(12) 특허협력조약에 의하여 공개된 국제출원

(19) 세계지식재산권기구
국제사무국

(43) 국제공개일
2009 년 7 월 16 일 (16.07.2009)

PCT

(10) 국제공개번호
WO 2009/088194 A2

(51) 국제특허분류:
G06F 12/00 (2006.01)

(21) 국제출원번호: PCT/KR2009/000034

(22) 국제출원일: 2009 년 1 월 5 일 (05.01.2009)

(25) 출원언어: 한국어

(26) 공개언어: 한국어

(30) 우선권정보:
10-2008-0002114 2008 년 1 월 8 일 (08.01.2008) KR

(71) 출원인 (US 을(를) 제외한 모든 지정국에 대하여): **한국과학기술원 (KOREA ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY)** [KR/KR]; 대전 유성구 구성동 373-1, 305-701 Daejeon (KR).

(72) 발명자; 겸

(75) 발명자/출원인 (US 에 한하여): **박규호 (PARK, Kyu-Ho)** [KR/KR]; 대전 유성구 구성동 한국과학기술원 6-3208, 305-701 Daejeon (KR). **백승훈 (BAEK, Sung-Hoon)** [KR/KR]; 대전 유성구 탑립동 577 번지, 305-510 Daejeon (KR).

(74) 대리인: **김성호 (KIM, Sungho)**; 서울 강남구 역삼동 635-7 삼정빌딩 7 층, 135-908 Seoul (KR).

(81) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 국내 권리의 보호를 위하여): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

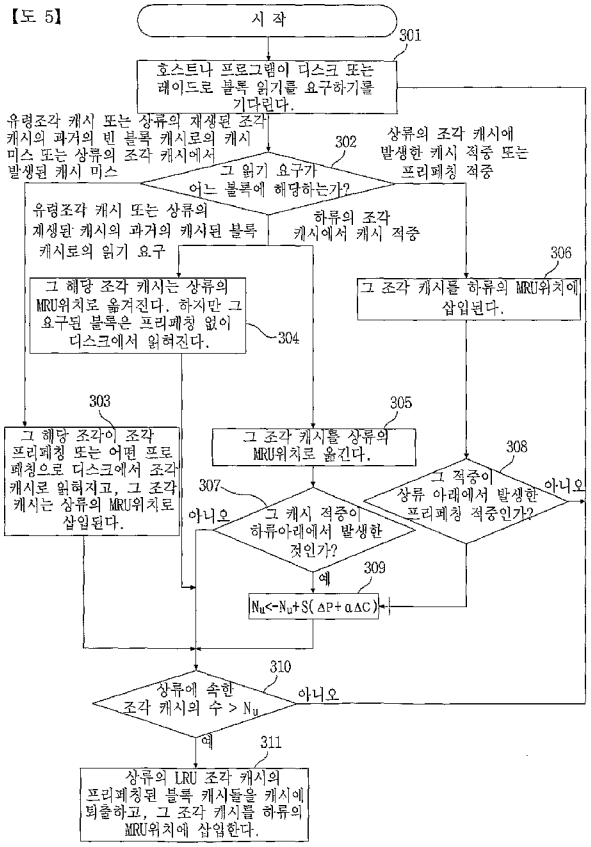
(84) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 역내 권리의 보호를 위하여): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), 유라시아 (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), 유럽 (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ,

[다음 쪽 계속]

(54) **Title**: PREPAGING DATA MANAGEMENT METHOD FOR A COMPUTER STORAGE DEVICE

(54) **발명의 명칭**: 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법

【도 5】



AA ... Start
BB ... Cache miss which occurred in an upstream fragment cache or cache miss of a previous empty block cache of an upstream replayed fragment cache or phantom fragment cache
CC ... Prepaging hit or cache hit which occurred in an upstream fragment cache
DD ... Read request to previous cached block cache of upstream replayed cache or phantom fragment cache
EE ... Cache hit in a downstream fragment cache
FF ... No
GG ... Yes
301 ... Host or program waits for a request for block reading from disk or RAID.
302 ... Which block does the read request correspond to?
303 ... The corresponding fragment is read as fragment cache from disk by fragment prepaging or some form of propaging, and the fragment cache is inserted into an upstream MRU position.
304 ... The corresponding fragment cache is moved to an upstream MRU position. However, the requested block is read from disk without prepaging.
305 ... The fragment cache is moved to an upstream MRU position
306 ... The fragment cache is inserted into a downstream MRU position.
307 ... Did the cache hit occur below downstream?
308 ... Is the hit a prepaging hit which occurred below upstream?
310 ... Number of fragment caches belonging upstream > Nu
311 ... Withdrawal from cache of prepaged block caches of the upstream LRU fragment cache, and insertion of the fragment cache into a downstream MRU position.

[다음 쪽 계속]

WO 2009/088194 A2

CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

공개:

— 국제조사보고서 없이 공개하며 보고서 접수 후 이를 별도 공개함 (규칙 48.2(g))

(57) Abstract: The present invention relates to a prepaging data management method for a computer storage device. In the prepaging data management method for a computer storage device according to the present invention, the full cache is managed in fragment cache units, and the fragment caches are divided upstream and downstream. The said prepaging data management method comprises: a first process involving control in such a way that on the upstream there are the said prepaged block cache and the said cached block cache while on the downstream side there is only the said cached block cache; a second process in which the number (Nu) of fragment caches which there can be on the said upstream side is updated using the differential value of the total of the prepaging hit rate and the cache hit rate; and a third process in which, when the number of the fragment cache comprised upstream is greater than the number (Nu) of the updated fragment caches mentioned hereinabove, the upstream LRU (Least Recently Used) fragment cache is moved downstream in accordance with LRU policy, and the prepaged block cache of the said fragment cache is eliminated from the said full cache.

(57) 요약서: 본 발명은 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법에 관한 것이다. 본 발명에 따른 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법은, 전체 캐시를 조각 캐시 단위로 관리하고, 조각 캐시들이 상류와 하류로 분할되며, 상기 상류는 상기 프리페칭된 블록 캐시와 상기 캐싱된 블록 캐시를 가지고, 상기 하류는 상기 캐싱된 블록 캐시만을 가지도록 제어하는 제 1 과정, 상기 상류가 가질 수 있는 조각 캐시들의 수(Nu)를 프리페칭 적중률과 캐시 적중률 합의 미분값을 이용하여 갱신하는 제 2 과정, 상기 상류에 포함된 조각 캐시의 수가 상기 갱신된 조각 캐시들의 수(Nu)보다 큰 경우, LRU(Least Recently Used) 정책에 따라 상류의 LRU 조각 캐시를 하류로 이동하되, 상기 조각 캐시의 프리페칭된 블록 캐시를 상기 전체 캐시에서 제거시키는 제 3 과정을 포함한다.

명세서

컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법

기술분야

[1] 본 발명은 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법에 관한 것으로, 특히 메모리에서의 프리페칭된 데이터를 관리하는 방법에 관한 것이다.

배경기술

[2] 프리페칭 기술은 1960년대를 거슬러 올라가 프로세서가 캐시(Cache) 라인 단위로 다수의 워드들을 미리 읽는 것으로 시작되었다. 그리고, 현재 프로세서에서의 프리페칭 기술이 많이 개발되었으며, 이러한 프리페칭 기술은 디스크에서 메모리로 미리 읽는 디스크 프리페칭에도 다소 적용되었다. 프로세서에서의 프리페칭과 디스크 저장장치의 다른 특징으로 디스크만을 위한 프리페칭 기술이 많이 연구되었다. 여기서, 디스크 프리페칭은 디스크의 읽기 비용을 낮추어 디스크 성능을 높이거나 연산동작과 디스크 입출력을 서로 겹치게 하여 시스템의 전체적인 성능을 높인다.

[3] 이러한 프리페칭된 캐시들을 관리하는 방법은 많이 제안되어왔다. 예를 들어, 캐싱된 데이터들의 빈도과 최근 접근시간들을 고려하는 ARC, LRFU, MQ, 2Q, LRU-2 등의 캐시 관리 기법들이 있고, 데이터의 순차성과 임의성으로 구분하는 SARC, DULO 등의 캐시 관리 기법들이 있다.

발명의 상세한 설명

기술적 과제

[4] 그러나, 이러한 종래 캐시 관리기법은 오로지 오프라인 프리페칭 기법들뿐이다. 이 기법들은 미래의 디스크 접근들을 모두 알고 있다고 가정하고 있기 때문에, 실제 시스템에서는 상용화되기 어려운 문제점이 있다.

[5] 또한, 종래 제안된 캐시 관리 방법들은, 잘못된 예측으로 프리페칭되고 사용되지 않는 데이터는 캐시를 오염시키기 때문에 프리페칭의 정확성만을 중요시하게 된다. 따라서, 이러한 정확성만을 중요시하게 되므로, 디스크의 읽기 서비스 시간이 늘어나게 되어 효율성이 떨어지는 문제점이 발생한다.

[6] 따라서, 본 발명의 목적은 프리페칭을 고려하여 효율성을 높일 수 있는 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법을 제안함에 있다.

[7] 또한, 본 발명의 다른 목적은 범용적으로 사용할 수 있는 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법을 제안함에 있다.

[8] 또한, 본 발명의 다른 목적은 프리페칭된 캐시들을 온라인으로 관리하는 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법을 제안함에 있다.

기술적 해결방법

[9] 상술한 바를 달성하기 위한 본 발명은, 컴퓨터 저장 장치에서의 프리페칭된 데이터를 관리 하는 방법에 있어서, 전체 캐시를 조각 캐시 단위로 관리하고,

조각 캐시들이 상류와 하류로 분할되며, 상기 상류는 상기 프리페칭된 블록 캐시와 상기 캐싱된 블록 캐시를 가지고, 상기 하류는 상기 캐싱된 블록 캐시만을 가지도록 제어하는 제 1과정, 상기 상류가 가질 수 있는 조각 캐시들의 수(*Nu*)를 프리페칭 적중률과 캐시 적중률 합의 미분값을 이용하여 갱신하는 제 2과정, 상기 상류에 포함된 조각 캐시의 수가 상기 갱신된 조각 캐시들의 수*(Nu)* 보다 큰 경우, LRU(Least Recently Used) 정책에 따라 상류의 LRU 조각 캐시를 하류로 이동하되, 상기 조각 캐시의 프리페칭된 블록 캐시를 상기 전체 캐시에서 제거시키는 제 3과정을 포함한다.

[10] 상기 Nu는 다음의 <수학식 1>에 의한 방법으로 되먹임시키며,

[11] 수학식 1

$$Nu \leftarrow Nu + S(\Delta P - \partial \Delta C)$$

[12] 여기서, 상기 ΔP는 임의 시간 동안 상기 상류 아래에서 발생한 상기 프리페칭 적중(히트) 수, 상기 ΔC는 같은 시간 동안 상기 전체 아래에서 발생한 상기 캐시 적중 수, 상기 α는 (상류 아래의 프리페칭된 블록 캐시 및 캐싱된 블록 캐시의 수) / (하류 아래의 캐싱된 블록 캐시의 수), 상기 S는 상수임을 특징으로 한다.

[13] 상기 상류 및 상기 하류의 조각 캐시들은 각각 LRU(Least Recently Used) 정책으로 관리됨을 특징으로 한다.

[14] 상기 제 1과정은, 상기 상류의 임의의 조각 캐시에 포함된 프리페칭된 블록 캐시 또는 캐싱된 블록 캐시로의 읽기 요구가 발생하면, 상기 조각 캐시를 상류의 MRU(Most recently used) 위치로 이동하는 과정을 더 포함한다.

[15] 상기 제 1과정은, 상기 하류의 조각 캐시에 포함된 캐싱된 블록 캐시로의 읽기 요구가 발생하면, 상기 조각 캐시를 상기 하류의 MRU(Most recently used) 위치로 이동하는 과정을 더 포함하고, 상기 읽기 요구가 캐시 미스를 발생시키면 상기 읽기 요구된 데이터는 조각 프리페칭 없이 디스크에서 읽혀진다.

[16] 상기 제 1과정은, 유령 조각 캐시 또는 상기 상류의 재생된 조각 캐시의 상기 과거의 캐싱된 블록 캐시로의 읽기 요구가 캐시 미스, 또는 상류의 조각 캐시로의 캐시 미스를 발생시키면, 해당 조각 캐시는 상류의 MRU(Most recently used) 위치로 이동하는 과정을 더 포함하고, 상기 읽기 요구된 블록은 조각 프리페칭 (Strip Prefetching: SP) 없이 디스크로부터 읽혀진다.

[17] 상기 제 1과정은, 유령 조각 캐시 또는 상기 상류의 재생된 조각 캐시의 과거의 빈 블록 캐시에 해당하는 읽기 요구가 발생되면, 해당되는 조각 캐시가 조각 프리페칭 (Strip Prefetching: SP) 또는 소정의 프리페칭 기법으로 디스크로부터 읽혀지고, 상기 상류의 MRU(Most recently used) 위치에 삽입되는 과정을 더 포함한다.

[18] 상기 제 1과정은, 상기 상류 또는 상기 하류의 조각 캐시, 또는 유령 조각 캐시에 속하지 않은 블록으로의 읽기 요구가 발생하면, 해당 블록을 위한 조각 캐시를 할당하고, 해당 조각 캐시를 상류의 MRU(Most recently used) 위치에 삽입하는

과정을 더 포함하며, 상기 조각 캐시는 조각 프리페칭(Strip Prefetching: SP) 또는 소정의 프리페칭 기법으로 읽혀진다.

[19] 상술한 바를 달성하기 위한 본 발명은, 컴퓨터 저장 장치에서의 프리페칭된 데이터를 관리 하는 방법에 있어서, 전체 캐시를 상기 블록 캐시 단위로 관리되고, 블록 캐시들이 상류와 하류로 분할되며, 상기 상류는 상기 프리페칭된 블록 캐시와 상기 캐싱된 블록 캐시를 가지고, 상기 하류는 상기 캐싱된 블록 캐시만을 가지도록 제어하는 제 1과정, 상기 상류가 가질 수 있는 상기 블록 캐시들의 수(*Nu*)를 프리페칭 적중률과 캐시 적중률 합의 미분값을 이용하여 갱신하는 제 2과정, 상기 상류에 포함된 블록 캐시의 수가 상기 갱신된 블록 캐시들의 수(*Nu*)보다 큰 경우, LRU(Least Recently Used) 정책에 따라 상류의 LRU 블록 캐시를 하류로 이동하되, 상기 블록 캐시의 프리페칭된 블록 캐시를 캐시에서 제거시키는 제 3과정을 포함한다.

[20] 상기 블록 캐시들의 수(*Nu*)는 다음의 <수학식 2>에 의한 방법으로 되먹임시키며,

[21] 수학식 2

$$Nu \leftarrow Nu + S(\Delta P - \partial \Delta C)$$

[22] 여기서, 상기 ΔP는 어떤 시간 동안 상기 상류 아래에서 발생한 상기 프리페칭 적중의 수, ΔC는 같은 시간 동안 상기 전체 아래에서 발생한 상기 캐시 적중의 수, α는 1 + (상류 아래의 프리페칭된 블록 캐시들의 수)/(전체 아래의 블록 캐시들의 수)이며, S 는 상수임을 특징으로 한다.

[23] 상기 상류 및 상기 하류의 블록 캐시들은 각각 LRU(Least Recently Used) 정책으로 관리됨을 특징으로 한다.

[24] 상기 제 1과정은, 상기 상류의 임의의 블랙 캐시에 포함된 프리페칭된 블록 캐시 또는 캐싱된 블록 캐시로의 읽기 요구가 발생하면, 상기 블록 캐시를 상류의 MRU(Most recently used) 위치로 이동하는 과정을 더 포함한다.

[25] 상기 제 1과정은, 상기 하류의 블랙 캐시에 포함된 캐싱된 블록 캐시로의 읽기 요구가 발생하면, 상기 블록 캐시를 하류의 MRU(Most recently used) 위치로 이동하는 과정을 더 포함한다.

[26] 상기 읽기 요구가 캐시 미스를 발생시키면, 해당되는 블록을 위한 블록 캐시를 새로 할당하고, 상류의 MRU(Most recently used) 위치에 삽입하는 과정을 더 포함한다.

유리한 효과

[27] 본 발명의 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법에 따르면, 프리페칭 적중률과 캐시 적중률의 두 변경(邊境) 사용률들을 동일하도록 프리페칭 블록 캐시들의 수와 캐싱된 블록 캐시들의 수를 최적으로 할당하고, 적응적 방법으로 적당한 시간에 사용되지 않는 프리페칭된 블록들을 메모리에서 퇴출시킴으로써, 종래 기법보다 높은 프리페칭 및 캐시 적중률로써

데이터 저장장치의 성능을 향상시키는 효과가 있다.

[28] 그리고, 본 발명에 따르면, 낮은 디스크 읽기 비용으로도 고성능의 이득을 얻을 수 있어, 비용을 절감할 수 있는 효과가 있다.

[29] 또한, 본 발명에 따르면, 어떠한 가정 없이 개발된 범용적 캐시 관리 방법으로, 별도의 하드웨어 변경없이 기존의 소형에서 대형 컴퓨터 또는 저장장치까지 간단히 적용할 수 있는 효과가 있다.

[30] 이상과 같은 본 발명에 대한 해결하고자 하는 과제, 과제 해결 수단, 효과 외의 구체적인 사항들은 다음에 기재할 실시예 및 도면들에 포함되어 있다. 본 발명의 이점 및 특징, 그리고 그것들을 달성하는 방법은 첨부되는 도면과 함께 상세하게 후술되어 있는 실시예들을 참조하면 명확해질 것이다. 명세서 전체에 걸쳐 동일 참조 부호는 동일 구성 요소를 지칭한다.

도면의 간단한 설명

[31] 도 1은 본 발명의 바람직한 실시예에 따라 4개의 디스크로 구성된 디스크 어레이에서 스트립 및 스트라이프의 구조의 일례가 도시된 도면

[32] 도 2는 본 발명의 바람직한 실시예에 따라 조각 캐시로 관리되는 캐시를 위한 적응적 캐시 솎아내기 기법을 설명하는 도면

[33] 도 3은 본 발명의 조각 캐시로 관리되는 캐시에서의 상류에서의 조각 캐시의 수를 계산하는 과정을 나타낸 도면

[34] 도 4는 본 발명의 바람직한 실시예에 따라 블록 캐시단위로 관리되는 캐시를 위한 적응적 캐시 솎아내기 기법을 나타내는 도면

[35] 도 5는 본 발명의 바람직한 실시예에 따라 조각 캐시로 관리되는 캐시를 위한 적응적 캐시 솎아내기 기법을 설명하는 흐름도

[36] 도 6은 본 발명의 바람직한 실시예에 따라 블록 캐시로 관리되는 캐시를 위한 적응적 캐시 솎아내기 기법을 설명하는 흐름도

[37] <도면의 주요 부분에 관한 부호의 설명>

[38] 101: 상류

[39] 102: 상류 아래

[40] 103: 하류

[41] 104: 전체 아래

[42] 120: 호스트가 요구했었던 블록 캐시(캐싱된 블록 캐시)

[43] 121: 프리페칭되었지만 호스트가 요구 안한 블록 캐시(프리페칭된 블록 캐시)

발명의 실시를 위한 최선의 형태

[44] 이하 본 발명의 실시예에 대하여 첨부한 도면을 참조하여 상세하게 설명하기로 한다. 다만, 첨부된 도면은 본 발명의 내용을 보다 쉽게 개시하기 위하여 설명되는 것일 뿐, 본 발명의 범위가 첨부된 도면의 범위로 한정되는 것이 아님은 이 기술분야의 통상의 지식을 가진 자라면 용이하게 알 수 있을 것이다.

[45] 우선, 본 발명은 정확성이 조금 떨어지는 프리페칭 방법으로도 고성능의

이득을 얻을 수 있는 프리페칭 데이터 관리 방법을 제안한다. 여기서, 정확성 문제는 프리페칭 적중률과 캐시 적중률의 합이 어떤 프리페칭의 프리페칭 적중률과 프리페칭 없는 캐시 적중률 중 어느 것보다도 높으면, 낮은 프리페칭 정확성의 문제는 해결될 수 있다.

[46] 이에 따라 본 발명에서는 프리페칭 적중률과 캐시 적중률의 합이 최대화하도록 함으로써, 고성능의 이득을 얻을 수 있게 된다. 이를 위해 본 발명은 프리페칭 적중률의 변경 사용률과 캐시 적중률의 변경 사용률을 매 순간순간 같도록 만들면서, 적절한 순간에 프리페칭되었지만 사용되지 않는 캐시들을 메모리에서 퇴출시키는 캐시 관리 방법, 즉 자동적이고 적응적인 캐시 관리 방법을 사용한다.

[47] 상세하게 살펴보면, 본 발명은 디스크와 같은 데이터 저장장치와 디스크를 위한 캐시 메모리를 포함하는 컴퓨터에서 프리페칭된 블록 캐시(121)와 캐싱된 블록 캐시(120)의 관리에 관한 방법이다.

[48] 우선, 본 발명의 실시예의 설명에 앞서 용어를 설명하면 다음과 같다.

[49] 프리페칭이란, 호스트나 프로그램이 디스크의 어떤 블록을 요구하지 않았으나 운영체제나 디스크 구동기가 어떤 블록을 캐시 메모리로 미리 읽어 놓는 동작을 말한다. 프리페칭된 블록 캐시(121)란, 프리페칭 되었으나 호스트나 프로그램이 요청하지 않은 디스크의 블록을 보유하고 있는 캐시이다. 캐싱된 블록 캐시(120)는 호스트 또는 프로그램이 이미 요구하였던 블록을 보유하고 있는 캐시이다. 빈 블록 캐시(122)는 데이터를 보유하고 있지 않은 블록 캐시이다.

[50] 프리페칭 적중이란, 프리페칭된 캐시 블록에 호스트나 프로그램이 요구하는 사건을 의미한다. 캐시 적중이란, 요구된 캐시 블록에 호스트나 프로그램이 요구하는 사건을 의미한다.

[51] 본 발명은 도 2에 나타낸 캐시 메모리를 조각 캐시로 관리하는 방법과 도 3에 나타낸 블록 캐시로 관리하는 방법으로 나뉜다. 이 두 가지 방법의 기본 운영원리는 같으나 구체적인 과정은 상이하다. 먼저, 도 2에 나타난 조각 캐시로 관리하는 적응적 캐시 솎음 기법(Cache culling method)을 설명하기로 한다. 그리고, 도 2에 나타난 캐시 솎음 기법은 조각 캐시 (Strip Cache) 단위로 프리페칭하는 조각 프리페칭(Strip Prefetching: 이하 'SP'라고 함) 기법과 잘 어울리는 방법이다. 여기서, 솎음 기법은 조각 프리페칭 기법에 한정되지 않지만, SP를 이용한 일례로써 본 발명을 설명하기로 한다.

[52] 조각이란 디스크에 연속한 블록들의 집합이다.

[53] 도 1은 다섯 개의 디스크들(1, 2, 3, 4)로 구성된 어떤 레이드(Redundant Array of Independent Disks: RAID)의 일 실시예를 보여주고 있다. 레이드에서의 스트라이프(Stripe)(30)는 디스크들(1, 2, 3, 4)의 조각들(Strip)(20)로 구성되어 있고, 조각(20)은 디스크 내에서 연속한 블록(10)들로 구성되어 있다. 단, 조각을 레이드에 한정하지 않고, 디스크에 연속한 블록들의 집합으로만 정의할 수도 있다.

[54] 조각 캐시(110)란, 상기 조각 단위로 관리되는 캐시이다. 조각 캐시(110)에는 캐싱된 블록 캐시(120)와 프리페칭된 블록 캐시(121)와 빈 블록 캐시(122)를 포함할 수 있다. 빈 블록 캐시(122)를 포함하지 않는 조각 캐시를 가득 채워진 조각 캐시라고 하고, 빈 블록 캐시(122)를 포함하는 조각 캐시를 부분적으로 채워진 조각 캐시라고 한다.

[55] SP로 프리페칭된 메모리를 효율적으로 관리하기 위해서, 본 발명인 적응적 캐시 솎음 기법은 캐시 적중률(Cache Hit Rate)과 프리페칭 적중률(Prefetching Hit Rate)의 합이 SP와 SP를 하지 않는 방식의 그 적중률들 보다 같거나 큼을 보장한다.

[56] 적응적 캐시 솎음 기법은, 프리페칭되고 요구되지 않은 블록 캐시들을 적정한 때에 적응적 방법으로 퇴출시킨다. 본 발명에서는 이러한 절차를 솎음 또는 솎아내기라고 부른다. 만약, 프리페칭되었으나 호스트가 요구하지 않은 블록 캐시들을 너무 일찍 솎아내면, 캐시 적중률은 증가하고 프리페칭 적중률은 감소한다. 만약, 프리페칭된 블록 캐시가 너무 늦게 솎아지면, 캐시 적중률이 감소하고 프리페칭 적중률은 증가한다.

[57] 도 1의 일 실시예에서는 캐시가 조각 캐시 단위로 관리되고, 각 조각 캐시는 네 개의 블록 캐시로 구성된다. 각 블록 캐시는 프리페칭된 블록 캐시 또는 캐싱된 블록 캐시 또는 빈 블록 캐시가 될 수 있다. 본 발명에서는 어떤 블록 캐시가 디스크 데이터를 위한 메모리를 갖고 있는지, 또는 프리페칭되었으나 참조되지 않았는지, 또는 캐싱된 블록 캐시인지 알 수 있는 추가적인 정보를 관리한다.

[58] 적응적 방법으로 적절한 순간에 프리페칭된 데이터를 요구된 데이터보다 먼저 퇴출 하기 위해서, 도 1에 도시된 조각 캐시들은 상류(upstream)(110)와 하류(downstream)(103)로 분할되어 있다. 상류(upstream)(110)와 하류(downstream)(103)는 LRU(least recently used) 정책으로 관리된다. 상류(101)는 프리페칭된 블록 캐시와 캐싱된 블록 캐시를 보유할 수 있지만, 하류(103)는 프리페칭된 블록 캐시를 포함하지 않는다. 새로 할당된 조각 캐시는 상류(101)의 MRU(Most recently used) 위치에 삽입된다. 만약, 상류(101)의 조각 캐시들의 수가 상류(101)가 가질 수 있는 조각 캐시들의 최대 개수 *Nu*를 초과하면, 상류(101)의 LRU(Least Recently Used) 조각 캐시(112)를 하류(103)의 MRU(Most Recently Used) 위치(113)로 옮긴다. 이때에 그 조각캐시의 모든 프리페칭된 블록 캐시들을 퇴출시킨다. 이 과정을 캐시 솎음이라고 한다.

[59] 본 발명의 적응적 캐시 솎음 기법은 변수 *Nu* (상류(101)가 가질 수 있는 최대 조각 캐시의 개수)를 적응적 방법으로 변경한다. 만약, *Nu*가 감소하면, 줄어든 프리페칭된 블록 캐시들 때문에 프리페칭 적중률은 감소하나, 캐시 적중률이 증가한다. 시스템의 성능은 프리페칭 적중률과 캐시 적중률의 합인 총 적중률에 의존한다. 본 발명은 총 적중률이 최대가 되도록 하는 제어기법을 포함한다.

[60] 우선, 적응적 솎음 기법의 캐시 관리과 구조에 대해서 설명하기로 한다.

[61] 조각 캐시의 모든 블록 캐시들이 캐시에서 퇴출될 때, 만약 그것의 형제(sibling)

조각 캐시들 중의 하나가 캐시에 살아 있다면, 그것은 유령 조각 캐시(105)가 된다. 유령 조각 캐시(105)는 과거에 호스트가 요구한 상태들 외에는 메모리를 가지지 않는다. 요구 상태는 그 조각 캐시의 블록들이 호스트나 프로그램에 의해서 요구되었었는지에 대한 정보를 알려준다. 형제 조각 캐시(115)들은 같은 레이드의 같은 스트라이프(30)에 속한 조각들(20)에 대응하는 조각 캐시들이다. 유령 조각 캐시(115)에 해당되는 어떤 디스크 입출력 요구가 발생하면, 그 유령 조각 캐시(115)는 과거의 요구상태를 유지한 조각 캐시(110)로 살아난다. 이렇게 과거에 유령이었던 조각 캐시를 재생된 조각 캐시라고 칭한다. 그러나, 재생된 조각 캐시는 과거의 요구 상태를 제외하면 다른 조각 캐시와 차이가 없다.

[62] 본 발명의 기법이 레이드에 사용된다면 유령 조각 캐시는 존재하고, 그렇지 않으면 유령 조각 캐시는 없다. 캐시에 살아 있는 형제 조각 캐시를 가지는 퇴출된 조각 캐시만이 유령 캐시가 될 수 있는 이유는, 스트라이프의 어떤 블록들을 RAID-5나 RAID-6을 구성하는 디스크들로 쓰기를 해야 할 때에, 형제 조각 캐시의 데이터들이 패러티를 갱신하는 데에 유용하기 때문이다. 또한, 유형 캐시는 과거의 요구 기록을 이용하여 성능을 향상시키기 위한 것이므로, 유형 캐시를 처리하지 않아도 적응적 솎음 기법은 동작한다.

[63] 호스트 또는 프로그램이 데이터 요구가 디스크 어레이 또는 디스크로 전달될 때마다, 상류(101)와 하류(102), 즉 두 리스트는 다음과 같은 법칙으로 관리된다.

[64] 상류(101)의 어떤 조각 캐시(110)에 포함된 프리페칭된 블록 캐시(121) 또는 캐싱된 블록 캐시(120)로의 읽기 요구가 발생하면, 그 요구된 조각 캐시는 상류의 MRU(Most recently used) 위치(111)로 옮겨간다.

[65] 하류(103)의 조각 캐시(110)에 포함된 블록(10)으로의 읽기 요구가 발생하면, 그 조각 캐시를 상류의 MRU위치(113)로 옮긴다. 만약, 그 요구가 캐시 미스를 발생시키면, 그 데이터는 SP없이 읽혀진다. 여기서, 캐시 미스란 요구한 블록이 캐시에 없어서 디스크로부터 읽어야 하는 상황을 의미한다.

[66] 유령 조각 캐시(115) 또는 상류(101)의 재생된 조각 캐시의 과거의 캐싱된 블록 캐시로의 캐시 미스, 또는 상류(101)의 조각 캐시로의 캐시 미스에 대하여, 그 요구된 조각 캐시는 상류의 MRU 위치(111)로 이동한다. 단, 그 요구된 블록은 SP 없이 디스크로부터 읽혀진다. 단, 과거의 캐싱된 블록 캐시란 그 해당 되는 블록 캐시가 유령이 되기 전에 캐싱된 블록 캐시였던 블록 캐시를 의미한다.

[67] 유령 조각 캐시(115) 또는 상류의 재생된 조각 캐시의 과거의 빈 블록 캐시에 해당하는 요구가 발생하면, 그 해당되는 조각은 SP 또는 어떤 프리페칭 기법으로 디스크로부터 읽혀지고, 상류의 MRU 위치(111)에 삽입된다. 단, 과거의 빈 블록 캐시란 그 해당되는 블록 캐시가 유령이 되기 전에 빈 블록 캐시였던 블록 캐시를 의미한다.

[68] 상류(101) 또는 하류(103)의 조각 캐시(110), 또는 유령 조각 캐시(115)에 속하지 않은 블록으로의 읽기 요구가 발생하면, 해당하는 블록을 위한 새로운 조각 캐시(110)를 할당하고, 그 조각 캐시를 상류의 MRU 위치(111)에 삽입한다. 단, 그

조각 캐시(110)는 SP 또는 어떤 프리페칭 기법으로 읽혀진다.

[69] 하류(103)에서 발생한 캐시 적중은 적중된 조각 캐시(110)를 상류(103)로 옮기지 않는다. 상류(103)의 부분적으로 채워진 조각 캐시가 상류로 옮겨질 수 있다면 상류가 차지한 메모리가 줄어든다. 왜냐하면, 상류(101)의 조각 캐시의 수를 *Nu*와 같게 하기 위해서 부분적으로 채워진 조각 캐시는, 어떤 가득 채워진 조각 캐시가 상류에서 퇴출시킬 수 있기 때문이다. 이 과정은 상류(101)와 하류(103)의 최적 분할을 깨트리게 한다.

[70] 형제 조각 캐시들은 RAID-5 또는 RAID-6 등이 디스테이지(레이드가 캐시에 있는 더티 데이터를 디스크로 쓰는 행위) 할 때에 더욱 유용하다. 그래서 유령 조각 캐시(115)의 캐싱된 블록 캐시(120)에 대한 호스트의 읽기 요구는 그 조각 캐시(110)를 프리페칭 없이 상류로 그 유령 조각 캐시(115)를 옮기고 그 유령 조각 캐시는 재생된다. 이 과정은 상류(101)의 메모리를 줄여서 더 많은 형제 조각 캐시들이 살 수 있게 한다. 왜냐하면, SP 또는 어떤 프리페칭이 없으면 그 유령 조각 캐시가 부분적으로 채워진 조각 캐시로 재생되기 때문이다. 만약, 재생된 조각 캐시가 오랫동안 사용되지 않으면, 그 조각 캐시는 상류에서 퇴출될 것이고, 따라서 상류는 자동으로 빼앗긴 메모리를 돌려받게 된다.

[71] 유령 조각 캐시(115)의 과거의 요구 상태는 유령 조각 캐시(115)가 재생되어도 유지된다. 이미 재생된 조각 캐시(110)나 유령 조각 캐시(115)의 과거의 캐싱된 블록 캐시 외의 블록에 해당하는 어떤 요구가 발생될 때에, 그 개별의 조각의 프리페칭된 블록 캐시(121)가 가치 있다고 여겨지므로, 그 요구를 위해서 SP가 수행된다.

[72] 본 발명은 캐시 치환 정책과 무관하다. 하류의 LRU 조각 캐시(114)는 어떤 하나의 퇴출 정책으로 선택될 수 있지만, 어떤 더 좋은 캐시 치환 정책으로 어떤 조각 캐시 또는 블록 캐시를 캐시에서 퇴출할 수 있다.

[73] 상류의 조각 캐시의 최대 수의 최적의 값 *Nu*를 찾는 것은 매우 중요한 부분이다. 프리페칭 적중률 P와 캐시 적중률 C의 합인 총 적중률(P+C)이 최대가 되게 하는 *Nu*가 최적이다. 총 적중률(P+C)이 최대가 되게 하는 *Nu*는 *Nu*에 대한 P+C의 함수의 기울기(slope)가 영이 되게 하는 *Nu*이다. 기울기는 미분값과 동일한다. 현재 *Nu*에 대한 P+C의 함수이 기울기를 구할 수 있다면, 다음의 <수학식 3>과 같이 기울기를 *Nu*에 되먹임하면 *Nu*는 자동으로 최적의 값으로 다가간다.

[74] 수학식 3

$$Nu \leftarrow Nu + slope$$

[75] 만약, *Nu*가 최적의 *Nu*보다 낮다면 슬로프(slope)가 양수가 되어서, 위의 되먹임은 *Nu*를 증가시키고, *Nu*가 최적의 *Nu*보다 높다면 슬로프(slope)가 음수가 되어서, 위의 되먹임은 *Nu*를 감소시킨다. 그래서, *Nu*는 자동으로 최적의 값으로 다가간다.

[76] 기울기는 *Nu*에 대한 P+C의 함수의 미분과 동일하다. 그리고, 그 미분은 각 프리페칭 적중률 P의 미분과 캐시 적중률 C의 미분의 합으로 다음의 <수학식 4>와 같이 나타낼 수 있다.

[77] 수학식 4

$$slope = \frac{d(P+C)}{dNu} = \frac{dP}{dNu} + \frac{dC}{dNu}$$

[78] 근사 미분 값은 도 3에서 보인 것과 같이 측정이 가능하다. 미분의 정의에 의하여, P의 미분은 어떤 주어진 시간동안 상류를 증가시키는 추가 할당 부분(150)에서 발생한 프리페칭 히트의 수

$$\Delta \dot{P}$$

와 거의 같다. 또한, 상기

$$\Delta \dot{P}$$

는 추가 할당 부분(150)과 인접한 상류 아래(120)에서 발생한 프리페칭 히트률 ΔP와 유사하다. 그러므로, *Nu*에 대한 프리페칭 적중률의 미분은 다음의 <수학식 5>와 같다.

[79] 수학식 5

$$\frac{dP}{dN_U} \cong \frac{\Delta \dot{P}}{\Delta N_U} \cong \frac{\Delta P}{\Delta N_U}$$

[80] 만약, 상류를 Δ*Nu*만큼 증가 시키면 하류는 αΔ*Nu*만큼 감소한다. 계수 α는 상류의 증가된 영역(150)과 하류의 줄어든 부분(151)의 빈 블록 캐시(122)들의 점유 비율에 의해서 결정된다. *Nu*에 대한 캐시 적중률 C의 미분은 하류의 줄어든 부분(152)에서의 캐시 적중률

$$\Delta \dot{C}$$

과 거의 유사하다. 만약, 캐시 적중률을 관측하려는 부분을 αΔ*Nu*개의 조각 캐시(110)가 아니라 상류 아래(102)와 같이 고정된 부분(104)으로 설정한다면,

$$\Delta \dot{C}$$

는 전체 아래(104)에서 발생한 캐시 적중률 ΔC와 거의 같다. 그러므로, C의 미분은 다음의 <수학식 6>과 같이 표현될 수 있다.

[81] 수학식 6

$$\frac{dC}{dN_U} \cong \frac{\Delta \dot{C}}{\Delta N_U} \cong \frac{\alpha \Delta P}{\Delta N_U}$$

[82] 이 근사화된 미분 값은 기존 기술인 적응적 치환 캐시에서의 순차 프리페칭(Sequential prefetching in Adaptive Replacement Cache: SARC)과 변방 사용률(Marginal Utility)과 유사한 점이 있다. 하지만, 변방 사용률로는 계수

의 존재를 설명할 수 없다. 더구나, 본 발명과 SARC 사이에는 몇 가지 다른 점이 있다. 첫 번째로, SARC는 어떤 블록이 퇴출되어야 하는지 결정하지만, 본 발명은 SARC에서 무시된 프리페칭된 블록의 퇴출만 고려한다. 두 번째로, SARC는 임의 접근 데이터와 순차 데이터와 관련있으나, 본 발명은 프리페칭 시나리오에서 프리페칭 적중과 캐시 적중을 고려한다. 세 번째로, 본 발명은 스트라이핑 디스크 어레이의 효율적인 관리를 위해 조각 단위로 캐시를 관리하지만, SARC는 단일 디스크를 위해 설계되어 있다.

[83] 현재 상류(101)와 하류(103)의 분할에서의 상류 크기에 대한 총 적중률의 슬로프(slope)는, 어떤 시간동안, 상류 아래(102)에서 발생한 프리페칭 적중 수 ΔP와 전체 아래(104)에서 발생한 캐시 적중 수 ΔC를 관측하여 다음의 <수학식 7>로 구하여진다.

[84] 수학식 7

$$slope \cong S(\Delta P - \partial \Delta C)$$

[85] 상류 아래(102)는 상류의 아래 부분이고, 상류 아래에 속한 조각 캐시의 수는 고정되어 있다. 비슷하게 전체 아래(104)는 상류와 하류를 결합하는 전체 리스트의 아래 부분이고, 전체 아래(104)에 속한 조각 캐시의 수는 상류 아래(102)와 같다. 초기 상태에서는 하류가 없거나 매우 작을 수 있다. 그래서, 전체 아래(104)와 상류 아래(102)는 서로 겹쳐질 수 있다.

[86] 상기 <수학식 7>의 비율 계수 S는 상기 <수학식 3>의 되먹임에 의한 적응속도를 결정한다. 상기 S가 클수록 *Nu*는 최적으로 값으로 빨리 접근하지만 오버 슈트(Overshoot)가 발생할 수 있다.

[87] 미분 되먹임을 이용한 적응적 캐시 솎음 기법에서 사용하는 수식은 상기 <수학식 3>과 상기 <수학식 7>를 결합한 것으로, 다음의 <수학식 8>과 같은 되먹임 수식이다.

[88] 수학식 8

$$Nu \leftarrow Nu + S(\Delta P - \partial \Delta C)$$

[89] 단, 계수 α는 (상류 아래의 프리페칭된 블록 캐시 및 캐싱된 블록 캐시의 수) /

(하류 아래의 캐싱된 블록 캐시의 수)로 정의 될 수 있다.

[90] 상기 <수학식 8>은 프로그램으로 구현되어 적응적 캐시 솎음 기법에서 최대의 총 적중률이 발생할 수 있도록, 상류(101)의 크기를 동적으로 제어한다.

[91] 본 발명은 조각 캐시로 관리되는 캐시 메모리뿐만 아니라, 도 4와 같이 블록 캐시로 관리되는 캐시 메모리에도 사용될 수 있다. 모든 블록 캐시들은 두 개의 리스트들 상류(201)와 하류(203)로 분리된다. 도 4에서 나타난 네모들은 블록캐시들을 나타낸다. 새롭게 디스크에서 읽은 블록 캐시는 상류(201)의 MRU(Most recently used) 위치(210)로 삽입된다. 캐시 적중 또는 프리페칭 적중이 발생한 상류(201)의 블록 캐시들은 상류(201)의 MRU(Most recently used) 위치(210)로 옮겨진다. 캐시 적중이 발생한 하류(203)의 블록 캐시들은 하류(203)의 MRU(Most recently used) 위치(212)로 옮겨진다. 캐시 미스가 발생하면 해당 블록을 위한 블록 캐시를 할당하고 그것을 상류의 MRU(Most recently used) 위치(210)에 삽입한다. 상류(201)에 속한 블록 캐시의 수가 상류(201)가 보유할 수 있는 블록 캐시의 최대 수 (*Nu*)를 초과하면, 상류에서 가장 오랫동안 사용하지 않은 LRU(Least Recently Used) 블록 캐시(211)를 상류에서 제거한다. 만약, 그 LRU(Least Recently Used) 블록 캐시(211)가 프리페칭된 블록 캐시(121)면 그것을 캐시에서 퇴출시키고, 캐싱된 블록 캐시(120)이면 하류의 MRU 위치(212)에 삽입한다.

[92] 상류 아래(202)는 상류에서 가장 오랫동안 사용되지 블록 캐시들의 집합이고, 상류 아래(202)의 블록 캐시의 수는 이미 설정된 값이다. 상류(201)와 하류(203)를 통틀어서 가장 오랫동안 사용되지 않는 그룹 캐시들의 집합들을 전체 아래(204)라고 한다. 전체 아래(204)의 블록 캐시의 수는 상류 아래(202)의 것과 같다.

[93] 소정 기간 동안, 상류 아래(102)에서 발생한 프리페칭 적중의 수 ΔP를 기록하고, 전체 아래(104)에서 발생한 캐시 적중 개수 ΔC를 기록한다. 그리고, 추가적으로, 그 기록 시간을 ΔP 또는 ΔC가 1이 되는 두 연속하는 시간 차이로 한다. 그러면, 상기 <수학식 8>로써 상류(201)의 크기를 조절한다. 단, 계수 α는 1 + (상류 아래의 프리페칭된 블록 캐시의 수)/(전체 아래의 블록 캐시의 수)이다.

[94] 도 5는 조각 캐시로 관리되는 캐시를 위한 적응적 캐시 솎아내기 기법을 설명하는 흐름도이다.

[95] 도 5를 참조하면, 호스트나 프로그램이 301 단계에서 디스크 또는 레이드로 블록 읽기 요구를 기다리는 중 읽기 요구가 발생하면, 302 단계에서 상기 읽기 요구가 어느 블록에 해당하는지를 판단한다. 만약, 읽기 요구가 유령 조각 캐시 또는 상류의 재생된 조각 캐시의 과거의 빈 블록 캐시로의 캐시 미스 또는 상류의 조각 캐시에서 발생된 캐시 미스이면, 그 해당 조각이 조각 프리페칭 또는 어떤 프리페칭으로 디스크에서 조각 캐시로 읽혀지고, 그 조각 캐시는 상류의 MRU(Most recently used)위치로 삽입되는 303 단계로 진행한다.

[96] 상기 302 단계에서 상기 읽기 요구가 유령 조각 캐시 또는 상류의 재생된 조각

캐시의 과거의 캐싱된 블록 캐시로의 읽기 요구이면, 그 해당 조각 캐시는 상류의 MRU(Most recently used) 위치로 옮겨진다. 하지만, 그 요구된 블록은 프리페칭없이 디스크에서 읽혀지는 304 단계를 수행한다. 상기 302 단계에서 하류의 조각 캐시에서 캐시 적중이면, 그 조각 캐시를 상류의 MRU로 옮기는 305 단계를 수행한다. 상기 302 단계에서 상류의 조각 캐시에 발생한 캐시 적중 또는 프리페칭 적중이면, 그 조각 캐시를 하류의 MRU위치에 삽입하는 306 단계를 수행한다.

[97] 상기 303 단계 또는 304 단계를 수행한 후에, 310 단계로 진행하여 상류에 속한 조각 캐시의 수가 *Nu*보다 큰지 확인하게 된다. 상기 310 단계에서 상기 상류에 속한 조각 캐시의 수가 *Nu*보다 크면, 311 단계에서와 같이 상류의 LRU 캐시의 프리페칭된 블록 캐시들을 캐시에 퇴출하고, 그 조각 캐시를 하류의 MRU위치에 삽입한 후 다시 310 단계로 돌아간다. 반면, 상기 310 단계에서 상기 상류에 속한 조각 캐시의 수가 *Nu*보다 작으면, 상기 301 단계로 진행한다.

[98] 한편, 상기 305 단계 이후에 307 단계에서 그 캐시 적중이 하류 아래에서 발생한 것이면, 309 단계에서와 같이 상기 <수학식 5>를 수행하고, 310 단계로 진행한다. 상기 306 단계 이후에 그 적중이 상류 아래에서 발생한 프리페칭 적중이면 309 단계로 진행하고, 그렇지 않으면 301 단계로 진행하게 된다.

[99] 도 6은 블록 캐시로 관리되는 캐시를 위한 적응적 캐시 솎아내기 기법을 설명하는 흐름도이다.

[100] 도 6을 살펴보면, 호스트나 프로그램이 401 단계에서 디스크 또는 레이드로 블록 읽기 요구를 기다리는 중 어떤 요구가 발생하면, 402 단계에서와 같이 상기 읽기 요구가 어디의 적중인지 미스인지를 판단하게 된다. 상기 요구된 데이터가 캐시에 없는 것이면, 403 단계에서와 같이 그 해당 조각 또는 어떤 프리페칭 기법으로 디스크에서 블록 캐시로 읽고 그 블록 캐시는 상류의 MRU(Most recently used) 위치에 삽입된다. 그런 후 409 단계에서와 같이 상류에 속한 블록 캐시의 수가 *Nu*보다 큰지 검사하는 단계를 수행한다. 상기 409 단계에서 상류에 속한 블록 캐시의 수가 *Nu*보다 크면, 410 단계로 진행하고, 아닌 경우 401 단계로 돌아가게 된다. 이후 410 단계에서는 상류의 LRU 블록 캐시가 프리페칭된 블록 캐시면 캐시에서 퇴출시키고, 그렇지 않으면 그것을 하류의 MRU 위치에 삽입한다.

[101] 다음으로, 상기 402 단계에서, 요구된 데이터가 하류에서 발생한 캐시 적중이면, 404 단계로 진행하여 그 조각 캐시를 상류의 MRU 위치로 옮긴 후, 406 단계에서와 같이 상기 캐시 적중이 하류 아래에서 발생한 것인지 검사하는 단계를 수행한다. 상기 캐시 적중이 하류 아래에서 발생한 것이라면, 408 단계에서 상기 <수학식 5>를 실행하는 단계를 수행하고 409 단계로 진행한다.

[102] 한편, 상기 402 단계에서, 요구된 데이터가 상류에서 발생한 캐시 적중 또는 프리페칭 적중이면, 405 단계로 진행하여 그 조각 캐시를 하류의 MRU 위치에 삽입한 후, 407단계에서와 같이 그 적중이 상류 아래에서 발생한 프리페칭

적중인지 검사하는 단계를 수행한다. 상기 적중이 상류 아래에서 발생한 프리페칭 적중이면 408 단계로 진행하고, 그렇지 않으면 상기 401 단계로 진행한다.

[103] 한편 본 발명의 상세한 설명에서는 본 발명에 의한 적응적 캐시 솎음 기법에 관해 설명하였으나, 본 발명의 범위에서 벗어나지 않는 한도 내에서 여러 가지 변형이 가능함은 물론이다. 그러므로 본 발명의 범위는 설명된 실시 예에 국한되어 정해져서는 안되며 후술하는 특허청구의 범위뿐 아니라 이 특허청구의 범위와 균등한 것들에 의해서 정해져야 한다.

산업상 이용가능성

[104] 본 발명에 관한 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법은, 프리페칭 적중률과 캐시 적중률의 두 변경(邊境) 사용률들을 동일하도록 프리페칭 블록 캐시들의 수와 캐싱된 블록 캐시들의 수를 최적으로 할당하고, 적응적 방법으로 적당한 시간에 사용되지 않는 프리페칭된 블록들을 메모리에서 퇴출시킴으로써, 종래 기법보다 높은 프리페칭 및 캐시 적중률로써 데이터 저장장치의 성능을 향상시킬 수 있고, 낮은 디스크 읽기 비용으로도 고성능의 이득을 얻을 수 있어, 비용을 절감할 수 있으며, 어떠한 가정 없이 개발된 범용적 캐시 관리 방법으로, 별도의 하드웨어 변경없이 기존의 소형에서 대형 컴퓨터 또는 저장장치까지 간단히 적용할 수 있다.

청구범위

[1] 컴퓨터 저장 장치에서의 프리페칭된 데이터를 관리 하는 방법에 있어서,
전체 캐시를 조각 캐시 단위로 관리하고, 조각 캐시들이 상류와 하류로 분할되며, 상기 상류는 상기 프리페칭된 블록 캐시와 상기 캐싱된 블록 캐시를 가지고, 상기 하류는 상기 캐싱된 블록 캐시만을 가지도록 제어하는 제 1과정;
상기 상류가 가질 수 있는 조각 캐시들의 수(*Nu*)를 프리페칭 적중률과 캐시 적중률의 합의 미분값을 이용하여 갱신하는 제 2과정; 및
상기 상류에 포함된 조각 캐시의 수가 상기 갱신된 조각 캐시들의 수 *(Nu)* 보다 큰 경우, LRU(Least Recently Used) 정책에 따라 상류의 LRU 조각 캐시를 하류로 이동하되, 상기 조각 캐시의 프리페칭된 블록 캐시를 상기 전체 캐시에서 제거시키는 제 3과정;
을 포함하는,
컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

[2] 제 1항에 있어서,
상기 조각 캐시들의 수(Nu)는 다음의 <수학식 1>에 의한 방법으로 되먹임시키며,
<수학식 1>

$$Nu \leftarrow Nu + S(\Delta P - \partial \Delta C)$$

여기서, 상기 ΔP는 임의 시간 동안 상기 상류 아래에서 발생한 상기 프리페칭 적중(히트) 수, 상기 ΔC는 같은 시간 동안 상기 전체 아래에서 발생한 상기 캐시 적중 수, 상기 α는 (상류 아래의 프리페칭된 블록 캐시 및 캐싱된 블록 캐시의 수) / (하류 아래의 캐싱된 블록 캐시의 수), 상기 S는 상수인, 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

[3] 제 1항에 있어서,
상기 상류 및 상기 하류의 조각 캐시들은 각각 LRU(Least Recently Used) 정책으로 관리되는, 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

[4] 제 1항에 있어서,
상기 제 1과정은,
상기 상류의 임의의 조각 캐시에 포함된 프리페칭된 블록 캐시 또는 캐싱된 블록 캐시로의 읽기 요구가 발생하면, 상기 조각 캐시를 상류의 MRU(Most recently used) 위치로 이동하는 과정을 더 포함하는, 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

[5] 제 1항에 있어서,
상기 제 1과정은,
상기 하류의 조각 캐시에 포함된 캐싱된 블록 캐시로의 읽기 요구가 발생하면, 상기 조각 캐시를 상기 하류의 MRU(Most recently used) 위치로

이동하는 과정을 더 포함하고,
상기 읽기 요구가 캐시 미스를 발생시키면 상기 읽기 요구된 데이터는 조각 프리페칭 없이 디스크에서 읽혀지는, 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

[6] 제 1항에 있어서,
상기 제 1과정은,
유령 조각 캐시 또는 상기 상류의 재생된 조각 캐시의 상기 과거의 캐싱된 블록 캐시로의 읽기 요구가 캐시 미스, 또는 상류의 조각 캐시로의 캐시 미스를 발생시키면, 해당 조각 캐시는 상류의 MRU(Most recently used) 위치로 이동하는 과정을 더 포함하고,
상기 읽기 요구된 블록은 조각 프리페칭 (Strip Prefetching: SP) 없이 디스크로부터 읽혀지는, 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

[7] 제 1항에 있어서,
상기 제 1과정은,
유령 조각 캐시 또는 상기 상류의 재생된 조각 캐시의 과거의 빈 블록 캐시에 해당하는 읽기 요구가 발생되면, 해당되는 조각 캐시가 조각 프리페칭 (Strip Prefetching: SP) 또는 소정의 프리페칭 기법으로 디스크로부터 읽혀지고, 상기 상류의 MRU(Most recently used) 위치에 삽입되는 과정을 더 포함하는, 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

[8] 제 1항에 있어서,
상기 제 1과정은,
상기 상류 또는 상기 하류의 조각 캐시, 또는 유령 조각 캐시에 속하지 않은 블록으로의 읽기 요구가 발생하면, 해당 블록을 위한 조각 캐시를 할당하고, 해당 조각 캐시를 상류의 MRU(Most recently used) 위치에 삽입하는 과정을 더 포함하며,
상기 조각 캐시는 조각 프리페칭(Strip Prefetching: SP) 또는 소정의 프리페칭 기법으로 읽혀지는, 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

[9] 컴퓨터 저장 장치에서의 프리페칭된 데이터를 관리하는 방법에 있어서,
전체 캐시를 상기 블록 캐시 단위로 관리되고, 블록 캐시들이 상류와 하류로 분할되며, 상기 상류는 상기 프리페칭된 블록 캐시와 상기 캐싱된 블록 캐시를 가지고, 상기 하류는 상기 캐싱된 블록 캐시만을 가지도록 제어하는 제 1과정;
상기 상류가 가질 수 있는 상기 블록 캐시들의 수(*Nu*)를 프리페칭 적중률과 캐시 적중률의 합의 미분값을 이용하여 갱신하는 제 2과정; 및
상기 상류에 포함된 블록 캐시의 수가 상기 갱신된 블록 캐시들의 수(*Nu*

)보다 큰 경우, LRU(Least Recently Used) 정책에 따라 상류의 LRU 블록 캐시를 하류로 이동하되, 상기 블록 캐시의 프리페칭된 블록 캐시를 캐시에서 제거시키는 제 3과정;

을 포함하는,

컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

[10] 제 9항에 있어서,

상기 블록 캐시들의 수(Nu)는 다음의 <수학식 2>에 의한 방법으로 되먹임시키며,

<수학식 2>

$$Nu \leftarrow Nu + S(\Delta P - \partial \Delta C)$$

여기서, 상기 ΔP는 어떤 시간 동안 상기 상류 아래에서 발생한 상기 프리페칭 적중의 수, ΔC는 같은 시간 동안 상기 전체 아래에서 발생한 상기 캐시 적중의 수, α는 1 + (상류 아래의 프리페칭된 블록 캐시들의 수)/(전체 아래의 블록 캐시들의 수)이며, S는 상수인, 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

[11] 제 9항에 있어서,

상기 상류 및 상기 하류의 블록 캐시들은 각각 LRU(Least Recently Used) 정책으로 관리되는, 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

[12] 제 9항에 있어서,

상기 제 1과정은,

상기 상류의 임의의 블랙 캐시에 포함된 프리페칭된 블록 캐시 또는 캐싱된 블록 캐시로의 읽기 요구가 발생하면, 상기 블록 캐시를 상류의 MRU(Most recently used) 위치로 이동하는 과정을 더 포함하는, 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

[13] 제 9항에 있어서,

상기 제 1과정은,

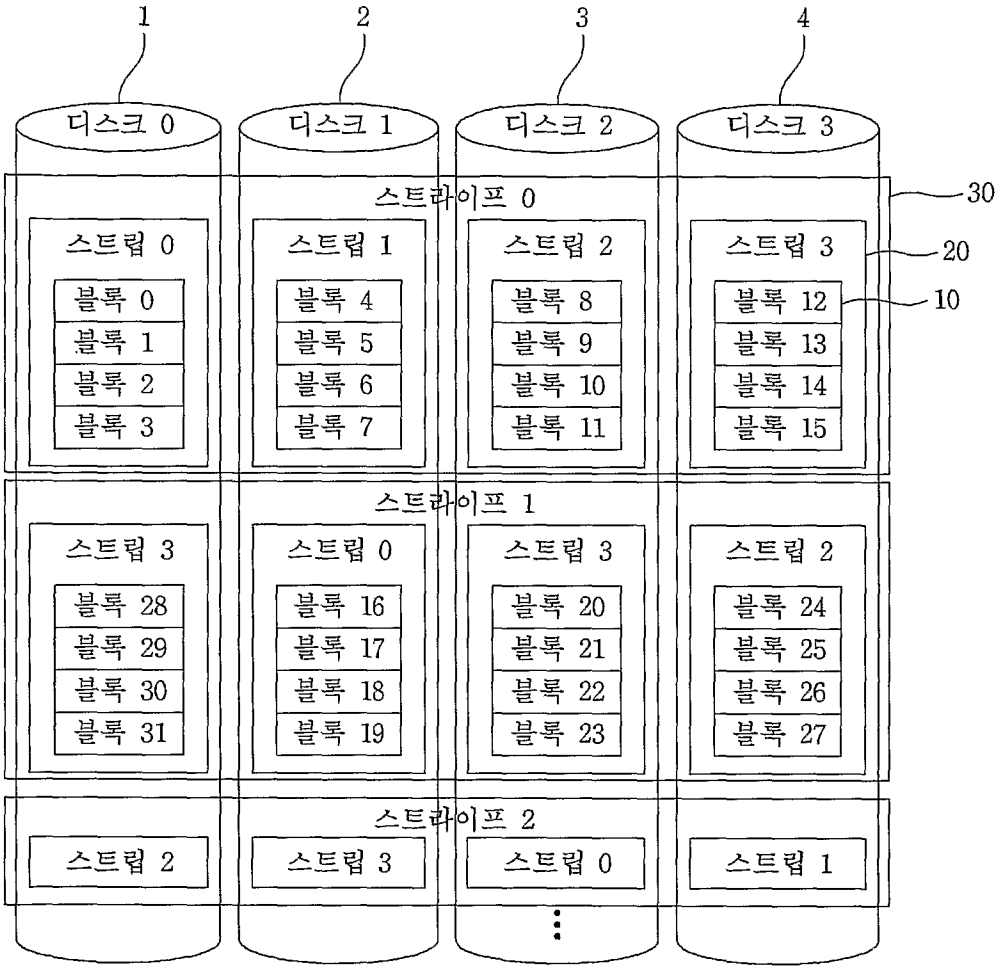
상기 하류의 블랙 캐시에 포함된 캐싱된 블록 캐시로의 읽기 요구가 발생하면, 상기 블록 캐시를 하류의 MRU(Most recently used) 위치로 이동하는 과정을 더 포함하는, 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

[14] 제 13항에 있어서,

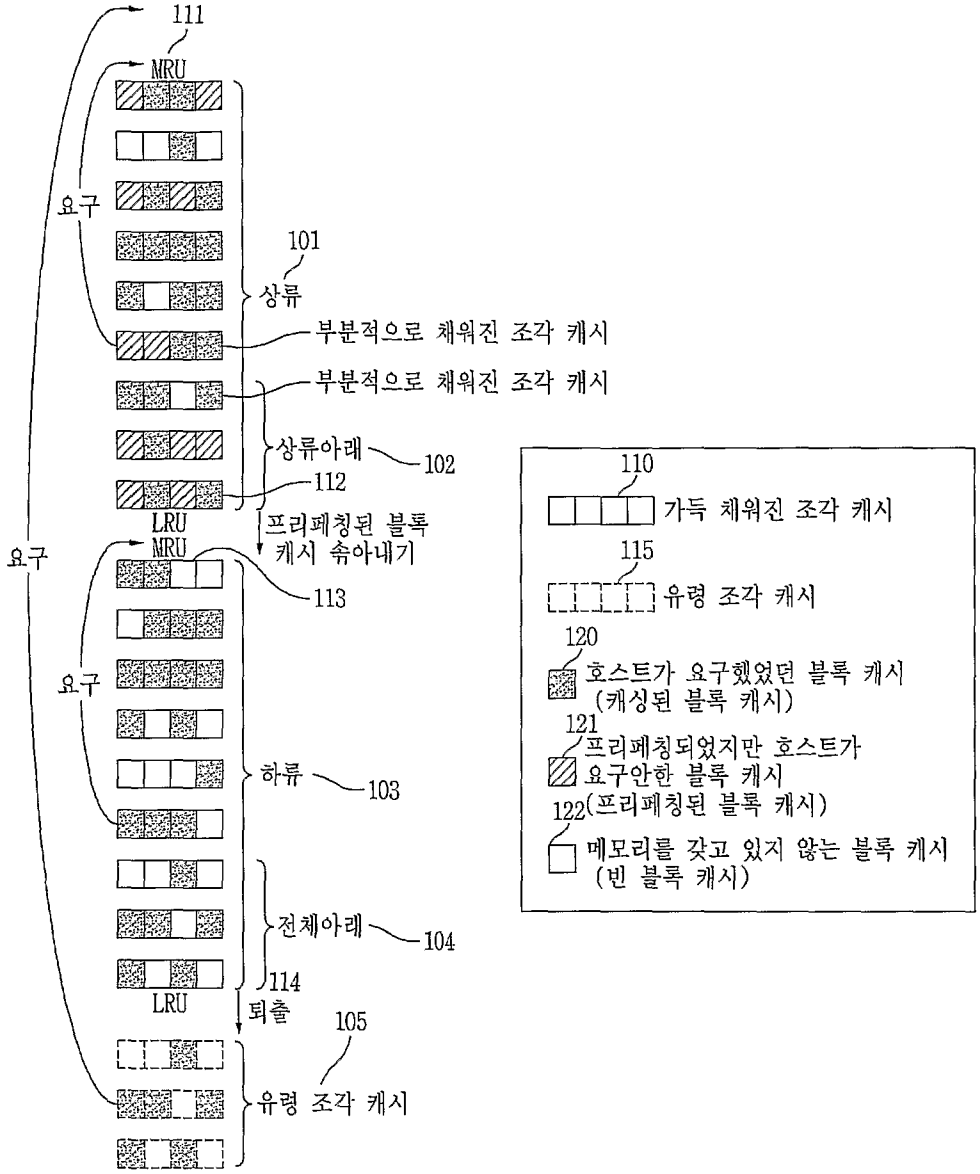
상기 읽기 요구가 캐시 미스를 발생시키면, 해당되는 블록을 위한 블록 캐시를 새로 할당하고, 상류의 MRU(Most recently used) 위치에 삽입하는 과정을 더 포함하는, 컴퓨터 저장장치에서의 프리페칭 데이터 관리 방법.

【도면】

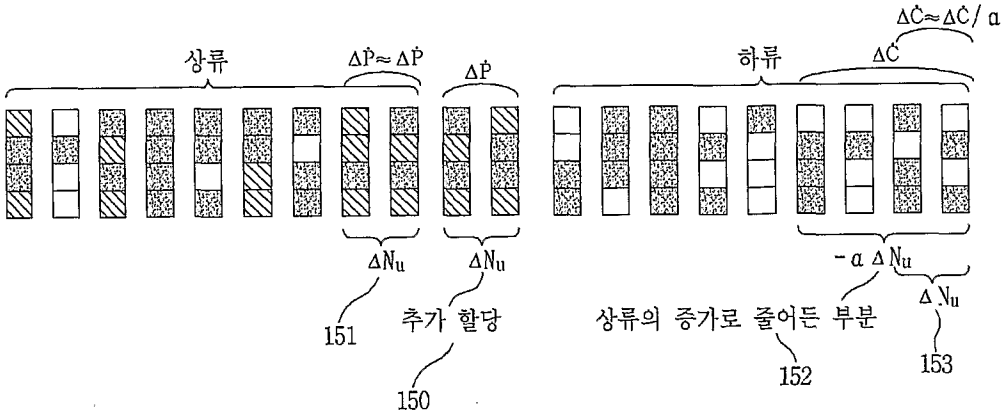
【도 1】


1
2
3
4
디스크 0
디스크 1
디스크 2
디스크 3
스트라이프 0
30
스트립 0
스트립 1
스트립 2
스트립 3
20
블록 0
블록 1
블록 2
블록 3
블록 4
블록 5
블록 6
블록 7
블록 8
블록 9
블록 10
블록 11
블록 12
블록 13
블록 14
블록 15
10
스트라이프 1
스트립 3
스트립 0
스트립 3
스트립 2
블록 28
블록 29
블록 30
블록 31
블록 16
블록 17
블록 18
블록 19
블록 20
블록 21
블록 22
블록 23
블록 24
블록 25
블록 26
블록 27
스트라이프 2
스트립 2
스트립 3
스트립 0
스트립 1

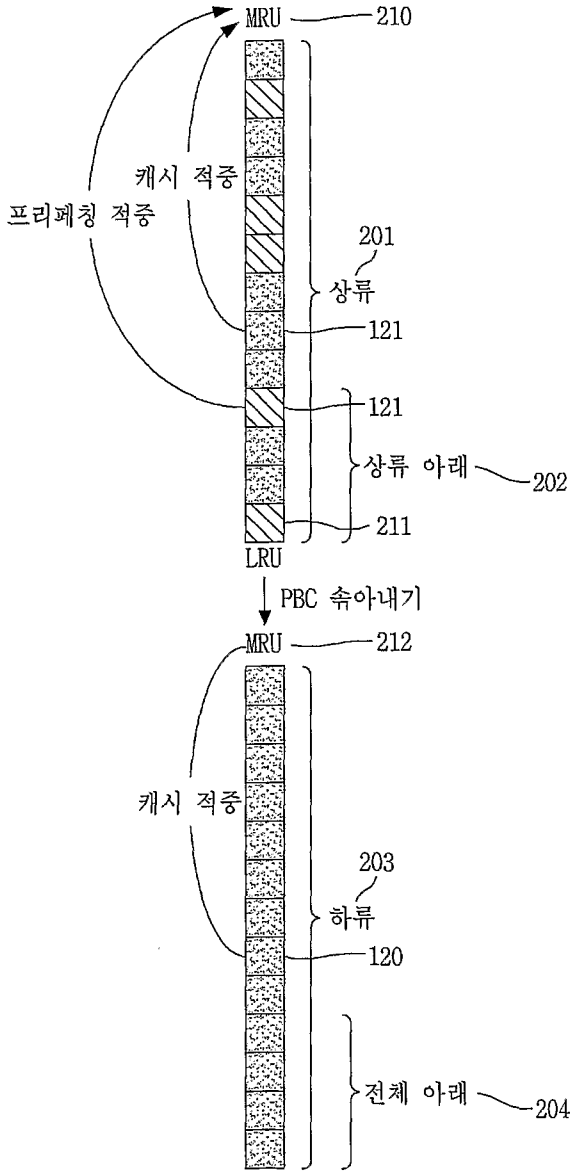
【도 2】


111
MRU
요구
101
상류
부분적으로 채워진 조각 캐시
부분적으로 채워진 조각 캐시
상류아래
102
112
LRU
프리페칭된 블록
캐시 솎아내기
요구
MRU
113
요구
하류
103
전체아래
104
114
LRU
퇴출
105
유령 조각 캐시
110
가득 채워진 조각 캐시
115
유령 조각 캐시
120
호스트가 요구했었던 블록 캐시
(캐싱된 블록 캐시)
121
프리페칭되었지만 호스트가
요구안한 블록 캐시
122(프리페칭된 블록 캐시)
메모리를 갖고 있지 않는 블록 캐시
(빈 블록 캐시)

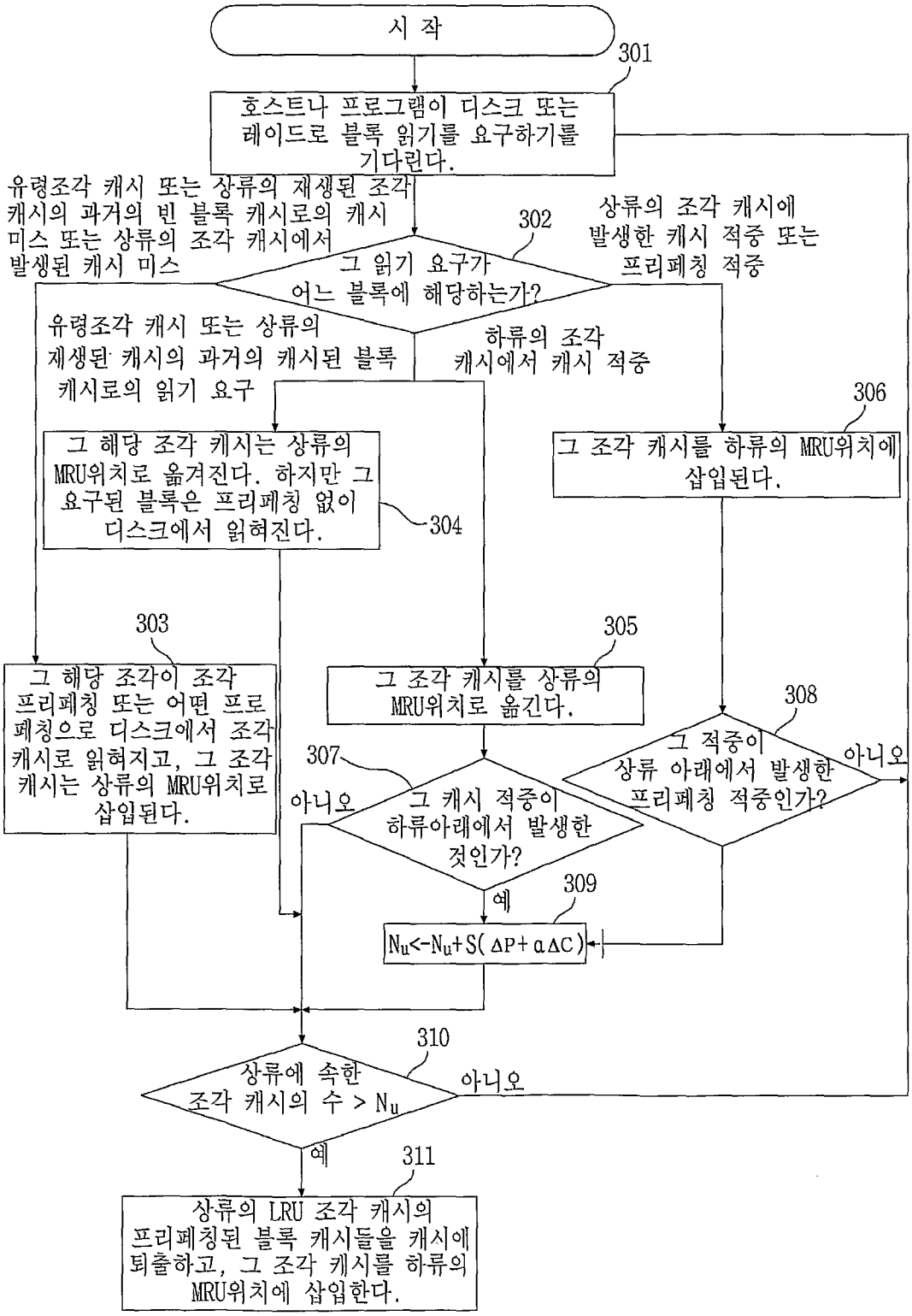
【도 3】


상류
ΔṖ≈ΔṖ
ΔṖ
하류
ΔĊ
ΔĊ≈ΔĊ/ α
ΔNu
ΔNu
-α ΔNu
ΔNu
151
추가 할당
150
상류의 증가로 줄어든 부분
152
153

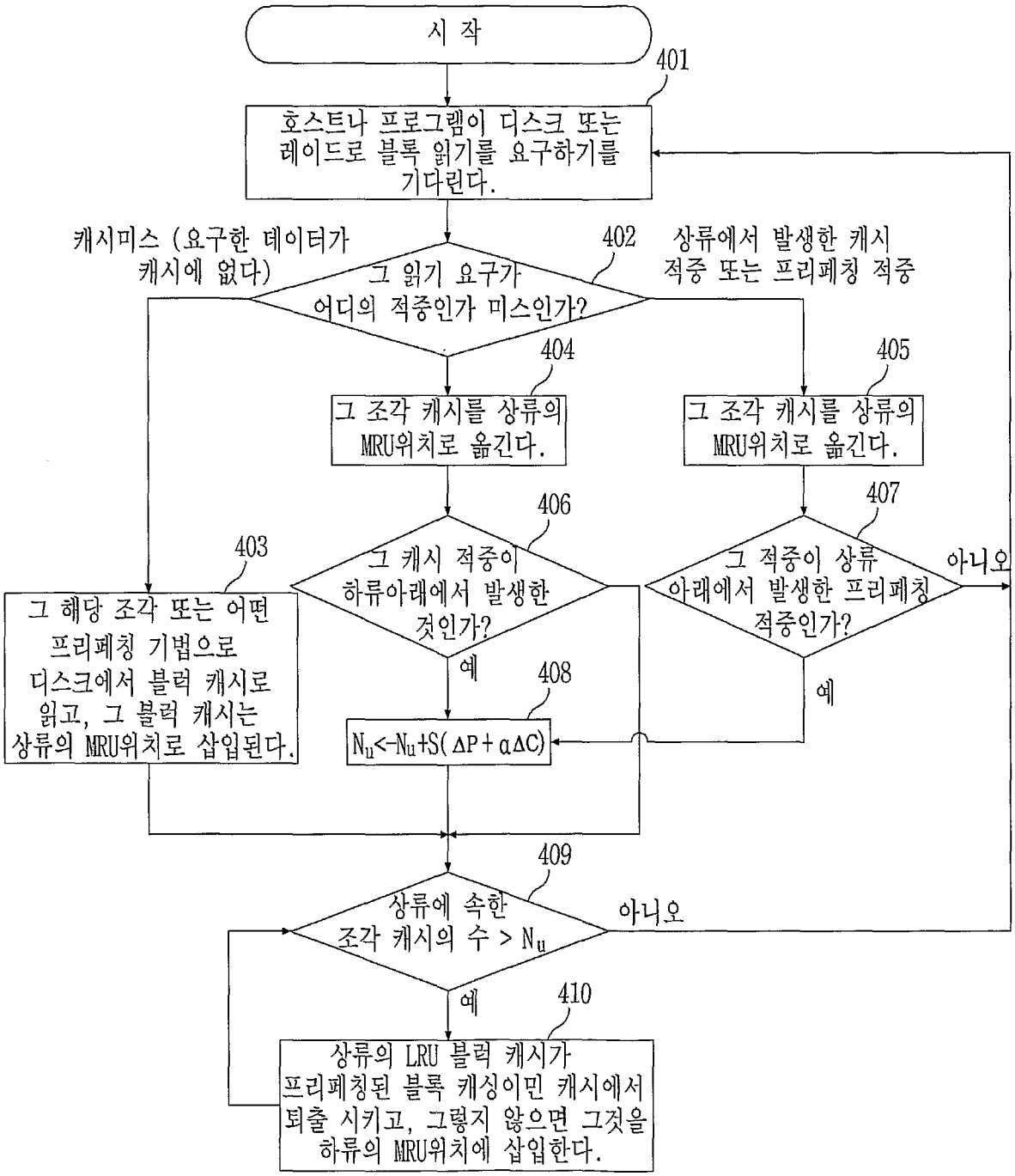
【도 4】


MRU
210
캐시 적중
프리페칭 적중
201
상류
121
121
상류 아래
202
211
LRU
PBC 솎아내기
MRU
212
캐시 적중
203
하류
120
전체 아래
204

【도 5】


시 작
301
호스트나 프로그램이 디스크 또는 레이드로 블록 읽기를 요구하기를 기다린다.
유령조각 캐시 또는 상류의 재생된 조각 캐시의 과거의 빈 블록 캐시로의 캐시 미스 또는 상류의 조각 캐시에서 발생된 캐시 미스
302
그 읽기 요구가 어느 블록에 해당하는가?
상류의 조각 캐시에 발생한 캐시 적중 또는 프리페칭 적중
유령조각 캐시 또는 상류의 재생된 캐시의 과거의 캐시된 블록 캐시로의 읽기 요구
하류의 조각 캐시에서 캐시 적중
306
그 조각 캐시를 하류의 MRU위치에 삽입된다.
그 해당 조각 캐시는 상류의 MRU위치로 옮겨진다. 하지만 그 요구된 블록은 프리페칭 없이 디스크에서 읽혀진다.
304
303
그 해당 조각이 조각 프리페칭 또는 어떤 프로 페칭으로 디스크에서 조각 캐시로 읽혀지고, 그 조각 캐시는 상류의 MRU위치로 삽입된다.
305
그 조각 캐시를 상류의 MRU위치로 옮긴다.
308
그 적중이 상류 아래에서 발생한 프리페칭 적중인가?
아니오
307
그 캐시 적중이 하류아래에서 발생한 것인가?
아니오
예
309
N_u<-N_u+S(ΔP+ αΔC)
310
상류에 속한 조각 캐시의 수 > N_u
아니오
예
311
상류의 LRU 조각 캐시의 프리페칭된 블록 캐시들을 캐시에 퇴출하고, 그 조각 캐시를 하류의 MRU위치에 삽입한다.

【도 6】


시 작
401
호스트나 프로그램이 디스크 또는 레이드로 블록 읽기를 요구하기를 기다린다.
캐시미스 (요구한 데이터가 캐시에 없다)
402
그 읽기 요구가 어디의 적중인가 미스인가?
상류에서 발생한 캐시 적중 또는 프리페칭 적중
404
그 조각 캐시를 상류의 MRU위치로 옮긴다.
405
그 조각 캐시를 상류의 MRU위치로 옮긴다.
406
그 캐시 적중이 하류아래에서 발생한 것인가?
407
그 적중이 상류 아래에서 발생한 프리페칭 적중인가?
아니오
403
그 해당 조각 또는 어떤 프리페칭 기법으로 디스크에서 블럭 캐시로 읽고, 그 블럭 캐시는 상류의 MRU위치로 삽입된다.
예
408
N_u<-N_u+S(ΔP + αΔC)
예
409
상류에 속한 조각 캐시의 수 > N_u
아니오
예
410
상류의 LRU 블럭 캐시가 프리페칭된 블록 캐싱이민 캐시에서 퇴출 시키고, 그렇지 않으면 그것을 하류의 MRU위치에 삽입한다.