Submarine: A Subscription-based Data Streaming Framework for Integrating Large Facilities and Advanced Cyberinfrastructure

Ali Reza Zamani, Moustafa AbdelBaky, Daniel Balouek-Thomert, J. J. Villalobos, Ivan Rodero, and Manish Parashar
Rutgers Discovery Informatics Institute (RDI^2), Rutgers University
Piscataway, New Jersey
Email:{alireza.zamani, moustafa.a, daniel.balouek, jj.villalobos, irodero, parashar}@rutgers.edu

ABSTRACT

Large scientific facilities provide researchers with instrumentation, data, and data products that can accelerate scientific discovery. However, increasing data volumes coupled with limited local computational power prevents researchers from taking full advantage of what these facilities can offer. Many researchers looked into using commercial and academic cyberinfrastructure (CI) to process this data. Nevertheless, there remains a disconnect between large facilities and cyberinfrastructure that requires researchers to be actively part of the data processing cycle. The increasing complexity of cyberinfrastructure and data scale necessitates new data delivery models, those that can autonomously integrate large-scale scientific facilities and cyberinfrastructure to deliver real-time data and insights. In this paper, we present our initial efforts using the Ocean Observatories Initiative project as a use case. In particular, we present a subscription-based data streaming service for data delivery that leverages the Apache Kafka data streaming platform. We also show how our solution can automatically integrate large-scale facilities with cyberinfrastructure services for automated data processing.

KEYWORDS

Real-time data delivery; large-scale scientific facilities; ocean observatories initiative; stream processing

1 INTRODUCTION

Open, large-scale scientific facilities are an essential part of the science and engineering enterprise. These facilities provide shared-use infrastructure, instrumentation, and data products that are openly accessible to a broad community of researchers and educators. For example, current experimental and observations facilities provide increasing volumes of data and data products that have the potential to deliver new insights in a wide range of science and engineering domains. However, while these facilities provide reliable and pervasive access to the data and data products, users typically have to download the data of interest and then process them, typically using local resources. Consequently, transforming these data and data products into insights requires local access to powerful computing, storage, and networking resources. These requirements can significantly limit the impact of the data, especially for researchers, educators, and students who do not have access to such capabilities. We are currently experiencing this limitation in the case of the Ocean Observatories Initiative (OOI) [13]. OOI currently serves data from 57 stable platforms and 31 mobile assets, carrying 1,227 instruments (~850 deployed), providing over 25,000 science data sets and over 100,000 scientific and engineering data products. OOI raw data and data products, such as high-definition video and hydrophone data, are rapidly growing in size and even modest queries can result in significant latencies for end users and can overwhelm their local storage and computing capabilities.

To address limited local computational power, users looked into using commercial and academic advanced cyberinfrastructure (ACI) services (e.g., Chameleon, XSEDE JetStream, AWS, etc.). ACI is playing an increasingly important role as platforms for computational and data-enabled science and engineering and can provide the necessary capabilities to allow a broad user community to process the data from large facilities effectively. However, despite clearly complementing each other, many large scientific facilities (for example, OOI) and advanced cyberinfrastructure remain largely disconnected. As a result, users are forced to actively be part of the process that queries and moves data from large facilities to computational services, which limits the potential utility of both the data and the facilities.

In this paper, we explore more effective data delivery mechanisms that can better integrate large facilities with cyberinfrastructure services. We present the architecture, implementation, and performance of a subscription-based data streaming service for data delivery of the OOI project and its integration with public CI services for automated data processing. Specifically, we enable users to create and manage query-based data streams and connect workflows with streams and stream-related events that when triggered can seamlessly orchestrate the entire data-to-discovery pipeline. Such a pipeline involves (i) executing the queries on the OOI CI; (ii) streaming the data to appropriate CI services – possibly using high-bandwidth interconnects (such as Internet2); (iii) staging the data close to computing and analytics resources (e.g., XSEDE JetStream [26]); (iv) launching the modeling and analysis processes to transform such data into insights; and (v) publishing results to the users. The proposed framework, named submarine, leverages *state-of-the-art* enterprise data streaming and processing solutions, namely, Apache Kafka [8], which provides robust and scalable solutions for data management. We also show how approximation techniques can be used to address network limitations and associated latencies.

DataCloud 2017, November 2017, Denver, Colorado USA
2017.

The main contribution of this work is to enable users with limited local computing, storage, and network capabilities to subscribe to data of interest and automatically process it (while it moves from observatories toward the users) using advanced infrastructure. Our proposed unified software stack, which has not been explored in prior works, relies on *Real-time Delivery*, *Data Processing* and *Workflow Description*. The latter provides users with mechanisms to describe the desired data (e.g., types and ranges), the computation (e.g., processing tools), the mechanisms for automated queries (e.g., feature detection), and the minimum acceptable quality of results (QoR).

The remainder of this paper is organized as follows. Section 2 presents the overall architecture of our system. A summary of the OOI project and motivating use case scenarios are presented in Section 3. A Kafka-based implementation of the system is presented in Section 4 followed by an experimental evaluation in Section 5. An overview of related work is presented in Section 6. Finally, the paper concludes in Section 7 outlining future work.

2 SUBMARINE SYSTEM ARCHITECTURE

The overall architecture of our system is depicted in Figure 1, and the main components are described below.

(i) *Messaging System.* The main component of our system is a messaging system, which can provide real-time access to data streams. The messaging system can also provide subscription-based data delivery and transfer mechanisms for users or other components of the system. Agents can be used to fetch the desired data from large facilities and publish them to the messaging system.

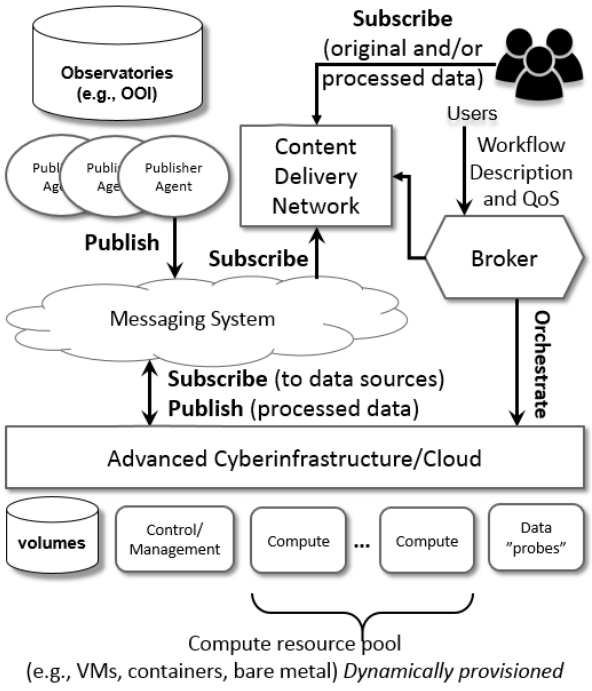

Figure 1: Overall System Architecture.

(ii) *Data processing using ACI.* Advanced cyberinfrastructure services can be used to process the data in real-time. Control/Management services and data probes can be used to subscribe to data of interest and use high-speed connections to transfer the data and store it locally. Computing resources are then provisioned on-demand to process the data, and the results are published back to the messaging system.

(iii) *Content Delivery Network.* To provide users with faster access to data, a subscription-based content delivery network can be instantiated by using network appliances that are within close proximity to the users. These appliances are connected to the messaging system using high-speed connections, can subscribe and replicate data, and deliver it to the users.

(iv) *Broker.* A broker is required to orchestrate the overall execution of the workflow. The broker takes as input the workflow description from the user and provisions the proper services to execute it. The broker can also redirect users to network appliances that are close to them. Finally, the broker can also select the proper tools necessary to deliver a solution to the user within a given QoR based on their network capacity. This can be achieved by leveraging approximation techniques to provide multiple resolutions of the desired data (e.g., using different sampling rates).

3 DRIVING APPLICATION

3.1 Ocean Observatories Initiative

The NSF Ocean Observatories Initiative (OOI) [12, 13, 18] is a networked ocean research observatory with arrays of instrumented water column moorings and buoys, profilers, gliders, and autonomous underwater vehicles (AUV) within different open ocean and coastal regions. OOI infrastructure also includes a cabled array of instrumented seafloor platforms and water column moorings on the Juan de Fuca tectonic plate. This networked system of instruments, moored and mobile platforms, and arrays provide ocean scientists, educators, and the public the means to collect sustained, time-series data sets to enable the examination of complex, interlinked physical, chemical, biological, and geological processes operating throughout the coastal regions and open ocean. The OOI has been built with an expectation of operation for 25 years.

OOI implements a geographically distributed, secure, highly available CI that is responsible for data acquisition and collection, data storage and processing, and on-demand delivery of data and data products to scientists and application developers. The core of the OOI CI software ecosystem (uFrame-based OOINet) is based on a Service Oriented Architecture (SOA), a set of datasets, instruments, platform drivers, and data product algorithms, which plug into the uFrame framework. uFrame is implemented using a combination of scalable and highly available open source distributed data management technologies (e.g., Apache Cassandra [10], RabbitMQ [23], Qpid [14], etc.) and custom development (e.g., parsers and drivers).

Data is gathered from both cabled and wireless instruments located across multiple research stations in the Pacific and Atlantic oceans. Once acquired, the raw data (consisting mostly of tables of raw instrument values) is transmitted to one of three operations centers. The data from the operations centers is then transferred

to the OOI CI for processing, storage, and dissemination. Two primary CI centers operated by the Rutgers Discovery Informatics Institute (RDI^2) are dedicated to OOI data management: the West Coast CI in Portland, OR, and the East Coast CI, at Rutgers University. Data from the Cabled Array components are initially received at the Shore Station in Washington. Then they are further processed using the East Coast CI that houses the primary computing servers, data storage and backup, and front-facing CI portal access point. The setup is then mirrored to the West Coast CI over a high-bandwidth Internet2 network link provisioned by MAGPI (Mid-Atlantic GigaPOP in Philadelphia) on the east coast and PNWGP (Pacific-Northwest GigaPOP) on the west coast. The data stores at the operational management centers are continuously synchronized with the data repositories located at the East and West Coast CI sites.

The OOI CI software ecosystem (OOINet) employs the uFrame software framework that processes the raw data and presents it in visually meaningful and comprehensible ways in response to user queries, which is accessible over the Internet through the CI web-based portal access point. A machine-to-machine (M2M) API provides programmatic access to OOINet through a RESTful API. In addition to the portal and API, OOI CI provides other data delivery methods such as a THREDDS server, a raw data archive, and an Alfresco server for cruise data. OOI CI software ecosystem permits 24/7 connectivity to bring sustained ocean observing data to a user anytime and any place. Anyone with an Internet connection can create an account or use CILogon and access the OOI data. Detailed architecture of the OOI CI network can be found online in [19].

The OOI CI design and implementation principles are based on industry best practices for the different aspects of the CI. The approach is based on a decentralized but coordinated architecture, which is driven by requirements, e.g., data storage capabilities, system load, security, etc. For example, the system is based on a multi-tier security approach with dedicated and redundant (highly available) firewall appliances at the CI perimeter. In addition to implementing industry best practices, the OOI CI cyber-security effort includes a comprehensive cyber-security program based on engagement with the NSF Center for Trustworthy Scientific Cyber-Infrastructure [11].

3.2 Use Case Scenarios

To support various use case scenarios, we define two different usage modes for scientists and end-users based on the architecture presented in Section 2. They are as follows:

(a) *Manual Query and Processing of Data.* Users issue a query for certain data/data-products and associate an analytics workflow with the query for processing the data. The query then triggers data staging resources to be provisioned in the Cloud or CI facility (e.g., at AWS or XSEDE's JetStream), the query to be executed using the OOI web services interface, and the resulting data streamed to the provisioned resources. Computational resources are then provisioned at the CI service, and the analytics workflow is executed to process the streamed data. Finally, the results of the analytics are made available to the users through a separate channel that they can subscribe to.

(b) *Subscription to a Specific Data Stream.* Users request real-time delivery of certain data/data products at a predefined sampling rate. The system then creates a streaming channel, which users can subscribe to, and publishes the desired data at the specified sampling rate to this channel. Using this delivery method, data is pushed to the user in near real-time instead of having users pull the data streams from OOINet.

In this paper, we have selected two representative instruments available in the OOI network as driving use cases: Bottom Pressure Tilt (BOTPT) and Digital Still Camera (CAMDS). While CAMDS allows us to explore different objectives and scenarios, BOTPT helps us conduct a performance evaluation of the streaming engine.

(1) **OOI Bottom Pressure Tilt:** The OOI Bottom Pressure Tilt (BOTPT) Instruments are deployed on the seafloor of the Axial Volcano caldera, approximately 300 miles west of the Oregon Coast. The onboard high-resolution (nano-resolution) pressure sensors sample sea floor (bottom) pressure at 20 S/sec and can effectively provide millimeter resolution of water depth. Currently, BOTPT instruments transmit the pressure data via the OOI submarine cable to the shore, and the real-time data are subsequently stored by the OOI CI. The pressure data are subsequently processed to create various derived products describing seafloor elevation changes, and rates thereof, associated with inflation/deflation of the magma chamber below the Axial volcano. Real-time access to such data products by the seismic and submarine volcano communities is critical for (i) detecting volcanic eruptions at Axial, (ii) monitoring pre- and post-eruption processes, (iii) and planning rapid responses, i.e., research cruises after event detection. This pressure data can also be used for tsunami early detection/warning, and so it is critical that it be made available for easy access by such organizations as the Pacific Tsunami Warning Center. The current OOI CI is not optimal for real-time processing, quality control/evaluation, event detection, and distribution of this high sample rate data to interested scientists and organizations. The ability to store and process this type of data in real-time and push these data products to multiple users on a subscription basis quickly and efficiently is a key requirement.

(2) **OOI Digital Still Camera:** We aim to provide ways for online processing of images from Digital Still Cameras, which are cameras with strobe lights for capturing high-resolution still imagery of water column biology, vents, diffuse flow, seeps, and macrofauna. We have developed an algorithm for object detection to implement data-driven (e.g., content-based) workflows with online analytics, which allows us to disregard dark images or images without regions of interest. OOI deploys multiple digital still cameras (Kongsberg) to provide real-time information on linkages between seismic activity and fluid flow as part of the Cabled Array. The Cabled Array, which includes 900 km of a modified telecommunications cable, provides unprecedented power (10 kV, 8 kW), bandwidth (10 GbE), and two-way communication to scientific sensor arrays on the seafloor and throughout the water column. As the first U.S. ocean observatory to span a tectonic plate, the OOI Cabled Array provides a constant stream of near real-time data from the seafloor and through the water column across the Juan de Fuca plate.

4 IMPLEMENTATION

In order to achieve the real-time data delivery required by the previously described scenarios, we implemented a solution that integrates large-scale scientific facilities with cyberinfrastructure services by leveraging *state-of-the-art* stream processing platforms (i.e., Apache Kafka). The overall architecture of our implementation is depicted in Figure 2. The main components of the system are described below.

(1) *Data Streaming.* The first step in our approach is to provide OOI data as real-time streams. This is achieved by using services that communicate with the OOI M2M interface to fetch the specified data at a predefined sampling rate. This data is then pushed to corresponding topics on an Apache Kafka installation. Kafka is installed on top of fast network appliances, which are optimized for storing, forwarding, or directly using large amounts of data, to provide fast access to the data streams. End-users can subscribe to topics created from the published data.
(2) *Subscription-based Data Transfer.* The data is also transferred to the cyberinfrastructure (e.g., JetStream or S3 Storage) using a Kafka client (running on a separate Virtual Machine - VM), which subscribes to the desired topic and stores the data locally at the corresponding site or zone.
(3) *Data Processing.* Once the desired data is fully transferred to the cyberinfrastructure. The system provisions on-demand computing resources to run the desired workflow (e.g., analytics) on the data. The processed data are then published to a new topic, which users can subscribe to, to receive the processed data.
(4) *Workflow Description.* A user interface is provided to allow users to issue queries to create new topics, subscribe to existing topics, or process data.
(5) *Subscription-based Content Delivery.* The system can also be extended by using multiple geographically distributed network appliances that are connected using high-speed connections (e.g., Internet2), which can enable a Content Delivery Network [25] that optimizes the data delivery based on the user location. The additional appliances can subscribe to desired streams based on user requests and replicate/synchronize the available data accordingly.

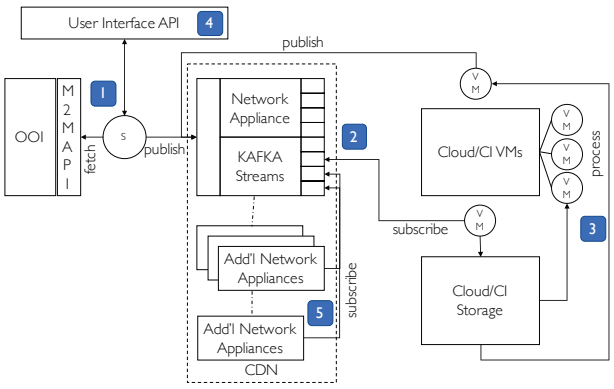

Figure 2: Kafka-based Implementation of the Submarine Framework.

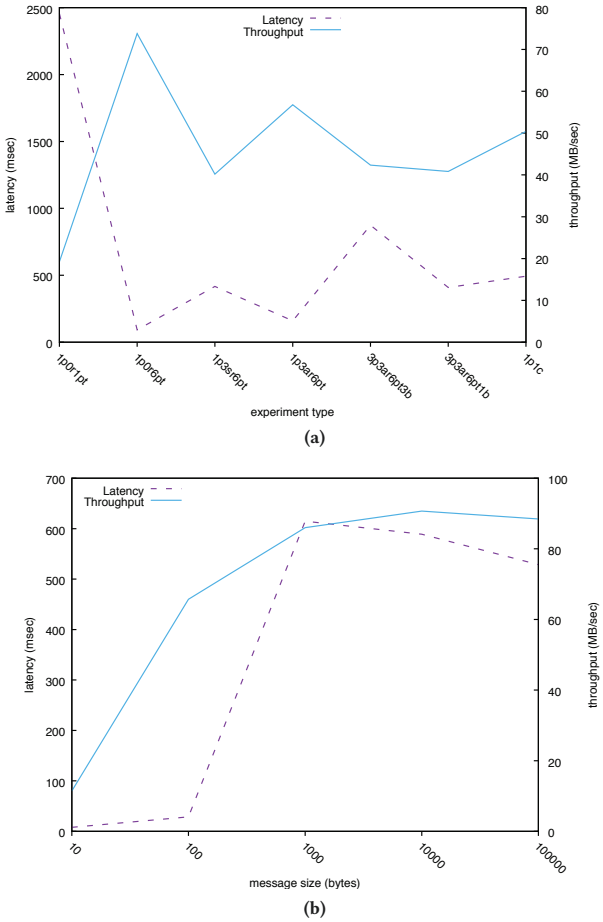

Figure 3: Performance evaluation results using BOTPT data.

5 EXPERIMENTAL EVALUATION

5.1 Performance Evaluation

In this section, we evaluate the performance of our framework using the OOI BOTPT data. We conducted our experiments using a cluster at Rutgers University. The cluster contained seven nodes, where each node had an Intel Xeon 2.4 GHz processor with eight cores, 7200 RPM SAS drive, 24GB of RAM, and 1Gb Ethernet. We used three nodes for the Kafka cluster, one node for Zookeeper, and three nodes for load testing. The experiments evaluated the performance of producing BOTPT data from OOI to the Kafka cluster as well as consuming data from the Kafka cluster. We measured the latency and raw throughput (MB/sec) by producing 50 million messages, where each message is 100 bytes (a total of ~5GB) for the following scenarios:
1p0r1pt: one producer with no replication and one partition.
1p0r6pt: one producer with no replication and six partitions.
1p3sr6pt: one producer with level 3 synchronous replication and six partitions. In this scenario, the partition master waits for an acknowledgment from the replicas before responding back to the producer.
1p3ar6pt: one producer with level 3 asynchronous replication and six partitions. In this scenario, the partition master does not wait

for an acknowledgment from the replicas before responding back to the producer.
3p3ar6pt3b: three producers with level 3 asynchronous replication and six partitions. The producers communicate with three different Kafka brokers.
3p3ar6pt1b: three producers with level 3 asynchronous replication and six partitions. The producers communicate with one Kafka broker.
1prod1cons: In all of the previous scenarios, we ran producers and no consumers, so all messages were persisted but not read. In this scenario, we run one producer and one consumer at the same time.

The results of these experiments are shown in Figure 3. Figure 3b show that 1p0r1pt has the highest latency (2,456.17 milliseconds) and lowest throughput (19.24 MB/sec), which is expected given that this scenario does not take advantage of the full cluster. The results also show that 1p0r6pt has the lowest latency (91.91 milliseconds) and highest throughput (73.80 MB/sec), which is also expected since no replication was required and therefore less data was transferred between brokers. We also evaluated the data production by increasing the message size from 10 bytes to 100,000 bytes. Figure 3b shows that the throughput increases as we increase the message size, which reaches its peak at 10,000 bytes for a throughput of 90.68 MB/sec. The throughput is slightly degraded for the largest message size (88.48 MB/sec), as we reach the hardware limits for the NIC and the hard drive. We also measured the latency and throughput when increasing the message size from 5 million to 50 million messages to see if there is any performance degradation. The latency decreased from 443.57 milliseconds to 159.66 milliseconds with an average throughput of 53.46 MB/sec. Finally, we evaluated the performance of consuming data by measuring the throughput and time taken to consume 50 million messages, where each message is 100 bytes (a total of ~5GB) using a topic with six partitions and level 3 asynchronous replication. The average throughput was 76.18 MB/sec and the average time taken was 63.8 secs. These results establish that the proposed framework is a viable solution that supports the requirements of scientists in the OOI project.

5.2 Digital Still Camera

In this section, we experimentally evaluate several scenarios for the Digital Still Camera data streaming (CAMDS) use case. First, we show that the growth in the number of streams, size of input data, and data generation rate imposes several limitations on general data streaming approaches. Afterwards, three new approaches are introduced to address these limitations.

In our experiments, we use a high-resolution digital camera, which is installed on the Pacific Ocean seabed, as our data producer device. The camera captures high-resolution digital images (one every seven seconds) and sends the images to a dedicated streaming engine. We use a total of 50 images as our input data set. Potential consumers (e.g., scientists interested in the data) subscribe the streaming engine to receive the images.

Figure 5 shows a simple streaming approach, which is used as our baseline. The image resolution and size are substantial, therefore, to reduce unnecessary overhead on the streaming engine, each image is sliced into smaller pieces before being sent to the streaming engine (image slicing step in Figure 5). Similarly, on the consumer side, the image slices are stitched back together to form the original image (image reconstruction step in Figure 5). Finally, to detect the available objects in the reconstructed images, an object detection algorithm is applied to each image (object detection step in Figure 5). Figure 4a illustrates one of the images that was captured by the camera and Figure 4b shows the processed image, which shows that an object (i.e., a fish) has been detected in the lower right side of the picture.

(a) Input Image (b) Output Image

Figure 4: Input raw image captured by Digital Still Camera vs. processed image.

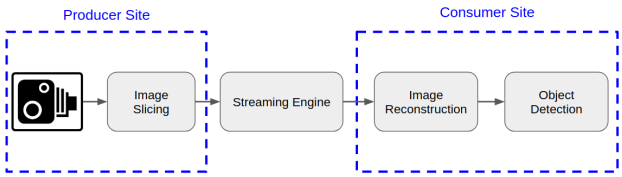

Figure 5: A Simple Workflow Implemented Using the Streaming Platform (Basic Streaming).

As mentioned in Section 3, infrastructure within large observatories are usually connected using high-speed network links. However, compared to the network connection between observatory infrastructure, the network bandwidth is substantially lower on the consumer side. Hence, to characterize the latency of streaming data for different network bandwidths and its overall impact on streaming performance, we used a Hierarchical Token Bucket (HTB) [5] tool to control the bandwidth between the consumer site and the streaming engine. In these experiments, latency represents the amount of time it takes for all slices of an image to get from the producer site to the consumer site. We used four distinctive bandwidths (1MB, 2MB, 5MB, and 10MB) for the connection between the consumer and the streaming engine. The latency results for the input image sequence are shown in Figure 6.

Figure 6 shows that in the case of high bandwidth connections between the consumer and the streaming engine (e.g., 5MB and 10MB), the latency remains within a boundary and the consumer can keep up with the producer's data generation rate. However, if the consumer is connected to the streaming engine using a low network bandwidth connection, the data transfer between the engine and the consumer becomes a bottleneck. Consequently, in these cases, the consumption rate is less than the production rate. Figure 6 also shows that in the case of low bandwidth connections, there is a linear increase in the latency for a sequence of consecutive images.

This can be attributed to the use of the streaming engine, which functions here as a buffer with its inputs greater than its output. As a result, as the input/output rate remains constant, the size of the data in the buffer grows and the queue time for new input data increases. It is easy to see that the latency accumulation can be worse (i.e., latency grows at a higher rate) if, for example, the camera capture rate or the image resolution/size increases. Moreover, this accumulation can also appear in high bandwidth connections when the input data (e.g., the number of images) increases.

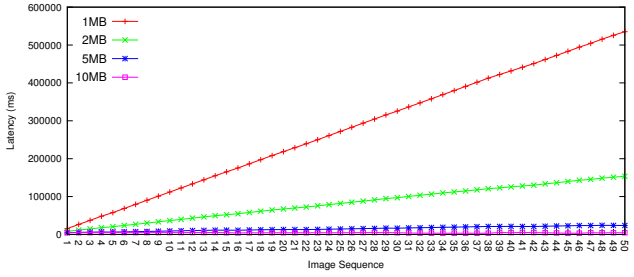

Figure 6: Latency achieved for a sequence of images using Basic Streaming.

To resolve this latency accumulation problem, we moved the computation from the consumer's site to the streaming engine to create a *streaming and processing engine*. This approach has many advantages. First, it allows the data to be processed as it moves toward the consumer. Moreover, this technique allows the data to be delivered to the consumer based on its content (i.e., if the data content is not interesting to the consumer it can be discarded). The early filtering of the content before it reaches the system's bottleneck helps reduce the load on the streaming engine, which partially resolves the latency accumulation problem. The overall flow of this technique is shown in Figure 7.

Figure 8 shows the latency results for this approach. Zero latency means that the image was discarded and not streamed to the user (i.e., there were no fish detected in the image). This technique is beneficial in the case where the input data does not contain useful information. In this experiment, 15 out of the 50 images contained at least one object, and the rest of the images (35) were discarded. Figure 8 shows that the latency accumulation was resolved and only the processed data was delivered to the consumer. However, due to the several extra processing stages at the streaming engine (i.e., Image Reconstruction, Object Detection, and Image Slicing), the overall latency of the delivered data is slightly higher than the basic streaming approach. Furthermore, this approach cannot solve the latency accumulation issue when all of the input data contain valuable information, (i.e., should be delivered to the consumer). This can be seen in Figure 8, where several successive images contain one or more objects (image 39 to 42) and should be received by the consumer. In this situation, if the consumer's connection bandwidth is low the latency starts to grow.

Another approach that we developed to address the latency accumulation issue is to continuously monitor the latency and use up-to-date latency information for future images. The general flow of this approach is presented in Figure 9. The main component of this approach is a decision-making step that compares the latency

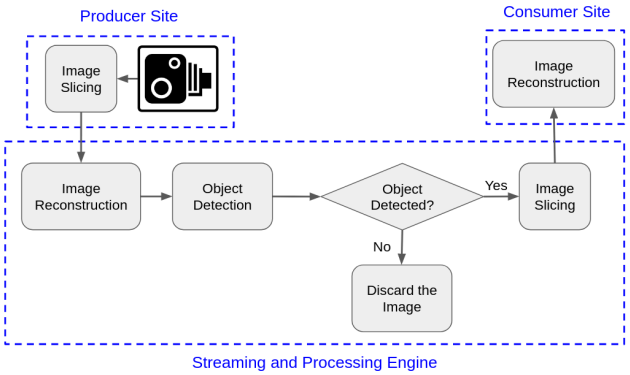

Figure 7: Early detection approach using a streaming and processing platform.

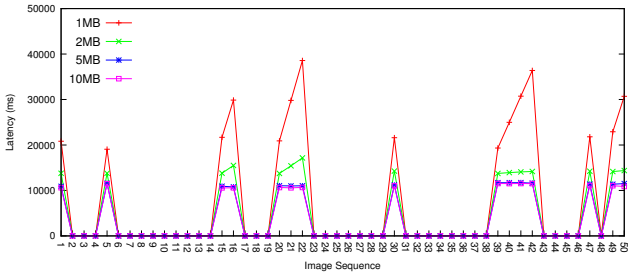

Figure 8: Latency achieved for each image using the proposed early detection approach.

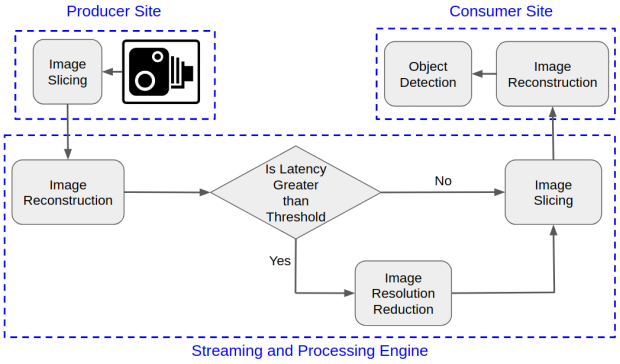

Figure 9: Content approximation approach using a streaming and processing platform.

information of the latest image delivered to a consumer with a predefined threshold. If the latency is more than the threshold, then the streaming engine will reduce the image resolution and send a low-resolution image to the consumer instead. Otherwise, the original high-quality image is delivered to the consumer.

The results of this approximation approach are presented in Figure 10. For high bandwidth consumers connections (i.e., 5MB and 10MB), the latency is always less than the threshold, and the latency accumulation problem never happens. However, for low bandwidth connections (i.e., 1MB and 2MB), approximation and resolution reduction solve the latency accumulation problem. In the 1MB and 2MB cases, figure 10 shows that the resolution reduction

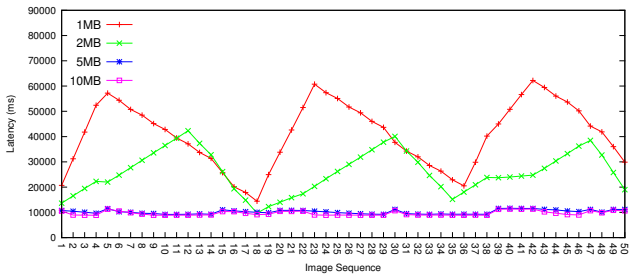

Figure 10: Latency achieved for each image using the proposed approximation approach.

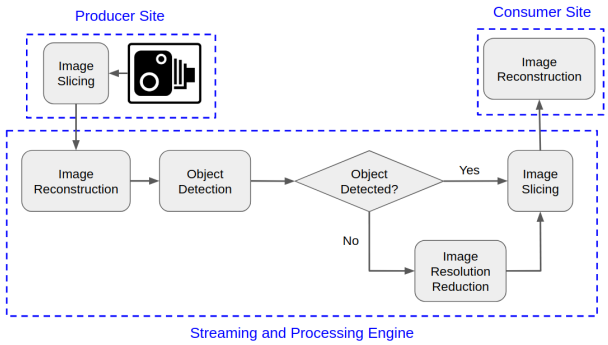

Figure 11: A hybrid early detection and approximation approach using a streaming and processing platform.

can decrease the amount of latency and keep it within a reasonable boundary. When the latency falls below the threshold, the original image quality is delivered to the consumer, which makes the latency grow again. It is also clear that reducing the image quality/resolution decreases the size of data that needs to be transferred, which reduces the pressure on the bottleneck connection.

Finally, we considered a hybrid approach that combines the early detection and the approximation approaches. Figure 11 demonstrates the flow of this hybrid approach. In this approach, we added a component to decide between low-resolution and high-resolution images. The decision for this component is based on the outcome of the object detection algorithm. If at least one object is found in the picture, a high-resolution image is delivered to the consumer. Otherwise, a low-resolution is delivered. The goal of this approach is to ensure that all images are delivered to the consumer while trying to address the latency accumulation problem. Similar to the early detection approach, this approach is beneficial in the case where most of the input data do not contain valuable information for the consumer.

Figure 12 collects the latency results of this hybrid approach. It can be observed that the latency gradually goes down when the image does not contain any objects. Moreover, in low bandwidth conditions, if several consecutive images contain useful information, the latency increases with almost a constant rate.

6 BACKGROUND AND RELATED WORK

The work presented in this paper is complementary to efforts such as GeoSciCloud [4], which explores how cloud services can support the core functionality provided by facilities, and Globus Software as

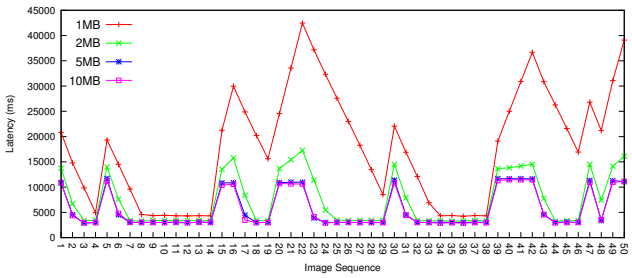

Figure 12: Latency achieved for each image using the proposed hybrid early detection and approximation approach.

a Service [2], which focuses on the transfer, sharing, and publishing of data. Farcas et al. [3] discussed requirements for service composition in large-scale software systems using OOI as a potential use case for their proposed architecture. The authors envisioned that a central data management system that ingests data and serves them to users on a query basis could be complemented with a highly distributed set of capabilities to facilitate a range of tasks ocean scientists would engage in. However, the implemented models based on subscription and stream processing platforms described in this paper were not envisioned at that time.

Processing large amount of data using heterogeneous resources connected to each other through wide area networks has been discussed in several papers which are complementary to our work. Vulimiri et al. [24] have presented the concept of wide area big data (WABD) and have argued that processing a large amount of data using distributed resource generates substantial network traffic which limits the overall performance of the system. Their proposed system which is called WANalytics explores how to push computation toward the edge to reduce network traffic. A similar concept has been discussed by Kloudas et al. [7] who explore how to map tasks to resources to reduce network traffic. Pu et al. [16] have explored minimizing latency in wide area analytics. They proposed a greedy heuristic optimization technique to find the best data and task placement. Our previous work [27, 28] proposed a computational model to extract waiting/queuing time of over-provisioned destination data centers and use network resources to process data at in-transit nodes during waiting time.

There are several streaming systems available such as Borealis [1], Storm [21] and Heron [9] that have been developed to run within one single data center. However, our streaming engine (i.e., the messaging system) that we described in section 2 is designed to run across distributed resources and data centers. In this paper, we specifically use Kafka [8], which is a distributed messaging system and a streaming platform. Kafka is mainly used for either real-time streaming of data pipelines between systems or applications or for building data-driven applications that can react to real-time data.

Stream processing/analytics was the main focus of several other research papers. Tudoran et al. [22] talked about a stream of events across data centers. Their proposed architecture monitors the available bandwidth between data centers and reacts accordingly to increase data transfer rate. Rabkin et al. [17] presented a streaming engine called Jetstream that addresses wide area stream queries with latency bound requirements and deals with network bandwidth limitations. Santos et al. [20] described the advantages of

distributed stream processing. They showed in their paper that filtering and preprocessing functions at edge clouds can reduce the impact of bandwidth limitations in distributed processing. Furthermore, in [6], replication based stream processing has been proposed over wide area network nodes to process downstream and react to events at the earliest time. Pietzuch et al. [15] have considered pushing distributed stream processing operators to network node automatically. Their proposed solution reduces the streaming latency and improves network utilization. However, aside from stream processing and analytics, our approach tries to provision processing resources at the proper location and utilizes approximation and content delivery techniques to overcome network limitations in delivering processed or raw data to the consumers.

7 CONCLUSION AND FUTURE WORK

This paper presented our initial efforts and experiences that complement our work in the OOI project, to explore more effective data delivery mechanisms, based on subscription-based data streaming, and to better integrate large facilities with cyberinfrastructure services. We presented the architecture, implementation, and performance of submarine, a subscription-based data streaming framework for OOI data delivery, and its integration with public CI services for automated data processing. The overarching objective of this effort is to improve the accessibility of data and the way scientists interact with both data sources and computational infrastructures, as well as the overall effectiveness and impact of current open, experimental, and observational facilities.

The presented work specifically targeted the end-to-end delivery and processing of the high-resolution pressure data and derived products from the OOI BOTPT instruments to users as well as CAMDS images, and leveraged enterprise data streaming and data processing technologies such as Apache Kafka to implement subscription-based data delivery and automated data processing. An experimental evaluation of the solution was also presented. Future work includes leveraging experiences with this solution to implement similar data delivery mechanisms for OOI high-bandwidth seismic (i.e., high-resolution tilt) data and acoustic data (e.g., hydrophones), as well as lower temporal resolution OOI seafloor pressure sensor data, to multiple users and data repositories.

ACKNOWLEDGMENTS

This research is supported in part by NSF via grants numbers ACI 1339036, ACI 1441376, ACI 1464317 and OCE 1745246. The research at Rutgers was conducted as part of the Rutgers Discovery Informatics Institute (RDI^2).

REFERENCES

[1] Daniel J Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Cetintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag Maskey, Alex Rasin, Esther Ryvkina, et al. 2005. The Design of the Borealis Stream Processing Engine.. In *Cidr*, Vol. 5. 277–289.

[2] Bryce Allen, Rachana Ananthakrishnan, Kyle Chard, Ian Foster, Ravi Madduri, Jim Pruyne, Stephen Rosen, and Steve Tuecke. 2017. Globus: A Case Study in Software as a Service for Scientists. In *Proceedings of the 8th Workshop on Scientific Cloud Computing*. ACM, 25–32.

[3] Claudiu Farcas, Emilia Farcas, and Ingolf Krüger. 2010. *Requirements for Service Composition in Ultra-Large Scale Software-Intensive Systems.* Springer Berlin Heidelberg, Berlin, Heidelberg, 93–115. https://doi.org/10.1007/978-3-642-12566-9_6

[4] GeoSciCloud. 2017. https://www.earthcube.org/group/geoscicloud-deploying-\multi-facility-cyberinfrastructure-commercial-private-cloud-based-systems. (2017). Last accessed on September 2017.

[5] Hierachial Toekn Bucket. 2017. https://en.wikipedia.org/wiki/Token_bucket. (2017). Last accessed on September 2017.

[6] Jeong-Hyon Hwang, Ugur Cetintemel, and Stan Zdonik. 2007. Fast and reliable stream processing over wide area networks. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*. IEEE, 604–613.

[7] Konstantinos Kloudas, Margarida Mamede, Nuno Preguiça, and Rodrigo Rodrigues. 2015. Pixida: optimizing data parallel jobs in wide-area data analytics. *Proceedings of the VLDB Endowment* 9, 2 (2015), 72–83.

[8] Jay Kreps, Linkedin Corp, Neha Narkhede, Jun Rao, and Linkedin Corp. 2011. Kafka: a distributed messaging system for log processing. NetDB'11. (2011).

[9] Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M Patel, Karthik Ramasamy, and Siddarth Taneja. 2015. Twitter heron: Stream processing at scale. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 239–250.

[10] Avinash Lakshman and Prashant Malik. 2009. Cassandra: structured storage system on a p2p network. In *Proceedings of the 28th ACM symposium on Principles of distributed computing*. ACM, 5–5.

[11] NSF Center for Trustworthy Scientific Cyberinfrastructure. 2017. https://trustedci.org. (2017). Last accessed on September 2017.

[12] Ocean Observatories Initiative Data Portal (OOINet). 2017. https://ooinet.oceanobservatories.org. (2017). Last accessed on September 2017.

[13] Ocean Observatories Initiative Web Site. 2017. http://oceanobservatories.org. (2017). Last accessed on September 2017.

[14] Open source amqp messaging. Qpid, Apache. 2017. http://qpid.apache.org. (2017). Last accessed on September 2017.

[15] Peter Pietzuch, Jonathan Ledlie, Jeffrey Shneidman, Mema Roussopoulos, Matt Welsh, and Margo Seltzer. 2006. Network-aware operator placement for stream-processing systems. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*. IEEE, 49–49.

[16] Qifan Pu, Ganesh Ananthanarayanan, Peter Bodik, Srikanth Kandula, Aditya Akella, Paramvir Bahl, and Ion Stoica. 2015. Low latency geo-distributed data analytics. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 421–434.

[17] Ariel Rabkin, Matvey Arye, Siddhartha Sen, Vivek S Pai, and Michael J Freedman. 2014. Aggregation and Degradation in JetStream: Streaming Analytics in the Wide Area.. In *NSDI*, Vol. 14. 275–288.

[18] Ivan Rodero and Manish Parashar. 2016. Architecting the cyberinfrastructure for National Science Foundation Ocean Observatories Initiative (OOI). In *7th International Workshop on Marine Technology: MARTECH 2016*. 99–101.

[19] Ivan Rodero and Manish Parashar. 2017. Ocean Observatories Initiative Cyber-Infrastructure White Paper http://nsfcac.rutgers.edu/ooi/ooi-ci-wp.pdf. (2017). Last accessed on September 2017.

[20] Ivo Santos, Marcel Tilly, Badrish Chandramouli, and Jonathan Goldstein. 2013. DiAl: distributed streaming analytics anywhere, anytime. *Proceedings of the VLDB Endowment* 6, 12 (2013), 1386–1389.

[21] Ankit Toshniwal, Siddarth Taneja, Amit Shukla, Karthik Ramasamy, Jignesh M Patel, Sanjeev Kulkarni, Jason Jackson, Krishna Gade, Maosong Fu, Jake Donham, et al. 2014. Storm@ twitter. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 147–156.

[22] Radu Tudoran, Olivier Nano, Ivo Santos, Alexandru Costan, Hakan Soncu, Luc Bougé, and Gabriel Antoniu. 2014. Jetstream: Enabling high performance event streaming across cloud data-centers. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*. ACM, 23–34.

[23] Alvaro Videla and Jason JW Williams. 2012. *RabbitMQ in action: distributed messaging for everyone*. Manning.

[24] Ashish Vulimiri, Carlo Curino, Brighten Godfrey, Konstantinos Karanasos, and George Varghese. 2015. WANalytics: Analytics for a Geo-Distributed Data-Intensive World.. In *CIDR*.

[25] Joel M Wein, John Josef Kloninger, Mark C Nottingham, David R Karger, and Philip A Lisiecki. 2007. Content delivery network (CDN) content server request handling mechanism with metadata framework support. (July 3 2007). US Patent 7,240,100.

[26] XSEDE JetStream Cloud. 2017. http://jetstream-cloud.org. (2017). Last accessed on September 2017.

[27] Ali Reza Zamani, Mengsong Zou, Javier Diaz-Montes, Ioan Petri, Omer Rana, Ashiq Anjum, and Manish Parashar. 2017. Deadline constrained video analysis via in-transit computational environments. *IEEE Transactions on Services Computing* (2017).

[28] Ali Reza Zamani, Mengsong Zou, Javier Diaz-Montes, Ioan Petri, Omer Rana, and Manish Parashar. 2017. A computational model to support in-network data analysis in federated ecosystems. *Future Generation Computer Systems* (2017).