I-BiDaaS Minimum Viable Product (MVP)

Dr. Dusan Jakovetic - December 27, 2018

The I-BiDaaS consortium is bringing several technological advances in the upcoming months, as it will be presented in the technology-focused month 12-deliverables D2.2, D2.3, D3.1, D4.1, D5.1, and D5.2. In short, we are presenting the I-BiDaaS minimum viable product – MVP (D5.2), as well as first reports on data fabrication (D2.2), visualization and monitoring (D2.3), batch analytics (D3.1), complex event processing (D4.1), and resource management (D5.1). We briefly outline here the overall I-BiDaaS MVP solution and the corresponding use case, as well as the batch analytics algorithm that lies at the core of the use case. Other technological advances will be described in the upcoming months' articles.

The I-BiDaaS MVP (Figure 1), lead by ITML, already incorporates a good number of I-BiDaaS technologies, demonstrates integration between a number of different components, and involves both batch and streaming processing. The corresponding use case is provided by CAIXA and is about finding relations between bank customers for security checks, given a set of their IP address connections to online banking services.

The use case (Figure 1) works with realistic synthetic data generated with the IBM's Test Data Fabrication tool (TDF). Within the batch analytics part, TDF first fabricates the data and saves it in a Sqlite file, after which the data is passed to Cassandra database via an adapter that is being developed by ITML. BSC is creating an analytics algorithm that finds relations between the customers based on their IP addresses and outputs groups (clusters) of interrelated customers. Within the streaming analytics part, TDF emulates near real-time transactions generation and passes them to the SAG's Universal Messaging (UM) via the MQTT messaging protocol. The SAG's Apama complex event processing engine then checks, with the help of the customer groups identified by the BSC's batch analytics algorithm, whether current transactions correspond to customers that are in a relationship or not, and accordingly generates alerts about potentially suspicious transactions. Both the batch and streaming analytics results are visualized via the AEGIS's advanced visualization toolkit.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780787.

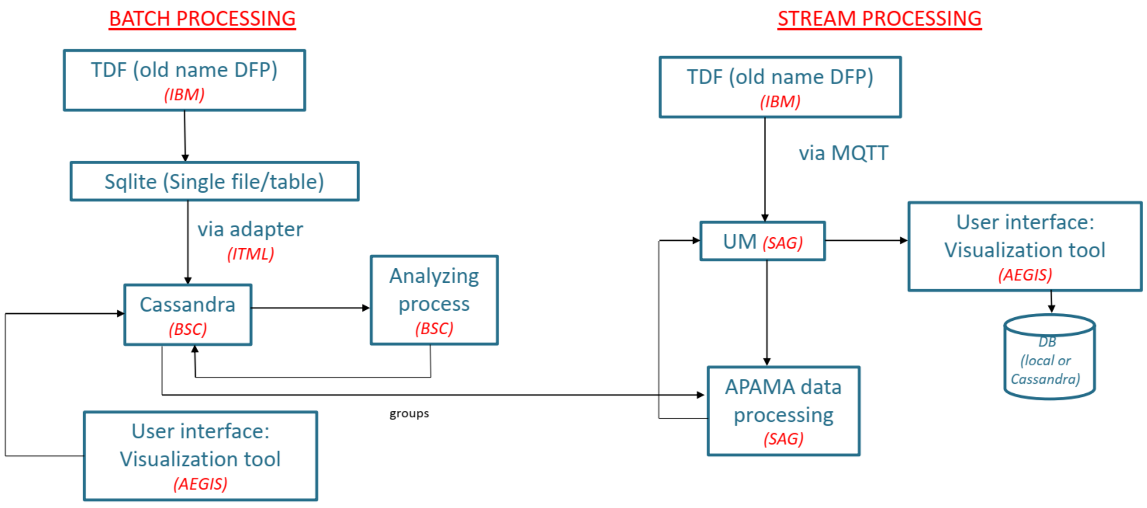

Figure 1: The I-BiDaaS MVP architecture.

At the core of the use case above lies the batch analytics algorithm on grouping bank customers based on their IP addresses. The algorithm implementation relies on the BSC's COMPSs programming model[1]. COMPSs provides an easy and convenient way to parallelize sequential codes. The programmers just need to identify the tasks that can exploit the parallelism of the hardware and to mark the parallelizable tasks using python decorators. Subsequently, the COMPSs runtime automatically identifies the dependencies between tasks (between input/output parameters of different tasks) and implements synchronization operations transparently to the user. In the I-BiDaaS MVP batch analytics algorithm, we have defined several parallel tasks, not only to exploit parallelism, but also to benefit from the automatic detection of the dependencies. The main tasks of our algorithm are the following:

- Divide the input data into chunks (randomly), and for each chunk, initialize the data structures with the input data in parallel;
- Merge the results of the initialization;
- Compute the blacklist with the public IP addresses that can be discarded;
- Identify the months that appear in the input data;
- For each month, compute the relations between users in parallel;
- Merge the information about relations detected for each month.

As the temporary data is stored in memory, it is necessary to add merge tasks to join the partial results. However, if we use a database to store this temporary data, we can omit the synchronization and the merge tasks, maximizing the degree of parallelism. We are exploring three different approaches on how to deal with this splitting-parallel computing-merging scheme (Figure 2): the approach based on COMPSs, the alternative using Spark/Hadoop engines and, finally, the one that can be obtained using Hecuba as the backend for the data.

[1] COMP Superscalar, an interoperable programming framework, Badia, R. M., J. Conejero, C. Diaz, J. Ejarque, D. Lezzi, F. Lordan, C. Ramon-Cortes, and R. Sirvent, SoftwareX, Volumes 3–4, December 2015, Pages 32–36, DOI: 10.1016/j.softx.2015.10.004

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780787.

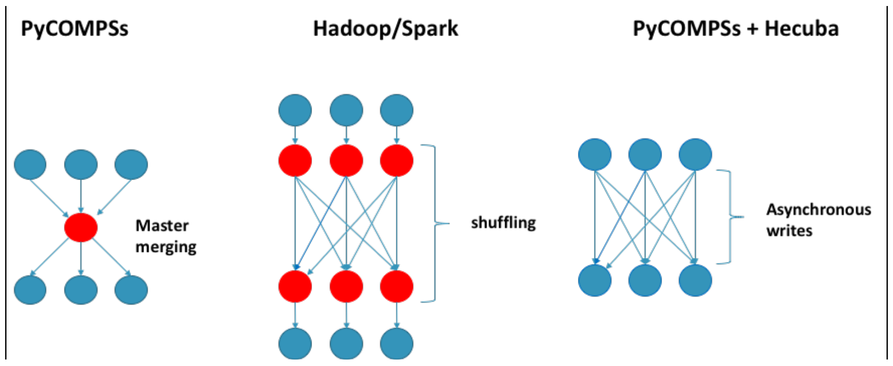

Figure 2: The split-compute-merge solutions for the I-BiDaaS MVP batch algorithm.

Find & Follow us

Website | Twitter | LinkedIn

Zenodo | OpenAIRE | GitHub

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780787.