Journal of Knowledge Information Technology and Systems

ISSN 1975-7700

http://www.kkits.or.kr

Power-Aware Hybrid Encoding Scheme for Mobile Cloud Services on Open Source Hardware Platforms

Boyun Eom, Choonhwa Lee[*]

Division of Computer Science and Engineering, Hanyang University

ABSTRACT

Cloud computing has been developed in various ways and many services have been migrated to this relatively new paradigm since its advent. One representative example is mobile cloud computing, MCC, in short. With introduction of faster network access and better empowered mobile devices than their predecessors, the integration of cloud computing and mobile devices would be the natural technology evolution path, which leads to the vision of mobile cloud computing. To provide the mobility capability, mobile devices have battery as an energy source and if the battery is discharged no more cloud service is available. The biggest advantage of mobile device which is mobility causes to the biggest disadvantage. Motivated by the importance of conserving energy on mobile devices, in this paper, we introduce an adaptive energy-aware encoding scheme for open source hardware platforms for RDA-based mobile cloud services which we have named in our work. We expect that an open source hardware platform can be a next important mobile device as well as a leader of IoT (Internet of Things). Through experiments we have found that the characteristic of hardware components on a device need to be considered carefully and the result showes that our proposed method is effective in saving energy consumption for providing RDA-based mobile cloud computing.

KEYWORDS: Mobile clouds, Energy consumption, Open source Hardware platforms

ARTICLE INFO: Received 21 September 2015, Revised 8 October 2015, Accepted 8 October 2015.

1. Introduction

It is expected that up to 26 billion devices will be connected to internet by 2020 [1]. Integrated sensors and actuators produce conceivable data almost in real time and provide a huge amount

[*]Corresponding author is with the Division of Computer Science and Engineering, Hanyang University, 222 Wangshimni-ro, Seongdong-gu, Seoul, 133-791, KOREA.
E-mail address: lee@hanyang.ac.kr

of information. As a producer of innumerable amount of data, these billions of smart devices builds the foundation for the Internet of Things (IoT). The open source hardware platform (OSHP), such as Arduino and Raspberry pi provides internet environment for these tiny electronic devices and enables these to be smart things. In this paper, we define an RDA(remote display architecture)-based mobile cloud services which have been implemented with a remote display framework. With the belief that these OSHP can be a major consumer for RDA-based mobile cloud services, we have applied the adaptive hybrid encoding scheme[2] onto an OSHP and developed it in more efficient way for saving energy consumption on those boards.

This paper is organized as follows. In Section 2, related work is discussed. Power-aware encoding scheme for OSHP is introduced in Section3. Then experimental result is presented in Section 4 and Section 5 concludes the paper.

2. Related Work

[3] defines mobile cloud computing as an integration of cloud computing technology with mobile devices to make the mobile devices resource-full in terms of computational power, memory, storage, energy, and context awareness. Many services have been proposed and implemented in the form of mobile cloud [4] and there have been many approaches to approaches to overcome the limit of energy on mobile devices[2, 5, 6].

An adaptive encoding scheme to reduce power consumption in delivering mobile cloud services has been introduced in [2]. In this paper, authors have investigated the correlation between energy consumption and various encoding types, and proposed a power-aware encoding scheme to maximize the connection lifespan of battery-driven devices to the cloud. In order to conserve power on the mobile devices, the proposed scheme tries to lower power consumption by CPU and wireless network interface card and two encoding schemes -RAW and ZRLE- were chosen. One of the most popular remote display solutions, VNC [7, 8, 9, 10] has been modified for the adaptive encoding scheme. The client shows the lowest CPU utility usage when it processes RAW type encoded data and the smallest energy consumption by wireless network interface card had been observed when the data from a server were encoded with ZRLE type which compresses graphic data in highest rate. In the approach, once the server begins the hybrid encoding scheme, it switches between the two encoders -*RAW* and *ZRLE*- depending on real-time transmission rate, adaptively.

3. Power-aware Hybrid Encoding Scheme for OSHP

The purpose of this study is to minimize energy consumption on OSHP by modifying the adaptive power-aware encoding scheme [2] for longer connection to use remote display architecture-based cloud services. Dislike smart phones explained in [2], devices of OSHP which have relatively lower performance of CPU require more considerations because the energy consumptions by CPU on these devices are usually smaller. While smart phones and tablet PCs have relatively high performance CPU(s) such as quad cores, a raspberry pi B+ is equipped with a single core at 700MHz. of CPU. This difference has several key performance implications.

As <Figure 1> shows, the rate of current discharge against CPU utilization becomes higher when the frequency of CPU clock is faster [10]. This implies the saving by reducing CPU utilization on a raspberry pi is significantly smaller than that of mobile devices such as smart phones or tablet PCs used in [2]. This is one important thing considered for OSHP.

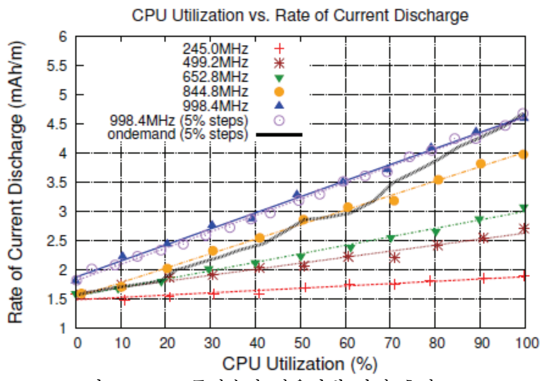


그림 1. CPU 주파수와 사용량별 방전 효과
Figure 1. Effect of CPU frequency and percentage CPU utilization on the rate of current discharge [10]

In this work, we used two encoding schemes, RAW and ZRLE, which were used in [2] and using a UNIX command, the CPU usage for processing each encoding scheme on an OSHP had been observed. For the case of ZRLE, CPU usage showed 7 ~ 25% but 1 ~ 8% of CPU was used when RAW was processed on raspberry pi B+. Considering the mean values for two cases (e.g., 16 for ZRLE and 4 for RAW), it can be assumed that roughly four more CPU power is required in processing ZRLE.

The energy consumption on a device during duration is,

$$\text{Energy Consumption} = \text{Current Discharge Rate} \times \text{Voltage} = (CDR_{dis}+CDR_{tr}+CDR_{CPU}+..) \times V$$

If the transmission rate is increased, the CDR(Current Discharge Rate) of transmission Rate, CDR_{tr}, also gets higher but the CPU utility becomes decreased as well as CDR of CPU, CDR_{CPU}.

Let's put $\triangle CDR_{tr}$ is the difference of CDRs for two transmission rates and $\triangle CDR_{cpu}$ is the difference of CDRs for different CPU utility. $\triangle CDR$ is the summation of $\triangle CDR_{tr}$ and $\triangle CDR_{cpu}$.

- $\Delta CDR_{tr} = CDR_{tr(n')} - CDR_{tr(n)}$
- $\Delta CDR_{cpu} = CDR_{cpu(n')} - CDR_{cpu(n)}$
- $\Delta CDR = \Delta CDR_{tr} + \Delta CDR_{cpu}$

Now, let's consider a situations where the transmission rate is increased twice (i.e., 400KB to 800KB) and the CPU utility is decreased in half (i.e., 80% to 40%).

To estimate current discharge rate(CDR) for Wi-Fi, following formula (1) is used [10]. Here, CDR is represented in mAH, and χ indicates a transmission rate in KB.

$$CDR_{tr}(\chi) = 0.0011\chi + 2.739 \tag{1}$$

The difference of CDRs by increasing transmission rate,ΔCDR_{tr}, from 400 to 800 as below:

$$\triangle CDR_{tr}=CDR_{tr}(400)-CDR_{tr}(800) = -0.44$$

Also, we can get $\triangle CDR_{cpu}$ from Figure 1 as below:

$$\Delta CDR_{cpu}=CDR_{cpu(80)}-CDR_{cpu(40)}=0.6$$

Therefore, the difference of CDR would be,

$$\Delta CDR = \Delta CDR_{tr} + \Delta CDR_{cpu} =-0.44+0.6$$

If the difference of CDR ($\triangle CDR$) becomes larger than 0 like in this case, it means that there is no saving of energy consumption by changing CPU utility and transmission rate. Therefore, finding target transmission rate which can make the difference of CDR gets less than 0 is required.

To calculate desirable target transmission rate when the CDR of CPU is decreased by half, it is needed to find the value which makes the difference of CDR($\triangle CDR$) becomes smaller than 0, such as (2):

$$\begin{aligned}\Delta CDR &= \Delta CDR_{tr} + \Delta CDR_{cpu}\\ &= [(CDR_{tr(target)}-CDR_{tr(current)})-(CDR_{cpu(current)}-CDR_{cpu(current/2)})]\\ &= [((0.0011 \times targetTransRate + 2.739) - (0.0011 \times currentTransRate + 2.739)) - (CDR_{cpu(current)}-CDR_{cpu(current/2)}] < 0 \quad (2)\end{aligned}$$

From the formula (2), the target transmission rate can be,

$$targetTransRate = currentTransRate + \frac{1}{0.0011} \times (CDR_{cpu(current)} - CDR_{cpu(current/2)})$$

After calculating the target transmission rate, now we need to find the order for RAW type encoded data under target transmission rate. Let's put the task T is to receive and decode n frames from a server. Then we can decompose task T as n numbers of task t, such as

$$T = t_1+t_2+t_3+..t_n$$

Also, the CDR which affects energy consumption can be decomposed as below:

$$sum(CDR(t_i))_{i=1,..n}=sum(CDR_{tr}(t_i)+CDR_{cpu}(t_i)+CRR_{disp}(t_i)+..)$$

For RDA-based mobile cloud, each t can be thought as an encoded image which is sent from server and needs to be decoded on client side. If ZRLE encoding was selected, considering the compression ratio, the bandwidth of four *t*s is the same size of that of one RAW-encoded 2]. So using the bandwidth calculated, the server gets the order for RAW typed image after the sequence of ZRLE type encodings.

Modified adaptive algorithm for OSHP to save energy is provided in <Figure 2>.

4. Experimental Measurements

4.1 Experimental Setup

We have deployed an RDA server onto the IaaS service, AWS(Amazon Web Service) EC2.

For a client, we have chosen Raspberry pi B+ and attached a touch screen. Below is the hardware specification of raspberry pi type B+ we used for OSHP.

```
(1) do{
(2)    msg=ReceiveFrameBufferUpdateRequest();
(3)    if (msg == firstRequest){
(4)        encode_with_ZRLE();
(5)        startTimer();   msgCnt = 1;
(6)    }else {
(7)       if (msg.batteryStustus < threshold){
(8)           transRate = calculate_Transmission_Rate();
(9)           size(raw) =   calculate_raw_data_size();
(10)          cdr_cpu_gain =   getCdr_cpu((int)(cpuUtility/4));
(11)         transValue = (getCdr_tr(transRate) + (cdr_cpu_gain) -   2.739) * (10000/11);
(10)         if (transRate < transValue)
(11)                 left = transValue -transRate;
(12)         else
(13)                 left = transValue;
(14)         t = (packetSize(raw) / left) * NumofFrameBufferUpdateRequestPerSec;
(15)         if (msgCnt % (coefficientValue* (t+1)) == 0){
(16)                 encode_with_RAW();
(17)                 msgCnt = 1;
(18)         }else{
(19)               encode_with_ZRLE();
(20)               msgCnt++;
(21)         }
(22)    } else
(23)    encode_with_ZRLE();
(24)  }
(25)  sendFrameUpdateMsg();
```

그림 2. 제안하는 알고리즘
Figure 2. Proposed Algorithm

RDA Client

- OS : Rasbian
- CPU : 1 (700MHz Broadcom BCM2835)
- Memory : 512MB RAM
- USB Wifi Adaptor (802.11n/b/g)
- 4-inch touch screen (HDMI)
- DSI display port for connecting the Raspberry Pi touch screen display

Under this circumstance, two different types of task -static and dynamic workloads- have been performed to measure power consumption on OSHP as in [2]. The first set is the case of editing a document where screen updates occurs less often and updated areas are limited. The second is the case of playing a video clip where the screen needs to be refreshed frequently and at a faster rate. These two tasks are referred as to static and dynamic in this paper, respectively.

The power consumption was measured with an external power meter as in <Figure 3>. The power cable of the raspberry pi plugged into the power meter which is goes into the power outlet. In this experiment, we measure only total energy consumption during the task.

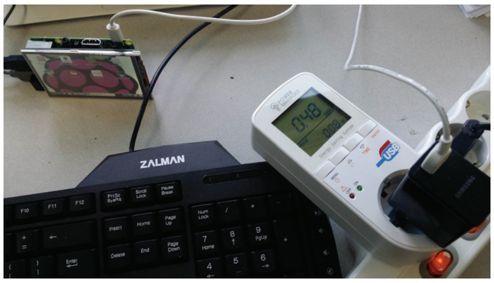


그림 3. 파워미터기를 이용한 에너지 측정
Figure 3. Energy Measurement Using a Power Meter

Static Workload

For a static workload, the scenario is to perform 3,000 keyboard inputs on an editor application, gedit and measure total energy consumption for three encoding schemes – zlib, zrle and hybrid. To emulate the situation as if a user press key evenly, we had added test codes for firing 3,000 key pressed events automatically on VNC Client side so that a key input value is sent to the server and the client receives the result screen from the server. It was called when the VNC client received *framebufferUpdate* message from a server that the *keypressed* events are called. This task has been performed for three encoding schemes and the energy has been measured for each task as shown in <Figure 4>.

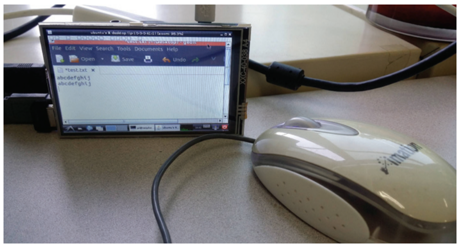


그림 4. 라즈베리파이에서 정적 작업 테스트 화면
Figure 4. Static workload test on raspberry pi

Dynamic Workload

A famous video clip, “Big Buck Bunny” whose resolution is 854 x 480 had been played for 9 minutes on the VNC Server and the total energy consumed on raspberry pi had been obtained as illustrate in <Figure 5>. The total energy consumption to use this kind of mobile cloud service on raspberry pi b+ had been also measured for three different encoding schemes.

그림 5. 라즈베리파이에서 동적 작업 테스트 화면
Figure 5 .Dynamic workload test on raspberry pi

4.2. Experimental Result

We took the average measurement after taking away the minimum and maximum values for each encoding scheme. The average energy consumption and durations for static workload with three encoding schemes are provided in <Figure 6>.

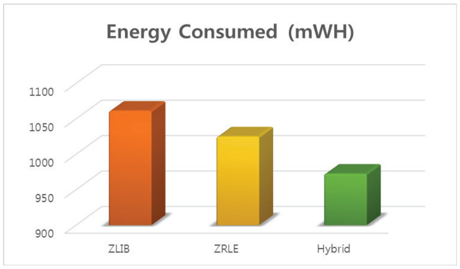


그림 6. 정적작업 테스트 결과
Figure 6. Static workload test on raspberry pi

The average energy consumptions for dynamic workload with three encoding scheme are provided in <Figure 7>.

Although the test result has showed the effectiveness of the proposed algorithm, it has not shown such significant improvement in saving energy on a client side as in [2]. This proves that different hardware characteristics of client devices result different pattern in consuming energy.

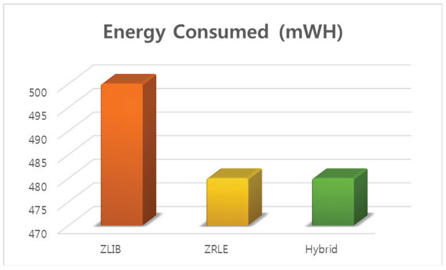


그림 7. 동적작업 테스트 결과
Figure 7 .Dynamic workload test on raspberry pie

We provide our experimental result on <Table 1>.

표 1. 테스트 결과
Table 1. Experimental Result

Encoding Type	ZLIB	ZRLE	Hybrid
Energy Consumption during Static Workload (mWH)	1,061.25	1,025	972.5
Energy Consumption during Dynamic Workload (mWH)	500	480	480

5. Conclusions

The mobile cloud service provides an enhanced experience on real-time basis, as well as scalability and it is required the ease of integration, reliability, and data synchronization as other cloud services. In this paper, we have proposed a modified power-efficient encoding scheme for open source hardware platforms which can be promising clients for RDA–based mobile clouds. The proposed hybrid encoding scheme considers power consumption by network interface cards as well as CPUs for its encoding decisions and switches encoding mode adaptively reflecting bandwidth.

Experimental result showes that the proposal outperforms one of well-known remote display solutions, VNC.

References

[1] Gartner. Available: http://www.gartner.com/newsroom/id/2636073

[2] B. Eom, C. Lee, H. Lee, and W. Ryu, *An adaptive remote display scheme to deliver mobile cloud services*, IEEE Transactions on Consumer Electronics, Vol. 58, 2014.

[3] M. O. A.Khan, S.Madani, and S.Khan, *A survey of mobile cloud computing application models*, Communications Surveys & Tutorials, IEEE, Vol. 16, pp. 393-413, 2013.

[4] D. H. a. L. Chen, *Mobile cloud for assistive healthcare (MoCAsH)*, presented at the Services Computing Conference (APSCC), Hangzhou, 2010.

[5] J. M. R. Murmuria, A. Stavrou, and J. Voas, *Mobile application and device power usage measurements*, Proceedings of the 6th IEEE International Conference on software Security and Reliability (SERE), pp. 147-156, 2012.

[6] A. A. K. Saipullah, N. Ismailand, and Y. Soo, *Measuring power consumption for image processing on android smartphone*, American Journal of Applied Sciences, Vol. 9, pp. 2052-2057, 2012.

[7] B. T. L. Zhang, Z. Qian, and Z. Wang, *Accurate online power estimation and automatic battery behavior based power model generation for smartphones*, in IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), Scottsdale, USA, pp. 105-114, 2010.

[8] T. Richardson, and K. Wood, *The RFB protocol*, AT&T Laboratories Cambridge, Jan. 1998.

[9] P. Simoens, F. Turck, B. Dhoedt, and P. Demeester, *Remote display solutions for mobile cloud computing*, Computer, Vol. 44, No. 8, pp. 46-53, 2011.

[10] H. Ko, J. Lee, and J. Kim, *Implementation and evaluation of fast mobile VNC systems*, IEEE Transactions. Consumer Electron., Vol. 58, No. 4, pp. 1211-1218, 2012.

오픈소스 하드웨어 플랫폼에서 모바일 클라우드 서비스를 위한 에너지 고려 하이브리드 인코딩 방법

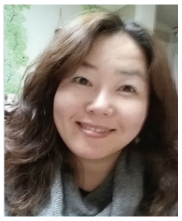
엄보윤, 이춘화
한양대학교 컴퓨터공학부

요 약

비교적 새로운 패러다임인 클라우드 컴퓨팅은 다양한 분야로 발전하고 있으며, 많은 서비스들이 클라우드 컴퓨팅 형태로 옮겨지고 있다. 대표적인 예가 모바일 클라우드 컴퓨팅, 즉 MCC(Mobile Cloud Cumputing)이다. 빠른 네트워크 접근력과 성능 좋은 모바일 단말들의 출현으로, 클라우드 서비스들과 이런 모바일 단말들과의 결합은 클라우드 컴퓨팅의 발전을 위해 너무도 자연스러운 단계로 보여진다. 그런데, 모바일 단말의 가장 큰 장점인 이동성은, 또한 이 단말들의 가장 큰 단점을 야기시킨다. 이동성을 위해 사용하는 밧데리가 서비스를 계속 사용할 수 있느냐 없느냐를 좌우하기 때문이다. 모바일 단말에서 이러한 에너지 절약의 중요성을 깨닫고, 이 논문에서는 오픈소스 하드웨어 플랫폼에서 원격 화면 전송 프레임워크를 사용하여 구현되는 클라우드 서비스들을 사용할 때 에너지 효율성을 고려하여 처리할 수 있도록 하기 위한 인코딩 방법을 제안한다. 단말에서 에너지 소모가 가장 큰 CPU와 무선랜카드의 에너지 사용량을 적응적으로 줄임으로써 전반적인 에너지 소모량을 줄이고자 하였다. 라즈베리파이를 이용한 두 종류 테스트 시나리오의 반복적인 테스트를 통해 제안하는 적응적 인코딩 방법이 기존의 RFB보다 에너지 효율성 면에서 뛰어남을 확인하였다.

Acknowledgments

This work was supported by the MSIP(Ministry of Science, ICT&Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2015-H8501-15-1006) supervised by the IITP(Institute for Information &communications Technology Promotion) and by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(No. 2013R1A1A2007616



Boyun Eom received her M.S. degree in computer & information science & engineering from University of Florida, Gainesville, US, in 2005. Currently, she is a senior researcher at ETRI. Her research interests include cloud computing, smart home, database, and middleware and services computing technology.

E-mail address: eby@etri.re.kr

Choonhwa Lee is an associate professor in the Division of Computer Science and Engineering at Hanyang University, Seoul, South Korea. He received his B.S. and M.S. degrees in computer engineering from Seoul National University, South Korea, in 1990 and 1992, respectively, and his Ph.D. degree in computer engineering from the University of Florida, Gainesville, in 2003. His research interests include cloud computing, peer-to-peer and mobile networking and computing, and services computing technology.

E-mail address: lee@hanyang.ac.kr