ERGO: A Scalable Edge Computing Architecture for Infrastructureless Agricultural Internet of Things

Deepak Nadig, Sara El Alaoui and Byrav Ramamurthy
Dept. of Computer Science & Engineering
University of Nebraska-Lincoln, Lincoln, NE, USA
{deepaknadig,sara,byrav}@cse.unl.edu

Santosh Pitla
Dept. of Biological Systems Engineering
University of Nebraska-Lincoln, Lincoln, NE, USA
spitla2@unl.edu

2021 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN) | 978-1-6654-4579-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/LANMAN52105.2021.9478811

***Abstract*—In this paper, we propose ERGO (edge architecture for Ag-IoT), an edge-computing architecture for infrastructureless smart agriculture environments. We also develop Ag-IoT application APIs and the associated microservice infrastructure. Our implementation and evaluations show that ERGO can operate independently of cloud-backed assistance, is highly scalable, modular, and affords composability benefits to Ag-IoT systems. We also demonstrate that ERGO outperforms traditional infrastructure in response latencies and transactional throughput, on average, by over 54% and 77%, respectively.**

***Index Terms*—Edge Computing, Ag-IoT, Infrastructureless.**

I. Introduction

With the emergence of the Internet of Things (IoT) in precision agriculture [1], [2], there is an increasing push towards integrating heterogeneous devices, machines and digital objects with automation in the virtual world. Although IoT empowers agriculture with smart and intelligent decision-making tools to integrate agricultural implements, knowledge and services for improved productivity and yield gains, critical challenges exist. In-field sensors and imaging systems generate large quantities of high-resolution datasets. Therefore, real-time data processing to obtain valuable insights can be challenging due to the lack of high-performance computing infrastructure, locally. Numerous solutions propose using cloud resources for data processing and decision-making insights, and they assume the availability of adequate bandwidth and connectivity for high-speed data movement. Thus, a key challenge is to provide a *scalable, low-cost edge computing solution for environments (e.g., rural areas) characterized by limited/intermittent connectivity* that are typical of agricultural Internet of things (Ag-IoT) ecosystems.

In this paper, we propose ERGO, a scalable edge computing architecture for Ag-IoT environments. ERGO works in connectivity-challenged environments, with limited (possibly periodic) or no wide area network (WAN) connectivity. Further, due to limited hardware resources available to edge computing infrastructure, we design ERGO to be highly composable and scalable to handle the dynamic needs of Ag-IoT.

II. ERGO Architecture

Our proposed ERGO architecture, shown in Figure 1, combines container orchestration with scalable web-services to provide Ag-IoT services over RESTful APIs. Our proposed architecture is shown in Figure 1a. It shares some common elements with existing edge-cloud solutions such as reconfigurable compute/storage/networking, standardized APIs for application containerization and security features. However, unlike edge-cloud solutions, our architecture is designed to work in resource-constrained environments with limited computational, network bandwidth and power/energy resources. Thus, we focus on seamless operation in a network with heterogeneous protocols and diverse topologies, while addressing the challenges of intermittent/disconnected operation.

As shown in Figure 1, the ERGO operator can manage application/service deployments through the Kubernetes (K8s) [3] operations APIs and can schedule workloads dynamically. The service framework exposes application APIs to the in-field devices. In-field devices can interact with ERGO either directly or through an IoT API aggregation gateway. The service framework also provides the ability to compose multiple applications into a single end-user service.

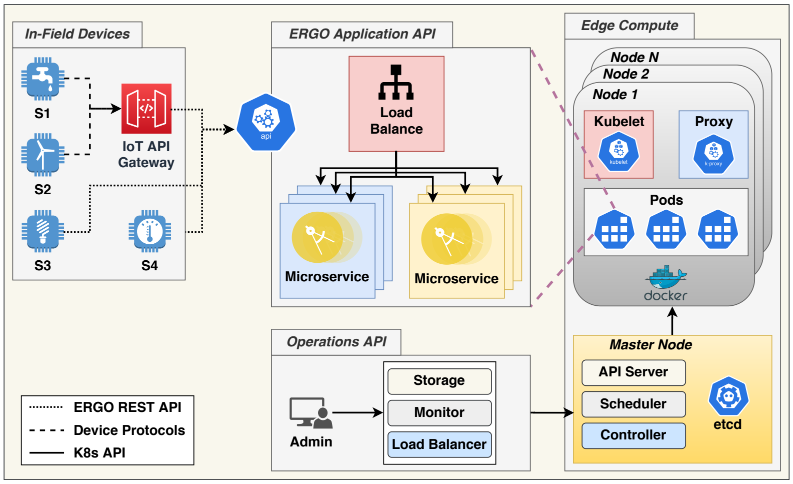
A. Disconnected/Intermittent Connectivity Operations

While the container orchestration platform allows for dynamically managing microservices, disconnected/intermittent connectivity creates new challenges that require special considerations to ensure seamless operation in the event of service failures. ERGO deploys a local image registry and scheduler service to ensure the availability of the application images and data to all nodes in the ERGO edge computing cluster. We also employ an image migration service to update both cluster management and application images periodically.

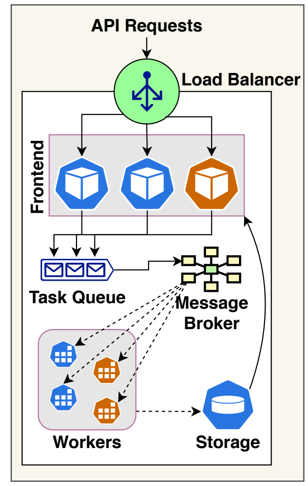
We implement our APIs on the Flask WSGI [4] framework. For interactive use, we also expose the API documentation using Swagger. Further, we employ response marshalling features to format, filter and render expected payload responses. To ensure modularity and to allow for namespace reuse and scalability, we use namespaces to organize the function-specific APIs. We use Flask Blueprints to manage API endpoint prefixes.

III. Results and Discussion

We evaluate the performance of ERGO service framework by load-testing ERGO applications and comparing it with traditional infrastructure comprising of a single monolithic server (labeled "*single-node*") with fixed applications. Our

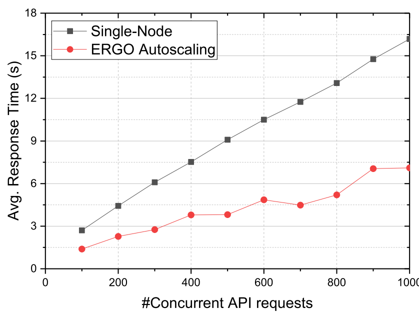


(a) ERGO Architecture.

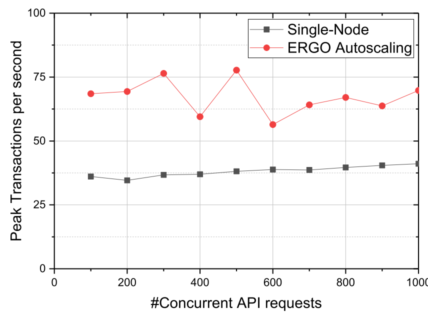


(b) ERGO Microservices.

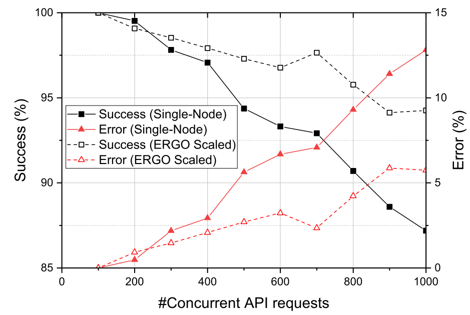
Fig. 1: Our ERGO Edge Computing Architecure.



(a) ERGO API response times.



(b) ERGO API throughput.



(c) Service success/failure rates.

Fig. 2: Performance evaluation of the Ag-IoT Application APIs.

evaluation framework consists of an IoT API gateway node interacting with the ERGO cluster wirelessly. We employ Apache JMeter [5] for functional load testing and evaluation. Our ERGO cluster allocates application/service deployments on 4 worker nodes, each with 4 cores and 4GB RAM. We evaluate ERGO performance using a single microservice deployment. We limit the hardware resources to 250 millicores and 512MB of RAM per service. We then evaluate the performance by sending a fixed number of concurrent API requests for 30 seconds. We repeat the test by increasing the number of concurrent API requests to range between 100 to 1,000/second. From Figure 2a, we observe that ERGO reduces the average response time (between $48\% - 62\%$, and about 54% on average) in comparison to the single-node infrastructure. Further, with autoscaling, we see an increase (between $57\% - 108\%$, and about 77% on average) in the peak transactions per second, leading to increased API throughput as shown in Figure 2b.

IV. Conclusions

In this paper, we present ERGO, an edge computing architecture for Ag-IoT environments characterized by limited/intermittent internet connectivity. We develop edge-enabled Ag-IoT services that are modular, composable, and highly scalable in heterogeneous, resource-constrained environments, which can augment highly instrumented Ag-IoT environments. Our exemplary applications and extensive performance evaluations demonstrate the efficacy of our proposed architecture In comparison to traditional architectures, on average, our proposed ERGO solution improves peak transaction throughput by over 77% and reduces response latencies by over 54%. Our future work will focus on developing a wider range of Ag-IoT microservices including machine learning workloads for the ERGO cluster.

Acknowledgments

This material is based upon work supported in part by the National Science Foundation under Grant No. CNS-1817105.

References

[1] O. Elijah, T. A. Rahman, I. Orikumhi, C. Y. Leow, and M. N. Hindia, "An overview of internet of things (iot) and data analytics in agriculture: Benefits and challenges," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3758–3773, Oct. 2018.

[2] C. Brewster, I. Roussaki, N. Kalatzis, K. Doolin, and K. Ellis, "IoT in Agriculture: Designing a Europe-Wide Large-Scale Pilot," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 26–33, September 2017.

[3] E. A. Brewer, "Kubernetes and the Path to Cloud Native," in *6th ACM Symposium on Cloud Computing*. ACM, 2015.

[4] "Flask (A Python Microframework)," https://flask.palletsprojects.com/, accessed: 2021-04-12.

[5] "Apache JMeter," https://jmeter.apache.org/, accessed: 2021-04-12.