International Journal for Modern Trends in Science and Technology, 6(8S): 185-191, 2020

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTSTCIET36
Available online at: http://www.ijmtst.com/ncracse2020.html

A Comprehensive Study on Design and Implementation of Bigdata-As-A-Service Stratum of Lifecycle Management of Smart Internet of Things Applications

G.Ramesh babu[1] | J.Sagar Babu[2] | R.Venkat

[1]Department of CS&E, Bapatla Engineering College, Bapatla, AP, India
[2]Department of CS&E, Nova College of Engineering College, Hyderabad, Telengana, India
[3]Department of CS&E, Vikas College of Engineering College, Vijayawada, AP, India

To Cite this Article
G.Ramesh babu, J.Sagar Babu and R.Venkat, "A Comprehensive Study on Design and Implementation of Bigdata-As-A-Service Stratum of Lifecycle Management of Smart Internet of Things Applications", *International Journal for Modern Trends in Science and Technology*, 6(8S): 185-191, 2020.

Article Info
Received on 16-July-2020, Revised on 15-August-2020, Accepted on 25-August-2020, Published on 30-August-2020.

ABSTRACT

Smart IoT systems require real-time and robust predictive analytics based on Machine Learning (ML) models. Building Big Data ML models is not only time consuming, but developers also lack the skills required for feature engineering, parameter tuning and model selection.The abundance of ML libraries and systems , data ingestion software, stream and batch processing engines, simulation technologies, and the number of hardware platforms accessible further exacerbates device architecture, fast development, and implementation issues. Finally, IoT's resource limitations demand that analytics engine execution be spread around the cloud edge spectrum. In order to address these challenging obstacles, we deliver Stratum, an event-driven Big Data-as-a-Service solution for IoT Lifecycle Analytics Management. Stratum provides developers with an elegant, declarative process, based on the model-driven architecture philosophy, to define device and network specifications. This paper illustrates the challenges Stratum is tackling, explaining its strengths through case studies in the modern world.

KEYWORDS: *AI/ML platform, Big Data Analytics, IoT,Cloud/Edge Resource Management, Deployment, Automation, Domain-Specific Modeling.*

INTRODUCTION

The Internet of Things (IoT)-based systems produce high-speed data volumes that need to be processed to extract useful information and make educated decisions across a number of application domains, e.g. , video analytics , predictive analytics, recommendation systems depend on the live and in-depth study of incoming data streams and historical data[1]. The creation, delivery and implementation lifecycle of IoT analytics tasks is considerably complicated in this context. Firstly, it includes creating one or more models of artificial intelligence (AI)/machine learning (ML) utilising vast collections of training data.This move involves developers to be informed of the variety of feasible ML models (e.g. linear models, decision trees or deep neural networks) and to be able to select from among the various ML libraries and frameworks. Second, once the ML models are qualified and ready to support the incoming demands, they have to be easily deployed and incorporated into the

target hardware network with the analytics pipeline.

Although the implementation of the ML model can be carried out in resource-rich environments, the resource-constrained complexity of the IoT structures and the real-time demands of the analytics tasks prohibit the flow of large volumes of data from the edge to the cloud for prediction. Alternatively, to partition and disperse the trained ML models across the cloud-edge continuum, successful resource management decisions are needed[2].

Unfortunately, IoT analytics programme developers still do not have the experience required to handle all the daunting lifecycle tasks of IoT analytics. There is therefore a compelling need to simplify the ML model creation phase and relieve the creator of liability for deciding the location of application analytics elements, tracking their usage of resources, and managing different data processing activities around the cloud spectrum[3].

Model-driven engineering (MDE)[4], specifically domain-specific modelling languages (DSMLs) and generative programming[5], is known to provide users with intuitive abstractions from error-prone and repetitive tasks while serverless computing relieves users of runtime infrastructure management issues. Within this article, we take advantage of the benefits of MDE and serverless computing and propose a Big Data-as-a-Service named Stratum for IoT Analytics Lifecycle Management. Our early research on Stratum[6] provided only the vision; this paper explores the architecture and tests Stratum 's capabilities in real-world use cases.

RELATED WORK

In this area, we utilize a propelling situation to feature the difficulties and determine the arrangement prerequisites for Stratum.

A. Spurring Case Study Eliciting Key Challenges

Consider an IoT use instance of a computerized tollgate that takes pictures of a vehicle's tag to charge cost [12]. This application will include a picture acknowledgment administration containing prescient analytics with the goal that it can consequently identify the tag of the entering or leaving vehicle from a cost court. The general framework will in this manner include a camera that snaps a photo of the tag, investigates the picture to distinguish the tag, and as needs be accuses the related record of the fitting cost.

Challenge 1: Model Development: Building prescient analytics requires creating AI/ML learning models dependent on chronicled datasets, which is a difficult errand and requires domain mastery. The designer is confronted with an assorted arrangement of ML capacities including order, relapse, suggestion (ALS), grouping, and so forth that are given by various libraries and systems, for example, Scikit-learn, Spark MLlib, and so on. Growing exceptionally precise models utilizing highlight building, model determination, hyperparameter tuning in those platforms is hard. In addition, to accelerate the training procedure, it should be driven by elite processing utilizing GPU and CPU, and should be circulated, if conceivable. Lamentably, all ML structures and calculations don't bolster conveyed model training. At last, the model advancement platforms must have the option to quickly choose the best models by assessing an enormous number of models in an appropriated way.

Challenge 2: Model Deployment: For IoT applications, for example, the tollgate use case or for automation partners like Google Home that utilization normal language induction models, the gadget to-cloud data full circle inactivity is impressive and henceforth handling at the edge is appealing as it diminishes dormancy and makes associated applications increasingly responsive. Along these lines, after the AI/ML model is trained, it should be quickly pushed to IoT gadgets for big data induction. The cloud resources include an assortment of servers including vitality effective multi-center processors and GPGPUs (e.g., NVidia TX1), or it very well may be a private cloud with restricted handling power. Hence, a viable choice of resources is vital.

Challenge 3: Data Movement and Management: Edge gadgets send the sifted data to the cloud utilizing one of numerous correspondence conventions, e.g., HTTP, MQTT and data ingestion administrations, for example, Apache Kafka, Apache Nifi, or Amazon Kinesis must be customized to tune in to approaching data streams and have the option to retain the data in databases or data lakes. When the data is ingested in the server, various endorsers might be keen on the result so they can run separate live analytics on the window of gushing data and picture the live examples [6]. Designers are probably not going to be specialists in these innovations and conventions to convey and autoscale the total data analytics pipeline.

Challenge 4: Determining the Right Hardware: Batch preparing systems, for example, Apache Hadoop or Spark can run profound analytics by conglomerating data from numerous data sources. These structures can coordinate the trained ML model for their indicative or prescient analytics as required. The difficulties lie in deciding the correct equipment to use from the wide assortment of gadget classes, picking deployment alternatives (on-preface or overcloud), and designing the deployment platform for the business application to convey genuine worth. In addition, all the exhibition measurements of the equipment parts should be checked for receptive and proactive dynamic to auto-scale the application.

B. Arrangement Requirements

In view of the evoked difficulties from Section III-A, we present the arrangement necessities that Stratum must meet.

Prerequisite 1: Flexible ML Service Development and En-

capsulation:To address Challenges 1 and 4, there is a need

for an AI/ML model structure system that can decouple the ML calculations from existing ML systems and create code for the basic structure from the clients' significant level ML model. The model ought to be epitomized, containerized, and uncovered by means of REST APIs.

Necessity 2: Automated Deployment of Application Components in Heterogeneous Environment: To address Challenges 2, and 4, we need a capacity that decreases designer exertion. The designer ought to indicate just negligible data utilizing a self-administration structure, and the framework at that point computerizes the deployment and data development challenges.

Necessity 3: Performance Monitoring and Intelligent Resource Allocation: To address Challenges 3, and 4, the arrangement must incorporate data and capacity administrations, plan the outstanding burden on container deployments that are coordinated by an execution motor with constant checking of framework measurements, and coordinate an exhibition data assortment technique. Utilizing these measurements, it must have the option to designate resources viably for the specific assignments powerfully and proactively.

PROPOSITION WORK:

PLAN AND IMPLEMENTATION OF STRATUM

This segment portrays the structure of Stratum and shows how it meets the prerequisites of Section III-B. Layer is acknowledged around the center ideas of Model-Driven Engineering (MDE) and generative programming. Layer gives a graphical interface to create and convey the ML pipeline utilizing more significant level instinctive deliberations. The theoretical language structure and semantics of the DSML are caught solidly in different metamodels. The client characterized specifications are utilized to mechanize the whole ML work process for the client. The Stratum DSML and its basic metamodels, constraint checkers, and code generative capacities are based on head of the WebGME modeling condition [13].1 Figure 1 shows the general use work process of the Stratum model-driven structure to understand a big data IoT analytics application.

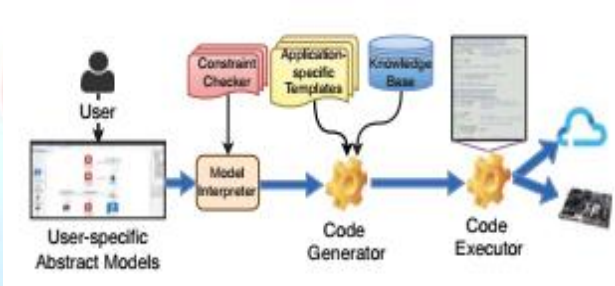

Fig. 1: Stratum Workflow to Realize an IoT Analytics Pipeline

A. Tending to Requirement 1: Development Kit for AI/ML Model Development

Layer involves an "Improvement Kit" for ML administration advancement and exemplification. It is an incorporated model-driven condition aid to fabricate ML pipelines by abstracting data preprocessing procedures, ML calculations that execute on existing ML structures, and assessment methodologies.

the metamodel [14]. The client models the deployment situation by utilizing the natural reflections utilizing the Stratum GUI. At that point, the Stratum model translator checks the rightness of the theoretical deployment model. A code generator inside the Stratum DSML produces the Infrastructure-as-Code (IAC) arrangement by parsing the client characterized model and sending it on the objective machine utilizing a layout based change and knowledge base. At long last, the IAC

arrangement is executed by the code agent to convey the ideal data analytics engineering on the objective machines over the cloud-edge range as appeared in Figure 1. The deployment is cultivated by means of the Stratum MDE approach without composing a solitary line of code.

1) Metamodel for Data Ingestion Frameworks: The meta-model for data ingestion devices (e.g., RabbitMQ, Kafka) catches the details of administrations for communicating with the Data Repositories and different microservices utilizing RESTful APIs. The client must choose the specific data ingestion device to send it on the objective platform. We likewise confirm the rightness of the client model dependent on the semantics characterized in the metamodel. For instance, in the event that the client chooses the AzureEventHubs as there wanted Data Ingestion Tool, at that point Amazon AWS or OpenStack or exposed metal can't be its objective deployment platform.

2) Metamodel for Data Analytics Applications: The live or bunch analytics structures, for example, Hadoop, Spark, Flink, and so on should be sent on the objective circulated frameworks. In addition, the trained ML model should be incorporated with these data analytics structures. So to convey the creation prepared AI pipeline, we caught the specifications for the ML libraries and structures, for example, Tensorflow, sci-unit learn alongside live and group analytics systems.

3) Additional Metamodels: Stratum likewise gives meta-models to catch the heterogeneity in resources (e.g., Raspberry Pi, NVIDIA Jetson TX1, uncovered metal server machines, and so forth), unique working frameworks and forms, just as data stores (e.g., AmazonS3, HDFS, AzureBlobStorage, Ceph).

B. Addressing Requirement 2: Framework for Performance Monitoring and Intelligent Resource Management

When the trained models are sent and begin executing, runtime resource management is required. Layer underpins autoscaling and load-adjusting utilizing the serverless worldview.

1) Performance Monitoring: It is basic to screen the framework measurements to comprehend the runtime execution of the foundation and the application. Layer use our ongoing work on FECBench [15] to gather distinctive framework measurements, for example, GPU-specific measurements, for example, power utilization, GPU use, temperature, and host-specific measurements, for example, CPU, plate, arrange, low-level reserve usage, memory data transmission use, utilizing gathered at various granularities from an appropriated set of resources and gather this data at a brought together server utilizing InfluxDB.

2) Resource Management: Stratum contains a Resource Manager to maintain the QoS of the application segments by scaling and moving the application segments. The inertness of prescient analytics applications is the summation of full circle inactivity (lrt) of data and the ML model execution time. We profile the ML model execution times on different data and target equipment before really sending the model, and consider its 95th percentile execution dormancy as assessed execution time (exechw_id,mlmodel).

Movement of ML Prediction Tasks. Before considering edge gadgets as a possible hub for executing prescient analytics, we check in the event that it has adequate memory to keep the model in memory. On the off chance that the edge hub can have the ML model, we profile the ML model on the edge gadgets. We likewise profile 95th percentile organize dormancy (lrt) among edge and cloud hub. We consider the movement of the ML model in the edge if exceeded,mlmodel < execcloud,mlmodel + lrt.

Auto-scaling of Application utilizing Serverless Paradigm: Let x signify the constraint determined by the Service Level Objective (SLO) of the ML model execution idleness, and let exechw_id,mlmodel,p mean 95th percentile execution dormancy on p CPU centers. For every server arrangement, we can register the quantity of solicitations n_req it can serve for an expectation administration while meeting the SLOs utilizing p CPU centers. By observing the all out number of approaching solicitations, we determined the all out number of machines (total_incomingrequests/nreq) required to deal with the profoundly equal outstanding burdens is. We determined what number of more machines to arrangement dependent on the distinction between the present status and wanted state. The Stratum Resource Manager conveys the ML model on these machines and starts the expectation administration consequently to deal with dynamic remaining burden proactively [16].

C. Conversation and Current Limitations

Presently, Stratum is equipped for producing just Python-based code, and just Scikit-learn and TensorFlow are incorporated. In any case, different

dialects, for example, Java, C++ can without much of a stretch be connected to Stratum, and other cloud libraries, for example, Amazon SageMaker, AzureML can be incorporated with Stratum without any problem. The structure of Stratum utilizes spry techniques so it tends to be stretched out effortlessly. We likewise fused the programmed rendition control in the Stratum system with the goal that we can review a specific adaptation of the structure later whenever required.

ASSESSMENT OF STRATUM

In this segment, we assess the straightforwardness, fast deployment, and resource management capacities of Stratum, alongside the openness, adaptability, and productivity of the ML model advancement system of Stratum.

A. ASSESSING THE RAPID MODEL DEVELOPMENT FRAMEWORK

We show how Stratum's MDE abilities ease ML devel-opement. Figure 2b shows how the ML designer can fabricate their ML pipeline utilizing the visual interface of Stratum. In the left-hand sheet (box1), all the structure squares are characterized utilizing the metamodel. The ML model engineer needs to relocate the necessary squares in the structure sheet (box 2) and must interface the squares to characterize the ML pipeline including pre-handling, ML calculation choice, hyperparameter tuning, model assessment, and best model choice measures. All the characteristics of the chose ML calculations, for example,max_depth,

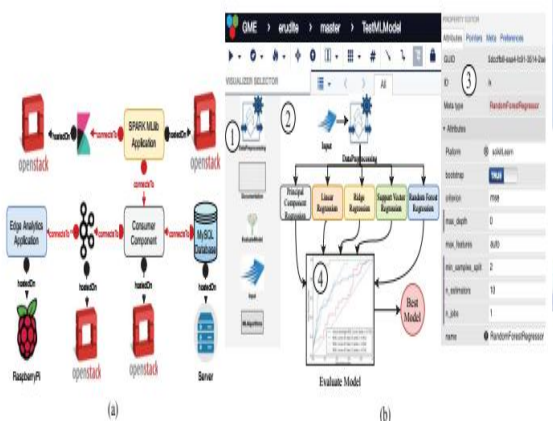

Fig. 2: (a) Example of Big Data Analytics System Deployment Model across Cloud/Edge Spectrum. (b) Usability of the ML model development Framework

Box 1shows the available selection of blocks available to create an ML pipeline as shown in Box 2. Attributes can be set using attribute selection panel in Box 3. Box 4 shows the model evaluation. standards, and so forth should be indicated by the client (or can take default esteems) from the correct sheet (box 3). The name of the qualities are reliant on ML calculations, and this viewpoint is caught by figuring out. The ML execution system should be referenced to tie the work process with a specific library or structure, for example, Scikit-learn, Spark MLlib, or Tensorflow.

All the ML calculations are exemplified in Docker containers, and various calculations can be executed in corresponding to accelerate the training and tuning stages. Additionally, in the information building hinder, the source data type, and way should be referenced, and furthermore data source type, e.g., CSV, the content is required. In the data preprocessing square, we just help straightforward data cleaning techniques, for example, sifting and standardization. In the assessment building obstruct, the strategy for assessment should be determined, and dependent on that, Stratum chooses the best model.

The model transformer can circulate various employments with various ML calculations over a group of associated machines and aids the designer to choose the best model or outfit of models dependent on the client's decision of assessment strategies. An example ROC bend appeared in box 4 portrays that the group of two ML techniques has the most elevated precision in the training stage on an example dataset, and it ought to be chosen as the best model. In this way, Stratum assists with building the ML model utilizing MDE procedures, and the ML engineer doesn't have to compose any code on bolstered ML structures. The system can likewise create the code in the note pad condition for the master client, where they can tune the ML model as required.

B.ASSESSMENT OF RAPID APPLICATION PROTOTYPING FRAMEWORK

As portrayed in Figure 2a, utilizing the visual interface of Stratum the application designer can build up the data analytics application. As portrayed in our past work [14], the business application work process is planned by hauling and associating the specific structure obstructs for application parts and foundation segments. As appeared in Figure 1, by parsing the client characterized dynamic model tree, the Stratum DSML makes the deployable model

(Ansible-specific for our situation) and utilizing NodeJS based module to execute the deployable model and make the framework of the application as depicted in Section. IV-B.

Figure 2a represents that utilizing the Stratum modeling condition, edge analytics application segments can be conveyed on Raspberry Pi machine, and data ingestion apparatus, e.g., Kafka can be sent on cloud VMs, which is maintained by OpenStack. The data purchaser application part can be likewise conveyed on OpenStack VM, which will expend the data from Kafka in a group or stream and store it in MySQL database, which is sent on head of the uncovered metal server. At that point, Spark with the MLlib library can be conveyed and arranged on OpenStack VMs, and a perception motor like Kibana can be coordinated with the work process. Peaceful APIs interface all the application parts, so the ML model can without much of a stretch be driven into prescient analytics applications during the management stage. We built up an example big data use case situation utilizing Stratum for assessment investigation of Twitter data stream utilizing the Figure 2a deployment model.

C. EXECUTION MONITORING ON HETEROGENEOUS HARDWARE

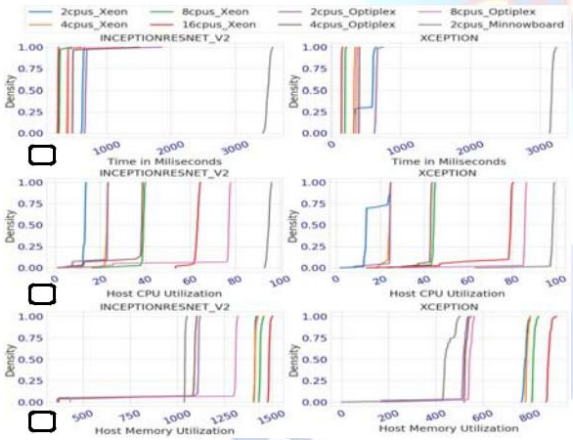

Fig. 3: (a) Latency of InceptionResnetV2 and Xception prediction services, (b)Host CPU utilization, (c) Host Memory utilization

D. RESOURCE MANAGEMENT

As referenced in Section. IV-C2, we profile the forecast administration on the specific equipment before sending it on the group of machines. In light of the quantity of approaching solicitations (dynamic outstanding burden), we scale here and there the quantity of ML model containers to ensure the pre-characterized QoS in an occasion driven way utilizing the Docker swarm group management instrument, as showed in our ongoing work [16].

CONCLUSION

As IoT-based analytics turns out to be progressively advanced, engineers are ending up lacking mastery in a wide scope of abilities while likewise being overpowered by the plenty of structures, libraries, programming dialects, and equipment available to plan and convey analytics applications. To address these profoundly pragmatic difficulties, this paper presents Stratum, which is a novel Big Data-as-a-Service for the lifecycle management of IoT analytics applications. Layer gives a graphical device that permits a client to graphically make the ML advancement and deployment pipeline utilizing its upheld highlights, and Its DSML understands the model and produce the code to mechanize and coordinate the application lifecycle management over the cloud-edge range.

REFERENCES

[1] E. Blasch, S. Ravela, and A. Aved, Handbook of dynamic data-driven applications systems. Springer, 2018.

[2] M. Satyanarayanan, “The emergence of edge computing,” Computer, vol. 50, no. 1, pp. 30–39, 2017.

[3] S. Shekhar, A. D. Chhokra, A. Bhattacharjee, G. Aupy, and A. Gokhale, “Indices: exploiting edge resources for performance-aware cloud-hosted services,” in 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC). IEEE, 2017, pp. 75–80.

[4] D. C. Schmidt, “Model-Driven Engineering,” IEEE Computer, vol. 39, no. 2, pp. 25–31, 2006.

[5] K. Czarnecki and U. W. Eisenecker, Generative Programming: Methods,Tools, and Applications. Reading, MA: Addison-Wesley, 2000.

[6] A. Bhattacharjee, Y. Barve, S. Khare, S. Bao, A. Gokhale, andDamiano, “Stratum: A serverless framework for the lifecycle management of machine learning-based data analytics tasks,” in 2019 USENIX Conference on Operational Machine Learning (OpML 19). Santa Clara, CA: USENIX Association, May 2019, pp. 59–61. [Online]. Available: https://www.usenix.org/conference/opml19/presentation/Bhattacharjee

[7] T. Li, J. Zhong, J. Liu, W. Wu, and C. Zhang, “Ease. ml: Towards multi-tenant resource sharing for machine learning workloads,” Proceedings of the VLDB Endowment, vol. 11, no. 5, pp. 607–620, 2018.

[8] D. Baylor, E. Breck, H.-T. Cheng, N. Fiedel, C. Y. Foo, Z. Haque,Haykal, M. Ispir, V. Jain, L. Koc, et al., "Tfx: A TensorFlow-based production-scale machine learning platform," in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2017, pp. 1387–1395.

[9] (2017) Meet michelangelo: Uber’s machine learning platform. [Online].Available: https://eng.uber.com/michelangelo/

[10] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Kon-winski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe,

et al., "Accelerating the machine learning lifecycle with mlflow," Data Engineering, p. 39, 2018.
[11] K. K. Kumar, S. G. B. Kumar, S. G. R. Rao and S. S. J. Sydulu, "Safe and high secured ranked keyword searchover an outsourced cloud data," 2017 International Conference on Inventive Computing and Informatics (ICICI), Coimbatore, 2017, pp. 20-25, doi: 10.1109/ICICI.2017.8365348.
[12] Y. Lee, A. Scolari, B.-G. Chun, M. Weimer, and M. Interlandi, "From the edge to the cloud: Model serving in ml .net," Data Engineering, p. 46, 2018.
[13] K. Kumar, S. Sinha, and P. Manupriya, "D-PNR: Deep license plate number recognition," in Proceedings of 2nd International Conference on Computer Vision & Image Processing. Springer, 2018, pp. 37–46.
[14] M. Maróti, R. Kereskényi, T. Kecskés, P. Völgyesi, and A. Lédeczi, "Online collaborative environment for designing complex computational systems," Procedia Computer Science, vol. 29, pp. 2432–2441, 2014.
[15] A. Bhattacharjee, Y. Barve, A. Gokhale, and T. Kuroda, "A model-driven approach to automate the deployment and management of cloud services," in 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion). IEEE, 2018, pp. 109–114.
[16] Y. D. Barve, S. Shekhar, A. Chhokra, S. Khare, A. Bhattacharjee,Kang, H. Sun, and A. Gokhale, "Fecbench: A holistic interference-aware approach for application performance modeling," in 2019 IEEE International Conference on Cloud Engineering (IC2E), June 2019, pp. 211–221.
[17] A. Bhattacharjee, A. D. Chhokra, Z. Kang, H. Sun, A. Gokhale, andKarsai, "Barista: Efficient and scalable serverless serving system for deep learning prediction services," in 2019 IEEE International Conference on Cloud Engineering (IC2E), June 2019, pp. 23–33.