Understanding Energy Efficiency of Databases on Single Board Computers for Edge Computing

Jian Liu
Computer Science & Engineering
Louisiana State University
jliu@csc.lsu.edu

Kefei Wang
Computer Science & Engineering
Louisiana State University
kwang@csc.lsu.edu

Feng Chen
Computer Science & Engineering
Louisiana State University
fchen@csc.lsu.edu

***Abstract*—With the rapid advancement in edge computing, a recent trend is to migrate data processing from data centers to the edge to avoid long data transmission latencies. Databases, as an indispensable component, play a crucial role in efficient data management on edge devices. However, a critical limitation of edge devices is the highly constrained energy resource. Databases often incur a heavy load of CPU and storage I/O activities, which raises a particular concern on power-constrained platforms. In this paper, we have conducted an experimental study on the energy consumption of three representative databases, namely SQLite, LevelDB, and MongoDB, on single board computers for edge computing. We find that by deploying an appropriate database according to specific scenarios, the energy consumption can be reduced by a factor of 58.3, and the bandwidth can be improved by a factor of 54.4. Based on our experimental results, we also present several important system implications associated with our findings. We hope our first-hand data and the obtained insight can provide useful guidance for database and edge system designers and practitioners to develop and deploy energy-efficient databases for edge computing.**

***Index Terms*—Energy efficiency; Single board computer; Database; Edge computing.**

I. Introduction

With the rapid advancement of edge computing technology, a recent trend is to migrate data processing from data centers to the edge of the Internet [1]–[7]. Such a practice brings several benefits. For example, excessively long-latency accesses to the remote data center can be avoided, the traffic over the network can be significantly reduced, and the risk of leaking user data can also be minimized for improved data privacy.

Databases, as a key component in data management, play a crucial role in data processing on the edge for various applications, such as image recognition, augmented reality, virtual reality, autonomous vehicles, and many others [8]–[11]. Traditionally, the design of database systems focuses on performance optimizations, such as increasing throughput and reducing latency. On edge systems, a unique and must-have consideration is *energy efficiency*. Edge devices are often deployed in environments with unstable or limited power sources. Some systems are powered solely by battery or by a reproducible but constrained energy source, such as solar power. In such environments, energy is an extremely scarce resource but crucial to the normal operations of devices in the wild. A critical challenge of deploying databases on edge systems is the associated energy constraint.

Databases on edge systems have several unique issues. First, most edge systems adopt an ARM-based architecture, which has distinct power properties compared to traditional x86-based architectures. Second, due to space constraint, edge systems typically rely on flash storage, such as Secure Digital (SD) and eMMC cards, while conventional servers normally adopt hard disk drives as storage. The two different storage technologies carry very distinct characteristics in terms of both performance and energy efficiency. Finally, edge systems have limited resources in almost all aspects, such as computing power, memory capacity, and I/O bandwidth, etc. All these distinctions demand a systematic study on the power implications of deploying databases on edge devices.

Unfortunately, it still remains unclear how databases could affect energy consumption on a typical edge device, not to mention a thorough understanding on the associated system implications. In this paper, we select SQLite [12], LevelDB [13], and MongoDB [14], three representative relational and non-relational databases for our studies. We deploy these database systems on three popular low-power and compact-size Single Board Computers (SBCs), namely Raspberry Pi (RPi) 3 and 4 [15], and ODROID C2 [16] boards. In order to benchmark the devices coupled with databases, we create a set of representative workloads using *Yahoo! Cloud Serving Benchmark* (YCSB) [17] with different workload patterns. Through extensive experimental studies, we have obtained the first-hand results by deploying three typical databases on various edge platforms with distinct capabilities. Our purpose is to study the impact of workloads, device hardware, and databases on the energy consumption of edge systems. In particular, we desire to answer the following three important questions through our experimental studies.

- *Question #1: Workload patterns (e.g., object size distribution, read and write ratio, etc.) can have different effects on database activities. What is the impact of workload patterns on energy consumption of databases?*
- *Question #2: Edge systems have different capabilities in terms of computing, memory, and storage, and the energy consumption characteristics are inherently distinct. What is the impact of edge platforms on energy consumption?*
- *Question #3: Database performance may not always be linearly consistent with energy consumption. What is the*

relationship between performance and energy consumption when handling different database workloads?

Addressing the above questions can provide us important insight on energy consumption of deploying databases on edge devices. To the best of our knowledge, this paper is the first work focusing on energy consumption of modern databases on edge systems. We hope that the findings and insight presented in this paper can provide valuable guidance for system designers and practitioners to develop and deploy energy-efficient databases for edge computing.

The rest of the paper is organized as follows. Section II introduces the background of this work. Section III describes the methodology and experimental setup for the evaluation. Section IV presents the experimental results. Section V discusses the future work. Section VI gives the related work. The final section concludes this paper.

II. Background

A. Databases

SQLite [12] is a light-weight but full-featured embedded SQL database engine, which has been widely deployed in mobile devices such as smartphone, tablet, etc. SQLite manages data in the form of tables based on the rigid relational data model, and it supports complex query statements within one table or across multiple tables. In addition, a complete SQLite database is contained in a single database file, which provides the desirable cross-platform flexibility.

LevelDB [13] is a high-performance NoSQL database developed by Google, which can be used for serving different edge applications. LevelDB is a light-weight database system with only 350-KB library size, favoring resource-scarce edge devices. Meanwhile, it is an embedded database with no client-server model support, making it a better fit in the small-form-factor devices. LevelDB has also been selected as one of the best databases for the edge by AZ Big Media [19].

MongoDB [14] is a light-weight, document-oriented NoSQL database. It is also becoming an important platform for intelligent edge [20]. Unlike LevelDB and SQLite, it uses JavaScript Object Notation (JSON) [21] API for data exchange and Binary JSON (BSON) [22] for data storage. The flexible data structure enables developers to use the same data model and syntax from the edge to the data center for fast application development.

B. Single Board Computers

With the increasing popularity of edge computing, the low-power ARM architecture has been widely adopted in a variety of edge devices (e.g., ODROID, RPi, smartphone, tablet). Compared with x86-based architectures, which are widely used in conventional server systems, the ARM architecture can significantly improve power efficiency, though at the cost of performance to some extent.

Single board computers, such as RPi and ODROID boards, are being widely used in various areas, such as education, home automation, industrial automation, etc. With customized Linux systems (e.g., Ubuntu, Raspbian), it is friendly to Linux practitioners without a steep learning curve. More recently, in order to enhance the usability and computing power of the edge devices, 64-bit devices (e.g., RPi 3 B+, RPi 4, ODROID C2) have been developed to replace the original 32-bit devices (e.g., RPi 2, ODROID C1). The computing power is increased significantly (e.g., the 64-bit ARMv8 Cortex-A72 adopted in RPi 4). More details about the SBCs used in this study are shown in Table I.

III. Methodology and experimental setup

A. Single Board Computers

In order to understand energy consumption with different hardware platforms and edge devices, we perform our experimental studies on a diverse selection of representative Single Board Computers (SBCs) and storage devices. We select three SBC platforms, namely ODROID C2 (OC2), RPi 3 model B+, and RPi 4 model B, covering hardware from different vendors and various generations of products of the same product family. All SBCs are equipped with the identical 64-GB SanDisk Ultra microSDXC card. A high-speed 64-GB eMMC v5.0 flash module is used in OC2 through the on-board eMMC module socket. A 256-GB Samsung 850 PRO SSD is connected to RPi 4 through the USB interface. Since RPi 4 provides two types of USB interfaces (USB 2.0 and 3.0), which have very different data transfer speeds, we also compare energy consumption of SSDs based on the two USB interfaces. In this way, we can make a comprehensive comparison on energy consumption between different platforms and storage devices under the same condition.

In our experiments, we install Ubuntu 18.04 Linux on the three SBCs. Although it only supports up to 1-GB memory on RPi4 [23], it enables us to compare the three devices consistently with the same operating system. All the SD cards, eMMC module, and SSD use Ext4 file system. Unless otherwise specified, we use default setting for the other parameters. We deploy three database systems on each SBC, namely SQLite 3.22.0, LevelDB 1.1, and MongoDB 3.6.6. In order to minimize the interference from other components, we disconnect all the peripheral devices (e.g., monitor, keyboard, mouse) from the main board. The wireless communications (e.g., Wi-Fi, Bluetooth) are also disabled during the experiments.

B. Benchmark Tools

In order to make the performance comparison between different databases with various workloads, we use Yahoo! Cloud Serving Benchmark (YCSB) [17] to generate three workload distributions, *Zipfian*, *Latest*, and *Uniform*. The *Zipfian* workloads access items according to the Zipfian distribution, where some items are frequently accessed while the others are not; the *Latest* workloads are similar to *Zipfian* workloads except that the most recently inserted items are more popular; the *Uniform* workloads access items randomly.

Since YCSB works on JAVA virtual machine, its data loading process demands a large capacity of memory. However, some edge devices are equipped with very limited memory (e.g., 1 GB), which can cause the data loading phase to be

TABLE I: Device hardware specifications of the single board computers.

	ODROID C2 [18]	**Raspberry Pi 3 Model B+** [15]	**Raspberry Pi 4 Model B** [15]
Chipset	Amlogic S905	Broadcom BCM2837B0	Broadcom BCM2711
CPU	Quad Core 1.5GHz 64-bit ARMv8 Cortex-A53	Quad Core 1.4GHz 64-bit ARMv8 Cortex-A53	Quad Core 1.5GHz 64-bit ARMv8 Cortex-A72
Memory	2GB DDR3	1GB LPDDR2	2GB LPDDR4
Flash Storage	MicroSD, eMMC5.0	MicroSD	MicroSD
Release Year	2016	2018	2019
Price	$40	$35	$35

TABLE II: Performance characteristics of storage devices.

Storage Devices	**Write (MB/s)**	**Read (MB/s)**
SD card	14.1	41.3
eMMC	128	140
SSD with USB 2.0	33.3	35.7
SSD with USB 3.0	153	224

Note: Data is measured using Linux `dd` command with 4-KB blocks.

interrupted in the middle. In order to deal with this issue, we have developed a simple data-loading tool (around 600 lines of C++ code in Linux) to directly load the data into the databases via the database interfaces. In particular, we first generate workloads using YCSB, and then use our tool to load data into the target database, which ensures that the memory demand would not exceed the available capacity on board. After data loading is completed, we start benchmarking the database by running the workloads as usual.

C. Testbed Setup

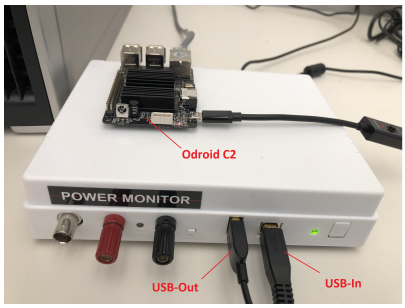


Fig. 1: The Monsoon power monitor and ODROID C2 board.

We use Monsoon Mobile Device Power Monitor [24] coupled with PowerTool 4.0.5.2 to measure the power and energy consumption of the SBC boards. Specifically, the SBC under test is connected to the USB-Out channel on the power monitor, which provides power for the normal run of the SBC. Another USB-In channel of the power monitor is connected to a DC power supply. The USB-In and USB-Out form a closed circuit, so that the power monitor can intercept and monitor the current variations, thereby providing the real-time power consumption data based on the current and voltage.

D. Measurement Metrics and Settings

Our measurement focuses on *energy consumption*, which is the amount of energy consumed to complete a task (in units of Joules), and *power consumption*, which is the energy consumption per unit time (in units of Watts). We also measure *bandwidth* and *latency*, two key performance metrics for databases, and study the relationship between energy consumption and database performance.

Our main experimental studies use the following settings for workloads, hardware platforms, and databases.

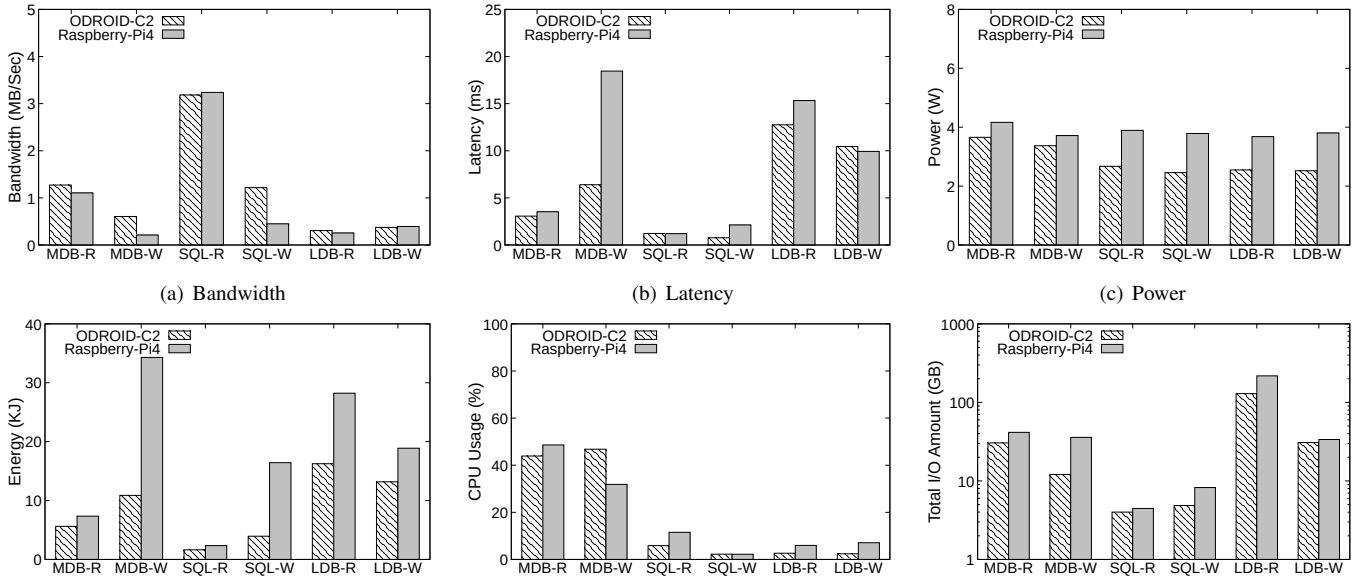
- **Workloads**: The *Zipfian* workload is chosen as a representative workload for the experiments. The total dataset size is around 4 GB. Item size is set to 1 KB. The total number of requests issued to the database reaches 2 millions, and the total amount of accessed data is about 2 GB. In addition, we run both read-intensive and write-intensive workloads respectively in each experiment for evaluating the database. For fair comparison, we flush the operating system page cache before each run. In our studies on the impact of workloads, we also include *Latest* and *Uniform* as additional workloads for studies. Specific settings can be found in Section IV-B.
- **Hardware platforms**: OC2 and RPi4 are selected as two representative edge devices. They are both equipped with 64-GB SanDisk SD cards. To fully exercise the quad-core processors on the devices, we use 4 threads to run most workloads. In the experiment of running the write-intensive workload on SQLite, we find that concurrent writes to SQLite can incur errors (e.g., an error signal `SQLITE_BUSY` is raised due to integrity issues), thus we use only one thread in this test. In our studies on the impact of hardware platform, we also include RPi3 as additional hardware for comparison. Specific settings are available in Section IV-C.
- **Databases**: In our experiments, the three databases, SQLite, LevelDB, and MongoDB run on the Ext4 file system. For all the three databases, we use the asynchronous I/O mode for better performance. Unless otherwise specified, the other parameters of databases use the default settings.

IV. Experimental Results

In this section, we present our experimental results and give a comprehensive analysis. We desire to seek answers in depth to the three key questions as mentioned previously. We will first give an overall comparison between the three database systems, and then present detailed experimental analysis from aspects of workloads, platforms, storage devices to obtain more insight on the effect of each individual component.

A. Overall Comparison

Figure 2 shows that SQLite achieves the best performance among the three databases for both read- and write-intensive workloads across the two platforms, OC2 and RPi4. For example, SQLite improves the bandwidth by a factor of 2.5 and 10.4 and reduces the latency by 60% and 90.4%, compared



(a) Bandwidth (b) Latency (c) Power

(d) Energy (e) CPU Usage (f) I/O Amount

Fig. 2: Overall comparison of performance, energy, and resource usage of three databases under *Zipfian* workload.

with MongoDB and LevelDB under read-intensive workloads on OC2. To explain the results, we have examined the CPU and disk usage to identify the bottleneck. We find that the CPU usage of MongoDB is about 43.9%. Although its is 38 percentage point (p.p) higher than that of SQLite, it is far from overloading the CPU. The main bottleneck is the limited storage I/O bandwidth on the SBCs (only dozens of megabytes per second). Unfortunately, both MongoDB and LevelDB generate more storage I/Os in terms of data amount than SQLite by a factor of 7.6 and 32.3, respectively.

We can also see that performance dominates the total energy consumption when the difference of power consumption is insignificant. Since SQLite achieves the highest bandwidth across different platforms, it takes the least time to complete the same workload. As such, the comparatively less execution time enables SQLite to consume the least energy. For example, although the power consumption of SQLite is 4.9% higher than that of LevelDB under read-intensive workloads on OC2, the significantly lower bandwidth of LevelDB makes it take longer time to complete the same task, which in turn causes it to consume more energy than SQLite by a factor of 9.9.

> ***Observation #1:*** *Generally, SQLite is the best choice if the specific information about the platform, storage medium, workload, etc. is unknown, due to its notably higher performance. Improving database performance is an effective method to reduce the energy consumption.*

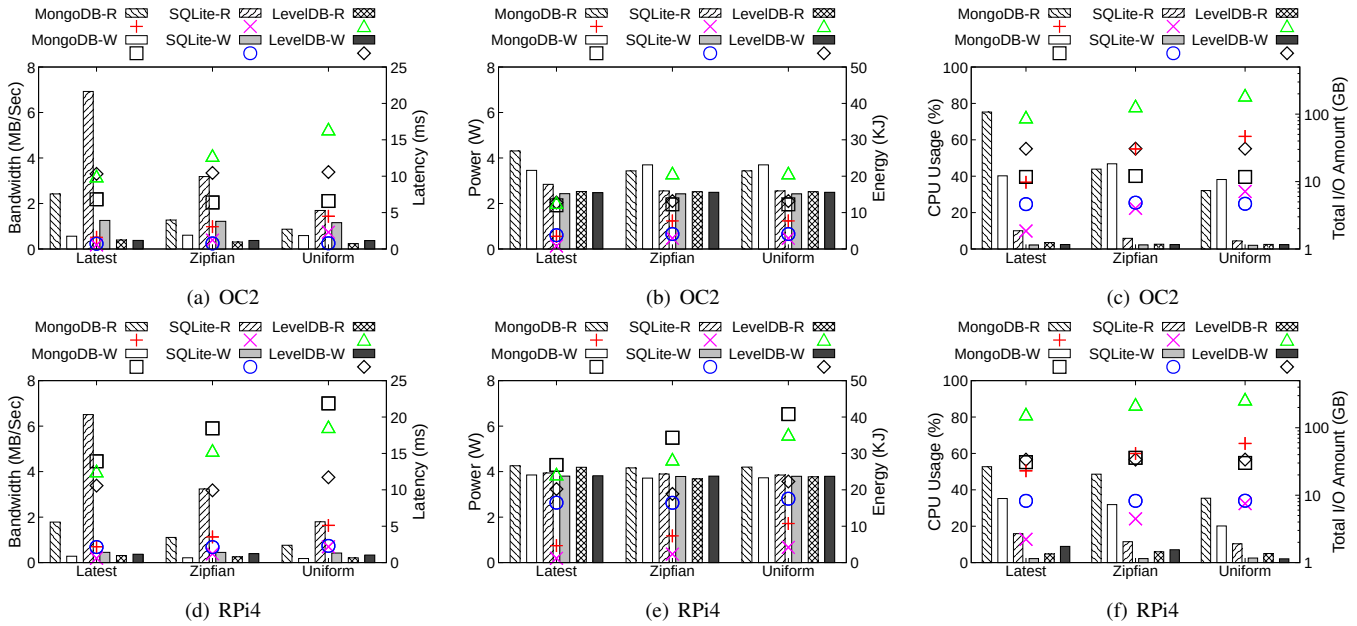
B. Effect of Workloads

Workloads can have a significant impact on energy consumption and performance of the databases. In this section, we benchmark the databases by adjusting the access patterns, read and write ratios, and key-value sizes to study the effect of workloads on databases. In the figures shown in this section, the bars represent the data for the left axis, and the points represent the data for the right axis.

1) ***Access pattern****:* We run three workload distributions in this test. As shown in Figure 3, the three databases in general achieve higher bandwidth and lower latency under *Latest* workload than the other two workloads on both OC2 and RPi4. For example, under read-intensive workloads on OC2, compared with *Zipfian* and *Uniform* workloads, the *Latest* workload with SQLite database achieves a higher bandwidth by a factor of 2.2 and 4.1, respectively. Meanwhile the latency is 54.1% and 75.6% lower, respectively. It is mainly because the *Latest* workload shows a stronger locality and more data accesses can be absorbed in memory. In the other two workloads, more storage I/Os are incurred due to the weaker locality. In fact, the total I/O amounts of SQLite under *Zipfian* and *Uniform* workloads are 2.2 times and 3.9 times of that running under *Latest* read-intensive workload on OC2.

2) ***Read and write ratio****:* Workloads with different amounts of read and write requests can impact the database behaviors, and consequently the performance and energy consumption. We configure YCSB to generate three representative workloads with different read and write ratios as read-intensive (R/W: 100/0), moderate (R/W: 50/50), and write-intensive (R/W: 0/100) using *Zipfian* distribution.

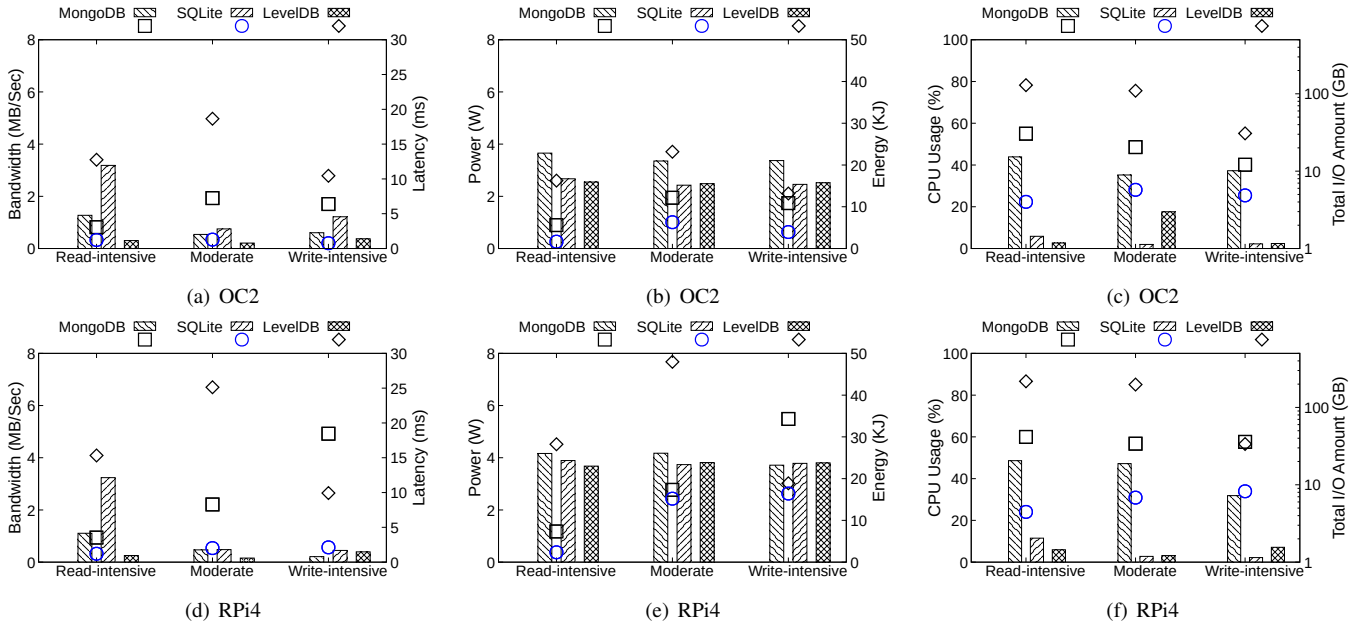
As shown in Figure 4, we find that SQLite and MongoDB achieve better performance under read-intensive workload on both OC2 and RPi4. For example, on OC2, SQLite and MongoDB achieve higher bandwidths with read-intensive workload than with write-intensive workloads by a factor of 2.6 and 2.1, respectively. Unlike the other two databases, LevelDB adopts the LSM-tree based data structure, which makes its write bandwidth 22.2% higher than its read bandwidth, and the total I/O amount under the write-intensive workload is only 23.8% of that under the read-intensive workload on OC2.



(a) OC2 (b) OC2 (c) OC2

(d) RPi4 (e) RPi4 (f) RPi4

Fig. 3: Performance, energy, and resource usage of three databases under *Latest*, *Zipfian*, and *Uniform* workloads.



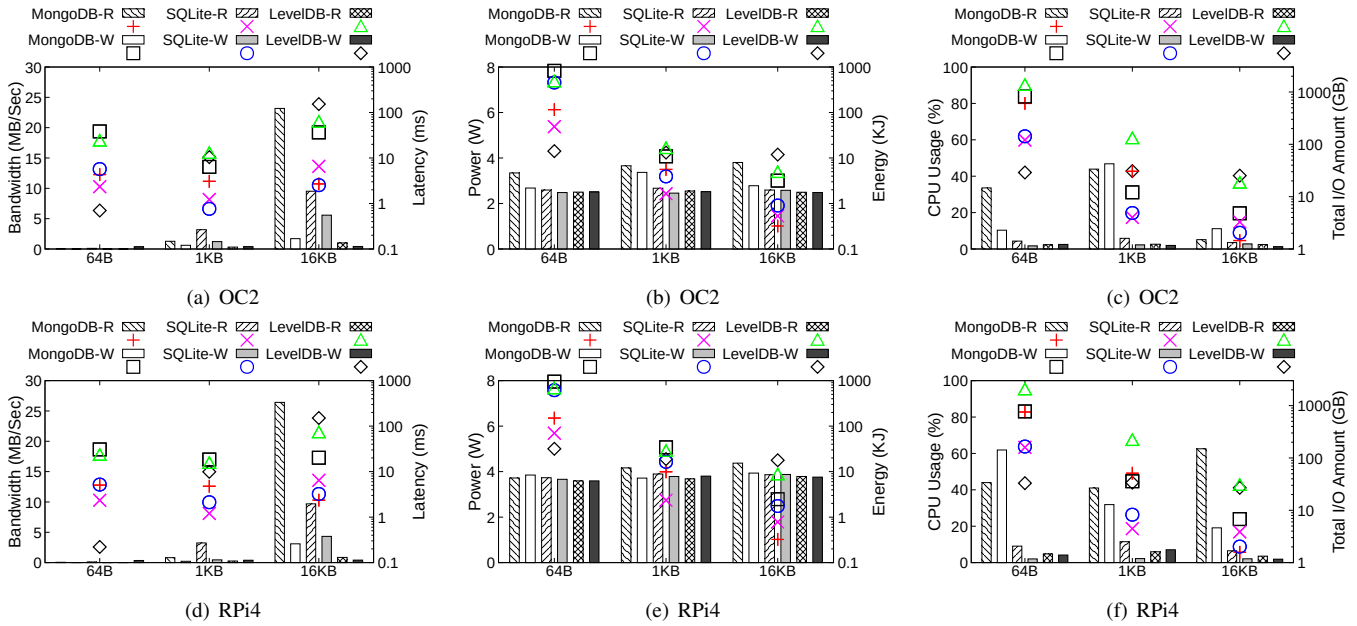
(a) OC2 (b) OC2 (c) OC2

(d) RPi4 (e) RPi4 (f) RPi4

Fig. 4: Performance, energy, and resource usage of three databases under *Zipfian* workload with different read-write ratios.

3) ***Key-value size****:* Key-value size also has a critical impact on databases. We select three representative key-value sizes, namely 64 B, 1 KB, and 16 KB to make comparison in Figure 5. In order to compare the incurred data amount of I/Os, we ensure the workloads with different key-value sizes access the same amount of data. We can see that the bandwidth increases significantly with the growing item size, because the I/O bandwidth can be better utilized with a larger key-value size. Latency is affected by I/O amplification, especially for small key-values. For example, the latency of SQLite with 64-B key-value size under read-intensive workloads on OC2 is 91.3% larger than that with 1-KB key-value size. This result can be explained by the much higher I/O amount involved with 64-B key-value size, which is 30.1 times of that with 1-KB key-value size. Further increasing the key-value size reduces I/O amplification, but the latency increases due to the larger amount of data that needs to be transferred for each request.

Compared to the other two databases, LevelDB benefits from the write-friendly LSM-tree data structure and performs well at handling small items under the write-intensive workload. For example, LevelDB outperforms SQLite and MongoDB in bandwidth under the write-intensive workload with 64-B key-value by a factor of 31.5 and 54.4 on OC2, and the corresponding I/O amount of LevelDB is much smaller,



(a) OC2 (b) OC2 (c) OC2

(d) RPi4 (e) RPi4 (f) RPi4

Fig. 5: Performance, energy, and resource usage of three databases under *Zipfian* workload with different key-value sizes.

which is only 20.3% and 3.5% of that caused by the other two databases respectively. MongoDB fits better for handling large-size key-value items under read-intensive workload, in terms of both bandwidth and latency, mainly due to the much less incurred I/O amount. For example, compared with SQLite and LevelDB under read-intensive workloads with 16-KB key-value based on OC2, MongoDB improves the bandwidth by a factor of 2.4 and 23.1, respectively, meanwhile reducing the latency by 59% and 95.6%.

The significant performance difference is also reflected in energy consumption. For example, for the small-size (64 B) write-intensive workload, LevelDB outperforms MongoDB in terms of bandwidth by a factor of 54.4, and MongoDB's energy consumption is 58.3 times of that with LevelDB.

> ***Observation #2:*** *LevelDB is more preferable to handle small-size write-intensive workloads, while MongoDB is more appropriate for working with large-size read-intensive workloads. The difference in power consumption caused by workloads has weak impact on the energy consumption compared to the significant performance gap.*

C. SBC Platforms and Storage Devices

To reflect the effect of different platforms and storage devices, we select three representative SBCs, namely ODROID C2, Raspberry Pi 3 B+, and Raspberry Pi 4 B, which cover the SBCs from different vendors as well as different generations of devices from the same vendor. We also use different flash storage devices, namely SD Card, eMMC, and SSD, to understand the effect of storage devices on databases. We use *Zipfian* workloads with 1-KB key-value size in this test.

1) ***SBC platforms****:* Figure 6 shows that OC2 achieves the best overall performance among the three platforms, while the performance of RPi3 is the worst. The key differentiating factor is the memory on device. OC2 is equipped with 2-GB memory on board, which allows it to serve more data requests from main memory rather than the slow SD card. For example, compared to running on RPi3, MongoDB generates 35.9% less I/O amount on OC2 under the read-intensive workload, outperforming RPi3 by a factor of 2.1 in bandwidth. Although the overall performance of RPi4 is relatively lower than OC2, it also outperforms RPi3 due to the faster memory (LPDDR4 vs. LPDDR2) and the better CPU (1.5GHz Cortex-A72 vs. 1.4GHz Cortex-A53).

2) ***Storage devices****:* Storage devices significantly affect the performance, as well as the power and energy consumption. We compare different databases running on SD, eMMC, and SSD. Since SSD with SATA interface cannot be directly connected to the SBC, we use a SATA/USB converter to connect the SSD and the SBC via the USB interface. RPi4 provides two different USB interfaces (USB 2.0 and 3.0), we benchmark the SSD using both interfaces.

By default, journaling of Ext4 is disabled for managing the system device (i.e., SD card). For fairness, we also evaluate the other attached devices including eMMC and SSD with journaling mechanism disabled. To further study the impact of journaling on energy efficiency of storage devices, we choose eMMC as an example to make comparison under two scenarios, with and without journaling. We use `tune2fs` tool in Linux to switch between the two journaling modes. We use the default journaling method (i.e., data=ordered) of Ext4, which only journals metadata operations [25].

In general, databases on SSD connected via the USB 3.0 interface achieve the highest performance, while databases on SD card have the lowest performance. For example, SQLite on SSD with USB 3.0 outperforms that on SD card by a factor of 2.5 in terms of bandwidth under the read-intensive workload.

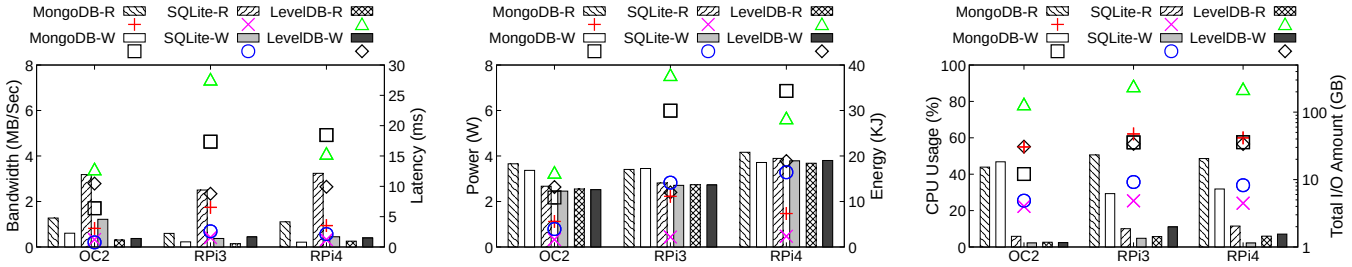


Fig. 6: Performance, energy, and resource usage of three databases under *Zipfian* workload on different SBC platforms.

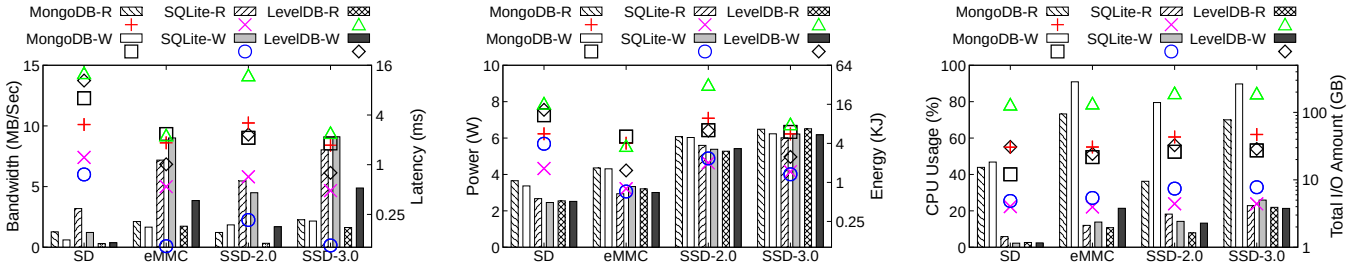


Fig. 7: Performance, energy, and resource usage of three databases under *Zipfian* workload with different storage devices.

The database performance on eMMC and SSD with USB 2.0 falls in the middle of the above-mentioned two. The overall performance is better on eMMC compared to that on SSD. For example, SQLite on eMMC improves the bandwidth by 31.6% compared with that on SSD with USB 2.0 under the read-intensive workload.

In addition, we also find that the default journaling method (i.e., data=ordered) has negligible impact on the performance and energy consumption, since it only journals the metadata. For example, SQLite based on eMMC without journaling improves the bandwidth only by 6.5% in contrast to that with journaling under write-intensive workload. Due to the space limit, it is not shown in the figure.

Since a relatively higher voltage is required, databases on SSD have much higher power consumption than running on eMMC. For example, the power consumption of SQLite on SSD with USB 3.0 is larger than that on eMMC under read-intensive workload by a factor of 2. Interestingly, although the database on SSD with USB 3.0 has a higher bandwidth and less running time compared to that running on eMMC, it has a larger energy consumption under the same situation. For example, SQLite running on SSD with USB 3.0 consumes 83% more energy than running on eMMC under read-intensive workload. It means that unlike our previous finding that performance largely determines energy consumption, power consumption is the deterministic factor in this experiment.

Observation #3: *The hardware resources on device, especially memory and storage, have a notable impact on the database performance due to the critical I/O bottleneck. Although SSD with USB 3.0 provides higher performance, the eMMC flash storage is more energy efficient due to the lower power consumption.*

V. Discussions

Our experimental studies have provided a set of valuable findings and system implications. In the meantime, we also note that due to the resource constraints, several aspects are worth further studies in the future. (1) Our current studies do not further differentiate energy consumption by different components of the database. An interesting question is, which part of the database design (e.g., logging, DB engine, compaction, etc.) contributes the most to the energy consumption, and how to optimize it by tuning or redesigning the database. In this work, we mainly treat the database as a whole. Disassembling the database for analysis could gain more insight. (2) In this work, our experiments are performed on SBC boards. Such compact computing devices are highly versatile and can be adopted to handle various kinds of edge workloads. In some application environments, however, such general-purpose SBC boards may not be suitable for the target workloads, such as machine learning, pattern recognition on the edge, which demand more computational power and storage capacity. In our future work, we plan to expand the device selection to cover a broader scope of application scenarios.

VI. Related Work

The limited energy resource is critical to edge devices. Prior research works have studied power and energy consumption on edge devices [26]–[33]. Khan et al. focus on the energy consumption of different data structures in edge computing. They find that the concurrent and locality-aware Delta tree outperforms B-link tree significantly in terms of energy efficiency and performance [28]. Chandra et al. propose three offloading energy-efficient approaches to improve the battery life of mobile devices, which includes using low-power CPU coexisting with the main CPU, using battery-backed RAM to reduce the flash I/Os, and offloading the computation-intensive

tasks to the nearby cloud [29]. Kaup et al. measure the relationship between power consumption and system resource utilization including CPU, Ethernet, Wi-Fi on Raspberry Pi (RPi), and further build a power model to improve the energy efficiency [27]. Bekaroo et al. measure the power consumption of the predefined key functionalities (e.g., device start-up, downloading a file, etc.) across five different devices including RPi, smartphone, etc. [30] Ardito et al. use sysbench and iperf to measure the power consumption of CPU as well as the network adapter based on RPi, and use linear regression to model the power consumption [31].

Recently, energy consumption on storage component of edge devices attracts interests in the community. Prior works find that energy consumption on storage components is a significant contributor that should not be neglected [8]–[11], [34]. Mohan et al. measure the energy consumption using different workloads on the SQLite database deployed on an Android smartphone, and they also analyze the impacts of various SQLite operations on the energy consumption [8]. Kim et al. develop an energy consumption model for I/O subsystem of wearable devices, thereby reducing energy consumption on I/O activities of SQLite database significantly [11]. Li et al. find that the energy consumption on storage software stack is 200 times more than the hardware through experiments on mobile platforms, and they also build a storage energy consumption model to better optimize energy utilization [10].

VII. Conclusion

In this paper, we present a comprehensive experimental study to understand the energy efficiency of databases on single board computers for edge computing. We have made several important findings on the performance, power, and energy consumption of three representative databases on edge devices and also discussed the related system implications. We hope that this work can provide valuable guidance for system designers and practitioners to design and deploy databases in an energy-efficient way in edge computing environments.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive feedback and comments. This work was supported in part by U.S. National Science Foundation under Grant CCF-1910958.

References

[1] 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.

[2] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable IoT architecture based on transparent computing," *IEEE Network*, vol. 31, no. 5, pp. 96–105, 2017.

[3] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, and C.-T. Lin, "Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment," *IEEE Access*, vol. 6, pp. 1706–1717, 2017.

[4] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[5] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[6] R. Sharma, S. Biookaghazadeh, B. Li, and M. Zhao, "Are existing knowledge transfer techniques effective for deep learning with edge devices?" in *Proceedings of 2018 IEEE International Conference on Edge Computing (EDGE)*, 2018, pp. 42–49.

[7] Q. Yang, R. Jin, and M. Zhao, "SmartDedup: Optimizing deduplication for resource-constrained devices," in *Proceedings of 2019 USENIX Annual Technical Conference (USENIX ATC)*, 2019, pp. 633–646.

[8] J. Mohan, D. Purohith, M. Halpern, V. Chidambaram, and V. J. Reddi, "Storage on your smartphone uses more energy than you think," in *Proceedings of the 9th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage)*, 2017.

[9] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with Eprof," in *Proceedings of the 7th ACM European Conference on Computer Systems (EuroSys)*, 2012, pp. 29–42.

[10] J. Li, A. Badam, R. Chandra, S. Swanson, B. Worthington, and Q. Zhang, "On the energy overhead of mobile storage systems," in *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST)*, 2014, pp. 105–118.

[11] J. Kim, S. Kim, J. Yun, and Y. Won, "Energy efficient IO stack design for wearable device," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 2152–2159.

[12] "SQLite," https://www.sqlite.org/index.html, 2021.

[13] "LevelDB," https://github.com/google/leveldb, 2021.

[14] "MongoDB," https://www.mongodb.com/, 2021.

[15] "Raspberry Pi," https://www.raspberrypi.org/, 2021.

[16] "ODROID," https://www.hardkernel.com/, 2021.

[17] "YCSB," https://github.com/brianfrankcooper/YCSB, 2021.

[18] "ODROID-C2," https://odroid.com/dokuwiki/doku.php?id=en:c2_hardware, 2021.

[19] "The best IoT databases for the edge – an overview and guide," https://azbigmedia.com/business/the-best-iot-databases-for-the-edge-an-overview-and-guide/, 2020.

[20] "The Internet of Things (IoT)," https://www.mongodb.com/use-cases/internet-of-things, 2021.

[21] "JavaScript Object Notation," https://www.json.org/, 2021.

[22] "Binary JSON," https://bsonspec.org/, 2021.

[23] "64 bit Ubuntu 18.04 on Pi4B," https://www.raspberrypi.org/forums/viewtopic.php?t=245908,, 2019.

[24] "Monsoon power monitor," https://www.msoon.com, 2021.

[25] "Ext4 filesystem," https://www.kernel.org/doc/Documentation/filesystems/ext4.txt, 2021.

[26] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of 2010 USENIX Annual Technical Conference (USENIX ATC)*, 2010, pp. 21–21.

[27] F. Kaup, P. Gottschling, and D. Hausheer, "PowerPi: Measuring and modeling the power consumption of the Raspberry Pi," in *Proceedings of 39th Annual IEEE Conference on Local Computer Networks*, 2014, pp. 236–243.

[28] A. M. Khan, I. Umar, and P. H. Ha, "Efficient compute at the edge: Optimizing energy aware data structures for emerging edge hardware," in *Proceedings of 2018 International Conference on High Performance Computing and Simulation (HPCS)*, 2018, pp. 314–321.

[29] R. Chandra, S. Hodges, A. Badam, and J. Huang, "Offloading to improve the battery life of mobile devices," *IEEE Pervasive Computing*, vol. 15, no. 4, pp. 5–9, 2016.

[30] G. Bekaroo and A. Santokhee, "Power consumption of the Raspberry Pi: A comparative analysis," in *Proceedings of 2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies*, 2016, pp. 361–366.

[31] L. Ardito and M. Torchiano, "Creating and evaluating a software power model for Linux single board computers," in *Proceedings of the 6th Int'l Workshop on Green and Sustainable Software*, 2018, pp. 1–8.

[32] J. Huang, A. Badam, R. Chandra, and E. B. Nightingale, "WearDrive: Fast and energy-efficient storage for wearables," in *Proceedings of 2015 USENIX Annual Technical Conference*, 2015, pp. 613–625.

[33] P. Olivier, J. Boukhobza, E. Senn, and H. Ouarnoughi, "A methodology for estimating performance and power consumption of embedded flash file systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 15, no. 4, pp. 1–25, 2016.

[34] D. T. Nguyen, G. Zhou, X. Qi, G. Peng, J. Zhao, T. Nguyen, and D. Le, "Storage-aware smartphone energy savings," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2013, pp. 677–686.