SAND2022-5205C

Scale-out Edge Storage Systems with Embedded Storage Nodes to Get Better Availability and Cost-Efficiency At the Same Time

Jianshen Liu
UC Santa Cruz

Matthew Leon Curry
Sandia National Laboratories

Carlos Maltzahn
UC Santa Cruz

Philip Kufeldt
Seagate Technology

Abstract

In the resource-rich environment of data centers most failures can quickly failover to redundant resources. In contrast, failure in edge infrastructures with limited resources might require maintenance personnel to drive to the location in order to fix the problem. The operational cost of these“truck rolls” to locations at the edge infrastructure competes with the operational cost incurred by extra space and power needed for redundant resources at the edge. Computational storage devices with network interfaces can act as network-attached storage servers and offer a new design point for storage systems at the edge. In this paper we hypothesize that a system consisting of a larger number of such small “embedded” storage nodes provides higher availability due to a larger number of failure domains while also saving operational cost in terms of space and power. As evidence for our hypothesis, we compared the possibility of data loss between two different types of storage systems: one is constructed with general-purpose servers, and the other one is constructed with embedded storage nodes. Our results show that the storage system constructed with general-purpose servers has 7 to 20 times higher risk of losing data over the storage system constructed with embedded storage devices. We also compare the two alternatives in terms of power and space using the Media-Based Work Unit (MBWU) that we developed in an earlier paper as a reference point.

1 Introduction

While the concept of the edge is not new [13, 14, 20], until recently edge devices have become a key driver of the growth of the global datasphere [12] as the number of connected edge devices skyrockets, and is projected to reach 43 billion by 2023 [5]. By 2025, it is estimated that 75% of data will be created and processed outside the cloud [16].

Though the trend is deploying more and more infrastructures at the edge to handle the increasing number of requests from edge devices, maintaining the large numbers of edge infrastructures becomes challenging. Failures in edge infrastructures might require maintenance personnel at remote sites to fix the problem. The operational cost of these “truck rolls” can quickly overwhelm the capital cost of redundant resources at the edge, and is estimated to be more than one thousand dollars per event [22]. On the other hand, unlike central data centers, edge data centers are likely to be restricted by environmental factors such as space and power, the stability of network connections, and temperature [2, 9]. These factors make the expense of provisioning and operating redundant resources at the edge competing with the cost of truck rolls.

In this paper, we focus on edge storage systems and investigate the benefits in data availability in addition to cost-efficiency of using embedded storage nodes to build edge storage systems. Embedded storage nodes encapsulate computing resources and storage media in a small form factor. We elaborate on the rationale of the benefits resulting from using embedded storage nodes as follows:

A. Improvement of data availability: Just like putting all your eggs in one basket is risky, for failover mechanism of a system, the more independent failure domains the mechanism spans, the higher availability the data stored in the system can be. A storage node is a failure domain since failures of critical components of a node, like CPU and DRAM, will cause inaccessibility to all the data hosted by that node. Therefore, a good failover mechanism should place redundant data on independent storage nodes. For example, a failure mechanism using data replication should place replicas of a data item on storage devices of different servers. More importantly, the less complex a failure domain is (i.e., the fewer number of disks attached to a node), the more reliable that failure domain becomes.

However, as discussed previously, edge data centers may suffer from environmental restrictions, using a large form factor of storage nodes, like general-purpose storage servers, could limit the number of failure domains an edge storage system can have. Using embedded storage nodes, on the one hand, makes it possible to have a larger number of nodes deployed at the edge under the same space restriction. On the other hand, the simpler system design of an embedded storage node makes each node more affordable, and thus more

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

nodes can be deployed under the same cost restriction. Finally, building storage systems using embedded storage nodes is a scale-out solution compared to using general-purpose storage servers, because fewer storage devices are co-located in the same failure domain. The availability benefit from scaling out a storage system has been shown in distributed database systems [6]. In general, using embedded storage nodes for edge storage systems can improve data availability by having a larger number of efficiently sized failure domains. In this paper, we will focus on validating this benefit.

B. Improvement of cost-efficiency: Embedded storage nodes are more power-efficient and space-efficient. The breakdown of Dennard scaling [19] starting around 2005 indicated that to continue improving the computing performance of processors, we need to input more power to the circuits than what we expected before. Furthermore, extra power causes more heat, requiring more space for heat dissipation. Embedded storage nodes have moderate computing power and a relatively simple system design, together contributing to better power-efficiency and space-efficiency. To show the benefits in these efficiencies, we used the Media-Based Work Unit (MBWU) [10] as a reference point to compare a general-purpose platform with an embedded platform regarding the performance of running a key-value store. Our results reveal that the embedded platform is 45.9% higher in power-efficiency and 79.7% higher in space-efficiency than the general-purpose platform. Details of the comparison and the related system configurations can be found in [10].

The contribution of this paper is an analysis of the data availability provided by using embedded storage nodes for edge storage systems in comparison to general-purpose servers. The rest of the paper is organized as follows: §2 introduces the analytical model and its assumptions of system configurations and model parameters for comparing the possibility of data loss between between two types of storage systems: general-purpose server-based storage systems and embedded storage node-based storage systems (Figure 1). §3 shows the evaluation results using our model with different parameters including different types of storage devices. Finally, §4 describes related work focused on data availability and the benefit of using embedded storage nodes.

2 The Analytical Model

Comparing a general-purpose server with an embedded storage node is challenging because they are built with components that are greatly different. To make the comparison more tractable, we carefully chose simplifying assumptions about system configurations and model parameters without impacting the generality of our results.

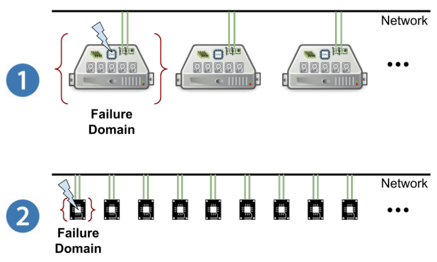

Figure 1: Storage Systems Constructed with Different Building Blocks: the first system uses general-purpose servers and the second system uses embedded storage devices

2.1 Assumptions of System Configurations

The assumptions of system configurations and our reasoning are as follows.

The units of deployment are homogeneous. We assume that all the servers have the same configuration for the storage system constructed with general-purpose servers (e.g., the same number and type of CPU cores, the same amount and type of DRAM, and the same number and model of block storage devices). Similarly, for storage systems constructed with embedded storage devices, we assume that all the devices are the same model. With this assumption, we can have a consistent failure rate for components of the same type belonging to the same type of deployment units. For example, all CPUs of the servers will have the same failure rate.

Both systems have the same level of network redundancy and power redundancy for all nodes. This allows us to omit network and power failure rates in our analysis.

Both systems use 3-way replication for data protection. There are other data redundancy techniques such as erasure coding. In this analysis we choose to focus on data replication and leave the analysis with other redundancy techniques to future work. We could increase the replication factor from 3 to 4 or even higher. However, as studied in [4], increasing the factor to 4 does not offer much difference in terms of the probability of data loss for the scale of nodes deployed at the edge. On the other hand, given the space restriction of the edge, employing an even higher level of replication would require more nodes within the space limitations of edge deployments.

Both systems use the copyset replication scheme [4] instead of the random replication scheme. Random replication is a simple technique used in many production storage systems like Hadoop Distributed File System (HDFS), Google File System (GFS), and Windows Azure. It can protect data against uncorrelated failures such as individual server or disk failures. However, with a sufficient number of data chunks stored, random replication stores replicas on any combination of k nodes, where k is the replication factor. Thus, even though every single data chunk is stored k times, random replication in the limit creates a virtual failure domain for any combination of k nodes. In other words, data loss is likely no matter which k nodes fail simultaneously. The copyset replication scheme reduces the possibility of data loss by limiting the number of combinations of k nodes (or "copysets") that share replicas, thereby reducing the probability that any combination of k node failures will lose data.

Independence of servers and storage devices. We also assume that the failures of different servers are independent, and the failures of different storage devices are also independent. Therefore, we can use Poisson distribution [21] to model the possibilities of hardware failures. Finally, we assume a general-purpose server hosts multiple block storage devices, and an embedded storage device consists of a single block storage device and some computing resources.

2.2 Assumptions of Model Parameters

We list the symbols defined for our analytical model in Table 1. The assumptions of the parameters are detailed below:

- $R_m = R_m^{'}$ and $R_d = R_d^{'}$. We assume that general-purpose servers and embedded storage devices have the same failure rate for their computing resources. Though the design of a general-purpose server is more complex than an embedded storage device, which might indicate that the failure rate of a general-purpose server is higher, we use this assumption that they are equal so that we can generate a conservative result from the comparison. In fact, even if $R_m^{'}$ is five times the value of R_m, in the case of $n = 4$, our model shows that the possibility of data loss of the embedded storage node-based system is still lower than that of the general-purpose server-based system. Similarly, we have the same assumption on the storage components from two deployment units.

- $R_d = f \cdot R_m$, where $f > 0$. This expresses the ratio of failures between computing resources and a storage component. For hard drives, f could be greater than 2, while for solid-state drives, f could be less than 1. We call f **the ratio of failure rates**.

- $m^{'} = c \cdot m$, where $c >= 1$. An embedded storage device could be less powerful than a general-purpose server. In this case, we may need multiple embedded storage devices to achieve a similar performance provided by a server. We call c **the ratio of computing performance**.

- $n >= 2$. We assume that a server will host multiple block storage devices. Specifically, we want at least two storage devices per server. We call n **the ratio of storage performance**.

- $m >= 3$. Since we use 3-way replication for data protection, we need at least three failure domains, which are equivalent to three servers.

Table 1: List of Model Parameters

Name	Description
m	the number of servers in the storage system
$m^{'}$	the number of embedded storage devices in the storage system
n	the number of storage devices in a server
R_m	the failure rate of a server excluding the storage components
R_d	the failure rate of a block storage device in a server
$R_m^{'}$	the failure rate of an embedded storage device excluding the storage component
$R_d^{'}$	the failure rate of the storage component in an embedded storage device
w	the scatter width of the copyset replication

We use "m" to stands for a "machine" and "d" for a "device" in the following notations: R_m, R_d, $R_m^{'}$, $R_d^{'}$.

2.3 Modeling the Two Systems

Scatter width defines the number of nodes the data on a node can be replicated to. According to the setup of the replication factor and the scatter width, the total numbers of copysets of the general-purpose server-based storage system and the embedded storage device-based storage system are $l_{gp} = \frac{wm}{6}$ and $l_{es} = \frac{wm^{'}}{6}$, respectively (see section 3.2 in [4]). For simplicity, we also assume that data will be replicated to storage devices of the same index. For example, if $\{1,4,7\}$ is a copyset, then the data in disk 1 of node 1 will be replicated to disk 1 of node 4 and disk 1 of node 7. We could have a different device mapping for replication, but it does not affect the result from the model.

For the storage system constructed with general-purpose servers, the event of data loss could be caused by one of the following three situations: (i) Failures of multiple servers: among these servers at least three fall in the same copyset. (ii) Failures of multiple storage devices: among these storage devices at least three whose hosts are in the same copyset, and the three devices have the same device index. (iii) A combination of failures with the number of failures ≥ 3: some failures in the combination cause data loss.

First of all, since the failures of servers are independent, we can express the possibility of failures involving exactly k servers by applying the probability mass function of the Poisson distribution:

$$P(\text{failures of } k \text{ servers}) = \frac{{R_m}^k e^{-R_m}}{k!} \quad (1)$$

Similarly, the possibility of failures involving exactly j storage devices is:

$$P(\text{failures of } j \text{ storage devices}) = \frac{R_d{}^j e^{-R_d}}{j!} \quad (2)$$

We can then express the possibilities of situations mentioned above that cause data loss, respectively, as follows

(i) $$P_m(k) = P(\text{failures of } k \text{ servers}) \times \frac{N_m(k)}{\binom{m}{k}} \quad (3)$$

(ii) $$P_d(j) = P(\text{failures of } j \text{ storage devices}) \times \frac{N_d(j)}{\binom{mn}{j}} \quad (4)$$

(iii) $$\begin{aligned} P_{m,d}(k,j) = P&(\text{failures of } k \text{ servers}) \\ &\times P(\text{failures of } j \text{ storage devices}) \\ &\times \frac{N_{m,d}(k,j)}{\binom{m}{k} \times \binom{mn}{j}} \end{aligned} \quad (5)$$

In equation 3, $N_m(k)$ is the number of k-combinations of servers, requiring that among each combination at least three servers fall in the same copyset. In equation 4, $N_d(j)$ is the number of j-combinations of all block storage devices, requiring that each combination contains at least three devices whose hosts are in the same copyset, and the three devices have the same device index. Finally, in equation 5, $N_{m,d}(k,j)$ is the number of combinations each of which contains failures of k servers and j storage devices, requiring that at least three failures in each combination are associated with a copyset. Specifically, for a specific combination, there exists a copyset, such that one failed server in the combination is within the copyset, and there are two failed storage devices in the combination whose device indexes are the same, and their hosts are also in the copyset. Or it could also be that two failed servers in the combination are in the copyset, and there is one failed storage device in the combination whose host also belongs to the copyset.

By adding up the possibilities of different cases, we can get the possibility of data loss of the storage system constructed with general-purpose servers:

$$\begin{aligned} P_{gp} = &\sum_{k=3}^{m} P_m(k) + \sum_{j=3}^{mn} P_d(j) \\ &+ \sum_{k=2}^{m} \sum_{j=1}^{mn} P_{m,d}(k,j) + \sum_{j=2}^{mn} P_{m,d}(1,j) \end{aligned} \quad (6)$$

Similarly, the possibility of data loss for the storage system constructed with embedded storage devices is:

$$\begin{aligned} P_{es} = &\sum_{k=3}^{m'} P'_m(k) + \sum_{j=3}^{m'} P'_d(j) \\ &+ \sum_{k=2}^{m'} \sum_{j=1}^{m'} P'_{m,d}(k,j) + \sum_{j=2}^{m'} P'_{m,d}(1,j) \end{aligned} \quad (7)$$

where

$$P'_m(k) = \frac{R'_m{}^k e^{-R'_m}}{k!} \times \frac{N'_m(k)}{\binom{m'}{k}} \quad (8)$$

$$P'_d(j) = \frac{R'_d{}^j e^{-R'_d}}{j!} \times \frac{N'_d(j)}{\binom{m'}{j}} \quad (9)$$

$$P'_{m,d}(k,j) = \frac{R'_m{}^k e^{-R'_m}}{k!} \times \frac{R'_d{}^j e^{-R'_d}}{j!} \times \frac{N'_{m,d}(k,j)}{\binom{m'}{k} \times \binom{m'}{j}} \quad (10)$$

Finally, to compare the possibility of data loss between the two storage systems, we can evaluate the ratio between P_{gp} and P_{es}:

$$\frac{P_{gp}}{P_{es}} \quad (11)$$

3 Evaluation

In the model, the expressions like $N_m(k)$ and $N_d(j)$ describe the number of combinations that could cause data loss. Since there does not always exist an optimal scheme that creates non-overlapping copysets that cover all the nodes [4], these values depend on the values of m and w and how the remaining nodes are grouped if there are any. For this reason, we need to compare the two systems based on a fixed range of k and j. As an example, we set $k + j <= 3$ to represent failures of exactly three components and determine the relative probability of data loss between the two systems:

$$ratio = \frac{P_m(3) + P_d(3) + P_{m,d}(1,2) + P_{m,d}(2,1)}{P'_m(3) + P'_d(3) + P'_{m,d}(1,2) + P'_{m,d}(2,1)} \quad (12)$$

With this setting, we have $N_m(3) = l_{gp}$, $N_d(3) = nl_{gp}$, $N_{m,d}(1,2) = N_{m,d}(2,1) = 3nl_{gp}$ and $N'_m(3) = N'_d(3) = l_{es}$, $N'_{m,d}(1,2) = N'_{m,d}(2,1) = 3l_{es}$.

According to equation 12, we can plot the result by fixing f and w with some reasonable values, and see how the relative probability of data loss relates to the number of servers m, the relative computing performance c, and the relative storage performance n. In the following figures from 2 to 5, we use $w = 4$ as it provides similar data recovery time as random replication on small clusters [4]. When showing the impacts of m and n, we use $c = n$ as a conservative setting, meaning that the total number of block storage devices in all servers is equal to the number of embedded storage devices used in the storage system. And when showing the impacts of m and c, we set $n = 12$ to represent a moderate size of server for edge cluster use cases.

Figure 2 shows how the changes in the number of servers and the ratio of storage performance affect the relative probability of data loss between the two systems. We use $f = 2$

as the ratio of failure rates for hard drives according to [17]. This figure shows that even though the total number of storage devices in all servers is equal to the total number of embedded storage devices, since the system constructed with embedded storage devices has more independent failure domains the failover mechanism (i.e., copyset replication) can span, the possibility of data loss of this system is much lower than that of the system constructed with general-purpose servers. For example, when every server hosts four storage devices, the relative probability of data loss between the two systems is as high as 7.1. In figure 3, when $c < 12$ the total number of storage devices in all servers is less than the number of embedded storage devices. This figure supports our hypothesis that the less complex a failure domain is, the more reliable that failure domain becomes.

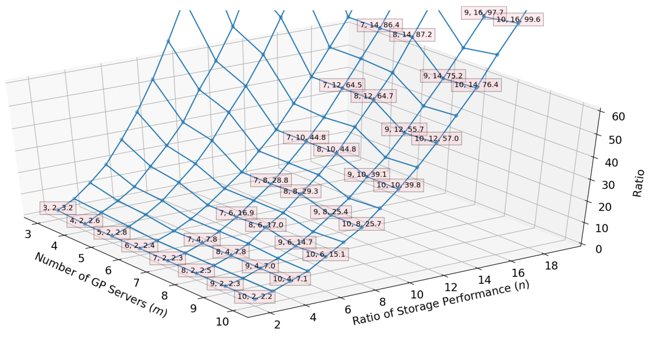

Figure 2: The Impacts of m and n on the Result Ratio (Hard Devices)

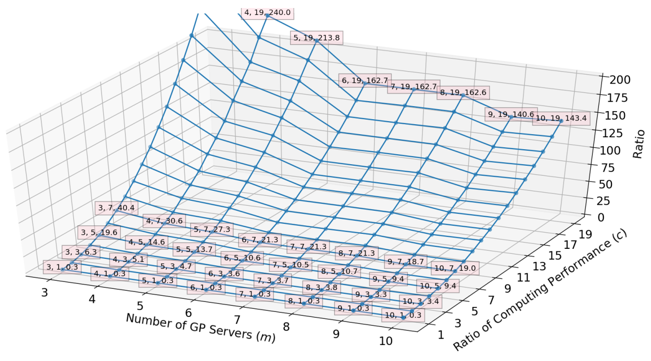

Figure 3: The Impacts of m and c on the Result Ratio (Hard Devices)

Figure 4 and 5 use $f = 0.06$ to emulate the low failure rate of solid-state drives (SSDs) due to their electrical design without any moving parts. According to [23], the failures of SSDs account for only 5.6% of all hardware failure events. Since the computing resources (like CPU and DRAM) in a server become the dominant risk of hardware failures, the growth of the curve is more severe than the previous case using hard drives as storage media. With four SSDs per server, the result ratio between the two systems reaches 20.7, indicating that storage systems using SSDs may be better to employ scale-out architecture constructed with less complex deployment units such as embedded storage devices to achieve better data availability.

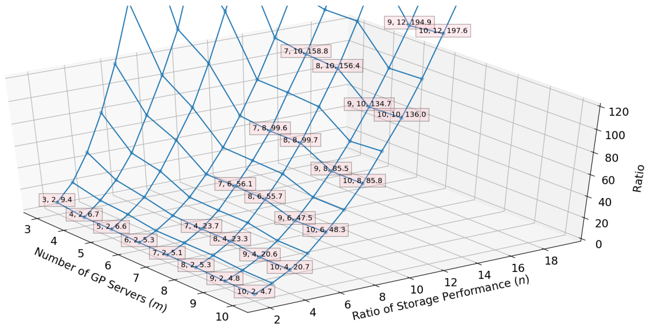

Figure 4: The Impacts of m and n on the Result Ratio (Solid-state Drives)

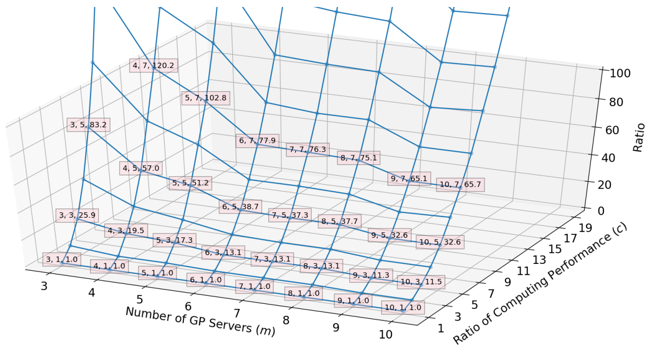

Figure 5: The Impacts of m and c on the Result Ratio (Solid-state Drives)

4 Related Work

Wang et al. [18] proposed a data availability model using a shifted declustering data layout technique, with which they demonstrated to have a significant bandwidth reduction during recovery compared with the copyset layout. Liu and Shen [11] developed a multi-failure resilient replication scheme that offers a lower probability of data loss than copysets. Compared with these techniques, our approach is to scale out to get more independent failure domains, is agnostic to these techniques, and can offer extended availability beyond what these techniques can do. FAWN [1] suggested that building a storage cluster with wimpy nodes is much more power-efficient, and can still meet the capacity, availability, throughput, and latency requirements set for a conventional storage cluster.

5 Conclusion

Providing high data availability at a low cost is important for edge infrastructures. The scale-out storage architecture using embedded storage nodes not only increases the number of failure domains and therefore improves data availability, but also reduces the operational cost of running storage systems of the architecture. As evidence for our hypothesis we determined in our evaluation, the storage system constructed with general-purpose servers can have 7 to 20 times higher risk of losing data than the storage system constructed with embedded storage devices.

6 Discussion Topics

The model: **a)** It is ideal to calculate the possibility of data loss for each of the two systems instead of the possibility based on specific ranges of k and j. To do this, we need a general way to calculate the values of $N_m(k)$, $N_m^{'}(k)$, $N_d(j)$, and $N_d^{'}(k)$. We have not found a general way to express the value of these expressions for any values of m and w, especially for those cases in which optimal schemes do not exist. As future work, we may use stochastic simulation to estimate the probability of a generated subset of size k that covers any pre-defined equal-sized subsets. **b)** We assume that the failures of different nodes are independent. This may not be true as some data need to be re-balanced when a failure occurs; the lifetime of some affected nodes could shrink because of having more data traffic than they normally have during regular time. Furthermore, we can involve the repair rate in the model so that we can simulate uncorrelated failures using the Markov model [3, 7].

The performance of embedded storage nodes: Embedded storage nodes are in general less powerful than general-purpose servers, partly because of their small form factor limiting the available space for packaging more computing resources. However, scale-out storage systems constructed with these embedded storage nodes can offer high aggregate bandwidth, which makes it especially applicable to bandwidth-sensitive workloads such as content delivery services [15]. For latency-sensitive workloads, as the performance of embedded processors surges rapidly generation by generation [8], we believe that running these workloads on clusters of embedded storage nodes is promising in the near future. Readers who are interested in the cost-benefit quantification of offloading data access functions to embedded storage nodes can refer to our earlier work [10].

The network complexity: The embedded storage node-based storage systems may require more network connection ports for communication, which seems to increase the likelihood of data loss because of the added network complexity. However, because the traditional storage devices in a server are also connected with ports (e.g., SAS/SATA ports), the ratio between the number of storage connection ports and the number of storage devices is unchanged. Since there is no evidence that these two types of ports have significant different failure rates, the added network complexity actually keeps the failure rates of the two types of systems in terms of connections of storage devices in balance.

The usefulness of the model: Resizing the failure domains of storage systems relates to the cost and the performance management of hardware. On the one hand, reducing the size of a failure domain in a system may increase the cost per gigabyte because each platform now hosts a smaller number of storage devices. Traditionally, we believe that the cost of the computing resources of a storage server could be amortized by increasing the number of storage devices within it. On the other hand, platforms with finer-granular resources could be more cost-effective than platforms with powerful aggregated resources. This model could be instrumental in system design to help determine the point of balance between the size of a failure domain, the cost, and the performance of the hardware.

7 Acknowledgements

We would like to thank the reviewers of this paper for their helpful feedback. This work was supported in part by NSF grants OAC-1836650, CNS-1764102, and CNS-1705021, and by the Center for Research in Open Source Software (cross.ucsc.edu). Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

[1] David G Andersen, Jason Franklin, Michael Kaminsky, Amar Phanishayee, Lawrence Tan, and Vijay Vasudevan. Fawn: A fast array of wimpy nodes. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 1–14, 2009.

[2] Saurabh Bagchi, Muhammad-Bilal Siddiqui, Paul Wood, and Heng Zhang. Dependability in edge computing. *Commun. ACM*, 63(1):58–66, December 2019.

[3] Richard Eric Brown, Shalini Gupta, Richard D Christie, Subrahmanyam S Venkata, and R Fletcher. Distribution system reliability assessment using hierarchical markov modeling. *IEEE Transactions on power Delivery*, 11(4):1929–1934, 1996.

[4] Asaf Cidon, Stephen Rumble, Ryan Stutsman, Sachin Katti, John Ousterhout, and Mendel Rosenblum. Copysets: Reducing the frequency of data loss in cloud storage. In *Presented as part of the 2013 {USENIX} Annual Technical Conference ({USENIX}{ATC} 13)*, pages 37–48, 2013.

[5] F Dahlqvust, Mark Patel, A Rajiko, and J Shulman. Growing opportunities in the internet of things. *McKinsey, July*, 2019.

[6] Jörg Domaschka, Christopher B Hauser, and Benjamin Erb. Reliability and availability properties of distributed database systems. In *2014 IEEE 18th International Enterprise Distributed Object Computing Conference*, pages 226–233. IEEE, 2014.

[7] Kevin M Greenan, James S Plank, Jay J Wylie, et al. Mean time to meaningless: Mttdl, markov models, and storage system reliability. In *HotStorage*, pages 1–5, 2010.

[8] Matthew Halpern, Yuhao Zhu, and Vijay Janapa Reddi. Mobile cpu's rise to power: Quantifying the impact of generational mobile cpu design trends on performance, energy, and user satisfaction. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 64–76. IEEE, 2016.

[9] Minh Le, Zheng Song, Young-Woo Kwon, and Eli Tilevich. Reliable and efficient mobile edge computing in highly dynamic and volatile environments. In *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 113–120. IEEE, 2017.

[10] Jianshen Liu, Philip Kufeldt, and Carlos Maltzahn. Mbwu: Benefit quantification for data access function offloading. In *International Conference on High Performance Computing*, pages 198–213. Springer, 2019.

[11] Jinwei Liu and Haiying Shen. A low-cost multi-failure resilient replication scheme for high data availability in cloud storage. In *2016 IEEE 23rd International Conference on High Performance Computing (HiPC)*, pages 242–251. IEEE, 2016.

[12] David Reinsel, John Gantz, and John Rydning. The digitization of the world from edge to core. *IDC White Paper*, 2018.

[13] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[14] Weisong Shi and Schahram Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[15] Athena Vakali and George Pallis. Content delivery networks: Status and trends. *IEEE Internet Computing*, 7(6):68–74, 2003.

[16] Rob van der Meulen. What edge computing means for infrastructure and operations leaders. *Gartner, online, available, www. gartner. com*, 2017.

[17] Kashi Venkatesh Vishwanath and Nachiappan Nagappan. Characterizing cloud computing hardware reliability. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 193–204, 2010.

[18] Jun Wang, Huafeng Wu, and Ruijun Wang. A new reliability model in replication-based big data storage systems. *Journal of Parallel and Distributed Computing*, 108:14–27, 2017.

[19] Wikipedia contributors. Dennard scaling — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Dennard_scaling&oldid=938947405, 2020. [Online; accessed 24-February-2020].

[20] Wikipedia contributors. Edge computing — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Edge_computing&oldid=937105715, 2020. [Online; accessed 21-February-2020].

[21] Wikipedia contributors. Poisson distribution — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Poisson_distribution&oldid=941164666, 2020. [Online; accessed 19-February-2020].

[22] Gwyn Wischmeyer. Soaring Field Service Costs Demand Investments in Process, Technology, February 2012. publisher: Technology Services Industry Association.

[23] Erci Xu, Mai Zheng, Feng Qin, Yikang Xu, and Jiesheng Wu. Lessons and actions: What we learned from 10k ssd-related storage system failures. In *2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19)*, pages 961–976, 2019.