Storage on the Edge: Evaluating Cloud Backed Edge Storage in Cyberphysical Systems

Sanjeev Sondur*, Krishna Kant*, Slobodan Vucetic*, and Brandon Byers†
*CIS Department, Temple University, Philadelphia, PA 19122
†Oracle USA, San Diego, CA 92121
* [sanjeev.sondur, kkant, vucetic]@temple.edu †brandon.byers@oracle.com

***Abstract*—Effective control of emerging cyberphysical systems such as smart transportation, smart health-care, etc. requires edge computing infrastructure that is often organized into three layers, namely edge (IoT) devices, edge controllers (ECs) and the cloud. In large infrastructures, ECs must be deployed densely in the proximity of edge devices and need to satisfy strict constraints on cost, size, cooling, etc. Thus, ECs cannot host large amounts of local storage and instead must make use of cloud storage in the background to provide an impression of large, fast local storage to host the IoT device data needed for online and real-time queries. In this paper, we provide insights into the configuration issues of such an edge storage infrastructure (ESI) based on the evaluation of commercial ESIs on several real-world edge computing workloads. We also show that the current ESI designs are lacking in several respects, and suggest some approaches for enhancing their capabilities to meet the stringent requirements of emerging edge computing applications.**

***Index Terms*—Edge Computing, Edge Storage Infrastructure, Object Storage, Configuration**

I. Introduction

An important goal for the ongoing deployment of the edge computing infrastructure and the 5G networks is intelligent control of a wide variety of cyberphysical systems affecting every aspect of the daily life of the citizens [1]. Edge computing together with high-bandwidth 5G networks is expected to support a large number of critical applications including traffic management, surveillance, wellness and health care management, smart manufacturing, smart distribution logistics, etc. [2]. Edge computing typically uses a 3-layer model as shown in Fig. 1, where the bottom layer form the edge devices that connect wirelessly to the middle layer of edge controllers (ECs) in the local region. The ECs receive data streams from the devices and do real-time analytics and short-term storage of the data. They also handle queries on the data which may involve collaboration with other ECs (e.g., tracking a car in the smart transportation context). The ECs connect to the cloud on the backend which provides long term storage for data and offline analytics on it.

A large cyber-physical system may require extensive EC infrastructure, with each EC managing many IoT devices including those generating large data streams (e.g., cameras) and providing efficient online access to large data sets (e.g., feature matching over cars on the road, or a large number of medical images). To minimize the deployment costs, an EC is likely to be housed in a small enclosure that is placed inconspicuously somewhere without the benefits of a typical machine room type of location. This results in serious limitations on the size, weight, power consumption, and physical access, and in turn requires a hardware/software design that automatically exploits the cloud resources where possible and desirable for computation and storage. In particular, the Edge Storage Infrastructure (ESI) [3] must be designed to exploit a small amount of fast local storage along with transparent data transfers to and from the cloud storage to create an impression of essentially unlimited local storage. In this paper, we focus on this aspect and discuss the issues and insights in configuring the local compute and storage resources to meet the QoS requirements of the edge computing applications.

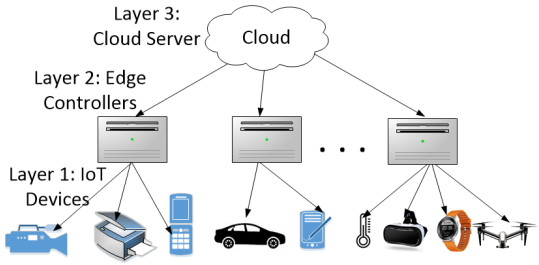

Fig. 1. Hierarchical architecture of edge computing.

The bottom layer in this model could consist of both smart devices (i.e., those with general purpose computing/storage capabilities) interacting directly with ECs or regular devices (e.g., a few ordinary cameras) interfaced locally with an "edge computing device" that then interacts with the ECs. In either case, the ESI model can be applied at the device layer also with remote storage located on the EC, although current commercial ESI offerings are not targeted for this purpose.

In this paper we present our insights into the workload, resource and performance characteristics of ESI based on an experimental evaluation of a commercial ESI system (also called as Storage Gateway) using demands of practical streaming workloads. The evaluation reveals a number of areas where the current ESI implementations must be improved to support emerging edge computing applications with increasingly varied QoS requirements including those where the tail latency must be controlled tightly.

The rest of the paper is organized as follows. We discuss the related work in section II and then introduce the ESI and its configuration issues in section III. In section IV, we present our experimental evaluation approach and then discuss

This research was supported, in part, by the NSF grant IIP-1439672

empirical findings in section V. Based on our findings, we recommend potential ESI enhancements in section VI. We conclude the paper in section VII.

II. Related Work

Edge Storage Infrastructure (ESI) is a relatively new paradigm in bridging the performance/latency gap between local SCSI access and back end Cloud OSS. However, it is well known that the storage system performance depends on the workload characteristics, deployed optimizations, and their specific configuration in a complex way [4] which makes accurate modeling very difficult. Costa et al. [4] support our complexity problem, their study relates to data compression and enable/disable data deduplication efforts. IBM authors, Ofer [5] used deep learning techniques in object storage systems to recommend the best strategy for cache eviction and refreshing data. Their study is the closest that relates to our work both in terms of the application of deep learning and working with cloud based object storage systems. While their study applies deep learning to caching techniques in an object store, we explore the resource allocation of object store based ESI. The work by Rao [6] relates to our problem domain wherein they use reinforced learning/ Artificial Neural Networks (ANN) for resource allocation to a virtual machine; their results support our observations that the traditional control theory is *inadequate* to capture the relationship between multiple control and system outputs limits. Storage configuration brings in substantial additional complexity due to stored data access and movement. Configuring for optimal performance depends on hardware and workloads, and is challenging because of the immense number of configurations their complexities and nonlinear system behavior [7]. Configurations also "are often difficult and knowledge-intensive to develop, brittle to various environment and systems changes, and limited in capacity to deal with non-steady-state phenomena [8]".

III. Edge Storage Infrastructure

An Edge Storage Infrastructure (ESI) such as a commercially available Storage Gateway can be viewed as "two-layer IO bus" (shown with markers in Fig 2): (Point A) front-end IO traffic from local users and (Point C) the back-end IO upload/fetch traffic to/from Cloud storage. The Edge Controller (Point B) has to match the vast gap between the bandwidth offered by SCSI transfer rates of (A) and back end unpredictable network conditions of (C), and a suitable caching mechanism for each so that data transfers to/from the backend Cloud Storage (Point D) can be properly handled. The advantage of ESI is that while the user data resides on the Cloud, it makes the access appear going to a local block device such as SCSI. For this, the ESI needs to manage the protocol conversion, where *SCSI IO blocks* of rather small size (e.g., 4 or 16 KB) have to be converted into *https/REST based* object store accesses, since the cloud interface is invariably object based. The object sizes could vary over a large range, although extremely large files are generally broken up into objects of some maximum size such as a few GB. This conversion affects IO acknowledgment, retries, buffering, latency, etc.

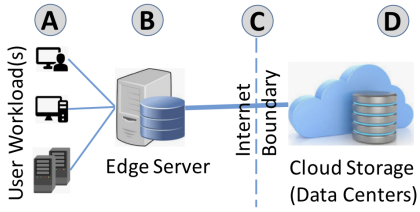

Fig. 2. Three Tier Study of Edge Storage Infrastructure.

An ESI should bridge the IO gap between the on-premise SCSI based disk operations (high IO rate, low latency, almost zero errors) and the backend Cloud object storage system (low IO rate, high latency, retry on timeouts/errors). The server capabilities, resource allocation, and configuration of the Edge Controller becomes an important factor that defines the latency/performance experienced by the users. The complexities involved in the Cloud-based object-store workload patterns are different from the traditional local block IO workloads. We will examine (i) these complex factors and their effect on the behavior of ESI, (ii) performance/ latency experienced by the user workloads, and (iii) the Cloud upload conditions (or failures). Beyond delivering user performance, Edge Controllers have to satisfy constraints such as cost/ space/ cooling, etc.

A. Brief Overview of Object Store

In object store, data is represented as an "object", which refers to a piece of data described (and pointed to) by the appropriate metadata. The metadata resides separately in a metadata server (MDS). The objects are stored on "object storage devices" (OSDs) that natively manage the mapping of the objects to the underlying device structure such as sectors or blocks. The metadata server together with the OSDs also implements access control to the objects so that it is not possible to directly access an object from OSDs. Instead, a query to the metadata server generates a capability that must be presented to the OSD for access to the data. A single metadata server typically serves multiple OSDs. To access an object, an application first contacts the metadata server, and then directly accesses the relevant OSD to retrieve the data using the capability provided by the metadata server. This makes the object store model quite scalable since accesses to multiple OSDs can proceed in parallel. This structure is shown in Fig. 3.

An object could represent any type of entity including entire file, fragment of a file, a contiguous set of database rows, a directory, etc. Object size is often limited, so that a very large file may have to be split into multiple objects. Most common types of data stored in the object store are unstructured data (but rich with metadata) such as images, audio, and video clips. Objects are typically identified with 64 bit Object ID and grouped within partitions with 64 bit partition ID. This gives each object a unique 128 bit namespace [9].

Rich metadata and flexibility of objects enable the user to find data based upon regular expressions or search in large

data-sets on metadata properties. This allows users to treat the Cloud as a large database of objects. As the size of the Cloud grows, so does the ability to find data based on required object-properties (e.g Films created before 1980).

Metadata processing is essential to suitably access an object and small sized metadata makes it easier to cache. Thus, dedicating adequate compute resources to process metadata is crucial for good storage retrieval performance.

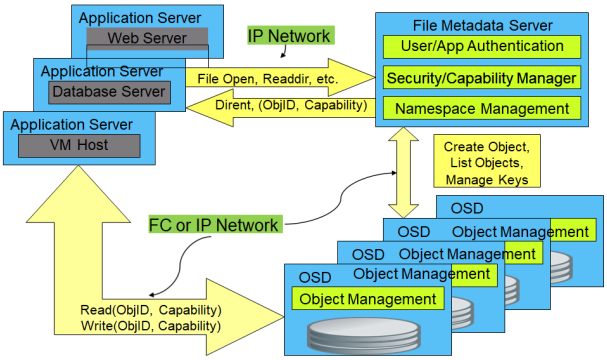

Fig. 3. Object Storage Infrastructure [9]

In traditional block file systems, block allocation and data transfer are very expensive. In block IO file systems API structure forces users to make more frequent API access (open, write, seek, close). OSDs grants (or denies) access to individual objects and fetch (or write) the objects using high level API calls that encapsulates low level details.

Because of these advantages, Object storage forms an ideal platform for data storage on the Cloud. Every object can be accessed directly with a unique ID and direct http/REST API, making data access faster. For these reasons, all commercial vendors of ESI devices use Object store as a Cloud Storage platform.

B. Characterizing the Behavior of ESI systems

Characterizing the behavior of cyber-systems is difficult due to the numerous parameters with complex inter-dependencies that are mostly unknown or poorly understood with respect to their impact on the overall performance, availability or user experience. These dependencies and complex behavior make analytic modeling of performance very difficult [10]. ESI combines the complexities inherent in a storage system, cache allocation, IO demands and the unpredictable nature of back end Cloud systems [11].

It is often very difficult to pinpoint how a change in system configuration can affect the overall performance of the system. Even the concept of performance itself can be subject to scrutiny when considering the complexities of subsystem interactions and the many ways in which performance metrics can be defined [12]. Similar to other cyber-systems, ESI too has many configuration parameters or ”knobs” with little clarity on how to set them or what precise impact they have on the output end.

1) Workload and Application Scenario: User workloads use objects of a varied size that may be uploaded to Cloud (write/put operation) or fetched from the Cloud (read/get operation). They represent real-world scenarios from traffic cameras/ smart city grids, video uploads, measurements from IoT devices, and medical images. Yi [16] has characterized the application scenarios and the demands placed by such users/workloads on Edge/Fog Computing Infrastructure. Additional workload properties and the influence of metadata is explained in later sections. We studied ESI with a large range of workloads with comparable studies from Yi [16], Varma [14], and YCSB [17]. Our scenario matches YCSB workloads with uniform distribution and large *write intensive* IO operations. Table I compares the different workloads used in our studies with a related example of real-world objects. In Table. I, DICOM (Digital Imaging and Communications in Medicine) refers to a standard for handling, storing, printing, and transmitting information in medical imaging. We are aware that user workloads include continuous incoming data streams like traffic images and discontinuous objects like IoT/Medical Images. We have considered both scenarios in our evaluations. In our experiments, we stressed the metadata structure with ownership O, flat dir. F or deep sub-directories D and permission P (*shown* as O,D,F,P in Table I).

2) Interface and Access to Data: Legacy and standard applications should be able to access data in ESI without any application re-writes. Typically, applications access data locally over SCSI bus, that operate at block IO levels (4K, 16K blocks). Delays due to data unavailability, connectivity, poor performance, etc. will degrade the user experience often resulting in application failures and time-outs. In addition to overheads such as protocol conversion (REST API to SCSI/NFS), ESI should satisfy the bandwidth gap by bridging the high performance/low latency user-side demands (Point A in Fig. 2) with high latency, unpredictable network on the back end (Point C in the same figure).

3) Security and Metadata: Security and metadata management in ESI system is a complex task. File access by a legacy application over SCSI should be translated to a OSD object operation. This involves translating the low level SCSI APIs such as file open/close and data write, to high level OSD APIs including metadata operations. Further, file attributes such as permissions, directory structure, etc. have to be transferred to OSD operations like Object ID, object path, metadata attributes. These metadata overheads involve considerable space and time, and consume both CPU and memory resources. Faster CPU speeds can translate the file-attribute (permission, sub-directory, name) to object metadata (ownership, object path, Object ID) faster. Thus, size of metadata operations and allocated resource (CPU speeds, metadata space) play an important factor in determining the user experience in an ESI system. All Object Storage Service (OSS) operations have to consult metadata for *relevant information* about the object. Caching metadata can improve performance and potentially avoid consulting the Cloud based OSD frequently. In this sense, resource allocation for metadata becomes an important factor in space available to pre-fetch metadata. Better user experience and higher performance depend on both data

ESI Workload		Confais IFPS [13]	IoT/Medical Images [14], [15]	
Workload Type	**Users, Objects, Obj. Size**	**Ops. and Obj. Size**	**Image Type**	**(Object) Image Size**
Tiny (O,D,F,P)	25 x 10,000 x 4 KB	–	Health Monitors	4 KB*
Small (O,D,F,P)	25 x 10,000 x 256 KB	100 x 256 KB	MRI/CT Scan	131 to 350 KB
Medium (O,D,F,P)	5 x 10,000 x 1MB	100 x 1 MB	DICOM Visible Light	1 MB
Large (O,D,F,P)	5 x 1,000 x 10 MB	100 x 10 MB	Mammography	27 MB
Huge (O,D,F,P)	2 x 200 x 1 GB	–	Pathology	1.3 GB

TABLE I
COMPARABLE WORKLOADS IN ESI SYSTEM (*ESTIMATED)

and metadata operations. Hence, compute (CPU) and storage (space) resource allocation to both data and metadata play an important part in determining the performance experienced by the users of an ESI system.

C. Describing an ESI system

The performance of an ESI system can be regarded as the throughput experienced by the client. In general, the performance depends on workload characteristics, EC hardware characteristics, and resources allocated to the workload. The consequence of a poor choice of any of these aspects would be experienced by the user as rejected requests or IO timeouts. Each one of these aspects itself is difficult to characterize because of a large number of parameters and complex relationships among them. To get around the difficulties of a detailed analytic characterization, we postulate a classification mechanism for each of them based on the most important parameters. The choice of these parameters cannot be automated and instead is generally derived from the domain knowledge and insights gained by the administrator regarding various attributes. Accordingly, based on extensive experimentation and our experience with the ESI, we represent hardware η_h and workload η_k classes as the following functions:

$$\eta_h = f_2(cs, nc, mc, bw, di) \quad (1)$$

where cs is the core speed, nc is the number of cores, mc is the memory size, bw is the memory bandwidth, and di is the disk IO capacity. Also,

$$\eta_k = f_3(ar, rs, ms) \quad (2)$$

where ar is the request arrival rate, rs is the request size, and ms is the metadata size. For hardware aspects, we use a classification consisting of the enumerated set {small, medium, large}. For workload, the classification is shown in Table I. The functions f_2 and f_3 are learned by training a neural net $\mathcal{N}$, the details of which are contained in [18].

The ESI resource allocation is the disk space allocation in this case and is denoted by η_r. This is split into three distinct partitions: data cache size db, metadata size md and log space size ls. The log size should have no influence on performance except that writebacks of the log would take up some backend IO bandwidth. Obviously $\eta_r = db + md + ls$ and η_r must be less than the total available disk space.

The performance measure p depends on the user requirements, and is adequately expressed as the desired read and write throughput in either MB/s or as objects/sec [15]. Now p can be expressed in terms of workload class η_w, Edge Controller class η_h and resource allocation class η_r as:

$$p = f_1(\eta_h, \eta_k, \eta_r) \quad (3)$$

where the function f_1 is also learned through the neural net $\mathcal{N}$.

IV. EXPERIMENTAL STUDY OF ESI

Using an "off-the-shelf" ESI solution, we focused on (1) workload performance (object write/put times) under different scenarios (2) performance vs network traffic exchange between ESI and Cloud OSS (3) influence of resource allocation on performance.

In order to study the behavior of ESI, we conducted several experiments to collect required data and analyze various factors influencing the behavior of the ESI system. Our experiments focused on collecting a wide range of data under different conditions, to satisfy all features/parameters in section III-C. We used a test environment comprising of servers of different capacity as shown in Table III. All servers have Ubuntu 14.04 with required tools and connected to the local network. We used different hardware configurations to study the influence of cores, core speeds, disk, and memory configurations. On each of these servers, we partitioned the disk for several cache configurations from 25 GB to 1000 GB. The server is connected to the HDD volumes on Cloud OSS using NFS protocol.

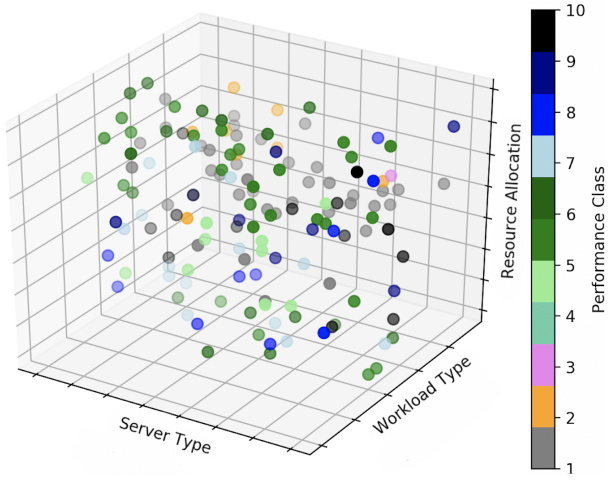

Fig. 4. Illustration of Sample Test Data.

For each execution of the workload η_k on a given ESI hardware η_h, we configured the resources η_r for a given allocation set (e.g. Data cache = 250GB, metadata = 100GB) and executed a pattern of workload η_k and collected the performance data p. We then clear the earlier configuration and workload data and *reconfiguring* for the next experiment.

For each execution, we collected metrics on execution time, metadata time (e.g. to create sub-directories, open and close files, etc.), throughput in bytes/sec. Alongside the workloads η_k, we captured details of the ESI server η_h (e.g. cores, core speed, memory, disk capacity, etc) and resource allocation η_r (i.e. data cache area, metadata, log size). A detailed description of the experiment setup, workload execution, resource configuration is given in our initial paper [18].

We captured nearly 990 data-sets and captured for various metrics ($p, \eta_h, \eta_k, \eta_r$). An illustration of the sample data set is shown in Fig. 4. The figure shows individual data points for various performance p observed for different combinations of workload η_k, server η_h and resource allocation η_r (Eq. 3). We analyze the collected data from various experiments and present our empirical findings below.

V. Evaluation and Empirical Findings

In this section, we evaluate the empirical results in terms of workload, resource allocation and performance. We compare our performance study with the related data from Confais [13]. Confais used YCSB tool [17] to benchmark Object Store performance between three edge storage systems. In YCSB terminology, our study classifies as write/ update heavy, uniform distribution workload. In addition, we study the influence of various parameters of Eq.$\{3, 2, 1\}$ on both the user side and the remote *Cloud upload* performance. Fig. 2 depicts the three tiers of our study as (A) user workloads, (B) Object Store on Edge Controller and (C) back end Cloud upload. The details of the data center hosted Cloud Object Storage (D) is hidden from the user and not included in this paper. Confais study is limited to multiple operations of varied file size (See Table[1] in Ref. [13]) and with no patterns in specific workloads. Table.II shows the objects count/size and performance observed in Confais. The matching workload type and average observed performance (from one sample configuration) in our study are given in the last three columns. The performance difference between our study and Confais is attributed to the number of objects and the architecture difference between ESI and Confais IFPS system. The comparison validates both the advantages of ESI (gain in performance) and the workloads used in our research.

Our findings are based on real-world customer workloads and empirical data collected through many months of effort. Our workload involves *entire file level* operations rather than block IO operations.

The time taken t_{total} to write the file involves all the data operations and metadata operations (including retries). Let t_{data} and t_{meta} denote, respectively, the object data (i.e., read or write) time and meta-data time. The latter includes sub-directory creation time t_{sub} (if any), file open time t_{open}, metadata tasks t_{mtask} and file close time t_{close}. Metadata tasks include journal maintenance, security, file locks, permissions, versioning, etc.

Storage systems are efficient in block writes and tuned to give the best IOPS rates. However, time taken for metadata tasks are significantly larger than the time for a simple IO operation ($t_{meta} > t_{data}$). If the ratio between time to write object and time to perform metadata tasks is smaller, then metadata time dominates the overall time and penalizes the overall performance. Conversely, if the object size is large (large IO operations), the relative time to perform metadata tasks is much smaller ($t_{meta} < t_{data}$); and contributes less to overall performance. We present the findings below and verify our research objective alongside the observations. In Fig. 5, 6, 7 & 8, x-axis denotes various configurations used, and y-axis denotes the corresponding performance observed in MBps.

Finding 1: *ESI provides good performance for large/huge workloads, but not for workloads with small file/object size.*

The workload with a large number of small files would pay the penalty in time spent on metadata operations compared to actual data IO time. In such cases, the time to persist the data is outweighed by the time required to complete the metadata tasks. Multiply this by 10,000 times for the Tiny, Small & Medium workloads in Table I, and the performance drops significantly. For tiny workloads, t_{meta} translates to one meta operation per 4KB written (or 256KB). The overall time for 10,000 metadata operations would be significantly larger compared to writing the data-bytes alone. This is confirmed by the poor performance seen in Fig. 7 for Tiny and Small workloads. The figure shows performance around 2 to 25 MB/s for Tiny and Small workloads under various resource configurations. Allocating additional resource does not help the situation. On the other end of the spectrum, for Large & Huge workloads, the number of metadata tasks is significantly smaller compared to the total write bytes, i.e. in Huge workloads, it is 400 metadata tasks for 400GB or about one meta-operation per one GB of data persisted. Hence, the contribution of metadata tasks is much smaller with large workloads. This is confirmed by high performance numbers measured during our research (as seen in Fig. 5). The extreme right of the figure shows performance for Large and Huge workloads, with few files of large GB file size (See magnified Fig. 6).

The key take away from this finding is that we need to segregate small and large data items, and store the former in local storage permanently. However, this would tend to reduce the available cache for large data items. Thus, the cut-off point of local only and remote backed storage must be determined so that the desired balance is achieved between the performance of small and large data items.

Finding 2: *Data and Metadata IO path of the ESI influences performance significantly. Effect of compute power (CPU, memory) is minimal compared to the IO path.*

This is borne out by our detailed study of different workloads and system configurations as shown in Table III. Server A has a single disk with significant capacity (1 x 5TB); and multiple partitions to host the data-cache and metadata. Server B has multiple disks controlled from a single HDD controller, with smaller capacity (3 x 500MB); and multiple partitions on different disks to host the data-cache and metadata. Both servers had one 32bit, 66MHz, SATA Controller. Clearly,

	Confais IFPS [13]			Edge Storage Infrastructure		
Image Type	**Object Operations & Size**	**Time taken (s)**	**Avg. perf (MB/s)**	**Workload**	**Objects (No. & Size)**	**Avg. perf. (MB/s)**
MRI/Small	1 x 256 KB	0.17	1.5	Small	10000 x 256 KB	10.70
MRI/Small	10 x 256 KB	0.17	0.15	Small	10000 x 256 KB	10.70
MRI/Small	100 x 256 KB	0.33	77.57	Small	10000 x 256 KB	10.70
DICOM/Medium	1 x 1 MB	0.22	4.54	Medium	10000 x 1MB	39.17
DICOM/Medium	10 x 1 MB	0.21	47.61	Medium	10000 x 1MB	39.17
DICOM/Medium	100 x 1 MB	1.07	93.45	Medium	10000 x 1MB	39.17
Mammography/Large	1 x 10 MB	0.34	29.41	Large	1000 x 10MB	177.90
Mammography/Large	10 x 10 MB	0.40	250.00	Large	1000 x 10MB	177.90
Mammography/Large	100 x 10 MB	3.92	255.10	Large	1000 x 10MB	177.90

TABLE II
COMPARING WORKLOAD AND PERFORMANCE

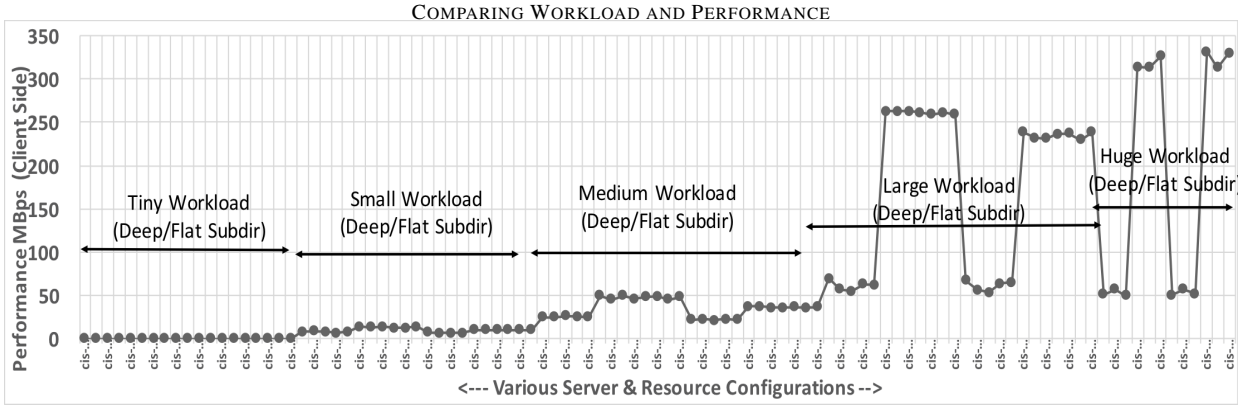

Fig. 5. Performance for Various Workloads under Different Server/Resource Allocations (Client Side).

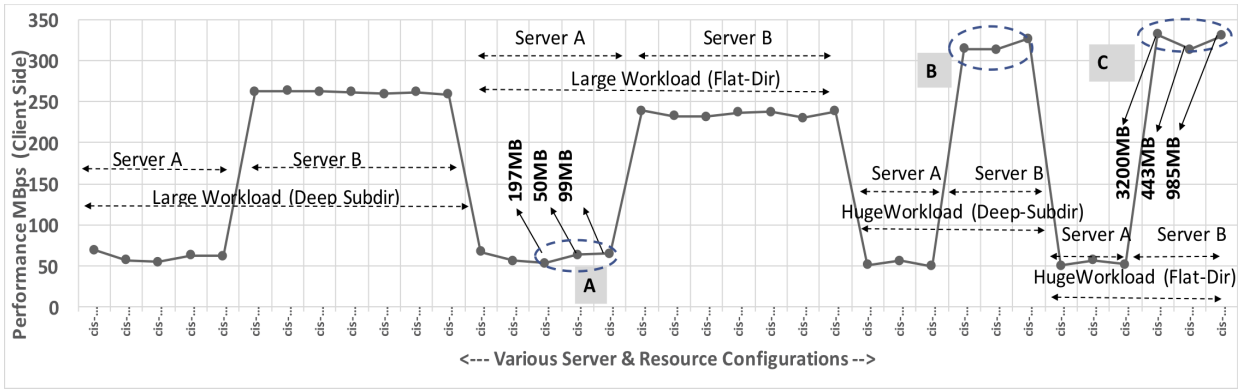

Fig. 6. Performance for Large Workloads under Different Server Configurations (Client Side).

both data-cache and metadata have a common IO path in server A. Though file operation is sequential (open, write, close); these operations contend to the same resource, and add to the bottleneck with one disk, and; multiple partitions on single large disk does not help the cause. Our study is IO intensive and all operations are dominated by the IO path. Clearly, a **multiple disk** infrastructure outperforms a single disk system as can be seen from the performance metrics in Fig. 6. The figure shows performance on server A and server B for Large workloads (both deep sub-dir and flat dir. structure). All workloads shown in the figure suffer from resource contention (data-cache & metadata) because of a single IO path and hence has lower performance (around 50 to 75 MBps). Server B refers to a host with multiple disks and shows better performance (225 to 270 MBps). Multi-disk (IO path) infrastructure improves performance by a large factor in large and huge workloads. A similar performance gain is seen for medium workloads (but at a smaller scale because of the ratio of object/metadata size).

The key takeaway from this observation is that an ESI design may benefit from several smaller disks with distinct controllers than a single large one. However, this needs to be balanced against the increased footprint of a multi-disk system.

Finding 3: *Data-cache needs to be correctly configured for the required workload. Allocating excessive capacity can be detrimental to performance.*

The ESI needs to perform background tasks not directly involved in the user IO path. These include garbage collection, data eviction to Cloud, data-refresh, and are hidden by the ESI vendor but still play an important role in the overall

Server	CPU cores	Memory	Hard Disk	NW Card
A	8 x 2.1 GHz	32 GiB	1 x 5TB	1 GB
B	4 x 1.8 GHz	16 GiB	3 x 500GB	1 GB

TABLE III
TEST ENVIRONMENT: SERVER CONFIGURATION

performance of the system. Allocating excessive cache can hurt these background processes, taking additional time to examine the data in the cache. Similarly, metadata maintenance or journaling needs to walk through the excessive metadata space. This is shown for both Server A (single disk) and Server B (multiple disks) with markers (A),(B) and (C) in Fig. 6. Marker (A) shows doubling the data-cache capacity from 99MB to 197MB can reduce performance in Server A (for large workloads). Similarly, marker (C) shows that 985MB data-cache can perform equally as 3200MB cache. Note: the two markers are for different workload types and different server types. In addition to not contributing to any gain in performance, over-allocation of capacity/resource can strain permissible thermal profile and cooling limits (or other constraints set by the administrator).

The key observation here is that simply using a default cache configuration may be undesirable. However, a proper setting requires knowledge of the workload characteristics. This should not be an issue in an edge computing environment, where the nature of each application shown be known and rather invariant.

Finding 4: *A better cloud upload performance can be achieved for larger objects.*

The ESI communicates with the Cloud Object Store (COS) over the Internet which can have unpredictable delays. We show *this* with empirical results in Fig. 7; here the dotted line represents the available bandwidth between the edge controller and the COS. Each Cloud operation pertains to an object; either to fetch (get) or write (put) *one complete object* from/to the Cloud. As stated earlier, object operations involve related metadata operation from the metadata server. A larger object can be fetched faster than multiple objects of small size, because of less overhead (ratio of data-to-metadata work). Fig. 7 illustrates this clearly with the top Cloud upload performance observed with workloads of large object size (i.e. Large and Huge workloads). Additionally, there exist an optimal configurations (shown as by the dotted circle) that are more efficient in reaching the COS, with higher performance rates (y-axis).

The clear takeaway from this observation is that a batched data upload to the cloud is highly desirable when individual items are small. For data that is generated continuously (e.g., a real-time camera stream), but in other cases, a proper batching itself could be a source of overhead.

Finding 5: *Metadata complexity significantly affects cloud upload performance. For example, Flat directory structure objects (smaller metadata) can be uploaded to Cloud faster than objects with deep sub-directories (bigger metadata).*

A closer comparison of workloads with deep sub-directories vs. flat-directory is shown in Fig. 7. The reasons for metadata operations presented earlier is noticeable in this figure. The difference in Cloud upload time may vary depending on the ratio of data-block to metadata. It should be noted that Cloud upload time over an unpredictable network is still unpredictable. As in earlier finding, there exist optimal configurations (shown as peaks) that have a better Cloud upload performance (y-axis).

Metadata size influences Cloud upload performance. As explained earlier, metadata server has to be consulted for every object operation. A larger/richer metadata structure, such as ownership, versioning, permissions, user-defined attribute like department, takes a longer time to process compared to a simple file name. For example, a deep sub-directory structure on the SCSI side block IO has to be translated to a partition ID and Object ID on the object store side. This takes time, CPU and memory resource.

The takeaway from this finding is that prior knowledge of metadata properties and rich metadata set can be exploited to allocate required disk and memory resource for metadata, to avoid frequent consultation with Cloud OSS for metadata operations.

Finding 6: *Insufficient resource allocation to workload can result in IO failures.*

Table IV shows cases of workload failure due to insufficient resources. Real-world workloads such as medical mammogram/pathology records or real time streams of traffic data cannot tolerate failure due to insufficient resource allocation. These workloads cannot be redone (or incur heavy redo costs for the failure of uploads). These workloads are referred to Large or Huge workload with low resource (data-cache) allocation in our study. Huge workload represents 200GB of data (200 x 1GB files), and the allocated data-cache were below this requirement. Unlike traditional cache systems, ESI cannot evict the write objects to free up data-cache space. The eviction to the Cloud is over an unreliable network, and cannot be guaranteed to complete at speeds required by the front end. Hence front end user/application write operations suffer latencies beyond acceptable limits, and latency sensitive applications will suffer IO failures. Allocated data-cache should account for user workload plus hidden operations like garbage collection, eviction, etc.

This finding points to the need for implementing a suitable admission control mechanism that controls the number of concurrent applications such that all admitted applications can receive an adequate amount of resources.

A. Completeness of Our Study

We considered workload covering a wide range of real-world scenarios. Our workload has an upper bound of 1GB objects and our study is restricted to few ESI servers. Our analysis shows that ESI systems perform better with large size objects, hence objects beyond our study boundary (i.e objects greater than 1GB) can benefit from ESI system and not suffer performance penalties. Of course, the server/resource allocation would change for larger objects. Using more data

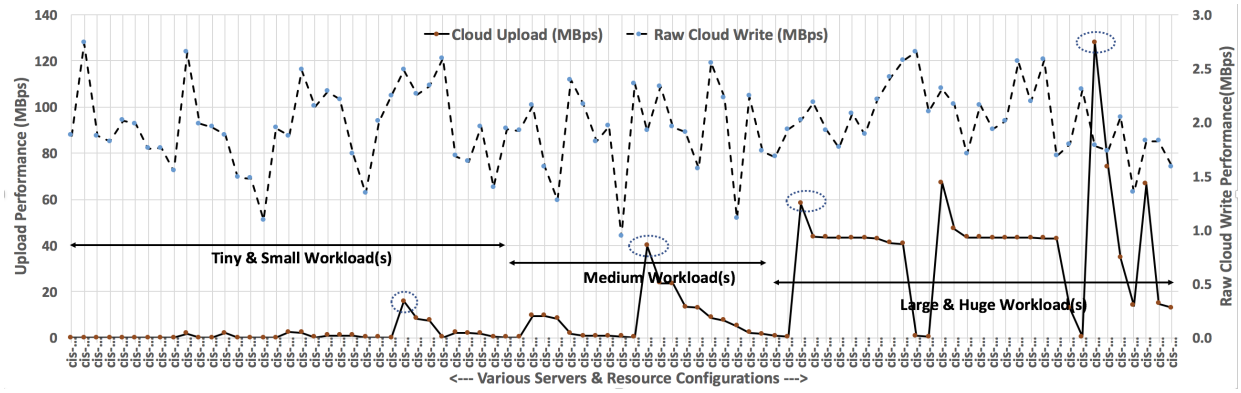

Fig. 7. Edge Storage to Cloud Upload Performance (Data Center Side).

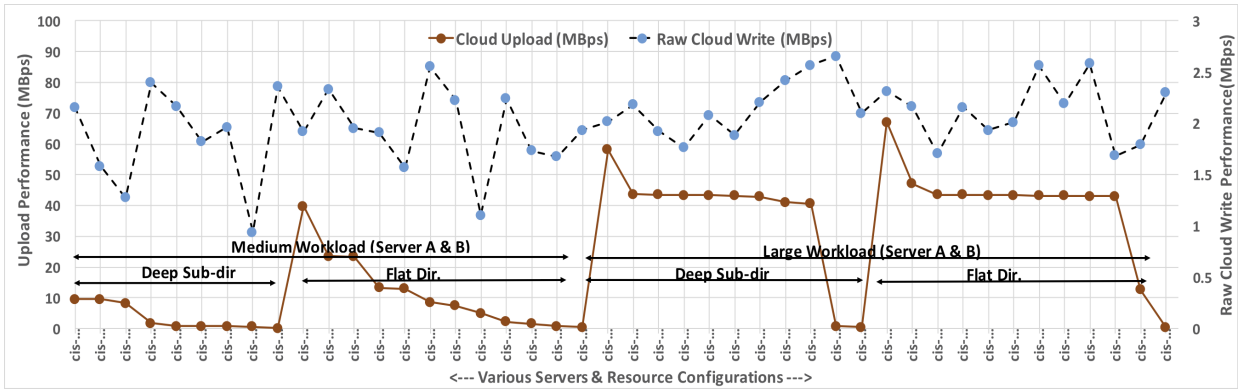

Fig. 8. Edge Storage to Cloud Upload Performance. (Medium & Large Workloads, Data Center Side)

Server	Dir type	Objects (No.x Size)	Cache Allocated	Cache Used	Test Status
A	Deep	200 x 1GB	99 GB	59%	Fail
A	Deep	200 x 1GB	197 GB	69%	Fail
B	Deep	200 x 1GB	50 GB	100%	Fail
B	Deep	200 x 1GB	99 GB	35%	Fail
B	Deep	200 x 1GB	197 GB	81%	Fail
A	Deep	200 x 1GB	497 GB	44%	Pass
A	Flat	200 x 1GB	99 GB	35%	Fail
A	Flat	200 x 1GB	197 GB	83%	Fail
B	Flat	200 x 1GB	99 GB	35%	Fail
A	Flat	200 x 1GB	497 GB	45%	Pass

TABLE IV
WORKLOAD VS. RESOURCE USED

collected from servers of various compute and storage capacity/ capability would enrich the data-set. A larger data-set from different ESI server would benefit the analysis and give a better picture of the influencing factors. Our study consisted of servers with HDD storage device. The analysis is still valid for systems using SSDs drives since we considered the importance of the IO path rather than the storage devices themselves. Our initial analysis and findings from one vendor apply equally to other vendor platforms since an Edge Controller invariably has some compute and storage limits, yet are tasked with satisfying the user demands (performance, workload, cost, cooling, space). Our survey of public documents from a few vendors show that all commercial ESI platforms provide limited options to control the resource allocation (CPU, memory, cache space) to achieve the required workload/performance goals.

VI. POTENTIAL ESI ENHANCEMENTS

Our study shows several areas of potential deficiencies in current ESI implementations for highly heterogeneous workloads. In particular, owing to the metadata related overhead, the ESI may not work well for small/medium sized objects and may have difficulty in meeting tight QoS requirements. This is likely to become more prevalent as more sophisticated real-time edge computing services are implemented that often require control over tail latencies (in addition to the average latencies).

Tail latency for read operations can benefit from *pinning* small/medium objects locally in the edge storage. Intelligent pinning of objects based on metadata (e.g. file size, directory depth, access pattern, etc.) avoids having to fetch small objects from the Cloud. It is important to maintain a balance between pinned and unpinned files since excessive pinning can consume valuable data cache space and trigger unwanted cloud eviction or cache cleanup overheads. The trade-off between prefetching metadata vs. pinning objects have to be studied since both can consume valuable resource and time. Both can help performance but the impact needs to be evaluated carefully. In particular, we need to address several issues in this regard: (1) determining the critical object size under which pinning may be beneficial, (2) ensuring that the overall pinned

content does not occupy a significant amount of space and thereby degrade the overall performance, (3) avoiding storage space fragmentation due to pinning, and (4) opportunistic syncing of the pinned objects to the cloud to reduce the data loss potential due to failure of the ESI. The pinned objects can be batched together for pushing to the cloud. We also observe that certain types of objects (e.g., MRI images with the small matrix size) are statically known to be small and this simplifies their treatment with respect to pinning and batch transfer.

Given the unpredictability of the available backend network bandwidth and user request arrival rate, it is important to do admission control [19] in order to provide suitable QoS to the requests in progress. Since the admission control concerns the availability of network bandwidth in the near future, the admission control needs to be coupled with the network bandwidth prediction. Such a prediction can be done using well known time-series prediction methods [20] but the performance of such a prediction needs to be evaluated experimentally. The prediction can also be exploited for opportunistic pushing of modified objects to the cloud. Prediction of network conditions can be exploited to prefetch metadata or partial objects during bandwidth-plenty periods.

One way to reduce the overhead for small objects is by intelligent bundling of multiple small objects into a single larger object, both for downloading from and uploading to the cloud. (Note that this would require a special module on the cloud side as well to suitably bundle outgoing data and unbundle incoming data.) Intelligent bundling involves consideration of several issues such as the overhead of bundling/unbundling and extra data transfer performed as a result of bundled transfers and needs to be investigated thoroughly.

Current ESI implementations do not support multiple QoS classes. Such a capability is essential with a heterogeneous mix of workloads with different transfer sizes and latency/throughput constraints. The QoS classification can be exploited in deciding what class of objects to pin. It can also be used in intelligently deciding which requests are allowed in by the admission control mechanism when there are bandwidth limitations.

VII. Conclusions

Right-sizing of an Edge Computing Infrastructure is important because of the inherent limitations defined by cost, power consumption, and cooling requirements. The complexities inherent in a storage system plus the unpredictable network makes detailed analytical modeling of an Edge Storage Infrastructure (ESI) difficult. Using empirical data under various workload/resource configurations, we studied an off-the-shelf ESI with a goal to understand the complex parameters that influence its behavior. Our findings highlighted the issues in configuring ESI to handle edge computing related IO. We also discussed several deficiencies in existing ESI designs and some potential approaches to enhancing the ESI designs. In the future, we will examine some of these issues and also evaluate ESIs equipped with SSDs and other emerging storage technologies such as Intel Optane. We believe that our study will also motivate ESI vendors to consider further enhancements to meet stringent QoS requirements of emerging edge computing applications.

References

[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[2] A. Pal and K. Kant, "Iot-based sensing and communications infrastructure for the fresh food supply chain," *Computer*, vol. 51, no. 2, pp. 76–80, 2018.

[3] D. Linthicum, "Edge Computing vs. Fog Computing: Definitions and Enterprise Uses," 2010. [Online]. Available: https://www.cisco.com/c/en/us/solutions/enterprise-networks/edge-computing.html

[4] L. B. Costa and M. Ripeanu, "Towards automating the configuration of a distributed storage system," in *2010 11th IEEE/ACM International Conference on Grid Computing*, Oct 2010, pp. 201–208.

[5] E. Ofer, A. Epstein, D. Sadeh, and D. Harnik, "Applying deep learning to object store caching," in *Proceedings of the 11th ACM International Systems and Storage Conference*, ser. SYSTOR '18. New York, NY, USA: ACM, 2018, pp. 126–126. [Online]. Available: http://doi.acm.org/10.1145/3211890.3211909

[6] J. Rao, X. Bu, C.-Z. Xu, L. Wang, and G. Yin, "Vconf: A reinforcement learning approach to virtual machines auto-configuration," in *Proceedings of the 6th International Conference on Autonomic Computing*, ser. ICAC '09. ACM, 2009, pp. 137–146.

[7] Z. Cao, V. Tarasov, S. Tiwari, and E. Zadok, "Towards better understanding of black-box auto-tuning: a comparative analysis for storage systems," in *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, 2018, pp. 893–907.

[8] G. Tesauro *et al.*, "Online resource allocation using decompositional reinforcement learning," in *AAAI*, vol. 5, 2005, pp. 886–891.

[9] "SNIA Object Store: Tutorial: Everything You wanted to Know About Storage," https://www.snia.org, 2018. [Online]. Available: https://www.snia.org

[10] M. Wang, K. Au, A. Ailamaki, A. Brockwell, C. Faloutsos, and G. R. Ganger, "Storage device performance prediction with cart models," in *The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004.(MASCOTS 2004). Proceedings.* IEEE, 2004, pp. 588–595.

[11] Y. Tanimura and H. Koie, "Operation-level performance control in the object store for distributed storage systems," in *2015 IEEE International Conference on Data Science and Data Intensive Systems*. IEEE, 2015, pp. 111–112.

[12] B. Eckart, X. He, H. Ong, and S. L. Scott, "An extensible i/o performance analysis framework for distributed environments," in *European Conference on Parallel Processing*. Springer, 2009, pp. 57–68.

[13] B. Confais, A. Lebre, and B. Parrein, "An object store service for a fog/edge computing infrastructure based on ipfs and a scale-out nas," in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 2017, pp. 41–50.

[14] R. Varma, "Storage media for computers in radiology," *The Indian journal of radiology and imaging*, vol. 18, pp. 287–9, 11 2008.

[15] "Performance Evaluation of Storage and Retrieval of DICOM Image Content ... ," 2010. [Online]. Available: https://www.oracle.com/us/technologies/embedded/embedded-dicom-wp-366893.pdf

[16] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proceedings of the 2015 Workshop on Mobile Big Data*, ser. Mobidata '15. New York, NY, USA: ACM, 2015, pp. 37–42. [Online]. Available: http://doi.acm.org/10.1145/2757384.2757397

[17] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010, pp. 143–154.

[18] S. Sondur and K. Kant, "Towards automated configuration of cloud storage gateways: A data driven approach," in *International Conference on Cloud Computing*. Springer, 2019, pp. 192–207.

[19] J. Ren, Y. Pan, A. Goscinski, and R. A. Beyah, "Edge computing for the internet of things," *IEEE Network*, vol. 32, no. 1, pp. 6–7, Jan 2018.

[20] A. S. Weigend, *Time series prediction: forecasting the future and understanding the past*. Routledge, 2018.