Application agnostic, empirical modelling of end-to-end memory hierarchy

Dhishankar Sengupta, Dell Inc.
Krishanu Dhar, Dell Inc.
Pradip Mukhopadhyay, NetApp Inc.
Abstract

Modeling (Analytical/trace-based/simulation) of memory hierarchy is typically discussed in silos. All the tools-and-techniques available allow us to measure and determine the memory requirement in pockets of the overall configuration. The standard practice of over-provision results in sub-optimal resource allocation and usage.

The solution is to stitch together the end to end memory requirement and provide knobs to try out various parametric manipulations. This approach allows tweaking various parameters and come up with a deterministic requirement of memory at various levels. E.g.: keeping block size constant, change the pattern (sequential-random-access), mix (Read-Write percentage) and the IO working set, cache-agnostic eviction policy, IO concurrency etc.
Issues & Proposal

- Analysis exists in silos
  - Analytical, simulation-based, trace-driven
  - Analysis does not cover the whole service tier (host – storage – disk)
- High degree of complexity for IT generalist

Solution!

- Empirical modelling of end-to-end memory hierarchy
  - Correlation of parameters across tiers
  - Impact of tuning parameters on applications
- Simplistic approach for IT generalist
End to End Memory Hierarchy

Host Side Cache / RoBoCache

Host Side Cache Miss

Storage Cache

Storage Side Cache Miss

Volume Manager

Remote Office / Branch Office
Multiple Hosts
Read Intensive

Dual/Single Controller
Multiple DIMMs

Storage Pools / Flash Pools
Different RAID Levels
High / Low rpm disks
Assumptions

- Read, Write Through cache
  - Read and Write cache seldom resides together

- We assume disks (HDD and/or SSD) will be under a Disk Subsystem, configured with some RAID level redundancy.
Parametric Classification

- **Warm-up**: Effectiveness of the cache depends on how quickly it is warmed up with the application working-set.

- **Intrinsic or Internal**: This is the physical property of the subsystem. E.g.: cache-block size, Read-Buffer-Size of a SSD disk.

- **Extrinsic or External**: Workload induced properties. E.g.: Workload Casualty - a parameter we propose to empirically measure the workload variance on memory hierarchy.
Parametric Analysis: Host Cache

- **Warm-up**
- **Intrinsic**
- **Extrinsic**

\[ f(\text{unique block access, total block referred, time granule}) \]

- **Cache Property** = \( f(S,D,B) \)
- **Eviction Coefficient** = \( f(\text{CacheAge,CacheHit}) \)
- **Cache Miss** = \( f(\text{warmup time, eviction coefficient}) \)

\[ \text{Run-length} = f(\text{Eviction Coefficient}) \]

\[ \text{Workload Casualty} = f(\text{App Blk Size, run-length}) \]
Parametric Analysis : Storage Cache

- Storage Cache
- Warm-up
- Intrinsic
- Extrinsic

\[ f(\text{unique block access, total block referred, time granule}) \]

- Cache Property \( = f(S,D,B) \)
- Eviction Coefficient \( = f(\text{CacheAge, CacheHit}) \)
- Cache Miss \( = f(\text{warmup time, eviction coefficient}) \)

- Run-length \( = f(\text{Eviction Coefficient}) \)
- IO Blk Size \( = f(\text{App Blk Size, no. of blks in granular time}) \)
- Workload Casualty \( = f(\text{IO Blk Size, run-length}) \)
Parametric Analysis: Disk Subsystem

Disk Subsystem

Volume Manager

Disks

HDD

SSD

Read Ahead = \( \frac{B}{\text{Stripe Size}} \times \text{StorageCache Miss} \)

Disk Array IOPs = \( f(\text{disk speed, latency, # of disks}) \)

Disk Array BW = \( f(\text{Disk IOPs, Stripe Size, #of Disks}) \)

Actual Reads = \( \frac{B}{\text{Stripe Size}} \times \text{Storage Cache Miss} \)

Disk IOPs = \( f(\text{Disk Speed, Latency}) \)

Disk BW = \( f(\text{Disk IOPs, Stripe Size}) \)

Rd from Rd Buffer = \( \frac{B}{\text{RdBufferSize}} \times (\text{StorageCacheMiss} - \text{RdBufferMiss}) \)

Rd from disk = \( \frac{B}{\text{PageSize}} \times \text{RdBufferMiss} \)

Type (SLC / MLC)

Write Buffer Size

Erasure Unit
Decision Tree Formation: Host to Storage

<table>
<thead>
<tr>
<th>Cache Age</th>
<th>Cache Hit</th>
<th>Cache behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>High</td>
<td>Retained</td>
</tr>
<tr>
<td>High</td>
<td>Low</td>
<td>Evicted</td>
</tr>
<tr>
<td>Low</td>
<td>High</td>
<td>Retained</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
<td>A function of time, ultimately leading to aged out.</td>
</tr>
</tbody>
</table>
Decision Tree Formation: Storage to Disk

- Storage Cache (S, D)
  - Eviction Coefficient
    - 0
    - 1
    - 10
    - 11
    - Less Cache Miss
    - More Cache Miss
      - Better Run-Length
        - Low WkLd Casually
      - Bad Run-Length
        - High WkLd Casually
    - Disk Array

- Disk Array (D, Stripe Size)
  - Disk Array (IOPs, BW)
    - Read Ahead
      - Good Read Ahead
      - Bad Read Ahead
    - Actual Read from Disk
Decision Tree Parameter Tuning

- Identify tunable parameters across the hierarchy
- Collapse of decision tree based on parameter tuning
- Impact on random and sequential workloads

Example: For Random / Sequential Workload

<table>
<thead>
<tr>
<th>Sub System</th>
<th>Tunable Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume Manager/Disk Array</td>
<td>Add more spindle</td>
</tr>
<tr>
<td>Volume Manager</td>
<td>Increase Stripe Size</td>
</tr>
<tr>
<td>Volume Manager</td>
<td>Add SSD as supplementary cache</td>
</tr>
<tr>
<td>Storage Cache</td>
<td>Modify Cache Block Size, B</td>
</tr>
<tr>
<td>Host Cache</td>
<td>Add more host side cache, (S, D)</td>
</tr>
</tbody>
</table>
Random Workload: Volume Manager: Tuning

- IO(s) are coming to the disks due to storage cache miss.

![Diagram showing Host Cache, Storage Cache, and Logical Volume Manager with arrows indicating improved throughput and cache hit rate based on sequentiality.]

- Add Spindles (High RPM Disks)
- Increase Stripe Size
Random Workload: Volume Manager: Tuning contd.

- Add SSD(s)
- Logical Volume Manager
- Storage Cache
  - Random data fetched faster from SSD(s)
  - Improved throughput
  - Flash pool acts like cache
  - Resulting in low latency
  - Cache miss is compensated by low latency in SSD(s)
Random Workload : Storage Cache : Tuning

Host Cache

Storage Cache

Logical Volume Manager

Improved throughput

Prefetch = f(deg. Of sequentiality)

Cache Hit = f(deg. Of sequentiality)

Improved latency

Modify Cache Block Size
Random Workload: Host Cache: Tuning

Add Cache

Host Cache

Throughput depends heavily on degree of sequentaility

More data eviction

Increase in pre-fetch

May result in choking of pipe due to wasted prefetch

Storage Cache

Logical Volume Manager
Sequential Workload: Tuning

- IO(s) are coming to the disks due to storage cache miss.
Novelty

- Systematic study of end-to-end memory hierarchy
- Decision Tree based problem analysis
- Identify glue point(s) between each subsystem
  - Effect of parameter on subsystems
  - Impact on the application
- Simplistic approach for IT generalist
Thank You
Backup
Definitions

/Set Associative/ Cache: Usually, the cache is a triplet of S, D and B.

S: The number of sets of a set associative cache; D: The degree of associativity; B: The cache block size.

These all are essentially the Intrinsic parameters of a cache sub system.

Cache Age: Intuitively an effectiveness of a cache can Depend on various factors. It can be the intrinsic parameters we discussed so far. It can be the eviction policy. We are taking help of Cache Age as one of the parameter to show how effective the cache is. It can be defined as the average time of a block residing in a cache before being evicted.

Working Set: The cache can only hold a portion of the data from the subsequent systems of the whole hierarchy. Intuitively the Working Set is the portion of the content that is currently available in the cache. Essentially the Working Set of a cache determines the hit percentage from the cache. If a program block does not exists in the cache, and hence in the working set, it results a cache miss. The data would subsequently be fetched from the lower hierarchy (say disk subsystem) and those blocks would be a part of the Working Set of the Cache.

Run Length: It is, intuitively, the spatial locality of the Working Set of the Cache. One can define it as the maximum stream of sequential references. Typically when a program block is read from the application, quite likely, the next read will ask for the subsequently program block. One simple example might be iterating over a set of elements. Hence the Degree of Sequentially is one parameter that will impact the hit ratio of the Cache subsystem.

Stripe Size: The contiguous blocks that can be retrieved in a single IO request from the disk array. By putting more disks under a RAID group, one can get a bigger stripe size. The bigger stripe size would positively influence the response time of reading blocks from disk subsystem by reducing random seek of the disk head.
**Parameter Details : Host Cache**

<table>
<thead>
<tr>
<th>Warm-up</th>
<th>Dependency</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>The net effect is the impact of proper warm up such that the cache will have proper working set, thus reducing cache miss. The intuition is, out of the total blocks referred, if the working set contains all the unique blocks the IO can refer, the cache hit will be maximized.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intrinsic Property Dependency Notes</th>
<th>Property</th>
<th>Dependency</th>
<th>Notes</th>
</tr>
</thead>
</table>
| **Cache Property (Set Associative)** | f (S, D, B) | S: Number of Sets  
D: Degree of associativity  
B: Cache block size |
| The below table can show the intuition behind this: |
| **Eviction Coefficient** | f (CacheAge, CacheHit) | Cache Age | Cache Hit | Cache behavior |
| High | High | Retained |
| High | Low | Evicted |
| Low | High | Retained |
| Low | Low | A function of time, ultimately leading to aged out. |
| **Cache Miss** | f (Cache warm up time, Eviction Coefficient) | @t=0 => everything is a miss  
@warmed up => Depends on Eviction Coefficient |

<table>
<thead>
<tr>
<th>Extrinsic Property Dependency Notes</th>
<th>Property</th>
<th>Dependency</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Workload Casualty</strong></td>
<td>f(Application Block Size, Run Length)</td>
<td>As existing blocks being evicted and replaced by newer blocks, the spatial locality of the cache is inversely affected. Hence the sequentially gets impacted; - so as the Run Length.</td>
<td></td>
</tr>
<tr>
<td><strong>Run Length</strong></td>
<td>f(Eviction Coefficient)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

22
## Parameter Details: Storage Cache

### Warm-up

The net effect is the impact of proper warm up such that the cache will have proper working set, thus reducing cache miss. The intuition is, out of the total blocks referred, if the working set contains all the unique blocks the IO can refer, the cache hit will be maximized.

<table>
<thead>
<tr>
<th>Property</th>
<th>Dependency</th>
<th>Notes</th>
</tr>
</thead>
</table>
| Cache Property (Set Associative) | f(S, D, B) | S: Number of Sets  
D: Degree of associativity  
B: Cache block size     |
| Eviction Coefficient      | f(CacheAge, CacheHit) | The below table can show the intuition behind this:  
|                           |            | Cache Age | Cache Hit | Cache behavior |
|                           |            | High      | High      | Retained      |
|                           |            | High      | Low       | Evicted       |
|                           |            | Low       | High      | Retained      |
|                           |            | Low       | Low       | A function of time, ultimately leading to aged out. |
| Cache Miss                | f(Cache warm up time, Eviction Coefficient) | @t=0 => everything is a miss  
@warmed up => Depends on Eviction Coefficient |

### Intrinsic Property Dependency Notes

**Workload Casualty**

As existing blocks being evicted and replaced by newer blocks, the spatial locality of the cache is inversely affected. Hence the sequentially gets impacted; - so as the Run Length.

<table>
<thead>
<tr>
<th>Property</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Length</td>
<td>f(Eviction Coefficient)</td>
</tr>
</tbody>
</table>

**IO Block Size**

The application block size may not have 1:1 correspondence to the storage side cache block size. Hence one application IO request might result multiple storage side cache IO request. Hence intuitively, it depends on the application block IO requests coming per unit time.

<table>
<thead>
<tr>
<th>Property</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>f(Application Block Size, number of blocks in granular time)</td>
</tr>
</tbody>
</table>
## Parameter Details: Total Disk Subsystem

<table>
<thead>
<tr>
<th>Property</th>
<th>Dependency</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Volume Manager</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read Ahead</td>
<td>((B / \text{StripeSize}) \times \text{Storage Cache Miss})</td>
<td>The number of actual IO request, in terms of the StripeSize, needs to be fetched from the Disk Subsystem</td>
</tr>
<tr>
<td>Disk Array IOPS</td>
<td>(f(\text{Disk Speed in rpm, Latency, Number of Disks}))</td>
<td>In case of SSD, disk rpm is not applicable</td>
</tr>
<tr>
<td>Disk Array Bandwidth</td>
<td>(f(\text{Disk IOPS, Stripe Size, Number of Disks}))</td>
<td></td>
</tr>
<tr>
<td>Actual Read from disk array</td>
<td>((B / \text{StripeSize}) \times \text{Storage Cache Miss})</td>
<td>This is same as Read Ahead. Whereas Read Ahead is prominent for sequential workload, this one is more influential for high degree of randomness.</td>
</tr>
<tr>
<td><strong>Disk Parameters: SSD</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Reads from ReadBuffer</td>
<td>((B / \text{ReadBufferSize}) \times (\text{Storage Cache Miss} – \text{Blocks missed from the SSD ReadBuffer}))</td>
<td>This is the Number of Reads from the SSD ReadBuf.</td>
</tr>
<tr>
<td>Erasure Unit (i.e. SSD Block) Size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type of SSD (SLC/MLC)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Read from SSD Disk</td>
<td>((B / \text{SSD Page Size}) \times \text{Blocks missed from the SSD ReadBuffer})</td>
<td>This is the number of Reads from the SSD Disk.</td>
</tr>
<tr>
<td>WriteBufferSize</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Disk Parameters: HDD</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disk IOPS</td>
<td>(f(\text{disk speed in rpm, latency}))</td>
<td></td>
</tr>
<tr>
<td>Disk Bandwidth</td>
<td>(f(\text{disk IOPS, StripeSize}))</td>
<td></td>
</tr>
</tbody>
</table>