(2011추계 우수발표논문) In-Storage Processing을 위한 SSD 소프트웨어 플랫폼 시뮬레이터 설계 및 구현

(A Design and Implementation of SSD Software Platform Simulator for In-Storage Processing)

박 대 동† 이 재 수††
(Daedong Park) (Jaesoo Lee)

홍 성 수†††
(Seongsoo Hong)

요 약 In-storage processing(ISP)는 저장 장치용 I/O 인터페이스의 병목현상으로 인한 전체 시스템 성능 저하 문제를 해결하기 위해 고안된 기법이다. 특히 최근의 플래시 메모리 기반 저장 장치들의 내부 데이터 전송 속도가 I/O 인터페이스의 발전에 비해 매우 빠르게 증가하기 때문에 ISP 의 필요가 더욱 증대되고 있다. 그러나 기존 연구들은 특정 시스템이나 응용에 제한되어 있어서 SSD 에 적용 및 범용적 응용 개발이 어렵다. 우리는 이런 문제를 해결하기 위해 ISP 를 위한 SSD 소프트웨어 플랫폼을 제안한다. 제안된 플랫폼은 ISP 응용의 동적 관리뿐만 아니라 이벤트 서비스를 통한 동적 기능 확장이 가능하다. 우리는 먼저 프로그래밍 모델을 제안하고 다음으로 이를 지원하는 SSD 소프트웨어 플랫폼을 고안한다. 마지막으로 제안하는 플랫폼을 검증하기 위한 시뮬레이터를 구현하고 실험을 통해 성능 및 기능 확장을 확인한다. 실험 결과 ISP 를 사용하는 응용의 응답 속도가 52.7% 감소하였으며 SSD 에 동적으로 새로운 기능을 추가하는 것이 가능함을 확인하였다.

키워드 : SSD 소프트웨어 플랫폼, ISP, OSD

Abstract In-storage processing (ISP) was developed to solve the performance degradation problem of I/O interface. Recently, the problem is getting worse because the speed of internal data bus of SSD has been improved much more than that of I/O interfaces. However, it is hard to use the existing ISP mechanisms for SSD because they aimed on specific systems or applications. In this paper, we propose SSD software platform to solve this problem. By using the proposed platform, user can dynamically install and remove various ISP applications and extend functionality through event service. We first propose a programming model and then design a SSD software platform. Finally, we implement a platform simulator and perform experiments to verify its performance enhancement and functional extensibility. The results show that the average response time of applications with ISP are decreased by 52.7% compared to that of without ISP. Also, user can dynamically attach a new functionality to SSD.

Key words : SSD Software Platform, ISP, OSD

1. 서 론

최근 저장 장치의 발달로 인해 플래시 메모리 기반의 SSD(Solid State Drive) 사용이 크게 증가하고 있다. 플래시 칩과 버스의 다양한 병렬화 기법 덕분에 SSD 내부의 데이터 전송 속도는 기존의 하드디스크에 비해 매우 빠를 뿐만 아니라 현재도 그 속도가 증가하고 있다. 반면 저장 장치와 호스트 시스템 사이의 I/O 인터페이스는 기존의 하드디스크를 위한 인터페이스를 공유하고 있으며 발전 속도는 SSD 의 발전 속도를 따라가지 못하고 있다. 그 결과 I/O 인터페이스가 데이터 전송의 병목 구간이 되어 전체 시스템의 성능을 저하시킨다. 단순히 I/O 인터페이스의 성능을 끌어올리는 것은 프로세서에 너무 많은 인터럽트를 발생시키게 되므로 다른 방법의 접근이 필요하다.

일반적으로 대량의 데이터 전송이 필요한 프로그램의 실제 연산은 간단한 경우가 많다. 예를 들어 대용량 데이터베이스에서 특정 조건을 만족하는 값들을 모두 찾는 경우 데이터의 변환과 값의 비교 연산이 모든 데이터에 대해 반복적으로 수행된다. 이런 연산을 저장 장치 내에서 수행하면 I/O 인터페이스를 통해 전송되는

· 본 연구는 삼성전자 메모리 사업부의 지원을 받아 수행하였음. (No. 0421-20100068, SSD를 위한 컴포넌트기반의 내장형 고성능 소프트웨어 플랫폼 연구)
· 이 논문은 제38회 추계학술발표회에서 'In-Storage Processing을 위한 SSD 소프트웨어 플랫폼'의 제목으로 발표된 논문을 확장한 것임
† 비회원 : 서울대학교 전기컴퓨터공학부
ddpark@redwood.snu.ac.kr
†† 비회원 : 삼성전자 메모리사업부
jslee@redwood.snu.ac.kr
††† 종신회원 : 서울대학교 전기컴퓨터공학부 교수
sshong@redwood.snu.ac.kr
논문접수 :
심사완료 :

정보과학회논문지: 컴퓨팅의 실제 및 레터 제XX권 제호(2011.XX)

데이터의 양을 크게 줄일 수 있다. 이처럼 저장 장치 내에서 응용의 일부를 수행하는 ISP 기법의 연구가 이루어졌다[1-3]. 그러나 기존 연구들은 특정 시스템이나 응용에만 제한되어 SSD 에 적용하기 어렵고, 범용적이고 다양한 응용을 개발하기 힘들다는 문제가 존재한다.

본 논문에서는 이러한 문제를 해결하기 위해 ISP 를 위한 SSD 소프트웨어 플랫폼을 제안한다. 먼저 응용 개발자를 위한 SSD 프로그래밍 모델을 고안하고 이를 지원하는 SSD 소프트웨어 플랫폼을 설계한다. 고안된 프로그래밍 모델은 ISP 응용뿐만 아니라 저장 장치 내의 특정 이벤트 발생 시 동작할 이벤트 처리기의 개발을 가능하게 한다. 또한 플랫폼 사용자는 쉽게 3rd 파티 응용을 저장 장치 내에 설치하고 관리할 수 있다.

우리는 제안하는 플랫폼의 성능 및 기능 확장을 검증하기 위해 시뮬레이터를 구현하고 실험을 수행했다. 실험 결과 많은 양의 I/O 가 발생하는 데이터베이스 응용의 select 쿼리 응답 시간이 47.3%로 감소하였다.

이 논문의 나머지 부분은 다음과 같이 구성된다. 2 장에서는 본 연구의 관련 연구를 정리하고 3 장에서는 해결하고자 하는 문제를 정의한다. 4 장과 5 장에서는 SSD 프로그래밍 모델과 소프트웨어 플랫폼에 대해 각각 설명한다. 6 장에서는 실험을 통한 검증을 보이고 마지막으로 7 장에서 결론을 내린다.

2. 관련 연구

저장 장치의 성능 및 기능 향상을 위해서 ISP 를 사용하는 방법과 파일 시스템의 개선을 통한 방법들이 활발하게 연구되어 왔다. 전자에는 프로세서의 처리 능력 한계로 인한 성능 저하 문제를 ISP 로 해결[1, 2]하는 연구와 분산 시스템에서 느린 네트워크로 인한 성능 저하 문제를 데이터가 위치한 노드에서 연산을 수행함으로써 해결[3]하는 연구들이 있다. 그러나 이들 연구는 매우 제한된 프로그래밍 모델로 작성된 ISP 응용 또는 미리 정해진 ISP 응용만을 수행할 수 있을 뿐만 아니라 특정 파일 시스템의 구조에 종속되어 있다. 따라서 3rd 파티가 다양한 목적의 ISP 응용을 개발하고 동적으로 설치 및 수행하기 어렵다.

파일 시스템을 통한 성능 및 기능 확장 연구들은 주로 특정 목적의 파일 시스템을 개발하는 것에 중점을 두었다. 범용 목적을 위한 일부 연구 중 대표적인 것으로 Object-based storage[4, 5]는 모든 데이터를 객체로 저장 장치에 기록하고 각 객체마다 목적에 맞는 다양한 속성을 부여한다. 또한 플래시 메모리와 같은 주요 저장 매체의 변화에 최적화된 파일 시스템도[6] 개발되었다. 그러나 이들 연구들은 3rd 파티에게 이를 활용할 수 있는 프로그래밍 모델을 제공하고 있지 못하다.

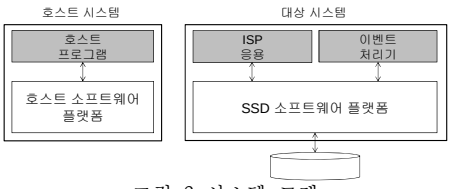

그림 2 시스템 모델

3. 문제 정의

3.1 시스템 모델

전체 시스템은 그림 1 과 같이 호스트 시스템과 대상 시스템으로 구성된다. 호스트 시스템은 사용자와 직접 상호작용 하는 시스템으로 데스크톱 PC 가 그 예이다. 호스트 시스템은 대상 시스템으로 데이터 읽기 및 쓰기 명령을 요청한다. 대상 시스템은 ISP 응용 및 이벤트 처리기가 동작하는 SSD 시스템이다. 두 시스템은 SATA 와 같은 I/O 인터페이스를 통해 서로 연결된다.

3.2 문제 정의 및 해결 방안

최근 개발되고 있는 SSD 내부의 데이터 전송 버스 속도는 6 Gbit/s 이상이며 현재도 그 속도가 증가하고 있다. 하지만 현재 가장 많이 사용되는 SATA2 인터페이스의 최대 속도는 3 Gbit/s 일 뿐만 아니라 실제 동작 속도는 일반적으로 이보다 낮다. 이로 인해 데이터 전송이 많은 프로그램의 동작 시 호스트 시스템과 SSD 사이의 I/O 인터페이스가 성능 병목 구간이 된다.

우리는 그림 2 와 같이 ISP 응용 및 ISP 지원 플래시 컨트롤러를 통해 호스트 시스템으로 전송되는 데이터의 양을 줄인다. 그림에서 화살표의 굵기가 데이터 전송량을 의미한다. 먼저 여러 개의 플래시 칩에서 전송되는 데이터를 사용해 ISP 지원 플래시 컨트롤러가 간단한 연산을 병렬적으로 수행한다. 이를 위해 ISP 지원 플래시 컨트롤러는 특수한 명령어 셋을 지원한다. 다음으로 ISP 응용을 통해 데이터에 대한 복잡한 연산을 수행함으로써 호스트 시스템으로 전송되는 데이터의 총량을 최소화한다. 이 경우 병목 구간은 SSD 내부로 변경된다. 뿐만 아니라 제안하는 시스템은 기존 ISP 기법들이 가지는 문제점을 해결할 수 있도록 동적 응용 설

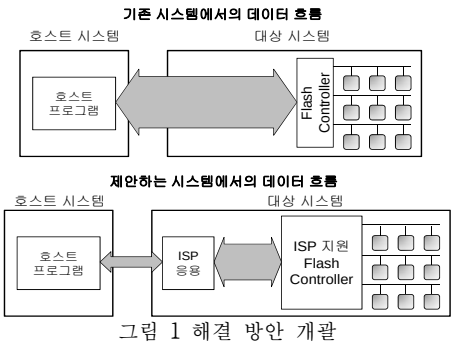

그림 1 해결 방안 개괄

치 및 이벤트 서비스를 지원한다. 4 장에서 먼저 ISP 응용 개발자를 위한 SSD 프로그래밍 모델을 제안하고 5 장에서는 SSD 소프트웨어 플랫폼의 설계를 기술한다.

4. SSD 프로그래밍 모델

본 장에서는 제안하는 SSD 프로그래밍 모델을 설명한다. SSD 를 사용하는 프로그램은 그림 1 의 회색 블록과 같이 호스트 프로그램, ISP 응용, 이벤트 처리기의 세 종류로 분류된다.

4.1 호스트 프로그램 인터페이스

호스트 프로그램은 호스트 시스템에서 수행되며 SSD 에 데이터를 읽고 쓰는 프로그램이다. 호스트 프로그램은 다음 세 종류의 인터페이스를 사용한다. 1) ISP 응용 관리 인터페이스, 2) ISP 응용과 통신하기 위한 인터페이스, 3) SSD 에 저장된 데이터 접근 인터페이스가 그것이다. 본 연구에서는 2)를 위한 인터페이스를 별도로 정의하지 않고 SSD 의 특정 위치에 데이터를 기록하여 호스트 프로그램과 ISP 응용이 통신하는 간단한 메커니즘을 사용하였다. 따라서 호스트 프로그램을 위해 다음 두 가지 인터페이스를 구체적으로 정의한다.

- ISP 응용 관리 인터페이스

  호스트 프로그램은 ISP 응용을 설치, 제거, 실행, 중지하거나 ISP 응용의 상태를 감시할 수 있어야 한다. 이를 위해 *install*, *start* 등의 ISP 응용 관리 인터페이스를 정의한다. 이들은 범용성을 높이기 위해 C 라이브러리로 구현되며 RPC 를 사용해 대상 시스템과 통신한다.

- SSD 에 저장된 데이터 접근 인터페이스

  호스트 프로그램은 대상 시스템의 데이터를 읽고 쓸 수 있어야 한다. 대상 시스템은 호스트 시스템에서 일반적인 저장 장치로 보이기 때문에 운영체제가 정의하는 파일 시스템 인터페이스를 사용한다. 본 연구에서는 *open*, *creat* 등의 리눅스의 파일 시스템 관련 시스템 함수들을 사용한다.

4.2 ISP 응용 인터페이스

ISP 응용은 대상 시스템에서 수행되며 호스트 프로그램의 기능 일부를 나누어 담당한다. 제안하는 시스템에서 ISP 응용은 다음 세 종류의 인터페이스를 사용한다. 각각은 1) 저장 장치의 데이터 접근을 위한 인터페이스, 2) 호스트 프로그램과 통신하기 위한 인터페이스, 3) HW 지원 데이터 병렬 처리를 위한 인터페이스이다. 앞에서 설명한대로 2)를 위한 인터페이스는 별도로 정의하지 않고 1)을 위한 인터페이스를 사용한다. 따라서 ISP 응용을 위해 다음 두 가지 인터페이스를 정의한다.

- 저장 장치의 데이터 접근 인터페이스

  ISP 응용은 이들 인터페이스를 통해 SSD 에 저장된 데이터에 접근할 수 있다. 우리가 제안하는 SSD 는 OSD(Object-based Storage Device)파일 시스템[4]을 사용하므로 모든 데이터는 객체로 플래시 메모리에 저장된다. 이 객체들에 접근하기 위해 OSD-2 표준[7]이 정의하는 *create_object*, *remove_object* 등의 명령어를 사용한다.

- HW 지원 데이터 병렬 처리 인터페이스

  대상 시스템에는 간단한 연산을 병렬적으로 수행하여 대용량 데이터를 빠르게 처리하는 HW 가 추가된다. 이 HW 연산 장치는 여러 개의 플래시 메모리 칩에서 동시에 데이터를 받아서 산술 연산, 비교 등을 수행하고 그 결과를 ISP 응용으로 전달한다. 따라서 이런 동작을 수행하는 HW 명령어들을 호출하기 위한 인터페이스를 정의한다.

4.3 이벤트 처리기 인터페이스

이벤트 처리기는 대상 시스템에서 수행되며 SSD 내에서 특정 이벤트가 발생했을 때 자동으로 수행되는 프로그램이다. SSD 내부의 이벤트에는 파일 시스템의 동작에 의해 발생하는 이벤트, FTL(Flash Translation Layer)의 동작에 의해 발생하는 이벤트, ISP 응용의 동작 및 예외 상황 발생으로 인한 이벤트가 있다. 객체의 생성이나 수정, 플래시 메모리의 쓰레기 수집(garbage collection)이나 hotspot 검출, ISP 응용의 메모리 오류 등이 SSD 내부 이벤트의 예이다. 이벤트 처리기의 개발을 위해 제공되는 인터페이스는 다음 세 종류가 있다.

- 이벤트 처리기 관리를 위한 인터페이스

  이벤트 처리기는 *register*, *unregister* 와 같은 인터페이스를 통해 자신을 특정 이벤트 발생시 호출되도록 등록하거나 반대로 해제할 수 있다. 또한 시스템에 등록된 이벤트 처리기들의 상태를 파악하기 위한 인터페이스를 사용할 수 있다. 이들 인터페이스는 유틸리티 프로그램에 의해 호출될 수도 있다.

- 대상 시스템의 데이터 접근 인터페이스

  이벤트 처리기는 SSD 에 저장된 데이터에 접근하기 위한 인터페이스를 사용할 수 있다. ISP 응용을 위한 인터페이스와 동일하게 OSD-2 표준 명령어를 사용한다

- HW 지원 데이터 병렬 처리 인터페이스

  ISP 응용과 마찬가지로 이벤트 처리기 역시 대용량의 데이터를 처리할 수 있다. 따라서 데이터의 병렬 처리를 위해 추가된 HW 를 활용하기 위한 인터페이스를 사용한다.

5. SSD 소프트웨어 플랫폼

본 장에서는 제안하는 SSD 소프트웨어 플랫폼에 대해 설명한다. 그림 3 은 전체 시스템의 소프트웨어 블록 다이어그램이다. 그림에서 볼 수 있듯이 SSD 소프트웨어 플랫폼은 다음 두 종류의 서브 시스템 블록들로 구분된다.

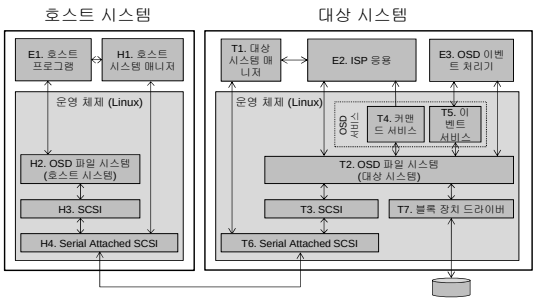

그림 3 해결 방안 개괄

- 호스트 시스템 소프트웨어 플랫폼의 서브 시스템
- 대상 시스템 소프트웨어 플랫폼의 서브 시스템

5.1 호스트 시스템

호스트 시스템의 소프트웨어 플랫폼을 구성하는 네 가지 서브 시스템들은 다음과 같다.

- H1. 호스트 시스템 매니저

  호스트 프로그램에게 ISP 응용을 관리하는 인터페이스를 제공하고 이들 인터페이스가 호출되면 RPC를 통해 대상 시스템으로 명령을 전달한다.

- H2. OSD 파일 시스템(호스트 시스템 부분)

  호스트 시스템에 위치하는 OSD 파일 시스템으로 Linux 표준 파일 인터페이스를 유저 레벨 프로그램에게 제공한다. 인터페이스가 호출되면 이를 알맞은 OSD-2 명령어로 변환해 SCSI 드라이버로 전송한다.

- H3. SCSI 드라이버

  OSD-2 명령어를 SCSI CDB(Command Descriptor Block)으로 변환하여 하단에 위치한 Serial Attached SCSI 로 전송한다.

- H4. Serial Attached SCSI

  SCSI CDB 를 SATA 패킷으로 변환하고 SATA 인터페이스를 통해 대상 시스템과 패킷을 주고받는다.

5.2 대상 시스템

대상 시스템 소프트웨어 플랫폼을 구성하는 서브 시스템들은 다음과 같다.

- T1. 대상 시스템 매니저

  호스트 시스템 매니저와 통신하면서 ISP 응용 관리 인터페이스 호출을 실제로 처리한다. 호스트 시스템과의 통신은 RPC 를 사용한다.

- T2. OSD 파일 시스템(대상 시스템 부분)

  대상 시스템에 위치하는 OSD 파일 시스템으로 OSD-2 명령어 인터페이스를 제공한다. SCSI 드라이버로부터 OSD-2 명령어를 전달받고 저장 매체에 기록된 object 에 직접 접근하여 알맞은 동작을 수행한다. Object 를 생성, 수정, 제거하고 object 의 attribute 를 설정하는 작업 등을 담당한다.

- T3. SCSI 드라이버

  Serial Attached SCSI 에서 전달된 SCSI CDB 패킷을 분해하여 OSD-2 명령어를 추출한다. 추출된 명령어에 따라 OSD 파일 시스템 인터페이스를 호출한다.

- T4. 커맨드 서비스

  호스트 프로그램과 ISP 응용간의 통신을 지원한다. 이를 위해 대상 시스템 초기화 시에 두 개의 특수 객체를 생성한다. 하나는 호스트 프로그램이 ISP 응용으로 명령을 전송할 때 쓰는 명령어 수신 객체이며 다른 하나는 ISP 응용의 처리 결과를 호스트 프로그램으로 전송할 때 쓰는 응답 객체이다. 커맨드 서비스는 주기적으로 명령어 수신 객체를 감시하여 새로운 명령어가 들어왔는지 검사하고 새 명령어가 수신 되면 알맞은 ISP 응용을 찾아서 수행한다.

- T5. 이벤트 서비스

  SSD 내의 이벤트 발생을 감시하고 이벤트 발생 시 해당 이벤트에 등록된 이벤트 처리기를 자동으로 호출한다. 이벤트 처리기의 등록, 해제 및 감시를 위한 인터페이스를 제공한다.

- T6. Serial Attached SCSI

  SATA 인터페이스로 들어온 ATA 패킷을 분해하여 SCSI CDB 를 검출하고 이를 SCSI 드라이버로 보낸다.

- T7. 블록 장치 드라이버

  블록 장치에 읽기 및 쓰기 명령을 내린다. 본 연구에서는 SSD 용 드라이버가 사용된다. 이 장치 드라이버는 블록 캐시, FTL 및 기타 기능을 포함한다.

6. 구현 및 검증

본 장에서는 제안하는 소프트웨어 플랫폼 검증을 위한 시뮬레이터의 구현에 대해 설명한 후 성능 향상 및 동적 기능 확장 검증 실험 결과를 설명한다.

6.1 SSD 소프트웨어 플랫폼 시뮬레이터 구현

우리는 그림 4 와 같은 하드웨어 위에 제안하는 SSD 소프트웨어 플랫폼 시뮬레이터를 구현하였다. 기존의 상용 SSD 는 범용 프로세서와 메모리가 없기 때문에 본 시스템을 위해 사용할 수 없다. 따라서 우리는 프로세서 및 메모리를 탑재한 NAS(Network Attached Storage)를 사용하여 ISP 가 가능한 SSD 를 에뮬레이션 하였다. 실험 환경과 제안한 시스템은 병목 구간과 연산 성능에 차이가 존재한다. 다음은 차이를 극복한 방법이다.

- 병목 구간의 차이

  실험 환경의 병목 구간은 NAS 와 SSD 사이의 인터페이스이다. 우리는 TCP 흐름 제어 기술을 통해 호스트 시스템과 NAS 사이가 병목 구간이 되도록 속도를 조절하였다.

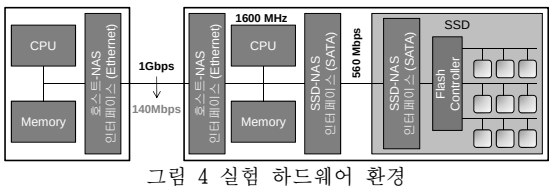

그림 4 실험 하드웨어 환경

• 연산 성능

제안하는 시스템은 SSD 에 탑재되는 400 MHz 의 프로세서와 데이터 병렬 처리를 위해 특별히 설계된 HW 를 연산 장치로 사용한다. 별도의 HW 장치 없이 제안한 시스템의 가능성을 확인하기 위해 시뮬레이터는 강력한 성능의 프로세서를 사용하였다.

6.2 성능 및 기능 검증

우리는 제안하는 시스템의 성능 및 기능 검증을 위해 두 가지 실험을 수행하였다. 먼저 ISP 를 통한 성능 향상을 검증하기 위해 간단한 데이터베이스 프로그램을 구현하고 사용자의 명령에 대한 응답 시간을 측정하였다. 실험에 사용한 데이터베이스는 {계좌, 이름, 나이, 성별, 잔고}를 임의로 생성하여 구축하였으며 특정 조건을 만족하는 데이터들을 검출하는 select 명령을 입력했을 때 결과가 나오기까지의 시간을 측정했다. 비교 대상은 호스트 프로그램만으로 구현된 경우와 select 쿼리 처리를 ISP 응용으로 분리해 구현한 경우다. ISP 응용은 일반적인 select 쿼리를 처리할 수 있도록 구현되었으며 동적으로 SSD 내에 설치 및 제거가 가능하다. 성능 실험 결과는 그림 5 와 같다. 가로 축은 데이터베이스의 크기(MB)이고 세로 축은 select 명령의 응답 시간(sec)이다. 실험 결과 ISP 응용을 사용하였을 때 응답 시간이 호스트 프로그램만을 사용하였을 때보다 평균 52.7% 감소하였다. 이를 통해 병목 구간이 SSD 내부 프로세서로 변경된 것을 확인할 수 있다.

다음으로 이벤트 처리기를 통한 동적 기능 확장을 검증하였다. 실험에서는 SSD 에 저장되는 mp3 데이터의 품질을 자동으로 변환하는 이벤트 처리기를 동적으로 설치하고 동작을 확인하였다. 구현한 이벤트 처리기는 SSD 의 여러 이벤트 중 파일 시스템의 *write_object* 명령어 발생 이벤트에 등록된다. 이벤트 처리기가 호출되면 데이터가 기록될 때 그 헤더를 검사하고 mp3 데이터인 경우 bitrate 를 128 Kbps 로 변환한다. 이벤트 처리기 등록 후 사용자가 320 Kbps 의 mp3 데이터를 SSD 로 복사하면 잠시 후에 자동으로 128 Kbps 의 mp3 데이터로 변환되어 기록되는 것을 확인할 수 있었다.

이와 같은 성능 및 기능 향상을 얻기 위해서 발생하는 추가 비용은 SSD 컨트롤러에 범용 프로세서를 추가하고 플래시 컨트롤러의 기능을 확장하기 위해 발생한다. 2012 년 현재 유통되는 Silicon Motion 사의 SM2250 SSD 컨트롤러는 개당 $3.5 수준으로 SSD 에서 차지하는 비중이 매우 적다. 본 연구에서 제안하는 400 MHz 성능의 ARM 프로세서는 개당 $1 미만이며 간단한 산술 연산 회로 정도를 플래시 컨트롤러에 추가하는 것으로 발생하는 생산 비용의 증가는 매우 미미하다. 따라서 본 연구 결과를 적용함으로써 발생하는 추가 비용은 $1 미만이 될 것으로 예측된다.

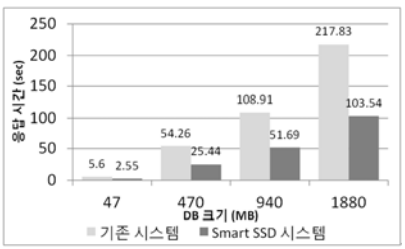

그림 5 DB 응용의 select 쿼리 응답 시간

7. 결론

우리는 SSD 프로그래밍 모델을 제안하고 이를 지원하는 SSD 소프트웨어 플랫폼을 설계하였다. 또한 시뮬레이션을 통해 제안된 SSD 소프트웨어 플랫폼의 유용성을 검증하였다. 제안된 SSD 소프트웨어 플랫폼은 3rd 파티 ISP 응용의 동적 다운로드 및 상태 관리가 가능할 뿐만 아니라 3rd 파티 이벤트 처리기의 등록을 통해 동적 기능 확장이 가능하다. 또한 이를 적용하기 위해 발생하는 SSD 의 생산 비용 증가 폭은 매우 작을 것으로 예측된다. 따라서 본 연구 결과를 활용하여 고성능의 SSD 개발 및 SSD 를 위한 다양한 응용 개발이 이루어질 수 있을 것으로 기대한다. 향후 연구에서는 HW 지원 병렬 연산 처리 활용의 극대화를 위한 FTL 및 플래시 컨트롤러의 데이터 저장 기법을 연구할 예정이다.

참 고 문 헌

[1] A. Acharya, M. Uysal, and J. Saltz, “Active Disks: Programming Model, Algorithms and Evaluation”, ACM SIGPLAN Notices, vol. 33(11), pp. 81~91, 1998.

[2] E. Riedel, C. Faloutsos, G. Gibson, and D. Nagle, “Active disks for large-scale data processing”, IEEE Computer, vol. 34(6), pp. 68~74, 2001.

[3] R. Reambasu, A. A. Levy, T. Kohno, A. Krishnamurthy, and H. M. Levy, “Comet: An Active Distributed Key-Value Store”, In Proc. of OSDI 2010.

[4] M. Mesnier, G. R. Ganger, and E. Riedel, “Object-Based Storage”, IEEE Communications Magazine, vol. 41(8), pp. 84~90, 2003.

[5] F. Wang, S. Brandt, E. Miller, and D. Long, “OBFS:a file system for object-based storage devices”, In Proc. of 12th NASA Goddard, 21st IEEE Conference on Mass Storage Systems and Technologies, April 2004.

[6] S. Ahn, J. Choi, D. Lee, S. H. Noh, S. Min, and Y. Cho, “Design, Implementation, and Performance Evaluation of Flash Memory-based File System on Chip”, Journal of Information Science and Engineering, vol. 23(6), pp. 1865~1887, 2007.

[7] T10, “The Object-Based Storage Device Commands-2”, INCITS, 2009.