Performance Issues in Cloud Computing: KVM Hypervisor's Cache Modes Evaluation

Borislav Đorđević[1], Nemanja Maček[2], Valentina Timčenko[3]

[1,3] Mihailo Pupin Institute, University of Belgrade, 15 Volgina Street, 11060 Belgrade, Serbia; borislav.djordjevic@pupin.rs, valentina.timcenko@pupin.rs

[2] Department of Computer Technologies, The School of Electrical and Computer Engineering of Applied Studies, Vojvode Stepe 283, 11000 Belgrade, Serbia
e-mail: nmacek@viser.edu.rs

Abstract: This paper examines the performance of bare-metal hypervisors within the context of Quality of Service evaluation for Cloud Computing. Special attention is paid to the Linux KVM hypervisors' different cache modes. The main goal was to define analytical models of all caching modes provided by hypervisor according to the general service time equation. Postmark benchmark software is used for random performance testing with two different workloads, consisting of relatively small objects that simulate a typical mail server. Sequential performance is evaluated with Bonnie++ benchmark software. The experiments were conducted separately with a single virtual machine, and, two and three virtual machines running on the hypervisor. The interpretation of obtained benchmark results according to the proposed models is the main contribution of this research.

Keywords: cloud computing; virtualization; hypervisor; KVM; cache modes

1 Introduction

Cloud computing (CC) is a concept of sharing hardware and software resources that are available on request. CC is based on virtualization, which allows hardware consolidation, provides resource isolation, leads to a higher level of security and reliability of the available IT infrastructure and significantly reduces maintenance costs [1-4]. The CC is closely related to the Quality of Service (QoS), which is a guaranteed level of performance and availability of service provided to users [5].

The hypervisor (virtual machine manager) is a software layer that creates and manages virtual machines. Guest operating systems (OS) are executed on virtual machines. There are the two types of hypervisors. Type-1 (also known as bare-metal or native) hypervisor is executed directly on the physical hardware, while type-2 (also known as hosted) hypervisor runs on the host OS, thus providing the guest OS with lower performance due to overhead produced by the host OS.

Linux Kernel-based Virtual Machine (KVM) [6] is type-1 hypervisor integrated into the Linux OS as a kernel module. This type of implementation allows KVM to follow modern kernel improvements. KVM uses a modified QEMU emulator for the block and network devices [7]. KVM provides a large number of tuneable parameters to the administrators, including three different caching modes, which differ in performance and achievable reliability.

In this paper, KVM hypervisor block device performance and reliability are examined. The obtained results are interpreted according to analytical models of workloads, reading, normal write cycles and flushing or direct writing in different caching modes. We have set up several preliminary research hypotheses on I/O performance, which were validated, along with the model, with the synthetic benchmarking. The main contribution of this paper is the proposed analytical modelling of workload and read/write (R/W) operations in the specific caching virtual environment, which allows us to make recommendations for the optimal KVM cache mode selection in specific situations.

2 Related Work

QoS in CC is a complex concept which includes many factors such as, performance, reliability, availability and security. These factors are mutually dependent, thus making the QoS evaluation a hard task. For example, security in CC is a research area that includes, but is not limited to the following issues: privileged user access, regulatory compliance, data location, data segregation, recovery, defence against the attacks and long-term viability. Availability and recovery are mutually dependent, as well as performance and reliability. Hence, analyzing all the factors would outreach the scope of this research.

The scope of this research is hypervisor performance evaluation as one of fundamental factors of the QoS. There are several different performance evaluation approaches reported in the literature that differ in methodology. For instance, the most common is the comparative performance analysis of Xen, KVM, VMware and OpenVZ hypervisors using different benchmark tools such as Bonnie++, IOzone, LINPACK and LMbench [8-12].

Some recent studies have focused on the emerging problem of fast input/output (I/O) support for a growing number of applications in the CC, thus targeting block device and general I/O performance analysis in virtual environment [9, 13-15]. Thus, it is often seen that the experimental results draw the attention mainly to the impact of performance overheads on the adoption of CC technology. Consequently, it is important to pay attention when making decisions or choices of virtual infrastructure management solutions, as there can be a rather limited capacity to react to changes on demand in stochastic and dynamic environments where not all the choices would be appropriate [9].

Another class of topics arise from the need for resource management optimization, virtual machine migration and resource replicas in CC [16-18]. These issues are highly correlated to the dynamic fluctuation of the system workload, further producing load imbalance, lower utilization and workload hotspots. Some of these approaches enforce the introduction of automatic management of virtualized resources in cloud environments, and the particular control system that would compute the necessary resource allocations for each used application, provide dynamic adjustment and virtual machine rearrangement in the cloud, mainly based on statistical machine learning techniques [17].

Studies that target a highly pervasive need for energy efficiency in the context of performance in CC are presented in [19-20]. These studies provide some fundamental insights on the impact of virtualization on energy usage, consider the energy overhead increase correlation with the increase of physical resources utilization, and also suggest some possibilities of CC server consolidation in data centers for reducing energy cost [19]. Alternatively, an approach for applying appropriate allocation schemes of dynamic requests for virtual servers is proposed for server farms applications [20], and efficient and secure use of system resources [21]. Additionally, CC technology can highly improve different business processes, e.g. in the context of universities where CC can provide a more intense data processing environment and enhance specific I/O quality dimensions [22].

The research presented in this paper belongs to the comparative performance analysis approach group. Although it is partially similar with the research presented in [9], which analyzes caching modes without any in-depth interpretation, all results presented here are interpreted according to the proposed analytical model. Best practices for block I/O performance and recommendations for the selection of appropriate KVM cache mode according to the type of storage that is used are discussed in [23]. For example, writethrough mode is recommended for local or direct-attached storage, as it ensures data integrity and provides acceptable I/O performance, while "none" mode is recommended for remote NFS storage, as it effectively turns all guest I/O operations into direct I/O operations on the host.

3 KVM Cache Modes

The operating system’s page cache improves the disk read/write operation performance. Within the Kernel-based Virtual Machine (KVM) environment, both the host and the guest OS maintain their own page caches. The page cache is copied to a permanent storage using flushing (fsync), while direct I/O requests bypass the page cache. There is also a disk R/W cache, resulting in three independent caches. There are three caching modes available for KVM guest

operating systems – writethrough, write-back and none, resulting in three different write operation options:

- data will be cached in the host-side page cache if the cache mode is set to writeback;
- data will be immediately flushed to the physical disk cache if the cache mode is set to none;
- data will be immediately flushed to the physical disk platters if the cache mode is set to writethrough.

If KVM caching mode is set to write-back, both the host OS page cache and the disk write cache are enabled for the guest. QEMU-KVM interacts with the disk image file with writeback semantics for guest's flushing and direct write cycles: write operations are reported to the guest as completed when the data is placed in the host page cache. The guest's virtual storage controller is expected to send down flush commands. Guest OS application's I/O performance is good, but the data is not protected from power failures. As a result, writeback caching mode is recommended only if the potential data loss is not a major concern.

If KVM caching mode is set to none, the host OS page cache is disabled. QEMU-KVM interacts with the disk image file with writethrough semantics for guest's flushing and direct write cycles; the host page cache is bypassed and I/O is performed directly between the QEMU-KVM buffers and the storage device. Write operations are reported to the guest as completed when the data is placed in the disk R/W cache. The guest's virtual storage controller is expected to send down flush commands. Guest's write performance in this mode is expected to be optimal because the write operations bypass the host OS page cache and go directly to disk R/W cache. However, due to host OS page cache being disabled, the guest's read performance is not as good as in writeback and writethrough modes. This cache mode is suitable for guests with large I/O requirements, and is generally the best choice, as it is the only mode that supports migration.

KVM's writethrough mode enables different caches for reading and writing. Host OS page cache and the disk cache are enabled for the guest's reading operations. QEMU-KVM interacts with the disk image file with write-through semantics for guest's flushing and direct write cycles, and write operations are reported as completed only when the data has been fully committed to the storage device. The guest's virtual storage controller does not need to send down flush commands. Guest's application read performance is good as in the write-back mode, as the host OS page cache is enabled for reading. However, this mode provides lowest writing performance and is prone to scaling problems, because data is written through to the physical storage medium. This cache mode is suitable for systems with small number of guests that have low I/O requirements.

4 Modelling the Workload and Cache Modes

Performance characteristics of each workload are based on times required to complete read and write operations. Both reading and writing can be either random or sequential. Thus, the total workload processing time T_W is given by:

$$T_W = T_{RR} + T_{SR} + T_{RW} + T_{SW}\,, \tag{1}$$

where T_{RR} denotes random read time, T_{SR} sequential read time, T_{RW} random write time and T_{SW} sequential write time. For the specified workload, expected access time for the file system includes five components given by the following equation:

$$T_W = T_{DIR} + T_{META} + T_{FL} + T_{FB} + T_J\,, \tag{2}$$

where T_W is the total time needed to complete all operations on the workload, T_{DIR} the time needed to complete all directory related operations, T_{META} the time needed to complete all metadata operations, T_{FL} the time needed to complete all free lists operations, T_{FB} the time needed to complete direct file blocks operations and T_J the time needed to complete journaling operations.

General service time equation that can be applied to any caching system is:

$$T_{srv} = P_{HIT} \cdot T_{CACHEsrv} + P_{MISS} \cdot T_{CACHEsrv}\,, \tag{3}$$

where $T_{CACHEsrv}$ denotes an effective disk access time with caching functionality (cache hit/miss service time), and P_{HIT} and P_{MISS} denote probabilities of hits and misses in the cache, respectively.

Cache service time is calculated as ratio of request size and cache transfer rate (the throughput of the cache).

4.1 Reading Operations in Different Cache Modes

Most of the timing equations presented here are based on the proper application of general service time equation (3).

Reading cycles in writeback and writethrough, as shown on Figure 1 (left), use all three caches. Application's reading service time is a function of guest OS page cache for the hit cycles, while host OS page cache and disk R/W cache are included in the miss cycles:

$$T_{srvR} \approx P_{HIT\_R\_guest} \cdot T_{srvR\_GuestPageCache} + P_{MISS\_R\_guest} \cdot T_{srvR\_VirtDisk}\,. \tag{4}$$

Virtual disk read service time is the function of host OS page cache for hit cycles, while disk R/W cache is included in the miss cycles:

$$T_{srvR\_VirtDisk} \approx P_{HIT\_R\_host} \cdot T_{srvR\_HostPageCache} + P_{MISS\_R\_host} \cdot T_{srvR\_PhysDisk}\,. \tag{5}$$

Physical disk service time is the function of disk R/W cache for hit cycles, while disk platters are included in the miss cycles:

$$T_{srvR\_PhysDisk} \approx P_{HIT\_R\_disk} \cdot T_{srvR\_DiskCache} + P_{MISS\_R\_host} \cdot T_{srvR\_DiskPlatters}. \quad (6)$$

Throughputs of guest and host OS page cache depend on the operating memory, while disk R/W cache throughput is considerably smaller and depends on the disk interface speed. Disk platters' reading service time is given as a sum of consumed seek time T_{seek}, generated rotational latency $T_{latency}$ and media transfer time T_{media}:

$$T_{srvR\_DiskPlatters} = T_{seek} + T_{latency} + T_{media}. \quad (7)$$

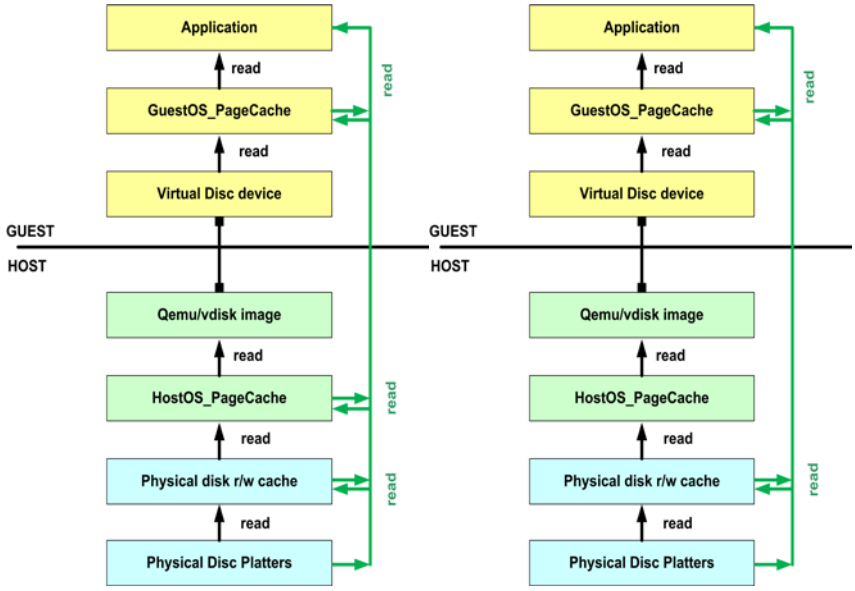

Figure 1

Reading when cache mode is set to writeback or writethrough (left) and set to none (right)

If the KVM cache mode is set to none, only guest OS page cache and disk R/W cache are active for reading operations, while host OS page cache is disabled, as shown in Figure 1 (right). Disabling the host OS page cache degrades reading performance, compared to writeback or writethrough mode.

Application's reading service time depends on guest OS page cache, while disk R/W cache is included into miss cycles, as given in (4). The virtual disk read service time, which is now an equivalent of physical disk service time given by equation (6), is the function of disk R/W for hit cycles, while disk platters are included into the miss cycles.

4.2 Normal Write Operations in all Cache Modes

Normal writing cycles use only guest OS page cache in all cache modes, as shown in Figure 2 (left). Host OS page cache and disk R/W cache are used for miss cycles as the disk block allocation function.

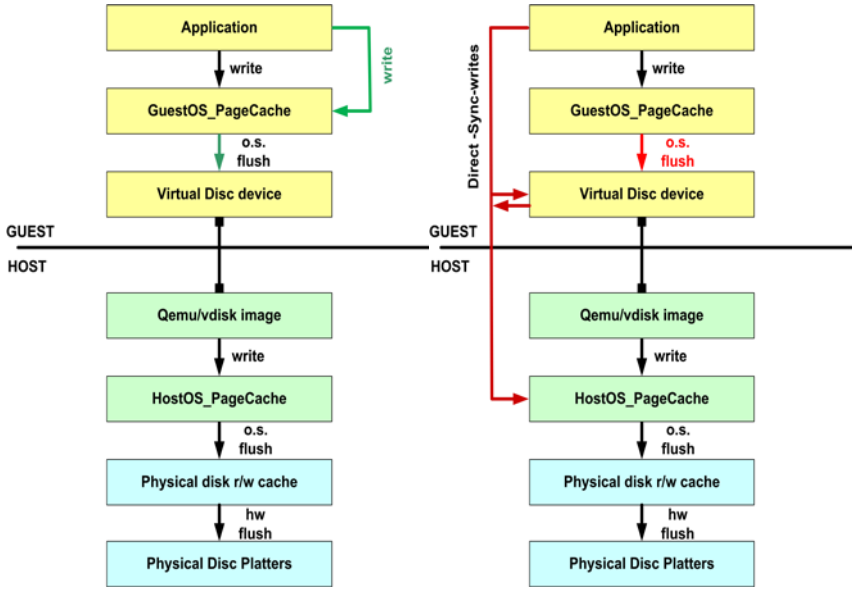

Figure 2
Normal writing operations in all cache modes (left) and flushing or direct-sync writing when cache mode is set to writeback (right)

Application's writing service time is a function of guest flushing time, guest OS page cache for hit cycles, while host OS page cache and disk R/W cache are included into the miss cycles as guest block allocation service time:

$$T_{srvW} \approx P_{HIT\_W\_host} \cdot T_{srvW\_GuestPageCache} + P_{MISS\_W\_guest} \cdot T_{GuestBlockAllocate} + T_{GuestFlush}. \quad (8)$$

If we exclude guest flushing, guest application's writing service time is almost identical to guest OS page cache service time; normal write performance without guest flushing time is almost identical for all three cache modes. However, guest flushing time is different for these modes: in writeback, flushing goes into the host OS page cache, in none caching mode flushing goes into the disk R/W cache, and in the writethrough mode it goes into the disk platters.

4.3 Flushing or Direct-sync Writing in Different Cache Modes

Let the direct write service time denote guest flushing and direct sync write time.

In writeback cache mode, only host OS page cache is active for flushing – direct writing, while disk R/W cache is active for miss cycles as disk block allocation function. Flushing and direct-sync write cycles are shown in the Figure 2 (right). Direct write service time is a function of host OS page cache for hit cycles, while disk R/W cache is defined as host block allocation service time in the miss cycles:

$$T_{srvDW} \approx P_{HIT\_W\_host} \cdot T_{srvW\_HostPageCache} + P_{MISS\_W\_host} \cdot T_{HostBlockAllocate} + T_{HostFlush} . \quad (9)$$

If host flushing is excluded, the guest's direct writing service time is almost identical to host OS page cache service time (very fast).

If cache mode is set to none, only disk R/W cache is active for flushing – direct writing, while disk platters in miss cycles are considered for disk block allocation function. Flushing and direct-sync write cycles are shown in the Figure 3 (left).

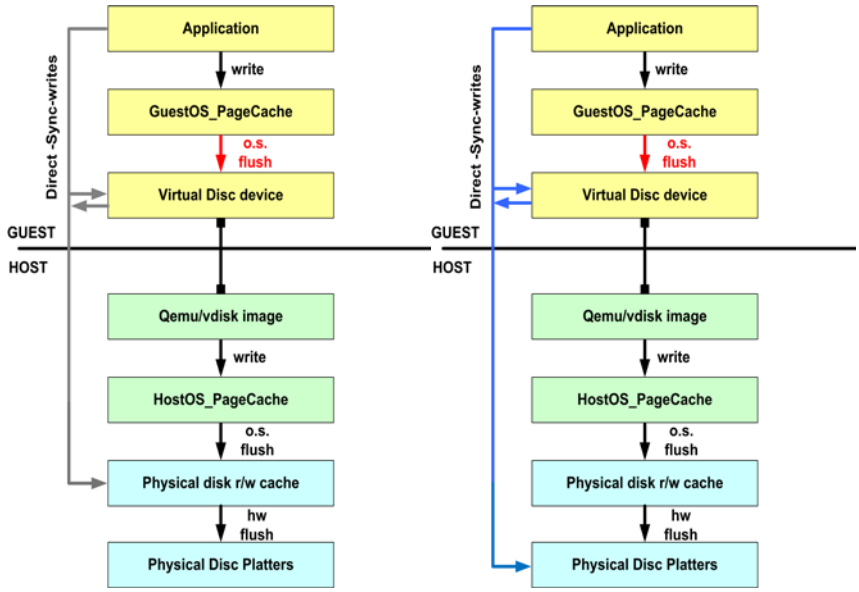

Figure 3

Flushing or direct-sync writing when cache mode is set to none (left) and set to writethrough (right)

Direct write service time is a function of disk R/W cache for hits, while disk platters are included in the miss cycles as disk block allocate service time:

$$T_{srvDW} \approx P_{HIT\_W\_disk} \cdot T_{srvW\_DiskCache} + P_{MISS\_W\_disk} \cdot T_{DiskBlockAllocate} + T_{DiskFlush} . \quad (10)$$

If disk flushing is excluded, the guest's direct writing service time is almost identical to disk R/W cache service time.

If writethrough cache mode is used, each guest flushing or direct write cycle finishes directly to disk platters, as shown in Figure 3 (right). Direct write service time is a function of disk platter service time, and host page cache and disk R/W cache for hits:

$$T_{srvDW} \approx P_{HIT\_W\_host} \cdot T_{srvW\_HostPageCache} + P_{HIT\_W\_disk} \cdot T_{srvW\_DiskCache} + T_{DiskPlatters}. \quad (11)$$

4.4 Hypotheses on KVM I/O Performance

KVM virtual environment provides three levels of caching (guest OS cache, host OS cache and disk R/W cache) and three different cache modes that mainly differ in write semantics and interaction with the host OS cache. Writeback and writethrough modes employ all three levels of caching in writeback and writethrough semantics, respectively, while "none" cache mode bypasses the host OS cache and employs disk R/W cache in writeback semantics. Single or multiple virtual machines running different applications can be executed on a hypervisor.

The following research hypotheses on overall I/O performance, based on the proposed model, are set:

- H1: I/O performance depends on the type of applications running on VMs.
- H2: I/O performance depends on number VMs running.
- H3: Number of VMs running on the hypervisor affects the amount of RAM available to the host OS cache, resulting in overall I/O performance degradation; this is more evident if the hypervisor runs on the system with smaller amount of RAM and large number of VMs running.

According to the presented model, writeback and writethrough cache modes are expected to provide a distinctive reading operations advantage over the "none" cache mode on the basis of equation (5). The preliminary hypotheses on reading operations performance are set as follows:

- H4: Writeback and writethrough are expected to provide remarkably better throughput for workloads with dominant random read components.
- H5: If the dominant component of the workload is the sequential reading, the writethrough and writeback should also provide much better performance under the following conditions: the host page cache is large enough and it relies on applied read ahead technology.
- H6: The impact of equation (5) directly depends on the size of available host page cache and on the usage of the read ahead technologies.

Write operations are correlated to flushing and direct sync cycles and the writeback mode should provide the best results according to equation (9). The preliminary hypotheses on writing operations performance are set as follows:

- H7: If the workloads' dominant components create a large number of random and sequential flushing and direct sync cycles, the usage of writeback caching should provide the system with the best performance.
- H8: Under the same circumstances, the writethrough mode is expected to provide the system with the worst performance, according to equation (11).
- H9: The impact of the equation (9) directly depends on the size of the available host page cache and on the usage of the block-allocation technologies. Also, as with the reading operations, the larger cache is expected to provide the system with better performance.

These hypotheses and the presented model are validated by a set of performance measurements (synthetic benchmarking) and result interpretation presented in next section of the paper.

5 Experiments

We have used Postmark benchmark [24] for hypervisor's random performance testing and Bonnie++ benchmark [25] for hypervisor's sequential performance testing. Postmark simulates Internet mail server workload. It creates a large pool of randomly generated files, performs a set of operations, such as creation, reading, and deletion, and measures the time required to perform these operations. Bonnie++ is a benchmark with the ability to perform several performance tests on the file system, including sequential throughput and CPU overhead monitoring during the test.

Experiments were performed on the dual core Intel Xeon CPU E3110 @ 3.00GHz server with 4GB RAM, 1TB hard disk (7200 rpm, 6Gb/s). Centos_OS_6.5_final is the underlying operating system with the ext4 as the test file system. Centos based Linux Kernel 2.6.32-358.18.1.el6.x86_64 is chosen as a native host for KVM hypervisors. Centos 6.5_final is also the guest operating system.

The test environment with only one virtual machine running is not completely valid representation of a cloud. Thus, we have performed three sets of experiments: one with the single VM running, and two additional tests with two and three VMs running. All the results obtained with one VM running are interpreted according to the analytical model presented in section 4 of this paper. The model is also validated with the results obtained from the experiments with multiple VMs, which are more representative of the CC environment.

5.1 Random and Sequential Performance Testing with Single VM Running

Random performance is measured with two different test sets. Obtained experimental results are given in Table 1 and graphically presented in Figure 4.

The workload specifications for the first test set are: small number of files (4,000) ranging in size from 1 KB to 100 KB, moderate number of create/delete operations, and smaller amount of read/write operations (1.6 GB for reading, 1.8 GB for writing). The workload specifications for the second test set differ in file size ranging in size from 100 KB to 300 KB, and a larger amount of read/write operations (4.6 GB for reading, 5.4 GB for writing).

Table 1
Postmark random performance testing results

		Cache mode		
Workload	Operation	Writeback	Writethrough	None
Test Set 1	Random read	59.77 MB/s	3.98 MB/s	5.41 MB/s
	Random write	69.89 MB/s	4.66 MB/s	6.33 MB/s
Test Set 2	Random read	26.83 MB/s	7.36 MB/s	8.57 MB/s
	Random write	31.15 MB/s	8.55 MB/s	9.94 MB/s

Figure 4
Random performance testing results

The writeback mode provides the best performance on both workloads. The results obtained from this test set indicate that random read and write components are dominant in (1), while the direct file block component is dominant in (2). The second test set enforces the workaround of sequential components. Writeback mode remarkably outperforms the writethrough mode. Although read (4-7) and normal write operation performances (8) are almost identical, as both modes employ host and guest OS page cache, there is a huge throughput difference due to the flushing and direct write cycles. Writeback mode uses the host OS page cache

with the writeback semantics given by (9) which is remarkably faster than writethrough synchronous writing mode (11). Big throughput differences indicate that the flushing and direct write cycles are very intensive, with the dominant random components of the workload; throughput difference decreases with the second test set, which increases the sequential component of workload.

Writeback mode remarkably outperforms the hypervisor with cache mode set to none as well. Although normal write operation performances (8) are almost identical, read cycles and flushing and direct write cycles provide huge throughput difference. Writeback mode's usage of host OS page cache with writeback semantics (9) provides faster write cycles than disk R/W cache with writeback semantics used if cache mode is set to none (10). Sequential components of workload are more pronounced in the second test set, resulting in smaller differences in read performance and flushing and direct write performance related to first test set.

The performance of hypervisor with the cache mode set to none is slightly better than the writethrough cache. Although normal write operation performances (8) are almost identical, read cycles and flushing and direct write cycles provide minor throughput difference. There are two reasons for the throughput differences. Writethrough read cycles through host OS page cache (5-8) are much faster than reading cycles without any cache. Flushing and direct write cycles using disk R/W cache with writeback semantics (10) are remarkably faster than writethrough synchronous writing mode (11). To conclude, writethrough mode reads data faster, data is written faster if cache mode is set to none. Differences in performance decrease if the workload with stronger sequential components is used.

Bonnie++ sequential throughput test is used to measure sequential writing, reading and rewriting performance of different KVM cache modes. Obtained experimental results are given in Table 2 and graphically presented in Figure 5.

The throughput differences for sequential writes providing writeback's superior performance appear as the result from formulas (9-11) for flushing and direct write cycles.

Rewrite operation reads the contents of the file, deletes the contents and writes new data into the file. Writeback's top performance results from flushing and direct write cycles. Throughput differences between writeback and writethrough modes in rewriting operations are decreased due to read cycles, as both cache modes use host OS page cache (4-7). Writeback mode outperforms the hypervisor's none cache mode, due to the flushing and direct write cycles (10-11) and due to the read cycles, by using host OS page cache.

The writeback slightly outperforms writethrough mode in sequential reading operations, as they both use host OS page cache. If caching mode is set to none, host OS page cache is not used resulting in 6 times lower throughput.

Table 2
Bonnie++ sequential performance testing results

	Cache mode		
Operation	Writeback	Writethrough	None
Sequential write	97.585 MB/s	37.797 MB/s	85.681 MB/s
Sequential rewrite	42.722 MB/s	26.006 MB/s	36.366 MB/s
Sequential read	618.881 MB/s	584.503 MB/s	98.082 MB/s

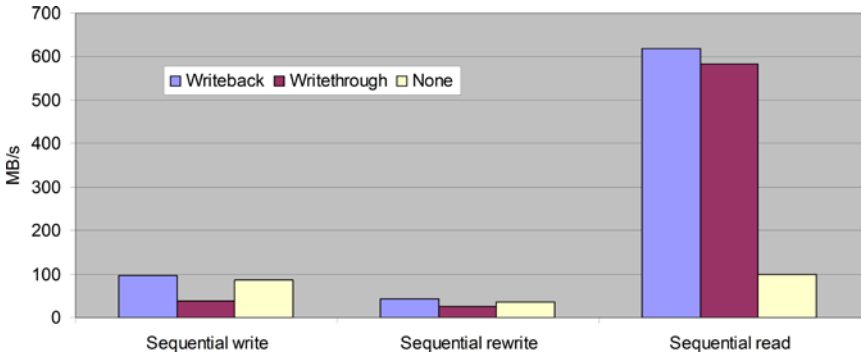

Figure 5
Sequential performance testing results

5.2 Validation of the Model and Hypotheses with Single VM Running

Regarding validation of the presented model for different caching modes, the experiments with one VM running have provided the results that were expected.

It is evident that with a single virtual machine running, the host operating system has more RAM than when multiple virtual machines are running. This results in the largest host page cache, i.e. the greatest impact to read and write operation performances, given by equations (5) and (9), respectively.

For the group of random tests, the performance of the writeback mode is, according to semantics given by equation (9), much better than the performance of writethrough mode, given by equation (11). The reading in writeback and writethrough, performed according to equation (5), is faster than reading when caching mode set to none. As the host page cache is larger than disk R/W cache, writeback also writes faster – which validates equations (9) and (10). Writethrough also provides better read performance than cache mode set to none according to (5), but significantly lower write performances, which validates equations (10) and (11).

According to the group of sequential tests:

- The results of the sequential write tests have shown that writeback caching mode is superior, resulting from (9), while according to (11) the writethrough is the worst option.
- Sequential rewrite operation combines sequential read and write operations on the same blocks. The performances of systems with the caching modes set to none and writeback mode are similar. This results from a big positive impact provided by disk R/W cache (10). Writethrough is the worst option again (11).
- With the largest possible host page cache size, the writeback and writethrough modes provide much better read performances (5) when compared to caching mode set to none.

5.3 Experiments and Validation with Multiple VMs Running

The next set of the experiments is carried out with multiple VMs running on the same hypervisor. The same Postmark workload is used for the two and three VMs running scenarios, and average values were used for result interpretation.

The results of random read and random write experiments with two and three VMs are given in Tables 3 and 4, respectively.

Table 3

Postmark random performance testing results (two VMs running)

		Cache mode		
Workload	Operation	Writeback	Writethrough	None
Test Set 1	Random read	21.15 MB/s	1.76 MB/s	2.28 MB/s
	Random write	24.73 MB/s	2.06 MB/s	2.67 MB/s
Test Set 2	Random read	4.75 MB/s	2.60 MB/s	3.26 MB/s
	Random write	5.51 MB/s	3.02 MB/s	3.78 MB/s

Table 4

Postmark random performance testing results (three VMs running)

		Cache mode		
Workload	Operation	Writeback	Writethrough	None
Test Set 1	Random read	8.20 MB/s	1.08 MB/s	1.89 MB/s
	Random write	9.59 MB/s	1.26 MB/s	2.21 MB/s
Test Set 2	Random read	3.09 MB/s	1.63 MB/s	2.57 MB/s
	Random write	3.59 MB/s	1.89 MB/s	2.99 MB/s

According to random test results, the performance of the virtual machine used for measurement decreases with each new VM added to the hypervisor, as each VM added to the system consumes a part of the available RAM (1 GB in our case).

Thus, there is less memory available to host OS, resulting in smaller host page cache. This has a direct impact on the equations (5) and (9): reading performances are decreased according to equation (5), and writing performances are decreased according to (9).

The results obtained from the first test set indicate that writeback caching mode slightly decreases its superior performance with the increased number of VMs running. However, it is still the very best option for the test set 1. Results of this test are in accordance with the expected ones, thus validating the analytical model properly.

Writeback caching mode outperforms the writethrough mode, mostly due to writeback semantics given with equation (9) that depends directly on the host page cache size, unlike writethrough semantic given with the equation (11). This difference decreases with each virtual machine added to the system, as each addition provides a negative impact on writeback performance, which is, as expected, in accordance with the equation (9).

Writeback provides faster reading operations if compared to the caching mode set to none, which is in accordance with (5) and mostly depends on the host page cache size. Writeback also provides better write operations performance, as given with equation (9) and (10). The increased number of VMs running decreases the performance difference between aforementioned modes, as equation (5) has weaker impact on writeback performance.

Writethrough provides the system with better reading operation performance than the system with caching mode set to none according to equation (5), which highly depends on host page cache size. But, when the write operations are analyzed, the so called "none" caching mode is outperforming writethrough, and this originates from differences expressed by equations (10) and (11). With the increasing the number of VMs running, this difference becomes even more evident. The explanation relies on application of the equation (5), as it has weaker effects with the writethrough, and in those circumstances the "none" mode overtakes the precedence in performance analysis.

According to the results obtained from the test set 2, the writeback mode loses some of its superiority with the increased number of VMs running, while system with caching mode set to none still outperforms writethrough mode. The explanation of this behaviour is similar to the one provided for the test set 1 – smaller amount of host page cache used for reading and writing.

The results of sequential read, write and rewrite operations obtained from Bonnie++ with two and three VMs are given in Tables 5 and 6, respectively.

According to the obtained test results, the performance decreases with each new VM added to the hypervisor.

Table 5
Bonnie++ sequential performance testing results (two VMs running)

	Cache mode		
Operation	Writeback	Writethrough	None
Sequential write	37.98 MB/s	22.58 MB/s	32.21 MB/
Sequential rewrite	10.62 MB/s	6.77 MB/s	13.96 MB/s
Sequential read	36.45 MB/s	32.35 MB/s	28.08 MB/s

Table 6
Bonnie++ sequential performance testing results (three VMs running)

	Cache mode		
Operation	Writeback	Writethrough	None
Sequential write	31.19 MB/s	14.64 MB/s	21.09 MB/s
Sequential rewrite	6.79 MB/s	4.39 MB/s	7.24 MB/s
Sequential read	23.90 MB/s	21.48 MB/s	20.40 MB/s

The results of the sequential write tests have shown that writeback caching mode dominates over the other two modes, for both scenarios with two and three VMs running. The explanation of such behaviour relies on equation (9), and it's comparison to equations (10) and (11), which is in accordance with the presented analytical model.

The best results in sequential rewrite operations testing are achieved when caching mode is set to none. When testing the rewriting operation, the overall effect of reduced host page cache from equations (5) and (9) influences in such a way that the "none" mode (which does not utilize the host page caching, but still utilizes disk R/W cache) provides better results than writeback mode. That behaviour was detected for the sequence that covers: sequential block reading, editing, and writing. Writethrough mode provides remarkable lower performance when compared to other caching modes.

The results of sequential read operations test indicate that, for both two and three VMs running, writeback and writethrough cache modes still provide higher I/O throughput. Thus, the presented model is completely validated for sequential read operations. When testing is performed with only one virtual machine, the available host page cache is large, thus writeback and writethrough modes significantly outperform the system with caching mode set to none, according to equation (5). This difference is decreased with each virtual machine added into the testing system, and in those circumstances the equation (5) has a weaker influence on the writeback and writethrough modes performances. With sufficient number of virtual machines, host page cache available to each virtual machine would be reduced and the cache mode performances would probably not differ that much.

Conclusions

While comparing the results obtained from testing the system with the one, two and three virtual machines running, we have detected the biggest performance difference between different cache modes employed in the single virtual machine scenario. This difference results from the hosts OS provided with largest amount of RAM. With the introduction of two or three VM, the performances of all cache modes were reduced, as well as the difference between them. The writeback mode employs three different caches in the writeback semantics resulting in superior performance. The best performances are achieved for random read/write workloads, as for sequential workloads when the amount of host OS page cache is large and the read ahead technique is dominating. However, writeback can endanger data integrity in case of power surges. Writethrough mode also employs three cache types with all caching outside of the virtual machines working with the writethrough semantics. Although data integrity is ensured, writethrough performs poorly, especially with the random workload when flushing and direct write cycles are dominant. Hypervisor's none cache mode, which employs disk R/W cache only outside of virtual machine, is faster than writethrough and slower than writeback mode, but does not provide total data integrity.

The results indicate that the amount of RAM has a huge impact on the performance, as it directly affects the host OS page cache size. According to the results, the writeback cache mode provides the system with the best performance if a small number of VMs is running. Each VM added to the system with fixed RAM size results in overall performance degradation, regardless of the employed cache mode, and decreased performance difference between the systems that employ "none" and writeback cache modes. Due to decreased impact of host OS cache it is expected that this performance difference between "none" and writeback modes will become almost negligible when a large number of VMs (10 or more VMs) is running. According to that, we strongly recommend the usage of writeback mode on hypervisors running a small number of VMs and "none" cache mode on hypervisors running a large number of VMs. It should be noted that that the amount of RAM available to the host directly affects the performance differences between systems that run large numbers of VMs – the larger amount of RAM available to the host results in bigger performance difference between systems that employ different cache modes.

Experimental results are not surprising and they are not contradictory to our preliminary hypotheses. The exception that was not expected is the rewrite operation results. This indicates that in some rare situations I/O performance does not benefit that much from the cache.

Further research will include KVM cache mode examination while running large number of virtual machines, as well as the impact of disk schedulers, I/O modes and virtual disk image formats towards virtualization performance.

Acknowledgement

This paper has been partially financed by Serbian Ministry of Education, Science and Technical Development (Development Projects III 43002, TR 32037 and TR 32025).

References

[1] K. Xiong, H. Perros: Service Performance and Analysis in Cloud Computing. In SERVICES 2009, 5th 2009 World Congress on Services, Bangalore, India, 2009, pp. 693-700

[2] J. G. Hansen, E. Jul, Lithium: Virtual Machine Storage for the Cloud. In 2010 SoCC'10: Proc. 1st ACMSymp. Cloud Comput., ACM Press, NewYork, USA, 2010, pp. 15-26

[3] D.-J. Kang, C.-Y. Kim, K.-H. Kim, S.-I. Jung: Proportional Disk I/O Bandwidth Management for Server Virtualization Environment. In 2008 Int. Conf. Comput. Sci. Inf. Technol., Piscataway, NJ, USA, 2008, pp. 647-653

[4] J. Nakajima, K. M. Asit: Hybrid Virtualization—Enhanced Virtualization for Linux. In 2007 Proc. Linux Symp, 2007

[5] T. Imada, M. Sato, and H. Kimura: Power and QoS Performance Characteristics of Virtualized Servers. In Proc. 2009 10th IEEE/ACM Int. Conf. Grid Computing (GRID), Piscataway, NJ, USA, 2009, pp. 232-240

[6] KVM, Kernel-based Virtual Machine. http://www.linuxkvm.org

[7] QEMU, Open Source Processor Emulation. http://www.qemu.org

[8] T. Deshane, Z. Shepherd, J. Matthews, M. BenYehuda, A. Shah, B. Rao: Quantitative Comparison of Xen and KVM. 2008 Xen Summit, Berkeley, CA, USA, USENIX Association, 2008

[9] D. Armstrong, K. Djemame: Performance Issues in Clouds: An Evaluation of Virtual Image Propagation and I/O Paravirtualization. The Computer Journal, Vol. 54, No. 6, 2011, pp. 836-849

[10] P. Padala, X. Zhu, Z. Wang, S. Singhal, K. G. Shin: Performance Evaluation of Virtualization Technologies for Server Consolidation. Tech. Report, HP Labs, USA, 2008

[11] X. Xu, F. Zhou, J. Wan and Y. Jiang: Quantifying Performance Properties of Virtual Machine. In 2008 Linux; Program Testing; Software Performance Evaluation; Systems Analysis; Virtual Machines, Vol. 1, Piscataway, NJ, USA, 2008, pp. 24-28

[12] J. Che, Q. He, Q. Gao, D. Huang: Performance Measuring and Comparing of Virtual Machine Monitors. In 2008 IEEE/IFIP 5th Int. Conf. Embedded and Ubiquitous Computing. EUC2008, Vol. 2, Piscataway, NJ, USA, 2008, pp. 381-386

[13] S. Y. Liang, X. Lu: An Efficient Disk I/O Characteristics Collection Method Based on Virtual Machine Technology, In 2008 Proc. 10th IEEE Int. Conf. High Performance Computing and Commun., HPCC2008, Dalian, China, 2008, pp. 943-949

[14] Y. Dong, J. Dai, Z. Huang, H. Guan, K. Tian, Y. Jiang: Towards High-quality I/O Virtualization. In 2009 ACM Int. Conf. Proc. Series, Haifa, Israel, 2009, pp. 12-22

[15] X. Liao, H. Jin, J. Yu, D. Li: A Performance Optimization Mechanism for SSD in Virtualized Environment. The Computer Journal, Vol. 56, No. 8, 2013, pp. 992-1000

[16] A. Sallam, K. Li: A Multi-objective Virtual Machine Migration Policy in Cloud Systems. The Computer Journal, Vol. 57, No. 2, 2014, pp. 195-204

[17] Q. Li, Q. Hao, L. Xiao, Z. Li: An Integrated Approach to Automatic Management of Virtualized Resources in Cloud Environments. The Computer Journal, Vol. 54, No. 6, 2011, pp. 905-919

[18] W. Zhao, P. M. Melliar-Smith, and L. E. Moser: Low Latency Fault Tolerance System. The Computer Journal, Vol. 56, No. 6, 2013, pp. 716-740

[19] Y. Jin, Y. Wen, Q. Chen and Z. Zhu: An Empirical Investigation of the Impact of Server Virtualization on Energy Efficiency for Green Data Center. The Computer Journal, Vol. 56, No. 8, 2013, pp. 977-990

[20] T. V. Do: Comparison of Allocation Schemes for Virtual Machines in Energy-Aware Server Farms. The Computer Journal, Vol. 54, No. 11, 2011, pp. 1790-1797

[21] L. Vokorokos, A. Baláž, N. Ádám: Secure Web Server System Resources Utilization. Acta Polytechnica Hungarica, Vol. 12, No. 2, 2015, pp. 5-19

[22] I. Petkovics, P. Tumbas, P. Matković, Z. Baracskai: Cloud Computing Support to University Business Processes in External Collaboration. Acta Polytechnica Hungarica, Vol. 11, No. 3, 2014, pp. 181-200

[23] Kernel Virtual Machines (KVM): Best Practices for KVM (second edition). IBM Corporation, 2012

[24] J. Katcher: PostMark: A New File System Benchmark, Technical Report TR3022. Network Appliance Inc, 1997

[25] Bonnie++ Benchmark Suite. http://www.coker.com.au/bonnie++/