I. Overview of Key Elements in Big Data Systems

The primary objective of this section is to delineate the crucial constituents of big data systems. The components under consideration are derived from the systematic literature review (SLR) as referenced by [1] and [2], complemented by a brief literary review of the existing knowledge.

Addressing the second research question (RQ2), the architectural references (ARs) presented in Table 1 were assessed and juxtaposed to pinpoint recurrent elements in big data architectural references. While some ARs, being brief, offered minimal insights, others, like NIST, were notably exhaustive. A prevailing observation was the tendency of ARs to be rooted in existing knowledge, substantiating the idea that crafting ARs from a foundation of pre-existing knowledge can be more potent than starting afresh.

In a systematic pursuit of this inquiry and subsequent to our data collation, we enumerated all the elements from every big data AR discussed prior. These elements find their description in Table 2. One challenge was the variation in terminologies used across different studies to delineate architectural components. While architectural definition languages, like Archimate, find limited adoption, most researchers resort to bespoke ontologies using simple diagrams. This renders comparison challenging, often necessitating translation from one conceptual framework to another.

An automated textual assessment was initially conducted on the naming of these components, aiming to discern trends and word preferences. The graphical representation of these terminologies is illustrated in Figure 3. Notably, terms such as 'big data application provider' and 'big data framework provider' emerged as common nomenclatures, possibly attributed to the influence of NIST BD AR (s17). Universally accepted terms included 'data consumer' and 'data provider', with 'layer' being the chosen descriptor for logically segmenting various AR components.

Distilling the above, to aptly address RQ2, we meticulously analyzed these components and sorted them based on functionality into: 1) Big Data Management and Storage, 2) Data Processing and Interface Applications, 3) Big Data Infrastructure.

A. Big Data Management and Storage

The inherent diversity of data is one of big data's quintessential hallmarks. This multifaceted nature of data necessitates the exploration and adoption of a variety of storage methodologies. An apt exemplification of this trend is the use of 'polyglot persistence'. In contexts where real-time and high-performance data is paramount, databases like ScyllaDB - renowned for their ultra-low latency and scalability - emerge as the optimal choice. Conversely, for scenarios demanding intricate relationships between data entities, the prowess of modern non-relational databases other than Neo4J, such as ArangoDB with its multi-model capabilities, becomes evident.

Selecting an appropriate database, or even multiple databases, embodies a pivotal architectural choice. This decision extends beyond mere storage, affecting patterns of data access and caching. For instance, in the realm of distributed systems specializing in micro-services architecture, many professionals are leaning towards the Command Query Responsibility Segregation (CQRS) pattern, known for bolstering the efficiency of event-driven applications. Two essential cornerstones of big data architectures are undeniably the storage type and its associated access pattern.

However, the prevailing sentiment in big data reference architectures (BD RAs) is an inclination towards monolithic storage strategies, epitomized by data warehouses and data lakes. While the conventional model of data staging and storage within data warehouses might seem ill-equipped to manage extensive big data volumes, it's intriguing to note the persistence of such methodologies in some modern proposals. A noteworthy element in many BD RAs is the data lake concept. Essentially, a data lake acts as a repository capable of accommodating a plethora of data forms, both internal and external. Data is typically fetched from these lakes to undergo transformation, a process delineated as LET (load, extract, transform), contrasting the older ETL (extract, transform, load) technique.

Drawing parallels between Business Intelligence (BI) and big data, one notices stark contrasts in the nature of their source data. In a similar vein, data lakes and data warehouses exhibit distinct characteristics. Traditional data warehouses predominantly rely on relational databases, which might restrict analytical flexibility and surge operational costs. Data lakes, however, offer a reservoir to store diverse data without pre-defined schemas, enhancing adaptability. Yet, this adaptability isn't without pitfalls. The unchecked dumping of data into lakes, disregarding its organization and consumption patterns, might inadvertently morph these lakes into inaccessible data swamps. Implementing data governance and proactive metadata can be potential antidotes to such scenarios.

Synthesizing the above, it's apparent that BD RAs are anchored in three dominant paradigms:

1) Enterprise data warehouse model.
2) Data lake model.
3) Multi-modal cloud-based model.

While the enterprise data warehouse model leans heavily on monolithic structures with ETLs and established data processing pathways, the data lake model resonates with a later-stage data processing approach. The third, cloud-centric paradigm seamlessly integrates distributed systems' elements and cloud functionalities. Some RAs discussed in [1] operate at an elevated abstraction layer, making it challenging to infer storage natures or data pipeline characteristics. One good example of such RAs is the NIST BD RA [3].

Yet, a broad spectrum of concerns often goes unaddressed in the realm of big data management. Critical areas like security, privacy, metadata management, and data quality sometimes appear sidelined. The scalability of some RAs in light of burgeoning data sources and their adaptability to evolving regional data privacy mandates also remain subjects of scrutiny. Discussions encapsulating data architecture, discoverability, and interoperability are conspicuously absent.

B. Data Handling and Interface Architectures

Big Data (BD) systems commonly navigate through two primary data handling realms: stream and batch processing. Stream processing, also known as rapid processing, addresses urgent tasks like identifying potentially fraudulent credit card activity. On the other hand, batch processing takes its time, ideal for in-depth analyses such as regression studies.

The chosen mode of processing, be it stream or batch, depends largely on three data attributes: its variety, volume, and velocity. For instance, real-time algorithms applied in stream processing often employ stateful in-memory data structures, like Hyperloglog, for instantaneous computations. Further customizations in streaming might involve strategies such as tuple-at-a-time or micro-batch processing. Contrarily, batch processing might avoid these techniques. Instead, an architect could lean towards methods like MapReduce or Bulk Synchronous Parallel (BSP) for more volume and velocity-driven data needs.

The literature presents various layers of granularity when discussing data handling. While certain studies, such as S19, delve deep into the intricacies of the data handling pipeline, others like S15 simplify it to terms like 'batch' or 'stream' processing.

Furthermore, we discerned two main approaches to data handling. One methodology integrates both batch and stream processing within a singular architectural frame, while the other differentiates between the two. This distinction is evident in the Lambda and Kappa architectural philosophies and their derived Reference Architectures (RAs). Some RAs, like S17, even introduce a triad: the batch node, streaming node, and interactive node.

In terms of BD interface communication, two patterns emerge. Some RAs showcase a simple 'access layer', as seen in S21 and S17. Others present multifaceted components tailored to unique needs, such as Machine Learning (ML) or Business Intelligence (BI), as observed in S16.

A few RAs emphasize the delineation between input/output interfaces and those between nodes, like s22. Yet, others might simply indicate interfaces with basic graphical cues, such as arrows. Despite their importance, contemporary RAs appear to lack a detailed focus on these interfaces, with only a few exceptions like s15's 'provider-request interface'.

REFERENCES

[1] P. Ataei and A. Litchfield, "The State of Big Data Reference Architectures: A Systematic Literature Review," in IEEE Access, vol. 10, pp. 113789-113807, 2022, doi: 10.1109/ACCESS.2022.3217557.

[2] Ataei, Pouya and Litchfield, Alan T., "Big Data Reference Architectures, a systematic literature review" (2020). ACIS 2020 Proceedings. 30.

[3] Framework, D. N. B. D. I. (2015). Draft nist big data interoperability framework: Volume 6, reference architecture. NIST Special Publication, 1500, 6.