CLOUD-NATIVE DATA MANAGEMENT FOR MULTI-MODEL ANALYTICAL WORKLOADS

A TECHNOLOGY REVIEW FOR ARCHITECTURE SELECTION

Matthias Pohl, Arne Osterthun

EUDAT 2026, Prague, Czech Republic

DLR

German Aerospace Center (DLR)
At a glance



- DLR is the research and technology centre of the Federal Republic of Germany for aeronautics and space as well as energy, transport, security and digitalisation
- German Space Agency for the realisation of German space activities
- Projektträger oversee funding programmes, support knowledge transfer
- National and international cooperations
- Supporter for industry, authorities, administrations, public stakeholders

11.000 Employees | 54 Institutes/Facilities | 30 Locations (Headquarter Köln) | Field Offices: Brüssel, Paris, Washington, Tokio

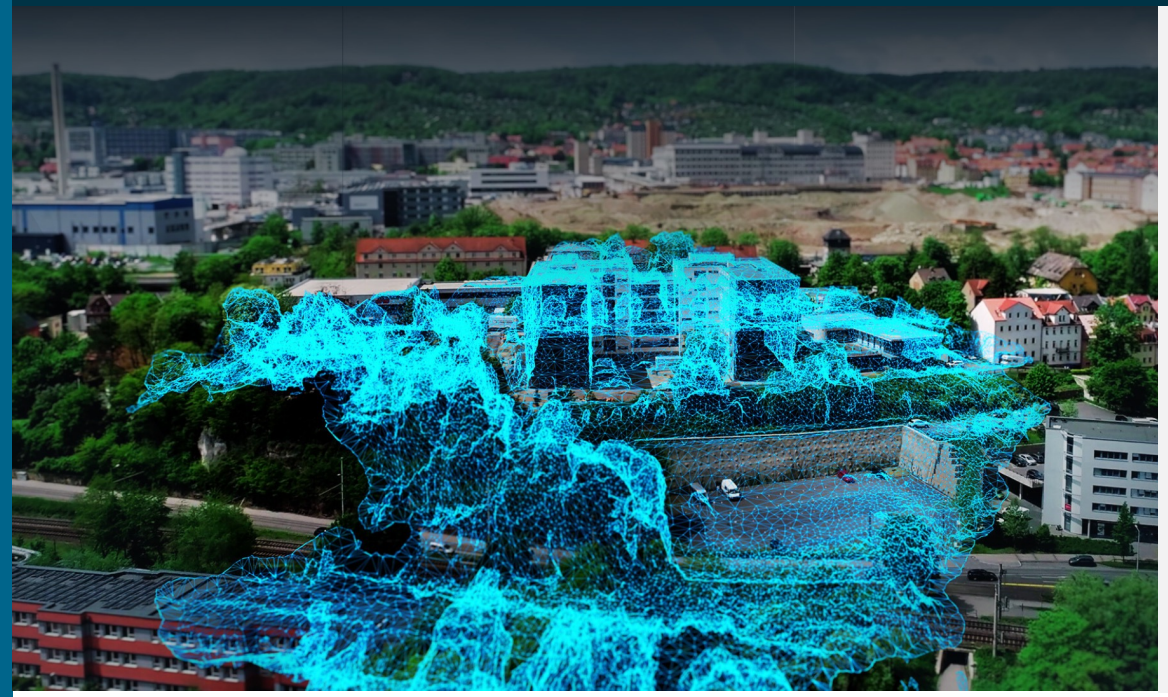
DLR Institute of Data Science

Our vision: From first bit to data-driven decision

Location Jena

80 Employees

Since 2017



- Research and development of innovative methods for an efficient use of complex data
- Interdisciplinary research along the entire data life cycle
- Application & transfer in the fields: aeronautics, space, energy, transport, security and digitalisation
- Exploit the potential of data
  - Knowledge from data
  - Robust decision support
  - Expand the possibilities of data
- HPDA-Cluster for handling large data volumnes and AI

CLOUD-NATIVE DATA MANAGEMENT FOR MULTI-MODEL ANALYTICAL WORKLOADS

A TECHNOLOGY REVIEW FOR ARCHITECTURE SELECTION

Matthias Pohl, Arne Osterthun

EUDAT 2026, Prague, Czech Republic

Motivation

- The cloud-native promise
  - Disposable replicas, replaced on failure
  - Horizontal scaling by adding instances
  - Each service owns its data
  - Declarative, automated operations
  - 82% of container users run Kubernetes in production
- The data reality
  - State must live somewhere durable
  - Invariants span services: consistency must be coordinated
  - Backups, failover and upgrades remain Day-2 work
  - CAP / PACELC still bound replicated stores
  - Licenses and tool governance change

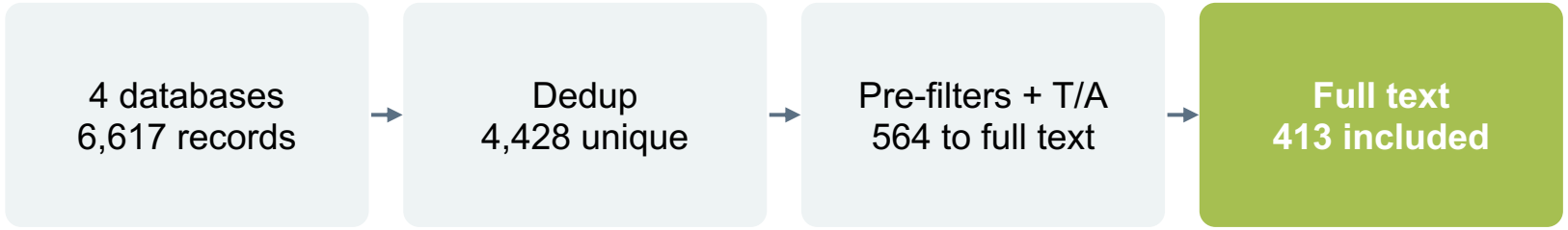
- Sources: (CNCF, 2026a; CNCF TAG Storage & DoKC, 2023; Brewer, 2012; Abadi, 2012; Kleppmann, 2017)

Research Question

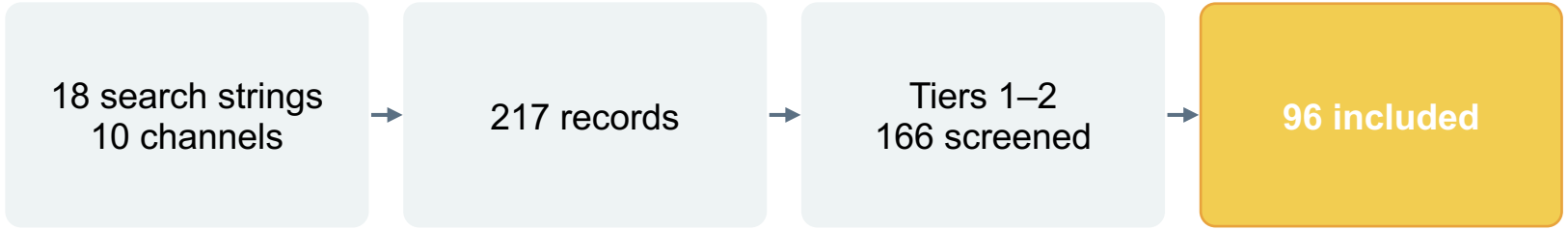
- **RQ1:** Which architectures and patterns manage data in cloud-native systems?
- **RQ2:** Which tools and platforms realize them?
- **RQ3:** Which characteristics (scalability, consistency, availability, portability, security, cost) do they address?
- **RQ4:** Which trade-offs and decision criteria are reported?

Methodology
Structured Literature Review

- SLR (Kitchenham & Charters, 2007)
- Academic Literature:



- Grey Literature:



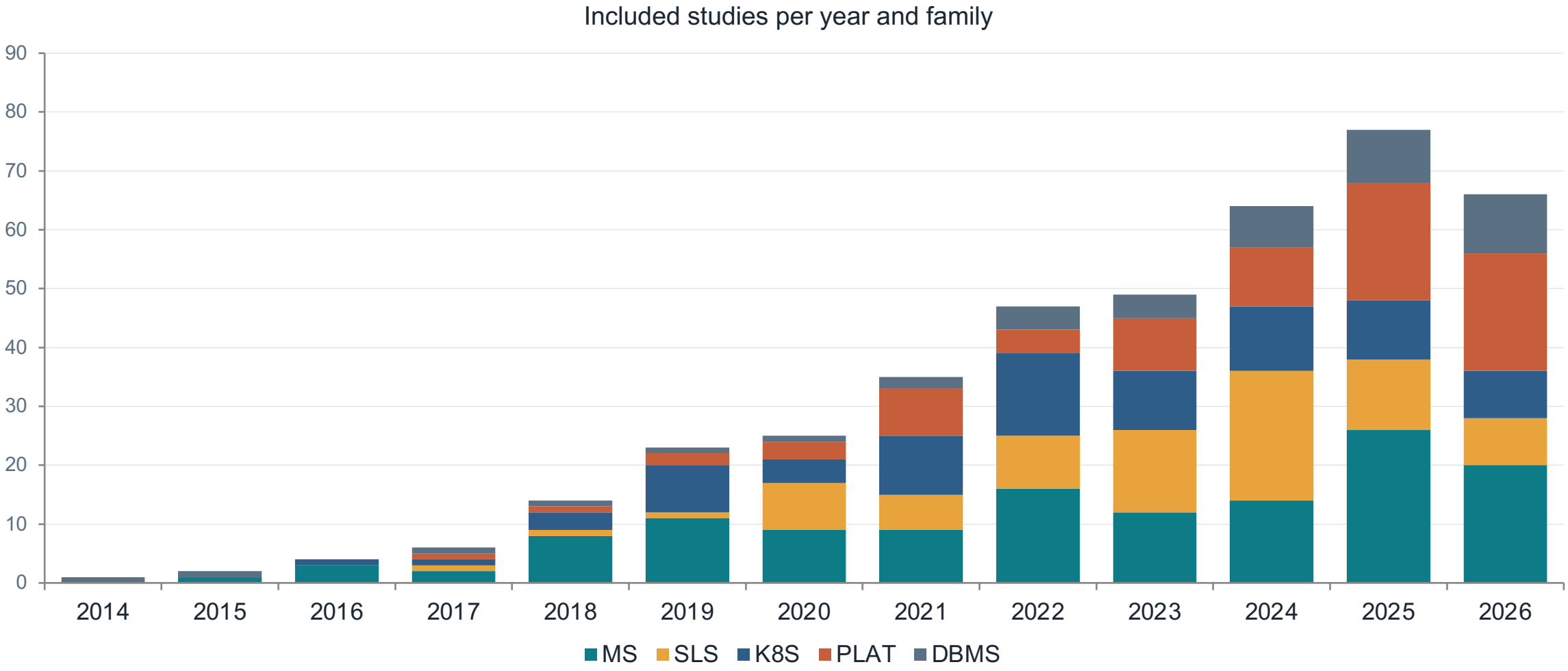
- Thematic synthesis
  - Codes per study: family (MS, SLS, K8S, PLAT, DBMS), contribution type, evidence type, RQs. Grouped per family, compared across families and with the grey lane.

Methodology
Search String & Criteria

- Cloud-native context
  - cloud native, microservice*, Kubernetes, containerized, serverless, FaaS, service mesh
- Data / state
  - data management, database*, stateful, state management, data lake*, lakehouse*, data mesh, event sourcing, CQRS, CDC, object storage
- Design / decision
  - architect*, pattern*, design decision*, trade-off*, taxonomy, survey, evaluation, selection

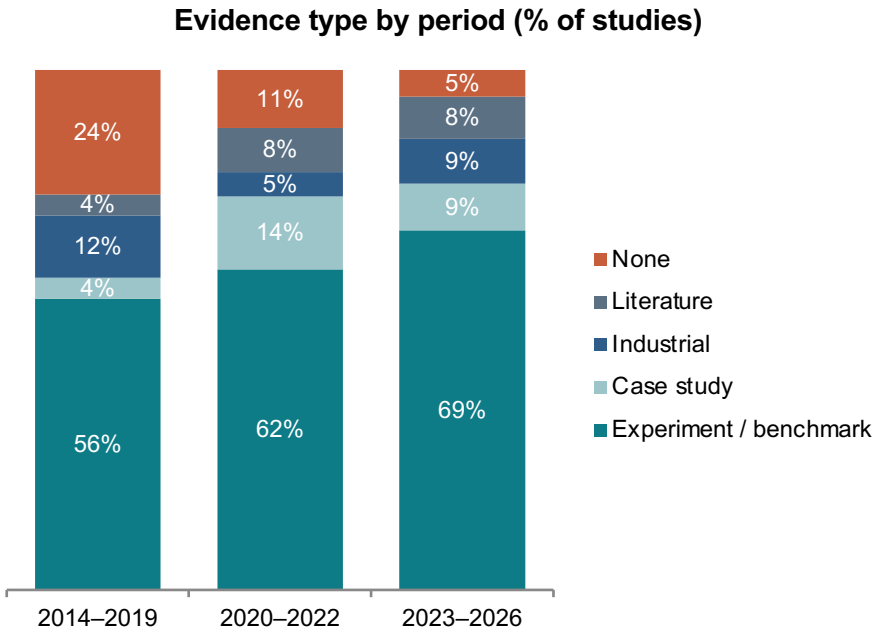
ID	Criterion
I1	Data management in cloud-native, container, microservice or serverless systems
I2	Proposes, describes, compares or evaluates an architecture, pattern or tool
I3	Peer-reviewed, or grey tier 1–2
E1	Only traditional VM / on-premises
E2	Narrow tuning or algorithm internals
E3	No full text, < 4 pages, not English
E4	Duplicate or superseded version
E5	Editorial, marketing, thin content

Results
Publications



Results
Publications – Type of Validation

- Mostly self-run, short, unreplicated testbeds
- Production evidence:
  - hyperscaler product papers
- Benchmarks emerging:
  - CloudyBench, Online Marketplace, Bexhoma
- Grey: ~60% documentation or opinion



- Sources: (Zhang et al., 2025b; van Renen & Leis, 2023; Laigner et al., 2025; Erdelt, 2024; Delnat et al., 2018; Grambow et al., 2019)

Results
Architectures and Patterns (RQ1)

- Microservices

- Ownership
  - Database-per-service is the reference point
  - Data-driven decomposition: ~25 studies derive boundaries from schemas, logs, stored procedures, and LLMs
  - Industry favors “data-first” migration
- Consistency
  - Saga dominates (37 studies): frameworks, isolation, verification
  - Alternatives: transparent ACID layers, coordination services, weak referential integrity
  - BAC theorem: no consistent and available whole-system backup
- Propagation
  - CQRS, event sourcing and CDC form the propagation layer
  - Debezium / Kafka in industrial case studies
  - Kafka pattern catalogue for streaming

- Sources: (Munonye & Martinek, 2020; Meric & Isakovic, 2026; Ma et al., 2022; Romani et al., 2026; Limon et al., 2018; Daraghmi et al., 2022; Stefanko et al., 2019; Nikolic & Dimitrieski, 2021; Zhang et al., 2019; Pardon et al., 2018; Bashtovyi & Fechan, 2023; Smid et al., 2019)

Results
Architectures and Patterns (RQ1)

- Stateful Kubernetes
  - Containerized DB persistence vs. managed services
  - Controllers for stateful availability, C/R operators
  - Storage-SLO scheduling, volume autoscaling
  - Stateful migration: ~20 studies (CRIU, MS2M, edge)
- Serverless state
  - Shared logs: Boki, IndiLog, FlexLog, DoubleTrack
  - Consistent caches co-located with functions
  - Actor/object models: Histrio, Nubes, OaaS, Netherite
  - Ephemeral storage, transactions, CRDT edge state

- Sources: (Li et al., 2024; Vayghan et al., 2021; Schmidt et al., 2023; Wen et al., 2023; Dinh-Tuan & Beierle, 2022; Calagna et al., 2025; Jia & Witchel, 2024; Wiesholler et al., 2024; Giantsidi et al., 2023; Wu et al., 2020; Sreekanti et al., 2020; De Martini et al., 2026; Burckhardt et al., 2025)

Results
Architectures and Patterns (RQ1)

- Data platforms
  - Lakehouse on object storage: BigLake, Ursa, Bauplan
  - Data mesh: reference architecture, policy-as-code governance
  - Stateful streaming: Samza, Kafka Streams
- Cloud-native DBMS
  - Disaggregated compute, log and storage
  - Products: Redshift, Aurora Serverless, DynamoDB, Firestore, PolarDB
  - Surveys and an experimental design study

- Sources: (Levandoski et al., 2024; Merli et al., 2025; Tagliabue et al., 2024b; van der Werf et al., 2025; Brambilla & Plebani, 2025; Noghabi et al., 2017; Dong et al., 2024; Pang & Wang, 2024; Armenatzoglou et al., 2022; Barnhart et al., 2024; Elhemali et al., 2022; Kesavan et al., 2023; Li, 2018)

Results
Tool Landscape (RQ2)

Tool group	Academic coverage (studies)	Grey coverage
Orchestration primitives	StatefulSet (17), operator (22)	Core (DoK patterns WP)
Kubernetes storage	Ceph (15), MinIO (15), Longhorn (3), OpenEBS (2)	Docs & licence/governance status
Operator-managed DBs	**Vitess (1), CloudNativePG (0)**	Docs, Slack & Zalando reports
Distributed SQL	CockroachDB (11), Spanner (14), TiDB (4)	Docs, licence changes
Streaming & CDC	Kafka (69), Flink (34), Debezium (6)	Outbox / CDC guidance
Backup / DR	**Velero (1)**	Docs, CNCF DR whitepaper
Stateful serverless runtimes	Orleans (11), Durable Functions (8)	Thin

- Most-used academic building blocks: Kubernetes (140), Docker (114), S3 (84), Redis (76), Lambda (73), Kafka (69)
- Sources: (CNCF TAG Storage & DoKC, 2023; ThoughtWorks, n.d.; Kubernetes, n.d.; Cockroach Labs, 2024; MinIO, 2026; Velero, 2026; CNCF, 2025; Truyen et al., 2018; Redžepagić et al., 2026; Mercl & Pavlik, 2019)

Results
Characteristics (RQ3)

Quality attribute	MS	SLS	K8S	PLAT	DBMS
Availability	73%	**78%**	**86%**	63%	**90%**
Scalability	65%	71%	44%	**78%**	**93%**
Performance	40%	**91%**	61%	63%	**88%**
Consistency	66%	59%	49%	49%	**76%**
Cost	27%	**79%**	32%	71%	**86%**
Fault tolerance	40%	39%	46%	42%	62%
Elasticity	10%	44%	18%	21%	**76%**
Security	18%	22%	31%	55%	38%
Portability	4%	10%	9%	13%	10%

Results
Decision Support (RQ4)

- 60 of 413 studies give explicit decision support

- Decision models & guidance
  - ADD model from 35 practitioner sources
  - DSL-based pattern selection
  - NoSQL & platform selection frameworks
  - Cost-based infrastructure design (BRAD)
- Empirical trade-offs
  - DB choice matters more than monolith vs microservices
  - Managed vs self-hosted DBs
  - EKS vs ECS; CockroachDB vs Patroni
  - Procedural vs transactional statelessness
- Limits & heuristics
  - BAC theorem; “new CAP” of streaming cost
  - Grey: fits one node? → RDBMS
  - Grey: inside vs outside Kubernetes
  - Grey: StatefulSet vs Dapr vs Durable Functions

- Sources: (Ntentos et al., 2019; Volynsky et al., 2022; Abbasi et al., 2026; Laigner et al., 2026; Yu et al., 2024; Meric & Isakovic, 2026; Kotakonda, 2026; Jerrari & Assayad, 2026a; Pardon et al., 2018; Merli et al., 2025; ThoughtWorks, 2018; Microsoft, n.d.-b)

Taxonomy
State Locality to Operator

	O1 Application	O2 Runtime / platform	O3 Data system
L1 Ephemeral	**Data-in-memory, caches** cache-aside in services	**Consistent caches** HydroCache, CausalMesh, Cloudburst	**Ephemeral storage** InfiniStore, Glider, HyDuo
L2 In-cluster	**DB in container image** app-managed rollout	**Operators, StatefulSets** C/R, migration, PV autoscaling	**Distributed DB in K8s** CockroachDB, Cassandra, Ceph · thin
L3 Data service	**DB-per-service + saga** CQRS projections, outbox	**Runtime state APIs** Dapr, ACID proxy layers · thin	**Managed cloud-native DB** Aurora, DynamoDB, Spanner
L4 Shared log	**Event sourcing, CDC** Debezium / Kafka pipelines	**Shared-log runtimes** Boki, FlexLog; Samza, StateFun	**Lakehouse, log-as-DB** Iceberg/Delta, BigLake, PolarDB

Typical Configurations

- Service-owned transactional state (L3 & O1)
  - Default for business microservices with local invariants
  - Cost: compensation logic and eventual consistency
- Platform-guaranteed state (L2/L3 & O2/O3)
  - Operator, managed DBMS, or stateful serverless runtime owns recovery
  - When Day-2 effort or global invariants dominate
- Log-centered state (L4)
  - Log or lakehouse is the system of record; services materialize views
  - When many consumers and auditability dominate

- Sources: (Richardson, n.d.-a; AWS, n.d.; CNCF TAG Storage & DoKC, 2023; Burckhardt et al., 2025; Levandoski et al., 2024)

Decision Framework

D1

Workload type

Operational or analytical?

Query pattern, volume, freshness

D2

Consistency scope

Which invariants, how strongly?

Invariants, compensation tolerance, isolation

D3

State locality

How close to compute?

Latency, state size, edge placement, recovery

D4

Guarantee owner

App, operator or provider?

Skills, Day-2 effort, PITR, cost model

D5

Propagation

How do changes flow?

Ordering, delivery semantics, schema evolution

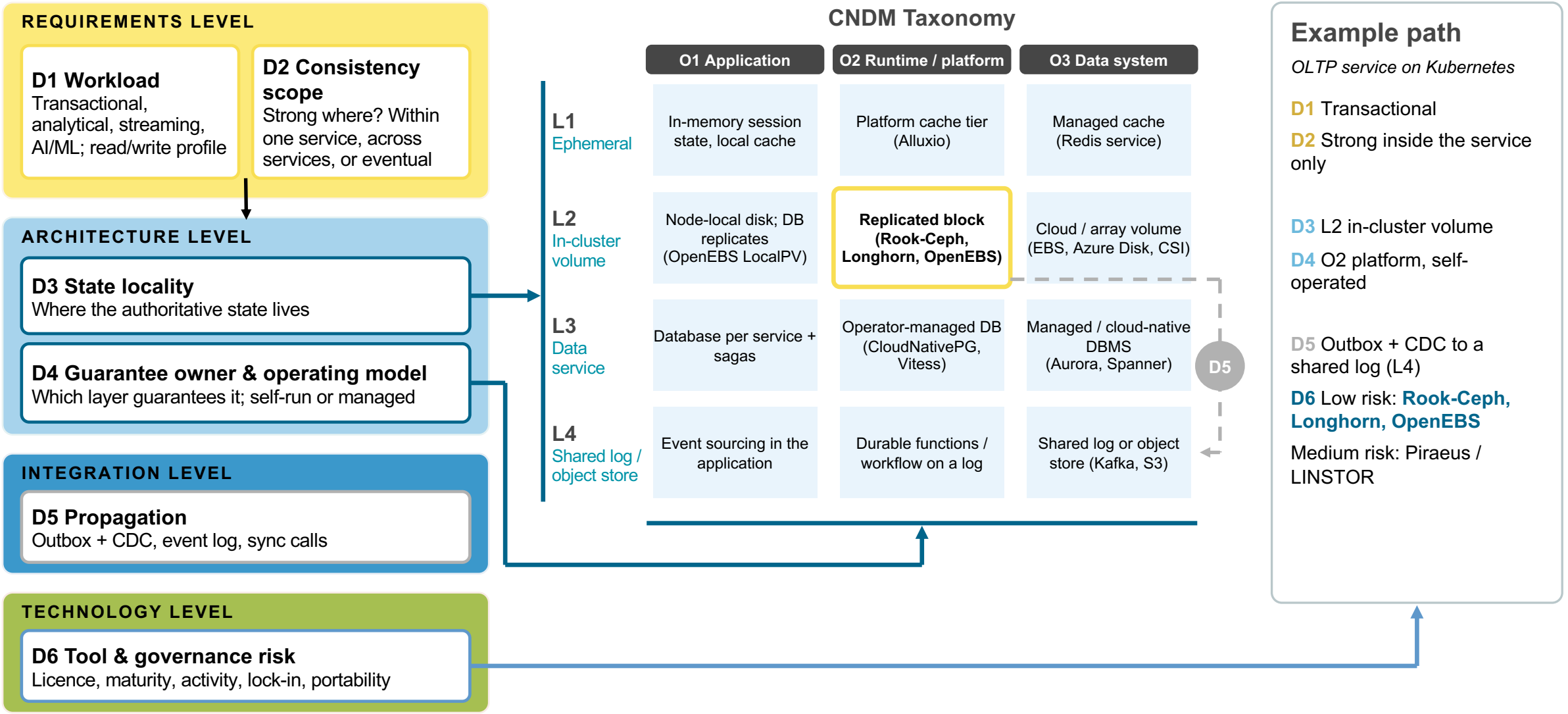
D6

Tool & governance

Mature, portable, stable licence?

CNCF maturity, licence, managed fallback

Decision Framework



Example path

OLTP service on Kubernetes

D1 Transactional

D2 Strong inside the service only

D3 L2 in-cluster volume

D4 O2 platform, self-operated

D5 Outbox + CDC to a shared log (L4)

D6 Low risk: Rook-Ceph, Longhorn, OpenEBS

Medium risk: Piraeus / LINSTOR

Imprint

Topic:	**Cloud-Native Data Management for Multi-Model Analytical Workloads - A Technology Review for Architecture Selection**
Date:	2026-11-26 (YYYY-MM-DD)
Author:	Matthias Pohl
Institute:	DW-DMA
Image sources:	All images “DLR (CC BY-NC-ND 3.0)” unless otherwise stated