차 례

1 과제 배경 및 목표

1.1 과제 배경

SSD(Solid State Drive)는 비휘발성 저장 장치로 기존의 HDD(Hard Disk Drive) 보다 빠른 입출력, 저전력, 저소음, 경량성을 장점으로 갖는다. 메모리가 NAND 플래시 메모리로 구성되어있어 데이터에 대한 제자리 쓰기 즉 덮어쓰기가 불가하다. 문제는 이와 같은 덮어쓰기가 불가능한 특성과 쓰기와 지우기 간의 차이에서 발생한다. 쓰기와 읽기의 단위는 페이지이며 이는 보통 4KB이고 지우기는 블록 단위로 주로 512KB 단위이다. 쓰기 위해서는 비어있는 블록이 필요하다. 덮어쓰기가 불가능하므로 업데이트시 기존 데이터를 두고 비어있는 페이지에 업데이트된 데이터를 쓴다. 그러나 블록들이 일정량 이상 가득 차 비어있는 공간이 없다면 반드시 메모리를 지워 블록을 비어있는 상태로 만들어야 한다. 이때 블록 단위 지우기는 해당 영역 내 위치한 삭제될 셀에 더하여 아직 유효한 셀까지 지운다. 이러한 데이터 손실을 방지하기 위해 지우기 이전에 블록 내 유효한 셀을 해당 블록 외부에 위치한 빈 셀로 이동시킨다. 이는 추가적인 쓰기 작업을 동반한다. 이를 수행하는 것이 GC(Garbage Collection)이며 해당 과정에서 발생한 추가적 쓰기에 의한 부하 발생을 WA(Write Amplification)라 한다.

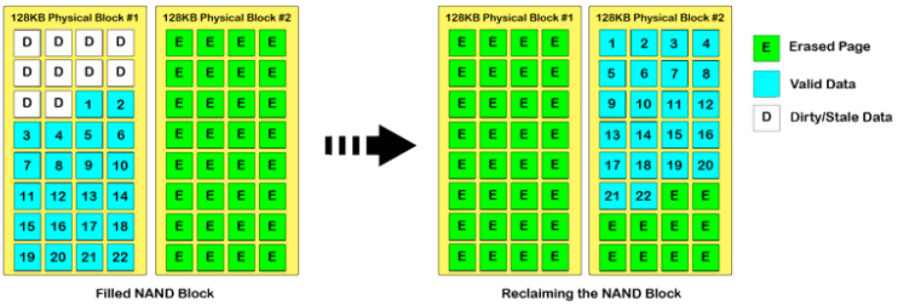


그림 1: 지우기 과정시 발생하는 유효 데이터의 이동

GC 작업을 개선하는 것은 곧 GC를 수행하는 FTL(Flash Translation Layer)을 개선하는 작업이다. 이러한 FTL은 다음과 같은 작업을 수행한다. HDD는 저장 단위를 섹터 기반으로 하고, SSD는 페이지와 블록 기반으로 한다. 많은 운영체제 및 파일구조가 이러한 HDD의 저장 단위를 기반으로 구성되어 있으므로 논리적 주소를 물리적 주소로 변환하는 과정에서 FTL이 이를 SSD 저장 단위에 맞게 변환해준다. 마찬가지로 HDD는 덮어쓰기가 가능하므로 HDD의 쓰기를 SSD에서 지우기와 쓰기로 치환하여 사용하도록 한다. 또한 NAND 플래시 메모리는 일정 횟수 이상의 쓰기 지우기가 진행되면 수명을 다하여 저장 기능을 상실하므로 쓰기 지우기를 일정 구역에 집중되지 않도록 분산시키는 Wear Leveling 작업을 수행한다. 이번 과제에선 이와 같은 FTL의 기능들 중 GC 작업만을 다룰 것이다.

1.2 과제 목표

1) 기계학습을 통한 Hot/Cold 분류

우리는 앞서 설명한 WA를 감소시키기 위하여 입출력 데이터를 Hot/Cold 데이터로 분류하는 방식을 사용하고자 한다. 구분 기준은 쓰기가 이루어진 후 업데이트 간격이 긴 것을 Cold, 짧은 것을 Hot으로 한다. 업데이트가 비교적 빈번하게 일어나는 Hot 데이터들을 같은 블록에 두고 업데이트가 거의 일어나지 않는 Cold 데이터 역시 모아둔다면 지우기가 수행될 때 블록 내에 아직 유효한 데이터의 수가 줄어든다. 이를 통해 같은 블록 내 유효한 데이터를 옮기는 과정에서 발생하는 오버헤드를 감소시킬 수 있다. 이러한 Hot/Cold에 대한 구분 기준은 실험 방향에 따라 추가될 수 있다. 우리는 이리헌 Hot/Cold 분류를 기계학습을 통해 진행할 것이다.

2) GC 알고리즘 구현

기계학습을 통해 얻어낸 모델을 기반으로 GC 알고리즘을 구현하고자 한다. 실험을 통해 구현된 알고리즘의 성능을 평가하기 위해서 실제 범용적으로 사용되는 SSD의 GC 알고리즘과 비교할 필요가 있다. 이러한 비교기준점이 될 GC 알고리즘의 선정은 가능한한 최신 알고리즘을 목표로 하되, 난이도와 진행 상황에 따라 적절한 것을 구현하여 실험을 진행할 것이다.

3) SSD 시뮬레이션을 통한 성능비교

성능 평가를 위하여 위에서 구현한 GC 알고리즘들을 SSD 환경에서 I/O 트레이스들에 대한 처리율을 비교할 필요가 있다. 이를 시뮬레이션 하기 위한 두 가지 방법이 존재한다. 하나는 오픈소스 소프트웨어로 존재하는 SSD 에뮬레이터를 사용하는 방법이다. 이는 실제 SSD와 유사한 환경에서 기능을 테스트할 수 있게 해준다. 다른 하나는 해당 성능 비교를 위한 FTL 계층을 직접 구현하여 실험을 진행하는 것이다. 우리는 모델 학습 및 진척도에 따라 두 가지 방법 중 한 가지를 선택하여 성능 비교 및 평가를 진행할 것이다.

2 기존 연구

머신러닝을 적용하기 이전에도 Hot/Cold 데이터 분류를 통한 성능향상을 도모한 알고리즘들이 존재한다. 그중 몇을 소개하자면 다음과 같다. FIFO(First In First Out) 알고리즘은 LRU(Least Recently Used) 블록부터 GC를 수행하는 방식으로 최근에 접근된 데이터는 재차 접근될 가능성이 높다는 점에서 사용되었다. Greedy 알고리즘의 경우 지우기 수행 대상 블록들

중 유효한 페이지가 가장 적은 블록에 GC를 수행하는 방식이다. 이를 통해 페이지 옮김으로 발생하는 부하가 적게 나타나도록 유도하는 것이다. Windowed 알고리즘은 FIFO와 Greedy를 결합하였다. 즉 LRU 블록들 중 유효한 페이지가 가장 적은 것에 GC를 수행한다. 마지막으로 d-choices가 있다. 이는 2 이상의 자연수 d개 블록을 uniformly random한 선택을 수행한다. 이어 해당 d개 블록 중 유효한 페이지가 가장 적은 블록에 GC를 진행한다.

이러한 알고리즘의 개선을 통한 성능 개선을 도모한 한 연구는[1] Hot/Cold 분류 기준에 접근 시간의 최신성을 추가로 사용한다. 개선된 알고리즘으로 MBF(Multiple Bloom Filter)를 도입하였고 이는 평가 기준점으로 사용한 WDAC (Window-based Direct Address Counting)를 통한 경우 보다 최대 65% 향상된 결과를 보였다.

다른 연구는[2] 워킹 로그 블록과 Cold 로그 블록을 사용한다. 워킹 로그 블록은 지우기 과정에서 존재하는 유효한 데이터를 수용한다. 이때에는 Hot과 Cold 데이터들이 워킹 로그 블록 내에 공존한다. 이후 쓰기 과정을 진행하다 워킹 로그 블록에 가용 저장공간을 모두 소진하면 해당 워킹 로그 블록에 지우기를 수행한다. 이때에도 유효한 데이터로 남아있는 데이터는 두 번의 지우기 과정에서 살아남았으므로 업데이트가 가끔 이루어지는 데이터임을 나타낸다. 그러므로 이들을 Cold 로그 블록으로 옮기고 해당 로그 블록에 위치한 데이터가 Cold 데이터가 된다. 해당 방식은 Greedy 알고리즘을 사용한 결과보다 쓰기 처리율이 최대 18% 향상했다.

이후 기계학습을 이용한 한 연구에선[3] 분류 기준에 단위 시간 내 업데이트 시간 간격을 추가한다. 분류 알고리즘으로는 I/O 트레이스 데이터에 비지도 학습이 적합하다는 판단하에 k-means 알고리즘을 선택했다. 이는 해당 알고리즘은 동적으로 I/O 트레이스의 접근 패턴을 분류하며 기타 분류 알고리즘보다 단순성, 속도, 유연성에 강점이 있다는 점에서 기인한다. 단점은 상대적으로 훈련 시간이 오래 걸린다는 점이고 이는 하드웨어 입출력 환경에 치명적이다. 이에 해당 연구에선 SSD 내부에 머신러닝 하드웨어 유닛을 도입할 것을 제안했다. 또한 분류에 있어 이전 방식에 더하여 Warm 블록이라는 개념을 도입한다. 업데이트 빈도수와 평균 업데이트 간격이 가장 작은 것을 Hot, 빈도수가 적고 업데이트 간격이 긴 것을 Cold, 마지막으로 그 중간 지역을 Warm 구역으로 분류했다. k-means의 k 값을 6으로 설정한 해당 실험에서 기존 SSD 하드웨어보다 머신러닝 하드웨어 유닛을 도입한 SSD에서 26.3% - 57.7%의 성능 개선을 얻었다.

위 연구에서 나아가 분류 기준을 추가한 연구[4] 역시 존재한다. 분류기준으로는 쓰기의 빈도, 평균 업데이트 요청 시간, 마지막으로 두 번째 기준의 평균을 구하는 데 사용된 모든 시간 간격의 표준편차를 사용한다. 분류 모델은 위에서 서술한 연구와 동일하게 k-means 분류 알고리즘을 사용했다. 또한 분류 시 큰 특징값이 군집 분류에 큰 영향을 가하므로 이를 감소시키기 위해 데이터의 표준화를 수행했다. 이에 더하여 각 특성의 중요도를 분류하기 위해 각각 빈도에

1, 평균 업데이트 요청 시간에 0.8, 업데이트 요청 시간 표준편차에 0.7의 가중치를 부여했다. 이러한 방법들을 통해 각 분류 기준이 균형 있게 작용하도록 하였다. 또한 적절한 k값을 구하기 위해 엘보우 기법을 사용하였고 이를 통해 k값을 6으로 설정했다. 이 연구 역시 위에서 서술한 연구와 같이 영역을 Hot, Warm, Cold로 나누었다. 이때 새로 도입한 업데이트 시간 간격 표준편차가 Warm 영역을 세세히 분류하게 해준다. 표준편차가 작은 경우 업데이트 빈도가 일정하며, 큰 경우 그렇지 않음을 보인다. 고로 Warm 영역 내에서도 표준편차가 큰 경우를 Cold 데이터로 분류했다. 해당 연구에서는 성능 비교는 수행하지 않았다.

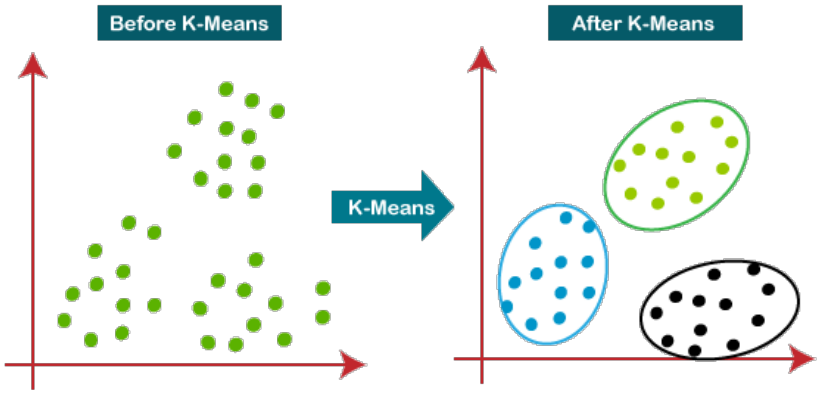


그림 2: K=3일 때 K-means의 분류

빈도를 기준으로 한 Hot/Cold 분류 이외에도 메모리의 DT(Death-Time) 예측을 기준으로 분류를 진행한 연구[5] 역시 존재한다. 해당 연구는 시계열 데이터 예측에 사용되는 CNN (Convolutional Neural Network)의 변종인 TCN(Temporal Convolutional Network)을 통해 메모리 DT 예측을 수행했다. 과적합 방지를 위해 Drop Out 계층을 사용하고, 500 뉴런으로 구성된 2개의 TCN 히든 레이어를 사용했으며 출력 계층으로 soft max를 사용했다. 결과는 AutoStream 기법과 비교했을 때 최대 14%의 성능 개선을 보였다.

3 제약사항과 해결방안

우리는 동일한 데이터에 대해 기존 알고리즘을 사용한 SSD 입출력 시간 소요와 머신러닝을 진행한 모델의 입출력 시간을 기록하여 비교해야 한다. 이를 위해서 SSD 에뮬레이터 혹은 직접 작성한 FTL 시뮬레이터를 사용할 것이다. 그 이유는 머신러닝을 통해 학습한 모델을 FTL에 적용하여 SSD를 물리적으로 구축하고, 실제 환경에서 실험하는 것은 해당 프로젝트의 영역을 벗어난 부분임에 기인한다. 그러므로 가상환경에서 실험을 진행하고 이를 분석하고자 한다.

4 시스템 설계

우리가 개선하고자 하는 바는 SSD의 지우기 과정 시 발생하는 WA이다. 이를 위해 Hot/Cold 분류를 기계학습을 통해 수행할 것이다. 우리는 앞서 이전 연구들에서 사용한 분류기준을 토대로 실험을 진행할 것이며, 실험 중 정확도 향상을 위해 새로운 기준을 도입할 수 있다.

4.1 기계학습

matplotlib로 학습 시각화를 진행 할 것이며, 기계학습 알고리즘에는 sklearn을 사용하고, 신경망 알고리즘을 위해 tensorflow를 사용할 것이다. 기계학습을 위하여 RNN (Recurrent Neural Network)과 LSTM(Long Short-Term Memory)을 이용한 지도학습을 시도할 것이다. RNN은 내부 상태를 사용하여 시계열 데이터를 처리에 사용된다. 하지만 RNN은 지점에 따라 손실함수 기울기가 빠르게 감소하여 장기적인 관계성을 갖는 데이터를 학습하는 데에 어려움이 있다.

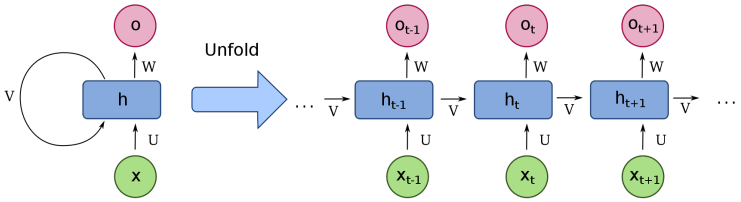


그림 3: Vanila RNN 구조

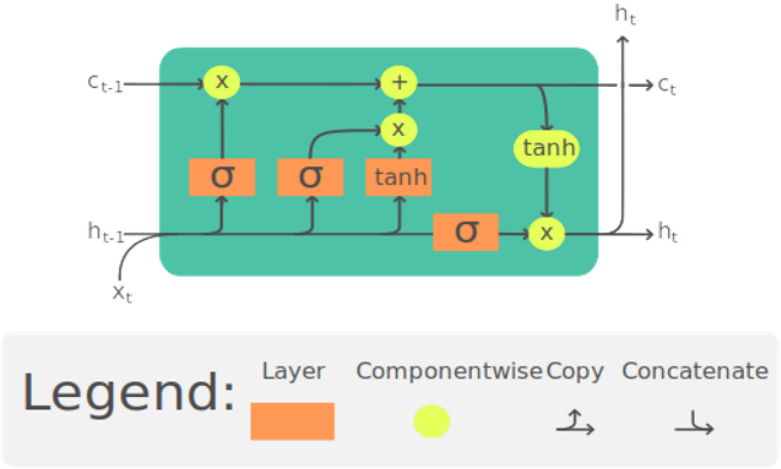


그림 4: LSTM 셀 내부구조

LSTM은 정보를 더 긴 시간 동안 유지하는 메모리 셀을 갖아 더 장기적인 관계성을 학습할 수 있다. 더하여 내부에 사용되는 게이트들이 입력된 정보가 메모리에 저장될지, 언제 출력될지, 언제 망각할지 결정한다. 이를 고려할 때 학습 데이터가 시간과 접근 빈도 등에 영향을 받으므로 LSTM이 더 나은 선택이 될 것이라 예상한다. 우리는 위 모델들의 분류 정확도를 평가하며 필요에 따라 새로운 모델 및 비지도 학습 모델 역시 학습을 시도할 것이다.

4.2 학습 데이터

학습 데이터로는 SNIA(Storage Networking Industry Association)과 UMass Trace Repository에서 제공하는 I/O 트레이스를 사용할 것이다. 이들 데이터는 제공되는 데이터마다 조금씩 상이하나 구성은 다음 양식을 예시로 들 수 있다.

0,20939840,8192,W,0.554041

첫 번째 필드는 I/O를 사용하는 애플리케이션 지정자이다. 두 번째는 논리적 블록 주소를 나타니며 세 번째는 레코드 사이즈를 표기한다. 네 번째는 OP코드로써 R(Read), W(Write)로 구성된다. 다섯 번째는 타임 스탬프로서, I/O 트레이스 시작점부터 경과 시간을 기록한 타임 스탬프이다. 이를 비교함으로 레코드 간 시간 차이를 확인할 수 있다.

4.3 SSD 시뮬레이션

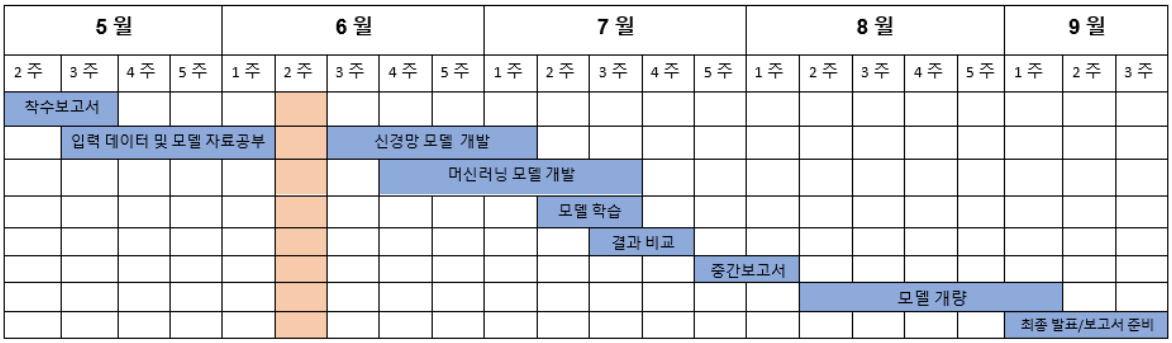
위 과정들을 통해 얻은 모델을 평가하기 위하여 SSD 에뮬레이터를 사용할 것이다. 이때 실제 SSD 제품에 사용되는 GC 알고리즘을 에뮬레이터에 적용할 것이며, 필요에 따라 직접 GC 알고리즘 시뮬레이터를 작성하여 실험할 것이다. GC 알고리즘은 가장 최근에 개발되고 널리 사용되는 것을 구현하고자 하나 프로젝트 진행 상황과 난이도를 고려하여 선정할 예정이며 Wear-leveling 작업은 해당 실험에서는 고려하지 않을 것이다. 이를 통해 평가 기준 알고리즘과 기계학습을 통해 얻어낸 모델의 성능을 평가하고 비교하여 개선된 성능을 얻어내고자 한다.

사용 가능한 SSD 에뮬레이터로는 FEMU, VSSIM, SIMPLESSD 등이 있다. 3가지 에뮬레이터 모두 GC 알고리즘을 변경 가능하며, 오픈소스 소프트웨어로 별도의 비용 없이 사용이 가능하다. FEMU는 C언어로 작성되어 있다. 또한 DRAM을 기반으로 작동하기 때문에 실험상 대용량 저장장치에 접근함에 따라 발생하는 추가적인 딜레이가 발생하지 않게 된다. VSSIM 역시 C언어로 이루어져 있다. 해당 에뮬레이터는 가상머신으로부터 실시간 워크로드를 받아 사용하여 실제 SSD 시스템 환경과 동일한 환경에서 실험을 진행할 수 있게 해준다. SIMPLESSD는 C++로 구성된다. 3가지 에뮬레이터 모두 풀 시스템 SSD 에뮬레이터이고 높은 성능을 보여주기 때문에 사용환경, 접근성, 구성 프로그래밍 언어 등을 고려하여 선정할 필요가 있다.

5 개발 일정 및 업무 분담

5.1 개발 일정

개발 일정은 아래와 같다



5.2 업무 분담

업무 분담은 아래와 같다

최성찬 - GC 알고리즘 구현 및 에뮬레이터 GC 알고리즘 수정

Ariunbold Odgerel - 신경망 모델 개발 및 훈련 데이터 처리

Ganchuluun Narantsatsralt - 머신러닝 모델 개발 및 성능 평가

공통 - 머신러닝 모델 훈련 및 자료조사

참고 문헌

[1] Dongchul Park et al. Hot data identification for flash-based storage systems using multiple bloom filters. In Mass Storage Systems and Technologies (MSST), 2011 IEEE 27th Symposium on, pages 1–11. IEEE, 2011.

[2] Ilhoon Shin. Hot/cold clustering for page mapping in nand flash memory. IEEE Transactions on Consumer Electronics, 57(4), 2011.

[3] B Li, HAML-SSD: a hardware accelerated hotness-aware machine learning based SSD management. In: 38th IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2019, p. 8942140.

[4] 정윤형, 정혜미, 김재호 (2021). SSD 쓰기 증폭을 줄이기 위한 머신러닝 기반 정교한 Hotness 분류 방안. 2021년 한국소프트웨어종합학술대회 논문집

[5] Chandranil Chakraborttii et al. Reducing Write Amplification in Flash by Death-time Prediction of Logical Block Addresses. SYSTOR '21, 2021