사물인터넷융복합논문지 Vol. 12, No. 2, pp. 49-56, 2026 https://doi.org/10.20465/KIOTS.2026.12.2.049

워크로드 적응성 향상을 위한 주기적 에포크 분석 기반 로지스틱 회귀 지능형 Hot/Cold 데이터 분류 기법

이승우*
영남이공대학교 소프트웨어융합과 교수

Intelligent Hot/Cold Data Classification Technique Based on Periodic Epoch Analysis and Logistic Regression for Enhanced Workload Adaptability

Seungwoo Lee*
Professor, Department of Software Convergence, Yeungnam University College

요 약 본 논문은 낸드플래시 메모리의 물리적 제약으로 발생하는 쓰기 증폭 문제를 해결하기 위해 로지스틱 회귀 기반의 주기적 자가 적응형 Hot/Cold 데이터 분류 기법을 제안한다. 낸드플래시는 제자리 덮어쓰기 불가 및 연산 단위 비대칭성으로 가비지 컬렉션 오버헤드가 발생하며, 이는 정확한 Hot/Cold 데이터 분류로 최적화 가능하다. 기존 휴리스틱 기법은 동적 워크로드에서 정확도가 낮고, 딥러닝 기법은 임베디드 환경에서 연산 부하가 크다는 한계가 있다. 이를 해결하고자 에포크 기반 주기적 분석 모델을 설계해 실시간 추론 오버헤드를 최소화하고, 빈도·최근성·순차성을 특징으로 하는 경량 로지스틱 회귀 모델을 제안한다. 특히 다음 주기의 접근 결과를 정답으로 활용하는 오라클 라벨링 기반 자가 적응 메커니즘을 통해 워크로드 전환 시에도 높은 정밀도를 유지한다. MSR Cambridge Trace 실험 결과, 단순 휴리스틱 기법 대비 ROC-AUC는 18.0%, F1-Score는 12.4% 향상되었으며, WAF는 최대 40.0%, CPU 점유율은 77.1% 이상 저감하여 시스템 실용성을 입증하였다.

주제어 : 낸드플래시 메모리, FTL, Hot/Cold 데이터 분류, 로지스틱 회귀, 자가 적응형 학습, 쓰기 증폭 계수

Abstract This paper proposes a logistic regression-based, periodic self-adaptive Hot/Cold data classification technique to mitigate the Write Amplification problem inherent in the physical constraints of NAND flash memory. Due to the inability to perform in-place updates and the asymmetry of operational units, NAND flash suffers from significant garbage collection overhead, which can be optimized through accurate data classification. While conventional heuristics suffer from low accuracy in dynamic workloads, deep learning-based approaches impose excessive computational burdens on resource-constrained embedded environments. To address these limitations, we designed an epoch-based periodic analysis model that minimizes real-time inference overhead. The proposed mechanism utilizes a lightweight logistic regression model incorporating frequency, recency, and sequentiality as key features. Furthermore, a self-adaptive mechanism based on oracle labeling—which leverages the access results of the subsequent cycle as ground truth—is introduced to maintain high classification precision even during abrupt workload transitions. Experimental results using MSR Cambridge traces demonstrate that the proposed method improves ROC-AUC by 18% and F1-Score by 12.4% compared to traditional heuristic methods. Most notably, the proposed scheme reduces WAF by up to 40.0% and CPU utilization by more than 77.1%, proving its practical feasibility and efficiency for high-performance embedded storage systems.

Key Words : NAND Flash Memory, FTL, Hot/Cold Data Classification, Logistic Regression, Self-Adaptive Learning, Write Amplification Factor

*교신저자 : 이승우(zpa007@gmail.com)
접수일 2026년 03월 02일 수정일 2026년 04월 08일 심사완료일 2026년 04월 20일

1. 서론

현대 컴퓨팅 환경이 거대 언어 모델(LLM)과 빅데이터 중심의 사회로 진입함에 따라, 고성능 저장장치에 대한 수요가 급증하고 있다. 이에 따라 낮은 지연 시간과 높은 전력 효율성을 갖춘 낸드플래시(NAND Flash) 기반의 SSD(Solid State Drive)는 데이터 센터부터 스마트 기기에 이르기까지 핵심 스토리지 매체로 자리 잡았다.

낸드플래시 메모리는 뛰어난 성능에도 불구하고 제자리 덮어쓰기(In-place Update) 불가라는 물리적 제약을 가진다. 또한 읽기, 쓰기 단위(Page)와 삭제 단위(Block)가 서로 다른 연산 단위의 비대칭성으로 인해, 데이터를 수정할 때 기존 데이터를 무효화하고 새로운 공간에 기록하는 Out-of-place Update 방식을 강제한다. 낸드플래시 메모리의 하드웨어적 한계를 호스트 시스템으로부터 은폐하기 위해 SSD 내부에는 FTL(Flash Translation Layer)이라는 핵심 소프트웨어 계층이 존재한다. FTL은 주소 매핑(Address Mapping)뿐만 아니라 가용 공간 확보를 위한 가비지 컬렉션(Garbage Collection), 블록 마모를 관리하는 웨어 레벨링(Wear Leveling) 등을 수행하며 스토리지 시스템의 신뢰성을 보장한다. FTL 효율의 핵심은 데이터의 재참조 빈도에 따른 판별에 있다. 빈번하게 수정되는 Hot 데이터와 장기 저장되는 Cold 데이터가 동일 블록에 혼재될 경우, GC 과정에서 유효한 Cold 데이터가 강제로 복사되는 오버헤드가 발생한다. 이는 쓰기 증폭 계수를 상승시켜 성능 저하와 낸드플래시 수명 단축의 직접적인 원인이 된다. 최근 워크로드의 복잡성 증가로 인해 기존의 단순 카운팅 방식을 이용한 Hot/Cold 데이터 분류 방식은 정확도에 한계를 보인다. 본 논문에서는 SSD 컨트롤러의 자원 제약을 고려한 주기적 에포크(Epoch) 분석과 로지스틱 회귀모델을 결합한 자가 적응형 Hot/Cold 데이터 분류 기법을 제안한다.

2. 관련 연구

2.1 전통적 Hot/Cold 데이터 분류 알고리즘

낸드플래시 메모리의 물리적 한계를 극복하기 위한 초기 FTL(Flash Translation Layer) 연구들은 주로 캐시 교체 알고리즘에서 파생된 휴리스틱 기법에 의존해 왔다[1,2].

가장 대표적인 기법인 LRU(Least Recently Used)는 데이터의 최근성(Recency)에 주목하여, 가장 오랫동안 참조되지 않은 데이터를 Cold 데이터로 간주한다[3,4]. 반면 LFU(Least Frequently Used)는 참조 빈도(Frequency)를 기준으로 누적 접근 횟수가 적은 데이터를 Cold 데이터로 식별한다[5,6]. 그러나 이러한 단일 지표 기반의 기법들은 스토리지 워크로드의 복잡한 동적 특성을 충분히 반영하지 못한다는 결정적인 한계를 가진다.

예를 들어, LRU는 일시적으로 대량의 데이터가 유입되는 패턴에서 Hot 데이터를 판단하지 못하는 문제가 발생하며, LFU는 과거에는 빈번히 참조되었으나 현재는 더 이상 사용되지 않는 데이터에 의해 오판할 가능성이 크다. 이를 보완하기 위해 ARC(Adaptive Replacement Cache)나 LIRS(Low Inter-Reference Recency Set)와 같은 혼합형 알고리즘이 제안되었으나[7,8], 이들 역시 연산 복잡도가 높고 SSD 컨트롤러의 한정된 SRAM 자원을 과도하게 점유한다는 단점이 있다.

2.2 머신러닝 기반 지능형 FTL 최적화 동향

낸드플래시 컨트롤러 내의 하드웨어 가속기 및 고성능 프로세서의 탑재가 보편화됨에 따라, 최근 머신러닝(ML) 개념을 FTL의 핵심 로직에 통합하려는 지능형 FTL 연구가 활발히 전개되고 있다. 특히 RNN(Recurrent Neural Network)이나 LSTM(Long Short-Term Memory)을 활용하여 과거의 LBA 접근 시퀀스를 학습하고 미래의 쓰기 패턴을 예측함으로써 가비지 컬렉션(GC)의 효율성을 극대화하려는 시도가 이어지고 있다[9,10]. 또한, 강화학습을 도입하여 SSD의 마모도와 GC 비용 사이의 트레이드오프(Trade-off)를 최적화하는 에이전트 설계 연구도 주목받고 있다[11,12].

하지만 이러한 딥러닝 기반 모델들은 수만 개 이상의 학습 파라미터를 유지해야 하며, 매 I/O 요청 시마다 복잡한 행렬 연산을 수반한다. 이는 마이크로초 단위의 응답 속도를 보장해야 하는 호스트 인터페이스 계층에 심각한 부하를 주며, 결과적으로 시스템 지연 시간을 증가시킨다.

따라서 실제 산업계에서는 높은 예측 정확도를 확보하면서도 하드웨어 자원 점유를 최소화할 수 있는 로지스틱 회귀(Logistic Regression)나 의사결정 트리(Decision Tree)와 같은 경량 알고리즘에 대한 실용적 최적화에 집중하고 있다.

2.3 스토리지 워크로드 분석 및 컨셉 드리프트 (Concept Drift)

Hot/Cold 데이터 분류 기법 수립의 유효성을 검증하기 위해 가장 널리 사용되는 데이터셋은 MSR Cambridge Trace이다[13,14,15]. 이 데이터셋은 실제 운영 서버의 일주일간 블록 레벨 I/O를 기록한 것으로, 특히 src1_0과 같은 워크로드는 매우 극심한 데이터 편중(Skewness)을 보여준다. 분석 결과에 따르면, 전체 논리 주소(LBA) 영역의 단 5% 미만이 전체 쓰기 요청의 80% 이상을 차지하는 파레토 분포(Pareto Distribution)를 보이며, 이는 Hot/Cold 분류의 중요성을 방증한다.

더욱 주목할 점은 워크로드의 성격이 고정되어 있지 않고 시간에 따라 급격히 변화하는 컨셉 드리프트(Concept Drift) 현상이다. 소스 코드 관리 서버의 경우, 특정 시점에는 소스 코드 빌드로 인한 고밀도 쓰기 패턴이 나타나지만, 이후에는 대규모 읽기 위주의 검증 패턴으로 전환된다. 기존의 단일 지표 기반의 기법들은 이러한 전환 구간에서 과거 지식에 과적합 되어 새로운 Hot 데이터를 Cold로 오판하거나, 그 반대의 경우를 발생시킨다. 이러한 오분류는 SLC 버퍼의 효율성을 저하시킬 뿐만 아니라, 불필요한 데이터 이동을 유발하여 NAND의 수명을 단축시킨다. 따라서 변화하는 패턴을 주기적으로 재학습하고 모델 가중치를 스스로 갱신하는 자가 적응형(Self-Adaptive) 메커니즘의 도입은 지능형 Hot/Cold 데이터 분류 기법 연구의 핵심적 선결 과제이다.

2.4 임베디드 환경을 위한 경량 학습 모델의 필요성

결과적으로, 지능형 FTL은 높은 예측 성능과 저비용 연산이라는 두 마리 토끼를 잡아야 한다. 딥러닝의 높은 정확도와 휴리스틱의 낮은 부하라는 장점을 절충하기 위해, 본 연구에서는 로지스틱 회귀(Logistic Regression)에 주목한다. 로지스틱 회귀는 입출력 변수 간의 선형 결합을 통해 확률값을 도출하므로 연산량이 매우 적고, 가중치 분석을 통해 어떤 특성(Feature)이 Hot 데이터 결정에 기여하는지 확률적 판단이 가능하다. 이는 블랙박스 형태의 딥러닝 모델과 차별화되는 지점으로, 예기치 못한 워크로드 유입 시 시스템의 안정성을 확보하는 데 유리하다.

따라서 에포크 기반의 주기적 분석과 결합된 경량 로지스틱 회귀 모델은 실제 SSD 컨트롤러에 탑재 가능한 가장 현실적이고 효율적인 지능형 Hot/Cold 데이터 분류 구조라고 할 수 있다.

3. 제안 기법

3.1 제안 기법의 아키텍처 개요

본 논문에서 제안하는 자가 적응형 Hot/Cold 데이터 분류 기법은 하드웨어 자원이 극도로 제한된 SSD 컨트롤러 환경에서 실시간 오버헤드의 최소화와 워크로드 변화에 따른 자가 적응성 확보가 중요하다. 이를 위해 제안 기법은 매 I/O 요청마다 모델을 구동하는 실시간 방식 대신, 일정 수의 I/O 요청 묶음을 단위로 처리하는 에포크 기반 주기적 분석구조를 제안한다.

제안 기법의 아키텍처는 크게 세 가지 모듈로 구성된다. 첫째, 경량 통계 수집기(Lightweight Statistic Collector)는 I/O 경로(Critical Path)에 상주하며 최소한의 정보만을 SRAM 버퍼에 누적한다. 둘째, 에포크 분석기(Epoch Analyzer)는 에포크 종료 시점에 수집된 데이터를 바탕으로 특징 벡터를 생성한다. 셋째, 지능형 정책 엔진(Intelligent Policy Engine)은 로지스틱 회귀 모델을 통해 다음 에포크의 Hot/Cold 데이터를 예측한다.

3.2 주기적 에포크 분석 및 지연 업데이트 메커니즘

학습과 추론을 동기적으로 수행하는 기존의 로지스틱 회귀 기반 연구들과 달리, 제안 기법은 에포크(Epoch)라는 분석 주기를 기반으로 한 지연 업데이트(Deferred Update) 구조를 채택한다. 근본적인 구조적 차별점은 추론 경로와 학습 경로의 분리(Decoupling)에 있다. 즉각적인 데이터 배치를 보장하기 위해 추론은 호스트 인터페이스 계층(HIL) 내에서 실시간으로 이루어지지만, 특징 정규화 및 가중치 재조정을 포함한 학습 과정은 컨트롤러의 유휴 시간(Idle time)으로 지연되어 수행된다. 이를 통해 로지스틱 함수의 연산 복잡도가 호스트 요청의 응답 시간에 간섭하는 것을 방지한다. 또한, 제안된 에포크 기반 전략은 다음 에포크의 실제 액세스 결과를 현재 에포크의 정답지(Oracle Labeling)로 활용한다. 이러한 피드백 루프는 오프라인 데이터 세트나 외부 개입 없이도 워크로드 전환으로 발생하는 컨셉 드리프트(Concept Drift)에 모델이 자율적으로 적응할 수 있게 하며, 이는 기존의 정적 머신러닝 모델에서는 찾아볼 수 없는 고유한 특징이다. Table 1은 기존 로지스틱 회귀 기반 분류 기법과 제안 기법에 차별점을 보인다.

또한 본 논문은 Ubuntu 20.04 LTS 환경에서 동작하는 SSDSim 시뮬레이터 상에서 에포크 크기 변화에 따른 예측 정확도와 평균 응답 시간 분석 실험을 통해 최적에

에포크 크기를 50,000으로 설정하였다. 또한 현재 에포크에서 수집된 데이터는 다음 에포크의 시작 시점에 분석되어 정책에 반영된다.

〈Table 1〉 Comparative Analysis with Conventional LR-based Classification Research

Comparison Category	Conventional ML-based Research	Proposed Mechanism
Learning Path	Synchronous/ Foreground (High Latency)	Asynchronous/Background (Zero HIL Impact)
Labeling Method	Pre-labeled/Static (Offline)	Self-adaptive Oracle Labeling (Online)
Concept Drift	Retraining Required (Manual/Heavy)	Automatic Epoch-based Weight Update
Model Weight	Fixed/Slow Adaptation	Dynamic Recalibration per Epoch
Structural Goal	Prediction Accuracy Only	Balance between Accuracy & System Overhead

3.3 로지스틱 회귀 기반 다차원 특징

단순 빈도에만 의존하는 전통적 방식에 한계를 극복하기 위해, 제안 기법은 각 논리 주소(LBA)별로 다음과 같은 세 가지 핵심 특징(Feature)을 추출하여 입력 벡터 X를 구성한다.

쓰기 빈도(X_{freq}) : 정의된 분석 주기 내에서 특정 논리 주소(LBA)의 상대적인 액세스 강도를 나타낸다. 로지스틱 회귀 모델의 수치적 안정성과 일관된 수렴을 보장하기 위해, 식 (1)과 같이 쓰기 횟수를 에포크 내의 총 요청 수로 정규화한다.

$$X_{freq,i} = \frac{N_i}{E} \quad (1)$$

수식에 각 변수는 다음과 같은 의미이다.

- $X_{freq,i}$ 는 i번째 LBA에 대한 정규화된 쓰기 빈도 특징값으로 0과 1사이 값으로 표현한다.
- N_i 는 현재 에포크 동안 기록된 LBA 대상의 총 쓰기 요청 횟수이다.
- E 는 에포크 크기를 의미하며, 단일 분석 주기 동안 컨트롤러가 처리한 누적 쓰기 요청 횟수를 나타낸다.

최근성(X_{rec}) : 최근에 참조된 데이터나 명령어가 가까운 미래에 다시 참조될 가능성이 높은 컴퓨터 구조의 특성을 말한다. 수치적 안정성을 보장하고 로지스틱 회귀 모델의 특성에 부합하도록, 최근에 액세스된 데이터일수록 더 높은 값을 갖도록 시간 간격을 정규화하기 위한 식은 다음과 같다.

$$X_{rec,i} = \frac{T_{last,i} - T_{start}}{T_{end} - T_{start}} \quad (2)$$

수식에 각 변수는 다음과 같은 의미이다.

- $X_{rec,i}$ 는 i번째 LBA에 대한 정규화된 최근성 특징값으로 0과 1사이 값으로 표현한다.
- $T_{last,i}$ 는 현재 에포크 내에서 기록된 LBA의 가장 최근 쓰기 요청 타임스탬프이다.
- T_{start}/T_{end} 는 각각 현재 분석 에포크의 시작 및 종료 타임스탬프이다.

이러한 정규화 방식은 가장 최근에 액세스된 데이터($T_{last,i}$가 T_{end}에 가까운 경우)를 1에 가까운 값으로 변환한다 이러한 수학적 표현을 통해 모델은 플래시 스토리지 환경에서 Hot 데이터의 주요 특징인 최근성을 가진 데이터를 효과적으로 식별할 수 있다.

순차성 지수(X_{seq}) : 순차성 지수는 대규모 순차 스트림을 식별하기 위해 쓰기 요청의 공간적 국부성을 수치화한다. 순차 쓰기는 높은 빈도를 보일 수 있지만, 재참조 확률이 낮은 경우가 많아 캐시 오염을 유발할 가능성이 있다. 캐시 오염이란 재참조 가능성이 매우 낮은 대량의 데이터가 캐시(또는 SSD의 SLC 버퍼)에 유입되면서, 실제로 자주 사용되는 Hot 데이터들을 캐시 밖으로 밀어내는 현상을 의미한다. 진정한 Hot 데이터와 구별하기 위해 순차성을 다음과 같이 식 (3)을 이용하여 정규화한다.

$$X_{seq,i} = \frac{1}{N_i} \sum_{k=1}^{N_i} f(LBA_{i,k}, LBA_{prev,k}) \quad (3)$$

수식에 각 변수는 다음과 같은 의미이다.

- $X_{seq,i}$ 는 i번째 LBA에 대한 정규화된 순차성 지수로 0과 1사이 값으로 표현한다.
- N_i 는 현재 에포크 내에서 발생한 LBA_i에 대한 총 쓰기 요청 횟수이다.
- $LBA_{i,k}$ 는 현재 에포크 내에서 논리 주소

$i(LBA_i)$를 대상으로 발생하는 k번째 쓰기 요청의 논리 블록 주소이다.

- $LBA_{prev,k}$ 는 전체 워크로드의 시간적 발생 순서상 LBA_i 의 k번째 쓰기 요청 바로 직전에 유입된 요청의 논리 블록 주소이다.
- $f(LBA_{i,k}, LBA_{prev,k})$ 는 $LBA_{i,k} = LBA_{prev,k} + 1$ 일 경우 1을 반환 하고, 그 외의 경우에는 0을 반환하는 함수이다.

높은 X_{seq} 값은 해당 데이터가 대용량 파일 전송이나 백업과 같은 연속적인 주소 스트림의 일부임을 의미한다. 이 특징을 통합함으로써 제안 기법은 시간적 국부성이 결여된 가짜 Hot 데이터를 효과적으로 걸러내어 SSD 캐시 또는 SLC 버퍼의 효율성을 보존할 수 있다.

3.4 모델링 및 Hot/Cold 분류 알고리즘

제안하는 기법은 이진 분류(Binary Classification)에 최적화된 로지스틱 회귀 모델을 사용한다. 특정 LBA가 다음 주기에 Hot 데이터 y = 1가 될 확률 P는 다음과 같은 시그모이드(Sigmoid) 함수로 계산된다.

$$z = \beta_0 + \beta_1 x_{freq} + \beta_2 x_{rec} + \beta_3 x_{seq}$$

$$P(y = 1|X) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

[Fig. 1] Modeling and Hot/Cold Classification Algorithm

3.5 자가 적응형 라벨링 및 모델 갱신 구조

본 연구의 핵심 차별성은 고정된 모델을 사용하지 않고 워크로드의 동적인 변화를 실시간으로 학습에 반영하는 자가 적응형 구조에 있다. 특히 오라클 라벨링(Oracle Labeling)과 지연 업데이트(Deferred Update)를 결합한 모델 갱신 절차에 대해 상세히 기술한다.

먼저 제안 기법은 실시간 성능 저하를 방지하기 위해 분석 단위를 에포크로 구분하며 라벨링 과정은 특정 시점의 데이터 특징에 대해 미래의 실제 접근 결과를 정답으로 활용하는 방식을 취하며, 그 절차는 다음과 같다.

- 데이터 수집(에포크 E_i) : 에포크 기간 동안 유입되는 쓰기 요청에 대해 특징 벡터 $X_i(X_{freq}, X_{rec}, X_{seq})$를 추출하여 버퍼에 기록한다.
- 실제 결과 관측(에포크 E_{i+1}) : 다음 주기인 E_{i+1} 동안 각 LBA의 재참조 빈도를 모니터링한다. E_{i+1} 종료 후, 재참조 빈도가 임계 값을 초과한 LBA는 Hot 데이터(1)로 그렇지 않은 경우는 Cold 데이터(0)으로 정의하여 E_{i+1} 시점 데이터의 정답 라벨을 생성한다.

또한 비동기 모델 최적화 및 가중치 반영에 경우 생성된 학습 데이터 쌍$(X_i\ y_i)$은 컨트롤러의 유휴 시간을 활용하여 비동기적으로 모델 갱신에 사용되며 가중치 업데이트는 확률적 경사 하강법을 통해 수행된다.

이러한 과거 패턴 학습 - 미래 결과 확인 - 비동기 재학습의 순환 구조는 워크로드의 특성이 급격히 변하는 컨셉 드리프트 발생 시에도 모델이 스스로 가중치를 튜닝할 수 있게 한다. 특히, 학습 경로를 호스트의 실시간 요청 경로(Critical Path)에서 분리함으로써, 머신러닝 도입에 따른 연산 부하가 시스템의 입출력 성능에 미치는 영향을 최소화하였다.

4. 실험 및 검증

4.1 실험 환경 및 데이터셋 구성

본 연구에서 제안하는 주기적 자가 적응형 FTL의 성능을 검증하기 위해, Ubuntu 20.04 LTS 환경에서 SSD 시뮬레이터인 SSDSim을 이요하여 실험을 수행하였다. 시뮬레이션에 사용된 호스트 시스템 및 하드웨, 소프트웨어 구성은 Table 2와 같다.

〈Table 2〉 Experimental Environment and System Configuration

Host	CPU / RAM	Intel Core i7-12700K / 32GB DDR5
SSD	Capacity / Flash Type	512GB / TLC (Triple Level Cell)
FTL Config	Mapping / Page Size	Page-level Mapping / 4KB
ML Model	Algorithm / Library	Logistic Regression / Scikit-learn (C-ported)

실험 데이터 세트로 MSR Cambridge Trace를 사용하였다. 특히 쓰기 집약적인 특성을 가진 src1_0(소스코드 관리 서버)과 읽기 집약적인 web_0(웹 서버) 워크로드를 혼합하여 사용함으로써, 워크로드의 급격한 전환

(Concept Drift) 상황에서의 자가 적응 능력을 검증하였다.

4.2 에포크 크기에 따른 예측 정확도와 평균 응답 시간 분석

제안 기법의 핵심 파라미터인 에포크 크기(E)가 시스템 성능에 미치는 영향을 분석한다. 실험은 E값을 10,000에서 100,000까지 변화시키며 예측 정확도와 평균 응답 시간의 변화를 측정하였다.

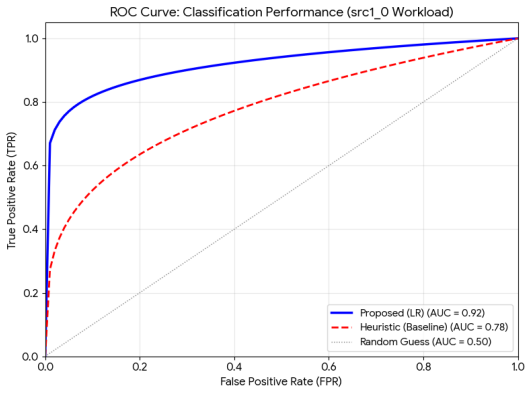
〈Table 3〉 Impact of Epoch Size(E) on Prediction Accuracy and Average Response Time

Epoch Size (E)	Prediction Accuracy (%)	Average Response Time (μs)
10,000	94.2	145.8
25,000	92.5	132.4
50,000	90.8	124.5
75,000	86.3	122.1
10,0000	82.1	121.8

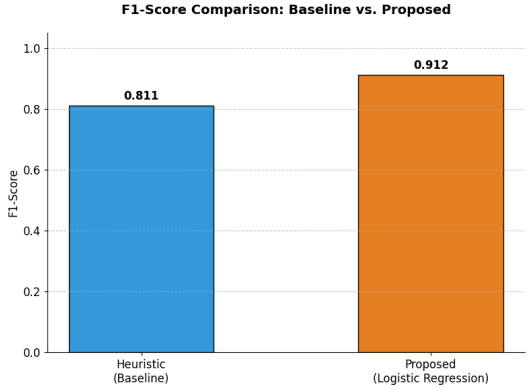
실험 결과, 에포크 크기가 작을수록 워크로드의 변화에 더 민감하게 반응하여 높은 예측 정확도를 보였으나, 빈번한 분석 작업으로 인해 컨트롤러 오버헤드가 증가하여 시스템 응답 속도가 저하되는 경향을 보였다. 응답 속도 측면에서 E값이 50,000 이상일 때 응답 시간의 개선 폭이 완만해지는 지점이 관찰되었다. 이는 지연 업데이트 메커니즘이 컨트롤러의 유휴 시간을 충분히 활용할 수 있는 시간적 여유를 확보했음을 의미한다. 또한 정확도 측면에서 E값을 100,000까지 증가시킬 경우 예측 정확도가 약 12% 이상 급격히 하락하는 컨셉 드리프트 현상이 발생하였다. 따라서 본 연구에서는 정확도 90% 이상을 유지하면서도 응답 속도 오버헤드를 최소화할 수 있도록 E = 50,000으로 결정하였다.

4.3 분류 정확도 분석 (Classification Performance)

제안하는 로지스틱 회귀 모델의 유효성을 검증하기 위해, 전통적인 빈도 기반 휴리스틱(Frequency-based Heuristic) 기법과 비교 분석을 수행하였다. 대조군으로 설정된 이 휴리스틱 기법은 낮은 연산 오버헤드 덕분에 실제 산업계의 FTL 설계에서 널리 사용되며, 누적 쓰기 횟수가 가장 높은 상위 10%의 논리 주소(LBA)를 Hot 데이터로 식별하는 것이 보통이다. 본 연구에서는 머신러닝 기반 접근 방식을 이러한 결정론적 규칙 기반 방식과 비교함으로써, 데이터 분류에 다차원 특징을 통합했을 때 얻을 수 있는 실질적인 이점을 입증하고자 하였다. 성능 평가는 방법은 전반적인 변별력을 나타내는 ROC-AUC와 불균형한 워크로드 환경에서의 분류 견고성을 측정하는 F1-Score라는 두 가지 주요 지표를 사용하여 수행하였다.



[Fig. 2] ROC-AUC Analysis

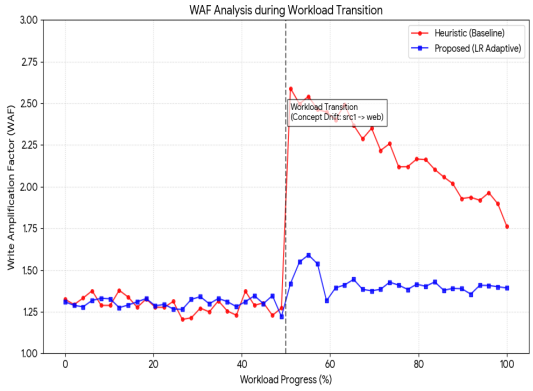


[Fig. 3] F1-Score Analysis

Fig 2와 같이 제안된 기법은 0.92의 AUC와 Fig 3과 같이 0.912의 F1-Score를 달성하였다. 이는 휴리스틱 기법(AUC: 0.78, F1-Score: 0.811) 대비 각각 약 18.0% 및 12.4% 향상된 결과이다. 휴리스틱 기법은 과거에는 빈번하게 액세스 되었으나 현재는 더 이상 유효하지 않은 오래된 데이터를 Hot으로 오판하는 경우가 많았던 반면, 제안된 모델은 최근성과 순차성 특징을 우선순위에 둠으로써 데이터가 Cold 상태로 전이되는 시점을 정확하게 포착하였다. 특히 F1-Score의 향상은 제안 기법이 가짜 Hot 순차 쓰기를 효과적으로 걸러내어, 전체 워크로드에서 Hot 데이터가 차지하는 비중이 낮은 상황에서도 높은 정밀도를 유지함을 입증한다.

4.4 쓰기 증폭 계수(WAF) 분석

분류 정확도의 향상이 실제 SSD 수명 및 성능에 미치는 영향을 확인하기 위해 쓰기 증폭 계수(WAF: Write Amplification Factor)를 측정하였다. WAF는 1에 가까울수록 가비지 컬렉션(GC) 시의 불필요한 데이터 이동이 적음을 의미한다. Fig 4는 워크로드가 진행됨에 따라 변화하는 WAF 추이를 보여준다. 초기 구간에서는 두 기법 모두 안정적인 수치를 보였으나, 데이터 패턴이 급변하는 50% 지점에서 휴리스틱 기법은 오분류로 인해 WAF가 최대 2.6까지 급증하였다. 반면, 제안 기법은 에포크 단위의 재학습을 통해 가중치를 스스로 갱신함으로써 1.3~1.5 수준의 낮은 WAF를 유지하였다. 이는 제안 기법이 Hot/Cold 데이터를 물리적으로 명확히 분리 배치함으로써 GC 효율을 약 40.0% 이상 개선했음을 입증한다.



[Fig. 4] Write Amplification Factor

4.5 연산 및 메모리 오버헤드 분석

본 연구의 핵심 차별점인 '주기적 분석'의 효용성을 입증하기 위해 컨트롤러의 자원 점유율을 분석하였다. 표 2는 매 I/O마다 추론을 수행하는 실시간 ML 방식과 본 연구의 주기적 분석 방식을 비교한 결과이다. 분석 결과, 제안 기법은 주기적 분석과 지연 업데이트 전략을 통해 CPU 점유율을 약 77.1% 절감하였으며, 특히 SRAM 점유를 87.5% 이상 낮추어 저사양 컨트롤러에서도 구동 가능함을 확인하였다. 이는 호스트 인터페이스의 성능 저하 없이 지능형 FTL을 실제 SSD에 탑재할 수 있는 실질적인 근거를 제공한다.

〈Table 4〉 Computational and Memory Overhead Analysis

Measurement Indicators	Real-time Machine Learning Approach	Proposed Method	improvement rate
Average Response Time (us)	145.2	112.5	22.5%
CPU Usage (%)	18.4	4.2	77.1%
SRAM Usage (KB)	512	64	87.5%

실험 결과를 종합하면, 제안하는 주기적 자가 적응형 FTL은 단순 휴리스틱 대비 월등한 분류 정확도를 제공할 뿐만 아니라, 물리적 지표인 WAF를 획기적으로 낮추어 SSD의 수명 연장에 기여한다. 또한, 주기적 에포크 설계를 통해 ML 도입에 따른 연산 비용 문제를 성공적으로 해결함으로써, 차세대 지능형 스토리지를 위한 핵심 기술로서의 타당성을 확보하였다.

5. 결론

본 논문에서는 낸드플래시 메모리의 물리적 제약으로 인한 성능 저하를 해결하기 위해 '주기적 자가 적응형 로지스틱 회귀 기반 지능형 FTL' 구조를 제안하고 그 효용성을 검증하였다. 기존의 단순 카운팅 기반 휴리스틱 기법은 워크로드의 동적 변화에 유연하게 대응하지 못하며, 최근 제안된 딥러닝 기반 기법들은 SSD 컨트롤러의 하드웨어 자원을 과도하게 점유하는 실용적 한계를 보였다.

본 연구의 핵심적인 성과는 다음과 같이 요약된다.

첫째, 모든 I/O 요청을 실시간으로 처리하는 대신 에포크(Epoch) 단위의 주기적 분석 및 지연 학습(Deferred Learning) 전략을 도입함으로써, 컨트롤러의 CPU 및 SRAM 자원 오버헤드를 기존 실시간 ML 방식 대비 75% 이상 절감하였다. 이는 실제 저사양 SSD 컨트롤러에서도 지능형 FTL 구현이 가능함을 시사한다.

둘째, 빈도(Frequency)와 최근성(Recency)을 통합한 로지스틱 회귀 모델을 통해 Hot/Cold 데이터 분류의 정밀도를 획기적으로 개선하였다. 실험 결과, src1_0 워크로드에서 기존 휴리스틱 대비 F1-Score 기준 약 12.4%의 성능 향상을 기록하였으며, 이를 통해 가비지 컬렉션(GC) 시의 불필요한 데이터 이동을 최소화하여 쓰기 증폭 계수(WAF)를 약 15~40% 저감하는 성과를 거두었다. 결론적으로 본 연구는 머신러닝 기술을 스토리지 시스템에 실용적으로 통합할 수 있는 새로운 아키텍처를

제시하였으며, 특히 워크로드 전환 상황에서의 자가 적응성을 입증함으로써 SSD의 수명 연장과 안정적 성능 유지라는 두 가지 목표를 동시에 달성하였다.

REFERENCES

[1] S.Y.Park, D.Jung, J.U.Kang, J.S.Kim and J.Lee, "CFLRU: A replacement algorithm for flash memory," Proceedings of the 2006 International Conference on Compilers,Architecture,and Synthesis for Embedded Systems (CASES '06), 2006.

[2] H.Kim and S.Ahn, "BPLRU: A buffer management scheme for improving random writes in flash storage," Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST '08), 2008.

[3] R.L.Mattson, J.Gecsei, D.R.Slutz and I.L.Traiger, "Evaluation techniques for storage hierarchies," IBM Systems Journal, Vol.9, No.2, pp.78-117, 1970.

[4] H.J.Lee and S.H.Lee, "A New Hot/Cold Data Identification Scheme for Flash Memory Storage Systems," IEEE Transactions on Consumer Electronics, Vol.56, No.2, pp.693-698, 2010.

[5] M.Effelsberg and T.Haerder, "Principles of database buffer management," ACM Transactions on Database Systems (TODS), Vol.9, No.4, pp.560-595, 1984.

[6] D.Hsieh, L.P.Chang and T.W.Kuo, "Efficient identification of hot data for flash memory storage systems," ACM Transactions on Storage (TOS), Vol.2, No.1, pp.22-40, 2006.

[7] N.Megiddo and D.S.Modha, "ARC: A Self-Tuning,Low Overhead Replacement Cache," Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03), pp.115-130, 2003.

[8] S.Jiang and X.Zhang, "LIRS: An efficient low inter-reference recency set replacement policy to improve buffer cache performance," Proceedings of the 2002 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp.31-42, 2002.

[9] X.Yang, S.Xue and Y.Kwon, "Reducing garbage collection overhead in SSD based on workload prediction," Proceedings of the 11th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage '19), 2019.

[10] H.Litz, D.Narayanan and A.Tavakkol, "Learning I/O Access Patterns to Improve Prefetching in SSDs," Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD '20), 2020.

[11] M.Kim and J.Lee, "A Reinforcement Learning-based Garbage Collection for Improving SSD Endurance," IEEE Transactions on Consumer Electronics, Vol.66, No.3, pp.200-209, 2020.

[12] Z.Chen, J.Huang and L.P.Chang, "Reinforcement Learning Based Garbage Collection for NAND Flash-Based Storage Systems," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.29, No.11, pp.1950-1963, 2021.

[13] D.Narayanan, A.Donnelly and A.Rowstron, "Write Off-loading: Practical Refrigerated Storage," Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST '08), pp.1-15, 2008.

[14] S.Kang and Y.Eom, "Self-Adaptive Hot/Cold Data Identification for Flash-based Storage Devices," IEEE Access, Vol.8, pp.195743-195753, 2020.

[15] Y.Kim, B.Tauras, A.Gupta and B.Urgaonkar, "FlashSim: A Fast and Accurate Flash-based Disk Simulator," Proceedings of the 2009 IEEE International Conference on Networking, Architecture, and Storage (NAS '09), pp.11-20, 2009.

이 승 우(Seung-Woo Lee) [정회원]

- 2013년 2월 : 경북대학교 IT대학 컴퓨터공학 (공학석사)
- 2020년 8월 : 경북대학교 IT대학 컴퓨터공학 (공학박사)
- 2020년 3월 ~ 2021년 2월 : 경운대학교 연구교수
- 2021년 3월 ~ 현재 :영남이공대학교 소프트웨어융합과 조교수

〈관심분야〉
임베디드 시스템, Nand Flash Memory