The Westermo test system performance data set

Per Erik Strandberg and Yosh Marklund

Westermo Network Technologies AB, Västerås, Sweden

November 2023

Abstract

There is a growing body of knowledge in the computer science, software engineering, software testing and software test automation disciplines. However, a challenge for researchers is to evaluate their research findings, ideas and tools due to lack of realistic data. This paper presents the Westermo test system performance data set. More than twenty performance metrics such as CPU and memory usage sampled twice per minute for a month on nineteen test systems driving nightly testing of cyber-physical systems has been anonymized and released. The industrial motivation is to spur work on anomaly detection in seasonal data such that one may increase trust in nightly testing. One could ask: If the test system is in an abnormal state – can we trust the test results? How could one automate the detection of abnormal states? The data set has previously been used by students and in hackathons. By releasing it we hope to simplify experiments on anomaly detection based on rules, thresholds, statistics, machine learning or artificial intelligence, perhaps while incorporating seasonality. We also hope that the data set could lead to findings in sustainable sofware engineering.

Keywords: software engineering, anomaly detection, open data, nightly testing, data visualization, seasonal data, embedded systems, cyber-physical systems, sustainable software engineering.

1 Specifications

Subject: Computer Science – Embedded Systems, and Software Engineering

Specific subject area: Performance metrics from servers in test systems.

Type and format of data: Tabular data in 19 CSV files, each containing 23 or 24 time series sampled about 86 thousand times. Total size is about 360 MB.

Data collection process: Performance data were acquired from servers using node exporter, stored with grafana and then exported to CSV.

Data accessibility: Available at GitHub: https://github.com/westermo/test-system-performance-dataset

Related research article: P E Strandberg. (2021). *Automated System-Level Software Testing of Industrial Networked Embedded Systems,* PhD Thesis, Mälardalen University. ISBN: 978-91-7485-529-6. [4]

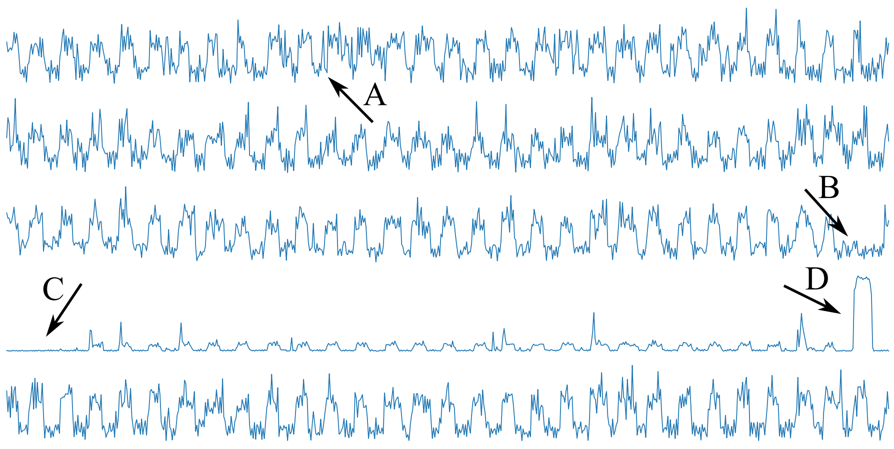


Figure 1: The metric load-15m on five systems (averaged for smoother curves).

2 Introduction

There is a growing body of knowledge in the computer science, software engineering, software testing and software test automation disciplines. However, there is a challenge for researchers to evaluate their research findings, innovations and tools due to lack of realistic data. This paper presents the Westermo test system performance data set. This is a data set with performance metrics from test systems that drive nightly testing at Westermo. In Figure 1 one such metric is visualized from five test systems. As can be seen, the signal has obvious peaks during the nights, when the test systems are working, and periods of inactivity during the days when humans might use the test systems for debugging and development.

Westermo develops cyber-physical systems for industrial communication such as robust switches and routers. These can be used in energy distribution, on-board rail, track-side rail, or for industry automation. A typical product supports many network communication protocols. In addition to robust hardware, the devices run an operating system based on GNU and Linux with the addition of proprietary code and third party code. This software is developed and maintained by several teams of software developers, and to ensure it has quality, Westermo has invested heavily on software test automation. One way of testing is automated functional testing where a test framework configures a test system, enables various protocols, sends certain stimuli and observes the effect such that it may give a pass or fail verdict – thereby answering ”does this software seem to work as expected?” See Figure 2 for an example of one such test system. The test framework runs on a server running Ubuntu GNU/Linux (not shown in the figure).

If the test framework does not give the verdict pass after a test case, this could have many reasons. By design, the desired cause is that the software under test has a problem of some kind. However, with a large and complex test framework, it could be the case that the problem is instead located in the code of the test framework. Furthermore, the root cause could be located in the hardware setup, e.g., if a cable has been connected in the wrong port, or a peripheral device was left without power, etc. Finally, it could be the cause that the problem comes from the server driving the nightly testing – a trivial cause could be a full disk. In order to learn more about automated detection of such anomalies we are releasing this data set.

The release of this data set follows the release of two previous data sets: the Westermo test results data set [5] and the Westermo network traffic data set [6]. Both these data sets have led to progress in knowledge in the form of validated ideas, algorithms and tools.

The Westermo test system performance data set was created during the AIDOaRt project [2] to explore anomaly detection. Figure 1 illustrates the load-15m performance metric over time for five test systems. Potential anomalies that a system could warn for include: (A) Did nightly testing not stop (the resting time of the test system is small)? (B) Did nightly testing not start (there seems to be a skipped heartbeat)? (C) Was this system under construction or not in use for three days (the curve looks flat)? (D) Is this a sign of

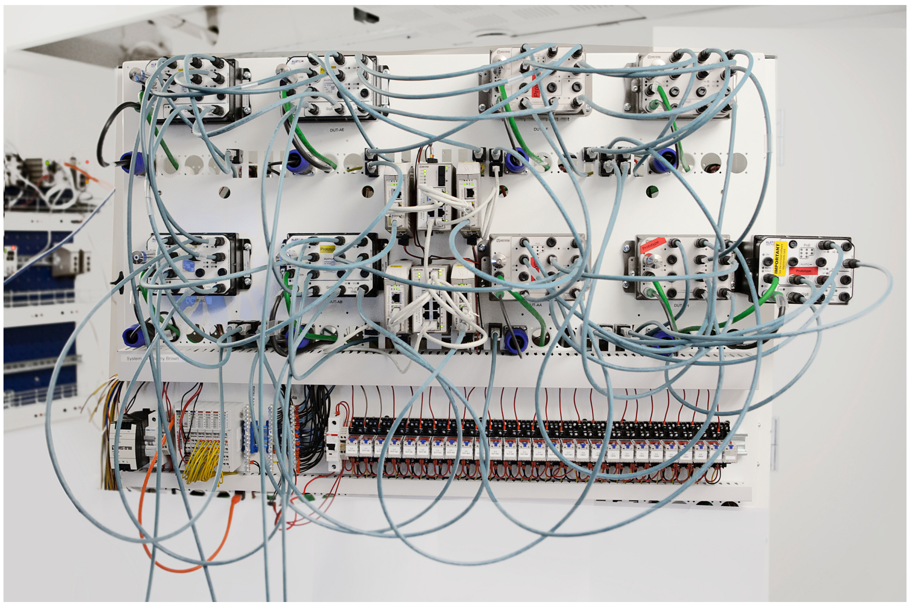


Figure 2: A Westermo test system with a network topology built up of switches, routers, other peripheral equipment, as well as a server (not shown).

abnormal load (a peak much higher than normal peaks)?

The dataset has been used in AIDOaRt hackathons as well as by a summer intern with the goal of replicating work done in a master's thesis [3] where isolation forest was one method used to detect outliers. Currently it is being used by a group of students in a project course in software engineering. These students will experiment with simulating test systems. First, the simulated systems will replay real data, but some additional test systems will be simulated to create abnormal data in order to trigger anomaly detection. The data produced will be stored in a database, such that a human operator could visualize various trends. We anticipate that visualizations of time series from many parallel metrics on many parallel test systems is non-trivial (in particular since tools like MS Excel rapidly crashes for even relatively trivial tasks and visualizations when exploring data from just one test system). Finally, the students will develop an anomaly detection toolkit could be developed to run in batches or in real time.

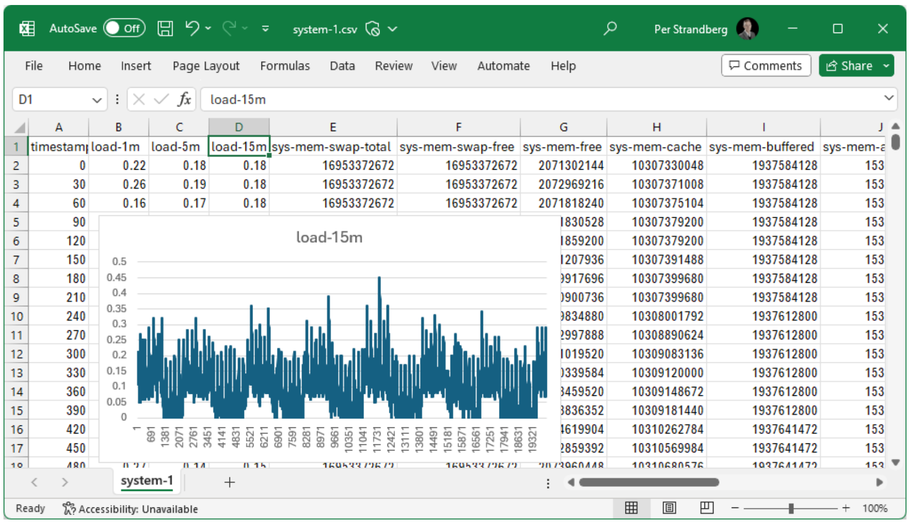
In addition, the data set could be used to work on sustainable software engineering [1]. One could ask: what is the carbon footprint of running nightly testing? How much waste do we get in terms of energy overuse, test results that cannot be trusted or human effort lost to debugging? Perhaps this data set could be used to shed light on some of these topics?

Users of the data could explore research questions on how to best define or combine anomaly detection based on rules, thresholds, statistics, machine learning or artificial intelligence, perhaps while incorporating seasonality[1]. In short, students, researchers and practitioners in the field of software engineering, statistics, software test automation, graphic design or artificial intelligence can benefit from the data. It can be used to evaluate algorithms, tools or visualizations for improved validity and generalizability.

3 Data description

The data is stored in 19 CSV files, one per test system. These represent tables of data. Each file starts with a header of names of metrics. Each line then contains values. The first column contains the timestamp as seconds into the data collection. There are almost 86 thousand lines per file. Figure 3 illustrates the

[1]https://en.wikipedia.org/wiki/Seasonality



	A	B	C	D	E	F	G	H	I	J
1	timestamp	load-1m	load-5m	load-15m	sys-mem-swap-total	sys-mem-swap-free	sys-mem-free	sys-mem-cache	sys-mem-buffered	sys-mem-a
2	0	0.22	0.18	0.18	16953372672	16953372672	2071302144	10307330048	1937584128	153
3	30	0.26	0.19	0.18	16953372672	16953372672	2072969216	10307371008	1937584128	153
4	60	0.16	0.17	0.18	16953372672	16953372672	2071818240	10307375104	1937584128	153
5	90						1830528	10307379200	1937584128	153
6	120						1859200	10307379200	1937584128	153
7	150						1207936	10307391488	1937584128	153
8	180						9917696	10307399680	1937584128	153
9	210						0900736	10307399680	1937584128	153
10	240						9834880	10308001792	1937612800	153
11	270						2997888	10308890624	1937612800	153
12	300						1019520	10309083136	1937612800	153
13	330						0339584	10309120000	1937612800	153
14	360						3459520	10309148672	1937612800	153
15	390						3836352	10309181440	1937612800	153
16	420						4619904	10310262784	1937641472	153
17	450						2859392	10310569984	1937641472	153

Figure 3: One CSV files loaded into MS Excel. The plot visualizes the first 20 thousand values for the load-15m metric. The plot represents about 0.05% of the data set.

first rows and columns of one such file. In the remainder of this section we describe the meaning of the columns.

Timestamp

A timestamp is a sequence of characters or encoded information identifying when a certain event occurred... They can have any epoch, can be relative to any arbitrary time, such as the power-on time of a system, or to some arbitrary time in the past. –Wikipedia on Timestamp[2]

Metric:

1. timestamp: Number of seconds since first data was collected.

Load

In UNIX computing, the system load is a measure of the amount of computational work that a computer system performs. The load average represents the average system load over a period of time. –Wikipedia on System load[3]

Metrics:

2. load-1m: System load over the last 1 minute.
3. load-5m: System load over the last 5 minutes.
4. load-15m: System load over the last 15 minutes.

Memory

Computer memory stores information, such as data and programs for immediate use in the computer... Besides storing opened programs, computer memory serves as disk cache and write buffer to improve both reading and writing performance. Operating systems borrow RAM capacity for caching so long as not needed by running software. –Wikipedia on Computer memory[4]

[2]https://en.wikipedia.org/wiki/Timestamp
[3]https://en.wikipedia.org/wiki/Load_(computing)
[4]https://en.wikipedia.org/wiki/Computer_memory

In computer operating systems, memory paging (or swapping…) is a memory management scheme by which a computer stores and retrieves data from secondary storage for use in main memory. –Wikipedia on Memory paging[5]

Metrics (in bytes):

5. sys-mem-swap-total: Swap size (constant).
6. sys-mem-swap-free: Available swap.
7. sys-mem-free: Unused memory.
8. sys-mem-cache: Memory used for cache.
9. sys-mem-buffered: Memory used by kernel buffers.
10. sys-mem-available: Memory available to be allocated (free and cache).
11. sys-mem-total: Total size of memory (constant).

CPU

A central processing unit (CPU)... is the most important processor in a given computer. Its electronic circuitry executes instructions of a computer program, such as arithmetic, logic, controlling, and input/output (I/O) operations. -Wikipedia on Central processing unit[6]

Metrics:

12. cpu-iowait: Summarized rate of change of seconds spent on waiting for I/O.
13. cpu-system: Summarized rate of change of seconds spent on kernel space threads.
14. cpu-user: Summarized rate of change of seconds spent on user space processes and threads.

Disk/Storage

These are metrics related to reading and writing (input and output) to and from a persistent storage (a disk).

Metrics

15. disk-io-time: Rate of change in time spent on storage i/o operations.
16. disk-bytes-read: Rate of change in bytes read.
17. disk-bytes-written: Rate of change in bytes written.
18. disk-io-read: Rate of change in amount of read operations.
19. disk-io-write: Rate of change in amount of write operations.

System

In computing, ... fork is an operation whereby a process creates a copy of itself... Fork is the primary method of process creation on Unix-like operating systems. –Wikipedia on fork (system call)[7]

Metrics

20. sys-fork-rate: Rate of change in number of forks.
21. sys-interrupt-rate: Rate of change of interrupts.
22. sys-context-switch-rate: Rate of change of context switches.

[5]https://en.wikipedia.org/wiki/Memory_paging
[6]https://en.wikipedia.org/wiki/Central_processing_unit
[7]https://en.wikipedia.org/wiki/Fork_(system_call)

Thermal

Temperature is a physical quantity that expresses quantitatively the attribute of hotness... The Celsius scale... is used... in most of the world...–Wikipedia on Temperature[8]

Metric

23. sys-thermal: Average rate of change in measured system temperature (Celsius). Please note that not all systems have access to a thermometer, so not all systems have this metric.

Server up

. . . a heartbeat is a periodic signal generated by hardware or software to indicate normal operation or to synchronize other parts of a computer system. . . Heartbeat messages are typically sent non-stop. . . When the destination identifies a lack of heartbeat messages. . . the destination may determine that the originator has failed. . . –Wikipedia on Heartbeat (computing)[9]

Metric

24. server-up: freshness check of reporting server, values above 0 indicates that the server is available.

4 Experimental design, materials and methods

The servers in the test systems run node exporter[10], a tool for exporting performance data from servers or other PCs. Data was exported and stored using grafana[11]. With a Python script, data was exported and slightly cleaned into CSV files.

The servers are located in the software testing laboratory of Westermo Network Technologies AB, in Västerås, Sweden.

5 Ethics statements

To protect Westermo, data has been anonymized, obfuscated, and only a sub-set of available test systems has been used. The public release of data has been approved by staff responsible for information security at Westermo Network Technologies AB at risk workshops during 2023.

6 CRediT author statement

The authors of this paper has contributed as follows. Conceptualization: PES, Methodology: PES & YM, Software: YM, Validation: PES & YM, Investigation: PES & YM, Data Curation: YM, Writing - Original Draft: PES, Writing - Review & Editing: PES & YM, Visualization: PES, and Supervision: PES. Other Westermo staff has contributed with Resources and Software.

7 Acknowledgments

Work was funded by Westermo Network Technologies AB, and the AIDOaRt project, a European ECSEL Joint Undertaking (JU) under grant agreement No. 101007350.

[8]https://en.wikipedia.org/wiki/Temperature
[9]https://en.wikipedia.org/wiki/Heartbeat_(computing)
[10]https://github.com/prometheus/node_exporter
[11]https://grafana.com/

8 Declaration of interests

The authors of this paper are employed at Westermo Network Technologies AB.

9 License

Suggested attribution: P E Strandberg and Y Marklund. (2023). The Westermo test system performance data set. Retrieved from https://github.com/westermo/test-system-performance-dataset

References

[1] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, M. Mahaux, B. Penzenstadler, G. Rodriguez-Navas, C. Salinesi, N. Seyff, C. Venters, et al. The Karlskrona manifesto for sustainability design. *arXiv preprint arXiv:1410.6968*, 2014. Online at: `https://arxiv.org/abs/1410.6968`.

[2] R. Eramo, V. Muttillo, L. Berardinelli, H. Bruneliere, A. Gomez, A. Bagnato, A. Sadovykh, and A. Cicchetti. AIDOaRt: AI-augmented Automation for DevOps, a Model-based Framework for Continuous Development in Cyber-Physical Systems. In *Euromicro Conferenece on Digital Systems Design*, 2021.

[3] S. Salahshour Torshizi. Software performance anomaly detection through analysis of test data by multivariate techniques. Master's thesis, Uppsala University, 2022.

[4] P. E. Strandberg. *Automated System-Level Software Testing of Industrial Networked Embedded Systems*. PhD thesis, Mälardalen University, 2021. Online at: `https://arxiv.org/abs/2111.08312`.

[5] P. E. Strandberg. The Westermo test results data set. *arXiv preprint:2211.13622*, 2022. Online at: `https://github.com/westermo/test-results-dataset`.

[6] P. E. Strandberg, D. Söderman, A. Dehlaghi-Ghadim, M. Leon, T. Markovic, S. Punnekkat, M. H. Moghadam, and D. Buffoni. The Westermo network traffic data set. *Data in Brief*, 50:109512, 2023. Online at: `https://github.com/westermo/network-traffic-dataset`.