10.48047/jocaaa.2026.35.01.38

Memory Deduplication and Copy-on-Write Optimization in Rapid Virtual Machine Provisioning Systems

Shruthi Karpur
Broadcom Inc., USA

Abstract

Traditional virtual machine replication based on full copy cloning results in heavy resource duplication as well as overheads in time that discourage dynamic scaling properties found in current cloud settings. Architectural improvements based on memory deduplication and copy -o n -w rite functionality radically alter provisioning processes to allow for instant bootstrapping of functional virtual machines from pre existing virtual machines. Hypervisor -l evel deduplication techniques track down and merge replica memory pages within different virtual machines, resulting in greatly decreased physical memory demands. Delta disk architecture satisfies copy -o n -w rite constraints at the storage stack, thereby allowing child disk images to dependently point to immutable parent structures while retaining only changed storage blocks within the local context. Virtual machine forking applies copy -o n -w rite concepts to full execution state replications, which greatly simplify lightweight virtual machine instance creations without incurring heavy duplication costs in traditional approaches. Implementation details involve managing trade -o ffs between deduplication scans to save memory resources, managing copy -o n -w rite page faults during divergence phases, and handling complex parent -c hild relationship dependencies among different virtual machines.

Keywords: Virtual Machine Provisioning, Memory Deduplication, Copy-On-Write Mechanisms, Hypervisor Optimization, Cloud Infrastructure

1. The Latency and Resource Cost of Traditional VM Cloning

Virtual machine migration can be considered a prime operational need in a data center, as a result of which there emerges a need for a smooth transition of running virtual machines from one physical machine to another. The technology for live migration facilitates a smooth transition from one physical machine to another, resulting in little or no disruption while migrating virtual machines, because, under the hood, there lie complexities [1]. The process for migrating a virtual machine includes transferring the entire state, such as the register contents, device context, and the entire memory allocated, from a source physical computer onto a destination computer.

The pre-copy migration technique deals with the problem of reducing downtime by iteratively copying memory pages from the source to the destination while the virtual machine keeps running on the source machine [1]. In the initial phase, all memory pages are sent to the destination, and a baseline memory image is built. Then, in iterative phases, changes to pages during previous transmission phases are tracked and sent back, gradually shrinking the set of dirty pages to be transferred. This iterative phase ends once the set of dirty pages reaches a sufficiently small threshold; then the virtual machine is paused briefly while state transfer is completed before resumption on the destination machine. However, being highly sensitive to memory write rates, if workloads tend to write pages often, they might do so quicker than the network can send the dirty pages to the destination, and hence may cause migration to continue for a prolonged period due to lack of convergence.

10.48047/jocaaa.2026.35.01.38

Resource allocation and placement optimization for the virtualized environment involves complexities beyond migration dynamics. Placement techniques for the virtualized environment face the task of analyzing and monitoring patterns of resource usage and performance characteristics for optimal virtual machine placement on physical resources [2]. Violations of the service level agreement arise due to conflicts of resources, leading to suboptimal performance of applications, requiring corrective actions through placement techniques. Placement strategies involving migrations of virtual machines can result in suboptimal performance during the migration of the virtual machines due to their complexity and conflicting criteria of placement strategies. The placement strategy involves simultaneous maximization of objectives like consolidation of virtual machines for power efficiency, performance isolation, and minimization of migration costs.

2. Memory Sharing Architecture and Hypervisor-Level Deduplication

Memory resource management in virtualized environments presents fundamental challenges in balancing overcommitment benefits against performance guarantees, requiring sophisticated mechanisms to reclaim and share physical memory across competing virtual machines. The hypervisor must implement transparent memory management techniques that operate without guest operating system awareness while maintaining isolation and security properties [3]. Ballooning mechanisms enable cooperative memory reclamation by installing a driver within guest operating systems that allocates memory on behalf of the hypervisor, effectively applying guest-level memory pressure to trigger internal paging mechanisms. This approach leverages guest operating system knowledge of page value and access patterns to make informed reclamation decisions superior to external hypervisor-level page replacement policies.

Content-based page sharing represents an opportunistic memory reclamation technique that identifies and consolidates identical memory pages across virtual machine instances, transparently reducing physical memory consumption without requiring guest modifications [3]. The sharing mechanism operates by computing cryptographic hash values for memory pages and maintaining a system-wide hash table mapping content fingerprints to physical page frames. When multiple pages hash to identical values, bytelevel comparison confirms true identity before consolidating references to a single physical page marked copy-on-write. This transparent sharing proves particularly effective in environments hosting multiple instances of similar operating systems or applications, where substantial memory content duplication naturally occurs in kernel code, shared libraries, and common application data structures.

The Difference Engine architecture extends basic page sharing concepts by implementing sub-page level deduplication and compression to maximize memory savings in virtualized environments hosting similar virtual machine instances [4]. Rather than requiring complete page-level identity, the system performs patch-based sharing where pages exhibiting substantial similarity are stored as a full base page plus compact binary patches representing differences. Memory pages are organized into sharing relationships where a single physical page serves as the base, with related pages represented as different patches applied at access time. Compression techniques further reduce memory footprint by identifying and compacting pages with high compressibility ratios. The architecture implements multi-level hashing to efficiently identify sharing candidates, employing coarse-grained hashing for initial filtering followed by fine-grained comparison for confirmation. The system dynamically adapts sharing strategies based on observed memory access patterns and modification rates, balancing compression overhead against achieved memory savings. Experimental evaluation demonstrates that sub-page sharing and compression deliver substantially higher memory reduction ratios compared to traditional full-page sharing in workloads with high similarity but imperfect identity, though the computational overhead of maintaining and applying patches introduces performance considerations requiring careful management.

10.48047/jocaaa.2026.35.01.38

3. Copy-on-Write Mechanisms and Delta-Disk Provisioning

Cloud storage infrastructure designs are faced with inherent challenges of ensuring data durability, accessibility for reading, and economic viability, necessitating the use of sophisticated designs that address these design challenges simultaneously on an infrastructure that spans different geographic locations. Cumulus, the backup solution, provides a solution for long-term archival needs by incorporating an efficient file system backup solution for cloud storage infrastructure, thereby proposing methods that reduce cloud storage bandwidth and cost, while ensuring reasonably good restore times [5]. This solution takes an incremental backup by identifying the changed file system blocks that had been modified previously in the backup process, thereby transmitting only the changed parts of the data to the storage infrastructure that is geographically separate. Chunking techniques for large files are applied by dividing the large files into chunks of varying sizes, which allows for the efficient identification of unchanged chunks even when insertion or deletion operations produce byte offset changes in the file. The architecture uses aggressive data deduplication to avoid storing the same content repeatedly in backup snapshots, computing crypto-fingerprints for every segment of the data, and a global hash map that resolves these fingerprints to storage locations [5]. Such a content-addressed storage infrastructure means that if two or more files, directories, or versions of a backup contain the same content, it will be stored just once, and all references will map to this unique storage slot. Moreover, the deduplication hash map also poses a challenge in managing metadata, since every backup segment submitted must first query the hash map for storage. The architecture also includes intelligent layout strategies that combine several related segments of a backup job into a large storage object, directly addressing the high per-object charges imposed by cloud storage systems that charge for the number of requests and the number of storage objects, rather than actual storage size.

The scanning strategies for memory deduplication should address two contradicting goals to increase sharing opportunities while reducing CPU costs used in page comparisons. The improvements in kernel same-page merging include analysis of input/output access patterns for efficient scanning of deduplication, targeting areas of memory that display similarity prerequisite properties [6]. The scanning process of traditional systems targets the exploration of memory pages randomly, wasting CPU usage in scanning unstable areas of volatile memory that lack sharing gains. The new system tracks access patterns and write operation counts for pages to segregate areas of memory of varying sharing gains and targets scanning in pages that see heavy access, while skipping areas experiencing frequent write operations that lead to quick dissolution of sharing of the mapped page due to copy-on-write page faults. The system enforces hints used in block device input/output operations to detect memory pages that are most likely stable in the system, including areas holding read-only file system files and executable code areas.

4. Process Forking and Snapshotless Cloning Techniques

This instantiation of virtual machines in an efficient and rapid manner can be considered extremely desirable in the field of cloud computing applications related to scalable processing. This requirement emerges due to demands on workloads within scalable computing applications to create several identical computing processing slots instantaneously. This requirement of scalable computing can be served through the implementation of the virtual machine “cloning” feature by the SnowFlock system based on an operating system “fork” facility. This enables creating “child” virtual machines that have identical memory states similar to those of a parent virtual machine [7]. The “cloning” feature of creating “child” virtual machines not only captures all execution states of an active virtual machine, like “registries” of an execution processor, memory contents, “pages” of an execution memory, but continues execution of “child” virtual

10.48047/jocaaa.2026.35.01.38

machines based on an identical execution state. The aforementioned traditional "cloning" facility requires an entire memory "copy" of an active virtual machine to complete execution within "child" virtual machines, thus requiring significantly extra processing time. The "fork" facility requires a "copy-on-write" processing action based on "modification" of memory execution states.

Virtual machine descriptions are containers that enclose the minimum state required to reconstruct the execution contexts on different physical machines [7]. Multicast transfer of these optimized descriptions allows multiple child clones to be created on a distributed infrastructure in parallel without incurring substantial network costs. However, the actual memory state resides on the parent machine at first, while child processes can trace pages on demand as execution progresses in response to the emerging patterns of memory access. This notion corresponds to the aggressive page prefetching strategies employed by the demand-tracing algorithm, referring to the prediction of potential access patterns to pages in the near future from their execution patterns. In addition, mechanisms are designed within the system to trace pages possibly accessed by all child processes. For example, executable code or read-only data structures are traceably multicasted to all child processes simultaneously.

Xen-Blanket extends VM mobility patterns to include FAST's concept of nested virtualization to achieve seamless VM migration over heterogeneous hypervisor platforms without VM downtime or cooperation [8]. This system introduces a thin layer of virtualization that translates a compliant virtual hardware interface for guest VMs to adapt to heterogeneous hypervisors. This design is ideal since it achieves a certain level of processing overhead in exchange for compatibility flexibility to allow for workload migration between heterogeneous hypervisors that may not have been compatible before. This system introduces effective memory and device states transfer functions that minimize downtime when migrating VMs among hypervisors when a virtualization layer is applied. Opportunities for page sharing may arise both from a virtualization layer and from a base hypervisor system, but with complexities when combining both systems' interactions via virtualization layers.

5. Performance Optimization and Scalability Considerations

Throughout this report, Optimization for memory page sharing in virtualized data center environments is achieved through smart and optimized virtual machine placement techniques to maximize the sharing opportunity by grouping virtual machine instances with a similar memory profile together in the same physical machine. Memory Buddies is one such system for workload-aware virtual machine placement. During placement, Memory Buddies takes memory similarity into account. The approach groups virtual machines with a likely sharing opportunity together in the same physical resource infrastructure [9]. Conventional virtual machine placement techniques only rely upon capacity and load balancing without any familiarity with memory characteristics. The optimized virtual machine placement framework includes memory fingerprinting for the computation of similarity measurements for virtual machine pairs. The placement optimization problem attempts to optimize overall memory savings within the data center by appropriately allocating virtual machines to physical machines in a way that allows similar workloads to share infrastructure [9].

This is an optimization problem whereby the system needs to solve an assignment problem with conflicting interests between placement for memory savings and other placement objectives, aiming at balancing loads. It uses a greedy approach that refines the placement process for beneficial migration opportunities as those involving the migration of a virtual machine to another physical machine with a substantial increase in sharing. This takes into account the cost of migration in making placement decisions, as frequent placement changes might be detrimental despite increasing sharing. Experimental analysis shows memory sharing ratios to be much higher in similarity-inspired placement compared to similarity-unaware placement

10.48047/jocaaa.2026.35.01.38

approaches, with this gain more visible in environments with diverse workloads. "Resource allocation within database server environments in virtualized storage systems is particularly complex due to the intricate relationships among available computational resources, storage system performance, and characteristics of query workloads. Dynamic resource allocation techniques require adaptation based on varying levels of workloads through adjusting virtual machine resources, adding or subtracting CPU and memory based on fluctuating query workloads [10].

The techniques require consideration of the non-linear relationships between different amounts of resources and application performance, suggesting that some resources act as 'bottlenecks' and hamper further efficiency of other abundant resources. A database environment is characterized by high dependency on storage input/output performance, suggesting that additional computational resources contribute little improvement if storage systems do not offer comparable storage throughput. The design involves model-based performance prediction techniques characterizing estimates of application-level potential of changes resulting from resource allocation. The design involves application-level performance metrics used for modeling relationships between different amounts of resources and query completion time taken by applications."

6. Architectural Challenges of CoW/Deduplication Integration and Management

Cloud infrastructure factors must be considered carefully during the performance evaluation of cloudbased applications because they contribute to performance latency variabilities that do not exist in traditional fixed-hosting solutions. Observational studies demonstrate the sensitivity of latencyintensive/cloud-aware applications that experience significantly increased response time variabilities on virtualized cloud platforms relative to physical environments because of mutual interference introduced by the sharing of common physical resources, depending on the nature of concurrent workloads on the cloud infrastructure [11]. Network latency variabilities are also observed in the cloud environment owing to software networking, which introduces additional virtualization stacks or because of the increased possibilities of software packet processing [8].

The performance evaluation approach for characterizing performance needs to consider the dynamics of availability of cloud resources because performance measurement based on immediate feedback cannot reflect the performance traits that are necessary for cloud applications [11]. Long-run benchmarking analysis helps analyze the performance degradation behaviors that take place only when the resources are subject to continuous load conditions and resource throttling by increased temperatures and resource leakage through memory exhaustion. The experiment proves that cloud applications requiring tight latency constraints have challenges in maintaining acceptable levels of performance in shared clouds, especially considering applications requiring sub-millisecond responses, where even minor interference affects performance adversely. Runtime measurements of cloud computing services have been found to have large variability in runtime performance, leading to difficulties in capacity planning, modeling, and comparing different candidate architectures for system design. There have been studies that examine the causes and scales of runtime variability as measured in sample cloud computing services to list the factors that influence variability in application runtime [12].

There is variability due to several factors, such as hardware variability, in which apparently similar machine types co-exist, such that those that seemingly have similar specifications actually have different processor performance, variability due to virtualization, which is dependent upon the pattern of hypervisor scheduling, as well as variability in network runtime due to congestion on shared resources. There is variability across different time scales that indicate either variation over a shorter time scale that depends upon transient resource contentions, as well as variability that is dependent upon changes in data center settings. Methods

10.48047/jocaaa.2026.35.01.38

of analyzing statistics allow for improved models of runtime variability that indicate scaled measurements to evaluate which data points influence measurements and thus should be removed from consideration. Its application to the study illustrates that without considering statistics, measurements are not reliable when systems have small variability in the measurement scales.

Conclusion

The traditional virtual machine provisioning model faces inherent challenges related to the prolonged nature of resource duplication obligations, thereby hindering scale-out agility. Optimizations that combine hypervisor memory deduplication operations with copy-on-write approaches abate the bottlenecks inherent in traditional cloning procedures by allowing transparent page sharing and decoupled resource allocations. Storage resource deduplication implemented by delta disks, together with the forking functionality of virtual machines, harnesses the power of copy-on-write operations, ensuring that the initial resource footprints occupy minimal proportions while maximizing the velocity of deployment. This approach requires critical tradeoffs in the operational deduplication scanning overhead costs against the actual memory savings achieved by adaptive algorithms that focus on fixed memory regions. Architectural harmony of the solution with orchestration platforms, scrupulous tracking of sharing efficiency indicators, and the intricately detailed management of the child-parent lifecycles are critical architectural elements of this technology. This solution, when applied in the context of virtualization, has been observed to deliver revolutionary improvements in the provisioning latency, resource agility, and cost expenditures inherent in the underlying infrastructure. This solution is an invaluable proposition for virtual desktop infrastructure operating in the presence of a large user population, development scenarios that necessitate short iteration cycles, and container workloads that drive demands for transient compute infrastructure, thus meeting the imperatives of responsiveness and scale-out agility that are germane to the latest cloud infrastructure stacks.

References

[1] Christopher Clark et al., "Live Migration of Virtual Machines," NSDI ’05: 2nd Symposium on Networked Systems Design & Implementation. [Online]. Available: https://www.usenix.org/legacy/event/nsdi05/tech/full_papers/clark/clark.pdf

[2] Norman Bobroff et al., "Dynamic Placement of Virtual Machines for Managing SLA Violations," 10th IFIP/IEEE International Symposium on Integrated Network Management, 2007. [Online]. Available: https://ieeexplore.ieee.org/document/4258528

[3] Carl A. Waldspurger, "Memory Resource Management in VMware ESX Server," Proceedings of the 5th Symposium on Operating Systems Design and Implementation, 2002. [Online]. Available: https://www.usenix.org/legacy/event/osdi02/tech/waldspurger/waldspurger.pdf

[4] Diwaker Gupta et al., "Difference Engine: Harnessing Memory Redundancy in Virtual Machines," 8th USENIX Symposium on Operating Systems Design and Implementation. [Online]. Available: https://www.usenix.org/legacy/event/osdi08/tech/full_papers/gupta/gupta.pdf

[5] Michael Vrable, Stefan Savage, and Geoffrey M. Voelker, "Cumulus: Filesystem Backup to the Cloud," 7th USENIX Conference on File and Storage Technologies. [Online]. Available: https://www.usenix.org/legacy/event/fast09/tech/full_papers/vrable/vrable.pdf

[6] Konrad Miller et al., "KSM++: Using I/O-based hints to make memory-deduplication scanners more efficient," 2012. [Online]. Available: https://www.cse.iitb.ac.in/~puru/courses/spring22/downloads/ksm++.pdf

[7] H. Andrés Lagar-Cavilla et al., "SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing," 2009. [Online]. Available: https://www.scs.stanford.edu/~rumble/papers/LagarCavillaEurosys09.pdf

10.48047/jocaaa.2026.35.01.38

[8] Dan Williams et al., "The Xen-Blanket: Virtualize Once, Run Everywhere," 2012. [Online]. Available: https://sites.cs.ucsb.edu/~rich/class/cs293b-cloud/papers/xen-blanket.pdf

[9] Timothy Wood et al., "Memory Buddies: Exploiting Page Sharing for Smart Colocation in Virtualized Data Centers," ACM SIGOPS Operating Systems Review, Volume 43, Issue 3, 2009. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1618525.1618529

[10] Gokul Soundararajan et al., "Dynamic Resource Allocation for Database Servers Running on Virtual Storage." [Online]. Available: https://www.eecg.toronto.edu/~amza/papers/fast09.pdf

[11] Sean Kenneth Barker and Prashant Shenoy, "Empirical Evaluation of Latency-Sensitive Application Performance in the Cloud," MMSys '10: Proceedings of the first annual ACM SIGMM conference on Multimedia systems, 2010. [Online]. Available: https://dl.acm.org/doi/10.1145/1730836.1730842 [12] Jörg Schad et al., "Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance," Proceedings of the VLDB Endowment, Volume 3, Issue 1-2, 2010. [Online]. Available: https://dl.acm.org/doi/10.14778/1920841.1920902