DEDIS: Exact Deduplication for Primary Distributed Storage*

J. Paulo J. Pereira

University of Minho & INESC TEC
{jtpaulo,jop}@di.uminho.pt

Abstract

The removal of duplicate data from primary storage volumes in a cloud computing environment is increasingly desirable, as the resulting space savings contribute to the cost effectiveness of a large scale multi-tenant infrastructure. However, traditional archival and backup deduplication systems are not suited for large scale virtualized infrastructures and the I/O demanding applications there deployed. In fact, few deduplication systems address primary data and are either restricted to special file systems or centralized designs.

This paper presents DEDIS, a dependable and fully distributed deduplication system for large-scale cloud infrastructures with a common primary storage pool abstraction. Unlike previous proposals, DEDIS does not depend on specific file systems with built-in aliasing operations while ensuring low storage I/O overhead, as described in our preliminary evaluation results. In more detail, DEDIS is implemented within Xen as a blktap driver and performs off-line deduplication among remote virtual machines.

1 Introduction and Background

Deduplication, as has been in use for a long time in archival and backup systems, is an appealing technique to mitigate the costs of storing large volumes of data [6]. The emergence of cloud computing brings novel opportunities, needs, and means to apply deduplication to general purpose primary storage volumes. Namely, cloud infrastructures must store persistently a large amount of virtual machine (VM) volumes that would benefit from deduplication space savings. In fact, existing studies show that deduplication can reduce the space occupied by general purpose VMs up to 80% [1].

Primary storage deduplication raises new challenges that are not addressed in traditional archival/backup deduplication systems. Most of these systems use in-band deduplication in which deduplication is performed in the storage write path before storing the written data persistently [6]. Such decision avoids writing duplicate content to the storage but, for I/O intensive applications, performing deduplication in the critical I/O path may introduce unacceptable overhead in write requests latency. In order to reduce the I/O latency overhead, some systems perform off-line deduplication that decouples deduplication from storage I/O and performs both asynchronously, which allows reducing the latency of storage I/O operations but requires additional storage space as data is only shared after being stored [1]. Other systems leverage data temporal and spatial locality for performing in-band deduplication without introducing significant storage I/O overhead [5]. However, for primary workloads exhibiting poor locality, these systems obtain increased I/O latency and reduced space savings. Moreover, most backup and archival storages assume immutable data, thus avoiding the need of copy-on-write (CoW) to prevent updates on data shared by several entities. Such does not hold true for primary storages where CoW mechanisms are required, increasing the complexity of reference management and impacting the latency of storage requests. Finally, few primary storage deduplication systems are thought for distributed infrastructures and either have centralized components or

*The title of the poster is also "DEDIS: Exact Deduplication for Primary Distributed Storage". If accepted, we will be presenting the poster at Eurosys 2013 without a demo.

depend on specific cluster file systems with special primitives for performing deduplication, that may pose as performance bottlenecks [1].

Next we present DEDIS, an off-line fully distributed deduplication system for a distributed infrastructure with a common storage pool abstraction where VMs primary volumes are persisted. DEDIS introduces negligible storage I/O overhead while providing exact deduplication, thus eliminating all duplicate data found across remote VM volumes. DEDIS does not depend on locality or special file system assumptions, requiring only a mechanism for intercepting I/O requests, which is common in most hypervisors.

2 DEDIS

Next, we overview DEDIS architecture and present some preliminary evaluation results.

Interceptor. In each server, a local module intercepts VMs I/O requests, at the block granularity, and maps logical to physical storage addresses. The physical location of each logical block is stored persistently as metadata. Physical block aliasing is enabled by pointing logical blocks to the same physical address. Shared physical addresses are marked as CoW for preventing updates that are written to an unused storage block.

Distributed Duplicates Index (DDI). A distributed module indexes the primary storage's blocks with unique content. For each block (entry), a checksum of the block's content, the block's physical address and the number of logical addresses sharing that block are stored. This information is used for aliasing duplicate blocks and to perform reference management and garbage collection of unreferenced blocks. Index entries are distributed over several servers and are located with a message routing service. Each entry has a small size allowing to store several entries in the same node and requiring a small number of DDI nodes for large clusters.

Share. I/O write requests are registered by local *interceptors* and are collected asynchronously by *share* module that runs locally in each server. For each written block, the block is marked preemptively as CoW, a signature of its content is generated and a remote call to the DDI is made to check for duplicates. If a match is found, the VM logical address is updated to the shared block and the old block freed, otherwise a new entry is added at the DDI so that future blocks with the same content can be aliased.

Garbage Collector (GC). The *GC* processes copied blocks or, in other words, aliased blocks that were updated and are no longer being referenced by a certain logical address. The number of references to a specific block can be consulted and decremented at the DDI and blocks can be freed if they are no longer being referenced.

Extent server. The extent server is a distributed coordination mechanism that manages a common pool of storage unused blocks. Unused blocks are necessary when a logical volume is created or when an aliased block is updated (i.e., copied on write). Storage extents are allocated with a large granularity and are then, within each physical host, used to satisfy individual block allocation requests, thus reducing the overhead involved in using a remote service [2]. Blocks freed by *GC* and *Share* are placed in the local extents and can be sent back to the *extent* server.

DEDIS was model checked and is resilient to crash failures and restart of nodes by using transactional logs [4]. Periodical checkpoints prune old log entries and update a persistent version of the logical to physical mapping, making recovery faster. Moreover, an open-source prototype of DEDIS[1] is implemented within XEN blktap mechanism and was evaluated with DEDISbench, an open-source benchmark that generates realistic content distributions, which is not supported in traditional disk I/O benchmarks [3]. This evaluation shows that DEDIS design and other optimizations, omitted from this paper due to space reasons, allow achieving negligible overhead, when compared to the default XEN blktap driver, with both intensive I/O and deduplication running concurrently. Namely, with DEDISbench simulating a stress I/O load in 6 VMs distributed among 2 machines, the overhead of DEDIS in I/O throughput is less than 18% and for a stable I/O load (50% of the stress load) this overhead is negligible. The evaluation also shows that the space saved with deduplication compensates the algorithm's metadata overhead and that the CPU, RAM and network resources usage is not significant.

[1] `https://launchpad.net/holeycow/block-based-deduplication`

3 Conclusion and Future Work

We present DEDIS, a fully distributed off-line deduplication system for cloud computing primary storage infrastructures. DEDIS only requires a mechanism for intercepting I/O requests, allowing to implement it with most hypervisors and optimizing the aliasing and copy-on-write mechanisms that are bottlenecks of current systems. Our preliminary results show that DEDIS achieves low storage I/O overhead while actively sharing blocks, thus not requiring deduplication to run in off-peak periods as in most off-line deduplication proposals. Currently, DEDIS prototype is being further evaluated and optimized in order to further prove that it is possible to achieve low storage I/O overhead and scale to large infrastructures.

Acknowledgements

Partially funded by Ph.D grant SFRH/BD/71372/2010.

References

[1] Austin T. Clements, Irfan Ahmad, Murali Vilayannur, and Jinyuan Li. Decentralized Deduplication in SAN Cluster File Systems. In *USENIX Annual Technical Conference (ATC)*, 2009.

[2] Dutch T. Meyer, Gitika Aggarwal, Brendan Cully, Geoffrey Lefebvre, Michael J. Feeley, Norman C. Hutchinson, and Andrew Warfield. Parallax: Virtual Disks for Virtual Machines. In *European Conference on Computer Systems (EuroSys)*, 2008.

[3] J. Paulo and J. Pereira. Dedisbench: A benchmark for deduplicated storage systems. In *International Symposium on Secure Virtual Infrastructures (DOA-SVI)*, 2012.

[4] Joao Paulo and Jose Pereira. Model checking a decentralized storage deduplication protocol. In *Fast Abstract in Latin-American Symposium on Dependable Computing*, 2011.

[5] Kiran Srinivasan, Tim Bisson, Garth Goodson, and Kaladhar Voruganti. iDedup: Latency-aware, Inline Data Deduplication for Primary Storage. In *USENIX Conference on File and Storage Technologies (FAST)*, 2012.

[6] Benjamin Zhu, Kai Li, and Hugo Patterson. Avoiding the Disk Bottleneck in the Data Domain Deduplication File System. In *USENIX Conference on File and Storage Technologies (FAST)*, 2008.