Section 4: Week 7: Time Series Data Management

Nate Bachmeier

TIM-7020: Database and Business Intelligence

January 19th, 2020

North Central University

Time Series Data Management

## Background

Data volumes continue to grow exponentially on an annual basis because of IoT, Cloud, Big Data, and Mobile (ICBM). This massive emersion of records introduces challenges for organizations as they need to gain timely insights into their data. Consider the raw resource costs associated with the data management lifecycle in terms of computing, storage, and networking. For instance, the Hadron collider generates 300GB/s, making an hour of sensor data approximately 8.6Pb (Basanta-Val et al., 2017)! Other examples exist across health care (e.g., medical monitoring devices), finance (e.g., high-frequency trading), retail (e.g., click streams), and manufacturing (e.g., industrial IoT) sectors as users and devices produce these time-series data streams. Traditional big data systems addressed these issues by staging data into an object or NoSQL store, then executing batch processing models such as Map-Reduce (Barika & Garg, 2019). While this approach works today, it will continue to become economically prohibitive going forward. Instead, the programming models need to evolve towards real-time stream processing that extracts and store subsets from the feed. As the programming model changes to iterative processing paradigms, it will cascade across the technology stack, fundamentally changing downstream interactions with data management systems. For instance, training deep learning models relies on offline batch processing versus real-time only systems that will require iterative learning algorithms (Bosch, Olsson, & Brinne, 2019) (Yang et al., 2019). Other changes will take place across infrastructure and business continuity monitoring, as time-series streams are highly contextual—traffic to the eCommerce site is 1000% higher than normal, are we under attack, or did our marketing campaign go viral? There are numerous other scenarios impacted, all of which increase agility for organizations towards dynamic market conditions.

## Problem Statement

The adoption of big data started to pick up steam around 2012 (Al-Sai, Abdullah, & Husin, 2019), and shortly aftward the industry needed to transition from relational to NoSQL technologies. This data management evolution needed to happen because the programming models could no longer meet the business needs through transactional patterns, and instead pivoted towards MapReduce solutions. The efficiency challenges with MapReduce and similar batch systems comes from their blunt instrument design. An analogy exists with “counting the number of red cars in a parking structure.” A batch processing approach would hire a dozen workers and comb through each lane to report back the aggregate sum. While this strategy works for the local mall, it becomes unrealistic at Disney World. Alternatively, a stream processing approach to the problem would place a couple of attendants at each gate that modifies a local counter. The use of real-time counters is more efficient in terms of computation, storage, and communication overhead—because of the iteratively updating model that does not need to retain the source information (the car in the lot).

Similarly, the accelerating growth of high-resolution time-series data continues to drive the need for innovation into shorter retention systems that deliver the same value. These changes to the programming model will cascade into the data management system and, therefore, into the requirements of downstream consumers, such as machine learning, business continuity, enterprise resource management, and safety systems. A clear understanding of these ramifications need to take place so that compensation strategies formed to remediate deficiencies across people (e.g., training), technology (e.g., tooling), organization structure (e.g., political resistance), process (e.g., governance), and data management (e.g., security and compliance) (Al-Sai, Abdullah, & Husin, 2019).

## Relevance and Significance

Between 2013 to 2020, the world’s data has grown from 4.4 to 44ZB and is showing no signs of slowing down (Mansouri, Nadjaran, & Buyya, 2017). This emergence of big data allows organizations to become more agile and respond to highly dynamic market conditions (Knabke & Olbrich, 2018), often at the expense of competitors with immature data management strategies (Al-Sai, Abdullah, & Husin, 2019). However, as these businesses continue down this path, the volume, variety, and veracity associated with integrating across multiple data sources become economically prohibitive to gain timely insights. The cost of sensors is also decreasing at an exponential rate, which is pushing industries to measure everything models. Consider a network of restaurant supply chains that can use cloud and IoT platforms to easily monitor and predict across ordering, transporting, preparation, and customer satisfaction —these sensors report into business intelligence systems everything from temperature anomalies to the performance of wait staff (McCrea, 2019) (Ma et al., 2018). Looking toward the forefront with Industrial IoT, many modern manufacturing plants are already becoming overwhelmed with the complexity of managing these time series streams (Frodigh, 2018).

So, (1) if all competitive organizations need to adopt big data strategies, such as those created by managing time series sensors, and (2) those complications are quickening every business cycle, then (3) finding solutions to those challenges is both highly relevant and significant to a broad audience. One paradigm shift that addresses these needs is real-time streaming data management solutions, as these technologies focus on iteratively processing and reduce the reliance on batch processing across external storage networks (e.g., SAN, data lakes, and data warehouses). That is not to say, data lakes and warehouses seize to exist more likely stream data management acts as an essential precursor working in tandem (McKendrick, 2019).

# Literature Review