Android CompCache Based on Graphics Processing Unit

Muder Almi'ani

Abdu Razaque

Saleh Atiewi

See next page for additional authors

Authors
Muder Almi'ani, Abdu Razaque, Saleh Atiewi, Mohammed Alweshah, Ayman Al-Dmour, and Basel Magableh

Android CompCache based on Graphics Processing Unit

Muder Almiani[*], Abdul Razaque[†], Shen Ziqi[§], Ge Yaqin[§], Saleh Atiewi[*], Mohammed Alweshah[‡], Ayman Al-Dmour[+], Basel Magableh[⊕]
[*]College of Information Technology, Al-Hussein Bin Talal University, Jordan
Email: malmiani@my.bridgeport.edu, Saleh@ahu.edu.jo
[†]Dept. of Computer Engineering & Telecommunication, International IT University Almaty, Email: a.razaque@iitu.kz
[§]Department of Computer Science, New York Institute of Technology, Nanjing Campus, China, Email: {zshen06,yge05}@nyit.edu
[‡]Al-Balqa Applied University, Amman, Jordan, Email: weshah@bau.edu.jo
[+]Department of Computer Science, Applied Science University, Manama, Bahrain, Email: Ayman70jo@yahoo.com
[⊕]Technological University, Dublin, Ireland, Email: basel.magableh@dit.ie

Abstract: Android systems have been successfully developed to meet the demands of users. The following four methods are used in Android systems for memory management: backing swap, CompCache, traditional Linux swap, and low memory killer. These memory management methods are fully functioning. However, Android phones cannot swap memory into solid-state drives, thus slowing the processor and reducing storage lifetime. In addition, the compression and decompression processes consume additional energy and latency. Therefore, the CompCache requires an extension.
An extended Android CompCache using a graphics processing unit to compress and decompress memory pages on demand and reduce the latency is introduced in this paper. This paper characterizes each data compression and decompression utility by measuring compression ratio, compression and decompression throughput, and energy efficiency to validate the process. Experimental results prove that data compression and decompression utilities can be beneficial to reduce the latency and perform faster compression and decompression compared with existing approaches.

***Index term*: Android CompCache; energy consumption; latency**

I. INTRODUCTION

Data compression is supposed to reduce the number of bits used to represent information [1]. The frequent and infrequent bits with short and long sequences, respectively, such as Huffman compression [2], are required to reduce the data size. Assuming that all data, emails, and other private information should be compressed lossless, using an appropriate data compression algorithm for compression and evaluating implementation quality, which can affect the performance and energy consumption, is necessary [3,4]. Many different models and coders are combined by algorithms of data compression. Additionally, utilities used for data compression and decompression can be beneficial to increase compression throughput, reduce latencies, and improve the efficiency of energy usage and storage [5,6]. Thus, this paper compares the most popular compression utilities, namely ZArchiver, Androidzip, Zip, 7Zipper, and Solid Explorer, to measure the metrics that affect the compression and decompression processes (e.g., compression ratio, compression throughputs, and efficiency). Three typical experiments, namely local, wired, and wireless, are conducted to evaluate utilities and validate the proposed idea.

This paper provides the following contributions:

- Local solves compression and decompression tasks established on mobile devices, such as Android phones.
- Wired solves compression tasks also established on mobile devices. Different from the first local experiment, data are transformed from the temporary page system to the remote server used for data communication.
- Wireless solves compression tasks established on mobile devices. This experiment differs from the second wired experiment because wireless finishes the compression tasks, in which data are transferred from remote servers to the mobile devices.
- This paper extends the CompCache to use the phone GPU to compress and decompress pages. The central processing unit (CPU) and graphics processing unit (GPU) are also compared on the basis of compression and decompression in terms of latency and/or energy consumption. However, identifying the difference between a GPU and a CPU, which is the key point to accomplish the improvement, is difficult.

The rest of the paper is organized as follows. Section II introduces the problem identification and significance. Section III discusses the related work. Section IV presents the proposed Android CompCache using GPU. Section V provides experimental results. Section VI finally concludes the entire paper.

II. PROBLEM IDENTIFICATION

Personal computers and workstations can handle energy consumptions for compression and decompression. By contrast, mobile phones experience problems due to compression and decompression because they have limited battery and memory space to handle this process. Consequently, mobile phones increase the latency and consume extended power during compression and decompression processes on data transfer.

Hence, setting up experiments to determine the metrics and then deciding the effective way of improving the processes is necessary to solve the aforementioned problem.

III. RELATED WORK

The most salient features of existing techniques are summarized and discussed in this section. Rai and Chaudhuri [5] proposed the approach that dynamically estimates the Quality of Service (QoS) level in GPU application. This approach employs a lightweight mechanism to adjust the GPU memory access rate dynamically such that the GPU can meet the required QoS level. This process frees up memory system resources that can be shifted to the co-running CPU applications. However, the proposed approach ignores the unexpected and uncertain conditions in the experiments.

Barr and Asanovi [6,7] introduced another related approach. Both approaches focus on the mobile device that can use wireless to measure energy consumption during the data compression process and also discuss the efficiency of energy usage. However, additional details on the frequency of executed instructions, the accuracy of experimental prediction, and the performance of compression are lacking. Furthermore, work in both approaches only choose a few compression testing methods. In addition, input data are limited from 1 MB text to 1 MB web data.

However, different kinds of compression, including the newest utilities, which can easily be used for compression and decompression, can be tested. Moreover, its range, which can cover additional data types, can be expanded.

Balasubramanian and Arun [8] conducted experiments and compared the compression of energy consumption in Android mobile phones. They only calculated the consumption in one process and ignored the consumption in other situations. The proposed approach in the present study involves the power consumption of the entire interaction process. Furthermore, the power consumption obtained through the power monitor is recorded and calculated. This process helps in evaluating accuracy.

Alameldeen and Adaptive [9] focused on the two features to improve compression and decompression and the complexity of running applications. They p attempted to solve both problems appropriately and facilitate efficient data compression. However, the decompression process is inaccurately addressed. By contrast, the proposed algorithm in this paper avoids the aforementioned problems and improves compression and decompression capabilities. Furthermore, channels between the temporary file system and remote servers can increase the number of data types, slightly switching from 0 to 1 or from 1 to 0, during the transfer of compressed data. The dynamic energy can also change during the transfer due to the relatively frequent capacitance of charging and discharging channels. Therefore, data compression maintains the balance between bandwidth and dynamic energy. Although bandwidth is ample, data transfer consumed at high dynamic energy must be analyzed. References [10-14] introduced a new algorithm to compress relatively unused files and store them in the memory using parallelization and vectorization techniques and resource management.

The CPU and GPU, which are adapted in the proposed model, are calculated using the proposed approach. In addition, local experiments are conducted to facilitate convenient compression tasks and reduction in the bit number. Therefore, these encoding schemes (CPU and GPU) can be used concurrently and efficiently to improve the compression and decompression processes. Furthermore, the extended CompCache enables memory swap into the physical disk.

The limitation of the proposed approach is the inclusion of the component only for solving compression tasks, especially memory. Otherwise, using the proposed approach in running applications is difficult due to a lack of memory. On the contrary, the proposed approach aims to swap memory that can be extended to compress page-cache pages. The process of compression and decompression is more efficient than swapping memory into physical disks. If the stored memory is required again, then the memory is decompressed and sent back. Thus, storage speed is improved.

IV. PROPOSED ANDROID COMPCACHE USING GRAPHICS PROCESSING UNIT

The performance and energy efficiency of compression and decompression are studied. Three functional ways are used for reducing the latencies.

- Local
- Wired
- Wireless

Several popular compression utilities, which include ZArchiver, Androidzip, Zip, 7Zipper, and Solid Explorer, are used on mobile development platforms to validate the performance of the proposed Android CompCache.

The number of samples executed on compression utilities is given by

$$n = \mathrm{C.T} \times \mathrm{SF}, \qquad (1)$$

where C.T is the total time of compressing a file, and SF is the frequency of the file.
Thus, the total energy of compression utilities (EC.T) is given by

$$\mathrm{EC.T} = \sum_{i=1}^{n} I_j * V_{PLATFORM,j} * t, \qquad (2)$$

where $\Delta t = 1 / \mathrm{SF}$, and $V_{PLAFORM,j} = V_{supply} - I_J \times R$.

$V_{PLAFORM}$ is assumed to be constant because of the ignorance of the voltage consumed by the shunt resistor to simplify the formula. The initial consumed energy EC.T(0), which is not included in the consumed energy when mobile phones go into idle, is investigated to calculate EC.T. Therefore, EC.T(0) can be calculated as follows:

$$\mathrm{EC.T}(0) = \mathrm{EC.T} - I_{idle} * V_{PLATFORM,idle} * C.T, \quad (3)$$

where $V_{PLATFORM,idle} = V_{SUPPLY} - I_{idle} * R$.

The same samples in the process of decompression tasks are calculated using the value decompression time D.T.

Calculating EC.T(I_{idle}) and ED.T(I_{idle}), which denotes the energy of the current idle(2) is easy when the total energy of compression EC.T(0) and decompression ED.T(0) is compared. The following metrics are used for the evaluation process.

- Compression ratio
- Reduced Bytes
- Energy efficiency

A. *Compression ratio*

Compression ratio is the ratio of uncompressed and compressed file that represents the level of compression. The effectiveness of compression utilities can also be realized. The compression ratio "CR" can be calculated as follows:

$$CR = \frac{SU}{SC}, \quad (4)$$

where SU is the size of uncompressed file, and SC is the size of compressed file.

B. *Reduced Bytes*

The performance of compression and decompression depends on the time and energy consumed in this process. Thus, the time of compression tasks C.T and decompression tasks D.T must be computed under the same condition of compression levels by using Android utilities. These utilities can record time and energy consumption.

The compression and decompression processes operate three times and record data to calculate the average time, which is the final time to be compared. However, the average time is not directly compared. Compression and decompression throughputs (Th.C and Th.D) counted by megabytes per second are compared.

Compression and decompression throughputs, which can be regarded as the number of reduced bytes $'R_{by}'$, represent the size of uncompressed pages sent into data transformation. These throughputs can be expressed as |SU-SC|.

$$R_{by} = (SU - SC). \quad (5)$$

Efficiency is measured to determine the throughput and help record the consumed energy in the process of compression and decompression. Moreover, compression and decompression throughputs are variables for data transfer testing. Throughput represents the capability of compression and decompression utilities. Thus, the performance of mobile devices is evaluated by using the value Th.C and Th.D.

C. *Energy efficiency*

First, a specific compression level is assumed. Then, the total energy of compression tasks (EC.T(0)) is measured by using Equation (3). The total compression energy of the idle current (EC.T(I_{idle})) can also be calculated.

where I_{idle} is equal to 0.25 or 0.5 A. Similarly, the total energy of decompression tasks is measured as the current equals the idle ((ED.T(I_{idle})). The consumed energy for every utility working in the specific compression level is measured thrice, and then average energy is calculated. However, consumed energy is not compared directly. Instead, the energy efficiency between compression and decompression (EE.C and EE.D) is differentiated.

The energy efficiency of compression and decompression represents the size of uncompressed pages used for data transfer. Moreover, energy efficiency can be regarded as the number of reduced bytes in the process of compression and decompression. Energy efficiency can be expressed as follows:

$$E_{eff} = \left(\frac{SU - SC}{EC.T}\right), \quad (6)$$

where E_{eff} is the energy efficiency.

V. EXPERIMENTAL RESULTS

The following three different experiments are conducted to calculate the three metrics introduced in the previous section.

- The first experiment is based on local. This experiment aims to focus on the performance of compression and decompression, which should calculate time and consumed energy during the process.
- The second experiment is based on wired.
- The third experiment is based on wireless.

The second and third experiments focus on the performance of utilities to calculate values of time and energy in the compression and decompression tasks.

The experimental result includes the metrics of compression and decompression tasks discussed in the previous section. Based on the conducted experiments, the following interesting results are determined.

- Overall compression ratio (Mixed networks)
- Compression and throughputs (Local)
- Compression and throughputs (Wired)
- Compression and throughputs (Wireless)

A. *Overall compression ratio*

Table 1 shows a list of used metrics, including values, explanations, and units. The solutions for these values are also presented in the table for easy reference and calculation.

TABLE 1: List of used metrics

Values	Explanation	Unit	Solution
SU	Size of uncompressed page	MB	Calculated
SC	Size of compression page	MB	Calculated
CR	Compression ratio	-	SU/SC
C.T[D.T]	Time of compression or decompression tasks	s	Calculated
T.UUP [T.UDW]	Time of uploading or downloading the original page	s	Calculated
EC.T [ED.T]	Total energy of compression or decompression tasks	J	Calculated
ET.UUP [UDW]	Total energy of uploading or downloading the original page	J	Calculated
EC.T(0) [ED.T(0)]	Total energy of compression or decompression as original current	J	$\mathrm{EC.T} - I_{idle} * V_s * C.T$ [$\mathrm{ED.T} - I_{idle} * V_s * D.T$]
Th.C [Th.D]	Compression or decompression throughput	MB/s	SU/C.T [SU/D.T]
Th.UUP [Th.UDW]	Original page of uploading or downloading throughput	MB/s	SU/T.UUP [SU/T.UDW]
EE.C [EE.D]	Compression or decompression energy efficiency	MB/s	SU/EC.T [SU/ED.T]

Figure 1 shows that the metric of the compression ratio increases following the change in compression levels. Different compression levels also have different compression ratios. Furthermore, different utilities involve corresponding compression ratios. The figure reveals that 7Zipper provides improved utilities in the compression experiment.

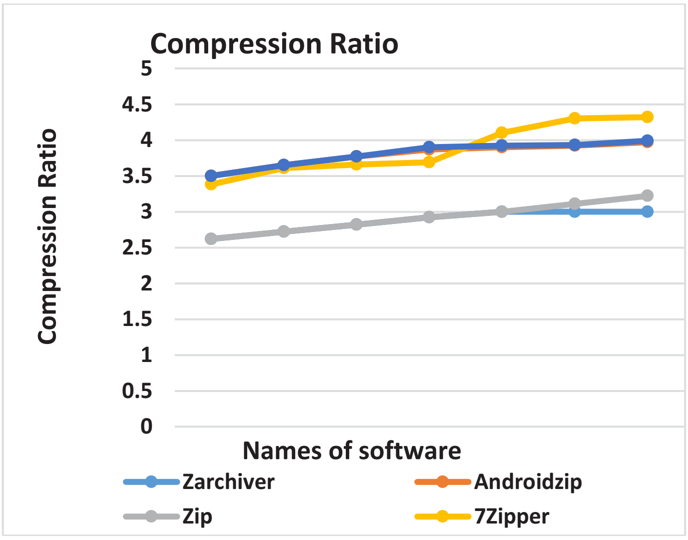

Figure 1: Overall compression ratio (CR)

B. Compression and decompression throughputs in local scenarios

Different compression utilities have different compression throughputs. The compression levels for all used utilities are related to compression throughput. This finding indicates that high compression levels cause low throughputs due to the increase in computational complexity. However, compression throughput also causes a change in magnitude. The low throughput leads to a large magnitude.

Table 2 shows the preparation for the local experiment. The experiments used ZArchiver, Androidzip, Zip, 7Zipper, and Solid Explorer to perform the compression and decompression tasks and facilitate the operation of these utilities on mobile devices, including Huawei, Oppo, Redmi Phone, Samsung, and Sony Ericsson.

TABLE 2: Preparation for the local experiment

Used Device/ Tools	Specification
Compression and Decompression	ZArchiver, Androidzip Zip, 7Zipper, Solid Explorer
Used Mobile Device	Huawei, Oppo Redmi Phone, Samsung Sony Ericsson

Figures 2 and 3 respectively show the decompression and compression throughputs in the local experiment. Different compression utilities have different compression and decompression throughputs, which are also related to varying compression levels. High compression levels can result in low compression throughputs, indicating high computational complexity. Androidzip has the highest decompression and compression throughputs under the same compression level.

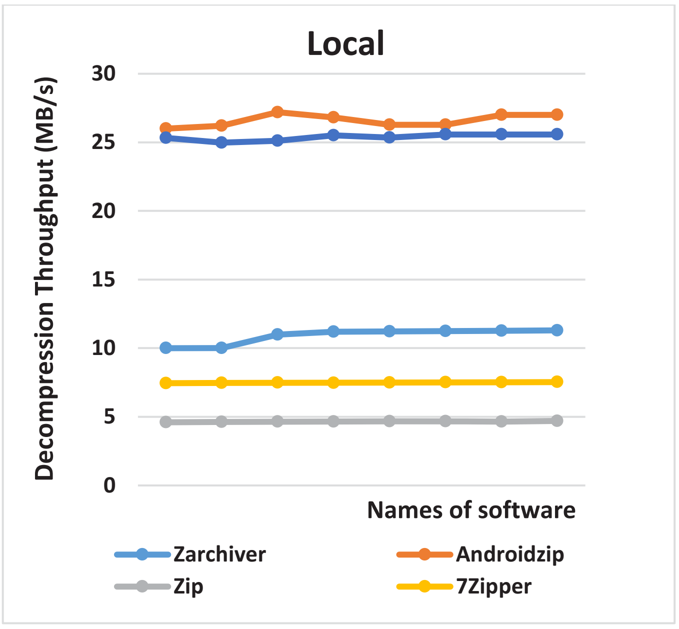

Figure 2: Decompression throughput

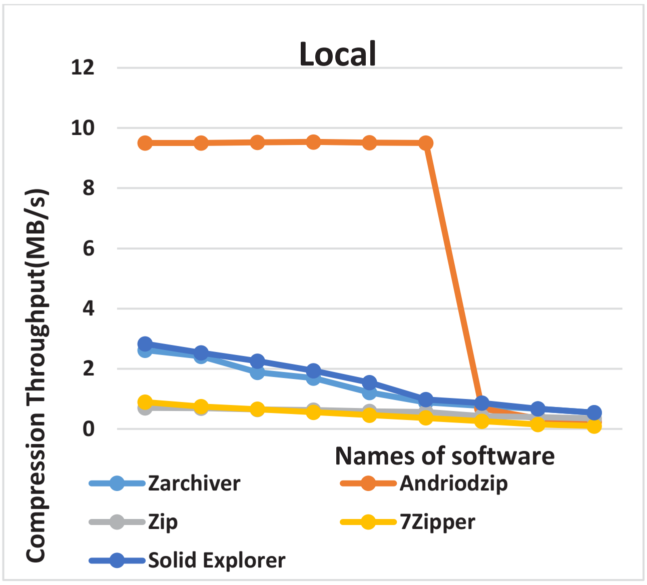

Figure 3: Compression throughput

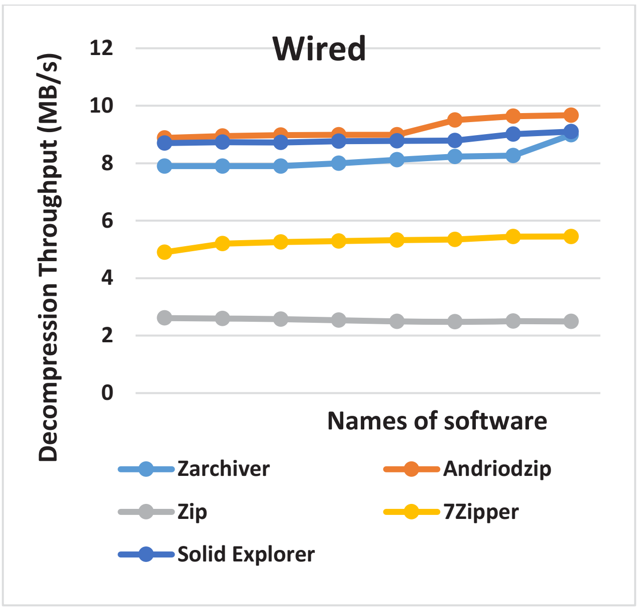

Figure 4: Decompression throughput

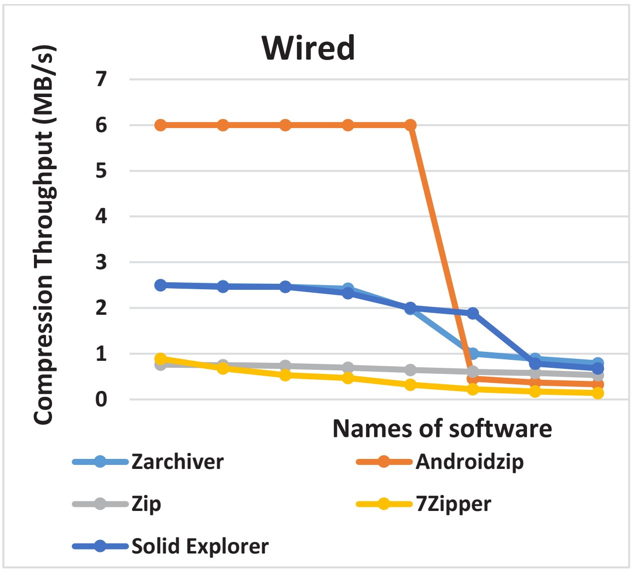

Figure 5: Compression throughput

C. Compression and decompression throughputs in wired scenario

Compression throughput is affected by the network data transfer. Thus, the throughput is smaller than the expression.

CR × Th.UUP

Figures 4 and 5 show compression throughputs in the wired experiment. The results indicate that the compression throughput is affected by remote devices, which means that throughput is always less than CR × Th.UUP. The decompression throughputs are also limited by remote devices and are always less than CR × (US/T.UDW).

D. Compression and decompression throughput in wireless scenarios

In the process of decompression tasks, used utilities can enlarge the bandwidth because wireless networks provide low throughput. Table 3 shows compression tasks, such as transferring data from the original input file into output files, and decompression tasks, such as transferring data from the compressed file into output files.

TABLE 3: Preparation for wireless experiment

Compression Tasks	Original input file (UF) Output files
Decompression Tasks	Compressed files (CF) Output files

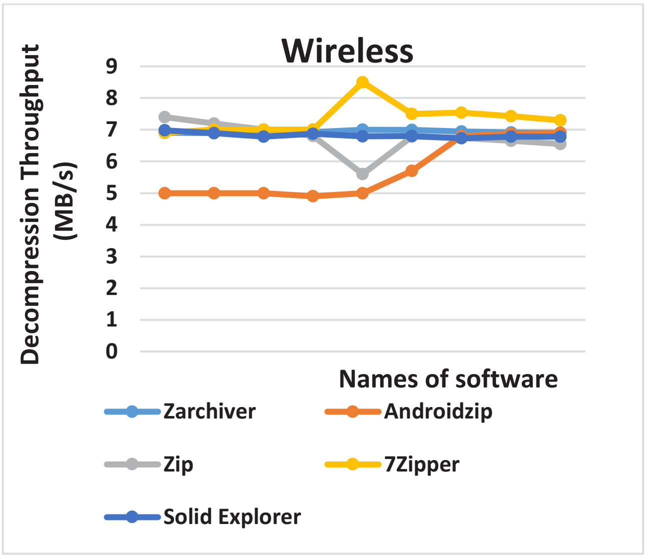

Figure 6: Decompression throughput

Figures 6 and 7 respectively show the compression and decompression throughputs in the wireless experiment. The results represent the real throughput in the process of transferring data via wireless technology. Similar to previous experiments, compression throughput is affected by wireless technology. Thus, this throughput is always less than CR × (US/T.UUP). In this experiment, compression increases the throughput of utilities. If throughput is sufficiently ineffective, then utilities should increase bandwidth.

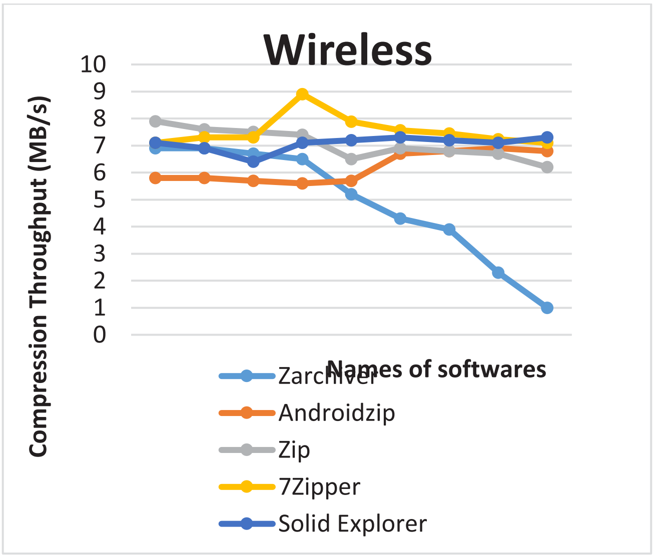

Figure 7: Compression throughput

VI. CONCLUSION

Compression and decompression utilities of ZArchiver, Androidzip, Zip, and 7Zipper are evaluated in this paper on the following three different networks: local, wired, and wireless. Furthermore, considerable attention is provided on compression ratio, performance (including time and energy), and energy efficiency in the process of compression and decompression under the control of compression levels. The results indicate that each software has different throughputs in each network. However, each software has better throughput in wireless networks than that in the two other networks.

REFERENCE

[1] Haque, Asadul, Shantanu Kode, Abdul Razaque, and Muder Almiani. "ConSec: An encryption policy for context aware security applications." In Information and Communication Systems (ICICS), 2016 7th International Conference on, pp. 96-101. IEEE, 2016.

[2] D. Huffman, "A Method for the Construction of MinimumRedundancy Codes," Proc. Ire, vol. 40, no. 9, pp. 1098–1101, Sep. 2012.

[3] Al-Rahayfeh, Amer, Abdul Razaque, Yaser Jararweh, and Muder Almiani. "Location-Based Lattice Mobility Model for Wireless Sensor Networks." Sensors 18, no. 12 (2018): 4096.

[4] BOWEN Yu, YU Zhang,and LUBIN Li " Energy measurement for mobile phone's data compression and transmission." Tianjin Municipal Science and Technology Commission under Grant No. 13ZCZDGX01098.

[5] Siddharth Rai. And Mainak Chaudhuri. "Improving CPU Performance Through Dynamic GPU Access Throttling in CPU-GPU Heterogeneous Processors." Parallel and Distributed Processing Symposium Workshops (IPDPSW). 2017 IEEE International.

[6] K. Barr and K. Asanovi, "Energy aware lossless data compression," in Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys'03), 2014, pp. 231–244.

[7] K. C. Barr and K. Asanovi, "Energy-aware lossless data compression," ACM Trans. Comput. Syst., vol. 24, no. 3, pp. 250–291, Aug. 2015.

[8] N. Balasubramanian, A. Balasubramanian, and V. Arun. "Energy consumption in mobile phones: a measurement study and implications for network applications." SIGCOMM. ACM, 2012.

[9] A.R.Alameldeen and D.A.Wood.Adaptive Cache Compression for High-Performance Processors. In ISCA-31, 2014.

[10] Nitin Gupta. " Compcache: in-memory compressed swapping." May 26, 2016.

[11] Razaque, Abdul, Syed Rizvi, Muder Almiani, and Amer Al Rahayfeh. "State-of-art review of information diffusion models and their impact on social network vulnerabilities." Journal of King Saud University-Computer and Information Sciences (2019).

[12] Razaque, Abdul, Muder Almiani, Mukazhanov Nurzhan Kakenuly, and Yaser Jararweh. "Energy Efficient Medium Access Control Protocol for Wireless Sensor Networks." In 2018 6th International Renewable and Sustainable Energy Conference (IRSEC), pp. 1-6. IEEE, 2018.

[13] Y. Jararweh, L. Tawalbeh, H. Tawalbeh and A. Moh'd, "Hardware Performance Evaluation of SHA-3 Candidate Algorithms," Journal of Information Security, Vol. 3 No. 2, 2012, pp. 69-76.

[14] Jararweh, Yaser, Mahmoud Al-Ayyoub, Maged Fakirah, Luay Alawneh, and Brij B. Gupta. "Improving the performance of the needleman-wunsch algorithm using parallelization and vectorization techniques." Multimedia Tools and Applications 78, no. 4 (2019): 3961-3977.