Design of Elastic Application for Seamless Cloud Computing

Kamal Bunkar, Prakash Choudhary, Neha Mahala, P K Bhagat

Abstract: ***Mobile cloud computing is an emerging technology. Unlike previous definition of mobile cloud computing, mobile node now has become a part of deployment of cloud application. But present issues in the currently existing network infrastructure create problems in connecting mobile devices to cloud hosting server. We present a protocol for elastic applications which uses application partitioning technique to facilitate the mobile cloud computing in disconnected network. Application partitioning is a technique that aims to split a complex application into components called as weblets. An elastic application consists of one or more weblets, each of which can be launched on a device if wireless network is not responding. The proper organization of weblets with client side data storage technique drives mobile cloud computing into a new era of wireless cloud computing. The initial phase of our work approach proposes algorithms for application partitioning and in the latter part we have explained protocol for elastic application through client side data storage mechanism. We present results of experimentation done by changing different cost parameters.***

Index Terms: ***About four key words or phrases in alphabetical order, separated by commas.***

I. INTRODUCTION

Cloud computing is a new technology which is being used from last few years. A lot of research is being carried out in this technology domain. Nowadays, much attention has been given towards this technology as many companies and the media have taken interest in it due to the opportunities offered. Cloud is a computing model which provides web-based software, middleware and computing resources as and when required.

There is no clear definition of the term "Cloud Computing" and there have been speculations regarding its origin. Some experts define cloud computing as - "cloud computing as a style of computing where scalable and elastic IT-related capabilities are provided "as a service" to customers using Internet technologies." (Gartner, 2012)[3].

The main aim of Cloud Computing is to share resources amongst the cloud service consumers, cloud partners, and cloud vendors in the cloud value chain. Various types of cloud offerings are the result of resource sharing at various levels such as infrastructure cloud (e.g. hardware, IT infrastructure management), software cloud (e.g. SaaS focusing on middleware as a service, or traditional CRM as a service), application cloud (e.g. Application as a Service, UML modeling tools as a service, social network as a service), and business cloud (e.g. business process as a service) [1].

Mobile cloud computing (MCC) will help to overcome limitations of mobile devices, in particular of the processing power and data storage. Mobile cloud computing is the usage of cloud computing technology in combination with mobile devices. Cloud computing exists when tasks and data are kept on the internet rather than on individual devices, providing on-demand access. Mobile cloud computing technology can give mobile device users a number of advantages.

Definitions of mobile cloud computing can be divided into two classes. The first refers to carrying out data storage and processing outside mobile devices. Mobile devices are simply terminals in cloud computing, only intended to provide a more convenient way of accessing services in the cloud. The second class of definitions refers to computing where data storage and processing are also carried out on mobile devices.

Mobile cloud computing poses challenges due to the intrinsic nature and constraints of wireless networks and devices. Mobile devices are generally less powerful and use batteries, whose capacity is fundamentally limiting. Another challenge is limited memory and less storage capacity. A key challenge for mobile cloud computing is network availability and intermittency. As mobile network infrastructures continuously improve, mobile users are becoming popular clients to consume any web resources, especially Web Services (WS). However, there are problems in connecting mobile devices to existing WS. A cloud application needs a constant connection that might prove to be an Achilles heel for the cloud computing movement. The aim of this paper is to provide seamless access to cloud computing framework in a disconnected network and increase the efficiency of mobile cloud computing. As we attempt to solve the problem mentioned above, we note that there is a high degree of certainty that mobile devices of today are not capable enough to perform heavy computational task.

There are several approaches to realize mobile cloud computing. Our approach is based on two properties. First, split the client-server web application into multiple functional units so that those functions of the server which are remote-able are duplicated on the mobile device. Then the application is executed either on cloud if the network is available or on the device if the network is not available. A web application is split into multiple functional units, called weblets [2]. We assume that implementation of weblets are high-level design consideration and application developer can determine how to implement weblets for mobile device based on partitioning algorithm.

Manuscript published on 30 October 2017.
***** Correspondence Author (s)
Kamal Bunkar*, BlueApp Software, Chindwada, India.
Prakash Choudhary, National Institute of Technology Manipur, India.
Neha Mahala, ISM Dhanbad, India.
P K Bhagat, National Institute of Technology Manipur, India. E-mail: pkbhagat22@gmail.com

In the proposed application partitioning algorithm, we have used a graph theoretic approach in which the features of the services of a web application are represented by nodes and the edges between nodes model the intercommunication between features. The goal of the partitioning algorithm is to group the application services into partitions in such a way that the resource constraints of the wireless device are satisfied, services of the application can execute separately on device in offline mode and the overall cost is minimized.

Second property is accessing web services using client side data storage. Client-side data storage is initiated by the web application and that the data is mostly user-specific [10]. In our project we used API of html5 called Web Storage which is used to store a web application for offline usage and to write data to a local database. SQLite is a relational database management system used for local storage in application software. We used a cache manager which is used to put resources (HTML, JavaScript, images, etc.), which are listed in a manifest-file, in cached memory and serves HTTP resources from local cache memory. A manifest-file of a web application contains list of all the files which are necessary for web application. The cache manager periodically checks for updated content and synchronize anything that is updated. The cache manager use scripting codes to insert data into local database and serve these data to UI.

A typical elastic application consists of following components - UI component, one or more weblets and a static XML file that contains metadata for the application called manifest, as explained in Fig. 1. The device elasticity manager (DEM) is responsible for configuring applications at launch time and making configuration changes during run time. Dynamic execution configuration of an elastic application is decided based on some cost saving objectives [6]. A mechanism is needed for efficient and intelligent dynamic execution configuration. A Naïve Bayesian Learning technique is used to find the optimal execution configuration [7].

Application partitioning can be achieved in two ways: static and dynamic [8]. Static application partitioning provides a separation of application components at design time. Dynamic application partitioning is usually done either at compile time or at run time. A substantial work has been done on compile time partitioning and tools have been developed to partition existing applications. These include JOrchestra[9] and Coign [3].

Web application services are software components that can be accessed over the internet using standard protocols and well defined interfaces such as Simple Object Access Protocol (SOAP) and REST. REST is an acronym for "REpresentative State Transfer". The key aspects of REST that make it desirable for mobile applications are - stateless, URL based, HTTP based responses etc. [5].

The thesis is organized as follows. Our proposed application partitioning algorithm, application architecture and protocol are presented in section 2. In the last part of section 2 we compare our proposed model with existing elastic application model. In section 3, we discuss the details of the implementation and results followed by conclusion in section 4.

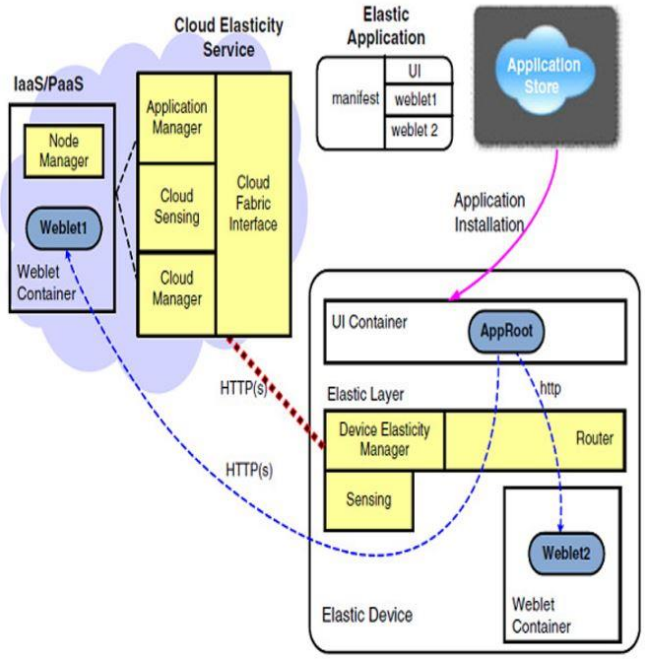

Figure 1. Reference Architecture for Elastic Application [2].

II. PROTOCOL DESIGN

A. Application Partitioning

Our proposed model has two important features. First, following the client/server split of web applications, a web application is split or partitioned so that some features of web application can run on the device and the other feature is client side data storage. In our partitioning algorithm, we used fine grained application partitioning. Fine grained partitioning is used for decomposition of a WS application at a finer level: a method or procedure level for example [8]. The work in [8] uses the above mentioned definitions to partition, whereas in addition, our proposed algorithm is based on the graph theory, mainly the dependency graphs. Web application represented as a dependency graph is used as input for partitioning algorithm. We assume that the application developers can determine the dependency graph for their web applications. Output of the partitioning algorithm is the set of weblets for every service that can be executed on the mobile device. We used two different cost functions in the partition algorithm, first is used at the design time to determine the partition of web application that is supposed to be deployed on the device. Second is used at run time to determine priority of weblets based on user pattern and runtime behaviors such as computation demand, data dependency.

The concept of application partitioning is to partition an application into remote-able weblets that can be executed independently. A component of the web application is said to be non remote-able if it satisfy any of the following criteria:

- Component directly accesses application's internal data.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

- Component accepts client requests for server storage services.
- Component accesses system services.

To sort out this problem, we use minimum device configuration as a parameter in design time objective function. A device configuration requirement includes minimum memory (MB), minimum possessor speed (GHz), minimum available storage capacity (MB), of the device. If execution demand of a dependency list does not fulfill by minimum device configuration then discard the dependency list from remote-able partition and check next dependency list. After completion of this step, remote-able partition contains only those dependency lists which are remote-able on device and executable on minimum device configuration.

B. Architecture

Our proposed architecture of the elastic application is given in Fig. 2. We adopt previous reference architecture [2] with some necessary modifications as per our requirement. Also, we have used terminology and definition from previous work [2] in our project. Because we adopt previous reference architecture in our work, both the architecture looks almost similar. But we have changed working of some modules and also added three modules in our architecture. We explain in last section of this chapter about detailed difference between both the architectures. As depicted in fig.3.1 cloud manager and device elastic manager are two important functional units of the complete architecture. On the device side, the key component is the device elasticity manager (DEM) which is responsible for configuring applications at launch time and making configuration changes during run time [2]. The configuration of an application includes: System log record, weblets table and the paths of the directory. Device manager contains a router. The router passes user requests to weblets.

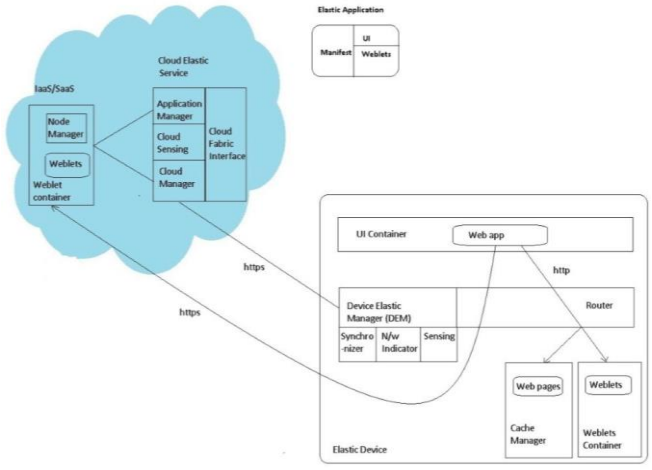

Figure 2. Proposed Architecture of Elastic web application.

When a weblet installs, the router will be aware of the new location and will pass requests from the user interface to the weblet (and passing replies back to the Interface). Each device also provides sensed data from the device such as processor type, memory status and battery state. This data is made available to the device elasticity manager and is used to determine whether weblet instance should launch on device or not. Cache manager stores web-pages in specific directory created by DEM and is responsible for displaying web-pages in offline mode. Network indicator is used to store the current status of the available network on device.

The Cloud Elasticity Manager (CEM) consists of application manager, and sensed information. The cloud manager is responsible for allocating resources on cloud node and releases the resources to underlying cloud nodes. It maintains usage information, including computations, bandwidth and storage, for the various functions running on the cloud. Also cloud manager maintains information of computation done by weblets on device including storage, computation etc. The application manager provides functions to install and maintain applications on cloud and weblets on device. To achieve our goal we added Cache manager, Network indicator and synchronizer in the reference architecture.

Cloud Elastic Manager (CEM) maintains record of every user activity. CEM uses user patterns for better prediction of weblets' priority. Device elastic manager contains a private key, which is used as a token. If network contains more than one device, private key of Device Elastic Manager uniquely identifies each DEM. CEM maintain user pattern record according to token of each user device.

C. Client Side Storage

We use client side storage for accessing web pages when network connection is not present. Client side storage is a technique to store data locally on the client device. Client side storage is a new feature of html5 which provides the possibility of caching web pages for offline use. There are two objectives provided by html5 to store data locally, they are - Local storage and Session storage. As compared to cookies, session storage provides larger storage space, better security policy etc. Local storage is used for long-term storage of data. Data remains stored even after window is closed. Data stored in session storage is lost when browser window closes. In our work we assume that the implementation of client side storage is dependent on security policy of cloud server and flexibility for storing data on client device. Html5 provides support for large database storage. With HTML 5, we can store our large data into database using SQLite database [11]. The flow diagram of elastic application is shown in Fig. 3.

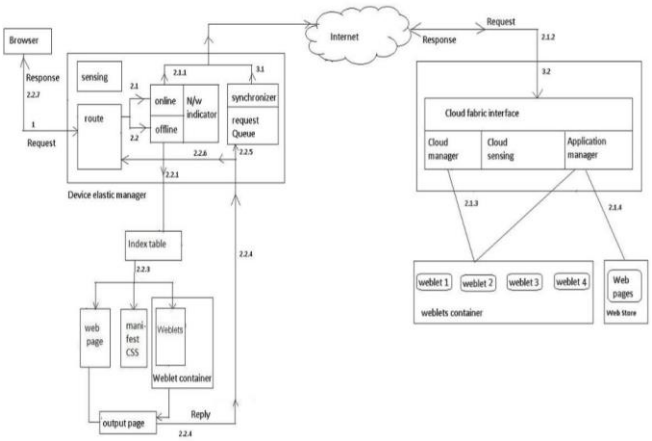

Figure 3. flow diagram of elastic application.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

D. Comparison with Reference Architecture

Xinwen Zhang [2] define elastic application as " an elastic application can augment the capabilities of a mobile device including computation power, storage, and network bandwidth, with the light of dynamic execution configuration according to device's status including CPU load, memory, battery level, network connection quality, and user preferences ". Xinwen Zhang defines reference elastic application model for connected network. So the computing task can be done partially on device and partially on cloud. Weblets changes their state according to state diagram defined in previous work. The DEM takes decision to migrate a running weblet from the device to the cloud or vice-versa. But in our proposed model, weblets migration is not allowed. If network is available then all the cloud computing tasks complete only on cloud. Otherwise task complete on device if network is not available.

In reference elastic application model, application partition is done by coarse grained application partitioning technique. Coarse grain partitioning suggests partitioning of a web service application at the package or the class level and wrap these partitions into multiple child web services [8]. In our work, we used fine grained application partitioning technique. Fine grained partitioning is for decomposition of a WS application at a finer level: a method or procedure level for example [8]. We used fine grained application partitioning technique because we believe that the design for disconnected network and the resource constrains are the main issues for handheld devices. Using fine grained partitioning technique we can better utilize the resources of devices. Another advantage of our partitioning algorithm is that we have removed duplicates from the sets of weblet. So the weblets would not consume unnecessary storage space in the device.

In reference elastic application model, DEM is responsible for determining optimal execution configuration. In previous works, optimal execution configuration computes on device whereas in our approach CEM decides whether weblets should execute on device or not.

Naïve Bayesian learning technique is used in the reference elastic application model for determining dynamic elastic execution configuration. Optimal execution configuration decides number of weblets to be executed on the device and the number of weblets executed on the cloud. Naïve Bayesian learning is a supervised learning technique. In supervised learning, classifier function trains a valid input object with training data and classify input object into any of the available classes. Naïve Bayesian technique used joint and individual probability of cost objectives [6] for classification. But in our approach we are using an objective function based on different parameters instead of Naïve Bayesian learning. Inputs for the objective function are exact values of cost objectives. Another difference is that we are using objective function two times, one at the design time and second at the run time. Both these objective functions are calculated using different parameters.

III. EXPERIMENTAL RESULTS

We have implemented the proposed application partitioning algorithm using eclipse IDE. For simulating cloud computing environment we have implemented server side code. Server code contains function like data storage on server, graph generation, mathematical computations etc. We used STAN (an eclipse plug-in) for creating dependency graph and for profiling and logging, 'perspective' plug-in, forgetting of dynamic/run-time information for every function. Both the plug-ins are available for public use.

We have tested our partition algorithm using sets of different cost objective coefficients like memory, time etc. First we set Time coefficient (TC) = 50, Memory Coefficient (MC) = 25, Inbound Coefficient (IC) = 15 and Call Coefficient (CC) = 10. Results of the test with these values are shown in Fig. 5. Then, we set Time coefficient (TC) = 25, Memory Coefficient (MC) = 50, Inbound Coefficient (IC) = 15 and Call Coefficient (CC) = 10. We then get result as shown in Fig. 6.

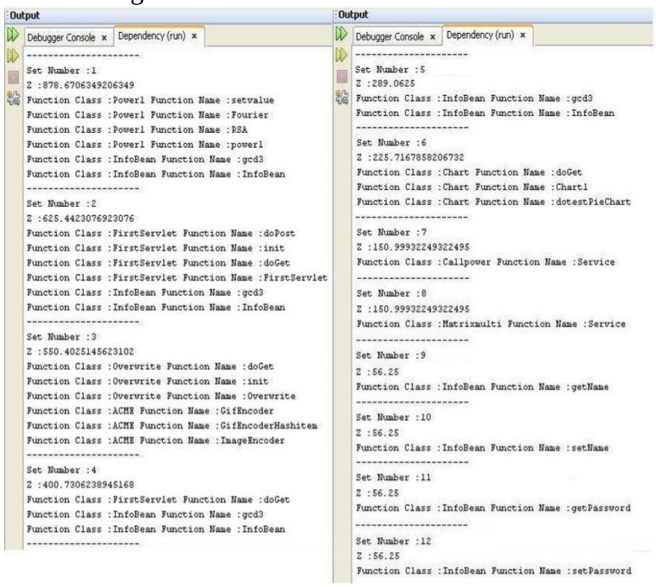

Figure 4. Dependency Lists of web application.

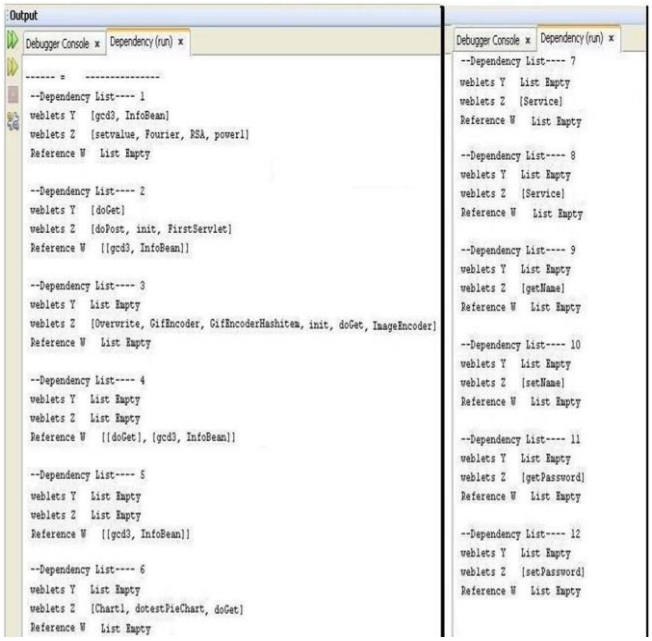

Figure 5. Output Weblets if Time Coefficient=50.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

As shown in Fig. 7 and Fig. 8, services "RecordStore" and "classifier" are the services which needed server's non-remote-able resources for execution of a service. So these two functions would not be in the dependency lists as shown in Fig. 4. From the experiment using two different configurations as mentioned above, we observe that the order of the dependency List 4 and 5 becomes interchanged. So we can say that a resource priority is an important factor to decide the order of weblets. Using this factor, we can decide which types of services user should use in offline mode, which also can satisfy device constraints. The details of execution time and memory requirements are given in Fig. 9 and Fig. 10 respectively.

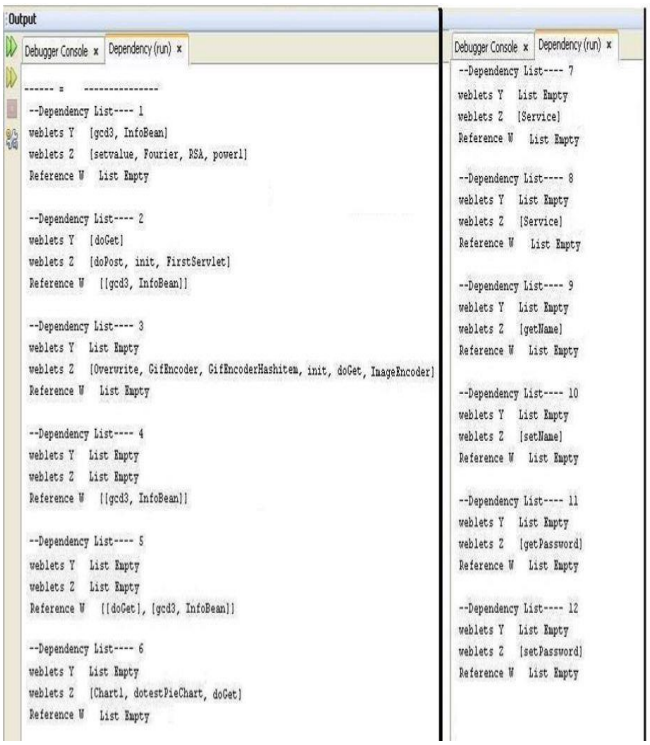

Figure 6. Output Weblets if Memory Coefficient =50.

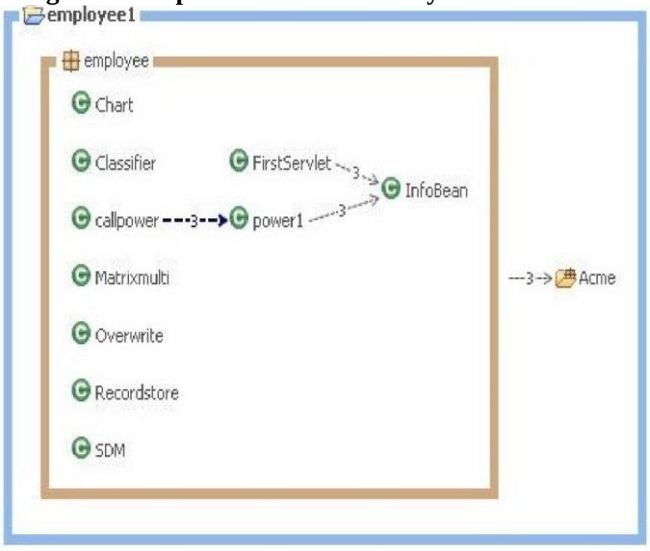

Figure 7. Dependency graph of a web application (class level view)

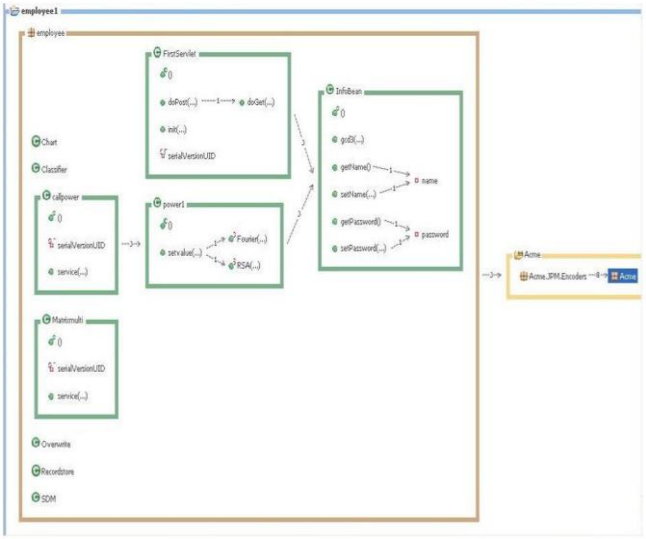

Figure 8. Dependency graph of Web application (function Level view)

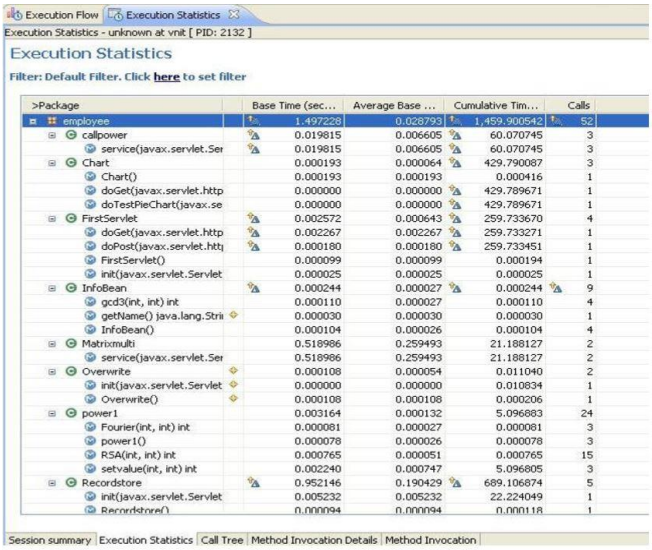

Figure 9. Execution Time of functions.

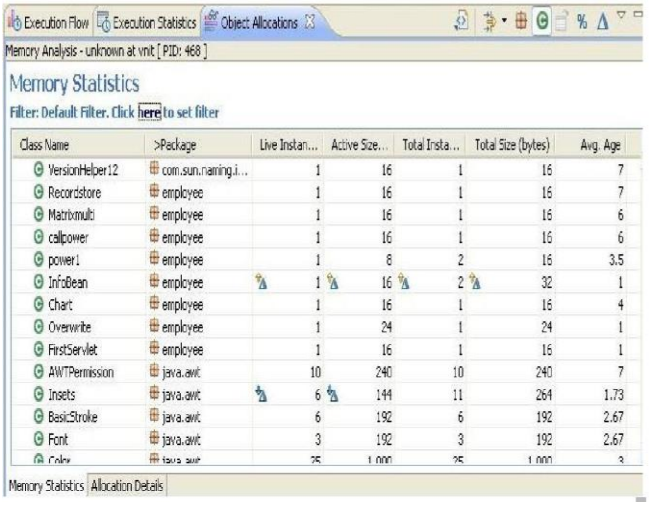

Figure 10. Memory Requirements of functions.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

IV. CONCLUSION

In this thesis, we have proposed an application partitioning technique for seamless cloud computing. The proposed partitioning algorithm is based on graph theoretic and algorithmic approach for partitioning a web application. The output of the partitioning algorithm is sets of weblets which can be executed on a device if and when the user is disconnected from the network. We have proposed a protocol to design and execute elastic application for seamless cloud computing.

We have tested our experiments with different values of cost coefficients such as memory coefficient, execution time coefficient and inbound coefficient etc. Observation shows that, as expected, the priority order of the set of weblets changes according to the values of the cost coefficient; how they change is evident from the experimentation results. From our experiment we can say that some services of the web application are remote-able and the user can use these services in offline.

REFERENCES

1. M. Armbrust et al., “Above the clouds: a berkeley view of cloud computing”, Technical report UCB/EECS-2009-28. EECS Department, University of California, Berkeley, 2009.
2. X. Zhang, A. Kunjithapatham, S. Gibbs, A. Kunjithapatham, S. Gibbs and S. Jeong, “Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing”, In Third International ICST Conference on Mobile Wireless Middleware, Operating Systems, and Applications. Mobile Networks and Applications, vol. 16, 2011, pp. 270–284.
3. G. C. Hunt and M. L. Scott, “The Coign Automatic Distributed Partitioning System”, Proceedings of the 3rd Symposium on Operating System Design and Implementation, 1999, pp. 187-200.
4. Gartner overview, [Online] Available: http://www.gartner.com/ technology/ initiatives/ cloud- computing.jsp.
5. J. H. Christensen, “Using RESTful Web-services and Cloud Computing to Create Next Generation Mobile Applications”, Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications, USA, 2009, pp. 627-634.
6. E. Walker, “The real cost of a CPU hour”, April 2009. IEEE Computer Society, 2009, pp. 35–41.
7. Y. Shi and Y. Yang, “An Algorithm for Incremental Tree-augmented Naïve Bayesian Classifier Learning”, 2010 International Conference on Artificial Intelligence and Computational Intelligence, 2010, pp. 6-10.
8. M. Asif, S. Majumdar and R. Dragnea, “Application Partitioning for Enhancing System Performance for Services Hosted on Wireless Devices”, [Online] Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.964&rep=rep1&type=pdf.
9. E. Tilevich and Y. Smaragdakis, “J-Orchestra: Automatic Java Application Partitioning”, In Proceedings of the 16th European Conference on ObjectOriented Programming, Malaga, Spain, 2002, pp. 178-204.
10. E. Tump, “The Risks of Client-Side Data Storage”. SANS Institute. [Online], Available: https://www.sans.org/reading-room/whitepapers/dlp/risks-client-side-data-storage-33669.
11. Create html5 offline web application, [Online] Available: http://www.catswhocode.com/blog/how-to-create-offline-html5-web-apps-in- 5-easy-steps.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)