UPPSALA
UNIVERSITET

IT16005

Examensarbete 30 hp
Januari 2016

Enabling Energy-Efficient Data Communication with Participatory Sensing and Mobile Cloud

Cloud-assisted crowd-sourced data-driven optimization

Peramanathan Sathyamoorthy

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Enabling Energy-Efficient Data Communication with Participatory Sensing and Mobile Cloud

Peramanathan Sathyamoorthy

This thesis proposes a novel power management solution for the resource-constrained devices in the context of Internet of Things (IoT). We focus on smartphones in the IoT, as they are getting increasingly popular and equipped with strong sensing capabilities. Smartphones have complex and chaotic asynchronous power consumption incurred by heterogeneous components including their onboard sensors. Their interaction with the cloud can support computation offloading and remote data access via the network. In this work, we aim at monitoring the power consumption behaviours of smartphones and profiling individual applications and platform to make better decisions in power management. A solution is to design architecture of cloud orchestration as an epic predictor of the behaviours of smart devices with respect to time, location, and context. We design and implement this architecture to provide an integrated cloud-based energy monitoring service. This service enables the monitoring of power consumption on smartphones and support data analysis on massive data logs collected by a large number of users.

Handledare: Edith Ngai
Ämnesgranskare: Subhrakanti Dey
Examinator: Justin Pearson
IT16005
Tryckt av: Reprocentralen ITC

Acknowledgement

I thank my supervisor Edith Ngai for her support and for helping me out whenever I have doubts. With the freedom she gave I have excelled and mastered many new skills.

Dedicated to

The eyes: ***Wife and Mom***
The shoulders: ***Dad and Father-in-law***
The arms: ***Brothers and Sisters***
Two little stars : ***Charu Nethra , Priyanga***
The 'indeed-in-needs' : All my great ***Friends***
&
The precious rarities: Motivating ***Relatives***

List of papers

This thesis is based on the following paper, which are referred to in the text by their Roman numerals.

I P. Sathyamoorthy , E.H. Ngai. Energy Efficiency as an Orchestration Service for the Internet of Things. In S. Forsstrom and M. Jacobsson, chairs, *Proceedings of SNCNW 2015, the 11th Swedish National Computer Networking Workshop*, pages 41-44. SNCNW 2015.

II P. Sathyamoorthy , E.H. Ngai, X. Hu and C.M. Victor. Energy Efficiency as an Orchestration Service for Mobile Internet of Things. 7^{th} *IEEE International Conference on Cloud Computing Technology and Science, Vancouver, Canada*, November 30 - December 2, CloudCom 2015, [Submitted and Accepted]

Reprints were made with permission from the publishers.

Contents

List of Tables

List of Figures

1. Introduction

> *"The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)"*
>
> – D. Clark, *End-To-End Arguments in System Design*, 1984

IoT is a convergence of a number of technologies such as sensors, IPv6, wireless communication and the Internet. Any real-world objects become smart just by satisfying a few conditions but are not limited to 1) uniquely identifiable; 2) being able to sense or actuate, and 3) being able to communicate [1]. The growth of smart objects are posing challenges to the research community in energy management, data analytic and security [2]. Among these challenges, security and privacy issues affect not only the technical system design level but also in the ethical, behavioural and policy levels. We have powerful analytical tools available with advanced data analysis algorithms [3]. On the other hand, energy management is more complex and chaotic, which is our focus in this thesis.

Berkeley National Laboratory defined energy efficiency as using less energy to provide the same service. The need for energy efficiency highly inevitable in almost every type of industries, companies and organizations including Information and Communications Technology (ICT). Energy management in Internet of Things(IoT) aims at reducing the electricity, which is beneficial for many industries to reduce their electricity bills. As the smart objects becoming smaller in size, their small sized batteries provide limited power for operations. Even the smart appliances are idle, they could indirectly waste a huge amount of energy in long term and eventually increase the electricity bills too. Although ICT can enable energy efficiency across all sectors, at present there is little market incentive to ensure that network-enabled devices themselves are energy efficient. In fact, up to 80% of their electricity consumption is used just to maintain a network connection. Even though the quantity of electricity used by each device is small, the anticipated massive deployment and widespread use makes the cumulative consumption considerable, as reported by International Energy Agency in [4].

Hereafter we narrow our focus on smartphones, which are smart devices that increasing in exponential order over the last ten years. Modern smartphones provide heterogeneous functionalities including a number of sensors.

They are one of the most representative and popular smart objects in the IoT. Nevertheless, smartphones are resource constraint with respect to the battery, memory and computation. It is common for them to offload computation and access remote data storage on the cloud servers via a network. Cloud computing in the IoT leads to thousands of cloud-supported applications and is growing steeply. As a consequence, smartphones are consuming a lot of energy for communication with the cloud. Due to the size limitation, the effort of making powerful batteries is not able to withstand the energy hungriness persisted in the smartphones. It is important to reduce energy consumption when developing new kind of applications.

Smartphones are usually running multiple applications with different operations at a time. It is very difficult to understand and identify the cause of high energy consumption in this asynchronous power consuming environment. It is necessary to provide profiling of power consumption from different levels, including system level as a whole, individual applications, and system calls in operation level. In this thesis, we propose the first iterative novel solution using *Cloud Orchestration* for power management on smartphones. Cloud orchestration aggregates power profiling data from the smartphones and coordinates data storage, data analysis, learning and decision making. From the profiling data, the orchestrator learns the power consumption behaviours and the usage pattern of the participating smartphones. It can answer questions like: which applications are the most energy consuming on the smartphone? What are the characteristics of applications that consume most of the energy? These findings can be used to further optimize the energy monitoring framework. For example, the energy profiler can predict and collect only the most important power consumption data logs on the smartphones. Our cloud orchestrator framework supports the dynamic workflow of processes and adaptive services fitting the needs of different users. It aims for providing overall system power management rather than making part of the system efficient. The cloud orchestration services can be selected and configured dynamically depending on the application characteristics, usage pattern, time and location context. Both offline and real-time services can be supported to provide long-term and large data analytic, or to give real-time alert on unusual events.

The Structure of the chapter. In this chapter, we first elucidate the ambiguity of the term *energy efficiency* (see Section 1.1) and introduce the challenges and the research questions (see Section 1.2) that we address in this thesis. Then we discuss the delimitations of the thesis (see Section 1.3). In the final section, we give the outline of the rest of this thesis (see Section 1.4).

1.1 Energy Efficiency

The term *energy efficiency* is referred to the way of achieving more services for the same amount of energy, or the same services for less amount of energy,

until otherwise specified. Thus, the thesis focuses on *efficient energy usage* rather than *energy conversion energy* and *energy conservation* which are commonly confused with energy efficiency. Energy conversion efficiency is the ratio between the useful output of an energy conversion system and the input, in energy terms. Energy conservation refers to reducing energy consumption through using less of an energy service.

Energy efficient communication is only considered as a constituting disciplinary of *green communications*. However, greenness and optimization of the end-to-end communication flow, computing, the involved intermediate hardware and other components are overwhelming scopes for this thesis. We are more towards energy efficiency as software engineering and communication science, designed and architect from data-driven intelligence and thus heavily limited to smart devices(users) and the applications. We also want the architecture to be flexible, adaptable and comprehend to future changes or optimizations in network layers. Thus by not compromising the improvements attained in domain knowledge.

1.2 Research Questions

Q1-Domain Knowledge: Do we have enough domain knowledge to provide an outstanding energy efficient data communications? What are the other factors worth considering? Why?

Q2-Cloud Solutions: Big data, large-scale cloud computing and services, crowdsourcing and IoT are new concepts that are aiming to boost data-driven intelligence and insights. How to integrate these concepts to design an architecture for energy efficiency?

Q3-Self Efficiency: Is energy consumption of the proposed optimizer within the service-enabled devices less than the energy efficiently made utilized?

1.3 Delimitations

Within the time frame and scope of the master thesis the following tasks are possible as future works

Deploying a fully functional smart profiler with the scale demanded by crowdsourcing: In the case of crowdsourcing it is challenging to get fine-grained logs of system level data. It is sophisticated to study in laboratory conditions by customizing OS image and to use advanced kernel level tools.

Integrating various cloud services as the orchestration that coordinates and enables energy efficiency in the participating devices: It is not so easy to set-up and configure cloud services in a short period of time from different vendors. It has the limitations of the cross-communication and linking of best services.

1.4 Outline

The rest of report is further divided as five chapters. The chapter, *Background*, explains in detail about the theoretical background and related works which influenced the thesis. The next chapter, *Design and Implementation*, explains the how the proof of concept is designed and prototyped. The fourth chapter, *Experimental Results*, highlights the findings with useful results and the challenges faced are discussed so that it would be resolved in future works. The fifth and sixth chapters, *Discussion & Conclusion and Future Works*, wrap up the thesis by discussing the extendibility of the thesis work and by summarizing the thesis work.

2. Background

2.1 Theory

In this section, we highlight fundamental concepts and information needed for the thesis.

2.1.1 Android: System and Performance Constraints

Android™ is an open source system software stack, built on top of the Linux kernel, that provides services and supports a variety of (mobile) devices. Figure 2.1 shows the Android software stack and the architecture of the Android operating system that are referred by applications developers, hardware developers, Original Equipment Manufactures (OEM) and carriers. This leads to variety of customizations and uncontrolled customization which would result in incompatible implementations. Hence, Android tightly integrate OEMs with the *Android Compatibility program*, developers with *Android SDK*, users with *Google Play*. For this thesis only devices with the compatible Android ecosystem is considered.

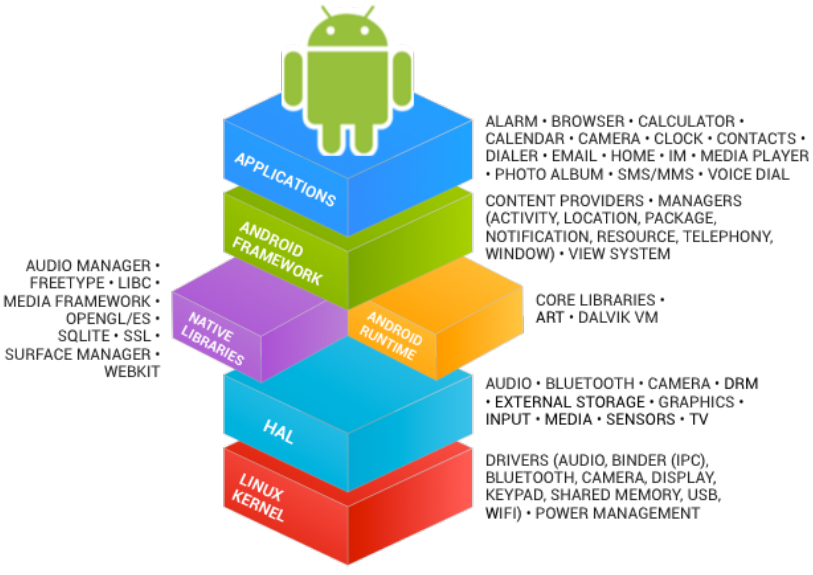


Figure 2.1. Android Stack (https://source.android.com)

2.1.2 Energy Management and Models

In smart devices, energy optimization is needed not only at the hardware level but also at the software level. At the software level, power usage could be optimized mainly at six levels. A detailed survey was done in [5] explaining energy management at the six levels together with a number of examples. The levels are:

- Enabling energy-awareness in operating systems
- Efficient resource allocation
- Context awareness of interaction patterns of users with the devices
- Communication interfaces and sensors management
- Integration of cloud computing services

Measuring power consumption of resources such as CPU, Memory, Display, Communication is extremely helpful in finding the energy-hungry applications, background services, and processes. In [6] energy cost of task offloading over *IEEE 802.11* WiFi and 3G mathematically modelled. For WiFi network, it uses protocol parameters such as *data rate, base rate, contention window size* to derive the formula. For 3G/4G, Radio Resource Controller (RRC) states were taken considered and the total energy consumption calculates as three parts: *promotion signalling, data transfer, tail energy*

2.1.3 Cloud Architecture and Characteristics

The U.S. National Institute of Standards and Technology(NIST) defines "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". There are five essential functional characteristics of cloud computing:

1. On-demand self-service
2. Broad network access
3. Resource pooling
4. Rapid elasticity
5. Measured service

2.2 Data Communication

Data communication is, in essence, digital data transmission between two or more computerized systems or computing devices, namely smartphone (client or peer) and cloud (server) via an interconnected telecommunication medium

such as the Internet. The important components of telecommunications are Data, Client, Server, Infrastructure of the network and protocols. Our focus is more on data communication in the realm of the client system and its applications and its communication interfaces.

Profiling data communications for energy-efficiency is little tricky. We should consider the protocols at various levels on the OSI(Open System Interconnectivity), especially the internet suite of protocols TCP/IP. However finding inefficiency in those protocols will only help to improve RFCs(Request for Comments). Communication interfaces such as WiFi, 3G, 4G LTE, Bluetooth, Near Field Communication, GPS are readily available to measure via APIs given by Operating System provider, namely, Android. Hence, events related to communication interfaces are very useful to profile.

2.3 Related works

Many efforts have been made to enable energy efficiency in smartphones and IoT in general. There is a range of solutions tried out in the *Hardware Architecture* level [8, 9], *Data communication* level [10, 11], *Network infrastructure* level [12] and in *Protocols* optimization [13]. Different tools have been developed to measure energy consumption on smart devices and smartphones. For example, power monitor meter has been used to provide the current with constant voltage 3.7V to the smartphone instead of using the battery [14]. This hardware setup can provide accurate power consumption measurements, but it is a bulky solution not suitable for ordinary users in their daily life.

As Intel summed up in [18], *Software Energy Efficiency* has the significance towards achieving *Computational Efficiency*, *Data Efficiency*, *Context Awareness* and *Idle Efficiency* in a broader sense. Nevertheless, current solutions which try to characterize power consumption on the smartphones usually focus on specific operations, such as communications [10] or interactions with certain hardware components, such as LCD or GPS. There are several common problems in most of the existing solutions, including 1) system as a whole was not considered; 2) trade-off between components was not properly considered; 3) interdependencies of the components were not properly studied; 4) the existing solutions are suboptimal. In order to address the above problems, we need a comprehensive approach to understand the energy consumption of individual applications as well as their interdependence and significance in the whole system. A comprehensive analysis can help the users to identify the most power consuming applications or operations on their smartphones. This can also make the power monitoring process more adaptive to the user behaviour and more energy-efficient in a long run.

Many existing solutions for power monitoring are run on the smartphones nowadays [15, 16]. They can monitor the percentage of battery consumed by different applications. The advantage of these solutions is simple to use, but

the limited memory and computation capability of smart devices make it hard to support more advanced data analysis. It is then difficult to support large-scale and long-term analyses of energy consumption data for both personalized or crowd-based monitoring. Regarding measuring energy consumption, the solid background has been provided in [19]. Internet-of-Things Architecture is a consortium rigorously developing architectural reference models. These models serve as initial guidance potentially towards a concrete architecture for the problem of interest and eventually towards the actual system architecture [20]. In [21], devices orchestration is explained from the business process point of view.

Carat [17] has presented a crowdsourcing approach for collection energy consumption data on smartphones and diagnosing energy anomalies from a community of clients. We share a similar concept of running the analysis on the cloud and further explore the opportunity of cloud orchestration services for smartphones. Instead of taking a black-box process-based approach, we propose a cloud orchestration approach for energy efficiency of smart devices. Cloud orchestration has the capabilities of coordinating different cloud services, such as data storage, analysis, and processing, in a comprehensive framework. Appropriate services can be selected according to the need of individual users. Heuristic profiling can be implemented to reduce the among the log data for energy profiling. This approach is useful in reducing the energy consumption in the energy profiling process itself. Our framework can be extended easily to include new data mining techniques and new services contributed by other users. It supports both individual profiling for personalized services and community analysis using crowdsourced data.

3. Method and Implementation

IoT initially had two visions, one is the *Things-oriented* vision and the other is the *Internet-oriented* vision. The Things vision emphasizes on the sensing and communication capability of different types of smart devices, which can be standalone or embedded into different real-world objects. The Internet vision focuses on the connectivity of the smart devices and their interaction with the Internet. Connecting smart devices to the Internet enables large data storage and analysis that are not feasible on the resource-limited devices. The Internet vision has driven cloud computing for the IoT to provide advanced data processing and data management capabilities.

Later, when new challenges introduced such as unique addressing and storing information, *Semantic-oriented* vision had arisen [22]. According to this new vision, the participating devices are categorised and the orchestration is configured to support scalable and controlled integrated solutions. In this work, we develop the idea of cloud orchestration to provide energy efficiency services for the IoT devices. A cloud orchestrator is a software system that manages the interconnections and interactions of different cloud-based services and processes. It supports dynamic workflows to connect various automated processes and associated resources according to the needs of users and the context environment.

3.1 Cloud Orchestration for Energy Efficiency

The main goal of our system design is to provide energy-efficient decision(s) back to the service enabled smartphones which are participating in the Orchestration. Orchestration, the concept existing in the music world was adopted in process automation of business world by automating, coordinating and managing complex systems, middlewares and services. Energy profiling may impose vulnerability in energy efficiency. Let us give an illustrative example. Even a single and careless piece of code (`while(battery.percentage) println(battery.percentage)`) may cause the system running into an infinite loop and drain all the battery. This small mistake can make any effort of power saving become vain. Hence, there is a need for an intelligent system, which is capable of coordinating different components, finding and categorizing the energy errors. The system should have access to powerful *dynamic control system engine* for fixing such errors. To assist bug fixing, we may need strong insights from the big data of crowdsourced logs/operations over

long period of time. *Orchestration* has the capabilities of integrating different types of clouds, processes and services for power management, which is an ideal solution.

3.2 EEaaS orchestration architecture

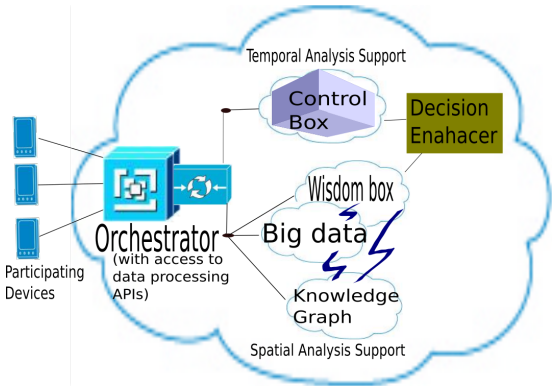


Figure 3.1. EEaaS orchestration for power management

We propose a cloud orchestration architecture, called EEaaS, for power management of IoT devices in Figure 3.1. The design is open and flexible which makes it easy to add, remove and merge with new models and services at any granular level in the orchestrator. The orchestrator coordinates the following components, including *Participating Devices*, *Data Processor*, *Big data storage*, *Knowledge graph*, *Wisdom box*, *Control box* and *Decision Enhancer*.

Participating devices and smart profiler

These are the smart devices in the IoT that interested in minimizing their energy consumption. Upon registration with the orchestrator, a fully customizable and lower energy consuming background *service application* is enabled in the smart devices. This service application sends low-level system call logs periodically and reports abnormal system behaviours spontaneously. These abnormal events may include accidental system crash or unusual battery drain by a specific application. To avoid security and privacy issues, logs are collected anonymously with a unique device profile. This application is not only a logs collector, it also acts as a local *self-controller* attempting to catch energy errors in time and optimize energy profiling in a long run. Its functionalities are regularly updated by the orchestrator. The participating devices report

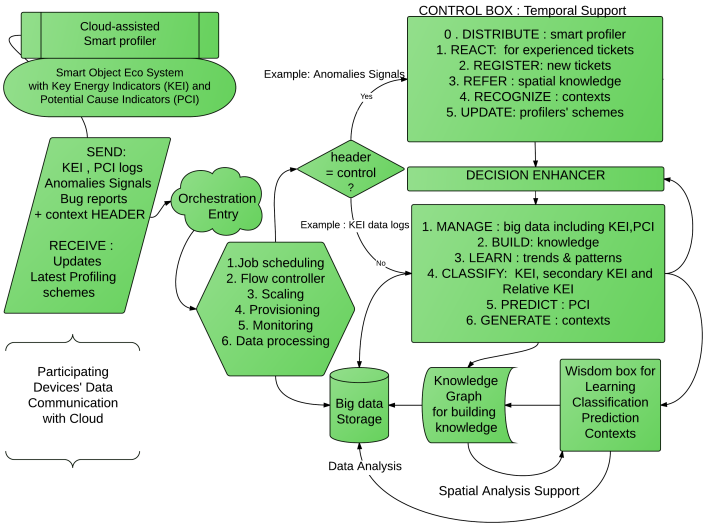


Figure 3.2. EEaaS orchestration insides the box

unusual events to the orchestrator in real-time. Since the unusual events contain only small amount of data, the communication overhead is not so much. On the other hand, the large volume of logged data on resource utilization is reported only when the smartphones are connected to the computer or WiFi network. This is to avoid the continuous data communication through mobile cellular networks. Data filtering and heuristic profiling can be performed to reduce the number of data samples in order to save energy.

Data processor

Data processor is a collection of APIs for various data processing methods accessible to the orchestrator. According to the context and needs of the user, appropriate data processing methods will be chosen by the orchestrator. The data processor supports both big data analysis and temporal data analysis. Advanced data mining techniques can be implemented in the data processor to perform data filtering and data aggregation. For example, it can characterize the energy consumption behaviour of different applications on the smartphones and identify the most power hungry applications.

Big data storage and modern tools

The data produced by the smartphones would be in massive scale over time. In order to handle these data-intensive operations, we need big data storage and

modern cloud programming paradigms such as *Hadoop* and *Apache Flink*. For the deep analysis of sample data, powerful computing languages such as *Python* and *R* are required.

Knowledge graph

When interpreting large volumes of data logs, dynamic knowledge graph is built and keep on updated. Knowledge graph is a knowledge base originally used by Google to enhance its search engine's search results with semantic-search information gathered from a wide variety of sources. Nodes are qualified classes and subclasses with attribute-value pairs so that it provides a clear and structured view of data. Using the knowledge graph, it is then easy to get specific resource utilization and energy consumption data for analysis with respect to location, device model, internet service provider and various specifications.

Wisdom box

Wisdom box contains a set of learning algorithms with a primary focus on building location specific context information (in the spatial domain). It can capture the location of the device and correlate the location with various usage and communication patterns. The wisdom box act as a predictor of trends in data, usage patterns, and system behaviour anomalies. It uses a combination of statistical algorithms and machine learning algorithms to make energy efficient decisions. The decisions that are independent of device, platform and applications are stored in the *Decision Enhancer* in the orchestration. The device, platform and applications specific decisions are updated in the knowledge graph and in the reference graph in the orchestration.

Control box

The control box is a builder of real-time and dynamic self-controller for the participating devices. It can also raise alerts and give time-sensitive feedbacks (temporal domain) to the users. The self-controller is implemented as a service as explained in 3.2. To make the self-controller, even more, intelligent, context related decisions in the spatial domain are used. Feedbacks are received from the participating devices to evaluate the performance of data log collection.

3.3 Architectural layers in the IoT context

In [23], the general architecture of IoT is well explained as four layers. Generally, *Perception Layer* is where all the information collected from the envi-

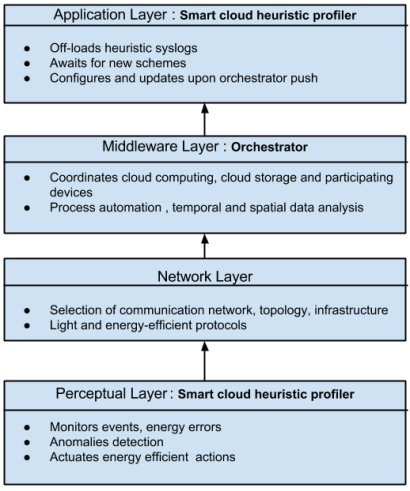


Figure 3.3. Architecture layers in IoT context

ronment using specialized equipment such as sensors. Considering the smartphones nowadays, the perception layer may include the on-board sensors such as a camera, microphone, GPS, light sensor, accelerometer, and gyroscope, etc. It is the layer that provides the digital representation of the physical world. *Network Layer* is for communication and transmission of information. It is supported by different communication modules, such as Bluetooth, GPRS, and WiFi in the smartphones. *Middleware Layer* by its various intelligent cloud computing supports the application layer and acts as an intermediate layer between the network layer and the application layer. The *Application Layer* is fully customized with respect to the users and the purpose of the applications. With respect to the cloud architectural design shown in figure 3.1, both perception layer and application layer points to smart profiler that sense and actuate participating devices. The smart profiler is explained in detail in section 3.2. In figure 3.3, it is shown how the complex architecture could be seen as the layers in the context of IoT.

3.4 Energy-efficient data profiling and communication

Given that data communication is one of the biggest challenges for energy efficiency, cloud-assisted data profiling would seem counter intuitive. Our system design takes smart logging with minimal data communications to reduce the overhead. It provides an alternative solution to fine-grained and real-time profiling, which is both data and computation intensive. We study existing profiling techniques and identify the most important energy features to design a heuristic profiler. This heuristic profiler collects data intelligently and main-

tains minimal communications with the cloud. Our approach can effectively reduce the overhead and energy consumption in the energy profiling process itself.

Key Energy Indicators

Managing and analysing big data is not a major cause of energy consumption. The real bottleneck is collecting data through frequent and intensive communications from the participating devices. Instead of sending all the data logs to the orchestration for analysing, we classify certain system calls and utilization as primary, called them *Key Energy Indicators* (KEI), such as CPU and memory usage. We further categorise the KEI into *Secondary Key Energy Indicators*(SKEI) and Relative Key Energy Indicators(RKEI). SKEI are not so frequently used but are important when they are active, such as GPS and Bluetooth. RKEI are active only when specific applications are running or occur in certain context. Examples of RKEI include camera and activity sensors.

Potential Cause Indicators

KEI capture the energy consumption behaviours and identify the causes of energy consumption. Here comes the *Potential Cause Indicators*(PCI), which can be learned by the orchestration over time. When the system becomes matured enough, the participatory devices collect and report less amount of data. Crowdsourcing further makes it easy to distribute the workload of data collection. In a long run, fewer logs and less communication from participatory devices are required. More sophisticated context-aware models in the orchestration have resulted. The workflows in the orchestration then become more energy-efficient, as it can adapt to the usage pattern and context dynamically (see figure 3.2).

In figure 3.2, the cloud-based smart profiler (top-left), sends the logged data or control messages from the participating devices to the cloud. The logged data are sent together with the *context header*, which is used by the orchestration to indicate the message type. For example, if the context header is *control*, then the orchestrator knows that there is no data logs in the message but events (issues) being reported from the participating devices. In this case, the events are forwarded to control box in order to address the issues. Otherwise, the other messages (data logs) are forwarded to the big data module supported by the spatial data processing unit for classification, prediction and context generation. The data processing unit helps the orchestration to learn what are the KEI so that less significant data can be identified and removed in the future. When the system becomes mature enough, the participatory devices will collect mainly the KEI, so that they can report less amount of data logs to save energy. Through iterative learning, the orchestration can build more sophis-

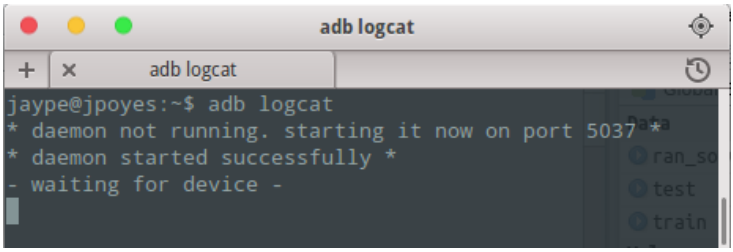


Figure 3.4. Android Debugger Logcat in Shell

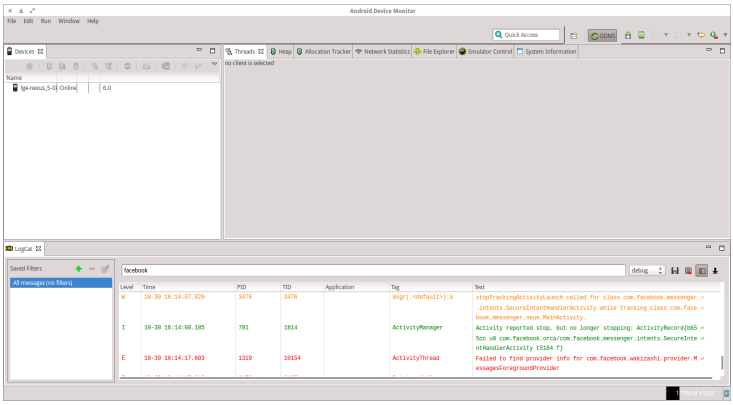


Figure 3.5. Dalvik Debug Monitor Server

ticated context-aware energy models for making a better prediction in energy profiling.

3.5 Implementation

In this section, we explain the prototype development in iterations, tools used and experimental set-up.

Phase 1 - Data Inspection

We first explored the logs of two smartphone models *Samsung SIII* with android version 4.4 and *Nexus 5* with android version 5.0 with android platform tools such as *logcat* for system debug information and *ddms*(dalvik debug monitor server) for debugging applications. These tools helped us to find out behaviours of the system and the applications running on the platform.

Then, we used Trepn [26], for collecting data in suitable formats, namely, *SQLite* and *CSV*. The eclipse plug-in provided by Trepn suffers from poor

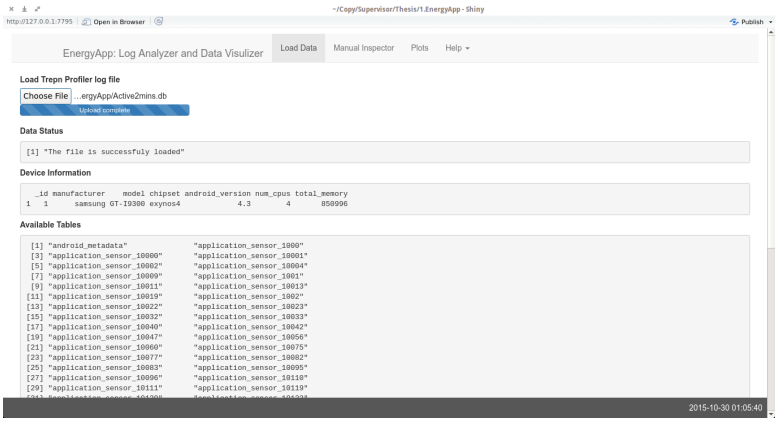


Figure 3.6. Data loader

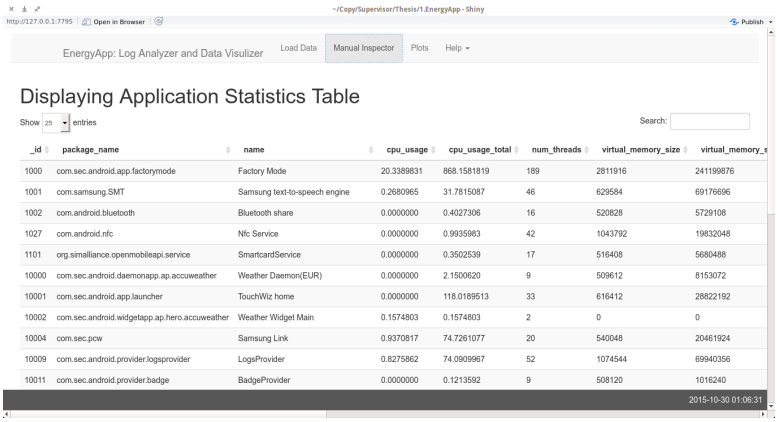


Figure 3.7. Data in sortable table

visualization. Also, the data in CSV format require a lot of cleaning to use. It also has heavy limitations on the number of parameters it can properly profile and hence not self energy efficient. However, the data had a lot of insights. We built a web application *EnergyApp* that helped to study the data in SQLite format. Few pages of the applications are shown in figures 3.6, 3.7 and in 3.8

Phase 2: Key Energy Indicators (KEI)

We realised soon when crowdsourcing methodology is adopted, we will be in need of big data management and analysis with unnecessary overhead and with less influencing features. It will also lead to inefficient profiler implementation. So, we derived KEIs to avoid such an overhead. We relied on two data

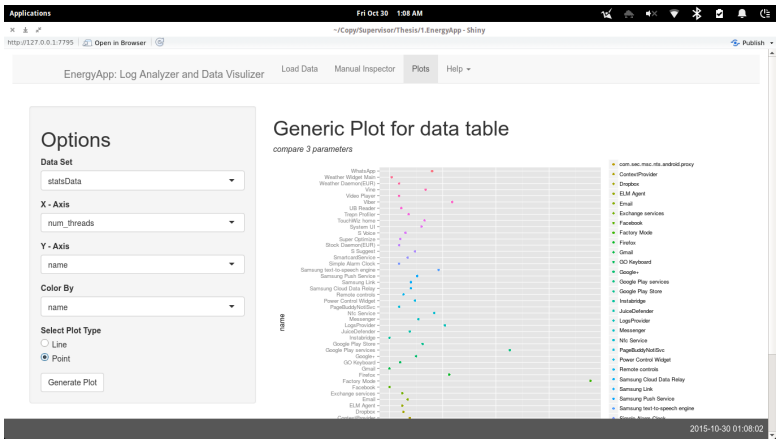


Figure 3.8. Basic Visualizer

Figure 3.9. EnergyDataAdmin (https://peramanathan.shinyapps.io/EnergyDataAdmin)

mining techniques, namely, *Decision Trees* and *Random Forest* to figure out the variables of importance by Gini values. This service is running as Software as a Service application at *https://peramanathan.shinyapps.io/EnergyDataAdmin* and only first version deployed. Homepage screenshot is shown in figure 3.9

Phase 3: Tools for Energy-Efficient Cloud Profiler Prototype

We propose a prototype for cloud profiler with key characteristics including:

- Real-time database with auto-synchronization support such as *Firebase*. Then it would behave smooth while offline and low latency conditions. Thus by utilizing the energy efficiently.

(a)

(b)

Figure 3.10. GraphQL Query language and runtime

- Distributed Data Protocol which is a kind of *publish-subscribe* methodology implemented by *Meteor* for mobile and web applications.
- Client-centric data response, no less, no more data such as *GraphQL*. It reduces the amount of data.
- As a background service possibly no GUI or less GUI. Exploit push notification panel icon as smart notifications. Safety and secure actuation.

We explain in detail about client-centric response because of its importance compared to others. These days applications are heavily using http RESTful services and ad-hoc endpoints to get data resources. The core problem with this approach is that the response data is entirely decided by the server. It means for requested endpoint, the server side application might be written to respond with more data than required. For example, the simple client application showing current weather is getting more information than what the application display on the screen. Facebook is one of the most energy consuming application due to engaged usage and we also realised the same in our experiments. It released a draft RFC and reference implementation *graphql* in 2105 as a remedy to the above-mentioned problem [27]. GraphQL is a data query language and runtime where queries are shaped similar to the data it needed and it is shown in figure 3.10.

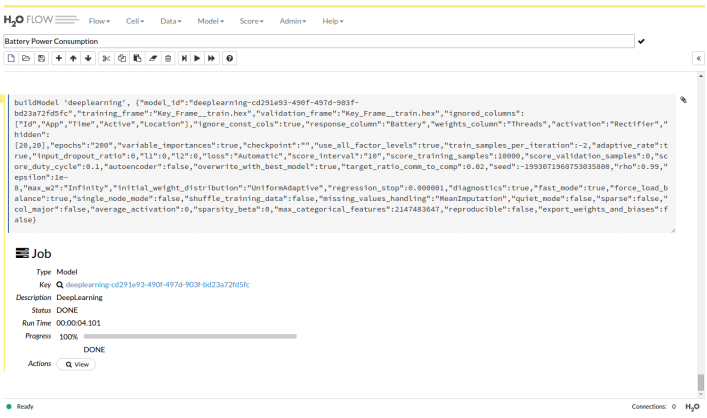


Figure 3.11. Available at `http://h2o.ai`

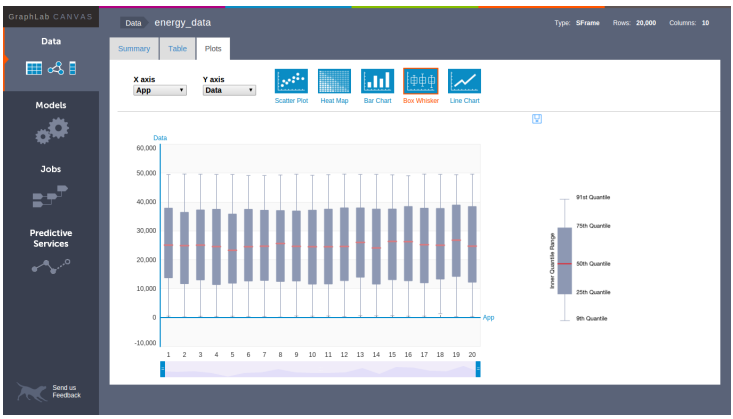


Figure 3.12. Available at `https://dato.com`

Phase 4: Testing Machine learning frameworks for deep learning

We first wanted to figure out the quality of predictions with crowd-sourced data. We have chosen two tools *H2O* supports R, Python and REST APIs and *Dato GraphLab* Amazon EC2 instance supports python and REST APIs. We tested these systems with simulated data (mimicking 100 users, 20 applications, KEI features). Figures 3.11 and 3.12 shows the action screenshot

4. Experimental Results

The smartphone is a system-on-chip architecture with three key components, *Application processor* to handle user applications, *Modem processor* to handle transmission, and reception and *Peripheral devices (I/O)* to interact with users. In smartphones, the power consumption of any I/O component is often higher than the power consumption of the CPU or at the least comparable. In [24], the drawbacks of power models derived from external power meters and software modelling are well explained. Software power modelling does not address the tail power states, which occur when the components remain power on and consume energy even though the CPU is idle. This problem can be addressed by system call tracing to check each component's power state though it may consume more energy. Nevertheless, energy profiling is a very important first step to characterize the power consumption on smartphones.

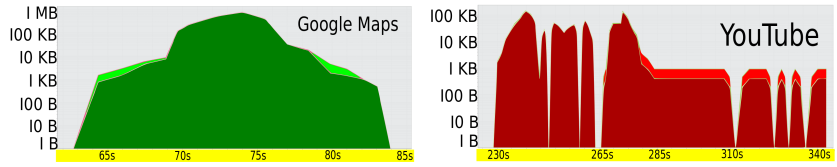
4.1 Power states and data logging

The Advanced Configuration and Power Interface (ACPI) specification has been evolving as common hardware interfaces in Operating System-directed configuration and Power Management (OSPM) for both the end devices and the entire systems. When profiling an individual application or entire platform, it is useful to fetch information about the states of system, device and processors (as shown in table 4.1), so that better decision can be made to achieve energy efficiency. We are currently using Qualcomm's Trepn Profiler [26]. We study the behaviours of the system and resources that applications consumed in the system including CPUs usage, memory usage, and data usage.

Global System States	Device Power States	Processor Power States
G0 Working	D0 - Fully-On	C0
G1 Sleeping	D1	C1
G2/S5 Soft Off	D2	C2
G3 Mechanical Off	D3hot	C3
	D3 - Off	

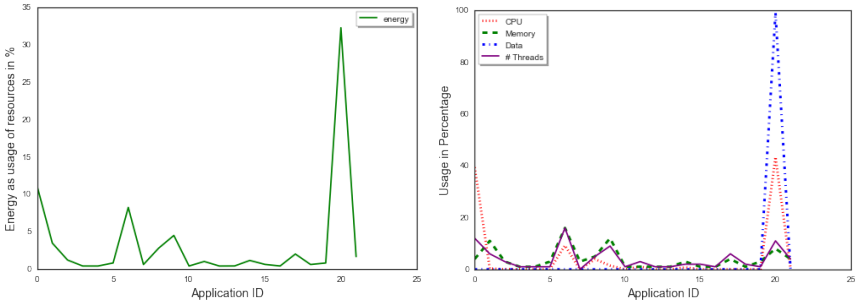
Table 4.1. *ACPI/OSPM defined power states*

As data communication is a major cause of fast energy drain in the smartphones, we show an interesting example on profiling the data communication patterns of mobile applications. Figure 4.1 shows the data communication usage patterns of two popular applications, *Google Maps* and *YouTube*. From the



(a) Google maps *(b)* YouTube

Figure 4.1. Snapshots of profiled data communication patterns



(a) Energy consumption of applications *(b)* Resource utilization of applications

Figure 4.2. Energy consumption and resource utilization

data logs, we observe that both Google Maps and YouTube are running two threads in their applications. The data communication of the two threads in each application share similar patterns, which are indicated by dark and light color in the figure.

Figure 4.1a shows a snapshot of data communication profiling of Google Maps. The y-axis is plotted in log scale, showing the size of data communication at different times. We observe that there is a sudden increase of data communication occurred at time interval [70, 75]. Through careful inspection, we find that it is due to the action of zooming in the map triggered by the user.

While profiling YouTube, a video is randomly picked for playing in the full screen mode. From figure 4.1b, we observe initial aggressive data communications due to pre-fetching. Then, the video is played smoothly with constant data communication from 285s. We believe that the intermittent communication pattern is due to the communication protocol and the reliability of the network.

ID	Application name	Energy consumption (%)
20	Facebook	32.270
0	Android System	11.076
6	Google Contacts Sync	8.238
9	Google App	4.490
1	com.qualcomm.qcrilmsgtunnel	3.464
8	System UI	2.788
17	YouTube	2.006
21	Messenger	1.656
2	Nfc Service	1.204
14	Google Keyboard	1.146
11	CaptivePortalLogin	1.008
5	Media Storage	0.808
19	ES File Explorer	0.804
15	Maps	0.616
18	Google Connectivity Services	0.604
7	Google Dialer	0.602
13	Hangouts	0.410
16	Google+	0.406
10	Calendar	0.404
3	Calendar Storage	0.402
4	User Dictionary	0.400
12	Fit	0.400

Table 4.2. *Ranked energy consumption of applications*

4.2 Cloud-based data profiling and analysis

4.2.1 Energy consumption

In order to understand and visualise the energy consumption pattern, the following experiment has been conducted. We collected the data from four users over a three month period from 1 March 2015 to 31 May 2015. The users have been using Samsung S4 or NEXUS 5 smartphones. The data has recorded the energy consumption and resource usage of the smartphones that were idle or running actively in daily use. We have chosen twenty-two applications that are run by all the four users in this data analysis. These 22 applications range from social media applications, messaging applications, navigation applications, to personal management applications. The data has been cleaned up and processed so that every resource utilization of each application has been summarized as a mean value in an hourly basis. Then, the summarized data are further aggregated to give an overview of energy consumption among all the applications

We compare the total energy consumption and resource utilization distribution among different applications in this experiment. This result helps us to identify the most power consuming applications and understand what resources have been utilized to make them so power hungry. Figure 4.2(a) compares the energy consumption of the 22 applications. The y-axis shows the total energy consumption of each application in percentage (among all applications). The x-axis shows the application ID from 0 to 21. From the figure, we observe that there are four to five applications, which consume the most energy compared with the others. We rank the energy consumption percentage of these 22 applications in Table 4.2. It shows that the most power consuming applications are Facebook, followed by Android Operating System, Google Contacts Sync, Google App.

We further investigate four key energy indicators (KEI) in these applications, including CPU load, memory, data communication, and a number of threads. Figure 4.2(b) shows the resource utilization of the applications in percentage. From the figure, we can see that the applications that consume most energy usually have high utilization in all four kinds of resources. Take Facebook as an example, it has the highest data communication and CPU usage among others. It also has relatively high number of threads and memory usage compared with other applications. We observe similar resource utilization patterns for applications that have high power consumption. The shape of the curves in figures 4.2(a) and 4.2(b) follow very similar patterns. It implies that these four selected resources are very important when profiling energy consumption for the smartphones.

4.2.2 CPU load

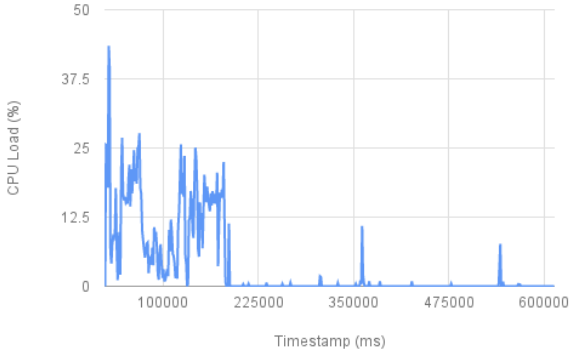


Figure 4.3. CPU load pattern of Facebook App

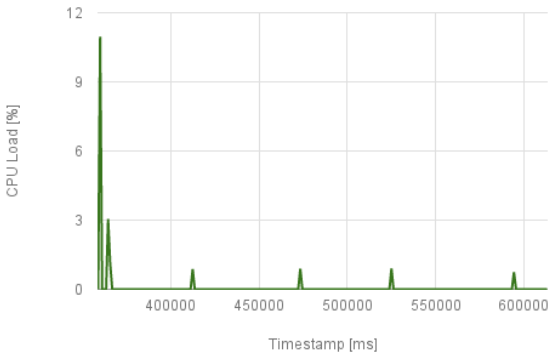


Figure 4.4. CPU load pattern of Google Maps

We also observe different CPU load patterns in the applications. Figure 4.3 shows the CPU load of the Facebook app. We can see that the Facebook app has high CPU use when the app is started. After that, the CPU load is quite random depending on the operations triggered by the user, such as uploading photos or sending messages. Figure 4.4 shows the CPU load pattern of Google MAPs, which is quite periodic due to the regular update of GPS locations. By observing the CPU patterns, it helps us to understand the operation characteristics and energy consumption of different applications.

4.2.3 Threads and memory use

Next, we analyse the correlation of threads and memory use in the applications. Figure 4.5 shows the average number of threads and the average virtual memory use of the 22 applications. As seen from the figure, there is a positive correlation between the number of threads and virtual memory use. Most of the applications use less than 50 threads. However, there are several applications using much more threads than the others. The top application, Google Contacts Sync, uses 162 threads and more than 12000MB virtual memory. The following applications with a high number of threads are Android System and Facebook, which use more than 100 threads. Google App uses almost 100 threads and a lot of virtual memory as well.

Figure 4.6 shows main-memory consumption of the applications. We divide the memory use into different ranges and count the number of applications in each range. The figure shows that most of the applications consume less than 25MB main memory. However, two applications, Facebook and Google Contacts Sync, consume almost 150MB main memory. Google App also has high main memory use of 100MB.

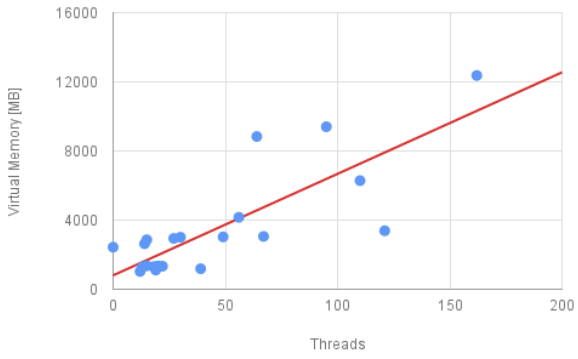


Figure 4.5. Threads and virtual memory use of applications

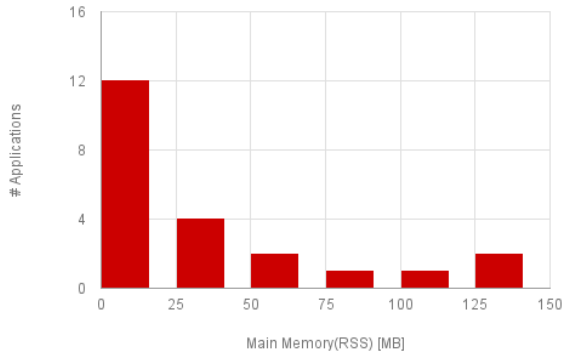


Figure 4.6. Main-memory usage of applications

4.3 Analysis for energy-efficient profiling

Understanding the characteristics of applications and energy consumption patterns on the smartphones are very useful for reducing the energy consumption in the profiling process itself. Through simple analysis, we can make initial observations on what applications consume the most energy and what applications consume insignificant amount. Our profiler can use this information to reduce the amount of data being collected on resource utilization. If we filter out the data from applications that consume less than 1% of the total energy in the system, we can greatly reduce the number of applications that require intense monitoring. Take the 22 applications in Table II as an example, we

```
# ["Device(Id)", "App", "CPU(MHz)", "Memory(MB)", "Data(KB)",
# "Threads", "Time", "Active", "Location", "Battery(%)",
    "Energy(0/1)"]
frmla <- Energy ~ CPU + Memory + Data + Threads + Time + Active
    + Location + Battery

dc_tree_classifier <- rpart(frmla, data=train, method="class",
                    control = rpart.control(minsplit = 50, cp =
                        0))

printcp(dc_tree_classifier)


# Root node error: 14229/40000 = 0.35573
#
# n= 40000
#
#         CP nsplit rel error   xerror      xstd
# 1 0.550144     0  1.000000 1.00000000 0.00672897
# 2 0.107281     1  0.449856 0.44999649 0.00515392
# 3 0.069646     4  0.109635 0.11026776 0.00272865
# 4 0.039989     5  0.039989 0.04062127 0.00167737
# 5 0.000000     6  0.000000 0.00063251 0.00021081
```

Listing 4.1. Recursive Partitioning and Regression Trees

can reduce the number of data samples by 50% while still keeping accurate data logs from applications that consume more than 94% of the total energy in the smartphones. In another word, it can save half of the energy in profiling without losing much accuracy in power monitoring. With cloud orchestration, we can configure the system dynamically to reduce the data samples of applications that are less frequently used or consume little energy.

4.4 Key Energy Indicators Prediction

We have simulated smartphone data for 100 users for 20 applications. We prepared training and test data samples of contains 40,000 records and 20,000 records respectively for an hour of usage. We, first applied simple decision tree regression `rpart`(Recursive Partitioning and Regression Trees) classifier available as R package which is shown in the code snippet 4.2. The visual representation of the cross-validation results shown in the figure 4.7. The decision tree classifier is tested against test data set which result in 92% accuracy. To do the experiment on real data, the data must be tidy and normalized and most importantly it requires sufficient amount of data. It involves careful pre-

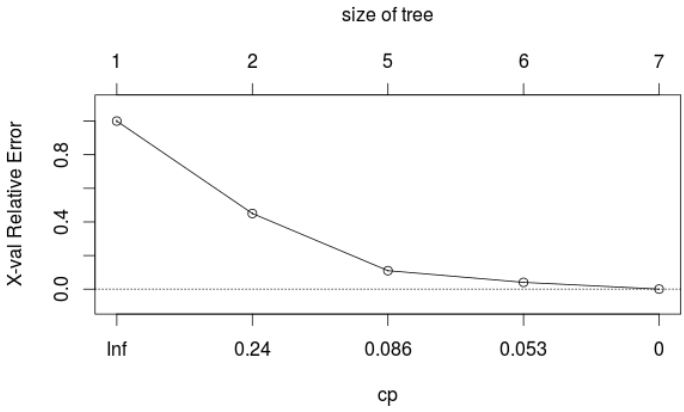


Figure 4.7. Cross Validation with Complex Parameter Table of recursive partitioning classifier (decision trees)

processing of data to make sure the validity of data. In real-time, the data is affected by noise and various interruptions.

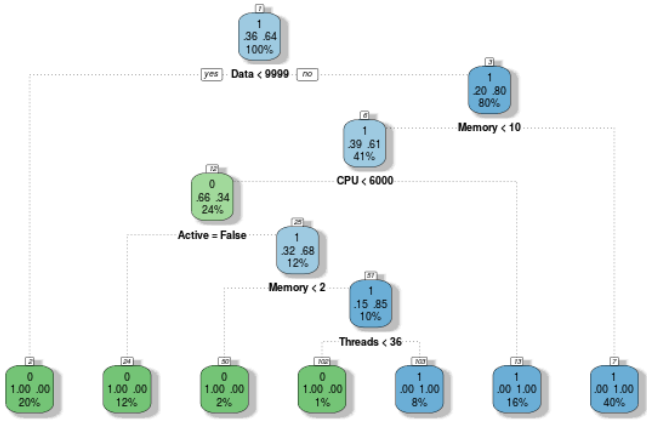


Figure 4.8. Decision Tree, Data in KB and Memory in MB

In the figure, 4.8, the interesting values of variable importance is predicted with respect to their influence. We found that the amount of data communicated and threads heavily influencing energy usage. The model predicts for the given values of features whether energy is high or low (0 or 1) as a binary classification problem. Recursive partitioning that is tree-structured models are used for classification. From 4.9, we confirmed that features such as the amount of data and number threads that applications and memory have

```
## Prediction Part

prediction <- predict(dc_tree_classifier, test, type = "prob")

# 'dc_tree_classifier' is a binary classifier (0 or 1)
# That could predict if the power consumption is low or high
# For the given resource utilization vector
# (Memory, Data, Threads,...) of events

# boolean vars := col 1: is_low(0), col 2: is_high(1)
# And is_low = ~is_high, test$Energy is either low or high
test$Predicted = prediction[,2]
test$Success = test$Energy == test$Predicted
paste("Error: ", 1-length(test$Success[test$Success ==
    TRUE])/nrow(test))

# --- result in --- #
# "Error: 0.0841499999999999"
```

Listing 4.2. Power Consumption Prediction of events w.r.t Resource Utilization

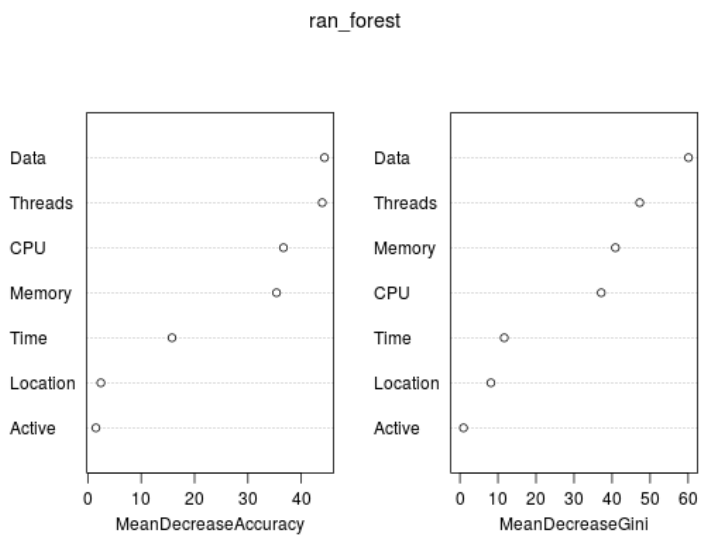


Figure 4.9. Random Forest

the most influence in energy drain by considering the Gini of random forest model.

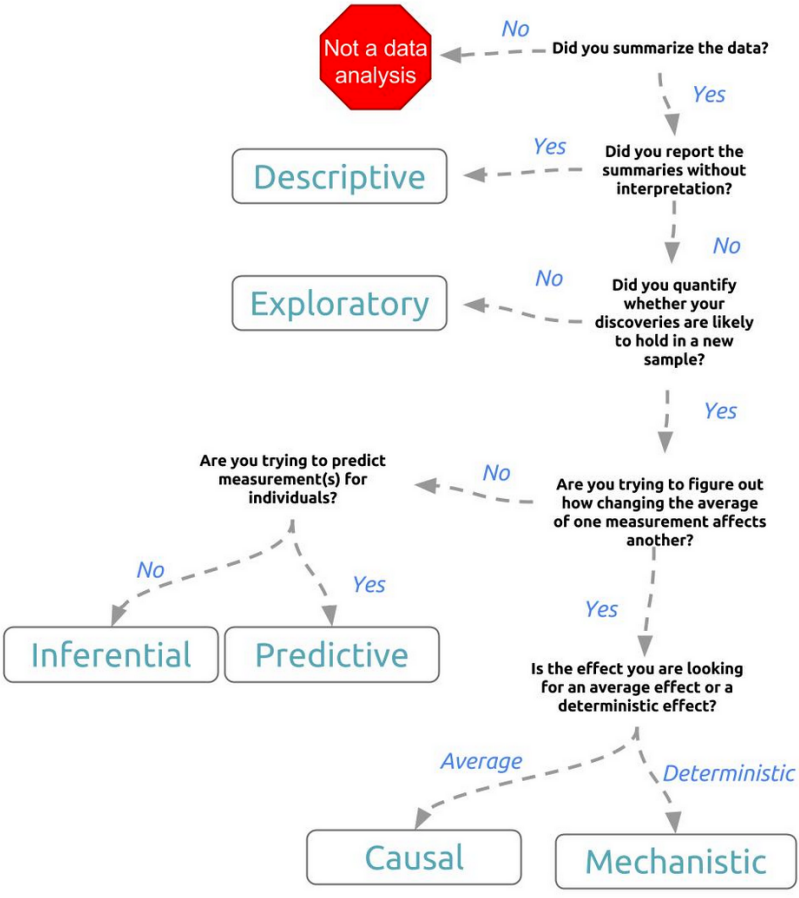


Figure 4.10. Categorization of data analysis question type

We can profile and collect data from smartphone with respect to the contexts where optimization techniques are applied to make the system energy-efficient. Later, by studying the nature of optimization problem we can apply suitable machine learning techniques. The figure 4.11, helps to chose the suitable machine learning methods. The detailed map is available at [30]. In order to define the data analysis question type, the flow chart in the figure 4.10 can be very useful[31].

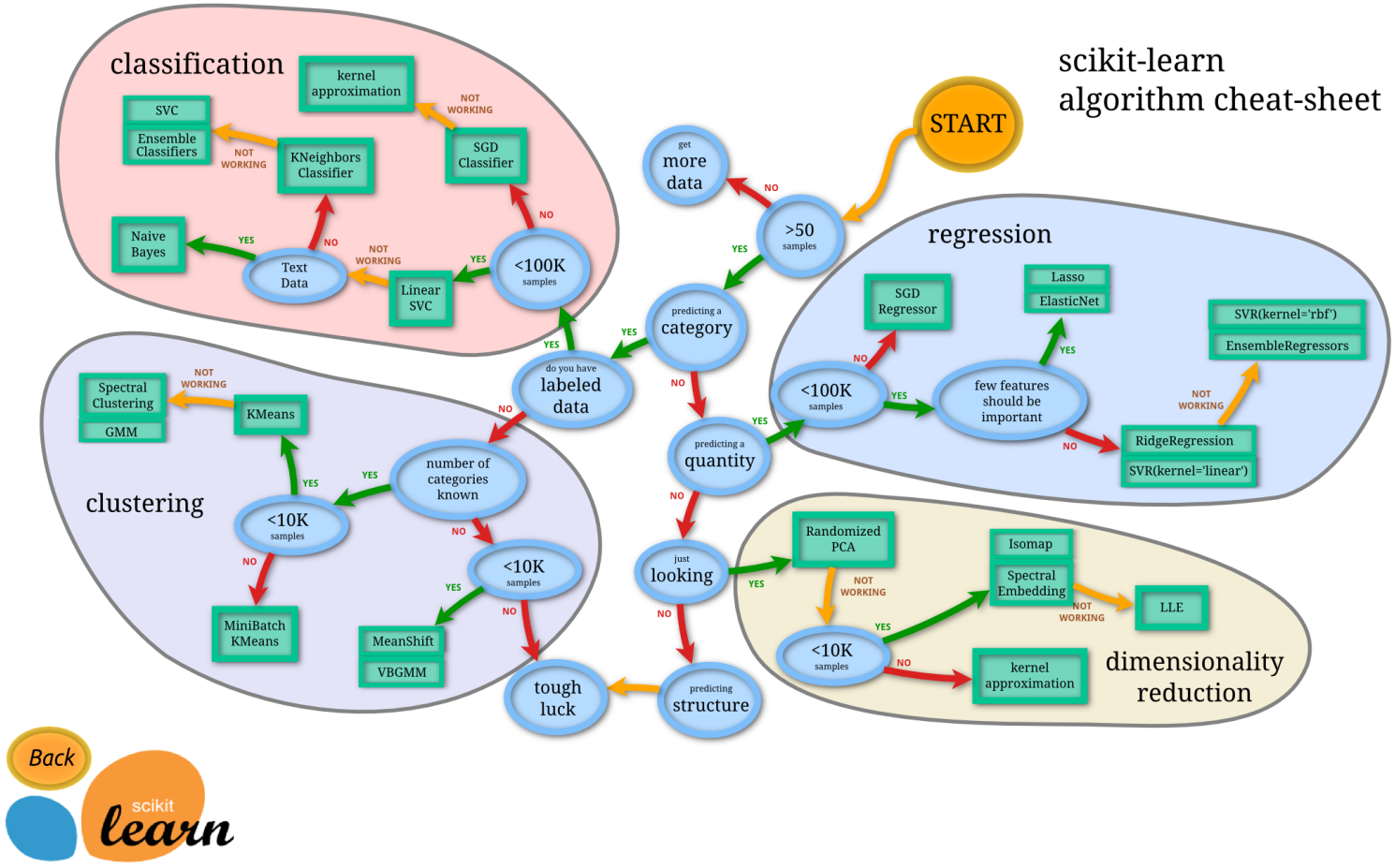


Figure 4.11. Machine Learning (http://scikit-learn.org/)

5. Discussions

Orchestrator is, in essence, the behaviour predictor of the participating devices with respect to time, location and as many as added contexts. The agent application installed in the smartphones report logs to the orchestrator. It will be facilitated by the local validator and action triggers which will be regularly updated by the orchestrator on demand to reduce energy consumption in data logging, communication and computation. In order to successfully deploy such orchestration service, we need to study and explore all the components and their interdependencies in detail. Questions that we plan to further investigate include 1) How to further reduce the energy consuming of profiler? 2) How to reduce data logs reporting and minimize energy consumption in data communication? 3) How to make orchestrator an epic predictor of device behaviours? 4) How to find optimal responsibilities of the local agent by ensuring minimal computation and resources? 5) Is the current solution the best fit for mass open source contribution? 6) What are the most appropriate tools for energy-efficient cloud orchestration implementation? We plan to implement advanced features in the Wisdom Box and Big Data modules and to test the prototype iteratively. Other than smartphones, we would like to extend this framework for testing with different types of IoT devices.

To put the user in the loop and to make them use the smartphones energy efficiently, we suggest a technique of smart notifications. For example, temporary overlay GUI elements, informative push notification icons, dynamic contextual changes in the notification icons can be exploited to avoid opening the energy optimizing application every time.

6. Conclusions and Future Work

In this paper, we proposed a novel cloud orchestration framework for improving energy efficiency for smartphones in the IoT. The major advantage of this cloud orchestration is that it supports dynamic workflow and configuration of different processes and services. We have described the components of the orchestration and their interactions. Our architecture design is flexible so that new components and advanced functions can be added to the system easily. The *big data*, *knowledge graph*, *control box* are openly accessible, so that both single user and mass collaborators can participate and add new methods to the system. We have conducted experiments using real resource utilization traces collected by four mobile users in a three month period. The results demonstrated that our profiler can successfully characterize the energy consumption of different applications and identify the most power consumption applications. It can also give feedbacks to the energy profiler to reduce energy consumption in data logging. The amount of data logs can be reduced significantly through learning the key energy indicators and application characteristics. This iterative learning process can progressively reduce the communication and computation overhead in energy profiling. There is a great potential that the big data knowledge can be used for solving other problems as well, such as energy bug detection, etc.

In the future, we would like to investigate advanced data mining and data filtering techniques to further reduce energy consumption in energy profiling. We will explore how data logging and communication can be optimized considering the application characteristics, usage pattern and operation context.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645-1660, September 2013.

[2] J. Chase, The Evolution of the Internet of Things. Dallas, TX: Texas Instruments Incorporated, 2013.

[3] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," Journal of Big Data, vol. 2, no. 1, February 2015.

[4] International Energy Agency, More Data, Less Energy - Making Network Standby More Efficient in Billions of Connected Devices, ©OECD/IEA, 2014 Licence:`https://www.iea.org/t&c/termsandconditions/`

[5] N. Vallina Rodriguez and J. Crowcroft, "Energy Management Techniques in Modern Mobile Handsets," IEEE Communications Surveys & Tutorials, vol. 15, no. 1, pp. 179–198, 2013.

[6] M. Altamimi, A. Abdrabou, K. Naik, and A. Nayak, "Energy Cost Models of Smartphones for Task Offloading to the Cloud," IEEE Transactions on Emerging Topics in Computing, vol. 3, no. 3, pp. 384–398, Sep. 2015.

[7] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[8] A. Tzanakaki, M. P. Anastasopoulos, S. Peng, B. Rofoee, Y. Yan, D. Simeonidou, G. Landi, G. Bernini, N. Ciulli, J. F. Riera, et al., "A converged network architecture for energy efficient mobile cloud computing," International Conference on Optical Network Design and Modeling, 19 May 2014 - 22 May 2014. Stockholm, Sweden, pp. 120-125.

[9] T. K. Kundu and K. Paul, "Improving Android Performance and Energy Efficiency," International Conference on VLSI Design (VLSI Design), 2-7 January, Chennai, India, pp. 256-261.

[10] P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, and Y. Qu, "eTime: energy-efficient transmission between cloud and mobile devices," IEEE International Conference on Computer Communications (INFOCOM), 14-19 April 2013, Turin, Italy, pp. 195-199.

[11] S. Nirjon, A. Nicoara, C.-H. Hsu, J. Singh, and J. Stankovic, "Multinets: Policy oriented real-time switching of wireless interfaces on mobile devices," IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 16-19 April 2012, Beijing, China, pp. 251–260.

[12] J. Tang, Z. Zhou, J. Niu, and Q. Wang, "An energy efficient hierarchical clustering index tree for facilitating time-correlated region queries in the Internet of Things," Journal of Network and Computer Applications, vol. 40, pp. 1–11, April 2014.

[13] A. Venčkauskas, N. Jusas, E. Kazanavičius, and V. Štuikys, "An Energy Efficient Protocol For The Internet Of Things," Journal of Electrical Engineering, vol. 66, no. 1, pp. 47–52, 2015.

[14] T. Zhang, X. Zhang, F. Liu, H. Leng, Q. Yu, and G. Liang, “eTrain: Making Wasted Energy Useful by Utilizing Heartbeats for Mobile Data Transmissions,” IEEE International Conference on Distributed Computing Systems (ICDCS), Columbus, Ohio, USA, 29 June - 2 July, 2015.
[15] Power Profiles for Android, https://source.android.com/devices/tech/power/index.html, July 2015.
[16] T. Dao, I. Singh, H. Madhyastha, S. V. Krishnamurthy, G. Cao, and P. Mohapatra, “TIDE: A user-centric tool for identifying energy hungry applications on smartphones,” IEEE International Conference on Distributed Computing Systems (ICDCS), Columbus, Ohio, USA, 29 June - 2 July, 2015.
[17] A. J. Oliner, A. P. Iyer, I. Stoica, E. Lagerspetz, and S. Tarkoma, “Carat: collaborative energy diagnosis for mobile devices,” ACM SenSys, November 11-15 2013, Rome, Italy, pp. 1-13.
[18] B. Steigerwald and A. Agrawal, “Developing green software,” Intel White Paper, vol. 9, 2011.
[19] A. Pathak, Y. C. Hu, and M. Zhang, “Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof,” ACM European Conference on Computer Systems (EuroSys), 10-13 April 2012, Bern Switzerland, pp. 29-42.
[20] A. Nettstrater, M. Bauer, M. Boussard, N. Bui, F. Carrez, et al., “Internet of Things-Architecture IoT-A Deliverable D1. 3–Updated reference model for IoT v1.5,” http://www.iot-a.eu/arm/d1.3, 2012.
[21] A. González García, M. Alvarez Alvarez, J. Pascual Espada, O. Sanjuán Martínez, J. M. Cueva Lovelle, and C. Pelayo G-Bustelo, “Introduction to Devices Orchestration in Internet of Things Using SBPMN,” International Journal of Interactive Multimedia and Artificial Intelligence, vol. 1, no. 4, p. 16-22, 2011.
[22] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” Computer Networks, vol. 54, no. 15, pp. 2787-2805, October 2010.
[23] H. Suo, J. Wan, C. Zou, and J. Liu, “Security in the Internet of Things: A Review,” International Conference on Computer Science and Electronics Engineering (ICCSEE), vol.3, 23-25 March 2012, pp.648-651.
[24] A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, and Y.-M. Wang, “Fine-grained power modeling for smartphones using system call tracing,” in Proceedings of the sixth conference on Computer systems, 2011, pp. 153-168.
[25] J. Paek, J. Kim, and R. Govindan, “Energy-efficient rate-adaptive GPS-based positioning for smartphones,” in Proceedings of the 8th international conference on Mobile systems, applications, and services, 2010, pp. 299–314.
[26] Qualcomm Technologies, Inc. "Trepn Profiler" `https://developer.qualcomm.com/mobile-development/increase-app-performance/trepn-profiler`, April 15, 2015.
[27] Copyright (c) 2015, Facebook, Inc. All rights reserved. `https://facebook.github.io/graphql`, October 2015.
[28] "Understanding the meteor internals" `https://meteorhacks.com/understanding-meteor-internals`, October 2015.
[29] "Firebase" `https://www.firebase.com/` , October 2015.

[30] "Scikit Learn" http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html November 22 2015.
[31] Jeff Leek, "The Elements of Data Analytic Style" Feb 2014.