OFFLOADING COMPUTATION OVER WEB TO MOBILE DEVICES

Report by

Aditya Tyagi (atyagi4@binghamton.edu)

Karan Gajwani (kgajwan1@binghamton.edu)

Chirag Onkarappa (cmahade1@binghamton.edu)

Guided by: Prof Mo Sha (msha@binghamton.edu)

Abstract:

There is great potential for boosting the performance of any computation devices by offloading computation-intensive parts of applications to the mobile device through cloud. The full realization of this potential is hindered by a mismatch between how individual different devices demand computing resources and how cloud providers offer them offloading requests usually require quick response, may be infrequent, and are subject to variable network connectivity, whereas cloud resources incur relatively long setup times are leased for long time quanta and are indifferent to network connectivity. In this paper, we present the design and implementation of a system, which bridges this gap by providing computation offloading as a service through mobile devices. It efficiently manages cloud resources for offloading requests to both improve offloading performance seen by mobile devices and reduce the monetary cost per request to the provider. It also effectively allocates and schedules offloading requests to resolve the contention for cloud resource. Moreover, it makes offloading decisions in a risk-controlled manner to overcome the uncertainties caused by variable network connectivity and program execution. We have implemented the system for Android and explored its design space through computation offloading experiments to Amazon EC2 across different applications and in various settings. If configured with the right design choices, it would have significant potential in reducing the cost of providing cloud resources to mobile devices while at the same time enabling mobile computation speedup. Also, we have designed cryptocurrency, aka QR coins. These QR coins are mined and paid to the mobile devices owner as they provide their physical resources for the computation.

Introduction:

The idea of offloading computation from web server to remote computing resources (ex: mobile devices) to improve performance and reduce energy consumption has been around for more than a decade. The usefulness of computation offloading hinges on the ability to achieve high computation speedups with small communication delays. In recent years, this idea has received more attention because of the significant rise in the sophistication of mobile applications, the availability of powerful clouds and the improved connectivity options for mobile devices. Despite great potential, a key challenge in computation offloading lies in the mismatch between how individual mobile devices demand and access computing resources and how cloud providers offer them. Offloading requests from a mobile device require quick response and may not be very frequent. Therefore, the ideal computing resources suitable for computation offloading should be immediately available upon request and be quickly released after execution. Aim of this project is to provide a platform to a client who want to take off computation from his device as he need more computation speed and memory. Therefore, we implemented a web server where client can upload his task. He uploads the file into our server. We then look for any idle registered mobile device. It is imperative that the mobile device needs to be registered and should have credentials in-order to download the task into his mobile device. For this purpose, we have developed an android app whose main function is to download the task from the server after checking the credentials of the mobile device's owner. We have also designed an algorithm through which we have designed QR coins. The payment would be done through QR coins. We have implemented RSA algorithm through which we have secured this transaction.

Categories and Subject Descriptors

Distributed Systems, Distributed applications

Keywords

Computation Offloading; Mobile Devices; Intermittent Connectivity; Resource Management

2. Background and problem statement

2.1.1 Computation Offloading

A basic computation-offloading system is composed of a client component running on the mobile device and a server component running in the cloud. The client component has three major functions. First, it monitors and predicts the network performance of the mobile device. Second, it tracks and predicts the execution requirements of mobile applications in terms of input/output data requirements and execution time on both the mobile device and the cloud. Third, using this information the client component chooses some portions of the computation to execute in the cloud so that the total execution time is minimized. The server component executes these offloaded portions immediately after receiving them and returns the results back to the client component so that the application can be resumed on the mobile device. Computation offloading trades off communication cost for computation gain. Previous systems usually assume stable network connectivity and adequate cloud computation resources. However, in mobile environments a mobile device may experience varying or even intermittent connectivity, while cloud resources may be temporarily unavailable or occupied. Thus, the communication cost may be higher, while the computation gain will be lower. Moreover, the network and execution prediction may be inaccurate, causing the performance of these systems to be degraded.

2.2 Problem Statement

The basic idea is to achieve good offloading performance at low monetary cost by sharing cloud resources among mobile devices. Specifically, our goal is to minimize the usage cost of cloud resources and provide an additional option in the form of mobile devices to our clients. As a result, the client can afford additional computational speed at affordable prices. The mobile owners will earn money in the form of QR coins. These coins are encrypted as we have implemented RSA algorithm. Therefore, all the transactions are secured without any redundancy.

2.3 Components of The System: Node.js, CSS, html, android studio.

1. **Node.js** Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code server-side. We used it to store data from the users, parse it to URL (Uniform Resource Locator) in form of text file and take the processed data back from mobile devices

2. **CSS and HTML**
   Used to create the web pages for our website
3. **Android Studio**
   To create mobile application to do the text processing. Used more than 24 opensource java libraries including Java.io.File in conjugation with SQLite to store the data on device, process it and send the processed text file to server. We also used zxing library to create QR code generator and scanner for use as cryptocurrency sender and receiver. To do so, we treated QR as bit-matrix and mapped our information in it. Given that open source server was used, we made more than 100 trials to confirm the working of app in conjugation with server.

3. Implementation

The design part of our software is quite complex. The reason is as we are working on three different domains simultaneously. We will first discuss the web development design part. The user will first load the computation task at this web address: http://offloadme.pythonanywhere.com/. The screenshot of that web page is as follows:

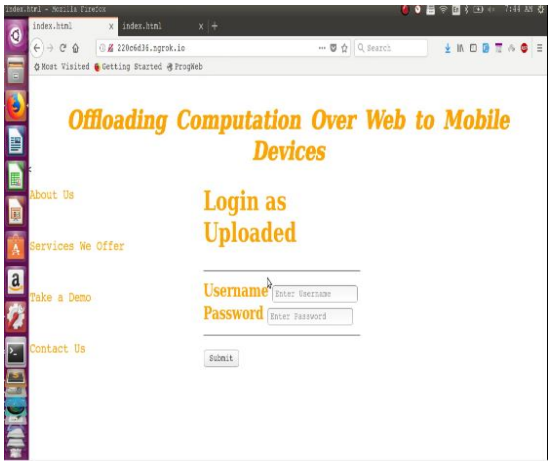

Figure 1.1

In this page, the client can login into our database. After logging into our database, the client can now upload his task. The Dashboard button is there in-order to return to the home page. This button is designed so that if user requires to go back to home page after login into our system. The user would automatically log out from our database. The Contact Us Tab provides our details and email id. If in any case, our clients' needs

our help, then it would help us to provide them adequate assistance. Also, uploaded along with is the genome text. Genome is the haploid set of chromosomes in a gamete or microorganism, or in each cell of a multicellular organism. Its sequence can help us to detect cancer in the early stage of diseases. So, we try to match the genome text with our application and further we will deal with string manipulation. The screenshots are as follows:

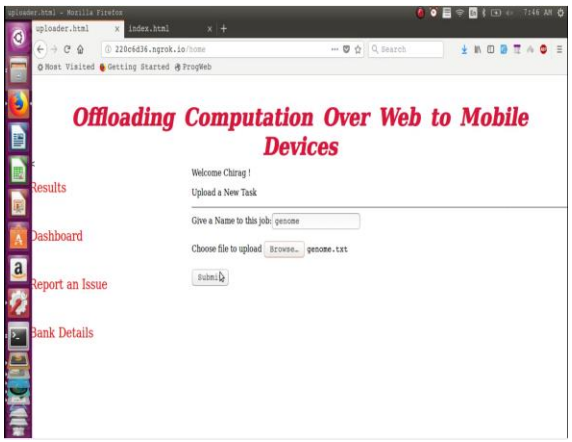

Figure 1.2

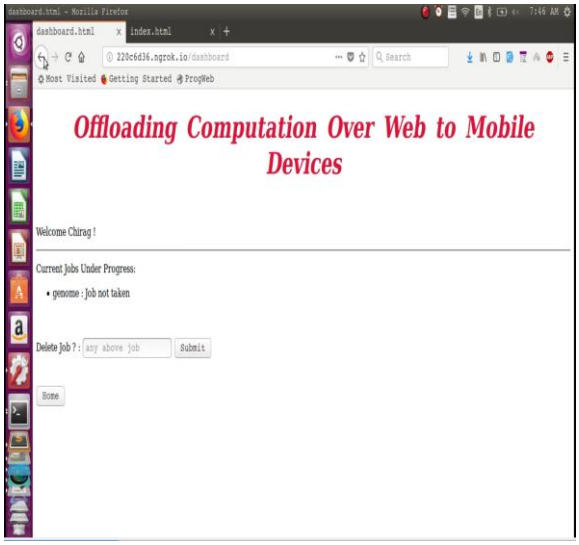

Figure 1.3

After the computation task gets completed we can get the result from the Result tab. In our genome text problem, we have required to search a specific pattern of strings. After searching that string, we need to replace that string. Our Result Tab button will show that genome text file which has that modified pattern of strings. We also have a tab button which deals with customer support." Report an Issue" tab will help us to find the bugs of our system.

After the task is uploaded, the task is computed by our android application. The name of our application is "Crytoconnect". Our app consists the functionalities of "Send Money", "Receive Money", "My Balance" and "Take Task". The functionality of take money and receive money consists the transaction of "QR coins".

These QR coins are encrypted through RSA algorithm. RSA algorithm is asymmetric cryptography algorithm. Asymmetric means that it works on two different keys i.e. Public Key and Private Key**.** As the name describes that the Public Key is given to everyone and Private key is kept private. The idea of RSA is based on the fact, that it is difficult to factorize a large integer. The public key consists of two numbers where one number is multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So, if somebody can factorize the large number, the private key is compromised. Therefore, encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024-bit keys could be broken in near future. But till now it seems to be an infeasible task. The option of "My Balance" displays the number of QR coins available in our account. The mobile device owner can also store his bank account details within this option.

4. System requirements and features

Non-Functional Requirement:

1. **Users details:** Secure access of confidential data.
2. **High Scalability:** The solution should be able to accommodate high number of customers and brokers. Both may be geographically distributed
3. **Flexibility**: Service based architecture which is versatile will be highly desirable for future extension.

Performance Requirements

1. **High Speed:** System should process voice messages in parallel for various users to give quick response then system must wait for process.
2. **Safety Requirements:** The data safety must be ensured by arranging for a secure and reliable transmission media. The source and destination information must be entered correctly to avoid any misuse or malfunctioning.

Security Requirements

Information security means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification or destruction. These fields are interrelated often and share the common goals of protecting the confidentiality, integrity and availability of information; however, there are some subtle differences between them. User password must be stored in encrypted form for the security reason. All the user details shall be accessible to only high authority persons. Access will be controlled with usernames and passwords.

Advantages & Disadvantages:

Advantages:

1. **Extensibility:** Extensibility allows new component to the system, replaces the existing once. This is done without affecting those components those are in their original place.
2. **Compatibility:** Compatibility is the measure with which user can extend the one type of application with another. The presentation tool is compatible with any type of Operating system. Because of this its usability is highly flexible.
3. **Serviceability:** In software engineering and hardware engineering, serviceability also known as supportability, is one of the aspects (from IBM's RASU (Reliability, Availability, Serviceability, and Usability). It refers to the ability of technical support personnel to install, configure, and monitor computer products, identify exceptions or faults, debug or isolate faults to root cause analysis, and provide hardware or software maintenance in pursuit of solving a problem and restoring the product into service.

Disadvantages:

1. Cannot get more than fifty hits per second.
2. App does not work with devices with API below nineteen.
3. Task assignments are last in first out
4. The time taken to execute task is directly proportional to file size.

Future Work

It is possible to incorporate iOS platform in our system. This system can be upgraded manifolds if we apply the concept of cross platform among mobile devices. This means that one major computation task can be subdivided among android app user and an iOS user. Both will complete the computation independently. We would need to build a system which would merge both computations irrespective of the platform. Sharing QR code from android to iOS can also be done to make sure the currency can be transferred cross platform

References:

[1] https://www.cc.gatech.edu/~khabak3/papers/COSMOS-MobiHoc'14.pdf

[2] https://pdfs.semanticscholar.org/1ce5/0c24b8e28c83765c7a78d810ef2e02250004.pdf

[3] http://www.buyya.com/papers/MobileCloud-IEEEComm2015.pdf

[4] https://searchsecurity.techtarget.com/definition/RSA

[5] https://sites.math.washington.edu/~morrow/336_09/papers/Yevgeny.pdf

[6] https://developer.android.com/support