Lessons learned from integrating batch and stream processing using IoT data

Hung Cao, Marcel Brown*, Lizhi Chen*, Riley Smith*, Monica Wachowicz
People in Motion Lab, University of New Brunswick
Fredericton, NB, Canada
{hcao3, marcel.brown, lizhi.chen, riley.smith, monicaw}@unb.ca

***Abstract*—The unbounded data streams generated by IoT sensors/devices are posing many technical challenges and requires a one-size-fits-all solution to cope with the massive amount and the high speed of the incoming IoT data arriving simultaneously. In this study, we try to integrate batch and stream processing in a unique system as a premise to handle *Volume* and *Velocity* aspects of IoT data simultaneously. In order to handle current, outdated, and historical IoT data streams, we built a cloud architecture to execute the analytical workflows using both batch and stream processing in a synergetic manner. A smart parking case study is used to evaluate the architecture and two experiments are implemented to demonstrate a web application for predicting parking spot availability. Herein, we learned our lessons that there are several hindrances to finding a middle ground where current, outdated and historical IoT data streams can be used in a strategic way.**

***Index Terms*—IoT data streams, batch processing, streaming processing, smart parking, cloud architecture**

I. Introduction

With the advent of the Internet of Things (IoT), a large number of sensors and devices are expected to interact and generate a massive amount of unbounded data streams. The insights generated from analyzing these IoT data streams can bring potential opportunities in sectors such as agriculture, health care, manufacturing, retail, marketing, smart cities, transportation, telecommunications, and tourism [1]–[3]. However, there is a trade-off between the completeness of observed data with respect to event times and the correctness of the analytical results since analytical tasks can either be executed with current available stream data or must wait until the completed stream data has arrived. This requires a complex data flow capable to extract value from different data streams at different points in time as well as integrate multiple analytical results over time in order to generate new insights. This challenge is further aggravated when fusing the extracted knowledge from historical IoT data to reach intelligent decisions in real-time.

Our research premise is that batch processing can deal with a large volume of IoT data, though it has not been traditionally designed to deal with the velocity of the incoming IoT data. In contrast, stream processing can cope with the very high data rate of the IoT data streams to yield the low-latency or speculative results, though it can not handle a massive amount of data in a short processing time. This paper proposes a cloud architecture for an IoT data flow where batch and stream processing are integrated to run analytical tasks in synergy by handling current, outdated, and historical IoT data streams.

The scientific contribution of this paper is to improve our understanding of the trade-off between completeness and correctness when using current, outdated and historical IoT data streams. A real-word case study, consisting of two implemented experiments, shows an IoT data flow (batch and stream processing) in smart parking and is used to describe the progress of our research work.

The remainder of this paper is organized as follows: Section II describes different components of our cloud architecture for IoT data; Section III is dedicated to implementing the architecture and describing our experiments on the smart parking as well as some preliminary results. Section IV discusses the lessons that we learned from the real-world experiments.

II. Our Cloud Architecture

This section describes the main modules of our proposed cloud architecture as shown in (Fig. 1). Each module is delivered by virtual machines running tools and data flows with persistent disk storage and consistent performance for both stream and batch processing.

Three types of data streams have been identified in our architecture:

- current IoT data streams are those with timestamps belonging to the the current time (i.e. now).
- outdated IoT data streams are when sometime has elapsed (i.e. just now).
- historical IoT data streams are those with timestamps belonging to the past.

In general, IoT applications will be developed based on the combination of different engines in our system. Depending on a specific scenario, the IoT applications can be implemented using different methods such as web apps, operational services, or 3rd party integrated applications. The whole system will be executed on core run times, which are managed throughout the OS, Resource Manager, or Container. There are several options to deploy our architecture such as stand alone or cluster deployment. Note that provision and orchestration are also deployed to mitigate difficulties in managing, distributing, and updating the system. Security is also taken into consideration.

This is a preprint. It will appear in *Proceeding of The 6th IEEE International Conference on Internet of Things: Systems, Management and Security,* Granada, Spain, October 22-25, 2019.

* M. Brown, L. Chen, and R. Smith contributed equally to this work.

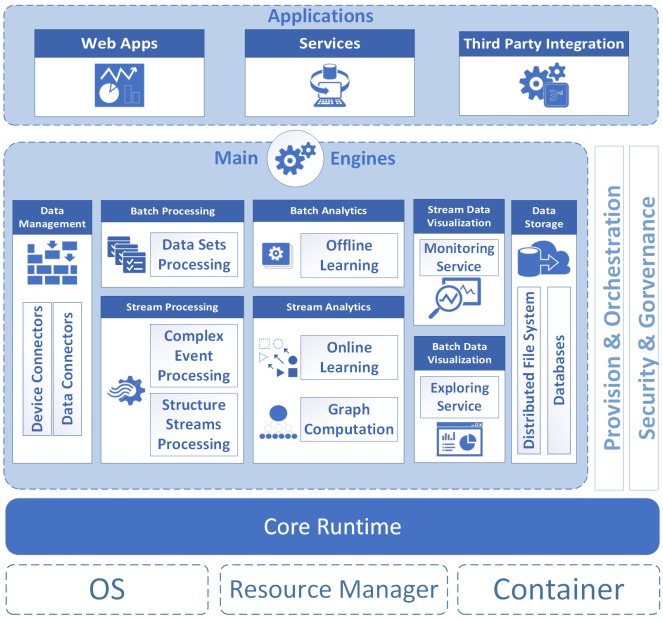


Fig. 1. The proposed cloud architecture for handling current, outdated, and historical IoT data streams.

In fact, our system aims to monitor and manage IoT data security across the different engines in the future.

The main engines of our IoT architecture can be described as one of the following:

1) Data management: There are two elements to manage the data flow in our cloud architecture: Device Connectors and Data Connectors. Device Connectors are the platforms to manage the network connection between the IoT devices, while Data Connectors are protocols to manage the flow of IoT data streams.

2) Stream Processing: This engine mainly deals with the continuous incoming of IoT data streams. It includes the Complex Event Processing (CEP) API and Structured Streams Processing API to manage and transform the raw data streams. The Structured Streams Processing API is used to build the programs that implement operations on current IoT data streams (e.g. filtering, updating state, defining windows, aggregating). The CEP API allows us to detect event patterns in an outdated IoT stream of events.

3) Streams Analytics: The main elements of this engine are the Online Learning Library and Graph Computation Library. The Online Learning Library is used to train the current stream data whenever they come to the system and it gradually builds the learning model. The Graph Computation Library builds graphs on the outdated data streams and analyzes them.

4) Stream Data Visualization: It is used to plot data whenever it comes to our system, with the aim of early detection of abnormalities for monitoring services.

5) Batch Processing: This component mainly deals with stream data that are accumulated in our data storage component. It includes the Data Sets Processing API. It is leveraged to build programs that implement operations on IoT historical IoT data (e.g., filtering, mapping, joining, grouping).

6) Batch Analytics: The Offline Learning Library is utilized to analyze the historical IoT data.

7) Batch Data Visualization: It plots the historical IoT data with the aim of assisting users with analysis and finding new insights through exploring services offered.

8) Data Storage: is the space where the incoming data streams or the results of the processing and analytical tasks reside. The storage space can be different types of databases (e.g. Document-based Store, Key-Value Store), a distributed file system, or an in-memory database.

III. Implementation Results

We built our cloud architecture using three virtual machines (VMs), which were provided by Compute Canada East Cloud as the IaaS resource. Each of them is a node, forming a cloud cluster. The detail specifications of each VM is available on Table I.

TABLE I
THE OVERVIEW OF THE CLOUD CLUSTER.

	VM1	VM2	VM3
Hostname	master.eastcloud	slave1.eastcloud	slave2.eastcloud
OS	CentOS 7.0 (x86_64)		
CPU	Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz		
# of Core	8	4	4
RAM	30GB	8GB	8GB
Disk (Main/Ephemeral)	20GB/1.15TB	20GB/800GB	20GB/800GB
IPv4	192.168.45.2	192.168.45.7	192.168.45.12

We have implemented a variety of open source software to deploy the main engines that were envisaged for our cloud cluster. Each of them plays an important role as a component in the overall cloud architecture. The implementation is illustrated in Fig. 2.

In this section, a smart parking scenario is used to evaluate our cloud architecture. The IoT data streams were generated in real-time whenever a driver parked his/her car and used the HotSpot Parking Application, combined with the pay station facilities of Saint John, NB, Canada. They were published to our Kafka broker as shown in Fig. 2.

We designed two experiments using a streaming and a batch processing workflows. The main goal was to improve the parking services. They are described below.

Experiment 1: The aim was to support the processing of IoT data streams on-the-fly, and then built an *Online Machine Learning (ML) Random Forest* model. The steps of the analytical workflow were: Ingest data streams every 5 seconds from Kafka broker → Perform data cleaning/preprocessing on-the-fly (using Flink) → Find patterns from data streams in that time window (using Flink CEP API) → Train an Online ML model (Tree/Random Forest using MOA) → Visualize for monitoring service (using Grafana).

Experiment 2: This experiment aimed to exploit the long term value from the IoT data by building a *ML Random Forest model in batch*. Therefore, we have used 2-year historical IoT data from our persistent database. The steps of our analytical

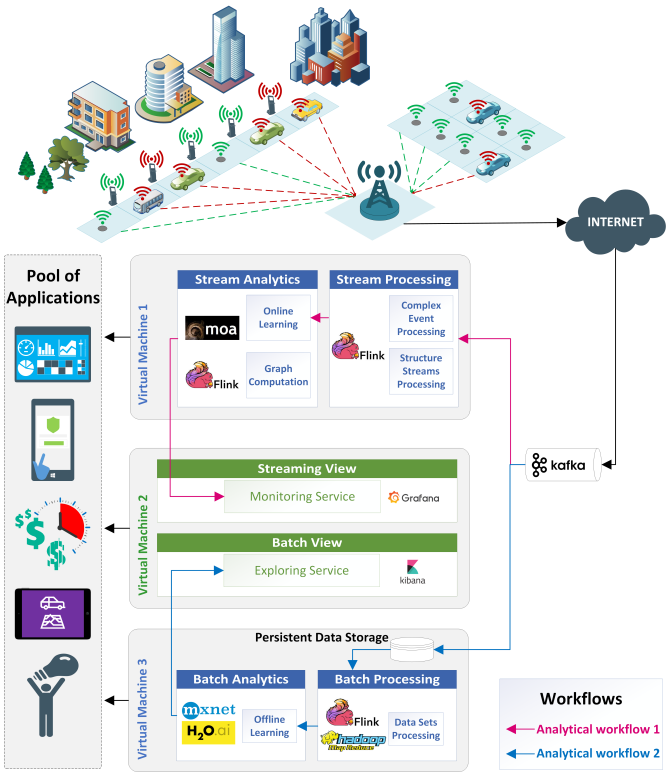
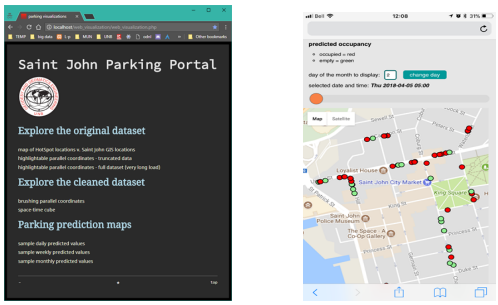


Fig. 2. Experiments of our smart transit scenario.

workflow were: → Train a ML Random Forest model in batch → Predict near future available parking spots (using $H_2O.ai$, MXNET) → Visualize the results for users (using Kibana).

Experiment 1 & 2 have successfully implemented and obtained preliminary results that were made available to users through a web application. We were able to predict parking availability using a Batch ML RF model and pre-process/clean incoming data tuples in batch and on-the-fly (Fig. 3).



(a) (b)

Fig. 3. Preliminary results from the smart parking scenario: *(a) Parking portal interface; (b) Web Application: Parking Availability in the next 2 hours (red: occupied; green: free)*

IV. Lessons Learned

Certainly there exists a continuum between stream processing and batch processing, and indeed our proposed IoT architecture is a step towards this direction. However, our overall argument is that finding a middle ground where current, outdated, and historical IoT data streams can be used in a strategic way is definitely not a trivial task. The challenges encountered are not related to an IoT architecture being better or worse than another, or that we should always seek for a minimum latency, or we should aim at one-size-fits-all solution.

In fact, our two experiments, although generating preliminary results, have provided us with new insights on the challenges ahead. They can be described as follows:

- ✘ *We should avoid duplicating our development efforts (i.e. modules) in order to support batch OR stream processing. Always having a pipeline (data flow) specific for streaming and another one for batch processing is not the best strategy when the aim is to support data analytics. Automated analytical tasks will require IoT architectures capable of handling one pipeline only, capable of both stream AND batch processing.*
- ✘ *The hurdles to be overcome in implementing and maintaining two systems as a unique solution is going to be unbearable for data scientists, especially in complex real-world scenarios.*
- ✘ *It is unclear how to identify the semantics of the computations from both batch and streaming processing in IoT. But one thing is clear, it will not only depend on the IoT application to define which streams are current, outdated, and historical. Networking latency needs to be studied in more depth in the near future, as well as how it will impact the semantics of the analytical computations.*
- ✘ *Unfortunately, the correctness of the results from batch and stream processing can fluctuate unpredictably. The completeness and correctness of both processing keep changing which makes it impossible to join the results before serving them in a visualization.*

Acknowledgments

This work was supported by the NSERC/Cisco Industrial Research Chair, Grant IRCPJ 488403-1. We would like to thank Compute Canada for providing us with the cloud resources, the City of Saint John, NB, Canada, and HotSpot Parking for giving us access to their IoT stream data.

References

[1] R. K. Lomotey, J. Pry, and S. Sriramoju, "Wearable iot data stream traceability in a distributed health information system," *Pervasive and Mobile Computing*, vol. 40, pp. 692–707, 2017.

[2] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, "Big iot data analytics: architecture, opportunities, and open research challenges," *IEEE Access*, vol. 5, pp. 5247–5261, 2017.

[3] H. Cao and M. Wachowicz, "The design of a streaming analytical workflow for processing massive transit feeds," in *2nd International Symposium on Spatiotemporal Computing*. Harvard University, Cambridge, MA, USA, 2017.