BOINC Computing With Android Smart Phones

Gary Tang

ICS 632

Introduction

According to Google, as of 2015 there are 1.4 billion Android users in the world. Taking advantage of the voluntary computing power of these phones can benefit many projects. The fastest supercomputer in the world, Tianhe-2 in Guangzhou, China, has 3.1 Million CPU cores. If we assume each Android phone has a single core, there are at least 1.4 billion Android CPU cores in the world.

This project created a matrix multiply application using distributed computing of Android smart phones. Participants from Virginia, and Hawaii volunteered for the project. The Berkeley Open Infrastructure for Network Computing (BOINC) was used to handle the communication between the participants' phone and the server. BOINC is an open-sourced system that supports volunteer distributed computing. It was originally developed by Space Sciences Laboratory at the University of California. The participants downloaded the BOINC client from the Google Play store. The BOINC server was implemented by me on a personal machine.

In this project, I looked at the following issues: (i)create a simple way for users to contribute to the project (ii)look at how a user's location can affect their contribution to the project (iii)study how partitioning input files can affect a project's efficiency.

Project Configuration

Setup BOINC Server

The BOINC server was deployed using a Virtual Box image provided by BOINC. The host computer was a 2012 Apple Mac Mini with Yosemite OS X, Intel Core i5 2.5 GHz processor and 4 GB of RAM. The Virtual Box image contained a Debian 64 Bit OS and was configured as follows: 1 CPU, 512 MB of RAM and 8 GB Hard drive. The server's network has a download speed of 33 Mbps and an upload speed of 6 Mbps. This Virtual Box configuration was unmodified to emphasize that the workload will be completed by the Android phones. Also, that a modest computer is able to schedule work for a BOINC project.

Set Up BOINC Project on Client app

An attempt was made to minimize the number of steps a participant would have to take in order to contribute to the project. Participants were instructed to download the BOINC app from the Google play store. Additional instructions were sent to each user. Each user was given a gmail account, BOINC server user account, and a BAM! manager account.

The BAM! manager was used to manage the BOINC project. This would allow the user to log into the BAM! manager and not have to worry about entering the project information. However, because my BOINC project was not an official project supported by Berkeley, the BAM! manager could not save my project details. As a result, instructions for a 10 step process was sent to each user. This caused the user to take additional 6 steps to complete the setup process. The additional inconvenience may have caused the delay of all users to complete the setup process. 3 out of the 9 volunteers completed the process after 12 hours the instructions were sent out. A total of 4 participants completed the experiments.

BOINC APP

Development

The matrix multiply app was developed in C and C++. In order to utilize the BOINC framework, the existing app needed to place BOINC API calls. One such API call was boinc_fopen which is similar to fopen in C. This app was then compiled with Android NDK and BOINC libraries. The Android NDK is a toolset that allows developers to implement parts of their Android Apps in native-code languages such as C and C++. The project has taken this path because this was the approach described on the BOINC website.

App Structure

The BOINC app requires several applications to function properly. The BOINC server executes two applications Matrix and Distribute. The Matrix application generates the matrix A and matrix B and separates it into tiles. At the end, task folders are generated. Inside each task folder is a name file. This name file tells the Compute app what task it is computing. The Distribute application copies the matrix A tile and the matrix B tile that a task requires into the task folder. The BOINC server requires an input and output xml file for each task. These xml files contain the preferences for the input and output files of the Compute app. The input and output files' preferences change for different matrix sizes and different matrix tile sizes. The bash scripts genInput and genOutput creates the input and output xml files.

The Compute app is the Android application that executes on the client's phones. This app along with the matrix A tile, the matrix B tile, and the name file is sent to each phone. The Compute app computes the matrix C tile and sends the results back to the BOINC server.

Experiments

Challenges

The Compute app was designed to compute matrix multiply on a matrix A and a matrix B. One of the critical parts of the design was the name file. This file tells the Compute app which task it is computing. The matrix A tiles and matrix B tiles were named according to which task it pertained to. All input files must be checked into the BOINC server's download folder in order to be sent to the client phones. The name files were all designed to have the same name but be placed in its own task folder. The BOINC server does not separate task files in its own task folder. Each input file has its own folder. Files with the same name cannot reside in the download folder. To work around this problem, for the experiments, the Compute app only computes with the first matrix A tile and the first matrix B tile. Therefore, only one name file was needed. This computation was repeated to simulate the computation of the whole matrix.

During preliminary testing, it was found that the BOINC server will halt user phone uploads when the upload directory has reach 1024 directories. Each time this has happened, the server's hard drive's capacity was over 40%. It is unknown when the BONIC server creates a directory in the upload folder. For one experiment, some directories will have one result while other directories will have two to three results. Therefore, it was assumed that one result will have one directory. Generally, all experiments from here on were designed to avoid creating 1024 directories.

Each participants had different time commitments. Also, the participants lived in two different time zones, 5 hr apart. Therefore, it was difficult to have all users participating in all of the experiments.

Smartphone Specs

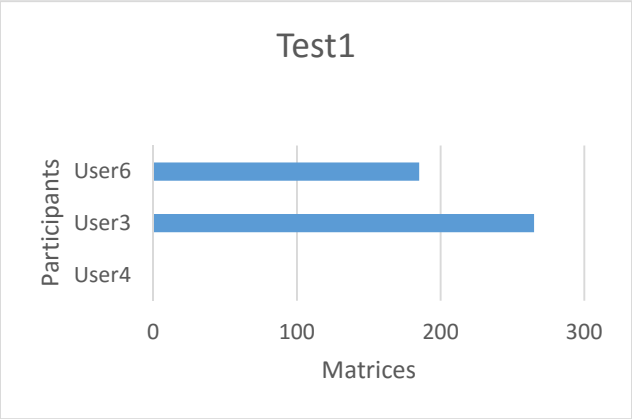
	Hawaii			Virginia
	User1	User3	User4	User5
OS	5.1.1	4.3	5.1	5.1.1
Model	LG V10	Sony Z	LTE Z812	Samsung S6
Processor (GHz)	1.8	1.5	1.2	2.1
Core	Hexa	Quad	Quad	Octa
RAM Avail. (GB)	3.7	1.8	0.9	2.6
Disk Avail. (GB)	25.5	8.0	1.7	19.2
Network	LTE	LTE	WIFI	WIFI

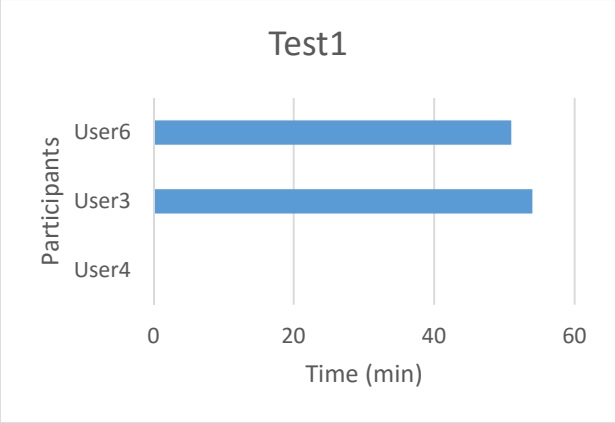
Location Based

These tests where done to examine how location can affect the amount of work a phone can complete. 450 matrices were computed. The matrix size and the tile size was 4000. Therefore,

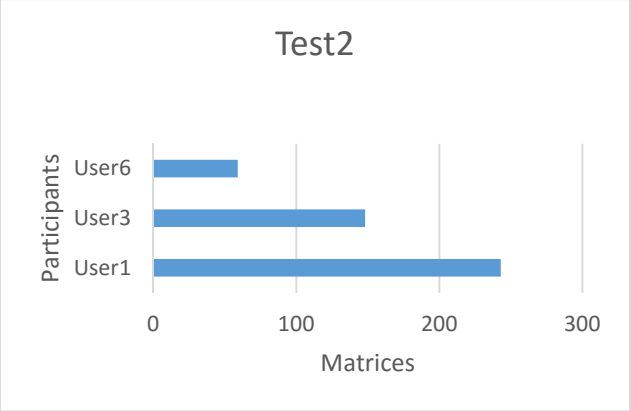
the matrix was not broken up into tiles. Matrix A was 75.6 MB and matrix B was 79.5 MB. It should be noted that user3's location was closest to the server. The BOINC server was located in Hawaii.

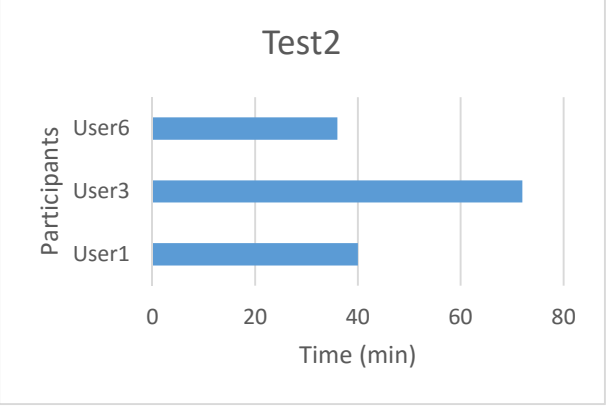
In test1, user3 calculated 30% more matrices than the 2nd most, 265 matrices. User4 did not compute any matrices. User4's phone had an error message stating there was not enough RAM. The two users that participated finished within 3 min of each other. The makespan of the experiment was 54 min.





In test2, user1 calculated 41% more matrices than the second highest user, user3. User3 made the makespan 32 min longer. The other two users finished 4 min within each other.





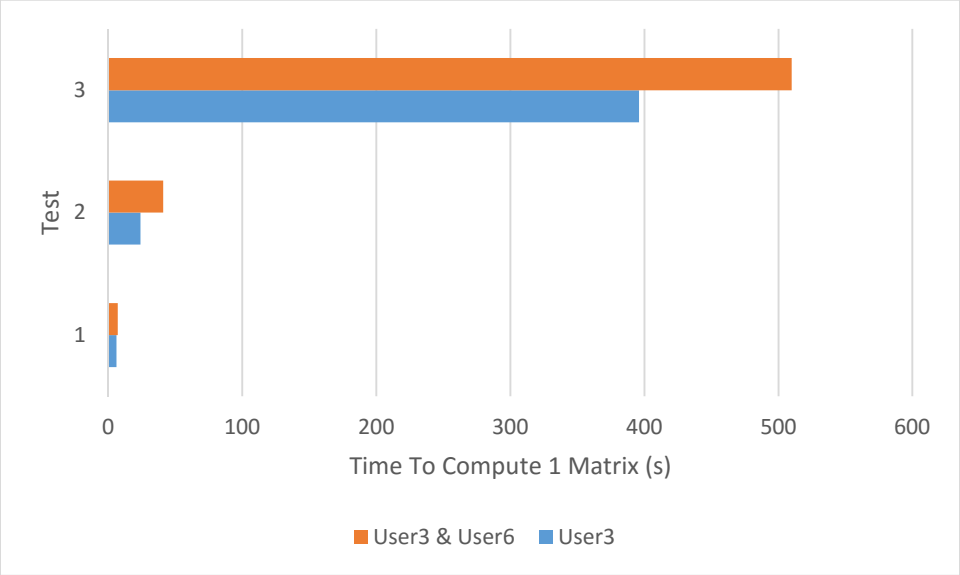
In both test, the user on the mainland (user5), calculated the least amount of matrices. A user's location did affect their phone's productivity. User5 was the furthest from the BOINC server. However, in test2, user1 completed the most matrices. Despite user3 being located closest to the BOINC server. Also, in test2, user3 took almost double the time user1 took to complete its matrices. These results can be attributed to user3's phone having the most modest hardware specs out of the three phones. It should be noted that both users in Hawaii were on their LTE cellular network. While user5 was on WIFI.

Different Tile Sizes

These test were done to examine how tile sizes affect a phone's time to compute matrices. Three matrices of 4000 with three different tile sizes were computed. Test1 had a tile size of 4000. Matrix A was 75.6 MB. Matrix B was 79.5 MB. Test2 had a tile size of 2000. The matrix A tile was 17.9 MB. The matrix B tile was 19.5 MB. Test3 had a tile size of 1000. The matrix A tile size was 3.5 MB. The matrix B tile size was 4.5 MB. The computation time was determined by total computation time / total number of matrices compute.

In the first experiment, user3 completed all three tests alone. When the matrix was not separated into tiles, the computation time was the fastest, at 6 s per matrix. Computation time increased as there was more tiles to compute. The slowest computation time was for the matrix separated into tiles of 1000, at 396 s per matrix.

In the second experiment, user3 and user5 completed all three tests together. These tests followed the same trend as when user3 completed them alone. Each test had slightly higher times than its counterpart. Test1, the fastest, had a speed of 7 s per matrix. Test3, the slowest, had a speed of 510 s per matrix.



All test show that despite the largest tile takes 170% more space than the smallest tile, it is much more efficient to compute. The largest matrix tile, matrix B in test1, was 79.5 MB. It's smallest counterpart tile, matrix B in test3, was 4.5 MB.

Conclusion

All three issues, (i)create a simple way for users to contribute to the project (ii)look at how a user's location can affect their contribution to the project (iii)study how partitioning input files can affect a project's efficiency, were addressed. Many lessons about BOINC, though not all discussed, were learned. A second iteration of experiments would go more smoothly. The following are future challenges that can be explored.

The BOINC server's modest hardware specs did not allow the creation of more and larger matrices. And so the Android phone's maximum potential was not explored. A more powerful BOINC server can be implemented to gather more data on Android phones' potential. It was also mentioned that matrix multiplication was simulated. The BOINC client app's design and BOINC server's workflow had compatibility issues. This problem should be further looked into.

Only users from two states participated in the experiments. More participants from different locations can be added. This can be used to examine how many users the BOINC server can schedule for. As the number of phones increase, how does this affect the project's efficiency? To aid in these efforts, a simpler BOINC set up processor should be explored. This will encourage more users to volunteer for the experiments.

Although the project's app was written in C and C++, writing the app in Java/Android should be explored. Doing so will allow the developer to use Android Studio which provides syntax checking, a GUI compiler and other helpful tools. Also, the developer can avoid the difficulties of writing an app in C and C++.

References

http://www.top500.org/lists/2015/11/
http://www.wsj.com/articles/google-says-android-has-1-4-billion-active-users-1443546856
https://boinc.berkeley.edu/
http://developer.android.com/tools/sdk/ndk/index.html