



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0063078
(43) 공개일자 2022년05월17일

(51) 국제특허분류(Int. Cl.)
G06F 3/06 (2006.01) G06F 12/02 (2018.01)
G06F 13/16 (2006.01) G06F 16/10 (2019.01)
(52) CPC특허분류
G06F 3/0643 (2013.01)
G06F 12/0246 (2013.01)
(21) 출원번호 10-2021-0072749
(22) 출원일자 2021년06월04일
심사청구일자 없음
(30) 우선권주장
1020200148903 2020년11월09일 대한민국(KR)

(71) 출원인
서강대학교산학협력단
서울특별시 마포구 백범로 35 (신수동, 서강대학교)
(72) 발명자
박성용
서울특별시 서대문구 가재울미래로 2, 107동 100
1호(남가좌동, DMC파크뷰자이)
김준형
서울특별시 강남구 삼성로64길 5, 107동 2205호(대치동, 대치현대아파트)
(74) 대리인
유미특허법인

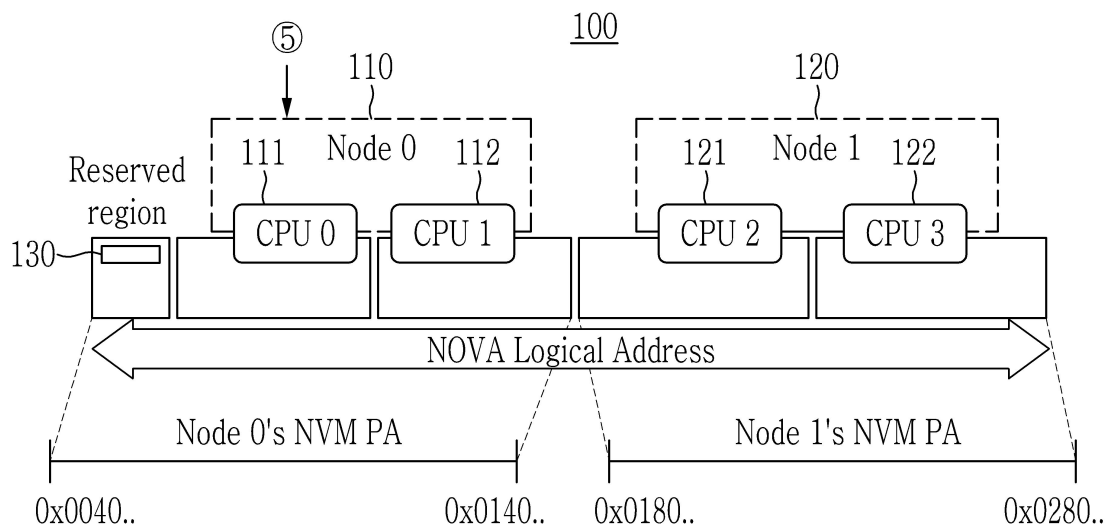
전체 청구항 수 : 총 8 항

(54) 발명의 명칭 NUMA 기반 매니코어 서버의 비휘발성 메모리 파일 시스템

(57) 요약

비휘발성 메모리 파일 시스템으로서, 데이터를 저장할 수 있는 비휘발성 메모리로 구성되고, 각 비휘발성 메모리에 단일 연속 가상 주소 공간으로 가상화된 가상 주소 공간이 각각 할당되고, 비휘발성 메모리의 크기에 따라 비휘발성 메모리의 전체 논리 주소 공간을 분할하여 복수의 노드 파티션을 생성하는 NUMA(Non-Uniform Memory Access) 노드와, 복수의 노드 파티션들의 수만큼 구비되며, 실행되는 스레드에 의해 NUMA 노드의 비휘발성 메모리에 데이터를 데이터를 쓰거나 NUMA 노드의 비휘발성 메모리에 쓰인 데이터를 읽어들이되, 비휘발성 메모리에 미리 설정된 물리적 주소 공간에 상기 데이터를 우선적으로 쓰기를 수행하는 복수의 CPU들을 포함한다.

대표도



(52) CPC특허분류

G06F 13/1657 (2013.01)
G06F 13/1694 (2013.01)
G06F 16/10 (2019.01)
G06F 3/0604 (2013.01)
G06F 3/061 (2013.01)
G06F 3/067 (2013.01)
G06F 3/0679 (2013.01)
G06F 2212/2542 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711081081
과제번호	2014-3-00035-006
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	SW컴퓨팅산업원천기술개발(R&D)
연구과제명	매니코어 기반 초고성능 스케일러블 OS 기초 연구 (차세대OS기초연구센터)
기 여 율	1/1
과제수행기관명	한국전자통신연구원
연구기간	2019.01.01~2019.12.31

명세서

청구범위

청구항 1

비휘발성 메모리 파일 시스템으로서,

복수의 CPU들, 그리고 물리 주소가 논리 주소로 가상화된 가상 주소 공간이 할당되어 있는 비휘발성 메모리를 각각 포함하며, 상기 비휘발성 메모리에 데이터가 저장되는 복수의 NUMA(Non-Uniform Memory Access) 노드들, 그리고

상기 복수의 NUMA 노드들에 각각 포함된 상기 비휘발성 메모리의 물리 주소를 저장하는 수퍼 블록

을 포함하고,

상기 적어도 하나의 CPU는, 스레드가 실행되면 상기 비휘발성 메모리의 물리 주소가 매핑된 공간에 우선적으로 데이터를 데이터 쓰기 수행하는, 비휘발성 메모리 파일 시스템.

청구항 2

제1항에 있어서,

상기 비휘발성 메모리의 크기에 따라 상기 비휘발성 메모리의 전체 논리 주소 공간을 분할하여 복수의 노드 파티션들이 생성되어 있는, 비휘발성 메모리 파일 시스템.

청구항 3

제2항에 있어서,

상기 적어도 하나의 CPU는 상기 노드 파티션들의 수만큼 상기 NUMA 노드에 포함되는, 비휘발성 메모리 파일 시스템.

청구항 4

제1항에 있어서,

상기 복수의 NUMA 노드들 중 제1 NUMA 노드의 가상 주소와, 상기 복수의 NUMA 노드들 중 제2 NUMA 노드의 가상 주소는 가상 주소 공간은, 연속되지 않은 비휘발성 메모리 주소를 단일 연속 가상 주소로 가상화된 주소인, 비휘발성 메모리 파일 시스템.

청구항 5

제4항에 있어서,

상기 복수의 CPU들 중 제1 CPU와 제2 CPU가 각각 스레드를 실행하여 상기 비휘발성 메모리에 상기 데이터를 쓰면, 각 스레드가 비휘발성 메모리의 로컬 우선 할당 정책을 기초로 CPU 코어의 메모리 할당자를 사용하여 데이터 페이지 또는 로그 페이지를 할당하는, 비휘발성 메모리 파일 시스템.

청구항 6

비휘발성 메모리 파일 시스템으로서,

데이터 페이지에 사용자 데이터를 저장하는 복수의 CPU들,

각 CPU별로 로그의 시작인 헤드 포인터와 끝인 테일 포인터로 구성된 각 CPU별 데이터 구조인 로컬 로그, 그리고

상기 각 CPU와 해당 로컬 로그 포인터를 인덱싱하는 포인터 배열인 글로벌 로그

를 포함하는, 비휘발성 메모리 파일 시스템.

청구항 7

제6항에 있어서,

상기 글로벌 로그는 비휘발성 메모리에 저장된 아이노드에 포함되는, 비휘발성 메모리 파일 시스템.

청구항 8

제7항에 있어서,

상기 각 CPU가 실행한 스레드가 상기 글로벌 로그를 참조하여 로컬 로그의 테일 포인터를 갱신한 후, 로그 페이지에 로그 항목을 추가하는, 비휘발성 메모리 파일 시스템.

발명의 설명**기술 분야**

[0001] 본 발명은 NUMA 기반 매니코어 서버에서 파일의 읽기/쓰기를 최적화하기 위한 비휘발성 메모리 파일 시스템에 대한 기술이다.

배경 기술

[0002] 최근 몇 년 동안 3D-Xpoint와 같은 비휘발성 메모리(NVM: Non-Volatile Memory) 장치를 위한 여러 파일 시스템이 제안되었다. 그 중에서 최첨단 NVM 파일 시스템인 NOVA(Non-Volatile memory Accelerated) 파일 시스템은 블록 기반 파일 시스템보다 높은 처리량과 낮은 읽기-쓰기 지연 시간을 보장한다. 또한, NOVA는 로그 구조화된 디자인을 통해, 파일 데이터와 메타 데이터의 일관성을 보장한다.

[0003] 이러한 NOVA는 모든 쓰기 작업에 대한 메타 데이터를 기록하는 노드 별로 로깅을 채택한다. 그러나 NOVA는 동시 공유 파일 I/O가 수행될 때, I/O 처리량 측면에서 전혀 확장성을 제공하지 않는다. NOVA가 노드 별 로그의 일관성을 보장하기 위해 inode를 획일적으로 잠그는데, 이로 인해 NOVA 및 고성능 NVM 장치의 동시 특성의 이점들이 무효화되기 때문이다.

[0004] NOVA의 획일적인 inode 잠금으로 인한 낮은 동시성으로 인한 확장성 문제를 해결하기 위해, 획일적인 잠금 대신 범위 잠금이라고 하는 정밀한 범위 기반 RW (Readers-Writer) 잠금 기법을 사용하는 것이 좋다. 범위 잠금 기반 NOVA는 공유 파일 I/O에 대한 성능 확장에 성공한다.

[0005] 그러나 범위 잠금 기법은 UMA(Uniform Memory Access) 아키텍처에만 적용되며, NUMA(Non-Uniform Memory Access) 아키텍처에서는 적용되지 않는다. 이는 NOVA가 NUMA 환경용으로 설계되지 않았기 때문이다.

[0006] 즉, NOVA는 단일 NUMA 노드에 모든 파일 데이터와 메타 데이터를 배치한다. 따라서 다른 노드에서 실행 중인 스레드는 원격 메모리 액세스를 통해 파일에 액세스해야 하므로, 성능이 크게 저하되는 문제점이 있다.

발명의 내용**해결하려는 과제**

[0007] 따라서, 본 발명은 NUMA 노드에 위치한 NVM 장치를 단일 주소 공간으로 사용할 수 있도록 가상화하는 NUMA 인식 NOVA 파일 시스템이, 로컬 NVM 장치에 파일 데이터와 메타 데이터가 우선적으로 배치되도록 로컬 우선 배치 정책을 채택하여, NUMA 기반 매니코어 서버에서 파일의 읽기/쓰기를 최적화할 수 있는 비휘발성 메모리 파일 시스템을 제공한다.

과제의 해결 수단

[0008] 상기 본 발명의 기술적 과제를 달성하기 위한 본 발명의 하나의 특징인 비휘발성 메모리 파일 시스템으로서,

[0009] 복수의 CPU들, 그리고 물리 주소가 논리 주소로 가상화된 가상 주소 공간이 할당되어 있는 비휘발성 메모리를 각각 포함하며, 상기 비휘발성 메모리에 데이터가 저장되는 복수의 NUMA(Non-Uniform Memory Access) 노드들, 그리고 상기 복수의 NUMA 노드들에 각각 포함된 상기 비휘발성 메모리의 물리 주소를 저장하는 수퍼 블록을 포함하고, 상기 적어도 하나의 CPU는, 스레드가 실행되면 상기 비휘발성 메모리의 물리 주소가 매핑된 공간에 우

선적으로 데이터를 데이터 쓰기를 수행한다.

- [0010] 상기 비휘발성 메모리의 크기에 따라 상기 비휘발성 메모리의 전체 논리 주소 공간을 분할하여 복수의 노드 파티션들이 생성되어 있을 수 있다.
- [0011] 상기 적어도 하나의 CPU는 상기 노드 파티션들의 수만큼 상기 NUMA 노드에 포함될 수 있다.
- [0012] 상기 복수의 NUMA 노드들 중 제1 NUMA 노드의 가상 주소와, 상기 복수의 NUMA 노드들 중 제2 NUMA 노드의 가상 주소는 가상 주소 공간은, 연속되지 않은 비휘발성 메모리 주소를 단일 연속 가상 주소로 가상화된 주소일 수 있다.
- [0013] 상기 복수의 CPU들 중 제1 CPU와 제2 CPU가 각각 스레드를 실행하여 상기 비휘발성 메모리에 상기 데이터를 쓰면, 각 스레드가 비휘발성 메모리의 로컬 우선 할당 정책을 기초로 CPU 코어의 메모리 할당자를 사용하여 데이터 페이지 또는 로그 페이지를 할당할 수 있다.
- [0014] 상기 본 발명의 기술적 과제를 달성하기 위한 본 발명의 또 다른 특징인 비휘발성 메모리 파일 시스템으로서,
- [0015] 데이터 페이지에 사용자 데이터를 저장하는 복수의 CPU들, 각 CPU별로 로그의 시작인 헤드 포인터와 끝인 테일 포인터로 구성된 각 CPU별 데이터 구조인 로컬 로그, 그리고 상기 각 CPU와 해당 로컬 로그 포인터를 인덱싱하는 포인터 배열인 글로벌 로그를 포함한다.
- [0016] 상기 글로벌 로그는 비휘발성 메모리에 저장된 아이노드에 포함될 수 있다.
- [0017] 상기 각 CPU가 실행한 스레드가 상기 글로벌 로그를 참조하여 로컬 로그의 테일 포인터를 갱신한 후, 로그 페이지에 로그 항목을 추가할 수 있다.

발명의 효과

- [0018] 본 발명에 따르면, 여러 NUMA 노드의 NVM 모듈을 가상화하고 로컬 우선 쓰기 정책을 통해 병렬 I/O를 수행할 때 발생할 수 있는 원격 액세스 수를 줄일 수 있다.
- [0019] 또한, NOVA 파일 시스템의 노드 별 로그를 잠금 없는 CPU 별 로그로 확장하여, 기존 NOVA 파일 시스템보다 높은 쓰기 처리량을 보여줄 수 있다.

도면의 간단한 설명

- [0020] 도 1은 일반적인 NVM이 NUMA 서버의 물리적 주소 공간을 구성하는 예시도이다.
- 도 2는 일반적인 NOVA 파일 시스템의 쓰기 I/O 흐름을 나타낸 예시도이다.
- 도 3은 본 발명의 실시예에 따른 NUMA 인지적인 메모리 관리 기능이 있는 NOVA 파일 시스템의 예시도이다.
- 도 4는 본 발명의 실시예에 따른 비휘발성 메모리 파일 시스템의 구조도이다.
- 도 5는 본 발명의 실시예에 따른 비휘발성 메모리 파일 시스템의 대역폭을 측정된 결과를 나타낸 예시도이다.
- 도 6은 본 발명의 실시예에 따른 NOVA 파일 시스템의 확장성을 평가한 예시도이다.
- 도 7은 본 발명의 실시예에 따른 컴퓨팅 장치의 구조도이다.

발명을 실시하기 위한 구체적인 내용

- [0021] 아래에서는 첨부한 도면을 참고로 하여 본 발명의 실시예에 대하여 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 용이하게 실시할 수 있도록 상세히 설명한다. 그러나 본 발명은 여러 가지 상이한 형태로 구현될 수 있으며 여기에서 설명하는 실시예에 한정되지 않는다. 그리고 도면에서 본 발명을 명확하게 설명하기 위해서 설명과 관계없는 부분은 생략하였으며, 명세서 전체를 통하여 유사한 부분에 대해서는 유사한 도면 부호를 붙였다.
- [0022] 명세서 전체에서, 어떤 부분이 어떤 구성요소를 "포함"한다고 할 때, 이는 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있는 것을 의미한다.
- [0023] 이하, 도면을 참조로 하여, 본 발명의 실시예에 따른 NUMA 기반 매니코어 서버의 비휘발성 메모리 파일 시스템 및 방법에 대해 설명한다. 본 발명의 실시예에 대해 설명하기 앞서, 일반적인 NUMA 서버와 NOVA 파일 시스템에

대해 먼저 도 1 및 도 2를 참조로 설명한다.

- [0024] 도 1은 일반적인 NVM이 NUMA 서버의 물리적 주소 공간을 구성하는 예시도이다.
- [0025] 도 1에 도시된 바와 같이, 종래에는 바이트 주소 지정이 가능한 파일 시스템인 NOVA 파일 시스템을 사용하였다.
- [0026] 종래의 매니코어 서버는 NUMA 아키텍처를 기반으로 한다. 그리고, 매니코어 시스템용으로 설계된 시스템 애플리케이션은 자신의 성능을 완전히 활용하기 위해 내부 아키텍처를 인식해야 한다.
- [0027] 그러나 NOVA 파일 시스템은 단일 NVM 공간에 상주하도록 설계된 NVM 기반 파일 시스템이다. 따라서, 도 1에 도시된 바와 같이 NUMA 기반 매니코어 머신에 설치된 여러 NVM 장치의 비 연속적 주소 공간 구성을 통해, 단일 NUMA 노드의 NVM 공간만 현재 NOVA 파일 시스템의 데이터 및 로그 페이지를 배치하는 데 사용할 수 있다.
- [0028] 이에 따라 다른 NUMA 노드에서 실행되는 스레드는 원격 메모리 액세스를 유발하게 되고, 이것이 확장성 병목 현상이 된다. 이러한 NOVA 파일 시스템에서 데이터 쓰기 흐름에 대해 도 2를 참조로 설명한다.
- [0029] 도 2는 일반적인 NOVA 파일 시스템의 쓰기 I/O 흐름을 나타낸 예시도이다.
- [0030] NOVA 파일 시스템은 로그 구조 설계를 채택하여 일관성을 보장하고, COW(Copy-On-Write) 방식으로 NVM의 노드별 로그에 수정 사항을 기록한다. NOVA 파일 시스템은 DRAM과 NVM으로 구성되고, NOVA 파일 시스템에서 사용되는 주요한 데이터 구조는 도 2에 도시된 바와 같다.
- [0031] DRAM의 인덱스 트리(10)는 각 파일에 대해 유지되며, 읽기/쓰기 작업의 파일 오프셋에 해당하는 데이터 페이지를 검색하는 데 사용된다.
- [0032] NOVA 파일 시스템의 inode(20)는 NVM에 저장되며, inode 로그(30)에 저장된 로그 항목의 시작과 끝에 대한 헤드 포인터(21) 및 테일 포인터(22)를 포함한다.
- [0033] inode 로그(30)는 일련의 로그 페이지이다. 각 로그 항목의 크기는 64B이고, 사용자 데이터는 4KB 크기로 데이터 페이지에 저장되며, CPU 당 메모리 페이지 할당자에 의해 NVM에 할당된다.
- [0034] 이와 같은 구조를 가지는 NOVA 파일 시스템에서 데이터 쓰기에 대한 I/O 흐름을 예로 하여 설명하면, 사용자가 파일의 10KB 내지 14KB의 쓰기 범위에서 데이터 쓰기를 요청한다고 가정한다. 그러면 NOVA 파일 시스템은 4KB 쓰기 작업 단위로 두 개의 새로운 데이터 페이지를 할당한다(①). 그리고 이전 데이터 페이지(Data 2)의 내용을 새로 할당된 페이지에 복사하고, 업데이트 할 범위에서 사용자 데이터를 덮어쓴다.
- [0035] 새로운 데이터에 대한 새 로그 항목은 마지막 완료(committed)된 로그 항목을 가리키는 inode(20)의 테일 포인터(22)를 사용하여 inode 로그(30)에 추가된다(②). 따라서 새 로그 항목은 테일 포인터(22) 바로 뒤에 기록된다. 만약, 테일 포인터(22)가 마지막 로그 항목에 있으면, 새 로그 페이지가 할당되고 로그 항목이 새 로그 페이지에 기록된다.
- [0036] 테일 포인터(22)는 새 로그 항목의 위치를 반영하도록 갱신된다(③). 마지막으로 DRAM의 인덱스 트리(10)가 업데이트되어(④), 새 인덱스 노드가 새 로그 항목을 가리키고, NOVA 파일 시스템의 쓰기 작업이 완료된다.
- [0037] 이와 같이, 범위 기반 RW 잠금을 통한 확장성 제한 NOVA 파일 시스템은 서로 다른 오프셋에서 동일한 파일에 데이터를 쓰거나 동시에 로그를 업데이트 할 때, 여러 스레드 간에 일관성이 없는 경우를 방지하기 위해 노드 별 잠금을 사용한다.
- [0038] 예를 들어, 스레드 T1과 T2가 동시에 같은 파일에 쓰고 있고, 노드 별 잠금이 없다고 가정한다. T1이 로그 항목을 쓰고 있지만(도 2의 ②) T1이 테일 포인터를 업데이트하기 전에 T2가 로그 항목을 추가하는 경우(도 2의 ③), T1이 쓴 로그 항목을 T2가 덮어쓸 수 있다. 이로 인해, T1의 데이터가 손실되는 문제점이 발생한다.
- [0039] 또한, NOVA 파일 시스템의 노드 별 잠금이 동일한 파일을 읽거나 쓰려고 시도하는 스레드를 직렬화하여, 병렬 I/O의 병목 현상이 발생되기도 하였다. 따라서, 이러한 문제점을 해결하기 위해 종래에 세분화된 범위 기반 RW 잠금이 제안되었다.
- [0040] 범위 잠금은 먼저 쓰기 요청의 범위를 확인하고, 쓰기 범위가 겹치지 않으면 잠금을 획득하고, 사용자 데이터(도 2의 ①)를 동시에 쓸 수 있다. 세분화된 범위 기반 RW 잠금은 UMA 아키텍처에서 파일 시스템 확장성을 보장하지만, NUMA 아키텍처에서 성능 확장에는 도움이 되지 않는다는 단점이 있다.
- [0041] 따라서, 본 발명의 실시예에서는 NUMA 노드에 위치한 NVM 장치를 단일 주소 공간으로 사용할 수 있도록 가상화

하는 NUMA 인식 NOVA 파일 시스템을 제안한다. 본 발명의 실시예에서는 두 개의 실시예에 따른 NUMA 인식 NOVA 파일 시스템을 제안한다.

- [0042] 먼저 제1 실시예에 따른 NUMA 인식 NOVA 파일 시스템은 NVM 장치의 가상화를 제공할 수 있다. 즉, 여러 NUMA 노드에 파일을 저장하기 위해, 각 노드에 있는 NVM의 인접하지 않은 물리적 주소 공간들을 하나의 논리적 주소 공간으로 가상화하여 사용한다. 그리고, 스레드가 실행되고 있는 CPU에 할당된 NVM 장치에, 파일 데이터와 메타 데이터를 우선적으로 배치하는 로컬 우선 쓰기 정책을 사용하는 제1 실시예에 따른 NUMA 인식 NOVA 파일 시스템을 제안한다.
- [0043] 또한, 본 발명의 제2 실시예에서는 잠금 없는 코어별 데이터 구조를 사용할 수도 있다. 잠금 없는 코어별 데이터 구조는, 다중 스레드가 동시에 쓰기 작업을 수행할 수 있도록 허용하여 확장성을 보장한다. 즉, 글로벌 로그 및 로컬 로그를 사용하는 노드 별 로그와 같은 잠금 없는 코어 별 데이터 구조를 사용하는 제2 실시예에 따른 NUMA 인식 NOVA 파일 시스템을 제안한다.
- [0044] 도 3은 본 발명의 제1 실시예에 따른 NUMA 인지적인 메모리 관리 기능이 있는 NOVA 파일 시스템의 예시도이다.
- [0045] NUMA 인지적 파일 시스템(100)은 여러 NUMA 노드(110, 120)에 파일을 배포하여, 원격 메모리 액세스를 줄일 수 있다. 본 발명의 실시예에서는 설명의 편의를 위하여 두 개의 NUMA 노드(110, 120)만을 도시하였으나, 반드시 이와 같이 한정되는 것은 아니다. 여기서 두 개의 NUMA 노드(110, 120)는 서로 인접해 있다고 가정한다.
- [0046] 본 발명의 실시예에서는 NUMA 인지적 NOVA 파일 시스템(100)을 설계하기 위해, 여러 NUMA 노드(110, 120)에서 연속되지 않은 NVM 주소 공간을 단일 연속 가상 주소 공간으로 가상화하여 사용한다.
- [0047] 이를 위해, 파일 시스템을 NVM에 마운트 할 때, 모든 NUMA 노드에서 슈퍼 블록(130)으로 NVM 장치의 정보인 물리적 메모리 주소를 가져온다. 결과적으로 NVM 장치의 물리적 주소가 선형으로 매핑되어, NUMA 인지적인 NOVA 파일 시스템이 사용자 데이터와 메타 데이터를 여러 NVM 장치에 쉽게 배치할 수 있다. 여기서 슈퍼 블록(130)은 예약된 영역(reserved region)에 위치한다.
- [0048] 원격 메모리 액세스를 완화하려면 메모리 할당 정책이 필요하다. 본 발명의 실시예에 따른 NUMA 인지적인 NOVA 파일 시스템(100)에는 스레드가 로컬 NVM 장치에 파일을 쓰는 것을 선호하는 로컬 우선 쓰기 정책을 적용한다.
- [0049] 도 3에 도시된 바와 같이 두 개의 NUMA 노드(110, 120)가 있는 NUMA 파일 시스템(100)에서의 메모리 할당 정책을 설명하면 다음과 같다.
- [0050] CPU-0(111) 및 CPU-1(112)에서 실행되는 스레드는 0x0040에서 0x0140 범위의 물리적 주소 공간에 파일을 쓰는 것을 선호한다. 반면, CPU-2(121) 및 CPU-3(122)은 0x0180에서 0x0280 범위에서 파일을 작성한다.
- [0051] 또한, NUMA 인식 NOVA 파일 시스템(100)은 비휘발성 메모리의 크기에 따라 전체 논리 주소 공간을 분할하고, NUMA 노드 파티션(⑤)별로 논리 주소 공간을 생성한다. 또한 NUMA 노드 파티션은 각 노드의 코어 수 즉, 각 NUMA 노드에 포함되어 있는 일정 수의 CPU 코어들로 나뉘며, NOVA 파일 시스템의 CPU 당 메모리 할당자에 의해 관리된다.
- [0052] 그리고 각 스레드의 데이터 및 로그 페이지는 스레드가 실행중인 코어에 할당된 영역에서 할당된다. 이를 통해 각 I/O 스레드는 먼저 로컬 NVM 공간에서 데이터 또는 로그 페이지를 할당할 수 있다.
- [0053] 다양한 스레드간에 공유되는 파일 데이터 구조는 NUMA 기반 시스템에서 파일 시스템의 확장성 병목 현상의 주요 원인이 된다. 파일이 서로 다른 NUMA 노드에서 실행되는 여러 스레드들에서 공유되면, 빈번한 원격 메모리 액세스로 인해 성능이 크게 저하된다.
- [0054] 또한, 공유 리소스는 일관성을 보장하기 위해 잠금으로 보호되므로 동시 업데이트 작업이 직렬화된다. 따라서, 본 발명의 실시예에서는 확장 가능한 NOVA 파일 시스템을 위한 CPU 별 데이터 구조를 제안하고, NOVA 파일 시스템의 노드 별 로그 구조를 잠금없는 CPU 별 로그로 확장한다. 이에 대해 도 4를 참조로 설명한다.
- [0055] 도 4는 본 발명의 제2 실시예에 따른 비휘발성 메모리 파일 시스템의 구조도이다.
- [0056] 도 4에 도시된 바와 같이, NOVA 파일 시스템(100)의 잠금없는 CPU 별 구조에 따르면, 두 가지 새로운 영구 데이터 구조인 글로벌 로그(140) 및 로컬 로그(150)로 구분할 수 있다.
- [0057] 글로벌 로그(140)는 도 4에 도시된 바와 같이, 각 CPU(161~163)와 해당 로컬 로그 포인터를 인덱싱하는 각 inode(⑥)에 대한 포인터 배열이다. 하나의 inode(⑥)에 하나의 글로벌 로그(140)가 CPU 수만큼의 로컬 로그 포

인터를 갖는다.

- [0058] 로컬 로그(150)는 CPU(161~163) 별 로그에 대한 헤드 포인터(151) 및 테일 포인터(152)가 있는 CPU 별 개인 데이터 구조이다.
- [0059] 두 개의 데이터 구조 즉, 글로벌 로그(140)와 로컬 로그(150)를 통해, 서로 다른 NUMA 노드의 여러 스레드가 파일 작업을 동시에 수행하고 확장성을 향상시킬 수 있다.
- [0060] 다음은 Intel Optane DC PM(Persistent Memory) 모듈이 장착된 매니코어 서버에서 NOVA 파일 시스템의 확장성을 평가한 예에 대해 도 5를 참조로 설명한다. 본 발명의 실시예에서는 6개의 Intel Optane DC PM 모듈이 각 소켓에 장착되어 있는 환경에서, 비휘발성 메모리 파일 시스템의 대역폭을 측정한 결과에 대해 설명한다.
- [0061] 도 5는 본 발명의 실시예에 따른 비휘발성 메모리 파일 시스템의 대역폭을 측정한 결과를 나타낸 예시도이다.
- [0062] 본 발명의 실시예에서는 단일 소켓 내 PM 모듈의 메모리 공간은 운영 체제에서 연속된 물리적 주소 공간으로 제공된다. 이 설정에서 PM에 대한 액세스는 단일 소켓의 6 PM 모듈에 걸쳐 인터리브된다. 그러나 서버는 소켓에서 PM 모듈을 집계하는 것을 허용하지 않는다.
- [0063] 또한, 본 발명의 실시예에서는 읽기 전용 및 쓰기 전용 워크로드로 코어 수를 늘려 Testbed의 각 메모리 모듈 대역폭을 측정하였으며 MLC(Memory Latency Checker) 도구를 사용하였다. MLC 도구는 여러 스레드가 메모리 장치에서 순차적으로 데이터를 읽거나 쓰는 워크로드의 최대 대역폭을 측정한다. 특히 정확한 메모리 대기 시간 측정을 위해 MLC 도구는 하드웨어 프리페처(prefetcher)를 비활성화한다.
- [0064] 도 5의 (a)는 PM의 측정된 쓰기 대역폭을 나타낸 예시도이다. 도 5의 (a)에서 Optane DC(src, dst)는 소켓 번호 src(또는, '소스 소켓 번호'라고도 지칭함)의 코어가 소켓 번호 dst(또는, '목적지 소켓 번호'라고도 지칭함)의 Optane DC PM에 할당된 메모리 버퍼에 액세스 할 때 I/O 작업을 수행함을 나타낸다.
- [0065] Optane DC (0,0)는 4개 코어에서 12.3GB/s의 최대 대역폭에 도달하지만, 코어 수가 증가함에 따라 점차적으로 성능이 저하된다. 반면에 Optane DC (0,1)의 최대 대역폭은 8 코어에서 6.5GB/s의 최고치에 도달하지만, 코어 수가 증가함에 따라 계속 감소한다.
- [0066] 도 5의 (b)에는 PM의 측정된 읽기 대역폭을 도시한 예시도이다. Optane DC (0,0) 대역폭은 코어 수가 증가함에 따라 선형적으로 증가하여 28개 코어에서 최대 40.04GB/s가 된다. Optane DC (0,1)의 대역폭은 코어 수가 증가함에 따라 증가하여 8개 코어에서 최대 15GB/s가 된다. 그러나 동일한 하드웨어 제한으로 인해 그 이후에는 대역폭이 떨어진다.
- [0067] 이상의 예시도를 요약하면, PM의 읽기 대역폭이 쓰기 대역폭보다 높다. Optane DC (0,0) 및 Optane DC (0,1)의 읽기 대역폭은 해당 쓰기 대역폭의 읽기 대역폭보다 각각 3.24x 및 2.31x 높다.
- [0068] 그리고, PM에서 NUMA 효과의 정도가 높다. 쓰기 측면에서 Optane DC (0,0)는 Optane DC (0,1)보다 8개 코어에서 1.86 배 더 높은 쓰기 대역폭을, 28개 코어에서 16.5배 더 높은 쓰기 대역폭을 보여준다. 읽기 측면에서 8코어에서 Optane DC (0,0)의 읽기 대역폭은 Optane DC (0,1)보다 1.98배 높고 28코어에서 35.7배 더 높다.
- [0069] 도 6은 본 발명의 실시예에 따른 NOVA 파일 시스템의 확장성을 평가한 예시도이다.
- [0070] 본 발명의 실시예에 따라 NOVA 파일 시스템의 확장성을 평가하기 위해, 다음 세 가지 버전의 NOVA 파일 시스템을 비교한다. 첫 번째 버전은 기존 NOVA 파일 시스템(NOVA(V))이고, 두 번째 버전은 범위 기반 락을 사용하는 NOVA 파일 시스템(NOVA(RL))이다. 마지막 버전은 본 발명의 실시예에 따른 NUMA 인지적 디자인과 범위 기반 락을 사용하는 NOVA 파일 시스템(NOVA(RL+NUMA))이다.
- [0071] 그리고, 다양한 워크로드 패턴을 생성할 수 있는 FxMark 벤치 마크가 평가에 사용된다. 구체적으로 DWOM, DWOL 및 DRBL의 세 가지 병렬 I/O 워크로드를 사용한다.
- [0072] DWOM 워크로드는 다중 스레드가 공유 파일에 쓰는 다중 스레드 공유 파일 쓰기(N-to-1 쓰기). DWOL 워크로드는 다중 스레드가 개인 파일에 쓰는 다중 스레드 개인 파일 쓰기(N-to-N 쓰기)를 수행한다. 그리고 DRBL 워크로드는 다중 스레드가 파일을 쓴 후 파일을 읽는 다중 스레드 개인 파일 읽기를 수행한다(N-to-N 읽기).
- [0073] 도 6의 (a)에 도시된 바와 같이 UMA 아키텍처에서 DWOM 워크로드를 사용하여 NOVA(RL)의 확장성을 확인하는 반면, NOVA(V)는 노드 별 잠금으로 인해 확장성을 제공하지 않는다. NOVA(V)는 1 개 코어에서 0.8GB/s의 최대 대

역폭을 나타내며 코어 수가 증가하면 약간 감소한다.

- [0074] 반면에 NOVA(RL)는 8 코어에서 2GB/s보다 약간 높은 최대 처리량을 보여주며 최대 28 코어까지 처리량을 유지한다. 이는 확실적인 잠금이 세분화된 잠금으로 대체되어 공유 파일이 병렬로 업데이트 될 수 있기 때문이다. 그러나 파일 로그를 동시에 업데이트 할 수 없으므로, NOVA(RL)의 경우 8개 코어 이후에는 처리량이 확장되지 않는다.
- [0075] 도 6의 (b)는 NUMA 아키텍처에서 DWOM 워크로드를 사용한 결과를 나타낸 예시도이다. NOVA(V) 및 NOVA(RL)는 UMA 환경에서의 결과와 동일한 결과를 보여준다. 이들은 NUMA를 인식하지 않기 때문에 NUMA 기반 Manycore 서버의 성능 향상이 없다.
- [0076] 반면에 NOVA(RL+NUMA)는 CPU 당 로그 및 로컬 우선 쓰기 정책으로 인해 최대 48 코어까지 확장 가능한 성능을 보여준다. NOVA(RL+NUMA)의 처리량은 테스트베드(Testbed)의 NUMA 경계인 28개 코어를 초과하여 48개 코어까지 최대 17.2GB/s로 증가한다.
- [0077] 또한 NOVA(RL+NUMA)는 lock-free 데이터 구조로 인해 NUMA 효과에도 불구하고 NOVA(RL)보다 높은 병렬성을 보여준다. 예를 들어 16개 코어에서 NOVA(RL+NUMA)의 처리량은 NOVA(RL)보다 31% 높음을 알 수 있다.
- [0078] 도 6의 (c)는 NUMA 아키텍처에서 DWOL 워크로드를 사용한 결과를 나타낸 예시도이다. DWOM 성능 향상을 위해 기본적으로 범위 잠금이 해제되어 있으므로 DWOL 워크로드를 사용하는 NOVA(RL)의 성능은 NOVA(V)의 성능과 크게 다르지 않다. 최대 16개 코어까지 확장되지만, 그 이후에는 코어 수가 증가함에 따라 약간 감소한다.
- [0079] 게다가 28 코어 이후의 성능은 NUMA를 인식하지 못하는 파일 배열로 인해 계속 저하되고 있다. 그러나 본 발명의 실시예에 따른 NOVA(RL+NUMA)는 DWOM 워크로드를 사용하는 실험과 유사하게 17.69GB/s의 처리량으로 최대 48개의 코어로 확장되는 것을 알 수 있다.
- [0080] 도 6의 (d)는 NUMA 아키텍처에서 DRBL 워크로드를 사용한 결과를 비교한 예시도이다. 병렬 읽기 측면에서 모든 버전의 NOVA 파일 시스템은 최대 28개의 코어로 확장된다. 그러나 NOVA(V) 및 NOVA(RL)는 28 코어 이후에는 확장되지 않는다.
- [0081] NOVA(V)의 경우, 단일 리더 카운터 문제로 인해 노드 별 잠금이 성능을 저하시킨다. 반면에 NOVA(RL)는 모든 파일이 노드 0에만 배치되기 때문에 NUMA 효과로 인해 잘 수행되지 않는다.
- [0082] 하지만 본 발명의 실시예에 따른 NOVA(RL+NUMA)는 로컬 쓰기 우선 정책을 사용하여 각 스레드의 파일을 로컬 NVM에 배치한다. 이를 통해 스레드는 로컬 메모리 액세스로 파일을 읽을 수 있고, 결과적으로 NOVA(RL+NUMA)는 56 코어까지 확장되며 56 코어에서 49.32GB/s의 최대 대역폭을 보여준다.
- [0083] 이와 같이, 본 발명의 실시예에 따른 NOVA 파일 시스템은 여러 NUMA 노드의 NVM 모듈을 가상화하고 로컬 우선 쓰기 정책을 통해, 병렬 I/O를 수행할 때 발생할 수 있는 원격 액세스 수를 줄였다. 그리고 NOVA 파일 시스템의 노드 별 로그를 잠금 없는 CPU 별 로그로 확장했다. 이에 따라 NUMA 인지적인 NOVA 파일 시스템은 Intel Optane DC Manycore 시스템에서 코어 수가 증가함에 따라 병렬 쓰기를 위해 확장 가능하며, 기존 NOVA보다 19.18 배 높은 쓰기 처리량을 보여준다.
- [0084] 도 7은 본 발명의 실시예에 따른 컴퓨팅 장치의 구조도이다.
- [0085] 도 7에 도시된 바와 같이, 비휘발성 메모리 파일 시스템(100)은 적어도 하나의 프로세서에 의해 동작하는 컴퓨팅 장치(200)에서, 본 발명의 동작을 실행하도록 기술된 명령들(instructions)이 포함된 프로그램을 실행한다.
- [0086] 컴퓨팅 장치(200)의 하드웨어는 적어도 하나의 프로세서(210), 메모리(220), 스토리지(230), 통신 인터페이스(240)를 포함할 수 있고, 버스를 통해 연결될 수 있다. 이외에도 입력 장치 및 출력 장치 등의 하드웨어가 포함될 수 있다. 컴퓨팅 장치(200)는 프로그램을 구동할 수 있는 운영 체제를 비롯한 각종 소프트웨어가 탑재될 수 있다.
- [0087] 프로세서(210)는 컴퓨팅 장치(200)의 동작을 제어하는 장치로서, 프로그램에 포함된 명령들을 처리하는 다양한 형태의 프로세서(210)일 수 있고, 예를 들면, CPU(Central Processing Unit), MPU(Micro Processor Unit), MCU(Micro Controller Unit), GPU(Graphic Processing Unit) 등 일 수 있다. 메모리(220)는 본 발명의 동작을 실행하도록 기술된 명령들이 프로세서(210)에 의해 처리되도록 해당 프로그램을 로드한다. 메모리(220)는 예를 들면, ROM(read only memory), RAM(random access memory) 등 일 수 있다. 스토리지(230)는 본 발명의 동작을 실행하는데 요구되는 각종 데이터, 프로그램 등을 저장한다. 통신 인터페이스(240)는 유/무선 통신 모듈일 수

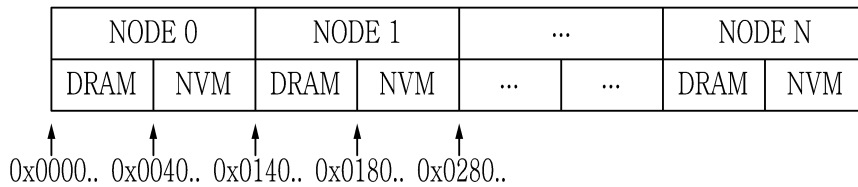
있다.

[0088]

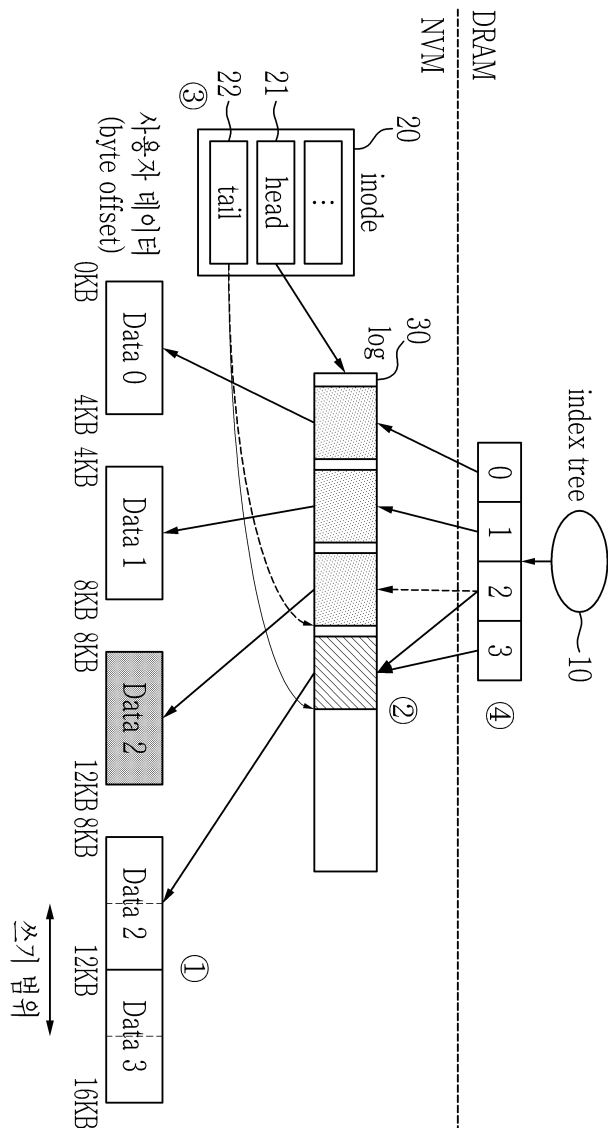
이상에서 본 발명의 실시예에 대하여 상세하게 설명하였지만 본 발명의 권리범위는 이에 한정되는 것은 아니고 다음의 청구범위에서 정의하고 있는 본 발명의 기본 개념을 이용한 당업자의 여러 변형 및 개량 형태 또한 본 발명의 권리범위에 속하는 것이다.

도면

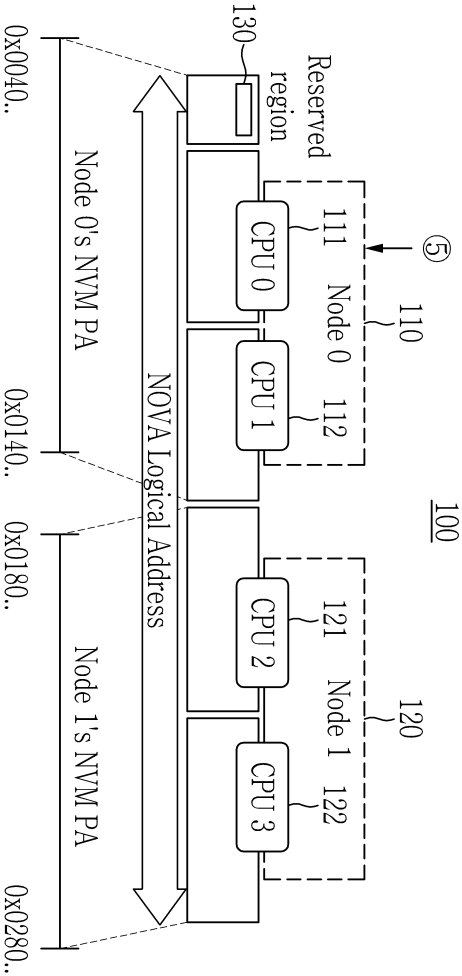
도면1



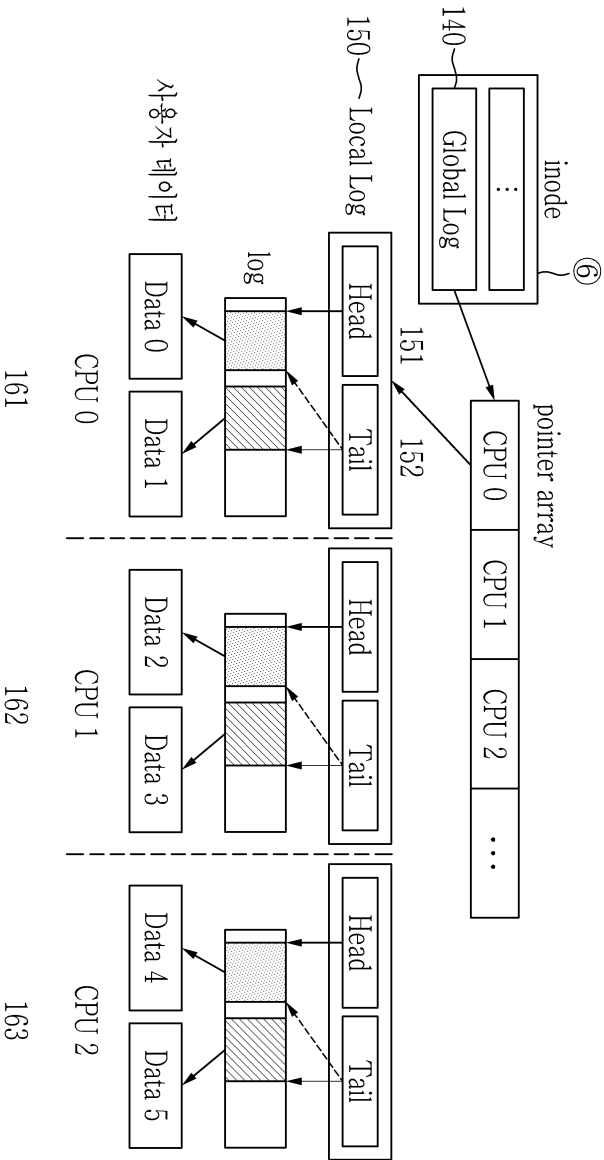
도면2



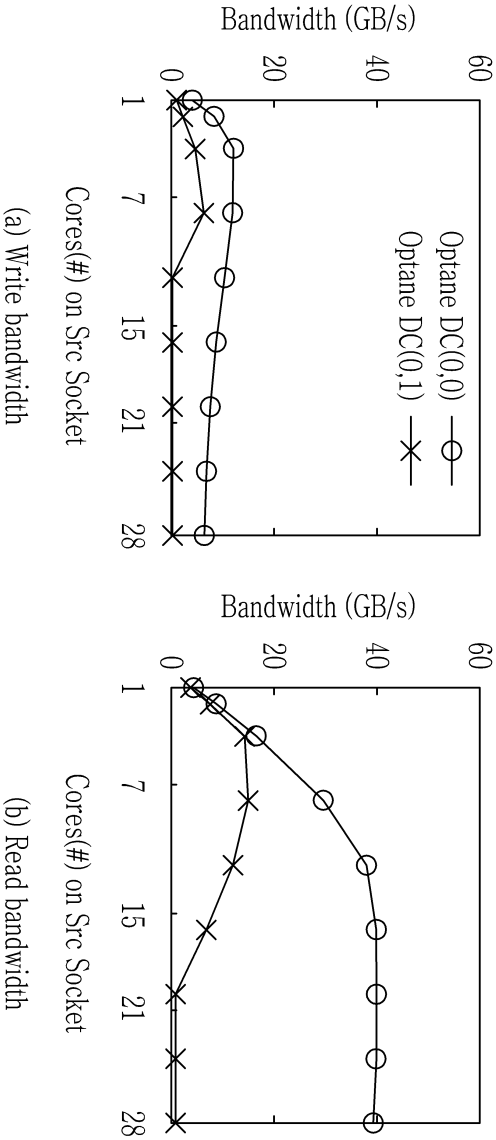
도면3



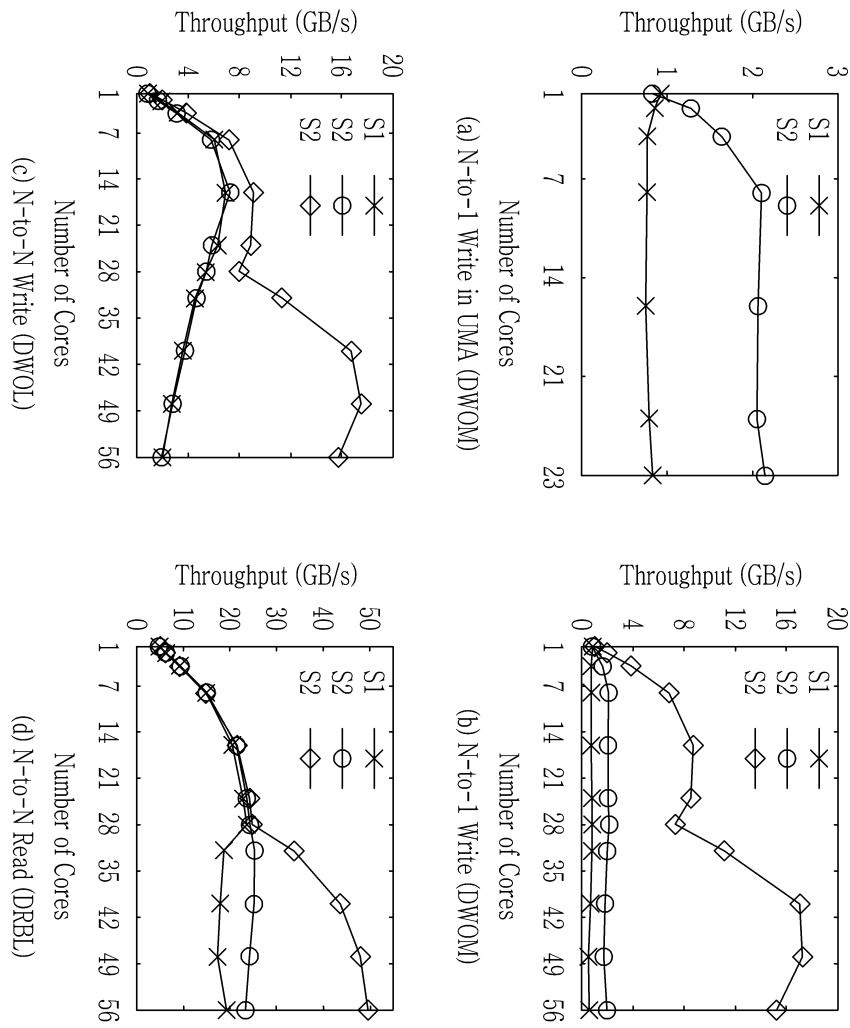
도면4



도면5



도면6



도면7

