Software Engineering and Artificial Intelligence Systems

Volume: 02 Issue: 01 (2026)

www.seais.org/

TALON: Tail-Latency-Aware Resource Scheduling for Privacy-Preserving Apache Spark

Ethan Caldwell, Clara Tremblay, Victor Lang*
University of Victoria, Victoria, BC, Canada
*Corresponding author: victorlang2931@gmail.com

Key word:
Apache Spark, tail latency, privacy-preserving analytics, resource scheduling, executor affinity, encrypted spill, differential privacy.

A B S T R A C T

Privacy-preserving Spark pipelines add executor-local costs—cryptographic context reconstruction, protected working sets, and authenticated spill—that interact with queueing and can amplify completion-time tails. This paper presents TALON, a tail-latency-aware scheduler that keeps privacy semantics fixed while jointly controlling task order, executor locality, and memory admission. TALON combines measured p50/p95 privacy-kernel models with an age-aware tail queue, a locality-aware placement score, and a deterministic makespan guard derived from a longest-processing-time reference schedule. Its concurrency-safe memory rule reserves minimum grants for other idle slots so that spill avoidance cannot silently collapse task parallelism. We calibrate the model with real AES-GCM, HMAC-SHA256, bounded Laplace aggregation, scrypt setup, and encrypted temporary-file measurements over bootstrapped UCI Optical Digits records. The evaluation is a reproducible Spark-style event-driven prototype rather than a patched Spark cluster. Across four 128-task workloads and 30 paired seeds, TALON reduces queue-inclusive p95 completion latency by 6.6–11.2% versus a strong reactive load-aware baseline, while makespan ranges from 5.1% lower to 5.4% higher. Cache-miss rate falls by 38.0–49.9%; encrypted spill is near parity in three workloads and 11.1% higher in the balanced case. Extreme-tail p99 is mixed, so the results support a bounded p95-oriented tradeoff rather than universal tail dominance..

INTRODUCTION

Apache Spark organizes data-parallel computation into stages and partition-level tasks, with executors providing reusable process-local state and resource slots [1], [2]. Spark SQL and Structured Streaming broaden this execution model to relational and incremental analytics [3], [4], while work such as Drizzle shows that coordination and scheduling granularity directly affect latency [5]. For ordinary analytics, a scheduler can often treat CPU time, memory, and locality as the dominant control dimensions. Privacy-preserving analytics complicate that picture because the task itself may carry cryptographic or privacy-system state whose cost is executor-local, and because memory pressure can turn intermediate data into encrypted spill rather than ordinary temporary I/O.

The systems literature has long recognized that average performance is insufficient for interactive services. Dean and Barroso highlighted how high-percentile delay can dominate user-visible behavior at scale [12], and low-latency schedulers such as Sparrow explicitly target short-job response time [8]. Cluster managers including Quasar, Firmament, and Altruistic Scheduling further demonstrate that quality-of-service and resource-efficiency goals can be made first-class scheduling concerns [13]–[15]. However, these systems do not model a privacy-specific combination that matters for Spark pipelines: a reusable executor context, a protected working set, and authenticated spill overhead that competes with concurrency. AERIS made an important step by exposing executor-local privacy state, protected working-set demand, and encrypted spill as schedulable signals [30]. Its measurement-backed treatment of those costs, together with its transparent reporting of queue-inclusive p95 behavior, motivates the complementary question pursued here: can a privacy-aware scheduler retain those useful signals while making completion-tail control an explicit objective rather than a side effect of task ordering? This paper answers that question with TALON, a tail-latency-aware task dispatcher and memory admission policy for Spark-style execution. TALON is deliberately narrow. It does not change the encryption algorithm, differential-privacy parameter, clipping bound, contribution policy, or released value. Instead, it observes only coarse metadata: partition size, operator class, opaque privacy-domain identifier, queue age, and executor load. From real local measurements it fits operator-level median and p95 service models. At each dispatch, TALON normally favors candidates with low predicted queue-inclusive p95 completion, but it periodically admits heavy tasks and activates a deterministic makespan guard when the remaining workload is predicted to exceed a stage budget. Executor selection then trades predicted tail completion against privacy-context reuse and load. A concurrency-safe memory rule reserves enough minimum grants to keep other idle executors launchable. The resulting design intentionally targets a Pareto point, not a universal optimum. In the paired trace study, TALON materially reduces p95 completion relative to a strong reactive scheduler, yet the p99 extreme tail is not consistently improved and one balanced workload spills modestly more data. These outcomes are reported because they expose the exact cost of prioritizing the 95th percentile. The contributions are fourfold: (1) a quantile-calibrated privacy-task cost model derived from real cryptographic and DP kernels; (2) a dual-queue tail policy with an explicit makespan guard; (3) a concurrency-safe memory admission rule that couples spill reduction with slot preservation; and (4) a reproducible evaluation artifact containing raw microbenchmarks, paired traces, task-level records, statistical comparisons, and ablations.

II. BACKGROUND AND RELATED WORK

A. Spark Resource Scheduling

Spark's stage/task abstraction exposes a natural task-dispatch control point [1], [2]. Mesos separates framework-level placement from cluster resource offers [6], and Dominant Resource Fairness provides a principled baseline for multi-resource allocation [7]. Sparrow shows that decentralized probing can reduce scheduling delay for fine-grained tasks [8]. Learned and measurement-driven approaches attack related problems at different layers: Decima learns scheduling policies from workload graphs [9], Ernest predicts analytics performance from sampled runs [10], and Rover adapts Spark SQL configurations online [11]. TALON differs in scope: it does not tune a job-wide configuration vector or learn an opaque policy; it uses interpretable task-level tail estimates because the privacy costs under study are local to specific executor/task pairings.

B. Confidential and Private Analytics

Confidential analytics systems protect data-in-use through enclaves, encrypted processing, or oblivious operators. Opaque integrates encrypted and oblivious analytics with Spark SQL [16]; VC3, SCONE, and Ryoan use trusted execution and isolation to protect outsourced computation [17]–[19]. Seabed, CryptDB, and Monomi enable analytics over encrypted representations with different leakage and trust models [20]–[22]. These systems establish that

confidentiality can add nonuniform state, memory, and I/O costs. TALON does not replace their security mechanisms; it treats privacy-operator cost as a scheduling input while leaving the security semantics intact.

Differential privacy introduces a semantic guarantee on released results [23]. PINQ, Airavat, and GUPT connect that guarantee to data-processing systems [24]–[26], while bounded-contribution SQL, Chorus, and PrivateSQL show how practical query engines can enforce or compile scalable differentially private mechanisms [27]–[29]. A resource scheduler must not improve latency by weakening epsilon, clipping, contribution bounds, or noise calibration. TALON therefore treats those parameters as invariants. The DP kernel in our artifact is used only to measure computational cost; no claim is made about utility on the handwritten-digit data.

C. Closest Privacy-Aware Scheduler

The closest prior system model is AERIS, which usefully demonstrates that reusable privacy state and encrypted spill deserve executor-scoped resource treatment [30]. TALON is complementary rather than a replacement: it preserves the same high-level idea that privacy overhead should be visible to the scheduler, but adds a measured p95 service model, queue-age sensitivity, a stage-level makespan guard, and an explicit concurrency reservation during memory admission. Our AERIS-inspired baseline is reimplemented under the present host, dataset, workload generator, and cost calibration; its numbers should therefore be read as a controlled comparison inside this artifact, not as a reproduction of another paper's reported values.

III. MODEL, OBJECTIVE, AND THREAT ASSUMPTIONS

A. Task and Executor State

For each ready task t, the controller observes $x_t=(n_t,o_t,d_t,s_t)$, where n_t is the partition row count, o_t is the privacy-operator class, d_t is an opaque privacy-domain identifier, and s_t is the stage identifier. The prototype contains four operator classes: bounded DP aggregation, HMAC-SHA256 pseudonymization, AES-GCM protection, and a combined HMAC+AES+DP transform. A domain identifier only names reusable executor context; it is not a secret key. Each executor e exposes its next available time, cumulative service load L_e, previous operator, and a two-entry least-recently-used cache of opaque domain identifiers.

Let W_t denote the estimated protected working set and g_t the task memory grant. The logical cluster has E=8 executors, M=96 MB of grantable memory, and a minimum per-task grant g_{min}=6 MB. The prototype approximates W_t from serialized feature bytes and operator intensity. Overflow $\max(0,W_t-g_t)$ becomes authenticated spill and incurs a cost calibrated by real AES-GCM temporary-file measurements. This logical memory pool is a stress-control parameter, not a claim about production Spark heap size.

B. Tail Objective

TALON's primary objective is queue-inclusive p95 task completion, not service time alone. Waiting time matters because a scheduler can make an individual task efficient after admission yet still delay it behind long work. For candidate task t on executor e with grant g, the controller uses the following predicted 95th-percentile completion surrogate:

$$C_{0.95}(t,e,g) = w_t + q_{0.95}(o_t,n_t) + I_{miss}\, q_{0.95}^{ctx} + q_{0.95}^{spill}(\max(0,W_t-g)) \quad (1)$$

where w_t is current queue age, $q_{0.95}(o_t,n_t)$ is the fitted p95 operator time, I_{miss} indicates a privacy-context miss, and $q_{0.95}^{spill}$ is derived from measured encrypted-spill timings. Because optimizing only p95 can defer the largest tasks and inflate stage completion, TALON constrains the choice with a makespan guard. At stage release r_s, it computes a longest-processing-time (LPT) reference duration $\hat{T}_{LPT}$ using the expected-cost model and defines a budget D_s with delta=0.08:

$$D_s = r_s + (1 + \delta)\,\hat{T}_{LPT}, \quad \delta = 0.08 \quad (2)$$

During execution, TALON re-estimates the LPT finish of remaining tasks using current executor availability. If that estimate exceeds D_s, the tail queue is overridden and the expected-heaviest task is selected. The guard is therefore not a fixed deadline promise; it is a

deterministic control mechanism that prevents sustained short-first ordering from accumulating heavy work at the stage boundary.

C. Threat Model and Privacy Invariants

We assume the driver, executor runtime, and chosen privacy operator are trusted to implement their specified semantics. Spill storage is treated as untrusted, so the benchmarked spill path uses AES-GCM authenticated encryption consistent with AES and GCM standards [33]. Network attackers, a malicious operating system, side channels, denial of service, compromised enclave hardware, and malicious Spark plugins are outside this prototype. The scheduler receives no plaintext records, keys, HMAC outputs, per-row DP contributions, or released DP answers.

Four invariants are enforced by construction. First, task placement never changes epsilon or clipping. Second, all spilled bytes use the same authenticated-encryption path regardless of policy. Third, domain identifiers remain opaque scheduler metadata. Fourth, tasks are reordered only within a stage whose tasks are modeled as independent partitions. Consequently, TALON alters performance behavior but not the mathematical definition of the privacy mechanism. This is an interface-level noninterference argument, not a side-channel proof.

IV. TALON DESIGN

A. Measurement-Calibrated Quantiles

A median-only model hides the variance that directly affects tail scheduling. TALON therefore fits separate nonnegative affine models to the measured median and empirical 95th percentile at each operator size. For operator o and row count n, $q_k(o,n)=a_{o,k}+b_{o,k}\,n$ for k in {0.50,0.95}. The model is intentionally simple: four measured row sizes are insufficient evidence for a high-capacity predictor, and an affine form keeps every scheduler decision auditable. Context reconstruction and spill are modeled separately from the operator so that state reuse and memory grants remain explicit control variables.

Table 1REPRESENTATIVE MEASURED PRIVACY COSTS

Primitive	Median m	P95 ms	Condition
DP	1.70	1.94	48k rows
AES	8.33	16.01	48k rows
HMAC	96.12	104.35	48k rows
COMBINED	109.74	124.42	48k rows
scrypt context	19.27	24.46	15 repetitions

At 48,000 rows, HMAC is the dominant primitive on this host: the median is 96.12 ms and the combined transform is 109.74 ms, compared with 8.33 ms for AES and 1.70 ms for the bounded DP aggregation. The scrypt context median is 19.27 ms. These values are hardware- and implementation-specific; the scheduler architecture does not assume the same ordering on another platform. Figure 1 shows the full measured scaling and the empirical p95 curves.

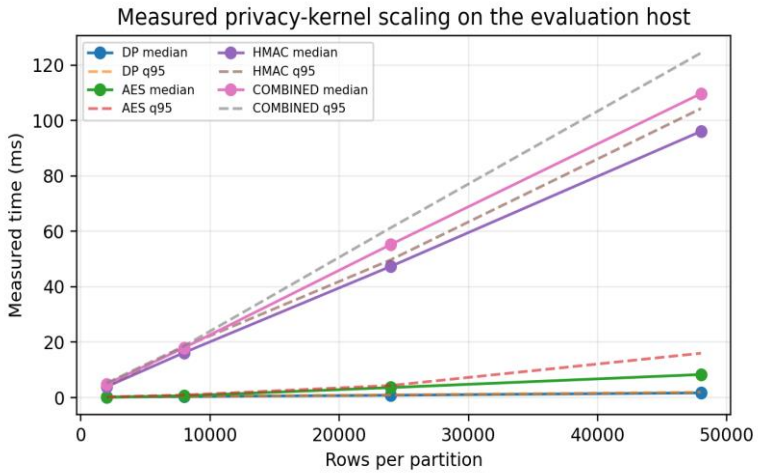


Figure 1Real local privacy-kernel timing used to calibrate median and p95 cost models.

B. Tail Queue with Heavy-Task Admission

When the makespan guard is inactive, TALON computes an age-adjusted task score q_0.95/(1+w/A), where A is twice the stage median q_0.95 with a 30 ms floor. Lower scores are preferred, so predicted short tasks are admitted early, but waiting progressively reduces a task's score. If a task from the heaviest decile waits longer than 2.5 times the stage q90 service estimate, an additional aging boost prevents starvation. Every fifth dispatch also considers a small heavy-task candidate set. These two mechanisms are intentionally simple and deterministic; they create a mixed queue rather than a pure shortest-processing-time policy.

The key safety mechanism is the budget check in (2). Before selecting from the tail queue, TALON simulates an LPT list schedule of the remaining expected costs using current executor availability. A predicted budget breach immediately promotes the expected-heaviest task. The no-guard ablation shows why this matters: a more aggressive tail queue can further reduce p95, but stage makespan grows sharply because the largest partitions collect at the end.

C. Locality-Aware Executor Scoring

For the selected candidate set, TALON evaluates each idle executor. A candidate executor score includes queue-inclusive p95 service, cumulative load, operation-switch overhead, and a context-miss cost. In simplified form,

Score(t,e) = C_0.95(t,e,g) + 0.03 L_e + 0.01 I_switch q_0.95 + 0.35 I_miss q_0.95^ctx (3)

The smallest feasible score wins. Context reuse is a soft preference, not a hard affinity: a cached domain can be ignored when queue age or load makes another executor better. This is important under domain shifts, where a strict placement rule would preserve locality at the cost of a growing queue. The no-affinity ablation removes context from this score while leaving the actual cache behavior intact.

D. Concurrency-Safe Memory Admission

Memory grants create a second tail hazard. A large grant can avoid encrypted spill but prevent other idle executors from launching, converting an I/O optimization into queueing delay. TALON begins from an adaptive privacy-aware grant and allows up to a 25% headroom, but reserves g_min for other idle slots that can immediately consume work. If U is currently reserved memory and R is the number of other launchable idle slots, the grant is bounded by

g_t <= M - U - R g_min (4)

Predicted short tasks are additionally capped at 12 MB in the prototype, preserving concurrency during the part of a stage that contributes most to p95. This rule does not guarantee minimum spill; instead, it makes the concurrency cost of spill avoidance explicit. When only a few tasks remain, R falls and the same task can receive more memory without suppressing parallelism.

E. Dispatch Procedure and Complexity

Table 2TALON DISPATCH AT AN IDLE-EXECUTOR EVENT

Step	Action
1	Update idle executors, queue ages, and current reserved memory.
2	Estimate remaining LPT finish; activate makespan guard if it exceeds D_s.
3	Choose heavy candidate under guard; otherwise build age-aware short-tail candidates plus periodic heavy admission.
4	For each candidate and idle executor, compute locality/load score in (3).
5	Choose lowest-score pair; assign concurrency-safe grant satisfying (4).
6	On completion, release memory, update executor load/history, and update the two-entry domain cache.

Sorting the ready set by predicted cost is O(N log N) per stage. At a dispatch event, TALON examines at most six tail candidates (or a small heavy set) across at most E idle executors, so the placement scan is O(E) with a small constant. The remaining-LPT estimate is O(N log E) if implemented with a heap. The prototype recomputes it directly because N=32 per stage; a Spark implementation could update it incrementally.

V. EXPERIMENTAL METHODOLOGY

A. Dataset and Real Microbenchmarks

The source records are the UCI Optical Recognition of Handwritten Digits dataset [31], accessed through scikit-learn [32]. The scikit-learn form contains 1,797 examples with 64 numeric features and ten classes. For a requested partition size, rows are sampled with replacement under a fixed seed and multiplied by independent Gaussian factors with mean 1 and standard deviation 0.001. This deterministic bootstrap is used only to scale computational work; the expanded partitions are not presented as new handwritten-digit observations, and the paper draws no classification or statistical inference from them.

The AES kernel serializes 64 float32 features and performs AES-GCM encrypt/decrypt verification. The HMAC kernel creates a 16-byte truncated HMAC-SHA256 pseudonym for each row ordinal. The DP kernel clips the first feature to [0,16], computes a mean for each of ten classes, and adds Laplace noise with scale 16/count at epsilon=1. The combined kernel executes HMAC, AES-GCM, and the DP aggregation in sequence. Operator measurements use four partition sizes (2k, 8k, 24k, and 48k rows), one untimed warm-up, and nine timed repetitions. Scrypt context setup uses 15 repetitions. Authenticated spill uses 1, 4, 12, and 24 MB payloads, one or four segments, and five repetitions per configuration, including encryption, temporary-file write/fsync, read, and decryption.

B. Spark-Style Event Prototype

The evaluation artifact is implemented in Python as an event-driven scheduler aligned with Spark stage barriers and executor occupancy. Each task holds one executor and a memory grant until completion. The next stage is released only when all tasks in the current stage finish. Executor state includes availability, cumulative service, previous operator, and an LRU privacy-domain cache. The simulator records start and finish times, queue time, service time, completion time, grant, spill, domain miss, executor identifier, and workload seed for every task.

This is not a patched Apache Spark cluster. The available environment did not contain a Spark runtime, and installing or emulating a distributed JVM stack solely to label the experiment 'Spark' would have weakened reproducibility. Instead, the artifact separates real local privacy-kernel measurement from deterministic Spark-style scheduling. It therefore supports controlled paired policy comparisons but omits network shuffle, JVM allocation and garbage collection, code generation, disk interference across physical executors, speculative copies, executor loss, dynamic allocation, and heterogeneous hosts.

C. Workloads, Baselines, and Protocol

Table 3TRACE WORKLOADS (32 TASKS/STAGE; 128 TASKS/TRACE)

Trace	Stages: mean rows / sigma	Stress
Balanced	HMAC 18k/.25; AES 24k/.25; Combined 20k/.25; DP 16k/.25	stationary
Skewed	HMAC 18k/.80; AES 32k/.80; Combined 26k/.80; DP 18k/.80	heavy partition skew
Bursty	DP 12k/.55; Combined 38k/.60; AES 16k/.45; Combined 42k/.70	alternating light/heavy
Shifted	HMAC 20k/.45; Combined 32k/.55; AES 22k/.50; Combined 36k/.60	domain mix shifts mid-stage

Partition sizes follow mean-preserving lognormal distributions and are clipped to 2,000–120,000 rows. In the first three workloads, four domain identifiers occur with probabilities 0.46, 0.26, 0.18, and 0.10. The shifted trace uses a 0.65/0.20/0.10/0.05 distribution in the first half of each stage and 0.08/0.12/0.30/0.50 in the second half. Each task also receives deterministic per-seed lognormal multipliers for base service, context setup, and spill so that policies face paired stochastic traces. We compare four main policies. Uniform uses FIFO order, 12 MB grants, and round-robin idle placement. Reactive uses longest-predicted-first order, least-busy placement, and a bounded 6–18 MB load-reactive grant. AERIS-inspired uses longest-predicted-first order, privacy-domain locality, cumulative-load regularization, and adaptive grants; it is a controlled reimplementation under this artifact's calibration. TALON is the full policy in Section IV. All policies use eight executors, M=96 MB, g_min=6 MB, and a two-domain cache. Thirty independent seeds are paired across policies for each workload.

Metrics are queue-inclusive p95 and p99 completion latency, p95 service time, stage/job makespan, total encrypted spill MB, privacy-context miss rate, peak reserved memory, and Jain fairness over per-executor service. Aggregate means use 95% confidence intervals 1.96s/sqrt(30). Paired percentage changes are computed seed by seed. A one-sided paired

Wilcoxon signed-rank test asks whether a baseline p95 completion metric is larger than TALON; the exact p-values are reported alongside effect sizes. The main evaluation contains 61,440 simulated task instances (4 workloads × 30 seeds × 4 policies × 128 tasks), all driven by the same real measurement calibration.

D. Execution Environment

Measurements were recorded on Linux 6.18.35 with five visible Intel Xeon Platinum 8370C CPUs at 2.80 GHz, Python 3.13.5, NumPy 2.3.5, pandas 2.2.3, SciPy 1.17.0, scikit-learn 1.8.0, and cryptography 46.0.4. The artifact includes the environment manifest, raw microbenchmark CSV, fitted models, per-run summaries, paired tests, task-level records, figures, and the code used to regenerate them.

VI. RESULTS

A. Main P95 Completion Results

Table 4TALON VS. REACTIVE OVER 30 PAIRED SEEDS

Trace	Reactive p95	TALON p95	p95 gain	Makespan gain
Balanced	215.9	201.5	+6.6%	+5.1%
Skewed	314.2	278.3	+11.2%	-3.2%
Bursty	530.8	475.8	+10.2%	-3.0%
Shifted	437.8	405.0	+7.5%	-5.4%

TALON reduces p95 completion in every workload relative to Reactive: 6.57% on balanced traces, 11.23% under skew, 10.17% on bursty traces, and 7.48% under the domain shift. The one-sided paired Wilcoxon p-values are 4.0e-8, 1.86e-9, 9.31e-10, and 9.31e-10, respectively. The improvement is therefore not driven by a few favorable seeds. Figure 2 also shows that the policy ordering depends on workload: Uniform has a slightly lower mean p95 than TALON in the skewed trace (269.36 vs. 278.35 ms), but its mean makespan is much larger (967.16 vs. 812.21 ms). This is exactly the tradeoff a single tail metric can hide.

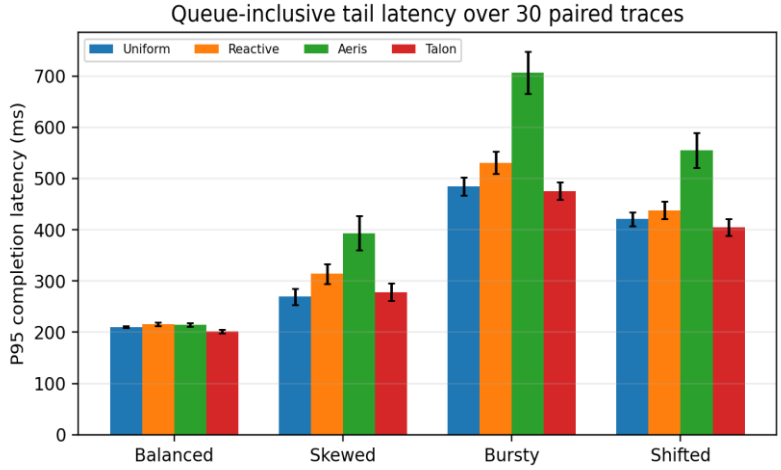


Figure 2Queue-inclusive p95 completion latency; error bars are 95% confidence intervals over 30 paired seeds.

B. Makespan–Tail Tradeoff

Against Reactive, TALON's makespan changes from 5.12% better on balanced traces to 3.23%, 3.01%, and 5.38% worse on skewed, bursty, and shifted traces. Thus, the measured cost

of the p95 improvement is bounded to roughly five percent in this test matrix rather than the much larger stage-tail expansion observed when the guard is removed. Against the AERIS-inspired policy, TALON is 3.75% faster on balanced, 0.33% slower on skewed, 17.42% faster on bursty, and 8.24% faster under domain shift; its p95 completion is lower by 5.92%, 28.19%, 31.92%, and 26.33%, respectively. Figure 3 plots the mean makespan/p95 points and makes the Pareto structure visible.

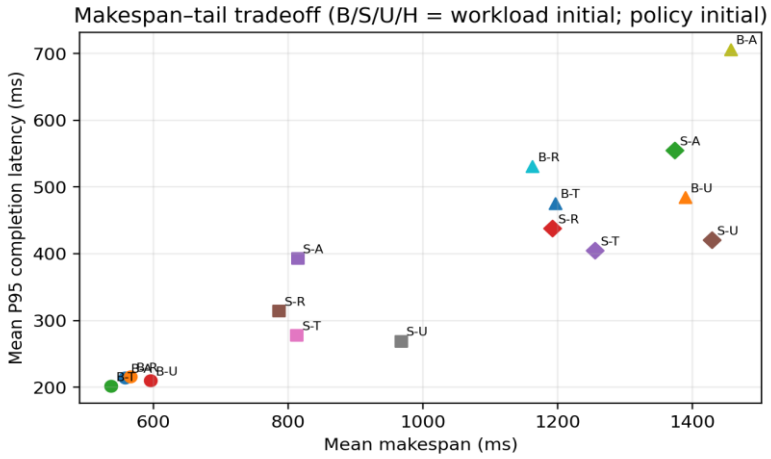


Figure 3. Mean makespan versus mean p95 completion. Labels combine workload initial and policy initial

The extreme p99 tail is less favorable. Relative to Reactive, TALON improves mean p99 by only 3.55% on balanced and 0.88% on skewed; bursty p99 is essentially unchanged (+0.16%), while shifted p99 is 2.42% worse. These differences are not consistently significant. The result is important because the scheduler explicitly protects the 95th percentile, not the maximum or 99th percentile. A production policy that requires p99 service-level objectives should either retune the quantile model or maintain separate p95 and p99 guards.

C. Privacy-Specific Overheads

TALON strongly reduces privacy-context misses relative to Reactive: 49.91% on balanced, 41.16% on skewed, 38.01% on bursty, and 44.93% under domain shift. The reduction occurs even though locality is soft, because the candidate/executor score accounts for context cost without forcing a task onto a loaded executor. This matters most when the setup median (19.27 ms) is nontrivial relative to AES or DP service, and less when HMAC dominates total service. Encrypted spill is more nuanced. Compared with Reactive, TALON reduces spill by 4.65% on skewed traces, 0.15% on bursty traces, and 3.26% under domain shift, but spills 11.07% more on balanced traces. In absolute terms, the balanced difference is 42.50 versus 38.66 MB across 128 tasks. The concurrency reservation intentionally allows this outcome: it will accept some spill rather than consume enough memory to suppress idle-slot admission. Relative to the AERIS-inspired policy, TALON spills substantially more, confirming that AERIS-style aggressive spill avoidance and TALON's p95 objective occupy different operating points rather than one dominating the other.

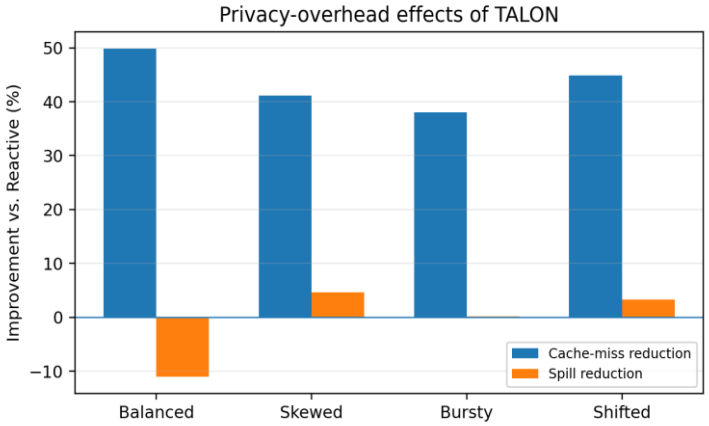


Figure 4TALON cache-miss and encrypted-spill changes relative to Reactive. Positive values are reductions.

D. Ablation Study

The makespan guard is the dominant structural ablation. Removing it lowers p95 further—from 278.35 to 232.50 ms on skewed traces and from 475.77 to 442.72 ms on bursty traces—but raises makespan from 812.21 to 1050.16 ms and from 1196.18 to 1472.40 ms. The full policy therefore gives up some p95 improvement to prevent long partitions from accumulating at the end of a stage. This directly validates the need for a constrained rather than pure short-first objective.

Removing locality from the executor score raises context miss rate from 0.26 to 0.45 on skewed traces and from 0.28 to 0.46 on bursty traces. It also increases p95 completion by 3.15% and 4.29%, respectively. By contrast, replacing TALON's adaptive grant with a fixed 12 MB grant has modest effects in this calibration: skewed spill is 2.23% higher with the fixed grant, while bursty spill is 0.89% lower. The memory component is therefore secondary to the queue/guard and locality mechanisms on this host; the paper does not inflate its contribution.

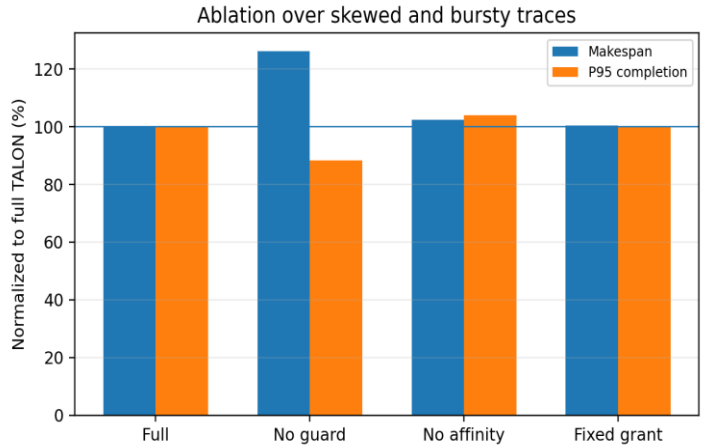


Figure 5. Ablation normalized to full TALON, averaged over skewed and bursty paired seeds.

E. Resource Safety and Fairness

The simulator enforces the 96 MB global grant bound at every dispatch. TALON's mean peak reservation remains within that limit for every workload. Jain fairness over per-executor service is approximately 0.99–1.00 across policies, indicating that the tail/locality score does not create persistent executor starvation in these traces. This fairness result should not be confused with job-level multi-tenant fairness, which the prototype does not model.

VII. DISCUSSION

A. What the Results Establish

The experiments support three bounded conclusions. First, privacy-aware scheduling can benefit from predicting a quantile rather than only mean or median service: queue-inclusive p95 improves consistently relative to a load-aware reactive policy. Second, a tail-first queue

needs an explicit stage-completion safeguard. Without the LPT-based guard, the scheduler can optimize the 95th percentile by sacrificing a small set of long tasks so aggressively that makespan becomes unacceptable. Third, executor-local privacy state remains useful even when tail latency is the primary objective; removing locality materially increases context reconstruction and p95.

The results do not establish that TALON is universally optimal. Uniform happens to achieve the lowest p95 in the skewed trace by allowing a much longer makespan, and the AERIS-inspired policy achieves much lower spill in every memory-stressed case. These comparisons are useful rather than problematic: they expose a three-way frontier among p95 completion, stage makespan, and encrypted spill. A deployment should surface the relevant policy knob instead of collapsing all three into an opaque scalar reward.

B. Spark Integration Path

A practical Spark integration could attach an opaque privacy profile to TaskSet metadata and maintain domain identifiers in executor heartbeat extensions. A TaskScheduler-level component could compute the tail/guard ordering, while an executor-side memory broker reserves off-heap or managed capacity before task launch. Completion events would update the measured service model, recent operator class, and domain cache. The scheduler need not receive ciphertext contents or DP outputs. Spill encryption should remain inside the privacy operator, not in the scheduler, so that placement logic cannot bypass the protection mechanism. Speculation and failure recovery require extra care. A speculative copy should not double-spend a DP privacy budget, and a failed executor must invalidate any assumed reusable context. Dynamic allocation changes E and therefore changes both the LPT reference schedule and the concurrency reservation in (4). These are engineering requirements for a Spark patch; they are not implemented in this artifact.

C. When to Use a Different Operating Point

TALON is appropriate when queue-inclusive p95 is operationally important and a small makespan tolerance is acceptable. If secure spill volume is the dominant cost—for example, when encrypted temporary storage is unusually slow or sensitive—an AERIS-style higher-memory operating point may be preferable. If p99 or maximum completion has a hard SLO, the guard should target that quantile or introduce explicit per-task deadlines. The core design is modular: the measurement layer can supply several quantiles, while the admission layer can change the reserve or spill penalty without touching privacy semantics.

VIII. LIMITATIONS AND THREATS TO VALIDITY

External validity is the principal limitation. The work is a measurement-backed trace prototype, not a multi-node Spark deployment. It does not model shuffle networking, JVM garbage collection, whole-stage code generation, object allocation, disk contention among physical executors, speculative execution, executor failure, adaptive query execution, dynamic allocation, or heterogeneous machines. The logical 96 MB memory pool is chosen to create controlled contention; it is not intended to resemble an executor heap. The real dataset is small and numerical. Bootstrap expansion preserves the computational shape of 64 float features but not the semantic complexity of production logs, joins, nested records, or images. Only four operator profiles and four privacy domains are modeled. The DP kernel is a bounded ten-group mean at epsilon=1 and is evaluated for timing only; repeated-query privacy composition and utility are outside scope. HMAC truncation and deterministic benchmark keys are measurement choices, not key-management guidance. Microbenchmarks are local and can be affected by CPU frequency, allocator state, filesystem caching, and noisy neighbors. We use warm-ups, multiple repetitions, separate median/p95 fits, and raw-data release to make the calibration auditable, but four row sizes cannot capture arbitrary nonlinear behavior. The simulator uses independent deterministic lognormal task multipliers and therefore omits correlated interference. Thirty seeds support paired effect-size estimation but remain too small for precise extreme-tail probability estimation, which is consistent with the mixed p99 result.

Finally, partition size and operator class can themselves be sensitive metadata under a stronger threat model. A deployment may need bucketing, access control, or privacy-preserving telemetry. None of these limitations invalidate the measured kernel timings or the deterministic policy comparison; they bound the claim to a reproducible scheduling design hypothesis for privacy-preserving Spark.

IX. CONCLUSION

This paper introduced TALON, a p95-oriented resource scheduler for privacy-preserving Spark-style analytics. TALON combines measured median/p95 privacy costs, an age-aware tail queue, soft executor-local privacy affinity, a deterministic LPT makespan guard, and concurrency-safe memory admission. In four paired trace families calibrated by real cryptographic and DP microbenchmarks, it lowers p95 completion by 6.6–11.2% relative to a strong reactive baseline while keeping makespan within a range from 5.1% faster to 5.4% slower. The same evaluation shows the limits of the approach: p99 is mixed, Uniform can win p95 under one skewed workload at substantial makespan cost, and a spill-focused privacy-aware policy can use less encrypted temporary storage. The central result is therefore a controllable frontier, not a universal winner. A full Spark implementation should retain the privacy-semantic invariants while validating the tail guard under shuffle, JVM, failure, and dynamic-allocation effects.

REFERENCES

Xing, S., & Wang, Y. (2025). Proactive data placement in heterogeneous storage systems via predictive multi-objective reinforcement learning. IEEE Access, 13, 117986-117998.

Ji, E., Wang, Y., Xing, S., & Jin, J. (2025). Hierarchical reinforcement learning for energy-efficient API traffic optimization in large-scale advertising systems. IEEE Access.

Xing, S., & Wang, Y. (2025). Cross-modal attention networks for multi-modal anomaly detection in system software. IEEE Open Journal of the Computer Society.

Liu, Y., Ren, S., Wang, X., & Zhou, M. (2024). Temporal logical attention network for log-based anomaly detection in distributed systems. Sensors, 24(24), 7949.

Ren, S., Jin, J., Niu, G., & Liu, Y. (2025). ARCS: Adaptive reinforcement learning framework for automated cybersecurity incident response strategy optimization. Applied Sciences, 15(2), 951.

Zhang, S. and Qiu, L. (2024). Data-driven assessment of pavement performance and resilience under extreme weather using the Long-Term Pavement Performance Program. AI and Data Science Journal, vol. 1, no. 1, pp. 57–66

Zhang, H. (2024). Graph-Aware Multi-Task Learning For Early Prediction Of Timing Violations And Routing Congestion In Digital Ic Physical Design. World Journal of Information Technology, 2(1): 13-20.

Wang, J., Tan, Y., Jiang, B., Wu, B., & Liu, W. (2025). Dynamic marketing uplift modeling: A symmetry-preserving framework integrating causal forests with deep reinforcement learning for personalized intervention strategies. Symmetry, 17(4), 610.

Zi, B. (2024). Cloud-native distributed systems for real-time payment intelligence. AI and Data Science Journal, 1(1), 51-56.

Zhang, S. & Qiu, L. (2023). An Interpretable, Uncertainty-Aware Framework For Bridge Deterioration Prediction And Risk-Based Maintenance Prioritization. Academic Journal of Architecture and Civil Engineering, 1(2): 21-28.

Jin, J., Xing, S., Ji, E., & Liu, W. (2025). Xgate: explainable reinforcement learning for transparent and trustworthy api traffic management in iot sensor networks. Sensors, 25(7), 2183.

Xing, S., Wang, Y., & Liu, W. (2025). Multi-dimensional anomaly detection and fault localization in microservice architectures: A dual-channel deep learning approach with causal inference for intelligent sensing. Sensors, 25(11), 3396.

Li, P., Ren, S., Zhang, Q., Wang, X., & Liu, Y. (2024). Think4SCND: Reinforcement learning with thinking model for dynamic supply chain network design. IEEE Access, 12, 195974-195985.

Zhao, X., Liu, J., Wang, Y., & Wang, J. (2026). CryptoMamba-SSM: Linear complexity state space models for cryptocurrency volatility prediction. IEEE Open Journal of the Computer Society, 7, 226-243.

Wang, J., Liu, J., Zheng, W., & Ge, Y. (2025). Temporal heterogeneous graph contrastive learning for fraud detection in credit card transactions. IEEE Access.

Sun, T., Yang, J., Li, J., Chen, J., Liu, M., Fan, L., & Wang, X. (2024). Enhancing auto insurance risk evaluation with transformer and SHAP. IEEE Access, 12, 116546-116557.

Zhang, H. (2025). Reinforcement Learning Approaches for Layout Optimization in Electronic Design Automation with Electromagnetic Compatibility Constraints. Frontiers in Robotics and Automation, 2(2), 77-93.

Han, X., Yang, Y., Chen, J., Wang, M., & Zhou, M. (2025). Symmetry-aware credit risk modeling: A deep learning framework exploiting financial data balance and invariance. Symmetry, 17(3), 341.

Zhao, X., Sun, T., Ren, S., Yang, J., & Liu, Y. (2025). RAG-Based AI Agents for Enterprise Software Development: Implementation Patterns and Production Deployment. Frontiers in Artificial Intelligence Research, 2(3), 501-520.

Chen, J., Cui, Y., Zhang, X., Yang, J., & Zhou, M. (2024). Temporal convolutional network for carbon tax projection: A data-driven approach. Applied Sciences, 14(20), 9213.

Sun, T., Wang, M., & Chen, J. (2025). Leveraging machine learning for tax fraud detection and risk scoring in corporate filings. Asian Business Research Journal, 10(11), 1-13.

Yue, X. & Zhu, R. (2024). An Explainable Hybrid Machine Learning Framework For Cash-Flow-Conditioned Multi-Horizon Liquidity Risk Early Warning: An Sme-Oriented Benchmark Study. World Journal of Management Science, 2(1): 63-72.

Cui, Y., Han, X., Chen, J., Zhang, X., Yang, J., & Zhang, X. (2025). FraudGNN-RL: a graph neural network with reinforcement learning for adaptive financial fraud detection. IEEE Open Journal of the Computer Society, 6, 426-437.

Hu, X., Zhao, X., Wang, J., & Yang, Y. (2025). Information-theoretic multi-scale geometric pre-training for enhanced molecular property prediction. Plos one, 20(10), e0332640.

Wang, M., Zhang, X., Yang, Y., & Wang, J. (2025). Explainable machine learning in risk management: Balancing accuracy and interpretability. Journal of Financial Risk Management, 14(3), 185-198.

Yang, J., Li, P., Cui, Y., Han, X., & Zhou, M. (2025). Multi-sensor temporal fusion transformer for stock performance prediction: An adaptive Sharpe ratio approach. Sensors, 25(3), 976.

Sun, T., Wang, M., & Chen, J. (2025). Leveraging machine learning for tax fraud detection and risk scoring in corporate filings. Asian Business Research Journal, 10(11), 1-13.

Bian, M., Li, P., Lin, Y., Wang, Y., & Teng, D. (2026). Assessing Green Public Supply Chains Resilience under Carbon Neutrality Goals: A Multi-Agent Reinforcement Learning Framework. IEEE Access.

Chen, J., Wang, M., & Sun, T. (2025). Intelligent tax systems and the role of natural language processing in regulatory interpretation. American Journal of Machine Learning, 6(4), 74-94.

Chen, Z., Liu, J., & Chen, J. (2025). Machine Learning Methods for Financial Forecasting in Enterprise Planning: Transitioning from Rule-Based Models to Predictive Analytics. Frontiers in Artificial Intelligence Research, 2(3), 541-564.

Chen, J., Liu, J., Liang, Y., & Zhou, M. (2026). KE-MLLM: A knowledge-enhanced multi-sensor learning framework for explainable fake review detection. Applied Sciences, 16(6), 2909.

Yue, X., Yang, J., & Liu, W. (2026). FraudDebate-Agent: A Multi-Agent LLM Framework with an Evidence-Based Debate Mechanism for Financial Statement Fraud Detection. Mathematics, 14(15), 2695.

Zhu, R. & Yue, X. (2024). An Explainable Machine Learning Framework For Predicting Dividend Sustainability And Financial Risk Of U.s. Reits Under Interest-Rate Shocks. Journal of Trends in Finance and Economics, 1(2): 48-57.