

I/O Scheduler Scheme for User Responsiveness in Mobile Systems

Jong Woo Park[†] · Jun Young Yoon^{††} · Dae-Wha Seo^{†††}

ABSTRACT

NAND flash storage is widely used for computer systems, because of it has faster response time, lower power consumption, and larger capacity per unit area than hard disk. However, currently used I/O scheduler in the operating system is optimized for characteristics of the hard disk. Therefore, the conventional I/O scheduler includes the unnecessary overhead in the case of the NAND flash storage to be applied. Particularly, when the write requests performed intensively, garbage collection is performed intensively. So, it occurs the problem that the processing of the I/O request delay. In this paper, we propose the new I/O scheduler to solve the problem of garbage collection performs intensively, and to optimize for NAND flash storage. In the result of performance evaluation, proposed scheme shows an improvement the user responsiveness by reducing 1% of the average read response time and 78% of the maximum response time.

Keywords : NAND Flash Storage, User Responsiveness, I/O Scheduler, Vertical Optimization

모바일 시스템에서 사용자 반응성을 고려한 입출력 스케줄링 기법

박 종 우[†] · 윤 준 영^{††} · 서 대 화^{†††}

요 약

낸드 플래시 저장장치는 하드디스크보다 응답시간이 빠르고, 전력 소모가 적으며, 단위 면적 당 저장 용량이 큰 장점을 가지고 있어 컴퓨터 시스템의 저장장치로 널리 사용되고 있다. 그러나 현재 사용되고 있는 운영체제의 입출력 스케줄러는 하드디스크의 특성에 최적화되어 있다. 따라서 기존의 입출력 스케줄러는 낸드 플래시 저장장치에 적용될 경우에 불필요한 오버헤드가 포함된다. 특히 쓰기 요청이 집중적으로 수행될 경우에 가비지 컬렉션 또한 집중적으로 수행된다. 이로 인하여 입출력 요청의 처리가 지연되는 문제점이 발생된다. 본 논문에서는 가비지 컬렉션이 집중적으로 수행됨으로 인하여 순간적으로 읽기 입출력 요청의 응답시간이 증가되는 것을 방지하고 낸드 플래시 저장장치에 최적화된 입출력 스케줄러를 제안하였다. 성능평가를 통하여 제안 기법이 평균 읽기 응답시간을 1%, 최대 응답시간을 78% 줄여 사용자 반응성을 향상시켰음을 보였다.

키워드 : 낸드 플래시 저장장치, 사용자 반응성, 입출력 스케줄러, 수직적 최적화

1. 서 론

낸드 플래시 저장장치는 하드디스크보다 응답시간이 빠르고, 전력 소모가 적으며, 단위 면적 당 저장 용량이 큰 장점을 가지고 있어 컴퓨터 시스템의 저장장치로 널리 사용되고 있다[1].

그러나 컴퓨터 시스템에서 사용되는 운영체제는 하드디스크에 최적화된 입출력 계층을 그대로 사용하고 있다. 이는 낸드 플래시 저장장치에서 수행되는 FTL(Flash Translation Layer) 알고리즘에서 외부 인터페이스를 하드디스크와 동일하게 가상화하기 때문에 가능하다. 그렇지만 낸드 플래시 저장장치의 특성은 하드디스크와 다르기 때문에 낸드 플래시 저장장치에 기존의 입출력 계층을 적용할 경우에는 불필요한 오버헤드가 포함된다[2, 3].

또한 기존의 입출력 계층을 사용할 경우에는 쓰기 연산이 집중적으로 수행되는 경우가 발생됨으로 인하여 가비지 컬렉션(Garbage Collection) 또한 집중적으로 수행된다. 이 때 입출력 요청의 응답시간이 순간적으로 증가되는 현상이 나타나고 있다[4].

* 이 논문은 2016년도 정부(교육부)의 지원으로 한국연구재단의 지원을 받아 수행된 기초연구사업입니다(NRF-2015R1D1A1A01057697).

† 정회원: 경북대학교 전자공학부 박사과정

†† 춘회원: 경북대학교 일베디드소프트웨어연구센터 연구원

††† 종신회원: 경북대학교 전자공학부 교수

Manuscript Received : May 2, 2016

First Revision : August 18, 2016

Accepted : September 2, 2016

* Corresponding Author : Dae-Wha Seo(dwseo@ee.knu.ac.kr)

따라서 본 논문에서는 읽기 입출력 요청의 응답시간이 순간적으로 증가되는 문제점을 해결하고, 낸드 플래시 저장장치에 최적화된 입출력 스케줄러를 제안하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 사용자 반응성과 읽기 입출력 응답시간을 줄이기 위한 관련 연구를 설명하고, 3장에서는 제안하는 입출력 스케줄링 기법의 수행 과정에 대하여 설명한다. 4장에서는 구현 방법을 기술하고, 5장에서는 성능 평가의 결과에 대하여 설명하며, 마지막으로 6장에서는 결론을 기술한다.

2. 관련 연구

이 장에서는 사용자 반응성에 대하여 설명하고, 읽기 입출력 요청의 응답시간을 줄이기 위하여 수행된 연구에 대해서 살펴본다.

사용자 반응성은 컴퓨터 시스템을 평가할 때에 사용자의 입력에 대한 반응을 말하며, 읽기 입출력 요청의 지연으로 사용자에게 데이터를 적시에 제공하지 못할 경우에 성능이 떨어진다[5]. 따라서 읽기 입출력 요청의 응답 시간을 줄이기 위하여 QASIO[5]와 Read Over Write (ROW)[6] 등의 연구가 수행되었다.

QASIO는 읽기 입출력 이전에 수행되어야 할 쓰기 입출력 요청이 입출력 계층에서 우선순위가 낮아 읽기 입출력 요청의 수행이 지연되는 현상을 해결하기 위하여 이 쓰기 입출력 요청을 우선적으로 처리함으로써 읽기 입출력 요청의 지연을 방지하였다.

ROW는 단순히 읽기 입출력이 있을 경우에 무조건 읽기를 우선적으로 처리하는 방법으로 읽기 입출력 요청의 지연을 방지하였다.

그러나 이 연구들은 평균 읽기 응답시간을 줄이는 효과는 있었지만, 순간적으로 증가하는 읽기 응답 시간을 해결하지는 못하고 있다. 그 이유는 서론에서 언급한 바와 같이 쓰기 입출력 요청의 집중 시에 발생되는 가비지 컬렉션 수행의 집중과 시스템 자원이 제한적인 모바일 시스템에 대하여 고려하지 않았기 때문이다.

따라서 평균 읽기 응답시간을 줄이는 것과 함께 최대 읽기 응답시간을 줄여 사용자 반응성의 QoS 요구사항을 만족시키기 위한 기법에 대한 연구가 필요하다.

3. 단순·공평 큐잉 입출력 스케줄링 기법(SFQ)

이 장에서는 순간적으로 증가하는 읽기 응답시간을 방지하여 사용자 반응성을 향상시키는 것에 초점을 두는 기법인 단순·공평 큐잉 입출력 스케줄링 기법(Simple Fairness Queuing I/O Scheduling Scheme, SFQ)에 대하여 기술한다.

3.1 기본 구조

Fig. 1은 SFQ의 기본 구조를 나타낸다. SFQ는 요청 큐 관리자, 입출력 요청 디스패처, 안정성 관리자 등의 기능적인 부분과, 기능 수행을 위한 자료 구조인 입출력 요청 큐로 구성된다.

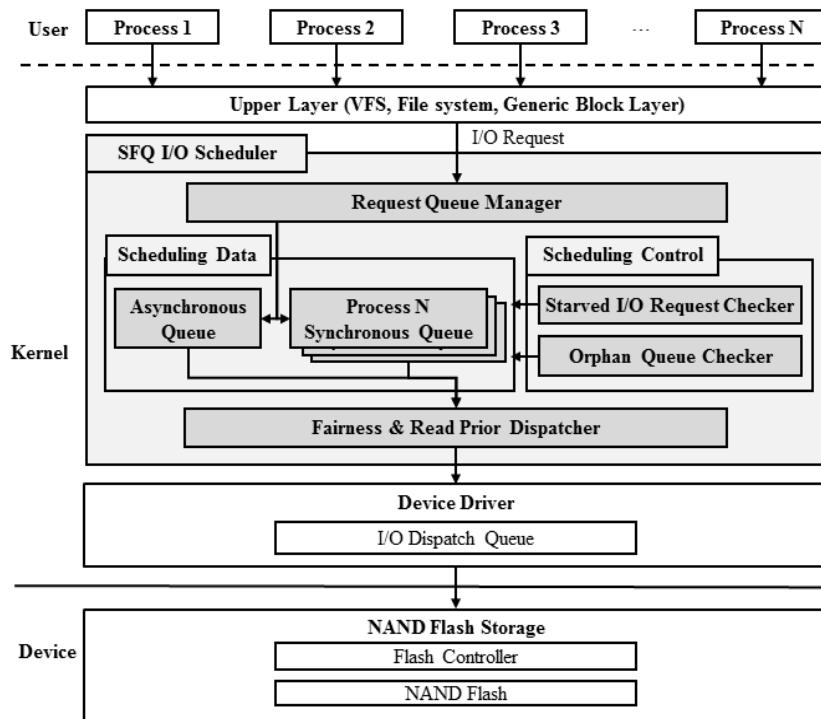


Fig. 1. Basic Structure of SFQ

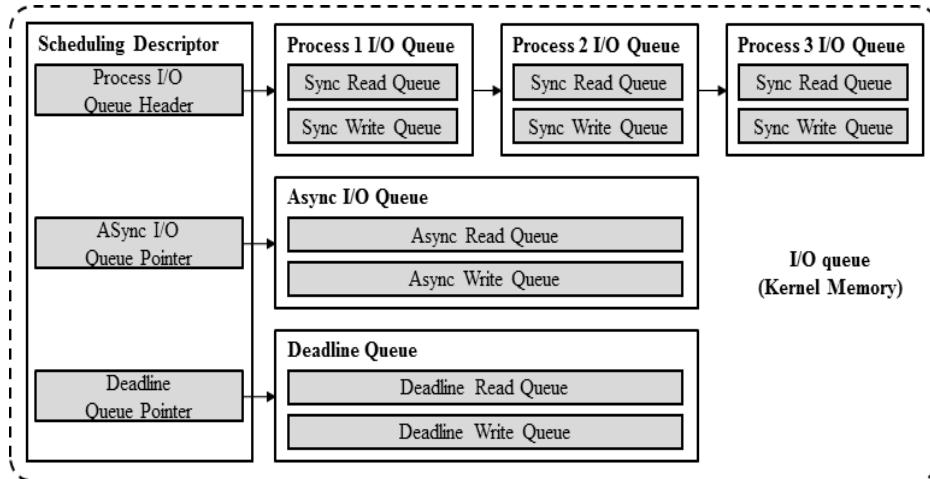


Fig. 2. Structure of I/O Request Queue

요청 큐 관리자는 상위 단계에서 제출한 입출력 요청을 읽기/쓰기 및 동기/비동기 여부에 따라 해당 입출력 큐에 등록하며, 입출력 요청 디스패처는 스케줄링 정책에 따라 처리할 입출력 요청을 디바이스 드라이버로 보낸다. 그리고 안정성 관리자는 입출력 스케줄러에서 발생되는 기아상태와 고아상태를 방지한다. 입출력 요청 큐는 프로세스 별 동기 입출력 큐와 하나의 비동기 입출력 큐로 구성되며, 동기 입출력 큐와 비동기 입출력 큐 각각은 읽기 입출력 큐와 쓰기 입출력 큐로 구성된다.

3.2 입출력 요청 큐

Fig. 2는 입출력 요청 큐의 구조를 나타낸다. 입출력 요청 큐는 스케줄링 디스크립터, 프로세스 별 입출력 큐, 비동기 입출력 큐, 그리고 데드라인 큐로 구성된다.

스케줄링 디스크립터는 입출력 스케줄러를 운영체제에 등록할 경우에 생성되는 자료구조이다. 프로세스 별 입출력 큐는 PID를 이용하여 해당되는 프로세스와 연결되며, 프로세스 입출력 큐 헤더에 이중 연결 리스트로 연결되어 있어 순차적으로 접근할 수 있다. 그리고 비동기 입출력 큐와 데드라인 큐는 각각의 큐 포인터로부터 접근할 수 있다. 각각의 입출력 큐는 읽기 및 쓰기 입출력 큐를 포함하고 있어 입출력 요청의 유형에 따라 분리하여 저장할 수 있다.

3.3 요청 큐 관리 기법

요청 큐 관리 기법은 상위 단계에서 내려오는 입출력 요청을 어느 입출력 큐에 등록할 것인가를 결정한다.

Algorithm 1은 입출력 요청의 특성에 따라 해당되는 입출력 큐에 등록하는 과정을 나타낸다. 알고리즘을 수행함에 따라 입출력 요청은 해당되는 프로세스 입출력 큐에 등록하거나 비동기 입출력 큐에 읽기/쓰기 여부에 따라 분류하여 저장한다. 입출력 요청에 해당하는 프로세스 입출력 큐의

존재 여부는 PID를 비교하여 확인되며, 존재하지 않을 경우에는 프로세스 입출력 큐를 생성하여 리스트에 추가된다. 그리고 모든 입출력 요청은 데드라인 큐에도 등록한다. 결과적으로 이 과정은 이후에 수행될 디스패치 알고리즘을 수행하기 위한 자료 구조를 생성하는 역할을 한다.

3.4 입출력 요청 디스패칭 기법

입출력 요청 디스패칭 기법은 입출력 요청 큐에 저장된 입출력 요청을 선택하여 디스패치 하는 과정을 수행한다.

입출력 요청 디스패칭 기법은 크게 입출력 큐를 선택하는 과정과 선택된 입출력 큐에서 어떤 입출력 요청을 디스패치 할 것인지를 선택하는 과정으로 나뉜다.

Algorithm 2는 입출력 큐를 선택하는 과정을 나타낸다.

프로세스별 입출력 큐는 라운드-로빈 정책으로 선택하여 프로세스 별 공평성을 보장하며, 프로세스 입출력 큐의 선택 횟수 제한을 소모하면 비동기 큐를 선택하게 하여 비동기 입출력 요청이 주기적으로 선택되게 한다.

Algorithm 3은 입출력 요청을 선택하는 과정을 나타낸다.

입출력 큐가 한번 선택되면 읽기 입출력 요청 2개, 쓰기 입출력 요청 1개를 처리한다. 그리고 읽기 입출력이 없거나 1개만 있더라도 쓰기 입출력 요청이 처리되게 하여 한번 선택된 입출력 큐는 무조건 입출력 요청을 3개 처리된다.

3.5 안정성 관리 기법

안정성 관리 기법은 입출력 스케줄러에서 발생되는 버그를 방지하기 위하여 수행되는 부가적인 기법이다.

기아상태 확인자는 일정 주기로 수행되어 만료시간이 지난 입출력 요청을 디스패치하며, 고아상태 확인자는 종료된 프로세스의 고아상태 큐를 해제함으로 불필요한 입출력이 수행되는 것을 방지한다.

Algorithm 1. 입출력 요청 큐 관리 알고리즘

```

while (I/O 요청)
    도착 시간을 저장;
    if (I/O 요청이 동기이면) {
        if (I/O를 요청한 프로세스에 해당하는 프로세스 I/O 큐가 없으
            면) 프로세스 I/O 큐를 생성하여 리스트에 추가;
        I/O 요청에 따라 동기 읽기 큐 또는 동기 쓰기 큐에 등록;
    } else
        I/O 요청에 따라 비동기 읽기 큐 또는 비동기 쓰기 큐에 등록;
    I/O 요청에 따라 테드라인 읽기 큐 또는 테드라인 쓰기 큐에 등록;
end while

```

Algorithm 2. I/O 큐 선택 알고리즘

```

while (true)
    if (동기 큐의 선택 횟수가 제한 이내이고, 프로세스 I/O 큐가
        있으면) {
        /* default = 3 */

        현재 선택된 프로세스 I/O 큐의 다음 프로세스 I/O 큐를
        디스패치 할 큐로 설정; 동기 큐의 선택 횟수++;
        if (고아상태 아님) {알고리즘 3 수행; return;}

        if (비동기 I/O 큐가 있으면) {
            비동기 I/O 큐를 디스패치 할 큐로 설정;
            동기 큐의 선택 횟수 = 0;
            if (고아상태 아님) {알고리즘 3 수행; return;}}
        return NULL;
    } end while

```

Algorithm 3. 입출력 요청 선택 알고리즘

```

while (true)
    if (읽기 횟수와 쓰기 횟수의 합이 큐 제한 이상이면) /*
        default = 3 */

        읽기 횟수 = 0; 쓰기 횟수 = 0; return;
    if (읽기 I/O 요청이 있고, 읽기 횟수가 남아 있으면) {
        default = 2 */

        선정된 큐의 읽기 요청 디스패치;
        읽기 횟수++; break;}

    if (쓰기 I/O 요청이 있고, 쓰기 횟수가 남아 있으면) {
        default = 1 */

        선정된 큐의 쓰기 요청 디스패치;
        쓰기 횟수++;}

end while

```

4. 구 현

본 논문에서 제안하는 SFQ는 모바일 시스템과 유사한 사양을 가지고 있는 ODROID-XU[6] 개발보드의 리눅스 커널에 구현하였으며, 리눅스 엘리베이터의 기본적인 API만을 사용하여 불필요한 API 호출 오버헤드를 줄였다.

구현에 사용된 ODROID-XU의 사양은 Table 1과 같다.

Table 1. ODROID-XU Specification

Processor	Exynos 5210(1.6GHz Quad + 1.2GHz Quad)
RAM	2GB LPDDR3 800MHz
Storage	eMMC 4.5 64GB
OS	Linux Kernel 3.4.5
Platform	Android 4.2.2

SFQ는 elevator_init_fn에서 입출력 요청 큐를 관리하기 위한 알고리즘과 기아상태 확인자의 알고리즘을 구현하였고, elevator_add_req_fn에서 요청 큐 관리 기법을 구현하였다. 그리고 elevator_dispatch_fn에서는 입출력 요청 디스패치 기법과 고아상태 확인자의 알고리즘을 구현하였다.

5. 성능 평가

본 논문에서는 임의 접근 워크로드를 생성할 수 있도록 수정된 ioping 1.8.1을 벤치마크 툴로 사용하였다. 그리고 저장장치의 1GB를 제외한 나머지 공간을 임의의 파일로 채우고, 벤치마크 툴을 병렬로 수행함으로서 최악의 조건으로 실험 환경을 구성하였다.

Fig. 3은 8개의 읽기 프로세스와 8개의 쓰기 프로세스를 동시에 수행했을 경우의 읽기 응답시간 및 쓰기 응답 시간을 측정한 실험 결과의 정규 분포를 나타낸 그래프이다.

실험의 결과로 SFQ는 CFQ보다 평균 읽기 응답시간은 약 1% 감소하였고, 최대 읽기 응답시간은 약 78% 감소하여 기존의 스케줄링 기법보다 읽기 응답시간의 성능이 더욱 향상되었음을 알 수 있다. 그리고 평균 쓰기 응답시간은 약 3% 감소하였고, 최대 쓰기 응답시간은 약 32% 감소하여 읽기 위주의 선택 기법임에도 불구하고 쓰기 응답시간의 성능이 비슷하거나 약간 향상되었음을 알 수 있다.

6. 결 론

본 논문에서는 낸드 플래시 메모리 기반의 저장장치가 탑재된 모바일 시스템에서 사용자 반응성을 고려한 단순·공평 큐잉 입출력 스케줄러에 대하여 제안하였다.

제안하는 스케줄링 기법은 모바일 시스템의 최악의 환경에서도 기존의 스케줄러보다 최대 읽기 응답시간과 평균 읽기 응답시간을 감소시켜 사용자 반응성을 향상시킬 수 있음을 확인할 수 있었다.

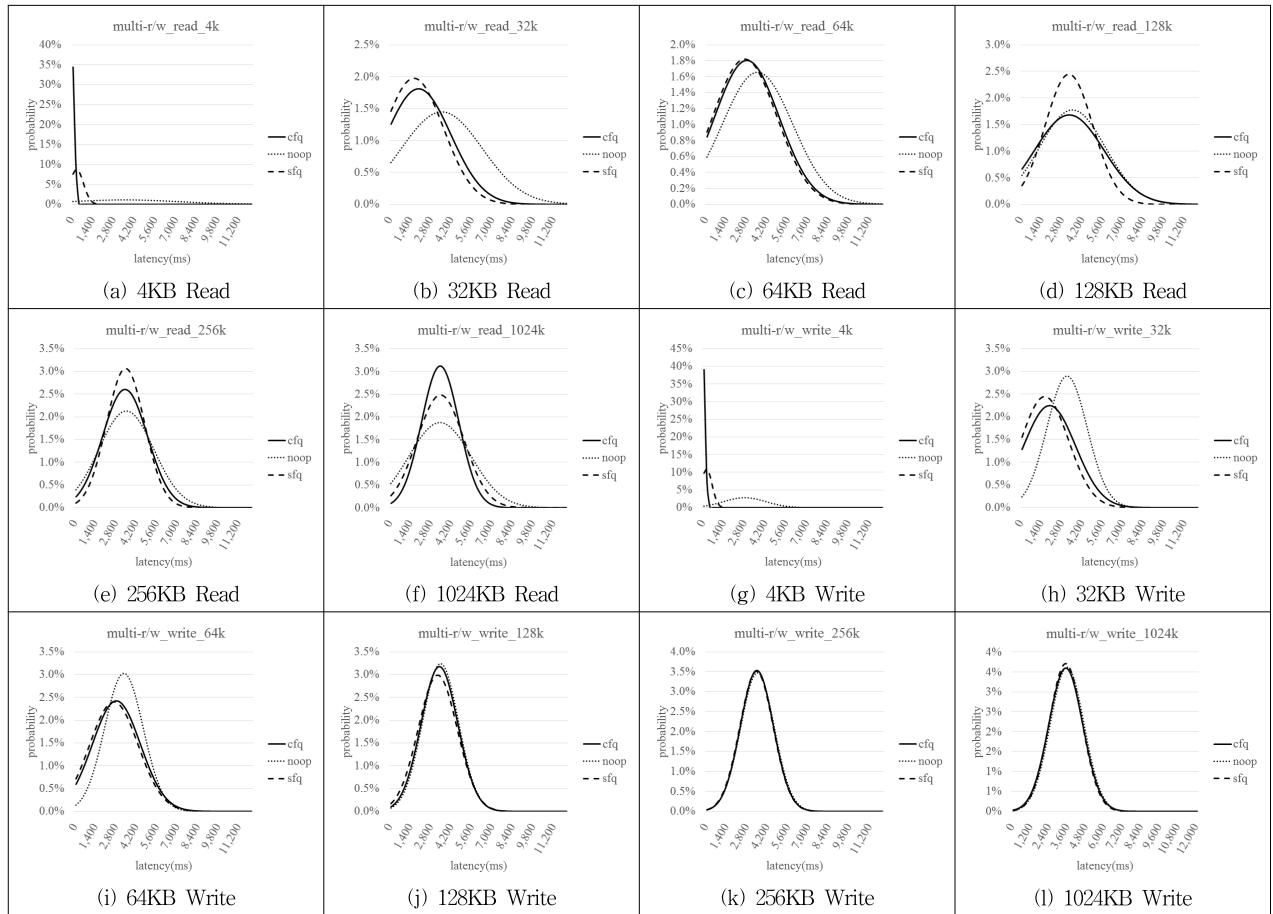


Fig. 3. Normal Distribution Graph of Latency in Multi-composite Request

References

- [1] Samsung Electronics, 64M × 8bit, 32M × 16bit NAND Flash Memory, datasheet, <http://arm9download.cncn.com/datasheet/K9F1208.pdf>.
- [2] Youngjoo Kim and Taeseok Kim, "Implementation of Linux I/O Scheduler Exploiting the Characteristics of SSDs," *Journal of KIISE : Computer Systems and Theory*, Vol.38, No.5, pp.223–232, 2011.
- [3] Sunwook Bae, Junghan Kim, and Young Ik Eom, "Enhancing Interactivity in Mobile Operating Systems," *Journal of KIISE : Computing Practices and Letters*, Vol.18, No.7, pp.533–537, 2012.
- [4] Myoungsoo Jung, Wonil Choi, Shekhar Srikantaiah, Joonhyuk Yoo, and Mahmut T. Kandemir, "HIOS: A Host Interface I/O Scheduler for Solid State Disks," in *Proceedings of the 41st Annual International Symposium on Computer Architecture*, 2014.
- [5] D. Jeong, Y. Lee, and J. Kim, "Boosting Quasi-Asynchronous I/O for Better Responsiveness in Mobile Devices," in *Proceedings of the 13th USENIX Conference on File and Storage Technologies(FAST '15)*, pp.191–202, 2015.

- [6] Qualcomm, RoW I/O Scheduler Specification [Internet], <https://gitmirror.org/shr/linux/source/0aa32b02356e765e2f04c15c64139becb2b89b2b:Documentation/block/row-iosched.txt>.
- [7] Hardkernel, Welcome to the Odroid Support Page: odroid xu [Internet], <http://odroid.com/dokuwiki/doku.php?id=en:odroid-xu>.
- [8] The Linux Kernel Archives, CFQ(Complete Fairness Queueing) [Internet], <https://www.kernel.org/doc/Documentation/block/cfq-iosched.txt>.
- [9] Noop scheduler [Internet], http://en.wikipedia.org/wiki/Noop_scheduler.



박종우

e-mail : jwpark0921@ee.knu.ac.kr
 2011년 대구대학교 컴퓨터·IT공학부(학사)
 2014년 경북대학교 전자공학부(공학석사)
 2014년~현 경북대학교 전자공학부
 박사과정
 관심분야: 플래시 스토리지, 운영체제,
 임베디드 소프트웨어, 모바일
 컴퓨팅 등



윤 준 영

e-mail : yoonpro7@gmail.com
2013년 대구대학교 컴퓨터·IT공학부(학사)
2015년 경북대학교 전자공학부(공학석사)
2015년~현 재 경북대학교 임베디드
소프트웨어연구센터 연구원
관심분야: 임베디드 시스템, 운영체제,
모바일 컴퓨팅 등



서 대 화

e-mail : dwseo@ee.knu.ac.kr
1981년 경북대학교 전자공학과(학사)
1983년 한국과학기술원 전산학과(공학석사)
1993년 한국과학기술원 전산학과(공학박사)
1983년~1995년 한국전자통신연구원
컴퓨터 S/W 연구실
2004년~현 재 경북대학교 임베디드소프트웨어연구센터장
1998년~현 재 경북대학교 전자공학부 교수
관심분야: 임베디드 소프트웨어, 병렬처리, 분산 운영체제 등