HERMIT: A Benchmark Suite for the Internet of Medical Things

Ankur Limaye, *Student Member, IEEE* and Tosiron Adegbija, *Member, IEEE*

***Abstract*—The growth of the Internet of Things (IoT) will transform the healthcare industry, and enable the emergence of the Internet of Medical Things (IoMT). In this paper, we present and analyze *HERMIT*, a benchmark suite for the Internet of Medical Things. The goal of HERMIT is to facilitate research into new microarchitectures and optimizations that will enable efficient execution of emerging IoMT applications. HERMIT comprises of applications spanning various domains in the healthcare industry, including Computerized Tomography (CT) scan, ultrasound, Magnetic Resonance Imaging (MRI), implantable heart monitors, wearable devices. HERMIT also includes supplementary applications for security and data compression. We analyze HERMIT on an IoT prototyping platform to derive insights into IoMT applications' compute and memory characteristics. We also compare HERMIT to three commonly used benchmark suites, MiBench, SPEC CPU2006, and PARSEC, and show that IoMT applications' characteristics differ from existing benchmarks. Our results motivate the need for a new benchmark suite to enable IoMT-targeted microarchitecture research.**

***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 emerging technology that provides exciting opportunities for innovative solutions and advancements in various industry sectors. In the IoT paradigm, millions of connected smart devices, sensors, and actuators collaborate to monitor and manage the physical environment and human systems. The IoT's goal is to achieve novel and innovative solutions with minimal human interference [1]. It has been projected that the IoT will constitute a $300 billion industry by 2020, and comprise of more than 26 billion interconnected devices [2].

The IoT's uses span a wide variety of application domains, ranging from personal environments like smart homes or transportation to large scale environments including smart offices, logistics, and management [3], [4]. One of the application domains that will be most impacted by the IoT is the healthcare industry [5], which will give rise to the *Internet of Medical Things* (*IoMT*) or medical IoT. The IoT is expected to have a profound $1.1 – $2.5 trillion annual economic impact in the healthcare domain by 2025 [6].

The authors are with the Department of Electrical and Computer Engineering, The University of Arizona, USA, e-mail: {ankurlimaye, tosiron}@email.arizona.edu.

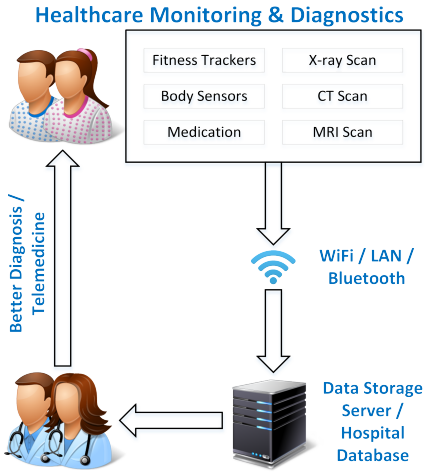


Fig. 1: Overview of the Internet of Medical Things (IoMT)

The IoMT, as illustrated in Fig. 1, will comprise of a network of interconnected medical devices that share data securely with the healthcare professionals to enable innovative medical solutions and services. Several emerging devices, such as pacemakers, portable Magnetic Resonance Imaging (MRI) and Computerized Tomography (CT) scan machines, and wearables will constitute the IoMT. While some of these devices currently have large form factors, we envision that miniature versions of medical devices, like portable ultrasounds [7], will continue to emerge. These devices will provide new opportunities for improved medical processes in various healthcare domains, such as healthcare monitoring, remote medical diagnostics, telemedicine, etc.

Our work is motivated by the need for tools to address some of the critical gaps in the IoMT. The traditional IoT paradigm comprises of low-power edge devices that collect data that is transmitted to centralized high-performance head nodes. The data is then analyzed and visualized on the head nodes to generate results and actionable information. This paradigm, however, can pose significant challenges and overheads, arising from the transmission of data between the edge and head devices. Due to limited bandwidth resources, this paradigm can result in significant delays in retrieving the required results, potentially leading to a bandwidth bottleneck. These bandwidth bottlenecks could be prohibitive in some use-cases, such as healthcare monitoring, where the latency of data analysis and visualization must satisfy stringent real-time constraints in order to prevent medical emergencies. Furthermore, the data transmission between the edge and head nodes can also result in significant energy overheads [8], [9].

Edge computing [10], [11] is an emerging solution to

these problems associated with the traditional IoT. In edge computing, the computation is moved closer to the edge devices, using two options: 1) a gateway can be incorporated between the edge and head devices, or 2) the edge devices are equipped with sufficient computational capabilities to generate actionable information from the data [12]. In this work, we focus on the second scenario, since it reduces communication overheads even further, as compared to the first.

Edge computing is especially critical for the IoMT due to the potentially life-critical nature of IoMT applications. IoMT edge devices must be equipped with right-provisioned processor architectures that can satisfy IoMT applications' computational demands, without introducing significant power and area overheads to the device. However, to provision medical edge devices with sufficient computational capabilities, we must first understand the execution characteristics of emerging IoMT applications, since these characteristics inform the required microarchitecture resources.

In this work, we performed an extensive, forward-looking study of IoMT applications. We also interacted with medical experts with foresight on potential applications that will execute on the IoMT. Our studies and interactions revealed that current benchmark suites do not sufficiently represent emerging IoMT applications. The dearth of benchmark suites that sufficiently captures IoMT application characteristics accounts for a critical technology gap; this gap can hamper microarchitecture research for emerging IoMT devices.

To address this gap, we present *HERMIT*[1], a suite of publicly available[2] benchmark applications chosen to represent different IoMT applications. Our contributions in this paper are summarized as follows:

1) We present an open-source and extensible benchmark suite, *HERMIT* that spans a variety of medical applications, including medical image processing algorithms, inverse Radon transform, implantable heart monitoring algorithms, activity monitoring algorithms, etc.
2) We have characterized HERMIT on a state-of-the-art IoT prototyping platform, the Raspberry Pi 3 [13], to derive insights into how well-provisioned the current platforms are for emerging IoMT applications.
3) We also extensively compare HERMIT to a range of common microarchitecture benchmark suites, including MiBench [14], SPEC CPU2006[3] [15], and PARSEC [16]. Our analysis shows that IoMT applications' characteristics (e.g., memory and branch characteristics) differ from current benchmark suites, motivating the need for a new representative benchmark suite.

II. Background and Related Work

Most IoT research was focused on developing software and communication protocols for IoT devices since the distributed system model was considered as the baseline implementation. However, with the emergence of the edge computing paradigm, emphasis must also be placed on developing efficient microprocessor architectures for the edge devices. To this end, emerging IoT applications must be characterized to gain insights about the applications' resource requirements and the limitations of existing microarchitectures. Benchmarks are a vital component to enabling the characterization of emerging applications, and directly influence the design of emerging devices and their microarchitectures [14], [15], [16]. The benchmark suite presented herein is motivated by our observation that the currently existent benchmarks do not sufficiently represent emerging IoMT applications' execution characteristics.

There are currently several general purpose benchmark suites that enable microarchitecture research in different application domains. MiBench [14] was developed as an open-source alternative to EEMBC [17] to target embedded system applications. Both MiBench and EEMBC are general purpose benchmark suites that contain applications for different application domains like automation, communication, and office software. PARSEC [16] comprises of multithreaded applications and enables the evaluation of a microprocessor's multithreading capabilities. SPEC CPU2006 and CPU2017 [15], [18] comprise of compute- and memory-intensive applications, and aim to assess the system's high-performance computing capabilities.

Apart from these general purpose benchmark suites, there are some application specific benchmark suites. MediaBench [19] is a benchmark suite specifically created for multimedia applications, and comprises of different audio, image, and video processing applications. ImpBench [20] provides a collection of applications targeted towards medical implants and includes applications for drug delivery simulations, data compression, encryption, and integrity checks. Another application specific benchmark suite is IoTAbench [21], which focuses on big data analysis for smart meters deployed in the city. To the best of our knowledge, there are no current benchmarks that directly target IoMT applications.

HERMIT represents a major step toward understanding IoMT applications from the edge computing perspective. We also believe that HERMIT is a step in the direction of understanding the execution requirements of a wide variety of emerging IoT applications. We have carefully selected a few representative applications that will potentially execute on the IoMT devices, and we plan to continue to expand this suite. We have compared the HERMIT benchmark suite to some of the most common benchmark suites: MiBench, PARSEC and CPU06—in order to demonstrate that IoMT applications have different execution characteristics. Thus, we hope that HERMIT will facilitate future IoMT-targeted microarchitecture research.

III. HERMIT Applications

Our goal in this work was to create a benchmark suite to enable microarchitecture researchers in developing processor architectures for next-generation IoMT devices. Since these devices will typically be resource-constrained, over-the-shelf or general purpose processors may be over-provisioned or under-provisioned for the devices. IoMT processors must be

[1]HERMIT stands for 'HEalthcaRe Monitoring for the Internet of Things.'

[2]The applications and inputs can be found at: www.ece.arizona.edu/tosiron/downloads.php.

[3]Hereafter, we refer to SPEC CPU2006 benchmark suite as CPU06.

right-provisioned for the intended applications, in order to adhere to the IoMT devices' resource constraints.

Even though IoMT applications are only a subset of the IoT application space, we observed that the healthcare application space is very expansive. The healthcare domain is vast and encompasses different sectors including health services and facilities (e.g. hospitals, nursing and residential care facilities, ambulatory services), medical devices and equipment, pharmaceuticals, drug manufacturing and delivery systems, etc. Thus, one of our primary objectives in developing the HERMIT benchmark suite is to provide a tractable and extensible representation of applications in the medical and healthcare domain.

We had a few criteria for the applications to be included in the benchmark suite: 1) they must represent emerging IoMT applications; 2) they must be perceived to be high-value by medical personnel; 3) they must support microarchitecture research; 4) they must be open-source; and 5) they must be characteristically diverse, and as a whole, different from existing benchmark suites. Thus, we began the process of compiling the HERMIT applications by first exploring the vast application space of the IoMT. We also interacted with medical personnel to help us identify some of the high-impact applications in the IoMT domain, especially from an edge computing perspective. Furthermore, we emphasized high compute- and memory-intensive applications, since they will impose more computational constraints on emerging medical devices.

HERMIT comprises of a carefully selected set of ten applications, which we are currently expanding to feature more applications. There are eight computation applications that cover different applications in the medical field, while two applications are crucial parts of the communication protocol. The current applications are: *Physical activity estimation (activity)*, *Sleep apnea detection (apdet)*, *Heart rate variability calculation (hrv)*, *Histogram equalization (imghist)*, *Inverse Radon Transform (iradon)*, *k-means clustering (kmeans)*, *ECG-QRS detection (sqrs)*, and *Blood pressure monitoring (wabp)*. We have also included two additional applications, *Advanced Encryption Standard (aes)* and *Lempel-Ziv compression (lzw)* to represent security and compression functions, respectively. We expect that security and compression will be necessary functions for all IoMT devices.

All the applications in the HERMIT benchmark suite are open-source, written in high-level languages (C/C++), and can be easily compiled on any Linux-based operating system. This section briefly motivates and describes the HERMIT benchmarks.

A. Physical activity estimation (activity)

Physical activity data plays a significant role in the diagnosis and treatment of many chronic diseases. Various inertial sensors (like accelerometers, gyroscopes, and pressure sensors), bio-sensors (like ECG, skin temperature, and heart rate monitor), wearable devices (like fitness bands) [22], or general purpose devices (like smartphones [23], [24], [25]), have facilitated robust ways to measure and estimate users' physical activities [26]. The physical activity estimation algorithm [27] included in HERMIT finds periods of activity based on heart rate time series from a bio-sensor. The algorithm calculates the 'activity index' based on: 1) the mean heart rate, 2) the cumulative power of the instantaneous heart rate time series over a given interval, and 3) a heart rate stationary index. This application can potentially execute on fitness wearable devices to determine the user's activity levels. This algorithm takes a 4.5KB instantaneous heart rate time series file as input and gives the duration of least physical activity during the time series as the output.

B. Advanced Encryption Standard (aes)

The IoT is especially susceptible to attacks due to the scale and mobility of IoT devices. These IoT characteristics necessitate security functions to be integral components of IoT devices. The security requirements in IoMT devices are even more critical since medical devices handle sensitive patient data and must maintain the data privacy [28]. Some medical devices, like pacemakers, are life-critical; a security breach can have fatal consequences.

Thus, we have included a security application as an essential application in HERMIT. We selected Advanced Encryption Standard (AES) [29] obtained from the Crypto++ library [30]. It supports a block length of 128 bits and key lengths of 128, 192 and 256 bits. In our implementation, we used default block and key lengths of 128 bits. The application performs both encryption and decryption operations. The application takes an 11KB plain text file as input, and generates an encrypted file. The encrypted file is then decrypted to generate the final output text file. Both of these files are compared to validate correct encryption and decryption.

C. Sleep apnea detection (apdet)

Sleep apnea is a sleep disorder in which the patient's breathing is repeatedly interrupted during sleep. Since the breathing is interrupted, it results in less oxygen supply to the body. If left untreated, sleep apnea can cause severe health issues, including high blood pressure, stroke, heart failure, diabetes, depression, or even death.

The sleep apnea detection application *apdet* [31] detects the periods of sleep apnea based on the periodic oscillations in cardiac interbeat (NN) intervals. Hilbert transform is used to calculate the instantaneous amplitudes and frequencies of the time series, and applies certain thresholds on the averages, standard deviations over a moving 5-minute window to detect the sleep apnea periods. We use a 5.6MB ECG file from PhysioNet's [32] Apnea-ECG Database as input to the application, which then prints the detected sleep apnea periods to an output file. The PhysioNet repository comprises of different open-source software and a database of related physiological signals.

D. Heart rate variability calculation (hrv)

Heart rate variability (HRV) is the measure of the variations in time intervals between adjacent heartbeats. Measuring heartbeat variability has both clinical and psychological significance. Lower long-term HRV has been found to be a predictor of mortality after heart attacks [33]. It is also a psychological marker for depression, anxiety, and stress. The heart rate variability (*hrv*) algorithm [34] calculates the time

and frequency domain HRV statistics. A 5.1MB ECG file from PhysioNet's database is input to this application. A text file containing the HRV statistics is generated after completing the execution.

E. Histogram Equalization (imghist)

Medical imaging techniques provide visual representations of internal organs and help doctors to provide an accurate diagnosis. Most of these images require pre-processing steps to correct for the sensor- and platform-specific distortions and improve the visibility of the image. Histogram equalization (*imghist*) is such an image processing technique used to increase the contrast of the image. It is a necessary step for digital X-ray, medical ultrasound, and MRI images. It stretches the dynamic range of pixel values to increase the contrast of the areas in the image with lower contrast. This application takes a 54KB grayscale image as input and outputs a grayscale image with better contrast.

F. Inverse Radon Transform (iradon)

This benchmark is motivated by the emergence of medical devices such as portable ultrasound and MRI devices, which will be major components of the IoMT. CT scan and MRI devices acquire projection data at different angles around a patient. Image reconstruction is the process that generates the images from these projections. The inverse Radon transform is used for reconstruction of the images. The iradon application [35] used in HERMIT reconstructs the output image from a 159KB 2D sinogram input data file for visualizing abnormal openings in the body.

G. K-means Clustering (kmeans)

Image segmentation techniques are used to partition an image into different segments belonging to the same object. In medical imaging, segmentation is necessary to classify and label different tissues or organs for better diagnostics. k-means clustering (*kmeans*) is one of the commonly used medical image segmentation algorithm [36]. The k-means algorithm implemented in HERMIT partitions different parts of the image into clusters based on pixel intensities and the nearest mean intensity values surrounding the pixel. A 54KB grayscale image is input to the application and generates a text file with the pixel locations grouped together in 10 bins.

H. Lempel-Ziv Compression (lzw)

The network bandwidth on the IoT will be limited, and shared by all the connected devices. Efficient use of the bandwidth is necessary when a large number of devices are connected. Data compression techniques are employed in these scenarios to reduce the amount of bandwidth used while transmitting the data. We included the Lempel-Ziv Compression algorithm [37] in HERMIT suite because it is a fast, low-overhead and lossless compression algorithm ideal for IoMT devices. The *lzw* benchmark takes an 11KB input text file and generates a compressed output file. For validation, the generated output is decompressed and compared with the original input file.

I. QRS Detection in ECG (sqrs)

The QRS detection algorithm (*sqrs*) [38] detects the QRS complex peaks and troughs in an Electrocardiogram (ECG) signal. The QRS complex is the combination of Q, R and S waves, and represents the right and left ventricular depolarization. The QRS complex attributes like the amplitude and time duration between intervals are used for diagnosing heart conditions like cardiac arrhythmias, ventricular hypertrophy, myocardial infractions, and other heart abnormalities. This algorithm is replicated from the PhysioNet repository. The algorithm only uses signal 0 of the ECG to detect the QRS complex. A 63KB test ECG signal from the PhysioNet's data bank is input to this application, and it gives an output annotation file with the detected QRS points.

J. Blood Pressure Monitoring (wabp)

Blood pressure is one of the most commonly measured human body reading. It has a direct correlation with the strain on the heart and arteries, and the blood flow to the organs. If the blood pressure is low, there is poor blood flow to all the organs and can cause organ failure. On the other hand, having hypertension (high blood pressure) puts too much strain and can lead to heart attack, stroke or hypertensive organ failure. The blood pressure monitoring algorithm considered in the HERMIT suite is also from the PhysioNet repository. This algorithm calculates the arterial blood pressure (ABP) from a continuous ABP signal. The algorithm uses the first derivative value of the ABP signal to calculate the blood pressure. ABP signals (from the PhysioNet's data bank) are input to the application, and a blood pressure log file is output.

IV. Evaluation Methodology

We executed the HERMIT applications on the Raspberry Pi 3 [13] platform and analyzed the applications' execution characteristics. We decided to use a physical platform, instead of simulations, in order to better understand the applications' execution on state-of-the-art IoT-targeted microarchitectures. The Raspberry Pi 3 is one of the most common low-cost and powerful IoT prototyping platforms on the market, and offers a viable option for prototyping IoMT applications.

The Raspberry Pi 3 Model B uses a Broadcom BCM2837 SoC featuring a 1.2GHz quad-core 64-bit ARM Cortex A53 CPU and a 400 MHz Broadcom VideoCore IV GPU. The ARM Cortex CPU has separate 2-way set associative 16 kB level one instruction and data caches, 512 kB unified level two cache, 1 GB LPDDR2 RAM and microSD slot for expandable storage. It also features Ethernet port and supports 802.11n wireless LAN. We installed the Ubuntu Mate 16.04 LTS operating system (Linux kernel 4.4.38-v7+) on the Pi 3.

For this paper, we focused on execution characteristics that are observable from the hardware performance counters; thus, we used the Linux *perf* utility to gather the execution statistics. We compiled all the HERMIT benchmarks using *gcc* without using any optimization flags. We executed the applications to completion 50 times each, to account for the variability between different runs, and analyzed the mean values of all the runs for various execution characteristics.

To evaluate the similarities and differences between HERMIT and current benchmark suites, we compared the HERMIT applications to three popular general purpose benchmark suites: MiBench, CPU06, and PARSEC. We used 15 applications from MiBench—*basicmath*, *bitcount*, *qsort*, *susan*, *jpeg*, *typeset*, *dijkstra*, *patricia*, *stringsearch*, *blowfish*, *sha*, *adpcm*, *crc32*, *fft* and *gsm*—all with the large input data set; 8 applications from PARSEC—*blackscholes*, *bodytrack*, *facesim*, *ferret*, *fluidanimate*, *freqmine*, *streamcluster* and *vips*—with simlarge input data set; and 20 applications from CPU06—*astar*, *bzip*, *gcc*, *gobmk*, *h264ref*, *hmmer*, *libquantum*, *omnetpp*, *sjeng*, *specrand*, *xalancbmk*, *calculix*, *gromacs*, *lbm*, *leslie3d*, *milc*, *namd*, *povray*, *soplex*, and *tonto*—with the reference input data sets.

V. HERMIT Applications' Characteristics

Table I presents an overview of the HERMIT benchmarks' execution characteristics, including, the instruction count, cache references, branch instructions, and instructions per cycle (IPC). One of our first observations is the variety in the applications' characteristics. For example, the instruction count ranges from about 3.5 million instructions to more than 8 billion instructions. *Hrv* is the largest application while *aes* has the smallest instruction count. The supplementary applications–*aes* and *lzw*–both have a very small instruction count. This is critical to ensure that these supplementary functions result in minimal overheads, and have fewer resource requirements than the actual medical applications. Another interesting observation is that *sqrs* and *wabp* had very similar characteristics. These benchmarks have similar characteristics because they both work on the same input data set, perform similar data operations, but derive two different conclusions. *Hrv* is the most compute and memory intensive application in HERMIT, as it has the lowest IPC and highest number of cache references and branch instructions. In what follows, we discuss the applications' execution characteristics in detail.

A. IPC Analysis

The instructions per cycle (IPC) is perhaps the most important performance characteristic of the microprocessor. IPC is the average number of instructions that are executed by the processor in each clock cycle. The IPC is mainly impacted by underlying microarchitecture characteristics like pipeline depth, execution paradigm (in-order vs. out-of-order), branch predictor efficiency, cache configuration, etc. The IPC is also affected by application characteristics, such as instruction mix, cache access patterns, branch instructions, etc. Higher IPC values suggest that the microarchitecture is right-provisioned for the executing applications, whereas, low IPC values reveal the need for performance optimizations to satisfy the applications' requirements.

Fig. 2 shows the IPC values of the different HERMIT benchmarks. We also included the branch and last level cache (LLC) miss rates to illustrate the impact of these characteristics on the overall performance. The average IPC of all the HERMIT applications on the Raspberry Pi 3 was relatively low at 0.598. *Apdet* had the highest IPC of 0.834, followed closely by *activity* with an IPC value of 0.815. On the other hand, *hrv* and *iradon* had the lowest IPC values of 0.248 and 0.251.

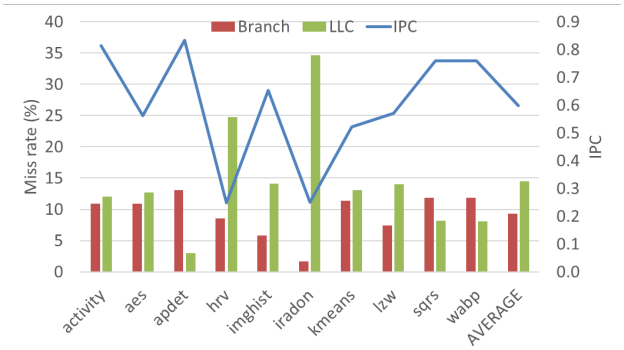


Fig. 2: IPC analysis: impact of branch and LLC miss rates on the IPC

We also investigated the impact of branches and LLC miss rates on the IPC (Fig. 2). In theory, a branch miss would flush the pipeline, and a cache miss will result in pipeline stalls. Both of these would increase the number of clock cycles required to complete the execution, thus decreasing the IPC value. The average LLC miss rate for all the applications was 14.451%. The two benchmarks with the lowest IPC values—*hrv* and *iradon*—also had the highest LLC miss rates of 24.779% and 34.600%, respectively, illustrating that the low IPC was because the benchmarks are memory-bound.

We observed that branch miss rates also significantly impacted the IPC values. Similarly to the LLC miss rates, benchmarks with higher branch miss rates exhibited lower IPC values. For example, even though *imghist* and *lzw* had similar LLC miss rates of 14.106% and 14.037%, respectively, *lzw* had a higher branch miss rate of 7.420% compared to *imghist*'s 5.779%. As a result, *lzw* had a lower IPC value of 0.570 compared to *imghist*'s 0.653.

Also, we observed that the LLC miss rates had a higher impact on the IPC than the branch miss rates. For example, *kmeans*' branch miss rate of 11.412% was similar to *sqrs*' and *wabp*'s branch miss rates of 11.831% and 11.860%, but *kmeans* had a higher LLC miss rate of 13.021% compared to *sqrs*' 8.147% and *wabp*'s 8.097%, resulting in a 31% lower IPC value for *kmeans*.

B. Memory Characteristics

We analyzed the memory characteristics of the HERMIT benchmarks to derive insights into the cache access patterns of the applications. Our analysis also provides insights into how well provisioned the memory hierarchy in the Raspberry Pi 3, representing the state-of-the-art, is for IoMT applications. We analyzed the number of cache accesses per thousand instruction (pti) and the cache miss rates for the L1 I-cache, L1 D-cache, and the LLC for all the benchmarks. The cache accesses per thousand instructions illustrates the stress a benchmark places on the cache, while the cache miss rates illustrate the cache's ability to handle the benchmark's memory intensity.

1) Cache accesses per thousand instructions: Fig. 3 depicts the cache accesses pti for all the benchmarks. The average L1 I-Cache, L1 D-Cache and LLC accesses pti across all the benchmarks was 572.482, 350.426 and 30.836, respectively.

TABLE I: HERMIT applications' characteristics

Application	Description	Instruction Count	Cache References	Branch Instructions	IPC
activity	Estimate Physical activity level	13,133,270	3,955,078	1,190,635	0.815
aes	Advanced Encryption Standard	3,471,322	1,240,650	327,100	0.562
apdet	Sleep Apnea detection	4,286,601,548	1,396,684,405	446,623,163	0.833
hrv	Heart rate variability	8,920,730,779	2,846,359,793	707,628,576	0.248
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
lzw	Lempel-Ziv-Welch Compression	5,671,513	2,135,652	636,136	0.570
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

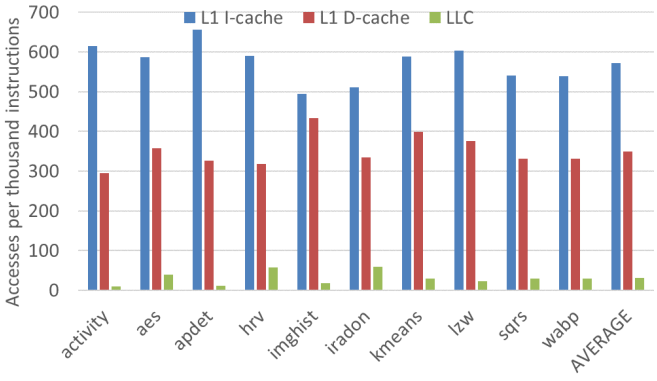


Fig. 3: Cache accesses

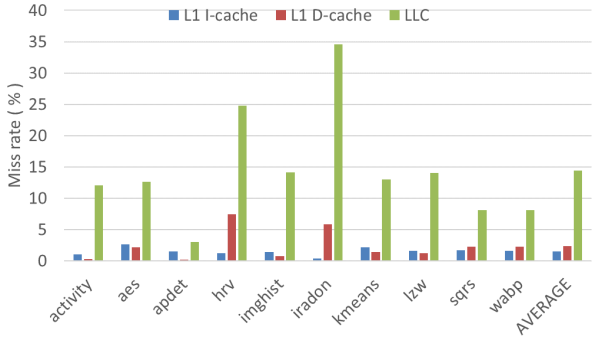


Fig. 4: Cache miss rates

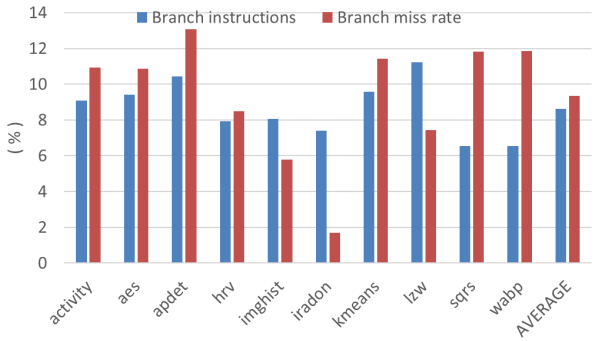


Fig. 5: Branch instructions

Apdet's L1 I-Cache accesses pti was the highest at 656.299, while *imgHist* had the lowest value of 494.784. *Activity* had the second largest L1 I-Cache accesses pti (614.261) but had the lowest L1 D-cache accesses pti at 294.426. This observation shows that *activity* is a compute-intensive application with a small input data size.

On the other hand, *imghist* exhibited the opposite trend with the lowest L1 I-Cache access pti, but the maximum 434.516 L1 D-Cache accesses pti. This suggests that the *imghist* performs limited operations repeatedly over a large input data, and generates large intermediate or output data. *Kmeans* and *lzw* exhibited higher than average L1-D cache accesses pti—399.863 and 376.357, respectively; however, this was because of large intermediate data generated during the execution, and not due to input data size. *Iradon* had the highest number of LLC cache accesses pti at 59.113, closely followed by *hrv* with 58.047 LLC accesses pti—91.701% and 88.244%, respectively, above the average value. In the next subsection, we discuss the caches' provisioning for the HERMIT benchmarks.

2) Cache miss rates: Fig. 4 depicts the cache miss rates for the HERMIT benchmarks. The average L1 I-Cache, L1 D-Cache, and LLC miss rates were 1.526%, 2.395% and 14.451% respectively, across all the benchmarks. *Iradon* exhibited the lowest L1 I-Cache miss rate of 0.356% but had the second highest L1 D-Cache miss rate of 5.852% and the highest 34.600% LLC miss rate. These results suggest that while the L1 I-Cache in the Raspberry Pi 3 is sufficiently provisioned for *iradon*, a larger L1 D-Cache and LLC is required to satisfy *iradon*'s data needs.

We made a similar observation for *hrv*. *Hrv*'s L1 I-Cache miss rate was slightly below average at 1.227%, but the L1 D-Cache miss rate was the highest at 7.424%—a 209.997% increase over the average—and the LLC miss rate was the second largest at 24.779%. For applications such as *iradon* and *hrv*, a larger data cache is required. Alternatively, the edge device could offload the execution to a more powerful central node (server) to reduce the resource requirements on the edge device.

C. Branch characteristics

Branch instructions, otherwise known as the control instructions, evaluate a logical condition, and depending on the outcome of the condition the branch is taken or the next instruction is executed. Branch instructions can be a major limiting factor in achieving optimal performance, since they can introduce control hazards in the pipeline. Thus, branch predictors exist in microprocessors to reduce the performance impact of branch instructions. The ARM Cortex A53 processor present in the Raspberry Pi 3 features a single-entry branch

target instruction cache, a 256-entry branch predictor with a 3072-entry pattern history prediction table [39].

We analyzed the HERMIT benchmarks' branch characteristics as a function of the percentage of branch instructions and the branch miss rates. The percentage branch instructions is an architecture-independent characteristic as it only depends on the instruction mix, and reveals the frequency of branch instructions in an application. However, the branch miss rate provides insights on the branch predictor's efficiency and accuracy.

Fig. 5 depicts the percentage branch instructions and the branch miss rates for all HERMIT benchmarks. On average, the benchmarks had 8.621% branch instructions, and the average branch miss rate was 9.333%. We observed that there was not much variation in the frequency of branch instructions among the different benchmarks. *Lzw* had the highest branch frequency at 11.216%, while *sqrs* had the lowest branch frequency of 6.547%.

Iradon exhibited the lowest branch miss rate of 1.678%—82.018% below average—while *apdet* had the highest at 13.060%. The branch miss rate is also an indicator of the applications' instruction mix, reflecting a trend towards conditional branches vs. loops. In general, conditional branches are harder to predict than loops; thus, higher branch miss rates reveal a greater number of conditional branches than loops. Thus, it can be estimated from the results that *iradon* has more loops than conditional branch instructions, while *apdet* has more conditional branches than loops.

VI. Comparison of HERMIT with existing benchmark suites

Table II presents the comparison of the hardware characteristics between the considered benchmark suites. We compared each HERMIT benchmark with 15 applications from MiBench, 8 applications from PARSEC, and 20 applications from CPU06. However, for brevity, Table II only shows the average values for different characteristics of all the benchmarks in the different suites.

The CPU06 benchmark suite consists of compute-intensive high-performance applications; the benchmarks exhibit the highest instruction count and maximum execution time. On the other hand, MiBench, which targets embedded system applications, has the shortest execution time and highest average IPC value of 0.911. In general, we observed that the most similar characteristics between HERMIT and the other benchmark suites are the branch instruction frequency, L1 I-Cache accesses pti, and L1 D-Cache access pti. However, we observed differences in other characteristics that warranted further analysis to quantify these differences.

To evaluate HERMIT's difference from the other benchmark suites, we used a statistical method—the Wilcoxon rank-sum test [40]—to test HERMIT's statistically significant difference from the three benchmark suites. Using the Wilcoxon rank-sum test method, we calculated the *p-value* for the data sets. A *p-value* of less than 0.05 indicates that the data sets are significantly different from each other [41].

Table II shows that HERMIT exhibits significant statistical difference from MiBench, PARSEC and CPU06 in 12, 5, and 7 of the 19 characteristics considered (the shaded fields in the table represent characteristics with a high statistical difference). For example, while HERMIT and MiBench were similar in some respects (e.g., execution time, percentage branch instructions), they were substantially different in other characteristics (e.g., cache miss rates, page faults, TLB misses, etc.). We observed similar trends in the comparisons with both PARSEC and CPU06. We also observed that HERMIT applications, despite their small sizes (in instruction count) compared to PARSEC and CPU06, exhibited high branch, L1-D, and LLC miss rates that were similar to PARSEC and CPU06.

Overall, we observed that HERMIT was substantially different from the other three benchmark suites in the instruction count, L1 I-Cache miss rate, and L1 I-Cache misses pti. Interestingly, HERMIT was most different from the MiBench benchmark suite, which illustrates the fact that state-of-the-art embedded systems benchmark suites may not sufficiently represent emerging IoMT applications' characteristics. These results also suggest that while specific benchmarks from the different suites may suffice for representing different IoMT applications; no one benchmark suite suffices, further motivating the need for the HERMIT benchmark suite.

Principal Component Analysis. To further analyze the differences between the different benchmark sutes, we also performed Principal Component Analysis (PCA) [42] on the hardware characteristics of all the benchmark suites. PCA is a statistical data analysis technique that helps in interpreting multivariate data, and converts a set of possibly correlated p variables $\{X_1, X_2, ..., X_p\}$ into a set of linearly uncorrelated p variables $\{Z_1, Z_2, ..., Z_p\}$, known as the principal components (PCs), such that:

$$Z_i = \sum_{j=1}^{p} a_{ij} X_j$$

The PC transformation has the following useful properties:

1) The total variance in the data remains same and the transformation does not result in any information loss:

$$\sum_{i=1}^{p} Var[X_i] = \sum_{i=1}^{p} Var[Z_i]$$

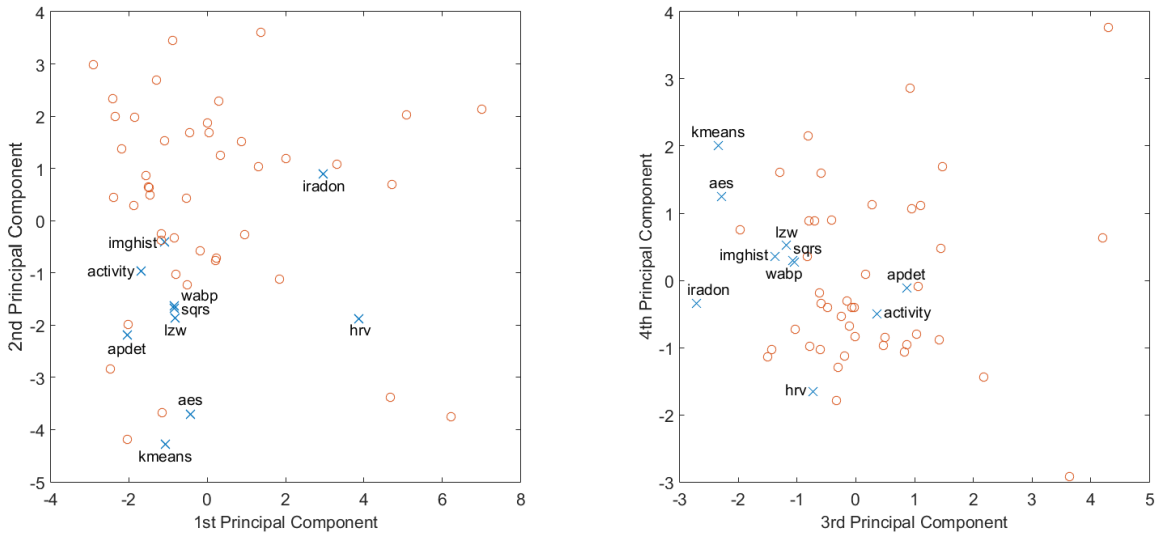
2) There is no information overlap between the PCs, and the PCs are uncorrelated to each other:

$$Cov[Z_i, Z_j] = 0, \forall i \neq j$$

3) Z_1 has the most information (variance), and each subsequent Z_i has less information than previous Z_{i-1}. This property is useful in reducing the dimensionality of the data. The components with lower variance can be removed from the analysis without significant loss of information:

$$Var[Z_1] > Var[Z_2] > ... > Var[Z_p]$$

As highlighted in 3), PCA is useful in reducing the data dimensionality without significant loss in information. In our analysis, we considered 19 different application characteristics as discussed in Table II. Using PCA, we reduced the [42 $\times$



(a) (b)

Fig. 6: Scatter-plot of all applications' PC values

19] data to [42 × 4] by considering only the first four PCs, accounting for 68.99% variance.

Fig. 6 depicts the scatter plot of PC values for the applications from HERMIT, MiBench, PARSEC and CPU06 benchmark suites. We have highlighted the HERMIT benchmarks with blue crosses, while the rest of the benchmarks are shown in orange circles. The benchmarks with similar PC values, will show similar hardware performance characteristics. For example, consider the benchmarks *sqrs*, *wabp* and *kmeans*. *Sqrs* and *wabp* have closer PC values, as seen in Fig. 6a and Fig. 6b, while *kmeans* has PC values significantly different from the other two. Table III gives the application characteristic values of these benchmarks, and validates that benchmarks with similar PC values (*sqrs* and *wabp*) have similar application characteristics. Another observation made from Fig. 6 is that almost all HERMIT benchmarks have different PC values than the benchmarks from other benchmark suites.

To determine if any specific benchmarks from the other benchmark suites can represent HERMIT benchmarks, we compared each HERMIT benchmarks with all 42 benchmarks considered from MiBench, PARSEC, and CPU06. We observed that there was no one benchmark from the other suites that accurately represented HERMIT in all execution characteristics. We then grouped the execution characteristics into five categories: performance characteristics (IPC and execution time), branch characteristics (branch instruction frequency, branch miss rate, and branch misses pti), L1 I-Cache characteristics (L1 I-Cache accesses pti, L1 I-Cache miss rate,

TABLE II: Comparison of HERMIT benchmarks with MiBench, PARSEC and CPU06

Hardware Characteristics	Average Values				Wilcoxon Rank-sum Test p-Values		
	HERMIT	MiBench	PARSEC	CPU06	MiBench	PARSEC	CPU06
Instruction Count	1,345,260,263	919,583,759	22,096,675,237	1,504,949,141,494	0.0429	0.00009	0.00003
IPC	0.597	0.911	0.641	0.537	0.0043	0.8968	0.0568
Execution Time (ms)	3.307	3.063	16.953	2,433.242	0.0557	0.0021	0.00005
Branch instructions (%)	8.621	9.687	5.678	9.920	0.3601	0.0031	0.5973
Branch miss rate (%)	9.333	5.979	11.528	8.109	0.0375	0.2743	0.1974
Branch misses pti	8.116	5.970	6.818	7.088	0.2555	0.6334	0.1569
L1 I-Cache accesses pti	572.482	537.063	553.402	557.215	0.4881	0.4598	0.5684
L1 I-Cache miss rate (%)	1.526	0.813	0.400	0.189	0.0284	0.00055	0.00003
L1 I-Cache misses pti	8.810	5.157	2.211	1.185	0.0375	0.00055	0.00003
L1 D-Cache accesses pti	350.426	335.935	357.159	382.168	0.9337	0.8968	0.1832
L1 D-Cache miss rate (%)	2.395	0.440	1.210	3.064	0.0030	0.3590	0.5973
L1 D-Cache misses pti	8.137	1.809	3.998	12.059	0.0025	0.2370	0.5401
LLC accesses pti	30.836	10.317	19.180	37.168	0.0030	0.0676	0.7836
LLC miss rate (%)	14.451	10.230	19.114	23.024	0.1416	0.2370	0.1038
LLC misses pti	5.585	0.665	4.808	10.336	0.00079	0.4598	0.6270
Page faults pmi	13.082	2.300	5.797	1.786	0.00079	0.1101	0.00018
Context switches pmi	0.618	0.403	0.173	0.016	0.1741	0.5148	0.00049
iTLB misses pmi	15.777	3.452	7.912	26.4310	0.00079	0.1728	0.0213
dTLB misses pmi	1,133.287	10.917	225.160	1,308.998	0.00065	0.6334	0.4586

TABLE III: Validating similarity based on PC values

Characteristic	*sqrs*	*wabp*	*kmeans*
IPC	0.760	0.756	0.523
L1 I Cache accesses pti	540.090	539.384	588.003
L1 I Cache miss rate (%)	1.668	1.647	2.176
L1 D Cache accesses pti	330.983	331.025	399.863
L1 D Cache miss rate (%)	2.306	2.302	1.386
LLC accesses pti	29.658	29.837	29.535
LLC miss rate (%)	8.147	8.097	13.021
Branches instructions (%)	6.547	6.552	9.567
Branch miss rate (%)	11.831	11.860	11.412

and L1 I-Cache misses pti), L1 D-Cache characteristics (L1 D-Cache accesses pti, L1 D-Cache miss rate, L1 D-Cache misses pti), and LLC characteristics (LLC accesses pti, LLC miss rate, and LLC misses pti). Using these groupings, we then performed PCA to determine benchmarks similarities for the different execution characteristics.

Table IV lists the benchmarks with the minimum Euclidean distance between the PCs for the different grouped characteristics. A Euclidean distance closer to zero indicates a higher degree of similarity among the execution characteristics. The two most similar benchmarks were *apdet* and *jpeg* (MiBench) for the performance characteristics as seen in Table IV. Similarly, *h264ref* (MiBench) had the closest branch characteristics to *imghist*. For L1 I Cache, L1 D Cache and LLC, *vips* (PARSEC) was closest to *iradon*, *fft* (MiBench) was closest to *apdet*, and *blackscholes* (PARSEC) was closest to *activity*. In general, our analysis indicates that a close representation of HERMIT applications using current benchmark suites is only possible using benchmarks from different suites to represent different execution characteristics.

VII. Conclusion and Future Work

In this paper, we present *HERMIT*, a benchmark suite for the Internet of Medical Things (IoMT). This benchmark suite is a step toward enabling research into highly efficient processors for emerging IoMT applications. We analyzed HERMIT for different execution characteristics and compared the benchmarks to three commonly used benchmark suites, MiBench, PARSEC, and SPEC CPU2006. Our comparative analysis revealed that the execution characteristics of HERMIT benchmarks substantially differ from those of existing benchmark suites.

Our future work involves extending HERMIT to represent more applications in the IoMT space. We also plan to investigate additional execution characteristics of HERMIT benchmarks, such as phase characteristics, which will enable optimizations for adaptable microprocessors. We will also further analyze how execution can be optimized using specialized processors, such as application specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs), and graphics processing units (GPUs).

References

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

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

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

[4] 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.

[5] 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.

[6] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, and A. Marrs, "Disruptive technologies: Advances that will transform life, business, and the global economy," Tech. Rep., May 2013.

[7] D. M. Becker, C. A. Tafoya, S. L. Becker, G. H. Kruger, M. J. Tafoya, and T. K. Becker, "The use of portable ultrasound devices in low-and middle-income countries: a systematic review of the literature," *Tropical Medicine & International Health*, vol. 21, no. 3, pp. 294–311, 2016.

[8] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy-aware wireless microsensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 40–50, Mar 2002.

[9] T. T.-O. Kwok and Y.-K. Kwok, "Computation and energy efficient image processing in wireless sensor networks based on reconfigurable computing," in *Proc. Int. Conf. Parallel Processing Workshops*, 2006, pp. 8 pp.–50.

[10] 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.

[11] 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.

[12] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *Proc. 1st MCC Workshop Mobile Cloud Computing*, Helsinki, Finland, 2012, pp. 13–16.

[13] 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.

[14] 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.

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

[16] 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.

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

[18] A. Limaye and T. Adegbija, "A Workload Characterization of the SPEC CPU2017 Benchmark Suite," in *Proc. IEEE Int. Symp. Performance Analysis of Systems and Software (ISPASS)*, Apr 2018.

[19] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "MediaBench: a tool for evaluating and synthesizing multimedia and communicatons systems," in *Proc. 30th Ann. Int. Symp. on Microarchitecture*, 1997, pp. 330–335.

[20] 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.

[21] 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.

[22] F. Tehranipoor, N. Karimian, P. A. Wortman, and J. A. Chandy, "Low-cost authentication paradigm for consumer electronics within the internet of wearable fitness tracking applications," in *Consumer Electronics (ICCE), 2018 IEEE International Conference on*. IEEE, 2018, pp. 1–6.

[23] I. Bisio, F. Lavagetto, M. Marchese, and A. Sciarrone, "Smartphone-based user Activity Recognition Method for Health Remote Monitoring Applications," in *Proc. 2^{nd} Int. Conf. Pervasive and Embedded Computing and Communication Systems*, Feb 2012, pp. 200–205.

[24] I. Bisio, A. Delfino, F. Lavagetto, and A. Sciarrone, "Enabling IoT for in-home rehabilitation: accelerometer signals classification methods for activity and movement recognition," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 135–146, 2017.

[25] I. Bisio, F. Lavagetto, M. Marchese, and A. Sciarrone, "Smartphone-centric ambient assisted living platform for patients suffering from co-morbidities monitoring," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 34–41, Jan 2015.

TABLE IV: Applications with similar execution characteristics to HERMIT applications

HERMIT	Execution Characteristics									
	Performance		Branch		L1 I-Cache		L1 D-Cache		LLC	
	Benchmark	Dist.	Benchmark	Dist.	Benchmark	Dist.	Benchmark	Dist.	Benchmark	Dist.
activity	*freqmine* (PARSEC)	0.0139	*typeset* (MiBench)	0.5374	*xalancbmk* (CPU06)	0.4638	*patricia* (MiBench)	0.1967	*blackscholes* (PARSEC)	0.1162
aes	*facesim* (PARSEC)	0.0723	*patricia* (MiBench)	0.5233	*patricia* (MiBench)	0.7940	*namd* (CPU06)	0.1392	*xalancbmk* (CPU06)	0.5146
apdet	*jpeg* (MiBench)	0.0080	*freqmine* (PARSEC)	0.4149	*qsort* (MiBench)	0.6140	*fft* (MiBench)	0.0633	*qsort* (MiBench)	0.1372
hrv	*gcc* (CPU06)	0.5375	*susan* (MiBench)	0.2085	*adpcm* (MiBench)	0.4693	*omnetpp* (CPU06)	1.0450	*astar* (CPU06)	0.2176
imghist	*stringsearch* (MiBench)	0.0638	*h264ref* (CPU06)	0.0340	*adpcm* (MiBench)	0.6462	*basicmath* (MiBench)	0.3459	*ferret* (PARSEC)	0.2176
iradon	*gcc* (CPU06)	0.5273	*leslie3d* (CPU06)	0.1476	*vips* (PARSEC)	0.2253	*soplex* (CPU06)	0.8293	*omnetpp* (CPU06)	0.7442
kmeans	*streamcluster* (PARSEC)	0.0440	*freqmine* (PARSEC)	0.5830	*stringsearch* (MiBench)	1.4187	*tonto* (CPU06)	0.2236	*stringsearch* (MiBench)	0.2041
lzw	*facesim* (PARSEC)	0.0441	*fft* (MiBench)	0.1945	*qsort* (MiBench)	0.8738	*tonto* (CPU06)	0.1010	*stringsearch* (MiBench)	0.1981
sqrs	*vips* (PARSEC)	0.0127	*typeset* (MiBench)	0.1457	*adpcm* (MiBench)	0.6749	*namd* (CPU06)	0.3426	*typeset* (MiBench)	0.2848
wabp	*vips* (PARSEC)	0.0126	*typeset* (MiBench)	0.1402	*adpcm* (MiBench)	0.6389	*namd* (CPU06)	0.3428	*typeset* (MiBench)	0.2764

[26] J. Qi, P. Yang, D. Fan, and Z. Deng, "A Survey of Physical Activity Monitoring and Assessment Using Internet of Things Technology," in *Proc. IEEE Int. Conf. Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, Oct 2015, pp. 2353–2358.

[27] G. B. Moody, "ECG-based indices of physical activity," in *Proc. Computers in Cardiology*, Oct 1992, pp. 403–406.

[28] 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.

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

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

[31] J. E. Mietus, C. K. Peng, P. C. Ivanov, and A. L. Goldberger, "Detection of obstructive sleep apnea from cardiac interbeat interval time series," in *Proc. Computers in Cardiology*, vol. 27, 2000, pp. 753–756.

[32] 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.

[33] R. E. Kleiger, J. Miller, J. Bigger, and A. J. Moss, "Decreased heart rate variability and its association with increased mortality after acute myocardial infarction," *The American Journal of Cardiology*, vol. 59, no. 4, pp. 256 – 262, 1987.

[34] J. E. Mietus, C. K. Peng, I. Henry, R. L. Goldsmith, and A. L. Goldberger, "The pNNx files: re-examining a widely used heart rate variability measure," *Heart*, vol. 88, no. 4, pp. 378–380, 2002.

[35] 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.

[36] 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.

[37] 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, 2003, pp. 101–16.

[38] 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.

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

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

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

[42] L. Eeckhout, H. Vandierendonck, and K. D. Bosschere, "Workload design: selecting representative program-input pairs," in *Proc. Int. Conf. Parallel Architectures and Compilation Techniques*, 2002, pp. 83–94.

Ankur Limaye is a Ph.D. student in the Department of Electrical and Computer Engineering at the University of Arizona. His research interests include computer architecture, workload characterization and performance analysis, and right-provisioned microarchitectures for IoT devices.

Tosiron Adegbija (M'11) received his M.S and Ph.D in Electrical and Computer Engineering from the University of Florida in 2011 and 2015, respectively and his B.Eng in Electrical Engineering from the University of Ilorin, Nigeria in 2005.

He is currently an Assistant Professor of Electrical and Computer Engineering at the University of Arizona, USA. His research interests are in computer architecture, with emphasis on adaptable computing, low-power embedded systems design and optimization methodologies, and microprocessor optimizations for the Internet of Things (IoT).

Dr. Adegbija was a recipient of the Best Paper Award at the Ph.D forum of IEEE Computer Society Annual Symposium on VLSI (ISVLSI) in 2014.