

# 목차

|                                               |   |
|-----------------------------------------------|---|
| <b>Ceph : 페타바이트 규모의 Linux 분산 파일 시스템</b> ..... | 3 |
| <b>"Ceph" 이름의 유래</b> .....                    | 3 |
| <b>Ceph의 목표</b> .....                         | 3 |
| <b>Ceph 아키텍처</b> .....                        | 3 |
| <b>Ceph 구성 요소</b> .....                       | 4 |
| <b>Ceph 클라이언트</b> .....                       | 5 |
| <b>Ceph 메타데이터 서버</b> .....                    | 6 |
| <b>Ceph 모니터</b> .....                         | 6 |
| <b>Ceph 오브젝트 스토리지</b> .....                   | 6 |
| <b>기타 흥미로운 기능</b> .....                       | 7 |
| <b>기타 분산 파일 시스템</b> .....                     | 7 |



# Ceph : 페타바이트 규모의 Linux 분산 파일 시스템

출처 : <https://www.ibm.com/developerworks/library/l-ceph/>

## "Ceph" 이름의 유래

"Ceph"는 대부분의 파일 시스템에서 사용하는 일반적인 약어 형태가 아닌 독특한 형태의 파일 시스템 이름이다. 이 이름은 "Sammy"라고 하는 UCSC의 마스코트에서 유래되었으며, 이 마스코트는 두족류(Cephalopods)에 속하는 껍질없는 연체동물인 바나나 슬러그이다(Ceph의 유래). 여러 개의 촉수가 있는 두족류는 분산 파일 시스템에 매우 적합한 상징성을 제공한다.

스토리지 산업의 아키텍트로서 필자는 파일 시스템에 대해 관심을 가지고 있다. 이러한 시스템은 스토리지 시스템에 대한 사용자 인터페이스이며 모든 시스템이 비슷한 기능 세트를 제공하는 하지만 확연히 차별화된 기능을 제공하기도 한다. Ceph도 별반 다르지 않다. 이 기사에서는 파일 시스템에서 찾아볼 수 있는 가장 흥미로운 기능 중 일부를 살펴본다. Ceph는 UCSC(University of California, Santa Cruz)에서 Sage Weil의 스토리지 시스템 관련 박사과정 연구 프로젝트로 시작되었다. 하지만 2010년 3월말부터는 주류 Linux 커널(2.6.34 이상)에서 Ceph를 볼 수 있다. 아직까지는 프로덕션 환경에서 사용하기에 부족한 Ceph이지만 평가 목적으로는 유용하다. 이 기사에서는 Ceph 파일 시스템을 살펴본 후 이 파일 시스템을 확장 가능한 분산 스토리지의 매력적인 대안으로 만들어 주는 고유한 기능을 설명한다.

## Ceph의 목표

분산 파일 시스템 개발은 복잡한 작업이지만 문제점이 올바르게 해결되면 막대한 가치를 제공할 수 있다. Ceph의 목표는 다음과 같이 정의할 수 있다.

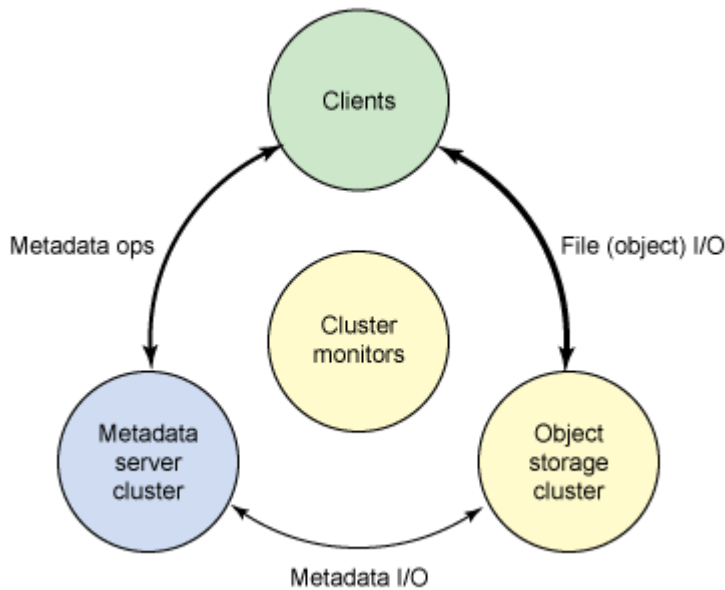
- 용량을 페타바이트 수준으로 손쉽게 확장
- 가변적인 워크로드를 효과적으로 처리할 수 있는 고성능(IOPS[input/output operations per second] 및 대역폭)
- 강력한 신뢰성

아쉽게도 이러한 목표는 서로 경쟁 관계에 놓일 수 있다. (예를 들어, 확장성을 높이면 성능이나 신뢰성이 낮아질 수 있다.) Ceph에는 매우 흥미로운 몇 가지 개념(예: 동적 메타데이터 파티셔닝과 데이터 분배 및 복제)이 있으며, 이 기사에서는 이러한 개념에 대해 간단히 살펴본다. Ceph의 설계에는 단일 장애 지점 문제를 해결하기 위한 내결함성 기능과 페타바이트 수준의 대규모 스토리지 오류가 예외가 아닌 일반적인 오류라는 가정이 통합되어 있다. 마지막으로 이 설계에서는 특정 워크로드를 가정하지 않는다. 대신 변화하는 분산 워크로드에 적절히 대응하여 최상의 성능을 제공할 수 있는 기능이 포함되어 있다. 이 모든 목표는 Ceph에서 제안한 향상된 기능을 통해 POSIX 시맨틱을 사용하는 기존 애플리케이션에 투명하게 배치할 수 있도록 지원하는 POSIX 호환성 목표와도 관계가 있다. 마지막으로 Ceph는 오픈 소스 분산 스토리지이며 주류 Linux 커널(2.6.34)에 포함되어 있다.

## Ceph 아키텍처

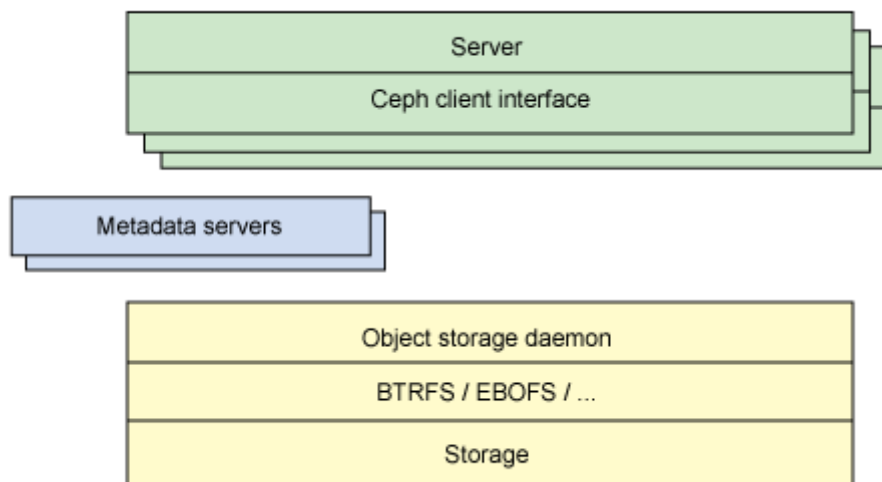
이제 Ceph 아키텍처와 상위 레벨의 핵심 요소를 살펴보자. 그런 다음 하위 레벨로 내려가면서 Ceph의 일부 주요 특성을 식별하고 자세히 살펴보자. Ceph 에코시스템은 크게 네 가지 세그먼트로 분류할 수 있다 (그림 1 참조). 첫 번째는 데이터의 사용자인 클라이언트이며, 두 번째는 분산 메타데이터를 캐시 및 동

기화하는 메타데이터 서버이고, 세 번째는 데이터와 메타데이터를 오브젝트로 저장하고 기타 주요 기능을 구현하는 오브젝트 스토리지 클러스터이며 마지막으로 모니터링 기능을 구현하는 클러스터 모니터가



있다.

그림 1에서 보듯이 클라이언트는 메타데이터 서버를 사용하여 메타데이터 작업(데이터 위치 식별)을 수행한다. 메타데이터 서버는 데이터의 위치와 새 데이터를 저장할 위치를 관리한다. 그리고 메타데이터는 스토리지 클러스터에 저장된다("Metadata I/O"로 표시됨). 실제 파일 I/O는 클라이언트와 오브젝트 스토리지 클러스터 사이에서 발생한다. 이러한 방식으로 열기, 닫기 및 이름 바꾸기와 같은 상위 레벨 POSIX 함수는 메타데이터 서버를 통해 관리되는 반면 읽기 및 쓰기와 같은 POSIX 함수는 오브젝트 스토리지 클러스터를 통해 직접 관리된다. 그림 2에서는 다른 관점에서 바라본 아키텍처를 보여 준다. 서버 세트는 메타데이터 서버와 오브젝트 레벨 스토리지의 관계를 파악하고 있는 클라이언트 인터페이스를 통해 Ceph 에코시스템에 액세스한다. 분산 스토리지 시스템은 스토리지 장치의 형식(Extent 및 EBOFS[B-tree-based Object File System] 또는 대체 형식), 데이터 복제, 장애 감지 및 복구를 관리하기 위해 설계된 대체 관리 계층 및 RADOS(Reliable Autonomic Distributed Object Storage)라는 후속 데이터 마이그레이션을 포함한 몇 개의 계층으로 분류할 수 있다. 마지막으로 모니터는 구성 요소 장애를 식별하고 알

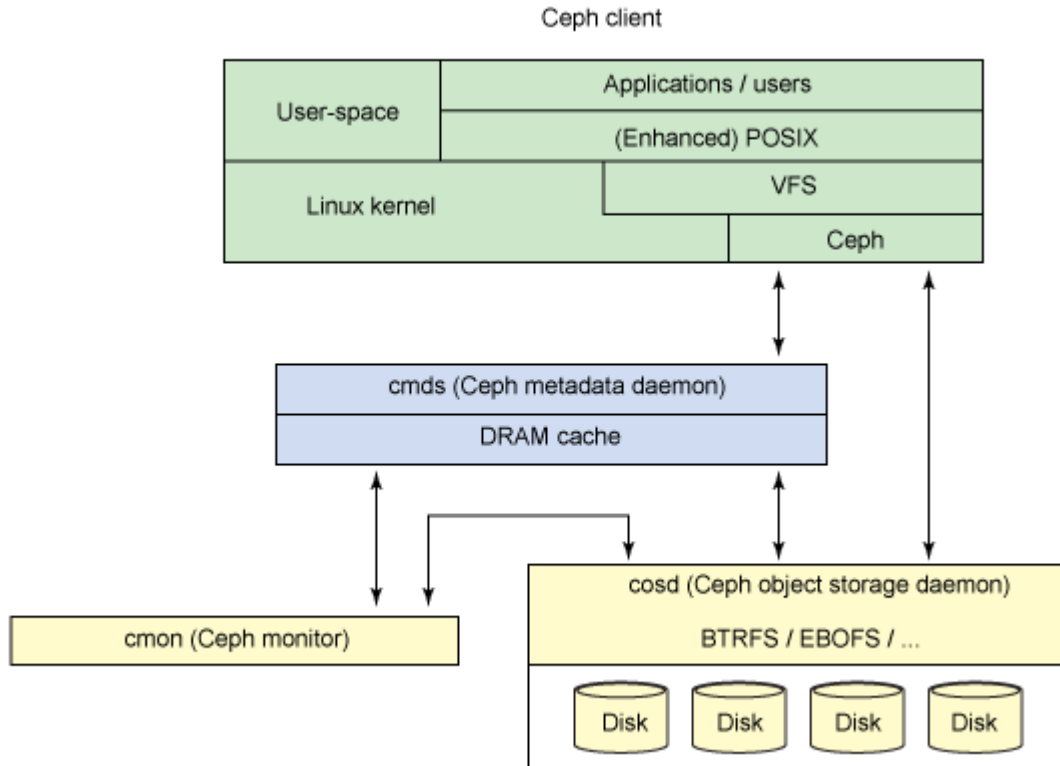


리는 데 사용된다.

## Ceph 구성 요소

지금까지 Ceph의 개념적 아키텍처를 살펴보았으므로 이제 Ceph 에코시스템 내에 구현된 주요 구성 요소에 대해 자세히 알아보자. Ceph와 기존 파일 시스템 사이의 주요 차이점 중 하나는 파일 시스템 자체

의 인텔리전스에 중점을 두지 않고 인텔리전스가 에코시스템 전체에 퍼져 있다는 것이다. 그림 3에서는 간단한 Ceph 에코시스템을 보여 준다. Ceph 클라이언트는 Ceph 파일 시스템의 사용자이다. Ceph 메타데이터 디먼은 메타데이터 서비스를 제공하는 반면 Ceph 오브젝트 스토리지 디먼은 실제 스토리지(데이터 및 메타데이터용)를 제공한다. 마지막으로 Ceph 모니터는 클러스터 관리를 제공한다. 다수의 Ceph 클라이언트, 다수의 오브젝트 스토리지 엔드포인트, 수많은 메타데이터 서버(파일 시스템의 용량에 따라)가 있을 수 있으며 적어도 하나의 중복 모니터 쌍이 있어야 한다. 그렇다면 이 파일 시스템이 어떻게 분산되는 것일까?

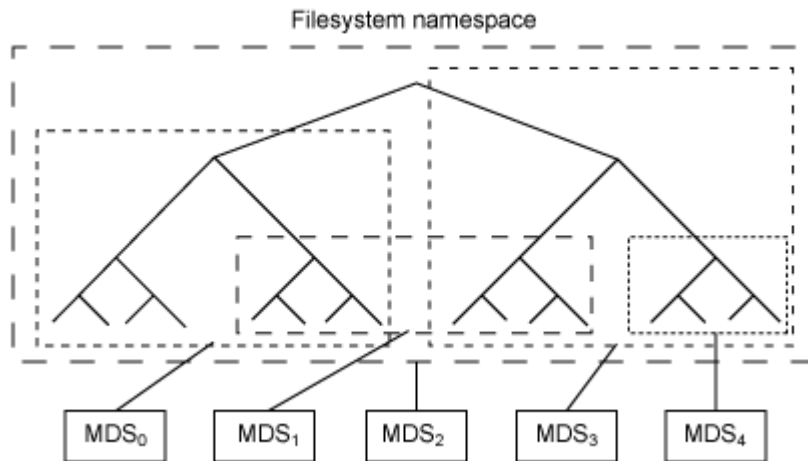


## Ceph 클라이언트

Linux에서 VFS(Virtual File system Switch)를 통해 파일 시스템에 대한 공통 인터페이스를 제공하므로 Ceph에 대한 사용자의 퍼스펙티브는 투명하다. 관리자의 퍼스펙티브는 스토리지 시스템 주위에 많은 서버가 있을 수 있기 때문에 확연히 다르다 (Ceph 클러스터를 작성하는 방법에 대한 자세한 정보는 참고자료 섹션 참조). 사용자의 관점에서 보면 사용자는 대용량 스토리지 시스템에 액세스할 수 있으며 기본 메타데이터 서버, 모니터 및 대규모 스토리지 풀에 통합된 개별 오브젝트 스토리지 장치를 인식하지 못한다. 사용자에게는 표준 파일 I/O를 수행할 수 있는 마운트 지점만 표시된다. Ceph 파일 시스템(적어도 클라이언트 인터페이스)은 Linux 커널에 구현된다. 거의 대부분의 파일 시스템에서는 모든 컨트롤 및 인텔리전스가 커널의 파일 시스템 소스 자체 내에서 구현된다. 하지만 Ceph의 경우에는 파일 시스템의 인텔리전스가 여러 노드로 분산되기 때문에 클라이언트 인터페이스가 단순해질 뿐만 아니라 대규모(및 동적) 확장이 가능하다. Ceph는 할당 목록(디스크의 블록을 지정된 파일에 매핑하기 위한 메타데이터)를 사용하지 않고 흥미로운 대안을 사용한다. Linux 퍼스펙티브의 파일에 메타데이터 서버의 inode 번호(INO)가 지정되며, 이 번호는 파일의 고유 ID이다. 그런 다음 파일은 파일 크기에 따라 몇 개의 오브젝트로 분할된다. INO와 오브젝트 번호(ONO)를 사용하여 각 오브젝트에 오브젝트 ID(OID)가 지정된다. OID에 대한 간단한 해시를 사용하여 각 오브젝트가 배치 그룹에 지정된다. 배치 그룹(PGID로 식별됨)은 오브젝트에 대한 개념적인 컨테이너이다. 마지막으로 오브젝트 스토리지 장치에 대한 배치 그룹의 매핑은 CRUSH(Controlled Replication Under Scalable Hashing)라는 알고리즘을 사용하는 의사 난수 매핑이다. 따라서 스토리지 장치에 대한 배치 그룹의 매핑은 메타데이터 대신 의사 난수 매핑 함수를 사용한다. 이 동작은 스토리지의 오버헤드를 최소화하고 데이터 분배 및 조회를 단순화하기 때문에 이상적이다. 할당을 위한 마지막 구성 요소는 클러스터 맵이다. 클러스터 맵은 스토리지 클러스터를 나타내는 장치를 효율적으로 보여 준다. PGID와 클러스터 맵을 사용하여 모든 오브젝트를 찾을 수 있다

## Ceph 메타데이터 서버

메타데이터 서버(cmds)의 역할은 파일 시스템의 네임스페이스를 관리하는 것이다. 메타데이터와 데이터는 둘 다 오브젝트 스토리지 클러스터에 저장되기는 하지만 확장성을 지원하기 위해 별도로 관리된다. 실제로 메타데이터는 네임스페이스를 적절히 복제 및 분배하여 핫스팟을 방지할 수 있도록 메타데이터 서버 클러스터에 분산된다. 그림 4에서 보듯이 메타데이터 서버는 네임스페이스의 일부를 관리하며 중복성 및 성능을 위해 관리 영역이 서로 겹칠 수 있다. 네임스페이스에 대한 메타데이터 서버의 매핑은 Ceph에서 동적 서브트리 파티셔닝을 사용하여 수행된다. 이 방법을 통해 Ceph는 변경되는 워크로드(메타데이터 서버 간의 네임스페이스 마이그레이션)에 적절히 대응하면서도 성능을 위해 지역성을 유지할



수 있다.

하지만 각 메타데이터 서버는 단순히 클라이언트 집단에 대한 네임스페이스를 관리하므로 기본 애플리케이션은 인텔리전트 메타데이터 캐시이다. (실제 메타데이터는 결국 오브젝트 스토리지 클러스터 내에 저장되기 때문이다.) 작성할 메타데이터는 단기 저널에 캐시된 후 최종적으로 실제 스토리지에 저장된다. 이 동작을 통해 메타데이터 서버는 최근 메타데이터를 클라이언트에게 되돌려 보낼 수 있다(일반적인 메타데이터 작업). 또한 저널은 장애 복구에 유용하다. 메타데이터 서버가 실패할 경우 해당 저널을 재실행하여 메타데이터를 디스크에 안전하게 저장할 수 있다. 메타데이터 서버는 inode 공간을 관리하면서 파일 이름을 메타데이터로 변환한다. 메타데이터 서버는 파일 이름을 Ceph 클라이언트가 파일 I/O를 위해 사용하는 inode, 파일 크기 및 스트라이프 데이터(레이아웃)로 변환한다.

## Ceph 모니터

Ceph에는 클러스터 맵의 관리를 구현한 모니터가 있지만 일부 오류 관리 요소는 오브젝트 저장소 자체에 구현되어 있다. 오브젝트 스토리지 장치가 실패하거나 새 장치가 추가되면 모니터가 이를 감지하고 올바른 클러스터 맵을 관리한다. 이 기능은 맵 업데이트가 기존 트래픽과 통신하는 분산 형식으로 수행된다. Ceph는 분산 환경에서의 합의를 위한 알고리즘의 모음인 Paxos를 사용한다.

## Ceph 오브젝트 스토리지

일반적인 오브젝트 스토리지와 마찬가지로 Ceph 스토리지 노드에는 스토리지뿐만 아니라 인텔리전스도 들어 있다. 일반적인 드라이브는 실행자의 명령에 응답만 하는 단순한 대상이다. 하지만 오브젝트 스토리지 장치는 다른 오브젝트 스토리지 장치와 통신 및 협업할 수 있도록 대상 및 실행자 역할을 둘 다 수행하는 인텔리전트 장치이다. 스토리지 퍼스펙티브에서 Ceph 오브젝트 스토리지 장치는 오브젝트를 블록에 매핑한다(일반적으로 클라이언트의 파일 시스템 계층에서 수행되는 작업). 이 동작을 통해 로컬 엔티티는 오브젝트를 저장할 수 있는 최선의 방법을 결정할 수 있다. 초기 버전의 Ceph에서는 EBOFS라는 로컬 스토리지에 사용자 정의 하위 레벨 파일 시스템을 구현했다. 이 시스템은 오브젝트 시맨틱 및 기타

기능(예를 들어, 디스크에 대한 커미트를 비동기적으로 알리는 기능)에 맞게 조정된 기본 스토리지에 대한 비표준 인터페이스를 구현했다. 오늘날에는 일부 필수 기능(예: 임베디드 무결성)을 이미 구현해 놓은 BTRFS(B-Tree File System)를 스토리지 노드에서 사용할 수 있다. Ceph 클라이언트는 CRUSH를 구현하고 디스크에 있는 파일의 블록 매핑을 모르기 때문에 기본 스토리지 장치에서 블록에 대한 오브젝트의 매핑을 안전하게 관리할 수 있다. 이를 통해 스토리지 노드에서 데이터를 복제할 수 있다(장치가 실패했을 경우). 장애 감지 및 복구가 에코시스템 전체에 분배되기 때문에 장애 복구를 분배하여 스토리지 시스템을 확장할 수 있다. Ceph에서는 이를 RADOS라고 한다(그림 3 참조).

## 기타 흥미로운 기능

마치 파일 시스템의 동적 특성과 적응성이 충분하지 않은 것처럼 Ceph에는 사용자에게 표시되는 일부 흥미로운 기능이 구현되어 있다. 예를 들어, 사용자는 Ceph의 모든 서브디렉토리(모든 내용 포함)에서 스냅샷을 작성할 수 있다. 또한 서브디렉토리 레벨에서 지정된 서브디렉토리(및 모든 중첩 내용)에 대한 스토리지 크기 및 파일 수를 보고하는 파일 및 용량 계산을 수행할 수 있다.

Ceph의 현재와 미래 이제 Ceph가 주류 Linux 커널에 통합되기는 했지만 아직까지는 실험용이라는 표현이 적합한 상태이다. 이 상태의 파일 시스템은 평가하기에는 유용하지만 프로덕션 환경에 사용하기에는 미진한 점이 남아 있다. 하지만 Ceph가 Linux 커널에 채택되었고 Ceph를 개발한 개발자가 지속적으로 개발하려는 열의를 가지고 있기 때문에 머지 않아 대용량 스토리지 요구를 충족하는 Ceph를 볼 수 있을 것이다.

## 기타 분산 파일 시스템

Ceph는 분산 파일 시스템 분야에서는 고유하지 않지만 대용량 스토리지 에코시스템을 관리하는 방법 면에서는 고유하다. 분산 파일 시스템의 몇 가지 예를 들자면 GFS(Google File System), GPFS(General Parallel File System) 및 Lustre가 있다. Ceph를 통해 우리는 분산 파일 시스템의 흥미로운 미래를 짐작할 수 있다. 그것은 바로 규모가 커지면서 대규모 스토리지 문제점을 극복해야 하는 고유한 과제가 눈앞에 다가왔다는 것이다.

이글은 IBM에 M.Tim Jones 라는 저자가 기고한 글이며 2010년도에 작성되었습니다.

<https://www.ibm.com/developerworks/library/l-ceph/>

From:  
<https://atl.kr/dokuwiki/> - AllThatLinux!

Permanent link:  
[https://atl.kr/dokuwiki/doku.php/ceph:%ED%8E%98%ED%83%80%EB%B0%94%EC%9D%84%ED%8A%B8\\_%EA%B7%9C%EB%AA%A8%EC%9D%98\\_linux\\_%EB%B6%84%EC%82%B0\\_%ED%8C%8C%EC%9D%BC\\_%EC%8B%9C%EC%8A%A4%ED%85%9C](https://atl.kr/dokuwiki/doku.php/ceph:%ED%8E%98%ED%83%80%EB%B0%94%EC%9D%84%ED%8A%B8_%EA%B7%9C%EB%AA%A8%EC%9D%98_linux_%EB%B6%84%EC%82%B0_%ED%8C%8C%EC%9D%BC_%EC%8B%9C%EC%8A%A4%ED%85%9C)

Last update: 2017/02/02 02:26

