Technical Disclosure Commons

Defensive Publications Series

August 16, 2019

Classifying SSD input-output streams to optimize storage overprovisioning

Bin Tan

Narges Shahidi

Manuel Benitez

Classifying SSD input-output streams to optimize storage overprovisioning

ABSTRACT

The data update mechanism for a solid-state drive (SSD) involves overprovisioning, e.g., a storage footprint larger than the actual data size, and write amplification, e.g., a physical amount of information written being a multiple of the actual amount of information. Both overprovisioning and write amplification overheads are at least partially a result of treating input data streams as statistically indistinguishable streams and subjecting them to fixed, pre-configured provisioning.

This disclosure describes techniques to classify the input-output streams of an SSD into various types, e.g., non-overlapped sequential data, chunked data streams of variable length, etc. Overprovisioning is optimized for each input stream based on its classification. The techniques thereby reduce both overprovisioning and write amplification overheads, resulting in SSD life and throughput that is significantly better than conventional techniques.

KEYWORDS

- Solid state drive (SSD)
- Flash memory
- Overprovisioning
- Write amplification
- Data stream
- Stream classification
- Non-overlapped data stream
- NAND flash

BACKGROUND

In solid-state drives (SSDs), data is generally not updated in place. Rather, data update in SSDs involves garbage collection, in turn facilitated by overprovisioning of storage capacity. Overprovisioning entails a storage footprint larger than the actual data size. Garbage collection results in write amplification, an undesirable phenomenon where the amount of information physically written is a multiple of the logical amount of information to be written. Both overprovisioning and write amplification overheads are at least partially a result of treating input data streams as statistically indistinguishable streams and subjecting them to fixed, pre-configured provisioning.

Non-volatile memories have features such as namespace and stream, which enable the host to tag the data stream based on the application that generates the data. If the data stream is thus tagged, the SSD can execute different garbage collection algorithms to improve performance. However, conventional execution of tag-based garbage collection requires deep knowledge of the data traffic and fixed overprovisioning for both tagged and untagged streams. Often, the data streams are random and deep data-stream knowledge is unknown. The present techniques of preconfigured, fixed overprovisioning result in poor usage of raw capacity, low write throughput, and high write amplification. In turn, these result in a shortened life and a poor response-time for the SSD.

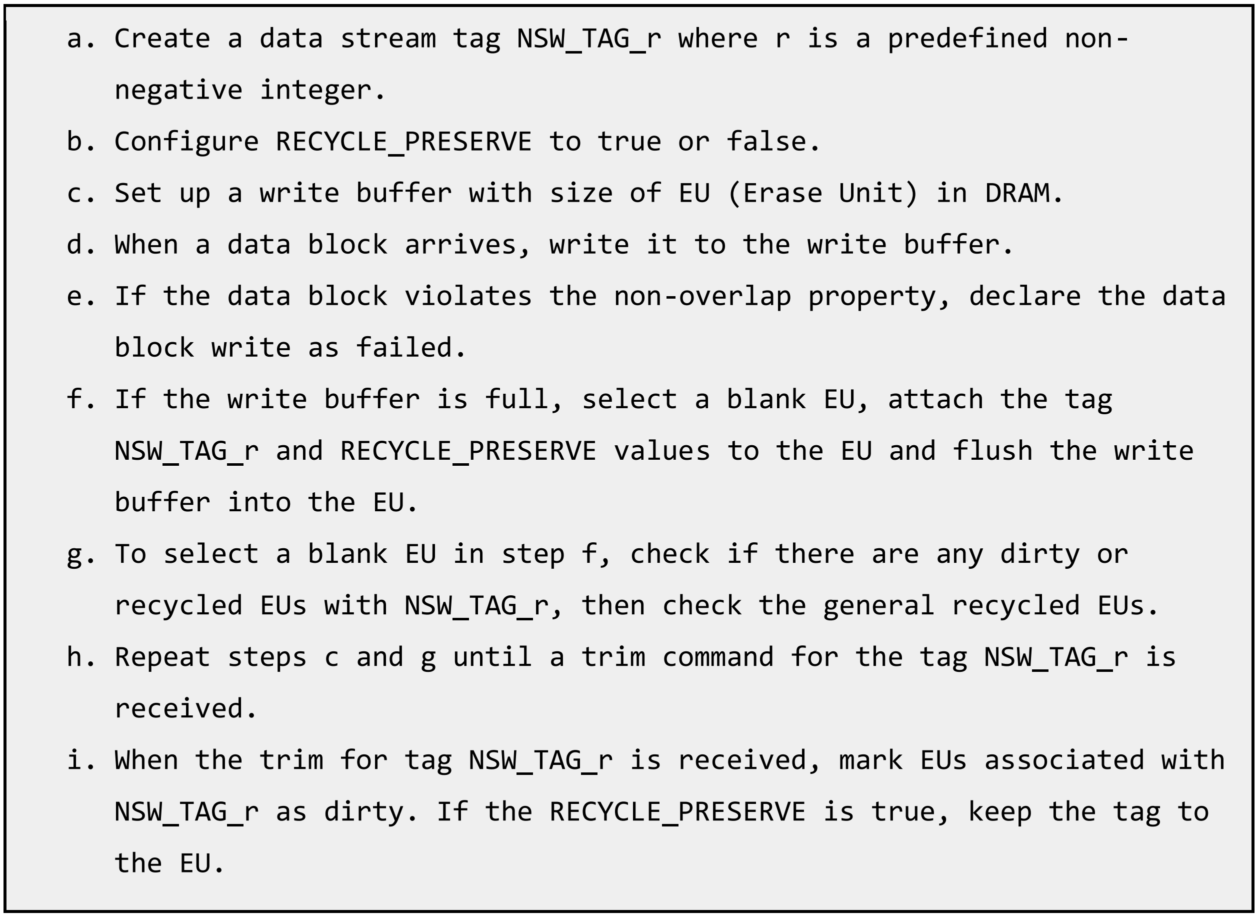
DESCRIPTION

This disclosure describes techniques to classify SSD input-output traffic into classes based on non-overlapped sequential write, sequential alignable data chunk, and data coldness characteristics. Classification is performed without prior deep-traffic knowledge.

Overprovisioning for each class is calculated and dynamically adjusted for each data stream such that high overall SSD performance is achieved.

Non-overlapped sequential write (NSW) data

NSW data is defined as data blocks that arrive at the SSD without overlapped logical block addresses. The total capacity needed by an NSW data stream is not be known until a trim command is received. The real, physical, overprovisioning capacity for such data streams is zero, and the effective overprovisioning (E_OP) is 100%.



```
a. Create a data stream tag NSW_TAG_r where r is a predefined non-
   negative integer.
b. Configure RECYCLE_PRESERVE to true or false.
c. Set up a write buffer with size of EU (Erase Unit) in DRAM.
d. When a data block arrives, write it to the write buffer.
e. If the data block violates the non-overlap property, declare the data
   block write as failed.
f. If the write buffer is full, select a blank EU, attach the tag
   NSW_TAG_r and RECYCLE_PRESERVE values to the EU and flush the write
   buffer into the EU.
g. To select a blank EU in step f, check if there are any dirty or
   recycled EUs with NSW_TAG_r, then check the general recycled EUs.
h. Repeat steps c and g until a trim command for the tag NSW_TAG_r is
   received.
i. When the trim for tag NSW_TAG_r is received, mark EUs associated with
   NSW_TAG_r as dirty. If the RECYCLE_PRESERVE is true, keep the tag to
   the EU.
```

Fig. 1: Treatment of NSW data streams

Fig. 1 illustrates the treatment of NSW streams, per techniques of this disclosure.

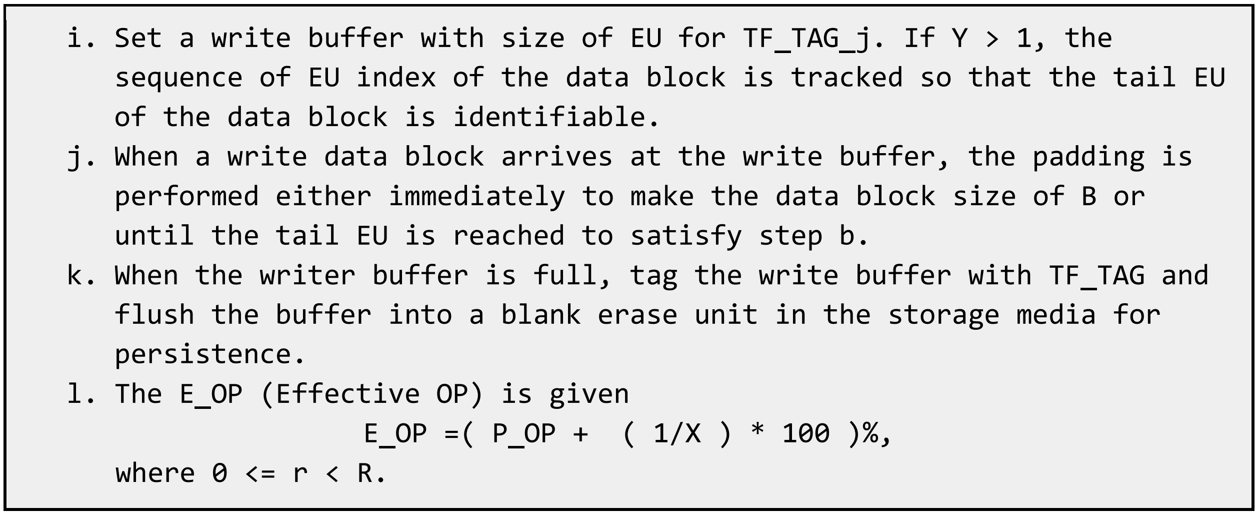
Data chunks that are sequentially alignable with reference to the erase unit of the SSD

An erase unit (EU) is the size of the smallest data storage unit that an SSD can erase and reclaim for the purposes of storing new data. The total user data capacity is denoted as TUC, and the total storage capacity is denoted as TSC. Integer parameters $B > (X \times \Delta)$, $\Delta \geq 0$, $X \geq 1$ and $Y \geq 1$ are configured such that

$$Y \times EU = X \times B,$$

where if $Y > 1$, $X = 1$. All data chunks with sizes in the range of $[B\text{-}\Delta, B]$ are classified as a SADC (Sequential Alignable Data Chunk) stream. Each of Y storage EUs store just X data chunks. For such data chunks, the techniques preserve data block sequential alignment by padding data with reference to the erase unit.

```
a. Create a data traffic tag as TF_TAG_j where j is a predefined non-
   negative integer.
b. Set EU the size of erase unit.
c. Configure integer parameters B > (X * Delta),  Delta >=0, X >= 1 and
   Y >= 1 such that
               Y * EU = X * B,
   where if Y > 1, X = 1.
d. Denote as SAC (Sequential Alignment Class) the class of SSD IO
   traffic with data block sizes in the range of [B-Delta, B] and
   satisfying step c.
e. Set the total user capacity as TUC.
f. Set the Minimum Over Provision to
              M_OP =( (
                    ( (X * Delta) / (Y*EU) ) +
                    ( (K * Y * EU)/TUC )
                      ) * 100 ) %,
   where K >= 1 is a pre-configured number to reserve K*Y blank erase
   units for flushing write buffer and persistent buffering.
g. Configure the storage overprovisioning rate as C_OP and the total raw
   storage capacity TSC as
                 TSC = TUC * (1 + C_OP), where C_OP >= M_OP.
h. Set the overprovisioning rate for performance enhancement as
                P_OP = (C_OP - M_OP).
```

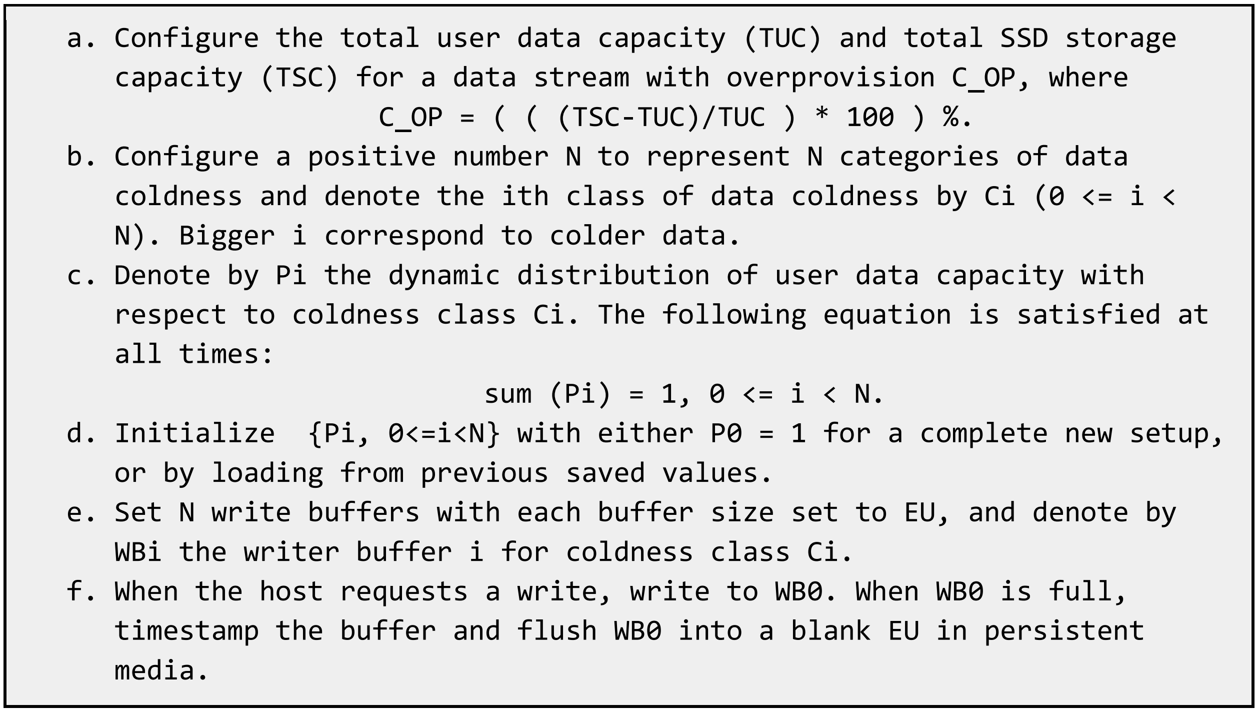


```
i. Set a write buffer with size of EU for TF_TAG_j. If Y > 1, the
   sequence of EU index of the data block is tracked so that the tail EU
   of the data block is identifiable.
j. When a write data block arrives at the write buffer, the padding is
   performed either immediately to make the data block size of B or
   until the tail EU is reached to satisfy step b.
k. When the writer buffer is full, tag the write buffer with TF_TAG and
   flush the buffer into a blank erase unit in the storage media for
   persistence.
l. The E_OP (Effective OP) is given
               E_OP =( P_OP +  ( 1/X ) * 100 )%,
   where 0 <= r < R.
```

Fig. 2: Treatment of SADC data streams

Fig. 2 illustrates the treatment of SADC data streams, per techniques of this disclosure. The condition *E_OP* > *C_OP* enables higher SSD capacity utilization, improves SSD throughput performance, and extends SSD life.

Data coldness and dynamical effective overprovisioning



```
a. Configure the total user data capacity (TUC) and total SSD storage
   capacity (TSC) for a data stream with overprovision C_OP, where
                C_OP = ( ( (TSC-TUC)/TUC ) * 100 ) %.
b. Configure a positive number N to represent N categories of data
   coldness and denote the ith class of data coldness by Ci (0 <= i <
   N). Bigger i correspond to colder data.
c. Denote by Pi the dynamic distribution of user data capacity with
   respect to coldness class Ci. The following equation is satisfied at
   all times:
                      sum (Pi) = 1, 0 <= i < N.
d. Initialize  {Pi, 0<=i<N} with either P0 = 1 for a complete new setup,
   or by loading from previous saved values.
e. Set N write buffers with each buffer size set to EU, and denote by
   WBi the writer buffer i for coldness class Ci.
f. When the host requests a write, write to WB0. When WB0 is full,
   timestamp the buffer and flush WB0 into a blank EU in persistent
   media.
```

```
g. When recycling a programmed EU, write the valid data from the EU into
   an appropriate WBi based on the recycled EU timestamp. The selection
   of the WBi is based on a predefined schema of linear categorization.
   For example, if the recycled EU timestamp is in the range of
   (current_timestamp + (i-1)*beta,  current_timestamp + i*beta], (beta
   > 0 being a pre-configured parameter) WBi is selected. The older the
   timestamp is, the bigger the coldness index i is.
h. When a write buffer WBi is full, flush the buffer into a blank EU in
   the SSD persistent media and assign a calculated-timestamp to it
   based on a predefined schema. For the linear categorization schema,
   the calculated-timestamp is current_timestamp + i*beta.
i. When either user data arrives at WB0, or valid data from recycling
   arrives at WBi, recalculate Pi for 0 <= i < N.
j. The runtime DE_OP (dynamic effective OP) is defined as
               DE_OP = C_OP / (1 - sum ( Fi(Alpha_i, Pi)) ),
   where 0 < i <N, Alpha_i is the weight factor for the coldness
   category Ci, and Fi are the effective functions. The simplest
   effective function is Fi(Alpha_i, Pi) = Alpha_i * Pi.
```

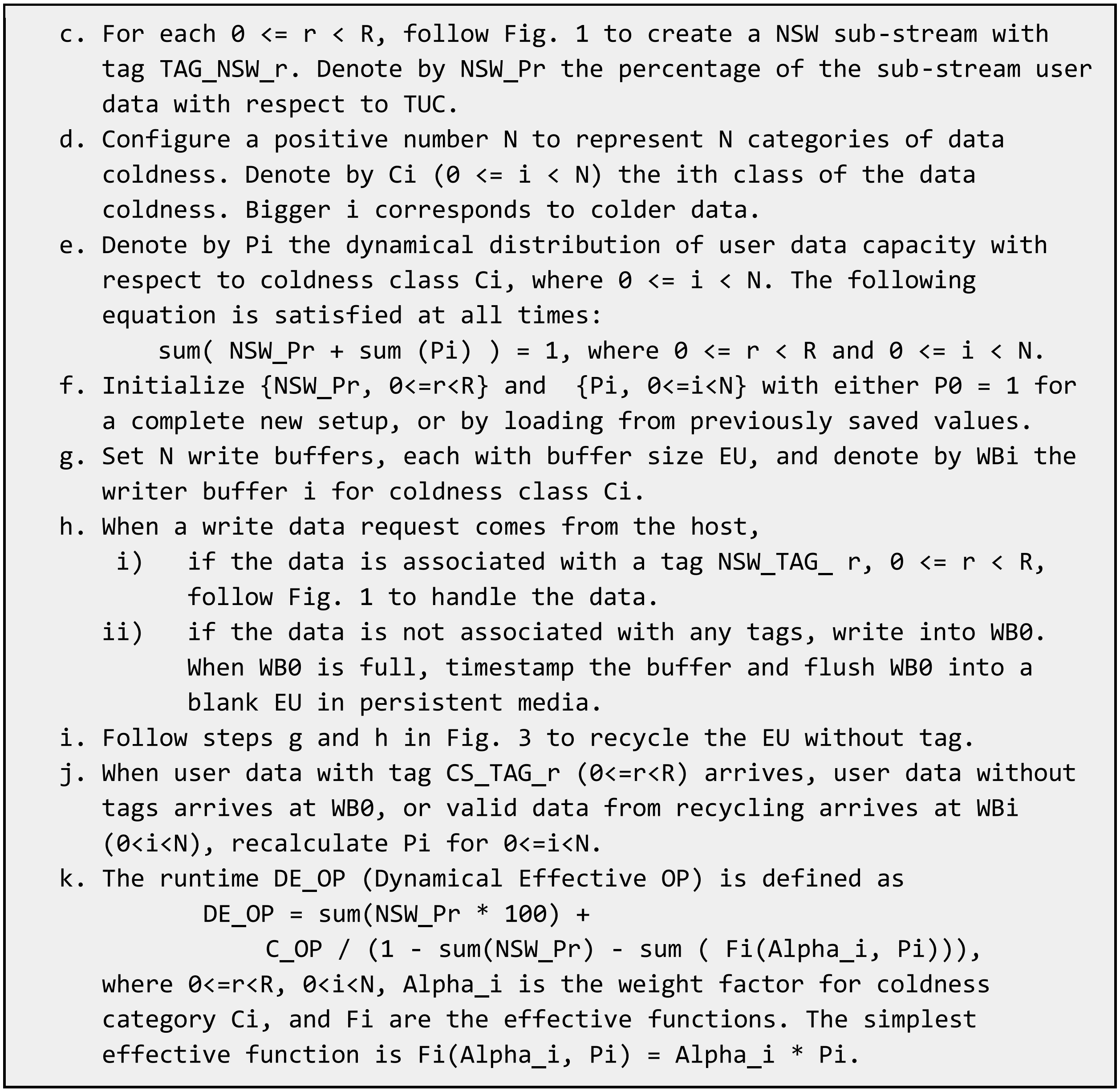
Fig. 3: Overprovisioning based on data coldness

The techniques adaptively classify SSD traffic dynamically distribute data storage overprovision based on data coldness as shown in Fig. 3. The condition *DE_OP > C_OP* achieved via the procedure illustrated in Fig. 3 enables higher SSD capacity utilization, improves SSD IO throughput performance, and extends SSD life.

<u>Classifying data using data coldness and NSW</u>

The techniques can also adaptively classify SSD IO traffic based on both data coldness and NSW sub-streams, as illustrated in Fig. 4.

```
a. Configure the total user data capacity (TUC) and total SSD storage
   capacity (TSC) for a data stream with overprovision C_OP, where
                C_OP = ( ( (TSC-TUC)/TUC ) * 100 ) %.
b. Configure an integer R >= 0.
```



```
c. For each 0 <= r < R, follow Fig. 1 to create a NSW sub-stream with
   tag TAG_NSW_r. Denote by NSW_Pr the percentage of the sub-stream user
   data with respect to TUC.
d. Configure a positive number N to represent N categories of data
   coldness. Denote by Ci (0 <= i < N) the ith class of the data
   coldness. Bigger i corresponds to colder data.
e. Denote by Pi the dynamical distribution of user data capacity with
   respect to coldness class Ci, where 0 <= i < N. The following
   equation is satisfied at all times:
       sum( NSW_Pr + sum (Pi) ) = 1, where 0 <= r < R and 0 <= i < N.
f. Initialize {NSW_Pr, 0<=r<R} and  {Pi, 0<=i<N} with either P0 = 1 for
   a complete new setup, or by loading from previously saved values.
g. Set N write buffers, each with buffer size EU, and denote by WBi the
   writer buffer i for coldness class Ci.
h. When a write data request comes from the host,
    i)   if the data is associated with a tag NSW_TAG_ r, 0 <= r < R,
         follow Fig. 1 to handle the data.
   ii)   if the data is not associated with any tags, write into WB0.
         When WB0 is full, timestamp the buffer and flush WB0 into a
         blank EU in persistent media.
i. Follow steps g and h in Fig. 3 to recycle the EU without tag.
j. When user data with tag CS_TAG_r (0<=r<R) arrives, user data without
   tags arrives at WB0, or valid data from recycling arrives at WBi
   (0<i<N), recalculate Pi for 0<=i<N.
k. The runtime DE_OP (Dynamical Effective OP) is defined as
          DE_OP = sum(NSW_Pr * 100) +
               C_OP / (1 - sum(NSW_Pr) - sum ( Fi(Alpha_i, Pi))),
   where 0<=r<R, 0<i<N, Alpha_i is the weight factor for coldness
   category Ci, and Fi are the effective functions. The simplest
   effective function is Fi(Alpha_i, Pi) = Alpha_i * Pi.
```

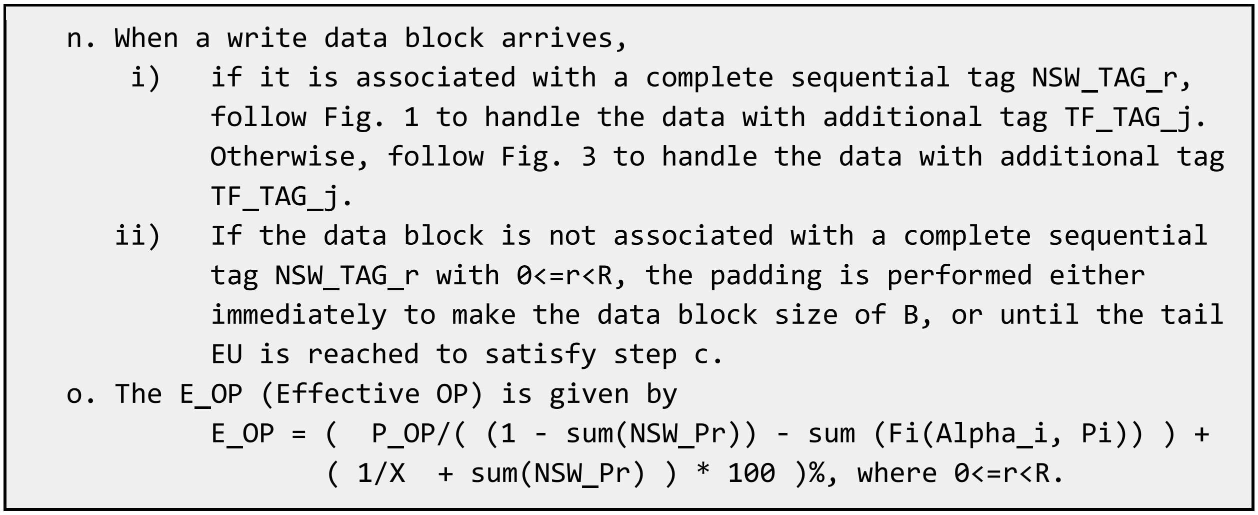
Fig. 4: Adaptive classification of SSD IO traffic based on both data coldness and NSW sub-streams

The condition *DE_OP* > *C_OP* achieved via the procedure illustrated in Fig. 4 enables higher SSD capacity utilization, improves SSD IO throughput performance, and extends SSD life.

Classifying SADC streams using data coldness

The techniques classify SADC streams into *N* data coldness classes. Data block sequential alignment is preserved by padding data with reference to the EU while NSW data sub-streams and data coldness classifications continue to exist.

```
a. Set the data traffic tag (TF_TAG_j), where j is a non-negative
   integer.
b. Set the size of Erase Unit (EU).
c. Configure integer parameters B > (X * Delta),  Delta >= 0, X >= 1 and
   Y >= 1 such that
                  Y * EU = X * B,
   where if Y > 1, X=1.
d. Denote as SAC (Sequential Alignment Class) the class of SSD IO
   traffic with data block sizes in the range of [B-Delta, B],
   satisfying step c.
e. Set the total user capacity (TUC) for the stream with tag TF_TAG_j.
f. Configure an integer R >= 0.
g. For each 0 <= r< R, follow steps from a to e of Fig. 1 to configure
   NSW substream tag NSW_TAG_r. Denote by NSW_Pr the percentage of the
   NSW substream user data with respect to TUC.
h. Set R_Pr the percentage of remaining user data capacity with respect
   to TUC,
               R_Pr = 1.0 - sum(NSW_Pr), where 0<=r<R.
i. Following Fig. 3, set N>=1 categories of data coldness for the
   remaining user data capacity.
j. Set the minimum over provision to
            M_OP =( (
                     ( (X * Delta)/(Y*EU) ) +
                     ( (K * Y * EU)/TUC )
                   ) * 100) %,
   where 0<=r<R, K >= 1 is a pre-configured number to reserve K * Y
   blank erase units for flushing write buffer and persistent buffering.
k. Configure the storage overprovisioning rate as C_OP and the total raw
   storage capacity TSC as
           TSC = TUC * (1 + C_OP), where C_OP >= M_OP.
l. Set the over provisioning rate for performance enhancement as
                    P_OP = (C_OP - M_OP) / R_Pr.
m. Set a write buffer with size EU for the remaining user data. If Y >
   1, the sequence of EU index of the data block is tracked so that the
   tail EU of the data block is identifiable.
```



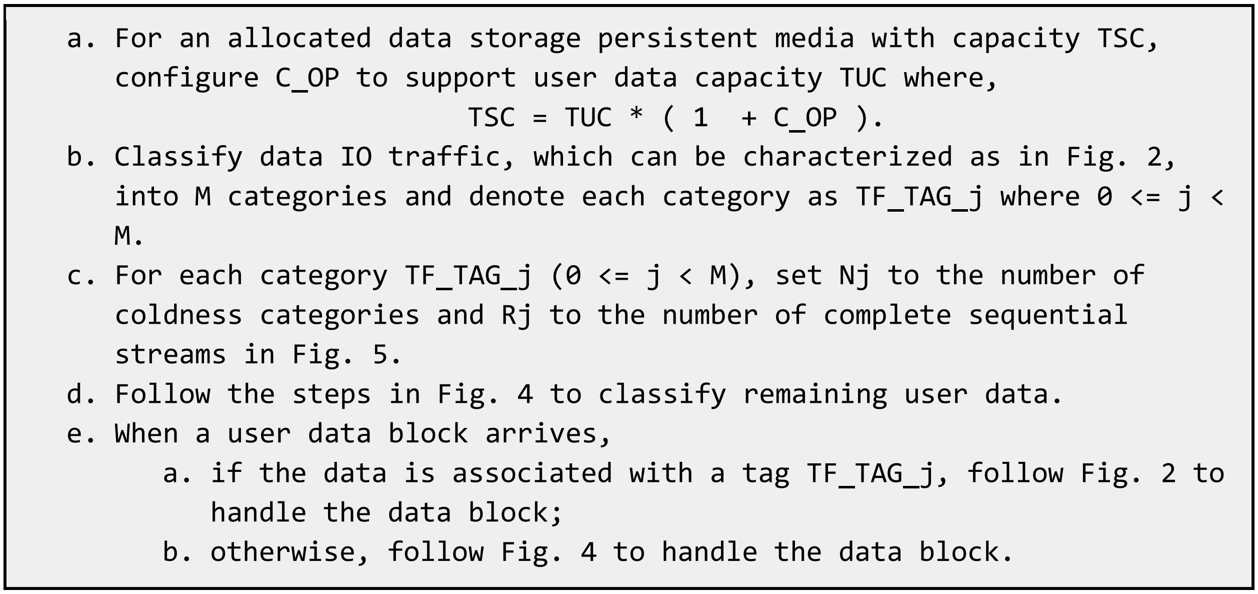
```
n. When a write data block arrives,
    i)   if it is associated with a complete sequential tag NSW_TAG_r,
         follow Fig. 1 to handle the data with additional tag TF_TAG_j.
         Otherwise, follow Fig. 3 to handle the data with additional tag
         TF_TAG_j.
   ii)   If the data block is not associated with a complete sequential
         tag NSW_TAG_r with 0<=r<R, the padding is performed either
         immediately to make the data block size of B, or until the tail
         EU is reached to satisfy step c.
o. The E_OP (Effective OP) is given by
         E_OP = (  P_OP/( (1 - sum(NSW_Pr)) - sum (Fi(Alpha_i, Pi)) ) +
                ( 1/X  + sum(NSW_Pr) ) * 100 )%, where 0<=r<R.
```

Fig. 5: Classifying SADC streams using data coldness

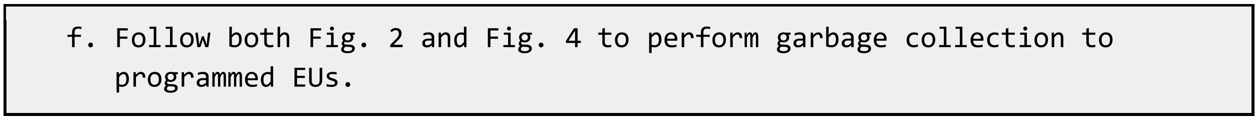
As is seen from Fig. 5, the effective overprovisioning (*E_OP*) is greater than the conventional storage overprovisioning rate (*C_OP*), enabling higher SSD capacity utilization, improving SSD IO throughput performance, and extending SSD life.

Classifying data streams using NSW, SADC, and data coldness

The techniques classify data streams and perform a garbage collection scheme to optimize the utilization, throughput, and life of an SSD.



```
a. For an allocated data storage persistent media with capacity TSC,
   configure C_OP to support user data capacity TUC where,
                          TSC = TUC * ( 1  + C_OP ).
b. Classify data IO traffic, which can be characterized as in Fig. 2,
   into M categories and denote each category as TF_TAG_j where 0 <= j <
   M.
c. For each category TF_TAG_j (0 <= j < M), set Nj to the number of
   coldness categories and Rj to the number of complete sequential
   streams in Fig. 5.
d. Follow the steps in Fig. 4 to classify remaining user data.
e. When a user data block arrives,
      a. if the data is associated with a tag TF_TAG_j, follow Fig. 2 to
         handle the data block;
      b. otherwise, follow Fig. 4 to handle the data block.
```



```
f. Follow both Fig. 2 and Fig. 4 to perform garbage collection to
   programmed EUs.
```

Fig. 6: Classifying data streams using NSW, SADC, and data coldness

Fig. 6 illustrates classifying data streams based on NSW, SADC, and data coldness.

CONCLUSION

This disclosure describes techniques to classify the input-output streams of an SSD into various types, e.g., non-overlapped sequential data, chunked data streams of variable length, etc. Overprovisioning is optimized for each input stream based on its classification. The techniques thereby reduce both overprovisioning and write amplification overheads, resulting in SSD life and throughput that is significantly better than conventional techniques.