CSE 535: Mobile Offloading Project

Abhiram Gulaikar
Arizona State University
Tempe, US
agulani1@asu.edu

Prathmesh Sambrekar
Arizona State University
Tempe, US
psambrek@asu.edu

Vinay Pandhariwal
Arizona State University
Tempe, US
vpandha1@asu.edu

Vinayak Vasant Kalunge
Arizona State University
Tempe, US
vkalunge@asu.edu

***Abstract*—With increasing number of devices and connectivity among them the number of tasks performed by these devices is also on the rise. This has resulted in popularity of task distribution among devices. This feature of sharing the task load of one device among other available devices is called mobile offloading. In this report we outline the same process using matrix multiplication problem. We follow a master slave architecture. The master is connected to the slave via Bluetooth. The master device distributes the task of matrix multiplication among slave devices by sending parts of matrices to slave devices which satisfy a certain criteria based on battery level and current location of the slave mobile devices. Also, we verify the mobile offloading process by checking the power consumption and time taken for matrix multiplication on master and slave devices.**

I. Introduction

Using mobile devices to do calculations has many apparent benefits, including more location flexibility, improved time management, simplicity of use, and increased productivity. However, it comes at the expense of increased energy use. The battery life of these mobile devices is directly affected by the drain on this electricity. Distributing the task among numerous devices in a network is one solution to this challenge. The aggregation of resources, together with the distribution of essential activities, considerably reduces each device's power usage. To increase efficiency and performance, distributed computing is a methodology in which components of a software system are shared among several machines. On separate slave mobile apps, we conduct a component of a distributed computing operation, namely ”Matrix Multiplication,” and then recombine all of the parts to generate the resultant matrix in the master mobile application. We kept track of battery and power usage both with and without the distributed strategy.

II. Author Keywords

Mobile Offloading, Distributed Computing

III. Project Setup Permissions

Our architecture consists of one Master mobile and several slave mobiles. The configurations of devices are:

1. Master Node
Model: OnePlus 7
OS: Android 11.0

2. Slave Node
Model: OnePlus Nord
OS: Android 11.0

3. Slave Node
Model: OnePlus 8
OS: Android 11.0

4. Slave Node
Model: Realme X
OS: Android 11.0

Bluetooth sockets are used to connect slave and master nodes. The master functions as a server in this design, using the Bluetooth Server Socket, while the slaves act as clients using the Bluetooth Client Socket.

We deployed the app on master and slave nodes before the initial bluetooth connection.

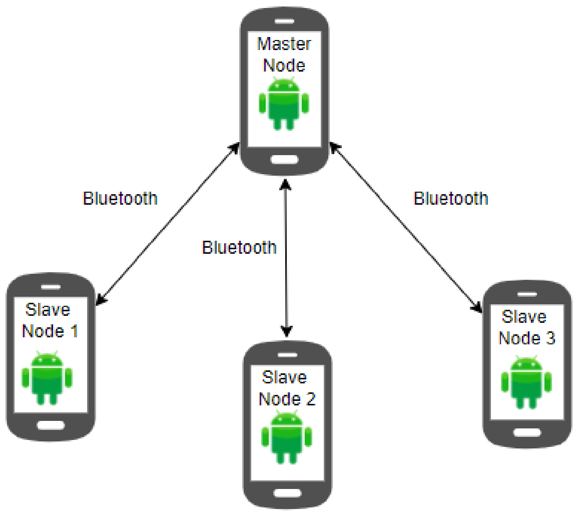


Fig. 1. Mobile Offloading Architecture

The communication channel used in the architecture is using Bluetooth. JSON data is sent through the channel for communication between the master and slave nodes.

IV. Implementation

Step 1: First of all when app opens it asks permission for location sensor. We have to allow the location permission to detect the location of the devices. We have to select the option to become Master or Slave node. The master starts bluetooth and bluetooth device discovery. The slave node opens a bluetooth connection using blutooth socket. The Master sends a bluetooth request to slave. Slave accepts the request and gets connected.

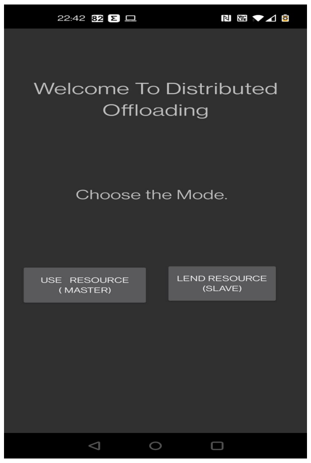


Fig. 2. Start Page

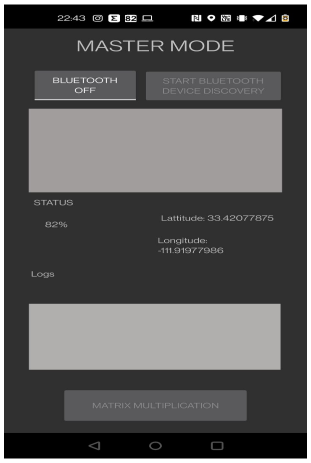


Fig. 3. Master Screen

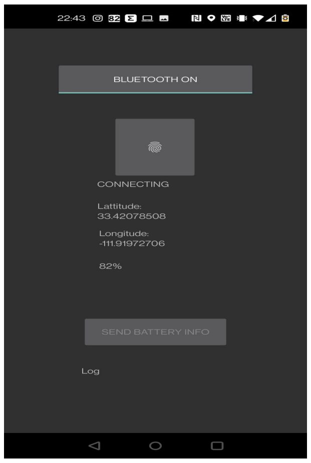
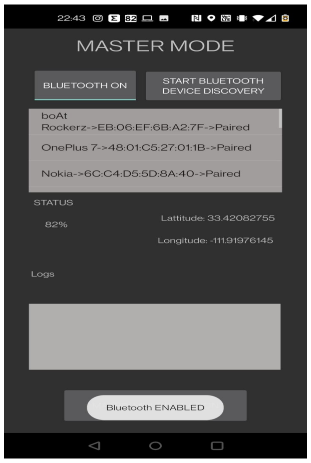


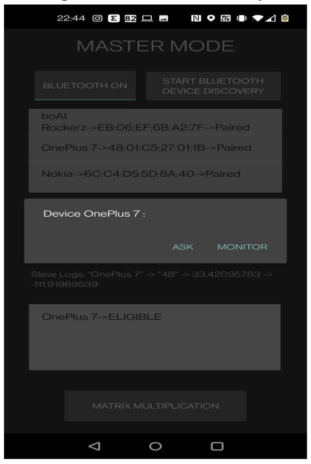
Fig. 4. Slave Screen

Step 2: After connection is setup master ask for battery and latitude longitude informaton from slave node. Master



boAt Rockerz->EB:06:EF:6B:A2:7F->Paired
OnePlus 7->48:01:C5:27:01:1B->Paired
Nokia->6C:C4:D5:5D:8A:40->Paired

Fig. 5. Device Discovery



boAt Rockerz->EB:06:EF:6B:A2:7F->Paired
OnePlus 7->48:01:C5:27:01:1B->Paired
Nokia->6C:C4:D5:5D:8A:40->Paired

Slave Logs: "OnePlus 7" -> "48" -> 33.42095783 -> -111.91969539

OnePlus 7->ELIGIBLE

Fig. 6. Ask for Permission

evaluates the battery and location of the slave and decides whether slave node is eligible for distriuted computing or not.

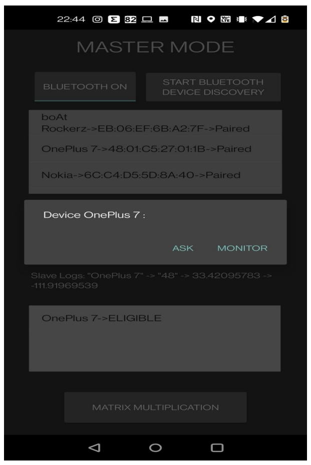
Step 3: For monitoring the slave battery and location information we select the eligible slave device and ask for Monitoring. Once the slave allows monitoring request, it sends its information every 10 seconds. This information is displayed on the Master node and saved in the batterymonitoringlog.txt file.

Step 4: For distributed computing first master ask eligible slave device for computing. Once slave device allows the request the device status changed from eligible to accepted. Similarly we have to ask for each eligible slave device.

Step 5: Now we can do the distributed matrix multiplicaton on the devices. Click on ”Matrix Multiplication” button to start the process. Now enter the 4X4 matrix into the input box(Number should be four digit with space separated). Then we press the ”Calculate” button to offload it to slaves.

Step 6: Once the slave device performs the necessary calculations, it returns the result back to the master. The result matrix is displayed on the screen.

Step 7: If a slave dies while receiving data from the master, the master’s mobile application’s continuous periodic monitoring capability will allow the master to determine if the



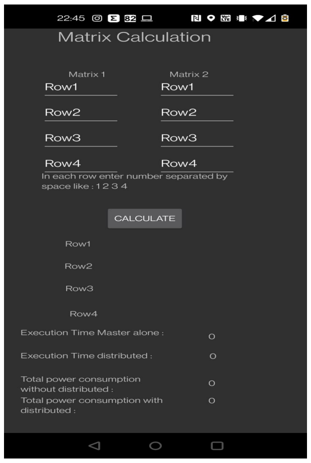
boAt Rockerz->EB:06:EF:6B:A2:7F->Paired
OnePlus 7->48:01:C5:27:01:1B->Paired
Nokia->6C:C4:D5:5D:8A:40->Paired

Device OnePlus 7 :

Slave Logs: "OnePlus 7" -> "48" -> 33.42095783 -> -111.91969539

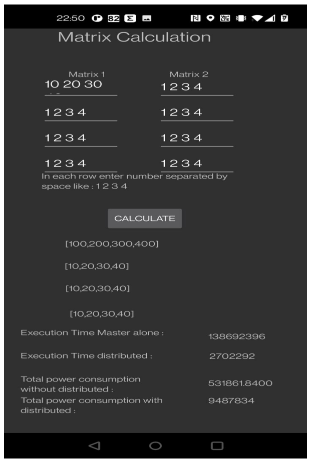
OnePlus 7->ELIGIBLE

Fig. 7. Monitoring Option



Execution Time Master alone : 0
Execution Time distributed : 0
Total power consumption without distributed : 0
Total power consumption with distributed : 0

Fig. 8. Matrix Multiplication Page



Matrix 1	Matrix 2
10 20 30	1 2 3 4
1 2 3 4	1 2 3 4
1 2 3 4	1 2 3 4
1 2 3 4	1 2 3 4

[100,200,300,400]
[10,20,30,40]
[10,20,30,40]
[10,20,30,40]

Execution Time Master alone : 138692396
Execution Time distributed : 2702292
Total power consumption without distributed : 531861.8400
Total power consumption with distributed : 9487834

Fig. 9. Time and Power Values

slave is alive or dead in a short amount of time, and to recover from this failure, we transfer the same data to an active slave.

Step 8: The execution time and battery consumption required for matrix multiplication on master device and on slave device is displayed on the master.

V. Completion of Tasks

Serial No	Task Name	Assignee
1	Mobile application that gathers and lists battery levels from phones.	Abhiram, Vinay
2	Service discovery app that uses Bluetooth to communicate requests for involvement to adjacent available phones.	Prathmesh, Vinayak
3	Dispatcher program that selects a phone that accepted the request based on a set of parameters.	Abhiram, Vinay
4	Send requests to start the battery monitoring program on the slave side based on the devices selected.	Prathmesh, Vinayak
5	A slave application that may receive a Bluetooth request from a master.	Prathmesh, Vinayak
6	If the user agrees, monitor the battery level and current position and transmit it back to the master.	Abhiram, Vinay
7	Application to initiate periodic monitoring by running a code snippet in Slave.	Prathmesh, Vinay
8	Replicating this architecture on the slave device to offload a matrix multiplication task.	Abhiram, Vinayak
9	Problem of Matrix multiplication using distributed architecture.	Prathmesh, Vinayak
10	Estimation of matrix multiplication execution time if done only by the master.	Prathmesh, Vinayak
11	Estimation of matrix multiplication execution time if spread between the master and slave with no failure.	Abhiram, Vinay
12	Estimation of the execution time of the distributed matrix multiplication between the master and slave with failure.	Abhiram, Vinay
13	Estimation of the master and slave nodes power consumption without computational distribution.	Prathmesh, Vinayak
14	Estimation of the master and slave nodes power consumption with computational distribution.	Abhiram, Vinay
15	Implementing a failure recovery algorithm that allows it to reassign a slave to another accessible slave node if one fails.	Prathmesh, Vinayak

VI. Limitation

With a grid size of 4 X 4, our application runs smoothly and can be scaled up in the future. Our program is ideally suited for Android devices running API version 29. Our app is only compatible with Bluetooth-enabled devices that have GPS capabilities. In rare circumstances, GPS may not be accurate enough to provide accurate latitude and longitude, preventing the master from connecting to slaves.

VII. Conclusion

We created a distributed computing architecture for Matrix Multiplication in this project. Our findings revealed a significant reduction in execution time when using distributed infrastructure, as well as lower power usage. We also demonstrated that our architecture can recover from failures of slave devices.

References

[1] "Android nearby api." https://developers.google.com/nearby, 2020.

[2] Bertocco, Matteo Ferraris, Franco Offelli, Carlo Parvis, Marco. (1998). A client-server architecture for distributed measurement systems. Instrumentation and Measurement, IEEE

[3] "Android broadcast system documentation." https: //developer.android.com/guide/components/ broadcasts, 2020.

[4] "Android broadcast receiver documentation." https://developer.android.com/reference/ android/content/BroadcastReceiver, 2020.

[5] : Z. Zhang, S. Li, "A Survey of Computational Offloading in Mobile Cloud Computing", IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, 2016