A Workload Characterization for the Internet of Medical Things (IoMT)

Ankur Limaye and Tosiron Adegbija
Department of Electrical & Computer Engineering
University of Arizona, Tucson, AZ, USA
Email: {ankurlimaye, tosiron}@email.arizona.edu

***Abstract*—We perform an extensive study of medical applications that will potentially execute on the Internet of Medical Things (IoMT), from an edge computing perspective. Using this study, we perform a workload characterization of potential IoMT applications and explore the microarchitecture implications of these applications. Our study includes workloads spanning a variety of medical applications including medical image processing algorithms, inverse Radon transform, and implantable heart monitors. We compare these workloads' characteristics to an existing embedded systems benchmark suite, MiBench, to reveal their differences and similarities. The analysis presented herein will enable the study and design of right-provisioned microprocessors for the IoMT, and provide a framework for studying the execution characteristics of workloads in other emerging Internet of Things application domains.**

***Index Terms*—Internet of Things, edge computing, Internet of Medical Things, right-provisioned microprocessors, low-power embedded systems, workload characterization, medical devices, healthcare.**

I. Introduction

The Internet of Things (IoT) is an exciting and transformative emerging technology trend that is gaining popularity in several application domains. In the IoT, a group of smart devices, comprised of sensors, actuators, and computation, can communicate and collaborate with each other for data acquisition, visualization, and use, without the need for human intervention [1]. The IoT has been described as a technology trend that will potentially transform life, business, and the global economy [2]. Gartner Inc. has projected that the IoT will include 26 billion installed units and will be a $300 billion industry by 2020 [3].

The IoT offers a wide range of application possibilities that span several domains, including personal, smart environment (home or office), transportation and logistics, and healthcare [4], [5]. One key domain that will be strongly impacted by the IoT is healthcare [6]. The application of the IoT in the medical/healthcare domain is referred to as the *Internet of Medical Things (IoMT)*, or healthcare IoT. The IoMT is a network of connected medical devices and applications whose purpose is to provide better and more personalized healthcare services. The IoMT is gaining traction due to the emergence of innovative products, such as portable ultrasound and magnetic resonance imaging (MRI) devices, wearables, which will enable the streamlining of medical processes, such as medical diagnostics, remote patient tracking and monitoring, etc.

Similar to other application domains, the IoMT is also prone to the concomitant overheads of the IoT. As devices increase on the IoMT, the resulting increase in acquired/transmitted data will pose significant bandwidth and latency overheads. In a medical diagnostics use case, for example, the IoMT could scale to a network of several portable medical devices that gather and transfer data to medical personnel for data analysis. Transmitting this data could potentially cause a bandwidth bottleneck in the communication network, and pose challenges for real-time scenarios, where the application latency must adhere to stringent deadlines (e.g., in diagnostic emergencies).

To address these challenges, *edge computing* is currently emerging as a viable IoT computing paradigm [7], [8]. Rather than transmitting the gathered data to a high performance central head node for data visualization and analytics, the edge devices (e.g., portable medical devices) are equipped with sufficient computational capabilities and algorithms to process and interpret the generated data. Thus, only actionable information, or significantly reduced data, is transmitted. This results in a reduction in the network bandwidth and data storage requirements, and faster data analysis at the head node.

However, a key challenge in IoMT edge computing is understanding the necessary computational capabilities and microprocessor architectures that satisfy IoMT applications' execution requirements, while adhering to the devices' resource constraints. To understand the microarchitectures that will support IoMT edge computing, we must first understand the execution characteristics of the applications that will potentially execute on IoMT devices.

In this work, we perform an extensive study of potential medical applications for the IoT, from an edge computing perspective. Using our study, we present seven potential IoMT applications and algorithms, and characterize these applications' execution characteristics with respect to computational and memory requirements. The applications span a variety of medical applications including medical image processing algorithms, inverse Radon transform, and implantable heart monitoring algorithms. We analyze the applications using the Raspberri Pi 3 [9], a common IoT prototyping platform, and Linux-based performance profiling tools. We show that the IoMT applications exhibit a variety of characteristics that necessitate different computational resources. Finally, we motivate the need for a new benchmark suite targeted towards the IoMT by comparing the IoMT applications with MiBench [10], a popular open-source embedded systems benchmark suite. Our analysis show that MiBench applications' characteristics differ from the IoMT workloads, suggesting that MiBench applications are insufficient for driving the design of future IoMT microprocessors.

II. Related Work

In order to enable the growth of the IoT, much prior research [4], [5] has focused on understanding and discovering insights into its various aspects. While most prior emphasis has been placed on software and communication aspects of the IoT, we must also understand the IoT edge nodes' processing requirements and workload execution characteristics, in order to develop right-provisioned microarchitectures. However, the vast number of potential applications and application domains makes it impractical to consider every application domain in workload characterization [8].

Several benchmark suites have been developed to enable workload characterization for different application domains. Some of the most popular ones include MiBench [10], PARSEC [11], EEMBC [12], SPEC2006 [13]. These benchmarks cater to a variety of application domains and architecture analysis. For example, PARSEC targets parallel multi-threaded and shared memory applications, and SPEC targets high performance desktop/server applications. Perhaps, most relevant and accessible to IoT microarchitecture research is the open-source MiBench, which consists of workloads representing various embedded system application domains.

Very few workloads or benchmark suites currently exist for microarchitecture studies that specifically target IoT applications. ImpBench [14] provides a collection of applications targeted towards processors used in medical implants, including compression, encryption, data integrity checks, motion detection algorithms, and drug delivery and monitoring algorithms. IoTAbench [15] was developed to target IoT Big Data scenarios, specifically focusing on a smart metering use case. Citybench [16] was developed to evaluate the resource description framework (RDF) stream processing engines in smart city applications and with smart city data streams. To the best of our knowledge, there is no existing work that studies workload execution characteristics for the IoMT. The absence of such a study is a critical gap, since the medical domain is one of the most potentially transformative IoT use cases [4].

To address this gap, our work represents a first step in the direction of understanding IoMT applications, from the context of edge computing. While the workloads studied herein do not exhaust medical applications, the workloads have been carefully selected to provide a solid foundation for future studies into IoMT application characteristics. To motivate our studies further, we have compared the IoMT workloads to MiBench. We chose MiBench since it is open-source and represents the embedded systems domain. We present analysis to show that IoMT applications exhibit disparate execution characteristics from MiBench, and necessitate a new set of workloads for IoMT-targeted microarchitecture research.

III. IoMT Workloads

One of our key goals was to select a set of applications that broadly represent emerging IoMT applications. Based on extensive surveys, and interactions with medical personnel, we selected a tractable set of seven applications that are common to a wide range of medical applications or considered to be high-impact for the IoMT. Specifically, the workloads are: *QRS detection in ECG, blood pressure monitoring, inverse Radon transform, histogram equalization, K-means clustering, advanced encryption standard (AES),* and *Lempel-Ziv compression*. Since we approach our studies from an edge computing perspective, two of the seven workloads represent compression and security applications; we envision that these two workloads will be common to most emerging IoMT devices.

All the workloads/algorithms considered are open-source and written in C++; they can be compiled easily on Linux-based operating systems[1]. In this section, we briefly describe the considered IoMT workloads.

A. QRS Detection in ECG

The QRS detection algorithm (*sqrs*) [17] is used from the PhysioNet [18] repository. The PhysioNet repository consists of a large database of physiologic signals and related open-source software. The algorithm detects the QRS peaks and troughs in an Electrocardiogram (ECG) signal. The algorithm uses only the signal 0 of the ECG for the QRS detection, and is optimized for adult human ECG. The input to this workload is a test ECG signal in PhysioNet's Data bank. The application outputs an annotation file with labeled detected QRS points.

B. Blood Pressure Monitoring

We also selected the blood pressure monitoring algorithm (*wabp*) from the PhysioNet repository. This algorithm calculates the arterial blood pressure (ABP) from a continuous ABP signal. The algorithm analyzes the first derivative of the ABP signal to determine the blood pressure. The workload takes as input ABP signals (available in the PhysioNet data bank), and outputs a blood pressure log file.

C. Histogram Equalization

Histogram equalization (*imgHist*) is an image processing technique to increase the contrast of the input image. It is a necessary pre-processing step for digital X-ray, medical ultrasound and the MRI images. The histogram equalization method stretches the dynamic range of pixel values; thus, areas in the image with low contrast get a higher contrast. The workload takes a grayscale image as input, and outputs a grayscale image with better contrast.

[1]The workloads and inputs are publicly available at: http://ece.arizona.edu/~tosiron/downloads.php

D. *Inverse Radon Transform*

The inverse Radon transform (*iRadon*) [19] is used in Computerized Tomography (CT) Scan and Magnetic Resonance Imaging (MRI) to reconstruct images for non-invasive examinations of the human body. The iRadon workload used herein reconstructs a 2D sinogram input data—a sinogram is used to visualize abnormal opening (sinus) in the body—to output images.

E. *K-means Clustering*

K-means clustering (*kmeans*) is one of the widely used algorithms for medical image segmentation [20]. Region based image segmentation is used to label the different organs inside an MRI brain scan for better diagnostics. The k-means algorithm partitions different parts of the image based on pixel intensities into clusters. In our workload, each pixel is clustered into a group depending on the nearest mean values. The workload takes as input a grayscale image, and outputs a text file with the pixel locations grouped together in 10 bins.

F. *Advanced Encryption Standard (AES)*

Medical device security is a critical issue in the healthcare industry [21], making security an essential application for medical devices. For our IoMT workloads, we have selected Advanced Encryption Standard (AES) [22] for data encryption. The AES algorithm, which supports a block length of 128 bits and key lengths of 128, 192 and 256 bits, was obtained from the Crypto++ Library [23]. In our workload, we use a default block length of 128 bits and key length of 128 bits. The workload takes as input a text file, and generates an encrypted file. The workload then decrypts the generated file and validates it by comparing with the original text file.

G. *Lempel-Ziv Compression*

Even though edge computing reduces the amount of transmitted data, data compression is still critical in IoMT devices for reducing the bandwidth requirements of data transmission. Lempel-Ziv Compression algorithm [24] (*lzw)* is attractive for the IoMT because of its efficiency, losslessness, and ease of implementation. The workload takes as input a text file, and generates a compressed output file. For validation, the workload decompresses the generated output and compares the result with the original input file.

IV. Evaluation Methodology

To analyze our workloads' execution characteristics, we used the Raspberry Pi 3—a common IoT prototyping platform [9]. We used an actual IoT prototyping platform, instead of simulations, in order to derive insights on how well provisioned current microprocessor architectures are for IoMT workloads' characteristics on a real device. The Raspberry Pi 3 board features a powerful motherboard, input and output peripheral connections with USB, LAN, and WiFi connectivity. The board is also priced cheaper than most other boards with similar functionality, making it very attractive for IoT devices.

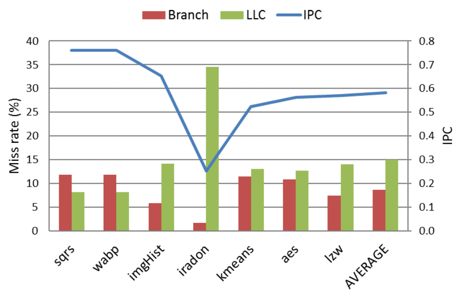

Fig. 1. IPC analysis. The figure also shows the branch and LLC miss rates to illustrate the impacts of these characteristics on the IPC

The Raspberry Pi 3 Model B, which we used, features a Broadcom BCM2837 SoC, equipped with a 1.2 GHz 64-bit quad core ARM Cortex-A53 CPU. The CPU has 32 kB Level 1 caches (separate 16 kB instruction and data caches) with 2-way set associativity, 512 kB Level 2 cache, and 1 GB LPDDR2 RAM. We installed the Ubuntu Mate 16.04 LTS operating system with Linux kernel 4.4.38-v7+ on the board. For instrumenting the hardware performance counters, we used the *perf_events* Linux utility. We compiled the IoMT workloads using gcc without any optimization flags. We ran each IoMT workload to completion 50 times, to gather execution statistics, and calculated the mean and standard deviations to measure the variability between different runs. We compared the IoMT workloads to MiBench, to represent existent embedded systems benchmarks, using the same experimental setup and methodology.

V. IoMT Workload Characteristics

Table I presents an overview of the IoMT workloads and their characteristics, including the instruction count, number of cache references, branch instructions, and the instructions per cycle (IPC). The *sqrs* and *wabp* workloads' characteristics were very similar, since they both worked on similar data inputs. Compared to the other workloads, *iradon* was the most compute and memory intensive, as evidenced by the low IPC and high number of cache references. In general, we also observed that *aes'* and *lzw*'s execution requirements did not supersede those of the functional workloads. This observation is critical for ensuring that compression and security do not incur significant runtime overheads or significantly reduce resource availability for functional workloads. The rest of this section details the workloads' execution characteristics.

A. *IPC Analysis*

The instructions per cycle (IPC) provides insight into a workload's compute-intensity, and is a measure of hardware performance for a workload. The IPC is affected by the workload's instruction mix and microprocessor hardware characteristics, e.g., pipeline depth, execution paradigm (in-order vs out-of-order), branch prediction accuracy, cache configurations, etc.

Fig. 1 shows the IPC for different IoMT workloads. We also investigated the impact of branch and last level cache (LLC) miss rates—the L2 cache was our LLC—on the IPC.

TABLE I
OVERVIEW OF IoMT WORKLOAD CHARACTERISTICS

Workload	Description	Instruction Count	Cache References	Branch Instructions	IPC
sqrs	QRS Detection in ECG	18,581,266	6,146,785	1,216,416	0.760
wabp	Blood Pressure Monitor	18,561,495	6,146,785	1,216,196	0.760
imgHist	Histogram Equalization	6,886,959	2,989,541	556,070	0.653
iradon	Inverse Radon Transform	174,419,063	58,586,880	12,927,117	0.251
kmeans	k-means Clustering	4,545,424	1,819,155	434,851	0.523
aes	Advanced Encryption Standard	3,471,322	1,240,650	327,100	0.562
lzw	Lempel-Ziv-Welch Compression	5,671,513	2,135,652	636,136	0.570

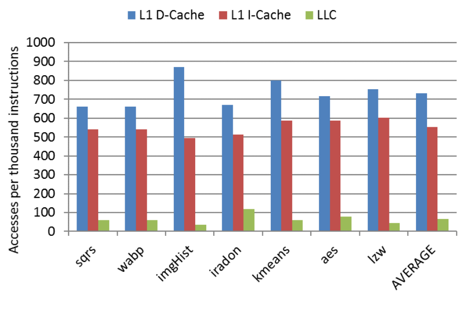

Fig. 2. Cache accesses

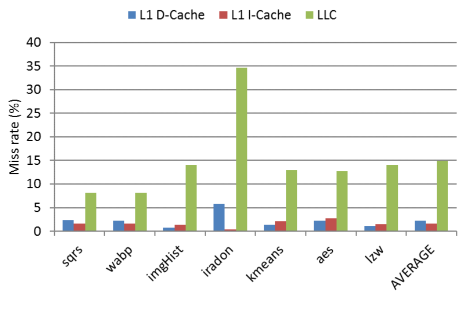

Fig. 3. Cache miss rates

High miss rates could result in pipeline stalls, which in effect, also reduce the IPC. The *sqrs* and *wabp* workloads had the highest IPC of 0.76, while iradon had the lowest IPC of 0.251. This can be attributed to *iradon*'s high memory intensity, for which the memory hierarchy was were not sufficient (details in Section V-B). *Iradon*'s memory intensity resulted in a high LLC miss rate of 34.6% and more pipeline stalls than the other workloads. On the other hand, *wabp*'s LLC miss rate was 8.2%, which is relatively low for the LLC (LLC miss rates are typically much higher than L1 cache miss rates, since most of an applications' spatial and temporal locality is exploited in the L1 cache).

We also observed that the LLC miss rate had a higher impact on IPC than branch miss rates for the workloads. For example, *kmeans'* branch miss rate of 11.4% was similar to *wabp*'s and *sqrs'* branch miss rate of 11.8%. However, *kmeans'* LLC miss rate was slightly higher (13.0% vs. 8.2%), resulting in a lower IPC of 0.523 as compared to *wabp*'s and *sqrs'* IPC of 0.76—a 31% reduction in IPC.

B. Memory Characteristics

We analyzed the IoMT workloads' memory characteristics using the cache accesses per thousand instructions (pti) and the cache miss rates. Fig. 2 and 3 depict the cache accesses per thousand instructions and cache miss rates, respectively. We analyzed these characteristics for all the caches in the Raspberry Pi 3: L1 instruction cache (L1 I-Cache), L1 data cache (L1 D-Cache), and L2 Cache (LLC).

1) Cache accesses: Fig. 2 depicts the cache accesses pti, which illustrates how much the workloads stress the caches. Across all the workloads, the average L1 I-Cache, L1 D-Cache and LLC were 552.034, 732.952 and 65.284 accesses per thousand instructions, respectively. *imgHist* had the highest L1 D-Cache accesses pti of 869.033, resulting from the large input data. *Kmeans* and *lzw* also had higher than average L1 D-Cache accesses pti of 799.726 and 752.714, respectively; the high number of accesses for these two workloads, however, resulted from the large amount of data generated during execution, not from large data inputs. On the other hand, *lzw* had the highest L1 I-Cache accesses pti of 603.378, while *imgHist*'s value was the lowest at 494.784. For most of the applications, the LLC accesses pti were relatively low—the lowest was 35.016 for *imgHist*—except for *iradon*, which had the highest LLC accesses pti at 118.225.

2) Cache miss rates: Fig. 3 depicts the cache miss rates, which provide a measure of cache performance. Given a cache configuration, the cache miss rate provides insight into a workload's cache resource requirement. Across all workloads, the average L1 I-Cache, L1 D-Cache and LLC miss rates were 1.643%, 2.288%, and 14.953%, respectively. *Iradon*'s L1 I-Cache miss rate was the lowest at 0.356%, while *aes* and *kmeans* had the highest I-Cache miss rates at 2.687% and 2.176%, respectively. On the other hand, *iradon* had the highest L1 D-Cache and LLC miss rates at 5.852% and 34.6%, respectively. These results suggest that even though *iradon* has a low instruction cache requirement, for which the Raspberry Pi 3's 16 kB cache is sufficiently provisioned, a larger L1 data cache would be required to satisfy its data needs. The L1 D-Cache and LLC miss rates for *iradon* are 155.76% and 131.40% above average, respectively. These characteristics suggest that the edge device executing the *iradon* application should either have a significantly larger data cache configuration. Alternatively, the edge device could offload the execution to a central processor capable of providing better performance.

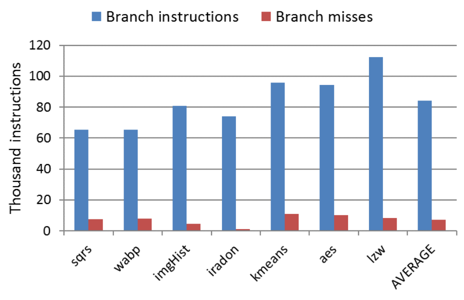

Fig. 4. Branch instructions

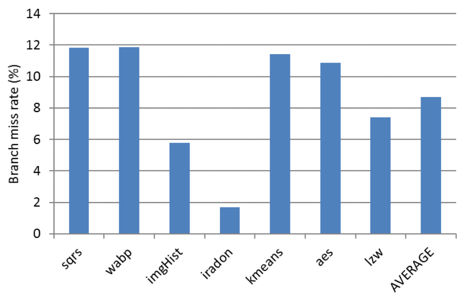

Fig. 5. Branch miss rates

C. Branch Characteristics

Branches, otherwise known as control instructions, could be a major limiting factor to achieving optimal performance, especially in pipelined microprocessors. Thus, branch predictors are important for improving the flow of instructions in the pipeline. We analyzed the IoMT workloads' branch characteristics as a function of the number of branch instructions per thousand instructions (pti) and the branch miss rate. The branch instructions pti is an architecture-independent characteristic that illustrates the frequency of branches and provides insight into the instruction mix. The branch miss rate illustrates how sufficient the branch prediction resources are for the executing application. For example, the ARM Cortex A53, which is featured in the Raspberry Pi 3, has a single-entry branch target instruction cache, a 256-entry branch target address cache, and uses a global branch predictor, with a 3072-entry pattern history prediction table [25].

Fig. 4 depicts the branches per thousand instructions. On average across all the workloads, there were 84 branches pti, representing an 8.4% frequency of branches, and 7.272 branch misses pti. The relatively few misses per thousand instructions suggests that the workloads have a large number of integer ALU operations. We also observed that there was not much variation in branch frequency among the different workloads. For example, *lzw* had the highest frequency at 112.163 branches pti, representing an 11.2% frequency, while *sqrs* had the lowest frequency at 65.465 branches pti, representing an 6.5% frequency.

Fig. 5 depicts the branch miss rates. On average across all the workloads, the average branch miss rate is 8.692%, with miss rates as low as 1.678% for *iradon*, and as high as 11.86% for *wabp*. Typically, the branch miss rates also reflect the trend towards conditional branches vs loops in the workload's instruction mix. Conditional branches are usually more difficult to predict, and result in higher miss rates, than loops, especially loops with a large number of iterations. For example, *iradon*'s low branch miss rate suggests that the workload contains more loops than conditional branches, unlike *wabp* which has a fairly high miss rate due to a higher number of conditional branches than loops.

D. Comparison with MiBench

To evaluate the similarity or disparity between currently existent embedded systems benchmarks and IoMT workloads, and the appropriateness of these benchmarks for IoMT microarchitecture research, we compared the IoMT workloads to MiBench [10]. From the MiBench suite, we used: *basicmath, bitcount, qsort, susan, jpeg, typeset, dijkstra, patricia, stringsearch, blowfish, sha, adpcm, crc32, fft,* and *gsm,* all with the large input data sets.

Table II depicts the comparison of hardware characteristics between MiBench and IoMT workloads. For brevity, we only show the means and standard deviations for the different characteristics. In addition, we used a statistical method—the Wilcoxon rank-sum test [26]—to quantify the similarity of the MiBench and IoMT workloads. Using this method for each execution characteristic considered, we calculated the *p-value*, which is a commonly used statistical significance value for evaluating the similarity between two data sets. A *p-value* of less than 0.05 indicates that the data sets are substantially dissimilar [27].

As seen in Table II, nine of the sixteen characteristics were substantially different between IoMT workloads and MiBench. Specifically, we observed that the MiBench performed much better on the Rasperry Pi 3 than the IoMT workloads. The MiBench workloads' mean IPC was 1.6X better than the IoMT workloads' IPC (0.938 vs 0.583). In addition, the memory characteristics were significantly different, especially with respect to the instruction and data caches. For example, the IoMT workloads' average instruction and data cache miss rates were 2X and 5.6X higher than than the MiBench workloads. These results reveal that the workloads differ significantly, and new workloads are required for researching new microarchitectures for the IoMT.

VI. Conclusion and Future Work

In this work, we characterized the execution behavior of potential Internet of Medical Things (IoMT) workloads, and discussed some their microarchitecture implications. Our comparisons with MiBench, a general purpose embedded system benchmark suite, revealed that the IoMT applications differ significantly from MiBench. These analyses motivate the need for new benchmark suites for IoMT microarchitecture research. Our studies also suggest that further research is needed in other IoT application domains to evaluate the workloads' execution characteristics. These workload characterizations are necessary for developing microarchitectures that sufficiently meet the workloads' resource requirements, while incurring minimal overheads.

TABLE II
COMPARISON OF IOMT WORKLOADS WITH MIBENCH

Hardware Characteristics	IoMT Workloads		MiBench Workloads		Wilcoxon Rank-sum Test
	Mean	Std. Dev.	Mean	Std. Dev	p-Value
Instruction Count	33162434.571	57972752.015	849213126.125	998664983.475	0.0019
IPC	0.583	0.161	0.938	0.251	0.0019
Branch instructions pti	83.986	16.157	89.573	35.141	0.7637
Branch misses pti	7.272	3.089	5.672	3.918	0.4423
Branch miss rate	8.692	3.612	6.755	4.066	0.3671
L1 D-Cache accesses pti	732.952	73.382	700.221	229.044	0.8151
L1 D-Cache misses pti	16.041	9.993	3.378	5.981	0.0009
L1 D-Cache miss rate	2.288	1.558	0.407	0.549	0.0009
L1 I-Cache accesses pti	552.034	38.484	521.237	113.382	0.6163
L1 I-Cache misses pti	9.237	4.066	5.001	6.345	0.0426
L1 I-Cache miss rate	1.643	0.665	0.805	0.930	0.0353
LLC accesses pti	65.284	25.071	20.267	20.728	0.0024
LLC misses pti	11.393	12.180	1.479	1.758	0.0009
LLC miss rate	14.953	8.366	11.452	9.309	0.1926
Page faults pmi	17.495	5.074	3.221	5.592	0.0015
Context switches pmi	0.729	1.414	0.338	0.566	0.4423

Another important observation from our analysis is that the Raspberry Pi 3, which is considered one of the most powerful IoT prototyping platforms, is insufficient for optimal execution of IoMT workloads. This observation motivates our plan to study microarchitectural optimizations that will satisfy IoMT workloads' resource requirements, without accruing overheads as compared to currently existent IoT platforms.

Finally, we plan to extend the analysis presented herein to include more workloads within the IoMT space, and other IoT application domains.

REFERENCES

[1] Ashton, Kevin, "That 'Internet of Things' thing," *RFiD Journal*, vol. 22, no. 7, pp. 97–114, 2009.

[2] McKinsey, "Disruptive technologies: Advances that will transform life, business, and the global economy," 2016. [Online]. Available: http://www.mckinsey.com

[3] "Gartner says the internet of things will transform the data center," 2014. [Online]. Available: http://www.gartner.com/newsroom/id/2684616

[4] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[5] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[6] Y. Yuehong, Y. Zeng, X. Chen, and Y. Fan, "The Internet of Things in Healthcare: An overview," *Journal of Industrial Information Integration*, vol. 1, pp. 3–13, 2016.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[8] T. Adegbija, A. Rogacs, C. Patel, and A. Gordon-Ross, "Enabling Right-Provisioned Microprocessor Architectures for the Internet of Things," in *Proc. Int. Mechanical Engineering Congr. and Expo. (ASME)*, 2015, pp. V014T06A001–V014T06A001.

[9] M. Maksimović, V. Vujović, N. Davidović, V. Milošević, and B. Perišić, "Raspberry Pi as Internet of things hardware: performances and constraints," *Design Issues*, vol. 3, p. 8, 2014.

[10] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Proc. 4th Annu. IEEE Int. Workshop Workload Characterization*, Dec 2001, pp. 3–14.

[11] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *Proc. 17th Int. Conf. Parallel Architectures and Compilation Techniques*, Oct 2008.

[12] "The Embedded Microprocessor Benchmark Consortium." [Online]. Available: http://www.eembc.org/

[13] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[14] C. Strydis, D. Dave, and G. N. Gaydadjiev, "ImpBench revisited: An extended characterization of implant-processor benchmarks," in *Proc. Int. Conf. Embedded Computer Systems: Architectures, Modeling and Simulation*, July 2010, pp. 126–135.

[15] M. Arlitt, M. Marwah, G. Bellala, A. Shah, J. Healey, and B. Vandiver, "IoTAbench: An Internet of Things analytics benchmark," in *Proc. 6th ACM/SPEC Int. Conf. Performance Engineering*, 2015, pp. 133–144.

[16] M. I. Ali, F. Gao, and A. Mileo, "Citybench: A configurable benchmark to evaluate rsp engines using smart city datasets," in *Proc. Int. Semantic Web Conf.*, 2015, pp. 374–389.

[17] W. Engelse and C. Zeelenberg, "A single scan algorithm for QRS-detection and feature extraction," *Computers in cardiology*, vol. 6, no. 1979, pp. 37–42, 1979.

[18] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[19] P. A. Toft and J. A. Sørensen, "The Radon transform-theory and implementation," Ph.D. dissertation, Technical University of Denmark, Department of Informatics and Mathematical Modeling, 1996.

[20] H. Ng, S. Ong, K. Foong, P. Goh, and W. Nowinski, "Medical image segmentation using k-means clustering and improved watershed algorithm," in *Proc. IEEE Southwest Symp. Image Analysis and Interpretation*, 2006, pp. 61–65.

[21] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel, "Security and privacy for implantable medical devices," *IEEE Pervasive Computing*, vol. 7, no. 1, 2008.

[22] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[23] "Crypto++ Library 5.6.5." [Online]. Available: https://www.cryptopp.com

[24] M. J. Knieser, F. G. Wolff, C. A. Papachristou, D. J. Weyer, and D. R. McIntyre, "A technique for high ratio LZW compression," in *Proc. Conf. Design, Automation and Test in Europe*, vol. 1.

[25] "ARM." [Online]. Available: http://www.arm.com

[26] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[27] M. J. Crawley, *Statistics: An introduction using R*, 2nd ed. John Wiley & Sons, 2014.