SSD 내 데이터 처리 연구 동향

성균관대학교 ▌강윤지·한규화·신동군*

1. 서 론

빅데이터 분석이 보편화되면서, 클라우드 컴퓨팅과 데이터 센터에서는 높은 스토리지 대역폭에 대한 요구가 급증하고 있으며, 이로 인해 빠른 대역폭을 가진 솔리드-스테이트 드라이브 (SSD)가 하드-디스크 드라이브 (HDD)를 대체하고 있다. 최근 들어 PCIe 슬롯 기반의 NVMe 인터페이스가 등장하면서 SSD는 PCIe 레인 당 약 1GB/s의 대역폭을 호스트에게 제공할 수 있다. 이러한 높은 대역폭으로 인해, SSD는 내부 펌웨어 오버헤드를 해소하기 위해 멀티-코어 구조를 도입하였다. SSD 내의 각 코어는 호스트에서 전달된 I/O 요청을 병렬적으로 처리하여 빠른 I/O 처리가 가능하다.

SSD는 플래시메모리의 erase-before-write 특성과 program/erase cycle 수의 제한 등을 관리하여 기존의 블록 인터페이스를 호스트에게 제공하기 위해 SSD 내장 펌웨어인 플래시 변환 계층 (FTL)을 포함한다. FTL은 호스트가 전달한 블록 주소를 플래시 메모리 페이지로 매핑하는 역할을 수행하고, SSD 내의 여유 공간이 부족할 시 가비지콜렉션(garbage collection)을 통해 무효화된 플래시 메모리 페이지를 회수하는 작업을 수행한다. 또한 플래시메모리의 오류 처리, 플래시메모리 블록들의 마모도 평준화 등의 작업도 수행한다.

최근에는 필터링이나 정렬과 같은 간단한 컴퓨팅 연산을 SSD로 오프로딩하는 연구들이 활발히 진행되고 있다[1-12]. 이러한 데이터 처리 방법을 In-Storage Computing (ISC)이라고 한다. ISC를 사용하면 스토리지 장치와 호스트 CPU가 병렬적으로 작업을 처리할 수 있기 때문에 전체 시스템 성능이 증가한다. 그리고, SSD 내부의 저전력 CPU를 사용한 연산을 통해서 전체 시스템의 전력 소모를 감소시킬 수 있다. 또한, 스토리지 내에서 필터링된 데이터만을 호스트로 전달하여 호스트와 디바이스 간의 데이터 전송을 줄일 수 있다. 특히, 최근 SSD는 다중 채널과 다중 칩을 통해서 IO 대역폭을 증가할 수 있지만 호스트와 SSD간의 PCIe 인터페이스의 대역폭 제약이 전체 대역폭을 제한하고 있는 문제가 있어 호스트와 디바이스 간의 데이터 전송량을 감소시키는 ISC 기법은 매우 유용하다.

삼성전자를 비롯한 여러 국내외 기업들에서는 ISC를 지원하는 SSD의 프로토타입을 개발하고, 맵-리듀스와 관계형 데이터베이스 쿼리의 일부를 SSD로 오프로딩했을 때의 성능 개선을 보여 주었다. 또한, 학계에서도 SSD에 작업을 오프로딩하기 위한 프레임워크를 제안하였으며, FPGA를 사용한 데이터 처리 가속, PCIe의 peer-to-peer 통신을 활용한 GPU로의 직접적인 데이터 전달 등 여러 방향으로 ISC를 사용하는 연구를 진행하였다.

SSD는 호스트 PC와 달리 ARM기반의 임베디드 프로세서를 사용하기 때문에, 호스트의 작업을 ISC를 통해 SSD에게 오프로딩하기 위해서는 다음과 같은 이슈를 고려해야 한다.

- 호스트 개발자가 디바이스 내에서 동작하는 프로그램을 쉽게 개발하고, 이를 사용할 수 있는 프로그래밍 모델이 필요하다. 프로그래밍 모델은 호스트에게 프로그램을 디바이스에 인스톨하고 이를 실행할 수 있는 방법을 제공해야하며, 연산 결과를 호스트로 가져오는 방법도 제공해야한다.
- SSD에서 사용하는 프로세서는 호스트 프로세서에 비해 낮은 성능을 제공하지만 같은 연산을 저전력으로 처리할 수 있고 여러 SSD에서 병렬 실행이 가능하다. 작업을 오프로딩할 때에는 디바이스와 호스트 간의 데이터 전송량을 줄이는 것을 목표로 하지만, 작업의 특성에 따라서 데이터 전송량을 줄이지 못할 수 있다. 따라서, 디바이스로 오프로딩할 작업을 선정할 시에 성능, 전력소모, 데이터 감소 등을 고려해야 한다.

* 종신회원

- ISC 작업은 SSD 내의 프로세서를 사용하기 때문에 연산 작업이 많아지면 FTL 작업을 방해하여 디바이스의 I/O의 성능에 영향을 줄 수 있다. ISC 작업과 FTL 작업을 소프트웨어 기반으로 격리하거나 하드웨어를 고정적으로 분리하는 방법 등을 통해 ISC 작업과 FTL 작업의 간섭을 최소화 하는 것이 필요하다.

본 문서에서는 현재까지 기업과 학계에서 발표된 ISC-SSD 관련 연구들을 소개하고 ISC-SSD를 개발하기 위해 해결해야 하는 문제점들을 찾아본다. 또한, ISC-SSD의 향후 전망에 대해 서술하며 마무리한다.

2. 빅데이터 처리 프레임워크

빅데이터에 대한 분석 및 처리를 위해 여러 개의 분산 컴퓨터를 사용하는 대표적인 프레임워크로 하둡(Hadoop)[13]과 스파크(Spark)[14]가 있다. 하둡은 2006년 개발되어 아파치 재단에 의해 오픈 소스로 관리되고 있다. 2012년에는 1.0 버전을 배포하였으며, 현재는 3.2.0 버전까지 개발이 완료되었다. 하둡은 하둡 분산 파일시스템 (HDFS)에 의해 분산 저장된 대용량의 비정형 데이터를 여러 개의 컴퓨팅 자원에서 병렬적 분산처리를 가능하게 지원하는 프레임워크이다. 하둡은 맵(map)-리듀스(reduce)의 매 단계마다 스토리지에 결과를 저장하고 다시 읽어오는 작업을 수행하기 때문에 분석 시간이 오래 걸린다. 이 문제를 해결하고자 DRAM에 분석을 위한 모든 데이터를 저장하고 DRAM 내에서 분석을 진행함으로써 맵-리듀스 분석을 가속화하는 스파크가 등장하였다.

스파크는 2009년 UC 버클리의 AMPLab에서 시작되어 현재는 아파치 재단에 의해 오픈소스로 관리되고 있다. 그림 1은 스파크의 에코 시스템을 보여준다. 스파크는 분산 저장된 다양한 비정형 데이터에 대한 분석이 가능하며, docker나 openstack 환경에서 사용 가능하다. 스파크는 비정형데이터를 DRAM으로 읽어와 resilient distributed dataset (RDD)을 구성한 후 분석을 수행한다. 스파크는 Scala, java, python, R을 통해 구현되어 있으며, Spark SQL, Spark Streaming, MLlib, GraphX를 통해 응용들에게 데이터베이스 쿼리의 분산처리, 스트리밍 방식의 데이터 처리, 머신러닝을 통한 데이터 분석, 그래프 구조 처리 등을 지원한다.

이러한 빅데이터 처리 프레임워크를 ISC를 통해 가속하기 위해서는 프레임워크에서 SSD의 자원을 관리하고 ISC 태스크를 할당하며 결과를 전송받을 수 있도록 구조적인 변화가 요구된다.

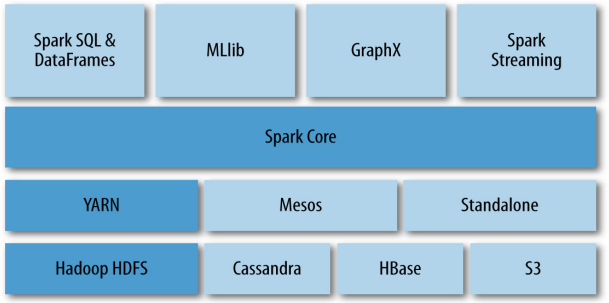

그림 1 Spark 에코 시스템 (출처: [15])

3. In-Storage Computing 연구 동향

SSD가 등장하기 이전부터 저장장치 내에서 데이터베이스 등 호스트 작업의 일부를 오프로딩하여 처리하는 전용 하드웨어를 제작하려는 시도들이 존재하였다. CASSM[16], RAP[17], RARES[18]는 HDD의 트랙마다 프로세서를 추가하여 ISC를 수행하는 방법을 제안하였다. 그러나, ISC를 위해 추가된 하드웨어가 차지하는 스토리지 내 공간으로 인해 해당 기술은 널리 사용되지 못하였다. DBC[19]와 SURE[20]는 HDD내의 헤드 마다 프로세서를 추가하는 방법을 제안하여 시스템의 성능을 향상 시켰지만, 복잡한 연산을 수행하지 못한다는 단점이 있다. 90년대 후반에는 디스크마다 하나의 CPU를 추가한 Active Disks [21] 연구가 제안이 되었고, 최근 ISC 기법들에서 사용하는 프로그래밍 모델들과 유사한 disklet이라는 스트림 기반 프로그래밍 모델을 제안하였다. 또한, disklet의 자원을 관리하는 디스크 내의 런타임인 diskOS를 도입하였다. 2013년에 Active flash[1]를 시작으로 SSD 내에서 ISC를 지원하기 위한 다양한 연구들이 발표되었다.

3.1 Active Flash [1]

Active Flash에서는 SSD 내에서 데이터를 처리하여 데이터 전송 시간과 에너지 비용을 줄여 HPC(High Performance Computing)에서의 데이터 분석을 효율적으로 수행하였다. Active flash에서는 에너지와 성능 모델링을 자세히 제시하여 SSD 컨트롤러에서 데이터 분석이 가능한 조건을 모델링 하였다. 또한 OpenSSD 개발 플랫폼에서 프로토타입을 제작하여 성능평가를 진행하였다. 명령 포맷은 연산, LBA 익스텐트의 리스트, 연산 옵션으로 간단하게 구성하였다. 실험 결과, SSD에서 간단한 데이터 분석 워크로드를 수행할 수 있음을 보여주었으며, 특히 에너지 부분에서 큰 효과를 얻었다.

3.2 SmartSSD [2, 3]

삼성전자는 2013년에 최초로 실제 하드웨어를 사용하여 MLC 플래시메모리 기반의 ISC-SSD의 프로토타입인 smartSSD를 구현하였으며, 이를 이용하여 데이터 집약적인 작업을 SSD로 오프로딩하여 성능을 측정하였다. SmartSSD의 내부 실행 엔진을 통하여 플래시메모리에 저장된 데이터를 가지고 호스트에서 요청한 서브 작업을 수행한다. 호스트에서는 SSD가 처리중인 I/O의 양을 모니터하여 연산 작업을 요청할 것인지 결정한다. 예를 들어, 현재 디바이스가 I/O가 없는 유휴 상태일 때에는 ISC 작업을 최대로 수행하고, 사용자로부터 대기 중인 I/O 요청이 있을 때에는 ISC 작업을 중단한다.

이 기법에서는 ISC 작업을 관리하고 실행하기 위해 I/O 명령을 추가하였다. 기본 통신 프로토콜에 관계없이 동일한 API 세트를 사용하기 위하여 확장된 객체 기반 인터페이스를 사용한다. 이 객체 기반 인터페이스에는 ISC 작업의 인자 값, 처리할 논리 주소와 데이터가 저장될 주소를 파라미터로 지정할 수 있다. ISC 명령에는 작업을 초기화하고 작업의 오브젝트를 생성하는 create_object, 작업을 실행하는 execute_object, 그리고 결과를 호스트로 읽어오거나 미리 지정한 논리 주소에 저장하는 read_object가 존재한다.

객체 기반으로 호스트와 디바이스가 통신을 할 때에는 양방향 통신을 사용하여 명령에 대해서 응답을 기다리고 다음 명령을 차례대로 수행한다. SmartSSD로 오프로딩 할 작업은 C 언어로 작성된다. SmartSSD는 동적 메모리 할당을 지원하지 않기 때문에 작업을 수행하기 전에 작업은 디바이스에 미리 로딩되어 있어야 하며 인풋과 아웃풋 오브젝트를 위한 메모리 공간도 예약되어 있어야 한다. SmartSSD는 웹 로그 분석기와 데이터 필터라는 두 가지 응용프로그램으로 성능을 평가하였다. smartSSD는 호스트에서 모든 작업을 처리하는 것 대비 에너지 사용을 50% 절약했으며, 2-3 배 성능도 향상되었다.

SmartSSD를 활용하여 DB를 가속하는 연구[3]도 진행했는데, SQL문으로 구성된 관계형 데이터베이스의 쿼리 처리를 위하여 세션 기반 프로토콜인 OPEN, CLOSE, GET을 정의하였다. OPEN은 세션의 시작을 알려주며 쓰레드와 메모리를 포함한 런타임 자원을 요청하고 CLOSE는 세션이 끝났음을 알리며 할당된 자원을 해지 한다. GET은 호스트에서 프로그램의 상태를 모니터링 하거나 프로그램의 결과를 얻을 수 있다. 추가로 다양한 API를 사용자 프로그램에 제공하고 있는데, 콜백 함수를 등록하는 API, thread 관련 API, 데이터가 로드하여 DRAM에 고정키는 데이터 API 등을 제공한다. SQL 서버를 사용한 실험에서 2.7배 이상의 성능 향상과 3배 이상의 에너지를 절약하였다.

3.3 Willow [4]

Willow에서는 SSD에서 실행할 어플리케이션의 구현 이슈들을 다루었다. 응용 프로그램 및 파일시스템/운영체제 프로그래머가 SSD의 기능을 확장하기 위해 SSD 어플리케이션의 프로그래밍 모델을 제안하였다. 그림 2는 Willow SSD의 구조를 보여준다. Willow SSD는 여러 SPU(storage processor unit)을 포함하고 있으며 SPU에서는 SPU-OS라고 불리는 작은 운영체제를 수행하고 있다. 호스트의 willow 드라이버는 호스트에서 SSD 어플리케이션을 제어하기 위하여 Host RPC Endpoint (HRE)라는 객체 세트를 만들고 관리한다. 드라이버는 RPC 요청을 수신하기 위한 HRE ID라는 고유한 식별자를 제공하고 SSD 어플리케이션이 사용자 공간 메모리와 willow간의 DMA 전송을 통해 결과를 받는다.

SSD내의 데이터 보호를 위해 SPU-OS에서는 각 프로세스의 권한 집합을 테이블 형태(Perm. Table)로 유지 관리한다. 호스트 커널 드라이버는 ISC 작업이 데이터에 대한 접근 권한을 필요로 하면 권한 RPC 요청을 SPU-OS로 보내어 권한 테이블을 업데이트 한다. 그리고 SPU 프로세서의 로컬 메모리에 코드와 데이터로의 접근만 허용하기 위해 SPU의 프로세서는 세그먼트 레지스터를 제공하고 현재 세그먼트 외부의 액세스를 허용하지 않는다.

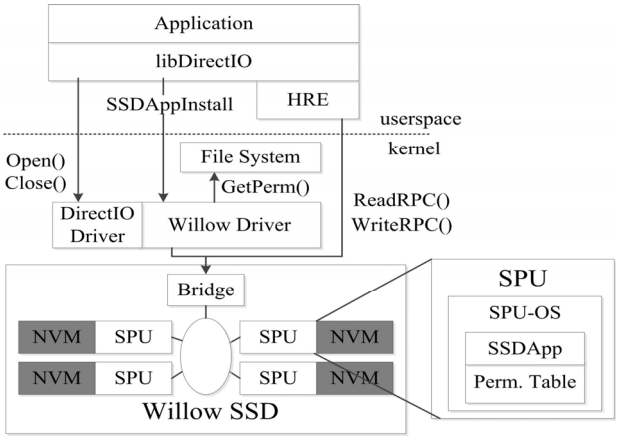

그림 2 Willow SSD의 구조

3.4 BlueDBM [5]

대규모 데이터 집합의 효율적인 분석을 위해서 여러 노드로 구성된 BlueDBM 구조가 제안되었다. 각 노드

는 BlueDBM 스토리지와 호스트 서버로 구성된다. 효율적으로 ISC를 사용하기 위하여 SSD 컨트롤러 오버헤드, 스토리지 오버헤드와 같은 부분을 최적화하여 I/O 응답시간을 감소시켰다. 먼저, SSD 컨트롤러의 주소 변환과 가비지콜렉션을 호스트의 파일 시스템으로 이동시켜 SSD 컨트롤러의 오버헤드를 줄이고, 모든 사용자 요청이 OS의 시스템 스택을 우회하여 하드웨어에 직접 전송될 수 있도록 하여 I/O 응답시간을 향상시켰다. 또한, 스토리지 디바이스간의 데이터 전송을 위하여 네트워크 소프트웨어 스택 오버헤드를 감소시켰다. 마지막으로 BlueDBM은 가속 프로그램 개발을 위한 Connectal[6]이라는 프레임워크를 사용하였다. Connectal은 RPC(Remote Procedure Call) 와 비슷한 인터페이스를 생성해주고 높은 대역폭 데이터 전송을 위한 메모리 매핑된 DMA 인터페이스도 제공한다.

3.5 Biscuit [7]과 YourSQL [8]

삼성전자가 제안한 biscuit은 개발자에게 ISC-SSD로 작업을 오프로딩할 수 있도록 데이터 플로우 기반의 고수준의 프로그래밍 모델을 제공한다. 그림 3은 biscuit의 프로그래밍 모델을 보여준다. Biscuit은 호스트에게

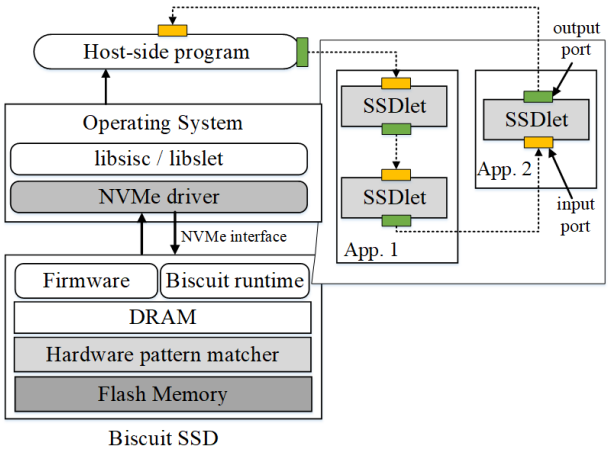

그림 3 Biscuit 프레임워크의 동작 과정

libslet과 libsisc 라이브러리를 제공한다. libslet은 SSD 내의 모듈 (SSDlet)을 프로그래밍 하는데 사용되며, libsisc는 호스트 측 프로그램에서 SSD 내의 모듈과의 통신을 위해 사용된다. Biscuit은 포트 I/O를 통해 SSDlet 간, 호스트 프로그램과 SSDlet 간의 데이터 흐름을 제어할 수 있도록 제공한다. 이를 기반으로 여러 SSDlet이 협력하여 하나의 작업을 수행할 수 있으며, SSD 내에서 맵-리듀스와 같은 고수준의 작업을 처리하는 것이 가능하다.

YourSQL은 MariaDB의 쿼리 플래너를 수정하여 Biscuit의 ISC 프레임워크를 사용할 수 있도록 하였다. YourSQL은 early filtering을 사용하여 데이터 전송량을 줄일 수 있는 쿼리를 판단하고, 해당 쿼리를 ISC-SSD에게 오프로딩 시킴으로써 전체 시스템의 성능을 개선하였다. YourSQL은 ISC-SSD로 오프로딩할 쿼리를 결정하기 위해 다음과 같은 과정을 거친다. 먼저, 쿼리의 종류, 쿼리를 수행할 테이블의 크기 등을 이용한 휴리스틱 알고리즘을 통해 점수를 산출하고, 높은 점수를 가진 쿼리만을 오프로딩 시킬 후보로 선택한다. 후보로 선정된 쿼리들을 오프로딩 하기 전에 ISC 샘플러를 통해 필터링을 통과한 데이터의 비율을 미리 측정해본다. 마지막으로, 높은 필터링 비율을 가지는 쿼리들을 ISC-SSD로 오프로딩하여 처리한다. YourSQL에서 사용한 ISC-SSD는 패턴 매칭을 수행하는 하드웨어를 추가하여 빠르게 필터링 작업을 수행할 수 있도록 했다.

YourSQL에서는 TPC-H 쿼리들에 대해 Biscuit을 사용할 때의 성능 개선 정도를 측정하였다. 그림 4는 YourSQL과 Biscuit을 사용한 TPC-H의 성능 향상을 보여준다. 총 22개의 쿼리 중 8개의 쿼리가 SSD에게 오프로딩 되었으며, 모든 쿼리에 대해 90%의 성능 개선을 보였다. 가장 많은 성능을 개선한 5개의 쿼리는 평균적으로 일반 SSD 대비 15배 높은 성능을 보였다.

3.6 Morpheus [9], Summarizer [10]

Morpheus와 Summarizer에서는 NVMe 인터페이스

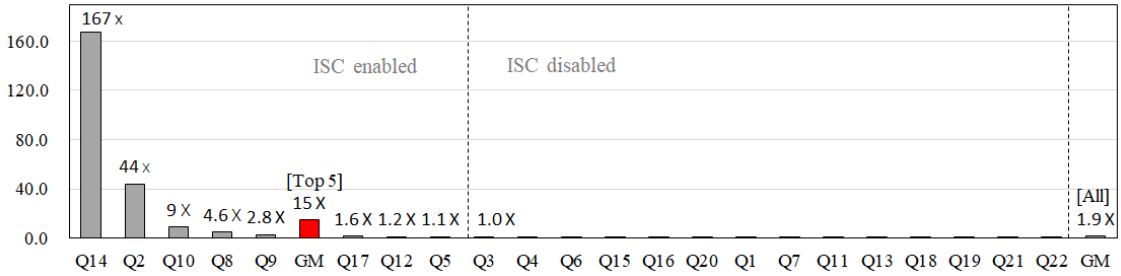

그림 4 YourSQL에서 측정한 TPC-H의 성능

를 확장하여 ISC 태스크를 SSD에게 등록하는 커맨드와 연산된 결과를 가져오는 커맨드를 추가하였다. ISC를 수행하는 읽기를 일반 읽기 커맨드와 구별하기 위해서 NVMe의 읽기 커맨드의 reserve된 공간을 사용하여, ISC가 필요한 읽기/쓰기 명령에 대해 flag를 세팅하고 연산하고자 하는 태스크의 ID를 등록한다. 그림 5는 이들 장치에서 ISC 작업을 수행하는 과정을 보여준다. 먼저 FTL은 호스트가 전달한 읽기 커맨드가 요청한 데이터를 플래시메모리에서 읽어온다. 데이터에 대한 읽기가 완료되면, NVMe 커맨드를 확인하여 ISC 작업이 추가로 필요한지 체크한다. ISC 작업이 필요한 경우, 해당 읽기 커맨드를 미리 할당해 둔 작업 큐에 등록하고, 해당 ISC 태스크가 작업 큐에 등록된 읽기 커맨드의 데이터에 대해 추가 작업을 수행한다. 모든 작업이 끝난 후 호스트는 연산의 결과만을 호스트로 읽어올 수 있다.

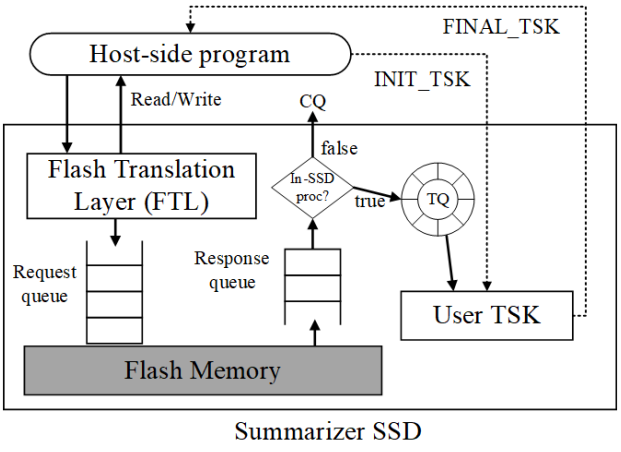

그림 5 Summarizer-SSD의 동작 과정

Morpheus는 특히 오브젝트의 비직렬화 (deserialization) 작업을 SSD로 오프로딩 시키는 방법을 제안하였다. 일부 프로그램들은 오브젝트를 스토리지에 저장할 때 이를 직렬화 (serialization)하여 ASCII 혹은 Unicode 형태의 텍스트(e.g. XML, CSV, JSON, TXT, YAML)로 변환하여 저장한다. 해당 논문에서는 스토리지에서 직렬화된 데이터를 호스트 메모리로 읽어와 오브젝트 형태로 비직렬화 하는데 소요되는 시간이 전체 수행시간의 64%에 해당한다고 지적한다. 또한, Morpheus는 PCIe의 peer-to-peer (P2P) 통신을 활용하여, 비직렬화된 오브젝트를 호스트 메모리를 거치지 않고 GPU로 전달하는 방법도 제안하였다.

Summarizer는 Morpeus를 확장하여 데이터 집중적인 작업을 SSD에게 오프로딩하는 API의 집합을 정의한다. Summarizer에서는 TPC-H 쿼리와 SSJoin 연산을 디바이스로 오프로딩하여 성능을 측정하였다. Summarizer에서는 ISC-SSD를 사용할 경우, 호스트 프로세서에서 모든 작업을 처리하는 것 대비 최대 20%의 성능 개선을 보였으며, SSD에서만 작업을 처리하는 것 대비 600%의 성능 개선을 보였다.

3.7 ExtraV [11]

ExtraV는 그래프 처리를 위한 프로그래밍 모델인 '그래프 가상화'라는 새로운 개념을 도입하여 스토리지의 하드웨어 가속을 효율적으로 활용하였다. ExtraV를 위한 시스템은 호스트 프로세서, 메인 메모리, 하드웨어 가속기인 AFU(Accelerator Function Unit), 그리고 스토리지로 구성된다. 그래프 가상화는 호스트 프로그램에게 그래프 데이터가 메인 메모리에 상주하고 있는 것처럼 보여 준다. 호스트 프로그램이 ExtraV에게 그래프 분석을 요청하면, ExtraV는 가속기 인터페이스의 기능을 활용하여 저장소의 그래프 데이터에 접근하고 하드웨어를 통해 데이터를 해석하여 주어진 시간 내에 호스트의 메인 메모리로 가져온다.

ExtraV에서는 분석할 그래프의 버텍스의 속성(attribute) 정보만을 메인 메모리에 로딩하고, 그래프 데이터는 CSR(Compressed Sparse Row) 형식으로 스토리지에 압축하여 관리한다. ExtraV의 프로그래밍 모델은 다음과 같다. 호스트 프로그램이 그래프를 처리할 때에는 버텍스 또는 엣지의 개별 정보를 요청하는 대신 스트리밍 쿼리를 보내게 된다. ExtraV에서 제공하는 스트리밍 쿼리 API에는 모든 버텍스에서 이웃 리스트를 얻는 쿼리, 버텍스의 리스트에서 다음 이웃된 버텍스을 찾는 쿼리 등이 있다. ExtraV에서는 확장-필터 기법을 사용하여 스토리지 디바이스의 대역폭을 향상시켰다. 먼저, AFU에서 압축된 그래프 데이터의 압축을 해지한 다음에(확장) 호스트 프로그램이 사용하지 않는 데이터들은 제거시켜(필터) 호스트로 전송한다. ExtraV에서는 FPGA를 사용한 프로토타입을 제작하여 여러 그래프 알고리즘을 사용하여 성능 평가를 진행하였다. Pagerank 실험에서 성능이 기존 그래프 기법보다 2-11배 향상한 것을 확인하였다.

3.8 GraFBoost [12]

GraFBoost는 BlueDBM 기반 시스템에서 플래시 스토리지에 랜덤하게 퍼져있는 그래프 데이터 구조를 통합하고 순차화를 수행하는 정렬-리듀스(sort-reduce) 알고리즘을 제안하고 이를 하드웨어로 구현하였다. 예를 들어, 그래프 연산을 수행하면서 버텍스 업데이

트를 즉시 진행하지 않고, 버텍스 업데이트 정보들을 별도의 파일에 기록해 둔다. 모든 연산이 끝난 후, 버텍스 업데이트를 수행할 때, 정렬-리듀스를 활용할 수 있다. 정렬-리듀스는 업데이트를 진행할 키-값들을 여러 블록으로 나누어 블록마다 키 값으로 정렬하여 순차 접근할 수 있도록 하고, 키-값 페어를 통합하여 엔트리 수를 감소시킨다. 논문에서는 이러한 통합연산을 하드웨어로 구현하여 성능 향상을 보여주었다. PageRank와 BFS 알고리즘의 성능평가에서 다른 그래프 알고리즘과 비교하여 2-4배의 우수한 성능을 보여준다.

3.9 산업계의 연구 동향

삼성전자는 2013년에 SmartSSD[2, 3]를 시작으로 2016년에는 biscuit[7]과 YourSQL[8]를 발표하였다. 해당 논문들을 통해 실제 하드웨어에서 빅데이터 분석과 데이터베이스의 쿼리를 SSD로 오프로딩할 때의 성능 개선 정도를 보여주었다. 또한 2018년에는 Xilinx 기반의 SmartSSD[22]를 발표하였는데, 이를 사용하면 FPGA를 통해 ISC 작업을 가속화하는 것이 가능하다.

미국의 NGD[23]에서는 Linux 운영체제를 내장한 고사양 멀티-코어 ISC-SSD를 개발하였다. NGD는 Microsoft Research와 함께 ISC를 통해 스토리지 내의 특정 이미지 파일을 찾는 작업을 수행하는 ISC-SSD를 개발하고 있다. 해당 ISC-SSD를 사용하면, 이미지 파일을 저장하기 위해 사용되는 호스트의 DRAM 사용량을 감소시킬 수 있다. 또한 모든 이미지를 호스트로 전송하지 않아도 되기 때문에, 호스트와 디바이스 간의 데이터 전송을 줄일 수 있으며, 호스트 서버의 자원을 다른 어플리케이션이 사용하는 것이 가능하다. 또한 ScaleFlux[24]를 비롯한 여러 업체가 FPGA 가속기를 포함한 SSD를 개발중이다.

4. 결 론

본 글에서는 In-storage computing 기술의 동작 방식과 최신의 관련 연구 동향에 대해 살펴보았다. 해당 기술은 90년대 후반부터 HDD를 기반으로 연구가 진행되어온 기술이지만, 고사양 멀티-코어를 사용하는 SSD의 등장과 많은 데이터 I/O를 발생시키는 빅데이터 분석에 대한 관심이 높아지면서 다시금 주목받고 있다. ISC-SSD 기술은 호스트의 작업을 SSD에게 오프로딩 시킴으로써 호스트와 병렬 처리를 통해 성능을 개선할 수 있을 뿐 아니라 호스트와 디바이스 간 데이터 전송량을 줄여 시스템의 에너지 소비도 줄일 수 있다. 현재까지의 ISC 기술은 데이터에 대한 간단한 연산 혹은 필터링을 통해 데이터 전송량을 줄이는 것에 초점이 맞춰져 있다. ISC-SSD를 일반적인 환경에서 사용하기 위해서는 응용 개발자에게 보다 친숙한 인터페이스를 제공하는 하는 것과 ISC 작업간의 간섭을 제거하는 것, ISC 작업을 사용함에 있어 보안 취약점의 해소 등이 필요하다. 이와 같은 문제를 해결하는 것이 향후 ISC 연구의 중요한 과제라고 볼 수 있다.

참고문헌

[1] D. Tiwari and et al., “Active flash: Towards energy-efficient, in-situ data analytics on extreme-scale machines”, Proceedings of the 11th USENIX Conference on File and Storage Technologies, pp 119-132, February 2013.

[2] Y. Kang and et al., “Enabling cost-effective data processing with smart SSD”, Proceedings of the IEEE 29th Symposium on Mass Storage Systems and Technologies, pp 1-12, May 2013.

[3] J. Do and et al., “Query processing on smart SSDs: opportunities and challenges”, Proceedings of the International Conference on Management of Data, pp 1221-1230, June 2013.

[4] S. Seshadri and et al., “Willow: a user-programmable SSD” Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation, pp 67-80, Octobuer 2014.

[5] S. W. Jun and et al., “BlueDBM: an appliance for big data analytics”, Proceedings of the 42nd International Symposium on Computer Architecture, pp 1-13, October 2015.

[6] M. King and et al., “Software-driven hardware development”, Proceedings of the 2015 ACMISIGDA International, Symposium on Field-Programmable Gate Arrays, pp. 13-22, 2015

[7] B. Gu and et al., “Biscuit: a framework for near-data processing of big data workloads”, Proceedings of the 43rd International Symposium on Computer Architecture, pp 153-165, June 2016.

[8] I. Jo and et al., “YourSQL: a high-performance database system leveraging in-storage computing”, Proceedings of the VLDB Endowment, pp. 924-935, August 2016.

[9] H. W. Tseng and et al., “Morpheus: creating application objects efficiently for heterogeneous computing”,

Proceedings of the 43rd International Symposium on Computer Architecture, pp 153-165, June 2016.
[10] G. Koo and et al., “Summarizer: trading communication with computing near storage”, Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture. pp 219-231, October 2017.
[11] J. Lee and et al., “ExtraV: Boosting graph processing near storage with a coherent accelerator”, Proceedings of the VLDB Endowment, pp 1706-1717, August 2017.
[12] S. W. Jun and et al., “GraFBoost: Using accelerated flash storage for external graph analytics”, Proceedings of the 45th International Symposium on Computer Architecture, pp 411-424, June 2018.
[13] Hadoop. http://hadoop.apache.org/.
[14] Spark. https://spark.apache.org/.
[15] B. Bengfort and et al., “Data Analytics with Hadoop: An Introduction for Data Scientists” O'Reilly Media, Inc., 2016.
[16] S. Y. W. Su and et al., “CASSM: a cellular system for very large data bases”, Proceedings of the 1st International Conference on Very Large Data Bases, pp. 456-472, September 1975.
[17] E. A. Ozkarahan and et al., “Performance evaluation of a relational associative processor”, ACM Transactions on Database System, vol. 2, no. 2, pp. 175-195, June 1977.
[18] C. S. Lin and et al., “The design of a rotating associative memory for relational database applications”, ACM Transactions on Database System, vol 1, no. 1, pp. 53-65, March 1976.
[19] K. Kannan., “The design of a mass memory for a database computer”, Proceedings of the 5th Annual Symposium on Computer Architecture, pp. 44-51, 1978.
[20] H.-O. Leilich and et al., “A search processor for data base management systems” Proceedings of the Fourth International Conference on Very Large Data Bases, vol. 4, pp. 280-287, 1978.
[21] A. Acharya and et al., “Active disks: Programming model, algorithms and evaluation”, Proceedings of the Eighth International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 81-91, October 1998.
[22] FPGA SSD, https://forums.xilinx.com
[23] NGD, https://www.ngdsystems.com
[24] ScaleFlux, https://www.scaleflux.com

약 력

강 윤 지

2013 성균관대학교 컴퓨터공학과 졸업(학사)
2015 성균관대학교 IT융합학과 졸업(석사)
2017~현재 성균관대학교 IT융합학과 박사과정
관심분야 : 플래시메모리, 파일시스템, 플래시 변환 계층
Email : oso41@skku.edu

한 규 화

2013 성균관대학교 컴퓨터공학과 졸업(학사)
2013~현재 성균관대학교 전자전기컴퓨터공학과 박사과정
관심분야 : 플래시메모리, 파일시스템, 플래시 변환 계층
Email : hgh6877@skku.edu

신 동 군

1994 서울대학교 계산통계학과 졸업(학사)
2000 서울대학교 전산과학과 졸업(석사)
2004 서울대학교 전기컴퓨터공학부 졸업 (박사)
2004~2007 삼성전자기술총괄SW연구소(책임연구원)
2007~현재 성균관대학교 컴퓨터공학과/소프트웨어학과 (교수)
2013~현재 정보과학회 논문지 (편집위원)
관심분야 : 플래시메모리, 파일시스템, 운영체제
Email : dongkun@skku.edu