제목: SSD IO 성능 테스트 – IODrive

작성자 : 송혁(hyok81@nate.com)

얼마 전 IODrive를 지엔오테크롤로지의 협찬을 받아서 일주일 동안 IO관련 테스트를 진행하였습니다.

서버용 제품으로, 벤더에서 제공하는 스펙만 본다면 기존에 테스트 했던 버텍스 또는 엠트론 보다 월등히 뛰어난 IOPS 수치를 보여주고 있습니다.
또한 특이한 것은 다른 SSD와 다르게 SATA 인터페이스가 아닌 PCI-E 인터페이스를 채택 하였습니다.
역시 서버용 제품이라, 가격은 생각보다 비쌌지만 서버에서 사용하는 엔터프라이즈 SAS디스크의 가격도 수십 만원에서 수백 만원 정도하기에,
적은 공간에 아주 많은 IOPS를 요구하는 비즈니스라면 일반 DISK를 사용하는 것 보다 적은 비용으로 구현 할 수 있을 것 입니다.

과연 어느정도의 IOPS 및 응답시간을 보여줄 지 매우 궁금하였습니다.
기존 엠트론 또는 버텍스 제품의 경우 랜덤 쓰기 작업이 읽기 작업 대비 현저히 느린 결과치를 보여주었습니다.
이러한 문제점이 랜덤쓰기가 많은 DBMS에서 메리트를 많이 상쇄 시켰습니다.

이번에 테스트한 제품은 IODrive 80GB 입니다.
스펙 상으로는 읽기, 쓰기에 대략 10만 IOPS을 제공해 준다고 하며, 최대 bandwidth는 읽기 700MB/s, 쓰기는 550MB/s 을 정도 입니다.
이 정도의 Bandwidth는 일반 서버용 DISK 몇 장만 RAID를 구성하면 충분히 뽑아 낼 수 있기에
비싼 가격과 적은 용량을 가진 SSD만의 작업이라고 볼 수 없습니다.

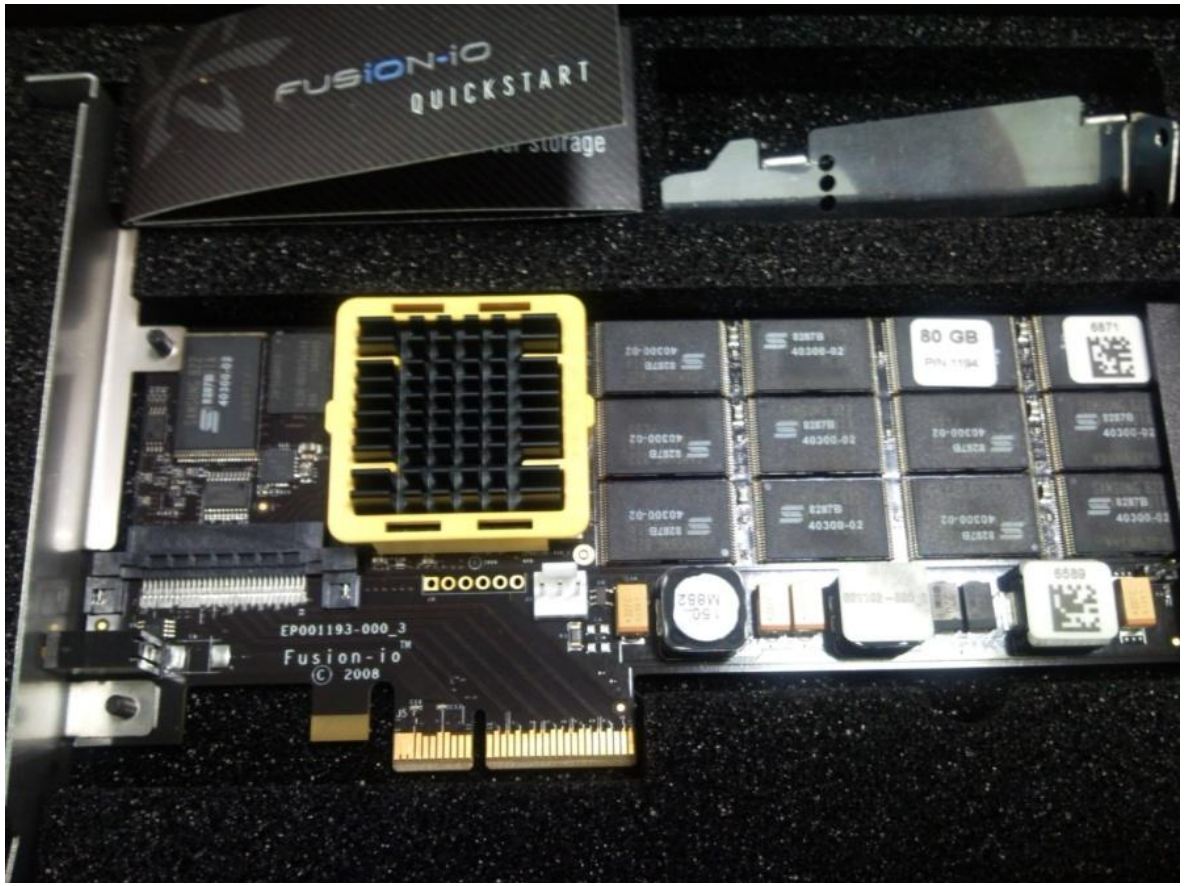
하지만 랜덤 IOPS에서는 이야기가 달라집니다.
보통 15K rpm 엔터프라이즈 SAS 인터페이스 DISK도 300~400정도의 IOPS를 보여주고 있기에,
IODrive가 제공하는 IOPS만큼을 디스크로 구성 하기 위해서는 대략 250장 정의 DISK가 필요하게 됩니다.
이렇게 비교해 보면 SSD의 랜덤 IOPS가 얼마나 많은 IOPS를 제공해 주는지 알 수 있습니다.

[ioDrive 스펙]

ioDrive Capacity	80GB	160GB	320GB
NAND Type	Single Level Cell (SLC)	Single Level Cell (SLC)	Multle Level Cell (MLC)
Write Bandwith	550 MB/s (random 16K)	600 MB/s (random 16K)	500 MB/s (random 16K)
Read Bandwith	700 MB/s (random 16K)	700 MB/s (random 16K)	700 MB/s (random 32K)
IOPS*	102,000(random 4K reads)	104,400(random 4K reads)	60,000(random 4K reads)
	91,000(random 4K writes)	103,925(random 4K writes)	79,000(random 4K writes)
	88,000(70/30 random 4K mix)	95,000(70/30 random 4K mix)	65,000(70/30 random 4K mix)
Access Latency	50μs Read	50μs Read	80μs Read
Bus Interface	PCI-Express x4		
Weight	Less than 2 ounces		
Operating System	RHEL 4 & 5 , SLES 9 & 10, Microsoft 64-Bit Windows**		
Operating System Sophisticated ECC (@5-TB write-erase/day)	24yrs	48yrs	16yrs

** Performance data provided by Medusa Lab. **64-Bit Windows XP, Vista, Server 2003 & 2008*

[IODrive 모습]



➢ 테스트 시작

처음 IO테스트를 진행할 때 80GB제품 두 개를 RAID0으로 묶어서 테스트할 예정이었으나,
하드웨어 문제가 있어 부득이하게 싱글로 테스트를 진행하였습니다.

간단한 테스트한 머신 스펙은 아래와 같습니다.

- CPU : Intel Xeon E54xx * 2 (Quad-Core)
- MEMORY: 2GB

테스트는 총 4가지로 테스트 하였습니다.

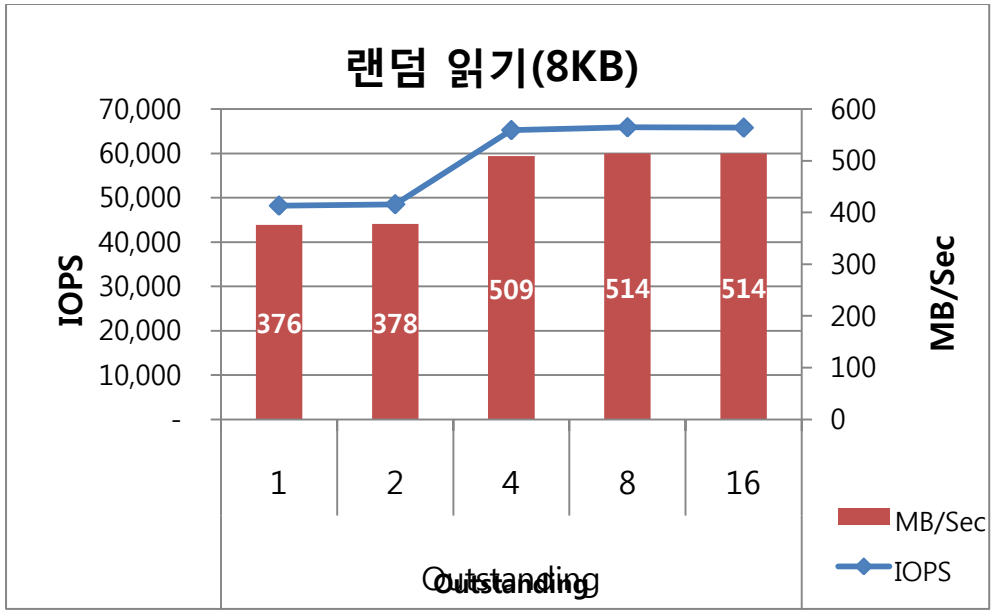
1. SQLIO
2. IOMeter
3. SQL Server – Index Scan vs Index Lookup
4. SQL Server – 물리적 조각화에 따른 SCAN성능 비교

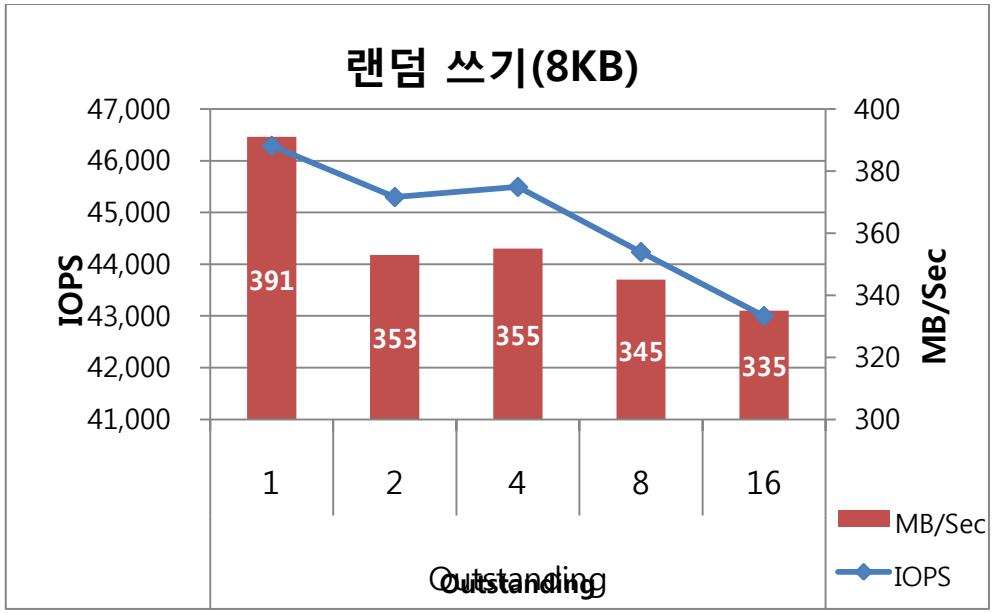
➢ 테스트 1 : SQLIO

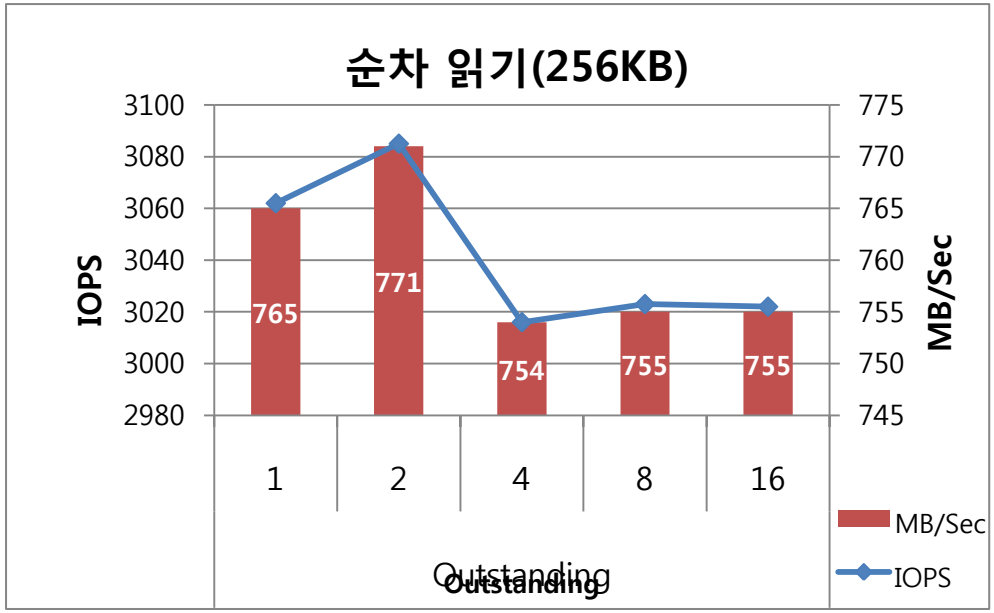
SQLIO는 단일형태의 IOPS를 산정할 수 있는 도구 입니다.
SQL Server에서 주로 사용되는 8KB랜덤, 256KB 순차 읽기/쓰기로 테스트를 진행하였습니다.
간단히 결과를 알아본다면 랜덤 읽기의 경우 대략 65,000정도 / 쓰기 46,000정도의 IOPS를 보여주고 있습니다.
대략 초당 전송속도(Bandwidth)로 환산해 본다면 대략 500MB정도입니다.

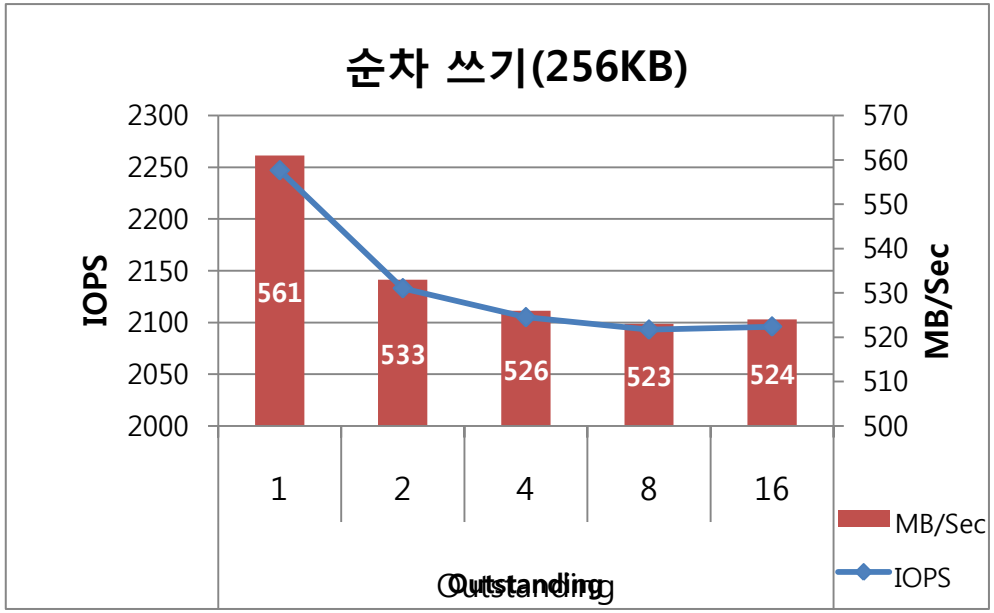
이정도의 수치라면 기존 엔터프라이즈 디스크와 비교하면 읽기의 경우 대략 220배 정도 쓰기의 경우 150배 정도 많은 IOPS를 제공해 주고 있습니다.
아래 표에서는 보여지지 않지만 대부분의 IO응답시간은 1ms이하로 매우 빠른 속도를 보여 주었습니다.

혹시 위 벤더에서 제공하는 스펙과 차이가 많이 난다고 생각하신다면, 위 테스트는 4K단위로 테스트 하였고, 여기서는 8K단위 테스트 했기 때문입니다.


랜덤 읽기(8KB)
IOPS
MB/Sec
70,000
60,000
50,000
40,000
30,000
20,000
10,000
-
600
500
400
300
200
100
0
376
378
509
514
514
1
2
4
8
16
Outstanding
MB/Sec
IOPS


랜덤 쓰기(8KB)
IOPS
MB/Sec
47,000
46,000
45,000
44,000
43,000
42,000
41,000
400
380
360
340
320
300
391
353
355
345
335
1
2
4
8
16
Outstanding
MB/Sec
IOPS


순차 읽기(256KB)
IOPS
MB/Sec
3100
3080
3060
3040
3020
3000
2980
775
770
765
760
755
750
745
765
771
754
755
755
1
2
4
8
16
Outstanding
MB/Sec
IOPS


순차 쓰기(256KB)
IOPS
MB/Sec
2300
2250
2200
2150
2100
2050
2000
570
560
550
540
530
520
510
500
561
533
526
523
524
1
2
4
8
16
Outstanding
MB/Sec
IOPS

➢ 테스트 2 : IOMETER

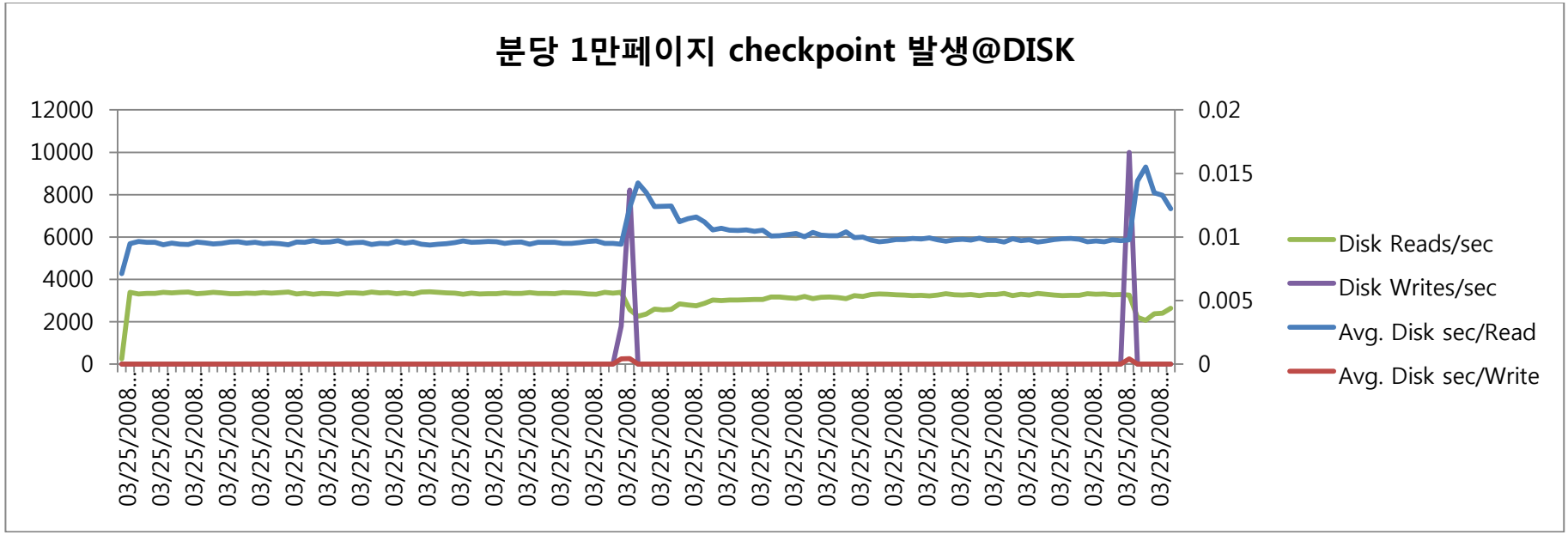
SQL Server를 사용하는 서비스에 대해 IO-Sub System에 대한 성능 측정을 해야한다면,
SQL Server를 기반으로 테스트를 진행하면 가장 효과적이고 정확한 테스트 결과를 볼 수 있을 것입니다.
하지만 SQL Server 의 IO패턴을 자유자재로 조작할 수 없기에 이렇게 테스트 쉬운 작업이 아닐 것입니다.

이번에 테스트하는 IOMETER는 여러 가지 IO패턴을 만들 수 있는 기능이 있어 SQL Server의 IO패턴을 비슷하게 흉내 낼 수 있습니다.
그래서 이번에는 SQL Server의 IO패턴을 비슷하게 구현하여 IO 성능 테스트를 진행하였습니다.

우선 테스트를 하기 전 SQL Server의 IO패턴에 대해서 알아볼 필요가 있습니다.
SQL Server의 IO패턴은 트랜잭션 로그를 제외하면 IO 패턴은 크게 두 가지로 볼 수 있습니다.

- ◆ Buffer Pool(이하 BP)에 없는 데이터를 읽기 위한 Read IO
- ◆ BP에서 디스크로 플러시 되지 않은 Dirty page를 디스크로 쓰는 checkpoint프로세스

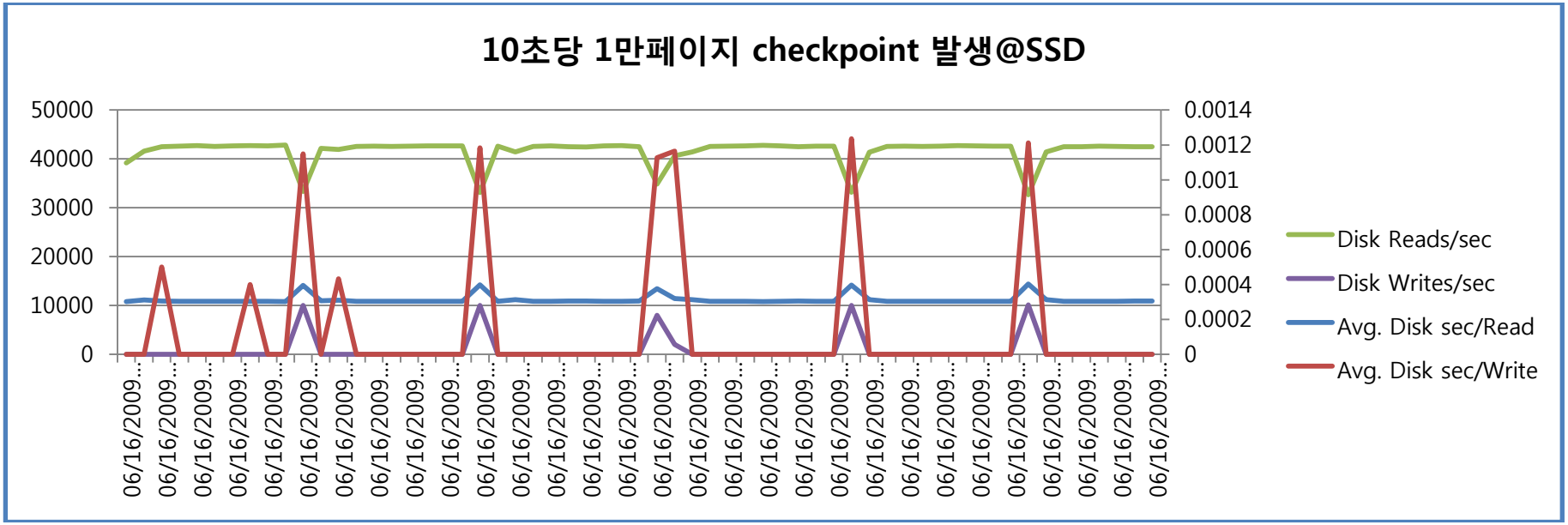
[디스크에서 분당 만 페이지씩 쓰기 작업]



우선 디스크에서 테스트한 결과를 살펴보면,
테스트 환경은 15K SAS 디스크 10장 (RAID10) 구성되었으며, 분당 만 페이지씩 쓰기 IO가 발생하게 됩니다.

간단히 그래프를 살펴보면 평균 읽기는 대략 3700쯤 되다가 쓰기IO가 수행되면 읽기 IO가 대략 2000정도 까지 떨어지는 현상을 보입니다.
쓰기 캐시가 장착된 모델이라 쓰기 시점에 쓰기 응답시간은 매우 빠르나 읽기 응답시간은 느려지는 것을 확인 할 수 있습니다.

[SSD에서 10초당 만 페이지씩 쓰기 작업]



SSD 에서의 테스트의 경우 위에서 테스트한 SQLIO로 어느정도의 수치를 보여주었고, 생각 이상의 IOPS를 제공해 주었기에 위 디스크와 다르게 10초당 만 페이지씩 쓰기 작업을 하였습니다.
간단히 그래프를 살펴보면 평균읽기는 42000정도 되다가 쓰기IO가 수행되면, 읽기IO가 대략 33000정도 까지 떨어지는 현상을 보입니다.
PCI-E 인터페이스 이고 제품에도 따로 Write Back 캐시 역할을 해 줄 수 있는 것이 없기에, 쓰기IO 작업 시 위와 디스크에서 테스트한 결과와 다르게 쓰기 응답시간이 느려지는 것을 볼 수 있습니다. 하지만 느려져도 1ms정도이기에 문제가 있는 수치라고 볼 수 없습니다.
또 하나 위 디스크와 다른 부분은 읽기 응답시간입니다.
쓰기 IO가 발생해도 꾸준히 대략 0.4ms 이하 수준을 보여주고 있습니다.

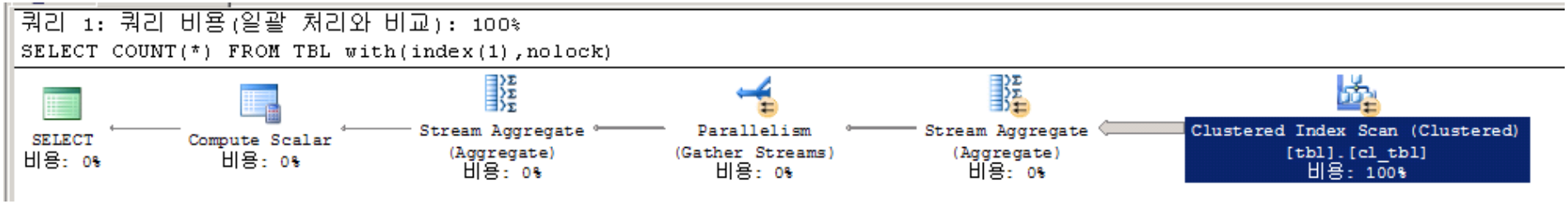
➢ 테스트 3 : SQL Server – Index Scan vs Index Lookup

대량의 데이터를 읽어야 하는 경우라면, 인덱스를 룩업하는 것보다 인덱스를 스캔하는 것이 빠르며, 그 속도 차이는 수배에서 수십배 빨랐습니다.
SSD는 위 테스트에서 살펴보았듯이, 많은 랜덤 IOPS를 제공해 주고 있으며, 응답시간 또한 기존디스크에서 볼 수 없는 속도를 내주게 됩니다.
그렇다면 SSD환경에서 모든 데이터를 읽어야 하는 경우에 룩업으로 처리시 어느 정도 성능을 보여줄지 궁금합니다.

아래 테스트는 7GB정도의 데이터를 입력 후 인덱스 힌트를 사용하여 인덱스 스캔과 인덱스 룩업에 대해 응답시간을 비교하였습니다.

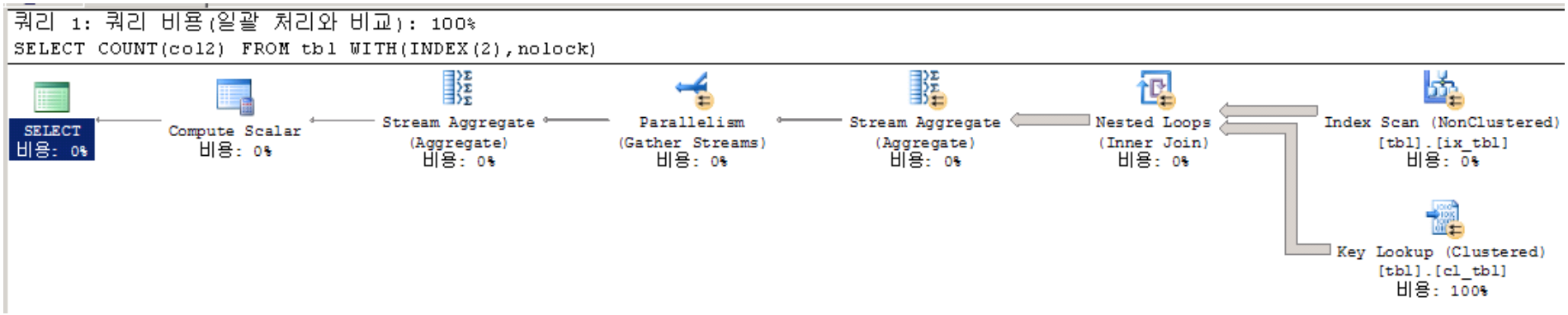
간단히 결론을 알아 본다면 SCAN작업은 10초, Lookup작업은 15초 정도에 끝났습니다.
역시나 SCAN이 Lookup보다 빨랐습니다. 하지만 시간이 1.5배 정도밖에 차이가 나지 않은 점을 주목해야 할 것 입니다.

```
SELECT COUNT(*) FROM TBL with(index(1),nolock)
/*
테이블'tbl'. 검색수9, 논리적읽기수1002491, 물리적읽기수14078, 미리읽기수1002490
, LOB 논리적읽기수0, LOB 물리적읽기수0, LOB 미리읽기수0.

SQL Server 실행시간:
CPU 시간= 2485밀리초, 경과시간= 10404밀리초
*/
```




```
SELECT COUNT(col2) FROM tbl WITH(INDEX(2),nolock)
  /*
  테이블'tbl'. 검색수9, 논리적읽기수12043916, 물리적읽기수2790, 미리읽기수1006198
```

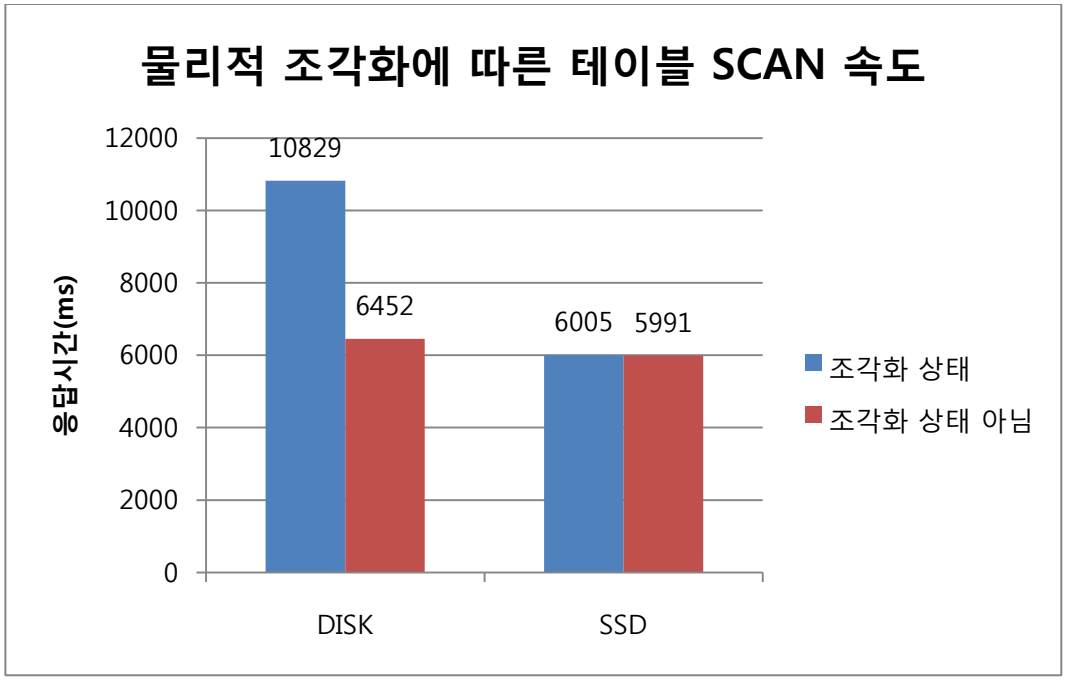
```
, LOB 논리적읽기수0, LOB 물리적읽기수0, LOB 미리읽기수0.
테이블'Worktable'. 검색수0, 논리적읽기수0, 물리적읽기수0, 미리읽기수0
, LOB 논리적읽기수0, LOB 물리적읽기수0, LOB 미리읽기수0.

CPU 시간= 31797밀리초, 경과시간= 15898밀리초
*/
```



➢ 테스트 4 : SQL Server – 물리적 조각화에 따른 SCAN성능 비교

예전에 테스트한 물리적 조각화에 따라 SCAN 성능이 SSD환경에서는 어떨지 궁금해서 테스트 하였습니다.
SSD는 기존의 디스크와 다르게 어떠한 데이터를 읽더라도 모두 동일한 시간에 읽을 수 있습니다.
하지만 기존 DISK는 물리적인 회전을 통해 데이터를 읽어야 하기에 데이터가 쓰여진 위치에 따라 응답시간이 차이가 발생하였습니다.
그래서 같은 파일을 인접한 곳에 저장하는 조각모음이 필요했었습니다.
하지만 SSD는 데이터가 어디에 있든 모두 동일한 시간으로 데이터를 가져올 수 있기에 따로 조각모음이 필요없다고 합니다.

디스크 환경에서 테스트한 스크립트를 이용하여 SSD에 동일하게 테스트를 진행하였습니다.
아래 수치를 보면 SSD는 물리적 조각화에 의해 전혀 성능적 영향을 받지 않았다고 볼 수 있습니다.
어느정도의 차이는 있지만, 테스트 환경에 따라 이정도의 오차는 충분히 무시할 수 있는 수치로 보여집니다.
SSD를 사용하면 더 이상 물리적 조각화로 인해 조각모음은 필요 없어 보입니다.
뭐. SSD에 물리적 조각화라는 말이 안어울리죠?




```
USE [master]
GO

CREATE DATABASE [TEST] ON  PRIMARY
( NAME = N'TEST', FILENAME = N'F:₩DBData₩TEST.mdf' , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
LOG ON ( NAME = N'TEST_log', FILENAME = N'D:₩DBlog₩TEST_log.ldf' ,  MAXSIZE = 2048GB , FILEGROWTH = 10%)

CREATE DATABASE [TEST1] ON  PRIMARY
( NAME = N'TEST1', FILENAME = N'F:₩DBData₩TEST1.mdf' , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
LOG ON ( NAME = N'TEST1_log', FILENAME = N'D:₩DBlog₩TEST1_log.ldf' , MAXSIZE = 2048GB , FILEGROWTH = 10%)

CREATE DATABASE [TEST2] ON  PRIMARY
( NAME = N'TEST2', FILENAME = N'F:₩DBData₩TEST2.mdf' ,SIZE=5GB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
LOG ON ( NAME = N'TEST2_log', FILENAME = N'D:₩DBlog₩TEST2_log.ldf' , SIZE = 5GB , MAXSIZE = 2048GB , FILEGROWTH = 10%)
```

```
DECLARE @I VARCHAR(100)
SET @I= '4'
WHILE 1=1
BEGIN
      EXEC ('ALTER DATABASE TEST MODIFY FILE(NAME = ''TEST'',SIZE='+@I+')')
      EXEC ('ALTER DATABASE TEST1 MODIFY FILE(NAME = ''TEST1'',SIZE='+@I+')')

      SET @I=@I+1
IF @I=5000    BREAK
END



/*
          테스트를위해4000만건대략5GB  정도의데이터를입력한다.
*/
SELECT TOP 40000000
   row_number() over(order by (select 1)) as col1
   ,cast('' as char(100)) col2
INTO test..tbl
FROM master.dbo.spt_values A,master.dbo.spt_values A1,master.dbo.spt_values A2,master.dbo.spt_values A3

SELECT TOP 40000000
   row_number() over(order by (select 1)) as col1
   ,cast('' as char(100)) col2
INTO test2..tbl
FROM master.dbo.spt_values A,master.dbo.spt_values A1,master.dbo.spt_values A2,master.dbo.spt_values A3
```

```
exec sp_msforeachtable 'checkpoint 1'
dbcc dropcleanbuffers

SET STATISTICS TIME ON
SET STATISTICS IO ON

-- 조각화 상태
SELECT COUNT(*) FROM test.dbo.tbl
테이블 'tbl'. 검색 수 9, 논리적 읽기 수 579712, 물리적 읽기 수 7258, 미리 읽기 수 579711, LOB 논리적 읽기 수 0, LOB 물리적 읽기 수 0, LOB 미리 읽기 수 0.
CPU 시간 = 10359밀리초, 경과 시간 = 6005밀리초

-- 조각화 상태 아님
SELECT COUNT(*) FROM test2.dbo.tbl
테이블 'tbl'. 검색 수 9, 논리적 읽기 수 579712, 물리적 읽기 수 7190, 미리 읽기 수 579711, LOB 논리적 읽기 수 0, LOB 물리적 읽기 수 0, LOB 미리 읽기 수 0.
CPU 시간 = 9891밀리초, 경과 시간 = 5991밀리초

SET STATISTICS TIME OFF
SET STATISTICS IO OFF
```

➢ 간단한 결론

테스트 내용을 간단히 정리해보면, DB 환경에 SSD를 보편적으로 사용하게 된다면, 현재 디스크 기반의 여러가지 문제가 사라질 것으로 보입니다.
또한 많은 IOPS를 충족시키기 위해 수많은 디스크를 RAID를 구성하지 않아도 되기에 외부 스토리지의 필요성이 적어질 것이며, 비용적인 측면으로 봐도 보다 저렴하게 시스템을 구성할 수 있을 것 입니다.

SSD가 DB 환경에 많이 사용 되는 그날을 기대해 봅니다.