SSD 의 구조와 그에 따른 특성의 이해 - 플래시 변환 계층과 웨어 레벨링, 오버 프로비저닝

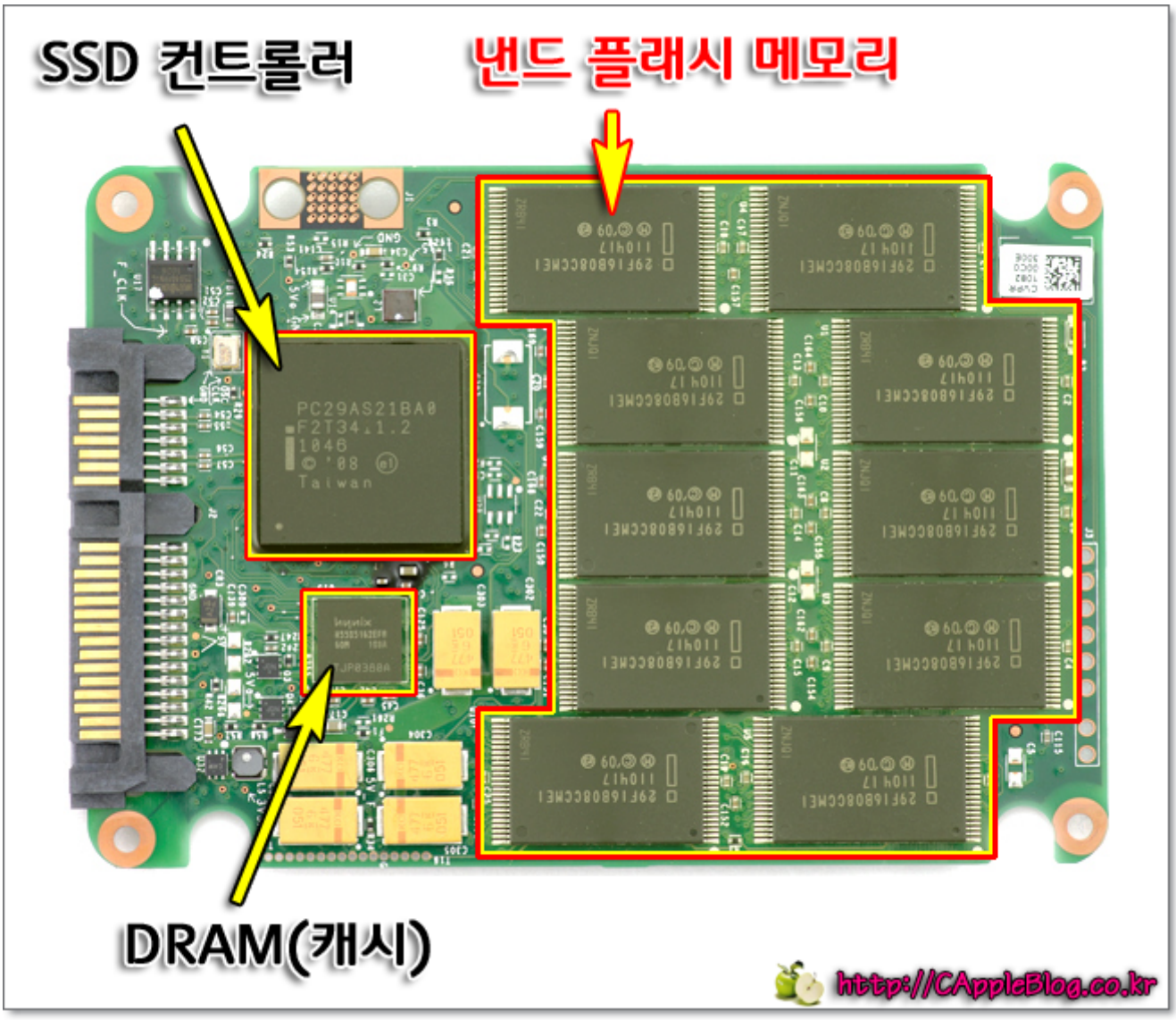
SSD 디스크와 HDD 디스크는 겉으로 보기엔 똑같아 보이고, 사용하는 방법도 동일합니다. 이는 윈도우에서 보기엔 SSD 디스크나 HDD 디스크나 동일하게 보이며, 파티션을 나누고 관리하는 것도 모두 동일하다는 이야기입니다. 즉, 우리가 실질적으로 윈도우에서 SSD 디스크를 사용하는 것이나, HDD 디스크를 사용하는 것이나 다를 게 없다는 것이죠. 이를 다른 말로 표현하자면 SSD 디스크와 HDD 디스크는 표층적으로 보이는 논리적인 구조는(디스크, 파티션, 섹터 등) 모두 동일하다고 할 수 있습니다.

하지만 익히 알다시피 SSD 와 HDD 는 하드웨어적으로는 종류가 전혀 다른 장치입니다. 일례로 HDD 는 데이터의 저장에 자기장을 사용하는 기계적인 장치인데 반해, SSD 는 데이터의 저장에 플로팅 게이트 트랜지스터를 사용하는 전기적인 장치이고, 물리적인 구조도 HDD 는 플래터를 실린더, 트랙, 섹터로 나눈 구조로 가지는 반면 SSD 는 칩 별로 플레인에 블록과 페이지 기반 구조를 가지니까요. 고로 달라도 너무 다르죠. 그래서 HDD 디스크에서 통용되던 정보들은 SSD 디스크에선 불필요하거나 전혀 다른 결과를 가져올 수도 있습니다. 가장 큰 예로 디스크 조각 모음의 경우 HDD 에선 성능을 위해 주기적으로 해주어야 했지만, SSD 에선 아무런 도움이 되지 않고(오히려 수명을 위해 일부러 데이터를 조각내서 저장), 오히려 SSD 의 수명을 단축시키는 결과를 가져오기도 합니다.

이번 글에서는 SSD 에선 HDD 와 달리 특정 작업에서 왜 그러한 결과가 나오는 것인지를 이해하기 위해 지금까지 캐플 블로그에서 지면 관계상(?) 하지 못했던 SSD 의 전체적인 이야기를 해보고자 합니다. 그럼 글을 시작하도록 하겠습니다.

SSD 의 물리적인 구조와 특성의 이해 - 블록과 페이지
1. SSD 의 전체적인 물리적 구조

일단 먼저 SSD 를 뜯어 보면 아래와 같이 생겼습니다.



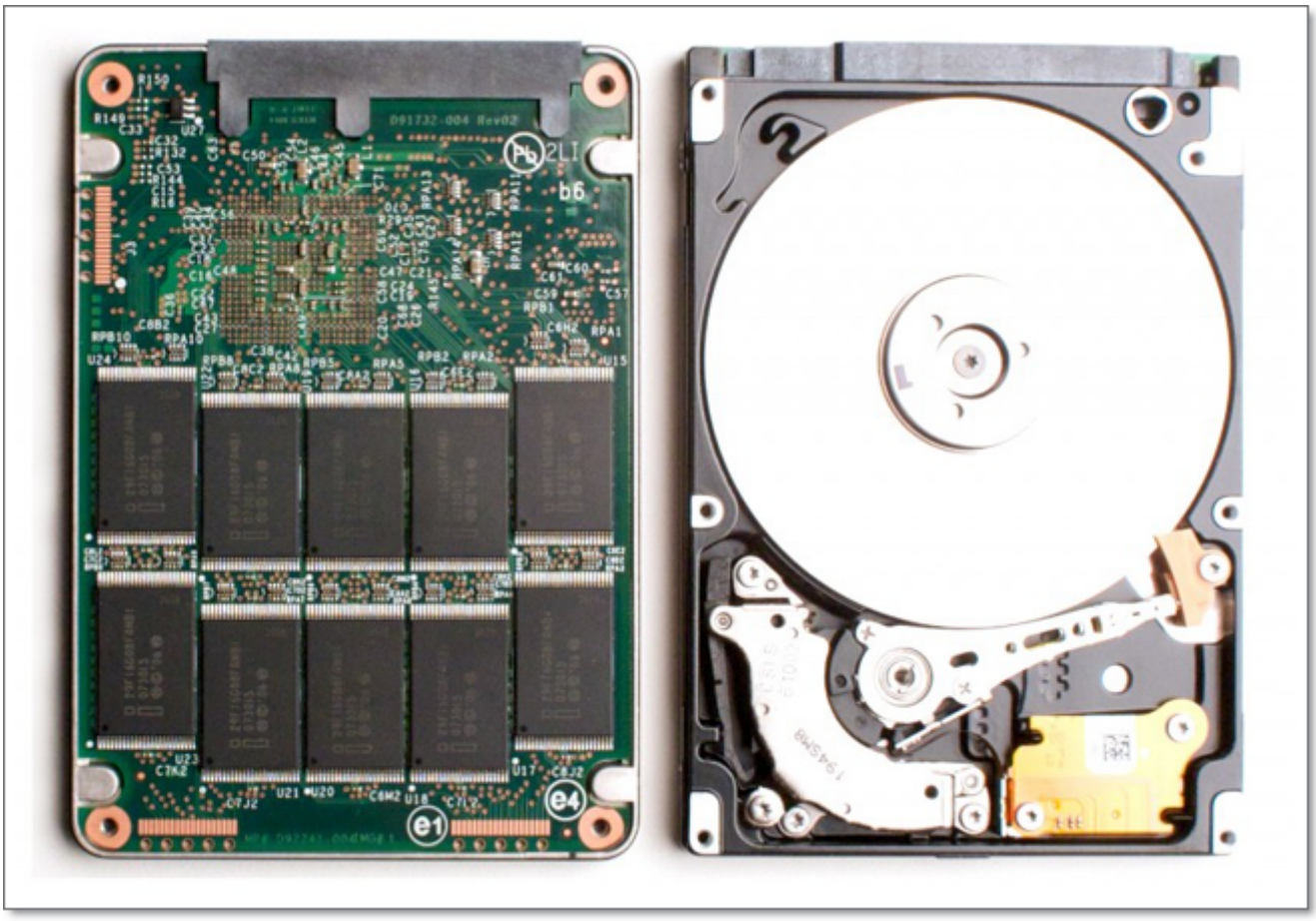
SSD 내부의 구조
[원본 출처 : Intel SSD 320 - www.storagereview.com]

지난 글에서 알아본 낸드 플래시 메모리가 쭈욱 달려 있는 걸 확인할 수 있습니다. SSD 에서 여러분의 데이터는 바로 저러한 낸드 플래시 메모리에 나눠 저장되는 겁니다. 그 외에 SSD 의 핵심이라고 할 수 있는 SSD 컨트롤러가 보이고, 캐시(버퍼)로 사용되는 DRAM 도 보이네요. 참고로 SSD 의 생명은 컨트롤러입니다. SSD 와 관련된 모든 기술의 집약체라고 할 수 있죠. 쉽게 컨트롤러는 낸드 플래시 메모리에 데이터를 저장하고, 낸드 플래시 메모리를 관리하고, 다시 데이터를 읽는 것까지 모든 것을 관장하는 가장 핵심 부품이라고 보시면 됩니다. 그래서 만약 SSD 컨트롤러가 저급하다면(설계 미숙) 제 아무리 훌륭한 SLC 를 출동시킨다 할 지라도 SSD 는 안전성, 신뢰성, 성능 등이 떨어질 수 밖에 없습니다. 또한 이러한 컨트롤러 설계에 자신이 있다면 주

력 제품에 대차게 TLC 를 사용하는 거고요.

자~ 일단! 제가 왜 SSD 의 내부 모습을 보여드렸냐면, SSD 는 HDD(하드 디스크) 와 달리 기계적인 구동부가 전혀 없고, 램과 같이 오직 전기 신호로만 움직이는 저장 매체라는 것을 알려드리기 위해서 입니다. 이것은 "SSD 에서는 왜 조각 모음이 필요 없는가?" 라는 물음에 대한 해답이 됩니다.

이 부분에 대해서 간단하게 설명을 하자면 HDD 는 아래와 같이 플래터라는 원판이 회전하고 그 위를 헤드 암이 기계적으로 왔다 갔다 움직이면서 데이터를 저장하고 읽는 구조인 반면, SSD 는 오직 전기 신호로만 데이터를 저장하고 읽습니다.



SSD 와 HDD 의 모습 비교
[좌] SSD, [우] HDD
[원본 출처 : hardwrk.com/blog/ssd-hdd-empfehlugen-fur-mabook-pro]

단편화란 무엇인가요? 파일이 저장 장치에서 물리적으로 연속된 공간에 저장되지 못하고, 여러 곳으로 쪼개져서 저장된 상태를 의미합니다. 조각 모음(Defragment)이 무엇인가요? 단편화(fragmentation)된 상태로 저장된 파일들을 재조합하여 물리적으로 연속된 공간에 다시 저장하여 단편화를 제거하는 겁니다.

HDD 는 일차적으로 기계적인 장치입니다. 즉, 모터로 플래터를 회전시키고 이러한 플래터에 저장된 데이터를 읽기 위해 헤드 암이 왔다 갔다 하면서 실제로 움직여야 한다는 겁니다. 만약에 파일이 중간에 끊기지 않고 연속된 공간에 저장되어 있다면 데이터의 시작 부분부터 끝까지 한 번에 쭈욱 읽으면 끝납니다. 그런데 어떠한 파일이 이처럼 연속된 공간에 저장되어 있지 않고 여기저기 흩어져서 저장이 되어 있다면, 쪼개진 파일 부분들을 모두 읽기 위해 이쪽에서 읽다가 끊어진 부분에서 다시 저쪽으로 옮겨서 읽어야 합니다. 즉, 죽어라고 왔

다 갔다 해야 한다는 거죠. 그래서 그 기계적으로 움직이는 거리만큼 데이터를 읽는데 더 오랜 시간이 걸리게 됩니다. 디스크의 성능이 떨어지는 거죠. 그래서 HDD 에서는 조각 모음이 필요합니다.

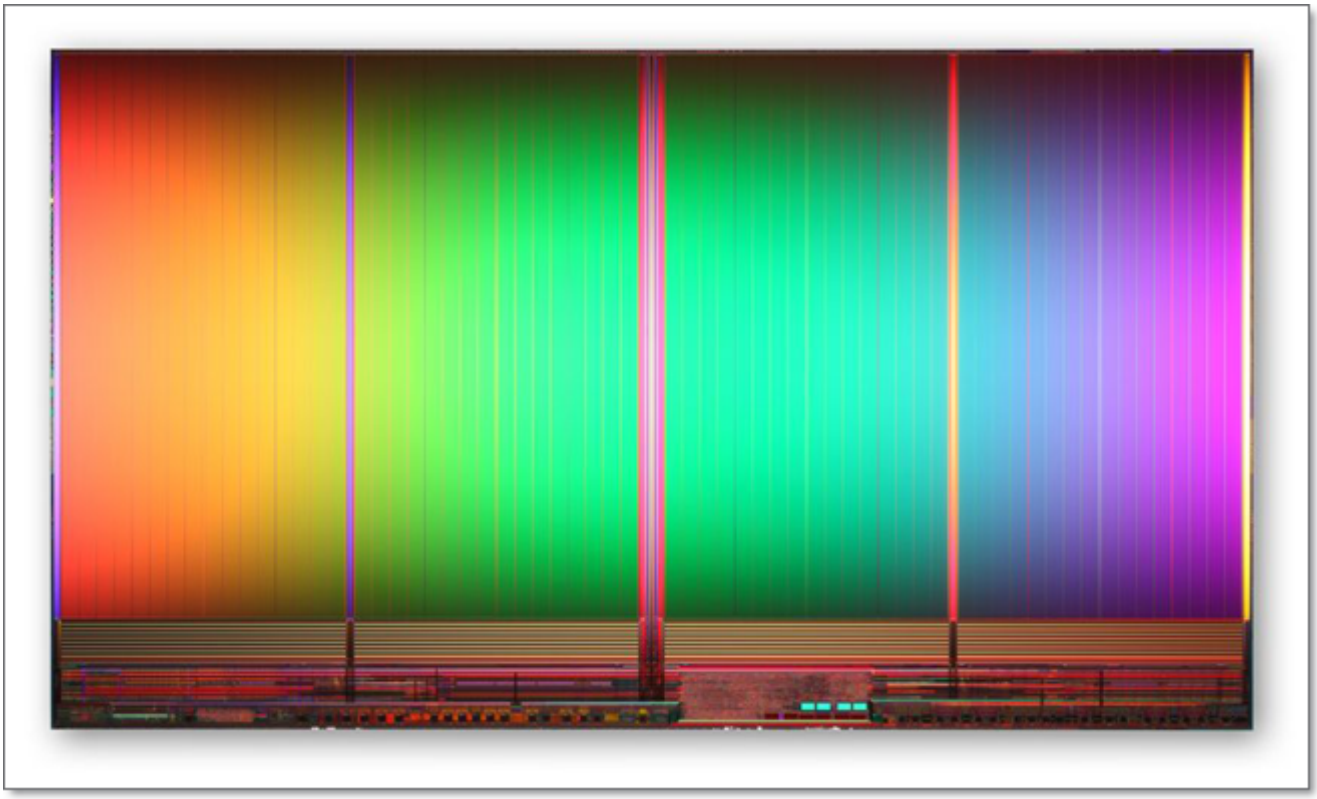
하지만 SSD 는 오직 전기 신호로만 작동하는 장치입니다. 그래서 파일이 물리적으로 쪼개져서 저장되어 있다고 할 지라도 이를 읽는데는 거의 차이가 없습니다. 읽을 데이터의 주소를 지정하면 즉각 튀어나오는 구조이기 때문이죠. 쉽게 데이터가 물리적으로 10CM 떨어진 위치에 저장이 되어 있다고 가정을 해보도록 하겠습니다. 전기 신호가 그 10CM 떨어진 위치에 도달하는데 얼마의 시간 차이가 날까요? 의미가 없다는 거죠. 그래서 데이터가 조각나 있어도 성능상의 차이는 현실적으로 없습니다. 그래서 오직 전기 신호로만 작동하는 램이나 SSD 와 같은 장치에선 조각 모음이 의미가 없습니다. 오히려 수명을 위해서 일부러 파일을 조각을 내고 분산시켜서 저장하기도 하죠.

아주 쉽게 생각하면, 서울과 부산에 각각 데이터를 쪼개서 놔두고(단편화) 이것을 읽어오라고 시켰을 때, HDD 는 차타고 먼저 서울에 들려서 데이터를 확인하고, 다시 차타고 부산으로 이동해서 데이터를 확인하는 구조입니다. 직접 움직이는 거죠. 이에 반해 SSD 는 서울에 전화해서 데이터가 뭔지 물어보고, 다시 부산에 전화해서 데이터가 뭔지 물어보는 구조 입니다. 직접 움직일 필요가 없죠. 즉, HDD 와 SSD 는 이러한 구조이기 때문에 만약 부산에 있는 데이터를 인천으로 옮긴다면(조각 모음) HDD 에선 굉장히 의미있는 작업이 되겠지만, SSD 에선 물어볼 전화 번호만 바뀌었을 뿐 데이터를 확인하는데 걸리는 시간은 똑같다는 겁니다. 간단하죠?

그리고 이러한 이유로 HDD 가 차를 전투기로 바꿔 성능을 올려도, 구조의 한계로 접근 시간에 있어서는 SSD 를 이길 수가 없습니다. 플래터의 회전수를 한 100만 RPM 정도로 올리고 헤드 암이 제로의 영역에서 춤을 추면 모를까 안 되는 건 안 되는 거죠. 플래터의 회전수가 100만 RPM 에 도달하기 전에 먼저 HDD 플래터가 회전을 감당하지 못해 폭발하는 게 빠를 겁니다. 그럼 파편 맞고 기도로만 만나시던 분을 직접 뵙게 되겠죠. [HDD 의 회전수가 15,000 RPM 에서 더이상 올라가지 않는 이유가 그겁니다. 플래터가 회전수를 이기지 못하고 산산조각나서 폭발해 버리니까요.]

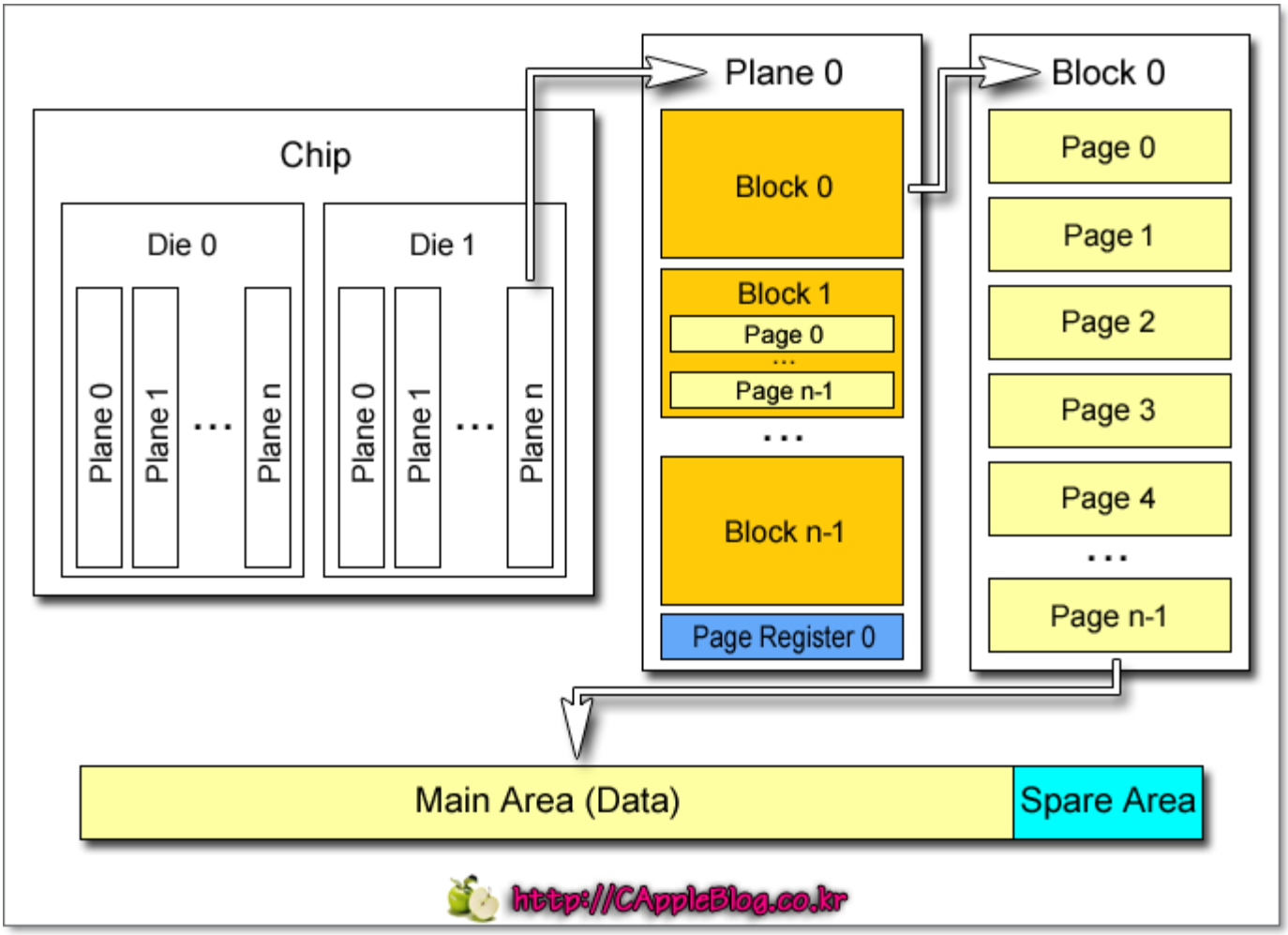
2. SSD 낸드 플래시 메모리 공간의 물리적인 구조 - 블록과 페이지

SSD 의 전체 모습에서 SSD 에는 하나가 아닌 여러 개의 낸드 플래시 메모리가 장착되어 있는 걸 확인할 수 있었습니다. SSD 는 이러한 낸드 플래시 메모리들을 채널로 묶고, 하나의 공간으로 통합하여 저장 공간을 마련합니다. 우선 낸드 플래시 메모리 칩은 실제로 아래와 같이 생겼습니다.



낸드 플래시 메모리의 실제 모습
(25nm IMFT 2-bit MLC NAND Flash, 8GB, 167mm$^{2)}$
[원본 출처 : www.anandtech.com]

예쁘죠? 참고로 위의 제품은「2 Die - 2 Plane」의 구조를 가진 8GB(64Gb) 낸드 플래시 메모리로「1 Plane = 1,024 Block」「1 Block = 256 Page」「1 Page = 8KB」 로 구성되어 있습니다.「1 LUN = 2 Plane = 2,048 Block」 즉, 페이지가 모여 블록이 되고, 블록이 모여 플레인이 되고, 이러한 플레인이 모인 것이 바로 낸드 플래시 메모리 칩입니다. 이러한 낸드 플래시 메모리의 구조를 간단하게 모식으로 나타내면 아래와 같습니다.

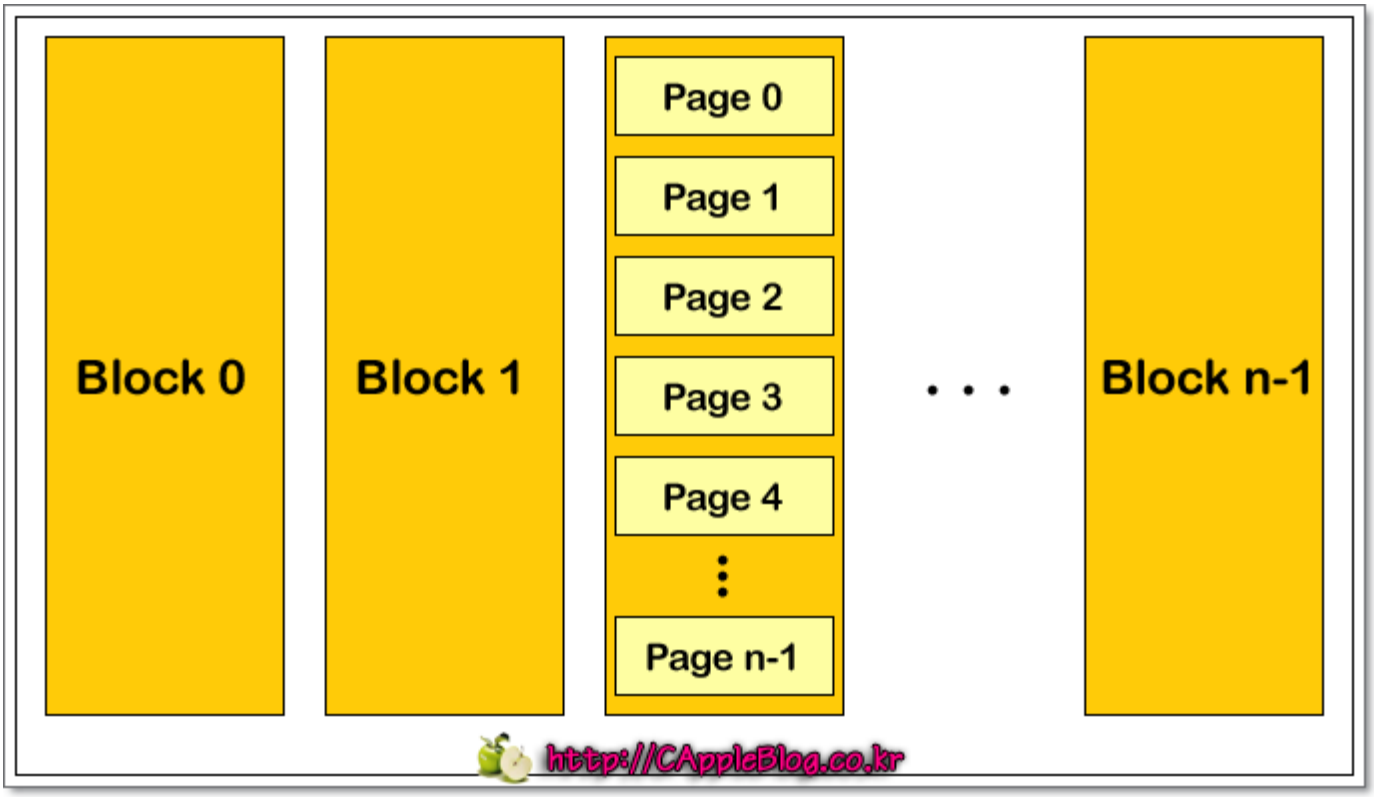


낸드 플래시 메모리의 구조

페이지(Page)	낸드 플래시 메모리에서 데이터 저장의 기본 단위. (셀의 집합)	읽기, 쓰기 작업의 최소 단위
블록(Block)	페이지의 집합.	지우기 작업의 최소 단위
플레인(Plane)	블록의 집합.	연산 처리의 단위

여러 가지 개념들이 있지만 낸드 플래시 메모리에서 가장 중요한 것은 데이터의 읽기와 쓰기, 지우기에서 사용되는 단위인 블록과 페이지입니다. 특히나 페이지는 낸드 플래시 메모리에서 실질적인 데이터 저장의 최소 단위(기본 단위)이기 때문에 이것은 반드시 기억해두시길 바랍니다. 그 외에는 솔직하게 관련 전공자나 관련 엔지니어가 아닌 이상 크게 의미가 없는 개념들이기 때문에 관심이 있다면 더 검색해보시면 좋고 아니면 말고 정도로 생각하시면 됩니다.

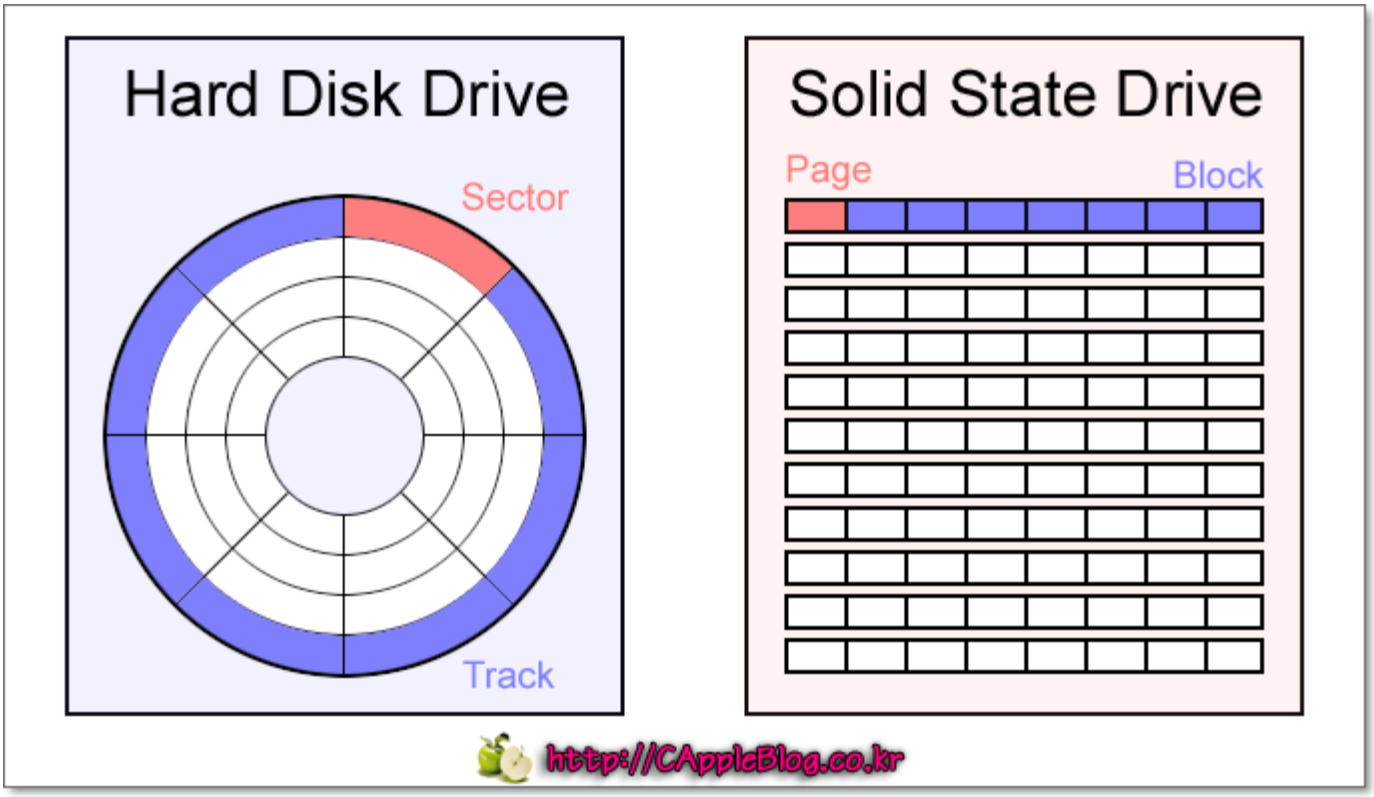
아무튼, SSD 는 이러한 낸드 플래시 메모리 칩 다수를 묶어 하나의 공간으로 사용합니다. 고로 간단하게 SSD 의 저장 공간을 정의내리면 블록과 페이지의 묶음이라고 할 수 있습니다. 그래서 이를 간단하게 추상적으로 나타내면 SSD 의 저장 공간은 아래와 같은 구조를 가지고 있는 것이죠.



SSD 저장 공간의 추상적인 구조

요즘은 보통 페이지가 4KB~16KB 의 크기를 가지며, 블록은 128KB~4096KB 의 크기를 가집니다. 과거엔 512Byte~2KB 페이지에, 64KB~128KB 의 블록도 사용되었지만, 낸드 플래시 메모리가 공정의 미세화와 MLC, TLC 로 고용량화되면서 점차 페이지와 블록의 크기도 늘어나고 있는 경향입니다.

이러한 SSD 의 구조를 쉽게 기존 HDD 의 구조와 비교해서 생각해보면, 블록은 트랙과 페이지는 섹터와 비슷하다고 보셔도 됩니다. [플레인은 실린더, 칩은 플래터] 물론 완전히 같은 의미를 가지고 있는 것은 아니지만요. 그점은 유의하시고요. 아무튼, 무엇보다 HDD 에서 데이터 저장의 최소 단위는 섹터이고, SSD 에서 데이터 저장의 최소 단위는 페이지란 것만 기억하시면 될 듯합니다.



HDD 와 SSD 의 저장 공간 비교

다시 SSD 이야기로 넘어와서 보시는 것처럼 SSD 에는 물리적으로 섹터가 존재하지 않습니다. 하지만 우리가 PC 에서 사용하는 FAT, NTFS 와 같은 파일 시스템들과 그 위에서 작동하는 운영체제들(윈도우), 프로그램들은 모두 섹터를 기반으로 하여 작동합니다. 즉, 간단하게 핵심만 짚어서 이야기하자면 우리가 흔히 사용하는 파일 시스템들은 섹터로 구성된 HDD 의 구조에 맞춰져 있습니다. [섹터 기반]

그런데 낸드 플래시 메모리에는 물리적인 섹터가 없죠. 블록과 페이지만 있을 뿐입니다. 그래서 낸드 플래시 메모리를 올바로 사용하기 위해선 그에 맞는 페이지 기반의 새로운 파일 시스템이 필요합니다. 기존에 PC 에서 사용되던 파일 시스템들은 그게 아니죠. 그래서 새로운 파일 시스템을 도입해야 하는데 PC 에서 사용되는 모든 운영체제와 프로그램들이 그에 맞춰 새로운 파일 시스템을 도입한다는 것은 사실상 무리가 따릅니다.

그래서 SSD 에서는(낸드 플래시 메모리에서는) 이 문제를 해결하기 위해 SSD 자체의 컨트롤러 단계에서 플래시 변환 계층(Flash Translation Layer, FTL) 이란 것을 사용하여 논리적으로 섹터 구조를 구축하는 방법을 사용하고 있습니다. 이러한 플래시 변환 계층은 낸드 플래시 메모리를 기존의 하드 디스크 드라이브처럼 사용할 수 있도록 도와주고, 그 외 낸드 플래시 메모리의 물리적인 특성(단점)들을 보완해 주는 역할을 하게 됩니다.

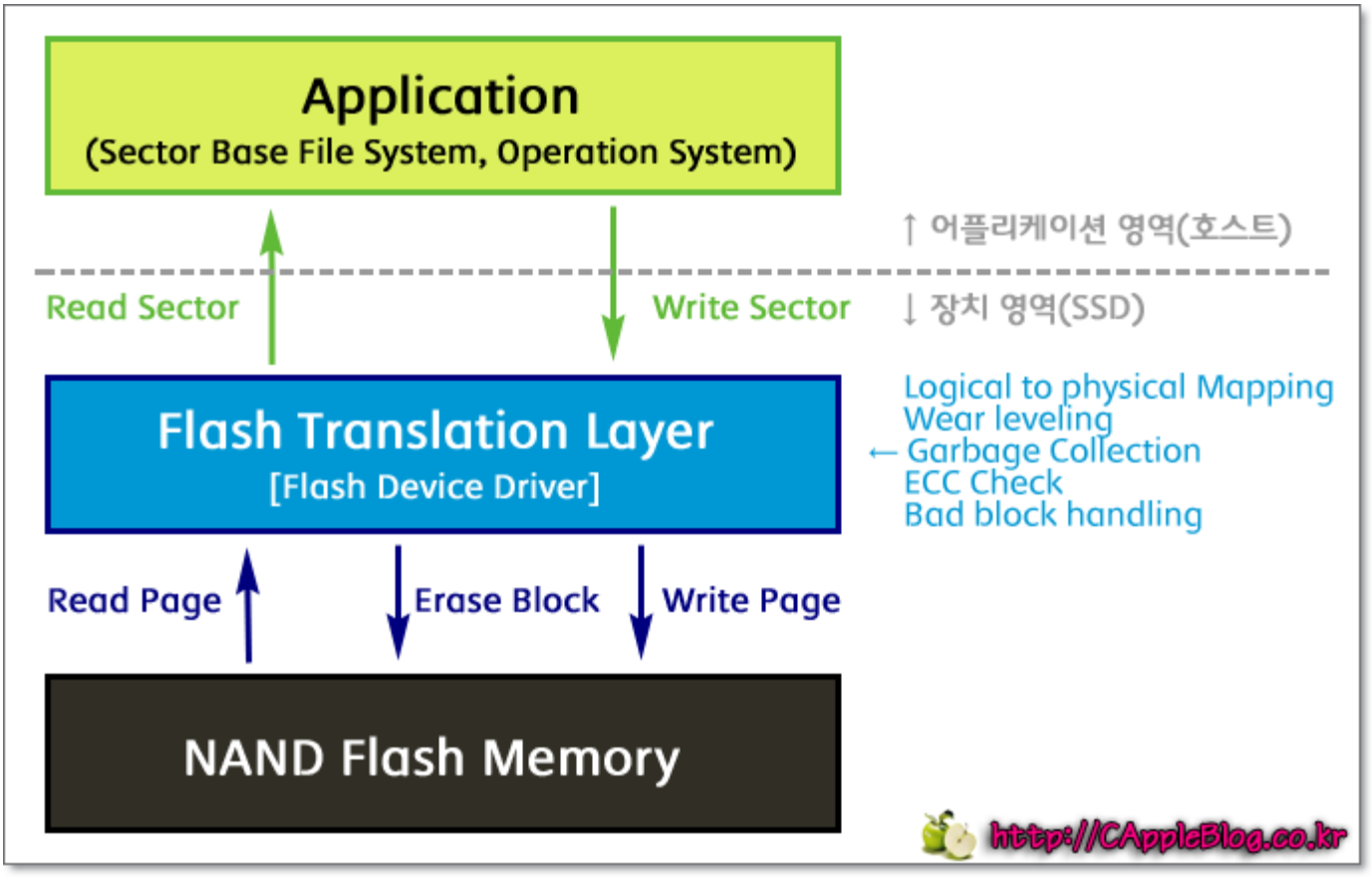
플래시 변환 계층과 웨어 레벨링 - 낸드 플래시 메모리의 특성을 숨겨라
1. 플래시 변환 계층(Flash Translation Layer)의 이해

PC 와 그 하위에서 사용되는 파일 시스템, 운영체제와 이를 기반으로 하는 프로그램들은 섹터를 기반으로 하고 있습니다. 하지만 SSD 는 블록과 페이지 기반의 저장 매체입니다. 즉, 원칙적으로 우리가 사용하는 기존의 PC 용 파일 시스템과 운영체제들은 구조적인 문제로(다르기 때문에) 직접적으로 플래시 메모리를 활용할 수 없습니다.

섹터 기반	Read Sector() Write Sector()	← 작업의 불일치 →	Read Page() Write Page() Erase Block() Copy Back Page()	**페이지 기반**

이 문제를 해결하기 위해 SSD 에서는 컨트롤러 단계에서 플래시 변환 계층이란 것을 적용하여 물리적인 블록과 페이지 위에 논리적인 섹터 구조를 구현하는 방식을 사용합니다. 이를 통해 기존 섹터 기반의 파일 시스템, 운영체제, 프로그램들이 아무런 문제없이 SSD 를 기존의 HDD 처럼 사용할 수 있게 만들어주는 것이죠. 또한 이러한 플래시 변환 계층에는 낸드 플래시 메모리라는 소자의 특성에 맞춰(쓰기보다 지우기가 느린 특성, 셀의 수명 한계 등) 필요한 부가적인 다른 기능들(웨어 레벨링, 가비지 컬렉션 등)이 포함되어 있습니다.

아래는 이러한 플래시 변환 계층의 구조를 간단하게 나타낸 것입니다. 참고로 아래의 그림은 설명을 단순화하기 위해 Flash Device Driver 를 플래시 변환 계층과 함께 표현하였으며, 그에 맞춰 기능들도 구별 없이 설명하였습니다. [즉, 실제론 아래의 이미지보다 많이 복잡하고 여러 가지 방식이 있습니다. 그러니 단순한 참고용으로만 이미지를 이해하세요.]

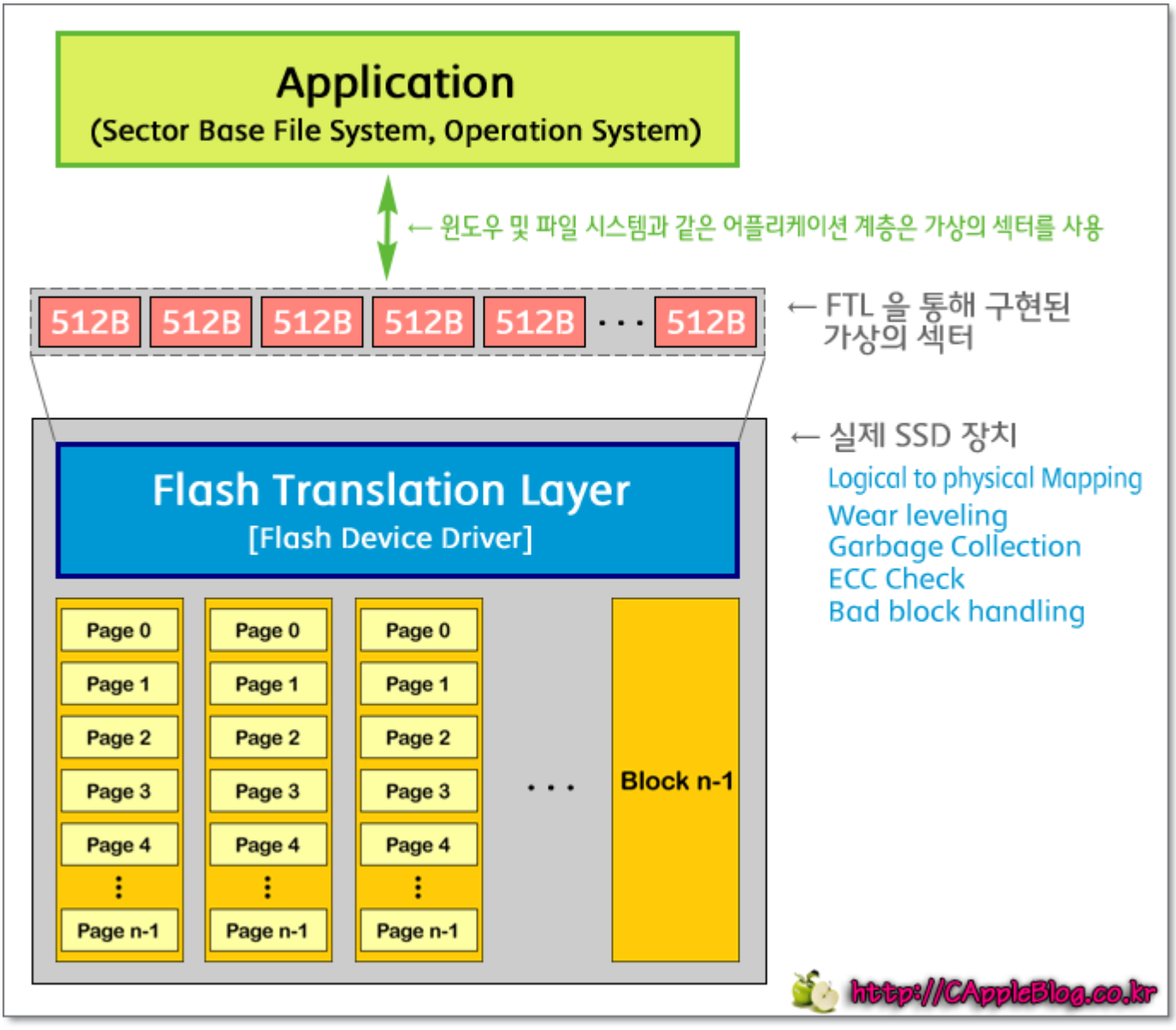


플래시 변환 계층의 간략한 구조

보시는 것과 같이 파일 시스템이니 운영체제들은 SSD 를 사용할 때 낸드 플래시 메모리를 직접 다루는 것이 아니라, 일종의 미들웨어인 플래시 변환 계층을 중간에 두고 이와 통신하는 구조를 가지고 있습니다. 그럼 그 하위에서 필요한 작업은 모두 플래시 변환 계층(과 플래시 장치 드라이버)가 알아서 하는 것이죠.

이는 다른 말로 우리가 사용하는 윈도우나 파일 시스템은 SSD 의 물리적인 영역에는 관여하지 않습니다. 모든 것은 SSD 컨트롤러 자체에서 해결하는 것입니다. [웨어 레벨링, 가비지 컬렉션 등의 작업은 모두 SSD 의 컨트롤러에서 자체적으로 진행하는 작업들입니다. 트림은 예외적으로 운영체제의 도움을 받지만(지워진 파일에 대한 정보만 제공 받음) 내부적인 작업은 역시나 SSD 컨트롤러 자체에서 진행합니다.]

이를 다시 블록과 페이지 그리고 논리적으로 구현된 섹터 구조로 표현하면 아래와 같습니다.



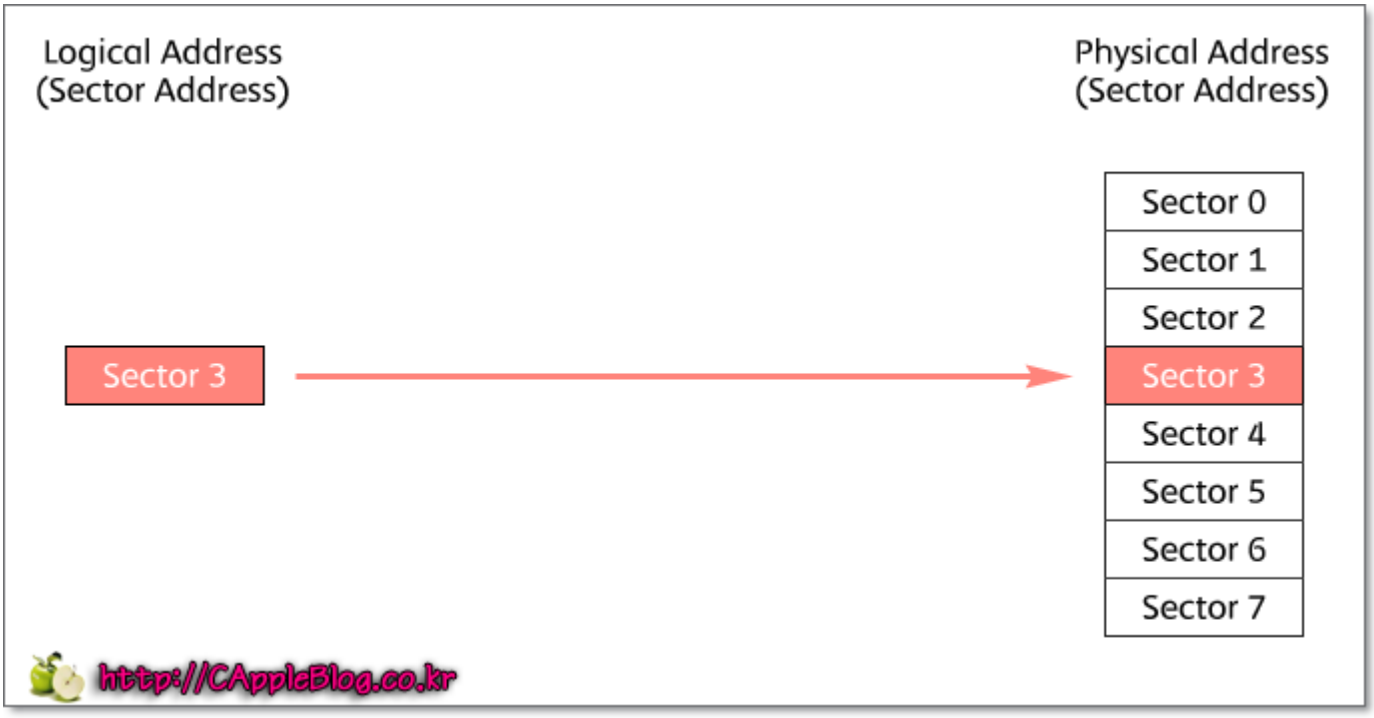
섹터 구조로 살펴본 플래시 변환 계층의 구조

즉, 우리가 윈도우를 통해 보고, 접하고, 사용하는 SSD 디스크는 모두 이러한 FTL 이 논리적으로(가상으로) 구현한 섹터 기반 디스크의 모습을 보고 있는 겁니다. 이거 굉장히 중요합니다. 기억해두세요. 일단은 여기에 해당하는 다른 이야기들은 좀 더 뒤에서 하기로 하고 우선 섹터에 집중하도록 하겠습니다.

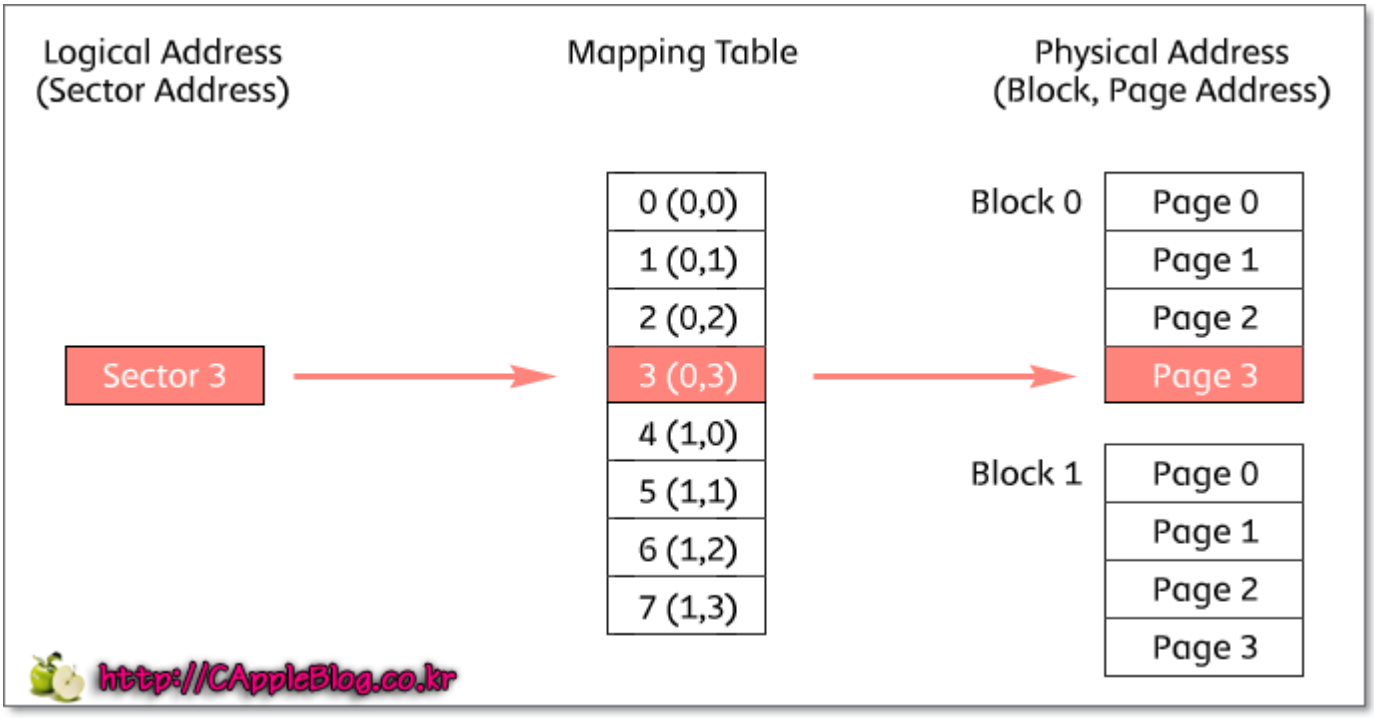
2. 플래시 변환 계층과 논리 주소, 물리 주소 - 주소 변환

파일 시스템이 논리적인 섹터 주소(Logical Address)를 통해 SSD 의 어떠한 데이터에 접근을 하면, FTL 은 이러한 논리적인 섹터 주소를 물리적인 블록과 페이지 주소로(Physical Address) 변환하여 최종적으로 낸드 플래시 메모리의 페이지에 도달하게 됩니다. 이를 위해 FTL 은 논리 주소와 물리 주소를 연결해주는 주소 변환용 매핑 테이블을 가지고 있습니다. 즉, 파일 시스템의 논리 주소(Sector Address)가 실제로 플래시 메모리의 어떠한 물리 주소(Block, Page Address)에 연결되는 것인지에 대한 표를 가지고 이 둘을 서로 연결시켜준다는 것이죠.

참고로 이러한 FTL 의 논리 주소 ↔ 물리 주소 매핑 방식에는 섹터 매핑(Sector Mapping), 블록 매핑(Block Mapping), 하이브리드 매핑(Hybrid Mapping) 등 여러 가지가 있습니다. 하지만 중요한 것은 논리 주소가 매핑 테이블에 의해 물리 주소로 변환되는다는 것이므로 그것만 기억하시면 될 듯합니다. 아무튼, 이를 기존의 HDD 와 비교하여 간단하게 그림으로 표현하면 아래와 같습니다. [이 때 페이지의 크기는 그냥 섹터와 동일하게 512Byte 라고 가정]



HDD 의 논리 주소 ↔ 물리 주소 구조



SSD 의 논리 주소 ↔ 물리 주소 구조

매우 단순하게 표현했지만 기본 구조는 위와 같습니다. 즉, 논리 주소로 접근을 하면, 중간에 매핑 테이블에 기록된 주소를 참조하여 최종적으로 페이지에 접근하는 것이죠. 참고로 그림은 간단하게 1 단계의 주소 변환 과정만을 표현했지만, 실제론 배드 블록의 관리 등을 위해 2 단계, 3 단계의 좀 더 복잡한 주소 변환 과정을 거치게 됩니다.

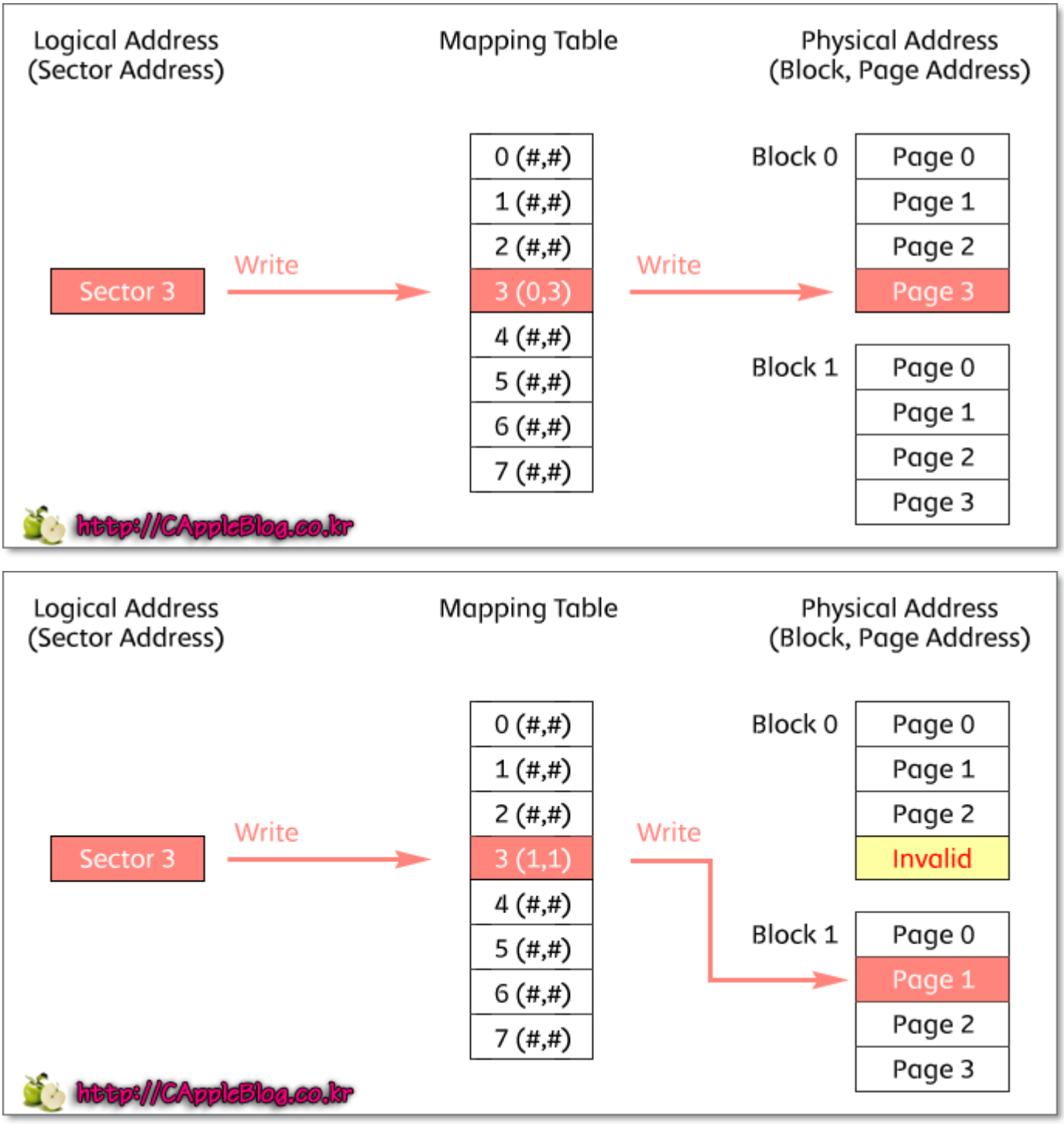
3. 플래시 변환 계층의 주소 변환과 웨어 레벨링 작업의 관계, 가비지 컬렉션

그런데 지난 SSD 의 이해 - 낸드 플래시 메모리의 구조와 SLC, MLC, TLC 방식의 차이 글에서도 다뤘듯이 플래시 메모리의 셀은 재기록 가능 횟수(P/E Cycles)가 정해져 있고 그에 따라 수명이 있습니다. 그리고 덮어쓰기 작업이 불가능합니다. 그래서 만약에 특정 페이지에 집중된 쓰기 작업이 일어난다면, (블록을 지운 후 다시 써야 하기 때문에) 해당 페이지가 포함된 블록 전체의 수명이 금새 바닥날 수 있습니다.

이러현 연유로 SSD 는 수명을 위해 포함된 모든 페이지들이 골고루 쓰여야 할 필요가 있습니다. 하지만 우리가 사용하는 FAT 이나 NTFS 와 같은 파일 시스템은 이러한 낸드 플래시 메모리의 사정을 전혀 고려하지 않고 있죠. 파일 시스템 자체를 뜯어 고친다는 것도 사실상 무리입니다. 그래서 이러한 문제를 SSD 자체적으로 FTL 단계에서 해결하게 됩니다. 그게 바로 그 유명한 웨어 레벨링(Wear leveling)입니다.

일단 SSD 는(낸드 플래시 메모리는) 페이지 레지스터(Page Register)를 통해 SSD 에 포함된 모든 페이지마다 개별적으로 총 몇 번의 재기록 작업이 이루어졌는지를 모두 기록하여 관리하고 있습니다. 이러한 상태에서 만약 어플리케이션 계층에서(윈도우에서) 어떠한 기록 작업이(Write Sector) 들어온다면, 플래시 변환 계층은 앞서 설명한 각 페이지의 재기록 횟수를 참고한 후, 주소 매핑 테이블에서 논리 주소와 물리 주소의 매핑 주소를 변환하는 방법을 통해 (비어 있는 페이지들 중) 아직 덜 사용된 페이지를 위주로 기록 작업을 진행하게 됩니다.

플래시 변환 계층에서 이러한 웨어 레벨링 기능이 작동하는 방식을 간단하게 파일의 수정을 예로 들어 설명을 해보도록 하겠습니다. 이렇게 파일을 수정하는 경우 다시 동일한 섹터로 기록이 이루어지겠죠? 그럼 웨어 레벨링은 아래와 같은 식으로 작업을 진행하여 특정 페이지에 집중되는 쓰기 작업을 피하게 됩니다.



웨어 레벨링 작업 - 주소 변환

즉, 웨어 레벨링은 주소 변환 테이블의 내용을 수정하여 논리 주소에 연결된 물리 주소 자체를 바꿔치기 하는 것이죠. 그리고 기존의 페이지는 다른 논리 주소로 연결시키고요. 간단하죠?

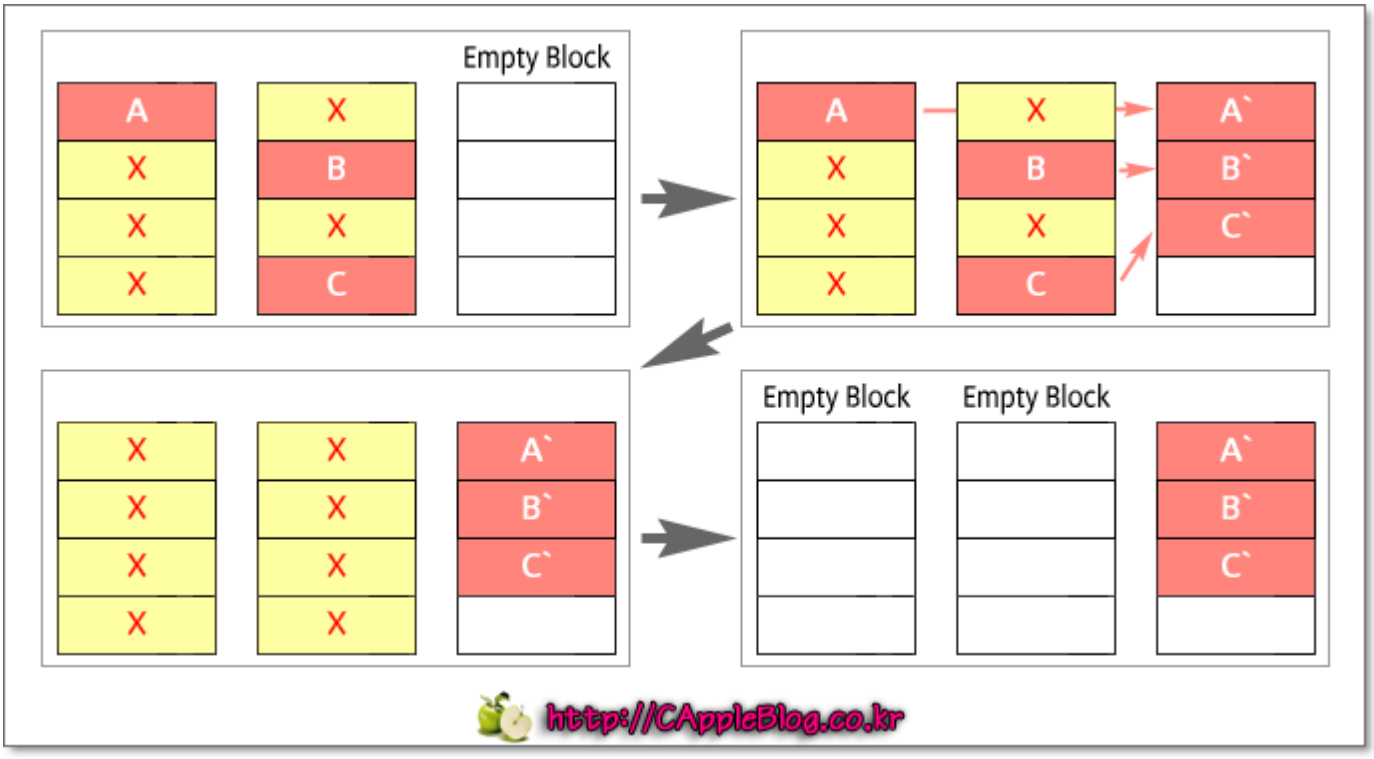
그런데 이 때 그림에서 표현했듯이 기존의 페이지에 있던 데이터는 우선은 그대로 놔둡니다. 왜냐? 이걸 지우려면 낸드 플래시 메모리의 특성상 블록 전체를 지운 후 상관 없는 다른 페이지들의 데이터는 다시 기록해야 합니다. 그럼 힘들게 주소 변환까지 시켜가면서 작업한 의미가 없어지잖아요? 가뜩이나 지우기 작업은 느리기도 하고 말이죠. 그래서 우선은 해당 페이지는 지우지 않고 필요 없는 쓰레기 페이지(Garbage)라고 마크

만 해둡니다. [Invalid 상태, 필요 없는 데이터]

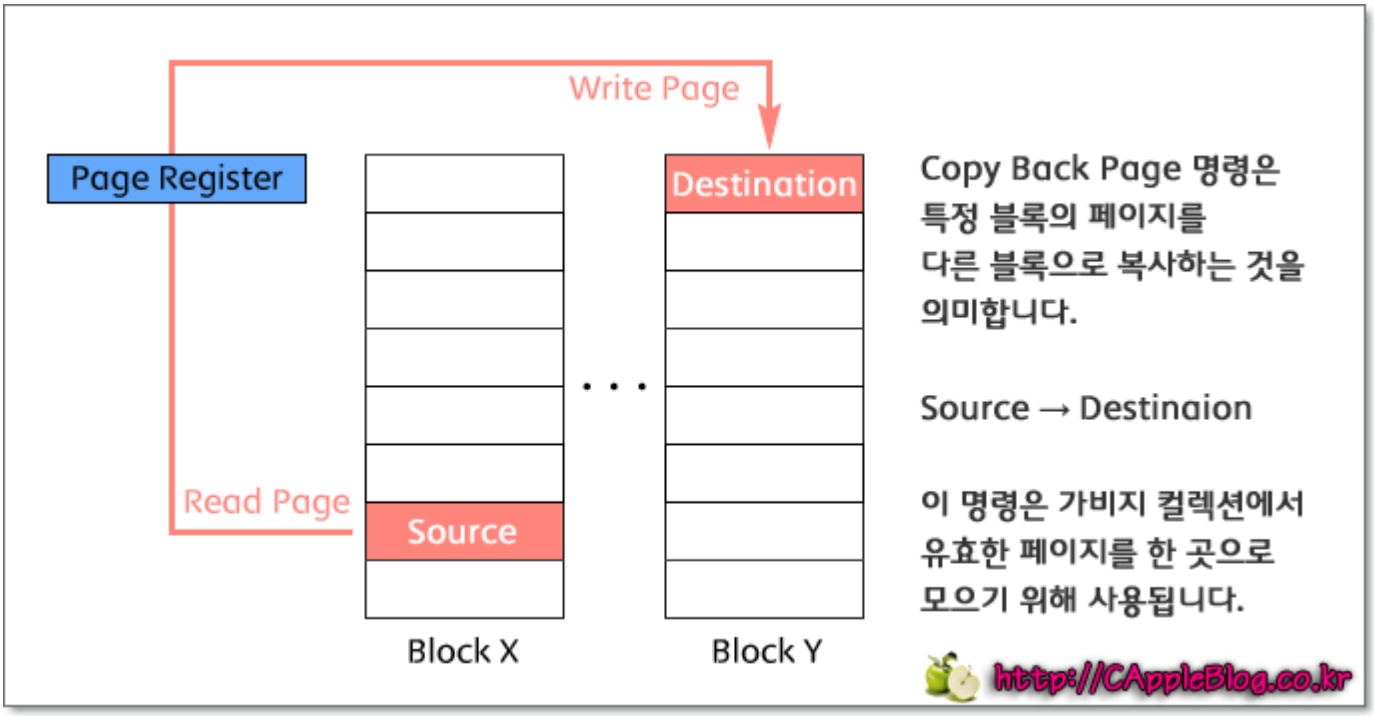
이렇게 사용하다보면 어쩔 수 없이 점차 필요 없는 데이터가 담긴 쓰레기 페이지들이 충분히 쌓이게 되겠죠? 이렇게 쓰레기 페이지들이 적당히 쌓이게 되면 적절한 시점에 한 번에 처리하는 겁니다. 그게 바로 가비지 컬렉션(Garbage Collection) 작업입니다.

즉, 웨어 레벨링을 통해 모든 페이지들이 골고루 쓰일 수 있게 함과 동시에 블록의 지우기 작업을 최대한 회피한 후, 적절히 쓰레기 페이지들이 쌓이면 가비지 컬렉션을 통해 몰아서 한 번에 지움으로써 블록의 지우기 작업 횟수를 그만큼 줄이는 거죠. 이를 토대로 낸드 플래시 메모리의 모든 페이지는 골고루 쓰일 수 있고, SSD 전체적으로 보았을 때 수명을 연장하는 효과를 얻을 수 있는 겁니다. 웨어 레벨링과 가비지 컬렉션, 환상의 한 쌍이죠.

참고로 가비지 컬렉션 작업은 다음과 같이 이루어지며, 이 때 카피백 페이지(Copy Back Page) 명령이 사용됩니다.



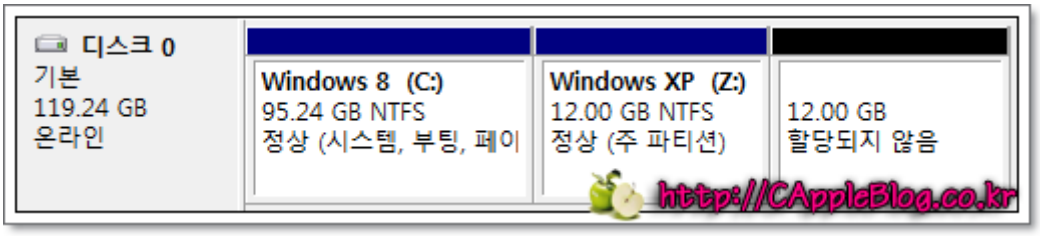
가비지 컬렉션 작업



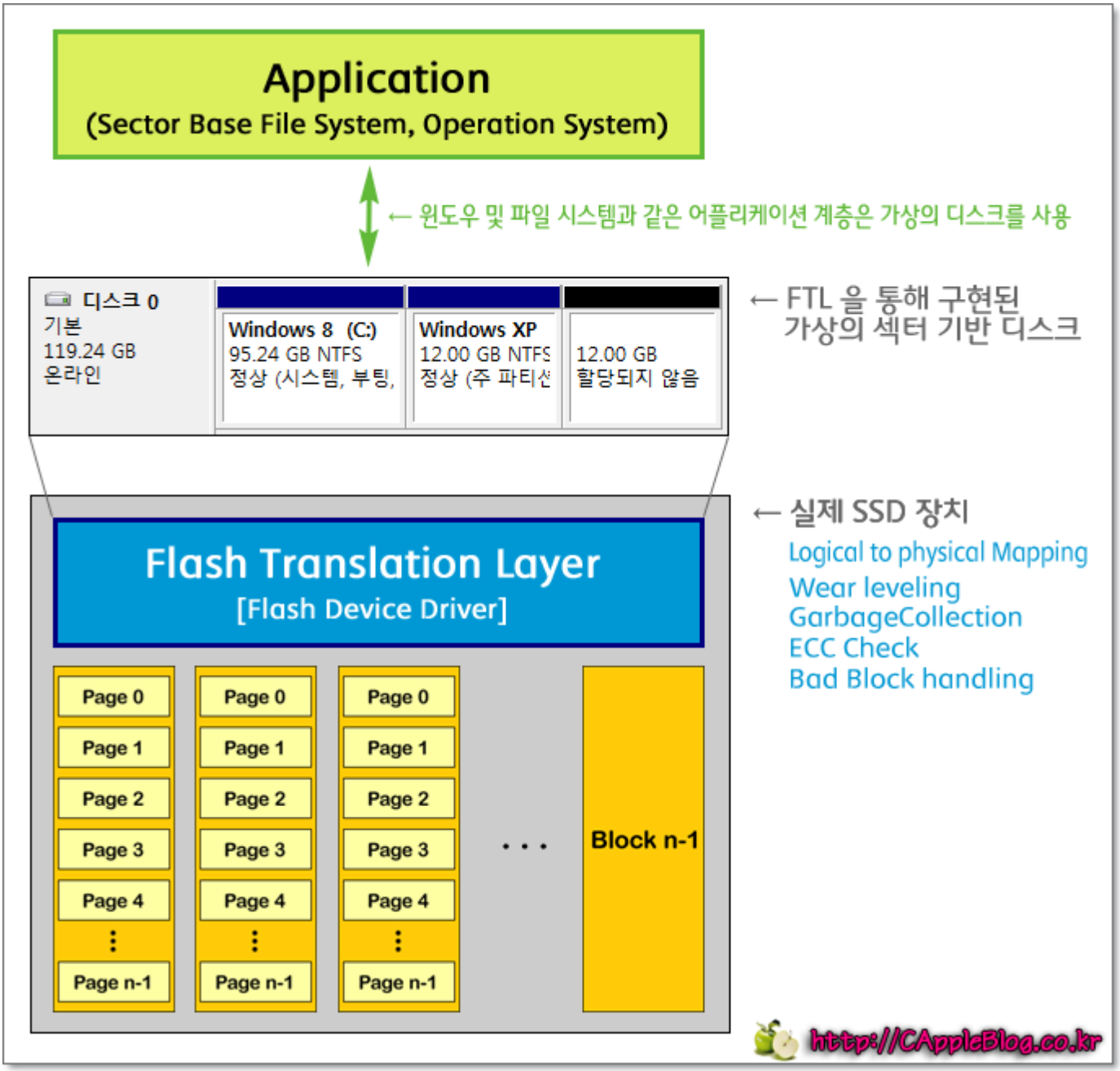
카피백 페이지 명령의 작업 구조

웨어 레벨링과 가비지 컬렉션 작업이 정확하게 무엇인지 이해하시겠나요? 이제 여기에서 아래의 두 가지만 기억하시면 됩니다.

먼저 지금까지 알아 본 웨어 레벨링과 가비지 컬렉션은 플래시 변환 계층(FTL) 영역에서 이루어지는 작업입니다. 이게 무슨 말이냐면, 웨어 레벨링과 가비지 컬렉션은 SSD 장치 단계에서 행해지는 작업이지, 그 위의 가상으로 생성된 논리 섹터 영역, 또는 어플리케이션 계층(호스트, OS, 파일 시스템)과는 상관이 없다는 겁니다. 일례로 아래와 같은 SSD 디스크가 있습니다.



여기에서 중요한 것은 이렇게 디스크 관리를 통해 볼 수 있는 디스크 자체가 플래시 변환 계층을 통해 논리적으로 구현된 가상의 디스크라는 겁니다., 웨어 레벨링과 가비지 컬렉션 작업은 계층 구조적으로 이보다 아래 단계인 SSD 장치 단계에서 이루어지는 것이고요. 즉, 위의 SSD 디스크는 파티션이 나눠져 있죠? 이렇게 파티션이 나눠져 있어도 웨어 레벨링과 가비지 컬렉션 작업에는 영향을 주지 않는다는 겁니다. 그러니까 간단하게 아까 살펴본 플래시 변환 계층의 구조에 대입해서 보면 아래와 같은 겁니다.



디스크로 살펴본 플래시 변환 계층의 구조

참고로 컨트롤러의 FTL 알고리즘에 따라 다르지만 오히려 이런 식으로 할당되지 않은 공간을 남겨두는 것이 SSD 의 성능과 수명에 도움을 주기도 합니다. [물론 반드시 그런 것은 아닙니다.] 이에 대한 자세한 내용은 뒤에서 하는 걸로 하고 아무튼, SSD 디스크에서 파티션을 나눠 사용하는 것은 웨어 레벨링과 가비지 컬렉션과는 상관이 없습니다. 즉, SSD 디스크의 파티셔닝 작업이 SSD 의 수명에 어떠한 영향을 미치지는 않는다는 거죠.

그러니까 원하시는대로 적절히 파티션을 나눠서 사용하시면 됩니다. "SSD 니까 파티션은 어떻게" 하실 필요는 없다는 거죠.
다음으로 HDD 는 논리 주소와 물리 주소가 일치하지만, SSD 는 지금까지 살펴본 것과 같이 논리 주소에 매핑되는 물리 주소가 일치하지 않습니다. 즉, 논리 섹터 0 하면 HDD 에선 항상 물리 섹터 0 이지만, SSD 에선 이게 어느 블록의 어느 페이지일 지는 매핑 테이블을 살펴보기 전까지는 며느리도 모른다는 겁니다. 게다가 이게 고정되어 있지도 않고 필요할 때마다 바뀌어 버립니다.

이게 왜 중요하냐면요. 보안 삭제 이야기가 나오면 꾸준하게 나오는 이야기가 있는데 "낸드 플래시 메모리는

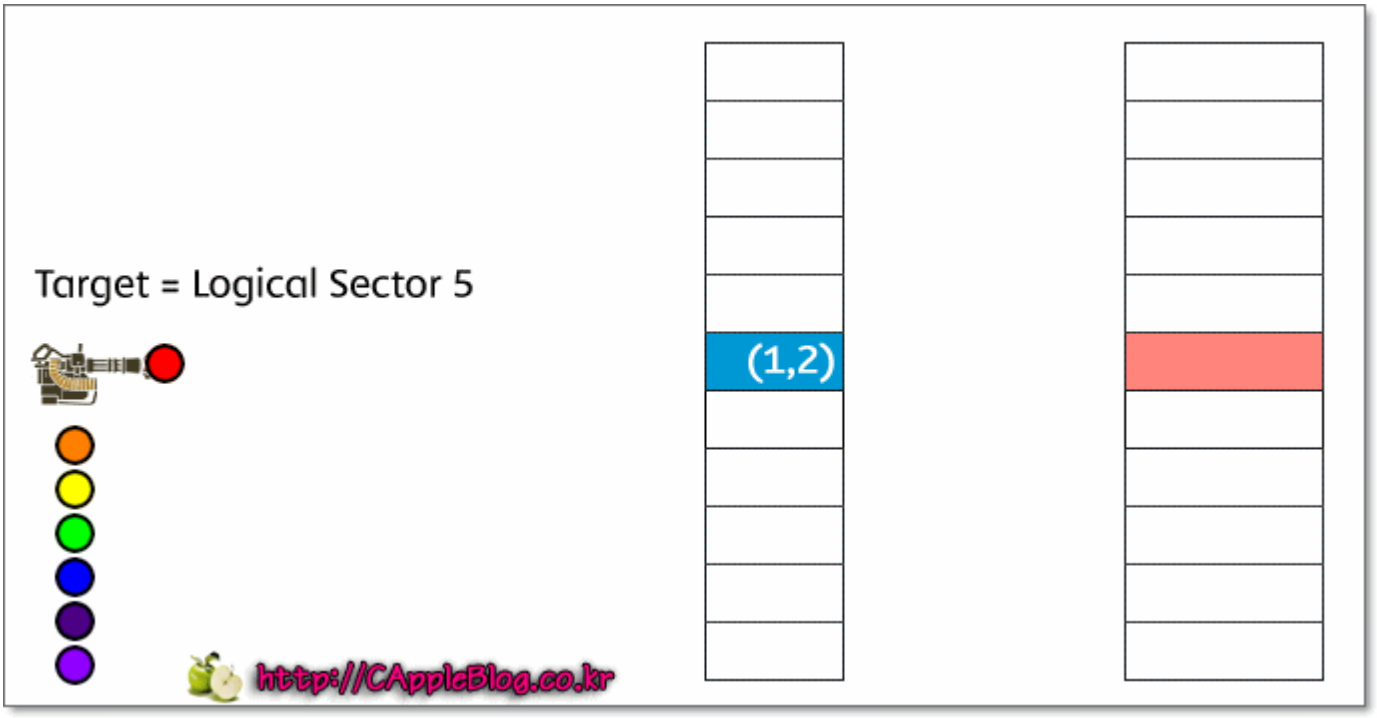
수명이 있기 때문에 보안 삭제 작업이 수명을 단축시키겠지만, 어쨌든 그래도 보안적인 측면에선 해줘도 좋은 것 아니냐?" 라는 이야기입니다. 이러한 질문에 몇 번 SSD 는 구조상 보안 삭제 작업이 필요 없고, 보안 삭제 작업을 진행해도 생각하는 결과를 보여주지 않기 때문에 의미가 없다고 말씀을 드린 적이 있습니다. 사실 이에 대한 필요한 설명은 위에서 모두 되었지만, 다시 한 번 짚어보도록 하죠.

일단 보안 삭제 작업이 어떻게 진행되는지는 보안 삭제란 무엇이며 보안 삭제 프로그램은 파일과 공간을 어떻게 삭제할까? 글에서 모두 설명을 드렸습니다. 아무튼, 일반적으로 보안 삭제란 특정 파일이나 영역을 의미 없는 데이터로 덮어쓴 후 지우는 것입니다. 이 때 지우기는 던져두고 어떠한 파일이 하나의 섹터에 위치해있고 이 파일을 7 번 덮어쓰는 작업을 진행한다고 해보도록 하겠습니다.

HDD 에서는 해당 덮어쓰기 작업이 아래와 같은 식으로 이루어집니다. [볼은 색깔별로 각기 다른 데이터를 의미하며, 데이터의 기록은 해당 볼을 섹터에 쏘는 것으로 표현]

HDD 에서의 덮어쓰기와 그 결과

하지만 SSD 에서는 플래시 변환 계층의 웨어 레벨링 기능으로 인해 동일한 작업이 아래와 같은 식으로 이루어집니다.



SSD 에서의 덮어쓰기와 그 결과

즉, SSD 에선 웨어 레벨링으로 인해 덮어쓰기를 통한 보안 삭제 작업 자체가 원하는 페이지에 집중되지 못하고, 분산되어 버리기 때문에 별 의미가 없습니다. 결과적으로 그저 쓰레기 페이지만 양산할 뿐 그냥 삭제하는 것과 별반 다르지 않죠. 그래서 트림(TRIM) 기능이 활성화되어 있다면 그냥 삭제하시면 되며, 덮어쓰기 방식의 보안 삭제 프로그램은 사용하지 않는 것이 좋습니다.

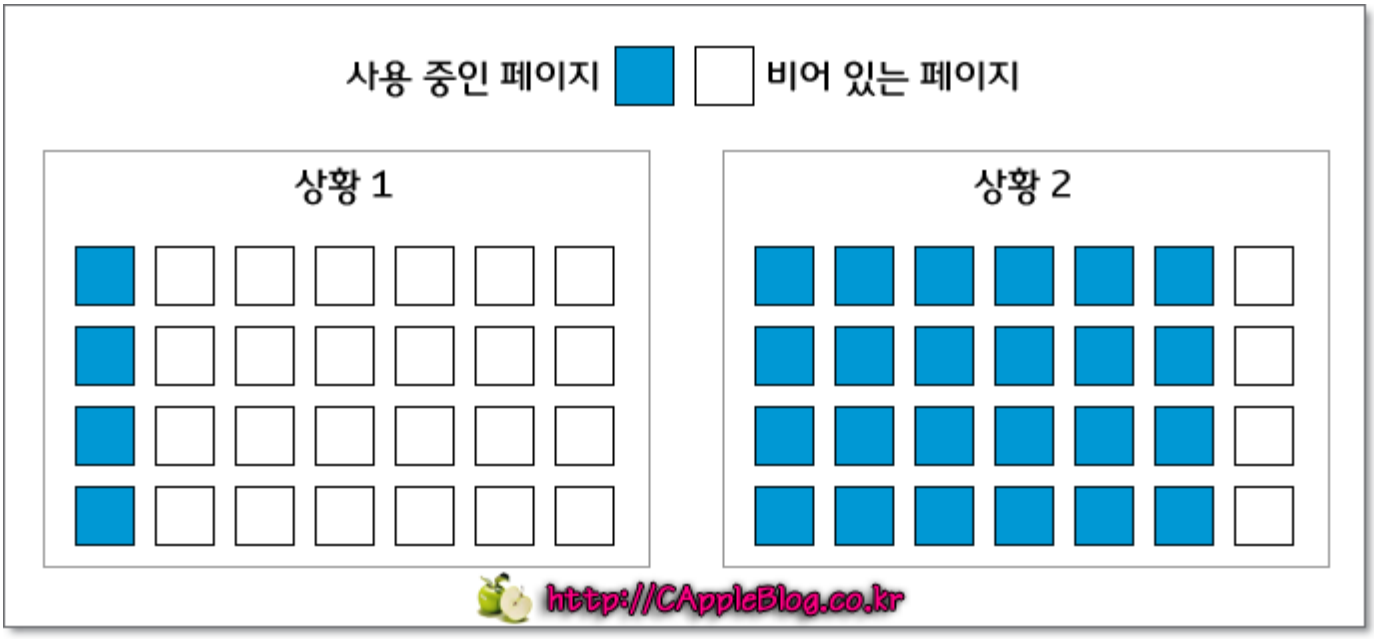
플래시 변환 계층에 포함된 웨어 레벨링과 가비지 컬렉션에 대한 설명은 여기까지 입니다. 참고로 SSD 에서는 성능의 향상을 위해 트림(TRIM) 기능이라는 것도 있는데, 이는 간단하게 윈도우에서 삭제한 파일을 SSD 에서도 실제로 삭제하는 기능으로, 운영체제와 연계되어 작동하는 기능이고 넓게 보면 가비지 컬렉션의 확장이라고 보셔도 좋습니다. 아무튼, 이번 글과는 크게 상관도 없고 이미 지난 SSD 의 특성과 TRIM 기능의 이해, 자동 TRIM(트림) 기능의 작동 여부 확인과 설정 방법 글에서 모두 설명하였기 때문에 여기에서는 따로 설명하지 않겠습니다.

아무튼, 이러한 웨어 레벨링과 가비지 컬렉션은 SSD 의 핵심이라고 보시면 됩니다. 웨어 레벨링 및 가비지 컬렉션과 같은 기능들이 얼마나 아름답게 구현되느냐에 따라 SSD 의 수명과 성능이 달라집니다. 그런데 이러한 주요한 기능들에는 치명적인 문제가 하나 있습니다.

오버 프로비저닝 공간의 이해 - SSD 의 성능 향상과 수명 연장을 위해
1. 오버 프로비저닝(Over Provisioning) 공간이란?

말했다시피 웨어 레벨링 및 가비지 컬렉션과 같은 기능들은 SSD 의 핵심입니다. 그리고 이러한 웨어 레벨링과 가비지 컬렉션의 목적은 "가능한 모든 페이지가 골고루 쓰여질 수 있도록 해주고, SSD 의 성능을 보장하며, 수명을 연장시키 것" 이죠.

그런데 웨어 레벨링 및 가비지 컬렉션과 같은 작업들은 그 작업 구조상 필연적으로 하나의 문제가 발생하게 됩니다. 간단하게 만약 SSD 의 저장 공간을 단순하게 페이지의 연속이라고 생각했을 때 아래의 두 가지 상황 중 어떠한 경우에 웨어 레벨링 및 가비지 컬렉션 작업을 진행하기에 더 이상적일까요?



당연하게도 비어 있는 페이지가 많은 상황 1 이 좋습니다. 상황 2 와 같은 경우는 특정 페이지들만 혹사를 당할 가능성이 매우 높으며, 이를 피하기 위해 보다 많은 작업들을 필요로 할 수 있습니다. 이는 분명 좋지 않은 모습이죠.

간단하게 정리하면 웨어 레벨링이나 가비지 컬렉션과 같은 작업들은 SSD 디스크의 여유 공간에 영향을 받는다고 할 수 있습니다. 만약 SSD 디스크가 거의 꽉 차게 되면(더티 상태) SSD 입장에서는 말 그대로 재앙이 시작되는 거죠. [이에 대한 대비가 충분하지 않거나, FTL 알고리즘이 좀 그렇거나]

그래서 SSD 에서는 여유 공간이 중요합니다. 여유 공간이 어느 정도 있어야 웨어 레벨링과 같은 기능들이 좀 더 원할하게 작동할테니까요. 그런데 SSD 를 사용하다 보면 분명 SSD 의 전체 공간이 꽉 채워지는 경우도 존재할 겁니다. 분명한 것은 SSD 는 그러한 상황에도 대비를 해야 한다는 겁니다.

그래서 SSD 에서는 오버 프로비저닝(Over Provisioning, OP) 공간이라는 것을 마련해두기도 합니다. [SSD 마다 다릅니다.]

더티 상태에서도 웨어 레벨링 작업이 효율적으로 진행되도록 도와주는 오버 프로비저닝 공간

즉, 오버 프로비저닝 공간은 웨어 레벨링, 가비지 컬렉션, 배드 블록 관리 등 SSD 를 운영하는데 필요한 핵심적인 기능들이 어떠한 상태에서도 원할하게 작동할 수 있도록 미리 마련된 예비 공간이라고 보시면 됩니다. 이렇게 오버 프로비저닝 공간이 마련되면 SSD 는 해당 오버 프로비저닝 공간을 활용하여 웨어 레벨링, 가비지 컬렉션과 같은 핵심 기능들을 최적화하게 됩니다. 이를 통해 SSD 의 성능이 항상 일정하게 유지되도록 해주고 수명 연장에도 도움을 주는 것이죠.

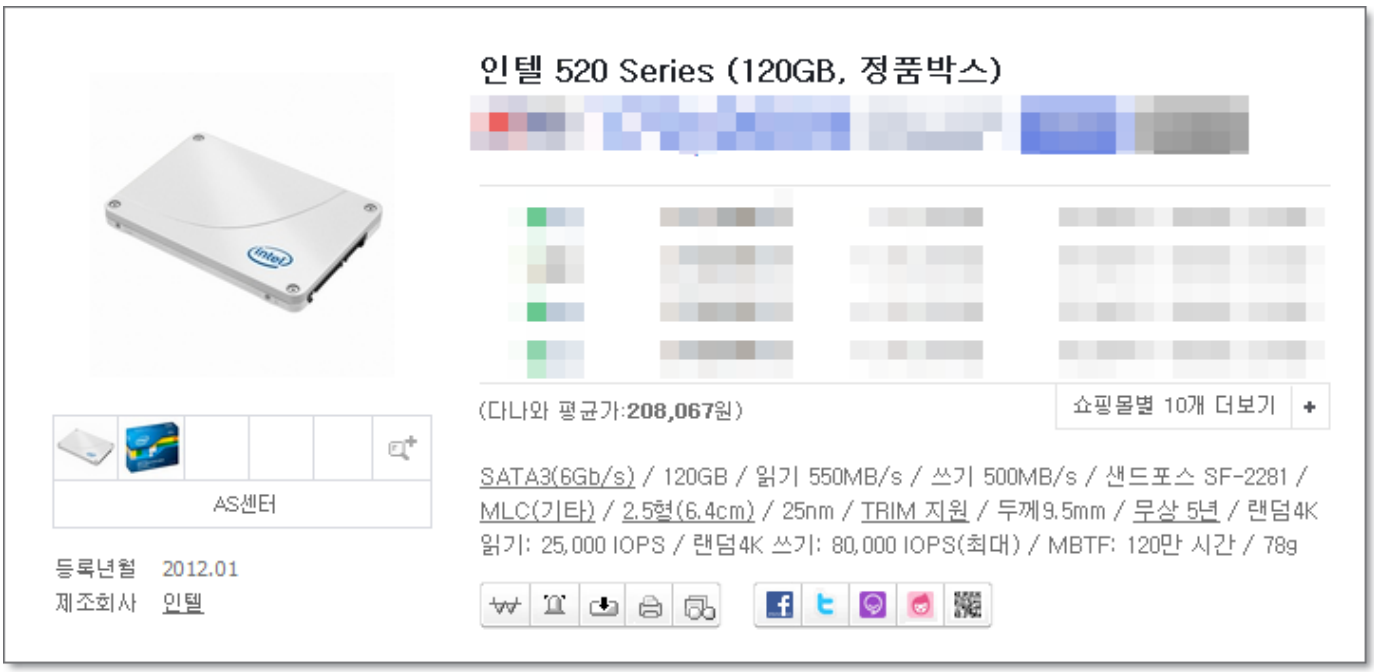
이러한 오버 프로비저닝 공간은 내부 오버 프로비저닝 공간과 외부 오버 프로비저닝 공간의 두 가지 형태가 있습니다.

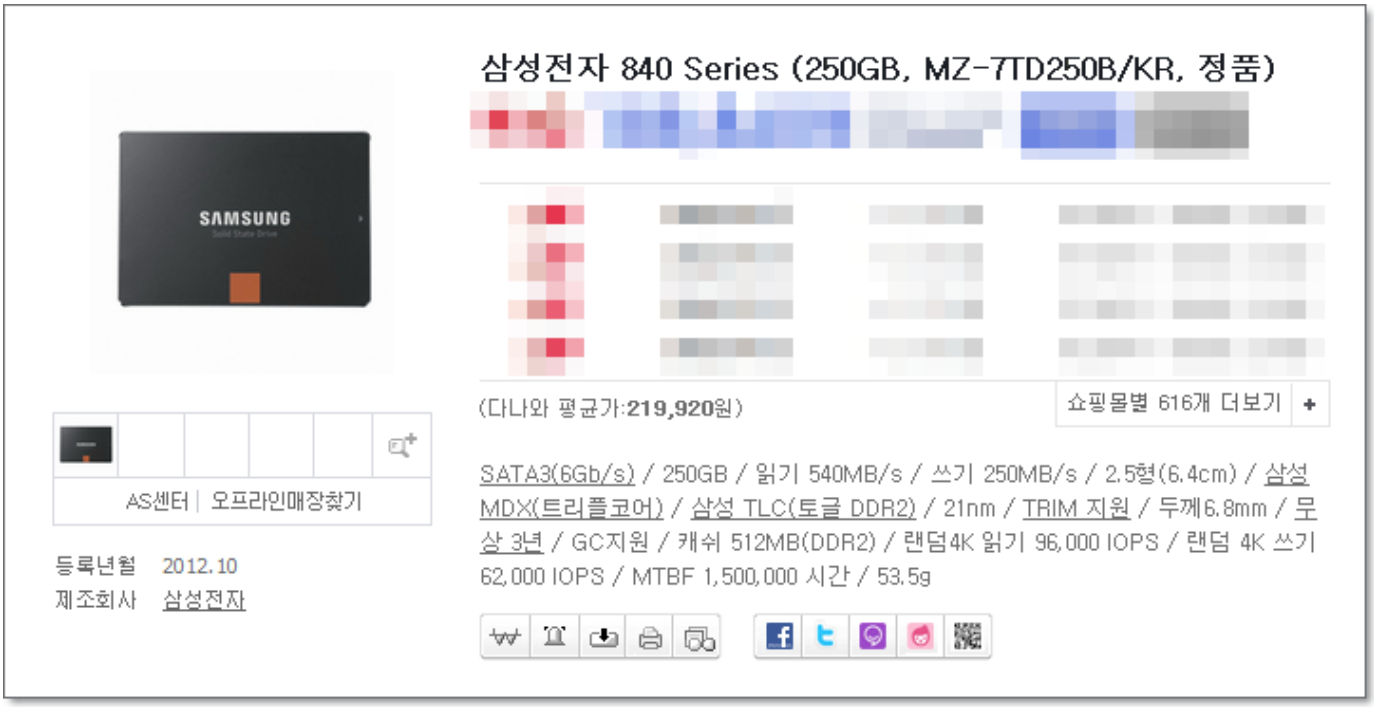
2. 내부 오버 프로비저닝 공간의 이해

먼저 내부 오버 프로비저닝 공간은 출시될 때부터 SSD 자체에 내장되어 오직 오버 프로비저닝 전용의 공간으로 사용되는 공간입니다. [흔하진 않지만 이를 일반 데이터 공간으로 전환할 수 있기도 합니다.] 즉, 제품 사양에 표기되는 용량 외에 숨어 있는 용량이 존재하고. 해당 공간이 바로 오버 프로비저닝 공간으로 활용되는 것이죠. 이러한 내부 오버 프로비저닝 공간은 SSD 의 장치 단계에서 설정되어 있는 공간이기 때문에 사용자가 직접 해당 공간의 용량이 정확하게 어떻게 되는지를 확인할 방법은 없습니다.

하지만 어떠한 제품이 얼마 정도의 내부 오버 프로비저닝 공간을 가지고 있을 거라고 예측은 할 수 있습니다. 일단 SSD 나 USB 플래시 메모리와 같이 낸드 플래시 메모리를 사용한 제품들은 특별한 경우를 제외하곤 낸드 플래시 메모리의 용량에 맞춰 2^X 의 용량을 가지거나 해당 용량의 배수로 조합할 수 있는 용량을 가지는 것이 일반적입니다. 쉽게 2GB, 4GB, 8GB, 16GB, 32GB, 64GBB, 128GB, 256GB, 512GB, 1TB..... 의 용량을 가지거나 또는 40GB, 80GB, 160GB, 320GB, 640GB..... 의 용량을 가지는 것이죠.

하지만 이러한 용량에 미묘하게 벗어나는 제품들이 있습니다. 일례로 60GB, 120GB, 250GB, 500GB 의 용량을 가진 제품들을 볼 수 있죠.





내부 오버 프로비저닝 공간이 적용된 제품들

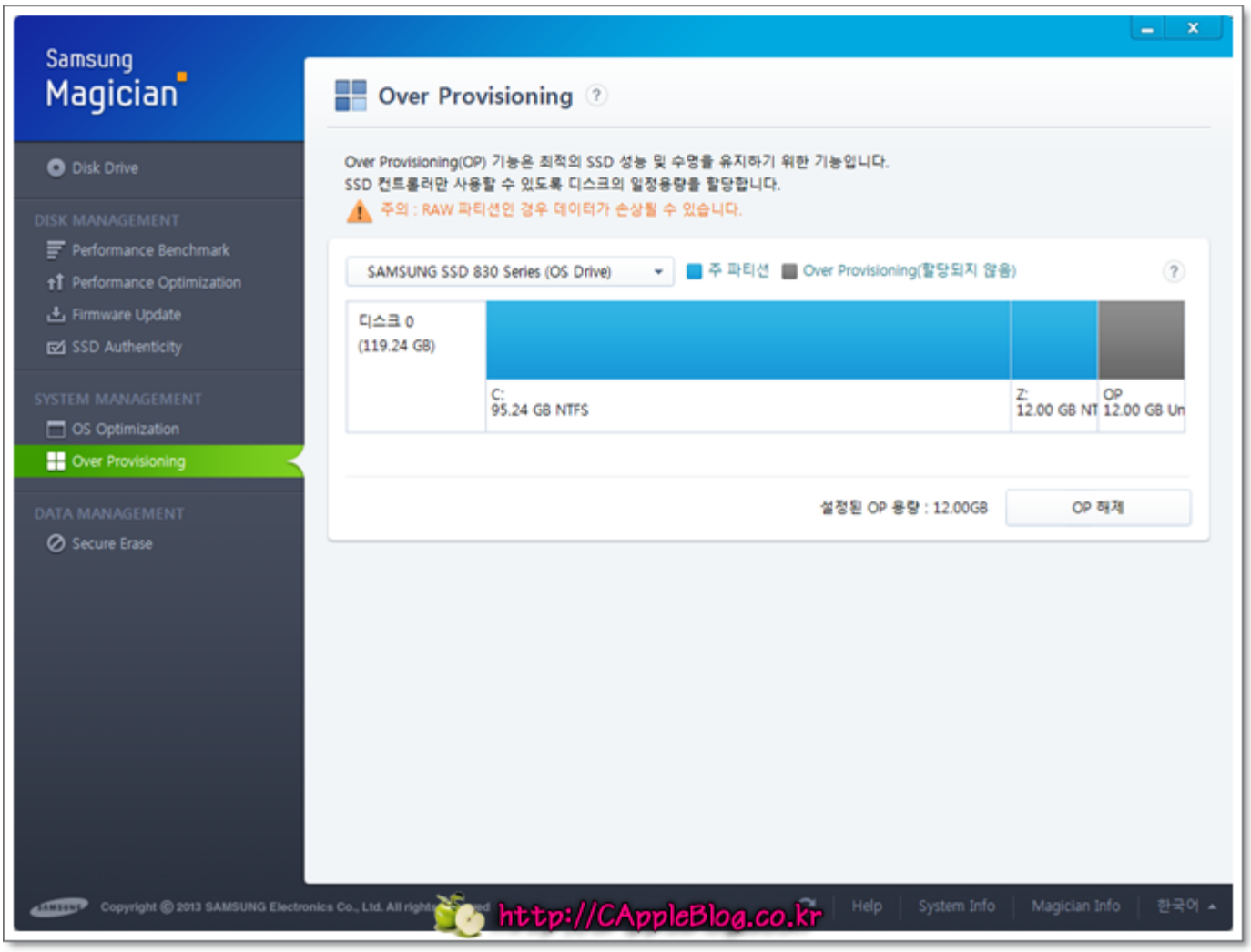
이러한 제품들은 모두 출시 단계에서 내부 오버 프로비저닝 공간이 설정된 제품이라고 보셔도 좋습니다. 흔히 사용되는 용량을 기준으로 120GB 제품은 128GB 에서 8GB 가 모자라고, 250GB 제품은 256GB 에서 6GB 가 모자라며, 500GB 제품은 512GB 에서 12GB 가 모자라죠? 그렇게 모자란 공간이 내부 오버 프로비저닝 공간으로 활용되고 있다고 보면 됩니다. [내부 오버 프로비저닝 공간이 용량에 맞춰 그만큼 퍼센테이지 비율로 나가지 않는 것은 대용량으로 갈 수록 이러한 오버 프로비저닝 공간의 필요성이 줄어들기 때문입니다.]

뭐 그렇습니다. 오버 프로비저닝 공간이라는 것 자체의 개념이 매우 간결하기 때문에 딱히 더 이야기드릴 것은 없네요. ^^;

3. 외부 오버 프로비저닝 공간의 이해

다음으로 외부 프로비저닝 공간은 앞서의 내부 오버 프로비저닝 공간과는 달리, 일반 데이터 공간과 오버 프로비저닝 공간이 따로 분리되어 있지 않고, 관리 프로그램을 통해 수동으로 설정하거나, 일반 데이터 공간에서 일부 공간이 미리 설정된 "어떠한 조건"에 맞으면 자동으로 해당 공간을 오버 프로비저닝 공간으로 사용하는 구조를 가지고 있습니다.

이러한 방식을 사용하는 제품으로는 대표적으로 삼성의 8X0 시리즈 제품들을 들 수 있습니다.



오버 프로비저닝 공간의 설정을 지원하는 SSD 관리 프로그램인 삼성 매지션(Samsung Magician)의 모습

해당 제품들은 SSD 관리 프로그램에서 오버 프로비저닝 공간을 설정하는 것을 도와주지만, 해당 관리 프로그램에서 제공하는 오버 프로비저닝 공간의 설정이란 것이 실제론 매우 단순한 파티션 축소(볼륨 축소) 기능이고, 이렇게 파티션 축소를 통해 확보된 할당되지 않은 공간을 오버 프로비저닝 공간으로 활용하는 것에 불과합니다. 그래서 해당 관리 프로그램을 통하지 않고, 윈도우의 디스크 관리나 DiskPart 와 같은 다른 파티션 관리 도구를 통해 할당되지 않은 공간을 미리 만든 후 관리 프로그램을 실행하면 위와 같이 자동으로 해당 할당되지 않은 공간을 오버 프로비저닝 공간으로 인식하는 것을 확인할 수 있습니다.

참고로 일단 정식 명칭은 아니지만 이러한 할당되지 않은 공간을 남겨두는 것은 일명 꼬리 자르기로 통용되고 있습니다. SSD 의 일부 영역을 할당되지 않은 영역으로 남겨두어 여유 공간을 일정하게 확보하는 것인데 이 또한 일종의 오버 프로비저닝이라고 보시면 됩니다. [컨트롤러가 그것을 얼마나 제대로 활용할 지는 컨트롤러 설계에 달려 있지만] 대체로 디스크의 마지막 공간을 할당되지 않는 공간으로 남겨두는 것이 쉽기 때문에 대부분 그렇게 작업하고, 윈도우의 디스크 관리에서 보자면 디스크의 마지막 공간을 자르는 것이기 때문에 꼬리

자르기라고 부르고 있습니다.

뭐 그렇습니다. 어떻게 오버 프로비저닝 공간에 대해서 이해가 되셨는지 모르겠네요. 아무튼, 오버 프로비저닝을 적용하든(적용되었든) 안 하든, 중요한 것은 SSD 에서는 웨어 레벨링 및 가비지 컬렉션과 같은 핵심적인 작업들이 원할하게 이뤄질 수 있도록 일정한 공간이 유지되어야 한다는 겁니다. 그래서 결국 모든 저장 장치들이 공통적으로 그러하듯 SSD 도 용량 많은 게 장땡입니다. ^^;

마치며...

요즘은 SSD 가 참 많이 보급되었습니다..이미 많은 분들께서 SSD 를 OS 용 디스크로 활용하고 계시니까요. 하지만 아직까진 SSD 를 사용하는데에 약간의 혼란은 있는 듯 합니다. 그래서 제 나름대로 SSD 를 이해하는데에 꼭 필요하다고 생각되는 부분들을 간추려서 정리를 해보았습니다. 몇 편 되지는 않지만 어떻게 이러한 저의 정리가 도움이 되었을지 모르겠네요.

아무튼, SSD 를 사용하면서 기억하실 것은 사실 별로 없습니다. 그저 기존의 HDD 를 사용할 때와 같이 그대로 사용하시면 되는 거죠. 그렇지만 기존까지 통용되던 어떠한 정보들은 SSD 장치의 특성과는 맞지 않는 부분들이 있는 것도 분명합니다. 대표적으로 조각 모음이나 보안 삭제와 같은 부분들이 있죠. 뭐 사실 그러한 몇 가지 부분들만 주의하시면 SSD 를 사용하면서 크게 주의해야 할 것은 없다고 보아도 좋습니다. 그래서 딱히 "SSD 는 이렇게 사용해라", "SSD 는 이렇게 최적화해라" 라고 콕 찝어서 말하기엔 좀 그렇네요. 제가 그런 쪽으로 정리하는 재주가 없기도 하고요. [그래도 다른 건 몰라도 AHCI 모드 설정은 꼭 설정하시길 바랍니다. SSD 자체가 AHCI 모드에 최적화되어 있습니다.]

뭐 그렇다는 거죠. ^^;;; 그럼 이번 글은 이쯤에서 마무리하도록 하겠습니다. 이상입니다. ^^