High Performance Computing Over Parallel Mobile Systems

Doha Ehab Attia
Department of Computer Science
Faculty of Computers and Information,
Cairo University
Cairo, Egypt

Abeer Mohamed ElKorany
Department of Computer Science
Faculty of Computers and Information,
Cairo University
Cairo, Egypt

Ahmed Shawky Moussa
Department of Computer Science
Faculty of Computers and Information,
Cairo University
Cairo, Egypt

***Abstract*—There are currently more mobile devices than people on the planet. This number is likely to multiply many folds with the Internet of Things revolution in the next few years. This may treasure an unprecedented computational power especially with the wide spread of multicore processors on mobile phones. This paper investigates and proposes a new methodology for mobile cluster computing, where multiple mobile devices including their multicore processors can be combined to perform possibly massively parallel applications. The paper presents in details the steps for building and testing the mobile cluster using the proposed methodology and proving the successful implementation.**

Keywords—Parallel computing; High-performance computing; Mobile computing; Cluster computing; Android OS

I. Introduction

The International Telecommunication Union (ITU) estimates that by the end of 2016 there will be almost 7.3 billion mobile subscriptions [1], which is equivalent to 95 percent of the world population. The number of mobile broadband subscriptions is also rising significantly due to the ever increasing popularity of mobile devices (e.g., smartphones, tablets, notebooks, iPads, etc.) with an estimate that the number of subscribers will reach 2.3 billions globally by the end of 2016.

Another important prediction is that the number of mobile devices per capita, already increasing significantly, is expected to reach 1.5 by 2020 [2]. These numbers and predictions understandably made the mobile communications one of the fastest growing fields of technology. Many people are starting to depend on their smartphones as a primary and, sometimes, only computing device. The constant development in the capabilities of mobile devices enabled those devices to be in some cases an adequate replacement for traditional computers where it can compute with comparable performance.

With both the rapidly growing capabilities and wide spread of mobile devices, the authors expect further growth of using mobile devices in general and smartphones in particular for more performance-intensive computing tasks that were previously handled by traditional computers. To deal with this tidal wave of high-performance applications the quality of experience on devices based on single core CPUs rapidly degrades when users run several applications concurrently, or run performance-intensive applications. Therefore, smartphone industry transitioned to multicore CPUs to cope with the performance challenges. The availability of multicore mobile devices makes them a prime candidate for parallel computing applications. Similar to what happened with traditional computers, the authors expect another extension of mobile parallel computing to include multi-node High Performance Computing (HPC) clusters.

Another critical factor to consider is the underutilization of mobile devices compared with the energy consumed. Most mobile computing and communication devices recharge their batteries because of the power leakage and the power consumed just to keep the devices on, without actual use. This observation applies more to the corporate mobile systems. This was another motivation for both the research community and industry to develop solutions to use the wasted energy in distributed mobile systems for larger combined computational power. For example, HTC and Samsung recently launched mobile applications (HTC Power to Give [3] and Samsung Power Sleep [4]) to enable smartphone owners to contribute unused computational power in mobile grids.

Using smartphones as a portable high performance computing infrastructure can be beneficial in many ways:

- Situations where access to HPC machines is non-existent and the only means for accomplishing resource-intensive computations are mobile devices. Those situations turned out to be widely spread in many military as well as civilian applications.
- Performance and energy-efficiency gains. By splitting the processing among several devices. This will, not only speed up the processing, but will also distribute the battery drain across all devices. Several solutions have been proposed to enhance the CPU performance [5], [6] and to manage the disk and screen in an intelligent manner to reduce power consumption [7], [8]. However, these solutions require changes in the structure of mobile devices, or they require a new hardware that results in cost increase and may not be feasible for all mobile devices. But distributing the workload over a large number of processors may reduce the power consumption of each device separately.
- Using the mobile cluster for educational purposes by creating an on spot parallel computing cluster using the

available smartphones in the classroom, especially when obtaining HPC machinery for education purposes is not feasible or possible.

- Another usage for the smartphone-based computing infrastructure is that an enterprise could benefit from significant energy savings and better operation by offloading tasks to clusters of smartphones and converging the clusters to the larger cloud.

Based on the above introduction and motivation, the research team investigated a solution based on developing a portable parallel computing cluster made of available smartphones. A key point is that the cluster uses standard parallel computing languages and techniques to facilitate the migration between the developed solution and standard HPC systems and clouds. Therefore, the solution is entirely based on standard C programming with Message Passing Interface (MPI).

The rest of the paper is organized as follows: Section 2 reviews the previous approaches identifying the shortcomings of each, which lead to the current research. Section 3, describes the details of building the proposed solution. Testing and evaluation of the developed mobile cluster are introduced in Section 4. Finally, Section 5 provides the conclusions and potential future applications.

II. Previous Approaches

There have been several parallel processing attempts on mobile devices, specifically on smartphones. In 2008 Daniel Doolan, Sabin Tabirca, and Laurence Yang implemented an MPI library on a Bluetooth network in a Java based environment [9]. The authors of the current paper attempted reproducing the reported Bluetooth cluster as described in the paper with no success. Despite the failure to replicate, we have three reservation on the published work today: (1) Doolan et al. reportedly redefined the MPI library to fit with the Bluetooth network. Hence, the MPI implementation is no longer standard to fit, and integrate, with other cross-compiled MPI systems, (2) The Java-based environment is on the decline [10] which makes the resulting cluster based on outdated technology, and (3) Typical HPC software systems are not developed with Java due to the virtual machine nature preventing full performance control.

Another attempt was carried out by Hinojos et al. [11]. The project is named BlueHoc, a system that enables distributed computation on Android smartphones via Bluetooth networks. It was designed for military fields where there might be no access to a fixed computing infrastructure. MPI was not used for the parallel programming on the BlueHoc system. An alternate approach was developed using the radio frequency communication (RFCOMM) protocol. This is clearly a nonstandard HPC programming library which, again, impedes the integration of parallel mobile systems into clouds and fixed HPC infrastructures.

Felix Büsching, Sebastian Schildt, and Lars Wolf developed another approach titled DroidCluster, based on installing Debian ARM on the SD card of a smartphone rather than direct Android programming. Their published paper lists a number of potential applications of portable clusters and uses the LINPACK benchmark to test the cluster performance [12].

In 2013 Chien-Chung Wu and Juyn-Jie Huang published a study implementing Android parallel programming based on the dual-core Cortex A9 [13]. This study focused on single node multicore programming with OpenMP rather than cluster computing with MPI.

Finally, Iulian Vîrtejanu and Costică Nitu developed a mobile cluster based on cross compiling the MPICH2 library for Android phones. The paper does not mention the communication protocol used to form the mobile cluster (e.g. ssh). Also, the detailed steps for cluster building were not mentioned [14].

III. Cluster Building

A. Software

In July 2016, Android's market share was 66.01%, exceeding all other smartphone's operating systems [10]. The availability of Android phones was not the only reason for choosing it for building our mobile cluster. The fact that Android is open source, Linux-based operating system allowed us to assume that all Linux applications can be easily ported to Android mobile devices. But that was proven incorrect. While Android is Linux-based it does not fully utilize the standard Linux kernel. Because Linux is open-source, the Android developers modified the Linux kernel to fit their needs [11][16]. There are some major differences between Android and Linux. For example, some of the standard GNU libraries are not included in Android. It does not rely on the GNU libc, for instance; it uses bionic instead [11].

Bionic is a C library that is not only much smaller in size than the GNU libc but also has less memory requirements. This means one cannot simply run Linux applications and libraries on Android [11]. For example, one of the libraries that cannot be directly ported to Android is the Message Passing Interface (MPI) library. MPI is an open source, portable library used in high-performance computing for message passing between parallel computing nodes. In order to build an MPI-based mobile cluster, we had to cross compile an MPI version with a library other than Bionic because Bionic does not support the full C/C++ standard [11]. In the current research project, the authors have chosen to cross compile and statically link MPICH which is a freely available implementation of the MPI standard to the Uclibc library.

Uclibc is a C library that is smaller in size than glibc; it was intended for Linux-based embedded systems [17]. A Uclibc based GNU toolchain was generated for the ARM architecture, the processor's architecture of the devices used in the current research. The authors used Buildroot, an open source tool to generate embedded Linux systems [18], to generate the required toolchain for the ARM architecture. In addition, The research team used the Android Debug Bridge (ADB) tool to access the used smartphone's shell. Finally, The researchers had to unlock the bootloader of the Android devices used in the experiment to enable access to the devices with root-level permission as a super user.

B. Hardware

Table I shows the hardware specifications for each of the two devices used in building the mobile cluster.

TABLE I. HARDWARE SPECIFICATIONS

specifications	Devices used	
	Device 1	*Device 2*
Number of cores	4 (1.2GHz)	2(1.5GHz)
Operating system	Android™ 4.4.2	Android™ 4.1.1
RAM	1GB	1GB
ROM	8 GB	8 GB

C. The steps for building a cluster of Android devices

1) Generate the toolchain. After downloading Buildroot software, go to the Buildroot directory and use make menuconfig command to open Buildroot configuration tool. After choosing the suitable configuration options, start the building process using make command. The Output will be found in '/PathToBuildroot/output/host/usr/bin' directory.

2) Compile MPICH for the ARM architecture. After downloading MPICH, create a new folder with the name “build”. Then use the following command to statically build MPICH (from the MPICH directory):

```
./configure –prefix=/mpich/build/
CC=/PathToBuildroot/output/host/usr/bin/arm-buildroot-
linux-uclibcgnueabi-gcc
CFLAGS="-I/PathToBuildroot/output/host/usr/include/ "
LDFLAGS="-L/PathToBuildroot/output/host/usr/lib/ "--
host=arm-linux --disable-shared --enable-static –disable-
fortran.”
```

3) Use /mpich/build/bin/mpicc command to compile your C code. e.g.: /mpich/build/bin/mpicc -o myCode mycode.c Then copy the generated “myCode” file to each device /data directory.

4) Change /system/etc/hosts file on each device to contain all IP addresses. Any file explorer application that requires root permission can be used. The hosts file should look like this:

```
10.0.0.1 localhost
10.1.1.1 hostname1
10.2.2.2 hostname2
```

NOTE: The default hostname ”localhost” was changed on all devices and each device was give a distinct hostname.

5) Copy the files in the generated /mpich/build/bin directory to each device in /system/xbin directory. The files can be copied from your computer to the mobile phones using ADB. Connect your phone to the computer, open the terminal and run:

```
adb push -p /mpich/build/bin /system/xbin
```

In case that ‘Read-only file system’ message appears you can mount your system to read/write by running the following commands from your terminal: type adb shell**,** the phone's terminal will open. Then type su command to have super user permissions. By using the following command mount -o rw, remount /system you should be able to copy the files. In case that ‘permission denied’ message appears: Run chmod 777 /system on the phone's terminal. Now MPICH should be running on the device to test it you can run the following command from the phone terminal: mpiexec -n 1 /data/myCode**.**

6) Setup ssh for communication between devices. First you have to download ssh server for your Android phones. In our case we used Android ports “Unix command line packages built for Android”. Use the mobile device's shell to run this command:

opkg install dropbear openssh. Then generate public and private key pairs in //.ssh/id_rsa (in system(root)) using this command /data/local/bin/ssh-keygen -t rsa. After copying the public key to the other phone //.ssh/authorized_keys (if the file does not exist, create it). Start dropbear on all devices by running: /data/local/bin/dropbear. Then Check that ssh is working by trying this command /data/local/bin/ssh root@ipaddress.

NOTE: The hotspot was enabled on one of the devices and the other devices connected to it without being connected to the Internet just to create a local network.

IV. TESTING AND EVALUATION

A matrix multiplication program was developed to test the mobile cluster. The cluster was tested by executing a sequential version of the program on a single device. Then multiple parallel versions with a different number of processors were executed on the mobile cluster. Finally, we compared the execution time of all the runs.

The following is the pseudocode of the matrix multiplication program that was executed on the developed mobile cluster

```
Set the matrix size
If rank equal zero
    Initialize the two matrices
    Distribute the matrices based on the matrix size and the
number of processes
If rank greater than zero
    Multiply the assigned part of the matrix based on the
process number
    Send the process result
If rank equal zero
    Receive/collect results from other processes
```

Table II shows the results of running the matrix multiplication code using square matrices of increasing sizes.

The table contains the time taken in seconds to execute a matrix multiplication C code using one processor on device1 then using four processors on device one then using six processors on device one and device two.

TABLE II. THE RESULTS OF RUNNING MATRIX MULTIPLICATION CODE ON MOBILE CLUSTER

Matrix Dimensions	Number of processes		
	1	*4*	*6*
200*200	1.136	0.552	2.082
400*400	7.452	3.296	5.802
600*600	27.03	9.686	11.962
800*800	68.194	25.652	34.436
1000*1000	136.152	49.192	56.168
2000*2000	1120.506	479.924	385.81
3000*3000	3994.886667	2231.788	1706.038
4000*4000	13107.08667	6638.176667	4844.5725
5000*5000	23148.62333	13920.55667	9351.23

Fig 1. illustrates the results. It shows that when the matrix size was small, the run time for the program was almost the same on single core and multiple cores. As the matrix size increase, the time taken increased dramatically on the single core, a clear indication of the need for extra computational power to cope with the growing problem size. As the number of processors increases the execution time decreases.

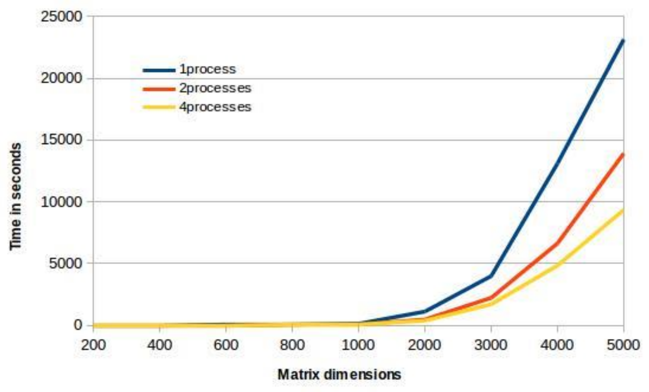

Fig. 1. The result of running matrix multiplication on mobile cluster

V. CONCLUSIONS AND POTENTIAL APPLICATIONS

A. Conclusions

The results show that using a cluster of smartphones as a high performance computing infrastructure is possible and can be in some cases an adequate alternative to the traditional cluster. It can also open the way to a plethora of new applications where the computational power of the smartphones are used to their fullest potential.

We also found some limitations in the mobile cluster, one of which is that the nodes of the cluster are constantly moving which means that some nodes may leave the cluster before finishing the job. Another limitation is that nodes of the cluster have to be predefined, the IP address of each node has to be mentioned in the hosts file before executing any job on the cluster.

B. Potential applications

In addition to the previously identified applications in the literature [12], the current research authors suggest and plant to work on the following potential applications:

- Personal identification by parallel biometrics computing using mobile devices. To overcome the limitations of the existing password-based authentication services on the Internet, we integrate personal features (ex: fingerprints, palmprints, hand geometry and face) into a hierarchical structure for fast and reliable personal identification and verification. To increase the speed and flexibility of the process, mobile devices can be used as a tool for parallel implementation in a distributed environment. The benefit of using a corporate or cloud mobile cluster for this application is two-fold. On one hand, it is increased security with distributed verification and manipulation of biomarkers. On the other hand, the use of parallel mobile clusters saves energy, computational resources, and consequently operation cost due to the saving of no longer needed dedicated infrastructure.

- Distributed key agreement. Securing the access to certain files/ places within the same institution by distributing the access key over different mobile nodes. The dynamic re-allocation of the access keys serves like the multiple keys safes to increase the security and enables the tracking of intrusion. This highly optimizes cloud security.

- Field data collection and processing. Capture, process and share data in places (such as military fields, agriculture fields, desert and geological excavations and navigations, etc.) where access to HPC (high performance computing) machines is non-existent. The utilization of the processing power of available devices, dynamic, mobile, ad hoc clusters can be used for the initial data collection and preprocessing. This may reduce or eliminate the need for data and program transmission.

- Video and image understanding applications.

ACKNOWLEDGMENT

The research team acknowledges the use and contribution of the HiPer-FC cluster and HPC group at the faculty of computers and information, Cairo Uuniversity.

REFERENCES

[1] Statistics. (2016). ITU. Retrieved 30 August 2016, from http://www.itu.int/en/ITUD/Statistics/Pages/stat/default.aspx

[2] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020 White Paper. (2016). Cisco. Retrieved 31 August 2016, from http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

[3] HTC Power To Give | HTC United States. (2016). HTC. Retrieved 30 August 2016, from http://www.htc.com/us/go/power-to-give/

[4] Power Sleep - DO GOOD WHILE YOU SLEEP. (2016). Samsung.com. Retrieved 30 August 2016, from http://www.samsung.com/at/microsite/powersleep/en/

[5] Kakerow R. “Low power design methodologies for mobile communication”, In Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2003; 8.

[6] Paulson LD. “Low-power chips for high-powered handhelds”. IEEE Computer Society Magazine 2003; 36(1): 21.

[7] Davis JW. “Power benchmark strategy for systems employing power management”, In Proceedings of the IEEE International Symposium on Electronics and the Environment, 2002; 117.

[8] Mayo RN, Ranganathan P. “Energy consumption in mobile devices: why future systems need requirements aware energy scale-down”, In Proceedings of the Workshop on Power-Aware Computing Systems, 2003.

[9] D. Doolan, S. Tabirca and L. Yang, "MMPI a message passing interface for the mobile environment", Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia - MoMM '08, 2008.

[10] "Operating system market share", Netmarketshare.com, 2016. [Online]. Available: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=9&qpcustomb=1.

[11] G. Hinojos, C. Tade, S. Park, D. Shires, and D. Bruno, “Bluehoc: Bluetooth ad-hoc network android distributed computing”, Int. Conf. on Parallel and Distrib. Process. Tech. and Appl.(PDPTA), pp.468-473, 2013.

[12] F. Busching, S. Schildt, and L. Wolf, “DroidCluster: Towards Smartphone Cluster Computing The Streets are Paved with Potential Computer Clusters”, In Distributed Computing Systems Workshops (ICDCSW), pp.114-117, 2012.

[13] C. Wu and J. Huang, "The Study of Android Parallel Programming Based on the Dual-Core Cortex-A9", 2013 Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2013.

[14] I. VÎRTEJANU and C. NIŢU, "Programming distrinuted applications for mobile platforms using MPI", U.P.B. Sci. Bull., vol. 75, no. 4, 2013.

[15] "Bionic (software)", Wikipedia, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Bionic_(software).

[16] W. Project!, "Android Open Source Project", Source.android.com, 2016. [Online]. Available: https://source.android.com/.

[17] "uClibc", Uclibc.org, 2016. [Online]. Available: https://www.uclibc.org/.

[18] "Buildroot - Making Embedded Linux Easy", Buildroot.org, 2016. [Online]. Available: https://buildroot.org/.