i defines the transmission period (seconds) of ith sensors.

Arrived raw data are then processed by DAC, and the extracted information from the raw data is stored in the server for future references. Total data processed by the jth DAC at each BATG are defined by:

$$TDP_j = \sum_{i=1}^{N'} (TDT_{P_i^t} - DPL_{P_i^t}) \times PF_j \tag{11}$$

where PF_j is the processing factor applied to the received data from sensors and necessary information is generated. Here, we assume that few data packets are lost by each sensor ($DPL_{P_i^t}$) at the transmission time.

6.1.3 Capacity limits

Each sensor has a capacity limit (in volume) of sensing and transmitting of data. Sensing capacity of the ith sensor at position P_i^t for a time instance is defined by:

$$SC_{P_i^t} \geq TDS_{P_i^t} \quad \forall P_i^t \in \mathcal{P} \tag{12}$$

The transmission capacity of the ith sensor is represented as:

$$TC_{P_i^t} \geq TDT_{P_i^t} \quad \forall P_i^t \in \mathcal{P} \tag{13}$$

The processing capacity of a DAC also has a limitation which is shown in the following equation:

$$PC_j \geq TDP_j \quad \forall P_i^t \in \mathcal{P}, \forall j \in M \tag{14}$$

6.1.4 Energy computation(EC)

The energy required for sensing and transmitting data by a sensor is generated from limited battery service. Utilized energy for sensing and transmitting data is measured by the equation given as:

$$SPW_{P_i^t} = TDS_{P_i^t} \times UCDA_{P_i^t} \tag{15}$$

$$TPW_{P_i^t} = TDT_{P_i^t} \times UCDT_{P_i^t} \tag{16}$$

where $UCDT_{P_i^t}$ represents the unit cost for data transfer (watts/bps) and $UCDA_{P_i^t}$ is the unit cost for data acquisition (watts/bps) of ith sensor at position P_i^t.

DAC receives data from the sensors. Depending on the various data types, ADDC converts it in a meaningful information and stored in servers by EERA in an energy-efficient way. The energy required to perform this operation by the jth DAC is estimated as:

$$PPW_j = \frac{TDP_j \times UCDP_j}{DP_j} + ACC_{jz} \tag{17}$$

where $UCDP_j$ represents the unit cost for data processing (watts/bps) and DP_j is data processing rate (bits/sec) of jth DAC, respectively. Average communication cost between jth DAC and zth server is represented by ACC_{jz}.

A sensor which has successfully completed its data transmission to the cloud requires a certain amount energy which has been represented as:

$$E_i = SPW_{P_i^t} + TPW_{P_i^t} + PPW_j + RC_i \quad \forall i \in N, \forall j \in M \tag{18}$$

where RC_i is the re-positioning cost (power) for the ith sensor.

6.1.5 Techniques

To describe the energy optimization strategy for sensors, we have explained it into two parts. In this model, sensors are movable but DACs are static. Sensors communicate with the DAC by using the WiFi communication protocol. WiFi has some communication range; when the distance between DAC and sensors is in that range, only the communication is established. At the start of a transmission period data transmitted from sensors to a DAC.

In the first part of our technique, we have shown how to measure the size of BATG ($SBATG_k$) for each transmission period. Clustering of sensors is carried out first when the sensors are under a DAC communication range. Based on the sensors position (distance between DAC and sensors), they are clustered into three groups central, right and left. Central group has minimum average distance, and then right and the left ones have the maximum average distance. A penalty value for each sensor group is measured based on the average distance between DAC and sensors of that group. Maximum of all group penalty values is set as the threshold value for communication. Once the communication is established, sensors are allowed to transmit data to the base station. The count of sensors that allowed to transmit data successfully has been tracked which is also known as success of sensors. If all the sensors are able to transmit data successfully in that BATG, then re-transmission is not required. But in case of any failure, sensors re-positioning is required and again same procedure will be followed from starting. Calculate the number of times sensors re-positioning is required (size of BATG) for success transmission of all sensors data with in a τ and then call the energy computation (EC) function for energy calculation.

Algorithm 1: EAAM Scheduling Strategy for Sensors

Input: Transmission period τ, N number of sensors, M number of DACs, η number of servers, Position Parameters (P,P')
Output: Size of Iteration ($SBATG_k$) for each transmission period
Initialize success = 0;
At a particular time instance t_i, all the sensors which are positioned at position $\mathcal{P}$ acquired data from the source;
for *(each transmission of BATG)* **do**
    Based on the position of sensors from DAC, sensors are divided into three logical groups (G_r,G_c,G_l) where G_c has the minimum average distance from DAC;
    Calculate the amount of data sensed by sensors using the equation 8;
    Calculate the Penalty Value and Threshold for each sensor using equation 4 and 6;
    if *(Penalty Value* $\leq$ *Threshold)* **then**
        Calculate the amount of data transmitted by the valid sensors using equation 10;
        Data processed and stored in the server by DAC which is measured by equation 11;
        Calculate the sum of individual success of sensor by using equation 7;
        Call the function EC();
if *(Sum of success* $<$ *N)* **then**
    Re-arrange the positions of the failure sensors by equation 1;
    Repeat the steps from starting for next BATG;
else
    Calculate the Iteration size $SBATG_k$ by using equation 2;
    Wait for the next transmission period of τ ;

Second part shows the estimated required energy for sensing, transmitting and processing of data for each iteration. Finally, it produces the mapping of DAC to server by using the virtualization technique.

Algorithm 2: Function EC()

Input: Data unit $DA_{P_i^t}$ & $DT_{P_i^t}$ & DP_i for sensed, transmit and processed, Unit power ($UCDT_{P_i^t}$ & $UCDA_{P_i^t}$ & $UCDP_i$) required by each sensor and DAC, power requirement for a single server PW_{S_z}, D_A and DC cost, Servers Max power PW_{MAX}, η number of servers, iteration size $SBATG_k$

Output: DAC to server data mapping: SDM,

if *($\eta > 0$)* **then**

Calculate the energy consumption for data acquisition $SPW_{P_i^t}$, data transfer $TPW_{P_i^t}$ and data processing $PPW_{P_i^t}$ using equation 15, 16 and 17 respectively;

Based on the estimated energy for successful sensors DAC uploads data to server in cloud;

return DAC to server mapping (SDM) of all sensors with required power for a TP;

6.2 Energy optimization strategy for SRs

On the other hand, U number of users can initiate SRs which need to solve satisfactorily by the servers. Execution of SRs on servers also requires energy. Hence, server consumes a huge amount of energy resources to perform resolving operation the SRs.

6.2.1 Energy computation (EC)

Energy estimated for a server is shown below:

$$PW_{S_z} = \sum_{i=1}^{U'} (IW_{SR_i} \times UC_{iz}) + DC \quad \forall z \in \eta, \forall i, U' \in U \tag{19}$$

where IW_{SR_i} and UC_{iz} represent the individual weight of the ith SR and unit cost to serve ith SR on zth server, respectively. Delay cost for communication is estimated as DC.

Individual weight of an SR is represented as:

$$IW_{SR_i} = \frac{SET_i}{DL_i} \tag{20}$$

where SET_i and DL_i denote the service execution time and the deadline for ith SR executions, respectively.

Each server has an upper limit of energy consumption. Overload may cause some serious physical damage. Hence, the upper bound of power for all servers is defined as:

$$PW_{MAX} \geq \sum_{z=1}^{\eta} PW_{S_z} \tag{21}$$

We have estimated the power requirement for performing our EAAM strategy. Though power and energy is not the same thing, there is close relation between then which is shown as:

$$Watts = \frac{joules}{time} \tag{22}$$

If the time is constant, then power and energy have a proportionate relation between them, which means if power increases, energy also increases.

6.2.2 Techniques

The last part of our algorithm explores the scheduling strategy for SRs in cloud. For each SR, all the parameters like execution time, deadline, etc., are known beforehand. At first, the individual weight of SRs is calculated and the energy required to perform the execution is estimated. Literally, the cloud has unlimited resource pool, but each server has a certain limitation of power consumption. If the energy required by all SRs is less than the maximum amount of energy required by all servers, then the SRs are guaranteed to serve; otherwise, task rejection will occur.

Algorithm 3: EAAM Scheduling Strategy for SRs

Input: U number of SRs, η number of servers, UC Unit power cost, Temporal parameters SET and DL, DC Delay cost, PW_{MAX} max server power consumption
Output: SR to PE mapping
for *(each SR of U)* **do**
  Calculate the individual weight each of the SRs using the equation 20;
  Calculate the amount of energy required by each of the SRs for execution using the equation 17;
  if *($PW_{MAX} \geq$ Execution power of all SRs by servers)* **then**
    Re-arrange the SRs according to the shortest deadline;
    Map the SRs to VMs for executions;
    Return the total energy consumption by the SRs;
  else
    Discard the SRs;

6.3 An example depicting the EAAM strategy

Let us assume that A, B, C, D, E, F, G are the mobile sensors which are sensing data from the real-world and transmitting the data to a DAC. Each sensor covers a certain distance or area which comes under a DAC through WiFi communication. Based on the distance between DAC and sensor, sensors are clustered into multiple (three) groups. One group of sensors, say G_c(C, D, E), is positioned at the center that has the average minimum distance from DAC. Other groups are positioned at right G_r (A, B) and left G_l(F, G). Left-positioning group has the average maximum distance

Table 1 Sensor parameters

N	*UCDA*	*T*	*UCDT*	*TP*	*DD*
A	0.003	0.3	0.009	0.07	10
B	0.005	0.1	0.008	0.09	12
C	0.005	0.1	0.008	0.09	4
D	0.002	0.5	0.007	0.10	5
E	0.003	0.4	0.006	0.05	7
F	0.002	0.5	0.007	0.10	15
G	0.004	0.2	0.010	0.04	17

from the DAC. As sensors are movable, position of sensors is frequently changing. Hence, grouping needs to be done for each transmission time.

6.3.1 Energy optimization strategy for sensors

Let us also assume that at the back end of cloud infrastructure is connected with 2 servers having 4 FPGA processing elements (PEs) each. As sensors are homogeneous in nature, each has the same sensing and transmitting speed. Sensor parameters are shown in Table 1 (elaborately discussed in Sect. 8).

Data acquisition rate (*DA*) and transmission rate (*DT*) of wireless sensors are constant, let say 512 bits/second and 1024 bits/second [4], but the sensing or transmission time is different. Unit cost for data sensing and transmission varies with the type of data it senses or transmits. We assume that a server has a maximum power capacity of $PW_{S_z} = 150$ watt[2]. Hence, PW_{MAX} will not be greater or equal to $150 \times 2 = 300$. Assuming that each sensor is equipped with a battery storing 27 kJ [25], this converts into a maximum power of $BT_{PW} = 856$ µW. Delay cost for communication is assumed to be $DC = 0.02$ (watt) and average communication cost ACC_{jz}=1 (watt), which depends on the distance length between the server and DAC and re-positioning cost is $RC_i = 0.1$ (watt). It is expected that few data (5% of total transmitted data by each sensor) packets will be lost at the transmission time. Data processing factor PF_j implied by DAC is a process that eliminates the redundant or unnecessary date before it is stored in the server. Implied data processing factors ensure that at an average of 20% of arrival data to DAC are discarded. Let us assume that the data processing rate (*DP*) of DAC is 1024 bits/second and unit cost for data processing *UCDP* is 0.2 watt. Power consumption for sensing, transmitting and processing is measured by Eqs. 15, 16 and 17, respectively.

Based on the threshold value, we will make a queue of sensors which are eligible to transmit data. Center position group has the highest priority as it has minimum average Euclidean distance between DAC and sensors and that has been indexed with 0.1, then right-wing group with priority index of 0.2 and the last is left-wing group with priority index of 0.3 to track the execution sequence. Penalty value of

[2] In [10], authors experimentally showed such typical max power consumption.

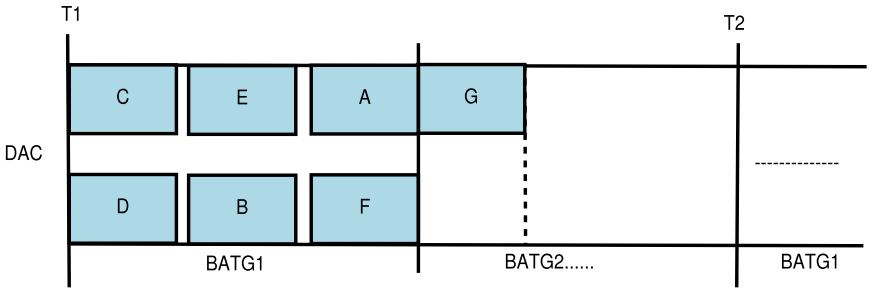


Fig. 5 Sensor data mapping with DAC to server

sensors estimated by using Eq. 4 is (2,2.2,0.4,0.5,0.6,4.5,5.1). Penalty threshold value calculated as 4.8. So, sensor G will be discarded from the transmission queue. Unsuccessful sensor G will take initiative to transmit data to the DAC in the next BATG. Each DAC has a processing capacity limit (let say, 512 bits), which varies with the number of servers associated with it. Based on the descending order of priority (lowest PI to highest), data are placed on servers. Hence, all the sensors expect G has transmitted data successfully to the server from DAC. Hence, the success of sensor in first BATG is $\sum_{i=1}^{7} S_{(i,t,G)} = 6$. Total energy consumed by sensors in the first BATG to place the data in the server is (2.37+2.37+2.12+2.12+2.21+2.01) = 13.02 (watt). In the second BATG, only the sensor G will try to send data to DAC with the same procedure, but in this case, DD_{ijk} will chance as the position of sensors will change as per Eq. 1. Energy consumption for the second BATG is 1.97 (watt). Hence, BATG size $SBATG_k$ is 2 (taken as a round figure).

A pictorial representation for success of sensors per BATG within a single transmission period (τ) is shown in Fig. 5.

6.3.2 Energy optimization strategy for SRs

On the server side, user-initiated service request (SR) can be real time or non-real time. Each real-time SR has some parameters like service execution time, deadline, etc. Let say, six real-time SR request for access to the server for data information. Sensors parameters are shown in Table 2. Based on these parameters, we can estimate the individual weight of each SR by using Eq. 20. The energy required by SRs to execute these requests on servers has been observed as 1.93, 2.76, 3.4, 4.7, 2.06, 4.2 watts, respectively. We have shown individual execution parameters, and allocation of SRs to the

Table 2 SR parameters

U	*SET*	*DL*	*IW*	*UC*	*POW*
SR_1	13	45	0.28	6	1.93
SR_2	22	60	0.36	7	2.76
SR_3	36	90	0.4	8	3.4
SR_4	60	120	0.5	9	4.7
SR_5	21	90	0.23	8	2.06
SR_6	72	180	0.4	10	4.2

VMs has been carried out according to the shortest deadline first (SDF). Hence, SR_1 will be allocated to VMs as it has the shortest deadline among all. Allocation sequence will be SR_1, SR_2, SR_3, SR_5, SR_4 and SR_6.

7 Employing first fit decreasing (FFD) technique in the proposed model

The VM allocation problems have similarities with the bin packing problem. There is a set of VMs with different resource demands (items with their associated weights) which have to be mapped into servers (packed into bins) in such a way that the overall resource utilization is maximized (the number of bins used is minimized). It has been observed that greedy heuristics for the bin packing problem give near-optimal solutions. One such heuristic technique is FFD which has been shown by the authors in [23, 29]. This FFD technique consists of sorting the items in decreasing order by weight and sequentially placing them into the first bin in which they fit. Applied in the context of virtualized data centers, this FFD technique would result in minimizing the number of active servers. We have employed this strategy in our algorithm and compared this FFD model with our proposed techniques for IoT-Cloud platform.

Algorithm 4: FFD based Scheduling Strategy

Input: Transmission period τ, N number of sensors, M number of DAC, η number of servers , Position Parameters (P,P')
Output: Size of Iteration ($SBATG_k$) for each transmission period
Initialize Total weight = 0;
Initialize Success of sensors = 0;
Estimate the amount of data sensed by sensor using the equation 8;
Sort all the sensors in decreasing order of its weight (amount of sensed data) ;
Transmission of data done by the sensor which first fit (highest weight among all sensors) in the DAC and calculated using the equation 10;
for *(sensor 1 to N)* **do**
    if *(Data Bandwidth of DAC* $\geq$ *Total weight)* **then**
        Data processed and stored in the server by the equation 11;
        Calculate the energy required for sensing, transmitting and processing data using equation 15, 16 and 17 respectively;
        Total weight = Total weight + weight;
        Success of sensors = Success of sensors + 1;
        Call the function EC();
if *(Success of sensors* $<$ *N)* **then**
    Process again the failure sensor's data;
    Call the Algorithm itself again;
else
    Wait for the next transmission period of τ ;

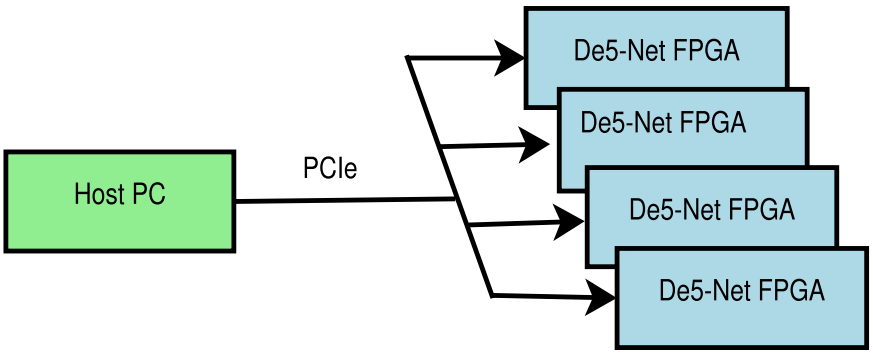


Fig. 6 PCIe communication architecture

8 Evaluation and results

In this section, we have discussed about the FPGA-based IoT-Cloud setup, simulation-based experimental setup, results and analysis and compare the results of the existing model with the proposed work.

8.1 FPGA-based IoT-Cloud setup

Intel (Altera) Terasic de5-net development kit supports Stratix V FPGA processor. We performed our testing on this Intel FPGA board with the specification as follow: Altera Stratix V GX FPGA (Terasic e5-net), 50MHz Oscillator, DDR3 SDRAM and 4 PCI (Peripheral Component Interconnect) Express hard IP blocks that support for PCIe Gen1/2/3 [9]. The Rectangular Regions (RRs) of the FPGA chip are used to perform operations on base station DAC and servers also. We have connected multiple Intel de5-net boards

OpenCL (Open Computing Language) application has been used with Altera SDK (Software Development Kit). This application allows the FPGA users to develop the platform for parallel computing in an abstract away on the traditional FPGA hardware by using C/C++ programming language. OpenCL has been implemented on a host PC that runs C/C++ programs, and multiple FPGA boards have been connected with the host PC motherboard through PCIe communication slot which is shown in Fig. 6. The OpenCL is responsible for managing data transfer between host and FPGA, allocating memory on FPGA device and invoking the parallel code executed on FPGA etc. Altera SDK for OpenCL creates several I/O interfaces, such as memory controller to read and write data from external and internal memory, and on top of that, it creates the PCIe communication link for data communication [34]. The actual bandwidth of PCIe is measured on DE5-Net FPGA board by implementing PCI Express Gen2 and Gen3 hard IP cores that support a maximum payload size of 512 Byte [13].

we have constructed synthetic SR set for our validation purpose. These SRs are taken from well-known Benchmark SR set named “EPFL” benchmarks [2]. This benchmark consists of numerous SR such as arithmetic and control unit. Power

Table 3 Benchmark SRs execution power (watts in unit)

PEs	SR_1	SR_2	SR_3	SR_4	SR_5	SR_6	SR_7	SR_8
FPGA	0.691	2.980	0.409	0.618	1.249	3.142	4.328	2.145

Table 4 capacities and efficiencies of IoT-Cloud

	Capacity	Efficiency
Cloud node	53.5 Million MIPS	500 MIPS/W
Cloud node (micro)	6.5 Million MIPS	100 MIPS/W
Wireless sensor	1 MIPS	480 MIPS/W
Smart device	2000 MIPS	50–1000 MIPS/W
Connected vehicle	2000 MIPS	1000 MIPS/W
WiFi link	150 Mbps	300 nJ/bit

consumption of synthetic SRs corresponding to FPGA has been evaluated on FPGA-based IoT-Cloud platform.

8.2 Simulation-based experimental setup

We have presented simulation-based results from the solution of the EAAM strategy to illustrate IoT services on smart environments. We have analyzed the execution time and energy efficiency of the "IoT-Cloud" solutions. The flexibility of "IoT-Cloud" infrastructure has been exploited by the services of the IoT. "IoT-Cloud" architecture is mainly composed of three layers: (1) a device layer, containing wireless sensors, (2) base station nodes hosting cloud applications, and (3) SR handling by the cloud servers. Due to the limited information disclosed by the cloud and network operators, in our paper, we have taken an approximate value utilizing the information given in [27] (cloud equipment) and [12] (wireless sensors), which are presented in Table 3.

Smart wireless sensor devices have a processing efficiency of 1000 MIPS/W, and each WSN has the processing complexity of 500 instructions per bit for analyzing the data. We have also considered that the packet size generated by the wireless sensors have an average length of 127 bytes (i.e., standard packet size in IEEE 802.15.4). The average communication delay which is the effect of queuing delays (due to their limited transmission capacities) is included in the simulation. We assumed an average delay of 5 ms for wireless links and the threshold value of 15 meters. Wireless nodes have an average processing delay of 4 ms, and cloud nodes have a processing delay of 1 ms. In the following, we have computed the EAAM strategy for different system parameters, such as transmission and processing efficiencies with respect to time and energy, number of users (SR) (Table 4).

We have evaluated the performance of EAAM using the MAC protocol MATLAB R2018b simulation data [31]. The parameters of simulations are listed in Table 5.

Table 5 Parameters of simulations

Parameters	Capacity
Area	100 ×100 to 200 × 200
Base station location	50 to 200
Number of sensors	40 to 60
Initial energy	0.5J
Sensing range	15
Transmitting circuit energy consumption (nJ/bit)	50

8.2.1 Experimental input parameters

Parameters that have been considered to evaluate performances are listed below:

1. *Number of Sensors* Number of sensors varied from 40 to 60 within a smart environment scenario.
2. *Number of DACs* Number of DACs or base stations varied from 2 to 6 with our network setup.
3. *Number of Servers* Number of servers varied from 2 to 6 with our FPGA-based simulation testbed.
4. *Number of PEs* Totally 8 to 24 PEs of same-type PEs have been used for simulation.
5. *Number of SRs* 30 to 50 number of SRs can be requested by the users at the same time which have been used for simulation.
6. *Processing Efficiency* We have considered processing efficiency of a DAC in between 100 to 500 MIPS/W to show the average power consumption.
7. *Packet generation rate* Packet generation rate by the wireless sensors ranging from 0.5 to 2 packets/second has been considered for the simulation.

Each result is generated by executing 100 different instances of each data set type and then taking the average over these 100 runs. Reconfiguration overhead is not shown here separately, as we have included it within SRs execution time.

8.3 Results and analysis

Tables 6 and 7 show the performance of the EAAM strategy over a different number of processing resources or servers, sensors and SRs. Runtime of EAAM strategy is measured in terms of second and power in watt. Here, we assumed that each server or DAC is composed of 4 PEs. The energy cost model in the simulations follows the Amazon EC2 and CloudSim simulator standard small instance type [16, 17].

Sensors sense the data and transmit the data to the DAC. More number of sensors means more data; hence, more processing is required. Resource utilization needs to increase for more processing; hence, more power is required. In Fig. 7a,

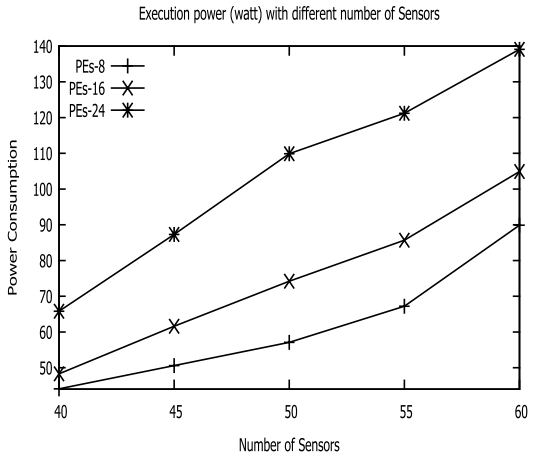
Table 6 Strategy execution time (sec) and power (watts) with different number of PEs

DAC	*PEs*	*N*	EAAM		FFD	
			POW_{Total}	*Runt-ime*	POW_{Total}	*Run-time*
2	8	40	45.3	47	44.1	45
		45	51.7	58	50.6	51
		50	58.2	78	57.1	77
		55	65.9	112	68.2	114
		60	80.4	136	89.9	138
4	16	40	48.6	55	48.3	54
		55	61.3	63	61.6	65
		50	70.7	87	74.5	91
		55	79.2	120	85.7	126
		60	95.9	141	104.9	161
6	24	40	58.2	58	65.8	64
		65	72.1	74	87.3	81
		50	89.9	115	109.9	123
		55	98.1	133	121.2	151
		60	119 .9	150	139.1	178

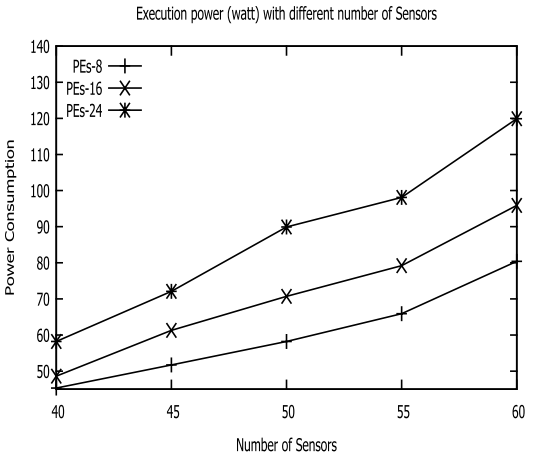
Table 7 SR execution power (watts) and time (sec)

η	*PEs*	*SRs*	EAAM		FFD	
			$POW_{execution}$	*Run-time*	$POW_{execution}$	*Run-time*
2	8	30	68.3	12.04	67.9	11.89
		35	83.1	15.48	83.7	14.97
		40	103.7	18.34	104.1	18.78
		45	121.2	22.93	120	23.11
		50	130.4	24.43	131.6	25.63
4	16	30	71.2	13.76	70	13.07
		35	88.7	16.05	85.1	16.98
		40	111.3	20.64	112.8	21.53
		45	130.1	24.43	129.7	26.27
		50	151.2	27.17	153.3	29.49
6	24	30	87.3	15.23	88.1	16.07
		35	99.2	18.77	98.3	20.39
		40	129.9	23.67	133.7	25.59
		45	144.5	27.09	149.2	28.99
		50	159.1	30.31	164.7	32.01

b, we show how power consumption increases with the number of sensors. One shows the power consumption for existing FFD strategy, and another shows result for EAAM technique. Graphs clearly depicts that the FFD strategy performance

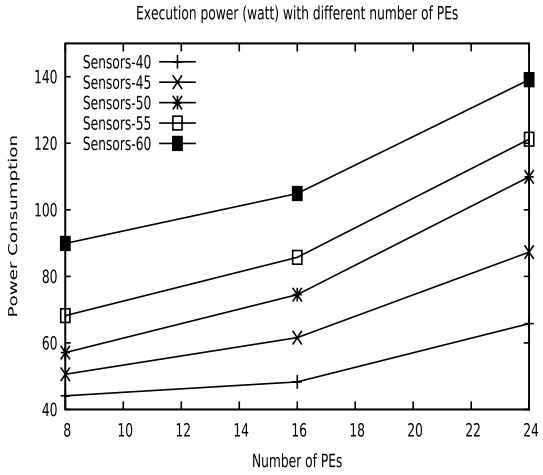


(a) Power consumption by "FFD" with different number of sensors

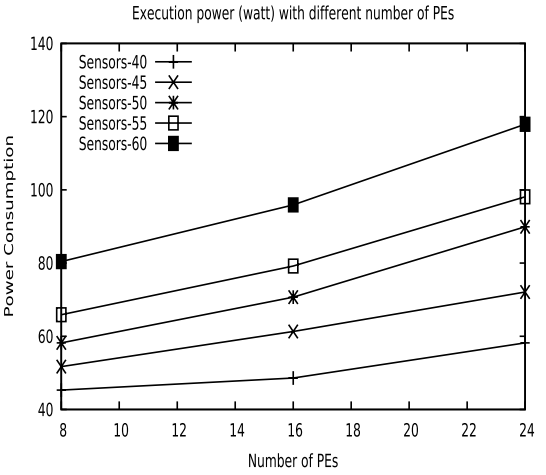


(b) Power consumption by "EAAM" with different number of sensors

Fig. 7 Power comparison



(a) Power consumption of "FFD" with different number of PEs

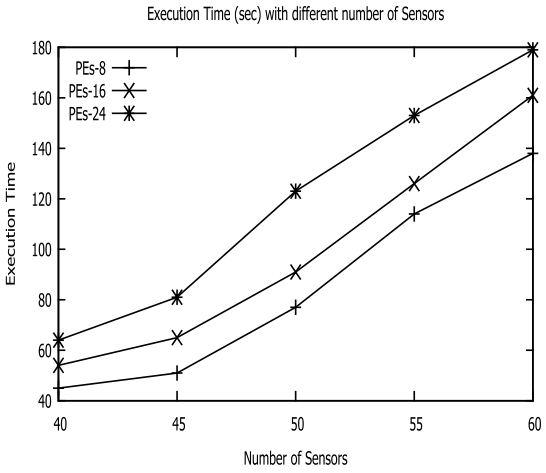


(b) Power consumption of "EAAM" with different number of PEs

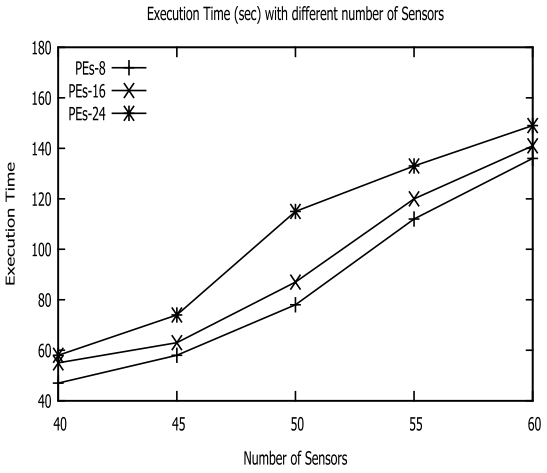
Fig. 8 Power comparison

degrades for larger scenario where EAAM has shown steady behaviors. Sensors count increases with keeping the PEs number as constant.

The increase in the number of servers or PEs increases processing capacity limit of DAC. Hence, DAC can accommodate more number of sensors for processing data to the servers means the number of success increases per iteration or BATG. Low resource (PEs) utilization raises the size of BATG as number of sensors increases. Hence, repeated action required by the sensors means more energy consumption. We have shown two power optimization techniques for IoT-Cloud infrastructure. Both of these techniques are quite energy efficient. FFD strategy exhibits good performance with the lower number of servers or PEs but once the count starts increasing, the "EAAM" outperformed the FFD. This observation is supported by Fig. 8a, b. These plots clearly depict that power consumption

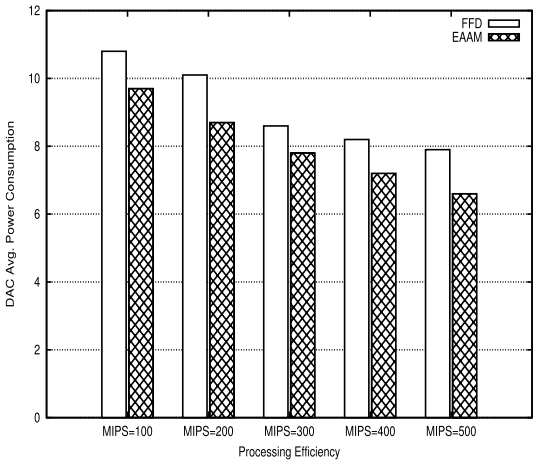


(a) Execution time by "FFD" with different number of sensors

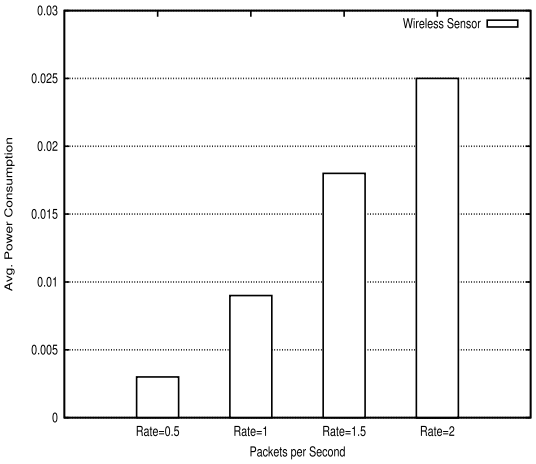


(b) Execution time by "EAAM" with different number of sensors

Fig. 9 Runtime comparison



(a) Power (watt) comparison of two strategies with different Processing Efficiency



(b) Power (watt) consumption by DAC for different packet size

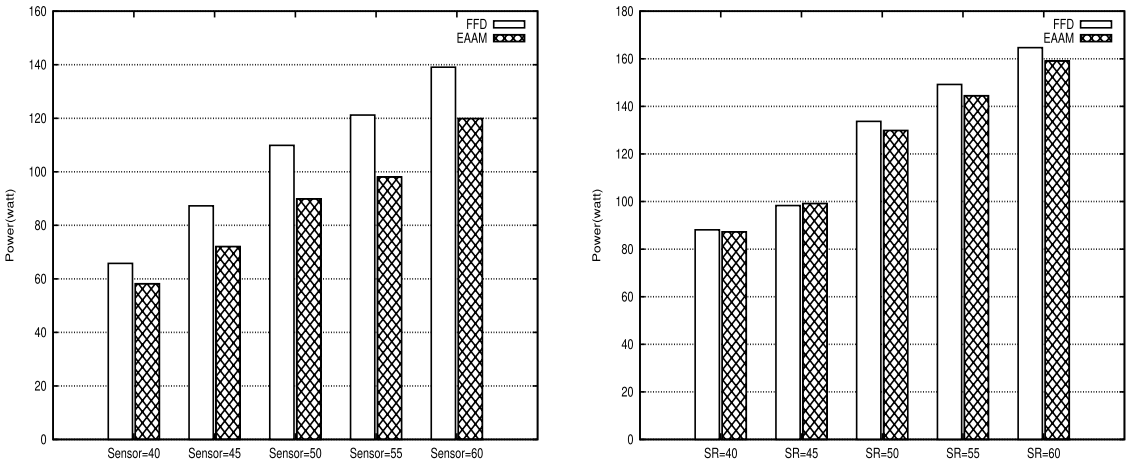
Fig. 10 Power comparison

rate gradually increases with the increase in servers or PEs count (sensors count remain constant).

Execution time has a proportional relation to the number of sensors which is shown in Fig. 9a, b. Figures depict that power increases linearly at a certain point of time but after that, it grows exponentially. From the figures, we can conclude EAAM is stable than the FFD strategy.

8.4 Comparison with existing work

The existing technique FFD and EAAM strategies performances are very close to each other at a certain point of time, but when the number of sensors or PEs starts



(a) Power (watt) comparison of two strategies with different number of sensors

(b) Power (watt) comparison of two strategies with different number of SRs

Fig. 11 Power comparison

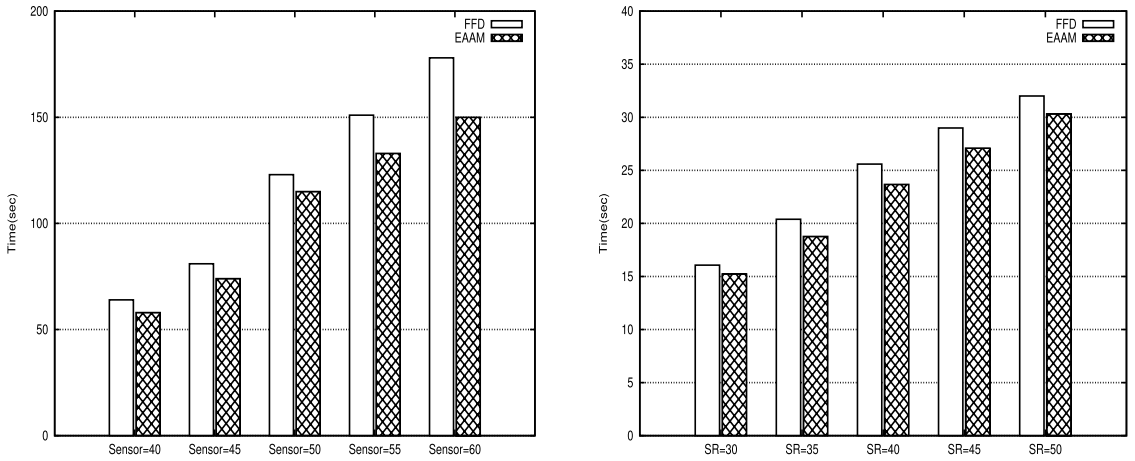
to grow rapidly, FFD fails to generate the optimized results where our proposed strategy EAAM produces better results. We have shown three bar graphs where we have demonstrated the performances of the two strategies.

Figure 10a describes the power comparison with respect to the processing efficiency of the system. Based on the data types, end devices are configured with different processing efficiency. Such as for video processing function, IoT-Cloud allocates higher processing efficiency, i.e., 500 MIPS/W. We have considered efficiency in between 100 and 500 MIPS/W and shown the average power consumption by the DAC to process data. Micro-cloud node has low processing efficiency than the larger cloud node, but that does not ensure power consumption will be reduced. DAC power consumption for both the mechanisms with different processing efficiency is described in that figure.

Similarly, Fig. 10b compares the power consumption of the DAC for different packet generation rates by the wireless sensors. Accuracy of the system services can be increased by the higher bit rate, but it has a negative impact on the power consumption. Due to the limitation of processing capacity of actuators, lesser packet rate actually helps to reduce the traffic and power consumption to the cloud. We can observe from the bar graph that the total power consumption is reduced for cloud infrastructure as long as the packet size is lower than 1.5 packets/sec.

We have shown the power consumption of the two strategies with different number of sensors in Fig. 11a where PEs count (with 24 PEs) remains constant. At the starting, when sensors count was low FFD performed better than our proposed strategy, but for a higher count of sensors EAAM outperformed the existing technique.

The bar graph in Fig. 12a exploits the execution time of both the strategies. Each strategy takes some time (seconds) to complete its operations, which is known as execution time or run time. Execution time increases with the number of sensors for a fixed number of PEs. The number of sensors increases means it will take a longer time to finish operations for all sensors activities (sensing, transmitting and



(a) Runtime (sec) comparison of two strategies with different number of sensors

(b) Runtime (sec) comparison of two strategies with different number of SRs

Fig. 12 Runtime comparison

processing). In this case, EAAM also produce better performances than FFD after a certain limit of sensors count (PEs count remains constant as 24).

User-initiated SRs are executed on the server side. SR consumed energy is based on the individual weight (IW_i). As the number of SR increases, it demands more number of resources (PEs) to serve the SR. Hence, more energy is required. We have observed the scenario where energy consumption is increasing with the number of SRs for a fixed number of PEs (24 PEs) for both of the strategies. In our case, EAAM with SDF technique to allocate the SRs in VMs has produced close results as FFD. We have compared both the results as shown in Fig. 11b. Execution time requirement for both the strategy is shown in Fig. 12b.

9 Conclusion and future work

In this paper, we have proposed a strategy for sensor data allocation and real-time SR allocation in "IoT-Cloud" architecture such that overall energy consumption is minimized. We developed the strategy called "EAAM" and evaluated with simulation-based experiment, and outcomes are promising. The proposed approach is capable of delivering higher resource utilization with low energy consumption over various simulation scenarios. The comparison with the existing work reveals that the proposed strategy outperforms the existing application management strategies.

Future work will devise the application management strategies for heterogeneous "IoT-Cloud" platforms where the serves will be composed of FPGAs, GPUs and CPUs etc. This individual processing capability will make the problem challenging in order to achieve high resource utilization. Additional work will focus on making the proposed architecture secured. Fear of piracy and cyber attacks are common nowadays; hence, future attempts will be toward making the platforms robust and secure from malicious attacks. Further future attempts will be made for software monitoring, fault detection, and automated error correction in "IoT-Cloud" systems.

References

1. Alamri A, Ansari WS, Hassan MM, Hossain MS, Alelaiwi A, Hossain MA (2013) A survey on sensor-cloud: architecture, applications, and approaches. Int J Distrib Sens Netw 9(2):917923
2. Amarú L, Gaillardon PE, De Micheli G (2015) The EPFL combinational benchmark suite. In: Proceedings of the 24th International Workshop on Logic and Synthesis (IWLS), No. CONF
3. Bandyopadhyay D, Sen J (2011) Internet of Things: applications and challenges in technology and standardization. Wirel Pers Commun 58(1):49–69
4. Barcelo M, Correa A, Llorca J, Tulino AM, Vicario JL, Morell A (2016) IoT-cloud service optimization in next generation smart environments. IEEE J Sel Areas Commun 34(12):4077–4090
5. Botta A, De Donato W, Persico V, Pescapé A (2014) On the integration of cloud computing and internet of things. In: International Conference on Future Internet of Things and Cloud (FiCloud). IEEE, pp 23–30
6. Buyya R, Calheiros RN, Li X (2012) Autonomic cloud computing: open challenges and architectural elements. In: Third International Conference on Emerging Applications of Information Technology (EAIT). IEEE, pp 3–10
7. Fahmy SA, Vipin K, Shreejith S (2015) Virtualized FPGA accelerators for efficient cloud computing. In: IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom). IEEE, pp 430–435
8. Filelis-Papadopoulos CK, Giannoutakis KM, Gravvanis GA, Tzovaras D (2018) Large-scale simulation of a self-organizing self-management cloud computing framework. J Supercomput 74(2):530–550
9. Firmansyah I, Yamaguchi Y, Boku T (2016) Performance evaluation of stratix v de5-net fpga board for high performance computing. In: International Conference on Computer, Control, Informatics and its Applications (IC3INA). IEEE, pp 23–27
10. Hsu CH, Slagter KD, Chen SC, Chung YC (2014) Optimizing energy consumption with task consolidation in clouds. Inf Sci 258:452–462
11. Huang M, Wu D, Yu CH, Fang Z, Interlandi M, Condie T, Cong J (2016) Programming and runtime support to blaze fpga accelerator deployment at datacenter scale. In: Proceedings of the Seventh ACM Symposium on Cloud Computing. ACM, pp 456–469
12. Ilyas M, Mahgoub I (2016) Smart dust: sensor network applications, architecture and design. CRC Press, Boca Raton
13. Janik I, Tang Q, Khalid M (2015) An overview of altera sdk for opencl: a user perspective. In: IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE). IEEE, pp 559–564
14. Kim B, Psannis K, Bhaskar H (2017) Special section on emerging multimedia technology for smart surveillance system with iot environment. J Supercomput 73(3):923–925
15. Kim HY, Kim PJ (2016) Embedded systems of Internet-of-Things incorporating a cloud computing service of FPGA reconfiguration. US Patent App. 14/999,341
16. Kim KH, Beloglazov A, Buyya R (2011) Power-aware provisioning of virtual machines for real-time cloud services. Concurr Comput Pract Exp 23(13):1491–1505
17. Kliazovich D, Bouvry P, Khan SU (2012) Greencloud: a packet-level simulator of energy-aware cloud computing data centers. J Supercomput 62(3):1263–1283
18. Li B, Li J, Huai J, Wo T, Li Q, Zhong L (2009) Enacloud: an energy-saving application live placement approach for cloud computing environments. In: IEEE International Conference on Cloud Computing. IEEE, pp 17–24
19. Memos VA, Psannis KE, Ishibashi Y, Kim BG, Gupta BB (2018) An efficient algorithm for media-based surveillance system (EAMSuS) in iot smart city framework. Future Gen Comput Syst 83:619–628
20. Mishra SK, Puthal D, Sahoo B, Jena SK, Obaidat MS (2018) An adaptive task allocation technique for green cloud computing. J Supercomput 74(1):370–385
21. Misra S, Chatterjee S, Obaidat MS (2017) On theoretical modeling of sensor cloud: a paradigm shift from wireless sensor network. IEEE Syst J 11(2):1084–1093
22. Nunez-Yanez J, Amiri S, Hosseinabady M, Rodríguez A, Asenjo R, Navarro A, Suarez D, Gran R (2019) Simultaneous multiprocessing in a software-defined heterogeneous FPGA. J Supercomput 75(8):4078–4095
23. Panigrahy R, Talwar K, Uyeda L, Wieder U (2011) Heuristics for vector bin packing. research. microsoft. com

24. Ren S, He Y, Xu F (2012) Provably-efficient job scheduling for energy and fairness in geographically distributed data centers. In: IEEE 32nd International Conference on Distributed Computing Systems (ICDCS). IEEE, pp 22–31
25. Sivagami A, Pavai K, Sridharan D, Murty SS (2010) Estimating the energy consumption of wireless sensor node: Iris. Int J Recent Trends Eng Technol 3(4):141–143
26. Suciu G, Vulpe A, Halunga S, Fratu O, Todoran G, Suciu V (2013) Smart cities built on resilient cloud computing and secure Internet of Things. In: 19th International Conference on Control Systems and Computer Science (CSCS). IEEE, pp 513–518
27. Vishwanath A, Jalali F, Hinton K, Alpcan T, Ayre RW, Tucker RS (2015) Energy consumption comparison of interactive cloud-based and local applications. IEEE J Sel Areas Commun 33(4):616–626
28. Vivek V, Srinivasan R, Blessing RE, Dhanasekaran R (2019) Payload fragmentation framework for high-performance computing in cloud environment. J Supercomput 75(5):2789–2804
29. Wadhwa B, Verma A (2014) Energy and carbon efficient VM placement and migration technique for green cloud datacenters. In: Seventh International Conference on Contemporary Computing (IC3). IEEE, pp 189–193
30. Xu H, Feng C, Li B (2013) Temperature aware workload management in geo-distributed datacenters. ACM Sigmetr Perform Eval Rev 41(1):373–374
31. Ye M, Li C, Chen G, Wu J (2005) EECS: an energy efficient clustering scheme in wireless sensor networks. In: 24th IEEE International Performance, Computing, and Communications Conference PCCC 2005. IEEE, pp 535–540
32. Zhang Z, Li C, Tao Y, Yang R, Tang H, Xu J (2014) Fuxi: a fault-tolerant resource management and job scheduling system at internet scale. Proc VLDB Endow 7(13):1393–1404
33. Zhu Z, Liu AX, Zhang F, Chen F (2018) FPGA resource pooling in cloud computing. IEEE Trans Cloud Comput. https://doi.org/10.1109/TCC.2018.2874011
34. Zohouri HR, Maruyama N, Smith A, Matsuda M, Matsuoka S (2016) Evaluating and optimizing opencl kernels for high performance computing with FPGAS. In: SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, pp 409–420