

ON-PREM TO CLOUD WITHOUT BREAKING THE BUSINESS: A DATA ENGINEERING BLUEPRINT FOR ENTERPRISE GCP MIGRATION

Santosh Kumar Maddali

Sr. Data Engineer

ABSTRACT

Large enterprises rarely abandon cloud migrations because the target technology fails; they abandon them because the business breaks along the way: reporting deadlines are missed, regulatory extracts diverge from certified numbers, and downstream consumers lose trust in the data. This paper presents a data engineering blueprint for migrating enterprise-scale on-premises data platforms to Google Cloud Platform (GCP) while treating business continuity as a first-class engineering requirement rather than a project-management afterthought. The blueprint combines four mechanisms: (1) a disposition-driven workload assessment that separates rehosting, replatforming, and refactoring paths; (2) a dual-run architecture in which legacy and cloud pipelines execute in parallel against synchronized inputs; (3) a continuous, per-run reconciliation framework with automated discrepancy alerting and loop-back reprocessing; and (4) a consumer-by-consumer strangler-pattern cutover with codified rollback runbooks. The blueprint is developed against a reference estate profile typical of large enterprises: a commercial warehouse tier, a Hadoop-era data lake, and several hundred interdependent scheduled pipelines, targeting BigQuery, Dataflow, Pub/Sub, Cloud Composer, and Looker. Rather than reporting results from a single program, the paper specifies an outcome model that states which effects the blueprint is designed to produce, the acceptance criteria that govern each migration phase, and the measurement protocol by which practitioners can evaluate them in their own estates. The paper closes with a discussion of dual-run economics, organizational preconditions for success, and the limitations of the approach.

Keywords:

Cloud Migration; Google Cloud Platform; BigQuery; Data Engineering; Data Reconciliation; Business Continuity

I. INTRODUCTION

Cloud computing has shifted from a cost-optimization tactic to the default operating model for enterprise analytics, offering elastic capacity, consumption-based pricing, and access to managed services that would be prohibitively expensive to build in-house [1], [2]. Yet for large organizations, the journey from an entrenched on-premises estate, typically a proprietary enterprise data warehouse (EDW) surrounded by a Hadoop-era data lake and hundreds or thousands of extract-transform-load (ETL) jobs, to a cloud-native platform remains one of the highest-risk programs a technology organization can undertake. A systematic review of cloud migration research identifies legacy interdependence, data gravity, and operational risk as dominant inhibitors, and notes that technical migration guidance far outweighs guidance on preserving business operations during the transition [3].

The central argument of this paper is that the hard problem of enterprise data platform migration is not moving data; it is moving data while the business continues to run on it. Financial closes, regulatory submissions, executive dashboards, and machine learning features all consume the platform daily. A migration approach that requires a "big-bang" cutover, a prolonged freeze on change, or a period of degraded data quality is, in practice, a plan to break the business. Accordingly, this paper frames continuity as an engineering requirement with measurable acceptance criteria (parity thresholds, SLA adherence, and rollback time objectives) rather than as a soft program-management concern.

The contributions of this paper are as follows:

- 1) A reference blueprint for migrating heterogeneous on-premises data estates (EDW, Hadoop, and change-data-capture-enabled operational systems) to a GCP-native architecture centered on BigQuery, Dataflow, Pub/Sub, Cloud Storage, Cloud Composer, and Looker.
- 2) A dual-run and reconciliation methodology, structured as a layered control hierarchy executed on every pipeline run with automated discrepancy alerting and loop-back reprocessing, that converts "the numbers match" from an assertion into a continuously verified property of the migration, with automated remediation for recoverable discrepancies.
- 3) A strangler-pattern cutover model that sequences consumer migration by risk tier, with explicit rollback runbooks and freeze-window governance.

- 4) An outcome and evaluation model that specifies the effects the blueprint is designed to produce and the measurement protocol for assessing them, together with a candid discussion of the costs and limits of the approach.

The remainder of the paper is organized as follows. Section II reviews related work. Section III presents the migration blueprint as the methodology. Section IV presents the outcome and evaluation model for the blueprint. Section V discusses implications and limitations, and Section VI concludes.

II. LITERATURE REVIEW

Foundational characterizations of cloud computing established the essential service and deployment models [2] and articulated the economic argument for elasticity and statistical multiplexing of demand [1]. Jamshidi et al. [3] systematized early cloud migration research and observed that most published approaches address application migration, with comparatively little treatment of stateful, interdependent data platforms, precisely the estate class this paper targets.

On the target-platform side, BigQuery descends from Dremel, whose columnar storage and multi-level serving trees demonstrated interactive analysis over web-scale datasets [4]; its modern managed form decouples storage from compute and exposes capacity through on-demand and reservation-based models [5]. The Dataflow model unified batch and streaming computation around event-time windowing and explicit correctness/latency/cost trade-offs [6], superseding the earlier generation of MapReduce [7] and resilient distributed dataset frameworks [8] that dominate legacy Hadoop estates. Kleppmann's synthesis of data-intensive system design [9] and Kimball's dimensional modeling canon [10] remain the touchstones for, respectively, the reliability semantics and the semantic-layer design that any migration must preserve.

For transition mechanics, Fowler's strangler fig pattern, which incrementally routes consumers to a new system while the legacy system continues to operate, originated in application modernization but generalizes naturally to data platforms [11]. Empirical software delivery research further shows that deployment frequency and mean time to restore, rather than change freezes, predict operational stability [12], motivating this paper's preference for many small, reversible cutovers over a single large one. Finally, the data quality literature provides the assessment dimensions (accuracy, completeness, consistency, timeliness) that a reconciliation framework must operationalize [13], while Marz and Warren's dual-path architectural thinking [14] prefigures the dual-run approach used here, albeit for a different purpose: this blueprint runs two complete platforms in parallel not to serve queries, but to prove equivalence before cutover.

The gap this paper addresses is the absence of an integrated, engineering-grade treatment that connects workload disposition, parallel operation, quantitative reconciliation, and consumer-sequenced cutover into a single blueprint for GCP-bound enterprise migrations.

III. METHODOLOGY

A. Guiding Principles

The blueprint rests on three principles. First, continuity is an acceptance criterion: every migration phase defines, in advance, the parity thresholds and SLA commitments that must hold for the phase to be declared complete. Second, every step is reversible until proven: no consumer is cut over without a tested rollback path, and no legacy asset is decommissioned until its successor has sustained parity through at least one full business cycle (including month-end and quarter-end peaks). Third, equivalence is computed, not asserted: reconciliation is implemented as automated pipelines that run continuously, publish results to dashboards, and gate cutover approvals.

B. Phase 0: Discovery and Workload Disposition

Migration begins with an automated inventory of the estate: table-level lineage harvested from ETL metadata and query logs, consumption mapping from BI-tool and database audit logs, and criticality scoring based on downstream regulatory or executive dependence. Google's migration framework of assess, plan, deploy, and optimize provides the outer loop for this phase [15]. Each workload is then assigned a disposition (Table 1). In mature estates a substantial share of tables and jobs is typically retired outright once consumption evidence shows they are unused; practitioners should size this share from their own consumption telemetry rather than from a rule of thumb, since it varies widely with estate age and governance maturity. The cheapest workload to migrate is the one that is deleted.

Disposition	Description	Typical Data Workloads	GCP Target
Retire	Decommission redundant or unused assets identified during discovery	Orphaned tables, obsolete reports, duplicated feeds	None (eliminated)
Retain	Keep on-premises temporarily due to latency, licensing, or regulatory constraints	Mainframe adjacency, jurisdiction-bound datasets	Hybrid connectivity (Interconnect)
Rehost	Lift-and-shift with minimal change to reduce migration risk and time	Hadoop/Spark jobs, self-managed databases	Dataproc, Compute Engine
Replatform	Move to a managed equivalent with modest modification; heterogeneous engines require schema and dialect conversion first	Hive tables; PostgreSQL and SQL Server schemas; Oracle schemas after conversion to PostgreSQL	Dataproc + GCS; Cloud SQL
Refactor	Re-architect to cloud-native services for elasticity and operational efficiency	EDW schemas, ETL pipelines, streaming feeds	BigQuery, Dataflow, Pub/Sub

Table 1. Workload disposition framework and representative GCP targets.**C. Target Architecture**

Figure 1 depicts the reference architecture. Historical bulk data moves through parallelized batch transfer (Storage Transfer Service or, for constrained network paths, Transfer Appliance) into a Cloud Storage landing zone organized into raw, staged, and curated layers. Ongoing changes flow through streaming change data capture (log-based CDC published to Pub/Sub), which keeps the cloud copy synchronized with operational sources for the entire dual-run period. Cloud Functions provides event-driven glue: object-finalization triggers on the landing zone launch validation and downstream processing without polling. Dataflow implements transformation and enrichment under the unified batch/streaming model [6], writing to BigQuery, which hosts the replatformed physical schemas alongside curated dimensional and analytical models that preserve Kimball-conformant dimensions [10]; relational schemas that remain operational in character are replatformed to Cloud SQL rather than the warehouse. Looker provides the governed semantic-model and dashboard layer through which reporting consumers are cut over. Cloud Composer orchestrates dependencies and enforces the same logical DAG contracts that governed the legacy scheduler, easing side-by-side comparison. A managed data catalog, together with Cloud IAM and VPC Service Controls, provides cataloging and lineage, access governance, and data-exfiltration perimeters, respectively.

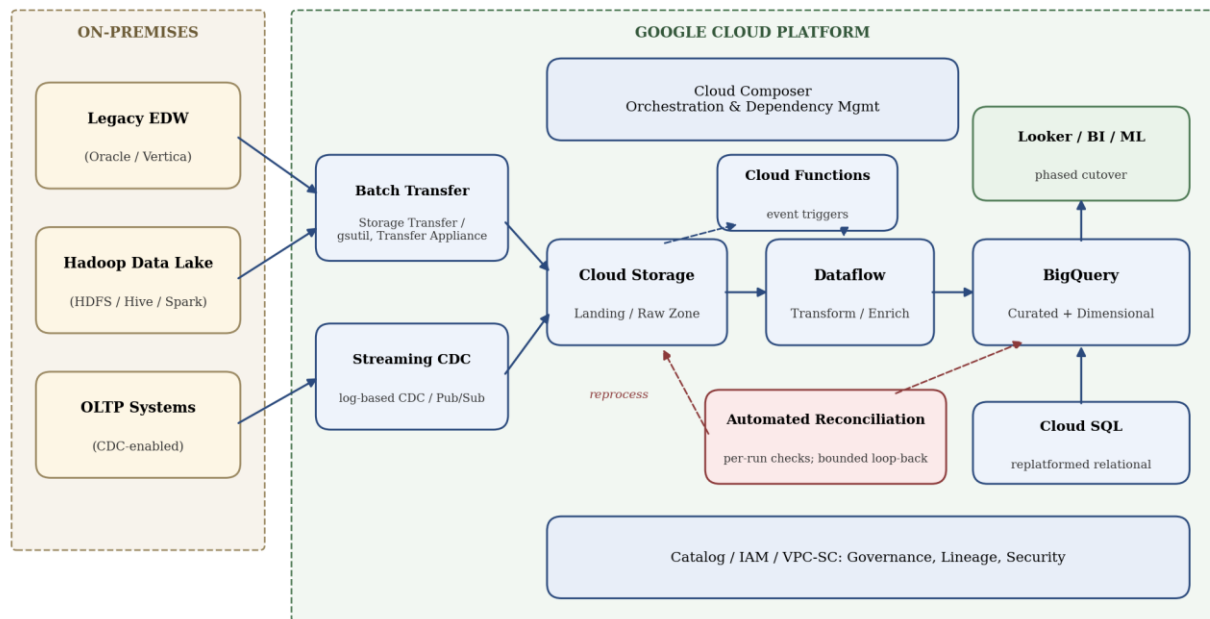


Figure 1. Reference architecture for dual-run enterprise migration to GCP.

D. Dual-Run Operation and the Reconciliation Control Hierarchy

The core continuity mechanism is dual-run: for each migration wave, legacy and cloud pipelines execute in parallel against synchronized inputs, and their outputs are compared automatically. Reconciliation is organized as a five-level control hierarchy (Table 2) and executed on every pipeline run, so that equivalence is a standing property of the platform rather than a periodic audit finding: exhaustive controls (row counts, aggregates) run on every load, while the most expensive controls (cell-level hash sampling, KPI dual-computation) run on risk-weighted schedules within each wave. Controls are implemented as declarative specifications compiled into Dataflow and BigQuery jobs, so that adding a table to reconciliation is a configuration change, not a development task. Any discrepancy raises an automated alert, and a loop-back mechanism reprocesses the affected partition from the landing zone without operator intervention. Reprocessing must be idempotent, so that a replayed partition fully replaces rather than augments prior output, and it must be bounded: after a program-defined maximum number of automated attempts, the partition is quarantined and escalated for engineering triage. This bound matters because automated replay can only clear recoverable faults such as transient execution or ingestion failures; a deterministic transformation defect will reproduce identically on every attempt and would otherwise drive an unbounded and expensive retry loop. Escalated exceptions are triaged into one of three classes, namely a defect in the new pipeline, a latent defect in the legacy pipeline, or an explainable representational difference (encoding, collation, floating-point precision), and tracked to closure. This classification matters: discoveries of latent legacy defects must be adjudicated with data owners rather than silently "fixed," because certified historical numbers may depend on them. The hierarchy operationalizes the accuracy, completeness, and consistency dimensions of the data quality literature [13] and echoes, in transitional form, the parallel-path reasoning of dual-architecture designs [14].

Level	Control	Method	Tolerance
L1	Row-count parity	Exact record counts per table/partition, legacy vs. target	Zero variance
L2	Aggregate parity	SUM/MIN/MAX/AVG on configured numeric measures; checksum on keys	Program-defined tolerance, e.g., within 0.001% where floating-point representation requires it
L3	Distributional parity	Null ratios, distinct cardinality, top-N frequency per column	Configurable threshold (e.g., within 0.01%)
L4	Cell-level sampling	Deterministic hash-based row sampling; full-row hash over a canonical representation (see Section III-D)	Zero variance on sample
L5	Business KPI parity	Dual-computation of certified executive metrics from both stacks	Zero variance

Table 2. Five-level reconciliation control hierarchy executed during dual-run.

Two definitions must be made explicit for reconciliation to be meaningful. The first is what constitutes synchronized inputs. Legacy and cloud pipelines can only be compared if they consume equivalent input populations, so each comparison is declared against a bounded window whose edges are defined rather than implied: bulk snapshots are pinned to a source-side consistency point, CDC streams are cut at a recorded log position or watermark, and event-time rather than arrival-time is used to assign records to the window. A window is eligible for comparison only once its late-arrival grace period has elapsed on both platforms. Records arriving after the grace period remain associated with their original event-time window and trigger a controlled replay or reopening of that window on both platforms before parity is recertified. Without this discipline, ordinary timing differences present themselves as row-count defects and reconciliation generates noise rather than signal. The second definition is the canonical representation used for hashing at L4. Two logically identical rows will hash differently across engines unless representation is normalized first, so the specification fixes null encoding, timestamp precision and time-zone offset, decimal scale and rounding, floating-point formatting, Unicode normalization form, string collation and trailing whitespace, character encoding, and column ordering. Values are serialized to that canonical form on both platforms before hashing.

E. Cutover Strategy: Consumer-by-Consumer Strangulation

Rather than cutting over data assets, the blueprint cuts over consumers, applying the strangler fig pattern [11] at the level of dashboards, extracts, and downstream applications. Consumers are tiered by blast radius: Tier 3 (internal, non-certified analytics) migrates first, providing production burn-in at low stakes; Tier 2 (operational reporting) follows; Tier 1 (regulatory, financial-close, and executive-certified outputs) migrates last, only after sustaining L5 KPI parity through a complete business cycle. Each cutover is executed against a runbook specifying the routing change, the validation checks, the decision authority, and a rehearsed rollback with an explicit, program-defined time objective (short in practice, since the legacy path remains warm throughout dual-run). Freeze windows are minimized deliberately: consistent with empirical findings that restore capability, not change scarcity, drives stability [12], a migration program following this blueprint should optimize for small reversible steps rather than long stabilization freezes.

F. Performance and Cost Engineering

Continuity also has a performance dimension: a migration that preserves correctness but doubles query latency still breaks the business. Replatformed schemas are therefore re-engineered for BigQuery's execution model [4], [5]: date-partitioned and clustered physical tables replace index-centric designs; incremental MERGE patterns replace full-refresh loads; and workload management uses reservations to isolate certified reporting from ad hoc exploration. Cost governance is instituted from day one, through per-project quotas, query bytes-scanned budgets, and storage lifecycle policies on the landing zone, because dual-run temporarily pays for two platforms, and undisciplined consumption can erase the migration's financial case before cutover completes.

G. Governance, Security, and Compliance

The migration inherits, and must demonstrably preserve, the control environment of the legacy estate. Access models are translated from perimeter- and grant-based database security to IAM roles bound at dataset and table

granularity, with VPC Service Controls establishing service perimeters against exfiltration. Column-level policy tags provide fine-grained access control for regulated attributes, and lineage metadata maintained in the catalog provides the audit trail that regulators expect when certified figures change platforms. Crucially, compliance sign-off is scheduled as a workstream with its own milestones, not as a gate discovered at the end.

IV. EXPECTED OUTCOMES AND EVALUATION MODEL

This section states what the blueprint is designed to achieve and how practitioners should measure whether it has. It is framed against a reference estate profile representative of large enterprises rather than a single organization: a commercial warehouse tier serving certified financial and operational reporting, a Hadoop-era environment (HDFS and Hive, with Spark transformation code in Scala or Python), and several hundred interdependent scheduled pipelines, with the reporting layer consolidating onto BigQuery fronted by governed dashboards. The values in Table 3 are design targets and illustrative ranges rather than empirically derived estimates: they indicate the order of effect the architectural properties described in Section III are intended to produce, not magnitudes established by controlled measurement. They are offered as hypotheses to be tested against a practitioner's own baseline using the protocol below, and any organization applying the blueprint should replace them with its own measured values before using them in a business case.

Three effects are central to the blueprint's thesis. First, continuity is treated as an acceptance criterion rather than an anticipated benefit: the target for SLA misses attributable to migrated workloads is zero, and a wave that cannot meet it does not proceed to cutover. Residual misses on datasets not yet onboarded remain the responsibility of the legacy platform and should decline monotonically as onboarding progresses. Second, equivalence is maintained by construction rather than by periodic audit: validations compare cloud outputs against on-premises outputs on every pipeline run, discrepancies raise automated alerts, and a loop-back mechanism reprocesses affected data without operator intervention, so that only discrepancies surviving reprocessing are escalated for engineering triage. Third, platform availability is expected to improve materially, since managed services carry their own availability guarantees in place of self-operated infrastructure that the organization must keep running itself. This effect should not be overstated: a service availability guarantee reduces infrastructure failure risk, but it does not address transformation defects, upstream data quality, orchestration errors, schema drift, or incorrect business logic, which remain the responsibility of the engineering controls described in Section III-D. Table 3 therefore separates platform availability from end-to-end pipeline and data incidents.

Metric	Baseline	Target State	Anticipated Effect
Critical-path batch duration	Indexed 100	SLA and workload defined	Estate-specific reduction, measured during dual-run
Reporting-layer query latency	Indexed 100	SLA and workload defined	Estate-specific reduction, measured during dual-run
SLA misses attributable to migrated workloads	Program-defined	Zero	Acceptance criterion, not an estimate
Platform availability incidents	Indexed 100	Materially lower	Managed-service availability guarantees replace self-operated infrastructure
End-to-end pipeline and data incidents	Indexed 100	Estate-dependent	Not addressed by service SLAs; depends on transformation and data-quality controls
Output validation coverage	Periodic, manual	Every run, automated	Alerting and loop-back reprocessing

Metric	Baseline	Target State	Anticipated Effect
Annualized platform run cost	Indexed 100	Parity to modest change	Return realized in reliability, not run cost

Table 3. Outcome model: design targets and illustrative target ranges (indexed).

Performance effects follow from the refactoring discipline of Section III-F, and their magnitude depends on how far legacy physical designs diverge from the target execution model: estates carrying heavy index-centric designs and full-refresh loads have the most to gain, while workloads already tuned for columnar scan-based execution will gain less. Run cost deserves particular candour in business-case framing. Practitioners should plan for run cost at or near parity rather than assuming infrastructure savings, with the return realized in reliability, elasticity, and analytical experience; the temporary dual-run premium should be budgeted explicitly as the price of continuity and retired as waves complete decommissioning.

The measurement protocol is a direct consequence of the dual-run design. Because legacy and cloud platforms execute the same pipelines, queries, and reports against synchronized inputs during parallel operation, practitioners can obtain like-for-like comparisons rather than before-and-after estimates: each metric in Table 3 should be measured on both platforms within the same dual-run window, over a workload mix spanning at least one full business cycle, and reported as an indexed ratio against the legacy baseline. Continuity metrics should be tracked continuously rather than sampled, since a single missed certified deadline is the outcome the blueprint exists to prevent.

V. DISCUSSION

The blueprint's most contestable design choice is the cost of dual-run: operating two platforms, plus a reconciliation estate, is expensive, and program sponsors will be tempted to shorten it. The premium is best understood as the price of an insurance policy: it reduces the probability and blast radius of catastrophic cutover failure and surfaces discrepancies before consumers are ever exposed to them. The economics favor dual-run precisely where the cost of being wrong is highest: certified, regulated, and executive-visible outputs. A conservative planning posture is to assume steady-state run-cost parity rather than guaranteed savings, with the return realized in reliability, elasticity, and analytical experience rather than in the infrastructure line item; the business case should be argued honestly on those terms.

The blueprint depends on three organizational preconditions. First, consumption evidence must be trusted: disposition decisions (especially retirement) collapse into negotiation unless query-log-based usage data is accepted as authoritative. Second, exception adjudication needs empowered data owners; reconciliation without a decision forum produces dashboards, not progress. Third, the semantic layer must be owned by engineering, not recreated ad hoc by each BI team, or parity at the table level will coexist with divergence at the dashboard level. Limitations. The blueprint assumes source systems can support CDC or repeatable bulk extraction; estates dominated by mainframe batch files require an additional staging tier. More significantly, the outcome model of Section IV is a set of design targets and illustrative ranges rather than an empirical evaluation: the blueprint has not been assessed here against a controlled comparison with alternative migration strategies, and effect magnitudes will vary with estate entropy, network capacity, and organizational readiness. Empirical validation across multiple estates remains open work. Finally, the blueprint is expressed in GCP-native services; the control structure (disposition, dual-run, per-run reconciliation with loop-back remediation, tiered strangulation) transfers to other hyperscalers, but the performance-engineering specifics of Section III-F do not transfer unmodified.

VI. CONCLUSION

Enterprise data platform migration succeeds or fails on continuity, and continuity can be engineered. This paper presented a blueprint that treats business continuity as a measurable property of the migration itself: workloads are dispositioned on consumption evidence; legacy and cloud platforms run in parallel against synchronized inputs; equivalence is validated on every pipeline run, with automated alerting and loop-back reprocessing; and consumers are cut over in risk-tiered waves under rehearsed rollback runbooks. For a reference estate combining a commercial warehouse tier and a Hadoop-era data lake targeting BigQuery, Dataflow, Pub/Sub, Cloud Composer, and Looker, the paper set out the acceptance criteria each phase must satisfy, the effects the design is intended to produce, and the measurement protocol by which practitioners can evaluate them. The framing of run cost deserves emphasis: organizations should plan for parity rather than savings and argue the business case on reliability and analytical experience. Future work includes empirical validation of the outcome model across multiple estates, extending the reconciliation hierarchy to machine learning feature parity, automating disposition

classification through analysis of lineage and consumption metadata, and formalizing the dual-run insurance economics as a decision model for premium sizing.

REFERENCES

- [1] M. Armbrust et al., "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, pp. 50-58, Apr. 2010.
- [2] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Gaithersburg, MD, USA, Special Publication 800-145, Sep. 2011.
- [3] P. Jamshidi, A. Ahmad, and C. Pahl, "Cloud migration research: A systematic review," IEEE Transactions on Cloud Computing, vol. 1, no. 2, pp. 142-157, Jul.-Dec. 2013.
- [4] S. Melnik et al., "Dremel: Interactive analysis of web-scale datasets," Proceedings of the VLDB Endowment, vol. 3, no. 1-2, pp. 330-339, Sep. 2010.
- [5] Google Cloud, "BigQuery documentation," Google Cloud. [Online]. Available: <https://cloud.google.com/bigquery/docs>
- [6] T. Akidau et al., "The Dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing," Proceedings of the VLDB Endowment, vol. 8, no. 12, pp. 1792-1803, Aug. 2015.
- [7] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107-113, Jan. 2008.
- [8] M. Zaharia et al., "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI), San Jose, CA, USA, 2012, pp. 15-28.
- [9] M. Kleppmann, Designing Data-Intensive Applications. Sebastopol, CA, USA: O'Reilly Media, 2017.
- [10] R. Kimball and M. Ross, The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, 3rd ed. Indianapolis, IN, USA: Wiley, 2013.
- [11] M. Fowler, "StranglerFigApplication," martinofowler.com, Jun. 2004. [Online]. Available: <https://martinfowler.com/bliki/StranglerFigApplication.html>
- [12] N. Forsgren, J. Humble, and G. Kim, Accelerate: The Science of Lean Software and DevOps. Portland, OR, USA: IT Revolution Press, 2018.
- [13] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino, "Methodologies for data quality assessment and improvement," ACM Computing Surveys, vol. 41, no. 3, pp. 1-52, Jul. 2009.
- [14] N. Marz and J. Warren, Big Data: Principles and Best Practices of Scalable Realtime Data Systems. Shelter Island, NY, USA: Manning, 2015.
- [15] Google Cloud, "Migration to Google Cloud: Getting started," Google Cloud Architecture Center. [Online]. Available: <https://cloud.google.com/architecture/migration-to-gcp-getting-started>