Victim Disk First: An Asymmetric Cache to Boost the Performance of Disk Arrays under Faulty Conditions

Shenggang Wan, Qiang Cao, Jianzhong Huang, Siyi Li, Xin Li, Shenghui Zhan, Li Yu, Changsheng Xie, Xubin He
Some important targets of modern storage systems

- **Performance**: Throughput or Response Time
- **Reliability**: MTTDL Mean Time To Data Loss
- **Others**: such as spatial utilization, scalability, manageability, power and so on
**Redundant Array of Inexpensive Disks**

**Classifications:**
- No redundancy RAID, e.g. RAID-0
- Mirror-based RAID, e.g. RAID-1, RAID-10
- Parity-based RAID, e.g. RAID-4, RAID-5, RAID-6, provides high performance, reliability with high spatial utilization
Weakness of Parity-based RAID

- High decoding cost
  - To CPU, it is recognized as computational complexity
  - To the storage device?
    - Extra Reconstruction I/O for User Requests
    - Extra Reconstruction I/O for System Recovery
Different I/O cost of user requests

- Miss to a Surviving Disk
- Request to a Surviving Disk

- Miss to a Faulty Disk
- Requests to all Surviving Disks
Existing Solutions

- **Cache**
  - Take disk or disk array as Cache: redirection of reads, piggy-backing of writes, WorkOut
  - Memory level cache: MICRO

- **Others**
  - Data layout
  - Scheduling
Observation: Different costs between access to the faulty data and surviving data as shown in the example.

Our solution: Victim Disk(s) First (VDF): Give higher priority to cache the blocks from the faulty disks when a disk array fails, thus reducing the I/Os directed to the faulty disks and reducing the extra reconstruction I/O for user requests.
**RGR: A new cache metric**

- **Miss Ratio**, the old metric to cache, limited in description of faulty condition.

- **Requests Generation Ratio (RGR):**
  - The ratio of the number of the requested blocks to the surviving disks and the number of the requested blocks to buffer cache, j/i in figure.
  - It takes into account different miss situations.
  - It can be used to describe the performance and the reliability in faulty condition, quantitatively and directly.
Formal Description of RGR

\[ RGR = \sum_{i=0}^{T-1} (p_i \times MP_i) \]

- \( T \): Total number of data blocks in a disk array.
- \( p_i \): Access probability of each block.
- \( MP_i \): Miss penalty of each block.
RGR and Performance

\[ BW = BW_U \times RGR + BW_R \]

- **BW**: Total serviceability of all surviving disks in terms of I/O bandwidth.
- **BW\(_U\)**: I/O bandwidth available to user workload, **throughput**.
- **BW\(_R\)**: I/O bandwidth for a reconstruction workload.
RGR and Reliability

\[ RD = \frac{Q}{BW - BW_U \times RGR} \]
VDF and RGR

• Essentially, VDF is to replace the block with minimum \((p_i \times MP_i)\) rather than only the \(p_i\), compared to the traditional cache algorithms.

• In many cases, the \(MP_i\) of blocks from faulty disks is larger than \(MP_i\) of blocks from surviving disks and tend to be kept in cache more probably, so we named this scheme as Victim Disk First.
Only read requests are considered here

- Usually, users are more sensitive to read latency.
- Non-volatile memory is deployed as a write cache.

RAID-5 is evaluated: one of the most popular parity-based RAID structures.

- MP\textsubscript{i} of blocks from surviving disk is 1.
- MP\textsubscript{i} of blocks from faulty disk is n-1, n is the number of disks.

Two popular cache algorithms: LRU and LFU

- LRU: Reciprocal of the interval access sequence number is used as \( p_i \) of each block in cache, relatively.
- LFU: Access frequency is chosen here.
Normal mode/faulty mode

- VDF cache only takes effect in faulty condition.
- Two types of stacks are employed to make a smooth conversion.
- Global stack takes charge in fault-free condition.
- Local stacks take charge in faulty condition.
Algorithm 1: VDF-LRU for RAID-5 with \( n \) disks

```
Input: The request stream \( x_1, x_2, x_3, \ldots, x_i, \ldots \)
VDF_LRU_Replace(x_i){
/*For every \( i \geq 1 \) and any \( x_i \), one and only one of the following cases must occur.*/
if \( x_i \) is in \( LS_k, 0 \leq k < n \) then
  /*A cache hit has occurred.*/
  Update \( TS \) of \( x_i \), by \( TS = GTS \);
  Move \( x_i \) to the heads of \( LS_k \) and \( GS \).
else
  /*A cache miss has occurred.*/
  if Cache is full then
    foreach block at the bottom of \( LS_j, 0 \leq j < n \) do
      if \( LS_j \) is a corresponding stack to a faulty disk then
        Its weight \( W = GTS - TS \);
      else
        Its weight \( W = (GTS - TS) \times (n - 1) \);
    Delete the block with maximum \( W \) to obtain a free block;
  else
    /*Cache is not full.*/
    Get a free block.
    Load \( x_i \) to the free block.
    Update \( TS \) of \( x_i \), by \( TS = GTS \);
    Add \( x_i \) to the heads of \( GS \) and the corresponding \( LS \).
  Update \( GTS \), by \( GTS = GTS + 1 \);
}
```

- \( 1/(GTS-TS) \) is \( p_i \)
- \( 1 \) or \( (n-1) \) is \( MP_i \) based on the miss conditions
- Choose the max \( ((GTS-TS) \times MP_i) \) to evict
Algorithm 2: VDF-LFU for RAID5 of $n$ disks

Input: The request stream $x_1, x_2, x_3, \ldots, x_i, \ldots$

VDF_LFU_Replace($x_i$)

/*For every $i \geq 1$ and any $x_i$, one and only one of the following cases must occur.*/

if $x_i$ is in $LS_k, 0 \leq k < n$ then
    /*A cache hit has occurred.*/
    Update $F$ and $TS$ of $x_i$, by $F = F + 1$;
    Move $x_i$ to right place of $LS_k$ and $GS$ according to $F$ and $TS$.
else
    /*A cache miss has occurred.*/
    if Cache is full then
        foreach block at the bottom of $LS_j, 0 \leq j < n$ do
            if $LS_j$ is a corresponding stack to a faulty disk then
                Its weight $W = F \times (n - 1)$;
            else
                Its weight $W = F$;
        Delete the block with minimum $W$ and $GTS - TS$ to obtain a free block;
    else /*Cache is not full.*/
        Get a free block.
        Load $x_i$ to the free block.
        Initialize the frequency $F$ and $TS$ of $x_i$, by $F = 1$ and $TS = GTS$;
        Move $x_i$ to right place of $LS_k$ and $GS$ according to $F$ and $TS$.

Update $GTS$, by $GTS = GTS + 1$;

---

- $F$ is $p_i$
- $1$ or $(n-1)$ is $MP_i$ based on the miss conditions
- Choose the min $(F*MP_i)$ to evict
Simulation

- **Targets:** Effect of VDF on reducing RGR

- **Traces:**
  - SPC-1-web: (Storage Performance Council)
  - LM-TBE, DTRS: (Microsoft)

- **Simulator:** VDF-Sim (about 3000 lines in C)
SPC Web Results of 8 disks

<table>
<thead>
<tr>
<th></th>
<th>65536</th>
<th>131072</th>
<th>262144</th>
<th>524288</th>
<th>1048576</th>
<th>2097152</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td>1.747</td>
<td>1.747</td>
<td>1.741</td>
<td>1.528</td>
<td>0.717</td>
<td>0.227</td>
</tr>
<tr>
<td>VDF-LRU</td>
<td>1.743</td>
<td>1.619</td>
<td>1.273</td>
<td>1.048</td>
<td>0.549</td>
<td>0.226</td>
</tr>
<tr>
<td>LFU</td>
<td>1.747</td>
<td>1.747</td>
<td>1.711</td>
<td>1.404</td>
<td>0.632</td>
<td>0.227</td>
</tr>
<tr>
<td>VDF-LFU</td>
<td>1.575</td>
<td>1.189</td>
<td>0.988</td>
<td>0.904</td>
<td>0.477</td>
<td>0.226</td>
</tr>
</tbody>
</table>
SPC Web Results of 262144 blocks

<table>
<thead>
<tr>
<th></th>
<th>5 Disks</th>
<th>6 Disks</th>
<th>7 Disks</th>
<th>8 Disks</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td>1.592</td>
<td>1.655</td>
<td>1.709</td>
<td>1.741</td>
</tr>
<tr>
<td>VDF-LRU</td>
<td>1.364</td>
<td>1.337</td>
<td>1.308</td>
<td>1.273</td>
</tr>
<tr>
<td>LFU</td>
<td>1.565</td>
<td>1.627</td>
<td>1.679</td>
<td>1.711</td>
</tr>
<tr>
<td>VDF-LFU</td>
<td>1.000</td>
<td>0.984</td>
<td>0.981</td>
<td>0.988</td>
</tr>
</tbody>
</table>
LM-TBE Results

Simulation Results of the LM-TBE Trace

<table>
<thead>
<tr>
<th></th>
<th>5 Disks</th>
<th>6 Disks</th>
<th>7 Disks</th>
<th>8 Disks</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td>1.494</td>
<td>1.462</td>
<td>1.553</td>
<td>1.555</td>
</tr>
<tr>
<td>VDF-LRU</td>
<td>1.441</td>
<td>1.386</td>
<td>1.551</td>
<td>1.497</td>
</tr>
<tr>
<td>LFU</td>
<td>1.515</td>
<td>1.371</td>
<td>1.517</td>
<td>1.429</td>
</tr>
<tr>
<td>VDF-LFU</td>
<td>1.469</td>
<td>1.234</td>
<td>1.521</td>
<td>1.433</td>
</tr>
</tbody>
</table>
 Targets: Effect of VDF on improving the throughput and shortening the reconstruction duration (MTTR).

 Trace: SPC-1-web
Collected block miss information from VDF-Sim, with the real time-stamp (micro second level).

Play the collected requests to the MD device, using direct I/O to bypass file system cache.

Figure 6: Architecture of VDF prototype.
Methodology

- **Open-loop testing**
  - Requests are re-played according to their timestamps (fixed $BW_u$).
  - To find the effect on reconstruction duration (MTTR).

- **Close-loop testing**
  - Requests are re-played one by one.
  - To find the effect on throughput.
### Open-loop Testing Results

<table>
<thead>
<tr>
<th>Disks</th>
<th>Blocks</th>
<th>LRU (s)</th>
<th>VDF-LRU (s)</th>
<th>Improvement</th>
<th>LFU (s)</th>
<th>VDF-LFU (s)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 disks</td>
<td>131072</td>
<td>2662</td>
<td>2543</td>
<td>4.5%</td>
<td>2710</td>
<td>1929</td>
<td>28.8%</td>
</tr>
<tr>
<td></td>
<td>262144</td>
<td>2958</td>
<td>1935</td>
<td>34.6%</td>
<td>2851</td>
<td>1531</td>
<td>46.3%</td>
</tr>
<tr>
<td></td>
<td>524288</td>
<td>1845</td>
<td>1407</td>
<td>23.7%</td>
<td>1786</td>
<td>1310</td>
<td>26.7%</td>
</tr>
<tr>
<td>6 disks</td>
<td>131072</td>
<td>1176</td>
<td>1147</td>
<td>2.5%</td>
<td>1175</td>
<td>964</td>
<td>18.0%</td>
</tr>
<tr>
<td></td>
<td>262144</td>
<td>1234</td>
<td>943</td>
<td>23.6%</td>
<td>1226</td>
<td>921</td>
<td>24.9%</td>
</tr>
<tr>
<td></td>
<td>524288</td>
<td>1027</td>
<td>818</td>
<td>20.4%</td>
<td>1005</td>
<td>806</td>
<td>19.8%</td>
</tr>
<tr>
<td>7 disks</td>
<td>131072</td>
<td>730</td>
<td>685</td>
<td>6.2%</td>
<td>733</td>
<td>652</td>
<td>11.1%</td>
</tr>
<tr>
<td></td>
<td>262144</td>
<td>758</td>
<td>659</td>
<td>13.1%</td>
<td>761</td>
<td>657</td>
<td>13.7%</td>
</tr>
<tr>
<td></td>
<td>524288</td>
<td>691</td>
<td>599</td>
<td>13.3%</td>
<td>687</td>
<td>598</td>
<td>13.0%</td>
</tr>
<tr>
<td>8 disks</td>
<td>131072</td>
<td>504</td>
<td>485</td>
<td>3.8%</td>
<td>509</td>
<td>485</td>
<td>4.7%</td>
</tr>
<tr>
<td></td>
<td>262144</td>
<td>558</td>
<td>501</td>
<td>10.2%</td>
<td>560</td>
<td>501</td>
<td>10.5%</td>
</tr>
<tr>
<td></td>
<td>524288</td>
<td>527</td>
<td>483</td>
<td>8.4%</td>
<td>526</td>
<td>479</td>
<td>8.9%</td>
</tr>
</tbody>
</table>

- **60GB dataset:** 15GB to reconstruct with 5 disks, 12GB with 6 disks...
Results by Changing Numbers of Blocks

<table>
<thead>
<tr>
<th>Throughput Improvement</th>
<th>131072</th>
<th>262144</th>
<th>524288</th>
<th>1048576</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDF-LRU to LRU</td>
<td>14.3%</td>
<td>22.3%</td>
<td>26.2%</td>
<td>22.1%</td>
</tr>
<tr>
<td>VDF-LFU to LFU</td>
<td>27.2%</td>
<td>46.8%</td>
<td>36.5%</td>
<td>25.4%</td>
</tr>
</tbody>
</table>
Future Work

- Integrate VDF into more general cache algorithms.
- Apply VDF to other RAID levels such as RAID-6 to evaluate the impact of VDF on concurrent failures.
Conclusions

- We present an asymmetric buffer cache replacement strategy, named Victim (or faulty) Disk(s) First (VDF) cache, to improve the reliability and performance of a RAID-based storage system, particularly under faulty conditions.

- The basic idea of VDF is to treat the faulty disks more favorably, or give a higher priority to cache the data associated with the faulty disks. The benefit of this scheme is to reduce number of the cache miss directed to the faulty disk, and thus to reduce the I/O requests to the surviving disks overall.

- Our results based on both simulation and prototyping implementation have demonstrated the effectiveness of VDF in terms of reduced disk I/O activities and a faster recovery.
Acknowledgments

- Our thorough shepherd, Erez Zadok, for his very detailed comments and helpful suggestions.

- The anonymous reviewers for their invaluable comments.

- National Basic Research 973 Program of China, the National Natural Science Foundation of China, the National 863 Program of China, and the Innovative Foundation of Wuhan National Laboratory for Optoelectronics.

- U.S. National Science Foundation (NSF).
Thank You!

Questions?