TMAPP – Typical Mobile Applications Benchmark

Joseph Issa, Quoc Le, Soohong Min, Johann Steinbrecher, Rajesh Turlapati
Silvia Figueira, JoAnne Holliday, Weijia Shang
Department of Computer Engineering, Santa Clara University
Santa Clara, CA 95053-0566
{jissa, qle, smin, j1steinbrecher, rturlapati, sfigueira, jholliday, wshang}@scu.edu

Moenes Iskarous, Dharmesh Jani
Intel Corporation
Santa Clara, CA 95054
{moenes.iskarous, dharmesh.b.jani}@intel.com

Abstract— **Analyzing the performance and power of Mobile Internet Devices (MID) is important for academia and industry as well as for consumers. It is important to assess the performance associated with a typical workload on MIDs so that applications and hardware can be optimized to reduce performance bottlenecks. In this paper, we present TMAPP, a benchmark developed to analyze the performance of MIDs under a typical workload. We specifically target SoC (System-on-Chip) Intel® Atom™ and ARM® CORTEX™-A8 based platforms, using different operating systems, such as MeeGo, Ubuntu and Angstrom Linux.**

Keywords; *performance analysis, mobile internet devices, media player performance, multimedia performance, ARM® CORTEX™-A8, INTEL® ATOM™, 3D game performance, 2D game performance, navigation performance, internet browser performance, OpenOffice performance*

I. INTRODUCTION

Mobile Internet Devices (MIDs) have gained widespread popularity by placing computational power and novel applications conveniently in the hands of end users. MIDs are hand-held computers, which are smaller than a laptop computer, but larger than a smart phone, with a larger and more usable screen. MIDs provide wireless Internet access and location-based services with GPS. MIDs also provide entertainment through multimedia and gaming.

There are many reasons for assessing the performance and power of MID and handheld devices, but the most common one is evaluating hardware and operating system configurations to obtain higher performance-per-watt on a given platform under a typical workload. The motivation behind the power and performance prediction is to facilitate the process of performance and power projection, which helps in the designing and fine-tuning of future processors for optimized performance and power.

Despite the widespread popularity of MIDs, current benchmarks (e.g., SPEC[29], MediaBench[30], MinneSPEC[31]) do not target their typical usage, making it hard to assess their performance and energy efficiency in a real scenario. In this paper, we present TMAPP – Typical Mobile Applications benchmark, which was developed specifically to assess the performance and possibly the energy efficiency of MIDs. TMAPP is formed by independent components, which are representative of these devices' typical usage. The usage model was based on available statistical analysis of MID usage across different population categories. The components are: productivity applications, such as OpenOffice, gaming, multimedia, GPS navigation, and web browser. The benchmark evaluates a processor/OS platform by measuring the performance of the device in a typical scenario. It can also assess energy efficiency, but this is out of the scope of this paper. The benchmark was developed to be browser and OS agnostic and to minimize the run-to-run variation. It was designed to run on INTEL® ATOM™ [6] as well as ARM® CORTEX™-A8[5] MIDs.

The remainder of the paper is organized as follows. Section II presents the platforms used. Section III describes the benchmark implementation. Section IV discusses memory dependency. Section V presents the experiments performed. Section VI concludes.

II. PLATFORMS

To illustrate TMAPP and its potential, we use it to analyze the performance of MID platforms. Our analysis include the impact of the MeeGo[1] operating system on the performance of the ARM® CORTEX™-A8 and INTEL® ATOM™ MIDs. We did experiment with other operating systems, which will be described in the experimental results section. MeeGo is an open-source Linux OS. MeeGo currently targets platforms such as netbooks, handheld computation-and-communication devices, in-vehicle devices, connected TVs, and media phones. All of these platforms have common user requirements in communication, applications, and Internet services in a portable or small device.

In our benchmark performance testing, we used the following platforms for running the benchmark:

- HP Netbook: MeeGo OS with INTEL® ATOM™ 1.66Ghz processor, Memory size = 1GB or 512MB, L2 Cache size = 512K.
- SHARP MID Netwalker[11]: Ubuntu OS[3] with ARM® CORTEX™-A8 @ 800 Mhz.
- Beagleboard[4]: Angstrom[2] and MeeGo OS's with ARM® CORTEX™-A8 @ 500MHz processor, Memory size = 512MB, L2 Cache size = 256KB.

III. BENCHMARK IMPEMENTATION

The goal of the benchmark is to compare processor performance and possibly energy efficiency using different operating systems, by measuring performance in different configurations. The benchmark consists of the following components: productivity applications, such as OpenOffice, gaming, multimedia, GPS navigation, and web browser. The benchmark components are automatically activated by a main script, and each generates a score in seconds or milliseconds, which represents the execution time or CPU time. The mechanism to determine the final score for several iterations for each component or subcomponent is to calculate the median for all runs of each component or subcomponent and use the Geometric Mean of the medians of all of them to generate one final score.

The Unix TOP command is used to obtain the CPU time. TOP provides an ongoing look at processor activity in real time. It displays a listing of the most CPU-intensive tasks on the system, and can provide an interactive interface for manipulating processes.

The benchmark was developed to be network independent, i.e., all data files required to run the benchmark components are stored in the local hard drive to eliminate any network interference, such as network congestion and latency, which may affect performance.

The remainder of this section provides an overview of each component and sub-components. Note that all the components of the benchmark are open source to facilitate the distribution.

1. Productivity

Productivity applications were included in the benchmark even though MIDs are not a convenient production platform, because they are widely used for reading productivity files.

1.1 OpenOffice Overview

The benchmark uses OpenOffice[28], which does not provide a distribution package for ARM® CORTEX™-A8. For this reason, we had to cross-compile it for ARM® CORTEX™-A8 to test its performance on both INTEL® ATOM™ and ARM® CORTEX™-A8 processors.

The benchmark assesses common and specific functions of the OpenOffice applications such as Writer, Calc, and Presentation. The applications are executed via a set of scripts, and the performance metric is the execution time.

1.2 OpenOffice Components

OpenOffice applications on MID devices have the following tasks in common: launching the application, opening a document, saving the document, and closing the document and the application. Each application also has supplementary features. We have tested three components: Writer, Calc, and Impress. Each component has common functions and also features specific to that component. For example, there is scrolling for Writer, calculation for Calc, and slide show for Impress.

The Writer module consists of launching the application, opening a Writer document, saving after modifying it, scrolling through from top to bottom, and closing the document and the application. The Calc module consists of starting the application, opening a Calc document with tables and graphs, calculating various data points in multiple tables and automatically creating graphs in the document, saving it, and closing the document and the application. Finally, the Impress module consists of starting the application, opening a presentation document with animation, modifying it and saving it, showing a slideshow from first to last page and closing the document and application.

2. Gaming

The Gaming component of the benchmark includes assessing the performance of MIDs with 3D and 2D-game workloads. The games selected are automated using X11 libraries played by the benchmark script. Our game benchmark components consist of the following: 3D Extreme Tux Racer (version 0.35)[18] and 2D Gnome-Blackjack[19]. Note that these games were selected because they are both open source and representative of the games played in low-power devices.

The 3D Extreme Tux Racer consists of a penguin racing down from a mountain course of snow and ice collecting herring. The game uses the OpenGL libraries and runs with –a option to (1) record average frames per second and (2) exit automatically in 60 seconds.

The 2D Blackjack game is a multiple-deck casino-rules blackjack game. The objective of the game is to have a higher count than the dealer played by the system without crossing over 21. This game is part of the gnome package. The game uses the Cairo 2D vector-based graphics software library. The game operates as follows:

- For every hand, the main thread in the benchmark automation script sends alternating 'Hit' and 'Stand' commands with a 2-second delay.
- The game ends in 70 seconds.
- The TOP command is used to obtain the game's share of the CPU time.

Table 1 shows the performance metrics for the 2D and 3D games' components.

Table 1: Game Performance Metrics

Game Components	Measured Performance Metrics
3D Game	Frames Per Second (FPS)
2D Game	Total CPU Time spent on 2D game

3. Multimedia

3.1. Overview

The multimedia component consists of the following applications: video playback, audio playback, picture preview, picture format convert, and video transcoding. The applications are executed via a set of scripts, and the performance metric is the execution time.

We did not include webcam and picture capturing, even though these two features are typical usage modules, because they do not provide heavy workload for the processor, and the decoding/encoding portion is covered in picture format convert and video transcoding.

3.2. Multimedia Components

For video decoding, we used MediaPlayer[7][8] software to playback a 640x480 resolution video clip using H.264 video codec. We disabled frame dropping in MediaPlayer so the workload will have to play all the frames, which will determine the processor performance for video playback.

For audio playback, we used MediaPlayer to playback an MP3 audio clip.

Picture animation is used in the benchmark to animate a pre-defined sequence of images. Picture animation is a command-line executable, which is part of the ImageMagick®[9] software package.

The conversion utility is also a command-line executable, part of the ImageMagick® software package. In our benchmark, we used the convert utility to convert images from 20 JPG files to PNG format, keeping the same image size.

Transcoding, in general, is a direct digital-to-digital conversion of one encoding to another. A typical usage model for transcoding is the need to convert video from one format to another, or to reduce video file size of the same format to meet certain storage capacity constraints. In our benchmark, we used an H.264 clip and converted it to an AVI video clip for 300 frames only. We chose 300 frames because of the time it takes to convert the entire video clip on slower MID devices. The transcoding was implemented by a media encoder tool for Linux, which used MPEG4 video codec, and scaled the video to a resolution of 640x480 to convert the 300 frames of the original video clip (1440x1080 H.264 format) to AVI format.

Table 2 shows the performance metrics for the multimedia components.

Table 2: Performance metrics for Multimedia

Multimedia Components	Measured Performance Metrics
Video Playback (H.264 clip)	Decoding time in seconds
Audio Playback (MP3 clip)	Decoding time in seconds
Picture Animation (20 JPG pictures)	Total elapsed time in seconds
Picture Format Convert (JPG to PNG)	Total elapsed time in seconds
Video Transcoding (MPEG4 to AVI)	Total elapsed time in seconds

4. Navigation

To assess GPS-based navigation, we selected the Navit[20] open source GPS application. This is a car and pedestrian navigation and routing engine. Navit supports several map formats such as: OpenStreetMap™[24], Garmin Maps, and US/Tiger maps.

All maps are stored locally on hard drive to eliminate network dependencies. The application calculates, based on the current GPS coordinates, the route to the destination according to Dijkstra algorithm[25]. This is a greedy solution to the shortest path problem in graph theory. It is used in most modern GPS navigation systems to calculate the shortest or fastest route. The application generates turn-by-turn directions based on the calculated route. These directions are displayed in the map screen as well as written to a terminal. Navit supports the configuration to use a speech synthesizer such as eSpeak[21] or Festival[22] to generate speech output from terminal strings. To input the current geographic coordinates, several sources can be used: NMEA GPS sensor, GPS device (external serial or USB GPS receiver), UDP port (friends tracking), and NMEA log file.

Since the navigation application is not run in terms of as fast as possible, to evaluate the performance of the application at a specific platform, the CPU time, obtained with the Unix TOP command, is used.

The configuration used is as follows:
- Two screen updates per second
- NMEA[26] log file as GPS input
- OpenStreetMap™[24] of Santa Clara, CA-95050 (317kB)
- Fixed destination
- Speech synthesizer for turn-by-turn directions

5. Web Browser

The most popular open source web browser Mozilla Firefox[13] was selected. In addition to Firefox, Google Chrome[14], a web browser developed by Google that uses the WebKit layout engine and application framework, has been used on INTEL® ATOM™ for comparisons with Firefox.

The workload for this benchmark is a set of HTML pages (see Table 3) downloaded and stored on the device. Currently, the list is chosen based on our usage model and Alexa top 500 sites[15].

This component measures loading time by inserting JavaScript commands into the pages. The *onload* event is used to know when the browser finishes loading a window or the entire frame in the window. The loading time (in milliseconds) is defined as the interval between starting loading and finishing loading all content within a document, all frames within a frameset, and all elements of a page.

Table 3: Web-page list

Type	URL	Html size (KB)	Image + Script + CSS size (KB)
Social + Blog	Facebook.com	28	528
	Myspace.com	48	448
	Blogger.com	16	136
	Wordpress.com	32	188
	LinkedIn.com	20	332
	Orkut.com	12	8
News	ESPN.com	200	932
	MSN.com	140	528
Multimedia	YouTube.com	96	336
	flickr.com	12	260
	photobucket.com	44	808
Search engine	Google.com	20	84
	Bing.com	40	24
	Yahoo.com	164	744
	Baidu.com	4	16
Cloud application	skydrive.live.com	20	216
	docs.google.com	16	116
Shopping	Amazon.com	228	620
	eBay.com	44	464
Popularity	Wikipedia.org	76	280
	Intel.com	28	340
	Microsoft.com	88	552
	Apple.com	16	652
	Weather.com	152	732

IV. MEMORY ANALYSIS

In this experiment, we analyzed the benchmark performance for different memory sizes. We used MeeGo OS on INTEL® ATOM™ with 512MB RAM and 1GB RAM, and we used MeeGo on ARM® CORTEX™-A8 (Beagleboard-xM) with 512MB RAM, while keeping processor frequency at the default setting. We noticed that there is no performance effect between MeeGo/INTEL® ATOM™ at 512MB and 1GB as shown in Figure 1. Therefore, in the remaining experiments, we do not take memory into account when we analyze processor performance for different frequencies.
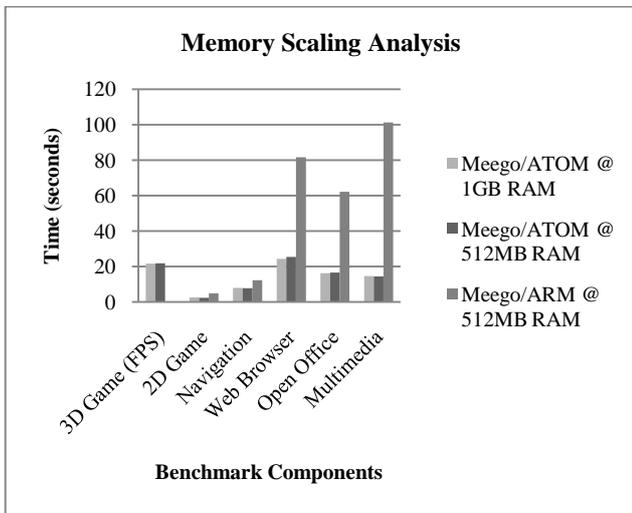
**Figure 1: Performance for different memory sizes**

V. EXPERIMENTAL RESULTS

To show the stability of the benchmark on each platform/system, we calculated run-to-run results based on the geometric mean of 10-run variations with respect to the average taken for runs 2-10. The reason we did not include the first iteration in our averaging is that usually the first iteration has the worst performance number (higher time) due to cash misses, so we excluded it from the average. All the benchmark components presented a run-to-run variation of less than 5% but, due to lack of space, we are omitting the graphs with those results.

We also measured either the execution time or CPU time for each component on multiple platforms based on ARM® CORTEX™-A8 and INTEL® ATOM™ with the same or different operating systems to show how different configurations affect performance. The results for each component are presented below.

1. Productivity

Figure 2 shows the results obtained for the OpenOffice components on multiple operating systems and platforms. To determine the performance dependencies on operating systems, we also evaluated the performance with respect to the change of CPU frequency on Beagleboard with ARM® CORTEX™-A8 processor, on two different operating systems such as MeeGo and Angstrom Linux. First, Figure 2 shows that INTEL® ATOM™ with high capacity is the best compared to other ARM® CORTEX™-A8 machines. Second, according to the change of CPU frequency from 500 to 700 MHz, this graph shows that the performance of the components depends greatly on the CPU performance. Therefore, this test demonstrates that the performance of OpenOffice components is significantly affected by the operating system and its capacity.
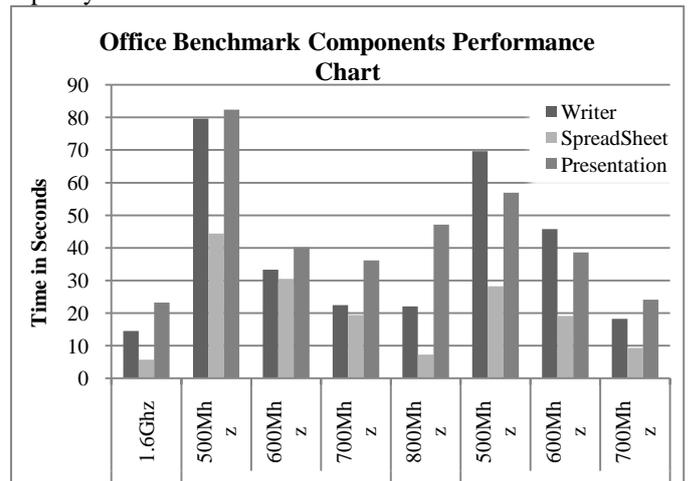
Figure 2: Performance at different platforms with different processor frequencies: HP, Sharp 500-800, Beagleboard 500-700

2. Games

Figure 3 shows the frames-per-second (FPS) obtained on MeeGo on INTEL® ATOM™. Figure 4 shows the FPS obtained on ARM® CORTEX™-A8, which is close to zero,

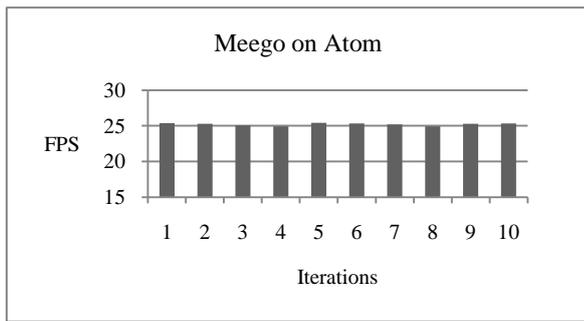because the OPENGL libraries used by the game are not hardware accelerated on the graphics card.

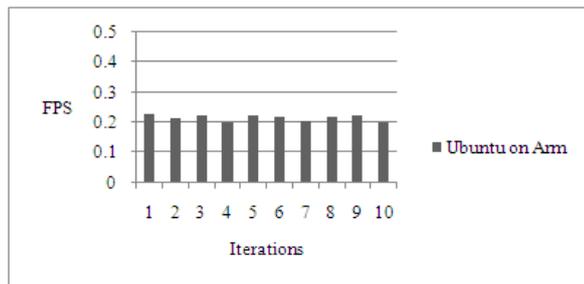**Figure 3: Etracer Game on MeeGo / INTEL® ATOM™**

Figure 4: Etracer 3D game on Ubuntu /ARM

Figure 5 shows the CPU utilization of the 2D game in different system/CPU configurations. The graph shows that the CPU utilization (runtime 60 Seconds) is the lowest on MeeGo on INTEL® ATOM™, followed by ARM® CORTEX™-A8 platforms.
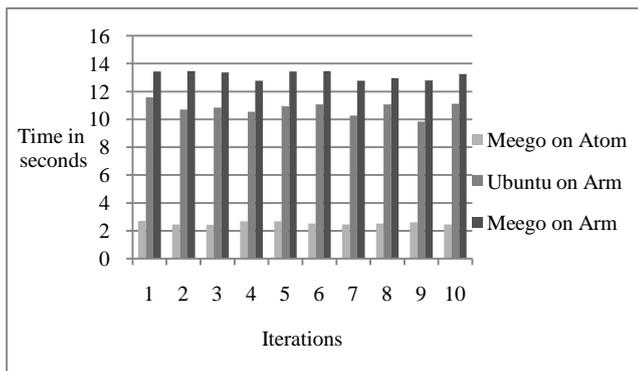

Figure 5: CPU utilization time on various OS and processors after 60 seconds of blackjack run

3. Multimedia

Figure 6 shows all multimedia sub-components timing chart. We also implemented processor frequency variation for ARM® CORTEX™-A8 processor to understand the sensitivity of the benchmark with respect to processor frequency. As expected, the performance for each multimedia sub-component is sensitive to processor frequency and, as frequency increases, time decreases. However, we noticed that the increase in unit frequency (MHz) does not translate to

equivalent reduction in execution time. As frequency increases, the return in performance seems to diminish until it gets to a point where frequency increase does not linearly scale with performance, or worse does not increase at all while power is increasing almost linearly with frequency. This is usually noticed at much higher frequencies than the frequencies shown in Figure 6.
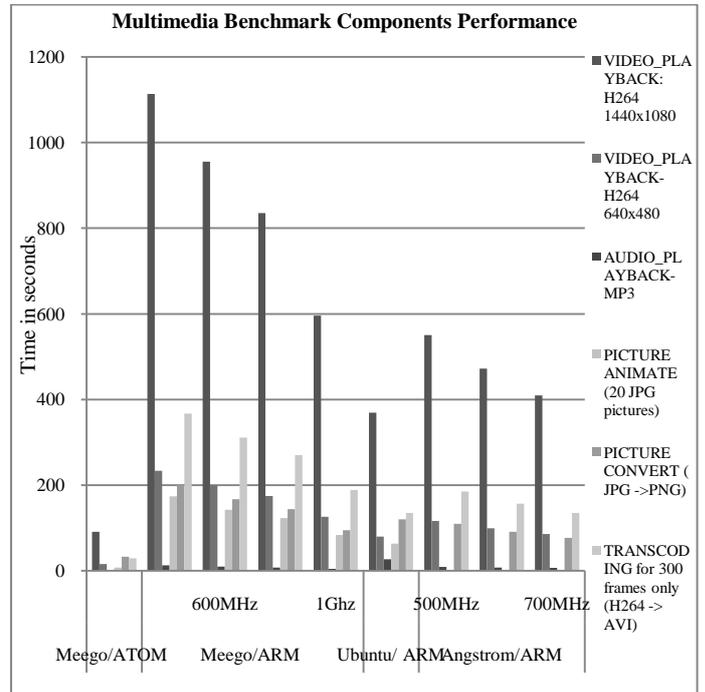

Figure 6: Multimedia components performance chart at different processor frequencies

4. Navigation

Figure 7 shows the frequency analysis of the navigation component running on MeeGo on ARM® CORTEX™-A8. The component was executed ten times on each frequency for 50 seconds per run. As expected, the total time at the CPU increases as the system clock decreases. The repetitive run analysis has shown that this component does not require any cache warm-up as the first iteration did not execute faster than repetitive runs. Therefore, the impact of the cache can be neglected, while the system clock frequency, as shown in Figure 7, has a considerable impact on the performance.
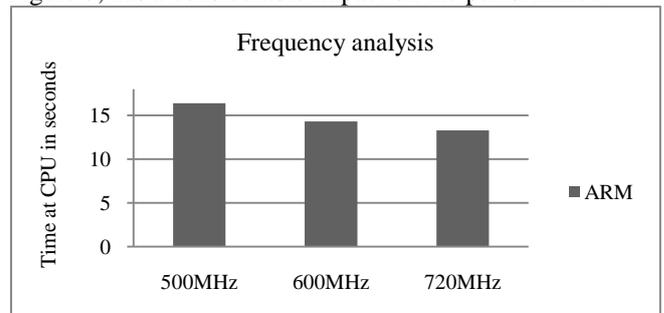

Figure 7: Running the benchmark on MeeGo OS at an ARM® CORTEX™-A8 platform. The component was executed for 50 seconds each time.

5. Browsing

The web browser component is tested with several different OSs with different processor frequencies as shown in Figure 8. As with the other components, the average loading time for each web page from Table 3 decreases as the processor frequency increases, except Angstrom 600MHz takes longer than MeeGo 500MHz, showing that the operating system may impact the performance. The loading time for ARM® CORTEX™-A8 processor is significant longer since it has lower frequency. There is about 18% increase in loading time for each 100 MHz increase of the ARM® CORTEX™-A8 processor frequency, with the processor frequency in between 500-700 MHz.
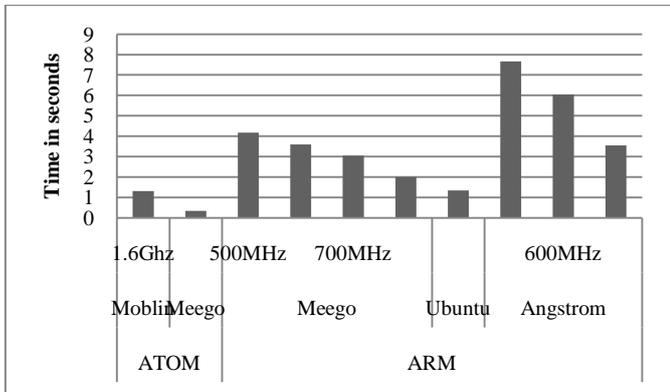
Figure 8: Web browser - performance chart at different processor frequencies on different platforms

VI. CONCLUDING REMARKS AND FUTURE WORK

In this paper, we presented a benchmark developed for assessing the performance and possibly the energy efficiency of MIDs. We ported the benchmark to different platforms (operating systems and processors) to determine performance variations and OS dependencies, and we show initial results obtained by comparing different platforms.

The benchmark can be modified by adding extra workloads, modifying existing workloads, and porting to different operating systems such as Android OS. The following is a list of future work planned for the benchmark:

- Include additional workloads or modify existing workloads by reviewing updated statistics for MID typical usage.
- Port the benchmark to Android-based operating system.
- Perform power analysis to calculate performance-per-watt for each benchmark component on all platforms covered in this paper.
- Analyze performance for wireless and wired connections for MIDs.
- Enhance the navigation component with augmented reality.
- Analyze Memory utilization and cache performance for 2D and 3D games.

TMAPP is available at http://tmapp.engr.scu.edu/. It is open source and we would be glad to have it run on different platforms. You can join our consortium to be established soon.

REFERENCES

[1] MeeGo OS ; http://www.meego.com

[2] Angstrom Distribution OS ; http://www.angstrom-distribution.org/

[3] Ubuntu OS ; http://www.ubuntu.com/

[4] Beagleboard; http://beagleboard.org/

[5] ARM® CORTEX™-A8 Processor ; http://en.wikipedia.org/wiki/ARM® CORTEX™-A8_architecture

[6] INTEL® ATOM™ Processor; http://www.intel.com/products/processor/Intel® Atom™/index.htm

[7] Media Player; http://www.mplayerhq.hu/design7/news.html

[8] Media Player Codecs : http://www.mplayerhq.hu/DOCS/HTML/en/codec-installation.html

[9] ImageMagick®: http://www.imagemagick.org/script/index.php

[10] FFMPEG; http://ffmpeg.org/

[11] SHARP NetWalker MID Device image installation; http://www.sharp.co.jp/support/mit/doc/recovery.html

[12] V, Gurkhe, V; "Optimization of an MP3 decoder on ARM® CORTEX™-A8 processor", TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region, March 2004.

[13] Firefox: http://www.mozilla.com

[14] Google chrome: http://www.google.com/chrome

[15] Alexa top sites: http://www.alexa.com/topsites/

[16] Poovey, J, A, "A Benchmark Characterizaiton of EEMBC Benchmark suite", Micro IEEE, Volume 29, Issue 5, Oct 2009.

[17] E. Simmons, "The Usage Model: Describing Product Usage during Design and Development". IEEE Software, Volume 23, Issue 3 (May 2006) Pages: 34-41 ISSN:0740-7459

[18] Extreme Tux Racer 3D game; http://tuxracer.sourceforge.net/

[19] BlackJack 2D game: http://packages.ubuntu.com/karmic/gnome-blackjack

[20] Navit: http://www.navit-project.org/

[21] eSpeak: http://espeak.sourceforge.net/

[22] Festival: http://festvox.org/festival/

[23] Navit planet extractor: http://maps3.navit-project.org/download/

[24] OpenStreetMap™: http://www.openstreetmap.org/

[25] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, "Introduction to Algorithms", Page 595, The Massachusetts Institute of Technology, Second Edition, 2001

[26] National Marine Electronics Association: http://www.nmea.org/

[27] MeeGo on Beagleboard : http://wiki.meego.com/ARM/Meego_on_the_Beagle

[28] OpenOffice; http://www.openoffice.org

[29] SPEC: http://www.spec.org/

[30] MediaBench: http://euler.slu.edu/~fritts/mediabench/

[31] MinneSPEC: http://www.arctic.umn.edu/minnespec/index.shtml