Integrated Application and Performance Monitoring at the IoT Edge

Yutthana Boonpalit[a], Siwakorn Suwanjinda[a], Jason Haga[b], Shava Smallen[c], Prapaporn Rattanatamrong[a], Vahid Daneshmand[d], Kensworth Subratie[d]
Thammasat University[a], National Institute of Advanced Industrial Science and Technology (AIST) [b],
University of California, San Diego[c], University of Florida[d]
oatyootna@gmail.com[a], siwakorn4437@gmail.com[a], jh.haga@aist.go.jp[b], ssmallen@sdsc.edu[c], rattanat@gmail.com[a], vdaneshmand@acis.ufl.edu[d], kcratie@ufl.edu[d]

Background

- Devices and protocols used within an IoT system may vary greatly in terms of power and connectivity requirements.
- The IoT system requires an IoT gateway to aggregate sensor data, translates between different protocols and process data before sending it to a cloud server.
- A smart home might collect sensor data and process it at a single gateway; on the other hand, a smart city management might have a gateway for each neighborhood in the entire city.
- Therefore, traditional ways of monitoring and troubleshooting used by system admins are no longer sufficient to give them the whole view of system status.

Challenges

- The system must be flexible enough to work with various kinds of commonly used IoT sensors and devices.
- The system must be scalable and power-efficient.

Aims

- Build an integrated solution for application.
- The system must be scalable and power-efficient.

Methodology

- Telegraf was deployed in a set of Raspberry Pi boards (Gateway Node) and InfluxDB and Grafana were deployed in a PRAGMA cloud server.

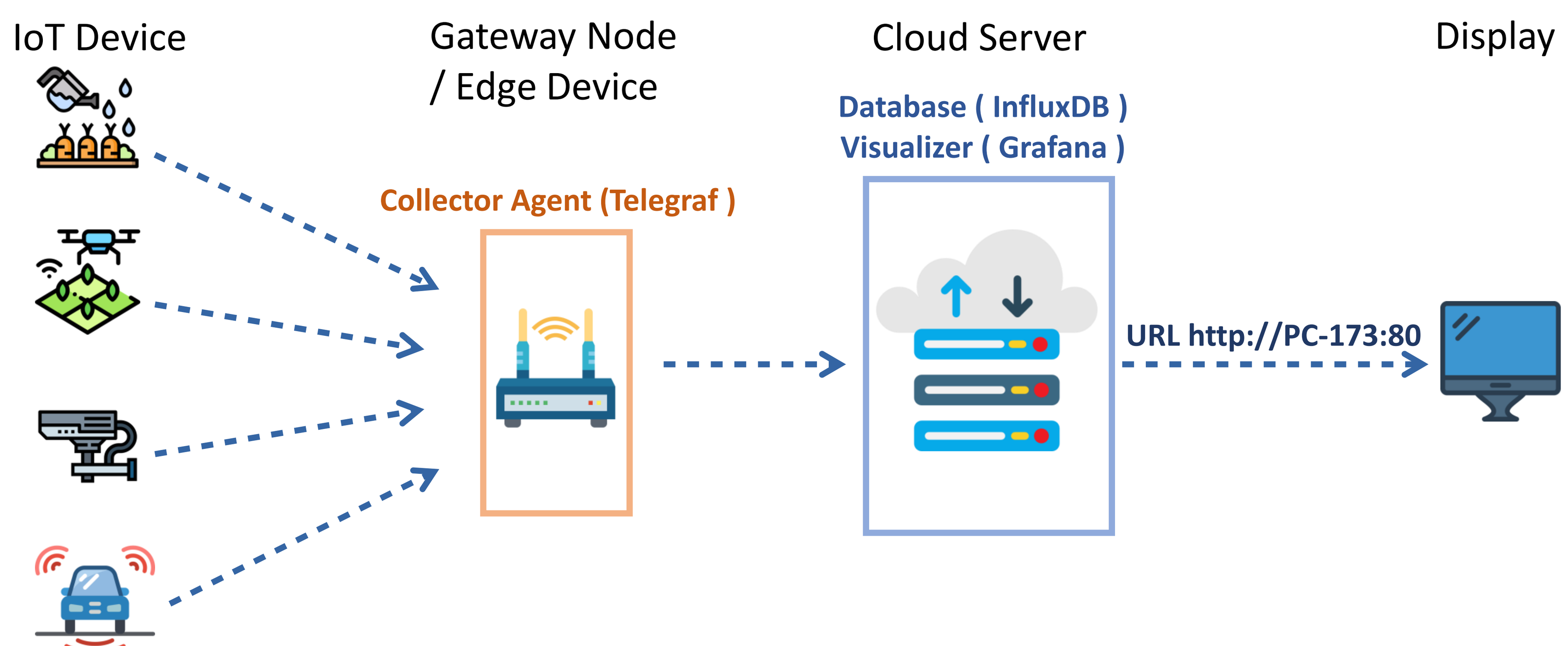

Figure 1: System architecture

- For application data, sensor data was simulated on the IoT gateway in two ways: (1) logging data from a temperature sensor, and (2) replaying logging data from the Smart Reservoir project's gateway from its GitHub repository.

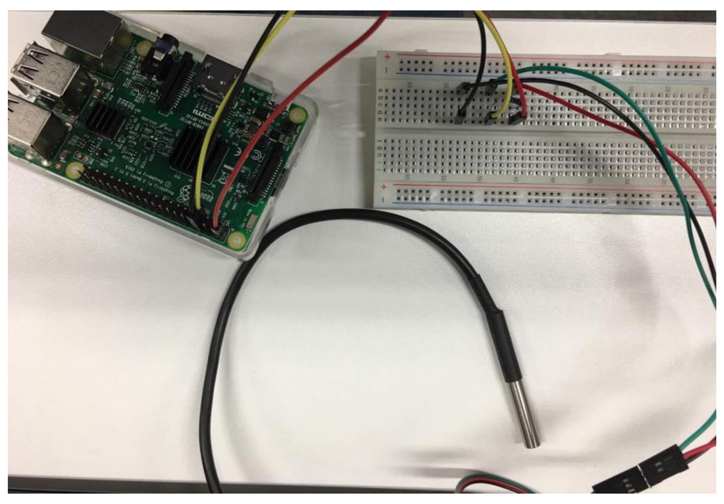

Figure 2: IoT gateway with temperature sensor

Figure 3: Grafana dashboard

- Experiments were performed to measure the overhead and power consumption of Telegraf in our setup compared to our set up with no Telegraf installed. We varied the frequencies of data collection and the number of monitored metrics.

Experiment Results

Impact of Metrics Collection

- **Memory usage is about 18% when Telegraf is running.**
- **When the number of metrics is increased from 1 to 10, Telegraf's use of CPU, memory, and disk I/O is negligible**
- **Telegraf's power consumption is negligible.**

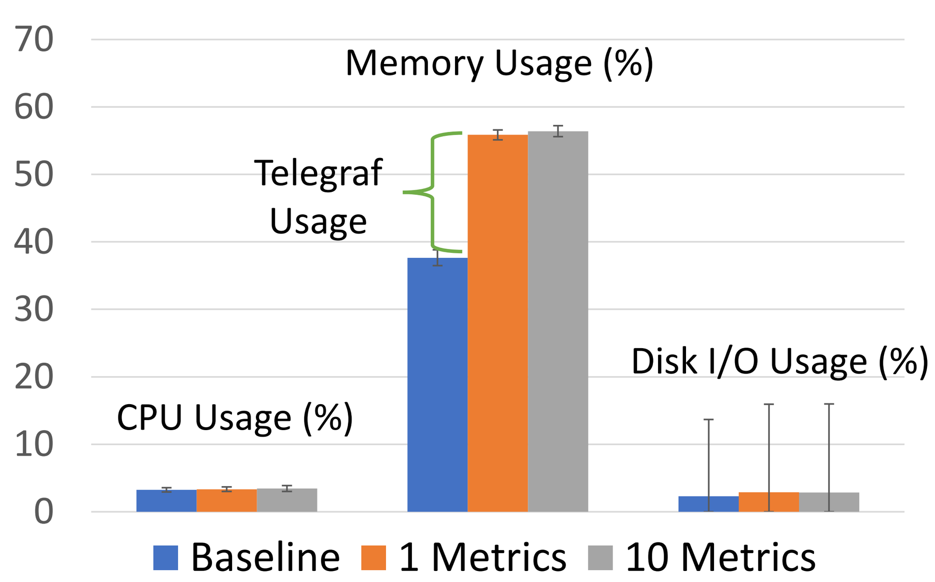

Figure 4: Telegraf system usage

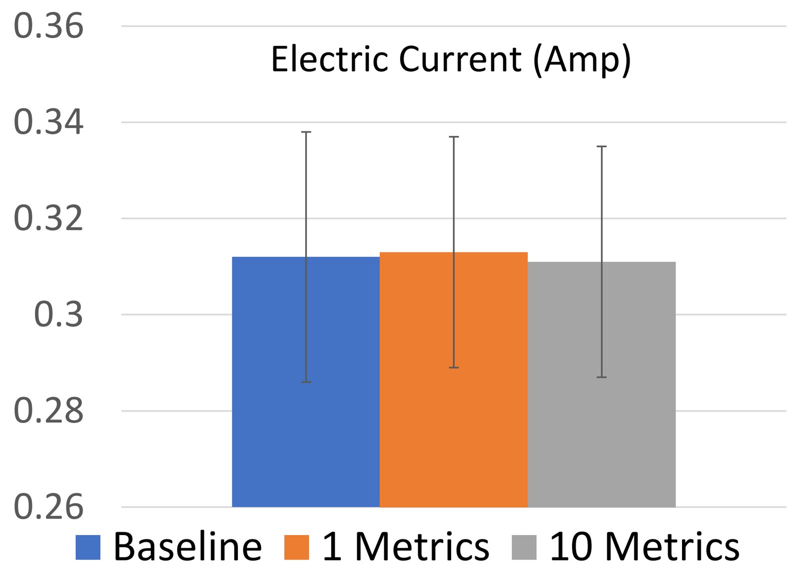

Figure 5: Telegraf power consumption

Impact of Collection Frequencies

- **Data collection rates greater than 1 Hz caused Telegraf to use about 18% more memory, CPU, and power consumption.**
- **Telegraf's use of disk I/O is negligible.**

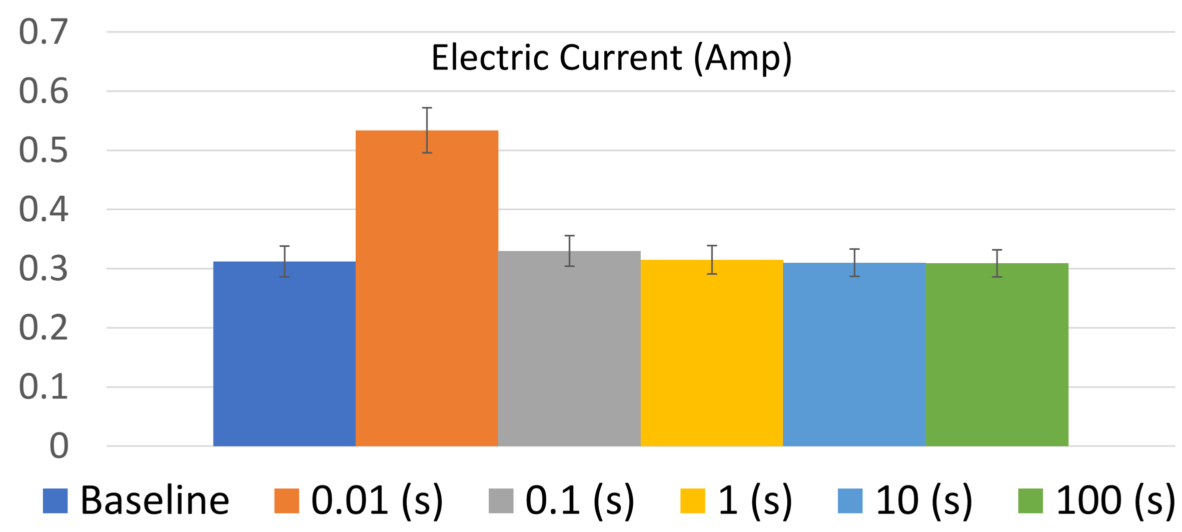

Figure 6: Telegraf power consumption

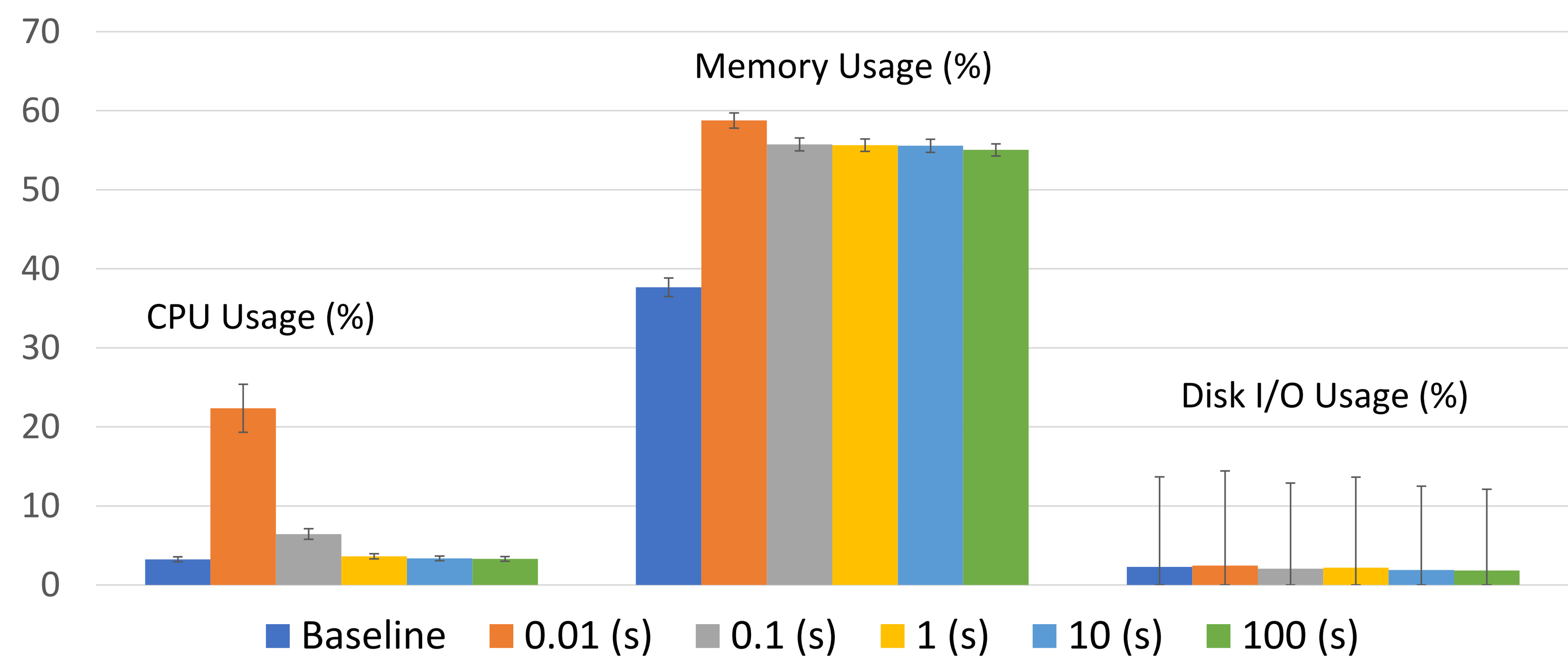

Figure 7: Telegraf system usage

Conclusion

- Telegraf's scalability and low power consumption make it ideal for deployment on Raspberry Pi.
- InfluxDB is suitable for storing time-series data and scales well.
- Grafana also scales well for data visualization.
- Data collection rates above 1Hz can induce some system overhead, that may require Raspberry Pi deployments to use AC power.

Future Work

- Investigate other approaches to collect application data in Telegraf.
- Investigate the impact of other approaches on system performance.

AIST
NATIONAL INSTITUTE OF ADVANCED INDUSTRIAL SCIENCE AND TECHNOLOGY (AIST)

UF UNIVERSITY of FLORIDA