Poster Abstract: Creating an Abstraction to Exploit IoT Synergies

Zach Leidall
University of Minnesota, Twin-Cities
Minneapolis, Minnesota 55454
leid0065@umn.edu

Abhishek Chandra
University of Minnesota, Twin-Cities
Minneapolis, Minnesota 55454
chandra@cs.umn.edu

Jon Weissman
University of Minnesota, Twin-Cities
Minneapolis, Minnesota 55454
jon@cs.umn.edu

ABSTRACT

The Internet of Things (IoT) is enabling a multitude of new applications. Current systems exhibit inefficiencies which stem from applications being isolated and opportunities for sharing and contention resolution ignored. We propose a new system called Constellation that enables synergies between applications which we believe can make IoT systems more efficient. We explore such synergies and provide preliminary results to legitimize our approach.

CCS CONCEPTS

•Computer systems organization →Sensors and actuators; Real-time operating systems; Distributed architectures; •Software and its engineering →Scheduling; Power management;

KEYWORDS

Internet of Things (IoT), Edge Computing, Real-time Scheduling

ACM Reference format:

Zach Leidall, Abhishek Chandra, and Jon Weissman. 2017. Poster Abstract: Creating an Abstraction to Exploit IoT Synergies. In *Proceedings of The 2nd ACM/IEEE International Conference on Internet-of-Things Design and Implementation, Pittsburgh, PA USA, April 2017 (IoTDI 2017)*, 2 pages. DOI: 10.1145/3054977.3057321

1 INTRODUCTION

The Internet of Things (IoT) introduces enormous possibilities. With these comes a new set of problems. Heterogeneity of devices, energy constraints, limited compute resources, and scalability all make necessary new systems and programming models for the IoT.

We have designed a system for alleviating these issues called Constellation. Constellation abstracts many of the more menial details of managing a dispersed networked system while also enabling seamless integration of devices from different manufacturers. Our system makes use of the growing paradigm of edge or 'fog' computing, which uses resources close to the network edge to augment IoT devices without the latency or other drawbacks of the cloud as detailed in [5]. The result is a powerful abstraction for making IoT applications more efficient and easy to develop.

Constellation also enables sharing of hardware and data across multiple IoT applications allowing for greater economies of scale. This differentiates our contributions from previous work which has

focused on enabling single applications which exist largely isolated from each other. Imagine a smart furnace which maintains the heat for a home with a set of temperature sensors. Now imagine the homeowner has an application on their smartphone that raises the temperature when approaching the home, and allows it to drop when away. Under the state of the art, the furnace would likely 'own' the sensors, in the sense that only it could query them and other applications must interface with it to use them (assuming the furnace's developers bothered to make it compatible). With a system like ours; however, both applications could use the temperature sensor by using a Constellation hub. This means only having to purchase one set of sensors for multiple applications.

2 ABSTRACTING COMPLEXITY

Constellation abstracts the complexities of a distributed system from the IoT developer by using a network query engine similar to how tinyDB [4] uses a database abstraction for a sensor network. This abstraction greatly reduces the implementation efforts required by IoT developers. In one test application, a smart tea-kettle, almost one hundred lines of Java code was compressed into just 12 lines when using our query engine. The query language used allows a programmer to specify requests for sensor readings, actuations as well as to create asynchronous events. Below is a table which outlines the basic syntax of Constellation's Query Language (CQL).

Statement Type	Statement Syntax
Find	find devices with <cond>+ as <id>
Read	read <attribute> from <id> where <cond>+
Actuate	actuate <action> on <id> params <list>
Periodic Event	event <id> <stmt>+ period <int> <unit>
Aperiodic Event	event <id> <stmt>+ when <cond>

Figure 1: Syntax of CQL Statements

The five statements above cover all of the common IoT use cases. Because Constellation handles things like connectivity and communication, the programmer need only be concerned with creating simple English-like queries to interface with the IoT. The use of this declarative syntax also enables the back-end query engine to perform optimizations which are detailed in Section 4, including recognizing commonalities between different applications and adjusting the executing of the queries accordingly. Finally, while most IoT applications are coded with a static set of devices in mind, CQL's declarative syntax allows for devices to be swapped/optimized at runtime if a more ideal set matching the query are found.

Another aspect that adds complexity to IoT systems is the heterogeneity of hardware. We solved this issue by utilizing a device driver interface. This interface has a list of specified functions which the driver must provide to enable the system to make calls to the device. This approach has been utilized in operating systems for years and is also used in IoT systems such as IoT-MAP [3] as well as Amazon's Alexa's Skills capabilities. The declarative statements issued by the user are then translated to device driver calls.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

IoTDI 2017, Pittsburgh, PA USA

© 2017 ACM. 978-1-4503-4966-6/17/04...\$15.00

DOI: 10.1145/3054977.3057321

3 GETTING MORE OUT OF THE EDGE

Many IoT applications require devices which are embedded and have a small form factor. This conflicts with creating applications that are efficient and cost-effective. A common solution is edge or 'fog' computing [2]. Edge computing utilizes local compute resources to avoid the large network latency incurred by the cloud. Potential devices include cloudlets, PCs, and mobile devices. Constellation is meant to run on such devices and is written in Java to be cross-platform. IoT applications, running in the cloud or on more local devices, will interface with the system over the network.

Most edge-based IoT systems focus on running code on the edge for efficiency [2]. While our system enables this kind of optimization, we focus our work on finding ways for the applications which utilize this edge infrastructure to experience greater economies of scale by having our system recognize and exploit commonalities between applications. Section 4 details some of our optimizations. Our evaluation then goes beyond edge computing to show the plethora of opportunities for resource sharing in IoT systems.

4 RECOGNIZING AND ENABLING SYNERGIES

Constellation goes beyond previous work by discovering and exploiting commonalities such as sensors and actuators between different applications to increase the efficiency of overall IoT systems.

4.1 Time-sensitive Caching

The first optimization we make is recognizing the potential for multiple applications to share the same sensor reading by caching it at the edge node. Caching in the IoT adds an additional constraint as data being retrieved is often time-sensitive. This introduces the issue of balancing staleness constraints of the applications against the energy constraints of the devices producing the data. Figure 2 shows some initial findings with the use of caching. The chart compares the read times of a standalone edge-based application along with a different application utilizing a Constellation node, both with and without caching. Notice that when the Constellation node uses its cache, the latency difference compared to using a standalone edge node is negligible. This helps to legitimize the use of Constellation despite the extra network hop it requires.

Figure 2: Latencies for Different Edge Configurations

4.2 Real-time Scheduling of IoT Tasks

IoT tasks require real-time scheduling to account for deadlines. Traditionally, real-time scheduling is used in embedded systems where deadlines are used for safety-critical applications. We enhance a real-time scheduling algorithm and adapt it to the IoT. Scheduling is made more difficult in an IoT setting as the nondeterminism of network latency and human interference make it difficult to accurately predict latencies to allocate processing time. In addition, IoT

tasks can operate on variable amounts of data and devices, a factor real-time systems often need not be concerned with. Our algorithm attempts to account for these factors to enhance IoT scheduling.

4.3 Period Synchronization

Constellation also increases efficiency by synchronizing periodic sensor readings. Periods of different lengths can be synced by determining their least-common-multiple and delaying each read by small increments until they can be used to satisfy multiple applications. This can greatly reduce energy use in the system.

4.4 Conflict Resolution

All of the optimizations mentioned have focused primarily on improving sensor readings. Our research seeks to prove that actuations can be made more efficient as well. Constellation has the potential to recognize multiple actuations involving the same device(s) and alter scheduling decisions based on this information. The declarative nature of CQL also enables us to potentially make different actuation choices in the event that two actuations conflict and an alternate method can be used. For example, consider the temperature control application from earlier. If another application issued a query seeking to make the room brighter, the system could choose to turn on lights rather than open the shades as the latter could lead to a temperature increase. The ability to specify *what* instead of *how* declaratively enables this optimization. Others such as [1] have utilized a similar, though less robust, query approach.

5 CONCLUSION

Constellation enables IoT applications to share resources and experience economies of scale without adding code complexity. This is done via its resource manager which enables the discovery and exploitation of commonalities between different applications. All this is done without the programmer needing to add any additional code or consider any additional hazards in their programming.

ACKNOWLEDGMENTS

We would like to thank Particle, an IoT start-up in the Twin-Cities, for providing hardware used in the experiments. We also thank the NSF for support under grants CSR-1162405 and CNS-1619254.

REFERENCES

- [1] Brian Amento, Bharath Balasubramanian, Robert J. Hall, Kaustubh Joshi, Guey-oung Jung, and K. Hal Purdy. 2016. FocusStack: Orchestrating Edge Clouds Using Location-Based Focus of Attention. In *Edge Computing (SEC), IEEE/ACM Symposium on*. IEEE, IEEE, Washington D.C., USA, 179–191.
- [2] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, ACM, Helsinki, Finland, 13–16.
- [3] Sehyeon Heo, Sungpil Woo, Janggwon Im, and Daeyoung Kim. 2015. IoT-MAP: IoT mashup application platform for the flexible IoT ecosystem. In *Internet of Things (IoT), 2015 5th International Conference on the*. IEEE, IEEE, Seoul, South Korea, 163–170.
- [4] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. 2005. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM Transactions on database systems (TODS)* 30, 1 (June 2005), 122–173.
- [5] Ben Zhang, Nitesh Mor, John Kolb, Douglas S Chan, Ken Lutz, Eric Allman, John Wawrzyniec, Edward Lee, and John Kubiawicz. 2015. Proceedings of the 7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15). In *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)*. Usenix, Usenix, Santa Clara, California, USA.