# Online detection of failures generated by storage simulator

# Online detection of failures generated by storage simulator

**Kenenbek Arzymatov, Mikhail Hushchyn, Andrey Sapronov, Vladislav Belavin, Leonid Gremyachikh, Maksim Karpov and Andrey Ustyuzhanin**

National Research University Higher School of Economics
11 Pokrovsky Boulevard, Moscow, Russia, 109028

E-mail: karzymatov@hse.ru

**Abstract.** Modern large-scale data-farms consist of hundreds of thousands of storage devices that span distributed infrastructure. Devices used in modern data centers (such as controllers, links, SSD- and HDD-disks) can fail due to hardware as well as software problems. Such failures or anomalies can be detected by monitoring the activity of components using machine learning techniques. In order to use these techniques, researchers need plenty of historical data of devices in normal and failure mode for training algorithms. In this work, we challenge two problems: 1) lack of storage data in the methods above by creating a simulator and 2) applying existing online algorithms that can faster detect a failure occurred in one of the components.

We created a Go-based (golang) package for simulating the behavior of modern storage infrastructure. The software is based on the discrete-event modeling paradigm and captures the structure and dynamics of high-level storage system building blocks. The package's flexible structure allows us to create a model of a real-world storage system with a configurable number of components. The primary area of interest is exploring the storage machine's behavior under stress testing or exploitation in the medium- or long-term for observing failures of its components.

To discover failures in the time series distribution generated by the simulator, we modified a change point detection algorithm that works in online mode. The goal of the change-point detection is to discover differences in time series distribution. This work describes an approach for failure detection in time series data based on direct density ratio estimation via binary classifiers.

## Introduction

Disk-drive is one of the crucial elements of any computer and IT infrastructure. Disk failures have a high contributing factor to outages of the overall computing system. During the last decades, the storage system's reliability and modeling is an active area of research in industry and academia works [1–3]. Nowadays, the rough total amount of hard disk drives (HDD) and solid-state drives (SSD) deployed in data-farms and cloud systems passed tens of millions of units [4]. Consequently, the importance of early identifying defects leading to failures that can happen in the future can result in significant benefits. Such failures or anomalies can be detected by monitoring components' activity using machine learning techniques, named change point detection [5–7]. To use these techniques, especially for anomaly detection, it is a necessity in historical data of devices in normal and failure mode for training algorithms. In this paper,

due to the reasons mentioned above, we challenge two problems: 1) lack of storage data in the methods above by creating a simulator and 2) applying new online algorithms that can faster detect a failure occurred in one of the components [8].

A Go-based (golang) package for simulating the behavior of modern storage infrastructure is created. The primary area of interest is exploring the storage machine's behavior under stress testing or exploitation in the medium- or long-term for observing failures of its components. The software is based on the discrete-event modeling paradigm and captures the structure and dynamics of high-level storage system building blocks. It represents the hybrid approach to modeling storage attached network [9, 10]. This method uses additional blocks with a neural network that tunes the internal model parameters while a simulation is running, described in [11]. This approach's critical advantage is a decreased requirement for detailed simulation and the number of modeled parameters of real-world system components and, as a result, a significant reduction in the intellectual cost of its development. The package's modular structure allows us to create a model of a real-word storage system with a configurable number of components. Compared to other techniques, parameter tuning does not require heavy-lifting changes within developing service [12].

To discover failures in the time series distribution generated by the simulator, we modified a change point detection algorithm that works in online mode. The goal of the change-point detection is to discover differences in time series distribution. This work uses an approach for failure detection in time series data based on direct density ratio estimation via binary classifiers [8].

## Simulator

*Internals*

The simulator uses a Discrete Event Simulation (DES) [13] paradigm for modeling storage infrastructure. In a broad sense, DES is used to simulate a system as a discrete sequence of events in time. Each event happens in a specific moment in time and traces a change of state in the system. Between two consecutive events, no altering in the system is presumed to happen; thus, the simulation time can directly move to the next event's occurrence time. The scheme of the process is shown in Figure 1.

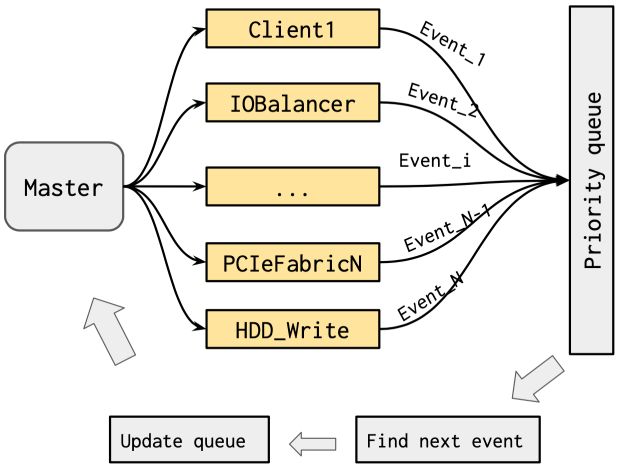


**Figure 1.** The event handling loop is the central part that responsible for time movement in the simulator. The Master process creates necessary logical processes (Client1, IOBalancer, HDD_Write, etc.) and populates a Priority Queue by collecting events from modeling processes. The last part of the implementation is running the event handling loop. It removes successive elements from the queue. That would be correct because we know that the queue is already time sorted and performed the associated actions.

The simulator's programming environment provides the functionality to set up a model for specific computing environments, especially storage area networks. The key site of interest is

**Table 1.** Resource description

| Resource | Real word entity | Parameters | Units | Anomaly type |
|---|---|---|---|---|
| CPU | Controller, server | Number of cores | Amount | Each component |
| | | Core speed | Flops | can suffer from |
| Link | Networking cables | Bandwidth | Megabyte/sec | performance **degradation** |
| | | Latency | Sec | or total **breakup** |
| Storage | Cache, SSD, HDD | Size | Gigabyte | |
| | | Write speed | Megabyte/sec | |
| | | Read speed | Megabyte/sec | |

exploring the storage infrastructure's behavior under various stress testing or utilization in the medium- or long-term for monitoring breakups of its components.

In the simulator, load to storage system can be represented by two action types: read file from disk and write file to disk. Each file has corresponding attributes, such as name, block size, and total size. With the current load, these attributes determine the amount of time required to perform the corresponding action. The three basic types of resources are provided: CPU, network interface, and storage. Their representation is shown in the Figure 3 and informative description is given in the Table 1. By using basic blocks, real-world systems can be constructed, as shown in the Figure 2.

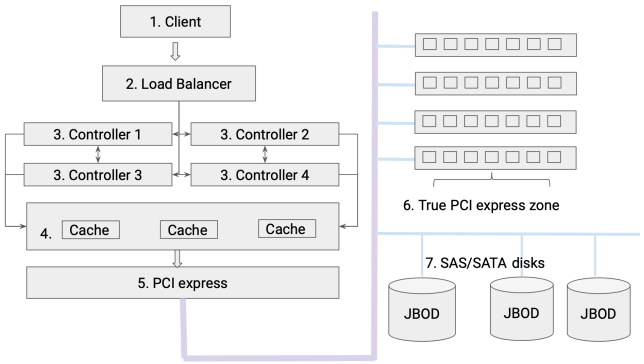


**Figure 2.** The example of the real storage system that can be modeled by using basic blocks

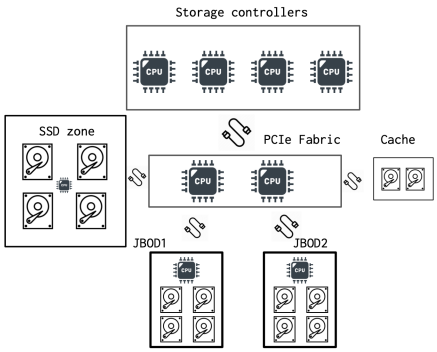


**Figure 3.** Basic resource entities in the simulator

*Comparison with the real data*

The data from the real-world storage system were used to validate the behavior of the simulator. A similar writing load scenario was generated on the model prototype, together with intentional controller failure (turn-off). The comparison is shown in the Figure 4. As we can see, the simulator's data can qualitatively reflect the components breakup.

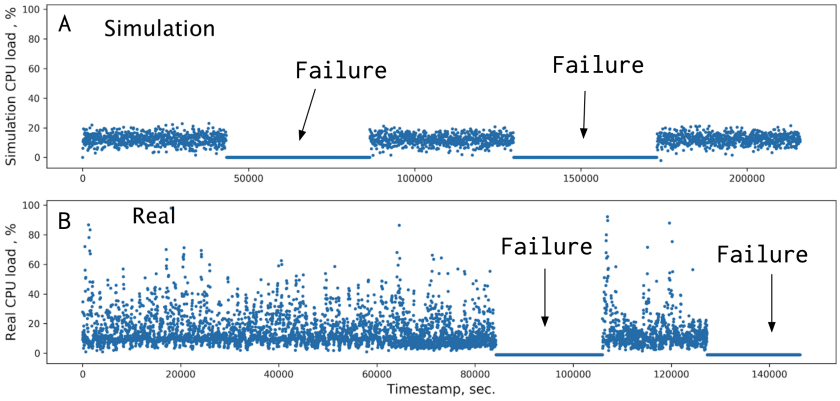


**Figure 4.** Comparison of the CPU load metrics between simulated (A) and real data (B). The periods marked 'Failure' correspond to a storage processor being offline

### Change point detection

Consider a $d$-dimensional time series that is described by a vector of observations $x(t) \in R^d$ at time $t$. Sequence of observations for time $t$ with length $k$ is defined as:

$$X(t) = [x(t)^T, x(t-1)^T, \ldots, x(t-k-1)^T]^T \in R^{kd}$$

Sample of sequences of size $n$ is defined as:

$$\mathcal{X}(t) = X(t), X(t-1), \ldots, X(t-n+1)$$

It is implied that observation distribution changes at time $t^*$. The goal is to detect this change. The idea is to estimate dissimilarity score between reference $X_{rf}(t-n)$ and test $X_{te}(t)$. The larger dissimilarity, the more likely the change point occurs at time $t-n$.

In this work, we apply a CPD algorithm based on direct density ratio estimation developed in [8]. The main idea is to estimate density ratio $w(X)$ between two probability distributions $P_{te}(X)$ and $P_{rf}(X)$ which correspond to test and reference sets accordingly. For estimating $w(X)$, different binary classifiers can be used, like decision trees, random forests, SVM, etc. We use neural networks for this purpose. This network $f(X, \theta)$ is trained on the mini-batches with cross-entropy loss function $L(\mathcal{X}(t-l), \mathcal{X}(t), \theta)$,

$$L(\mathcal{X}(t-l), \mathcal{X}(t), \theta) = -\frac{1}{n} \sum_{X \in \mathcal{X}(t-l)} \log(1 - f(X, \theta)) - \frac{1}{n} \sum_{X \in \mathcal{X}(t)} \log f(X, \theta),$$

We use a dissimilarity score based on the Kullback-Leibler divergence, $D(\mathcal{X}(t-l), \mathcal{X}(t))$. Following [14], we define this score as:

$$D(\mathcal{X}(t-l), \mathcal{X}(t), \theta) = \frac{1}{n} \sum_{X \in \mathcal{X}(t-l)} \log \frac{1 - f(X, \theta)}{f(X, \theta)} + \frac{1}{n} \sum_{X \in \mathcal{X}(t)} \log \frac{f(X, \theta)}{1 - f(X, \theta)}.$$

According to [8], the training algorithm is shown in Alg. 1. It consists of the following steps performing in the loop: 1) initializing hyper-parameters 2) preparing single datasets $\mathcal{X}'_{rf}$ and $\mathcal{X}'_{te}$ 3) calculating loss function $J$ 4) applying gradients to the weights of neural network.

**Algorithm 1:** Change-point detection algorithm.

**Inputs:** time series $\{X(t)\}_{t=k}^{T}$; $k$ – size of a combined vector $X(t)$; $n$ – size of a mini-batch $\mathcal{X}(t)$; $l$ – lag size and $n \ll l$; $f(X, \theta)$ – a neural network with weights $\theta$;
**Initialization:** $t \leftarrow k + n + l$;
**while** $t \leq T$ **do**
  take mini-batches $\mathcal{X}(t-l)$ and $\mathcal{X}(t)$;
  $d(t) \leftarrow D(\mathcal{X}(t-l), \mathcal{X}(t), \theta)$;
  $\bar{d}(t) \leftarrow \bar{d}(t-n) + \frac{1}{l}(d(t) - d(t-l-n))$;
  $loss(t, \theta) \leftarrow L(\mathcal{X}(t-l), \mathcal{X}(t), \theta)$;
  $\theta \leftarrow \text{Optimizer}(\text{loss}(t, \theta))$;
  $t \leftarrow t + n$;
**end**
**return** $\{\bar{d}(t)\}_{t=1}^{T}$ – change-point detection score

## Results

To check the change-point algorithm against the simulation data, four time-series datasets were prepared: 1) controller's CPU load metric 2) load balancer request time 3) data traffic to storage devices and 4) differences change of used space. Their time-series are shown on the upper halves of Figures 5, 6, 7 and 8.

As shown in the bottom halves of the figures above, the algorithm can identify data points where distribution changes. A red line on each plot is a CPD score line. The higher values it has, the more confident algorithm about a change point occurred at this timestamp.

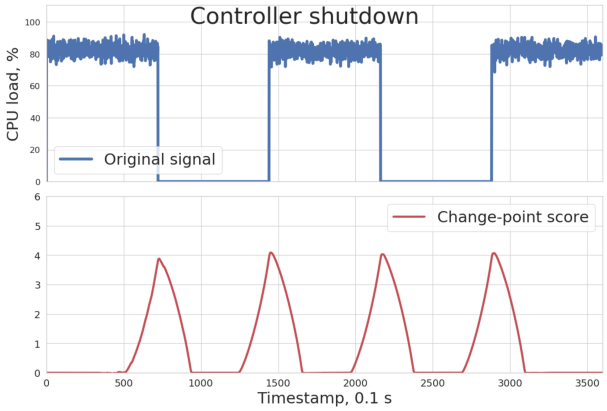


**Figure 5.** Controller failure

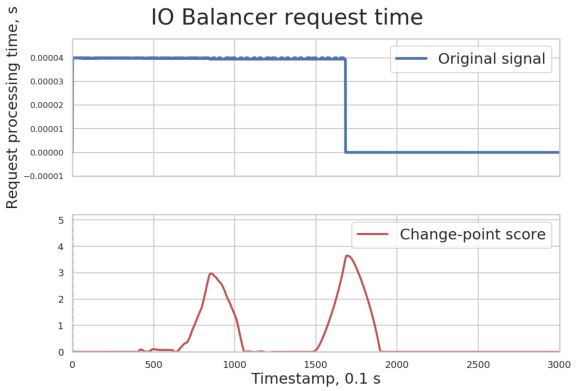


**Figure 6.** IO balancer time series

## Conclusion

The simulator for modeling storage infrastructure based on the event-driven paradigm was presented. It allows researchers to try different I/O load scenarios to test disk performance and model failures of its hardware components. By providing large amounts of synthetic data of anomalies and time series of a machine in various modes, the simulator can also be used as a benchmark for comparing different change-point detection algorithms. In this work, the density ratio estimation CPD algorithm were successfully applied to the simulator data.


**Figure 7.** Storage traffic

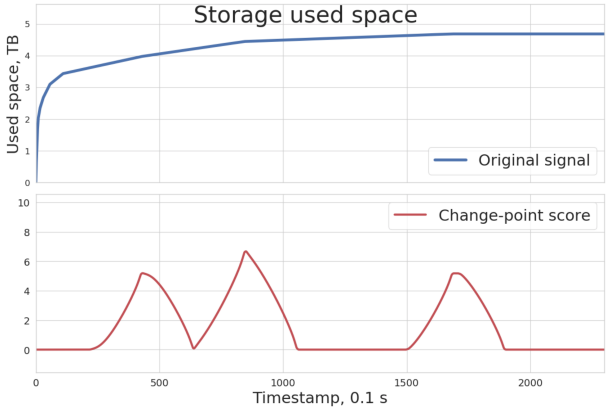


**Figure 8.** Storage used space

## Acknowledgments

This research was supported in part through computational resources of HPC facilities at NRU HSE.

## References

[1] Yang J and Feng-Bin Sun 1999 A comprehensive review of hard-disk drive reliability *Annual Reliability and Maintainability. Symposium. 1999 Proceedings (Cat. No.99CH36283)* pp 403–409
[2] Strom B D, Lee S, Tyndall G W and Khurshudov A 2007 *IEEE Transactions on Magnetics* **43** 3676–3684
[3] Elerath J G 2000 Specifying reliability in the disk drive industry: No more mtbf's *Annual Reliability and Maintainability Symposium. 2000 Proceedings. International Symposium on Product Quality and Integrity (Cat. No.00CH37055)* pp 194–199
[4] Ganguly S, Consul A, Khan A, Bussone B, Richards J and Miguel A 2016 A practical approach to hard disk failure prediction in cloud platforms: Big data model for failure management in datacenters *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)* pp 105–116
[5] Aminikhanghahi S and Cook D J 2017 *Knowledge and information systems* **51** 339–367
[6] Kawahara Y and Sugiyama M 2009 *Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation* pp 389–400 (*Preprint* `https://epubs.siam.org/doi/pdf/10.1137/1.9781611972795.34`) URL `https://epubs.siam.org/doi/abs/10.1137/1.9781611972795.34`
[7] Liu S, Yamada M, Collier N and Sugiyama M 2013 *Neural Networks* **43** 72 – 83 ISSN 0893-6080 URL `http://www.sciencedirect.com/science/article/pii/S0893608013000270`
[8] Hushchyn M, Arzymatov K and Derkach D 2020 Online neural networks for change-point detection (*Preprint* `2010.01388`)
[9] Karpov M, Arzymatov K, Belavin V, Sapronov A, Ustyuzhanin A and Nevolin A 2018 *International Journal of Civil Engineering and Technology* **9** 220–226
[10] Arzymatov K, Sapronov A, Belavin V, Gremyachikh L, Karpov M, Ustyuzhanin A, Tchoub I and Ikoev A 2020 *PeerJ Computer Science* **6** e271 ISSN 2376-5992 URL `https://doi.org/10.7717/peerj-cs.271`
[11] Belavin V, Sapronov A, Arzymatov K, Karpov M, Nevolin A and Ustyuzhanin A 2018 *Advances in Systems Science and Applications* **18(4)** 1–12
[12] Mousavi S S, Schukat M and Howley E 2017 *Lecture Notes in Networks and Systems* 426–440 ISSN 2367-3389 URL `http://dx.doi.org/10.1007/978-3-319-56991-832`
[13] Fishman G S 1978 *Principles of discrete event simulation* Wiley series on systems engineering and analysis (New York: Wiley) ISBN 9780471043959
[14] Hushchyn M and Ustyuzhanin A 2020 Generalization of change-point detection in time series data based on direct density ratio estimation (*Preprint* `2001.06386`)