Enabling Scalable and Reliable Real Time Data Services for Sensors and Devices in StreamCI

Jaewoo Shin*, Lan Zhao†, Carol X. Song†, Rajesh Kalyanam†, Jian Jin‡, Jacob D. Hosen§, Ananth Grama* and Dongyan Xu*
* *Department of Computer Science*
† *Research Computing*
‡ *Department of Agricultural & Biological Engineering*
§ *Department of Forestry & Natural Resources*
{shin152, lanzhao, cxsong, rkalyanam, jinjian, jhosen, ayg, dxu}@purdue.edu
Purdue University

Abstract—**Rapid advances in technology over the past decade have enabled collection of large amounts of data, in particular, through data streams from sensors and devices. Effective utilization of such data has been hampered by the lack of ready-to-use resources for data providers to manage the data and for data consumers to access the data through facile APIs. In this paper, we present StreamCI, a scalable cloud-based sensor data collection and analysis system that enables researchers to easily collect, process, store, and access large volumes of heterogeneous sensor data. StreamCI provides a web portal for users and administrators to easily register new data sources and monitor the status of data ingestion pipelines. The back end of StreamCI provides real time data ingestion/query APIs, data access control, and data processing pipelines using an open source software stack including RabbitMQ, Node.js, MongoDB, Certbot, Grafana, and HUBzero. Containerization and orchestration of services using Kubernetes improves scalability, as demonstrated by our experimental results. The StreamCI system has been used in multiple research and education domains including the collection and processing of plant health sensor data by plant phenotype researchers, collection of real-world air quality sensor data and its use in data analysis coursework for ecology students, and the collection and analysis of advanced manufacturing data for cybersecurity research.**

Index Terms—**StreamCI, Cyberinfrastructure, Sensor data, MyGeoHub, Scalability, Kubernetes**

I. Introduction

Rapid advancements in sensing and communication technologies have resulted in large volumes of streaming data from Internet of Things (IoT), smart manufacturing, and precision agriculture, to name a few. Effective management and utilization of this data can enable significant advances in diverse areas, ranging from cybersecurity to sustainability and food security. Managing streaming sensor data brings significant challenges to researchers and practitioners, who are typically not familiar with the complexity of underlying computer systems and often lack the skills needed to develop and host their own solutions. These data typically come from different sensor devices, in high volumes, often have different formats, and use different network protocols. Moreover, these data are continuously generated and typically need to be handled and processed at real time. There have been a few efforts in recent years to address these challenges, however, a scalable facile cyberinfrastructure (CI) solution that seamlessly supports collecting, processing, managing, and analyzing such heterogeneous and continuous data is lacking. One notable effort is the CHORDS project [1], which provides data services to help geoscience researchers to acquire, navigate, and distribute real-time data. Users can insert or query data using the CHORDS portal or via HTTP requests. The system enables users to visualize the data using Grafana [2] visualization services. While the system is easy to use, there are several limitations in terms of the types of data and queries supported, and the scalability of the system.

To help overcome these challenges, we have developed StreamCI, an easy-to-use, flexible, and scalable cloud based streaming sensor data management CI system that enables individual researchers to easily collect, manage, and access real time sensor data. StreamCI is built upon a prototype system called SACI [3]. which provides basic support for data ingestion and processing using a RabbitMQ and a container based architecture. StreamCI extends SACI with value added services including OAuth based authentication, fine-grained data access control, automated SSL certificate renewal, and scalable and reliable services through the use of Kubernetes and Purdue's Geddes composable cloud infrastructure [4].

In the following sections, we first describe the system design and implementation of StreamCI, followed by benchmarking results that illustrate the reliability and scalabilty of the system. We then describe how StreamCI has been used to support research and education activities in three use cases from diverse disciplines. We discuss future work and conclusions in Section V.

II. System Design and Implementation

Developed as part of the NSF funded GeoEDF project [5], the overall goal of StreamCI is to develop a turnkey solution that enables researchers to easily connect their sensor data sources with automatically generated StreamCI data processing and management pipelines with minimal configuration steps that can be typically completed in just a few mouse

Presented at Gateways 2022, San Diego, USA, October 18–20, 2022

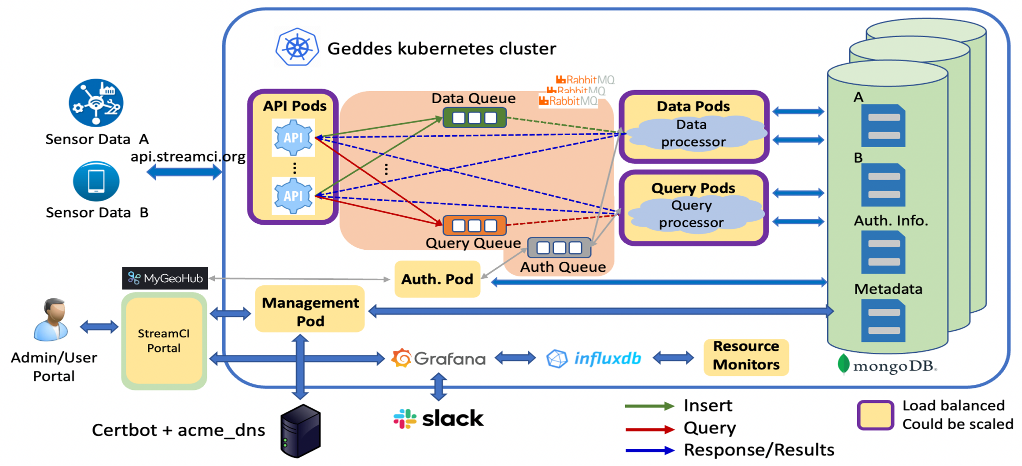


Fig. 1: StreamCI Architecture Overview.

clicks. As shown in Figure 1, StreamCI is a scalable and extensible cloud based sensor data management and analysis platform that is developed using an open source software stack. It provides a portal interface that allows data owners to easily manage their data streams. They can register new data sources, view a sample of the latest ingested data, the number of data records ingested, and the last ingestion time on an easy-to-use dashboard, plot time series using a flexible data viewer, and modify the data table schema by themselves. Once users register a new data source in StreamCI, it automatically sets up all the required data pipelines at the back end. It also allows administrators to approve new registration requests and monitor the overall status of the system including CPU/memory usage, data access response time, message queue availability, and system logs, to name a few. The StreamCI portal is implemented as a HUBzero [6] web component and deployed on MyGeoHub [7], which is a science gateway for geospatial data driven research, education, and collaboration.

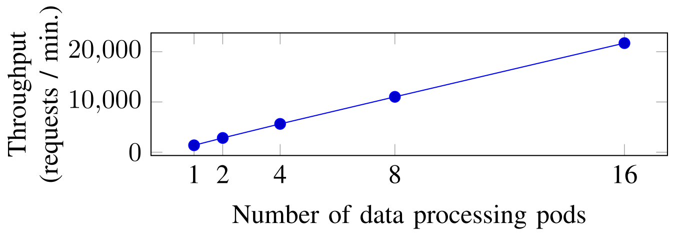
The back end of StreamCI is deployed on Purdue's Geddes composable system and consists of several groups of container pods managed by Kubernetes, including: (1) API pods that provide data ingestion and query REST APIs implemented using Node.js; (2) RabbitMQ pods that implement a high-availability, mirrored queue cluster for dynamically dispatching user requests to either a data queue, a query queue, or an authentication queue; (3) data processing pods that receive ingested data from the data queue and perform some preprocessing steps, such as metadata/schema validation, before storing the data in a MongoDB database table; (4) query processing pods that perform authorization and query syntax validation before fetching data from the database; (5) an authentication pod that receives an authentication request from a data or query pod and creates or renews a user's authentication status by performing OAuth authentication to the MyGeoHub OAuth server; (6) a management pod that manages the data sources and their metadata, provides registration and data management APIs to the portal, and automatically renews and installs an SSL certificate for the REST API endpoints to support secured message protocol using Certbot and acme-dns; and (7) a replica set of MongoDB that stores data, metadata, and authentication information. On top of these core components, StreamCI implemented a resource monitoring/alert system by integrating InfluxDB, Grafana, and Slack to allow system administrators to quickly respond to system failures.

One of the common requirements from the community we have been working with is to support sensor data that is geospatially referenced. StreamCI supports ingestion of GeoJSON data type (i.e., Point, LineString, Polygon, and other GeoJSON types standardized in RFC 7946) and geospatial queries. When a user specifies an attribute type as GeoJSON at the data source registration step, the system creates a geospatial index on the attribute in MongoDB, which enables users to query the data with geospatial query operators, such as $geoIntersects and/or $geoWithin.

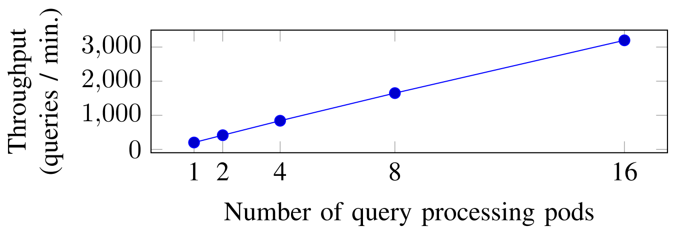
StreamCI allows users to customize access control on their data for sharing and collaboration. Each user is assigned a role from three categories for each data source: SuperUser, DataProvider, and GeneralUser. SuperUser has read and write access to all data sources. DataProvider by default only has read and write access to her own data sources and read access to public data. GeneralUser does not own any data sources and can only read data that is set to be public. The system lets users read (query) or write (ingest) data into the system only if the user has the appropriate permissions. In addition, StreamCI allows users to set the privacy level when they ingest data into the system. By default, the privacy level is set to private, but can be changed to public or shared with specific users.

III. Performance Evaluation

As shown in Figure 1, the core components of StreamCI (i.e., API, Data, and Query pods) are orchestrated via Kubernetes and dynamically scaled. The system monitors the CPU usage of the components. Once the usage hits a threshold, it assigns more pods and balances the workloads. To show how many data ingestion and query requests can be processed by StreamCI, we measured the ingestion and the query throughput on Geddes, a community cluster at Purdue equipped with Dell Compute nodes with two 64-core AMD EPYC Rome processors. We deployed StreamCI with five MongoDB replica sets (each with 4 CPU and 16 GB Memory) and three mirrored RabbitMQ clusters (2 CPU/16 GB each) for stable and reliable services. For the processing pods, we assigned the CPU/memory usage limit for API (0.5CPU/8GB), Data (0.2CPU/2GB), and Query (0.2CPU/8GB), respectively. To measure the system performance of varying data/query processing pods, we disabled auto-scaling of the API pods and launched 16 API pods for all experiments.



(a)



(b)

Fig. 2: Data ingestion/query throughput

First, we measured the data ingestion throughput of StreamCI by varying the number of data processing pods. For this test, we used dummy data sources that send data with three random string values (length of 4, 8, and 16 respectively), two random float values (to simulate latitude/longitude coordinates), two integer values, and one timestamp. The test data producers ran on separate VMs and spawned over 120 data streams. We then captured the maximum number of data units processed per minute by 1, 2, 4, 8, and 16 data processing pods. As shown in Figure 2a, 1 data processing pod could process 1,375 requests per minute and the throughput increased linearly as more processing pods were deployed. In this test, if we have enough data processing pods so that a request does not need to wait in a message queue, it takes less than 20 ms for an ingestion request to be processed and return the status.

Similar experiments were carried out to measure the data query performance in StreamCI. We first fed dummy data into StreamCI at 4,250 units per minute and the test queries were set to fetch the recent [1,2] minute(s) of data (i.e., the size of query results is [4,250, 8,500]). Similar to the ingestion test, 1 query processing pod could process 202 queries per minute on average and the capacity increased as more resources were assigned for query processing. Although the response time of a query strongly depends on the query type and the size of the results, our test queries could be returned within 200 ms on average.

In our StreamCI deployment, horizontal pod autoscaling is set on the API, Data, and Query pods so that the resources are dynamically increased or decreased for a given data ingestion or query workload. This feature is critical for providing reliable real-time or near real-time services to handle large volumes of streaming data.

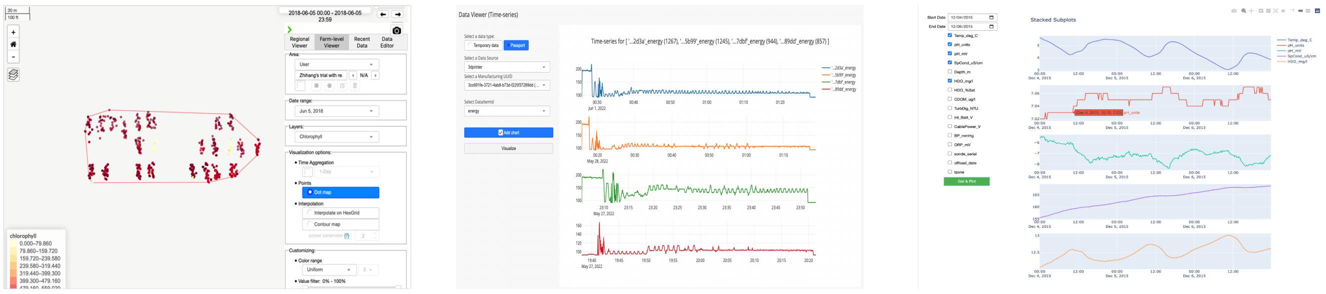
IV. Use Cases

The StreamCI system is designed to be flexible enough to work with sensor data from different domains. It has been successfully used by several research projects in multiple disciplines to collect and manage data from diverse sensor devices.

A. Plant Health Sensor Data Collection

Developed by Professor Jin's research group in the Department of Agricultural and Biological Engineering at Purdue University, LeafSpec [8] is an innovative low cost handheld hyperspectral imager that can scan a leaf non-destructively and provide measurements of the plant's physiological features including leaf moisture content, chlorophyll content, nitrogen content, pathogen and insect diseases, and stresses from chemical sprays within 10 seconds. It has been used by researchers to perform advanced plant phenotyping research to raise and select crops that are high yield and resistant to pests and drought stress as well as by farmers to monitor the health of their crops and to direct fertilizer and pesticide application in the field. In addition to getting immediate feedback while scanning a plant using the device, it is also important to collect the data on the server side and make it available for more advanced analysis and decision making. To this end, StreamCI has been successfully used to provide a platform for real-time crop growth data collection, processing, and exploration from distributed handheld LeafSpec sensors.

In each measurement, a raw leaf image is processed in real-time by the onboard software, and the result can be viewed through a smartphone mobile app, which is connected with the sensor via bluetooth. The current measurement results include the spectra of leaf pixels, leaf's morphological features, and plant physiological features such as Normalized Difference Vegetative Index (NDVI), Leaf Relative Water Content (RWC), Leaf Nitrogen Content, etc. The mobile app immediately ingests the georeferenced (with the smartphone's GPS) imaging results to StreamCI via its REST API, making it possible for users to access and visualize field collected plant hyperspectral imaging data on various maps online in real time. Figure 3a shows a group of plant sensor data collected at Purdue's ACRE farm in July 2018 for different maize phenotypes using a customized web application tool that interacts with StreamCI's MongoDB back end using its REST API.



(a) (b) (c)

Fig. 3: (a) LeafSpec plant health data collected at Purdue ACRE farm; (b) Energy usage comparisons of data from a 3D printer; (c) Air quality sensor data used in ecology classes.

B. Secure Manufacturing Research

StreamCI has also been adapted and extended for the CyManII secure architectures. The goal of this project is to build a scalable, generic end-to-end architecture that connects manufacturing shop floors to the cloud for the collection, transmission, curation, and analytics of manufacturing data, for a wide range of applications including manufacturing anomaly detection, quality monitoring, energy profiling and optimization, and supply-chain management. StreamCI was enhanced to offer the following key features/functions: (1) integrates disparate manufacturing data streams from instrumented processes in manufacturing; (2) exposes a consistent API and data schema for analytics applications to achieve cyber-physical security; (3) integrates energy monitors and implements basic energy accounting services; (4) performs authentication, encryption, and privacy services for exporting data to external (global) supply chain ledgers; (5) implements feature extraction, compression, and storage indices; and (6) maintains metadata of manufacturing data streams, along with associated manufacturing semantics and state, for product provenance tracking.

C. Ecological Sensor Class

StreamCI has been used to collect ecological sensor data for Purdue's "Ecological Sensors and Data" course in the spring semester of 2021. Three environmental sensors were set up by the instructor and the data were pushed to StreamCI endpoints in real time using the webhook mechanism provided by Particle.io. More specifically, air quality sensors publish data such as temperature, humidity, and dust to the Particle platform at various frequencies. Once the webhook detects a data event, it reads the data attributes to send to StreamCI, makes a JSON-formatted object for ingestion, and sets additional fields for user authentication to StreamCI before sending the data to StreamCI's REST endpoint. More than 1 million records have been received from the sensors. Figure 3c shows one example time series plot of the air quality sensor data collected in StreamCI. During the course, students learned how to use environmental sensors to collect and analyze data to understand ecosystems using sensor data from the StreamCI system.

V. Conclusions

In this paper we presented the design and implementation of StreamCI, a general purpose scalable cyberinfrastructure for practitioners and researchers to collect, manage, and make their sensor data available to the community. It has demonstrated success in applications from multiple disciplines to harvest and utilize large volumes of sensor data. Our future work includes enhancing the user/admin portal functions, developing real time data quality control services, and implementing a standalone StreamCI that can be easily deployed on most existing cloud resources.

Acknowledgment

- This project is funded by the NSF awards no. 1835822.
- This material (IV-B) is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Advanced Manufacturing Office Award Number DE-EE0009046. The views expressed herein do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

References

[1] R. Gooch and V. Chandrasekar, "Integration of real-time weather radar data and internet of things with cloud-hosted real-time data services for the geosciences (chords)," in *2017 IEEE international geoscience and remote sensing symposium (IGARSS)*. IEEE, 2017, pp. 4519–4521.

[2] "Grafana." [Online]. Available: https://grafana.com/

[3] J. Shin, L. Zhao, C. Song, R. Kalyanam, and J. Jin, "Saci - a cloud based real-time sensor data management and analysis platform," Gateways 2020 conference, October 2020.

[4] "Geddes," Purdue University Research Computing. [Online]. Available: https://www.rcac.purdue.edu/compute/geddes

[5] R. Kalyanam, L. Zhao, X. C. Song, V. Merwade, J. Jin, U. Baldos, and J. Smith, "Geoedf: An extensible geospatial data framework for fair science," in *Practice and Experience in Advanced Research Computing*, 2020, pp. 207–214.

[6] M. McLennan and R. Kennell, "Hubzero: A platform for dissemination and collaboration in computational science and engineering," *Computing in Science & Engineering*, vol. 12, 2010.

[7] R. Kalyanam, L. Zhao, C. Song, L. Biehl, D. Kearney, I. L. Kim, J. Shin, N. Villoria, and V. Merwade, "Mygeohub—a sustainable and evolving geospatial science gateway," *Future Generation Computer Systems*, vol. 94, pp. 820–832, 2019.

[8] L. Wang, J. Jin, Z. Song, J. Wang, L. Zhang, T. U. Rehman, D. Ma, N. R. Carpenter, and M. R. Tuinstra, "Leafspec: An accurate and portable hyperspectral corn leaf imager," *Computers and Electronics in Agriculture*, vol. 169, p. 105209, 2020.