공학석사 학위논문

모바일 시스템에서의 저장 장치 성능 변화에 따른 응용의 성능 변화 탐색 및 활용

2013년 2월

서울대학교 대학원

전기컴퓨터공학부

임 제 헌

모바일 시스템에서의 저장 장치 성능 변화에 따른 응용의 성능 변화 탐색 및 활용

지도교수 김 지 홍

이 논문을 공학석사 학위논문으로 제출함
2013년 2월

서울대학교 대학원
전기컴퓨터공학부
임 제 헌

임제헌의 석사 학위논문을 인준함
2013년 2월

위 원 장 _____ 하 순 회 (인)

부위원장 _____ 김 지 홍 (인)

위 원 _____ Mckay (인)

초 록

스마트폰 또는 태블릿 PC 등의 스마트 장치에 탑재되는 낸드 플래시 메모리 기반의 저장 장치는 임의의 읽기/쓰기에 대한 성능이 우수하고 전력 소모량이 낮다는 장점이 있지만, 블록 당 삭제 횟수에 의해 수명이 결정된다는 문제점이 있다. 따라서 낸드 플래시 메모리 기반의 저장 장치에서는 수명의 최적화를 주제로 많은 연구들이 이루어지고 있다.

낸드 플래시 메모리 기반의 저장 장치는 쓰기 요청의 처리 속도와 수명 사이에 트레이드오프 관계가 존재한다. 즉 쓰기 성능의 희생으로 수명을 얻을 수 있게 되는 것이다. 또한 스마트 장치에 탑재되는 저장 장치는 여러 응용 프로그램에서 요구되는 최대의 쓰기 성능을 지원하도록 설계되었다. 이러한 저장 장치의 특성을 활용한다면 저장 장치의 쓰기 속도를 낮추더라도 응용 프로그램의 성능에 변화가 없는 응용 프로그램에 대해서는 저장 장치의 쓰기 성능을 낮게 조절함으로써 수명의 이득을 얻을 수 있게 된다.

위와 같은 특성을 갖는 응용 프로그램이 존재하는 지의 실험을 진행하기 위해서 저장 장치의 성능 에뮬레이션 환경을 개발하였다. 이를 활용하여 20개의 응용 프로그램에 대하여 저장 장치의 성능 변화에 따른 응용 프로그램의의 성능 변화를 탐색하는 실험을 진행한 결과, 실제 성능이 저장 장치의 성능에 영향을 받지 않는 응용 프로그램들이 존재하는 것을 확인하였다. 또한 응용 프로그램 별 성능의 저하를 발생시키지 않기 위해 요구되는 저장 장치의 성능 탐색 실험을 수행하였다. 실험을 통해 찾은 응용 프로그램별 최적의 저장 장치 성능 정보를 응용 프로그램 개발자가 프로그램 개발 시 저장 장치로 전달할 수 있도록 API를 개발하여 응용 프로그램

의 특성에 맞게 저장 장치의 성능이 조절되고, 나아가 수명의 이득을 얻을 수 있도록 하였다.

주요어 : 저장 장치의 수명, 응용 프로그램의 저장 장치 사용 특성, 저장 장치 에뮬레이션, 저장 장치의 성능과 신뢰성 관계
학 번 : 2011-20920

목 차

그 림 목 차

표 목 차

제 1 장 서 론

제 1 절 연구 배경

스마트폰과 태블릿 PC 등의 스마트 장치에는 낸드 플래시 메모리 기반의 저장 장치가 탑재된다. 낸드 플래시 메모리 기반의 저장 장치는 임의의 읽기/쓰기 작업에 대한 빠른 성능과 낮은 전력 소모량의 장점을 가지지만, 블록 당 삭제 횟수가 제한되어 있고 이 횟수가 저장 장치의 수명을 결정한다는 문제점을 가지고 있다. 현재 스마트 장치에는 2비트의 정보를 하나의 메모리 셀에 저장하는 Multi-Level Cell (MLC) 타입의 낸드 플래시 메모리가 사용되고 있는데, 가격 경쟁력을 올리기 위해서 3비트의 정보를 하나의 메모리 셀에 저장하는 Triple-Level Cell (TLC) 타입의 낸드 플래시 메모리가 점차적으로 사용될 것으로 예상되고 있다. TLC 기술은 셀 당 비트 수를 증가시켜 비트 당 가격을 하락시키지만, MLC 기술에 비해 플래시 메모리의 수명이 수배까지 짧아진다는 단점이 있다. 이러한 특성을 갖는 TLC 기반의 플래시 메모리가 스마트 장치에 탑재될 경우 저장 장치의 수명을 최적화하는 연구가 중요한 문제로 작용하게 된다.

플래시 메모리의 수명을 최적화하기 위한 여러 방법 중 하나로 칩의 쓰기 속도와 수명 간의 관계를 활용할 수 있다. 칩의 쓰기 속도와 수명 간에는 트레이드오프 관계가 존재하는데, 저장 장치의 쓰기 성능을 희생함으로써 수명을 연장할 수 있게 되는 것이다. 이러한 관계를 활용하여 저장 장치의 수명을 연장하는 것이 가능하게 하기 위해서는, 먼저 스마트 장치에서 응용의 실행 시 저장 장치의 쓰기 속도를 낮추더라도 응용의 성능에 유의미한 영향을 끼치지 않는 경우가 있다는 사실과 이를 증명하는 것이 필요하다.

스마트 장치에 탑재되는 저장 장치가 갖는 다른 하나의 특성으로는 저장 장치의 성능이 여러 작업에서 요구되는 최대 성능에 맞추어져 설계된다는 점이 있다. 저장 장치는 응용 프로그램(이하 “응용”)에서 사진을 저장하거나 동영상 등의 자료를 다운로드 받을 경우에, 또한 앞으로 사용될 자료의 캐싱이 필요한 경우 등 여러 경우에 사용된다. 이 때 저장 장치에서의 연산 처리로 인해 발생하는 지연을 최대한 줄이기 위해서 저장 장치는 많은 I/O 연산을 발생시키는 작업에서 요구되는 최대 성능을 가지도록 설계된다. 이러한 사실은 응용의 실행 시 저장 장치의 쓰기 속도를 낮추더라도 응용의 성능이 크게 변화하지 않을 수 있다는 가능성을 제공한다.

본 논문에서는 저장 장치의 쓰기 속도 변화에 따른 응용의 성능 변화 정도 실험에 대한 결과를 소개하고, 저장 장치의 쓰기 속도를 에뮬레이션 할 수 있도록 개발한 환경을 소개한다. 나아가 응용의 성능 변화 정도를 활용하여 저장 장치의 수명을 최적화할 수 있는 방법을 제안한다.

제 2 절 연구의 동기와 기여

플래시 메모리 기반의 저장 장치는 모바일 장치를 구성하는 여러 핵심적인 구성 요소 중 하나라고 할 수 있다. 플래시 메모리에서 정보는 셀의 임계전압에 의해 결정되는데, 임계전압 사이의 에러 마진에 따라 플래시 메모리의 P/E 사이클이 영향을 받게 된다.

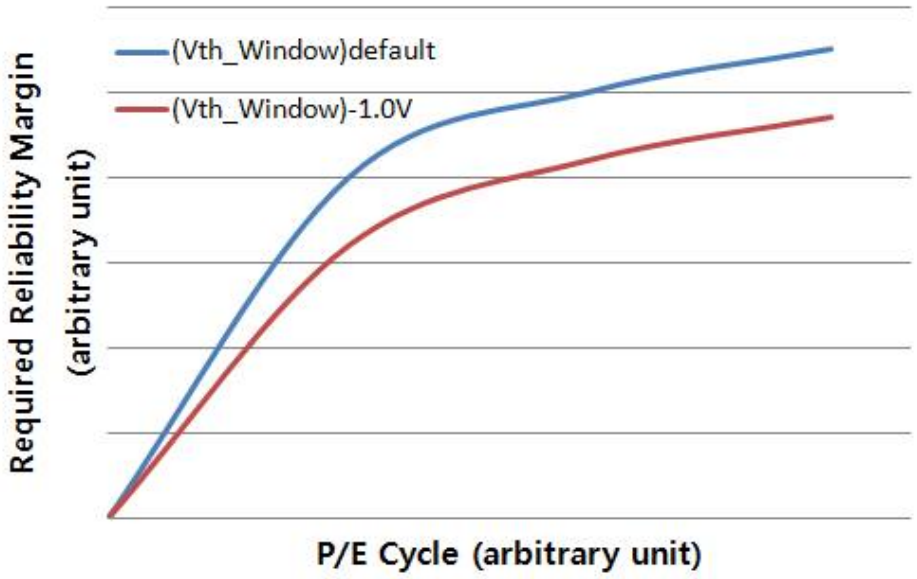

그림 1 P/E 사이클과 요구 신뢰성 마진과의 관계

그림 1의 그래프는 플래시 메모리의 P/E 사이클과 메모리에 저장한 정보가 에러로 인해 실제 정보와 다르게 읽히는 상태가 발생하지 않기 위해 요구되는 마진과의 관계를 나타낸다. 그래프를 통해서 같은 신뢰성 마진을 갖는 경우에는 (Vth, window)가 작은 경우에 더 많은 P/E 사이클을 얻을 수 있음을 확인할 수 있다. 따라서 P/E 사이클의 증가를 통해 플래시 메모리의 수명을 연장하기 위해서는 (Vth, window)의 값을 줄이는 것이 중요하다.

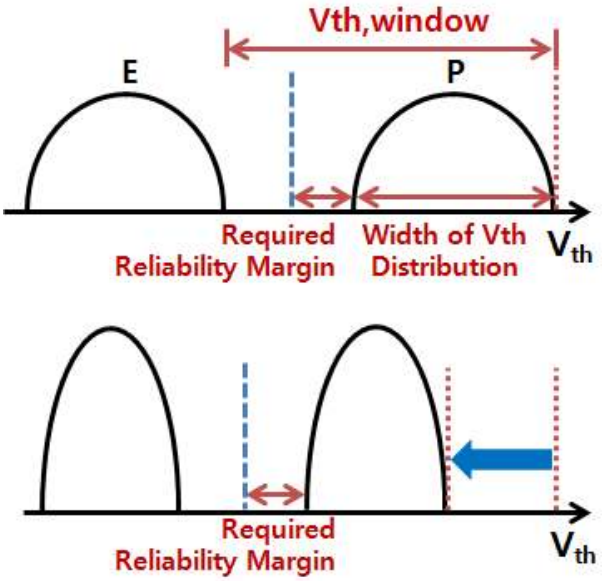

그림 2 (Vth, window)와 산포폭의 관계

그림 2는 (Vth, window)와 산포폭 사이의 관계를 나타낸다. P/E 사이클의 증가를 통한 수명 연장을 위해서는 (Vth, window)의 값을 줄여야 하는데, 이를 위해서는 산포폭 역시 감소해야 한다.

이러한 산포폭을 줄이기 위한 방법의 일환으로 프로그래밍에서의 파라미터를 조정하는 방법이 있다. 플래시 메모리의 프로그램 매커니즘 중 ISPP(Incremental Step Pulse Programming) 방식은 필수적인 기법 중의 하나로, 이는 임계전압의 분포를 일정 수준 이상으로 좁게 만들기 위하여 전압의 인가와 확인을 반복하는 기법이다. 플래시 메모리 셀 내의 플로팅 게이트들은 셀마다 그 물리적 특성이 조금씩 다르므로 임계전압을 한 번에 큰 폭으로 옮긴다면 그 분포가 에러 마진 이상으로 넓어지기 때문에 전압의 인가와 확인의 반복은 필수적이다. 이 때 단계마다 인가되는 전압을 작게 하고 그 만큼 확인하는 작업을 자주 반복한다면 프로그래밍에 걸리는 시간은 더 많이 소요될지라도 임계전압의 분포를 매우 세밀하게 만들 수 있고, 이는 산포폭을 줄이게 되어 (Vth, window)의 값이 작아지는 결과를 가져온다. 그림 3에서 이와 같은 관계를 확인할 수 있다.

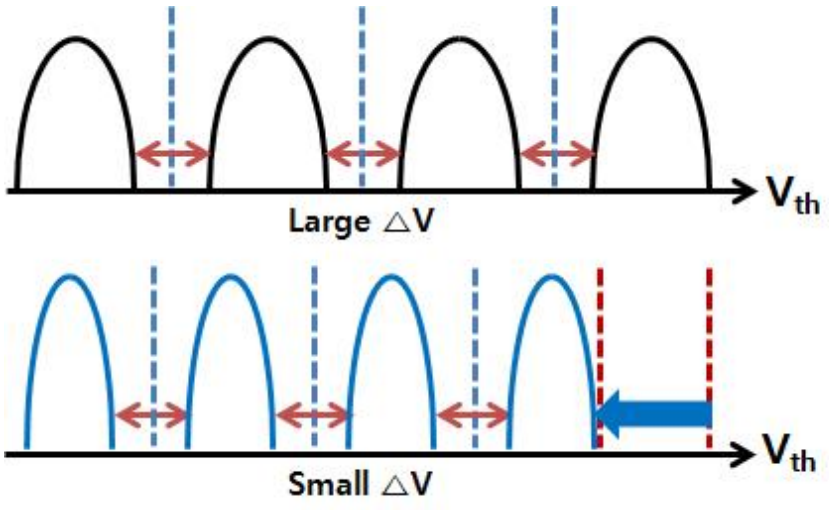

그림 3 프로그램 시 인가되는 전압과 (Vth, window)의 관계

이는 플래시 메모리의 성능과 내구성 사이에 트레이드오프 관계가 존재함을 의미하고, 따라서 충분한 시간이 주어졌을 경우 내구성을 높이는 방향으로 프로그래밍에서의 파라미터를 조정하여 플래시의 수명을 연장할 수 있음을 알 수 있다. 이와 같은 관계를 정리하면 그림 4와 같다.

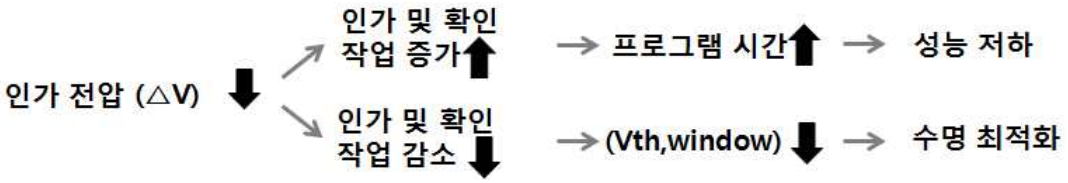

그림 4 플래시 메모리의 성능과 수명 사이의 관계

지금까지의 기술 개발은 플래시 메모리가 사용되는 시스템에서 필요로 하는 최대의 성능을 보장하도록 이루어져 왔고, 이에 따라 실제 시스템이나 응용 프로그램이 요구하는 일반적인 수준보다 필요 이상의 성능을 내기 위해 수명을 희생하도록 설계된 측면이 있다. 모바일 장치의 저장 장치가 요구하는 플래시 메모리의 집적도가 높아짐에 따라 수명은 기하학적으로 악화될 것이기 때문에 성능과 수명 사이의 트레이드오프 관계를 활용하여 시스템과 응용 프로그램이 요구하는 최적의 성능을 제공하면서 수

명의 이득을 얻을 수 있도록 하는 최적화가 필요하다.

모바일 장치에서 저장 장치의 성능와 수명 사이의 트레이드오프 관계를 활용하여 수명을 최적화하기 위해서는 저장 장치의 쓰기 속도를 늦추더라도 응용의 성능 저하가 사용자가 용인할 수 있는 수준으로 발생한다는 전제가 필요하다. 이 때 응용의 성능이란 반응 시간을 의미하는 것으로, 사용자의 입력으로 인해 발생한 응용 내부의 하나의 작업 처리 결과가 화면에 모두 그려지는 시점을 가리킨다. 반응 시간은 여러 기준을 통해 정의될 수 있는데, 본 논문에서는 사용자 경험을 해치지 않는 수준에서 저장 장치의 수명을 연장시키는 방법을 연구하는 것을 목표로 하기 때문에 사용자 관점에서의 반응 시간이라 할 수 있는 작업의 결과 화면이 모두 그려지는 시점을 선택하여 사용한다.[1]

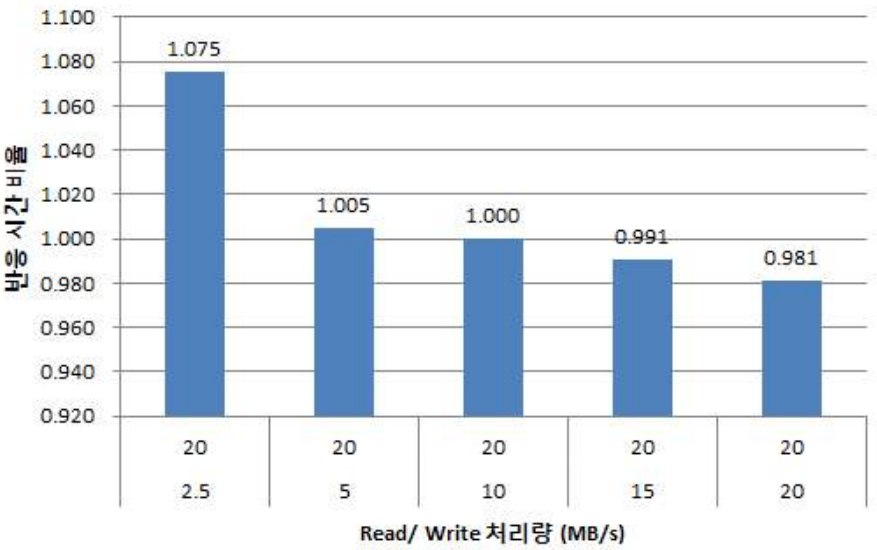

그림 5 Write 처리량에 따른 응용의 반응 시간 변화 비율

그림 5는 저장 장치의 쓰기 속도를 조절하면서 측정한 포털 사이트 응용의 성능 변화를 나타낸다. 이 때의 성능이란 반응 시간의 일종인 응용의 론칭 시간으로, 응용의 시작으로부터 첫 웹 페이지가 화면에 모두 그

려지는 시점까지의 시간을 나타낸다. 그래프를 통해 저장 장치의 쓰기 속도가 10MB/s에서 5MB/s로 떨어질 때 응용의 성능은 0.6%가 감소하였고, 1MB/s일 경우에는 7.5%가 감소하였다는 것을 확인할 수 있다.

Read / Write 처리량 (MB/s)	20 / 2.5	20 / 5	20 / 10	20 / 15	20 / 20
반응 시간	5.32367	4.992084	4.968229	4.921073	4.872583

표 1 Write 처리량에 따른 응용의 반응 시간

표 1은 응용의 론칭 시간의 절대값을 나타낸다. 저장 장치의 쓰기 속도가 10MB/s에서 5MB/s와 1MB/s로 변화할 경우에 각각 약 0.03초와 0.35초의 시간이 증가한 것을 확인할 수 있다. 이러한 론칭 시간 증가가 사용자 경험에 미치는 영향을 정량적으로 평가하기는 어렵지만, 반응 시간과 관련된 연구에서 제시한 기준을 참고하면 0.03초와 0.35초의 론칭 시간 증가는 사용자 경험에 영향을 주지 않는 무시할 수 있는 수준이라고 판단할 수 있다.[2]

위와 같이 저장 장치의 쓰기 속도가 느려지는 방향으로 변화한다고 하더라도 응용의 성능에는 사용자가 용인할 수 있는 수준의 영향을 끼치는 응용의 작업 같은 경우에는 저장 장치의 쓰기 성능과 수명 사이의 트레이드오프 관계를 활용할 수 있게 된다. 저장 장치의 쓰기 속도를 조금 느리게 함으로써 수명 상의 이득을 얻어 저장 장치를 더 오래 쓰는 것이 가능해지는 것이다.

본 논문의 기여 중 첫 번째로, 저장 장치의 쓰기 성능에 따른 응용의 성능 변화를 탐색하는 실험을 진행하였다. 응용의 론칭 또는 론칭 이외의 작업 수행 시 저장 장치로의 쓰기 작업을 발생시키는 20개의 응용에 대하

여 저장 장치의 쓰기 속도를 변화시키면서 응용의 반응 시간이 어떻게 변화하는지를 측정하였고, 실험 결과를 토대로 각 응용이 요구하는 저장 장치의 최적의 쓰기 성능을 탐색하였다.

두 번째로, 실제 스마트폰에서 동작하는 저장 장치의 성능 에뮬레이션 환경을 개발하였다. 저장 장치의 쓰기 속도 변화에 따른 응용의 성능 변화 실험을 진행하기 위해서는 저장 장치의 속도를 변화시킬 수 있는 환경이 필요하다. 현재 스마트 장치에 탑재되어 있는 저장 장치의 쓰기 속도를 임의로 조절하는 것은 불가능하기 때문에 메모리에 저장 장치를 에뮬레이션 할 수 있는 램디스크를 설치하고, 응용의 설치 및 쓰기 작업이 해당 램디스크로 발생하도록 수정하였다. 또한 파일시스템으로부터 I/O 요청을 받아 디바이스로 전달하는 플래시 변환 계층에 저장 장치의 속도 제어 모듈을 추가하여 쓰기 속도를 조절할 수 있는 환경을 구현하였다.

마지막으로 저장 장치의 성능 에뮬레이션 환경을 활용하여 응용이 요구하는 저장 장치의 최적의 쓰기 성능을 탐색한 결과를 응용 개발자가 응용의 개발 시 활용할 수 있는 방안을 제시하고, 실제 사용자가 느끼는 응용의 성능 저하 없이 저장 장치의 쓰기 속도를 효과적으로 조절할 수 있음을 확인하였다.

제 3 절 논문 구성

본 논문의 구성은 다음과 같다. 2장에서는 최근 연구된 플래시 메모리의 수명 연장을 위한 연구 중 본 논문에서 연구하려는 방향과 비슷한 성격을 갖는 기법을 소개한다. 또한 응용의 성능의 한 종류인 반응 시간의 정의 및 측정과 관련된 내용에 대하여 자세히 설명하도록 한다. 3장에서는 저장 장치의 성능 변화에 따른 응용의 성능 변화를 측정하기 위해 필요한 저장 장치의 성능 에뮬레이션 환경 개발 내용을 설명한다. 4장에서는 위의 실험을 수행하기 위해 추가로 필요한 반응 시간 측정 도구와 네트워크 성능 에뮬레이션 환경에 대해 언급한다. 5장에서는 개발한 환경을 활용하여 탐색한 응용별 최적의 저장 장치 성능을 활용하는 방안에 대해 제시한다. 6장에서는 실험 환경과 저장 장치의 성능 변화에 따른 응용의 성능 변화 분석 실험 결과 및 분석 내용을 설명한다. 마지막으로 7장에서는 결론과 함께 향후 연구 내용에 대해 소개한다.

제 2 장 관련 연구

제 1 절 플래시 메모리의 수명 연장 연구

낸드 플래시 메모리 기반의 저장 장치는 하드 디스크에 비해 빠른 임의의 읽기/쓰기에 대한 성능, 내구성, 적은 전력 소모량 등으로 인해 다양한 시스템의 보조기억장치로 사용되고 있다. 하지만 블록 당 삭제 횟수에 의해 저장 장치의 수명이 결정된다는 단점이 있기 때문에 저장 장치의 수명을 최적화하기 위한 여러 연구가 진행되고 있다.

저장 장치의 수명과 관련된 여러 연구 중 본 논문에서 수행하려는 연구와 비슷한 성격을 지니는 READY[3]는 쓰기 성능을 throttling 하여 엔터프라이즈 시장에서의 SSD가 필요로 하는 수명을 만족시킬 수 있도록 하였다. 이 때 적은 throttling 오버헤드로 SSD의 수명을 보장하기 위해서 플로팅 게이트 트랜지스터의 self-recovery 효과를 활용하였다. READY 기법의 실험 결과는 이 기법이 쓰기 응답 시간에 있어서 적은 수준의 감소만을 통해서 요구되는 수명을 보장한다고 제시하였다.

제 2 절 반응 시간 연구

HCI(Human Computer Interaction) 연구 그룹들에 의해 컴퓨터의 반응 시간이 사용자의 만족감, 스트레스, 생산성 등에 영향을 미친다는 사실이 밝혀졌다. 이와 더불어 사용자가 용인할 수 있는 스마트 장치의 반응 시간의 기준을 찾는 연구도 이루어졌는데, 그 기준은 다음과 같다. 사용자 입력에 대한 반응이 입력으로부터 150ms 이내에 완료된다면 사용자는 이러한 지연을 인지하지 못한다. 반응 완료가 150ms와 1000ms 사이에 이루어진다면 사용자는 이 시간을 인지할 수 있지만 작은 시간이기 때문에 무시할 수 있다. 반응 완료가 1000ms 시간 이후에 일어나게 된다면 사용자는 이 시간을 인지할 수 있을 뿐만 아니라 지연으로 생각하기 때문에 사용자 경험에 부정적일 영향을 미치게 된다.[2]

또한 사용자 입력에 의해 발생한 작업의 반응 완료 시점을 어느 순간으로 볼 것인지에 대해서도 연구가 진행되고 있다.[3] 반응 시간은 크게 세 가지로 나누어 생각해 볼 수 있다. 먼저 작업을 발생시킨 사용자의 입력으로부터 사용자의 다음 입력에 대해 응용이 반응을 시작하는 순간까지의 시간이 있다. 사용자가 응용과 상호작용을 시작하는 것이 가능해지는 순간이라는 점에 의의가 있지만, 화면에 유의미한 유저 인터페이스가 그려지지 않더라도 사용자의 입력에 반응하는 경우도 있기 때문에 실제 사용자가 인지하는 반응 시간보다 작은 시간을 나타낼 가능성이 있다. 두 번째로 유저 인터페이스가 화면에 모두 그려지는 시점까지의 시간이 있다. 사용자는 화면의 인터페이스를 확인하고 다음 작업을 진행하기 때문에 사용자 관점에서의 반응 시간을 가장 잘 나타낸다고 할 수 있지만, 모든 인터페이스가 화면에 나타나기 전에도 다음 작업을 입력할 수 있기 때문에 실제 사용자가 인지하는 반응 시간보다 큰 시간을 가리키는 경우가 존재한다. 마지막으로 사용자가 입력한 작업에 대해 시스템 내부적으로 처리가 완료되는 시점까지의 시간이

있다. 사용자 입력에 대한 결과가 화면에는 모두 그려졌더라도 내부적으로는 데이터 캐싱 등의 인터페이스를 그리는 것과 무관한 작업들이 수행되게 된다. 이러한 반응 시간은 온전히 시스템 관점에서의 반응 시간으로 사용자 경험의 최적화를 위해 사용되는 반응 시간으로는 부적절한 성격을 지니고 있다.

제 3 장 저장 장치의 성능 에뮬레이션 환경

제 1 절 저장 장치의 성능 에뮬레이션 환경 전체 구조

저장 장치의 성능 변화가 응용의 성능에 미치는 영향을 탐색하기 위해서는 스마트 장치에 탑재된 플래시 메모리 기반의 저장 장치의 속도를 조절하는 것이 필요하다. 하지만 실제 저장 장치의 속도를 조절하는 것은 불가능하기 때문에 저장 장치의 성능을 에뮬레이션 할 수 있는 환경을 개발하였다. 그림 6은 스마트 장치에 탑재된 저장 장치의 성능 에뮬레이션 환경의 전체 구조를 나타낸다.

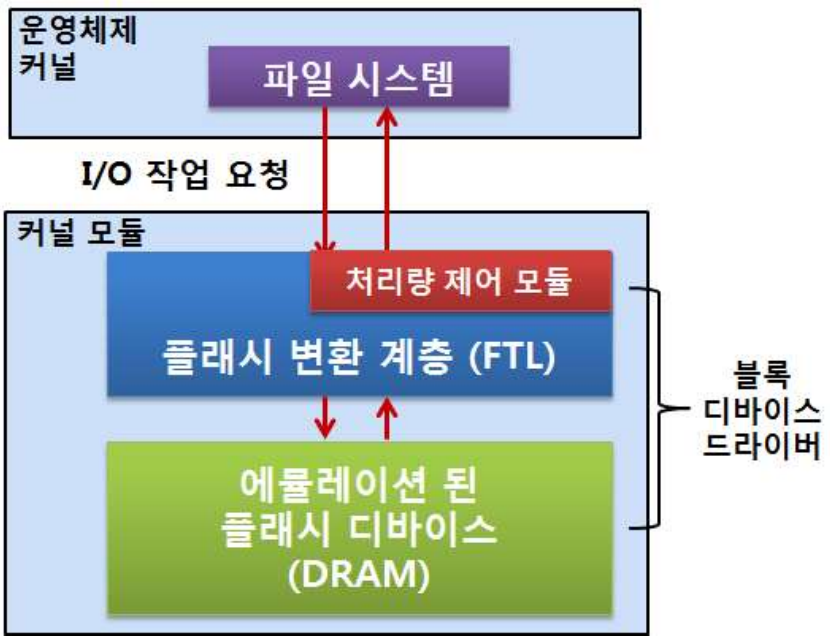

그림 6 저장 장치의 성능 에뮬레이션 환경 구조

저장 장치의 성능 에뮬레이션 환경은 블록 디바이스 드라이버로 구성하여 커널 모듈로써 스마트 장치에 추가하여 사용할 수 있도록 설계 및 구현되었다. 저장 장치의 성능 에뮬레이션 환경은 크게 세 가지로 나눌 수 있는데, 실제 저장 장치를 에뮬레이션 하기 위한 플래시 변환 계층과 플래시 디바이스, 저장 장치의 성능 조절을 위한 플래시 변환 계층 내의 처리량 제어 모듈이 그 것이다.

실제 플래시 메모리 기반의 저장 장치에는 플래시 변환 계층(Flash Translation Layer, FTL)이라는 시스템 소프트웨어가 존재하는데, 파일 시스템으로부터 내려오는 I/O 요청의 논리 주소를 플래시 메모리의 물리 주소로 변환하는 작업을 수행한다. 플래시 메모리는 데이터 쓰기 연산을 수행하기 위해서는 연산을 수행하려고 하는 해당 영역이 지워져 있어야 한다는 특성을 지니는데, 이러한 하드웨어적인 한계점을 극복하기 위하여 쓰기 요청을 미리 지워져 있는 빈 공간에 쓰이게끔 주소를 변환하여 재지정하는 작업을 수행하는 플래시 변환 계층을 필요로 하게 되는 것이다. 또한 특성 페이지에 대한 쓰기/지우기 연산이 일정 횟수를 넘으면 그 페이지의 데이터가 손상될 수 있다는 문제점을 완화시키기 위하여 흩어져 있는 페이지들을 한 곳으로 복사하여 모아놓고 일괄 삭제시킴으로써 지우기 연산의 횟수를 감소시키는 가비지 컬렉션의 작업을 수행한다. 저장 장치의 성능 에뮬레이션 환경에서의 대상 저장 장치는 플래시 메모리 기반의 저장 장치이기 때문에 해당 저장 장치에서 필요로 하는 플래시 변환 계층이 모듈에 포함되어 있다. 해당 계층에서는 앞서 언급한 바와 같이 플래시 변환 계층의 주요 작업인 주소 변환 작업과 함께 가비지 컬렉션의 작업을 수행한다.

실제 저장 장치를 에뮬레이션 하여 성능을 조절하기 위해서 DRAM 메모리에 저장 장치의 동작을 에뮬레이션 하는 램디스크를 설치하는 작업을 수행하였다. 플래시 디바이스 모듈에서는 실제 플래시 메모리 기반의 저장 장치에서 발생하는 읽기/쓰기 연산을 DRAM 메모리로 수행하도록 하

여 저장 장치의 동작을 에뮬레이션 한다.

마지막으로 저장 장치의 성능을 조절 할 수 있도록 플래시 변환 계층 내에 처리량 제어 모듈을 개발하였다. 이에 대한 자세한 설명은 다음 절에서 계속 이어진다.

제 2 절 저장 장치의 처리량 제어 모듈

저장 장치의 성능을 조절하기 위해서 플래시 변환 계층 내에 저장 장치의 처리량 제어 모듈을 개발하였다. 저장 장치의 성능을 조절한다는 것은 단위 시간 내에 처리 가능한 요청의 개수, 즉 처리량을 조절한다는 것을 의미한다. 단위 시간 내에 처리되는 요청의 개수를 조절하기 위해서는 목표 처리량을 달성하기 위해서 하나의 요청이 처리되는데 걸려야 하는 시간을 계산한 후, 요청의 읽기/쓰기 작업 이후 해당 시간만큼의 지연을 주면서 그 시간 동안에는 다른 요청이 처리되지 못하도록 함으로써 처리량을 조절할 수 있다.

먼저 목표 처리량(throughput)이 주어졌을 경우 단위 시간(1초) 내에 처리 가능한 요청의 개수(N)는 그림 7과 같이 구할 수 있다. 이 때 목표 처리량은 MB/s 단위로 주어진다고 가정하였다.

$$N = (Throughput * 1024) / 2$$

그림 7 단위 시간 내에 처리 가능한 요청의 개수

현재 구현되어 있는 플래시 메모리 기반의 저장 장치 에뮬레이션 디바이스에서 하나의 페이지 크기를 2K로 설정하였기 때문에, 위의 식을 통해서 단위 시간 내에 처리 가능한 페이지의 개수를 구할 수 있다.

위에서 구한 단위 시간 내 처리 가능한 페이지의 개수를 단위 시간으로 나누어 주면, 목표 처리량을 달성하기 위해서 하나의 페이지를 처리하는데 걸려야 하는 시간을 계산할 수 있다. 본 논문에서는 이 시간을 MTR(Max Time for Request)이라 정의하도록 하겠다.

플래시 변환 계층에는 파일 시스템으로부터 블록 디바이스 드라이버로 I/O 요청이 전달되었을 때 수행되는 device_make_request() 함수가 존재

한다. 해당 함수 내에서는 파일 시스템으로부터 전달 받은 I/O 요청을 위해 주소 변환과 필요 시 가비지 컬렉션 작업을 수행하고, 읽기/쓰기의 요청한 작업을 처리한 후 작업이 완료되었음을 파일 시스템에 전달하게 된다. 저장 장치의 처리량 제어 모듈은 I/O 요청이 처리되었음을 파일 시스템에게 알려주기 이전에 MTR을 기반으로 계산한 지연을 주어 파일 시스템에서 다음 요청을 보내는 시간을 늦추게 하고, 이를 통해 처리량이 조절되도록 한다. 지금까지 설명한 I/O 요청의 처리 구조를 그림으로 나타내면 그림 8과 같다.

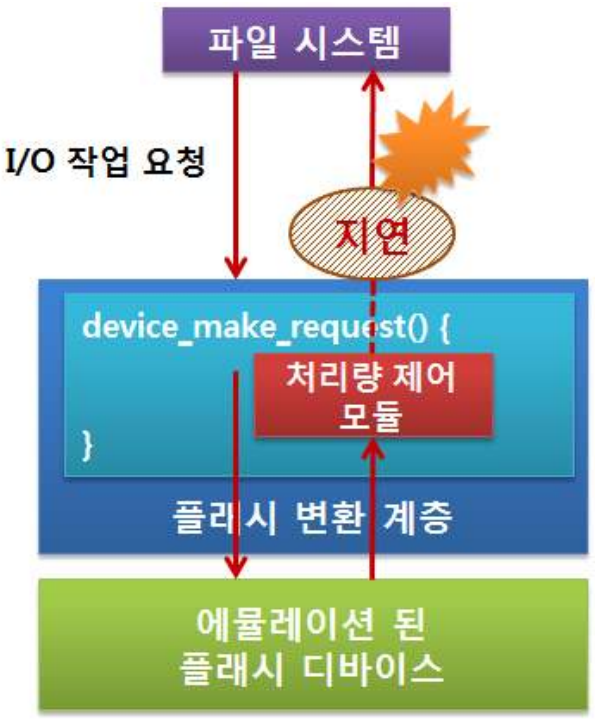

그림 8 저장 장치의 처리량 제어 모듈

처리량 제어 모듈에서는 I/O 요청 별 지연을 주기 이전에 목표 처리량과 단위 시간을 제어하고 실제 지연을 얼마나 줄 것인지에 대한 계산을 하는 등의 작업을 수행해야 한다. 먼저, 현재의 단위 시간 내에 처리된 요청의 양이 목표 처리량을 초과하였다면 단위 시간 내의 남은 시간은 지연

으로 처리해 더 이상의 요청이 처리되지 않도록 하여 단위 시간 내에 처리되는 양이 목표 처리량을 넘지 않도록 한다. 다음으로 이전에 처리한 요청이 들어온 시간과 현재 요청이 들어온 시간이 다른 단위 시간에 속해 있으면 단위 시간 내 처리된 요청의 양을 초기화하는 작업을 통해 새로운 단위 시간이 시작되었음을 설정해야 한다. 이후 현재의 요청의 크기를 보고 지연 시간을 계산한다. 파일 시스템에서의 요청의 기본 단위는 4K이지만 write buffer를 사용하는 일반적인 경우 여러 요청이 합쳐져 큰 단위로 들어올 수 있기 때문에 현재 요청에 몇 개의 페이지가 속해있는 지를 계산하고, 이 값에 앞에서 계산한 MTR을 곱하여 현재 요청에 주어져야 하는 지연 시간을 계산한다. 지연은 커널 타이머를 이용하여 주어지도록 구현하였다. device_make_request()에 요청이 들어오게 되면 해당 요청은 completion lock을 획득하여 다음 요청이 동시에 처리되는 것을 막는데, 커널 타이머를 이용하여 설정한 지연 시간이 모두 지난 다음에 요현재 요청이 가지고 있는 completion lock을 풀어주도록 함으로써 하나의 요청이 처리되는데 걸리는 시간을 조절하고, 나아가 저장 장치의 처리량을 조절할 수 있게 된다. 위와 같은 I/O 요청 별 지연을 주는 방법을 적용하였을 경우의 요청 처리 과정을 보면 그림 9와 같다.

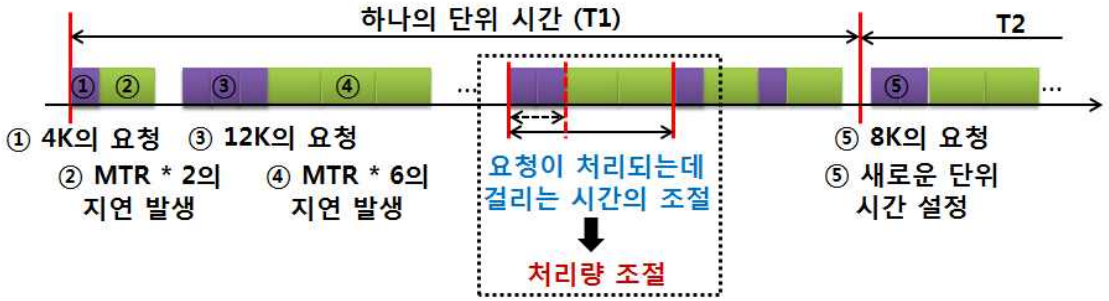

그림 9 I/O 요청 별 지연을 적용하였을 경우의 요청 처리 과정

개발한 저장 장치의 처리량 제어 모듈을 적용하였을 경우 실제 처리량

이 잘 조절되는 지를 검증하는 과정이 필요하다. 이를 위해서 일정한 크기의 파일에 쓰기 작업을 수행하는 간단한 프로그램을 작성한 후 각각에 대해 15번의 실험을 수행하여 구현한 처리량 제어 모듈을 검증하였다.

표 2는 쓰기 작업의 목표 처리량과 처리량 제어 모듈을 통해 조절된 저장 장치의 처리량 사이의 차이를 나타낸다.

목표 처리량 (KB/s)	2560	5120	10240	15360	20480	30720	40960
제어 처리량 (KB/s)	2026.5	5008.1	9810.0	14972.1	19872.9	29210.3	39329.6
오차율 (%)	1.05	2.19	4.18	2.52	2.96	4.91	3.98

표 2 처리량 제어 모듈을 통해 제어한 처리량의 오차

위 표를 토대로 결과를 분석해 보면, 목표 처리량과 처리량 제어 모듈을 통해 조절된 제어 처리량 사이에는 약 5% 미만의 오차가 존재하여 처리량이 잘 조절되고 있음을 확인할 수 있다.

제 3 절 응용의 설치 경로 제어 모듈

저장 장치의 쓰기 성능을 변화시키면서 응용의 성능 변화를 탐색하기 위해서는 실험을 원하는 응용을 성능 조절이 가능한 디바이스에 설치해야 한다. 응용이 론칭되거나 작업을 처리함에 있어서 관련된 파일들을 저장 장치로부터 읽어오는 작업을 수행하기 때문에 해당 파일들이 모두 성능 조절이 가능한 디바이스에 위치하고 있어야 저장 장치의 성능 변화에 따른 응용의 성능 변화를 정확하게 측정할 수 있다.

안드로이드 플랫폼에서 응용 프로그램은 APK(android package file)라는 패키지 형태로 존재한다. 응용 프로그램을 설치한다는 것은 이 APK 파일을 받아 압축을 풀어 필요한 파일들을 적절한 경로에 위치시키는 작업을 의미한다.[4] 응용의 설치 시 APK 파일로부터 응용을 실행하는데 필요한 데이터베이스와 라이브러리 등의 파일과 달빅 가상 머신에서 실행될 컴파일 된 클래스 파일을 추출하여 정해진 경로에 저장하게 된다. 안드로이드 플랫폼에서 시스템 응용이 아닌 일반적인 응용의 APK 파일은 /data/app/ 폴더에, 응용을 수행하는데 있어 필요한 데이터는 /data/data/ 폴더에 저장되며 달빅 가상 머신에서 수행되는 덱스(dex) 파일은 /data/dalvik-cache 폴더 아래에 저장된다.[5] 저장 장치의 성능 변화에 따른 응용의 성능 변화 실험을 수행하기 위해서는 응용의 실행과 관련된 파일의 설치 경로를 변경하는 작업이 필요하기 때문에 안드로이드 플랫폼 내부에 응용의 설치 경로 제어 모듈을 구현하였다. 해당 모듈의 구조는 그림 10과 같다.[6]

그림 10 응용의 설치 경로 제어 모듈 구조

안드로이드 프레임워크의 패키지 매니저 내에 존재하는 응용의 설치 경로 제어 모듈은 APK 파일과 응용을 수행하는데 있어서 필요한 데이터, 그리고 덱스 파일이 저장되는 경로를 사용자가 설정한 경로로 변경한다. 우리는 이 위치를 저장 장치의 동작 에뮬레이션이 가능하도록 설치한 램 디스크가 마운트 된 폴더의 경로로 지정하였다. 안드로이드 런타임의 달빅 가상 머신 내에 존재하는 응용의 설치 경로 제어 모듈에서는 응용의 실행을 위해 필요한 덱스 파일을 로딩하는 경로를 사용자가 설정한 경로로 변경한다. 해당 모듈을 통해서 사용자는 응용의 설치가 원하는 경로에 이루어지도록 제어할 수 있게 된다.

제 4 장 저장 장치의 성능 변화에 따른 응용의 성능 변화 탐색

제 1 절 응용의 성능 변화 탐색 실험을 위한 설정

저장 장치의 쓰기 성능 변화가 응용의 성능에 미치는 영향을 탐색하기 위해서는 응용의 설치 경로 제어 모듈이 탑재된 플랫폼이 올려져 있는 스마트 장치가 필요하다. 실험을 원하는 응용을 램디스크가 마운트 된 경로에 설치하여야 저장 장치의 성능 변화에 따른 응용의 성능 변화를 측정할 수 있다.

또한 응용의 성능 변화 탐색 실험을 수행하기 위해서는 저장 장치의 성능 에뮬레이션 환경과 더불어 응용의 성능을 측정하기 위한 사용자 관점에서의 반응 시간을 측정하는 도구와 네트워크 속도를 에뮬레이션 할 수 있는 환경이 필요하다. 스마트 장치의 많은 응용은 저장 장치와 더불어 네트워크 모듈을 사용한다. 마켓 응용에서 새로운 응용을 다운로드 받아 설치하는 등의 작업은 저장 장치의 성능뿐만 아니라 네트워크 모듈의 성능에도 영향을 받을 가능성이 크다. 따라서 저장 장치의 성능이 응용의 성능에 미치는 영향을 정확하게 탐색하기 위해서는 저장 장치와 더불어 네트워크 모듈의 속도도 변화시켜 가면서 응용의 성능 변화를 측정할 필요성이 있다. 네트워크로부터 데이터를 받아오고, 이 데이터를 저장 장치에 쓰는 작업을 수행할 경우 저장 장치에서 요구되는 성능은 네트워크 모듈의 성능에 종속적이기 때문이다.

저장 장치의 성능 변화에 따른 응용의 성능 변화 탐색 실험을 수행하기 위한 환경은 그림 11과 같이 나타낼 수 있다.

그림 11 저장 장치의 성능에 따른 응용의 성능 변화 실험 환경

사용자 관점에서의 반응 시간 측정 도구는 연구실에서 기 개발한 도구를 활용하였다. 네트워크의 성능 에뮬레이션 환경은 기존의 도구와 환경을 활용하여 개발하였는데, 각각에 대해 아래에서 좀 더 자세하게 설명하도록 하겠다.

제 2 절 사용자 관점에서의 반응 시간 측정

사용자 입력에 대한 반응 완료 시점의 정의는 여러 가지가 가능하다. 사용자 입력에 대해 응용이 처음으로 반응하는 시점과 모든 유저 인터페이스가 화면에 그려지는 시점, 시스템 내부적으로 사용자가 요청한 작업에 대한 처리가 완료되는 시점 등이 존재한다. 이를 그림으로 나타내면 그림 12와 같다.

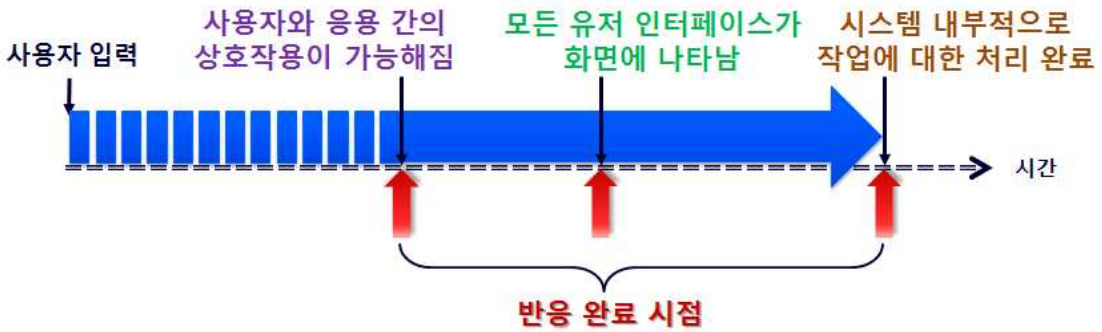

그림 12 사용자 입력에 대한 반응 완료 시점

스마트 장치에서는 사용자 경험이 최적화의 대상으로 매우 중요하다. 여러 가능한 반응 시간의 정의 중 사용자 경험과 밀접한 관련이 있는 반응 시간은 모든 유저 인터페이스가 화면에 나타나 사용자가 다음 작업을 요청하기 위한 판단을 내릴 수 있게 되는 시점일 것이다. 따라서 본 논문에서 수행된 모든 실험에서 응용의 성능은 사용자 입력으로부터 이에 의해 발생한 요청에 대한 결과가 화면에 모두 그려지는 시점까지의 시간을 의미한다. 사용자 관점에서의 반응 시간을 측정하기 위해서 연구실에서 이미 개발한 RTmeter라는 도구를 활용하였다.

제 3 절 네트워크 성능 에뮬레이션 환경

응용별 특성에 적합한 저장 장치의 성능을 탐색하기 위해서는 다른 컴포넌트의 설정과의 연관 관계를 밝혀내고 이를 활용하는 것이 필요하다. 본 논문에서는 저장 장치 이외의 컴포넌트를 네트워크로 제한하였고, 네트워크 성능 에뮬레이션 환경을 개발하였다.

스마트 장치에서 이용 가능한 네트워크 모듈은 3G와 WiFi, LTE로 한정되어 있기 때문에 다양한 네트워크 속도에 대한 응용의 성능 변화를 탐색하기 위해 리버스 테더링 기술을 활용하였다. 리버스 테더링이란 스마트 장치와 호스트를 USB 인터페이스를 이용하여 연결하고, 스마트 장치에서 호스트의 빠른 네트워크를 이용할 수 있도록 하는 기술이다.[7]

리버스 테더링을 활용하여 스마트 장치에서 호스트의 네트워크를 이용할 수 있도록 설정한 이후에는 네트워크 속도를 원하는 수준으로 조절할 수 있는 매커니즘이 필요하다. 저장 장치의 성능을 조절하는 것과 비슷한 맥락으로 네트워크의 성능을 조절할 수 있는 환경이 필요한데, 이를 위해서 리눅스에서 제공하는 도구를 활용하였다.[8]

제 5 장 응용별 최적의 저장 장치 성능의 활용

앞서 소개한 저장 장치와 네트워크의 성능 에뮬레이션 환경과 사용자 관점에서의 반응 시간 측정 도구를 활용하여 저장 장치의 성능 변화에 따른 응용의 성능 변화 탐색 실험을 진행하면 응용별 최적의 저장 장치 성능을 찾을 수 있게 된다. 이번 절에서는 실험을 통해 얻은 응용 별 최적의 저장 장치 성능을 활용하여 실제 저장 장치의 수명을 최적화하기 위한 기법을 소개한다.

블록 당 삭제 횟수에 의해 수명이 결정되는 플래시 메모리 기반의 저장 장치에서 수명은 중요한 최적화 대상이다. 저장 장치의 쓰기 성능과 수명 사이에 트레이드오프 관계가 존재하여 이를 활용할 수 있게 된다면 수명 상의 큰 이득을 얻을 수 있게 된다. 연구 동기 부분에서의 실험을 통해 저장 장치로의 많은 쓰기 요청을 발생시키는 응용 중 저장 장치의 쓰기 성능을 저하시키더라도 사용자 관점의 응용의 성능에는 큰 영향을 미치지 않는 응용이 존재한다는 사실을 보였다. 이와 같은 응용에 대해서는 저장 장치의 쓰기 속도를 느리게 조절함으로써 수명을 최적화할 수 있는 여지가 존재한다. 이러한 기회를 최대한 활용하여 저장 장치의 수명을 최적화하기 위한 방안으로 여러 가지가 있을 수 있지만, 본 논문에서는 응용 개발자가 응용의 개발 시 활용할 수 있는 API(Application Programming Interface)를 개발하여 소개한다.

응용의 론칭 또는 작업이 요구하는 저장 장치의 성능은 다를 수 있다. 각각의 경우에 가장 적합한 저장 장치의 성능을 탐색하기 위해서 응용 개발자는 저장 장치와 네트워크의 성능 에뮬레이션 환경과 반응 시간 측정 도구를 활용하여 실험을 진행한다. 각 작업에 대해 적합한 저장 장치의 성능을 탐색하였다면 응용 개발자는 해당 속도를 요구하는 쓰기가 어떤

코드에 의해 발생했는지 알 수 있으므로, 이러한 쓰기를 발생시킨 부분에 개발한 저장 장치의 성능 조절 API를 추가하여 실제 저장 장치에서 쓰기 속도가 조절될 수 있도록 할 수 있다. 응용 개발자에게 제공하기 위해 개발한 API 목록은 표 3과 같다.

API 이름	API설명
DWSOn	DWS (Dynamic Write-speed Scaling) 모드의 시작
DWSOff	DWS 모드의 중지
isDWS	현재 DWS 모드의 ON / OFF 정보를 개발자에 전달
isForeground	해당 응용이 포어그라운드 또는 백그라운드에서 실행되고 있는지의 정보를 개발자에 전달
getWriteSpeed	현재 저장 장치의 성능 정보를 개발자에 전달
writeDWS	해당 쓰기 작업의 DWS를 위한 최적의 값을 시스템에 전달

표 3 저장 장치의 쓰기 성능 조절을 위해 응용 개발자에 제공되는 API 목록

isForeground()는 안드로이드 플랫폼 내부의 액티비티 매니저에 의해 관리되고 있는 실행 중인 응용의 목록을 활용하여 해당 응용이 포어그라운드 또는 백그라운드에서 실행되고 있는지의 정보를 응용 개발자에 전달한다. 이 API는 액티비티 매니저 클래스에 실행 중인 응용의 목록을 얻고 가장 상위에 존재하는 응용의 정보를 추출하여 응용 개발자가 입력한 응용의 이름과 일치하는 지의 여부를 검사하여 알려주는 새로운 메소드를 추가함으로써 구현할 수 있다.

isForeground()를 제외한 다른 API는 커널에서 관련 정보를 관리 또는 처리하는 시스템 콜을 추가하여 구현하였다. 이들 API의 구조는 그림 13과 같다.

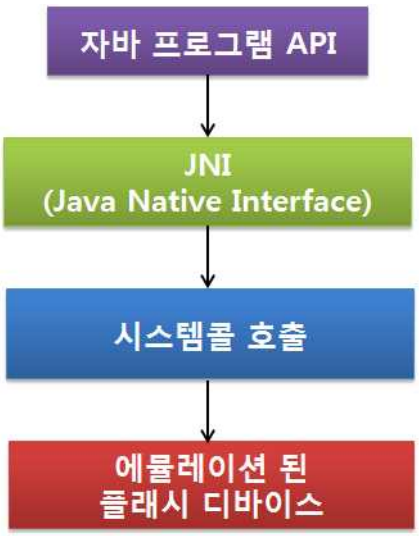

그림 13 저장 장치의 쓰기
성능 조절 API의 구조

응용 개발자가 자바를 사용하여 응용을 개발할 경우에 저장 장치의 쓰기 성능 조절을 위한 API를 사용하면 안드로이드 내에서 자바 코드가 C 또는 C++과 같은 다른 언어로 작성된 코드와 상호작용할 수 있도록 제공되는 JNI를 통해서 커널에 등록한 시스템 콜을 호출할 수 있도록 해준다. isForeground()를 제외한 5개 API의 구현을 위해 위의 구조를 활용하여 자바 라이브러리와 JNI에 새로운 메소드를 추가하고, 각 메소드가 호출하는 시스템 콜을 개발하였다.

DWSOn()과 DWSOff()는 커널 내부에서 DWS 기능을 총괄하는 변수를 하나 두어 그 값을 맞게 설정함으로써 해당 기능을 수행할 수 있다. isDWS()는 앞에서 언급한 DWS 기능을 총괄하는 변수의 현재 값이 응용 개발자에게 전달되도록 하면 된다. getWriteSpeed()는 오프라인으로 측정한 저장 장치의 쓰기 성능을 응용 개발자에게 제공하도록 한다. 마지막으로 writeDWS()는 해당 쓰기 작업이 응용 개발자가 설정한 쓰기 성능으로 수행되도록 하는데, 이 때 시스템 콜은 해당 응용의 성능을 저하시키지 않는 수준에서 가장 큰 폭으로 낮출 수 있는 저장 장치의 쓰기 속도를

에뮬레이션 된 플래시 디바이스에 전달하여 해당 속도로 쓰기 요청이 처리되도록 조절한다.

writeDWS()를 위한 시스템 콜에 대해 좀 더 자세히 알아보면, 에뮬레이션 된 플래시 디바이스에 요구되는 저장 장치의 성능 정보를 전달하는 시스템 콜을 구현하기 위해서는 블록 장치 연산을 위해 사용되는 바이오 디스크립터의 수정이 필요하다. 블록 장치로의 연산이 발생할 경우 연산의 종류와 크기 등의 정보를 저장하는 바이오 구조체가 생성되어 디바이스 드라이버에 전달된다.[9] 이러한 바이오 구조체에 해당 연산이 쓰기 연산일 경우 가장 적합한 성능을 나타내는 변수를 추가하여, 이 값이 다른 정보와 함께 디바이스 드라이버에 전달되도록 한다.

위와 같은 구조와 상세를 갖는 저장 장치의 성능 조절 API를 통해서 응용 개발자는 응용 개발 시 실험을 통해 탐색한 쓰기 성능을 안드로이드 프레임워크와 커널을 통하여 디바이스에 전달할 수 있고, 디바이스는 개발자로부터 받은 정보를 활용하여 쓰기 요청이 처리되는 속도를 조절함으로써 저장 장치의 수명이 연장되도록 할 수 있다.

제 6 장 실험 결과

제 1 절 실험 환경

실험은 갤럭시 넥서스 스마트폰에서 진행되었다. 갤럭시 넥서스에 탑재된 플래시 메모리 기반 저장 장치의 성능을 측정해 본 결과 읽기 연산의 경우 20MB/s, 쓰기 연산의 경우 10MB/s가 된다는 것을 확인할 수 있었다. 이는 해당 저장 장치에 있는 일정한 크기의 파일을 읽거나 저장 장치로 일정한 크기의 파일을 쓰는 간단한 프로그램을 작성하고, 프로그램이 수행되는데 걸리는 시간을 구하여 계산한 값이다. 이후에 수행된 모든 실험에서 저장 장치의 성능이 좋아지거나 나빠진다는 판단은 이 값을 기준으로 하였다.

저장 장치의 성능에 따른 응용의 성능 변화 실험을 위해서 저장 장치로의 쓰기 연산을 발생시키는 대표적인 응용의 론칭 또는 작업 20개를 선정하였다. 저장 장치의 쓰기 성능을 2.5MB/s, 5MB/s, 10MB/s, 15MB/s, 20MB/s로 변경시키면서 20개의 작업의 성능 변화를 실험하였다.

제 5장에서 제안한 응용 개발자가 응용 개발 시 활용 가능한 저장 장치의 성능 조절 API의 개발이 완료되지 않아 작업 별로 최적의 저장 장치의 쓰기 성능을 수동으로 설정한 후 응용의 성능을 측정하였다.

제 2 절 실험 결과

먼저 20개의 작업을 대상으로 수행한 저장 장치의 쓰기 성능 변화에 따른 응용의 성능 변화 실험 결과를 각기 다른 경향을 보이는 대표적인 3가지 경우로 나누어 소개한다. 그림 14는 3가지 응용에 대해 저장 장치의 쓰기 속도를 5가지로 조절할 때의 응용의 반응 시간 비율 실험 결과를 나타낸다.

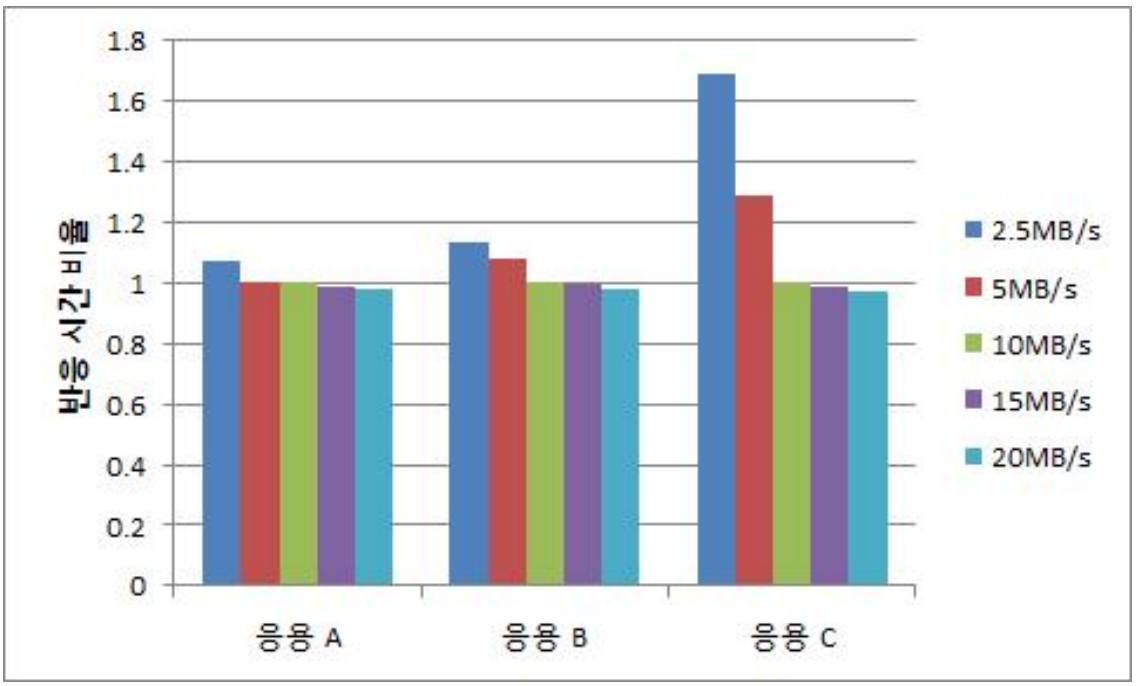

그림 14 저장 장치 성능 변화에 따른 3개의 응용의 반응 시간 변화 비율

Read / Write 처리량 (MB/s)	20 / 2.5	20 / 5	20 / 10	20 / 15	20 / 20
반응 시간	48.95665	46.81018	43.31296	43.10635	42.57235

표 4 Write 처리량에 따른 응용 설치 작업의 반응 시간 변화

Read / Write 처리량 (MB/s)	20 / 2.5	20 / 5	20 / 10	20 / 15	20 / 20
반응 시간	20.135498	15.34845	11.31372	11.18189	10.96854

표 5 Write 처리량에 따른 컨텐츠 다운로드 작업의 반응 시간 변화

응용 A는 포털 사이트 응용으로 사용자의 입력을 통해 응용을 론칭한 시점부터 첫 웹 페이지가 화면에 모두 그려지는 시점까지의 시간을 반응 시간으로 측정하였다. 저장 장치의 쓰기 속도가 2.5MB/s까지 떨어지더라도 사용자 경험에 영향을 미치는 반응 시간은 약 7.5%만 증가한 것을 확인할 수 있다. 표 1을 통해 반응 시간의 절대값으로 확인해 보아도 쓰기 속도가 10MB/s에서 2.5MB/s로 변화할 때 반응 시간은 약 0.37초 증가하는 수준에 그친다는 것을 알 수 있다. 저장 장치의 쓰기 성능이 응용의 성능에 직접적인 영향을 미치지 않는다고 해석할 수 있을 것이다. 저장 장치의 쓰기 속도가 변화할 때 이와 같은 양상을 보이는 작업들은 반응 시간이 저장 장치의 성능에 큰 영향을 받지 않는다는 것을 의미하기 때문에, 굳이 저장 장치의 성능을 좋게 하여 응용을 수행할 필요가 없다. 저장 장치의 성능을 조금 떨어뜨리더라도 응용의 성능은 큰 변화 없이 유지할 수 있고, 저장 장치의 성능과 수명 사이의 트레이드오프 관계를 활용하여 수명 상의 이득을 얻을 수 있다.

응용 B는 브라우저 응용으로 약 28MB의 컨텐츠를 다운받아 저장하는 작업을 수행하였다. 반응 시간은 컨텐츠의 다운로드가 완료되는 데까지 걸리는 시간으로 하였다. 저장 장치의 쓰기 속도가 5MB/s, 2.5MB/s로 떨어질 때 응용의 성능은 약 8.1%, 13% 감소하는 것을 확인하였다. 표 4를 통해 반응 시간 변화의 절대값은 약 3.5초, 5.6초라는 것을 확인할 수 있다. 이러한 수치는 사용자가 해당 작업을 포어그라운드에서 수행하느냐 또는 백그라운드에서 수행하느냐에 따라서 반응 시간의 변화를 지연으로 인식할 수도, 그렇지 않을 수도 있게 된다. 또한 응용 B의 작업에는 저장 장치뿐만 아니라 네트워크 모듈도 관여를 하게 된다. 네트워크로부터 필요한 데이터를 받아서 저장 장치로 저장을 하는 이와 같은 작업을 수행하는 경우에는 응용의 성능이 저장 장치와 더불어 네트워크 모듈의 영향을 받을 가능성이 크게 존재한다. 따라서 앞서 소개한 네트워크의 성능 에뮬레이션 환경을 활용하여 네트워크의 속도를 변화시키면서 응용의 성능이 어떻게 변화하는지를 추가로 탐색해 볼 필요가 있다. 이 결과는 뒤에서 자세하게 다루기로 하겠다.

응용 C는 마켓 응용으로, 약 20MB 크기의 응용을 다운로드 받은 후 저장 장치로 설치하는 작업을 수행하였다. 이 때의 반응 시간은 네트워크를 사용하여 응용을 다운로드 받는 시간은 제외하고, 백업 폴더에 저장된 응용의 APK 파일의 압축을 풀어 저장 장치로의 설치만을 수행하는 작업이 완료되는 시점까지의 시간이다. 저장 장치의 성능이 10MB/s에서 5MB/s, 2.5MB/s로 떨어짐에 따라 반응 시간이 28%, 69% 증가하였다. 표 5의 값을 통해 반응 시간 변화의 절대값은 약 4초, 8.8초라는 것을 확인할 수 있다. 이러한 경우에도 응용이 포어그라운드 또는 백그라운드에서 실행되고 있는지에 따라 사용자가 늘어난 반응 시간을 지연으로 인지하는지의 여부가 달라지긴 하겠지만, 최대 8.8초라는 큰 시간의 차이가 발생하기 때문에 저장 장치의 쓰기 속도를 조절하여 수명 상의 이득을 얻기 보다는 사용자 경험을 해치지 않기 위해 저장 장치의 성능을 변화시키지 않

는 것이 타당한 것으로 보인다.

위의 실험 결과를 토대로 20개의 응용 또는 작업을 가지고 수행한 저장 장치의 성능 변화에 따른 응용의 성능 변화 실험에서 응용은 3가지로 분류 가능함을 확인하였다. 응용의 성능이 저장 장치의 영향을 받지 않는 경우와 그 반대인 경우, 그리고 스마트 장치의 다른 구성 요소의 영향을 받아 실행 때마다 특성이 달라지는 응용이 그 것이다. 다음 절에서 응용 A가 저장 장치의 성능 변화에도 불구하고 반응 시간에 변화가 없었던 이유를 분석하고, 응용 B를 네트워크 모듈의 속도를 변화시키며 추가로 수행한 실험 결과를 소개한다.

제 3 절 실험 결과의 분석

제 2절의 실험 결과에서 응용 A의 경우 저장 장치의 쓰기 속도 변화가 응용의 성능에 큰 영향을 미치지 않는 이유를 분석하기 위해 각각의 쓰기 작업이 발생한 시간을 구해 쓰기 발생량과 반응 완료 시점 간의 관계를 분석하였다. 그림 15는 시간에 따른 누적 쓰기 발생량과 반응 완료 시점 간의 관계를 나타낸다.

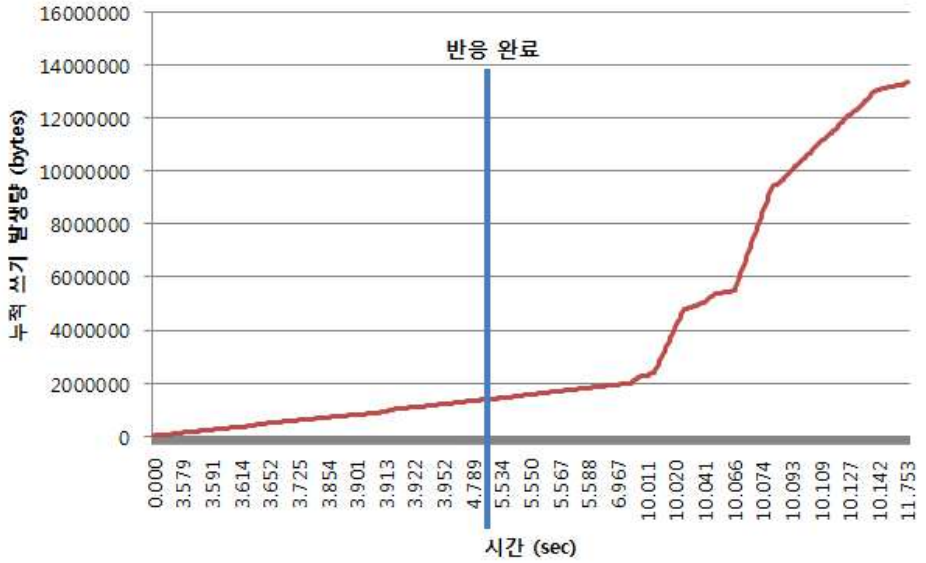

그림 15 쓰기 발생량과 반응 완료 시점 간의 관계

응용 A의 론칭 시 약 13MB의 저장 장치로의 쓰기 작업이 발생한다. 약 13MB의 쓰기 작업 중 반응 시간 이전에 발생하는 쓰기는 약 2MB로 약 15%만이 실제 반응 시간에 영향을 미치는 쓰기 작업이라는 것을 확인할 수 있다. 따라서 쓰기 속도를 늦추더라도 반응 시간에는 큰 영향을 미치지 않게 되는 것이다. 이러한 경우에는 쓰기 속도를 느리게 조절하여 사용자 경험의 저하 없이 저장 장치의 수명을 연장시킬 수 있게 된다.

그림 16은 응용 B의 컨텐츠 다운로드 작업을 네트워크 모듈의 속도를 변화시키면서 수행하였을 경우 반응 시간의 변화를 측정한 결과이다.

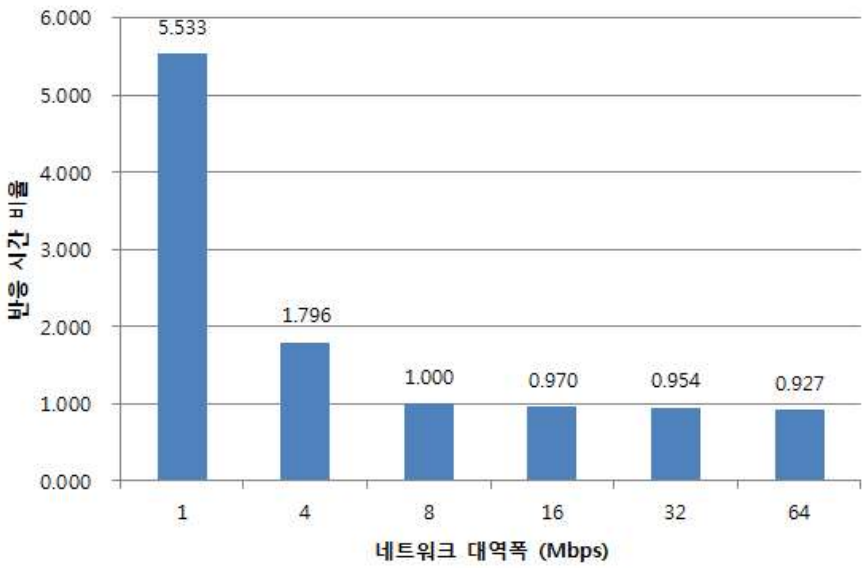

그림 16 네트워크 속도에 따른 응용의 반응 시간 변화 비율

실험할 당시 WiFi 모듈의 속도는 약 8Mbps로 측정되었고, 이 값을 기준으로 하여 네트워크 대역폭을 변화시키며 반응 시간의 변화를 측정하였다. 네트워크 대역폭이 1Mbps일 때(연결 상태가 양호하지 않은 3G 모듈을 사용할 경우의 속도) 응용을 다운로드 받는데 걸리는 시간이 약 5.5배 증가하는 것을 확인할 수 있다. 보통의 3G 모듈의 속도인 2 ~ 4Mbps 일 때에는 약 80% 정도 시간이 증가하였다.

네트워크 속도 변화에 따른 반응 시간 변화와 제 2절에서 확인 가능한 저장 장치의 쓰기 속도에 따른 반응 시간 변화를 종합하여 분석해보면 컨텐츠를 다운로드 받는데 걸리는 시간은 저장 장치의 쓰기 속도보다 네트워크 속도에 더 큰 영향을 받는다는 것을 알 수 있다. 컨텐츠의 다운로드와 같은 작업은 네트워크를 통해서 받아온 데이터를 저장 장치에 쓰는 작업이기 때문에 저장 장치에서 필요로 하는 성능은 그 당시의 네트워크 속

도에 종속적이고, 네트워크 속도가 낮아 시간 당 받아오는 데이터의 양이 적다면 저장 장치의 쓰기 속도를 굳이 빠르게 할 필요가 없다는 사실을 유추 가능하다. 따라서 저장 장치의 쓰기 속도를 조절하여 수명을 최적화 하는데 있어서 다른 컴포넌트의 현재 설정을 참고하여 쓰기 속도를 효과적으로 조절하는 것이 필요하다.

제 7 장 결 론

제 1 절 결 론

본 논문에서는 스마트 장치의 플래시 메모리 기반 저장 장치는 성능과 수명 사이의 트레이드오프 관계를 가지고 있고, 저장 장치가 응용이 요구하는 최대의 성능에 맞추어 설계되었다는 점에 착안하여 저장 장치의 쓰기 성능을 낮추어도 성능에 변화가 없는 응용이 존재하는지를 알아보는 실험을 진행하였다. 이러한 실험을 진행하기 위해서 실제 저장 장치의 성능을 에뮬레이션 할 수 있는 환경을 개발하였다.

개발한 환경을 활용하여 실험을 진행해 본 결과, 실제 응용의 성능이 저장 장치의 쓰기 성능에 영향을 받지 않는 경우가 존재하였고 이러한 경우에는 쓰기 성능을 해당 응용에 적합한 값으로 동적으로 조절하여 수명의 이득을 볼 수 있다는 사실을 파악하였다.

또한 개발한 환경을 활용해 응용별 최적의 저장 장치 성능을 찾고 이를 응용 개발자가 응용 개발 시에 명시하여 해당 응용을 실행할 경우 저장 장치의 성능을 설정 값으로 조절할 수 있도록 하는 API를 설계하였다.

제 2 절 향후 연구

본 논문에서는 개발한 저장 장치의 성능 에뮬레이션 환경을 이용하여 찾은 응용별 최적의 저장 장치 성능 값을 응용 개발자가 응용 개발 시에 활용할 수 있도록 하는 API를 설계하였다. 이러한 방식은 해당 응용을 위한 저장 장치의 쓰기 성능 값이 응용 개발 시에 고정된다는 단점을 가지고 있다. 따라서 향후에는 응용 개발자와 상관없이 시스템 내부에서 저장 장치로의 쓰기 연산을 발생시키는 컨텍스트를 찾고, 이 컨텍스트를 활용하여 응용에 적합한 저장 장치의 쓰기 성능을 저장 장치에 제공할 수 있는 기법을 연구할 계획이다.

참 고 문 헌

[1] J. Lim, et al., "LTmeter: an app launching analyzer for personal smart devices," Int. Conf. on Ubiquitous Information Technologies & Application, 2011.

[2] N. Tolia et al., “"Quantifying Interactive User Experience on Thin Clients,"" IEEE Computer, Vol. 39, No. 3, pp. 46-52, Mar. 2006.

[3] Sungjin Lee et al., "Lifetime Management of Flash-Based SSDs Using Recovery-Aware Dynamic Throttling," USENIX Conference on File and Storage Technologies, 2012.

[4] APK (file format), http://en.wikipedia.org/wiki/APK_(file_format)

[5] Dex File Format, http://www.retrodev.com/android/dexformat.html

[6] The Android mobile platform, http://www.emich.edu/compsci/projects/Master_Thesis_-_Benjamin_Speckmann.pdf

[7] Reverse USB Tethering, http://malsandroid.blogspot.kr/2012/01/reverse-usb-tethering.html

[8] Traffic Control HOWTO, http://linux-ip.net/articles/Traffic-Control-HOWTO/

[9] The bio structure, http://www.makelinux.net/books/lkd2/ch13lev1sec3

Abstract

NAND flash memory-based storage has the merit of rapid performance of random read/write and low power consumption, but the lifetime of the storage is restricted to the number of erase count.

There is trade-off relationship between the write performance and the lifetime of the flash memory-based storage, namely by reducing the write performance the lifetime of the storage can be increased. The other characteristic of the storage in the smart device is that the write performance is decided according to the required maximum performance among diverse applications. If there is an application that the response time is not affected by the write performance of the storage, we can optimize the storage lifetime by adjusting the write performance for that application low.

We developed the mobile storage evaluation environment to find out applications that the response time is not affected by the write performance. By conducting an experiment to investigate the relationship between the response time and the write performance, we identified that there are applications with such characteristics. This means that the lifetime of the storage can be optimized by controlling the write performance dynamically. We have a plan to develop the application programming interface for application developers that conveys the information of suitable write performance for that application to the storage and makes it possible to regulate the write performance.

keywords : Storage Lifetime, Characteristics of Storage Usage, Storage Emulation, Relationship between Storage Performance and Lifetime
Student Number : 2011-20920