The 37th International Conference on Conceptual Modeling, October 22-25, 2018 Xi'an, China

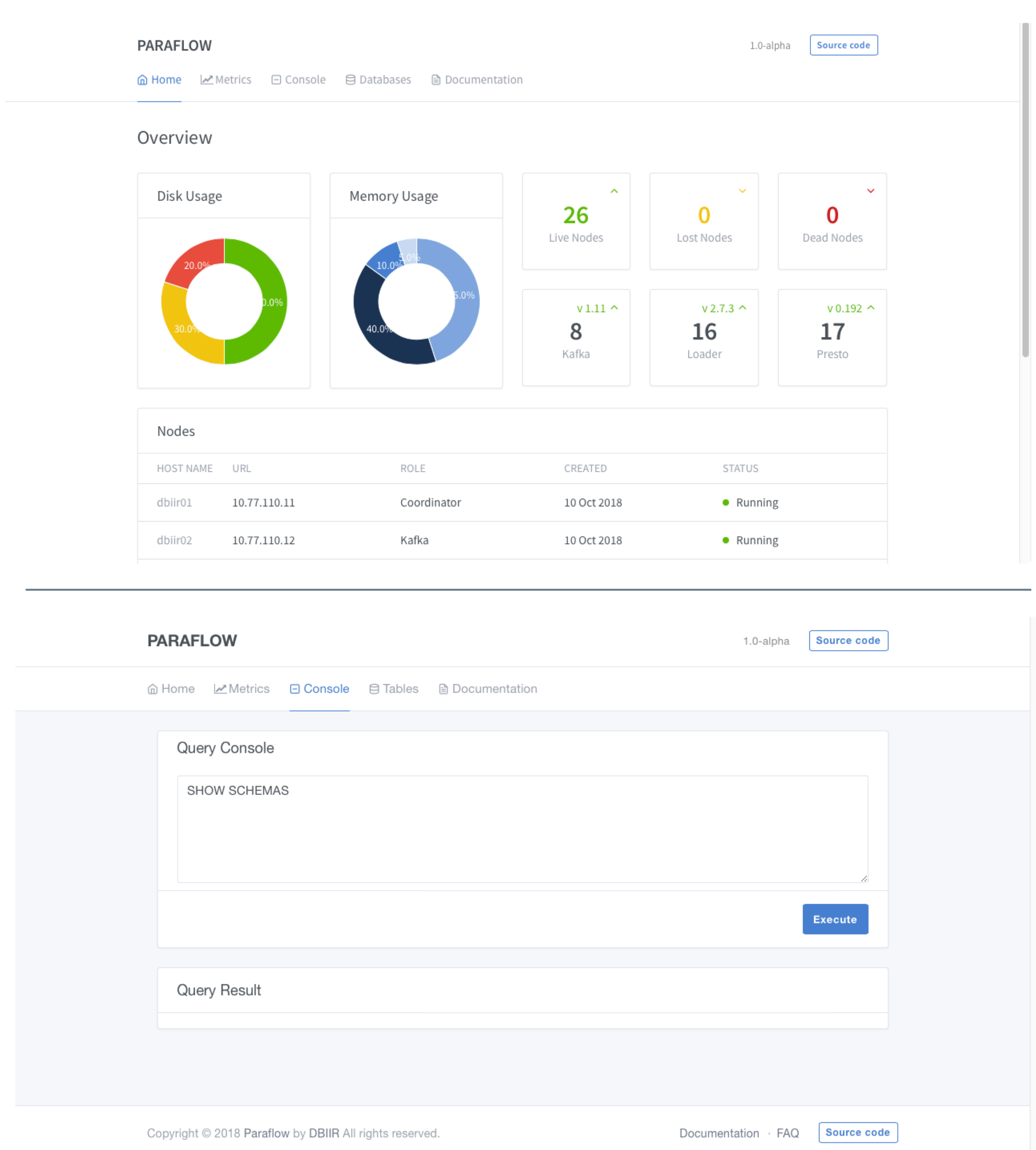
Towards Real-time Analysis of ID-Associated Data

Guodong Jin, Yixuan Wang, Xiongpai Qin*, Yueguo Chen, Xiaoyong Du
DEKE, MOE and School of Information, Renmin University of China, China
qxp1990@ruc.edu.cn

Motivation

- ID-associated data are sequences of entries, and each entry is semantically associated with a unique ID. Examples are user IDs in user behaviour logs of mobile applications and device IDs in sensor records of self-driving cars.
- Nowadays, many big data applications generate such types of ID-associated data at high speed. Typically, most queries over ID-associated data are ID-centric – they retrieve and analyze data of a specific group of IDs over a period of time.
- To generate valuable insights from such data timely, the system needs to ingest high volumes of them with low latency, and support real-time analysis over them efficiently.

Demonstration Interface

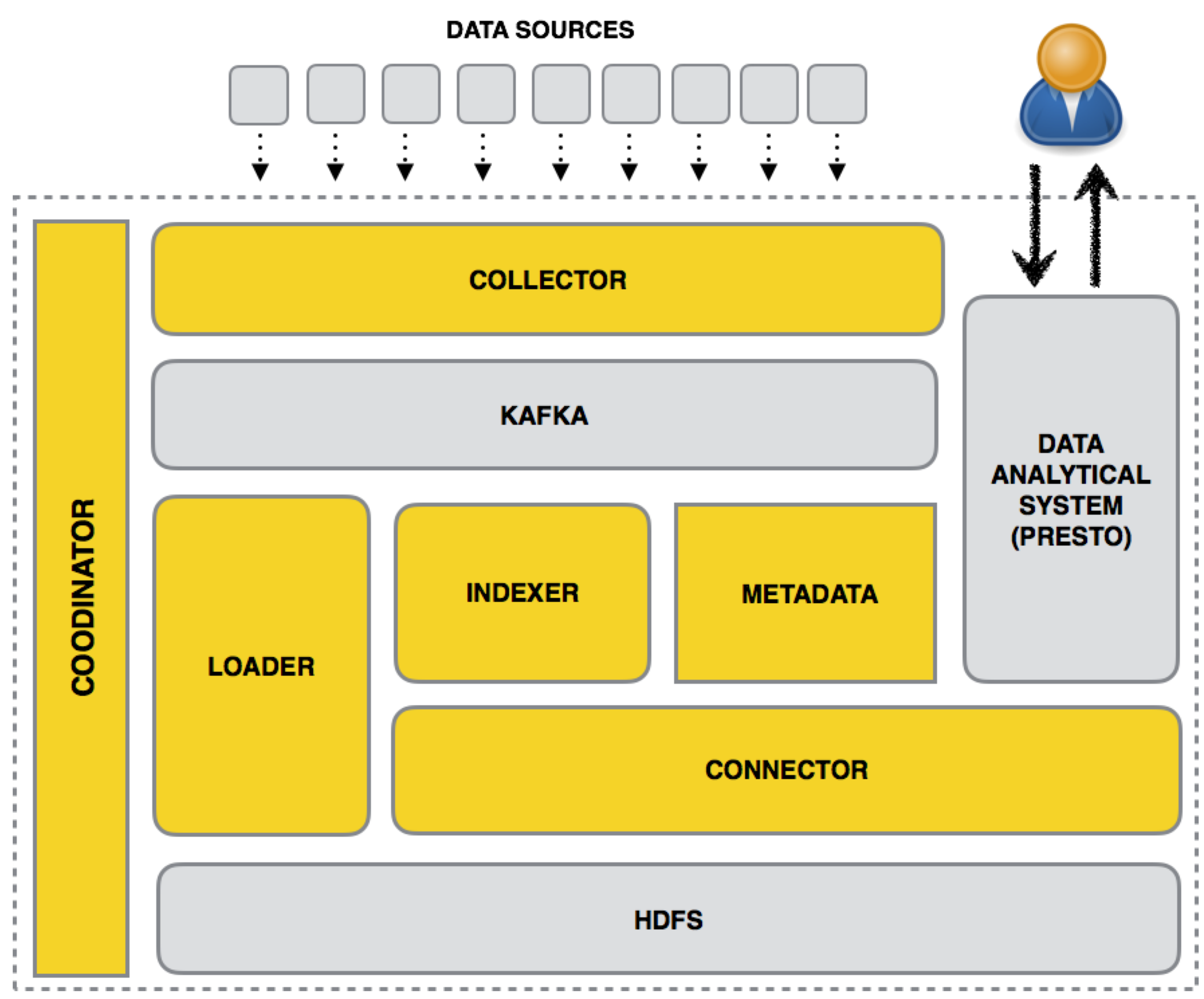


Acknowledgement & Open Source

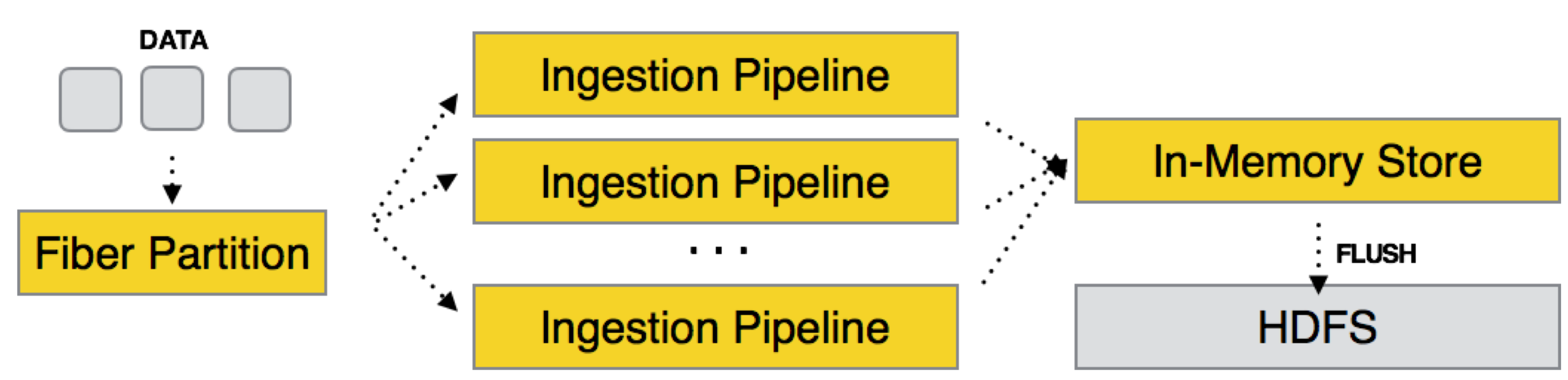
This work is supported by Science and Technology Planning Project of Guangdong under grant No.2015B010131015, 863 key project under grant No.2015AA015307, and the National Science Foundation of China under grants No.61472426, U1711261, 61432006.

OPEN SOURCE on GitHub:
https://github.com/dbiir/paraflow

System Architecture



Key Techniques



Fiber Partitioning

The partitioning scheme is based on the consistent hashing. For each data entry, a hash function is applied to get a hash value of its ID, and the range of hash values is split into *k* intervals, mapping to *k* fibers.

Parallel Ingestion Pipeline

Data pulled from Kafka are maintained as fiber streams in separate threads. In each thread, data are sorted by their associated IDs and generation time. Once the sorted buffer reaches its max size or user-defined lifetime (elapsed time since the last reset), it compacts all data into the memory store and resets.

In-Memory Columnar Storage

Memory store organizes data as columnar segments(on-heap, off-heap, on-disk). To reduce memory footprints, lightweight compression schemes such as run length encoding, bit packing, and dictionary encoding are utilized to compress segments. With little overhead, queries can run directly on these lightweight compressed segments.