Storage Performance Evaluation for IoT Gateway Implementation Using Raspberry Pi 2

Zan-Wai Kong, Boon-Yaik Ooi, and Chee-Siang Wong
Faculty of Information and Communication Technology, Department of Computer Science, Universiti Tunku Abdul Rahman, Jalan Universiti, Bandar Barat,
31900 Kampar, Perak, Malaysia.
zforzen@1utar.my

***Abstract*—IoT gateway is a core module exists in many of the IoT architectures that plays a role to connect WSNs to the internet, or specifically to the Cloud. However, conventional internet gateway is not sufficient to be IoT gateway. One of the most critical issue faced by IoT gateway is unstable internet connection especially when using cellular network. This work proposes that IoT gateway should have temporary storage to mask network issue. The objective of this work is to find out the most efficient solution for IoT gateway with temporary storage based on the elements of hardware, scheduler and storage method including database versus flat file on Raspberry Pi and NAND flash. From our experimental results, we found that the most efficient solution for temporary storage in IoT Gateway is using 4-threaded flat file I/O with Deadline scheduler.**

***Index Terms*—Gateway; Internet of Things; NAND Flash.**

I. Introduction

Internet of Things (IOT) is an emerging paradigm built up with a continuum of uniquely addressable 'things' which is able to communicate with each other through a worldwide dynamic network (internet), with the bolster of protocols and approaches such as IPV6, ubiquitous computing, pervasive computing, sensing technology and other domains. It is a system that bridges real and digital world is formed to enable symbiotic interactions seamlessly between the two parties [1]. In other words, IoT describes a vision where a huge network of uniquely identified smart objects (things) with different characteristics (sensors, actuators etc.) connected at any-time, any-place, any-thing, working together to provide variety of services on demand to end users [2]. The range of services varies from services to citizens (e.g., smart home, mobile health) to industrial applications (e.g., smart grid, efficient transportation and logistics) [3].

Generally, IoT architectures commonly consisted of three working layers: perception layer, network layer and application layer[4]–[6]. In brief, the layers are mainly to transmit the perceptual information from WSN (perception layer) to end user devices (application layer), through network protocols and internet (network layer) [4]. According to Xu *et. al* [7], the communication architecture can be categorized into three groups. (1) Front-end proxy solution: connection is performed by middleware proxy and no direct connection between WSNs and the Internet. (2) Gateway solution: a gateway is located and acts as a bridge between WSNs and the Internet. (3) TCP/IP overlay solution, an overlay network constructed on either WSNs or the Internet. In this work, we will focus on the gateway solution.

Most of the existing IoT frameworks with gateway solution can be classified into two major groups: with and without Cloud Computing. IoT architectures without Cloud Computing proposed that end user devices in application layers are allowed to accessed into sensor gateways directly in networks layer to obtain perceptual information; whereas architectures with Cloud Computing have cloud server(s) in between of end devices and sensor gateways, which store and process the information uploaded from sensor gateways and allow end devices access to obtain information on demand [7]–[11]. Relatively, approaches with Cloud Computing tend to be more practically applicable as the data generation rate of numerous sensors could be extremely high and requires a high processing power back-end machine for data processing job that cannot be handled by light-weighted sensor gateway [6], [12], [13].

According to [4], an IoT gateway should have minimum requirements of (1) multi interfaces to handle incoming data from different sources and forward data to different destination, (2) protocol conversion to handle and standardize heterogeneous data and (3) high manageability in terms of remotely managing connected sensors and gateway itself. To achieve the IoT vision of "connectivity at any-time, any-place", wired broadband to the internet would not be suitable as a medium for gateway-cloud connection. Cellular network with less geographical limitations becomes a better alternative. However, inconsistency of cellular networks quality might at some point causes the upload bandwidth unable to cope with the incoming data rate from sensors. It eventually leads to network congestion, which is an issue that cannot be resolved by IoT gateway with only minimum requirements.

Azzam *et. al* [6] proposed a framework of IoT Gateway with temporary storage, allows incoming data from sensors to be temporary stored while pending to be uploaded to cloud, resolved the issue of network congestion. Yet, temporary storage performance is still an ambiguous subject. Since the continuous incoming data rate from sensor networks could be very high, I/O speed of temporary storage should also be decent enough to handle it.

The main contributions of this paper is test and find out the most efficient method for temporary storage implemented on

SD card based on elements of hardware, scheduler and storage method. The rest of the paper is explained as below: Section 2.0- Related work, Section 3.0- Methodology of tests, Section 4.0- Experimental Results and Section 5.0- Conclusion.

II. Related Work

In terms of data storage, from low to high level, the most essential consideration would be the storage hardware. In the implementation of embedded computers as an IoT gateway, flash memory is widely used as a storage medium, where SD card is a paradigm of flash memory for modern day programmable embedded computers. Therefore, SD card speed is undoubtedly the basis of storage performance. SD cards in the market are classified with grades according to their read/write speed and performance consistency. UHS classes are the index to identify the minimum speed of an SDHC/ SDXC card, cards with minimum speed of 10MB/s and 30MB/s are graded as UHS class 1 and UHS class 3 respectively [14]. The SD card models with grades above, SanDisk SDHC ultra (UHS class 1) and SanDisk SDHC Extreme Pro (UHS class 3) are the example of flash memory that are used in electronic devices in the market.

The higher level in data storage is the I/O scheduler used in the embedded computer OS. Noop and Deadline are two schedulers available in the Linux kernel used by embedded computers. The Noop scheduler is the simplest I/O scheduler for Linux kernel, works by inserting all incoming I/O requests into FIFO queue and performs request merging, while the Deadline scheduler's main goal is to guarantee a start service time for an I/O request by imposing a deadline on all I/O operations to prevent resource starvation.

Upon hardware and kernel level, the next computer architecture level that affects the data storage performance is the I/O method. There are a lot of ways to write continuous incoming data into files and allowed to read the stored data concurrently. Since embedded computers nowadays do come with multicore processors, multithreaded I/O also helps in data storage [15]. One of the most popular implementations is by using thread-safe database that features data organization and query, eases data insertion and searching work. Another method is using simple flat file storing, which contains only basic formatting and record with no structured relationship. Flat files storing might cumber the data searching work, but it tends to have the highest speed due to its simplicity.

III. Methodology

The main objective of this work is to test the most efficient method to store the sensor data, based on the following criteria: (i) SD card type, (ii) I/O scheduler and (iii) storage method. Generally the whole work is done on Raspberry Pi 2 Model B with a 900MHz quad-core ARM Cortex-A7 CPU and 1GB RAM which acts as an IoT gateway platform.

As mentioned above, the first criterion is the SD card type. In this research, two types of SDHC card are used as test mediums, which are SanDisk Ultra with 48MBps read/write speed and SanDisk Extreme Pro with 90/95MBps read/write speed as advertised. The second criterion is the I/O scheduler, with the variable of Noop and Deadline scheduler built-in with the Raspberry Pi 2. The last criterion is read/write method, where the database PostgresSQL, SQLite and flat file I/O are compared in term of read/write speed based on input size and number of threads.

Table 1
Hierarchy of Storage Performance Test

	Raspberry Pi 2 Model B		
Level 1	SanDisk Ultra (Low-end)		SanDisk Extreme Pro (High-end)
Level 2	Deadline		No-op
Level 3	PostgresSQL	SQLite	Flat file I/O

The test will be performed by hierarchy as shown in Table 1, starting from the lowest architecture level. The option with better result will be used as a basis to carry on the next level test.

Level 1: A set of simple file I/O tests using flat file read/write is run on both SD cards with default I/O scheduler using C++ program. The test involves concurrent read/write rows of string from/to text files, with the manipulated variables of (1) number of rows range from 5k to 5M and (2) number of threads range from 1 – 32. Each individual test is repeated for 10 times for normalized results. The card with better performance will proceed to level 2 test.

Level 2: The same file I/O test similar as previous level is run on the same SD card using different I/O scheduler. The variance with better performance will be selected as default I/O scheduler of next level test.

Level 3: The C++ test program is modified to work with SQL databases. The similar tests with same variables working on different I/O platforms (PostgresSQL, SQLite, Flat file I/O) are performed under the environment set by previous level tests.

Eventually, the temporary storage solution with best performance will be decided with the combination of best performers from three levels.

IV. Experimental Results

This section shows the experimental results and comments according to the result observations.

A. Criterion 1: SD card Type

According to Figure 1, SanDisk Ultra has slower read/write speed as number of threads increases (increment becomes very significant from 8 threads), whereas SanDisk Extreme Pro according to Figure 2 gives consistently higher I/O speed regardless the number of threads.

Table 2 shows the time variance obtained by repeating same test to read/write 5M rows of data using flat file storage method. The overall time variance of SanDisk Ultra significantly overwhelmed SanDisk Pro Extreme's, showed that SanDisk Extreme Pro has higher performance consistency. Therefore, the remainder of tests are set up with the SanDisk Extreme Pro.

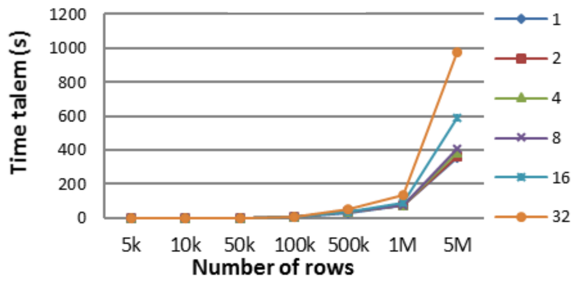

Figure 1: Read/write time vs number of rows with different number of thread for SanDisk Ultra

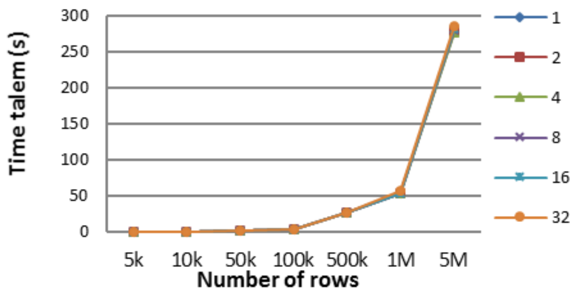

Figure 2: Read/write time vs number of rows with different number of thread for SanDisk Extreme Pro

Table 2
Variance of time taken for repeated I/O test

Number of threads	Variance	
	Low end NAND flash storage (SanDisk Ultra)	High-end NAND flash storage (SanDisk Extreme Pro)
1	89.47	1.48
2	148.99	0.95
4	433.53	2.70
8	112.12	1.11
16	987.60	4.69
32	1757.68	10.06

B. *Criterion 2: Test on the effect of I/O scheduler speed on data storage performance.*

According to Figure 3 and Figure 4, the two I/O schedulers did not give significant difference in terms of performance. The tests gave almost the same results under all variables. It might be due to the SanDisk Extreme Pro itself was initially performing with its maximum speed even with slower scheduler. Since the two do not give much difference, the Deadline scheduler is selected to carry out next level test as it is the default I/O scheduler of Raspberry Pi 2.

C. *Criterion 3: Test on the effect of storage method on data storage performance.*

Databases in general has much higher overheads compare to flat file I/O. As shown in Figure 7, flat file I/O able to read/write 5M rows of data within 300 seconds (100k rows takes less than 3 seconds) with different number of threads (4 threads gives insignificantly shortest time), whereas Postgres SQL according to Figure 6 has best case of handling 100k rows of data with 195 seconds using 32 threads and SQLite according to Figure 5 takes around 2400 seconds to achieve 100k rows of data regardless of the number of thread. The overheads occurred in databases is because database has to ensure the ACID (Atomicity, Consistency, Isolation, Durability) properties.

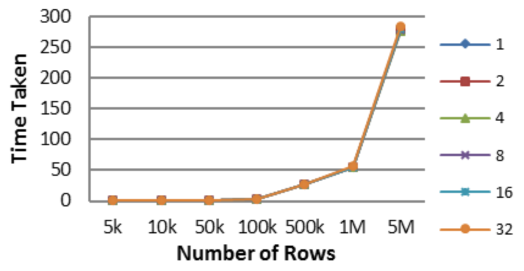

Figure 3: Flat file I/O read/write time vs number of rows with different number of thread (Deadline)

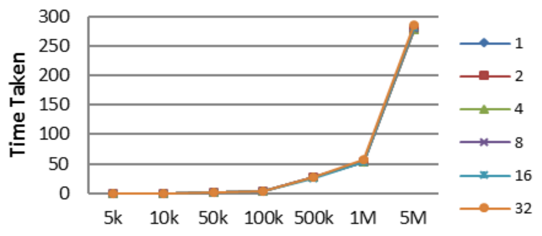

Figure 4: Flat file I/O read/write time vs number of rows with different number of thread (Noop)

V. CONCLUSION

According to the experimental results, the most efficient solution for temporary storage in Smart IoT Gateway is using 4 threaded flat file I/O with Deadline scheduler on high end NAND flash storage such as SD Extreme Pro. This combination gives the fastest storage performance in terms of read/write speed. The flat file I/O platform might require further enhancement to deal with data-query-alike tasks in IoT systems. However, with the wide margin between performances of flat file I/O and databases, it is highly possible to integrate minimum query features to flat file I/O and still having better performance than regular databases. Interestingly, from our experimental result, we found that there are performance differences between of low end and high end NAND flash storages even though the Raspberry Pi 2 bus speed is capped at around 19.4MBps. Although both low end and high end NAND flash storages can perform faster than Raspberry Pi 2 bus speed, the high end NAND flash storage does have performance advantages over the low end NAND flash storage especially in handling concurrent IO activity much better

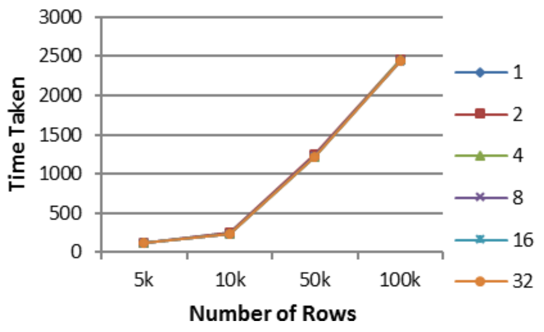

Figure 5: SQLite read/write time vs number of rows with different number of thread

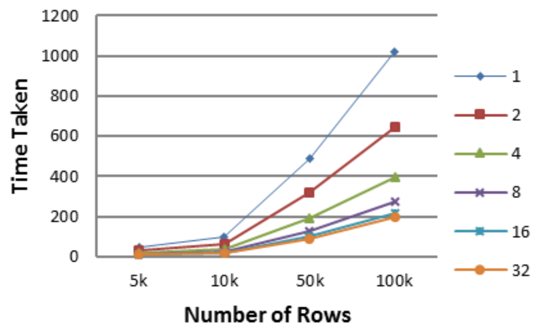

Figure 6: PostgresSQL read/write time vs number of rows with different number of thread

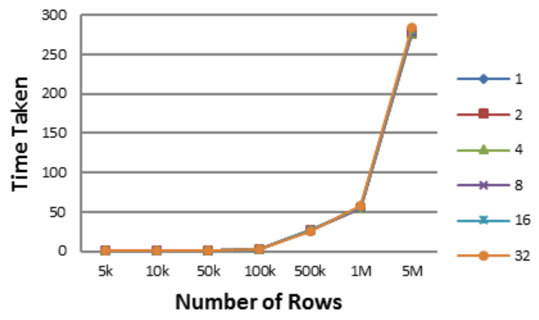

Figure 7: Flat file I/O read/write vs number of rows with different number of thread

References

[1] E. Borgia, "The Internet of Things vision : Key features , applications and open issues," *Comput. Commun.*, vol. 54, pp. 1–31, 2014.

[2] L. Coetzee and J. Eksteen, "The Internet of Things – Promise for the Future ? An Introduction," *IST-Africa Conf. Proc.*, pp. 1–9, 2011.

[3] L. Costantino, N. Buonaccorsi, C. Cicconetti, and R. Mambrini, "Performance analysis of an LTE gateway for the IoT," *2012 IEEE Int. Symp. a World Wireless, Mob. Multimed. Networks, WoWMoM 2012 - Digit. Proc.*, 2012.

[4] H. Chen, X. Jia, and H. Li, "A Brief Introduction To IoT Gateway," *ICCTA2011*, p. 4, 2011.

[5] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "IOT Gateway: BridgingWireless Sensor Networks into Internet of Things," *2010 IEEE/IFIP Int. Conf. Embed. Ubiquitous Comput.*, pp. 347–352, 2010.

[6] M. Aazam and E.-N. Huh, "Fog Computing and Smart Gateway Based Communication for Cloud of Things," *2014 Int. Conf. Futur. Internet Things Cloud*, pp. 464–470, 2014.

[7] Ran,Xu, Y. Shuang-Hua, L. Ping, and C. Jiangtao, "IoT Architecture Design for 6LoWPAN Enabled Federated Sensor Network," *11th World Congr. Intell. Control Autom.*, pp. 2997–3002, 2014.

[8] S. K. Datta, C. Bonnet, and N. Nikaein, "An IoT gateway centric architecture to provide novel M2M services," *2014 IEEE World Forum Internet Things*, pp. 514–519, 2014.

[9] D. Min, Z. Xiao, B. Sheng, and G. Shiya, "Design and implementation of the the multi-channel RS485 IOT gateway," *2012 IEEE Int. Conf. Cyber Technol. Autom. Control Intell. Syst.*, pp. 47–88, 2012.

[10] C. P. Kruger, G. P. Hancke, S. Networks, and H. Kong, "Rapid Prototyping of a Wireless Sensor Network Gateway for the Internet of Things Using o ff -the-shelf Components," *2015 IEEE Int. Conf. Ind. Technol.*, pp. 1926–1931, 2015.

[11] M. Ha, S. H. Kim, H. Kim, K. Kwon, N. Giang, and D. Kim, "SNAIL gateway: Dual-mode wireless access points for WiFi and IP-based wireless sensor networks in the internet of things," *2012 IEEE Consum. Commun. Netw. Conf. CCNC'2012*, pp. 169–173, 2012.

[12] M. Aazam, P. P. Hung, and E. N. Huh, "Smart gateway based communication for cloud of things," *IEEE ISSNIP 2014 - 2014 IEEE 9th Int. Conf. Intell. Sensors, Sens. Networks Inf. Process. Conf. Proc.*, no. April, pp. 21–24, 2014.

[13] J. De Huang and H. C. Hsieh, "Design of gateway for monitoring system in IoT networks," *Proc. - 2013 IEEE Int. Conf. Green Comput. Commun. IEEE Internet Things IEEE Cyber, Phys. Soc. Comput. GreenCom-iThings-CPSCom 2013*, pp. 1876–1880, 2013.

[14] "Speed Class - SD Association." [Online]. Available: https://www.sdcard.org/developers/overview/speed_class/. [Accessed: 22-Sep-2015].

[15] B. W. Low, B. Y. Ooi, and C. S. Wong, "Scalability of database bulk insertion with multi-threading," *Commun. Comput. Inf. Sci.*, vol. 181 CCIS, no. PART 3, pp. 151–162, 2011.