# Table of Contents

- Content-Based Addressing ................................................................. 1
- Cost ........................................................................................................ 1
- Experimental Analysis ........................................................................ 1
- lxbatch ................................................................................................. 2
- SLC3 and SLC4 .................................................................................... 2
- Summary ............................................................................................... 2
- Implementation ..................................................................................... 2
Content-Based Addressing

Content-Based Addressing, aka. Content-Based Hashing, is a useful technique for speeding up the transfer of VM images across the network.

Consider the above illustration. Three block device files contain filesystem images for VMs. For simplicity, they each contain only four blocks.

Given they are ext2 or ext3 images, each file within the image is 1k aligned. Depending on the size of the image or configuration, ext2 and ext3 filesystems can also 2k or 4k align files.

Contents of different blocks are calculated with a hash algorithm, e.g. SHA. In the illustration, sha(block1) = A, sha(block2) = B, etc. for the first image.

Blocks that are in common for several images are called "hot" or "golden" blocks. A more descriptive naming scheme is "primary" for blocks that exist in all images, "secondary" for blocks in all but one image - i.e. given m images, an n-ary block exists in m - n images. In the illustration, A is a primary block, and B is a secondary block.

If two VM images share many blocks, then these blocks need not be transferred two times. Indeed, when an image is to be transferred to a target machine which already contains a set of similar images, only parts of the image needs to be transferred.

For example, if a target VMM machine already has the first VM image in its image repository, only half of the second image would have to be transferred to the same VMM, since blocks A and B are already in the repository.

Cost

The cost of generating a hash table is linear, but the table should also be sorted, which amounts to \( n \times \log n \), where \( n \) is the number of blocks in the image.

The fractional cost of a hash table is \( \text{length(hash)} / \text{block size} \). The cost of an SHA hash is 20 bytes. Given that a block can be 1k, 2k or 4k, the total cost is 2.0, 1.0 or 0.48 %, respectively, of the total image size.

Experimental Analysis

The following experiments can be considered somewhat worst-case scenarios. In the "lxbatch" experiment the images are larger than what data would normally be in a VM image. In the "SLC3 and SLC4" experiment, the images are more asymmetrical than what one could expect to find in an image repository.
**lxbatch**

The two root filesystems of lxb7562 and lxb7564 have been compared, looking for hot blocks.

The filesystems are 4k aligned and have been compressed (without deleting data) to 1400000 blocks (5.3 GB).

Scanning and comparing the blocks shows that 1176582 blocks are shared, which is 84% of the total image. This means that 16% (873 MB) of the image would need to be transferred.

The overhead of the hash table would be $20 \times 1400000 \text{ bytes} = 26.7 \text{ MB}$.

**SLC3 and SLC4**

Two images, SLC3 and SLC4, generated by OS Farm, are compared. Both are 1k aligned but the SLC4 image is also 4k aligned, so 1k blocks are compared. The SLC3 and SLC4 images have 351232 and 780288 1k blocks, respectively.

If SLC4 already exists on the target machine, 52% of SLC3 would need to be transferred. The overhead of the hash table would be $20 \times 351232 = 6.70 \text{ MB}$.

If SLC3 already exists on the target machine, 78% of SLC4 would need to be transferred. The overhead of the hash table would be $20 \times 780288 = 14.9 \text{ MB}$.

-- Main.hbjerke - 07 Jun 2007

**Summary**

Bad or worst-case image transfer scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Normal transfer</th>
<th>Content Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>lxbatch to lxbatch</td>
<td>5.3 GB</td>
<td>0.88 GB</td>
</tr>
<tr>
<td>SLC3 to SLC4</td>
<td>343 MB</td>
<td>185 MB</td>
</tr>
<tr>
<td>SLC4 to SLC3</td>
<td>762 MB</td>
<td>609 MB</td>
</tr>
</tbody>
</table>

**Implementation**

JCBT is a Java implementation of Content Based Transfer. It can speed up VM image propagation and reduce network congestion.

This topic: Virtualization > ContentBasedAddressing
Topic revision: r4 - 2007-08-27 - HenrikHemmestadBjerke

Copyright &© 2008-2019 by the contributing authors. All material on this collaboration platform is the property of the contributing authors. Ideas, requests, problems regarding TWiki? Send feedback