Realizacija datotečnog sustava na SSD pogonu

Večenaj, Matija

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Matija Večenaj

REALIZACIJA DATOTEČNOG SUSTAVA NA SSD POGONU

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Matija Večenaj

Matični broj: 44855/16-R

Studij: Informacijski sustavi

REALIZACIJA DATOTEČNOG SUSTAVA NA SSD POGONU

ZAVRŠNI RAD

Mentor:

Izv. Prof. Dr. Sc. Igor Balaban

Varaždin, kolovoz 2019.

Matija Večenaj

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema ovog rada jest datotečni sustav i njegova realizacija na SSD pogonu. Najprije se u razradi obrazlažava uloga flash memorije bez koje SSD pogon ne bi bilo moguće realizirati. Nakon toga, slijede općenite informacije o SSD pogonu kao što je povijest razvoja, karakteristike, usporedba sa starijim tehnologijama i podjela istih s obzirom na sučelja. Objasnit će se i arhitektura SSD-a, s posebnim naglaskom na kontroler i memoriju. Sada slijede ključni dijelovi rada, a prvi je datotečni sustav. Pojašnjeno je značenje istog kod ovog medija i dan je pregled i pojašnjenje datotečnih sustava koji se koriste na SSD-ovima i u različitim operacijskim sustavima. Sljedeće, objašnjena je NAND flash tehnologija, uz pomoć koje je većina današnjih SSD-ova realizirana. Objašnjene su njezine specifičnosti, nedostaci i prednosti kao i razne izvedbe i usporedba s ostalim memorijama. Naposljetku, slijedi razrada važnog firmwarea FTL-a koji je odgovoran za kvalitetu performansi SSD pogona i objašnjene su njegove zadaće i funkcionalnosti.

Ključne riječi: SSD; flash memorija; datotečni sustav; FTL; NAND; pohrana

Sadržaj

1. Uvod

Solid-state drive (SSD) je medij za pohranu podataka temeljen na integriranim krugovima. Iz naziva istog, vidi se da SSD ne sadrži disk, što ukazuje da se ne sastoji od mehaničkih pokretnih dijelova kao danas pomalo zastarjeli čvrsti disk (HDD). SSD je zapravo nasljednik HDD-a budući da donosi bolje performanse (brzine čitanja i pisanja), veću izdržljivnost, veću energetsku učinkovitost kao i manju masu i dimezije. Ipak, SSD pogon ima i nedostataka, a to je trenutno veća cijena po gigabajtu memorije.

SSD je izgrađen od memorijskog kontrolera i *flash* memorije. Memorija može biti realizirana u NOR ili NAND obliku. U NAND obliku, SSD-ovi pohranjuju podatke u ćelije koje mogu pohraniti od 1 do 4 bita podataka (SLC, MLC, TLC, QLC). SSD-ovi koji se temelje na NAND Flash-u polagano će „odumirati“ tijekom vremena ako budu ostavljeni na dulje vrijeme bez napajanja. To uzrokuje da istrošeni pogoni počnu gubiti podatke obično nakon jedne godine [1]. SSD-ovi mogu koristiti tradicionalna sučelja za tvrdi disk ili novija sučelja koja iskorištavaju određene prednosti flash memorije u SSD-ovima. Tradicionalna sučelja (npr. SATA i SAS) i standardni HDD formati omogućuju takve SSD-ove da se koriste kao zamjene za HDD-ove u računalima i drugim uređajima. Novije inačice formata kao što su mSATA, M.2, U.2 i EDSFF i sučelja većih brzina kao što je NVMe preko PCI Expressa mogu povećati performanse u odnosu na performanse HDD-a [2].

2. Metode i tehnike rada

Prilikom pisanja ovoga rada, korišteni su uglavnom internetski izvori. Neki od njih su bili službeni članci, neki su bili znanstveni radovi, a neki su bili javno dostupni članci koji su mogli biti uređivani. Analizom i razradom tih izvora nastao je ovaj rad. Od alata koji su se koristili jedino je bitno spomenuti da je pisan u *Microsoft Word*-u.

3. Razrada

3.1. Flash memorija

3.1.1. Općenito

Flash memorija je vrsta trajne memorije koju je moguće bristati i na nju ponovno pisati, a koja briše podatke u jedinicama koje zovemo blokovi. Blok pohranjen na flash memorijskom čipu mora biti obrisan prije no što se podaci mogu zapisivati ili mijenjati na čipu. Koristi se za poslužitelje, pohranu i u mrežnoj tehnologiji, kao i u širokom rasponu uređaja, uključujući USB flash pogone, mobilne telefone, digitalne fotoaparate, tablet računala, PC kartice u prijenosnim računalima i ugrađenim kontrolerima. Na primjer, NAND flash-bazirani SSD pogoni se često koriste za ubrzavanje ulazno-izlaznih performansi intenzivnih aplikacija. NOR flash memorija se često koristi za držanje kontrolnog koda, kao što je osnovni ulazno - izlazni sustav (BIOS), na računalu. Sve više se koristi za računanje u memoriji kako bi se ubrzala učinkovitost i povećala skalabilnost sustava koji upravljaju i analiziraju sve veće skupove podataka [3].

Dr. Fujio Masuoka zaslužan je za izum flash memorije kada je radio za Toshibu 1980-ih. Flash memorija nastala je iz izbrisive programabilne memorije samo za čitanje (EPROM) i električno izbrisive programibilne memorije samo za čitanje (EEPROM). Flash je tehnički varijanta EEPROM-a, ali industrija zadržava pojam EEPROM za izbrisivu memoriju na razini bajta i primjenjuje izraz flash memorija za veću izbrisivu memoriju temeljenu na blokovima. Uređaji koji koriste flash memoriju brišu podatke na razini bloka i prepisuju podatke na razini bajta (NOR flash) ili višestruke bajtne stranice (NAND flash) [4].

3.1.2. NOR i NAND

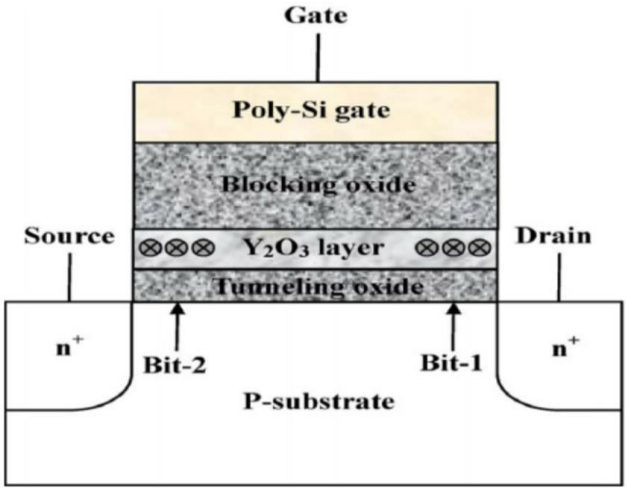
NOR flash memoriji je potrebno više vremena za brisanje i pisanje, ali pruža poptunu adresu preko koje je moguće nasumično pristupiti bilo kojoj memorijskoj lokaciji. To ju čini prikladnom zamjenom za starije čipove memorije samo za čitanje (ROM), koji se koriste za pohranu programskog koda koji se rijetko treba ažurirati, kao što je BIOS računala ili firmware. Izdržljivost može biti samo 100 ciklusa brisanja za flash memoriju na čipu, do ipak tipičnijih 100 000 ciklusa brisanja [3]. NOR flash je temelj za rane flash bazirane prijenosne medije. NOR flash brzo čita podatke, ali je uglavnom sporiji od NAND-a kada je u pitanju brisanje i pisanje. NOR flash je također skuplji za proizvodnju od NAND-a. Dvije glavne vrste NOR flash memorije su paralelne i serijske. NOR flash je izvorno bio dostupan samo s paralelnim

sučeljem. Paralelni NOR nudi visoke performanse, sigurnost i dodatne značajke. Primarna namjena uključuje industrijske, automobilske, mrežne i telekomunikacijske sustave i opremu. Serijska NOR memorija ima manji broj pinova i manje pakiranje, što ga čini jeftinijim od paralelnog NOR-a. Primjeri za serijski NOR uključuju osobna i ultra-tanka računala, servere, HDD-ove, pisače, digitalne kamere, modeme i usmjerivače.

NAND flash memorija je smanjila vrijeme brisanja i pisanja i zahtijeva manju površinu za čip na ćeliji, što omogućuje veću gustoću pohrane i nižu cijenu po bitu nego NOR flash. Također ima i do 10 puta veću izdržljivost NOR-a. Međutim ulazno – izlazno sučelje NAND flash memorije ne sadrži vanjsku adresnu sabirnicu za nasumični pristup. Umjesto toga, podaci se moraju čitati po stranicama, koje su veće od bajtova, ali manje od blokova. Na primjer, stranica može biti 4KB, a blok 128KB. NAND također troši manje električne energije od NOR-a za operacije koje zahtijevaju mnogo zapisivanja [4]. To čini NAND flash neprikladnim kao zamjena za programski ROM, budući da većina mikroprocesora i mikrokontrolera zahtijeva slučajni pristup. Stoga, NAND flash je sličan drugim sekundarnim uređajima za pohranu podataka, kao što su tvrdi diskovi i optički mediji, te je zato vrlo pogodan za upotrebu u uređajima za masovno pohranjivanje, kao što su memorijske kartice.

3.1.3. Karakteristike

Arhitektura flash memorije uključuje niz memorija s velikim brojem flash ćelija. Osnovna ćelija flash memorije sastoji se od tranzistora za pohranu s kontrolnim vratima i plutajućim vratima, koja je izolirana od ostatka tranzistora tankim dielektričnim materijalom ili oksidnim slojem. Plutajući ulaz pohranjuje električni naboj i kontrolira protok električne struje. Elektroni se dodaju ili uklanjaju iz plutajućeg vrata kako bi se promijenio napon praga tranzistora za pohranu. Promjena napona utječe na to je li ćelija programirana kao nula ili jedan [4].



Slika 1. Arhitektura ćelije [6]

Flash je najjeftiniji oblik poluvodičke memorije. Za razliku od dinamičke memorije s izravnim pristupom (DRAM) i statičkog RAM-a (SRAM), flash memorija je trajna, nudi manju potrošnju energije i može se izbrisati u velikim blokovima. Također NOR flash nudi brzo slučajno čitanje, dok je NAND flash brza kod serijskog čitanja i zapisivanja. SSD s NAND flash memorijskim čipovima pruža znatno bolje performanse od tradicionalnih magnetskih medija, kao što su HDD i traka. Flash diskovi također troše manje energije i proizvode manje topline od HDD-ova [3].

Glavni nedostaci flash memorije su specifičan način potrošnje i interferencija između ćelija kako se matrice smanjuju. Bitovi mogu biti pokvareni s vrlo čestim pisanjima i brisanjima, koji na kraju razgrade oksidni sloj koji će zarobiti elektrone. Elektroni mogu izaći ili se zaglaviti u oksidnom izolacijskom sloju što rezultira greškama i zatajenju bitova.

Danas, proizvođači flash pogona imaju poboljšanu izdržljivost i pouzdanost kroz algoritme za ispravljanje pogrešaka, izravnavanje trošenja i senzora koji upozoravaju na trošenje medija.

3.2. Općenito o SSD pogonu

3.2.1. Povijest

Najraniji SSD pogoni uglavnom su dizajnirani za potrošačke uređaje. Debi *Apple iPod*-a u 2005. godini označio je prvi značajan uređaj temeljen na flashu koji široko prodire na tržište potrošača. *Dell EMC* poznat je kao prvi proizvođač koji je uključio SSD-ove u hardver za pohranu u poduzeću kada je 2008. dodao tu tehnologiju svojim nizovima diskova *Symmetrix* [1]. To je bio događaj koji je označio uporabu hibridnih nizova diskova koji su bili sačinjeni od flash pogona i HDD-ova. U većini slučajeva, korporacijski SSD-ovi u hibridnim nizovima koriste se za caching u flashu. To je zbog veće cijene i manje izdržljivosti SSD-a u usporedbi s HDD-ovima. Najraniji komercijalno dizajnirani SSD diskovi izrađeni su pomoću poslovne *multi-level cell (enterprise MLC)* flash tehnologije, koja ima poboljšane cikluse pisanja u usporedbi s potrošačkim MLC. Na tržištu se prodaju noviji SSD-ovi poduzeća s troslojnim ćelijama (TLC). SSD-ovi napravljeni s 3D NAND-om predstavljaju sljedeću točku u razvoju SSD pogona. IBM, Samsung i Toshiba proizveli su i prodali SSD-ove s 3D NAND-om, u kojima su ćelije flash memorije postavljene jedna na drugu u vertikalnim slojevima. Stručnjaci tvrde da SSD-ovi počinju mijenjati tradicionalne diskove u nekim slučajevima uporabe, iako se očekuje da će flash pogoni i HDD-ovi zajedno koristiti u mnogim poduzećima u bliskoj budućnosti. Na primjer, SSD-ovi su prilagođeni za visoke performanse, ali manje za dugoročno arhiviranje i backup, za koji se obično koristi čvrsti disk [1] .

3.2.2. Značajke

Postoji nekoliko značajki koje karakteriziraju dizajn SSD-a. Budući da ne koristi pokretne dijelove, SSD ne podliježe mehaničkom kvaru koji se javlja kod tvrdih diskova. Također je tiši i troši manje energije nego HDD. Budući da SSD-ovi teže manje od tvrdih diskova, oni dobro pristaju za prijenosna i mobilna računala. Osim toga, softver SSD kontrolera uključuje analitiku predviđanja koja upozorava korisnika unaprijed o mogućem kvaru pogona. Flash memorija na SSD-ovima se obično izrađuje s jednom razinom ćelije (SLC) ili više razina na ćeliji (MLC) . SLC pogoni pohranjuju 1 bit podataka po ćeliji flash medija. SSD-ovi bazirani na MLC-u udvostručuju kapacitet pogona pisanjem podataka u dva segmenta. Noviji SSD-ovi, poznati kao TLC, prodaju se kao takvi na kojima se pohranjuju 3 bita podataka po flash ćeliji. TLC je jeftiniji od SLC-a ili MLC-a, što ga čini atraktivnom opcijom za proizvođače flash uređaja na potrošačkoj razini. SSD-ovi temeljeni na TLC-u daju veći kapacitet i jeftiniji su od MLC ili SLC-a, iako s većom vjerojatnošću za odumiranjem bitova. U 2018. godini Intel i Micron uveli su četveroslojni NAND [2].

3.2.3. Usporedba SSD-ova i HDD-ova

Tradicionalna testiranja HDD-ova imaju tendenciju da se usredotoče na karakteristike performansi koje su loše sa HDD-ovima, kao što je latencija rotacije i vrijeme traženja. Budući da se SSD-ovi ne moraju vrtjeti niti tražiti podatke, oni se mogu pokazati daleko boljim u odnosu na HDD-ove u takvim testovima [1]. Međutim, SSD uređaji imaju poteškoće s mješovitim čitanjima i zapisima, a njihova se učinkovitost s vremenom može smanjiti. Većina prednosti SSD-ova u odnosu na tradicionalne tvrde diskove je zbog njihove sposobnosti da pristupaju podacima potpuno elektronički umjesto elektromehanički, što rezultira većim brzinama prijenosa. S druge strane, tvrdi diskovi nude znatno veći kapacitet za svoju cijenu.

Neki testovi pokazuju da su SSD-ovi znatno pouzdaniji od HDD-a, ali drugi pak obrnuto. Međutim, SSD diskovi su jedinstveno osjetljivi na iznenadni prekid napajanja, što rezultira prekinutim zapisima ili čak slučajevima potpunog gubitka pogona. Pouzdanost HDD-ova i SSD-ova uvelike varira među modelima. Kao i kod tvrdih diskova, postoji razlika između troškova i performansi različitih SSD-ova. SSD-ovi na jednoj razini ćelija (SLC), iako su znatno skuplji od onih na više razina ćelija (MLC), nude značajnu prednost u brzini. Kada je riječ o HDD-ima, ponovno zapisana datoteka će općenito zauzeti isto mjesto na površini diska kao i izvorna datoteka, dok će u SSD-ovima nova kopija često biti zapisana na različite NAND ćelije u svrhu izjednačavanja trošenja. Algoritmi za izjednačavanje trošenja su složeni i teško ih je iscrpno testirati [1]. Kao rezultat toga, jedan od glavnih uzroka gubitka podataka u SSD-ovima su bugovi firmware-a.

Brojni čimbenici utječu na vijek trajanja SSD-ova i HDD-ova, uključujući vlažnost i oksidaciju metala unutar pogona. Podaci kod oba tipa medija s vremenom će se degradirati, a HDD-ovi općenito podržavaju veći broj pisanja na pogon dnevno. Kod SSD-ova, flash memorija podržava ograničen broj zapisa na pogon dnevno. Razina zadržavanja podataka se smanjuje kako se sve više podataka zapisuje u flash ćelije. Pokretni dijelovi HDD-ova povećavaju šanse za kvarove, pa proizvođači HDD-a to nadoknađuju senzorima za zaštitu pogona i drugih komponenti u računalnim uređajima. Izlaganje toplini je još jedan čimbenik koji utječe na život pogona, posebno za SSD-ove. Kada SSD radi na visokim temperaturama tijekom duljeg vremenskog razdoblja, to može doprinijeti curenju elektrona iz NAND flash memorije.

Tablica 1. Usporedba svojstava HDD-a i SSD-a (prema [1])

Svojstvo (obilježje)	SSD	HDD
Cijena	Skuplji, iako je prisutan pad cijena	Jeftiniji, cijena pala, ali u manjem postotku
Kapacitet	Do 100 TB	Do 14 TB
Pouzdanost	Bez napajanja, istrošeni SSD-ovi gube podatke nakon jedne do dvije godine (ovisno o temperaturi) pa nisu namijenjeni arhiviranju podataka	U suhoj i hladnoj okolini, HDD-ovi mogu čuvati podatke vrlo dugo, ali mehanički dijelovi postanu nepouzdani i disk se neće pokrenuti nakon nekoliko godina u arhivi
Vrijeme pokretanja	U jednom trenutku – nekoliko milisekundi ako se izlazi iz načina za čuvanje energije	Može trajati nekoliko sekundi, ovisi i o broju diskova
Brzina sekvencijalnog pristupa	Tipična najveća brzina prijenosa mjeri se između 200 MB/s i 3500 MB/s , ovisno o modelu	Pozicioniranje glave i čitanje staze uzrokuje brzine oko 200 MB/s. Također ovisi o brzini okretaja: 3600 – 15000 RPM
Brzina nasumičnog pristupa	Uobičajeno ispod 0.1 milisekunde	Mnogo dulje vrijeme čitanja, između 2.9 ms pa do 12 ms
Fragmentacija	Fragmentacija je na SSD-ovima zanemariva; defragmentacija uzrokuje potrošnju ćelija zbog dodatnih pisanja po NAND flashu	Neki datotečni sustavi (npr. Windowsov NTFS) postaju fragmentirani nakon čestih pisaja, pa je defragmentacija potrebna za očuvanje optimalnih performansi
Buka	Nepostojeća, nema mehaničkih dijelova	Karakteristični zvukovi vrtnje i kliktanja
Temperature	Česti kvarovi između 30 C i 40 C , a oko 70 C javalja se zastajkivanje	Temperature iznad 35 C skraćuju životni vijek diska, a iznad 55 C dolazi do smanjenja pouzdanosti.
Omjer performansi čitanja i pisanja	Jeftiniji pogoni obično imaju puno sporije brzine pisanja od čitanja, inače ujednačeni	HDD-ovi uglavnom imaju dulja vremena traženja za pisanje nego za čitanje
Potrošnja energije	Otprilike upola manje nego HDD	Uobičajeni 3.5 inčni HDD-ovi troše do 20W energije

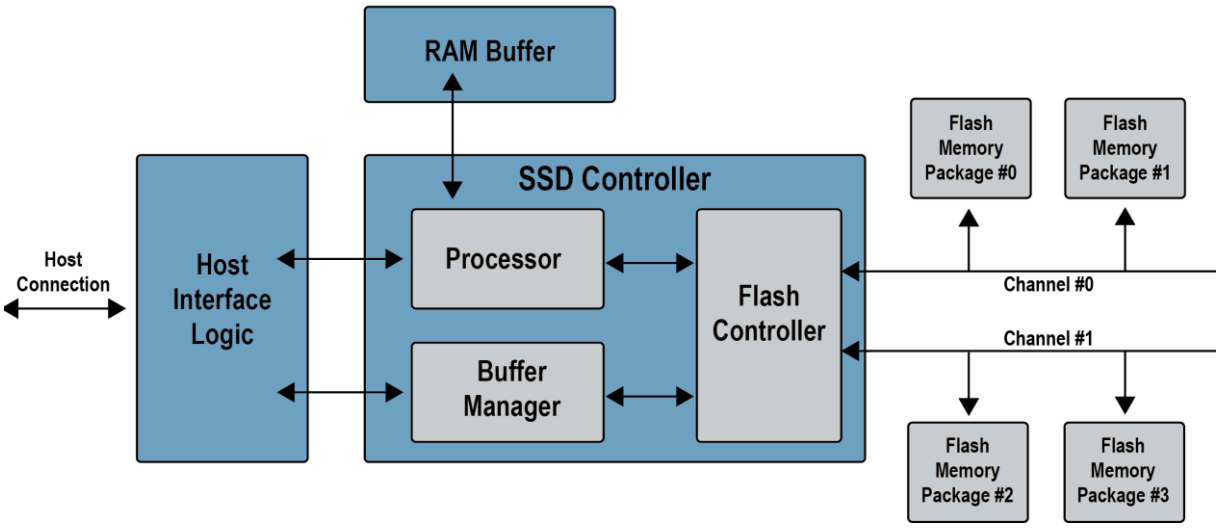
3.2.4. Oblici

SSD-ovi nemaju fizička ograničenja HDD-a. To omogućuje proizvođačima pogona da nude SSD-ove u različitim oblicima. Najčešći faktor je 2.5-inčni SSD koji je dostupan u više visina i koji podržava *serial-attached SCSI* (SAS*), Serial Advanced Technology Attachment* (SATA) i *nonvolatile memory express* (NVMe) protokole. Ovo su pet vrsta SSD-ova grupiranih prema sučelju [5]:

1) **SATA SSD** - SATA je sučelje koje je kompatibilno s HDD-ovima i SSD-ovima koji se temelje na SATA-i. Najnovija SATA iteracija je SATA III, koja donosi brzine prijenosa od 6Gb / s i obično dolazi u 2,5-inčnom obliku. Od svog dolaska na tržište 2003. godine, SATA je prošla nekoliko revizija kako bi pratila zahtjeve performansi SSD-a tehnologija.
2) **mSATA SSD** - SATA Revision 3.1 je predstavio mini-SATA (mSATA) na mobilnim računalnim uređajima kao što su prijenosna računala i *utlrabooks* u 2011. MSATA konektor izgleda slično PCI Mini Card konektoru, ali nije funkcionalan ako je priključen na PCIE utor. Iako još uvijek možemo pronaći mSATA SSD-ove, bolje dizajnirani M.2 ih je na kraju zamijenio.
3) **SATA Express -** SATA Express (SATAe) nastao je pod SATA Revision 3.2 i kompatibilan je s SATA i PCIE pogonima. Na priključak (utor na matičnoj ploči) može stati dva SATA 3.5 ”pogona ili dva PCIE SSD-a, ovisno o vrsti brzine. Međutim, treba napomenuti da je specifikacija SATAe sada zastarjela i to zbog M.2 sučelja kao i toga da većina matičnih ploča nije uključivala dotične priključke.
4) **PCIE SSD** - PCIE utori mogu sadržavati od X4 do X16 staza (ili više), ovisno o matičnoj ploči. PCIE SSD-i se spajaju na ove utore, omogućujući postizanje iznimnih brzina prijenosa koje lako nadmašuju najbolje SATA pogone. PCIE 2.0 pogon, na primjer, može postići 2GB/s sa samo jednom PCIE stazom.
5) **M.2** - Izvorno nazvan *NGFF (Next Generation Form Factor)*, M.2 konektori se mogu pronaći na mobilnim uređajima i na PC matičnim pločama. M.2 konektori su također kompatibilni sa SATA III i PCIE uređajima, što ih čini svestranijim. Naravno, ako postoji konektor, onda mora postojati i uređaj. M.2 pogoni mogu biti specifični za SATA ili PCIE i smatraju se budućnošću SSD-ova. M.2 oblici temelje se na mjeri istih u milimetrima.

3.3. Arhitektura SSD pogona

SSD pogon sastoji se, između ostalog, od dva ključna dijela, a to su kontroler i memorija koja pohranjuje podatke. Ostali dijelovi mogu na primjer biti pretvarači istosmjerne struje koji može služiti kao unutarnje napajanje za pogon, a često su prisutni i senzori za mjerenje temperature samog pogona. SSD-ovi temeljeni na flashu često koriste malu količinu DRAM-a kao priručnu memoriju slično kako se koristi i kod međuspremnika HDD-ova. Uobičajeni memorijski sustav je sastavljen od nekoliko NAND memorija. Također se uobičajeno koristi 8 bitna sabirnica (koju se naziva kanalom) koja spaja različite memorije sa kontrolerom. Operacije u tim kanalima mogu biti izmjenične. Na primjer, niz od višestrukih operacija pisanja mogu se usmjeriti na isti kanal komunicirajući s različitim NAND memorijama [7]. Zahvaljujući njima, propusnost kod SSD-ova je znatno unaprijeđena.



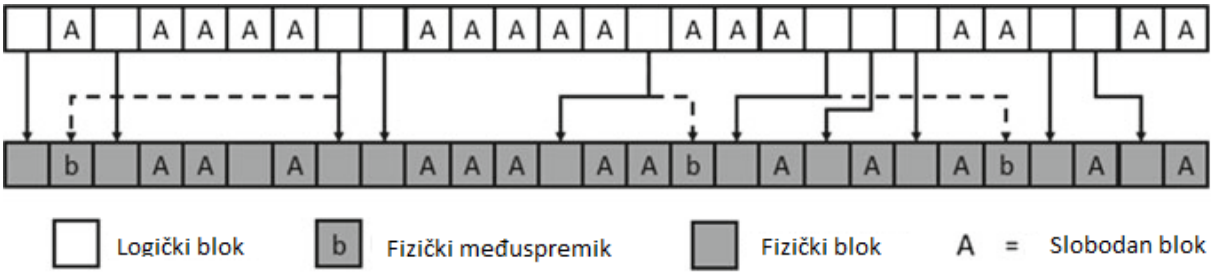
Slika 2. Arhitektura SSD pogona [8]

3.3.1. Kontroler

Svaki SSD uključuje kontroler koji objedinjuje elektroniku koja spaja komponente NAND memorije sa samim računalom. Kontroler je ugrađeni procesor koji izvršava kod na razini firmwarea i jedan je od najvažnijih čimbenika performansi SSD-a [1]. Dakle, kontroler je odgovoran za dvije glavne zadaće: osigurati najoptimalnije sučelje i protokol prema računalu i flash memorijama i učinkovito upravljanje podacima što podrazumijeva osiguravanje najveće brzine prijenosa, integriteta podataka i zadržavanje pohranjenih informacija. Da bi mogao izvršiti navedene zadaće, izrađen je aplikacijski specifičan uređaj koji objedinjuje 8 i 16 bitni procesor zajedno sa nezavisnim sklopovljem za izvršavanje vremenski kritičnih zadataka. Kontroler se može generalno podijeliti na četiri dijela, koji su implementirani ili u sklopovlju ili u firmwareu.

Gledajući od računala prema flashu, prvi dio je sučelje računala. Ono implementira potreban industrijski sandardiziran protokol (*PCIe, SAS, SATA..*) koji omogućuje ostvarivanje logičke i fizičke komunikacije i zapravo interoperabilnosti između SSD-a i računala. Ovaj je blok mješavina sklopovlja (međuspremnici, upravljački programi...) i firmwarea , koji dekodiraju naredbeni niz kojeg poziva računalo i upravlja tokom podataka u i iz flash memorije [7].

Drugi je dio flash datotečni sustav (*Flash File System – FFS*) koji je odgovoran za samo korištenje SSD pogona kako je i slučaj sa HDD-ovima. Flash datotečni sustav je uobičajeno implementiran u obliku firmwarea unutar kontrolera i to na način da svaka razina odrađuje određenu funkciju. Glavne funkcije su: upravljanje iskorištenošću (*wear leveling management)* , sakupljanje smeća (*garbage collection*) i upravljanje neupotrebljivim blokovima (*bad sector management*) [7]. Za sve te funkcije, često se koriste tablice pomoću kojih se mapiraju sektori i stranice iz logiče u fizičku domenu (flash translacijski sloj – FTL) kao prema slici 3 dolje. Gornji red je logički pogled memorije,a donji red jest onaj fizički. Iz perspektive računala, podaci su transparentno pisani i prepisani unutar zadanog logičkog sektora. Ipak, zbog ograničenja FLT-a, pisanje preko iste stranice nije moguće pa je potrebno alocirati novu stranicu (sektor) u fizičkom bloku, a prethodni označiti kao nevažeći. Stoga je jasno da će doći do trenutka kada će se trenutni fizički blok napuniti i javiti će se potreba za drugim blokom koji se uzima iz već predodređenih i spremnih blokova koji će preuzeti tu logičku adresu. Sve potrebne translacijske tablice su pohranjene na samom SSD pogonu, pa to znači da je na istom smanjena kapaciteta za pohranu podataka .



Slika 3. Upravljanje logičkim i fizičkim blokovima (prema [7])

3.3.2. Memorija

Postoji nekoliko realizacija memorije na SSD-ovima, to su: flash temeljena memorija, DRAM temeljena memorija i nešto manje prisutna 3D-XPoint temeljena memorija koju su razvili *Intel* i *Micron*.

Flash temeljena memorija je upravo ona najčešća. Većina proizvođača koristi NAND flash memoriju za svoje pogone zbog manje cijene u usporedbi s DRAM memorijom i zbog same mogućnosti zadržavanja podataka bez potrebe za konstantnim napajanjem što osigurava očuvanje podataka i prilikom nestanka električne energije. Flash bazirani SSD-ovi su sporiji od DRAM rješenja, a neki su rani dizajni čak i sporiji od HDD-ova nakon učestale upotrebe. Taj su problem riješili kontroleri koji su izašli 2009. i kasnije [1]. Ovakve izvedbe obično dolaze u standardnim diskovnim oblicima (2.5 i 3.5 inča), no i u manjim izdanjima poput M.2 oblika. Jeftiniji pogoni obično koriste višeslojne ćelije (od tri sloja i više), koji su sporiji i manje pouzdani od onih jednoslojnih.

SSD-ovi temeljeni na memoriji koja zahtijeva konstantno napajanje poput onih DRAM (*Dynamic Random Access Memory)* pogona jesu karakteristični po svojim vrlo visokim brzinama pristupa podacima (< 10 ms) i koriste se najprije za ubrzavanje aplikacija koje bi bile spore ako bi se izvršavale na HDD-ovima ili flash baziranim SSD-ovima. Ovakvi SSD-pogoni obično uključuju internu bateriju ili eksterni izvor izmjenične struje zajedno sa rezervnim sustavima za pohranu podataka da bi osigurali trajanje istih u slučajevima nestanka napajanja diska iz vanjskih izvora [1]. Kada nema struje, baterija opskrbljuje snagu potrebnu za kopiranje informacija iz RAM-a u rezervne spremnike. Kada se napajanje ponovno ostvari, podaci se ponovo kopiraju u RAM, i SSD nastavlja normalno raditi.

3.4. Datotečni sustavi kod SSD pogona

3.4.1. Datotečni sustav

Uloga datotečnog sustava je organizacija datoteka i omogućavanje pohrane, dohvaćanje i izmjene podataka na određenom fizičkom uređaju poput tvrdog diska, niza flash memorije ili optičkog diska. Datotečni sustav je od iznimne važnosti za operacijski sustav, te može biti u interakciji ili čak dio nekih protokola (NFS, SMB) koji omogućuju mrežni pristup spomenutim datotečnim sustavima. Mora upravljati kapacitetom za pohranu, smješatajem podataka, direktorijima i imenima datoteka, integritetom podataka i dozvolama pristupa. Uz sve to, karakteristike i mogućnosti datotečnog sustava moraju odgovarati uređaju na kojem se koristi kao i željenoj namjeni aplikacije za koju se koristi [9].

Datotečni sustavi koji se koriste kod tvrdih diskova uglavnom se mogu koristiti i kod SSD pogona. Ipak, iako su slični prema sučeljima, postoje neki problemi. Na niskoj razini funkcioniranja, SSD pogoni zahtijevaju posebno upravljanje poput kontrole istrošenosti i raznih algoritama koji brinu o otkrivanju i uklanjanju pogrešaka. Očekuje se da će datotečni sustav za SSD pogon podržavati TRIM naredbu koja pomaže pogonu reciklirati odbačene podatke. Uglavnom se te operacije odvijaju u samom mediju pa je moguće koristiti uobičajene datotečne sustave. Poželjno je da operacijski sustav na prikladan način organizira particije pogona, čime se izbjegavaju prekomjeni ciklusi čitanja, prepravljanja i pisanja. Uobičajeno je da na osobnim računalima particije započinju sa veličinom od 1 MB, koja je prilagodljiva na sve uobičajene veličine stranica i blokova kod SSD pogona. Noviji operacijski sustavi, to jest njihov instalacijski softver i diskovni alati, time automatski upravljaju [1]. Najćešće korišteni datotečni sustavi na Linux operacijskom sustavu jesu ext4, Btrfs, XFS, JFS i F2FS. Kod macOS-a, od verzije 10.6.8 TRIM podrška vrijedi samo kod Apple SSD pogona, dok se kod ostalih pogona TRIM mora omogućiti na drugačiji način. Kod Microsoft Windowsa, od verzije 7 standardni NTFS datotečni sustav podržava TRIM funkciju koju i sam automatski izvršava ako je prepoznati medij SSD. Neki od datotečnih sustava optimiziranih za flash memoriju i SSD medije jesu: CHFS, exFAT, F2FS, FFS2, HFS+, JFFS(2), LSFS, LogFS, NILFS, OneFS, TFAT, TrueFFS, UBIFS, YAFFS, ZFS i drugi.

3.4.2. exFAT

Extended File Allocation Table skaćeno exFAT je datotečni sustav razvijen od strane Microsofta 2006. godine koji je posebno optimiziran za flash memoriju. On je vlasnički datotečni sustav čije su karakteristike patentirane od strane Microsofta. Ovaj datotečni sustav se koristi gdje NTFS nije zadovoljavajuća opcija, a potreban je veći datotečni limit od standardnog FAT32 datotečnog sistema (4 GiB). Sve inačice Windows operacijskog sustava od Vista SP1 i XP SP2 ga podržavaju. Za razliku od FAT32 datotečnog sustava, veličina klastera doseže kapacitet i do 32 MB, a ograničavanje pristupa se vrši uz listu za kontrolu pristupa [9]. Ovaj datotečni sustav podržava datoteke najveće veličine 16 exabajta ($2^{64} - 1$ bajtova) , skalabilnost za veličine diskova do 128 petabajta, podržava brojku od 2 796 202 datoteka po direktoriju, ima poboljšanu alokaciju slobodnog prostora, veličinu klastera do 32 MB i još mongo drugih karakterisitika. Kako je ovaj datotečni sustav u vlasništvu Microsofta i budući da nisu objavljivali specifikacije za isti, manjak dokumentacije je naškodio razvoj *open source* upravljačkih programa. Stoga, podrška za exFAT je ograničena na Microsoftove proizvode pa je zbog svega navedenoga pokušaj da exFAT postane univerzalni sustav ostao neostvaren [11].

3.4.3. NILFS

NILFS (*New Implementation of Log-structured File System*) je datotečni sustav temeljen na logovima i razvijen za operacijski sustav Linux. NILFS je datotečni sustav kod kojega se medij za pohranu smatra kao kružni buffer, a novi blokovi se uvijek pišu na kraj. On također koristi snimke (*snapshot*) koje nisu ograničene ničime osim kapacitetom prostora. Developeri ističu izradu kontinuiranih snimaka koje se mogu koristiti ukoliko dođe do problema kod datotečnog sustava koji je možda uzrokovan od strane korisnika [12]. NILFS bilježi podatke u kontiniuirani log na koji se oni samo nadodaju, a nikada ne prepisuju preko postojećih. Zahvaljujući takvome pristupu, smanjeno je vrijeme traženja i minimiziran slučaj gubitka podataka koji se javlja kod pogrešaka s konvencionalnim datotečnim sustavima. Također, NILFS može kontinuirano i automatski sačuvati trenutno stanje datotečnog sustava bez ometanja rada istog [13]. Korisnicima omogućava povrat datoteka ako su ih nenamjerno prepisali ili izbrisali u nekom skorijem vremenu. Stvara i brojne sigurnosne točke svakih nekoliko sekundi ili nakon uspješne operacije zapisivanja. NILFS2 realizira i online sakupljanje smeća što je pogodno za povrat prostora na SSD pogonima.

3.4.4. APFS

APFS (*Apple File System*) je datotečni sustav objavljen 2017. godine i razvijen od strane organizacije *Apple Inc* koji se koristi u računalima s operacijskim sustavom Mac OS. Cilj mu je poboljšati i ispraviti glavne propuste njegovog prethodnika zvanog HFS+ (Hierarhical File System). APFS je optimiziran za flash i SSD pogone sa velikim naglaskom na enkripciju. Ovaj datotečni sustav pogodan je za naprave sa relativno malim ili velikim prostorom za pohranu. Koristi 64 bitne *inode* brojeve i omogućuje sigurniju pohranu. APFS, kao i HFS+, koristi TRIM naredbu za bolje performanse i upravljanje prostorom. Također, može ubrzati brzine čitanja i pisanja na iOS-u i MacOS-u kao i prostor na iOS napravama zbog načina na koji računa dostupne podatke [14]. Može kreirati klonove koji omogućuju operacijskom sustavu stvaranje efikasnih kopija na istom svesku (eng. *volume*) bez zauzimanja dodatnog prostora. Promjene u koloniranoj datoteci bilježe se kao delta pa samim time smanjuju potrebnu količinu prostora za pohranu kopija. APFS podržava snimke (*snapshot*) za izradu instanci datotečnog sustava u tom trenutku koje se mogu samo čitati. Podržava i enkripciju cijelog diska (FDE) kao i enkripciju datoteka sa sljedećim opcijama : bez enkripcije, enkripcija s pomoću jednog ključa i enkripcija s pomoću više ključeva gde je svaka datoteka kriptirana odvojenim ključem. Podržava izrazito velik broj datoteka na jednom svesku (preko 9 kvintilijuna). APFS je dizajniran da izbjegne korupciju metapodataka uzrokovanu sustavskim greškama. Umjesto prepisivanja postojećih metapodataka, izrađuje potpuno novi zapis, ukazuje na njega i tada otpušta prijašnje. To sprječava korupciju zapisa koji sadrže djelomično nove i djelomično stare podatke. Također nije više potrebno zapisivati promjene dvaput, kao što je to bio slučaj kod HFS+ datotečnog sustava [14].

3.4.5. NTFS

NTFS (*New Technology File System*) je datotečni sustav razvijen i u vlasništvu organizacije Microsoft. Počevši s Windows NT 3.1 verzijom, to je standardni datotečni sustav za Windows operacijski sustav [15]. Koristi se i na HDD medijima kao i na SSD pogonima. Može pohraniti datoteke veličine do 16 TB, dok su particije najveće veličine 256 TB. Mogućnost pohrane većih datoteka je posebice važno unaprjeđenje u odnosu na prijašnji FAT32 datotečni sustav. Podržava LZNT1 kompresiju i koristi liste s kontrolom pristupa. Koristi MFT (*Master File Table*) koja pohranjuje lokaciju i podatke za pristup informacijama. Ta tablica se povećava s vremenom pa dolazi do fragmentacije. Mala veličina klastera od 4 KB osigurava da manje mjesta ostane neiskorišteno za manje datoteke na disku. Zbog toga je i takva veličina klastera dobrodošla kod SSD pogona, pa NTFS ima dobre performanse i na SSD-ovima [9].

Kao i kod većine datotečnih sustava, glavna jedinica pohrane kod NTFS-a jest datoteka. Datoteka je skup bilo kakvih podataka i može sadržavati bilo što: programe, tekstualne datoteka, zvučne zapise i tako dalje. Operacijski sustav ne prepoznaje razliku između istih. Korištenje određene datoteke ovisi o interpretaciji iste od strane aplikacije koja koristi spomenutu datoteku. Kod NTFS-a, sve datoteke su pohranjene u istom obliku: kao skup atributa. Oni uključuju i sve podatke o samoj datoteci. Kako će datoteka biti pohranjena kod NTFS-a ovisi o veličini same datoteke. Struktura svake datoteke je temeljena na sljedećim informacijama i atributima koji su pohranjeni za svaku datoteku: *header* (H), standardni informacijski atribut (SI), atribut imena datoteke (FN), atribut s podacima i sigurnosni deskriptor (SD). Ovo su sve osnovni atributi i postoje još neki dodatni. Ako je datoteka dovoljno mala da svi njezini atributi mogu stati unutar MFT zapisa za tu datoteku, tada se pohranjuje unutar MFT-a u potpunosti. Ovo ovisi uvelike o veličini MFT zapisa koji se koristi. Ako je datoteka prevelika, NTFS započinje s brojnim ekspanzijama koje miču atribute iz MFT-a. Redoslijed koraka je sljedeći [24]:

1. Najprije će NTFS pokušati spremiti cijelu datoteku u MFT zapis, ako je to moguće.
2. Ako je datoteka prevelika za MFT zapis, on će da taj atribut u MFT-u sadržavati pokazivače na blokove podataka pohranjene u sekevencijalnom redoslijedu izvan MFT-a koji se nazivaju sljedovi podataka (*data runs*).
3. Moguće je da datoteka postane tako velika da nema mjesta u MFT zapisu za sve te pokazivače unutar podatkovnog atributa. Takva datoteka neće imati podatkovni atribut u svojem glavnom MFT zapisu, nego pokazivač koji se nalazi u glavnom MFT zapisu i pokazuje na drugi MFT zapis koji sadrži listu pokazivača podatkovnog atributa.
4. NTFS će nastaviti proširivati ovu strukturu ako se kreiraju vrlo velike datoteke. Moći će kreirati mnogo MFT zapisa, ako je potrebno, da pohrani velik broj potrebnih pokazivača prema sljedovima podataka (*data runs*).

U sljedovima podataka se nalazi većina podataka o datoteci kod NTFS-a. Sadržavaju blokove graničnih klastera na disku. Pokazivači u podatkovnim atributima za tu datoteku sadrže referencu na početak i broj klastera. Početak je prepoznatljiv koristeći virtualni broj klastera (VCN). Korištanje „pokazivač+duljina“ principa znači da kod NTFS-a nije potrebno čitati svaki klaster datoteke da bi se odredilo gdje se nalazi sljedeći [24].

3.5. NAND tehnologija

NAND flash memorija je vrsta memorije koja ne zahtijeva konstantno napajanje da bi zadržala podatke. Koristi se u MP3 playerima, digitalnim kamerama, USB pogonima i naravno SSD pogonima. Važna je po tome što je smanjila cijenu po bitu, a povećala kapacitete čipova što joj je omogućilo da konkurira sa tvrdim diskovima. NAND memorija sprema podatke u obliku blokova i oslanja se na strujne krugove da bi pohranila te podatke. Također, ima ograničen broj ciklusa pisanja. To znači da ćelije polako odumiru s vremenom, što pogoršava performanse,a taj koncept je poznat pod imenom trošenja (*wear-out)*. Kako bi se to ublažilo, proizvođači često znaju opskrbiti medij s više memorije nego što je to naznačeno [16]. Neka istraživanja su pokazala da primjenom vrlo visoke temperature na NAND čipove, elektrone koji su ostali „zaglavljeni“ mogu na taj način osloboditi. To u teoriji značajno poboljšava dugovječnost SSD pogona, no ovo još nije u potpunosti potvrđeno kao praksa kod uobičajenih korisnika [17]. Postoji nekoliko vrsta NAND flash memorije. Ono što ih razlikuje je broj bitova koje koristi pojedina ćelija. Što je više bitova pohranjeno u svakoj ćeliji, to je NAND flash jeftiniji za izvedbu. Ovo su podtipovi NAND flash memorije [16] :

- **SLC** (*Single level cells)* – memorija kod koje se jedan bit pohranjuje u jednu ćeliju. Zahvaljujući tome, ima najveću izdržljivost, ali je i najskuplja vrsta memorije.
- **MLC** (*Multi level cells*) – memorija kod koje se dva bita pohranjuju u jednu ćeliju. Budući da se brisanja i pisanja odvijaju dvostruko češće nego kod SLC vrste, ova je vrsta manje izdržljiva, ali je i jeftinija za proizvodnju pa se i često koristi.
- **TLC** (*Triple level cells)* – memorija koja koristi tri bita na jednoj ćeliji. Mnogo proizvoda namijenjenih potrošačkom sloju korisnika ih koristi budući da je jeftina za proizvodnju, ali naravno donosi niže performanse od SLC i MLC memorija.
- **QLC** (*Quad level cells)* – memorija koja koristi četiri bita na jednoj ćeliji. Ova memorija je najjeftinija za proizvodnju, pa zato i ima najlošije performanse.
- **3D NAND** – ova podvrsta memorije stavlja ćelije jednu na drugu. Zapravo to znači da su organizirane vertikalno u nekoliko slojeva, a najćešće koriste MLC ili TLC NAND flash [18].

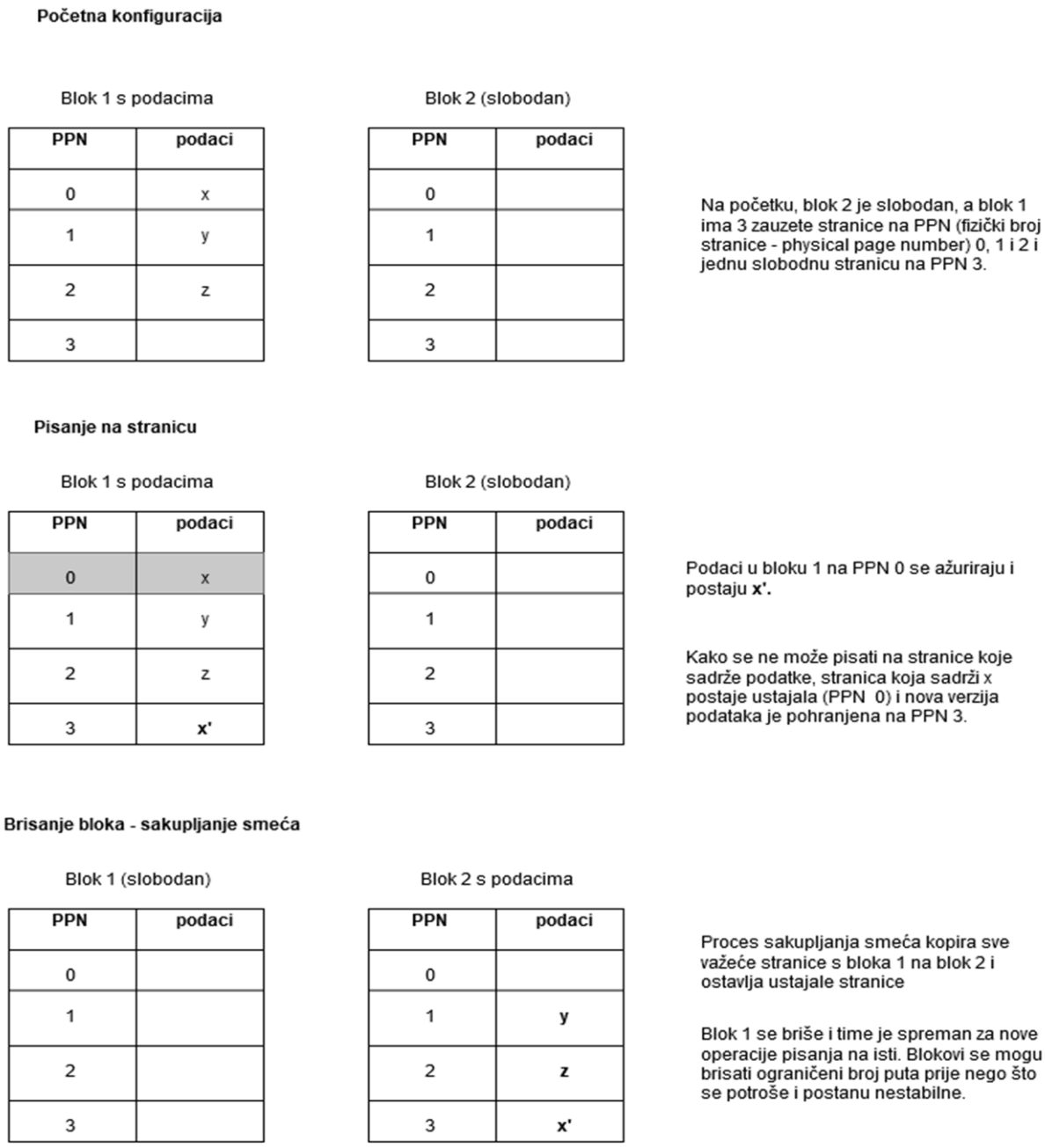
U tablici 2, dana je usporedba svih NAND flash podvrsta memorije kao i performanse/latencije HDD-ova, glavne RAM memorije i pričuvne L1 i L2 memorije.

Tablica 2. Usporedba svojstava NAND flash memorija i drugih vrsta memorija (prema [17])

	SLC	**MLC**	**TLC**	**HDD**	**RAM**	**L1**	**L2**
P/E ciklusa	100 000	10 000	5 000	-	-	-	-
Bitova po ćeliji	1	2	3	-	-	-	-
Latentnost traženja (ms)	-	-	-	9000	-	-	-
Latentnost čitanja (ms)	25	50	100	2000 – 7000	0.04 – 0.1	0.001	0.004
Latentnost pisanja (ms)	250	900	1500	2000 – 7000	0.04 – 0.1	0.001	0.004
Latentnost brisanja (ms)	1500	3000	5000	-	-	-	-

U posljednjem desetljeću, SSD pogoni temeljeni na NAND flash realizaciji su dobili široku primjenu na potrošačkoj razini, ali i kod sustava sa potrebom za visokim performansama s ciljem pružanja visokih performansa i visoke razine pouzdanosti. Takva memorija nudi visoke performanse kod čitanja i pisanja u usporedbi s tradicionalnim HDD-ovima. Ipak, postoji nekoliko prepreka koje treba prebroditi da bi se ova memorija mogla adekvetno koristiti kao rješenje za pohranu. NAND flash čip je sastavljen od dva ili više kalupa, a svaki od njih ima nekoliko ravnina. Svaka ravnina se pak sastoji od tisuća blokova i nekoliko registara koji se koriste kao međuspremnici. Blok uglavnom sadrži 64, 128 ili 256 stranica, a veličina stranice je 2, 4, 8 ili 16 KB. Zbog jedinstvene organizacije NAND flash ćelija, nije moguće čitati ili pisati s/po samo jednoj ćeliji. Memorija je grupirana i pristupa joj se sa specifičnim svojstvima [19]. Poznavanje ovih svojstava je iznimno važno za optimizaciju strukture podataka kod SSD pogona.

Za početak, nije moguće čitati manje od jedne stranice odjednom. Moguće je zatražiti samo jedan bajt od operacijskog sustava, ali će on vratiti cijelu stranicu SSD-u, čitajući mnogo više podataka nego je to bilo potrebno. Nadalje, prilikom pisanja na SSD, zapisivanja se odvijaju u inkrementima veličine stranice. Pa ako sama operacija pisanja utječe na samo jedan bajt, cijela će stranica mora biti zapisana. Zapisivanje više podataka nego je to potrebno je poznato pod nazivom pojačanog pisanja (*write ampliification*) što uzrokuje neefikasnost. Na primjer, zapisivanje jednog bajta na stranicu će rezultirati zapisivanjem cijele stranice, koja može biti veličine i do 16 KB što ostavlja mnogo neiskorištenog prostora. Uz nepotrebno zauzimanje prostora, također dolazi do više unutarnjih operacija nego što je potrebno. Na stranicu u NAND flashu moguće je pisati jedino ako je ona u slobodnom stanju. Kada se mijenjaju podaci, sadržaj stranice se kopira u unutarnji registar, podaci se ažuriraju i nova verzija je pohranjena na slobodnu stranicu, što se zove operacija čitanja-izmjenjivanja-pisanja (*read-modifly-write*). Podaci se ne ažuriraju na licu mjesta, budući da je slobodna stranica različita od one stranice na kojoj su prvotno bili podaci. Kada su podaci trajno na pogonu, prvotna stranica je označena kao ustajala i takva će ostati dok god ne bude izbrisana. Nije moguće brisati pojedine stranice, nego je moguće brisati cijele blokove odjednom. Operacija brisanja je automatski prepoznata od stane procesa sakupljanja smeća u kontroleru SSD-a kada je potrebno ustajale stranice povratiti u prvotno stanje da bi se došlo do mjesta za pohranu [19]. Kao ilustracija ovih ograničenja, slijedi prikaz s operacijom zapisivanja na jednostavan SSD pogon.



Slika 4. Zapisivanje podataka na SSD pogon (prema [19])

Dakle, operacije pisanja ili P/E (*program/erase)* operacija na flash memoriji jedino može promijeniti vrijednost ćelije iz 1 u 0. Operacija brisanja je potrebna da bi se ćelije označene s 0 prebacile na vrijednost 1, što će ju pripremiti za novu operaciju pisanja ili P/E operaciju. Operacije čitanja i pisanje mogu funkcionirati na stranici u NAND flash memoriji, dok operacija brisanja može raditi samo na razini bloka. Ova razlika predstavlja prepreku u dizajniranju kontrolera kod SSD pogona. Još jedna prepreka kod flash memorije je ciklus brisanja. Ćelije flash memorije polako odumiru s vremenom i korištenjem, pa SSD pogoni temeljeni na NAND flash memoriji moraju koristiti poseban firmware zvan flash translacijski sloj (*Flash Translation Layer* - FTL) koji će upravljati operacijama [20].

3.6. Flash Translation Layer (FTL)

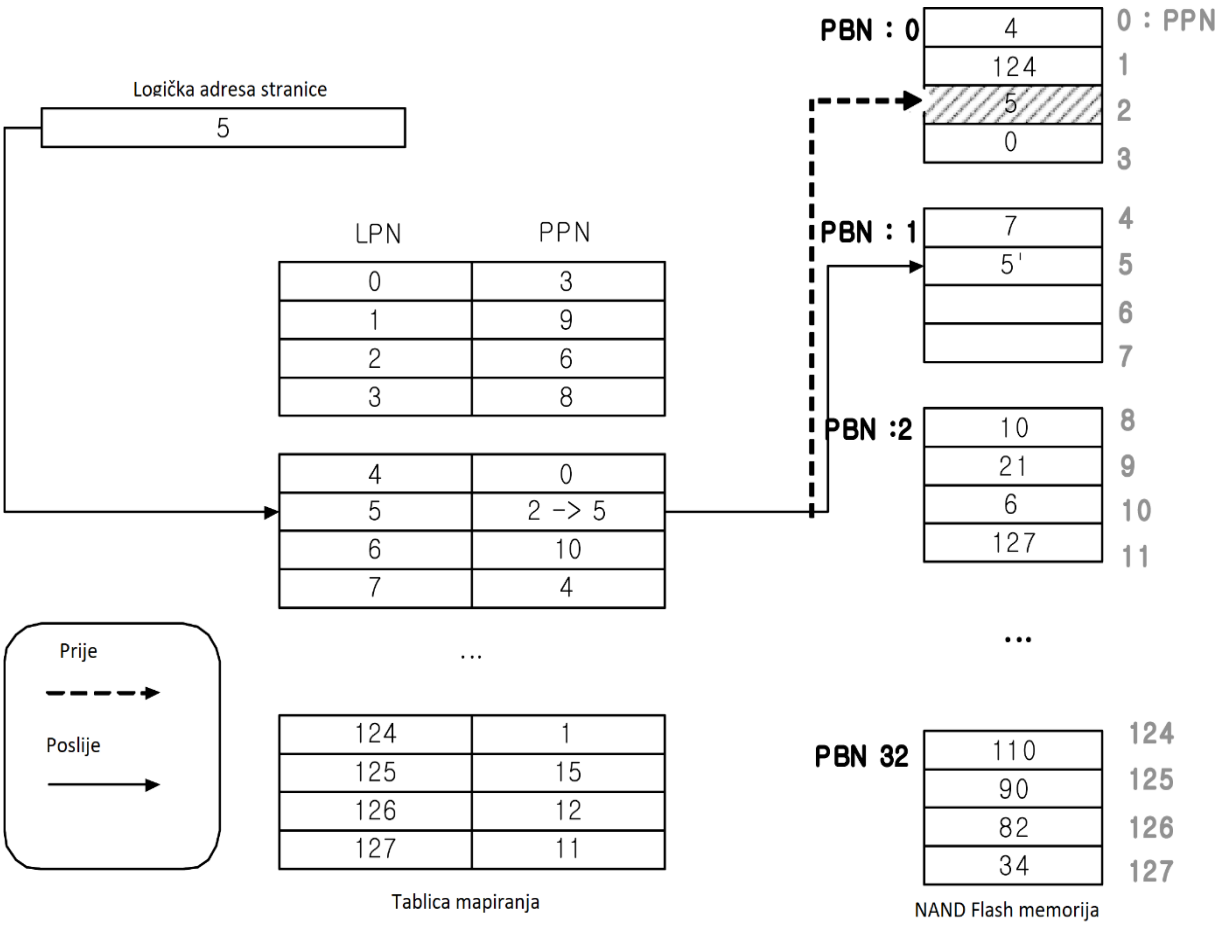
Glavni faktor koji je olakšao korištenje SSD pogona je isto sučelje koje se koristi i kod HDD pogona. Iako kod HDD-ova prikazivanje liste logičkih adresa blokova (LBA) ima smisla, budući da se na HDD-u može pisati preko sektora, to kod flash memorije nije tako. Iz tog razloga, dodatna komponenta koja je potrebna da bi se sakrile unutarnje karakteristike NAND flash memorije, a prikazale samo liste logičkih blokova adresa sučelju. Upravo se ta komponenta zove flash translacijski sloj (FTL), a nalazi se u kontroleru SSD pogona [19]. FTL je iznimno važan jer implementira glavne algoritme potrebne za upravljanjem resursima unutar SSD-a.

3.6.1. Mapiranje podataka

FTL mora implementirati mehanizam koji omogućava adresno preusmjeravanje s logičke adrese stranice na onu fizičku adresu u SSD-u. FTL sadrži tablicu kojom prati ažuriranja logičkih stranica. Postoji nekoliko strategija kojima se ovo pokušava postići [20].

Pure page mapping (čisto mapiranje stranica) koje sadrži tablicu koja mapira svaku logičku stranicu fizičkoj stranici gdje su pohranjeni podaci. Mapiranje stranica je učinkovito, ali nije skalabilno. Naime, kako veličina SSD pogona raste tako raste i ukupan broj fizičkih stranica pa će se i veličina tablice također povećati. Neka su istraživanja predložila pristup koji ovaj način čini primjenjivim tako što se tablica sprema u flash čipove, a potreban dio se predaje kontroleru kada on to zahtijeva [20]. Iako je ova strategija prilično prilagodljiva, velik nedostatak iste je taj da tablica za mapiranje zahtijeva mnogo RAM-a što značajno može utjecati na cijenu proizvodnje.

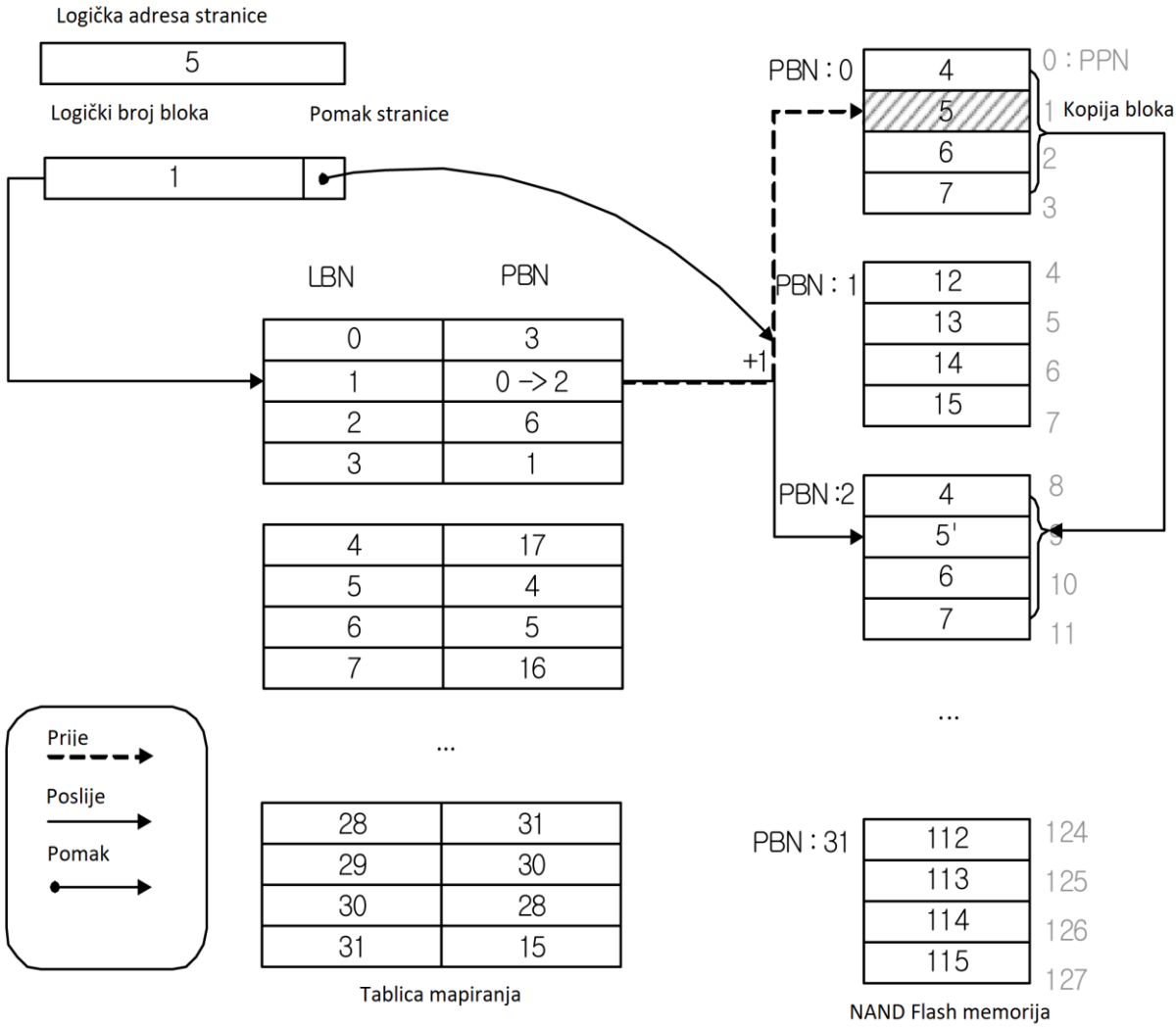
Kao primjer, na slici dolje, kada se javlja zahtjev za pisanje prema logičkoj adresi stranice 5, FTL će najprije tražiti odgovarajući fizički broj stranice koristeći logički broj stranice čiji su indeksi identični u tablici mapiranja. Kao rezultat podudarajućih adresa, odgovarajući fizički broj stranice je prepoznat pod brojem 2 u NAND flash memoriji. Ipak, na njemu se trenutno nalaze važeći podaci pa se potrebni podaci za zapisivanje moraju preusmjeriti na slobodnu stranicu u flash memoriji. Kako je druga stranica bloka s fizičkim brojem 1 slobodna, podaci će biti zapisani na tu lokaciju. U isto vrijeme, tablica mapiranja će biti ažurirana da pokazuje na novu stranicu s važećim podacima [22].



Slika 5. Mapiranje stranica (prema [22])

Block mapping (mapiranje blokova) pokušava smanjiti čim je više moguće veličinu tablice za mapiranje tako što mapira logičke u fizičke blokove adresa. Pomak kod stranica u logičkim i fizičkim blokovima mora biti identičan. Na svakoj stranici koja se ažurira van mjesta, blok mora biti zamijenjen drugim slobodnim blokom i sve nepromijenjene stranice unutar bloka, zajedno sa promijenjenom stranicom, moraju se prebaciti na slobodan blok. Ovo uzrokuje velik pad performansi što nije prihvatljivo kako je brzina pisanja već mnogo manja nego brzina čitanja zbog dugog vremena izvršavanja operacija u flash ćelijama. Na primjer, pretpostavimo da SSD ima 256 stranica po bloku. To znači da mapiranje blokova zahtijeva 256 puta manje memorije nego mapiranje stranica, što je rezultira mnogo boljim upravljanjem kapacitetom. Ipak, ono zahtijeva zapisivanje na disk zbog dodatne sigurnosti što uzrokuje pojačano zapisivanje i čini ovu strategiju prilično neefikasnom [19].

Na sljedećoj slici je prikazan primjer kada dođe do zahtjeva za pisanje na logičku adresu stranice 5. Stranica s logičkom adresom je podijeljena na logički blok 1 i na pomak stranice 1. Najprije se treba odrediti fizički broj bloka za odgovarajući logički broj bloka 1. Nakon što je odgovarajući fizički broj bloka 0 našao para, logički pomak stranice je dodan na određeni fizićki broj bloka, a svi nadolazeći podaci su zapisani na fizički broj stranice 1. Kako fizički broj stranice 1 već sadrži podatke, oni će morati biti zapisani na slobodan blok (ovdje na fizički broj bloka 2). Isto tako, druge stranice u istom bloku s fizičkim brojem stranice 1 će biti locirane i kopirane na isti slobodan blok budući da je samo jedan logički blok povezan sa samo jednim fizičkim blokom [22].

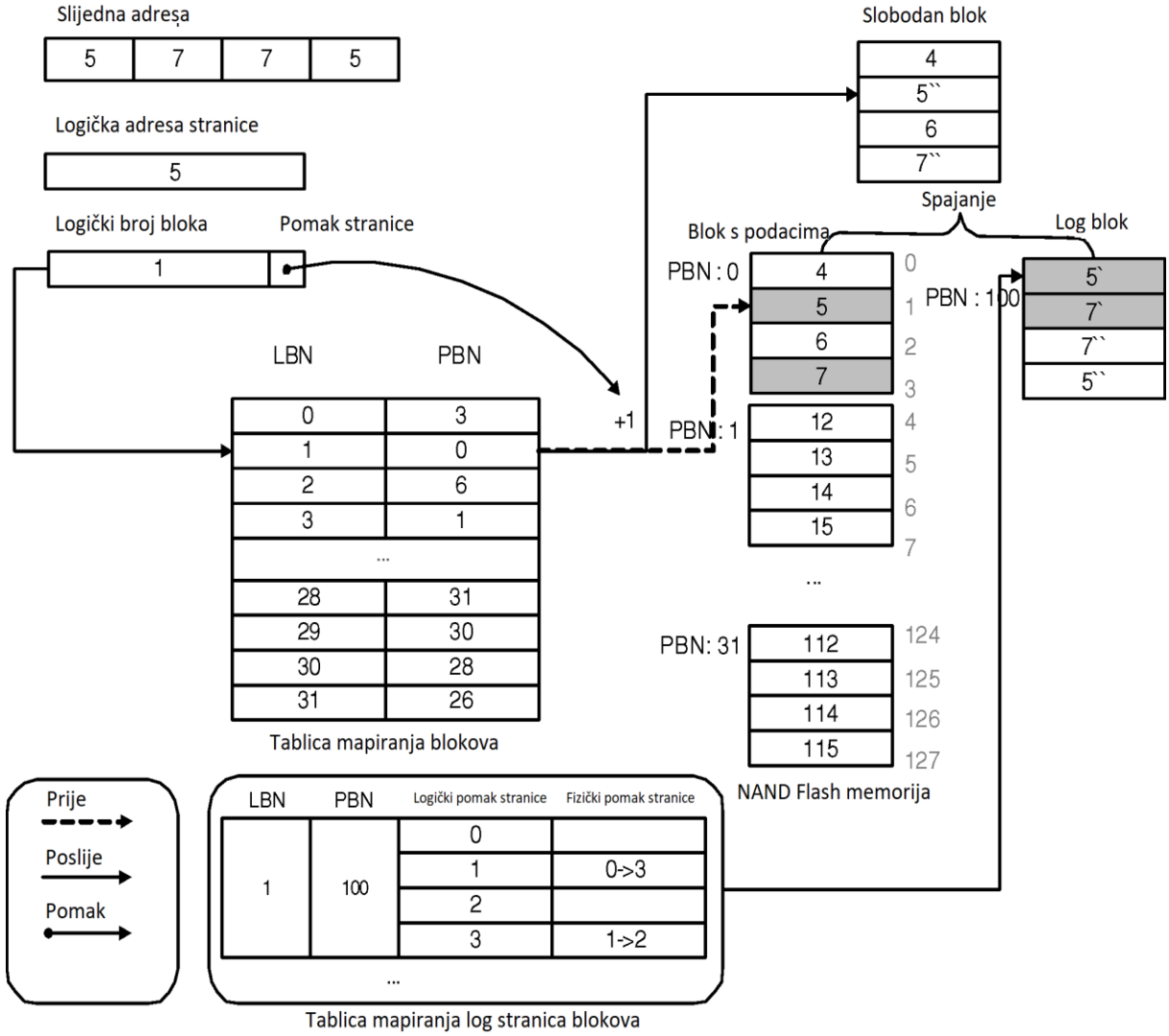


Slika 6. Mapiranje blokova (prema [22])

Hybrid mapping (hibridno mapiranje) je kombinacija čistog mapiranja stranica i mapiranja blokova koja ima cilj smanjiti veličinu tablice i poboljšati performanse ažuriranja. U ovom pristupu, fizički blokovi su podijeljeni u dvije grupe: blokovi podataka i log blokovi. Svaki blok podataka je mapiran logičkom bloku kojeg će ažuriranje stranice spremiti u log blok, a stara stranica će postati nevažeća. Kada je broj slobodnih blokova na najmanjem broju, izvršit će se operacija spajanja koja je odgovorna za vraćanje nevažećih stranica da bi oslobodila prostor. Operacija spajanja označava log blok i ponovno kreira blok s podacima iz važećih stranica unutar log bloka. Spajanje može biti obavljeno na nekoliko načina, što pak ovisi o sadržaju označenog log bloka [20]:

- *Switch merge* (zamjensko spajanje) – ovo spajanje se koristi kada blok sadrži stranice podataka u slijednom redoslijedu. Spajanje ovdje samo mijenja blok podataka sa log blokom i ažurira tablicu mapiranja. Naposlijetku, stari blok s podacima koji sadrži sve nevažeće stranice će bit izbrisan.
- *Reorder merge* (preuređeno spajanje) – ovo se spajanje koristi kada blok sadrži stranice jednog bloka s podacima, ne u slijednom redoslijedu. U ovom slučaju, sve stranice će biti kopirane u drugi slobodan blok, ali u točnom redoslijedu, dok će i blok podataka i log blok biti izbrisani. Novi blok će zamijeniti stari blok s podacima i tablica će biti ažurirana.
- *Partial merge* (djelomično spajanje) – javlja se kada blok sadrži neke stranice u bloku podataka koji su u odgovarajućem redoslijedu, a ostatak bloka je slobodan. U ovom slučaju, ostatak stranica će biti kopiran iz drugih lokacija u log blok i taj log blok će zamijeniti stari blok s podacima. Naravno, stari blok će biti izbrisan, a tablica ažurirana.
- *Full merge* (puno spajanje) – puno spajanje je najzahtjevnije spajanje od svih. Kod njega je najprije označen log blok koji sadrži važeće stranice iz više nego jednog bloka s podacima. U ovom slučaju, svi blokovi s podacima imaju barem jednu važeću stranicu unutar označenog log bloka koja mora biti ponovno kreirana. Puno spajanje ponovno sastavlja sve stranice iz ovog seta s blokovima podataka i ponovno iz izrađuje i zamjenjuje sa starim blokovima podataka. Naposlijetku, svi blokovi podataka i log blokovi označeni u ovom setu će biti izbrisani.

Na slici dolje se nalazi primjer kod kojega se pretpostavlja da fizički broj bloka 0 sadrži podatke logičkih brojeva stranica 4, 5, 6 i 7 koji su zapisani na alocirani log blok, dok dolazne operacije pisanja zahtijevanju logičke brojeve stranica 5, 7, 7 i 5. Zadnje dvije operacije pisanja pišu preko prve dvije. Kao rezultat toga, samo je zadnji zahtjev prema logičkom broju stranica (7, 5) važeći za logički broj bloka 1. Ti zahtjevi su označeni kao 5'' i 7'' na slici. Kada log blok nema dodatnih slobodnih stranica ili kada je logički blok koji sadrži tražene stranice promijenjen od strane prethodnog logičkog bloka, tada se log blok i odgovarajući blok s podacima spajaju u slobodan blok. Naposlijetku, spojeni slobodan blok postaje novi podatkovni blok, a prvobitni podatkovni blok i log blok postaju dva slobodna bloka [22].



Slika 7. Hibridno mapiranje (prema [22])

3.6.2. Dodjeljivanje stranica

Funkcionalnost dodjeljivanja stranica (*page allocation)* upravlja ponašanjem SSD-a tijekom raznih ulazno – izlaznih pristupa i utječe na šanse za ostvarivanje paralelnog pristpa na način da specificira metodu koja će riješiti konflikte kod resurasa SSD-a. Ova funkcionalnost također određuje odredišni kanal, smjer, čip i ravninu transakcije veličine stranice i pri tome slijedi specifičnu listu prioriteta razina paralelizma. Ova dodjela specificira fizičku lokaciju u kojoj se nalazi određeni PPN (*Physical Page Number* – broj fizičke stranice). Lokacija uključuje ID kanala (CID), ID paketa (WID), ID čipa (DID) i ID ravnine (PID), a unutar ravnine, određuje broj bloka i pomak stranice kojoj se treba pristupiti. Ako ovo svojstvo adekvatno radi s logičkim adresama stranica pomoću resurasa na SSD-u (kanali, čipovi...) onda kontroler može istovremeno izvršavati transakcije da bi se postiglo manje vrijeme potrebno za odaziv [20].

Različiti mehanizmi dodjeljivanja stranica koriste različite naredbe da bi odvojili zahtjeve u raznim nivoima paralelizma. CWDP (*Channel, Way, Die, Plane*) koristi ostatak PPN-a za druge kanale da bi specificirao odredišni broj kanala (CID). Kvocijent je zatim primjenjen za izračun WID-a. Ovaj se proces ponavlja za DID i PID u sljedećim koracima. Nakon njih, kvocijent će se koristiti za pronalaženje bloka unutar ravnine i za pomak stranice toga bloka. Slično, izračuni kod DPWC (*Die,Plane, Way, Channel)* mehanizma odvijaju se redoslijedom DID, PID, WID i CID. Metoda računanja ID-jeva u statičnim pristupima čini njihove performanse jako ovisnima o uzorcima adresnog pristupa i mapiranju podataka u FTL-u [20].

3.6.3. Sakupljanje smeća

Kao što je bilo rečeno, pisanje preko stranica koje sadrže podatke nije moguće. Ako podaci na stranici trebaju biti ažurirani, nova inačica je zapisana na slobodnu stranicu, a stranica koja je sadržavala prijašnju inačicu podataka je označena kao ustajala. Kada blokovi sadrže ustajale stranice, moraju se obrisati prije nego što se može na njih pisati. Sakupljanje smeća (*garbage collection*) je proces kojeg obavlja SSD kontroler i koji osigurava da se ustajale stranice obrišu i vrate u slobodno stanje da bi se nadolazeće operacije pisanja mogle uspješno izvršiti [19].

Za početak tu je upravljanje istrošenošću (*wear leveling*). Zbog nepredvidivog izvršavanja operacija pisanja, učestalost ažuriranja fizičkih stranica nije identično pa to dramatično utječe na broj brisanja različitih blokova, što rezultira brzom potrošnjom nekih blokova, dok su drugi blokovi daleko od svoje granice izdržljivosti. Da bi se dugovječnost SSD-a poboljšala, FTL koristi skup mehanizama koji će uravnotežiti istrošenost, poput mapiranja stranica s obzirom na njihovu izdržljivost ili izmjenjivanja podataka između često korištenih blokova sa onim manje izkorištenim blokovima [20].

Cjelokupan kapacitet SSD pogona je uobičajeno viši nego onaj vidljivi korisniku. Nevidljivi kapacitet, koji iznosi oko 7% do 25% ukupnog kapaciteta, se naziva prekomjernom opskrbom (*over-provisioning space*), a njime upravlja FTL. Dva su glavna razloga zbog kojih se ovaj princip prekomjerne opskrbe koristi. Najprije, FTL koristi ovaj kapacitet za upravljanjem lošim blokovima. Trajno iznačava blokove koji su dosegli svoj kapacitet izdržljivosti kao loše blokove i mijenja ih sa upotrebljivim blokovima iz kapaciteta prekomjene opskrbe. Nadalje, FTL koristi ovaj kapacitet kada je prisutan velik broj operacija pisanja. Da bi pružio prihvatljive performanse, posebice kod aplikacija sa malim nasumičnim pisanjem, kontroler upravlja operacijama pisanja u strukturi loga (*log structure*). Ova log struktura se može pohraniti u kapacitetu namijenjenom za prekomjernu opskrbu. Kada je dostupan kapacitet prekomjerne opskrbe došao do predefinirane granice, FTL mora osloboditi više blokova da bi imao dovoljan broj slobodnih blokova [21]. Ovaj se proces zove sakupljanje smeća *(garbage collection*), koji će biti pobliže pojašnjen.

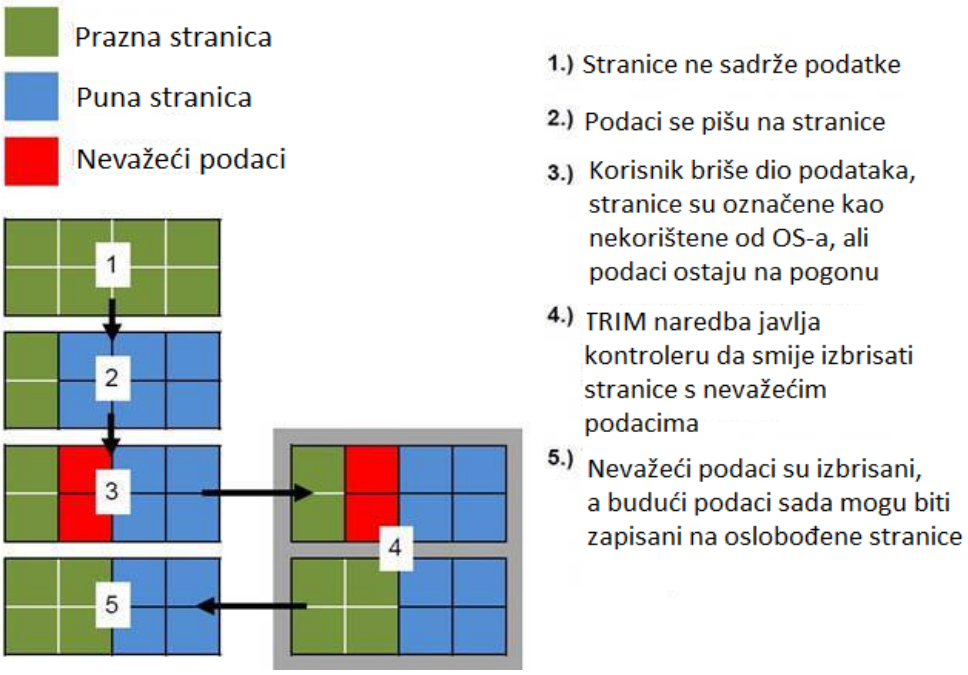
Svako izvršavanje sakupljanja smeća ima tri koraka. Naprije, funkcija *GCselect* odabire jedan ili više blokova među kandidatima. Tada se važeće stranice koje su označene među blokovima kandidatima premještaju u čiste blokove i njihovi pripadni zapisi u tablici mapiranja su ažurirani. Naposlijetku, označeni blokovi kandidati su izbrisani i dodani među slobodne blokove. Jasno je da je ovaj proces iznimno ključan kada je riječ o performansama SSD pogona. U idealnom slučaju, sakupljanje smeća mora pružiti mnogo slobodnih stranica, a pritom čim manju „cijenu“. Najveći utjecaj na performanse u sakupljanju smeća ima premještaj važećih stranica između blokova. Taj utjecaj se može poboljšati tako da se provodi označavanje blokova kandidata koji imaju najveći broj nevažećih stranica. Na ovaj način, kopiranje važećih stranica povlači minimalne dodatne operacije pisanja dok je više slobodnih stranica dostupno na raspolaganju. Ipak, funkcija *GCselect* mora imati obzira na dugovječnost blokova i treba stremiti ujednačenoj distribuciji brisanja preko svih ćelija i blokova.

Vrijeme potrebno za pronalazak najboljih blokova kandidata trebalo bi biti kratko. Mnogi *GCselect* algoritmi su već predložili neke metrike za poboljšanje istog. Ovo su neki od njih [20]:

- *Greedy (*škrti) – ovaj algoritam označava blok sa najmanjim brojem važećih stranica. Također, očekuje se da će smanjiti utjecaj na performase prilikom premještaja. Neki nedostaci uključuju lošu raspodjelu trošenja i visoku latenciju.
- RGA – ovaj algoritam označava nasumičan skup *d* blokova i označava onog s najmanjim brojem važećih stranica među njima. Ubrzava vrijeme premještanja, ali pogoršava raspodjelu trošenja.
- *Random* (nasumično) – algoritam označava blok nasumično bez nekih posebnih kriterija odabira prema stranicama (važeće ili nevažeće). Izvedba RANDOM+, ponavlja ovu funkciju dok ne dođe do bloka s barem jednom nevažećom stranicom.

3.6.4. Ostale funkcionalnosti

Pretpostavimo da program zapisuje datoteke na sve logičke adrese blokova SSD-a. Takav bi se SSD smatrao punim. Sada pretpostavimo da se sve te datoteke izbrišu. Datotečni sustav bi javio da je dostupno 100% slobodnog mjesta, iako bi pogon bio pun, budući da kontroler ne može znati kada su logički podaci obrisani od korisnika. Kontroler će vidjeti slobodan prostor samo kada se preko logičkih adresa blokova, na kojima su bili podaci, ponovno nešto zapiše. U tome trenutku, proces sakupljanja smeća će izbrisati blokove povezane s izbrisanim podacima, oslobađajući tako stranice za nove operacije pisanja. Kao posljedica, umjesto brisanja blokova odmah kada je poznato da sadrže ustajale podatke, brisanje će biti odgođeno što značajno degradira performanse [21]. Još je prisutan jedan problem: budući da stranice koje sadržavaju obrisane datoteke nisu poznate kontroleru, proces sakupljanja smeća će ih nastaviti micati uokolo da bi postigao ravnomjeno trošenje što povećava zagušenost prema sučelju iz nepotrebnog razloga. Rješenje problema odgođenog brisanja je TRIM naredba, koja može biti poslana od operacijskog sustava kontroleru da ga obavijesti o stranicama koje se više ne koriste u logičkom prostoru. S takvom informacijom, proces sakupljanja smeća zna da više nije potrebno micati stranice uokolo nego ih slobodno može obrisati kada god bi to bilo potrebno. TRIM naredba će raditi samo ako ju podržavaju SSD kontroler, operacijski sustav i datotečni sustav. Većina današnjih SSD pogona podržava TRIM naredbu, što će značajno poboljšati buduće performanse budući da se proces sakupljanja smeća može početi odvijati čim prije je to moguće [21].



Slika 8. TRIM primjer (prema [23])

Neki SSD kontroleri pružaju funkcionalnost sigurnog brisanja (*ATA Secure Erase*), čiji je cilj vratiti performanse pogona u početno („kao novo“) stanje. Ova naredba briše sve podatke koje je korisnik pisao i resetira FTL tablicu mapiranja, no naravno ne može imati utjecaj na fizička ograničenja kod potrošnje ćelija. Iako ova funkcionalnost zvuči vrlo obećavajuće, proizvođač je odgovoran za njezinu adekvatnu implementaciju [21].

Nativan red izvršavanja naredbi (*Native Command Queueing*) je svojstvo SATA-e koje omogućuje SSD pogonu prihvaćanje mnogo naredbi od strane korisnika da bi ih izvršio istovremeno koristeći unutarnji paralelizam. Uz smanjivanje latencije zbog pogona,neki pogoni također koriste NCQ da se bolje nose sa latencijom sučelja. Na primjer, NCQ može dati prvenstvo dolazećim naredbama da bi osigurao da pogon uvijek ima naredbe za obrađivanje dok je CPU sučelja zauzet [21].

Iznenadan nestanak električne energije je pojava koja se ne može izbjeći. Tako neki proizvođači uključuju *super*-kondenzator u njihovoj izvedbi SSD pogona koji bi trebao sadržavati dovoljno energije da bi se izvršili ulazno-izlazni zahtjevi u sabirnici (ukoliko dođe do nestanka energije) i da pogon ostane u konzistentnom stanju. Problem je da taj kondenzator ne ugrađuju svi proizvođači (ili neki drugi oblik zaštite od nestanka električne energije), a oni koji ih ugrađuju ne nepominju nužno to u specifikacijama. Tada, kao i kod *Secure Erase* naredbe, nije jasno je li ili nije mehanizam pravilno implementiran i hoće li on pružiti zaštitu pogonu od gubitka podataka kada dođe do nestanka električne engergije [21].

4. Zaključak

SSD pogoni su kompleksni i napredni sustavi za pohranu podataka. Njihova implementacija na niskoj razini zahtijeva mnogo brige za kvalitetno funkcioniranje istog budući da je realiziran s pomoću specifične flash tehnologije. Potrebno je brinuti o istrošenosti, dugovječnosti i općenitim performansama SSD-a. Uloga firmwarea je ovdje od ključne važnosti bez koje SSD ne bi funkcionirao ni približno efikasno kako funkcionira s njime. Samim time i datotečne sustave prilagođene za SSD-ove nije lako implementirati, ali se zahvaljujući mnogo boljim performansama od prošlih generacija uređaja za pohranu podataka to svakako isplatilo pa se i realiziralo. Danas je ova tehnologija prilično razvijena i proizvođači SSD-ova brinu o većini navedenih ograničenja i zahtjeva, pa je sigurno reći da je SSD pogon suvremeno rješenje za problem pohrane podataka.

Popis literature

[1] *Solid-state drive* , 2019. [Na internetu]. Dostupno na: https://en.wikipedia.org/wiki/Solid-state_drive [Pristupljeno 29.6.2019.]

[2] M. Rouse, *SSD (solid-state drive)* , 2019. [Na internetu] . Dostupno na : https://searchstorage.techtarget.com/definition/SSD-solid-state-drive [Pristupljeno 29.6.2019.]

[3] *Flash memory* , 2019. [Na internetu]. Dostupno na : https://en.wikipedia.org/wiki/Flash_memory [Pristupljeno 29.6.2019.]

[4] M. Rouse i dr. , *Flash memory* , 2018. [Na internetu] . Dostupno na : https://searchstorage.techtarget.com/definition/flash-memory [Pristupljeno 29.6.2019.]

[5] Z. Painter, *5 SSD Interface Types and How They Affect Your Computer's Performance* 2018. [Na internetu] Dostupno na : https://blog.silicon-power.com/index.php/guides/ssd-interface-types/ [Pristupljeno 29.6.2019.]

[6] S. Taylor, Ćelija flash memorije [Slika] , preuzeto 29.6.2019. sa: https://www.researchgate.net/profile/S_Taylor3/publication/276039677/figure/fig6/AS:319924485541893@1453287349737/Schematic-representation-of-flash-memory-cell-structure-using-the-Y-2-O-3-as-a.png

[7] Micheloni R. i Crippa L. , *Solid-state drives (SSDs) ,* 2017. [Na internetu] , Dostupno na:https://www.springer.com/cda/content/document/cda_downloaddocument/9783319517346-c1.pdf?SGWID=0-0-45-1604043-p180537768 [Pristupljeno 6.8.2019.]

[8] *Solid state architecture* [Slika], preuzeto 6.8.2019. sa: https://www.datalight.com/assets/images/Tabbed%20Pages/Images/Ind%20SSD%20Solid-State%20Architecture.png

[9] P. Schmid , *SSD file system* , 2012. [Na internetu] . Dostupno na: https://www.tomshardware.com/reviews/ssd-file-system-ntfs,3166.html [Pristupljeno 14. 8. 2019.]

[10] *List of file systems*, 2019. [Na internetu]. Dostupno na: https://en.wikipedia.org/wiki/List_of_file_systems#File_systems_optimized_for_flash_memory,_solid_state_media [Pristupljeno: 14.8.2019.]

[11] *exFAT* , 2019. [Na internetu] Dostupno na: https://en.wikipedia.org/wiki/ExFAT [Pristupljeno: 14.8.2019.]

[12] *Btrfs and NILFS ,* 2007. [Na internetu] Dostupno na: https://lwn.net/Articles/238923/ [Pristupljeno 14.8.2019.]

[13] *NILFS*, 2019. [Na internetu] Dostupno na: https://en.wikipedia.org/wiki/NILFS [Pristupljeno: 14.8.2019.]

[14] *Apple File System* , 2019. [Na internetu] Dostupno na: https://en.wikipedia.org/wiki/Apple_File_System [Pristupljeno 14.8.2019.]

[15] *NTFS ,* 2019. [Na internetu] Dostupno na: https://en.wikipedia.org/wiki/NTFS [Pristupljeno 14.8.2019.]

[16] *NAND Flash memory ,* 2019. [Na internetu] Dostupno na: https://searchstorage.techtarget.com/definition/NAND-flash-memory [Pristupljeno 20.8.2019.]

[17] E. Goossaert, *Coding for SSDs – Part 2,* 2014. [Na internetu] Dostupno na: http://codecapsule.com/2014/02/12/coding-for-ssds-part-2-architecture-of-an-ssd-and-benchmarking/ [Pristupljeno 20.8.2019.]

[18] M. Rouse , *3D NAND Flash,* 2016. [Na internetu] Dostupno na: https://searchstorage.techtarget.com/definition/3D-NAND-flash [Pistupljeno: 20.8.2019.]

[19] E. Goossaert, *Coding for SSDs – Part 3 ,* 2014. [Na internetu] Dostupno na: http://codecapsule.com/2014/02/12/coding-for-ssds-part-3-pages-blocks-and-the-flash-translation-layer/ [Pristupljeno: 20.8.2019.]

[20] N. Shahidi, *Flash Translation Layer Design In Solid State Drives ,* 2017. [Na internetu] Dostupno na: https://etda.libraries.psu.edu/files/final_submissions/14344 [Pristupljeno: 20.8.2019.]

[21] E. Goossaert, *Coding for SSDs – Part 4,* 2014. [Na internetu] Dostupno na: http://codecapsule.com/2014/02/12/coding-for-ssds-part-4-advanced-functionalities-and-internal-parallelism/ [Pristupljeno 22.8.2019.]

[22] C. Park i dr., *A Reconfigurable FTL (Flash TranslationLayer) Architecture for NAND Flash-Based Applications* , 2008. [Na internetu] Dostupno na: http://csl.skku.edu/papers/tecs08.pdf [Pristupljeno 27.8.2019.]

[23] *Recovering Evidence from SSD Drives in 2014: Understanding TRIM, Garbage Collection and Exclusions* , 2014. [Na internetu] Dostuno na: https://articles.forensicfocus.com/2014/09/23/recovering-evidence-from-ssd-drives-in-2014-understanding-trim-garbage-collection-and-exclusions/ [Pristupljeno: 28.8.2019.]

[24] NTFS Directories and Files [Na internetu] Dostupno na: http://www.dewassoc.com/kbase/windows_nt/ntfs_directories_and_files.htm [Pristupljeni 30.8.2019.]

Popis slika

Popis tablica