University of Amsterdam

System and Network Engineering Master

Research Project

Performance measurement and tuning of remote acquisition

Author:
Łukasz Makowski
lukasz.makowski@os3.nl

Supervisor:
Ir. Ruud Schramp

Thursday 14th April, 2016

Abstract

The paper introduces the concept of forensic triage and the novel method of its application — the acquisition of iSCSI attached storage over Wide Area Network (WAN) link. Next, in the created lab environment, it tests the potential methods of I/O performance improvement (prefetching, parallelism). The gathered metrics show that for the links with Round-Trip Time (RTT) introduced, performance degraded. Finally, the author discusses flaws in the established method and argues about the future work in this area.

1 Introduction

Digital forensics is a field focusing on the several aspects leading to providing the evidence of criminal activities on the basis of analysed digital media. Palmer [1] extends this term, dividing it into the methods handling preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence.

1.1 Acquisition

The acquisition in the means of digital forensics is creating a data image that is unambiguously equivalent to the original one. Often this includes the checksums creation; if required these can be used to cross-validate the authenticity of created copy. Next, the acquired image is handed to an investigator who analyses it for potential evidence. This stage utilises various tools and methodologies depending on data. However, it can be generalised that all of those aim to help in accessing and extracting potentially relevant information. The examples can be the deleted files, internet browsing history or the data hidden in an unallocated media structures. By its means, the forensic analysis is a creative and non-trivial process requiring well-trained and experienced staff. This has already been identified as a serious bottleneck in the digital forensics process chain.

1.2 Forensic triage

One of the remedies to that problem is implanting the concept of triage into the forensics domain. As specified by Garfinkel [2], the triage is a method of prioritising work based on the assignment, rather than its content. In digital forensics, this means that before starting the actual analysis, the acquired images should be assessed and assigned with priority. Taking into account that the size and electronic data devices equipped with integrated storage drastically increases [3], the triage could quickly assess large groups of data, pointing to the most interesting object which should be examined in detail. To realize this idea, a

forensic triage concept assumes the development of tools that can quickly report data the media handles e.g. image files, credit card numbers. Next, based on that information, further prioritisation can be performed.

1.3 Challenges of a remote acquisition

In the age of widespread Internet connections, it is often necessary to correlate evidence gathered in several geographically spread locations occurs. With the defined approach, the investigator needs either to visit every location of analysed data, or all the evidence must be delivered to him. However, it would be more efficient if the data could be assessed remotely (with triage). This could help not only with prioritising the investigation, but it gives a notion whether a particular image is a perspective for a current case. Perhaps the simplest solution is transferring the images over WAN links to the central location handling the analysis. However, when including the factors like WAN bandwidth limitations and storage size often exceeding terabytes, this can be seen as inefficient. Especially where getting acquainted with a *remote* hard drive is urgent, it is not possible to acquire all data in a reasonable period.

1.4 Remote triage concept

One of the possible solutions is creating Internet Small Computer Systems Interface (iSCSI) connections from the distributed images to a central location where an investigator resides. This allows the investigator to interact with the data as it would be locally attached. This technique has its disadvantages; iSCSI protocol performance degrades significantly when the link experiences packet loss or large RTT. As iSCSI uses Transport Control Protocol (TCP) for data transmission, tuning both TCP and iSCSI stacks parameters can probably help to improve on the achieved throughput. The usage of Input/Output (I/O) optimisation methods such as caching, parallelism or scheduling might enhance the performance as well.

Assuming the goal is to gain as efficient data transfer rate as possible, the research will focus on determining whether it is possible to apply I/O optimization techniques to the previously defined pattern and what the effect of link delay between the iSCSI initiator and server will be on the triage process read performance.

2 Related work

Garfinkel [2] describes the concept of triage and proposes applying it to forensics. The introduced approach is a block level data analysis, where the file system metadata is ignored. Instead, the focus is put on the sequential parsing of

raw disk data, searching for matching patterns (e.g. email addresses credit card numbers). The author describes the design of his proof of concept tool - `bulk_extractor`, simultaneously characterising and benchmarking the multi-core processing. Finally to showcase the concept usefulness, two real world cases are presented. `Bulk_extractor` could provide meaningful results after a few hours of execution, unambiguously indicating the information stored on the analysed media.

Roussev [4] discusses the current processing performance of acquisition tools. He argues that the developers could have taken a more significant effort to utilise that processing of the available hardware. His paper defines universal objectives which should be the objectives during the development process. These are briefly: parallelism, acquisition and processing phases concurrency and the immediate presentation of partial results. Later, the so-called Low Latency (LL) requirement for the triage is formulated. Minimisation of the time between the query and response is pointed as a crucial requirement for any forensic tools. The paper aims to quantitatively characterise commonly used acquisition techniques, further introducing a concept of latency-optimized target acquisition (LOTA). The author claims such a framework has the capability to fully utilise any analysed disk throughput.

VanDeBogart et al. [5] propose a new approach to the application-directed access. The solution enables an application to submit prefetch requests to the filesystem layer, placing the data in the cache beforehand. The research introduces an aggressive fetching scheme, where the data is read from a disk subsystem until the system's memory is not significantly utilised.

In the paper by Cao et al. [6] the design of the Application-Controlled File System (ACFS) is explained. It discusses techniques that can lead to better disk performance. Apart from the application-directed access, it is proposed to use prefetching combined with caching and disk scheduling. The authors emphasize that, in the single process case, the disk scheduling has point only for an asynchronous I/O or when the prefetching requests (not the data access itself) are reordered, whereas in a multiprocess scenario, the choice and tuning of the kernel I/O allocation policy has the most significance.

Research papers of Zhang et al. [7] and Oguchi et al. [8] focus on the improvement of sequential write performance to the iSCSI block device. Both papers provide detailed tuning information on the TCP protocol, open-iSCSI initiator, and iSCSI Enterprise Target (IET) software. The tests are performed for the links with emulated network delay. Although partially overlapping, the papers provide unique tuning advice that leads to an overall performance gain.

Yamaguchi et al. [9] analyse the model of iSCSI Short Block Access, as real-world examples of such I/O, file or database access patterns are given. Researchers compare several iSCSI implementations presenting their read performance for 1 and 8KB blocks. The conclusion states that for iSCSI software, knowledge about underlying layers characteristics (TCP, Ethernet) is crucial to reach optimal

throughput.

3 Method

3.1 Lab environment setup

The goal was to simulate WAN link characteristics between two endpoints. For this purpose, these units were defined and setup (Figure 1):

- iSCSI client
- iSCSI server
- WAN emulating bridge
- sample data image

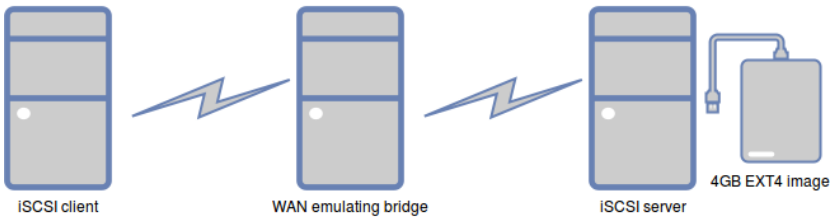


Figure 1: Topology scheme

Next, iSCSI software has been deployed and configured so the test sample image was provided by the server and remotely available at the client side.

3.1.1 Hardware

The lab environment has been built out of the three components: iSCSI client, server, and network emulator. Each was implemented on a separate Dell Precision Tower 5810 workstation, equipped with the Intel Xeon E5-1650 CPU (6 cores, 2 threads each) and 32GB of RAM.

3.1.1.1 Storage

Each workstation had a 512GB Solid State Drive (SSD) drive. These were used for the purpose of Operating System (OS) installation. Additionally, in case of WAN emulating bridge also as a storage holding dumped network traffic. Moreover, a 3TB conventional Hard Disk Drive (HDD) was installed on the iSCSI server, which is where the sample data image was placed. The motivation

Component	Software
iSCSI client	Open-iSCSI client [11]
iSCSI server	iSCSI Enterprise Target (IET) [12]
WAN emulating bridge	netem [13]

Table 1: Core component software

to use a HDD for this purpose was that in the author's opinion, this technology is still dominating in large storage setups. The sample acquired data was 4GB EXT4 Linux filesystem image, originally being the part of Lansing Information Systems Security Association [10] 2013 forensic challenge.

3.1.1.2 Network

The workstations were using its standard GigaEthernet (GE) interfaces. However, the WAN emulating bridge has been installed with an additional one. Next, using Linux `brctl` utility, a software bridge interconnecting iSCSI client and server was created.

3.1.2 Software

The operating system used was Ubuntu Server 14.04 LTS. The iSCSI and WAN emulating software used in the research (Table 1) was determined based on the selected iSCSI related papers [7] [8].

3.2 I/O optimisation - methods of choice

3.2.1 Prefetching

The technique of prefetching is commonly used by OS to improve the storage devices performance from the user/application point of view. From the high-level perspective it can be described as a solution converting multiple read operations into a single, substantial one. The internals of this solution can be observed based on the Linux OS example (Figure 2). On the kernel level, a small read request (red) is being intercepted and converted to a single sequential and asynchronous read-ahead reads (grey). As it populates the OS page cache (green), this enables any future reads hitting within this space to be served significantly faster.

The research uses two prefetching based approaches — read-ahead and read-behind. The difference between these two is how the prefetching offset is determined. The first one prefetches the data following application requested block, whereas the latter aims to populate the cache with blocks placed behind it (Figure 3).

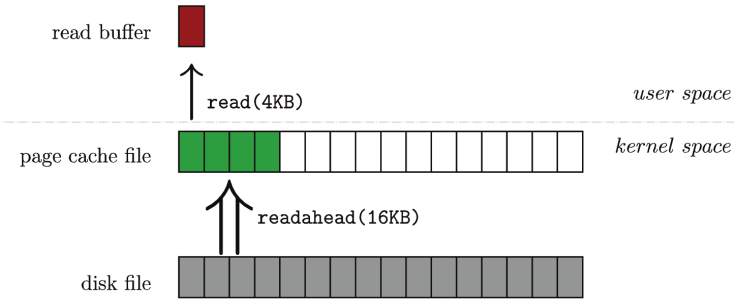


Figure 2: Read-ahead in Linux [14]

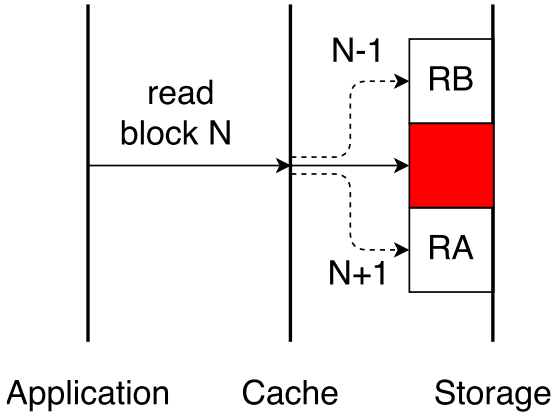


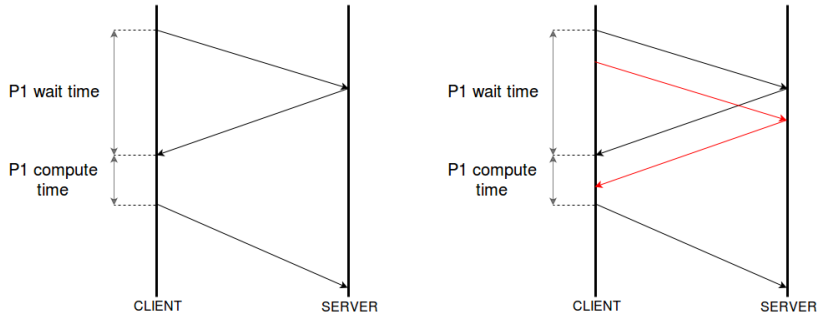
Figure 3: Read-ahead (RA) and read-behind (RB) comparison

3.2.1.1 Fusecoraw

To test the behaviour of applications using read-ahead and read-before methods, fusecoraw [15] proof of concept tool has been employed. It implements the client side read and write caching scheme over the Linux File System in User Space (FUSE). Modifications were made to include simplified implementation of *read-ahead* and *read-behind* operations. These are `pread64()` calls following initial read request inside single upper-level FUSE implementable `read()` function. The usage of FUSE allowed the seamless integration of prefetching with the test triage application. The acquisition process was performed on the fusecoraw provided pseudo-device instead of the raw iSCSI disk.

3.2.2 Parallelism

In the times when the workloads tend to be scaled rather horizontally than vertically, introducing the concurrency to an application seems almost indispensable. Figure 4 presents the typical area when the performance improvement may be



(a) Single process workload

(b) Second process introduced

Figure 4: Benefits of concurrency

achieved. Assuming the process is synchronous, after submitting a request to the server it will be passively waiting for a response (Figure 4a). As the figure 4b shows, this waiting period may be utilised by another process.

Although the advantages of presented pattern are straightforward, it can not be universally used for every workload. For example, it is practically infeasible to try to blindly saturate a single resource (i.e. iSCSI disk) with tens of processes. However, a carefully picked number of concurrent workers may have the potential to deliver beneficial results.

3.2.2.1 Triage.py

The decision to create a tool trying to mimic the behaviour of the triage system has been made. The assumed requirements were the adjustable worker threads number and a file-based scanning approach, as defined by [2]. It was essential to read a block device's content with no modification to its data. The solution used by The SleuthKit (TSK) to cope with this problem is the filesystem driver implementation with the overall omission of write operations. Ultimately, in the research developed `triage.py` [16] script, TSK tools (`fls, icat`) were parallelised using the Python programming language.

As depicted in figure 5, `triage.py` runs two types of threads, which communicate through a global queue. The scanning worker traverses the directory and returns its content as (filename, inode) tuples. The file worker dequeues this information, and if the name matches a predefined regular expression, it attempts to gather the actual file.

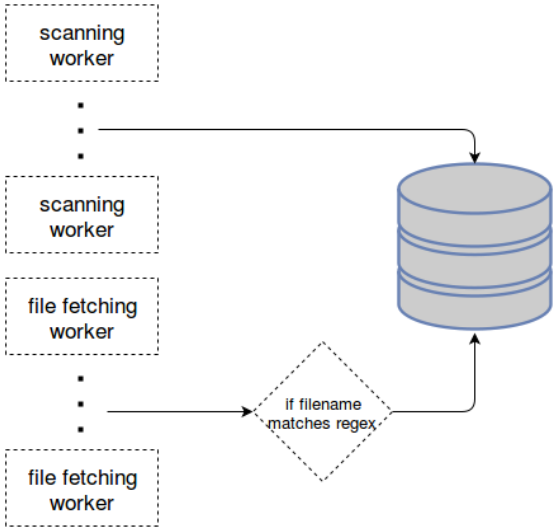


Figure 5: triage.py architecture overview

3.3 Tests scenarios

3.3.1 Prefetching

read-behind [B] / read-ahead [B]	0	8192	65536
0	X	X	X
8192	X	X	-
65536	X	-	X

Table 2: Chosen read-ahead and read-behind values

The prefetching test set (Table 2) parameters were selected as the multiplicity of `fusecoraw` default block size — 8192 bytes.

3.3.2 Parallelism

`Triage.py` can theoretically use any chosen value determining the workers count. For the parallelism test it has been limited, as specified in Table 3.

directory scanner \ file fetcher	1	2	4
1	X	-	-
2	-	X	-
4	-	-	X

Table 3: triage.py workers setup

3.3.3 Combining I/O optimisation and RTT WAN parameters

One of the crucial objectives of this research was to test the techniques defined in sections 3.2.1, 3.2.2 over the WAN link. A simplified model was introduced, where only the link RTT was emulated.

Given the I/O optimisation methods defined, each tests specified in tables 2, 3 was conducted for three network RTT parameters (Table 4). The delay was *added* to the native link RTT between the iSCSI client and server, which, considering this research, was assumed negligible.

RTT [ms] \ test type	prefetching	parallelism	repetitions
0	X	X	3
10	X	X	3
20	X	X	3

Table 4: Test sets summary

3.3.4 Pre-test procedure

To assure the correctness and repeatability of measurements, the procedure (Table 5) was implemented before every test.

Step	**Description**
1	The iSCSI initiator process was restarted
2	The OS block device read-ahead size was set to 0
3	The OS pagecache (including cached dentries and inodes) was flushed
4	The fusecoraw mountpoint was recreated and its cache flushed (prefetching test only)

Table 5: Pre-test procedure

3.4 Metrics

To depict the acquisition process performance, two separate metrics were gathered.

3.4.1 Average throughput

First, the interest was in how the TCP connection used by the iSCSI process will behave under different loads. One of the most straightforward characteristics was chosen - an average throughput. The tool used to analyse dumped packet capture (PCAP) files was `tcptrace` [17], which was one of its standard metric presents *throughput* parameter. In its essence, as per the tool's manual [18] it is the *total bytes of data sent excluding retransmitted bytes and any bytes sent doing window probing*, divided by the time difference between the first and last packet of the analysed connection.

3.4.2 Elapsed time

The second measured metric was the execution time of `triage.py` tool. GNU `time` command provided *elapsed time* metric was chosen as the one accurately representing the time spent by the CPU to execute user space, and kernel level code runtime. Effectively, the elapsed time presents the time period the user must wait to acquire a final triage process results.

4 Results

4.1 Prefetching

4.1.1 Throughput

The conducted measurements present that, despite the prefetching setting used (Table 2), with the increase of link delay an average connection throughput decreases (Figure 6).

With RTT 0, it can be seen that all variants of 8192 byte block performed slightly better than the reference test. However, RTT 10 and 20 acquisition with prefetching disabled outperforms the rest.

4.1.2 Execution time

Figure 7 depicts that the delay growth combined with any tested prefetching setting leads to the total execution time being longer than with no optimisation applied.

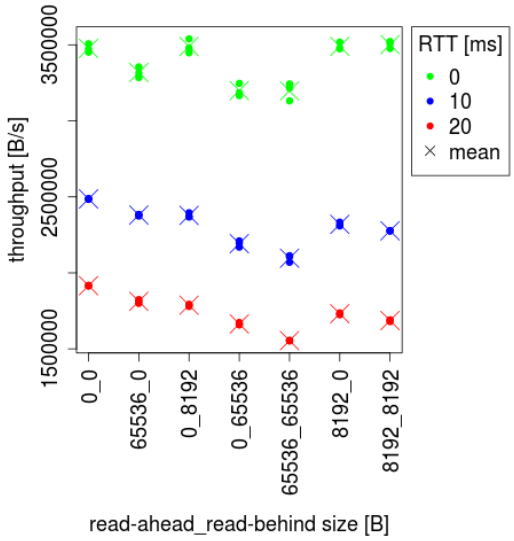


Figure 6: Prefetching, average throughput (higher is better)

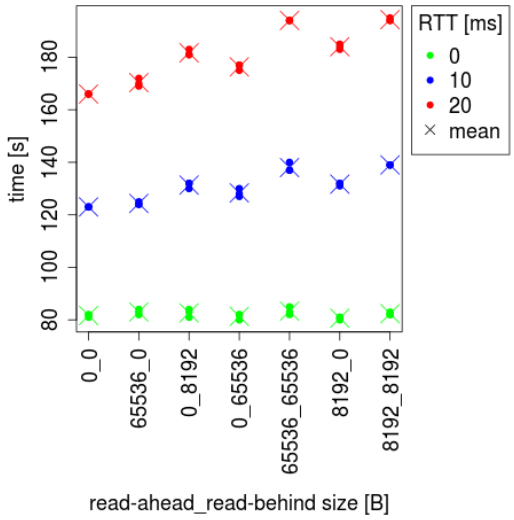


Figure 7: Prefetching, elapsed time (lower is better)

For the experiments with the RTT introduced the prefetching lead to the degradation of execution time comparing to RTT 0 case. In the RTT 0 example, read-ahead equal to 8192 and 65536 bytes resulted in a shorter acquisition process completion.

4.2 Parallelism

4.2.1 Throughput

The results (Figure 8) show that in the RTT 0 case, two and four workers setting achieved an average throughput higher than the reference single process run.

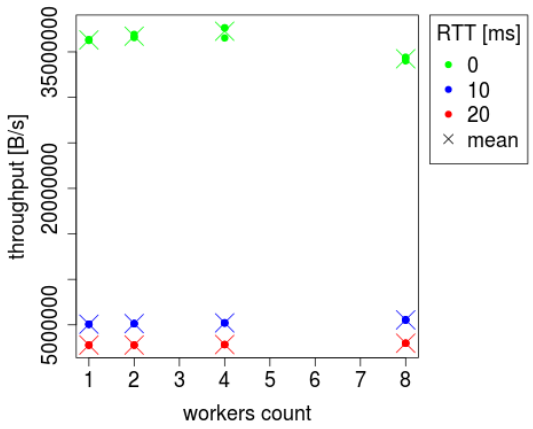


Figure 8: Parallelism, average throughput (higher is better)

A score of eight working threads showed a meaningful drop. For the relative delay equal to 10 and 20 milliseconds, the higher the number of processes, the higher the average relative throughput reached.

4.2.2 Execution time

The elapsed time graph (Figure 9) depicts that the eight threads test was the fastest in all the cases.

In the behaviour of the RTT 0 scenario, despite having the worst average bandwidth, eight workers could still do the job faster than other acquisition runs.

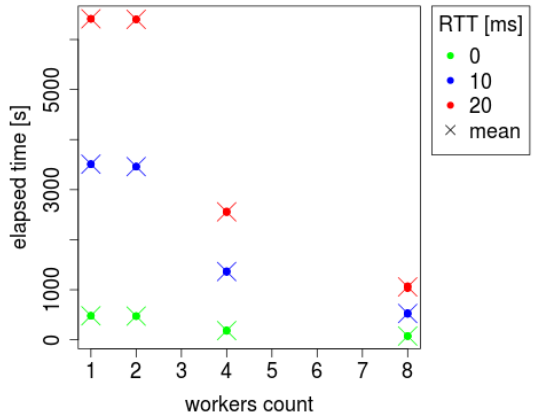


Figure 9: Parallelism, elapsed time (lower is better)

5 Discussion

5.1 Prefetching

In the first case, the approach used influenced the process in a negative way. When the delay was introduced to the connection, none of the chosen read-ahead and read-before parameters brought any benefits.

The used technique degraded the performance noticeably. The simplified implementation of prefetching (Section 3.2.1.1) definitely increased the overall response time of the FUSE provided block device. Moreover, the chosen test scenario covered only the small area of possible size settings.

5.2 Parallelism

The outcome of the parallelism test appears to prove the usefulness of spreading the workload between multiple workers. Figure 9 illustrates that more workers resulted in a lower elapsed time score. The factor that was expected to enable observed improvement was an increase of throughput. However, figure 8 does not confirm it. Although the average throughput was improved for a reference test, the RTT 10 and 20 cases do not show any observable difference.

Even though the test procedure was designed with the cache awareness in mind, it assumed the lack of client-side read cache for a block device. The `blktrace`

tool log created as the part of the post-mortem analysis appears to confirm this factor as the potential flaw in the taken method.

6 Future work

Similarly to Yamaguchi et al. [9] it would be beneficial to investigate specific iSCSI implementation subtleties against the model of *Short Block Access* introduced by the authors. Potential research could also attempt to extend defined high latency (one way delay of 4ms) to the higher values.

Furthermore, one may apply conclusions from the papers focusing on the practical side of delayed iSCSI I/O tuning [7] [8] to triage domain. Specifically, to extend the scope of conducted tests to random reads, which were not examined in this research.

ATA over Ethernet (AoE) protocol is an interesting alternative to the iSCSI. However, as it runs directly on top of Network Layer 2 — it is not routable. Research discovering the possibilities of spanning AoE over multiple network domains while still becoming performance advantageous could shed new light on the remote acquisition problem.

The delay of a network link is one factor decreasing remote triage performance. Offloading the triage workload to be computed on the remote endpoint and concurrently streaming the results back could partly cancel link latency implications.

References

[1] Gary Palmer. *A Road Map for Digital Forensic Research*. Tech. rep. DFRWS, 2001.

[2] Simson L. Garfinkel. 'Digital media triage with bulk data analysis and bulk_extractor'. In: *Computers & Security* 32 (2013), pp. 56–72. ISSN: 0167-4048. DOI: http://dx.doi.org/10.1016/j.cose.2012.09.011. URL: http://www.sciencedirect.com/science/article/pii/S0167404812001472.

[3] Matthew Komorowski. *A history of storage cost*. 2009. URL: http://www.mkomo.com/cost-per-gigabyte.

[4] Vassil Roussev, Candice Quates and Robert Martell. 'Real-time Digital Forensics and Triage'. In: *Digit. Investig.* 10.2 (Sept. 2013), pp. 158–167. ISSN: 1742-2876. DOI: 10.1016/j.diin.2013.02.001. URL: http://dx.doi.org/10.1016/j.diin.2013.02.001.

[5] Steve VanDeBogart, Christopher Frost and Eddie Kohler. 'Reducing Seek Overhead with Application-directed Prefetching'. In: *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*. USENIX'09. San Diego, California: USENIX Association, 2009, pp. 24–24. URL: http://dl.acm.org/citation.cfm?id=1855807.1855831.

[6] Pei Cao et al. 'Implementation and Performance of Integrated Application-controlled File Caching, Prefetching, and Disk Scheduling'. In: *ACM Trans. Comput. Syst.* 14.4 (Nov. 1996), pp. 311–343. ISSN: 0734-2071. DOI: 10.1145/235543.235544. URL: http://doi.acm.org/10.1145/235543.235544.

[7] Y. Zhang and M.H. MacGregor. 'Tuning Open-iSCSI for Operation over WAN Links'. In: *Communication Networks and Services Research Conference (CNSR), 2011 Ninth Annual*. May 2011, pp. 85–92. DOI: 10.1109/CNSR.2011.21.

[8] Masato Oguchi et al. 'Performance Improvement of iSCSI Remote Storage Access through Optimization of Multiple Layers.' In: *JSW* 8.3 (2013), pp. 538–546. URL: http://dblp.uni-trier.de/db/journals/jsw/jsw8.html#OguchiHMOY13.

[9] S. Yamaguchi, M. Oguchi and M. Kitsuregawa. 'Analysis of iSCSI short blocks access'. In: *Digital Information Management, 2007. ICDIM '07. 2nd International Conference on*. Vol. 2. Oct. 2007, pp. 574–576. DOI: 10.1109/ICDIM.2007.4444285.

[10] *Lansing chapter of ISSA*. URL: http://lansingmi.issa.org/ (visited on 10/04/2016).

[11] *Open-iSCSI project information*. URL: http://linux-iscsi.org/wiki/Open-iSCSI (visited on 10/04/2016).

[12] *The iSCSI Enterprise Target project*. URL: http://iscsitarget.sourceforge.net/ (visited on 10/04/2016).

[13] *netem utility information*. URL: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem (visited on 10/04/2016).

[14] Fengguang Wu. 'Sequential file prefetching in Linux'. In: *Advanced Operating Systems and Kernel Applications: Techniques and Technologies: Techniques and Technologies* (2009), p. 218.

[15] Eric van den Haak. *Remote data acquisition on block devices in large environments. A study into copy-on-read and copy-on-write methods*. Tech. rep. University of Amsterdam, 2014.

[16] Łukasz Makowski. *triage.py source code*. 2015. URL: https://github.com/maq123/triage/blob/master/triage.py.

[17] Shawn Ostermann. *tcptrace - Official Homepage*. 2015. URL: http://tcptrace.org.

[18] Manikantan Ramadas. *TCPTRACE Manual*. 2003. URL: http://www.tcptrace.org/tcptrace-manual/manual/index.html.