Harmonized solution design of data stations

We take Klepmann (2017) as our starting point, who states that "Many applications today are *data-intensive*, as opposed to *compute-intensive*. Raw CPU power is rarely a limiting factor for these applications—bigger problems are usually the amount of data, the complexity of data, and the speed at which it is changing."

Generically, we want:

Reliability	Scalability	Maintainability
tolerating hardware & software vaults	Measuring load & performance	Operability, simplicity & evolvability
human error	Latency percentiles, throughput	

We focus on analytical data systems, with different patterns from transactional data systems.

0.1 Detailing the layers of a data station

TO DO: provide detailed layers, and explain how interoperability works across the layers:

- Storage layer (technology): all reference architectures stipulate use of S3-compliant blob storage
- Data and metadata (application): resides in the data station
- We propose to move towards open table formats, that is, Apache Iceberg, whereby storage and compute can be separated

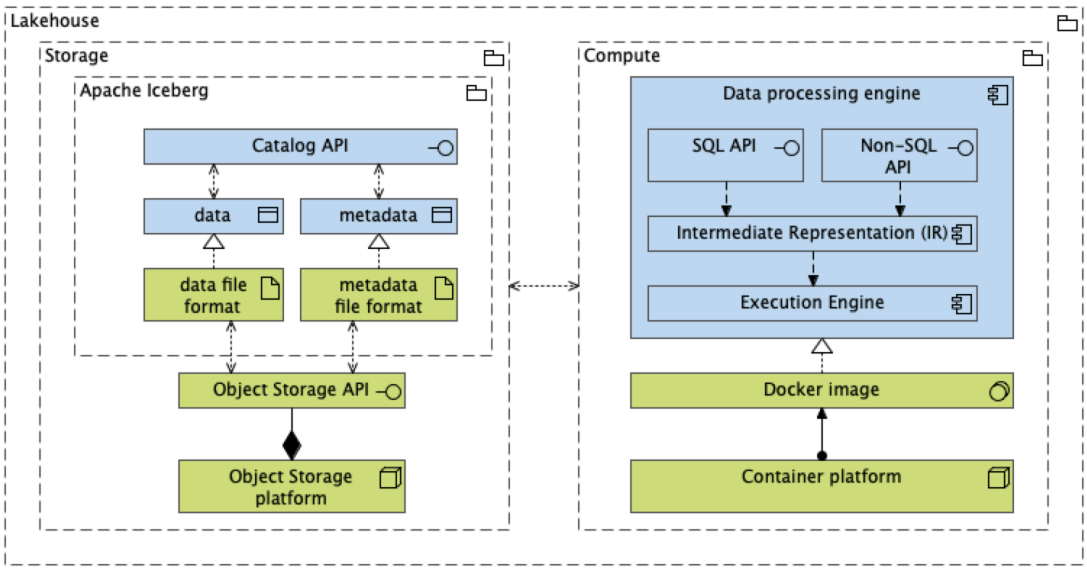

Figure 1: Solution of a minimal lakehouse that sits at the core of a data station

0.2 Detailing the data conformity zone

TO DO: explain that

- data conformity zone is essentialy a lakehouse pattern
- the architecture of a lakehouse has stabilized and converged towards:
  - **Colum-oriented storage and memory layout:** Apache Arrow ecosystem, including Apache Flight
  - **Late-binding with logical data models most suited for analytics:** ELT pattern with zonal architecture
    - *staging zone:* hard business rules (does incoming data comply to syntactic standard), change data capture
    - *linkage & conformity zone:* concept-oriented tables, typically following a data vault modeling principle, ascertain referential integrity across resources, with tables per concept and linking tables. Mapping to coding systems. Entity resolution for record linkage at the subject level
    - *consumption zone:* convenient standardized views like an event table (patient journey, layout for process mining) with uniformity of dimensions using a star schema

0.3 Detailing the trains

TO DO: explain

- difference between centralized and distributed federated learning (causes lots of confusion)
- basically Train is a generalization of all types of computes
- difference between
  - Train for secondary use, which usually with batch-wise, less strict latency requirements
  - Train for primary use, like API call and messaging, with stricter latency requirements. This also includes deployment of AI for inference

0.4 The lakehouse architecture as the *de facto* standard for populating data stations

The PHT architecture does not specify *how* the data stations should be populated with data. Also the DSB2 only describes how the 'Data, Services and Offerings descriptions' building block should provide data providers the tools to describe a data product appropriately and completely, that is, tools for metadata creation and management.

One of the key questions of this paper is to detail the 'data conformity zone' as defined in the Cumuluz canvas as the functionality through which the data station is populated

0.5 Parking lot: operational readiness assessment

- Andre: how do we manage the project, through which governance do we want to decide the priorities
  - We want to run it as a proper open source project

    - But we need to make sure other people contribute to it (not just Frank and Bart)
    - SURF as new candidate are managing organization
    - Andre: action for follow up
- Point Johan: decision principles i) logs from nodes; ii) logs from central server. What is the philosophy what we want to make visible, because it has implications for security
    - User point of view: we want some level of central visibility
    - Software point of view: should we build this in vantage6, or should we integrate another existing open source package
    - Johan:
        - we should separate out different types of logs
        - maintenance dashboard
    - Yannick: clarifies finding in debugging SocketIO time-outs
        - splitting pay-loads
        - optimizing proxy
        - Bart & Andre: known issues, in previous version of vantage6 it was solved in fork
    - action Tim, Marlou & Yannick: PR July is a short-term
    - Larger question: should we continue with SocketIO?
        - Melle: use SocketIO for short messages, but don't use it for logging training
- Hardware failure
    - vantage6 v5 will be Kubernetes, but both MDW and DHD are in favour of keeping Docker as a simpeler solution
    - Andre: we agree on technical perspectives
    - Ideas
        - keep v4 in Docker
        - build v6 with Kubernetes
        - possibility for allowing hybrid, Docker option in v5
        - contractual
            - MDW: have general contracts
            - DHD: does specify in contract that it is Docker Engine
        -

Bibliography