Application for task assignment on mobile devices

Iury Cleveston[a], Lucas Bortolini Fronza[a]

[a]*Department of Computer Science, Kettering University*

Abstract

This paper describes an implementation of a task assignment system for mobile devices. In our solution, each user submits tasks to the server and the server distributes each task to other registered users, so that the task execution occurs in a faster way. Our API was written in PHP, using Web Services. The mobile application was written in Java for Android.

Keywords: Web Services; Task Assignment; Mobile Processing; Android Platform.

1. Introduction

Each year the number of mobile devices increase. By 2020, globally there will be 6.1 billion smartphone users [1]. Today, each person has at least one device and these devices are in idle mode most of the time. Knowing that there are computational problems that need more processing power like rendering videos, applying filters to images and mathematical calculations of various kinds. Our objective is to develop an application that will break these problems into smaller tasks and will assign them to idle devices among the network.

1.1. The problem

Even though nowadays having a large number of high-end smartphones on the market, most users have devices that are not very powerful. However, the need for greater processing power increases every day, most of this processing demand comes from a mobile game or any other task.

To overcome this need and taking advantage of idle devices, in theory would be possible to break these tasks that demand a lot of processing power into smaller tasks, then they would be assigned to idle devices. Each device would process these tasks at the same time thus decreasing the processing time by taking advantage of parallel processing.

1.2. The proposal

The main idea is to develop a mobile application that would run in the background of each smartphone. This application would submit tasks for other smartphones and would receive tasks from others to process. All this would be done without the need for user interaction.

To test our idea and to make it become a reality we proposed an Android application that manages the task allocation problem. This application is a little bit different than our main idea because the need of testing and it will be explained in Section 3.

2. Related Works

An article related to our solution was written by researchers under the title Multi-Task Assignment for CrowdSensing in Mobile Social Networks [2].

This paper describes a solution for task assignment in a MSN network, it presents two algorithms for delivering tasks, one that operates in the online scenario and one in the offline scenario. The goal of these algorithms is to provide the execution of tasks in the shortest time possible.

3. Experimental Setup

To try out our idea, we have developed an application for smartphones with Android operating system. The application is supported by a Web Application Programming Interface (API) which contains all the application logic. This makes possible to separate the logic of the view, making it easier to build a cross-platform application.

To develop the task assignment application, we implemented a simple algorithm that allows preliminary test to be carried out. In the algorithm we define, a mobile device creates a new task, which is sent to the web server. The web server is responsible for breaking down the task into smaller parts, that is, into subtasks. For this experiment we decided to break all the tasks in three subtasks to be easier to test and evaluate the results. Each of these parts then waits for devices that are available to perform processing. After each of the subtasks has been processed, the device that sent the request for the task can then retrieve the data that was processed. This device performs a post-processing with all the results and then have the task, which before would all be done by him, accomplished.

The illustration of our algorithm can be seen in the picture bellow:

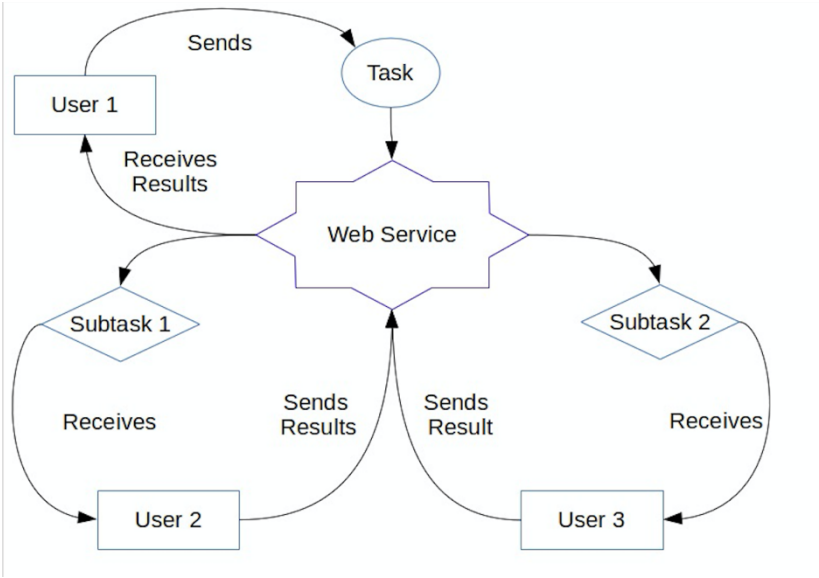

Fig 1. Application

We made the application in a way that different kind of tasks can be added to the API in the future. So, when a task is sent to the server, it also carries the type of the task. In order not to waste so much time implementing the processing part and to be able to focus on the task assignment algorithm, we implemented only one type of task. The task that we implemented was the vector sorting.

3.1. Application Programming Interface

Our *API* was developed by making use of Web Services, one of the main reasons for using this technology is that communication occurs only between devices, without interaction with humans. This provides an exchange of

information more efficient and fast, and allows access from all over the world. Thus, the use of Web Services has proved to be the best option to implement our solution.

We know that Web services can be developed by following two paradigms: *REST* and SOAP. Our *API* was written following the *REST* model, which allows responses in *JSON* format. Which is more advantageous because it introduces lower overhead and faster parser.

The *API* was implemented in Linux 4.2 environment by using the Archlinux distribution. This choice was made based on the requirements of our solution, because we know that Linux systems provide free resources and enables better control of the services used. To this end, the following packages were installed:

- PostgreSQL 9.2,
- Nginx 1.9.9,
- Redis 3.0.5,
- PHP 5.6 with pgsql, mcrypt, gd extensions,
- phpFPM 5.6;

We emphasize that the choice of the operating system does not interfere with the operation of the *API*. Our solution was written to allow the implementation of other Linux distributions and other operating system such as Windows and OSX. Even changing the cache and database management system is perfectly viable and achievable quickly.

3.1.1. Programming Technology

Our *API* was written in *PHP,* using the object orientation paradigm. The chosen design pattern was the *MVC* model, which provides the separation among models, views and controllers. This approach allows us to unlink the system modules, preventing the spread of errors and speeding development. In addition, it provides a more robust solution by allowing the implementation of new features more quickly and easily.

However, only the *PHP* does not meet all our needs, because many of the problems occur because of the particular details of the system, such as accessing the database, creating routes and creating users session. To disregard these types of problems, we use a *PHP* framework called *Yii* 2, this framework is open source and enables us to create efficient solutions, without worrying about specific details of each system.

This approach allows us to implement current resources such as Redis, which is a cache system used to store database queries and routing rules to the controllers. In addition, we can control the traffic of each user on the system.

3.1.2. Data Storage

PostgreSQL was used to store the data, which is an extremely convenient alternative, it implements sophisticated features that were only found in systems paid, like Oracle. This software works through the *SQL* language, which allows us to create and manipulate all the data.

Our database was written in *SQL* language, using the relational data model. We can see in the picture below the data structure in PostgreSQL, the following tables were used to implement our solution:

- CLIENT, used to store user information,
- TASK, used to store task information submitted by users,
- PROCESS, used to store the subtasks,
- SCORE, used to store the user's score;

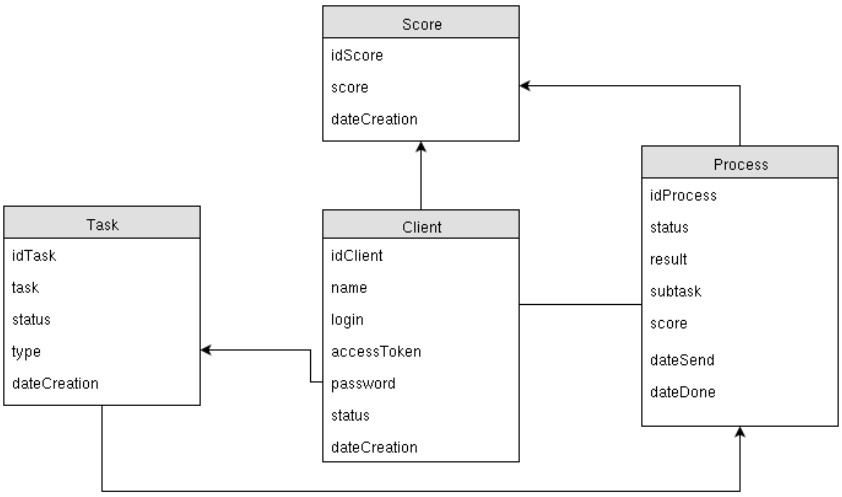

Fig 2. Database Model

In addition, the primary key of each table was designed to use BigInt fields, which provides capability to store several records, making our solution scalable. And the field that stores the task was design to use data type *JSON*, which provides functions for data manipulation, implemented by PostgreSQL itself.

That said, to store this data on the server we need a convenient way to access the Web Service, this is done via the *HTTP* protocol, this protocol allows a couple of commands to access, modification and delete data.

For our API, the following commands were used:

- GET, used to obtain data,
- POST, used to create data,
- PUT, used to update data,
- DELETE, used to delete data.

All the routes are accessed through the controllers written in *PHP*, that is, each controller implements actions, such as creation, updating and deletion of records.

The controllers developed were:

- USER, used to manage users' informations and actions,
- TASK, used to manage tasks' information,
- SUBTASK, used to manage subtasks' informations;

In addition to these controllers, the *MVC* model forces us to create models, that is, classes that represent our tables in the database. Thus, PHP classes were created for each one of our tables.

3.1.3. Routing

The use of our *API* starts by user creation, task creation, task execution and getting the results. We begin the explanation for the creation of a user, so that the mobile application needs to access the *POST /user route* and sends the following information in *JSON* format:

BODY

```
{
  "login":"iury",
  "password":"1234"
}
```

Fig 3. User Creation

When executed, the data are inserted into the CLIENT table in the server and the return of this call is the following *JSON*:

Fig 4. User Information

This return contains all the information of the created user, one field that deserves special attention is the accesstoken. This code is required to carry out all future calls to the *API*, it contains the user's access code.

If the user is created, but does not have the accesstoken, it can perform the so-called *POST /user/login*, sending the same information required in the registration process. When the user performs this action, with the correct data, the *API* provides a new accesstoken and they can perform future calls.

For the insertion of each user's tasks, the mobile application needs to access the *POST /task* route, sending the data in *JSON* format, as shown in image below:

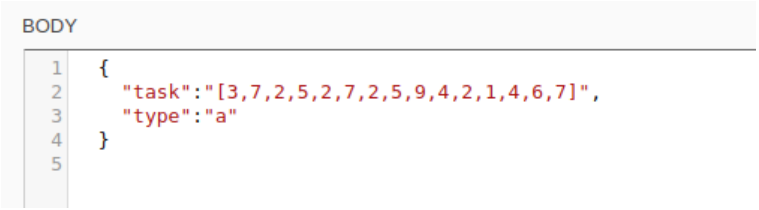

Fig 5. Task Creation

In the presented solution, we sent a disordered array. When this route is executed, the task is inserted in the *TASK* table in the server and immediately is broken into three parts and placed in the PROCESS table. In return the user receives a message TRUE or FALSE, meaning if the insertion occurred successfully or not.

When the user wants to perform tasks from other users, it simply accesses the *GET /subtask* route. This route provides a list of all available tasks for execution, as shown below:

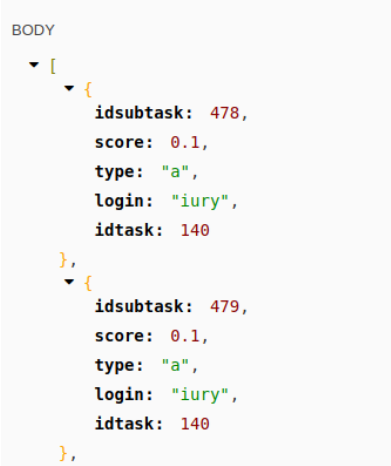

Fig 6. Subtasks List

Receiving this list the user can select the desired task for processing, the mobile application automatically retrieves the data by accessing *GET /subtask/{idsubtask} route*, this route will return all data in the sub-task, as shown in the image below:

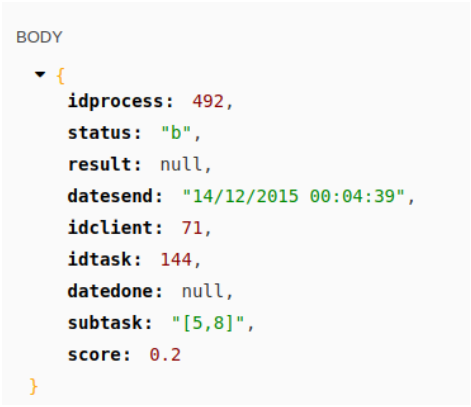

Fig 7. Task Data

The field *subtask* contains the data to be processed. With the data received, the mobile application will make the execution of this sub-task, and when it will be completed, just perform the *PUT* command with the data processed on the same route used to get the raw data. This command will update the data in the PROCESS table, as well as crediting the score for the user in the SCORE table.

Thus, when the execution of the subtask ends, the *API* will check whether the other subtasks have been completed. If the answer is affirmative, the *API* updates the status of the main task and from that point the user can get the processed data from the server through the *GET /task /{idtask}* route.

3.2. Android Application

The Android Application consists mainly of the view of the application. As said before, the Web API is responsible for the application logic. In an Android Application, an Activity is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a

map. Each activity is given a window in which to draw its user interface. The window typically fills the screen, but may be smaller than the screen and float on top of other windows [3].

3.2.1. Activities

In our application we have the following Activities:

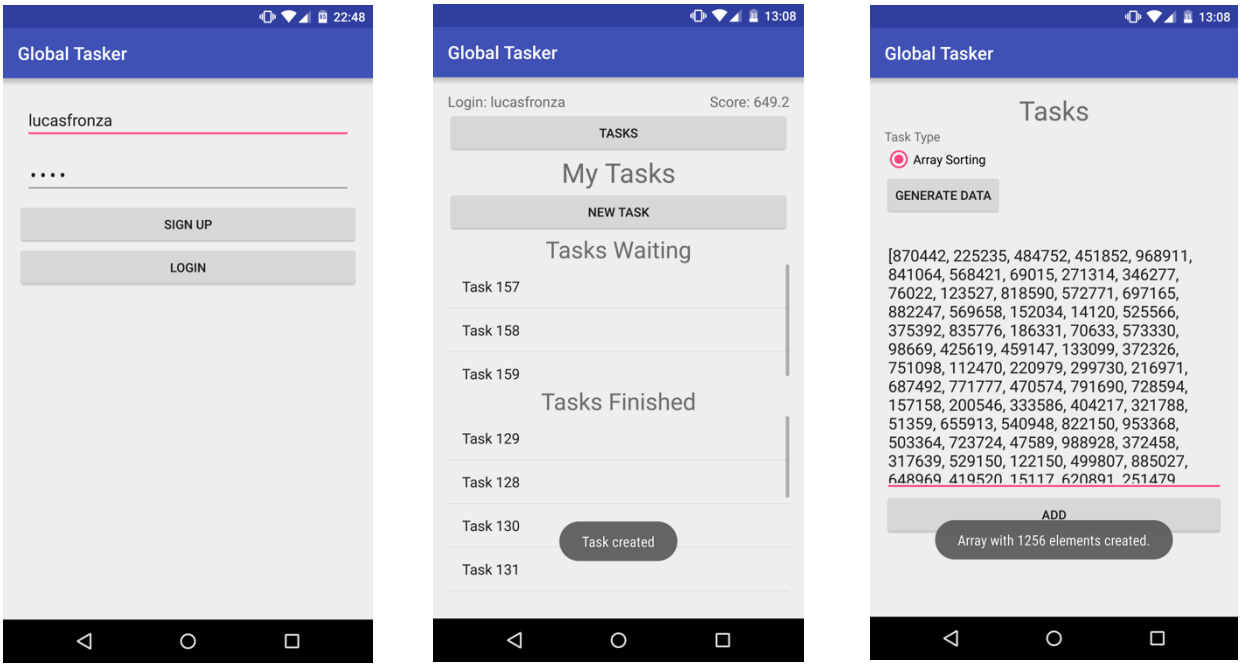

Fig. 8. (a) Main Activity (b) Tasks Activity (c) New Task Activity

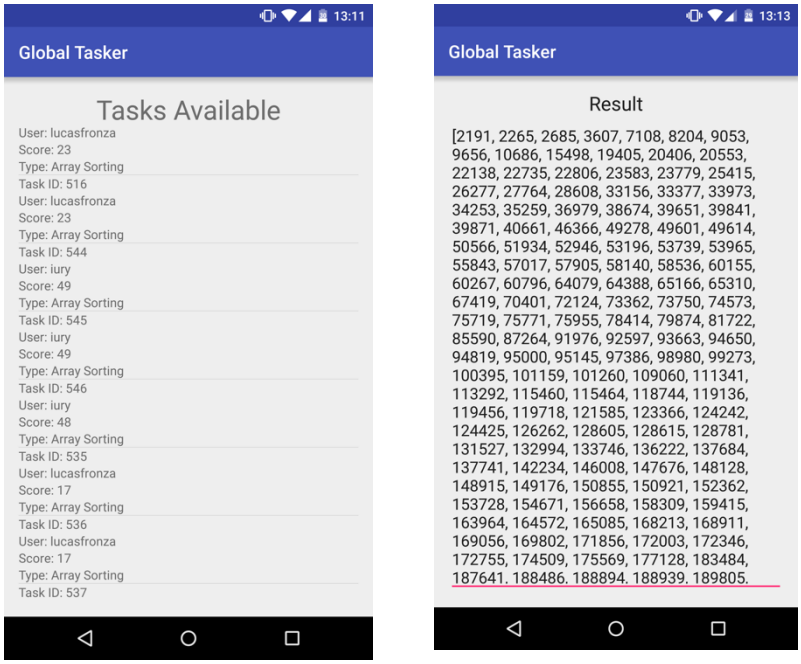

Fig. 9. (a) Subtask Activity (b) Result Activity

3.2.1.1. Main

The main activity is the first one that is called when the application is started. Our application starts with a Sign Up/Login view. To use the application, the user must have a Username and a Password. This is used to identify which task belongs to each user.

When the user clicks to sign up or to login, a request is made to the Web API, then if data match, the user successfully logs in and is redirected to the next activity.

3.2.1.2. Tasks

This activity shows which user is logged in and his score, allows the user to see which tasks he sent and are waiting to be processed and which tasks are finished. The user can then go to New Task to send a new task, go to Tasks to see a list of tasks from others that are waiting to be processed or click in a finished task to see the result of this task.

3.2.1.3. New Task

In this activity, it is possible to select the type of the task (by this time there is only one type available) and there is a button to generate random arrays to facilitate testing. By clicking in the Add button, the task is sent do the Web API to be broken in three subtasks.

3.2.1.4. Subtask

The Subtask Activity get from the server all the subtasks that are available to be processed and list this tasks to the user. Then, the user can click in a task and this task data will be retrieved from the served and processed by him, then it is sent back to the server.

3.2.1.5. Result

In the Tasks Activity, when the user clicks on a task that already finished the Result Activity is called. This activity retrieves the processed data from the server, make a post-process and show the result to the user. In this case, the post-processing that is made is the merge of the three arrays that were sorted by others.

3.2.2. Helper Classes

Besides the Activities we developed other classes to help. Mostly, we developed classes to make the requests for the API. As we worked with Get, Put and Post requests, were made one class to help with each one of this requests.

4. Results

The development of this application proved that is possible to create a task assignment application. The application successfully sent the tasks and retrieved them. It would be possible know to add more types of processing in the API.

It was also added a score for each task taking into account how much processing. This score had no essential role in this step, but, at first, would serve to encourage users to perform tasks. Each task a user finish, he wins points.

5. Conclusion

In the task assignment problem, there are some difficulties that we have to consider. One of this difficulties that exist is how to break a task in smaller task. When the task is a vector ordination it is easy to think in a way to break it to smaller tasks, but when the task is something more complex like an image processing, math calculations and simulations, it is much more difficult to create an algorithm to break this task in subtasks.

Another existing problem is the communication overhead. To make it worth to break a task and assign to others instead of processing by itself, the time that it takes to process the smaller tasks plus the time of sending and retrieving

data from the web server have to better than the time it would take to process by itself. Besides that, even if making the processing through the Web API was slightly, we need to take in account if it is worth it given that the web server has costs. So these two things, cost and processing power have to be taken into account to know a task should be assigned to others or not.

References

1. http://techcrunch.com/2015/06/02/6-1b-smartphone-users-globally-by-2020-overtaking-basic-fixed-phone-subscriptions.
2. http://cis-linux1.temple.edu/~jiewu/research/publications/Publication_files/xiaomingjun_infocom2015.pdf.
3. http://developer.android.com/guide/components/activities.html.