

# Fast and Streaming Analytics in ATM Cash Management

Terpsichori-Helen Velivassaki<sup>a</sup> and Panagiotis Athanasoulis  
*SingularLogic, Achaias 3 & Trizinias st., Kifisia, Attica, Greece*

**Keywords:** Cash Management, Fast Analytics, Stream Analytics, Operational Database.

**Abstract:** Cash management across a network of ATMs can be greatly improved, exploiting a multitude of information sources, which however generate huge datasets, not possible to be analysed via traditional methods. Business Intelligence in the Banking, Financial Services and Insurance (BFSI) sector can be even more challenging when exploitation of real-time information streams is desired. This paper presents the uCash ATM cash management system, running on top of CloudDBAppliance, supporting fast analytics over historical data, as well as streaming analytics functionalities over real-time data, served by its Operational Database. The paper discussed integration points with the platform and provides integration hints, thus constituting a real use case example of the CloudDBAppliance platform, allowing for its further exploitation in various application domains.

## 1 INTRODUCTION

Even in the digital age, cash remains an essential component of the payment system, being the largest retail payment category by number of transaction for transactions under \$10 (Reserve Bank of Australia, 2017). During the last few years, in their attempt to achieve mobility to better serve their customers, banks across the globe are investing on next-generation ATMs, featuring advanced capabilities and offering customers the ability to perform more types of financial transactions than the ones are currently handled by customer service representatives. This will create a whole new perspective of the user ATM experience, also increasing the spatiotemporal heterogeneity of the cash demand and the data produced by the ATMs, derived from the new business value propositions of these new ATMs (ATM Industry Association, 2018). Additionally, as user mobility increases, so does the complexity of predicting the expected user withdrawal and deposit transactions volumes.


Banks and financial institutes nowadays face significant challenges in managing effectively ATM replenishment, based primarily on static prediction algorithms. This results in high cash-out rates in certain ATMs, while having excess leftover cash in other ATMs, returned to banks. Accordingly,

ineffective cash management in ATMs leads to considerable customer dissatisfaction due to delays until reloading empty ATMs, as well as complex and expensive cash logistics to reload the network.

Ideally, smart cash management should be able to catch cash demand, affected by external factors and not easily predictable, as e.g. the monthly salaries/pensions. For example, an ATM, located on the same level of a large shopping Mall as a store with special discounts for a single day only, could have increased cash flow. Also, cash demand in ATMs close to social events is expected to be higher than usual. Moreover, ATM traffic could be affected by combined factors, such as weather and location. Indicatively, ATMs close to beaches/seaside might realize increased demand during warm and sunny days or cash demand could decreased for any ATM during cold days. Another dependence can be found at seasonality, e.g. ATM cash demand is expected to grow during the days before bank holidays.

In this context, appropriate cash management is bound to ultra-fast analytics over huge datasets, produced by large interconnected ATM channel networks. Business objectives mandate for the capability of processing complex queries over thousands of columns of operational data, created by thousands of ATMs in the order of seconds.

On the other hand, strong data privacy and

<sup>a</sup>  <https://orcid.org/0000-0002-0362-4607>

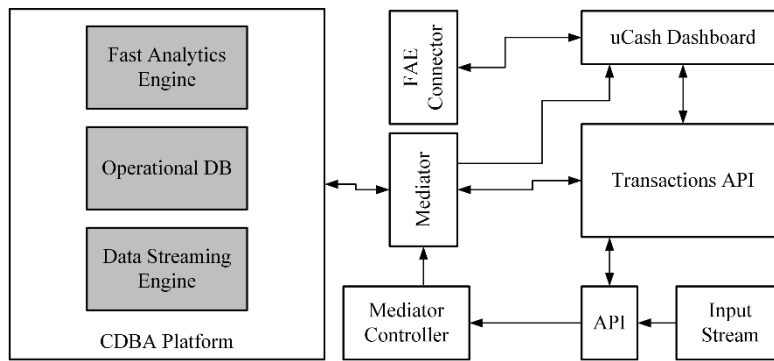


Figure 1: General architecture of the Cash Management application.

security requirements renders ATM cash management too critical to rely on existing clouds.

In this paper, ultra-fast ATM Cash Management system (uCash) is presented as a critical application, running on top of the CloudDBAppliance platform (CloudDBAppliance, 2019), incorporating a scalable Operational Database (DB), a Fast Analytics Engine (FAE) and a Data Streaming Engine (DSE), while ensuring high availability. The proposed system is able to identify both expected and unexpected changes in external factors affecting cash withdrawal from ATMs. To this, the proposed system relies on streaming information received from ATM channels, the social media, cooperating retailers, social trending sites, the weather, financial services or other sources are properly exploited (Velivassaki, et al., 2019). Also, uCash should be smart enough to identify short- or long-term changes, affecting cash demand from ATMs, in order to feed accordingly the cash logistics.

The rest of the paper is organized as follows. Section 2 presents the architecture of the uCash system. Section 3 highlights the exploitation and integration of the CloudDBAppliance platform and Section 4 concludes the paper.

## 2 UCASH ARCHITECTURE

The uCash system development is led by the general architecture presented in Figure 1. It delivers a high-level overview of the final uCash architecture, skipping the details of CloudDBAppliance platform integration, which will be step-wise detailed in the next paragraphs.

Overall, the uCash architectural components could be spilt in greater sets of components as follows:

1. Input Streams which include simulation components and external data provider

components. Simulation components comprise a set of microservices simulating events nearby ATMs as well as offers by commercial places nearby ATMs. External data provider components interact with public services to get information related to weather and currency rates. In particular, the relevant data include weather forecasts for arbitrary time periods concerning the places where ATMs are located (each simulated data item finally streamed into the uCash system contains a detailed 48-hours forecast) as well as the currency rates. To avoid excess activity from these components, caching strategies have been applied at the providers side, only invoking the external data providers when really needed.

2. The uCash application package including the uCash Dashboard and the Transactions API. The former is a front-end component interacting with the Transactions API and implicitly with the CloudDBAppliance platform and is used by the various actors of the system.
3. A set of mediation services enabling the seamless and technology-agnostic interaction of the uCash application package with the CloudDBAppliance platform. The components incorporated in this set of services are the Fast Analytics Engine (FAE) Connector, the Mediator (enabling integration with the Operational Database and the DSE) and the Mediator Controller that interacts with the Transactions Simulator to get the simulated data and, then, forward them to the Mediator.

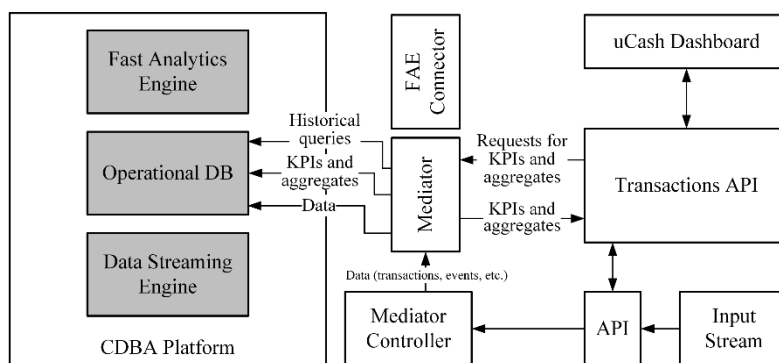


Figure 2: uCash integration with the Operational Database.

### 3 EXPLOITATION OF THE CLOUDDBAPPLIANCE PLATFORM

The integration of uCash with the CloudDBAppliance platform and its core technical components provide a set of benefits towards response times of simple or complex queries over the huge dataset used by uCash, which are summarized below.

1. Quick retrieval and aggregation of historical data (catalysed by the Operational DB);
2. Quick retrieval of aggregated KPIs based on real-time and historical data (catalysed by the Fast Data Analytics framework);
3. Real-time retrieval of KPIs and aggregates based on streaming data (catalysed by the DSE);
4. Real-time indications of correlation among the streamed data parameters (catalysed by the DSE ML algorithms).

The above summarise not only the integration context of uCash with the CloudDBAppliance platform, but also highlight the merits of the latter under a big-data perspective, a domain where traditional IT and data management architectures simply cannot satisfy.

#### 3.1 Integration with the Operational Database

The integration with the Operational DB is of paramount importance, since it not only acts as an in-memory database offering hot storage services at mass scale, but also as a single-entry point of data that can be exploited by the rest of the

CloudDBAppliance platform core components. In order to integrate with the Operational DB, uCash makes use of a Java-based Mediator component which exposes the JDBC connection by means of a RESTful API<sup>1</sup>. By means of this Mediation API, the uCash application package are able to:

- a) Create and maintain a connection pool for the incoming requests;
- b) Register transactions information to the Operational DB;
- c) Retrieve transactions information (aggregated or not) from the Operational DB.

Figure 2 depicts the information flow of the relevant uCash integration activities.

In detail, every time a transaction is generated, it gets communicated to the Mediator Controller which, in turn, interacts with the Mediator by transforming the incoming JSON data format to a JDBC-compliant statement, to be stored into the Operational DB.

Next, when an actor using the uCash Dashboard accesses the dashboard and requests a certain set of Key Performance Indicators (KPIs) to showcase, the following steps take place:

1. The request gets forwarded to the Transactions API, which transforms the request into a JDBC compatible-one and communicates it to the Mediator.
2. The Mediator performs the query against the Operational DB and gets the results.

The results are communicated to the uCash Dashboard via the Transactions API.

<sup>1</sup> As also documented for the case of the DSE, the Mediator actually comprises two mediator services, the one enabling

the interaction with the Operational DB and the other being responsible for interacting with the DSE component.

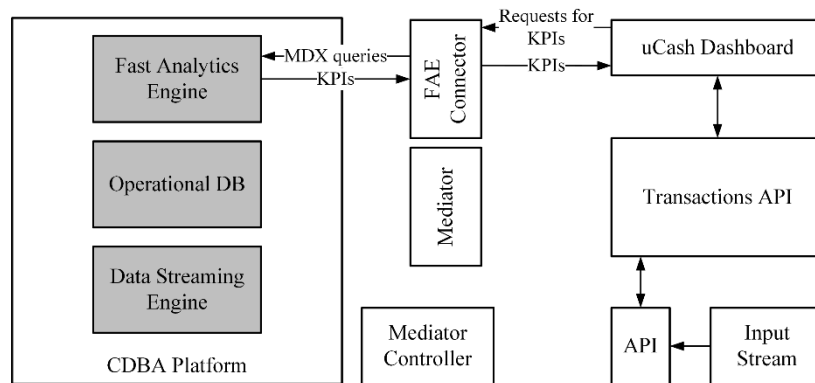


Figure 3: uCash integration with the Fast Analytics Engine.

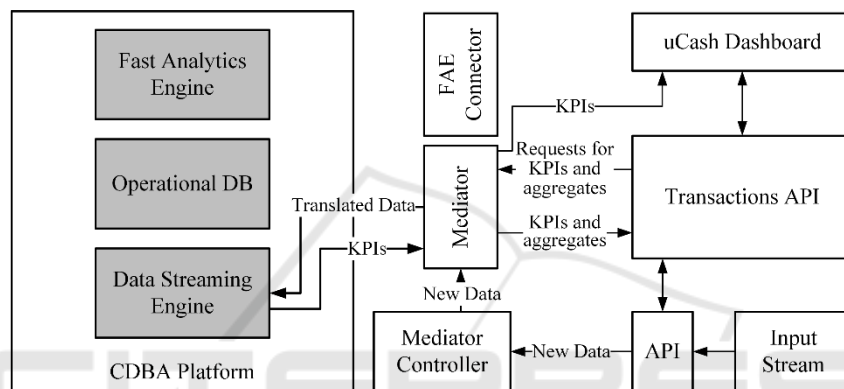


Figure 4: uCash integration with the Data Streaming Engine.

### 3.2 Integration of Fast Data Analytics Engine

The scope of this integration is the quick retrieval of KPIs and aggregates over historical data. Figure 3 presents the overall integration of the uCash application package with The Fast Analytics Engine. The integration flow comprises four steps as follows:

1. The user of the uCash Dashboard asks for the visualization of a set of KPIs.
2. The request of the KPIs is redirected to the FAE connector which, in turn, based on the input received, builds an accompanying Multidimensional Expressions (MDX) query.
3. The built MDX query gets communicated to the fast analytics engine in order to get the requested KPI/aggregated values;
4. The answer from the fast data analytics engine is, then, forwarded back to the AP connector which tunnels the question back to the uCash Dashboard.

It is worth mentioning that the fast data analytics

engine will be pre-configured to understand which data should be retrieved from the Operational DB so that KPI requests towards the fast data analytics engine are rendered possible.

### 3.3 Integration with the Data Streaming Engine

As already hinted and as happens with the case of the Operational DB, the integration with the DSE is catalyzed by the Mediator Controller and the Mediator service.

Indicatively, whenever a new transaction is generated and passes through the Mediator Controller, it gets pushed to a publish/subscribe bus for which the Mediator acts (at this point of the data flow) as a subscriber. The received input is then communicated to the DSE Mediator part of the Mediator which, in turn, translates the incoming data into tuples, able to be consumed by the DSE. As an example, for communicating the data to the query operated in the context of the integrated Machine Learning (ML) algorithms the transformations that take place are:

- a) Formulate the weather (current and forecast) data into ranked groups of weather types (e.g. Rain, Cloudy, Rain, Snow etc) so that they are better contextualized;
- b) Flatten the incoming JSON-formatted information into comma-delimited triplets of type:  
 $\langle \text{timestamp} \rangle, \text{atm\_} \langle \text{atm\_id} \rangle \_ \langle \text{variable} \rangle, \langle \text{value} \rangle$  where the timestamp indicator is of type YYYYmmDDHHMM.

Next, the input is forwarded to the DSE, activating the enabled streaming queries (e.g. for calculating aggregate cash demand for one or more ATMs) and the ML algorithms calculating the correlation indices among the provided variables. When the online processing is over, the results are received by the Mediator which, in turn, publishes them back to the Publish Subscribe bus, feeding the Mediator Controller. The latter informs in real time the uCash Dashboard. Figure 5, below, depicts the above process.

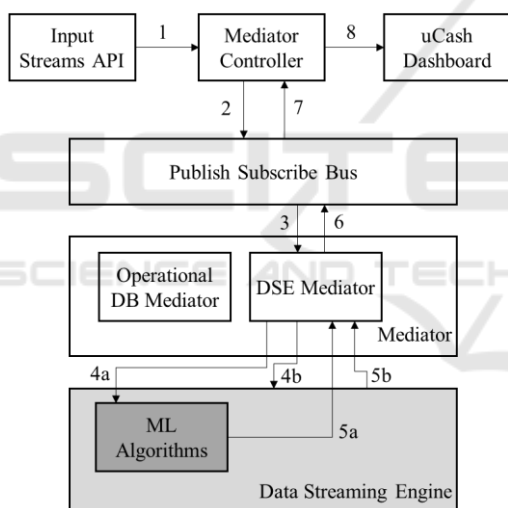


Figure 5: Detailed steps of integration with the DSE.

## 4 CONCLUSIONS

This paper presented the uCash Cash Management system as a use case of the CloudDBAppliance platform. Implementation details have been provided, including the high-level architecture of the system, which highlights use case specific components and interaction points with the CloudDBAppliance platform. The uCash system has been developed on top of the CloudDBAppliance platform, exploiting appropriately the platform merits in the Bank sector. It has to be noted that the integration and exploitation

hints presented in this paper can be enablers for the platform exploitation in other application areas and under various scenarios, as well. As a next step, the end to end functionalities of the uCash system, will be validated under the final CloudDBAppliance platform, coupling ultra-fast software with high-end hardware components.

## ACKNOWLEDGEMENTS

This work has been partially conducted within the CloudDBAppliance project Grant number 732051, co-funded by the European Commission as part of the H2020 Framework Programme.

## REFERENCES

- ATM Industry Association, 2018. *Business Value Propositions for Next Generation ATMs*. [Online] Available at: <https://www.atmia.com/files/committees/consortium-for-next-gen-atms/bvps-for-next-generation-atms-published.pdf>
- CloudDBAppliance, 2017. *D7.1: Use Cases Requirements Analysis*, s.l.: H2020-732051 CDBA Deliverable Report.
- CloudDBAppliance, 2017. *D7.2: Use Cases Design*, s.l.: H2020-732051 CDBA Deliverable Report.
- CloudDBAppliance, 2018. *D7.3: Use Cases Implementation Version 1*. s.l.: H2020-732051 CDBA Deliverable Report.
- CloudDBAppliance, 2019. *CloudDBAppliance - European Database Appliance*. [Online] Available at: <http://clouddb.eu/>
- Reserve Bank of Australia, 2017. *Reserve Bank Bulletin*, Sydney: s.n.
- Velivassaki, T.-H., Athanasoulis, P. & Trakadas, P., 2019. *uCash: ATM Cash Management as a Critical and Data-intensive Application*. Heraklion, s.n.