ESP-JETA
ESP Journal of Engineering & Technology Advancements
ISSN: 2583-2646 / Volume 6 Issue 2 April to June 2026
Paper Id: JETA-V6I2P130/ Page No: 269-277

Review Article

Load-Aware Dispatch Framework for Concurrent ETL and ML Model Execution in Enterprise Systems

Voolla Sandeep kumar
University of Southern California, Los Angeles

Received Date: 16 June 2026 Revised Date: 24 June 2026 Accepted Date: 30 June 2026

Abstract: *Enterprise systems are moving an ETL workload and machine learning (ML) model execution to the same computational substrate, but research on the treatment of dispatch control in both workload categories has been disjointed. This review examines recent literature relevant to load-aware dispatch for the concurrent execution of ETL workloads and ML models in enterprise systems. It focuses on orchestration logic, lifecycle management, resource competition, data freshness, latency protection, and operational governance. The review concludes that the recent journal literature has a lot to offer in regard to MLOps architecture, quality assurance, deployment automation, and intelligent data preparation, but does not discuss in detail the joint ETL-ML dispatch under common enterprise load. Practical experience shows repeatedly that the breakdown of coordination is possible at the border of data pipelines, model pipelines, and infrastructure schedulers. Some of the most prominent gaps include the poor cross-layer observability, the inability to support mixed I/O- and latency-sensitive workloads, the inability to support freshness-latency trade-offs in policies and the inability to treat the queue isolation, workload classification, and feedback control in a production environment. A topic-specific conceptual framework is thus created to relate data-plane dynamics, model-serving constraints, and dispatch policy choice. The field is important since the enterprise value is more and more relying on the stable coexistence of constantly updated streams of data and constantly used predictive services.*

Keywords: *Concurrent ETL-ML Execution, Enterprise Systems, Load-Aware Dispatch, Mlops, Resource Orchestration*

I. INTRODUCTION

The need for load-aware dispatch of concurrent ETL and ML workloads has emerged from structural changes in enterprise computing. Analytical pipelines and ML services are increasingly deployed on shared computational substrates rather than being operated as separate data-platform and experimentation environments. ETL workflows now coexist with batch scoring, near-real-time inference, feature refresh, validation, and retraining tasks that often execute within overlapping resource windows. In this instance, dispatch is not simply a normal job scheduling issue since the dispatcher must not only decide the order of execution, but also ensure the data, model latency, reliability, and execution safety are maintained and are operated under workload classes with vastly different bottleneck profiles. Recent research on machine learning lifecycle assurance has clarified that the production ML systems require orchestration of control of data, code, models, tests, and infrastructure but the literature has not mapped that lifecycle perspective into a conceived dispatch theory of running ETL and ML in enterprise settings concurrently [1].

The definition of workload coupling has been changed by industrial-scale ML as well. The delay in ETL in the traditional data estates primarily affecting downstream analytics or reporting. Online feature availability, model input validity, and stale predictions can be distorted in online customer-facing or decision-support services by ETL delay in current enterprise systems, as well as by distorting online feature availability and compromising model input validity. The socio-technical system with complex dependencies, changing data states and coordination challenges in their operation, which extend way beyond the design of the algorithm, is what Lwakatare et al. call industrial ML [2]. The point above is what the current topic is about because the overload of one particular queue is unlikely to be the cause of failures of dispatch in the mixed ETL-ML environments. More often than not failure can take the form of a snowball: an ETL burst runaway pushes storage bandwidth, model-serving latency grows, fallback logic is invoked, retraining windows creep into business times of high activity. The load-conscious dispatch that must be checked must therefore be a research problem of first-order, rather than an implementation problem, which is of peripheral interest.

The relevance of the title topic can also be supported by evidence (based on cases) of the ML implementation. Paleyes et al. show that the deployment issues tend to have discrepancies between the assumptions in the laboratory and the realities of production especially non-stationary data, breakability of interfaces and monitoring of the operations [3]. These discrepancies in enterprise systems operating ETL and ML at the same time become dispatch-relevant. The dispatch policy establishes the priority according to which data preparation functions are carried out, inference services are preempted, how the backlog is throttled and when the throttling of the corrective service or admission control is activated. Much of the

existing literature continues to consider deployment a model-serving problem or a CI/CD problem, whereas enterprise operators have to grapple with a more general problem: model execution can be only as stable as data movement and data transformation ecosystem. The title topic is thus at the crossroads between data engineering, software engineering, workflow control, and operation on the cloud-native level.

The importance of the topic has increased as the ML delivery is moving towards the patterns of continuous integration, continuous delivery, and continuous deployment. Shahin et al. state that the code changes, model artifacts, and data dependencies, validation phases and rollback constraints are more complicated to implement in ML-specific delivery pipelines compared to the traditional software pipelines [4]. The short-term consequences of the discovery are on the parallel ETL-ML run. Enterprise system dispatch logic cannot be based on fixed priority ladders in the training, scoring, feature refresh, and transformation job offerings that all require shared CPU, memory, network, and storage resources. A dispatch model which does not pay attention to workload semantics can maintain the nominal cluster usage at the expense of business performance. As a comparison, a load-conscious framework needs to categorize tasks in terms of urgency, sensitivity to dependencies, cost of staleness, risk to service level, and recoverability. This type of structure would involve both control plane and data plane observability as well.

The architectural literature on MLOps can provide an initial point of view but not a complete solution. Kreuzberger et al. sum up the recent work and introduce the structured representation of MLOps with the focus on automation, reproducibility, monitoring, and lifecycle orchestration [5]. That literature dictates the need to have coordinated pipelines, but little journal treatment of dispatch policies to run both ETL and ML concurrently appears in the literature. The gaps in research in at least five directions are therefore: cross-layer workload sensing, policy mechanisms to tradeoff between freshness and latency, queue isolation to disallow mixed I/O and compute contention, to offer feedback loops between runtime telemetry and dispatch choice, and to offer enterprise governance structures to enable controlled degradation during overload. Accordingly, this review critically examines literature published between 2020 and 2025 on load-aware dispatch, MLOps, data-pipeline orchestration, automated data preparation, and production ML governance. It identifies what the available literature explains, where it remains insufficient for concurrent ETL-ML execution, and how reported findings can be interpreted for enterprise dispatch design. The following sections discuss the evidence base and streamline methodological approaches, synthesize reported results, and determine contradictions and limitations, and give plausible future research directions.

II. LITERATURE REVIEW

The new literature in the journals that can be used in a load-sensitive dispatch framework of concurrent ETL and ML model development within enterprise systems may be split into three overlapping clusters. The former is about production ML software engineering and governance and includes lifecycle assurance, testing, deployment practice and quality attributes. The second cluster concerns the data preparation, automated pipeline building and model-search mechanisms which indirectly determine the dispatch complexity, through the variation of the scale and variability of the execution graph. The third cluster deals with orchestration and infrastructure management, such as containerized implementation and process models to run ML in production. Empirical research on a dispatcher which explicitly co-optimizes ETL and ML under enterprise load is uncommon, and thus the review needs to read between the lines. That poverty itself is a significant finding of the literature as opposed to an incidental constraint on the current review.

One pattern seen early in the literature is that operational fragility is a common occurrence at crossings. Giray models ML systems as systems that are software-intensive, having unique lifecycle and maintenance overheads, particularly with regards to data reliance and environmental volatility [6]. The results of Braike and Khomh demonstrate that the quality of the ML programs is exceptionally sensitive to the data perturbation, oracle construction, and secret failure modes [7]. Serban et al. show that best practices of software engineering in ML are not fully embraced and shared in the industry, and this leads to redundant software versioning, testing, reproducibility, and the lack of discipline in deployment [8]. Martinez-Fernandez et al. further the latter when they say that AI-based systems are quality-critical socio-technical assemblies not only possess problems associated with architecture, data, operations, and governance but also engineering. In the common implication in the case of the title topic, the reduction of dispatch policy to an infrastructure level queue ordering is impossible since the interactions between data state and model state and platform state generate operational risk.

A second trend is relating to the function of upstream data work. Bilalli et al. demonstrate that the process of intelligent data preparation is becoming more and more automated, but automation itself opens up new search spaces, dependencies of lineages, and variability of execution [10]. He et al. and Zoller and Huber also come to a similar conclusion in the AutoML literature: automation can make the model development more efficient, but, at the same time, search-based and pipeline-based processes will consume more resources and make operations more complicated [11], [12]. It applies to enterprise cases, where data preparation and model selection are not background tasks that are unlikely to cause harm. An

AutoML execution, feature recalculation loop or schema adaptation may compete on the serving tasks, which are latency-viable when at peak load, but which have the same pools of storage bandwidth, CPU and memory. An interesting discussion of the title topic should thus consider automated data and model pipeline generation to be dispatch amplifiers.

A third trend is related to execution substrate and process control. Casalicchio and Perciballi refer to container orchestration as a governance and automation problem, but not an infrastructural convenience per se [13]. According to Studer et al., CRISP-ML(Q) is a process model that is quality aware and formalizes handoffs and control points across the lifecycle of the ML [14]. Tamburri expands on that discussion by pointing out sustainability pressure on MLOps, such as operational waste, unreliable pipelines, and coordination costs [15]. Even the contributions remain rather consistent with the exact title, yet the ensemble of the studies justifies the necessity of enterprise dispatch to have an integrated approach. Container orchestration specifies the location of execution, process models specify which transitions are allowed, and MLOps concerns specify which control objectives are important. A bridge between all three is not well developed in literature.

Table 1: Summary of Key Findings

ef	Focus	Key Findings
6]	Software-engineering view of ML systems under production change	Operational burden concentrates in data dependence, evolving environments, and lifecycle coupling, indicating that dispatch policy must account for model-state and data-state variability rather than host load alone.
7]	Testing of ML programs under uncertain inputs	Validation difficulty rises when data distributions shift or upstream preparation changes, implying that ETL timing and transformation behaviour have direct consequences for safe model execution windows.
8]	Adoption of software-engineering practices in industrial ML	Partial uptake of versioning, automated testing, and reproducibility controls creates uneven operational readiness, which weakens consistent dispatch behaviour across enterprise teams and pipelines.
9]	Quality attributes of AI-based systems	Reliability, maintainability, and traceability emerge as system-level properties, supporting a dispatch design that includes lineage awareness, rollback support, and fault isolation.
10]	Intelligent data preparation and automated transformation selection	Automated preparation expands transformation search spaces and increases execution variability, making ETL less predictable and more sensitive to queueing and resource contention.
11]	AutoML state of the art	Search-heavy model automation improves exploration but can create bursty compute demand and unstable runtime footprints that complicate concurrent execution with ETL workloads.
12]	Survey and benchmark of AutoML frameworks	Framework heterogeneity produces large differences in orchestration overhead and pipeline breadth, implying that dispatch frameworks require method-aware admission control.
13]	Container orchestration in cloud-native systems	Orchestration layers offer placement and isolation benefits but do not automatically solve mixed-workload interference, especially under shared storage and network pressure.
14]	Quality-aware process model for ML operations	Formal lifecycle gates clarify approval, validation, and monitoring transitions, suggesting that dispatch should be coupled to quality state rather than raw resource availability alone.
15]	Sustainability and operational discipline in MLOps	Unstable pipelines, redundant recomputation, and control fragmentation generate operational waste, supporting a load-aware framework that suppresses unnecessary work during stress periods.

The literature thus provides a solid basis in which to argue how the topic of title issue is important but lacks solid basis on how the issue of joint dispatch ought to be put into action. Software engineering can study the existence of lifecycle coupling, data preparation studies can be used to understand why ETL behaviour is too volatile to naively schedule it, and orchestration studies can find points of control in containerised systems. Despite this, there is scanty direct comparative evidence on ETL-ML concurrency. Journal literature about the trade-off of a delayed transformation task between enterprise dispatch and an inference queue with strict latency requirements, or how a dispatcher should react to degradation of both data freshness and serving latency, is very scarce. The lack of that constrains strong causal claims.

Another important limitation is measurement heterogeneity. Some of the papers reviewed consider quality, governance or lifecycle structure as an alternative to runtime dispatch results [6], [9], [14]. Other articles are concerned with the range of automation or multitasking capacity of frameworks but not their performance with mixed workloads during contention [10], [11], [12]. The literature is therefore rich but limited in content of the elements of a dispatch framework and little overall evaluation of one. It is this disaggregated state that makes most enterprise deployments ad hoc based on a priority rule, hand negotiated maintenance windows, or infrastructure heuristics, which lack awareness of business semantics. So solemn an examination of the title subject matter must conclude that the area is conceptually active but functionally under-integrated.

III. METHODOLOGY

The lack of the ideas of control is not the methodological gap of the literature, but the lack of their integration. The current literature provides partial approaches which may be combined to form a topic-based dispatch logic. Lifecycle assurance studies provide state-awareness [1], [14], as well as validation gates. The studies of deployment and MLOps offer constraints on automation, observability, and feedback on operations [4], [5], [15]. Data preparation and AutoML studies can shed light on the variability of upstream execution graphs, and the heavy search and burstiness of upstream [10], [11], [12]. Orchestrating the containers, placement, isolation and policy hooks are introduced [13]. To implement load-sensitive dispatch architectures to execute ETL and ML models concurrently in enterprise systems must combine these strands into a single control loop that monitors load, classifies workload semantics, chooses a dispatch action and corrects action with feedback.

Figure 1 provides such a structure in a way which is clung to the title topic closely. The figure isolates sensing, policy, and execution capabilities and brings ETL and ML telemetry to the forefront, instead of hiding both within a generic scheduler. This picture helps to confirm one of the main points of this review: making sure that the freshness state, backlog state, model QoS state and infrastructure contention state are correlated before selecting a control action.

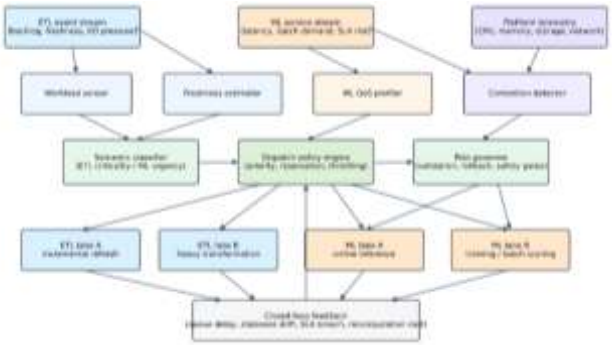

Figure 1: Load-Aware Dispatch Control Architecture for Concurrent ETL and ML Execution

This framework has four methodological families that are proposed in the literature. The former is the case of the first family which is known as a static or quasi-static reservation, where critical paths through inference or critical data refresh are guaranteed capacity; these are better with respect to predictability but not very good with respect to a rapidly changing enterprise demand [13]. Second is rule-based feedback control, which throttles, admission control or dynamic reprioritization is applied based on the queue length, staleness or violation of latency; this family is aligned with MLOps observability thinking, but might be semantically rich [5], [15]. The third control is search-based or optimization-oriented control and is observable in the form of AutoML and automated literature on pipelines; they can scale policy choice but might be unacceptable in real time enterprise considerations [11] as well as [12]. The fourth one is quality-gated orchestration embodied by lifecycle assurance, process-model work; this family guarantees safety and reproducibility but may be too hard when overloaded requires controlled degradation instead of an outright block [1], [14].

Single-layer reasoning is one of the conceptual weaknesses that are evident in such families. The most common data preparation papers are the ones that reason on the pipeline-construction layer, software engineering papers on the governance layer and orchestration papers on the infrastructure layer. The particular title subject matter is the one that requires multi-layer reasoning as a single dispatch action can modify the data freshness, inference latency, cost and fault exposure in unison. The conceptual model of the enterprise systems would then require at least six integrated control questions that comprise: what workload is coming in, the urgency of the workload, what resource is actually limited, whether the work can be safely performed, what lane to receive the workload and what corrective action to be taken in case the measured results do not match the expectation. The existing literature has partial answers to each of the questions, but seldom in one platform of assessment.

IV. RESULTS AND DISCUSSION

The reviewed literature shows the same background information: Non-model work plays a crucial role in the stability of production ML. This is the conclusion that can be found in the lifecycle assurance literature, empirical deployment literature, software engineering literature, and process-model literature [1], [3], [6], [8]. The title topic that is most applicable in implication is that ETL activities cannot be shunned to a background queue, which is simply handled using fairness of infrastructure. Typically include extraction, schema alignment, incremental refresh and validation processes in the enterprise system transform the difference between whether model execution is meaningful or not. A weakly invalid dispatch framework can be used to produce superficially healthy services with low inference latency and silence degradation of data freshness. Conversely, the framework with a skew towards achievement of ETL without service-level protection of serving paths might ensure freshness and violation of user-observable service-level guarantees. The policy goals that do not neglect the semantic freshness and service latency are justified by the reported studies.

Table 2: Method Comparison

ef	Method	Strengths	Limitations
6]	Software-engineering analysis of ML lifecycle dependencies	Reveals hidden coupling among data change, deployment state, and operational maintenance, which is crucial for dispatch policy design in shared enterprise platforms.	Does not provide runtime dispatch algorithms or mixed-load performance evaluation.
7]	Testing-oriented control of ML program behaviour	Exposes data-shift sensitivity and validation fragility, supporting dispatch rules that delay unsafe model execution after disruptive ETL events.	Testing logic alone cannot decide resource allocation under simultaneous ETL and serving pressure.
8]	Empirical assessment of best-practice adoption	Identifies reproducibility and automation gaps that explain unstable execution behaviour across teams and pipelines.	Observational scope limits direct inference about scheduler design or queue-control policies.
9]	Quality-attribute framing for AI-based systems	Connects reliability, traceability, and maintainability to system architecture, helping dispatch frameworks incorporate non-performance objectives.	High-level quality framing offers limited guidance for fine-grained runtime arbitration.
10]	Intelligent data preparation and transform recommendation	Supports adaptive ETL planning and reduction of manual transformation effort in dynamic data environments.	Transformation search and materialization steps can intensify burst load and widen resource variance.
11]	State-of-the-art AutoML search methods	Provides adaptive search and automatic configuration mechanisms that inspire policy tuning for dispatch.	Search-heavy execution can be computationally expensive and poorly aligned with low-latency production windows.
12]	Benchmarking of AutoML frameworks	Enables comparative understanding of orchestration breadth, resource footprint, and pipeline variability.	Benchmark settings do not directly model concurrent ETL plus online serving interference.
13]	Container orchestration mechanisms	Offers placement control, namespace separation, and baseline isolation tools suited to enterprise deployment.	Storage and network contention remain difficult under mixed ETL and model-serving load.
14]	Quality-aware lifecycle process model	Anchors dispatch to approval state, validation state, and monitoring readiness, reducing unsafe execution.	Process discipline may increase operational latency during overload if no degraded-mode policy exists.
15]	Sustainability-oriented MLOps governance	Highlights waste reduction, redundant recomputation control, and operational coherence, all relevant to overload management.	Conceptual scope exceeds concrete runtime scheduling detail.

The literature also suggests that a scheduler which is merely utilization based is not enough. Giray points out the complexity of the lifecycle of ML systems [6], and Serban et al. show that best-practices adoption in industries is not even [8]. This imbalance is important in mixed ETL-ML settings, as the dispatch behaviour is forced to counter the imbalanced maturity of teams, as well as the imbalanced readiness of pipelines. Execution with high priority should not be provided only

due to the fact that a job is large or urgent in terms of the locality. An operational load-sensitive model must have workload classification, which consists of dependency criticality, rollback cost, downstream coupling, and observability confidence. This definition aligns with CRISP-ML(Q) that is the quality checkpoints of the lifecycle [14]. The literature described thereby lead to the discussion of the generic scheduling efficiency to the controlled operability.

Table 2 compares methods used that were extracted in the literature reviewed. One can observe that there is no current method that can bring a balance between the latency protection, freshness preservation, overhead control and enterprise governability. The reservation of important paths is done by using static reservations; however, they do not respond well to changing loads. Search-based automation enhances better adaptivity but may also generate more spikes of workload. Quality-gated approaches enhance discipline at the expense of being too conservative. Container orchestration better placement and isolation but substrate-level orchestration does not necessarily encode the business semantics needed to support the coexistence of ETL-ML.

Table 3 shows the trends of the results reported or suggested by the literature. These findings are mostly qualitative because of the laxity of direct journal standards of co-current ETL-ML dispatch. However, there is consistent evidence that coordinated control increases the transparency in the operation as compared to fragmented control that creates hidden contention and gradual failure. The special lanes or the reservation are handy inference-serving stability, but the integrity and freshness of the data have to be clearly dealt with as well as the policy. The benefit of lifecycle-awareness in approaches is not always reported as a reduced raw latency, but the greatest benefit is usually reduced coordination failure.

Table 3: Results comparison

ef	**System / Context**	**Metric / Basis of Comparison**	**Outcome**
6]	Production ML systems with evolving data and deployment environments	Operational complexity, maintenance burden, dependency coupling	Strong evidence that runtime behaviour cannot be managed safely without awareness of data and lifecycle dependencies.
7]	ML program validation under input and model uncertainty	Test adequacy, oracle difficulty, failure detectability	Confirms that unsafe execution risk rises when upstream ETL alters input properties without coordinated control.
8]	Industrial ML practice across organizations	Adoption of versioning, testing, reproducibility, deployment discipline	Shows that uneven engineering maturity undermines predictable dispatch and recovery behaviour.
10]	Data preparation workflows with automated transformation support	Pipeline variability, preparation effort, transformation complexity	Indicates that ETL workloads can become less predictable as automation breadth increases.
11]	Automated model search pipelines	Search breadth, computational demand, automation capability	Suggests that adaptive search improves exploration but magnifies contention risk in shared enterprise clusters.
12]	AutoML framework comparison	Runtime footprint, orchestration breadth, comparative efficiency	Demonstrates substantial heterogeneity across frameworks, supporting method-aware admission control.
13]	Containerized application orchestration	Isolation capability, deployment flexibility, management scope	Improves control surfaces for dispatch yet does not eliminate mixed I/O and latency interference.
14]	Quality-assured ML process transitions	Lifecycle control, gate clarity, monitoring readiness	Improves governance and reduces unsafe transitions but can slow response without dynamic exception handling.
15]	MLOps operational governance	Waste reduction, pipeline coherence, sustainability logic	Supports suppressing redundant ETL or retraining work during stress, improving enterprise resilience.

The corpus reviewed has been transformed into a trend view of analysis and this is seen in figure 2. The figure does not claim that the number of publications is a factor of maturity of the field but shows the changing locus of focus of field between lifecycle assurance and testing to automation and orchestration across the period of the reviewed article. The

identical tendency contributes to the thesis that this discipline has already compiled numerous components of a dispatch structure without creating enough evidence of direct journal yet on mutual ETL-ML runtime arbitration.

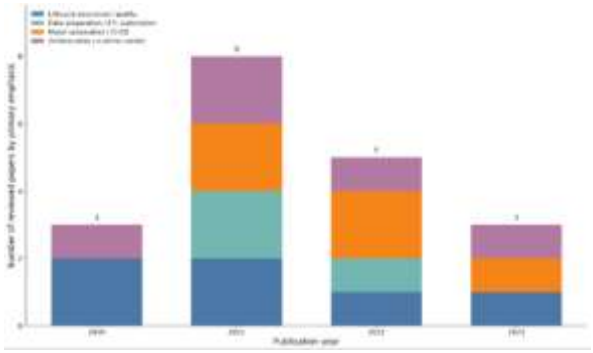

Figure 2: Thematic Emphasis in the Reviewed Literature, 2020–2023

A more comparative question is with respect to appropriateness of methods in the goals of enterprise control. Figure 3 provides an answer to that question, but with a dot-matrix design rather than a heatmap, as the literature is more of a directional comparison than a quantitative scoring. This number shows that the reservation and orchestration strategies are more or less resistant to latencies and isolation, whereas the quality-gated and data-aware strategies more influence the reproducibility, freshness integrity, and safe transition control.

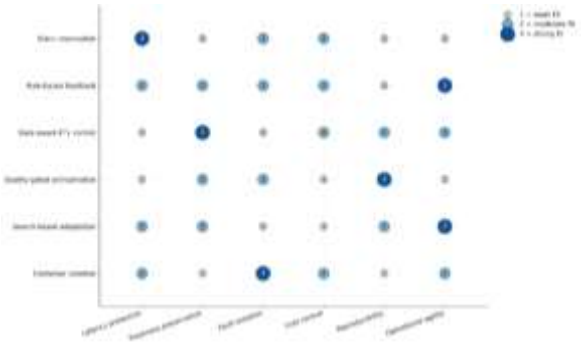

Figure 3: Method-Family Fit Across Enterprise Dispatch Objectives

The literature is also indicating that change in dispatch choice based on workload regime rather than being constant globally should occur. An example of operating map is illustrated in Figure 4, where there are two pressure axes: ETL load intensity, and ML latency criticality. The images support a domain-specific conclusion: the one that is the essence of the title topic: effective enterprise dispatch is regime-based, and the policy that would be optimal with heavy ETL, but moderate serving pressure is not the same as the policy that would be optimal with extreme serving pressure and moderate ETL demand.

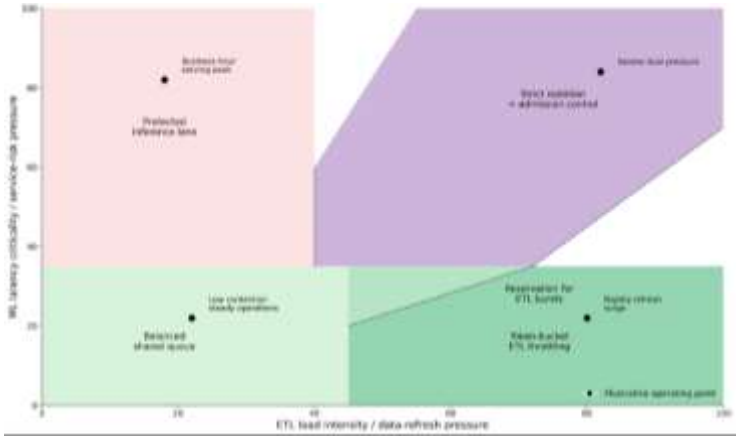

Figure 4: Operating-Region Map For Enterprise ETL-ML Dispatch Policy Selection

Together, the research studies recorded, and the comparative analysis done in Figures 2-4 are suggestive of four substantive conclusions. First, direct dispatch research of mixed ETL-ML enterprise workloads does not exist at a level of maturity despite the existence of enabling components maturing. Second, cross-layer observability is the most prevalent

capability that is not offered. Any dispatch structure cannot be intelligent unless it has telemetry between backlog and freshness and latency, and the validation status and substrate contention. Third, managing overload needs to be controlled degradation and not binary, success or failure. The use of the protective lanes, the control of admission, the selective validation, suppression of the unnecessary work appears to be mentioned numerous times in the sources [4], [13], [15]. Fourth, semantics should be in enterprise dispatch. When it is fed by a schema-repair ETL task feeding a controlled model, an exploratory retraining run can be very expensive to business risk but can be fed by virtually the same hardware with almost no business risk.

Contradictions also appear. More automation oriented literature praises that dynamic pipeline generation and search is possible to be created [10], [11], [12] but governance-oriented literature warns of the increase of hidden operational debt with increased automation [1], [14], [15]. Elasticity and portability are highly regarded in orchestration literature [13], and stability and controlled transitions are highly regarded in quality-oriented literature [1], [14]. Those are not incompatible positions, but the contradiction is important since the enterprise dispatch systems need to make decisions regarding whether or not elasticity is useful and whether or not elasticity just hastens unsafe or wasteful work. The most plausible interpretation of the reviewed literature is that dispatch should be both load-aware and state-aware. Load awareness without lifecycle-state awareness produces brittle systems because the dispatcher may respond to resource pressure while ignoring validation status, data freshness, rollback readiness, or model risk. Conversely, lifecycle-state awareness without load awareness can produce systems that are disciplined but operationally unresponsive during contention or overload.

V. FUTURE DIRECTIONS

Future studies on the specific title topic ought to go beyond the general MLOps architecture and discuss explicit mixed-workload dispatch investigations. Empirical research that is journal quality and considers ETL and ML in real-world enterprise scenarios, such as shared storage pressure, schema drift, bursty feature refresh, online inference deadlines and intermittent retraining demand, are the most urgently required. Although the existing literature has valuable components, the field has not yet created standard benchmark scenarios with which to collectively challenge ETL-ML. In the absence of these benchmarks, comparisons between dispatch methods are more of an impression than cumulative.

The second priority has to do with semantic telemetry. The majority of studies reviewed focused on observability, quality, or orchestration on a broad level but further research ought to establish a small number of dispatch-relevant signals which relate business semantics to runtime control. Freshness half-life, stale-prediction exposure, recomputation cost, dependency depth queue aging, and validation confidence and rollback readiness are some of the candidate signals. A possible path to go is to standardize data lineage metadata to serving telemetry in a manner such that dispatch can react to the interpretation of delay, rather than delay itself. This would lead to smarter scheduling of the enterprise without having to optimize the system unrealistically.

The third direction is the policy learning with safety restrictions. Involving search-based and automated techniques, it may be suggested that adaptive policy tuning may be more effective in dynamic environment as compared to fixed rules [11], [12]. Nonetheless, policy learning of enterprise ETL-ML dispatch cannot be considered as unconstrained reinforcement or black-box optimization. Further research on safe policy adaptation should examine safe adaptation of policy where the adoption of some workload classes, like regulated inference or freshness-critical ETL steps, are safeguarded by hard guards. A combination of the rule-based safety envelope and adaptive tuning of lower risk parameters appears to be especially promising.

The fourth orientation is organizational and architectural integration. Time and again, the literature indicates that the fragmented ownership of data engineering, ML engineering, platform engineering and governance functions is a cause of a failure of operations [2], [8], [9]. Future studies ought to look into the structures of actor-responsibility and technical dispatch logic. Enterprise systems must have dispatch frameworks, which are not only mathematically sound, but also explainable, auditable, and governable. A layered design where lifecycle state, quality gates, orchestration controls, and runtime dispatch have a common set of controls is the most realistic short term priority.

VI. CONCLUSION

An ETL and ML model load-sensitive dispatch framework to execute the two concurrently in enterprise systems has become a needed and yet inadequately amalgamated field of study. The reviewed literature illustrates the sound progress of machine learning lifecycle assurance, deployment automation, field of software engineering, intelligent data preparation, automated model search, container orchestration, and process design, which are quality sensitive. Nevertheless, only little first-hand evidence of dispatch policies that specifically arbitrage ETL and ML workloads on shared enterprise load exists.

The main point of this review is that it is impossible to reduce efficient enterprise dispatches to queue ordering or host utilization. The problem is inter-layer in nature. At the same time, dispatch choices modify the freshness of data, model

correctness, the latency danger, fault uncovering, operation wastefulness, and the governance stance. The existing literature offers requisite building blocks, but the blocks are still scattered in the neighbouring research streams. The conceptual structure concerning a given subject matter, therefore, becomes a necessity to correlate workload sensing, semantic classification, policy choice, quality gating, and feedback correction.

The scholarly concern of the subject is in its applied urgency. Enterprise value is becoming more and more reliant on foreseeable co-existence among ever-fresh streams of data and ever-consumed predictive services. Ongoing gaps in research involve a lack of benchmarks of mixed workloads, inadequate semantic telemetry, limited evidence on controlled degradation, and lack of integration of both technical and organizational control structures. The future will be based on direct empirical investigations that will consider ETL and ML concurrency as a systems problem on their own, as opposed to it being a secondary effect of data platforms today.

- **Interest Conflicts:** The author declares that there is no conflict of interest concerning the publishing of this paper.

V. REFERENCES

[1] Ashmore, R., Calinescu, R., & Paterson, C. (2021). Assuring the machine learning lifecycle: Desiderata, methods, and challenges. ACM Computing Surveys, 54(5), 1–39.

[2] Lwakatare, L. E., Raj, A., Bosch, J., Olsson, H. H., Crnkovic, I., & Ågren, S. M. (2020). Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. Information and Software Technology, 127, 106368.

[3] Paleyes, A., Urma, R.-G., & Lawrence, N. D. (2022). Challenges in deploying machine learning: A survey of case studies. ACM Computing Surveys, 55(6), 1–29.

[4] Shahin, M., Ali Babar, M., & Zhu, L. (2022). Continuous integration, delivery and deployment of machine learning models: A systematic review. Journal of Systems and Software, 189, 111331.

[5] Kreuzberger, D., Kühl, N., & Hirschl, S. (2023). Machine learning operations (MLOps): Overview, definition, and architecture. IEEE Access, 11, 31866–31879.

[6] Giray, G. (2021). A software engineering perspective on engineering machine learning systems: State of the art and challenges. Journal of Systems and Software, 180, 111031.

[7] Braiek, H. B., & Khomh, F. (2020). On testing machine learning programs. Journal of Systems and Software, 164, 110542.

[8] Serban, A., van der Blom, K., Hoos, H. H., & Visser, J. (2021). Adoption and effects of software engineering best practices in machine learning. Empirical Software Engineering, 26(5), 1–44.

[9] Martínez-Fernández, S., Franch, X., Jedlitschka, A., Oriol, M., & Trendowicz, A. (2022). Software engineering for AI-based systems: A survey. ACM Transactions on Software Engineering and Methodology, 31(2), 1–59.

[10] Bilalli, B., Abelló, A., Aluja-Banet, T., & Wrembel, R. (2022). Intelligent data preparation: A survey. The VLDB Journal, 31(4), 853–885.

[11] He, X., Zhao, K., & Chu, X. (2021). AutoML: A survey of the state-of-the-art. Knowledge-Based Systems, 212, 106622.

[12] Zöller, M.-A., & Huber, M. F. (2021). Benchmark and survey of automated machine learning frameworks. Journal of Artificial Intelligence Research, 70, 409–472.

[13] Casalicchio, E., & Perciballi, V. (2020). Container orchestration: A survey. Systems, 8(4), 43.

[14] Studer, S., Bui, T. B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S., & Müller, K.-R. (2021). Towards CRISP-ML(Q): A machine learning process model with quality assurance methodology. Machine Learning and Knowledge Extraction, 3(2), 392–413.

[15] Tamburri, D. A. (2020). Sustainable MLOps: Trends and challenges. Journal of Data and Information Quality, 12(3), 1–19.