Quantifying Performance and Energy Efficiency of a Real Time Data Analytics System

Abdul Rehman
abrehman@iu.edu
Indiana University Bloomington
Bloomington, Indiana, USA

Prateek Sharma
prateeks@iu.edu
Indiana University
Bloomington, Indiana, USA

ABSTRACT

We have quantified the performance and energy footprint of a real time data analytics system (Edge-IoT-Analytics-Box). Edge-IoT-Analytics-Box is a system that can host multiple Maching Learning models and use them to setup real time data analytics pipelines. Our analysis indicate that this system deployed on a Jetson Orin AGX device results in 0.1 second (bestcase) and 0.3 seconds (worstcase) of inference time for GPU centric models with only 0.8 Watts of average power consumption increase. For CPU centric models, the inference time is 0.1 seconds (best case) and 1.5 seconds (worst case) with a power consumption increase of 3.2 Watts.

Our experiments indicate that the tail latency saturates at 1.2 seconds for GPU centric models and 2.5 seconds for CPU centric models. Point of inflexion is reached rapidly (25 requests per second) for CPU centric models as opposed to GPU centric models (40 request per second).

KEYWORDS

Data Analytics, High Performance Computing, Energy Efficiency, Edge Computing, Machine Learning

1 INTRODUCTION

With the advent of IoT, the amount of data being generated is increasing exponentially. Trends in data provides valuable insights that can be used to make informed decisions and avoid potential chaos. For instance, real time anomaly detection of devices on a railway network can predict potential failures and avoid accidents. Similarly, real time analysis of traffic data to forecast potential congestion and suggest alternate routes can save time and fuel.

However, processing this data in real time is a challenging task. Some examples of data analytics system include HASTE [5], GeoFlink [3], research works [4] etc.

Edge-IoT-Analytics-Box [1] is one such distributed system that allows users to deploy multiple machine learning models on edge devices and then use them to build analytics pipelines for real time data processing. Pipeline can be targeted at any analytics job like forecasting, anomaly detection, classification etc.

Figure 1 shows the design of the Edge-IoT-Analytics-Box system. There are two distinct layers of the system. Data Storage Layer and Continuous Data Analytics Layer. Primary purpose of the Data Storage Layer is to reliably fetch data from sensors and store it into a time series database. Continuous Data Analytics Layer, provides a flexible closed loop system for fetching data, analyzing it and pushing the analysis back to the database.

, ,
.

Data Storage layer is built using MQTT protocol and Influx database. Data is sourced from MQTT protocol compatible devices. Streaming service provides a service to connect to a MQTT broker and InfluxDB, which continuously fetches the data and pushes it to the database in real time.

Continuous Data Analytics layer is built using in-house Analytics Framework that we have developed and Torch Serve to host DNN models. The continuous data analytics layer is hosted on Jetson Orin - a low cost, low power device. Each analysis/prediction loop in this layer starts with fetching data from the database, followed by making an inference request to the model - hosted on torch serve - and pushing the data back to the database. InfluxDB dashboard allows to visualize the real time data and predictions.

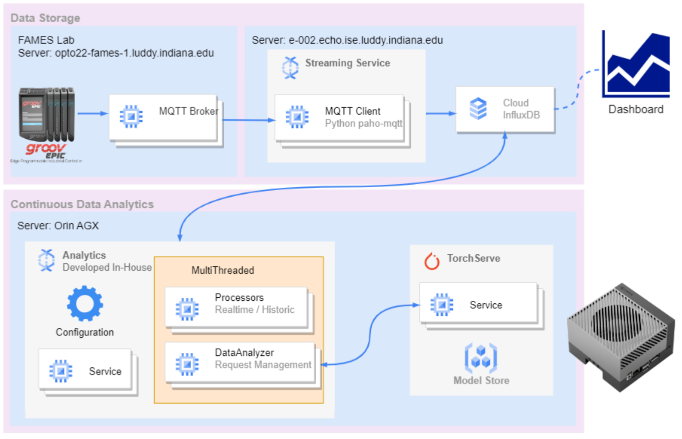


Figure 1: Architecture of Edge-IoT-Analytics-Box System

Given the complexity of this system and the number of components involved, it is important to quantify the best case and worst case performance before even attempting to optimize it or giving Quality of Service guarantees. In this paper, we quantify the performance of Edge-IoT-Analytics-Box system deployed on a Jetson Orin AGX (32GB) device. Inference time for each request is collected as the number of pipelines are increased. Distribution of inference time is analyzed to derive insights about the possible limitations of the system.

The rest of the paper is organized as follows

- Section 2 describes the metrics of interest
- Section 3 describes the methodology used to quantify the performance and energy cost of the system
- Section 4 presents the result of the analysis
- Section 5 discusses potential areas of improvement
- Section 6 concludes the paper

2 METRICS OF INTEREST

Inference Time: Time it takes for a http request to complete end to end.

Power Consumption: Average power consumption of the system in Watts while requests are being serviced for a given duration.

Pipeline: A pipeline is a sequence of data processing components that are connected in series to fetch data from a database, process it using a machine learning model and then store the results back to the database.

Number of Pipelines: Number of realtime analytics being done in parallel.

3 METHODOLOGY

3.1 Setup

Edge-IoT-Analytics-Box [1] system is deployed on the Jetson Orin AGX device which has 2048-core NVIDIA Ampere architecture GPU with 64 Tensor Cores and 12-core Arm® Cortex®-A78AE CPU. Two models are being used for prediction - ARNN model and ARIMA model. ARNN is a GPU centric model and ARIMA is a CPU centric model. A pipeline consists of reading sensor data from InfluxDB, forecasting it's value for next 10 seconds and storing the forecasted result back to InfluxDB. Sensor data is a time series data that is being streamed to the database every second or minute depending on the type of the sensor. Figure 2 shows an example of the sensor data.

Inference time is collected for each individual request at the initiators end. During the experiment, power consumption of GPU, CPU and the system as a whole is also collected using tegrastat. Performance analysis specific code is available at **performance_analysis** directory in the codebase [2].

3.2 Experiments

Single Experiment: Pipelines are setup for given number of sensors. Setup is allowed to warm-up for 10 seconds before metric collection is started. Metrics are collected for 30 seconds, before the setup is stopped. **performance_analysis/run_single.sh** script in the codebase is used to run the single experiment.

Sweeping Number of Pipelines: For each given combination of models, a sweep is done for number of pipelines ranging from 10 sensors to 50 sensors with an increment of 5 sensors. As an example to sweep the number of pipelines for ARNN model, it would require 9 single experiments. **performance_analysis/run_series.sh** script in the codebase is used to run all the experiments one after another.

Combination of models:

- ARIMA only
- ARNN only
- ARIMA and ARNN

In total 27 experiments were executed to quantify the performance of the system. **performance_analysis/postprocessing** has the scripts for post processing the results and drawing figures. Collected data is available at under **performance_analysis/series_results** in directories **arima_only**, **arnn_only** and **both_models**.

4 RESULTS

Distribution of inference time for each request is shown in Figure 4 and the mean power of the system for each experiment is shown in Figure 3. Top edge of the violin plot indicates the tale latency for each instance and the fat portion of the violin plot indicates the mean inference time. Power plots show CPU and GPU portion of the power in addition to the total system power consumption.

Key findings from the results are as follows:

- System is stable for GPU centric models as opposed to CPU centric models in terms of tail latency.
- GPU is power efficient as opposed to CPU.
- Point of inflexion is reached rapidly for CPU centric models as opposed to GPU centric models.

Tail Latency: As the number of pipelines are increased from 10 to 50, the tail latency saturates at 2.3 seconds for ARIMA at 40 parallel requests per second. The distribution for inference time is evenly distributed with a mean at around mid point of 1.2 seconds - see Figure refsubfig:carima. In contrast, for ARNN saturated tail latency is 1.2 seconds with a very low mean inference time of 0.1 seconds - see Figure refsubfig:carnn. It is a GPU centric model that is why the inference time is low and more stable as requests increase. Under the hood, cuda is time multiplexing the requests.

Power Efficiency: Average power consumption of the system is 23 Watts for ARIMA and 23 Watts for ARNN when requests are being made as opposed to 19 Watts when the system is idle. This 4 Watts increase in power consumption is due to the CPU and GPU being used for inference. Off this 4 Watts, 0.8 Watts of increase is due to GPU and 3.2 Watts of increase is due to CPU. Figure 3a shows that as CPU centric requests increase the power consumption of the system rises rapidly. In contrast, Figure 3b shows that the power consumption of the system is stable as the number of pipelines are increased.

Inflexion Point: Point of inflexion is reached rapidly for CPU centric models as opposed to GPU centric models. For ARIMA, the point of inflexion is reached at 25 parallel requests per second. In contrast, for ARNN, the point of inflexion is reached at 40 parallel requests per second.

Concurrent requests for ARNN and ARIMA: When both of the models are being used for prediction, tail latency saturates at 2.3 seconds and mean inference time has clearly two means one at 1.5 seconds for the ARIMA and other at 0.3 seconds for the ARNN model.

5 POTENTIAL AREAS OF IMPROVEMENT

Since the inflexion point is reached rapidly for CPU centric models as opposed to GPU centric models, we can throttle the number of requests for CPU centric models to 25 requests per second and for GPU centric models to 40 requests per second. This will ensure that the tail latency does not increase significantly and the energy cost per inference is low.

Additionally, given the fact that power consumption for GPU is low as opposed to CPU, we can build GPU centric models for prediction in place of CPU centric models. This will ensure that the system is power efficient and the tail latency is stable as the number of requests increase.

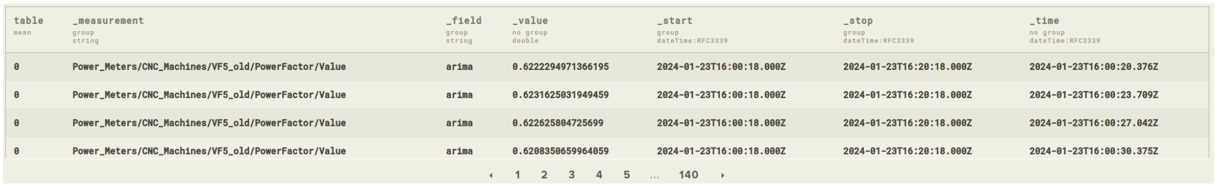
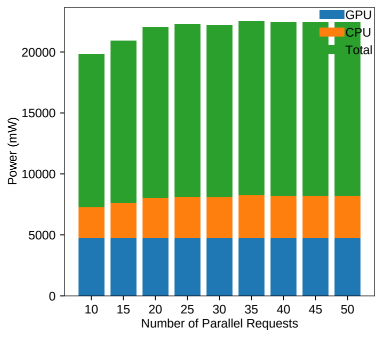


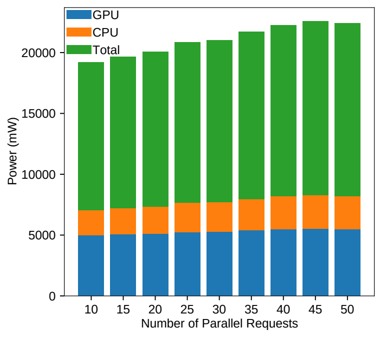
table mean	_measurement group string	_field group string	_value no group double	_start group dateTime:RFC3339	_stop group dateTime:RFC3339	_time no group dateTime:RFC3339
0	Power_Meters/CNC_Machines/VF5_old/PowerFactor/Value	arima	0.6222294971366195	2024-01-23T16:00:18.000Z	2024-01-23T16:20:18.000Z	2024-01-23T16:00:20.376Z
0	Power_Meters/CNC_Machines/VF5_old/PowerFactor/Value	arima	0.6231625031949459	2024-01-23T16:00:18.000Z	2024-01-23T16:20:18.000Z	2024-01-23T16:00:23.709Z
0	Power_Meters/CNC_Machines/VF5_old/PowerFactor/Value	arima	0.622625804725699	2024-01-23T16:00:18.000Z	2024-01-23T16:20:18.000Z	2024-01-23T16:00:27.042Z
0	Power_Meters/CNC_Machines/VF5_old/PowerFactor/Value	arima	0.6208350659964059	2024-01-23T16:00:18.000Z	2024-01-23T16:20:18.000Z	2024-01-23T16:00:30.375Z

‹ 1 2 3 4 5 … 140 ›

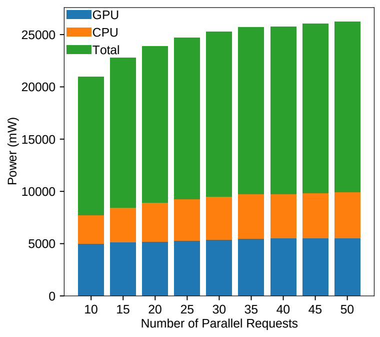
Figure 2: Sensor Data



(a) ARIMA

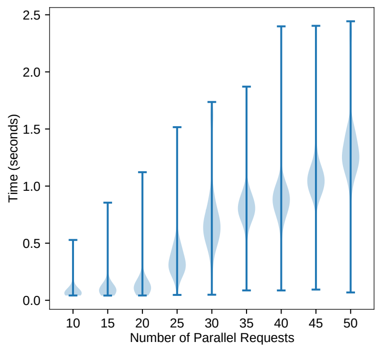


(b) ARNN

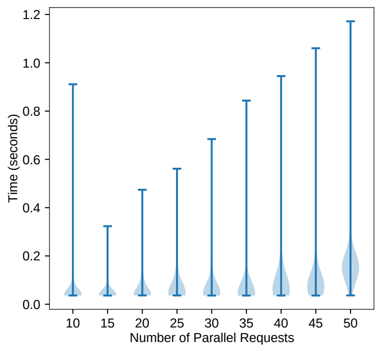


(c) BOTH

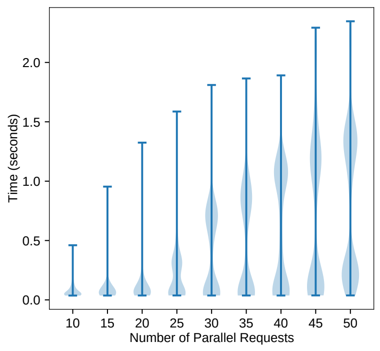
Figure 3: Power Consumption



(a) ARIMA



(b) ARNN



(c) BOTH

Figure 4: Inference Times

6 CONCLUSION

We have quantified the performance and energy efficiency of a real time data analytics system. Our analysis indicate that this system deployed on a Jetson Orin AGX device is better suited for GPU centric models with stable Quality of Service in response to high request throughput. CPU centric models perform poorly as the number of requests increase. Additionally, we have determined the point of inflexion for the system which can be used for future work to throttle the number of requests to guarantee quality of service.

REFERENCES

[1] [n. d.] Edge-iot-analytics-box. https://github.com/COS-IN/Edge-IoT-Analytics-Box/tree/performance_analysis. Accessed: 2024-04-22. ().

[2] [n. d.] Edge-iot-analytics-box - performance analysis code/results. https://github.com/COS-IN/Edge-IoT-Analytics-Box/tree/performance_analysis/performance_analysis. Accessed: 2024-04-22. ().

[3] Salman Ahmed Shaikh, Komal Mariam, Hiroyuki Kitagawa, and Kyoung-Sook Kim. 2020. Geoflink: a distributed and scalable framework for the real-time processing of spatial streams. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (CIKM '20). Association for Computing Machinery, Virtual Event, Ireland, 3149–3156. ISBN: 9781450368599. DOI: 10.1145/3340531.3412761.

[4] Mbarka Soualhia, Chunyan Fu, and Foutse Khomh. 2019. Infrastructure fault detection and prediction in edge cloud environments. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing* (SEC '19). Association for Computing Machinery, Arlington, Virginia, 222–235. ISBN: 9781450367332. DOI: 10.1145/3318216.3363305.

[5] Zhong Yang, Bolong Zheng, Chengdong Tong, Lianggui Weng, Chenliang Li, and Guohui Li. 2021. Haste: a distributed system for hybrid and adaptive processing on streaming spatial-textual data. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (CIKM '21). Association

for Computing Machinery, Virtual Event, Queensland, Australia, 2363–2372. ISBN: 9781450384469. DOI: 10.1145/3459637.3482435.