DroidCluster: Towards Smartphone Cluster Computing - The Streets are Paved with Potential Computer Clusters -

Felix Büsching, Sebastian Schildt, Lars Wolf
Technische Universität Braunschweig
Institute of Operating Systems and Computer Networks (IBR)
Mühlenpfordtstraße 23, 38106 Braunschweig, Germany
Email: {buesching | schildt | wolf}@ibr.cs.tu-bs.de

***Abstract*—What is the processing-power of an omnibus? Can a train compute a climate model? Today's smartphones are becoming more and more powerful and have a performance similar to former high-end workstations. This power can also be used in a joint and cooperative way by building local and mobile ad-hoc clusters. In this paper we will show that setting up a smartphone cluster is not only possible, but it is also a reasonable thing to do, considering the sheer amount of mobile devices and the applications that could benefit from it.**

***Keywords*-Cluster Computing; Smartphone; Mobile Cluster**

I. Introduction

Cluster- and Grid-Computing are well-known and frequently investigated topics. There has been plenty of research work during the past 30 years and when looking at Cloud-Computing nowadays, there is still recent and ongoing work in that area.

In mobile computing, there is obviously a trend to miniaturize and save energy, but at the same time mobile devices are also becoming more powerful. In fact, nowadays smartphones have the computational capabilities of high-end workstations from a few years ago. The SGI Indigo-2 from the middle of the 1990ies was considered a high-end graphical workstation. The Indigo-2 used a MIPS based processor, whose architecture is very similar to the MIPS cores used in many embedded systems today. In the Indigo-2's highest configuration the CPU reached 195 MHz. Performance metrics that are easily surpassed by even entry level smartphones. And when yesterday's clustered workstations could compute climate models or simulate nuclear explosions, clusters of today's smartphones could do so as well.

Many successful scientific projects such as the pioneer *Seti@home* , *Einstein@home* , the *World Community Grid* or the *distributed.net* projects have shown, that for certain workloads, volunteer computing is a viable alternative to buying or renting big compute clusters. The main motivation for these projects, are lower costs compared to classic compute clouds [1]. While the main idea was to tap into the computing power available in private hands, these days mobile devices such as tablets and smartphones are about to overtake PC's in market penetration. In this paper we argue, that current mobile computing platforms are becoming powerful enough, and are widespread enough, that they should be considered as viable compute resource for computation intensive tasks such as the work offloaded to volunteer computing projects today. As all these devices are also equipped with WiFi, which offers plenty of bandwidth at reasonably good latencies, all the ingredients for forming capable ad-hoc computing clouds are there.

In the remainder of the paper, we start in section II by suggesting some scenarios where it is reasonable to use the computational resources of mobile devices. Section III gives an overview about the current state and development of technology for mobile computing. We present a feasibility study, implementing and evaluating a small MPI cluster using ordinary Android mobile phones, in section IV. Finally, in section V, we finish with some concluding remarks.

II. Applications

Not only are mobile computing platforms getting faster, but smartphones and tablets are on their way becoming the primary (or only) computing device for many people. The market penetration for mobile phones in general is much higher than for PCs. In 2010 shipments of smartphones surpassed PC shipments [2] for the first time.

In the following we present some visionary applications for mobile clouds which can be established in an ad-hoc and on-demand way to solve computation problems in cooperation. Additionally, the existing volunteer computing projects mentioned in the introduction could definitely be ported to mobile devices as well.

A. Rolling Clouds

Consider a train as an example where the ad-hoc formation of a mobile computing cloud will make sense: Most passengers have mobile phones, and they can use the carriage's sockets to charge them, so that energy drain is not an issue. For example a Siemens Velaro D high-speed train has a total capacity of 460 passengers. A second class carriage has 76 seats. Within a carriage mobile devices can easily form a closely coupled computing cloud working on principles such as MPI [3] using WiFi. Due to communication constraints, a computing cloud encompassing a whole train should be built on an architecture for loosely coupled distributed systems, similar to what current public distributed computing platforms such as BOINC [4] use: Dispatch tasks

to devices (or in this scenario closely-coupled sub-clouds) and aggregate results in a backend. The WiFi infrastructure already built into modern trains for providing passengers with internet access can be used to facilitate communication between carriages.

In the near future, when you embark on a train journey, you will plug in your mobile phone. While your mobile phone is charging it will form a mobile cloud with the devices of fellow travelers. An oncoming train will transmit environmental data from a public sensing system installed at this train's destination. The ad-hoc computing cloud will use this data to calculate a fine grained forecast of local weather and ozone concentrations at your destination. Before you leave the train you put on your raincoat because your device warned you to expect heavy cloudbursts.

B. Corporate Environments

The smarter our phones become, the less the battery seems to last: Each day millions of office workers enter their cubicle, switch on their PC, and plug in their private phone. As an employer you can either tolerate this, and pay for the electricity, or prohibit it. The problem with prohibiting is: It is hard to enforce, and surely not good publicity if you fire an employee for "stealing electricity".

Now, as an employer, you can fight back: Every private device that contributes back to the company's computing cloud, is allowed to be charged. In a software shop, it would be quite easy to deploy *distcc* onto the mobile devices. *distcc* is a distributed compiler framework: When compiling a big software project, compilation units are dispatched to different compile nodes and the object files are later aggregated on the host that initiated the build. If you compile a big project, it does not matter, that a mobile phone maybe needs 5 seconds to compile a file, that the developer's workstation can compile in 0.5 seconds: As the whole build process takes longer than 5 seconds you can still save time by distributing to many different nodes. Thus, your employees will not steal electricity, instead they will bring their own private equipment to increase their productivity.

C. Cooperative Cracking

Moxie Marlinspikes tool WPACracker uses a 400 CPU cluster running in the Amazon cloud. At Black Hat DC 2011 Thomas Roth successfully demonstrated another Cloud Cracking Suite (CCS) that is able to crack WPA-encryption in a reasonable time. Combing the computation power of mobile devices locally brings the needed power to crack WLAN encryption on site. Using a sufficiently large number of smartphones combined with the ability to share their resources and to coordinate a distributed attack will significantly lower the time for any brute-force based intrusion. Please note that this might be illegal in some countries.

III. Mobile Computing Hardware Evolution

Mobile devices follow the technological development of common general purpose PC platforms, only at a much faster pace. As a rule of thumb all architectural improvements from general purpose computing arrive in the mobile market, as soon as current technology allows them to be implemented within the power envelope available to mobile devices. Only, in the mobile space development is faster. These days, due to the boom in the smartphone and tablet sector, mobile platforms are evolving faster and decreasing the lead of general purpose architectures.

The LG P500 used for the demonstration in this paper has been a Midrange Smartphone as of 2011. It includes a 600 MHz Qualcom MSM7227 CPU based on an ARM 11 core. In 2012 the midrange already moved up to ARM Cortex A8 based designs at around 1 GHz. The A8 is a dual issue in-order architecture including the NEON SIMD extensions and a much improved FPU. Today's high end mobile devices implement the Cortex A9, which allows for dual-core and quad-core SoC configurations and is a superscalar dual-issue fully out-of-order design. The A9's successor, the A15, will reach the market soon, including further architectural improvements. The market for mobile computing SoCs is highly competitive: Not only is there a fierce competition on the SoCs based on ARM cores, but also on the cores itself: ARM architectural licensees offer compatible core architectures competing with ARM's Cortex designs. Qualcomms upcoming Krait architecture will be a ARMv7 compliant core competing against the Cortex A15 based designs. Apart from the ARM world the PPC and MIPS family of SoCs are also still very much alive

That CPUs architectures coming from the embedded and mobile markets are a viable fit for high performance computing environments, is also demonstrated by Calxeda's "EnergyCore", which is an ARM based quadcore SoC with an integrated 80 GBit fabric switch targeted for many-core clusters [5]. This technology is implemented by HP's Redstone research platform, which integrates 2800 Calxeda cores into a a single rack. [6].

In contrast to common PC platforms from 10 years ago, current smartphones also include capable GPUs. All contemporary mobile GPUs support Open GL ES 2.0, which is the OpenGL variant for embedded systems. The 2.0 variety requires programmable shaders, which have been the first step to enable GPGPU on common PC platforms. Furthermore, cutting edge mobile GPU solutions such as Imagination Technologies PowerVR SGX series or ARM's own MALI T600 series even support OpenCL. This allows leveraging experiences form GPGPU computing on the desktop and applying it to mobile platforms.

We measured the performance of various Android devices available in the lab using a LINPACK benchmark available in the Android market[1]. Table I summarizes the results. While the absolute values are not so interesting, we want to point out, how the changes in performance reflect the rapid architectural innovations that we can currently witness in the mobile SoC market.

This short overview of the current state of mobile devices shows that the computing power available in small mobile devices already surpassed the computing power of high-end

[1] https://market.android.com/details?id=com.greenecomputing.linpack

workstations from a few years ago and is rapidly moving into the area dominated by current low end and (in the graphics area) mid-range stationary computing.

IV. Feasibility Study

To demonstrate that it is feasible to build an Android cluster with currently available hardware and software, we built a small proof-of-concept cluster with 6 Android nodes.

As evaluation software we chose to run LINPACK as it is a standard benchmark for HPC systems, which are usually composed of many compute nodes running concurrently. While it is of course possible to build a compute cluster out of ARM nodes [6], our goal is to be minimally invasive: We do not want to heavily modify or even replace the installed Android systems. If distributed computing solutions get deployed on mobile systems, it will be important that they can run alongside the Android system and not interfere with the devices primary function.

To distribute the calculation, we use a LINPACK implementation based on a MPI library. Message Passing Interface (MPI) [3] is a standard describing the message exchange in parallel computations in distributed systems. MPI applications consist of communication processes that are executed parallel on distributed cores or systems. The parallel processes are usually working on the same problem and are exchanging messages, e.g. via TCP.

LINPACK benchmarks [7] are frequently used to measure a system's floating point computing power, although the original LINPACK is first of all a routine library for solving systems of linear equations. Today, LINPACK is the standard benchmark for the TOP500 list, which aims to gather the most powerful (known) computer systems in the world.

A. Test Environment

Last year, we equipped a student lab with six cheap LG P500 Android phones. Regarding the computational power this is considered a low end phone today: It is equipped with a 600MHz MSM7227 processor (see table I) and 512MiB RAM. Therefore the performance results measured in this study are not expected to be impressive, but rather show the principal feasibility and scalability distributing LINPACK on this less than ideal environment.

The installed Android system has only been modified minimally: To enable the installation of the needed testing tools, root access to the smartphones was obtained by running the *z4root* tool. Then we created a Debian ARM installation in a folder on a SD Card using debootstrap. The rooted phone allows to get shell access and then *chroot* into the base Debian system installed by debootstrap. Please note, that by using this method we do not interfere with any software already installed on the phone. The Android system can be used just normally. The Debian userland runs under the same kernel the phone has been booted with.

In the Debian system we installed a MPI library and HPL which is the actual MPI based LINPACK implementation. We also installed an SSH server, so the phones could be comfortably configured over the network (even with all the advances in the smartphone sector, using a real keyboard is still a bit more comfortable than typing shell commands on a 2" virtual keyboard).

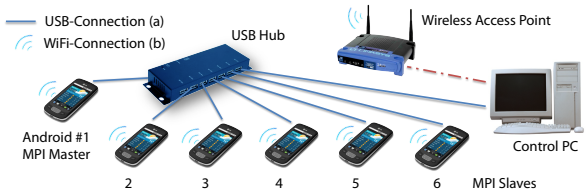

Figure 1. System Overview

B. Test Setup

While normally for MPI applications you want to use low latency links between nodes, for a mobile phone the only realistic available option is WiFi. With a single access point and without any additional routers this would theoretically allow clustering up to 1024 nodes (this is due to limitations in IEEE 802.11's network allocation vector, probably you would run into problems much earlier...). As the used version of Android did not support the IEEE 802.11's ad-hoc mode, we setup a separate access point which was used by the mobile phones. In a real scenario, this access point could also have been provided by an Android device, as all Android phones capable of WiFi tethering can operate as access point.

As WiFi connections are not the most reliable links possible, especially considering the institute's highly crowded 2.4 GHz band (the P500 does not support 5GHz WiFi), for reference we also included the next best solution: When attaching the phones via USB to a PC, a virtual ethernet interface for each phone will be established on the host. We used this USB-ethernet connection and manually set up the routing on the PC. Thus, all phones in the cluster were able to communicate with each other by a wired connection. The setup of the DroidCluster can be seen in figure 1.

We run the LINPACK benchmark over the WiFi and the USB links for 1 to 6 nodes. The used HPLinpack configuration file can be requested by the authors.

C. Results

Figure 2 shows the averaged results of the LINPACK runs. It can be seen that with our HPL configuration a single LP500 reaches 5.81 MFlops. This is in line with the performance expected for this architecture when comparing it with the performance reached by the third party LINPACK benchmark for the P500 listed in table I.

The experiment shows, that despite the less than optimal links, the cluster scales reasonably well up to 6 nodes. While the USB links scale slightly better than the WiFi link, the WiFi cluster still reaches 75% of the optimal value (assuming unrealistic linear scaling) for 6 nodes. The small ditch for 5 nodes in the USB configuration is due to the fact that the HPL benchmark ran with normal priority, as the goal was not to take over the whole system, but run the computation alongside the normal system's operations. Therefore

Table I
LINPACK PERFORMANCE OF DIFFERENT ANDROID SYSTEMS

System	CPU	MHz	ARM Core	Android Version	MFLOPS △
Huawei U8120	Qualcomm MSM7225	528	ARM11	2.3.7	3.7
LG P500	Qualcomm MSM7227	600	ARM11	2.2	4.0
HTC Legend	Qualcomm MSM7227	600	ARM11	2.3.7	7.5
Samsung Galaxy S	Samsung Exynos 3110	1000	Cortex A8	2.3.7	17.7
HTC Nexus One	Qualcomm QSD 8250	1000	Qualcomm Scorpion	4.0.3	31.0
Medion Lifetab P9514	Nvidia Tegra 2	2x1000	Cortex A9	3.2	54.4
Samsung Galaxy Nexus	Texas Instruments OMAP 4460	2x1200	Cortex A9	4.0.2	75.0

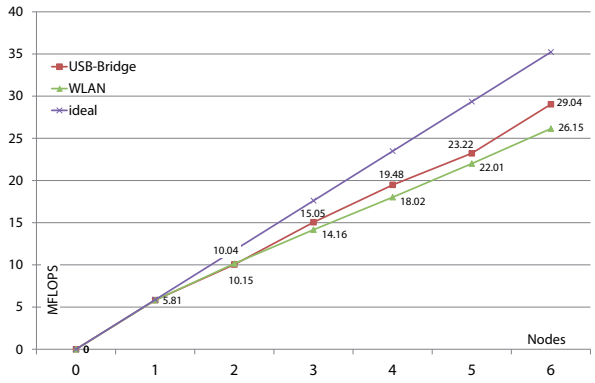

Figure 2. Combined computation power (Mega-FLOPS) of clustered smartphones (1 to 6 phones) running Linpack and MPI.

tasks normally running on the Android can influence the computation task.

Also note that a tightly coupled approach such as MPI to distribute a task puts the most stress on the links. For many use cases a more lightly coupled approach where you send larger tasks to nodes and wait for the results is surely possible and promising; and this is expected to yield even better scaling characteristics.

V. CONCLUSIONS

The current evolution in mobile computing platforms follows the developments in the desktop world, only at a much faster pace. Innovation for mobile computing platforms is driven by a highly competitive market and by the fast adoption of tablets which are situated between mobile and desktop computing and therefore are always pursuing the highest performance. Mobile computing platforms today surpass the computational power of workstations from a few years ago. This combined with the fact that desktop and server hardware is vastly outnumbered by mobile devices deployed today, leads to the conclusion that we should find ways to fully utilize these computational capacities.

We implemented a small feasibility study showing that Android systems today are PC-like enough so that it is easily possible to deploy standard tools and mechanisms from the stationary computing world to successfully distribute computational tasks. We have shown that even using this rather crude methods, it is possible to integrate Android devices into a distributed cluster in a way that does not interfere with the running Android system and applications.

We fully expect that in the future distributed computing frameworks better adapted to the special challenges in the mobile computing world will be developed. A limitation is that it is mostly not a good idea to run CPU intensive applications on battery, but as we have shown there are applications which can be used when the devices are charging anyway.

We also think mobile ad-hoc clouds are environmentally friendly: While running computations uses some extra energy it also means the hardware is utilized more. The amount of energy a device consumes during its lifetime is negligible compared to the energy put into it for production. As mobile platforms continue to evolve, it is to be expected that the "computation per watt" that can be harvested from these machines will soon be as high as for stationary computers. Then we have the situation where, for certain applications, a bunch of mobile devices can replace a stationary server by donating their combined idle times, which is a real benefit in an environmental as well as in a cost sense.

ACKNOWLEDGMENT

This work has been supported by the NTH School for IT Ecosystems.

REFERENCES

[1] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. P. Anderson, "Cost-benefit analysis of Cloud Computing versus desktop grids," in *2009 IEEE International Symposium on Parallel & Distributed Processing*, May 2009, pp. 1–12.

[2] J. Menn, "Smartphone shipments surpass PCs," *Financial Times*, Feburary 8th 2011.

[3] *MPI: A Message-Passing Interface Standard - Version 2.2*. Message Passing Interface Forum, 2009.

[4] D. Anderson, "BOINC: A System for Public-Resource Computing and Storage," in *Fifth IEEE/ACM International Workshop on Grid Computing*. IEEE, 2004, pp. 4–10.

[5] Calxeda Incorporated, "Calxeda Launches the EnergyCore™Processor; Delivers 10 Times the Performance for the Same Power," Press Release, November 2011.

[6] Hewlett-Packard, "HP Shapes the Future of Extreme Low-energy Server Technology," Press Release, November 2011.

[7] J. J. Dongarra, P. Luszczek, and A. Petitet, "The LINPACK Benchmark: past, present and future," *Concurrency and Computation: Practice and Experience*, vol. 15, no. 9, pp. 803–820, 2003.