Poster Abstract: Scaling IoT Device APIs and Analytics

Omprakash Gnawali
University of Houston
gnawali@cs.uh.edu

David Moss, Dmitry Shirkalin
People Power Co.
{dmoss,dmitry}@peoplepowerco.com

Russ Clark, Brian Jones, William Eason
Georgia Institute of Technology
{russ.clark,brian.jones,bill.eason}@gatech.edu

***Abstract*—Many IoT applications consist of two types of actions: interaction with the device, which can be sensors or actuators, and interaction with the data, for example, to reveal insights. In this poster, we introduce a software stack that provides these functionalities in a scalable manner. The API for device interaction is designed with generality in mind so that widest possible array of devices are supported and in large numbers. The analytics framework, called Composer, is designed to allow user code to be easily integrated into data analytics. We present the design, describe the implementation and deployment, and present some evaluation results. We share the performance data from a live deployment with tens of thousands of active users to demonstrate the scalability of the design.**

I. Introduction

The two common functionalities required in an IoT stack are device interaction and data analytics. Many IoT applications have physical devices such as sensors and actuators. Traditional examples are sensing applications. Lately, these applications increasingly have components that interact with the users or the environment.

An IoT technology provider must deal with a large number of such devices. Depending on how the system is designed, these devices perform a large number of small requests to the server. In some systems, these devices keep a large number of persistent connections. Thus, IoT software stack must overcome these scaling challenges.

The other aspect that is becoming increasingly common is the ability to analyze the data, in nearly real time, and perform action in response. Several examples exist in the industry and research [1], [2]. The data analytics needs can be diverse depending on the application and often extends far beyond simple graphs or triggers. It is increasingly common for data analytics to include sophisticated statistics, data modeling, or machine learning functionalities. Thus, an IoT technology provider must provide a way to plug in data analytics modules into the core of data processing pipeline. Only with such flexiblity will it be possible to support diverse IoT applications with tens of thousands of active users.

In this poster, we present an architecture that scales to meet these demands. We briefly describe the design rationale, use cases in which the architecture is being used, and share some performance results.

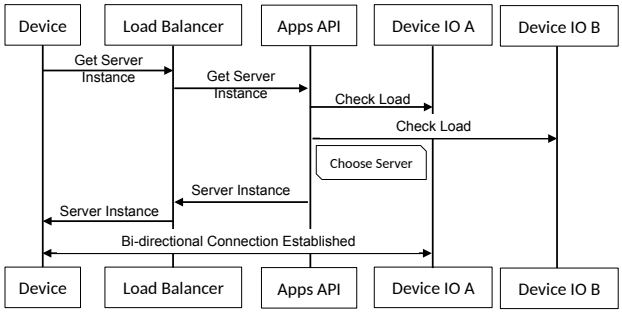

Fig. 1. Scalable Device IO Server Architecture. Device IO servers handle the API calls made by the devices such as sensors.

II. Design and Implementation

Although the software stack on the device and the Cloud has many components, we focus our discussion on Device IO and Analytics.

A. Scaling Device IO

Figure 1 shows the architecture of the infrastructure designed to process a large number of device API calls. In addition to periodically uploading sensor data to the cloud server, device API calls typically include a long-polling persistent bi-directional connection with the server to facilitate the rapid delivery of commands from the server to the sensor device. For each long-polling persistent connection, one port is continuously consumed on the side of the cloud server. Under these conditions, standard load balancers become ineffective. Therefore, under this design, devices that are directly connected with the cloud server must load balance themselves across a pool of available servers. This is accomplished by periodically (currently every 24 hours) using a load-balanced Application API to retrieve a server instance to which the device should connect. When the Application API server receives this request, the server evaluates a pool of candidate device servers (e.g., A or B) and returns the address for the best candidate. This architecture has been used to handle 20,000-40,000 open connections in a server and may scale further.

B. Scaling the Analytics

Scaling the Analytics is even harder than scaling the device APIs. Figure 2 shows the architecture of Composer, the data analytics portion of the infrastructure. The key insight is

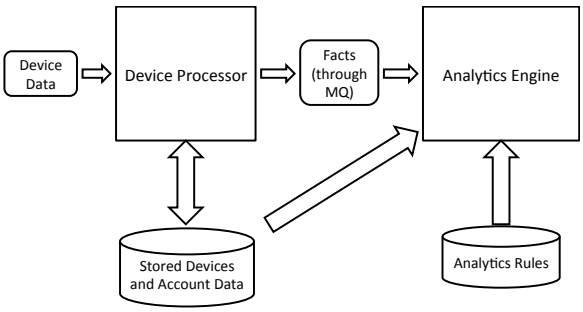

Fig. 2. Data flow architecture to support Analytics (Composer).

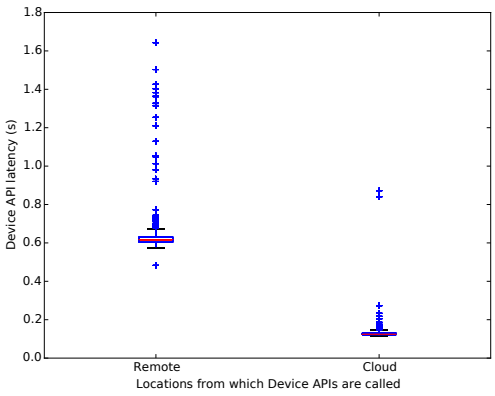

Fig. 3. Device IO API Latency.

carefully designing the dataflow within the system so that the flows do not block each other and have minimal dependency among the components despite the analytics requiring not only data sources of different types (e.g., fresh data from the devices, synthesized facts, or stored data) but also the ability to execute rules and analytics applications. In Composer, carefully sanitized Python scripts perform the analytics tasks and can trigger actions.

III. Evaluation

During the week of February 8th 2016, the system received about 30 million data posts from devices (device IO API), or in average 48 API device posts per second. Average response time was 500 ms. In parallel the server processed 27.5 million analytics requests, which is 46 analytic events per second. All of this was done on a server with 8 CPU's, which are shared between device processing and apps API calls. Device processing took 4-6% of CPU and database server another 15-25% of CPU. Figure 4 shows the performance of the server over one week duration.

To drill down the latency performance further, we generated two sets of device post APIs from a machine. The first is from a machine at home and the second from the Cloud to provide the lower bound or baseline performance. Figure 3 shows the results suggesting that the device API with low latency is feasible even with modest server resources.

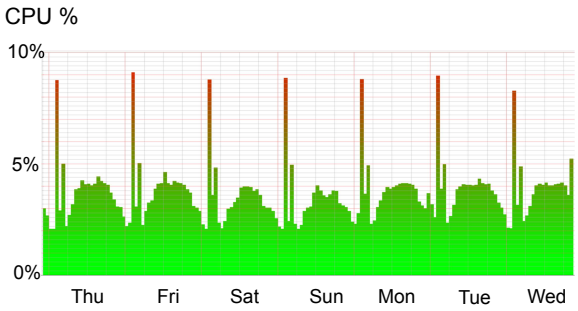

Fig. 4. CPU load due to data analytics.

IV. Deployments

The sytem has been deployed to support IoT applications with tens of thousands of daily active users. We are currently also deploying the system to support several smart home and educational settings.

In one application, the device APIs are offered to students and researchers who require cloud-enabled software to connect devices and sensors. The Composer analytics software will allow researchers to create analytics that run in real-time on top of the data flowing into the server. The *command center* helps manage the deployments across a community, provide a direct access to users and device data, and offers methods to communicate with end users who are taking part in a study.

We are beginning to explore the use of the described system for supporting aging-in-place and remote patient monitoring for clinical decision-making. Similar to existing systems that support remote patient monitoring, we will deploy a system of sensors and actuators that can record everyday behaviors, such as amount of time in each room and patterns of movement. Using Composer to perform analytics, we will examine the potential to discover possibly problematic behaviors (the stove is on and no one present) from the data and offer alerts or notifications to the resident to allow them to take action or dismiss. For individuals managing a chronic disease, more advanced sensing devices in combination with analytics may allow caregivers to intervene if behaviors, such as night-time bathroom trips increase - a known indicator of urinary tract infection as well as congestive heart failure exacerbation [3].

V. Conclusions

In this poster, we presented an IoT stack that supports a large number of device API calls and allows integration of user-defined analytics into the core of the dataflow. The system has been used to support IoT applications with tens of thousands of daily active users. In the future, we plan to use the system to support educational and smarthome applications.

References

[1] C.-J. M. Liang, B. F. Karlsson, N. D. Lane, F. Zhao, J. Zhang, Z. Pan, Z. Li, and Y. Yu, "Sift: building an internet of safe things," in *ACM SenSys 2015*.

[2] B. Ur, E. McManus, M. Pak Yong Ho, and M. L. Littman, "Practical trigger-action programming in the smart home," in *ACM CHI 2015*.

[3] M. J. Rantz, M. Skubic, R. J. Koopman, L. Phillips, G. L. Alexander, S. J. Miller, and R. D. Guevara, "Using sensor networks to detect urinary tract infections in older adults," in *IEEE Healthcom 2011*.