RESEARCH ARTICLE OPEN ACCESS

Techniques for Extending Computation and Resources in Mobile Devices

Mr.S.Jagadeesan,M.Sc, MCA., M.Phil., ME[CSE].,
(Assistant professor, Department of Computer Application, Nandha Engineering College/Anna University, Erode-52)
G.Nandhini, MCA.,
(Department of Computer Application, Nandha Engineering College/Anna University, Erode-52)

----------**********----------

Abstract:

The project contains options to synchronize the offline folder contents to the web. The word documents, excel worksheets and other files can be selected and uploaded to the web application. In addition, they can be converted into bytes and saved in the database provided in the web server location. In any place, the administrator can view the contents in the server, select a file and download to the system wherever the administrator is working. the files can be uploaded from the web page also. The windows application can be used to download the content from the web site to the local machine. Thus the application synchronizes the web content and windows content. The application acts like content storage as well as document version manager. Like wise, all the documents are stored in database also with the name, the system from which it is uploaded, the data and time of uploading as well as the original file copy is also kept in the server. The files uploaded from the system in remote places other than the local office if presents in the web site, then the administrator can be intimated such that some files present in the web site is not downloaded to the local system. This helps the administrator to synchronize the document efficiently.

Keywords **— Mobile ad hoc networks (MANETs), certificate revocation, security and threshold.**

----------**********----------

I. INTRODUCTION

Cloud computing is internet-based computing in which large groups of remote servers are networked to allow sharing of data-processing tasks, centralized data storage, and online access to computer services or resources. Clouds can be classified as public, private or hybrid. Cloud computing is a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle Applications. The main enabling technology for cloud computing is virtualization. Virtualization

software allows a physical computing device to be electronically separated into one or more "virtual" devices, each of which can be easily used and managed to perform computing tasks. Cloud computing adopts concepts from Service oriented Architecture (SOA) that can help the user break these problems into services that can be integrated to provide a solution. Cloud computing provides all of its resources as services, and makes use of the well-established standards and best practices gained in the domain of SOA to allow global and easy access to cloud services in a standardized way.

Cloud computing is a kind of grid computing; it has evolved by addressing the QoS (quality of service) and reliability problems. Cloud computing provides the tools and technologies to build data/compute intensive parallel applications with much more affordable prices compared to traditional parallel computing techniques. Resource-intensive tasks can be relegated to the cloud such that mobile device constraints are mitigated. Therefore, it enables the

use of services for which mobile constraints act as barriers to entry. Mobile devices are used for functionality which goes far beyond traditional calls and messaging. Nowadays devices are used for music, gaming, photos and videos, as well as managing health, fitness and finances. However – for any device whether cheap or high-end - factors such as weight, size and battery type negatively affect resources like the processor, memory, battery life and disk capacity.

II. RELATED WORK AND MOTIVATION

The benefits of MCC can be reaped by anyone who makes active use of their mobile phone, whose usage goes beyond simple texting and calls, and who regularly accesses the internet from their devices. These benefits include easier access to resources, lower processing power consumption, longer battery life, as well as lower bandwidth requirements, datausage, data charges and latency. A possible limitation would be that for widespread adoption of MCC to occur, a change in culture and mindset is required. This is because MCC is a new concept, and some people might find this hard to adopt. Another limitation is that trust, security and privacy concerns act as a barrier to adoption. There is also the possibility that the results are not significant enough to trigger a change in culture.
Manual verification of both Remote server and local system document contents. Documents are not maintained chronological in both Remote server and local system. Documents worked out in both local systems and remote systems cannot be maintained. Time consumption is more in document version verification. To avoid the drawbacks in the existing system and reduce the time consumption, a system is required such that if the application is doing synchronization of both local system documents and web site document contents, then

the documents are maintained effectively and at the same time, then can be referenced in future easily. Hence the new system is being proposed. The new system reduces the time consumption in document verification between document uploaded in web site as well as in local systems. The advantages are listed below: Automatic verification and synchronization of both mail server and local system document contents. Documents can be uploaded from the windows application (for local system) as well as in web pages (for remote systems). Documents are maintained chronological in both mail server and local system. Documents worked out in both local systems and remote systems are easily maintained. Time consumption is less in document version verification. Currently, smart devices cannot be exploited to their full potential due to issues, such as short battery life, low process- ing power and low connectivity, even though device memory, storage, CPU, screen size, sensing technology, etc., has im- proved [8]. However, from a user's point of view, a device can never have enough battery life, be too small or too light [9]. Enhancing mobile devices with computing capabilities via the cloud means that devices can support applications with requirements greater than their capabilities, as extra resources are available via the cloud. Devices can therefore be designed to be simpler and less powerful than the applications they will be running [6]. MCC is a concept which aims to mitigate these limitations by extending the capabilities of smart devices by employing cloud services, as required [5]. Resource-intensive tasks can be relegated to the cloud or other resources such that the mobile device constraints are mitigated. A range of connections can be used, such as wireless (e.g. WIFI and Bluetooth) and mobile broadband [10]. It therefore facilitates the use of services to which mobile constraints would have been a barrier to access. In MCC both the data storage and processing occur external to the mobile device, while in cloud computing it is usually only the data storage which is external [11].

A. *mobile cloud computing*

Resource-intensive tasks can be relegated to the cloud or other resources such that the mobile device constraints are mitigated. In MCC both the data storage and processing occur external to the mobile device, while in cloud computing it is usually only the data storage which is external. Several existing architectures for achieving this have been proposed. Following are some of these architectures as categorised in [7]: augmented execution, elastic parti- tioned/modularised applications, application mobility and ad- hoc mobile cloud. We add a fifth

category; cyber foraging. require the setup and deployment of expensive and specific hardware.

Augmented Execution

Processes on mobile devices are offloaded to the cloud in order to overcome smart device limitations, such as processing power, battery life and memory. In this module considered the Limited Memory-Client/ Excessive Memory-Server. The primary functionality outsourcing (intensive computation is offloaded), background augmentation (processes which do not require user interaction are offloaded), mainline augmentation (changes in the execution path can be offloaded), hardware augmentation (augmentation of underlying platform) and augmentation through multiplicity (running multiple parallel copies of an application to make optimal decisions). In this module, computationally expensive tasks are overloaded to a web service containing a clone of the device, while simple tasks are kept on the device. Once the web service in cloud task completes the execution, results are integrated back to the mobile node. Processes on mobile devices are offloaded to the cloud in order to overcome smart device limitations, such as processing power, battery life and memory [7]. Chun and Maniatis [12] define five types of augmented execution; primary functionality outsourcing (intensive com- putation is offloaded), background augmentation (processes which do not require user interaction are offloaded), mainline augmentation (changes in the execution path can be offloaded), hardware augmentation (augmentation of underlying platform) and augmentation through multiplicity (running multiple paral- lel copies of an application to make optimal decisions). Virtual Machine (VM) techniques enable this type of architecture as it supports the offloading of VM overlays by suspending these overlays, transferring them to the cloud and allowing them to be resumed. Two approaches which make use of VMs techniques to achieve augmented execution are CloneCloud [12], [13] and Satyanarayanan et al. [9]. In CloneCloud, computationally expensive tasks are overloaded to a cloud containing a clone of the device, while simple tasks are kept on the device. Once the cloud task completes execution, results are integrated back to the smart phone. Satyanarayanan et al. propose a system based on cloudlets; a cluster of computers (or a single computer) which is on standby for nearby devices and can be accessed by a low latency, high speed, wireless, local area network.

B. Elastic/Partitioned Modularized Applications

In this module, some of the processes executed in Client and remaining processes are executed from the server. The applications are partitioned into components such that these can be remotely executed individually on a cloud. With static partitioning, applications are partitioned into fixed modules during compile time or runtime. In contrast, Dynamic partitioning is the partitioning of applications at runtime, periodically (on an interval basis) or casually. Dynamic configuration based on factors such as CPU status, memory, power, bandwidth, and user preferences determines where an application is run; whether on the device or in the cloud server. Factors such as resource consumption on the mobile device and application computational requirements are dynamically factored in the partitioning algorithm. This approach is ideal for applications which have varying and unpredictable computational loads. MAUI [14] offers a combination of fine-grained control over mobile code and VM migration. Runtime profiling and historic information (e.g. latency, bandwidth, etc.) is used to period- ically determine which methods should be offloaded, if any. Zhang et al. [6], [15] propose the partitioning of applications into individually configurable elastic components; weblets. Dynamic configuration based on factors such as CPU status, memory, power, bandwidth, and user preferences determines where an application is run; whether on the device or in the cloud. Chroma [16] makes use of runtime application-specific knowledge (referred to as tactics) to determine whether to execute remotely. Historical data and resource monitoring (e.g. CPU, network, battery, etc.) are balanced against the estimated cost to predict the best action. Giurgiu et al. [17] propose a platform which automatically distributes different bundles of an application between a mobile device and the cloud. Applications are modelled as consumption graphs, and resources such as latency, data transfer and cost

are used as parameters to optimise the offloading decisions. In our tests we apply concepts of this approach whereby we model the offloading of parts of the application's storage requirements to the cloud. The difference between this and our approach is that in the former, both execution and storage are offloaded to the cloud, while in our tests focus was place on the storage aspect.

C. Ad-hoc Mobile Cloud

In this module, the given task by a mobile node, the process execution is happened in the client if the server is unavailable. An ad-hoc mobile cloud is a set of mobile devices which replace the cloud infrastructure by providing their computation resources to other devices. Mobile devices have storage, computational power, battery power and sensing capabilities, which provides an opportunity for exploiting the collective power of these devices. This is especially useful in situations where there is little or no connection to the network, limited power (short range communications consume less energy). A drawback of offloading com- putation is its dependency on the network, which means that if the connection is not available or if it is unstable then offloading and resuming execution of the application is not possible [9], [19]. Fernando, Loke and Rahayu [20] suggest the ad-hoc mobile cloud as a solution to this. An ad-hoc mobile cloud is a set of mobile devices which replace the cloud infrastructure by providing their computation resources to other devices [7]. Mobile devices have storage, computational power, battery power and sensing capabilities, which provides an opportunity for exploiting the collective power of these devices. This is especially useful in situations where there is little or no connection to the network, limited power (short range communications consume less energy), or when data charges are to be avoided [4]. It also has the advantages of distributed ownership, less hardware mainte-nance, and enhanced efficiency when accessing data residing on other devices [21]. This approach has the challenge that a collaborative environment in which spontaneous peer device discovery and network interaction must be supported. The Mo- bile Cloud Computing Framework [20] is one such approach. This framework forms a cloud of local mobile devices, thus not requiring any additional infrastructure. Huerta-Canepa and Lee [22] also propose a cloud made up of co-located mobile devices, usually having a common goal. Another framework which uses the same approach is Hyrax [21]. In this case, mobile nodes process local tasks whose results are aggregated at the mobile cloud level. MobiCloud [23] transforms tradi- tional ad-hoc networks into service clouds, whereby a device can therefore push its computing and storage services to its corresponding Service Node. This approach is modelled in our tests whereby we model the offloading of an application's storage requirements to a peer mobile device. This is similar to most current ad-hoc mobile cloud approaches, whereby tasks are distributed over several mobile peers. In our approach, we distribute data storage capabilities rather than processing tasks.

D. Application Mobility

Application mobility is based on the concept of process migration, whereby processes can be paused, transferred to a different machine and seamlessly resumed. The difference between this and Augmented Execution is that in application mobility, migration can occur between different underlying mobile architectures. In this module, the application mobility, one process is executed in client/ other process is processed in the server which is Cloud Selection Based. It is built on a distributed file system, which support the transmission of state from one site to another, and the change in state from suspend to resume or vice versa. The thin clients are that internets suspend / resume have an asynchronous dependence on the network, while thin client would traditionally be synchronously dependent on the network. The difference between this and Augmented Execution is that in application mobility, migration can occur between different underlying mobile architectures. One such implementation of this architecture is Internet Suspend/Resume (ISR) [18]. This implementation is built on VMs and a distributed file system, which support the transmission of state from one site to another, and the change in state from suspend to resume or vice versa. An advantage of ISR over other architectures, such as thin clients is that ISR have an asynchronous

dependence on the network, while thin clients would traditionally be synchronously dependent on the network. An issue with this architecture is that as VM states are being transferred over the network, this might take too long and substantial bandwidth might be consumed. Similarly to augmented execution, we decided to exclude application mobility from our tests as we wanted to focus on simple, lightweight and inexpensive solutions, while VMs can be complex, heavy and also relatively expensive.

E. Cyber Foraging

The process or the given task in a node is partially executed in an un-trusted server. It is a method whereby proximate available resources (called surrogates) are utilized for remote execution of applications. These resources are dynamically discovered and utilized even if the servers are not explicitly trusted by the user. Surrogates are based on two main premises; un-trusted and unmanaged. This reduces the total cost of ownership and maintenance and thus encourages their broad adoption. The offload tasks to a remote server are based on several factors including computation complexity, network strength, device capability and data locality. Cyber Foraging is defined by Balan et al. as: ”a mechanism to augment the computational and storage capabilities of mobile devices” [24] It is a method whereby proximate available resources (called surrogates) are utilised for remote execution of applications. These resources are dynamically discovered and utilised even if the servers are not explicitly trusted by the user. Surrogates are based on two main premises; untrusted and unmanaged. This reduces the total cost of ownership and maintenance and thus encourages their broad adoption. Scavenger [25] is one framework which employs cyber foraging to offload tasks to a remote server over WIFI (known as a surrogate). The decision to offload and to which surrogate is based on several factors including computation complexity, network strength, device capability and data locality. While our approach does not model cyber foraging sce- narios (which dynamically discovers cloud resources in the environment), it does model the use of resources located in the environment. 6) Discussion: In augmented execution, processes are of- floaded from a mobile device to the cloud (usually using VM techniques). Application mobility is similar to this, except that suspending and resuming processes can be done on different underlying mobile architectures. With elastic parti- tioned/modularised applications, applications are partitioned into modules. Partitioning can be done on a static or dynamic basis, at compile time or at runtime. Unlike augmented ex- ecution and application mobility, granularity of partitioning lies at the application level, rather than the process level. Due to this, most approaches in elastic partitioned/modularised applications require a re-writing or modification of application code, while this is not the case for approaches which use virtual machines (that is augmented execution and application mobility). In ad-hoc mobile cloud, mobile devices replace the clouinfrastructure by pooling their resources and offering these to other devices in the pool. This approach requires dy- namic discovery of resources and network interaction. This is similar in concept to cyber foraging, whereby resources in the environment are dynamically discovered and utilised. The difference between these two approaches is that in an ad-hoc mobile cloud all resources are in mobile devices, while in cyber foraging resources are cloud servers. Moreover, approaches in ad-hoc mobile cloud are geared towards sharing and collaboration of resources (computation, data, etc.) so a mobile node can both offer and consume resources. On the other hand, mobile nodes in cyber foraging are focused on offloading to an external cloud, which means that mobile nodes utilise (rather than offer) resources.

B. Cost Models and Resource Allocation In most approaches to MCC, applications (or some parts of them) are offloaded to the cloud. However, offloading compu- tation comes at a cost; when the overhead of computing of- floading is taken into consideration it might not be worthwhile to offload computation [4]. Offloading is only cost-effective if the communication effort is small when compared to the computation effort [7], [19]. Moreover, when calculating the energy efficiency of an application, one should always factor in the energy overhead for features, such as communication, privacy, security and reliability [19] as these might sometimes be

overlooked. Factors determining offloading include module execution time, resource consumption, battery, bandwidth and security [7]. In order to provide accurate information, algo- rithms used to predict the cost of offloading must be fast and work in real time using the latest information. Miettinen and Nurminen [26] and Kumar and Lu [19] have both devised a formula which evaluates the cost/benefit of offloading. Comparing Kumar and Lu's cost model to Miettinen and Nurminen, the former present a more complex formula in which a greater number of variables are modelled. For example, number of instructions, speed of instructions, watts consumed in idle state, etc. Meanwhile, Miettinen and Nurminen simply model these variables as an amount of com- putation which can be performed. In both models, transferred data is taken into consideration, however exchanged bandwidth is only considered in Kumar and Lu. All in all, we think that Kumar and Lu might provide a more realistic model of our costs, as it takes more variables into consideration rather than grouping them together, thus factoring more operations. Other approaches use historical profiling, parametric and stochastic methods [4], as well as naive Bayesian learning techniques, linear programming and partitioning algorithms to calculate the cost model. For example, in [6] data (device status, cloud status, application performance counters and user preferences) collected from both the device and the cloud is used as input for optimising algorithms to enhance configu- rations for power consumption, monetary cost, performance attributes, security and privacy. Using the above research, we have based our tests on the following criteria: CPU, Memory, Battery Usage, Data Usage and Time, as will be discussed in more detail within Section IV-B.

F. Data collection

For the purposes of testing the application with realistic data the document is collected from the user. It has the following sub modules.

Add Document / Work Book

The user can upload the document to the web using this module. During this entry, the document or workbook or any other file is selected using open file dialog control. The logged user name, the user name used for logging to the operating system, the system name and IP Address along with entry time and optional remarks are added to the database record. In addition, the document data is converted into binary data and saved in the database in addition with the file upload in the web site folder.

View Document / Work Book

The administrator views the documents status whether 'new', 'modified' or 'synchronized' using this module. The web database records are queried and filled in data table object and using data grid view control, the records are displayed. table object and using grid view control, the records are displayed. The paging option is set so that the number of records per page is controlled.

G. synchronization

View Document Status

The users can view the uploaded documents or workbook status whether the new documents are added, if any existing document / work book is modified or those documents / workbooks are synchronized is implemented in this module. The web database records are queried and filled in data table object and using grid view control, the records are displayed. The paging option is set so that the number of records per page is controlled.

System to Web Synchronize Single Document

The administrator views the new or modified documents records in the system using this module. Then the document is synchronized from system to web. The offline software folder contains 'Downloads' folder in which the document to web site is being saved.

Web to System Synchronize Single Document

The administrator views the new or modified documents records in the web site using this module. Then the document is synchronized from web site to system. The offline software folder contains 'Downloads' folder in which the document from web site is being saved.

H. web services - application

Admin Login

The administrative user is logging into the web application through this module. A number of user name and password is saved in to the table; of which any one of the user name and password is to

given to enter into the application. During login, the logged details are displayed.

Download Document

The administrator downloads the document from browser to the web using this module. In this link, all the document or workbook or any other file is selected using link button control.

Upload Document

The administrator uploads the document from browser to the web using this module. During this entry, the document or workbook or any other file is selected using file upload control. The logged user name, the user name used for logging to the operating system along with entry time and optional remarks are added to the database record. In addition, the document data is converted into binary data and saved in the database in addition with the file upload in the web site folder.

View Document Status

The administrator views the documents status whether 'new', 'modified' or 'synchronized' using this module. The web database records are queried and filled in data table object and using grid view control, the records are displayed. The paging option is set so that the number of records per page is controlled.

III. MODEL OF THE CLUSTER-BASED SCHEME

A. High-Level Architecture

1) Execution Scenarios: We designed an application to model different execution strategies which could be applied to the healthcare scenario described in Section VI. Following are these execution strategies:

• Local Datastore: A local datastore allows users to access, edit and save their data on their local mobile device. This enables offline access, and is applicable to instances where access to the cloud is limited or non-existent. For example first responders working in the field at a disaster site. In instances where a cloud is used to back up the data collected in the field, this would be synchronised once the cloud becomes available. Challenges to the adoption of cloud computing in remote areas include a lack of connectivity and adequate bandwidth [27] which makes solutions such as this ideal in these scenarios.

• Server Datastore: In this execution scenario, data is stored on a remote server and all data operations require access to this server. This

scenario can be applied for instances where medical personnel have a high-speed connection to the cloud, such as in a hospital. The use of a centralised, remote datastore facilitates the sharing of data between different medical personnel.

• Peer-to-Peer Datastore: This scenario applies to instances where access to the cloud is not available, and collab- oration is ideal. Data can be shared between different mobile devices using peer-to-peer protocols. In this way, devices can still access data found on other devices. As an example, this would be ideal for sharing information between medical personnel working in remote areas with no network connection who need to share and collaborate on patient information.

2) Scenario Test Cases: In this section we describe how the execution scenarios described in Section III-A1 were modelled in the application:

• Local Datastore (SQL): We modelled the local datastore scenario via SQLLite , an embedded SQL database.

• Local Datastore (NoSQL): The local datastore execution scenario was also modelled via Couchbase 1 Mobile, an embedded NoSQL database. This provides a comparison to the SQL database featured in the previous test case.

• Server Datastore: Couchbase Server was used to model the server datastore scenario, whereby Couchbase Server is a NoSQL distributed database.

• Peer-to-Peer Datastore: The Couchbase technology was also used to enable the peer-to-peer execution scenario.

3) High-Level Architecture Diagram: Figure 1 is a high level architecture diagram for our prototype. The main com- ponents of the system are described:

• Mobile Application: Prototype application through which tests are initiated. Android was our choice of technology for this.

• Couchbase Mobile/Couchbase Mobile Peer-to-Peer: Couchbase mobile client used as a local storage database. Can be synchronised with other Couchbase databases, whether on the cloud or in peer-to-peer mode.

• SQLLite: Local, embedded SQL database.

• Couchbase Server: Couchbase server database in the cloud.

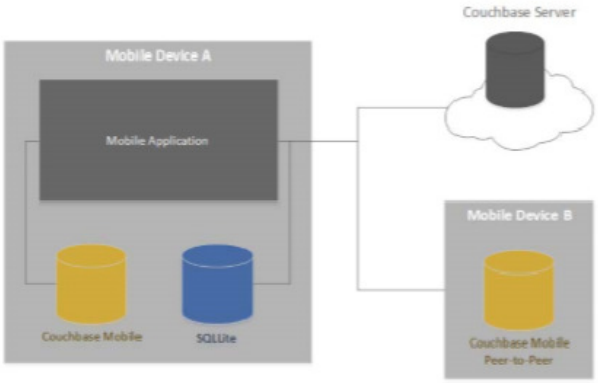

Fig. 1. High Level Architecture Diagram

B. Data Collection

The nature of healthcare data (e.g. medical records) is very confidential in nature. Therefore there are ethical, trust and privacy concerns to be addressed when using such data. For the purposes of testing our application with real- istic data while forgoing these concerns, we used sam- ples of anonymised data extracted from 6 million patient records found in the Medical Quality Improvement Consor- tium (MQIC) Database [28]. The MQIC is a big data ware- house provided by GEHealthcare , which pools anonymised, clinical data from various United States healthcare providers. Therefore, this warehouse is a store of healthcare information spanning millions of patient records representing a wide cross section of healthcare-related issues. Two datasets were used for testing. The first dataset is large in terms of number of records, but narrow (it has very few columns). The second dataset is much smaller in terms of number of records, but contains several columns. Thus, we were able to test the application using datasets with different characteristics.

1) MQIC Patient Data 100k Sample: This dataset2 is a random sample from the (approximately) six million patient records found in the MQIC database. It should not be inter- preted as statistically significant (sample is not indicative of the whole dataset), rather it is meant to represent general trends.

2) MQIC Patient Data Detailed Sample: This dataset 3 is also a random sample from the MQIC database. The dataset has 2029 records.

C. Technologies

In this section we discuss some of the alternative technolo- gies which were considered for the remote server database. More specifically, Couchbase Server is compared to Mon- goDB4, Cassandra5 and CouchDB6. We required two types of databases for our prototype; a local, embedded database and a remote, server database. For the first type, we chose two databases, SQLite and Couchbase Lite. The decision for SQLite was due to its popularity, light- weightiness and versatility; arguably it is the most widely deployed and widespread SQL database in the world7. Couch- base Lite was selected due to the features it supports; its synchronisation capabilities and its cross-platform support.

Data generated for our experiments was collected via device and application monitoring. Monitoring the device via the ADB tool generated large logs which needed to be analysed manually and from which specific units of data were extracted. Following is a breakdown of this data:

1) CPU: CPU usage was monitored via the use of ADB commands. Commands are run on a per–device (rather than a per–process) basis. They output raw statistics regarding the CPU usage of different processes and services, as a percentage of the total CPU consumption. In particular, we were interested in the CPU load, which is represented as three averages over progressively longer periods of time (one, five and fifteen minute averages). A load of 1.00 represents a CPU at full capacity, while a load greater than 1.00 represents load which is over the capacity of the CPU. Subsequently, a load less than 1.00 means that the CPU is not at full capacity. Loads are relative to the number of cores, so in a multi-processor environment dual-core environment, the 100% load would be 2.00. Therefore, lower numbers are better.

2) Memory: Memory usage was also monitored using the ADB tool. Statistics were also outputted to an external file from which they were analysed. The MemInfo command is run on a process basis. Logs list a process's current memory allocations in

kilobytes. Al- though several allocations are included (e.g. private RAM, proportional set size, etc.), we only took into consideration Proportional Set Size - Total for our results as this indicated the amount of memory which was only being used by the our application process and is not shared with any other processes. Indeed it is the amount of memory which can be reclaimed once the process is terminated.

3) Battery Usage: Similar to the previous, Battery Usage was also monitored via the ADB tool at a device level. The BatteryStats command generates raw battery usage statistics and provides a view of historical power events. However, we were mainly interested in the estimated power use and computed drain which is measured in Milli-Ampere Hour (mAh) units. This rates the amount by which a battery will discharge over a given period of time (usually that of an hour).

4) Data Usage: The Android TrafficStats class is used programmatically to provide network statistics. While this was implemented at the application level, statistics are displayed at a device level. Stopping all other processes during testing is a way of ensuring that statistics pertain only to the running application. 5) Time: Duration of tests (Time) were calculated program- matically at an application level. Using the Android System class the difference between the start date/time and the end date/time can be calculated.

RESULTS

In total, 144 tests10 were conducted; 4 execution scenarios for 4 different file sizes, where each test was carried out 3 times. Tests were carried out on 3 different devices. Due to the number of variables present in the results, these have been collated for easier analysis.

A. CPU(load)

Table I and Figure 2 represent the results for CPU tests as CPU load for different devices and execution scenarios.

TABLE I
CPU (LOAD) RESULTS

	Local Datastore (SQL)	*Local Datastore (NoSQL)*	*Server Datastore*	*Peer-to-Peer Datastore*
Device A	13.52	11.16	12.61	11.59
Device B	8.89	9.36	9.20	9.42
Device C	1.30	1.02	1.23	1.07

B. Memory Allocation (MB)

Table II and Figure 3 show results for memory allocation tests, given in MB.

TABLE II
MEMORY ALLOCATION (MB) RESULTS

	Local Datastore (SQL)	*Local Datastore (NoSQL)*	*Server Datastore*	*Peer-to-Peer Datastore*
Device A	30	80	69	60
Device B	73	59	136	140
Device C	36	41	129	104

[10]For further details go to: https://www.dropbox.com/sh/sdcwaq3z5x77t89s/AABULcIX1dwei6-BoJzs6kwra?dl=0

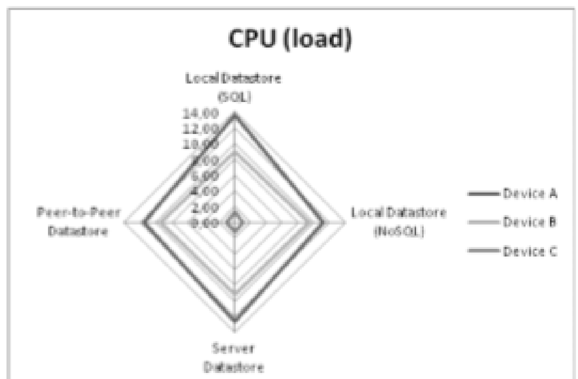

Fig. 2. CPU - Execution Scenarios - Devices Chart

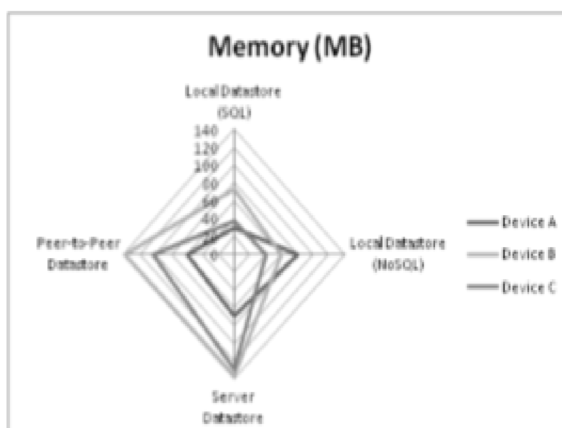

Fig. 3. Memory - Execution Scenarios - Devices Chart

CONCLUSIONS

Through this project, the data management process becomes easy. All the day-to-day activities are assigned to them through browser interface. The administrator can view the contents in the server, select a file and download to the system wherever the administrator is working. Like wise, the files can be uploaded from the web page also very easy manner. The new system eliminates the difficulties in the existing system. It is developed in a user-friendly manner. The system is very fast and any transaction can be viewed or retaken at any level. Error messages are given at each level of input of individual stages. Many of the time-tested practices and technologies for managing trust relationships in traditional enterprise IT environments can be extended to work effectively in both private and public clouds.

REFERENCES

[1] J. Pramis, "Number of mobile phones to exceed world population by 2014," 2013.

[2] S. Hossain, "What is mobile cloud computing?" 2013.

[3] D. J. Frank, W. Haensch, G. Shahidi, and O. H. Dokumaci, "Optimizing cmos technology for maximum performance," IBM journal of research and development, vol. 50, no. 4.5, pp. 419–431, 2006.

[4] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," Future Generation Computer Systems, vol. 29, no. 1, pp. 84–106, 2013.

[5] M. Shiraz, A. Gani, R. H. Khokhar, and R. Buyya, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing," Communications Surveys & Tutorials, IEEE, vol. 15, no. 3, pp. 1294–1313, 2013.

[6] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," Mobile Networks and Applica- tions, vol. 16, no. 3, pp. 270–284, 2011.

[7] D. Kovachev, Y. Cao, and R. Klamma, "Mobile cloud computing: a comparison of application models," arXiv preprint arXiv:1107.4940, 2011.

[8] H. Qi and A. Gani, "Research on mobile cloud computing: Review, trend and perspectives," in Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on. IEEE, 2012, pp. 195–202.

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," Pervasive Computing, IEEE, vol. 8, no. 4, pp. 14–23, 2009.

[10] S. Dihal, H. Bouwman, M. de Reuver, M. Warnier, and C. Carlsson, "Mobile cloud computing: state of the art and outlook," Info, vol. 15, no. 1, pp. 4–16, 2013.

[11] W. Jia, H. Zhu, Z. Cao, L. Wei, and X. Lin, "Sdsm: a secure data service mechanism in mobile cloud computing," in Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on. IEEE, 2011, pp. 1060–1065.

[12] B.-G. Chun and P. Maniatis, "Augmented smartphone applications th
rough clone cloud execution." in HotOS, vol. 9, 2009, pp. 8–11.