Advances in Science and Technology Research Journal

Advances in Science and Technology Research Journal, 2025, 19(11), 502–514
https://doi.org/10.12913/22998624/209581
ISSN 2299-8624, License CC-BY 4.0

Received: 2025.05.30
Accepted: 2025.09.15
Published: 2025.10.01

Power law distributions evidences in cache memory bytes – Windows performance counter

Bartosz Kowal[1*], Dominik Strzałka[1]

[1] Department of Complex Systems, Rzeszów University of Technology, Al. Powstańców Warszawy 12, 35-959 Rzeszów, Poland
* Corresponding author's e-mail: b.kowal@prz.edu.pl

ABSTRACT

In this paper, the main attention is paid to the analysis of Windows operating system counter Memory Cache Bytes. Thanks to the Windows Perfmon tool, it was possible to gather long time series that show interesting, from statistical point of view, behavior of different Windows Desktop operating system versions (Windows 7, 8, 10) in idle and loaded mode. The comparative analysis of this counter behavior, understood as a time series that represents cache memory, will show that based on internal memory management mechanisms power law distributions are omnipresent phenomena. We focused on McCulloch, Kout methods and Hill estimator for calculations of alpha stability index and Q-Q plot with Anderson-Darling test for distribution normality test. All of the tested time series indicated the existence of deep probability heavy-tailed distributions for extreme values, confirming that operating system has to deal with anomalous cache memory behavior. This feature is common for all tested 64-bit hardware configurations regardless of the workload mode.

Keywords: operating systems, counters, performance tests, long-term behavior, long-range dependencies, Lévy processes, power laws.

INTRODUCTION

The research problem of computer systems understood as mass service systems, performance, has been one of the key challenges for more than fifty years [1, 2]. Today, in comparison to past times, it is possible not only to consider this challenge in terms of theoretical analysis but also to collect huge real datasets that can be further analyzed [3, 4]. On the basis of obtained statistics, experts are able to create exact statistical models that can be used for modeling. Data sets can be collected after experiments with different hardware configurations, operating systems generations. They can also include different types of workloads generated by benchmarks and human users. Here, in this paper, the authors focus on Windows operating system built-in solutions. Compared with the approaches known in the literature [1, 2], where analytical solutions assume a single, unchanging Gaussian-domain distribution, our study not only collects data but also applies several statistical approaches to reveal data-set properties that are particularly interesting from a statistical point of view. In real data, compared to the models considered in literature, we can have a mixture of different probability distributions with power-law properties. One of the main motivations for this paper is to discover and obtain new knowledge about the real behavior of computer memory system considered as a whole. This means that hardware is processing tasks and is also working together with operating system software. The research presented here constitutes an extension of earlier works, in which the cache-byte counter was analyzed, revealing the presence of self-similarity and long-range dependencies [5, 6]. That work is now complemented by an additional analysis of power-law distributions. Previous research is supported by new sets of collected data with

new hardware configurations. However, to meet the challenges of accurately analyzing computer system performance, it is important to look for statistical methods. These methods must enable more accurate data collection, analysis, and interpretation.

The analysis of counters and probability distribution densities for collected data is necessary to gain a deeper understanding of system behavior at the cache level, which plays a key role in system performance. In the literature, a similar approach can be seen in: origins of the long-range correlations of ionic current fluctuations in membrane channels [7], water levels [8], financial and volcanic time series [9], paper citations [10], mechanical systems [11]. In paper it is shown that precise monitoring of operating system counters allows identification, for example, cache efficiency patterns, which have a direct impact on the speed and the stability of entire system application. In the case of computer systems analysis this is not widely used approach with well-established methodology, but for instance some examples can be referred to: computer programs [12], software [13], system-on-chip (SoC) and power-efficient network-on-chip (NoC) topologies [14]. Regarding memory-system analysis based on collected time series, it is worth mentioning paper [15]. There memory-access traces were collected over time and were analyzed using autocorrelation and alpha-stable process models to characterize time-series patterns in CPU benchmark workloads are used. But in comparison to the presented paper, the authors mostly focused on CPU. In paper [16], chaos theory and nonlinear dynamical systems analysis are used to analyze system performance counters, including memory usage. However, it is not related to the existence of power-laws. The understanding of probability distributions, especially the observation of "heavy tails", enables the prediction of extreme events that may require intervention or optimization [17]. For this reason, in-depth analysis of counters and their probability density distributions becomes a key element in the process of managing the performance and reliability of computer systems.

In successive sections, the following topics will be discussed. After the introduction, in Section 2 the architecture and different types and groups of counters will be considered. Next, a comparison of different counter-groups will be presented as a review of their development in accordance with the development of Windows operating systems. In Section 4 several practical examples related to the use of this internal systems solution will be shown. Finally, the paper will be summarized in Section 5.

A SHORT REVIEW OF PERFORMANCE COUNTERS HISTORY IN OPERATING SYSTEMS: FROM WINDOWS XP TO WINDOWS 11

In the family of Windows operating systems, counters as internal elements of the system appeared for the first time in Windows NT3.1 [17]. Initially, direct access was limited and required additional actions done by the user, for instance, system upgrade with installation of software package, but later in new versions of operating systems direct access to counters via perfmon tool was allowed. This created the possibility of dealing with long-term records of operating system counters. Because there are no similar comparisons of counters in the literature, this paper gives some details on these issues.

The Microsoft company calls each type of data collected by the operating system, for example, the amount of RAM used, % CPU usage, etc., a performance counter [18, 19]. The data of such counters contains detailed information about the behavior of the system. Each counter in MS Windows has an assigned counter type. This type determines how the information collected by the counter is calculated and displayed. Looking at the current available in MS Windows 10 (21H1) list, there are more than 800 different counters. It should also be emphasized that, depending on the installed hardware and the MS Windows operating system version, the number of counters and groups may vary. This is because, for example, equipment manufacturers can write their own counter for their device.

Each counter in MS Windows is based on some mathematical functions that allow calculations to be done. For example, in MS Windows 10 (21H1), there are 34 types of counters, showing the methods of downloading raw data, e.g., from the processor, and then passing this value to the appropriate type of counter. An example of a counter type is presented in Table 1 [20].

As can be seen in Table 1, the method of retrieving data is immediate, i.e. when the command

Table 1. Type of counter: PERF_COUNTER_RAWCOUNT [21]

Description	This counter type shows the last observed value only
Data read time	Instantaneous
Formula	Shows data in RAW form
Average	$\frac{SUM(N)}{x}$ *N* is used to describe the raw counter data.
Counter example	Memory\Cache bytes

is invoked, the data are read and then processed by the function returning the result. This type returns data that has been calculated as the average of the last two measurements over time.

One of the default limits of the built-in data collection tools is the available time interval, which typically is from 1 second to 1 minute [22]. However, it is possible to collect data at smaller intervals, even every 100 ms using, for example, the performance API or using PowerShell [18]. Collecting data from system counters more frequently than the default limit of 1 second (it is assumed in the Perfmon – performance monitor – system tool) may be justified in situations requiring precise, real-time performance analysis. For time-intensive applications such as network traffic or video games, shorter measurement intervals can help identify and resolve performance problems as they arise. However, an increased frequency of data collection can also lead to a significant system load, especially when monitoring multiple counters simultaneously. Therefore, it is crucial to strike a balance between the need for detailed data, the potential impact on system performance, and the type of counter itself [22]. In the case of sampling strategy, authors decided to take 1 second interval. This decision was based on the observation that data variability in smaller time intervals was very low. Also, several tests were done with different sampling rates: 10, 30, and 60 seconds, and in any case the existence of non-heavy-tailed distributions was confirmed. This is direct proof that heavy-tails (power-law distributions) properties are scale independent and may do not vanish at coarse resolutions. Moreover, higher (i.e. above 1 second) sampling rates lead to the loss of data precision – accidentally such extreme phenomena like memory dumps, computer resets, high jumps in memory usage can be deleted from time series.

For example, the cache bytes counter indicates the size, in bytes, of the portion of the system file cache that currently resides in and is active in physical memory. According to the documentation, this counter only shows the last measured value [21, 23]. Table 2 shows the counter data returned with a resolution of 100 milliseconds and 1 second. As can be seen in the table, the variability in smaller time intervals is so small that it is not recommended to use sampling smaller than 1 second. However, this is not a rule, and it is recommended to adapt the method of selecting the intervals for each counter separately [22].

COMPARATIVE ANALYSIS OF COUNTERS IN WINDOWS SYSTEMS – FROM XP TO 11

In the previous chapter, a short description of system counters was given. Over the years, technological progress has forced operating system designers to generate more information. Taking into account the beginning of the Windows desktop operating system family investigated in the paper, the authors checked how many counters and counter groups there are in XP-11 operating systems. Windows XP allowed collecting only 554 counters from 30 different groups, e.g. cache, memory, and processor. From the time perspective and subsequent versions of the system, this number seems to be quite small. For example, the next generation of the system, i.e. Windows Vista, already has 965 counters in 60 groups, while Windows 7 64 bit allows to collect data from 1.473 counters in 91 groups. As can be read in Table 3, in each subsequent generation of the operating system, there is a significant increase in the number of built-in system counters.

One of the issues of this type of tests, is the existence of various compilations or updates of operating systems, e.g. service packs for Windows XP, Vista, and 7, or various compilations of later operating systems.

The individual operating system builds were also detailed during the tests:

Table 2. Sample data received from the cache bytes counter in two-time intervals

Time	Counter value cache bytes [100 ms]	Counter value cache bytes [1 s]
12:40:00.079	539754500	
12:40:00.199	539758600	
12:40:00.309	539758600	539758600
12:40:00.418	539762700	
12:40:00.526	539775000	
12:40:00.635	539914200	
12:40:00.744	539963400	
12:40:00.855	539963400	
12:40:00.965	539959300	
12:40:01.072	539959300	
12:40:01.182	539959300	
12:40:01.289	539967500	
12:40:01.399	539967500	539967500
12:40:01.508	539967500	
12:40:01.617	540004400	
12:40:01.726	540004400	
12:40:01.835	540004400	
12:40:01.945	539848700	

- Windows XP Professional, Service Pack 3,
- Windows Vista Business, Service Pack 1,
- Windows 7 Professional x86, Service Pack 1,
- Windows 7 Professional x64, Service Pack 1,
- Windows 8 Professional x86, Build 9200,
- Windows 8 Professional x64, Build 9200,
- Windows 8.1 Professional x86, Build 9200,
- Windows 8.1 Professional x64, Build 9200,
- Windows 10 Professional x86, version 22H2,
- Windows 10 Professional x64, version 22H2,
- Windows 11, version 23H2.

The next test focused on analyzing the counters between the presented generations of the MS Windows system to specify the groups and the counters themselves that are in each system, regardless of the processor architecture. Data analysis revealed the presence of 347 counters distributed into 19 unique counter groups that are common to all tested generations of the MS Windows operating system.

These groups include categories such as memory, processor(*), and network interface(*), which emphasizes their fundamental importance for monitoring and diagnosing the performance of the operating system. The complete list of groups and the sum of the counters are presented in Table 4. The presence of these counter groups in all generations of the Windows operating system indicates stability and continuity in Microsoft's approach to the development of diagnostic and system monitoring tools, especially in the areas of memory and processor. Such a broad coverage indicates a key approach to operating system administration, focused on the performance, reliability, and availability of Windows operating systems in various work environments.

Analysis of data from counters

The analysis of system counter data is a complex process that requires detailed planning and understanding of the operational context. The key challenges are the amount of data generated by the counters and their effective storage. This data can be collected from a variety of sources with a potentially high sampling rate, which can lead to a rapid increase in data volume and requires the use of optimized solutions for their processing.

Table 5 shows how different sampling settings affect the amount of data we collect. One key observation is that measuring data every 100 milliseconds may not always be accurate, because sometimes there are measurement time intervals in the range of 95–115 milliseconds. This shows the limitation of data collection precision. Moreover, the increased amount of data generated by more frequent sampling requires not only greater processing throughput from monitoring systems

Table 3. Number of groups of counters and MS Windows Desktop system counters over the subsequent generations

OS	Counter groups	Counters
Windows XP	30	554
Windows Vista	60	965
Windows 7 32bit	91	1473
Windows 7 64bit	91	1473
Windows 8 32bit	105	1696
Windows 8 64bit	105	1698
Windows 8.1 32bit	118	1803
Windows 8.1 64bit	110	1728
Windows 10 32bit	146	2161
Windows 10 64bit	153	2439
Windows 11	171	2879

but also increased memory capacity for storing these data. In turn, the additional counters added for monitoring at 100-ms intervals proportionally increases the size of the collected data. This highlights the need to make careful resource calculations before initiating long-term monitoring sessions. The results and conclusions presented in Table 5 provide important information for designers of monitoring systems, who must take into account both the precision of the collected data and the system's efficiency in terms of information processing and storage.

The analysis and interpretation of particular counters itself is another challenge, considering that the number of available counters can range from 554 in XP to as many as 2879 in Windows 11. This diversity requires system administrators to have a deep understanding of each counter specifics and its potential impact on system performance.

Another problem with data analysis is the fact that counters can be classified into one of 34 types. In these types, data can be processed and returned in a variety of ways. For example, some counters provide the average number of operations performed in each second of the sampling period. Others display the average hit ratio for all operations during the last two sampling periods, allowing one to evaluate the effectiveness of a given operation. The others show the ratio of a subset to its set as a percentage, allowing a quick assessment of the proportions of individual elements in the context of the whole [20].

To show the issues related to counter analysis, the memory group and the cache bytes counter were selected. This counter is one of the most important counters that indicates the size (in bytes) of the portion of the system file cache that currently resides in and is active in physical memory. This counter, due to its presence in all systems under analysis, highlights its significant role in monitoring system performance.

Table 4. Performance counters that are present in all generations of the Microsoft Windows operating system

Counter group	Number of Counters
Browser	20
Cache	27
ICMP	27
Job Object Details(*)	27
LogicalDisk(*)	23
Memory	29
NBT Connection(*)	3
Network Interface(*)	17
Objects	6
Paging File(*)	2
PhysicalDisk(*)	21
Print Queue(*)	13
Process(*)	27
Processor(*)	15
Redirector	37
System	17
Telephony	9
Terminal Services Session(*)	15
Thread(*)	12

DATA ANALYSIS BASED ON THE CACHE BYTE COUNTER

The main assumption of the experiment was to perform a long-term observation of the performance of Microsoft Windows 7, 8, and 10 operating systems (x64 architecture), running in two environments: on identically configured virtual machines without any user influence, and on 24 physical computers with user interface (eight per OS version). A total set of 6 virtual machines took part in the research; 2 virtual machines for each of Microsoft 7, 8, 10 operating systems with the following configuration parameters: 2 GB RAM, 2 vCPU, 50GB HDD. The data collected includes uninterrupted work of operating systems for a total of approximately 30 days. To avoid undefined behavior of operating systems, computers do not

Table 5. Comparison of data collections efficiency

Number of counters	Sampling rate	Number of rows	Data size (bytes)	Data size per second (bytes/sec)
1	1 s	3.562	276,990	76.94
1	100 ms	32.535	2,530,428	703.45
2	1 s	3.562	383,074	106.41
2	100 ms	32.535	3,498,994	971.94

have access to the Internet or internal network, and the user does not perform any operations.

In addition to the virtual machines, the study also employed a group of 24 physical computers, divided into three groups of eight machines each running Windows 7, Windows 8 and Windows 10 (x64 architecture) with varied hardware configurations. Unlike the VM set, these physical machines were used by real users carrying out their normal day-to-day tasks without any artificially generated load throughout the observation period. In practice, no two users have the same machines, application sets, or daily routines. Hardware details and data sets are given in [24]. In addition, tracking every action of participants throughout the experiment would create unresolvable privacy and consent problems and can significantly disturb performed experiments.

Figure 1 shows the cache bytes counter behavior for each operating system separately. As can be seen, in each version of the presented operating system, the behavior of this counter is different, even without any user influence. This raises a problem with the interpretation of results and the further proposals of different models, e.g. for optimizing cache management.

The diversity in the cache bytes counter behavior in different versions of the Microsoft Windows operating system, shown in Figure 1, indicates the need for an in-depth analysis of these patterns, which could contribute to a better understanding of system memory management. Instead of focusing on differences, the key point can be the search for common patterns, such as characteristic changes in data values or typical probability distributions, which would enable the creation of more universal models of cache processes.

In the future, such an analysis could help identify potential performance issues before they occur, making systems more stable and responsive. Finally, a better understanding of the behavior of this counter could contribute to the development of advanced monitoring techniques that could adapt to dynamically changing operating conditions of the system, which is crucial in the management of complex computer systems. However, this work is focused on examining the statistical properties of the data collected from counters in order to learn the characteristics of processes occurring in cache memory during processing in the operating system.

Cache memory bytes counter data analysis

The next stage of the analysis is to determine the probability distribution for the cache byte counter, whose behavior is a random process described by the random variable X. This will enable understanding the dynamics of processing over time and the overall cache utilization profile. Such an analysis will help determine whether there are statistical similarities in OS Windows cache management between operating systems. This method focusses on understanding the variability and characteristics of the data.

Two methods are used mainly to analyze the random variable X probability distributions, represented by the time series $X_i(t)$, $i \in \{1,..., n\}$ – histograms and kernel density estimators. This analysis focusses on the use of kernel density estimators, especially the Epanechnikov kernel with smoothing according to Silverman's rule, which is considered to be particularly effective [25–26].

Due to the fact that the counter indicates the total value of cache bytes in a given second, i.e. only the positive ones, it is more difficult to read dynamic changes. An approach was chosen in which the probability density analysis of each tested data set was performed using increments, i.e. the difference of cache bytes in its current and previous value: $X(t) - X(t-1)$. This simple approach allows us to check how the number of bytes of the cache used in a given operating system changes every 1 second. Figure 2 shows an example cache byte counter values chart for selected machines (with order reference to Figure 1).

Based on the increments shown in Figure 2, four complementary steps can be carried out to

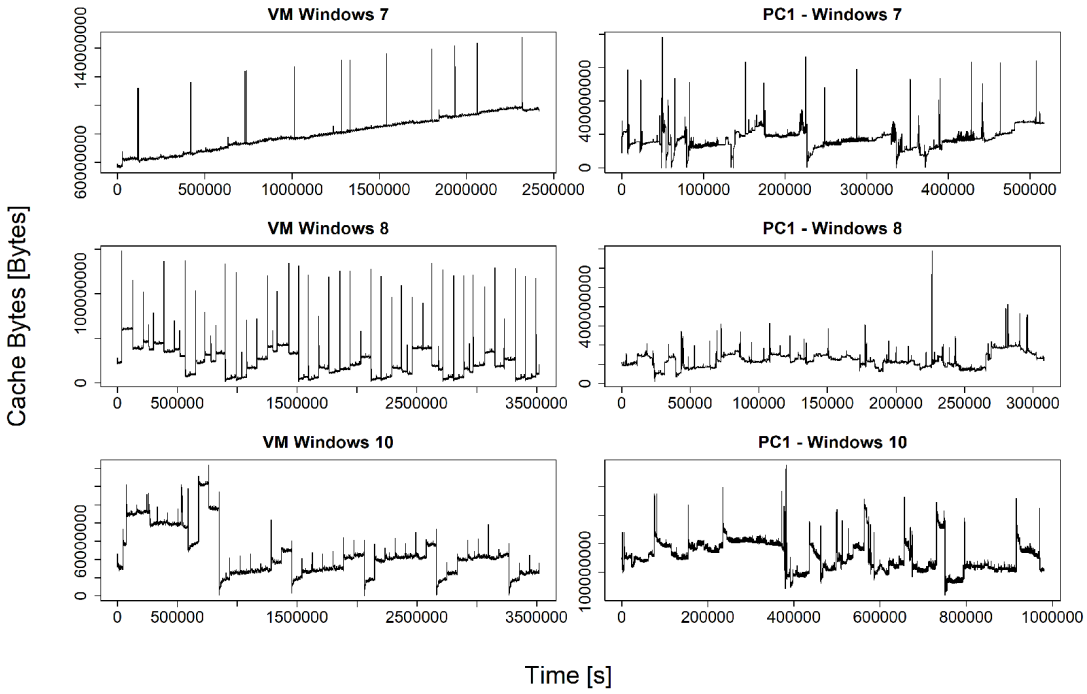

Figure 1. Cache bytes comparison between selected machines: left column – virtual machines, right column – PCs from group 1

verify the hypothesis of the heavy-tailed distributions occurrence in all tested cases:

1. Tail index estimation α (McCulloch, Kout, Kogon-McCulloch methods) [27].
2. Analysis of Q–Q charts and the Anderson-Darling test for compliance with the normal distribution.
3. Hill exponent estimation – see reference [28] for more details.
4. Graphical analysis.

Table 6 shows the results of the three α heavy-tail index estimation methods for all tested operating systems and hardware configurations. Due to specific properties of analyzed time series (e.g. large number of zeros; in some time series it is even 90% of the records), some numerical algorithms for α stability index returned NA. NA indicates that selected methods could not reliably estimate this parameter, but the reasons for this result are not known. For example, the McCulloch quantile estimator relies on matching five sample quantiles to precomputed lookup tables that map those quantiles to the stable distribution parameters α and β. Whenever two or more required quantiles coincide or the result falls outside the range of the tables, the procedure fails to converge, and the function returns the fallback value α = 0.5. This value could be merely an error flag and could therefore be treated the same as NA. There are also data for which the McCulloch, Kout and Kogon-McCulloch (IG) methods returned results indicating the presence of heavy-tails. In the case of the McCulloch method, returning a value of 0.5 suggests a possible problem with the algorithm's operation and could be also treated as NA. Thus, also graphical tests were performed, like Q-Q plots and visualizations of probability distributions.

From the results presented in Table 6, it is clear that because not always used numerical methods can effectively prove the existence of heavy tails, thus a further test can be performed comparing the empirical distributions with the theoretical normal distribution. It will help to confirm or reject the thesis about the existence of distribution "heavy tails".

One such test is the empirical analysis of Q-Q plot for selected machines, which is presented in Figure 3. Graphical analysis and comparison with the normal distribution in all analyzed cases showed significant differences in the shape of the distribution, with dominant heavy tails in the data, especially visible in the extreme quantile values.

In the case of the Anderson-Darling test, each tested time series returned a *p*-value of

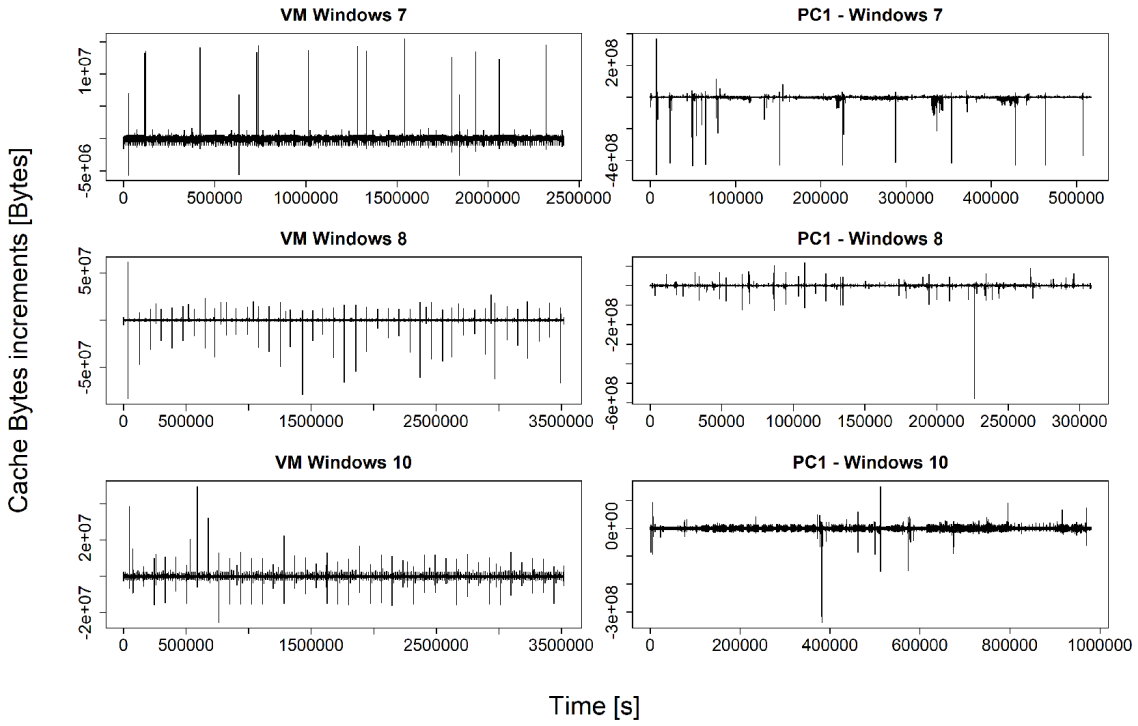

Figure 2. Cache byte increments comparison between selected machines

3.7×10^{-24}, which indicates a strong rejection of the null hypothesis of compliance with the normal distribution – the tested data statistically significantly deviate from the normal distribution. It should be emphasized, according to paper [5], that the data can exhibit long-range dependencies, which violates the classical assumption of samples independence. The Hurst exponent determined for the studied series significantly exceeds 0.5, "confirming the existence of deep statistical long-term dependencies in all studied time series" – the samples are strongly dependent over time. Nevertheless, the extremely low p-value of the Anderson-Darling test (on the order of 10^{-24}) clearly indicates a deviation from normal distribution. In this paper, an emphasis was placed on qualitative conclusions derived from the Q-Q plots and estimator values (e.g., Hill's). By the use of data increments and very large sample size, the influence of dependencies between subsequent observations was minimized, and the characteristics of heavy tails of the distributions were highlighted. In the future work, it is planned to utilize significant verification methods that take into account dependencies (e.g., block bootstrap procedures) to formally confirm the obtained results.

Another test for the occurrence of heavy-tailed distributions is the analysis of the Hill exponent, which is shown in Figure 4.

The analysis of the Hill exponent for all the operating systems and load mode shows that there are heavy tails for a certain range of the ordinal statistics k. One of the difficulties in estimating this parameter is finding the optimal value of the parameter k that will be free of bias; therefore, one method is to take the last 5–20% of the k value and look for the value of the α parameter there or use one of the optimization methods.

In order to find the optimal order statistic k, the Tea package from the Crane repository [29] was used to select the dAMSE function, which aims to provide the optimal number of order statistics k for the Hill estimator. The results of the estimated α parameter are presented in Table 7.

Although the numerical dAMSE method produces unstable, extreme values of α in some situations (when the optimal k falls outside the recommended range of the last 10–20% of the order statistics k), the empirical, graphical Hill exponent confirms the existence of heavy-tails. This is a general problem with calculations of the α stability index [30]. There is no commonly accepted, universal method, thus in paper several were used to compare obtained results [31, 32]. Some obtained results indicate extreme α values calculated by

Table 6. Distribution of α index stability estimation

OS	McCulloch [α]	Kout [α]	IG [α]
VM1 Windows 7 2GB	0.5	NA	NA
VM2 Windows 7 2GB	0.5	NA	NA
VM1 Windows 8 2GB	0.5	NA	NA
VM2 Windows 8 2GB	0.5	NA	NA
VM1 Windows 10 2GB	0.5	0.127	0.0093
VM2 Windows 10 2GB	0.5	0.1279	0.0053
PC1 Windows 7	0.5	0.236	0.303
PC2 Windows 7	0.5	0.739	0.709
PC3 Windows 7	0.576	0.408	0.454
PC4 Windows 7	0.586	0.483	0.487
PC5 Windows 7	0.5	NA	NA
PC6 Windows 7	0.5	NA	NA
PC7 Windows 7	0.663	0.413	0.563
PC8 Windows 7	0.5	0.299	0.367
PC1 Windows 8	0.5	NA	NA
PC2 Windows 8	0.685	0.142	0.372
PC3 Windows 8	0.5	NA	NA
PC4 Windows 8	0.5	NA	NA
PC5 Windows 8	0.5	NA	NA
PC6 Windows 8	0.554	0.766	0.786
PC7 Windows 8	0.79	0.515	0.683
PC8 Windows 8	0.568	0.209	0.18
PC1 Windows 10	0.5	NA	0.037
PC2 Windows 10	0.597	0.268	0.306
PC3 Windows 10	0.5	NA	NA
PC4 Windows 10	0.5	NA	NA
PC5 Windows 10	0.5	NA	NA
PC6 Windows 10	0.5	NA	NA
PC7 Windows 10	0.5	NA	NA
PC8 Windows 10	0.5	0.158	0.212

numerical methods, but also a graphical approach was taken together with Q-Q plots. Extreme α value could be a prerequisite for a non-heavy-tailed distribution. In the case of Table 7, α values were returned by the dAMSE method, which can yield unexpected optimal k values that reflect α values outside the Lévy domain. Therefore, in case of inconsistencies, each series should be manually analyzed in addition to the dAMSE method.

To better visualize the distribution features of the analyzed increments of cache bytes in the graph (Figure 5), the vertical axis is presented on a logarithmic scale. In Figure 5, the blue curve shows the estimated real probability distribution using Epanechnikov kernel, whereas the red curve represents the theoretical normal distribution. The results provide information about the analyzed systems' performance: in each of the tested virtual machines the cache byte counter values do not change without user intervention for at least 90% of the system operation time. This conclusion may give the false impression that there is no visible process dynamics in the operating system (colloquially: the system does nothing) – the system should remain in a stable state. A look at Figure 5 shows that the probability density graph visualized on a log-lin scale shows the potential occurrence of the so-called "heavy tails" in the probability distribution (right and left tails show extreme events and are slowly vanishing). These types of distributions are characteristic of processes with

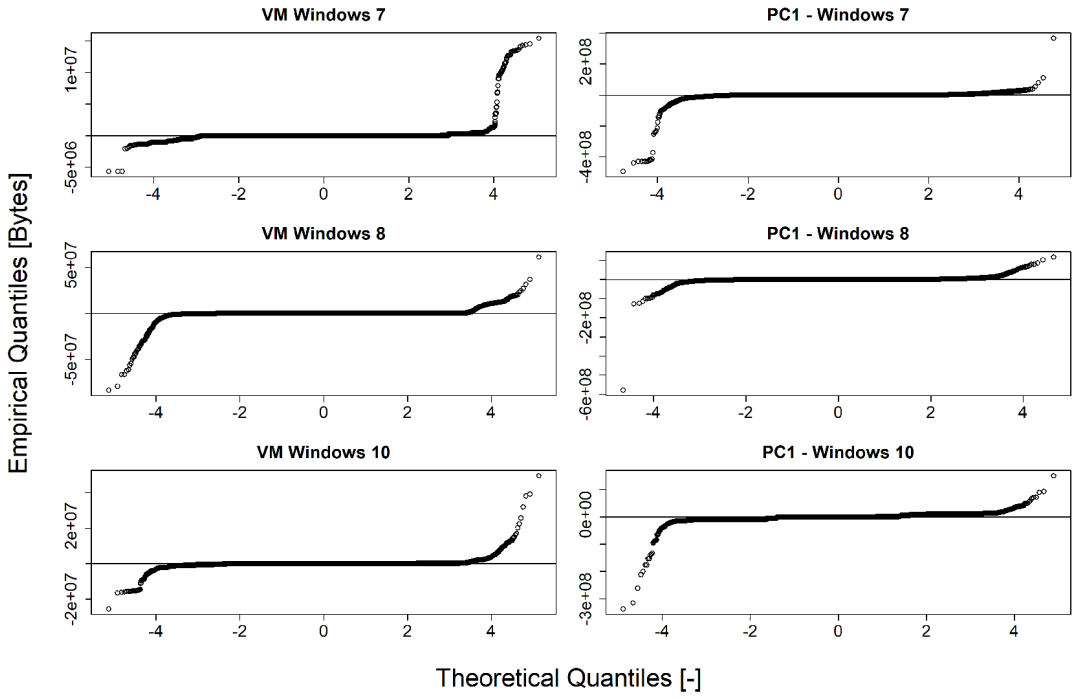

Figure 3. Q-Q graphs for selected machines

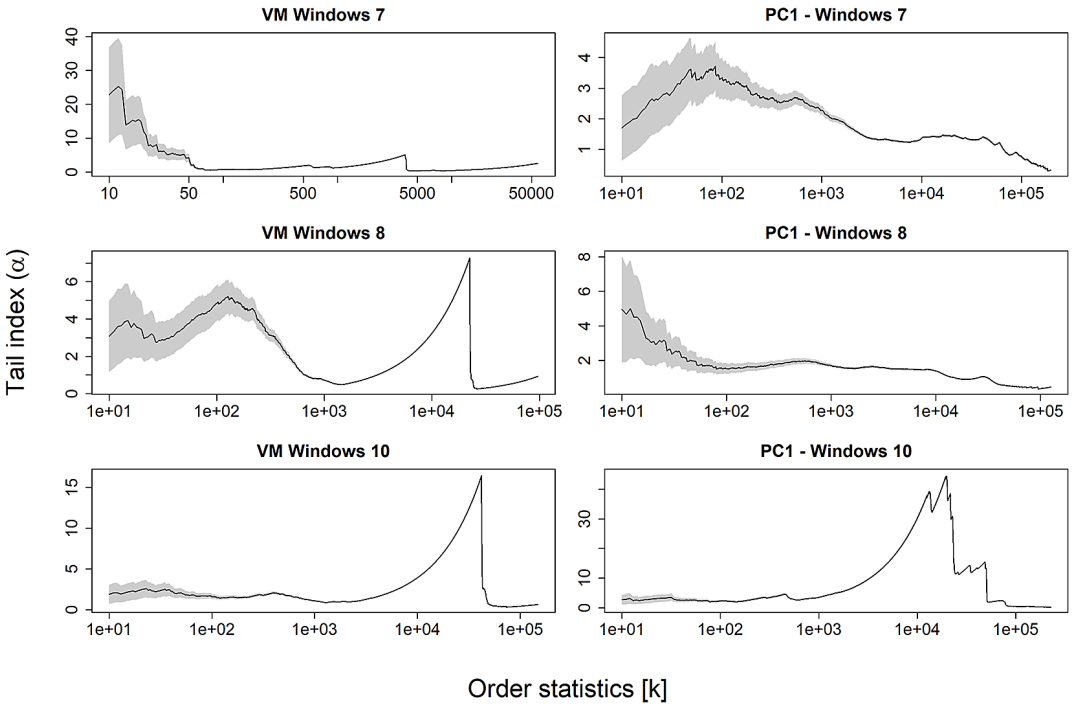

Figure 4. Hill estimator for selected machines

extreme values. This effect is known in many different cases [17] and means that the process cannot always be modelled using a random walk – from a statistical perspective. An approach based on the Lévy processes may be needed. Figure 5 shows that the heavy tails appear in the probability distributions for each operating system regardless of its version and its load. This effect can be much better observed if the probability distribution is plotted on a double logarithmic scale, but it will be only visible for increments greater than 0.

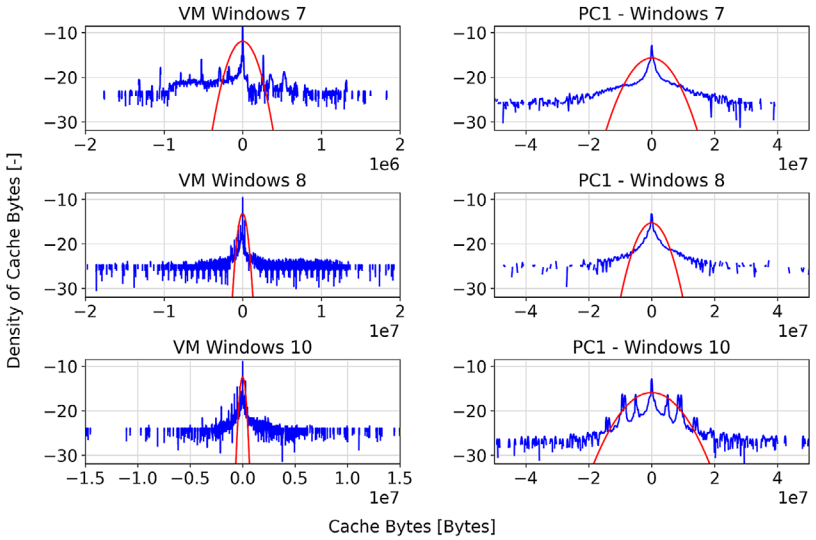

Figure 5. Probability distribution of cache bytes increments for selected machines. The Sensity Of Cache bytes is presented on a log scale. The red line shows the fitted normal distribution; blue line shows the real distribution. Normal distribution was estimated by calculated mean and variance

Table 7. The α parameter estimation results for the Hill estimator

OS	Optimal k	Estimated α
VM1 Windows 7 2GB	4100	0.571
VM2 Windows 7 2GB	4213	0.433
VM1 Windows 8 2GB	6214	2.007
VM2 Windows 8 2GB	5785	1.941
VM1 Windows 10 2GB	4303	1.688
VM2 Windows 10 2GB	3503	1.314
PC1 Windows 7	772	2.406
PC2 Windows 7	4031	1.633
PC3 Windows 7	29	2.715
PC4 Windows 7	223	1.178
PC5 Windows 7	3662	1.338
PC6 Windows 7	392	1.194
PC7 Windows 7	751	2.105
PC8 Windows 7	32	1.815
PC1 Windows 8	1254	1.592
PC2 Windows 8	424	2.295
PC3 Windows 8	54	4.188
PC4 Windows 8	6385	0.994
PC5 Windows 8	37	2.985
PC6 Windows 8	4859	1.194
PC7 Windows 8	603	3.387
PC8 Windows 8	116	164.929
PC1 Windows 10	142	1.979
PC2 Windows 10	37	54.728
PC3 Windows 10	52	3.145
PC4 Windows 10	888	2.497
PC5 Windows 10	2501	1.629
PC6 Windows 10	45	4.587
PC7 Windows 10	2702	1.858
PC8 Windows 10	129	2.463

CONCLUSIONS

In this paper authors were able to discover interesting patterns that can be also found in computer systems behavior. This is the main achievement of presented work. It seems to be a little bit surprising bearing in mind that computer systems technical time is going much faster than in other physical systems. Power-law distributions exist in many different natural phenomena, and their appearance is one of the first key indicators of complex behavior with long-range dependencies. In the case of computer systems, it is an indicator of their dynamical behavior that was caused by human generated workload. From the presented paper, in comparison to the so far used analytical models from queuing theory, it is clear the computer systems managed by Windows operating systems can behave in scale-free regime with dynamical behavior that influences their performance. This is visible at least via the cache memory counter.

The proposed paper draws attention to the presence of "heavy tails" in the analyzed data, which indicates the characteristic Lévy processes in computer systems. This observation not only confirms the complexity of computer systems behavior, but also highlights the need for further research in this direction. This problem is not well-recognized in literature, since most of the existing approaches refer to classical queuing models proposed more than 40 years ago. The development of current operating systems allows us to collect

data sets and detect hidden patterns and origins of long-range correlations. This opens a new field for future analyses aimed at delving into the mechanisms responsible for these unique patterns in probability distributions. The considerations presented in the paper show evidence that such patterns can exist in personal computers with MS Windows operating systems when the workload is generated by users.

The existence and understanding of Lévy processes and their impact on computer systems performance dynamics was indicated as crucial to the development of more effective monitoring and optimization methods for computer networks, but in the case of computer systems is still neglected. Since the problem is new, a more detailed study requires other methods for the calculations of the α stability index. Some of them are based not on the features of probability or density distributions but on time series. Among them there are: maximum likelihood estimation (MLE), least squares estimation (LS), weighted least squares estimation (WLS), percentile method (PM), method of moments (MoM).

Therefore, it is crucial to adapt the data collection method to the specificity of the monitored counters and expectations regarding the accuracy of the analyses. Even more, it is good to have real data that is collected in the system where there is a workload (system is not in the idle state) generated by users or other computer programs.

REFERENCES

1. Kleinrock L. Queueing Systems Volume 1: Theory. Journal of the American Statistical Association. 1975.
2. Patterson DA, Hennessy JL. Computer architecture: a quantitative approach. San Francisco (CA): Morgan Kaufmann Publishers Inc. 1990.
3. Fortier PJ, Michel H. Computer Systems performance evaluation and prediction. Digital Press; 2003.
4. Lazowska ED, Zahorjan J, Graham GS, Sevcik KC. Quantitative system performance: Computer system analysis using queueing network models. Englewood Cliffs (NJ): Prentice-Hall, Inc. 1984.
5. Kowal B, Strzalka D. Statistical long-range dependencies and statistical self-similarity in computer systems processing – the case of cache bytes counter. Adv Sci Technol Res J. 2024;18(8):311–18. https://doi.org/10.12913/22998624/194428
6. Strzałka D, Dymora P, Mazurek M. Modified stretched exponential model of computer system resources management limitations—The case of cache memory. Physica A: Statistical Mechanics and its Applications. 2018;491:490–497. https://doi.org/10.1016/j.physa.2017.09.012
7. Mercik S, Weron K. Stochastic origins of the long-range correlations of ionic current fluctuations in membrane channels. Phys Rev E. 2001;63(5):1–10. https://doi.org/10.1103/PhysRevE.63.051910
8. Wang H, Song S, Zhang G, Ayantobo OO, Guo T. Stochastic volatility modeling of daily streamflow time series. Water Resources Research. 2023;59(1):1–18. https://doi.org/10.1029/2021WR031662
9. Mariani MC, Asante PK, Bhuiyan MAM, Beccar-Varela MP, Jaroszewicz S, Tweneboah OK. Long-range correlations and characterization of financial and volcanic time series. Mathematics. 2020;8(3):441. https://doi.org/10.3390/math8030441
10. Klebanov LB, Kuvaeva-Gudoshnikova YV, Rachev ST. Heavy-tailed probability distributions: some examples of their appearance. Mathematics. 2023;11(14):3094. https://doi.org/10.3390/math11143094
11. Żórawski W, Vicen M, Trelka-Druzic A, Góral A, Makrenek M, Adamczak S, Bokuvka O. Microstructure and mechanical properties of cold sprayed amorphous coating. Adv Sci Technol Res J. 2024;18(8):73–85. https://doi.org/10.12913/22998624/193479
12. Zhang H. Discovering power laws in computer programs. Information Processing & Management. 2009;45(4):477–483. https://doi.org/10.1016/j.ipm.2009.02.001
13. Louridas P, Spinellis D, Vlachos V. Power laws in software. ACM Trans Softw Eng Methodol. 2008;18(1):1–26. https://doi.org/10.1145/1391984.1391986
14. Teuscher C, Chung H, Grimm A, Amarnath A, Parashar N. The power of power-laws: Or how to save power in SoC. 2011 International Green Computing Conference and Workshops. IEEE; 2011. https://doi.org/10.1109/IGCC.2011.6008603
15. Zou Q, Zhu Y, Tan Y, Deng Y, Chen W. Temporal characterization of memory access behaviors in SPEC CPU2017 workloads: Analysis and synthesis. Future Generation Computer Systems. 2022;130:33–45. https://doi.org/10.1016/j.future.2021.12.009
16. Alexander Z, Mytkowicz T, Diwan A, Bradley E. Measurement and dynamical analysis of computer performance data. In: Cohen PR, Adams NM, Berthold MR, editors. Advances in Intelligent Data Analysis IX. Lecture Notes in Computer Science, vol 6065. Berlin, Heidelberg: Springer 2010;6065:18–29. https://doi.org/10.1007/978-3-642-13062-5_4
17. Vidyasagar M. Modeling extreme events using heavy-tailed distributions [internet]. fusion methodologies in crisis management. Springer

International Publishing. 2016; 455–65. https://doi.org/10.1007/978-3-319-22527-2_21

18. Microsoft. Performance Counters [Internet]. Microsoft; [cited 2025 May 20]. Available from: https://learn.microsoft.com/en-us/windows/win32/perfctrs/performance-counters-portal
19. Microsoft. Windows Performance Monitor [Internet]. Microsoft; [cited 2025 May 20]. Available from: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-r2-and-2008/cc749249(v=ws.11)
20. Microsoft. Counter Types [Internet]. Microsoft; [cited 2025 May 20]. Available from: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc785636(v=ws.10)
21. Microsoft. PERF_COUNTER_RAWCOUNT [Internet]. Microsoft; [cited 2025 May 20]. Available from: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc757032(v=ws.10)
22. Huffman C. Performance monitor. Elsevier eBooks. 2014; 11–56. https://doi.org/10.1016/b978-0-12-416701-8.00002-8
23. Microsoft. Memory Object [Internet]. Microsoft; [cited 2025 May 20]. Available from: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc778082(v=ws.10)
24. Kowal B, Strzalka D. Cache Memory Bytes – Windows performance counter (Version 1). figshare. 2025. https://doi.org/10.6084/m9.figshare.29589116.v1
25. Racine JS. Nonparametric Econometrics: A Primer. FNT in Econometrics. 2007;3(1):1–88. https://doi.org/10.1561/0800000009
26. Soh Y, Hae Y, Mehmood A, Hadi Ashraf R, Kim I. Performance Evaluation of Various Functions for Kernel Density Estimation. OJAppS. 2013;3(1):58–64. https://doi.org/10.4236/ojapps.2013.31B012
27. Kharrat T, Boshnakov GN. StableEstim: Estimate the Four Parameters of Stable Laws using Different Methods [Internet]. 2014. https://doi.org/10.32614/cran.package.stableestim
28. Hofert M, Hornik K, McNeil AJ. qrmtools: Tools for Quantitative Risk Management [Internet]. 2015. https://doi.org/10.32614/cran.package.qrmtools
29. Ossberger J. tea: Threshold Estimation Approaches [Internet]. 2017. https://doi.org/10.32614/cran.package.tea
30. Caeiro J, Gomes M I. Threshold selection in extreme value analysis. in extreme value modeling and risk analysis. Chapman and Hall/CRC 2016. 2016; 89–106. https://doi.org/10.1201/b19721-8
31. Terdik G, Gyires T. Lévy flights and fractal modeling of internet traffic. IEEE/ACM Trans Networking. 2009 Feb;17(1):120–129. https://doi.org/10.1109/tnet.2008.925630
32. Nolan JP. Univariate stable distributions. springer series in operations research and financial engineering. Springer International Publishing. 2020. https://doi.org/10.1007/978-3-030-52915-4