Analysis and Evolution of Journaling File Systems

2017.07.26

DC Lab

서영근

Content

Introduction

- 파일시스템이 어떻게 작동하는 지 아는 것은 중요하다.
  - ✓ 전통적으로 성능만을 체크하는 방식이 아닌, 보다 전문적인 분석 방법이 필요

- 이를 가능하게 하는 두 가지 방법
  - ✓ SBA(Semantic block-level analysis)
  - ✓ STP(Semantic trace playback)

- 위와 같은 방법을 이용, 여러 design flaw나 성능 문제를 분석할 수 있다.
  - ✓ Ex) tangled synchrony, limit parallelism, eager writing 등 (뒤에서 다룰 예정)

Methodology

- SBA
  - ✓ 파일시스템의 internal behavior와 정책을 알 수 있는 분석 방식
  - ✓블럭 단위의 분석을 한다.
  - ✓ 디스크 트래픽의 양, 읽기 / 쓰기 요청의 block number 등의 정보를 알 수 있다.

- STP
  - ✓파일시스템을 어떻게 바꾸는 것이 성능에 영향을 주는지 체크할 때 좋은 분석 방식
  - ✓SBA block 드라이버가 생성하는 block level trace를 가지고 분석을 한다.

Ext3 file system - SBA

◆ 실험 환경

✓ 3가지 workload에 관해 전체 throughput, journal data, fixed_location, 총 9가지 case를 실험 할 것이다.

✓3가지의 workload는

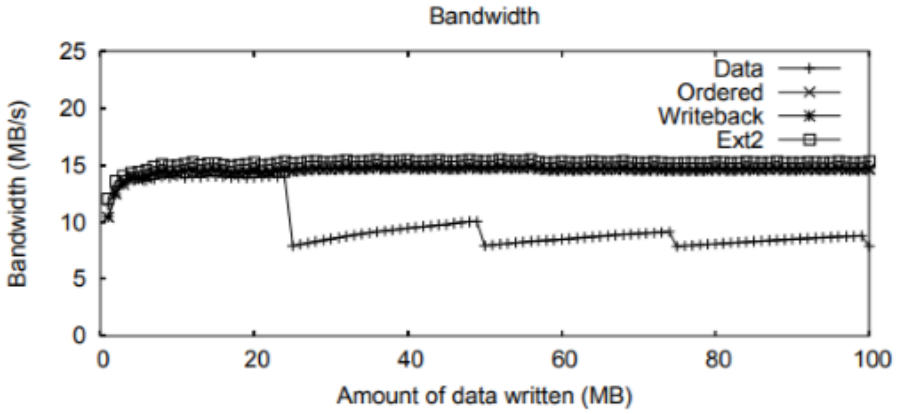
첫째: sequential한 input이 들어오는 케이스,

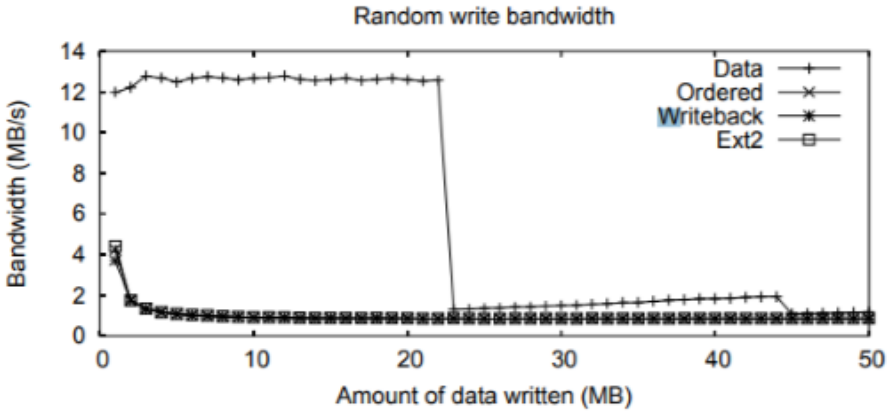
둘째: random input에 대해 256 write 마다 fsync를 부르는 케이스,

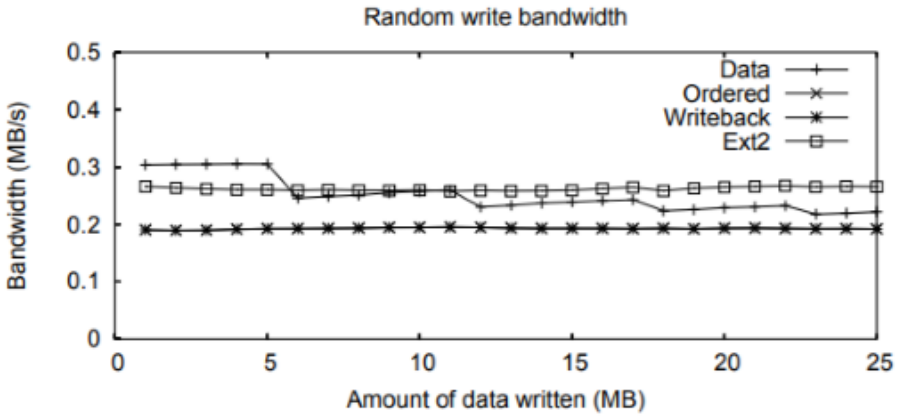
셋째: fsync를 매번 부르는 케이스 이다.

Ext3 file system - SBA

◆1. bandwidth(throughput)







Ext3 file system - SBA

◆1. bandwidth(throughput)

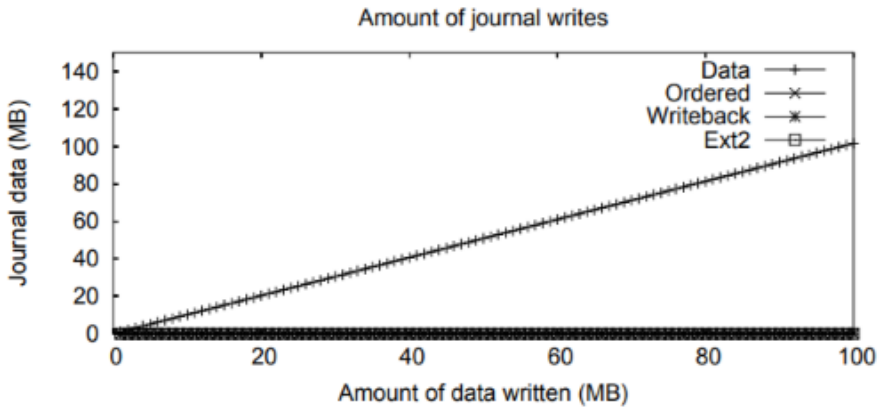
- ✓ Sequential, 256 fsync, all fsync 순으로 전체적인 throughput 이 줄어듬을 알 수 있다.(y축 단위 참조)
- ✓ sequential에서 ext2가 약간이나마 제일 좋은 성능을 보여준다. 그 이유는 저널링에 오버헤드가 존재하기 때문이다.
- ✓ Ordered, writeback 모드는 ext2랑 거의 비슷한 성능을 보인다.
- ✓ 데이터 저널링 기법은 값이 일정하지 않다. -> x축을 기준하여 '톱날' 모양을 하고 있다.

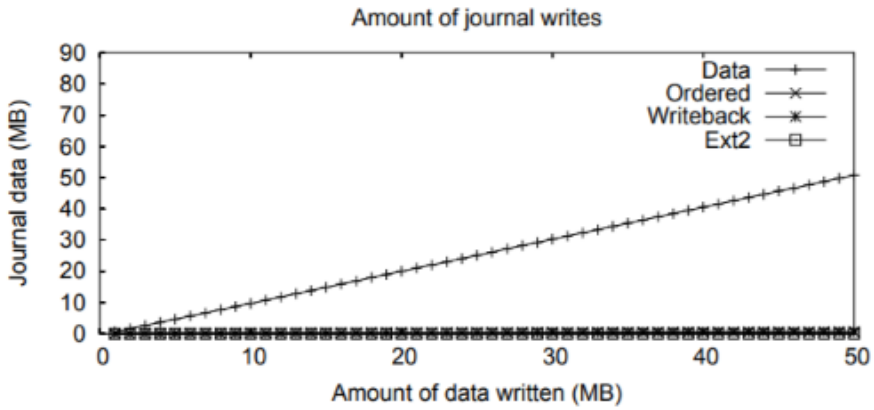
◆ 결론

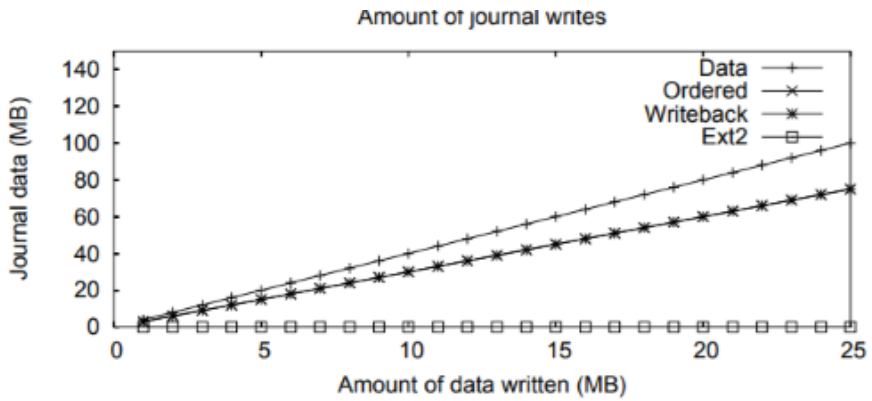
- ✓ 단순히 workload간 성능 비교는 가능하지만, 차이의 이유는 모른다.
- ✓ 이를 알기 위해 journal data, fixed_location을 나눠서 분석해 보는 것!

Ext3 file system - SBA

◆ 2. journal data







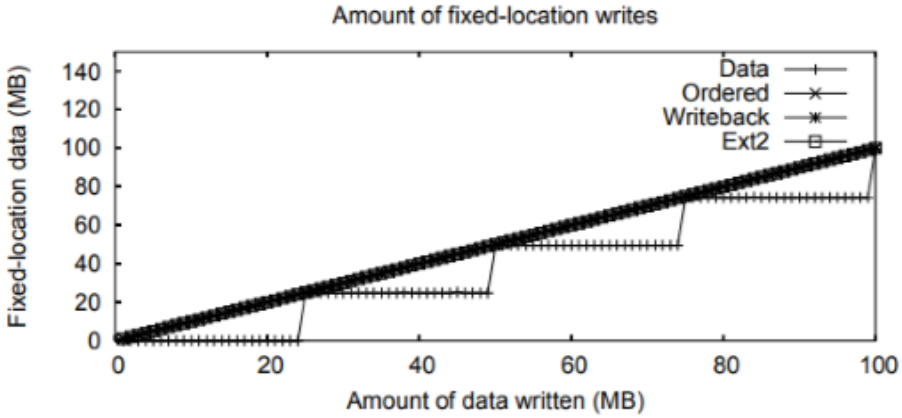
Ext3 file system - SBA

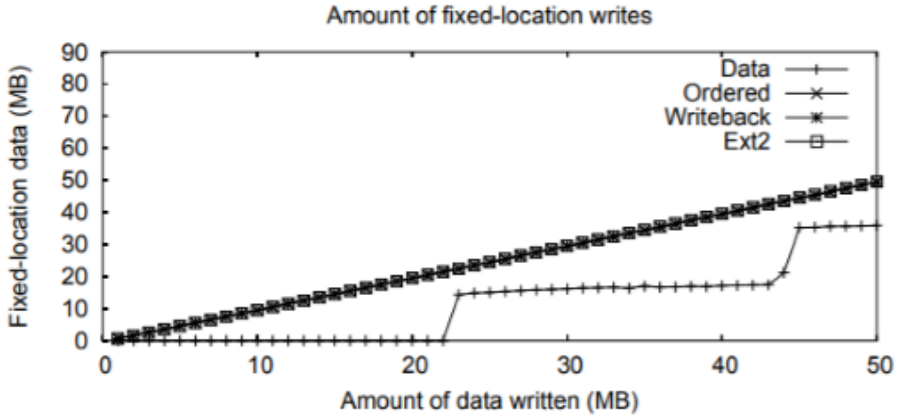
◆ 2. journal data

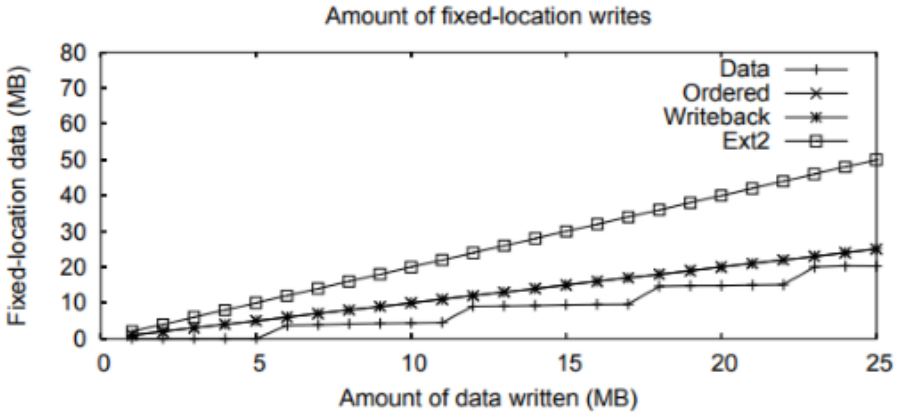
✓ 데이터 저널링 모드는 데이터 까지 저널링 하므로 다른 모드에 비해 저널링 양이 압도적으로 많은 것을 알 수 있다. -> 다른 두 모드는 메타데이타만 저널링한다.

Ext3 file system - SBA

◆ 3. fixed location



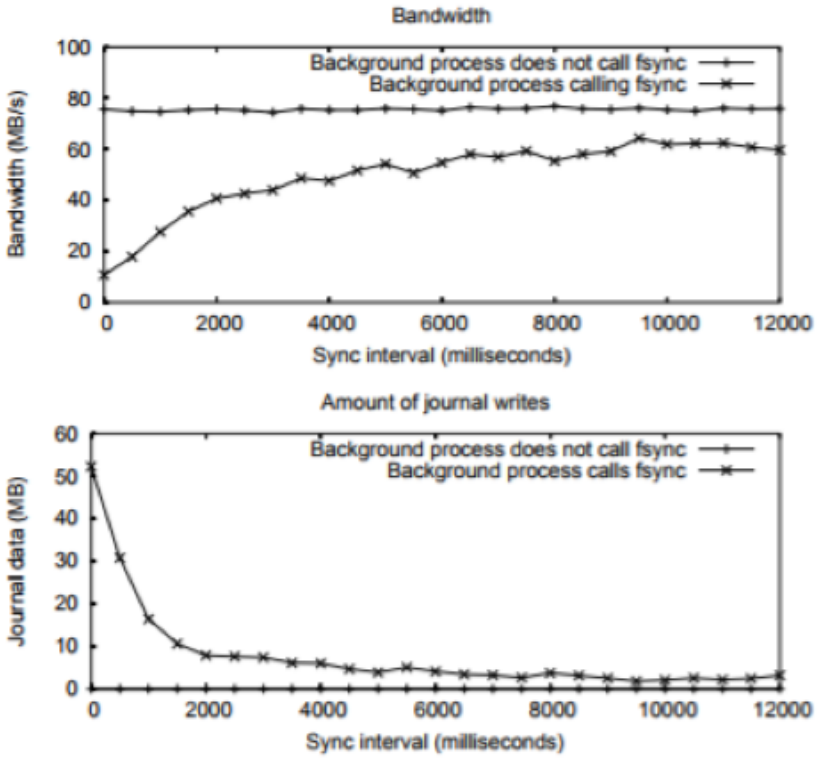




Ext3 file system - SBA

◆ Concurrency

✓ 50mb의 파일을 읽는 foreground 프로세스와 4kb의 블록을 읽는 background 프로세스가 같이 실행되는 환경에서 후자의 프로세스의 fsync 옵션을 바꿔가면서 실험해보자!



◆ Concurrency

✓ 첫번째 그래프를 보면 background 프로세스가 fsync를 하지 않는 다면 성능이 max지만 fsync 를 interval을 짧게 부름에 따라 foreground 프로세스의 성능이 좋지 않음을 알 수 있다.

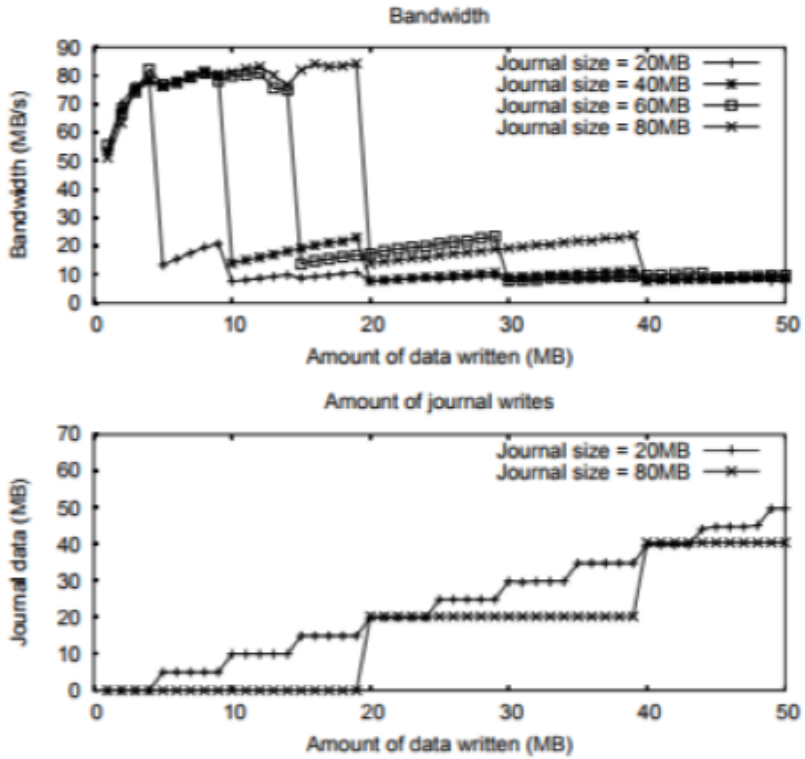
✓ 두번째 그래프에서 set interval이 짧을 수록 저널데이터가 많다는 뜻이다.

✓ 그 데이터는 foreground 프로세스 , background 프로세스 둘다의 데이터라 fsync 시 한 그룹에 두 데이타가 다 들어있으면 비동기도 씽크를 맞춰야 한다. -> tangled

Ext3 file system - SBA

◆ Journal commit policy (journal size)

✓저널 데이터 모드에 집중해서 분석해 보자



✓저널 사이즈의 1/4 지점에서 bandwidth drop 이 일어난다.

✓첫번째 drop이 있기전 bandwidth는 in-memory 속도와 같다.

✓마찬가지로 저널 데이터도 1/4 지점에서 일어난다.

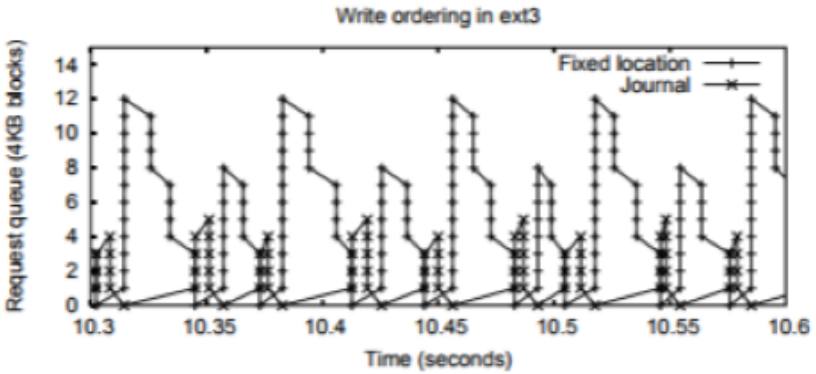
-> fsync 호출

ordered와 writeback도 같은 원리 이다.

Ext3 file system - SBA

◆ Interaction of journal and fixed-location traffic

✓저널에 쓰는 것과 fixed-location에 쓰는 것의 타이밍은 consistency를 위해 철저히 관리되야 한다.



✓저널은 데이타 write가 끝난 다음에 commit 된다.

✓Ordered journaling의 정책을 확인 할 수 있다.

문제점

✓첫번째의 waiting은 필요가 없다.

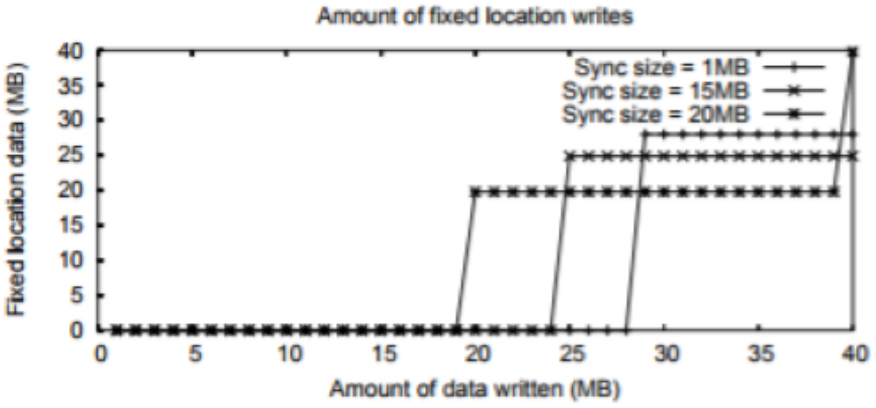
✓저널과 데이터의 device가 구분되어 있다면 필요하지 않다.

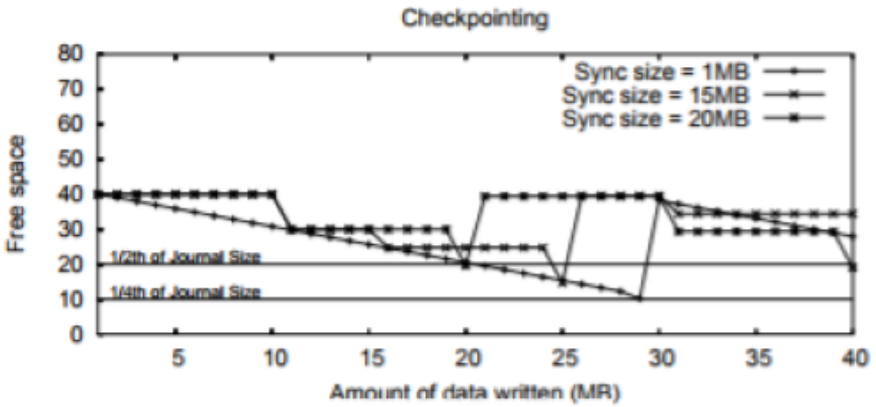
-> falsely limited parallelism

Ext3 file system - SBA

◆ Checkpoint policy (journal size)

✓ 저널 사이즈가 40mb인 환경에서 테스트





✓ 1mb fsync 인 경우 28mb일때 checkpoint

✓ 20mb fsync 인 경우 20mb일때 checkpoint

✓ 두번째 그래프의 경우 free space가 저널 공간의 1/2 에서 1/4 사이에 checkpoint가 일어나는 것을 알 수 있다.

✓ fsync 사이즈가 작을 수록 checkpoint를 연기하여 실행하는 것을 알 수 있다.

Ext3 file system - SBA

◆ Eager writing

- ✓ Ordered mode 랑 data mode는 metadata를 write 하기 전에 data를 먼저 writing하는데, 이를 eager writing 이라 한다.
- ✓ 장점 : data를 자주 flush하므로 disk queue가 크지 않아 전체적인 성능이 향상 된다.
- ✓ 단점 : temporary files 들이 write 될 확률이 높아 I/O 횟수가 늘어난다.

Ext3 file system - SBA

◆ 정리

✓ Best performance 를 내는 저널링 모드는 workload 종류에 따라 다르다.

(ex. random -> data journaling) 그러나 저널 크기나 데이터가 쓰인 양이 더욱 더 큰 영향을 미친다.

✓ Ext3은 compound transaction 에 취약하다. tangled fsync 때문.

✓ Ordered 모드에서 저널 데이타 와 fixed-place 데이타는 겹치지 않고 서로를 기다리는데 처음에는 기다릴 필요가 없다.

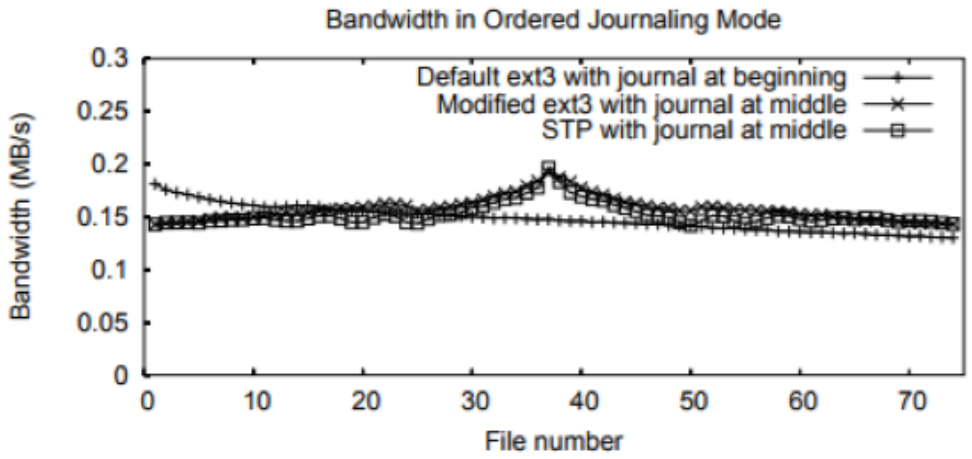
✓ Eager writing : ordered 와 data 저널링에서 일어나는 것, I/O load가 늘어날 가능성이 있다.

Ext3 file system - STP

◆ 원리

✓다양한 변화주기

◆ 첫번째, 저널 location



✓저널 location 위치에 따라 bandwidth가 다르다. 일반적인 ext3은 partition 시작에 저널 공간이 있는데 이를 중간으로 이동하면 ext3의 worst 값을 피할 수 있는 장점이 있다.

Ext3 file system - STP

◆ 두번째, 저널 mode

✓ 원래 저널링 모드는 마운트 단위로 하나의 모드를 이용한다.

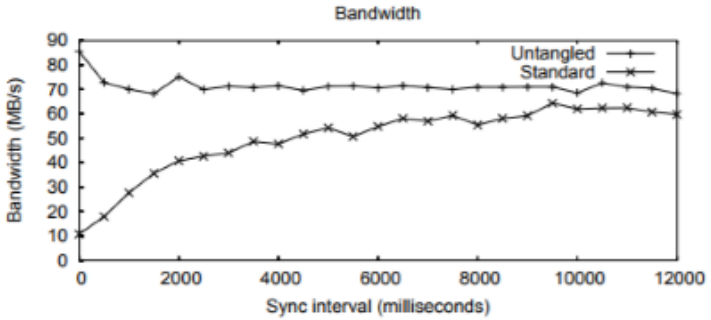
✓ 하지만 성능 향상을 위해 하나의 트랜잭션 마다 그에 맞는 저널링 모드를 적용시켜 보자!

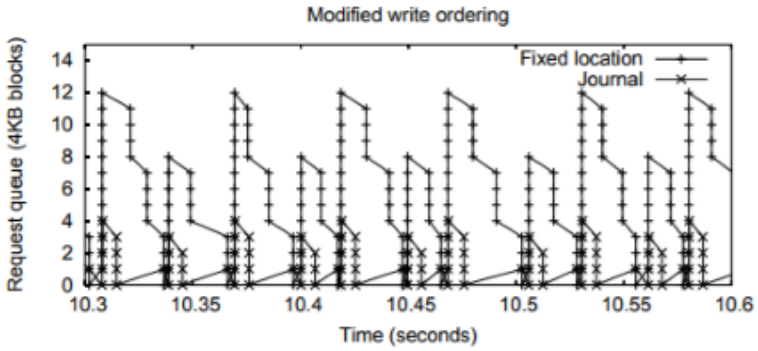
✓ Sequential : ordered // random : data

✓ 그 결과 ordered 모드는 83.39초, data 모드는 86.67초인데 비해 adaptive 저널링은 51.75초 소요 되었다.

Ext3 file system - STP

◆ 세번째, 트랙잭션 grouping





✓궁극적인 목적은 비동기 프로세스가 동기 프로세스의 영향을 받지 않게 하는 것 -> untangle

✓Untangle 상태인 프로세스 in-memrory update가 가능하다.

ReiserFS - SBA

◆ background

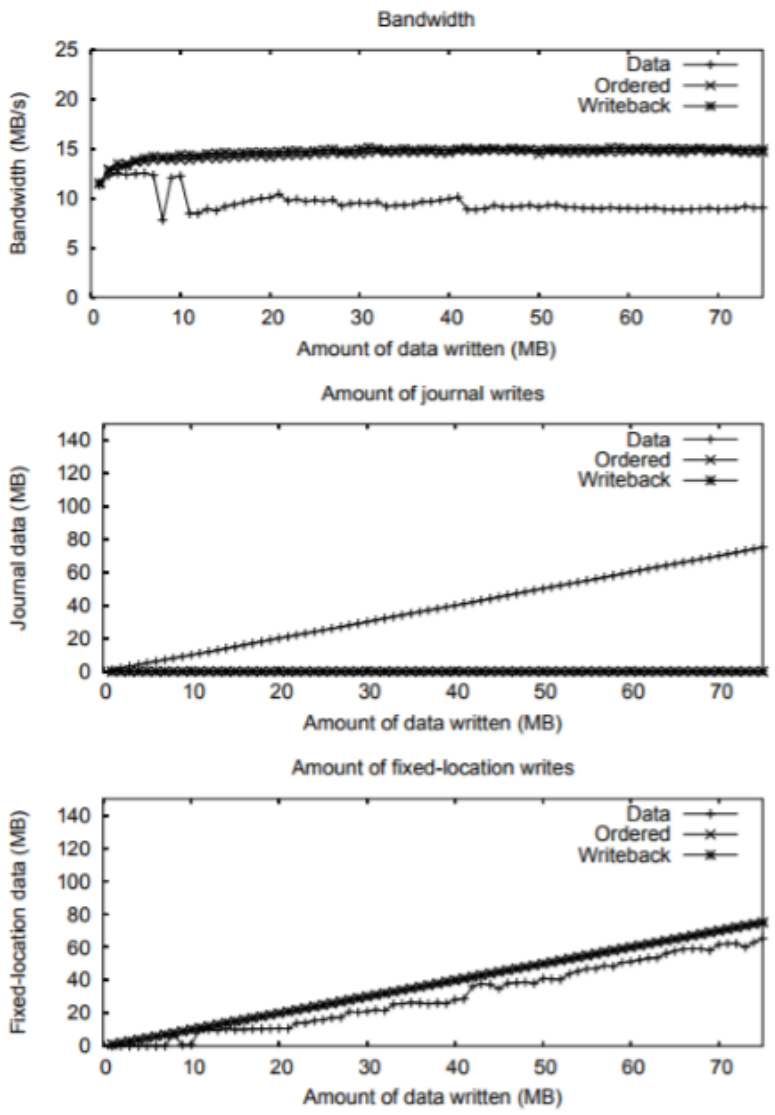
✓ Ext3 와 거의 비슷한 파일시스템이다. (저널링 모드 3개, compound transaction)

✓2가지가 다르다.

1. 데이터 구조로 B+트리 사용
2. 저널의 포맷이 다르다. // ext3 = 파일 // reiserfs = sequence block 최대크기 32mb

ReiserFS - SBA

- Basic behavior
  - ✓ Ext3와 거의 비슷하다.
  - ✓ 차이점이 있다면, data journal에서 톱날 모양의 그래프가 사라졌다.



ReiserFS - SBA

◆ Hournal commit policy

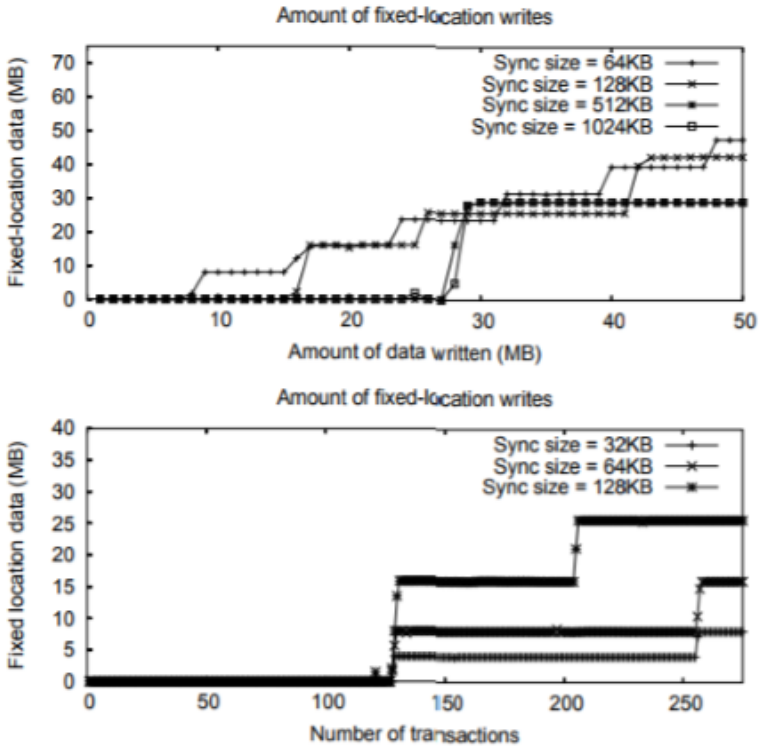
- ✓ Ext3은 데이터 저널링 기준 1/4 의 데이타가 차면 commit 됐었다.
- ✓ ReiserFS는 저널 size가 8mb보다 크냐 작냐에 따라 900block // 450 block 으로 나눠서 생각한다.
- ✓ ReiserFS의 저널 크기는 32mb이므로, 상당히 여유롭게 commit 한다.

ReiserFS - SBA

◆ Checkpoint policy

✓ Ext3 보다 복잡한 구조로, 저널의 남는공간 고려하는 것이 아니고, concurrent transaction 의 number도 고려한다. 7/8이 차면 checkpoint

✓ 두번째 그림에서, 128개 이상의 transcation 이 있으면 checkpoint 된다.



ReiserFS - SBA

◆ Finding bugs

✓ SBA 기법을 이용하다 보면 파일시스템에 정책만 분석하는 것이 아니고 파일시스템이 잘 구현 되었는지도 자연스럽게 알게 된다.

✓ 다음은 논문에서 찾아낸 4가지 버그이다.

• In the first transaction after a mount, the fsync call returns before any of the data is written.
We tracked this aberrant behavior to an incorrect initialization.

• When a file block is overwritten in writeback mode, its stat information is not updated.
This error occurs due to a failure to update the inode's transaction information.

• When committing old transactions, dirty data is not always flushed.
We tracked this to erroneously applying a condition to prevent data flushing during journal replay.

• Irrespective of changing the journal thread's wake up interval, dirty data is not flushed.
This problem occurs due to a simple coding error.

IBM JFS

◆ IBM JFS

✓ 이 파일시스템을 분석할 때 논문에서 코드를 분석하는 것이 아닌 일정 트래픽을 필터해서 시스템에 문제가 있는지 체크하는 기법을 사용하였다.

이를 통해 여러가지 정보를 알아냈는데,

✓ 1. ordered 저널링 모드를 사용한다는 것

✓ 2. logging at the record level, 즉 전체를 로깅하는게 아니고 바뀐 structure만 로깅한다.

✓ 3. 그룹 concurrent update를 이용하지 않는다. 동기랑 비동기랑 bandwidth 가 달라진다.

✓ 4. 저널 write는 timer에 의해 일어나지 않고 다른 방법을 통해서만 된다. 즉, infinite write delay가 생길 가능성이 있다. 이는 reliability에 좋지 않다.

Windows NTFS

◆ Windows NTFS

- ✓ 윈도우의 기본 파일 시스템
- ✓ ntfs의 모든 오브젝트는 file이다. metadata, journal 등 전부다 파일
- ✓ 저널은 파일시스템 중간에 위치한다.
- ✓ 데이터 저널링을 하지않고, 블록레벨 저널링을 하지 않는다. JFS 랑 같음
- ✓ Ordered 저널링을 한다.

감사합니다