**CPSC454-01 Phase 2 Progress Report**

**Project Members:**

Christian Angeles:

* Email: [christian.angeles@csu.fullerton.edu](mailto:christian.angeles@csu.fullerton.edu)
* Major: Computer Science (Undergraduate)

John Zavala:

* Email: [johnzy27@yahoo.com](mailto:johnzy27@yahoo.com)
* Major: Computer Science (Undergraduate)

**Project Title:** Mobile Cloud Computing

Topics:

* Cloud Security
* Distributed Computing
* Mobile Cloud Computing
  + Android platform; computational offloading to cloud server

**Project Description:**

Our goal is to simulate computational offloading of a compute-intensive task on an android application to a cloud server. This server will complete the task via distributed computation. The data will be an array of at least 100,000 elements conceptualizing a large data set. Data that is too large to store on an android device. The program on the cloud server will be performing an exhaustive search and comparison on such data to simulate the computational offloading. In order to create a compute-intensive environment, the algorithm is going to search and compare the 100,000-element-sized array with itself. Forcing the algorithm to perform with a time complexity of O(n2). A slower run time will allow us to see a difference in computational performance.

**Project Setup Environment:**

Amazon Web Services (AWS) EC2, StarCluster, Message Passing Interface (MPI), Android

**Project Skills Needed:**

Languages: Java, Python

Operating System: Android, Ubuntu

**Completed Task List:**

Amazon Web Services (AWS) EC2

* Account creation and familiarization
  + created instances
  + security group configuration
  + ssh connections to AMIs using private keys
* Implemented cloud server environment with StarCluster toolkit

StarCluster

* Learned how to use the toolkit to manage cluster (terminal commands)
  + start, stop, terminate, and create a cluster
  + upload/download files to nodes
  + adding/removing a node
* Configured settings to automate creation of cloud cluster
  + Ubuntu 13.04 starcluster AMIs (master and two slave nodes)
  + Network File System (NFS) sharing with Amazon’s Elastic Block Storage (EBS)
  + Security Group (inbound/outbound SSH connections)
  + Password-less SSH by creating a public keys for VMs

Message Passing Interface (MPI)

* Learned how to implement a program in C/C++ and Python
  + mpi4py library for Python
  + Initialize and Finalize MPI program
  + rank/process
  + size of the MPI “world”
  + communication between processes
    - Point-to-Point
    - Collective Communication
      * Broadcasting
      * Scattering and Gathering

Cloud Server Program

* Written in Python
  + simpler to write MPI program in Python
  + saves time from compiling C/C++ code
  + MPI Initialize and Finalize is done for you
* MPI program
  + divides the array into segments
    - number of segments are based on how many processes are allocated to complete the task
      * programmed to dynamically scale
  + array elements are initialized to 0 except for the last index
  + the algorithm to search and compare for a number greater than 0 is a simple double for-loop
    - a number is placed at the last index for the worse-case time complexity
  + implemented randomization
    - a random index will be assigned a number greater than 0
    - shows dynamic scaling and expected run time
  + implemented display output to identify which process belongs to which node
  + implemented timer
    - shows how long it takes for the program to finish

Socket Programming

* Not implemented at this time

Android Application

* Not implemented at this time