Middleware Power Saving Scheme for Mobile Applications

N Z Jhanjhi
School of Computing & IT, Taylor's University
Selangor, Malaysia
noorzaman.jhanjhi@taylors.edu.my

Fatimah Abdualaziz Almusalli
CCSIT, King Faisal University,
Ahsa, Saudi Arabia
f_almoslly@hotmail.com

Sarfarz N Brohi
School of Computing & IT, Taylor's University
Selangor, Malaysia
SarfrazNawaz.Brohi@taylors.edu.my

Azween Abdullah
School of Computing & IT, Taylor's University
Selangor, Malaysia
azween.abdullah@taylors.edu.my

***Abstract*—Smartphones popularity, usage and users dependency has been increased over the years. The popularity increase is linked with several factors such as smartphones size, ease in use, and several supported multipurpose apps. This all is enable due to the advanced integrated technologies in smartphones such as Wi-Fi, multi Sensors, GPS, high-speed CPU, a real world coloured display, Bluetooth, NFC etc. These capabilities attracts users and developers highly to build and join the smartphone community. Smartphones performance and functionalities are improving with time on both hardware and software side. However, power consumption is the key concern from all aspects. Rapid increase in number of apps and use is not inclined with smartphones batteries growth. Hence the demand for power saving applications increasing gradually to keep them intact. A great number of researches have been conducted to introduce the several power saving approaches. Memory data access for optimization carries a significant improvement in power consumption especially for data-intensive applications. Memory transformation, presents great optimization, such as from Array of Structure (AOS) to Structure of Array (SOA). This works well by reducing the memory access counts, which results as an overall memory access require power consumption. This research is the extended version of [20], where middleware power saving scheme was developed. This research introduces memory optimization through middleware transformation service, which converts the AOS to SOA and resulting will increase significant power saving for mobile application by reducing the memory access counts.it extended the battery life by minimizing its use.**

Keywords: ***Data layout transformation, Power Saving, Mobile Application, Array of structure AOS, Structure of arrays SOA***

I. INTRODUCTION

The smartphone due to his characteristics being smart, lighter, thinner device having promising features, have main role of social and cultural connectivity as well as a best source of storing and accessing data. The Smartphone due to his characteristics being smart, lighter, thinner device having promising features, have main role of social and cultural connectivity as well as a best source of storing and accessing data. The market share for smartphones increases by 13.0% as of 2nd quarter of 2015 [1], it becomes necessity of our daily life.

This rapid development penetrated in each and every aspect of our daily life. The patients are using the smartphones for reporting their diseases remotely in case getting advises, patients are using smartphones for taking blood pressure readings, diabetic's readings, walking steps count etc. Which can be stored in cloud and easily access and advised by the panel of the physicians. In 2014, 1:6 visits in medical health were virtual [2] which helps to save the operational and other cost for the physicians and patients as well. Online shopping, accessing information, banking etc. become norm through mobile applications [3], in addition, integrated applications of web with voice become normal [4]. Smart phones become major part of our daily life, almost everybody owns smart phones, and it is expected to have more than 80% of people by 2020 [5].

Nowadays smartphones are built on latest technology such as full HD (High Definition), Snapdragon 820, Exynos 8890 a powerful processor, finger print scanners, higher Camera resolutions up to 23MPs (Megapixel) and more [6]. However, these all are power demanding and challenges for the smartphone batteries. Battery matters mainly for smart phones, Fig. 1 describes details about battery in milli Ampere hour (mAh) for selected smart phones [7]. To address the battery life issue, a great number of scientist/researchers suggested multiple solutions, including power efficient hardware, lithium-ion battery, screen resolutions, operating system, smart applications, developing energy efficient code and applications etc. [8-9].

Smartphone memory and processing has major share for power consuming almost 15% of the overall system and this increases with usage. 7-10% consumed by SD card and RAM while performing read and write operations. [10]. There is a gap between CPU frequency and memory, to overcome this, a data transformation is required to optimize the memory [11]. The popular memory optimization transformation is AOS to SOA [12], which motivates this research to introduce Data layout transformation service.

978-1-5386-7167-2/18/$31.00 ©2018 IEEE

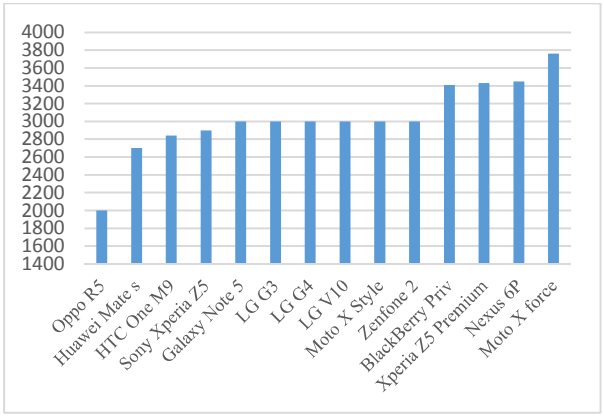


Figure 1. Most common smartphone battery sizes in mAh [7].

II. BACKGROUND

A. *The Power Consumptions in Smartphone*

The increase in the number of applications and functions with smartphones demands for more power and optimized use of existing battery at both software and hardware end. Fig.2 depicts that major battery drain of 47.9% due to the smart phone applications *[13]*, hence developing efficient power consuming applications is essential.

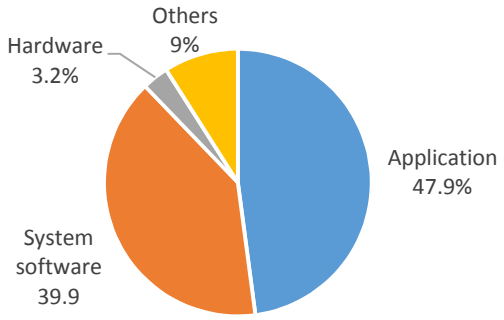


Figure 2. Component based battery drain [13].

Smart phone application developers can be assisted by providing details of power consumption at source line level. In [14] researchers introduces the method for power consumption in smart phones, the modal works efficiently almost 100%, with the combination of hardware and smart phone applications using profiling and statistical model. Power concern attracts the attentions of several researcher [15]. Energy efficient middleware and annotation language was developed by [16], it was built based on two main keys, first to delay the execution, until the device minimize the operation cost, second use an abstract model to describe desired device status. In [17] Tempus was introduces which can help to write power management policies. In [20] researchers focus on the impact of programming tools/languages used for smart phone application development, they also studied the impact of compiler optimization. Dynamic flexible resolution scaling system was introduced by [21] which help to save the power consumed through smart phone display.

Graphic Processing Unit (GPU) and CPU took major share of battery drain [22], half of the energy can be saved by controlling the resolution such as 60.5% reduction at most and 30.1% on the average for 15 applications. In [23] researchers used Dynamic Voltage and Frequency Scaling (DVFS) along User Driven Frequency Scaling (UDFS) to control the power consumption ideally for GPU and CPU. The waked technique [24] was applied by the researchers to control the power consumption especially for the smart phone components which are power hungry such as Wi-Fi. The authors focused on smart phone browser application power saving [25]. In [26] researcher concentrate on power saving techniques for live video streaming applications. Researchers in [27] focuses on Hyper Text Transfer Protocol (HTTP) request power saving by introducing an optimization method for it. A framework was introduced in [28] which enhanced 40% of energy on profile basis. In [29] researcher consider the power consumption by localization applications and GPS signals. Fig.3 describes major smart phone components based on battery drain [31]. A great number of research have been conducted targeted to save the batter life based on hardware and smart phone applications.

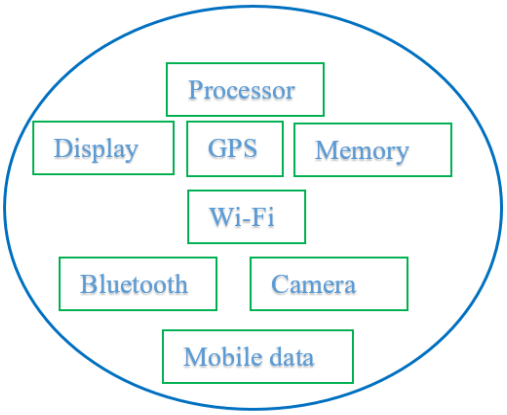


Figure 3. Major power share components of smart phone [31]

B. *Power Optimization based Hardware-Based Vs Software-*

Hardware based major components of smartphone are microprocessor, wireless network interface for Wi-Fi, storage, touch screen, memory etc. while software related major component are such as operating system, system applications, user level applications etc. Following sub section will discuss software and hardware based optimization related to the application development.

This sub section briefly discuss about hardware and software based optimization related to smartphone application development. In [21] researchers introduced several approaches/techniques to reduce the power consumption of smartphones base on different hardware alterations. Hardware based techniques are well known, as they produce better results at user and developer level. However, it limits the scope by going individual basis on each hardware component such as in case of GPS power saving, hardware optimization will lead for alteration in hardware component only and it will show results for GPS only, not for the entire smartphone. Hence, software based approach is more reliable in long term at developer end as it will work for several applications. As discuss earlier that 47.9% of the battery drain is linked with software based applications. Application not well written in code and applications with bugs drains more battery on regular basis. A complicated and illogical code written application always

utilizes more resources including memory, processor etc. and uses more share of the battery [33]. Most of the researchers focuses main power consumption components such as CPU, GPS and Wi-Fi which uses comparatively more battery.

C. AOS VS SOA

For better understanding of AOS and SOA, it is necessary to understand data layout, data layout where data could be organized accessed in the memory, when data may have multiple values as of 3D point. Data layout is extremely important for graphical or computing applications. Data layout may have high impact on the performance of application computing and development. Data layout has mainly two prominent structures such as AOS and SOA data layout. Fig. 4 a. and 4 b are the base for AOS and SOA, where in Fig 4 a represents continuous data storage in memory (AOS layout), while Fig 4 b data stores in separate area. While Fig 5 a. depicts the syntax for AOS and Fig 5 b depicts SOA.

Figure.4 AOS (a) vs. SOA (b).

```
struct S {float a;          struct S {float a[N];
          float b;                    float b[N];
          float c;                    float c[N];
};                          };
struct S myData[N];         struct S myData;
        (a)                         (b)
```

Figure 5. Syntex for AOS (a) and the SOA (b)

SOA comparatively have better locality, as algorithm doesn't require all the fields at the same time. However, AOS normally used in case if algorithm require all the fields [34]. In both of these memory access varies due to the different layout [35]. Reading code comparatively more ease with AOS and also easy to maintain. However, it is not the efficient related to the processor. While code become harder to maintain comparatively with AOS in SOA but it is more ease in code readable, and access [36], the issue in SOA is between the maintainability of the code and accessibility of the code. Memory optimization for power saving is highly important, Table 1 depicts a summary about layout transformation techniques.

TABLE 1. SUMMARY ON DATA LAYOUT TRANSFORMATION TECHNIQUES

Contributed	**Achieved Results**
Investigated power consumption & impact of memory usage [19-20].	21.70% increase in power consumption per each array access.
AoS to SoA performance was tested in multi-threaded system [37].	Better results achieved such 5% - 30%
Geometry computations were improved by converting AOS to SOA shuffle being used [38].	Using shuffle SIMD implemented, received better performance.
OpencCL program written in AOS, improved data layout. [39].	2.83 times speedup for different AoS programs.
Heterogeneous Many-Core Systems, Automatic Data Layout Transformation for [36].	Significant increase in performance of 177%
Data layout transformation approach to know where transformation technique can be more suitable [40].	Significant improvement of 49.5% on average. (cost for implementation high)
Xevolver code transformation framework, by defining a rule that could transfers AOS to SOA [41].	The Transformation increases the battery by reducing the memory hit.
Binary code analysis based transformation method [42].	Evaluating and transforming data layout transformations
Using SIMD Data Layout Templates, Transformation from AOS to SOA [43].	Transformation Increases the speed up and enhance the performance.

III. PROPOSED SYSTEM ARCHITECTURE

How data layout transformation service will work and will convert different data layouts in the memory is described in this section. The proposed transformation service will convert the written code in the form of AOS to SOA in the memory. The transformation will work based on three steps such as first is Parsing, where it will apply on the source code to change and convert its new form of code. Fig. 6 describes the working steps of proposed data layout transformation service.

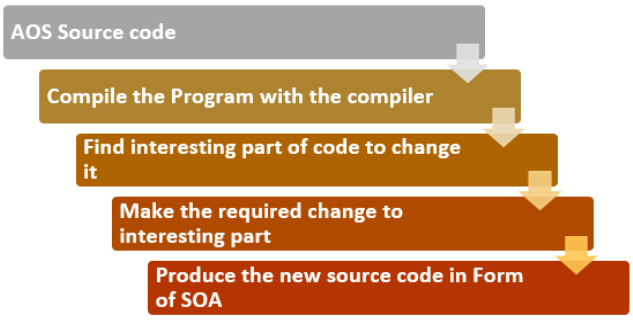


Figure. 6 Work flow for Data Layout Transformation.

Data Layout Transformation (DLT) service, will change the data layout of the available source code from AOS to SOA, where developer will required to pass his written code to the data layout transformation service. After parsing at second step transformation service will identify the code of interest where layout form could be change further in next iteration. DLT will use Abstract Syntax Tree (AST) which will help at the compiler stage to find out convertible segment of the code. At the third and final step all required changes will be done with the available code and code will be rewritten, the final code generated by the DLT tool will be able to execute by the processor and achieved data can be saved in the form of SOA. Further Fig. 7 describes conversion steps for stated operations.

Data Layout Transformation (DLT) tool achieved conversion will make processor performance even faster, fetched data from cache and memory will be used, it is also known as speculative execution (optimization technique). This will results in saving the cache, memory as well as the

bandwidth which will results the faster execution in optimized was within less amount of time, which will further leads to the less consumption of the battery in different aspects. In speculative execution focus is to avoid the delay for next round of execution. DLT approach makes it confirm that all executable data is in appropriate form where process may use most of the data smoothly for execution rather to waste resources or accept any other job partially, this organized way of data layout saves processors time, execution and power of smartphones.

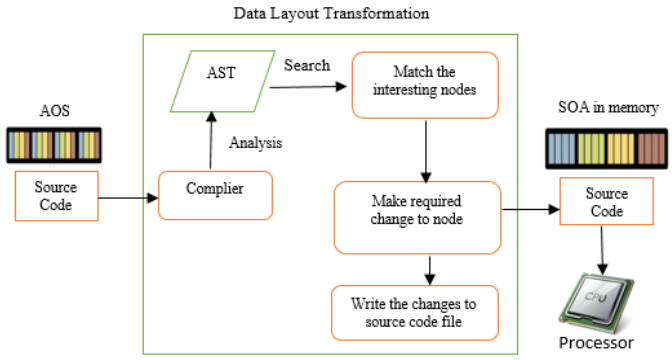


Figure.7 DLT Conversion Steps.

IV. Results and Discussions

This section presents the achieved results, Android API 19 KitKat was used for the testing purpose. In the test, we used the Central Processing Unit (CPU) having 86x architecture, equipped Supplemental Streaming SIMD Extensions (SSSE) & Advanced Vector Extensions (AVX) instruction. The results shows the better speed based on less interactions were used in case of SOA which reflects less amount of work required in case of SOA, At the same time compiler requires more efforts in term of interactions with AOS.

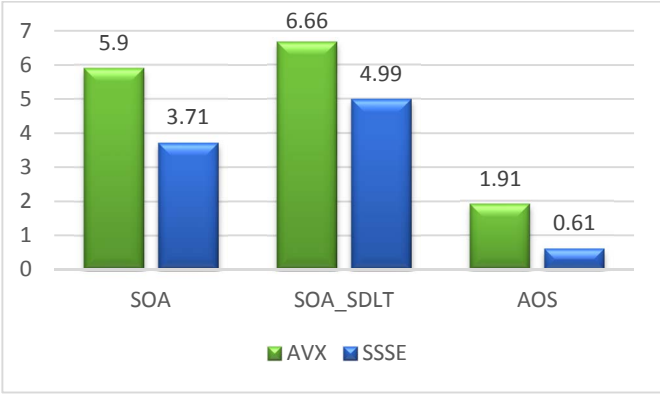
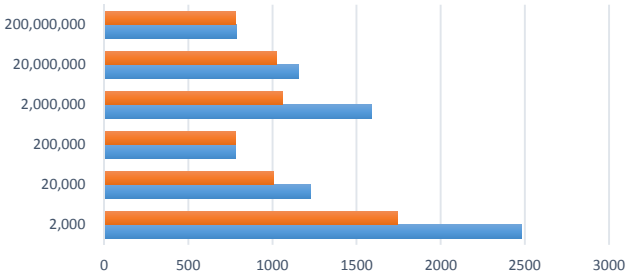


Figure 8 Speed Gain by Different data layout

An example of [54] was used to carry out the layout test of AOS, which is vectorized manually, used SOA_SDLT to compare the speed results. In the above test, at three points the code was normalized. In addition, the iterations were covered by using the loop to access the complete structure of data. In SOA_SDLT version 8.230 over 3.510 the achieved speed of array was 1000,000. The SDLT achieves significant results of 2.35 times over the AOS code. This efficient performance support the CPU in term of better use of cache, memory and better power utilization of the smartphones, which leads to the less power consuming. In the same way Fig 9 depicted more results related to the elapsed time and power consumption at the different level/scaling the array such as of array size at 2000 structure elements to 200,000,000, for different ranges of iterations and considering their average. The results shows lesser time (better) for SOA 1.28 times over AOS. Further Table no 2 describes for different precision floating point tested applications, where SOA has significant improvement over AOS which is almost 12 times. Fig. 10 describes performance comparison and significant achievement of SOA over AOS. The achieved results are for Intel processor with a great achievement of 90% in case of data layout conversion from AOS to SOA using SDLT.



	2,000	20,000	200,000	2,000,000	20,000,000	200,000,000
SOA	1741	1005	778	1060	1020	777
AOS	2482	1222	781	1585	1152	783

Figure.9 Elapsed Time for AOS and SOA SOA by the Nanosecond.

Table 2 Performance Comparison Results for Single-Precision [69]

Test	Samples	Iterations	C++ AOS Version	C++ SDLT Version	Improvement
1	100,000	2048	20.7	1.7	12 times
2	100,000	4096	41.3	3.5	11 times
3	1,000,000	2048	217.1	17.3	12.5 times
4	1,000,000	4096	451.3	34.7	13 times

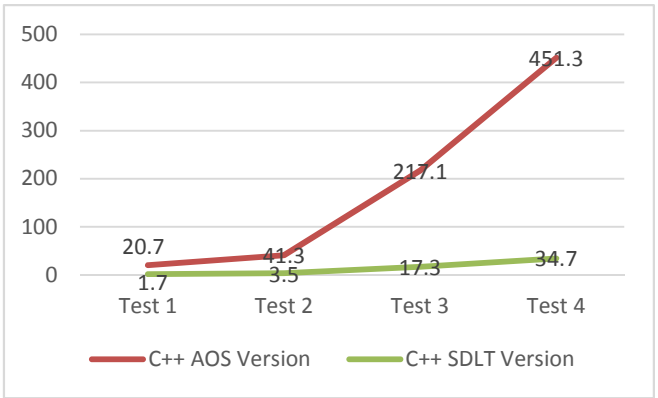


Figure 10 Performance of SOA and AOS by Seconds.

Fig.10 depicts SOA and AOS power consumption in different cases over the time. In addition, Fig. 11 shows a detailed comparison of AOS and SOA for different cases, where results shows clearly that SOA consumes less power over AOS. The Intel Acer Iconia processor was used for achieved results, where test was run for the time of 10 minutes, for the number of iterations 1000 using the array of 200000000 structure elements. The same test were checked with simple

configurations and again found that SOA uses less amount of power over AOS such as 1.14 times.

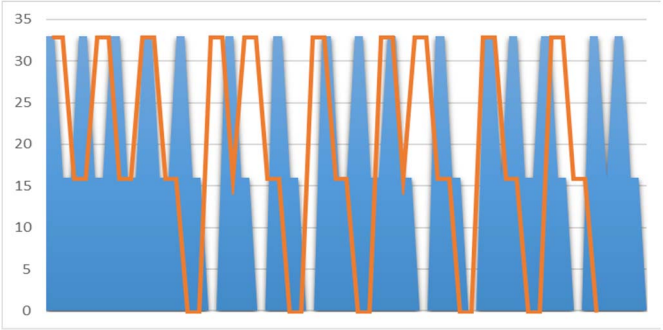


Figure 11 Power Consumption by AOS and SOA

V. Conclusions

Power saving application development becomes goal, which require more efforts at the developer end. Code handling for power saving is complex and prone to error, and misleading of apps, which may cause for the more power drain. 15% of the power consumed by the memory and processor, hence high demand of optimization is arises here. DLT is providing a great aspect of optimization, and most commonly optimization techniques using DLT is the AOS to SOA conversion. Conversion from array of structure to structure of array is time taking job. This research presented an efficient Data Layout Transformation Service (DLTS). DLTS converts the earlier written code in the form of AOS to source code which can be stored in SOA in the form of data. DLTS produced significant results using SOA layout, it decreases the memory attempts or access time, efficient in time execution at processor end, and eventually it utilizes the less amount of power and enhances the battery life. Results shows a significant improvement based on DLTS. The performance of SOA is better than AOS in different aspects such as in memory access, fastest processing instructions at COU, and power savings up to 90%. SOA using DLTS is significantly suitable for power saving technique at application level of smartphones.

References

[1] "IDC: Smartphone OS Market Share, 2015 Q2. (2015, August). Retrieved March 09, 2016, from http://www.idc.com/prodserv/smartphone-os-market-share.jsp".

[2] "Topol, E. J. (2015, January 9). The Future of Medicine is in Your Smartphone. The Wall Street Journal. Retrieved from http://www.wsj.com".

[3] "Anita Singh. (2015, August 6) Smartphones are Most Popular Means of Getting Online. The Daily Telegraph. Retrieved from http://www.telegraph.co.uk".

[4] "B. J. (2015, September 29). Can a Smartphone Be a Tool for Learning? - BBC News. Retrieved from http://Www.Bbc.Com".

[5] "Planet of The Phones. (2015, February 28). Retrieved, from http://www.economist.com/news/leaders/21645180-smartphone-Ubiquitous-Addictive-And-Transformative-Planet-Phones".

[6] "E. P. (2015, December 17). Back to The Future: What Will a Flagship from Late 2016 Look Like? Retrieved from http://www.androidauthority.com/what-to-expect-flagship-phone-2016-653616/".

[7] "G. S. (2015, December 14). Which Android Phones Charge The Fastest? Retrieved March 03, 2016, from http://www.androidauthority.com/which-phone-charges-the-fastest-660181/".

[8] "Burn-Callander, R. (2015, November 21). Is This The End of Constant Smartphone Recharging? Retrieved March 03, 2016, from http://www.telegraph.co.uk/".

[9] "S. K. (2015, April 6). Smartphones Could be Charged in 60 Seconds With New Battery. Retrieved from http://www.telegraph.co.uk".

[10] "Naik, B. A., & Chavan, R. K. (2015). Optimization in Power Usage of Smartphones. International Journal of Computer Applications, 119(18).".

[11] "Faria, N., Silva, R., & Sobral, J. L. (2013, February). Impact of data structure layout on performance. In Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on (pp. 116-120). IEEE.".

[12] "Stratton, J. A., Rodrigues, C., Sung, I. J., Chang, L. W., Anssari, N., Liu, G., ... & Obeid, N. (2012). Algorithm and data optimization techniques for scaling to massively threaded systems. Computer, (8), 26-32.".

[13] "Ma, X., Huang, P., Jin, X., Wang, P., Park, S., Shen, D. & Voelker, G. M. (2013). eDoctor: Automatically diagnosing abnormal battery drain issues on smartphones. In Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementat".

[14] "Li, D., Hao, S., Halfond, W. G., & Govindan, R. (2013, July). Calculating Source Line Level Energy Information for Android Applications. in Proceedings of The 2013 International Symposium on Software Testing and Analysis (Pp. 78-89). ACM.".

[15] "Datta, S. K., Bonnet, C., & Nikaein, N. (2012, June). Android Power Management: Current and Future Trends. in Enabling Technologies for Smartphone and Internet of Things (Etsiot), 2012 First IEEE Workshop on (Pp. 48-53). IEEE.".

[16] "Nikzad, N., Chipara, O., & Griswold, W. G. (2014, May). APE: An Annotation Language and Middleware for Energy-Efficient Mobile Application Development. In Proceedings of The 36th International Conference on Software Engineering (Pp. 515-526). ACM.".

[17] "Nikzad, N., Radi, M., Chipara, O., & Griswold, W. G. (2015, November). Managing the Energy-Delay Tradeoff in Mobile Applications with Tempus. In Proceedings of the 16th Annual Middleware Conference (pp. 259-270). ACM".

[18] "Chen, X., & Zong, Z. (2016, October). Android App Energy Efficiency: The Impact of Language, Runtime, Compiler, and Implementation. In Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communica".

[19] Noor Zaman, Fatimah Abdualaziz Almusalli, (2017) Review: Smartphone Power Consumption & Energy Saving Techniques", in IEEE Innovations in Electrical Engineering and Computational Technologies (ICIEET), April 2017 Karachi Pakistan".

[20] "Fatimah Abdualaziz Almusalli, Noor Zaman and Raihan Rasool (2017) "Energy Efficient Middleware: Design and Development for Mobile Applications", in 19th IEEE International Conference on Advance Communication Technology ICACT 2017, February 2017 Korea".

[21] "He, S., Liu, Y., & Zhou, H. (2015, September). Demo: Optimizing Smartphone Power Consumption through Dynamic Resolution Scaling. In Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (pp. 185-187). ACM.".

[22] "Rumi, M. A., & Hasibul Hasan, D. M. (2015, September). CPU Power Consumption Reduction in Android Smartphone. In Green Energy and Technology (ICGET), 2015 3rd International Conference on (Pp. 1-6). IEEE.".

[23] "Ahmad, E., & Shihada, B. (2015, October). Green Smartphone Gpus: Optimizing Energy Consumption Using Gpufreq Scaling Governors. In Wireless and Mobile Computing, Networking and Communications (Wimob), 2015 IEEE 11th International Conference On (Pp. 740-74".

[24] "Shah, M. A., Naheed, N., & Zhang, S. (2015, September). Energy Efficiency in Smartphones: A Survey on Modern Tools and

Techniques. In Automation and Computing (ICAC), 2015 21st International Conference on (Pp. 1-6). IEEE".

[25] "Zhao, B., Hu, W., Zheng, Q., & Cao, G. (2015). Energy-Aware Web Browsing on Smartphones. Parallel and Distributed Systems, IEEE Transactions on, 26(3), 761-774.".

[26] "Rajaraman, S. V., Siekkinen, M., & Hoque, M. A. (2014, September). Energy consumption anatomy of live video streaming from a smartphone. In Personal, Indoor, and Mobile Radio Communication (PIMRC), 2014 IEEE 25th Annual International Symposium on (pp. 201".

[27] "Li, D., Lyu, Y., Gui, J., & Halfond, W. G. (2016, May). Automated Energy Optimization of HTTP Requests for Mobile Applications. In Proceedings of the 38th International Conference on Software Engineering (ICSE).".

[28] "Datta, S. K., Bonnet, C., & Nikaein, N. (2014, August). Self-Adaptive Battery and Context Aware Mobile Application Development. In Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International (Pp. 761-766). IEEE.".

[29] "Tawalbeh, L. A., Basalamah, A., Mehmood, R., & Tawalbeh, H. Greener and Smarter Phones for Future Cities: Characterizing the Impact of GPS Signal Strength on Power Consumption".

[30] "Saha, S., Chatterjee, S., Gupta, A. K., Bhattacharya, I., & Mondal, T. (2015, December). TrackMe-a low power location tracking system using smart phone sensors. In Computing and Network Communications (CoCoNet), 2015 International Conference on (pp. 457-4".

[31] "Carroll, A., & Heiser, G. (2010, June). An Analysis of Power Consumption in a Smartphone. In USENIX annual technical conference (Vol. 14).".

[32] "Alam, F., Panda, P. R., Tripathi, N., Sharma, N., & Narayan, S. (2014, March). Energy Optimization in Android Applications Through Wakelock Placement. in Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014 (Pp. 1-4). IEEE.".

[33] "Pathak, A., Jindal, A., Hu, Y. C., & Midkiff, S. P. (2012, June). What is Keeping my Phone Awake? Characterizing and Detecting No-Sleep Energy Bugs in Smartphone Apps. In Proceedings of The 10th International Conference on Mobile Systems, Applications, an".

[34] "Mei, G., & Tian, H. (2016). Impact of data layouts on the efficiency of GPU-accelerated IDW interpolation. SpringerPlus, 5(1), 1.".

[35] "Strzodka, R. (2011). Abstraction for AoS and SoA Layout in C+. GPU Computing Gems Jade Edition, 429.".

[36] Tseng, Y. Y., Huang, Y. H., Lai, B. C. C., & Lin, J. L. (2014). Automatic Data Layout Transformation for Heterogeneous Many-Core Systems. In Network and Parallel Computing (pp. 208-219). Springer Berlin Heidelberg..

[37] "Stratton, J. A., Anssari, N., Rodrigues, C., Sung, I. J., Obeid, N., Chang, L., ... & Hwu, W. M. (2012, May). Optimization and architecture effects on GPU computing workload performance. In Innovative Parallel Computing (InPar), 2012 (pp. 1-10). IEEE.".

[38] "P., & (Intel), S. M. (2010, September 27). 3D Vector Normalization Using 256-Bit Intel® Advanced Vector Extensions (Intel® AVX). Retrieved May 09, 2016, from https://software.intel.com/en-us/articles/3d-vector-normalization-using-256-bit-intel-advanced-ve".

[39] "Kofler, K., Cosenza, B., & Fahringer, T. (2015). Automatic Data Layout Optimizations for GPUs. In Euro-Par 2015: Parallel Processing (pp. 263-274). Springer Berlin Heidelberg.".

[40] "Li, P. Y., Zhang, Q. H., Zhao, R. C., & Yu, H. N. (2015, June). Data layout transformation for structure vectorization on SIMD architectures. In Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th".

[41] "Yamada, T., Hirasawa, S., Takizawa, H., & Kobayashi, H. (2015, December). A Case Study of User-Defined Code Transformations for Data Layout Optimizations. In 2015 Third International Symposium on Computing and Networking (CANDAR) (pp. 535-541). IEEE.".

[43] "Haine, C., Aumage, O., Petit, E., & Barthou, D. (2014). Exploring and Evaluating Array Layout Restructuring for SIMDization. In Languages and Compilers for Parallel Computing (pp. 351-366). Springer International Publishing.".

[43] "Reinders, J., Jeffers, J., & Sodani, A. (2016). Intel Xeon Phi Processor High Performance Programming Knights Landing Edition..".