DOI: https://doi.org/10.15276/ict.03.2026.35
UDC 004.41:004.75

Applicability conditions, architecture, and quantitative models of a hardware second layer in two-layer distributed event processing

Oleksandr M. Kupriianov[1)]
ORCID: https://orcid.org/0009-0003-8155-1321; kupriyanov.a062@stud.op.edu.ua. Scopus Author ID: 60718473000
Andrii O. Maksymenko[1)]
ORCID: https://orcid.org/0000-0003-3117-1657; maksymenko.a.o@op.edu.ua. Scopus Author ID: 57215216446
[1)] Odesa Polytechnic National University, 1, Shevchenko Av. Odesa, 65044, Ukraine

ABSTRACT

Relevance. Events that share an aggregate identifier are functionally dependent: they must be applied to the projection state strictly in order, so the throughput of one aggregate is limited by a single serial executor, horizontal scaling does not raise this per-key ceiling, and software executors additionally exhibit pause-driven latency tails. **The aim of the article** is to determine the formal conditions under which the second (projection) layer of a two-layer distributed event-processing architecture should be computed in reconfigurable hardware rather than in software, and to construct and validate quantitative models of such an implementation. **Tasks:** formalization of the projection as a per-key state transducer; derivation of applicability attributes and decision conditions; specification of the reference architecture of the serialization layer, the hardware core, and the host software; construction and validation of models of throughput, latency, cost, conflated egress, and checkpointing. **Methods:** the theory of cost register automata and join-semilattices, queueing theory, renewal theory, the Young-Daly checkpoint model, and discrete-event simulation. **Scientific novelty:** for the first time, a formal applicability framework – six attributes with connecting propositions and four decision conditions – is proposed that determines exactly which event-processing algorithms benefit from a hardware projection layer; for the first time, an exact closed-form expression is obtained for the egress rate and the mean age of information of a rate-limited last-value cache fed by a Poisson update stream; the two-layer in-memory architecture is improved by relocating the projection computation into a hardware core with a fixed-offset serialization contract; the quantitative models of two-layer event processing are further developed for the hardware substrate. **Practical significance:** decision surfaces and break-even curves that allow practitioners to justify the choice between the hardware layer, tuned software, and stream frameworks, together with implementation principles and a verification protocol that reduce the risk of the hardware path. **Results:** a single hardware accelerator sustains a high volume of small-packet events per second—significantly outperforming conventional processor cores and standard software serial rates; it maintains ultra-low and predictable tail latencies while eliminating pause-afflicted software spikes; it achieves high cost-efficiency compared to software alternatives at scale under current cloud pricing; and it substantially reduces auction fan-out egress while maintaining a bounded mean staleness.**Conclusions:** the transfer of the second layer into hardware is a contract of six attributes; where the contract holds, the gains are structural rather than incremental, and where it fails, stream frameworks remain the right substrate. All closed forms are validated by simulation with a negligible worst-case relative error.

Keywords: Stream processing; field-programmable gate array; hardware description language; event sourcing; command-query responsibility segregation; conflation; age of information; cost register automata; checkpointing; Kafka Streams; market data

For citation: **Kupriianov O. M., Maksymenko A. O. "Applicability conditions, architecture, and quantitative models of a hardware second layer in two-layer distributed event processing". *Informatics. Culture. Technology*. 2026; Vol. 3 No. 1 (3): 444–460. DOI: https://doi.org/10.15276/ict.03.2026.35**

INTRODUCTION

Relevance of the research. Stateful event-processing systems must frequently validate every incoming event against the current projection of accumulated state: a new bid on an auction is compared with the current best bid, a stock reduction is checked against the available quantity, an order is validated against position limits. Events sharing an aggregate identifier are therefore functionally dependent: event $e_{k,i+1}$ of aggregate k can only be interpreted against the projection state produced by $e_{k,i}$, so every round-trip to remote storage serializes the aggregate's entire stream, and no amount of horizontal scaling raises the per-key throughput ceiling.

The In-Memory Quorum-Partitioned Stateful Processing (IQPS) architecture, breaks the ceiling with a two-layer design: Layer 1 (K consumers) routes events by consistent hashing to Layer 2 (M in-memory projection instances), which checkpoint to persistent storage every N events under quorum control and rely on stream replay for recovery.This paper asks the next question: if the per-key update logic is small and fixed, why execute it on a central processing unit (CPU) at all? The

hardware variant studied here – hereinafter IQPS-F, where "F" refers to the field-programmable gate array (FPGA) – replaces the Layer-2 software projection with a reconfigurable-hardware core described in the VHSIC Hardware Description Language (VHDL).

The four duties the user-visible specification assigns to that core map onto classic architectural elements:

1) **reaction synthesis** – for each triggering event, compute the reaction event(s) (Bid Posted → Bid Accepted/Bid Rejected), i.e., the output function of a state transducer;

2) **projection update** – the state-transition function of the same transducer, executed against on-card memory;

3) **snapshot export** – every N events or T_{ckpt} seconds (whichever comes first), write a consistent state image to a staging area from which the host synchronizes it to persistent storage — the IQPS checkpoint, relocated;

4) **batched, conflated emission** – every E events or T_{emit} seconds, release reaction events to the delivery queue, *and* guarantee per-subscriber notification rates of at most Q messages per second by conflation (e.g., a participant's auction rank is delivered at most Q times per second regardless of how many bids changed it).

Duty 4 is not merely batching: it is a semantic filter that deliberately drops superseded notifications. Section 4 shows this is sound exactly when the notification stream forms a last-write-wins (LWW) join-semilattice, and Section 6 quantifies both the egress reduction and the staleness cost with an exact renewal analysis validated by simulation. Duty 4's rate guarantee mirrors industrial practice in market-data distribution, where exchanges conflate feeds at fixed intervals (CME Globex at 50 ms; the real-time optimized feeds of the London Stock Exchange Group at 3 updates per second) [2], [3].

Because a hardware parser cannot chase pointers through a self-describing format, Layer 1 gains a duty in IQPS-F: besides routing, it serializes events into a fixed-offset binary schema (Section 5.1), assigns per-partition sequence numbers, and injects tick events so that time itself becomes part of the deterministic input stream.

Research questions. RQ1: what formal attributes must the per-event algorithm possess for the FPGA substrate to dominate (in throughput and/or cost) an in-memory software projection, and under what conditions does it dominate general stream frameworks (Kafka Streams, Apache Flink, Apache Spark)? RQ2: how should the system be organized – the serialization layer, the FPGA core, and the surrounding application – and which principles and protocols should govern the implementation?

The remainder of the article is organized as follows: Section 2 analyses recent research and publications; Section 3 states the aim, tasks, and methods of the research; Section 4 formalizes the applicability conditions (RQ1); Section 5 specifies the architecture (RQ2); Section 6 presents the quantitative models and results; Section 7 discusses use cases, the framework boundary, and threats to validity; Section 8 concludes.

ANALYSIS OF RECENT RESEARCH AND PUBLICATIONS

Software stream processing. Kafka Streams, Apache Flink, and Spark Structured Streaming provide keyed state, windowing, and exactly-once semantics over partitioned logs [4], [5]. Their per-core throughput for keyed stateful operations is bounded by (de)serialization, state-store access (frequently RocksDB with the write amplification of log-structured merge trees), and changelog replication; benchmark studies place framework-class throughput at 10^5-10^6 records per second per core, with end-to-end latencies from milliseconds (Flink, Kafka Streams) to micro-batches of 100 ms and above (Spark) [6], [7], [8]. Google's fleet-wide profiling attributes 22-27 % of all warehouse cycles to the "datacenter tax" – remote procedure calls, (de)serialization, compression, hashing – precisely the work a wire-to-wire hardware path eliminates [9]. The LMAX architecture demonstrates the software optimum for our workload class: a single-threaded business-logic processor over an in-memory working set sustains about 6 million orders per second [10]; we treat this as the best published software serial rate throughout.

FPGA devices in event processing. Hardware feed handlers parse exchange protocols at line rate with microsecond, jitter-free turnaround [11]. Research into query-to-gateware compilers established that select-project-aggregate streams compile to constant-rate circuits [12]. Fleet [13] and recent streaming aggregation engines [14] generalize this to massively parallel streaming; KV-Direct reaches 1.22 billion key-value operations per second per server by moving the update logic onto programmable network interface cards (NIC) [15] – an existence proof for duty 2 at rates three orders of magnitude beyond software. Recent industry implementations have further established fleet-scale FPGA operation and transformation of live traffic [16]. Consensus in a Box [17] runs ZAB-style replication in an FPGA at microsecond latencies, foreshadowing a hardware implementation of the IQPS quorum itself.

Theory imported. We characterize hardware-friendly projections via cost register automata (CRA) and regular functions [18]; conflation soundness is addressed through the theory of the join-semilattices that underlie conflict-free replicated data types [19]; staleness is managed using the established Age of Information (AoI) metrics [20]; checkpoint cadence follows the standard Young–Daly optimization model [21]; queueing behaviour is analysed via the classical Pollaczek–Khinchine and Lindley formulas. The “killer microseconds” argument of Barroso et al. [22] frames why CPU stacks structurally cannot serve the microsecond regime the FPGA occupies.

The analysis shows that each ingredient of the proposed solution – line-rate hardware parsing, in-fabric keyed state, hardware consensus, and conflated distribution – has been demonstrated in isolation, but the literature offers neither formal conditions that determine when the projection layer of a two-layer event-processing architecture should move into hardware, nor a validated set of quantitative models of such a system as a whole. Closing this gap is the subject of the present article.

THE AIM, TASKS, AND METHODS OF THE RESEARCH

The aim of the article is to determine the formal conditions under which the second (projection) layer of the two-layer IQPS architecture should be implemented in reconfigurable hardware, to construct the reference architecture of such an implementation, and to build and validate its quantitative models.

To achieve this aim, the following tasks are solved:

1) formalization of the Layer-2 computation as a per-key state transducer and derivation of the applicability attributes A1-A6 together with the decision conditions D1-D4 relative to CPU scale-out and stream frameworks;

2) specification of the reference architecture – the serialization contract of Layer 1, the internal organization of the FPGA core, and the surrounding host software – together with a ten-principle implementation protocol;

3) construction of six quantitative models M1–M6: throughput ceilings, sojourn-time tails, cost with amortization of non-recurring engineering (NRE) expenses, conflated egress and staleness, checkpoint cadence, and the decision surface;

4) validation of every closed form by discrete-event simulation;

5) determination of the resulting quantitative boundaries between the hardware layer, tuned software, and stream frameworks.

Research methods. The theory of cost register automata and regular functions; the theory of join-semilattices; queueing theory – the M/D/1 and M/G/1 models in Kendall’s notation together with the Pollaczek–Khinchine and Lindley formulas; renewal theory and Age-of-Information analysis; the Young-Daly checkpoint model; discrete-event simulation with batch-means confidence intervals.

Scientific novelty. For the first time, a formal applicability framework is proposed – six attributes A1-A6 with connecting propositions and four decision conditions D1-D4 – that determines exactly which event-processing algorithms benefit from transferring the projection layer into an FPGA. For the first time, an exact closed-form expression is obtained for the egress rate and the mean age of information of a rate-limited last-value cache fed by a Poisson update stream. The

two-layer IQPS architecture is improved by relocating the Layer-2 projection into a hardware core with a fixed-offset serialization contract in Layer 1. The quantitative models of two-layer event processing are further developed for the hardware substrate and confirmed by simulation.

Practical significance. The decision surfaces and break-even curves of Section 6 give practitioners a principled basis for choosing between the hardware layer, tuned software, and stream frameworks; the implementation principles and the verification protocol of Section 5 reduce the engineering risk of the hardware path; the applicability boundary of Section 7 identifies the workload families – auctions and matching, pre-trade risk control, market-data distribution, leaderboards, telecommunication counters, telemetry sketches – for which the approach is justified.

SYSTEM MODEL AND FORMAL APPLICABILITY CONDITIONS (RQ1)

1. The projection as a state transducer

Fix a key (aggregate) space K and a per-key state space S. The Layer-2 computation is a Mealy-style transducer per key:

$$\delta: S \times \Sigma \to S, \quad \omega: S \times \Sigma \to \Gamma^{(\leq c)} \quad , \tag{1}$$

where Σ is the (serialized) input event alphabet; Γ is the reaction alphabet, and c is a constant bound on the number of reactions per input (duty 1 = ω, duty 2 = δ).

The global projection is the keyed product of independent per-key transducers; events of distinct keys commute. This is the standard event-sourcing fold of the command-query responsibility segregation (CQRS) style, restricted per key. Fig. 1 summarizes the model and anchors the attributes introduced below.

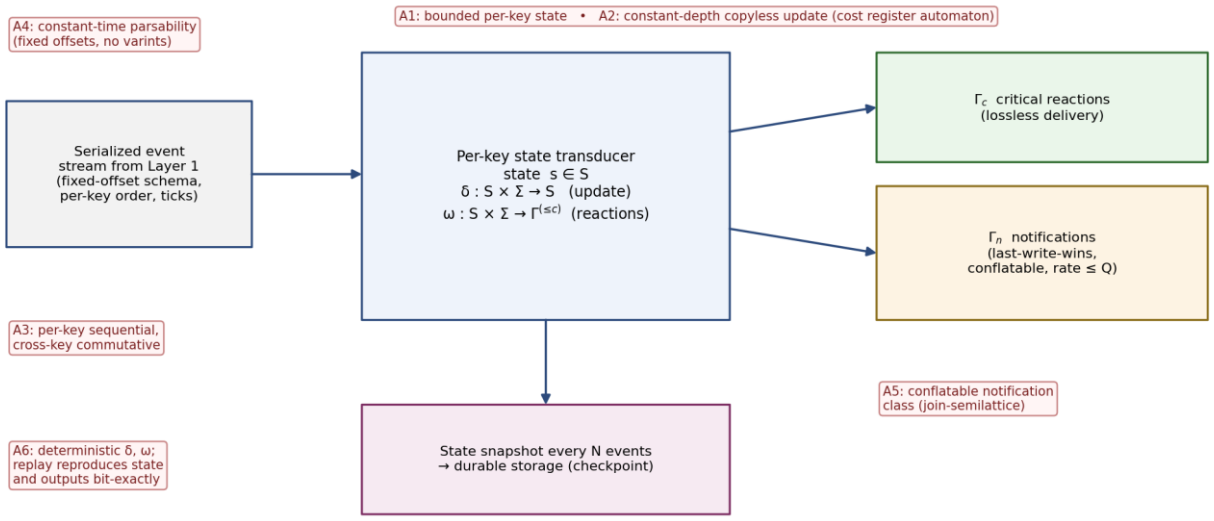


Fig. 1. **The per-key state transducer and the applicability attributes A1-A6**
Source: **compiled by the authors**

A useful yardstick for "how much machinery δ, ω may contain and still be regular" is the **cost register automaton**: finitely many registers over a value monoid, updated by copyless combinations of registers and event fields, with output drawn from registers. CRA capture counters, minimum/maximum/argmax, top-k of bounded k, sliding aggregates over algebraic (invertible or associative) operators, threshold logic, and rank bookkeeping of bounded tables – and exclude unbounded joins, data-dependent iteration, and pointer-linked structures. StreamQRE [18] shows that the same class covers realistic financial computations.

2. Applicability attributes

We say that a Layer-2 workload is **hardware-eligible** when it satisfies the following attributes.

A1 (bounded per-key state). $|enc(S)| \leq C_S$ bits for a compile-time constant C_S, and the live key working set fits the device memory hierarchy: $\Sigma_k |enc(S_k)| \leq 40$ MB for state resident in block or ultra RAM (BRAM/URAM), or ≤ 16 GB for state resident in high-bandwidth memory (HBM) (Alveo U55C figures [23]).

A2 (constant-depth update). δ and ω are expressible as a CRA with copyless updates whose monoid operations have bounded-depth combinational circuits (addition, comparison, multiplexing, hashing, fixed-point multiplication). The update then compiles to a d-stage pipeline with initiation interval II = 1 – one event per clock cycle – independently of event history.

A3 (per-key sequential, cross-key commutative). Correctness requires only per-key event order (the IQPS functional-dependency premise). This licenses key-partitioned parallel pipelines with no inter-pipeline synchronization.

A4 (constant-time parsability). Events admit a fixed-offset binary layout – every field at a static byte offset, variable-length parts bounded and length-prefixed, in the style of Simple Binary Encoding (SBE). Parsing is then wiring plus a few multiplexers, not a computation. (This is what forces serialization into Layer 1.)

A5 (conflatable notification class). Reaction events split into a critical class Γ_c (must all be delivered: Bid Accepted/Bid Rejected) and a notification class Γ_n (rank updates) such that Γ_n per (subscriber, topic) pair carries last-write-wins semantics: the observer's correct belief after receiving a subsequence is determined by the latest element received. Formally, Γ_n values are ordered by generation sequence, and the observer state is a join-semilattice homomorphism image of the latest value [19].

A6 (deterministic and replayable). δ, ω are pure functions of (state, event); all nondeterminism – time, ordering, tie-breaks – is resolved upstream by Layer-1 sequencing and injected tick events. This makes the core a replicated state machine: replay of the log through the same bitstream reproduces states and outputs bit-exactly.

3. Propositions

Proposition 1 (pipelinability). Under A2 and A4, the per-pipeline sustainable rate is $f_{clk}/II = f_{clk}$ events per second, with deterministic latency d/f_{clk}, for any event mix. Sketch: A4 gives a constant-latency decoder; A2 gives a fixed-depth datapath; II = 1 requires resolving the read-after-write (RAW) hazard when consecutive events hit the same key, which the forwarding network of Section 5.2 does in the style of processor bypass paths; absent forwarding, the per-key rate degrades to f_{clk}/L for state-access latency L (quantified in M1).

Proposition 2 (hot-key bound; scale-out invariance). Under A3, horizontal scaling multiplies aggregate capacity but leaves the per-key ceiling at the serial rate μ^{serial} of one executor. If the peak single-key arrival rate $\lambda_{key} > \mu_{sw}^{serial}$, no software deployment of any width meets demand, while a hardware pipeline raises the ceiling by $f_{clk}/\mu_{sw}^{serial}$. This is the formally decisive case for IQPS-F: 250 MHz against 0.1–6 million events per second per software executor (M1) – a 42–2,500-fold serial headroom.

Proposition 3 (conflation soundness). Under A5, any delivery policy that (i) never reorders a (subscriber, topic) channel and (ii) always delivers the latest generated value no later than the next gate opening yields observer states identical to full delivery at every delivery instant. Sketch: last-write-wins semantics mean that the observer function factors through "latest received"; policy (ii) makes "latest received at delivery instants" equal to "latest generated at delivery instants". Dropping intermediate values is therefore unobservable at delivery instants; the only cost is *staleness between deliveries*, which Proposition 4 bounds. Critical-class events are exempt: they are delivered losslessly with per-key sequence numbers.

Proposition 4 (exact conflated egress and staleness). Let updates to one (subscriber, topic) channel arrive as a Poisson process of rate λ and let the conflation gate enforce a minimum spacing $T_{gate} = 1/Q$ with the policy "at gate opening send the freshest pending value; if none is pending, the next arrival is sent immediately".

Then, with $q = \lambda/Q$, the egress rate and the mean age of information at the receiver are:

$$R(\lambda, Q) = \frac{\lambda Q}{\lambda + Qe^{-q}}. \quad (2)$$

$$\bar{A}(\lambda, Q) = E[G] + \frac{E[X^2]}{2E[X]}. \quad (3)$$

$$E[G] = \frac{1 - e^{-q}}{\lambda} - \frac{e^{-q}}{Q}, \qquad E[X] = \frac{1}{Q} + \frac{e^{-q}}{\lambda}, \qquad E[X^2] = \frac{1}{Q^2} + \frac{2e^{-q}}{\lambda Q} + \frac{2e^{-q}}{\lambda^2}, \quad (4)$$

where X is the inter-send interval, G is the age of the carried value at send time, and $\bar{A}$ is the time-average age of information at the receiver (propagation delay excluded). Asymptotically $\bar{A} \to 1/\lambda$ as $\lambda \ll Q$ (every update is delivered fresh), and $\bar{A} \to 1/(2Q) + 1/\lambda$ as $\lambda \gg Q$. Sketch: sends are renewal points by memorylessness of the Poisson stream; $X = 1/Q + Y$ with $Y = 0$ with probability $1 - e^{-q}$ and $Y \sim Exp(\lambda)$ otherwise; G is an $Exp(\lambda)$ age truncated at the gate width, independent of the subsequent X; the age-of-information sawtooth then gives the stated renewal-reward ratio [20]. Simulation confirms both expressions to within 1.3 % across $\lambda/Q \in [0.1, 10^3]$ (Section 6, M4).

Corollary (egress hard cap). Total notification egress is bounded by $\Sigma_{channels} R \leq M_{sub} Q$ regardless of input load – the property that makes duty 4 a load-shedding guarantee, not an optimization.

Proposition 5 (replay equivalence). Under A6, a snapshot at sequence number s plus replay of events $s + 1..n$ through the same core is bit-equivalent to uninterrupted execution to n; consequently, a consistent global checkpoint is just a per-partition sequence-number cut, and no Chandy–Lamport-style marker protocol is required. This is why the IQPS rule "checkpoint every N events" survives unchanged in hardware, and why Flink-style barrier alignment is unnecessary within a partition.

4. Formal comparison conditions versus software deployments

Let λ denote the aggregate arrival rate, λ_{key} the peak per-key rate, L_{p99} the tail-latency service-level objective (SLO), W the live state size in bytes, and let μ_{sw}, μ_{hw} denote per-executor service rates. IQPS-F **strictly dominates** software Layer 2 when A1-A6 hold and at least one of the following conditions is met:

D1 (serial ceiling): $\lambda_{key} > \rho_{max} \mu_{sw}^{serial}$ – no software width suffices (Proposition 2);

D2 (tail SLO): the required L_{p99} lies below the software floor (approximately 10-50 μ_s for kernel-bypass-tuned code, milliseconds for framework-class systems; M2 quantifies the pause-driven divergence of the 99.9th percentile);

D3 (fleet economics): λ exceeds the cost break-even λ^* of M3 (≈ 3.1 million events per second against framework-class software at 2026 cloud prices, host included);

D4 (egress physics): the required notification fan-out $\lambda_{key} \times M_{sub}$ exceeds what any single software node can even generate, while the conflation cap $M_{sub} Q$ fits one card's egress (the 10^4-fold reduction of M4).

Conversely, software (Kafka Streams, Flink, Spark) remains the right substrate when any of A1-A6 fails or none of D1-D4 holds – concretely: state beyond device memory or with pointer-rich structure (A1/A2 fail); evolving business logic whose release cadence outpaces bitstream qualification; ad-hoc or dynamically optimized queries, large joins, iterative or machine-learning workloads (Spark's home turf [5]); rates ≤ ~1 million events per second with SLO ≥ ~10 ms, where a handful of cores costs less than the cheapest FPGA path (M3); and organizations for which the NRE and the scarcity of register-transfer-level (RTL) and verification skills dominate (Section 7). Spark Structured Streaming in particular is excluded from every sub-100-ms regime by its micro-batch floor [5], but remains superior for large windowed analytics over the same event log – a natural division of labour in which both consume the Layer-1 archive.

In summary, hardware eligibility is a checkable contract: the six attributes of Fig. 1 determine whether the projection compiles to a constant-rate circuit, and the four decision conditions determine whether doing so pays. The next section turns the contract into an architecture.

ARCHITECTURE OF THE HARDWARE SECOND LAYER (RQ2)

1. Layer 1: routing and serialization

Layer 1 terminates client protocols, validates and authenticates, and then performs four hardware-facing duties; the general organization is shown on the left of Fig. 2.

Serialization contract. Events are encoded in a fixed-offset binary schema in the style of FIX Simple Binary Encoding: little-endian scalar fields at static offsets, enumerations as integers, bounded strings length-prefixed, one schema/version word up front. The schema is the single source of truth, compiled twice from one definition file: to Layer-1 codecs (Java, C++, JavaScript) and to a VHDL record with a decoder entity. This co-generation principle eliminates the classic hardware/software drift failure mode. Self-describing formats (JSON, Avro with schema resolution, Protobuf with variable-length integers) are explicitly excluded on the hot path: variable-length decoding and optional fields reintroduce data-dependent control flow, violating A4; the datacenter-tax measurements [9] indicate what re-introducing them would cost even in software.

Sequencing. A per-partition sequencer assigns dense sequence numbers (p, s), establishing the total per-key order that A3 and A6 require. The sequenced stream is simultaneously (i) multicast toward Layer 2 and (ii) appended to the durable archive (a Kafka or Redpanda topic, or Aeron Archive [24]) — the archive, not the FPGA, is the source of truth, exactly as in IQPS.

Virtual time. Layer 1 injects tick events at a fixed cadence (e.g., 1 ms) through the same sequencer. All temporal logic in the core – auction deadlines, the conflation timers used for the T_{emit}/T_{ckpt} alternatives, timeouts – keys off tick counts, never a wall clock, preserving the replay equivalence of Proposition 5. (This is the hardware analogue of the watermark and trigger discipline of the Dataflow model.)

Transport. Preferred: User Datagram Protocol (UDP) multicast with redundant A/B feeds and retransmission by negative acknowledgements (NACK) served from the archive – the exchange-proven pattern – either over a switch to network-attached cards or over PCI Express (XDMA/QDMA engines) when the card is host-attached (the case of the F2 cloud instances, which expose no direct network pins to the FPGA). Kernel-bypass techniques (DPDK, AF_XDP) or Aeron on the Layer-1 side keep the software half of the path off the kernel's tail-latency floor [22].

2. The FPGA core

The core (target: AMD UltraScale+ or Versal families – e.g., the Alveo U55C card with two 100-Gigabit-Ethernet ports and 16 GB of HBM [23], or the VU47P device of the F2 instances) is organized as a feed-forward AXI4-Stream dataflow (Fig. 2); every block is a separately verifiable entity with a static-latency contract.

Ingress and decode. The 100-Gigabit media-access controller (MAC) → UDP/IP filter → SBE decoder (A4: pure wiring plus multiplexers, 2-3 cycles) → sequence-gap detector (gaps trigger NACK/replay; the pipeline never processes out of order within a partition).

Key dispatch. A hash of the aggregate identifier based on a cyclic redundancy check (CRC) selects one of *P* update pipelines (A3); a per-pipeline skid FIFO absorbs transient imbalance; credit counters propagate backpressure to the MAC, where only notification-class traffic may ultimately shed (critical events are lossless by construction – the archive retransmits).

Update pipeline (duties 1-2). Per pipeline: state lookup in banked URAM/HBM (a cuckoo-hashed key table for sparse key spaces; direct indexing for dense ones), a d-stage δ/ω datapath generated from the CRA description, and a write-back stage. Back-to-back events on the same key are handled by a forwarding (bypass) network comparing in-flight keys across stages – the standard solution of the RAW hazard that keeps II = 1 (Proposition 1); designs without forwarding fall to f_{clk}/L per key, still 10-60 times faster than software (M1). State words carry an error-correcting code (ECC); a parity-scrub finite-state machine (FSM) walks the state banks in idle cycles.

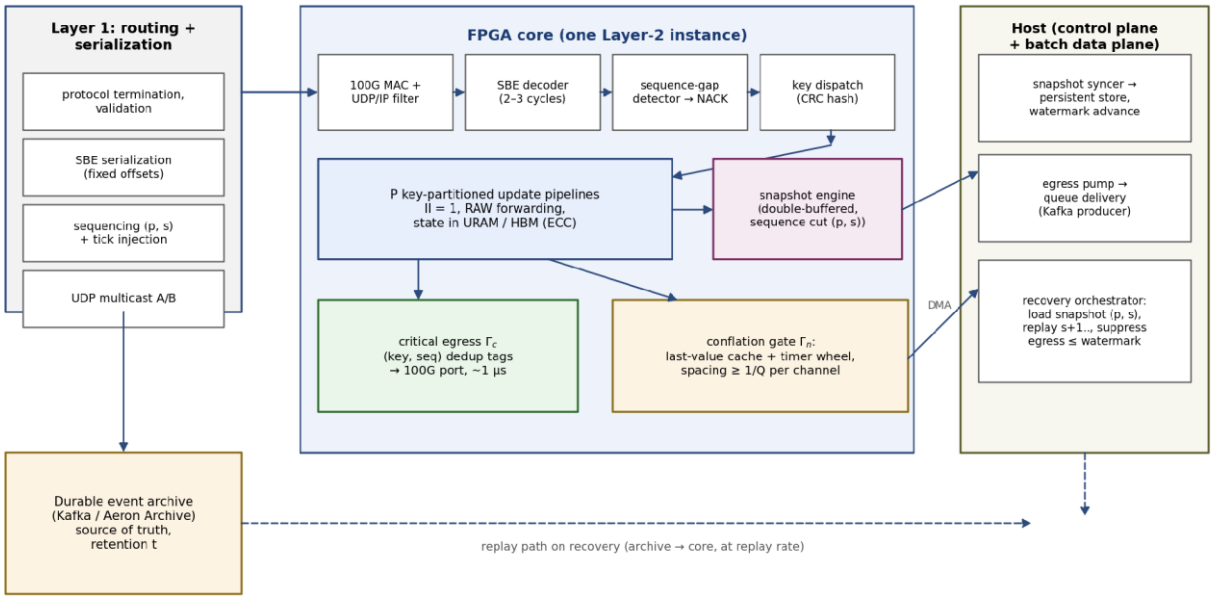


Fig. 2. **The IQPS-F reference architecture: Layer 1, the FPGA core, the host, and the durable archive**

Source: **compiled by the authors**

Dual-class egress (duty 4). The outputs of ω split by class. **Critical** events go directly to an egress serializer and out the second 100-Gigabit port (and, mirrored, into a host DMA ring for queue delivery) with (key, seq) tags for consumer-side deduplication – a latency budget of about 1 µs wire-to-wire. **Notification** events enter the **conflation unit**: a last-value cache (LVC) indexed by (subscriber, topic) in URAM/HBM plus a hierarchical timer wheel that opens each channel's gate at rate Q (Propositions 3-4). Gate openings drain the freshest value into per-destination coalescing buffers flushed every E messages or T_{emit} ticks into the egress ring of direct memory access (DMA) – this joint policy implements the requirement "every E or by timer, and never above Q per channel" verbatim. Both the LVC and the wheel are $O(1)$ per event; capacity is BRAM/HBM-bound (10^4 `channels × 64 B ≈ 640 KB – trivially URAM-resident;` 10^7 channels move to HBM).

Snapshot engine (duty 3). Every N events or T_{ckpt} ticks per partition, the engine snapshots at a sequence cut (Proposition 5): the state banks flip to a shadow copy (double buffering – the pipeline keeps running), and the shadow streams over DMA to a host staging buffer tagged with (p, s). The host synchronizes it to persistent storage (Redis in the IQPS prototype lineage) and only then advances the durable watermark; archive retention t must cover at least the un-checkpointed suffix plus the recovery time, exactly as the IQPS retention constraint prescribes. M5 shows that the bandwidth cost is negligible (< 0.03 % at the Young–Daly cadence), so N is in practice chosen by the recovery-time objective, not by overhead.

Clocking and clock-domain crossing. The MAC, HBM, PCI Express, and core clocks are distinct domains bridged by asynchronous FIFOs; the core clock target (250-300 MHz) is deliberately conservative for timing closure on congested super-logic regions.

Development and verification protocol. The δ/ω datapath is authored in VHDL-2008 (or generated via high-level synthesis (HLS) for faster NRE – M3 models both), but the verification methodology is where correctness lives: (i) a golden software model (the existing Node.js/Redis rules module is the natural seed) executed in lockstep over randomized event logs, with state-hash comparison at every snapshot cut; (ii) constrained-random and corner stimulus (gaps, duplicates, hot-key bursts, tick starvation) under cocotb/UVM; (iii) SVA/PSL assertions for the invariants — per-channel egress spacing $\geq 1/Q$, no critical-event drop, monotone sequence acceptance,

conservation laws of the domain (an accepted bid strictly improves the book); (iv) formal property checking on the conflation gate and the snapshot state machines (bounded, control-dominated — ideal formal targets); (v) replay-equivalence continuous integration: every candidate bitstream must reproduce archived production logs bit-exactly (A6 makes this decisive, cheap, and regression-proof).

3. The surrounding host application

The host CPU beside the card runs no per-event logic; it owns the control plane: bitstream and schema lifecycle, auction and instrument configuration via memory-mapped registers, health and telemetry counters. Its data-plane duties are batch-shaped (Fig. 2, right): the **snapshot syncer** (staging buffer → persistent store → watermark advance), the **egress pump** (DMA rings → a Kafka producer for queue delivery; the direct 100-Gigabit path serves latency-critical subscribers), and the **recovery orchestrator**: on restart or failover, load the newest durable snapshot (p, s), replay the archive from $s + 1$ at the card's replay rate with egress suppressed below the previously published watermark (duplicates are additionally guarded by (key, seq) deduplication at consumers — the idempotence discipline every event-sourced consumer needs anyway). High availability follows the exchange pattern rather than the database pattern: a warm standby card consumes the same multicast feed with egress suppressed; failover is a permission flip, not a state transfer; the IQPS quorum layer (checkpoint agreement of m_2 of m replicas) composes unchanged on top, and Consensus in a Box [17] indicates that even that vote could eventually move into fabric.

4. Implementation principles (P1-P10)

P1: a fixed-offset co-generated schema (A4). P2: a single writer per key – one pipeline owns a key, no cross-pipeline locks (A3). P3: all nondeterminism upstream – sequence numbers and ticks, never wall clocks in fabric (A6). P4: the archive is the source of truth; the card is a cache of the fold. P5: snapshots are sequence cuts, taken without stalling (Proposition 5). P6: egress is idempotent and deduplicable – (key, seq) on every message. P7: two egress classes with a machine-checked safety property – critical lossless, notifications conflatable with spacing $\geq 1/Q$ (Propositions 3-4). P8: credit-based backpressure end-to-end; loss, when forced, lands only on the conflatable class. P9: everything bounded at compile time – state, queues, loop trip counts (A1/A2). P10: golden-model replay equivalence as the release gate.

In summary, the architecture concentrates all per-event work in a feed-forward fabric pipeline whose blocks carry static-latency contracts (Fig. 2), keeps the archive – not the card – as the source of truth, and pushes every source of nondeterminism upstream; principles P1-P10 and the replay-equivalence gate make the hardware path auditable and regression-proof.

QUANTITATIVE MODELS AND RESULTS

All models are implemented in an accompanying Python software package (NumPy and Matplotlib; a custom exact Lindley-recursion simulator, renewal simulators of conflation, and a vectorized failure-renewal checkpoint simulator; pinned random seeds). Every closed form is validated against discrete-event simulation over 26 checks. Headline validation: the formulas (2)-(4) of Proposition 4 agree with simulation to within 1.3 % over $\lambda/Q \in [0.1, 10^3]$; all Pollaczek-Khinchine means fall inside batch-means 95 % confidence intervals; the Monte-Carlo checkpoint optimum matches the Young-Daly value within the flatness of its objective (overhead at either optimum below 0.03 %).

Parameterization discipline. Hardware numbers are datasheet and price-list values [23], [25], [26]; software tiers bracket the published range: framework-class 10^5 events per second per core [6], [7], tuned in-memory 10^6 (interpolation), LMAX-class 6×10^6 [10]. Sizing uses utilization $\rho \leq 0.7$ for tail headroom. Every assumption is tagged in the package parameters, and sensitivity to it is one edit away.

1. M1– throughput ceilings

A 100-Gigabit-Ethernet port carries 96.2 million UDP events per second at a 64-byte payload (148.8 million at minimum frame; 64.4 million at 128 bytes) – so a single pipeline at 250 MHz (II = 1) is ingress-bound, not compute-bound; four pipelines saturate both ports of a U55C card with a four-fold headroom. The per-key serial ceilings (Proposition 2) are: framework-class 0.1 million per second, tuned 1 million per second, LMAX-class 6 million per second, FPGA without forwarding 62.5 million per second, FPGA with forwarding 250 million per second – a 42-fold jump over the best software even for a single hot auction, and 2,500-fold over framework-class (Fig. 3). The 180 million key-value operations per second per NIC measured for KV-Direct [15] independently corroborate the order of magnitude.

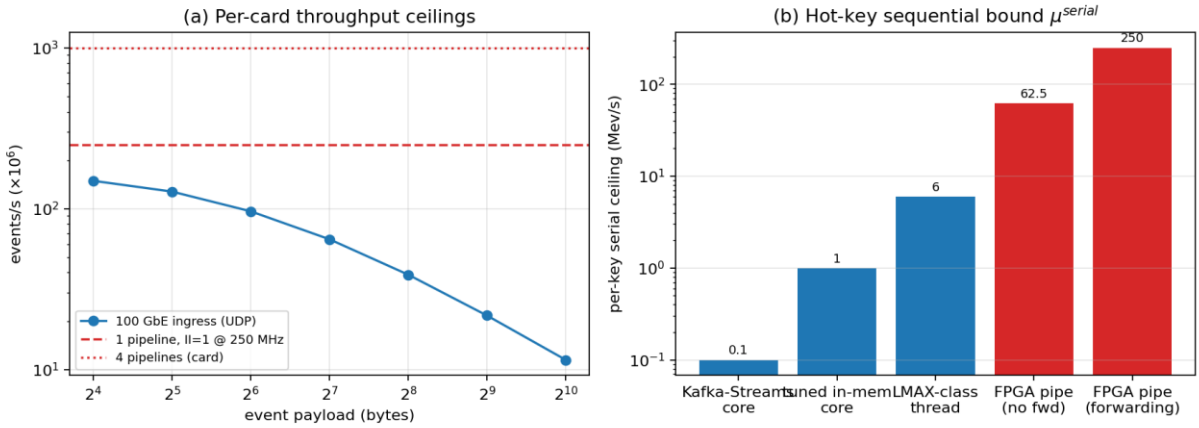


Fig. 3. **Throughput ceilings**
Source: **compiled by the authors**

2. M2 – sojourn-time tails

Layer-2 service is modelled per partition as M/D/1 for the FPGA (a deterministic 0.8 µs wire-to-wire, the mid-range of published handlers) and M/G/1 for software: a lognormal body (mean 2 µs, coefficient of variation 1.2) plus a rare-pause mixture (probability 5×10^{-5}, mean 20 ms – garbage collection and compaction). At $\rho = 0.7$: the FPGA median is 1.4 µs and the 99.99th percentile is 11.4 µs; the software median is 8.9 µs, but the 99.99th percentile is 200 ms – four orders of magnitude, driven entirely by pause-induced busy periods that no horizontal scaling removes (they recur on every node). This is condition D2 made quantitative: if the SLO speaks in microseconds at the 99.9th percentile, the software curve is not on the chart (Fig. 4). The Pollaczek-Khinchine means match simulation within the confidence intervals at every ρ (the software mean at $\rho = 0.9$ carries a ±43 % confidence interval from 400 pause samples – reported, not hidden).

3. M3 – cost and the non-recurring-engineering frontier

Cloud, on-demand, sized at $\rho \leq 0.7$ (c7i at \$0.0446 per vCPU-hour [26]; f2.6xlarge, one VU47P plus a 24-vCPU host, \$1.98 per hour [25]): the F2 line undercuts framework-class software above $\lambda^* \approx 3.1$ million events per second, undercuts tuned software above ≈ 31 million events per second, and never undercuts LMAX-class software within one card's ingress bound – an honest boundary: against hyper-optimized software, the FPGA case rests on D1/D2/D4 (hot keys, tails, fan-out) and, in on-premises terms, on energy (a U55C at a 165 W wall delivers ≈ 0.4-0.8 million events per second per watt against ≈ 0.01 million for framework-class cores – a 35-70-fold advantage; against the marginal core power of LMAX-class software it is rough parity, 0.6-1.2–fold, though server-level idle floors favour the card). On premises, NRE dominates: at \$240,000 (hand-written VHDL) the break-even against tuned software sits at 6×10^8 events per second even at 60-month amortization, while at \$80,000 (HLS) it falls to $\approx 1.8\times10^8$ events per second at 56 months (Fig. 5) – the quantitative form of the standard advice to prototype in HLS and hand-craft only the proven hot datapath. Deployments below ~10^8 events per second justify hardware on D1/D2/D4, not on raw dollars per throughput.

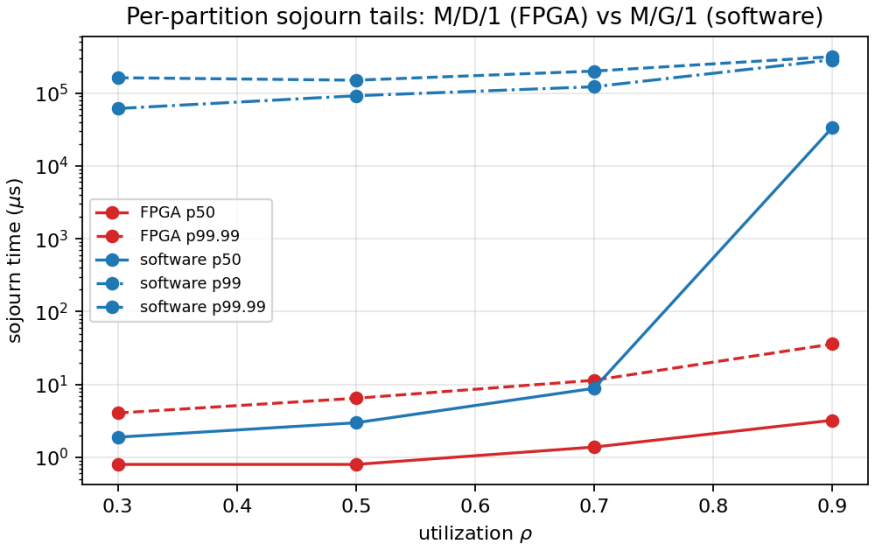


Fig. 4. **Sojourn-time tails**
Source: **compiled by the authors**

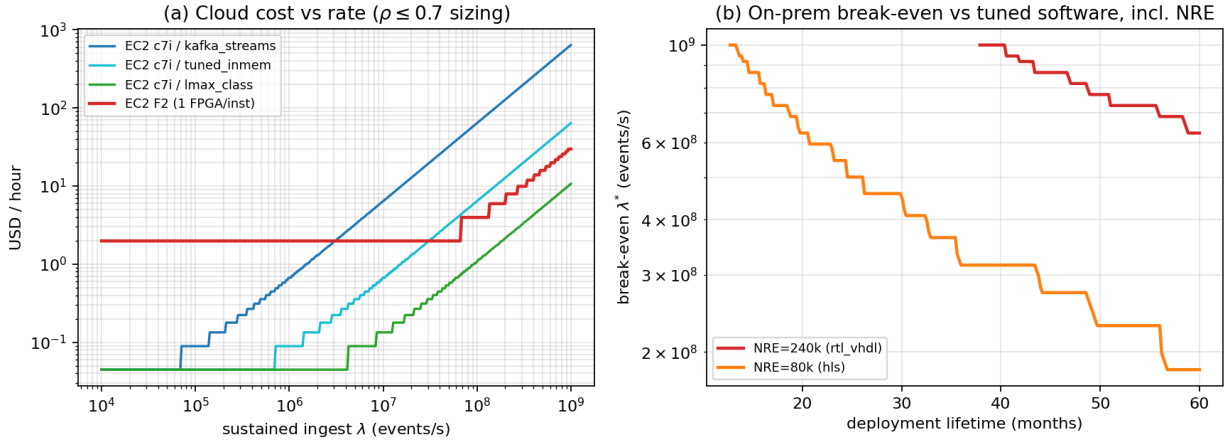


Fig. 5. **Cost versus rate; the non-recurring-engineering frontier**
Source: **compiled by the authors**

4. M4 – conflation: egress and staleness

The formulas of Proposition 4, validated to within 1.3 %: egress saturates at the hard cap $M_{sub}Q$ (Corollary), and the mean staleness collapses to $1/(2Q) + 1/\lambda$ under overload (Fig. 6). For the reference auction (10^4 participants, 10^5 bits per second, $Q = 10$ per second – conservative relative to exchange practice [2]): naive rank fan-out would demand 10^9 messages per second ≈ 512 Gbit/s of egress (infeasible for any single node of any technology); the conflation unit delivers the same observer-visible ranks at every delivery instant (Proposition 3) in 10^5 messages per second ≈ 0.05 Gbit/s – a 10,000-fold reduction – with a mean staleness of 50 ms and the load-independent guarantee that no participant ever receives more than 10 updates per second. Choosing Q becomes a principled trade: $A \approx 1/(2Q)$ under load, so a 100 ms staleness budget implies $Q = 5$ per second per channel and an egress cap of $5M_{sub}$ messages per second.

5. M5 – snapshot cadence and recovery

With double-buffered snapshots at a DMA share of $b = 8$ GB/s, the checkpoint cost $\delta = S/b$ is 8 ms (64 MB of state) to 2 s (16 GB). The Young-Daly cadence at a mean time between failures (MTBF) of 6 months gives $T^* = 502$ s (64 MB) to 7,940 s (16 GB) with overhead ≤ 0.03 % – i.e., failure-optimal checkpointing is essentially free, and the binding constraint is the recovery time instead:

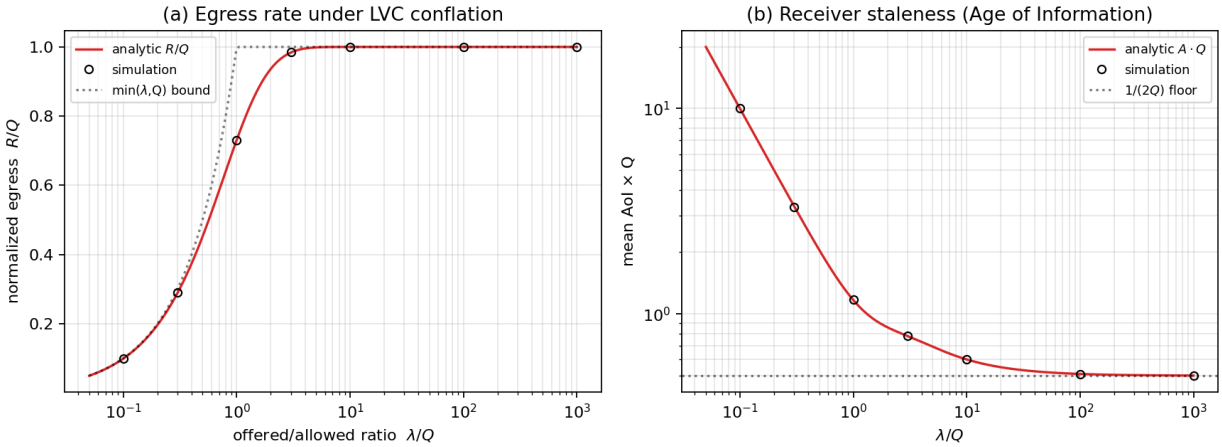


Fig. 6. **Conflation egress and the age of information**
Source: **compiled by the authors**

$$R(T) = t_{fail} + \frac{S}{b} + \frac{\frac{\lambda T}{2}}{\mu_{replay} - \lambda}. \quad (5)$$

At $\lambda = 10^7$ events per second, 4 GB of state, and a replay rate of 100 million events per second, a 60 s recovery-time objective (RTO) forces $T \leq 531$ s ($N \leq 5.3 \times 10^9$ events) – two orders below the Young-Daly value of 3,971 s (Fig. 7).

The resulting rule is:

$$N = \lambda \cdot min(T_{Daly}, T_{RTO}) \quad (6)$$

with hardware, the binding term is always T_{RTO}. Warm-standby failover (Section 5.3) removes replay from the availability path entirely, leaving these bounds for cold restarts. The IQPS quorum-checkpoint protocol consumes the exported (p, s) −tagged snapshots unchanged.

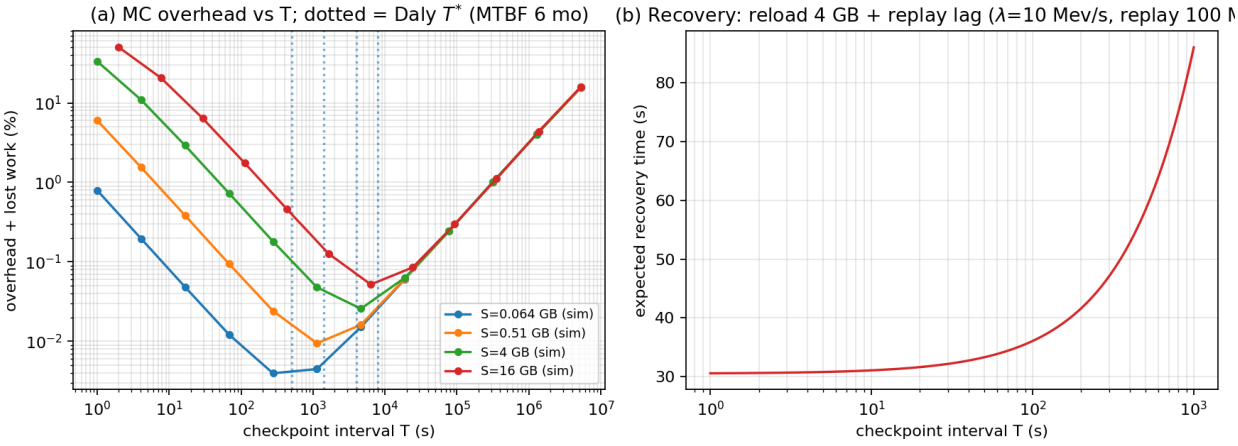


Fig. 7. **Checkpoint overhead and recovery**
Source: **compiled by the authors**

6. M6 – the decision surface

Superimposing D1 (per-key ceilings) and D2 (tail floors) yields the deployment map of Fig. 8. Illustrative placements: a typical software-as-a-service aggregate (10^4 events per second, 50 ms) sits deep in “any software”; the reference hot auction (10^5 events per second on one key, 1 ms) already exceeds a framework-class core’s ceiling – tuned software or hardware; a single level-2 order book at 5×10^6 events per second with a 50 µs 99th percentile crosses into LMAX-class-or-FPGA; a pre-trade risk gate (5×10^7 events per second, 5 µs) is FPGA-only. The map’s message is the thesis of

the article in one picture: scale-out moves a system to the right along the aggregate axis but never along the per-key or tail axes; only a faster serial executor does.

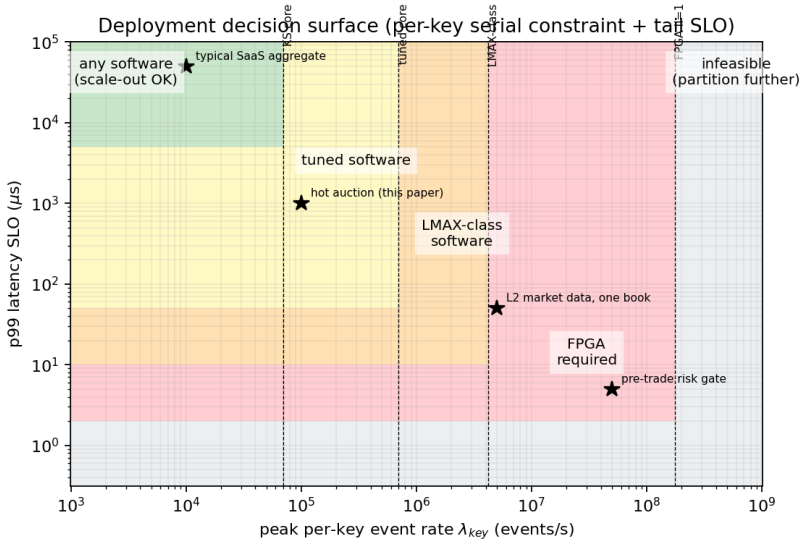


Fig. 8. **The decision surface**
Source: **compiled by the authors**

Taken together, models M1-M6 give the quantitative answer to RQ1: the hardware layer changes the two axes that horizontal scaling cannot – the per-key serial ceiling and the latency tail – caps egress by construction, makes failure-optimal checkpointing free, and beats framework-class software on cost above approximately 3.1 million events per second.

DISCUSSION: USE CASES, THE FRAMEWORK BOUNDARY, AND THREATS TO VALIDITY

1. Use cases and the framework boundary

The attributes of Section 4 pick out a recognizable family of workloads, each anchored to the attribute that qualifies it. Auction and matching engines (this article's running example and the industry's founding use case) satisfy all six attributes, with D1 activated by hot instruments and D4 by participant fan-out. Pre-trade risk gates (checks in the style of SEC Rule 15c3-5) are pure A2 threshold automata under D2 SLO of single-digit microseconds. Market-data distribution with per-subscriber entitlements and conflation is duty 4 sold as a product – existing exchange practice [2], [3] validates both the semantics and the Q values. Real-time leaderboards and matchmaking in gaming are rank projections under massive fan-out (A5/D4) with modest state (A1). Per-flow charging in telecommunications and usage counters of the 5G user-plane function are A1/A2 counters at D1-scale per-flow rates, already adjacent to deployed SmartNIC practice [16]. Network telemetry sketches (heavy hitters, detection of distributed denial-of-service attacks) are the canonical constant-space streaming algorithms – CRA over approximate monoids [14]. Budget pacing in real-time-bidding advertisement auctions shares the auction shape at 10-100 ms SLO; it qualifies on D4/D3 rather than D2. What unifies the family: small algebraic state, fixed logic, per-key ordering, punishing rates or tails, and notification streams whose consumers only care about the latest value.

The boundary with frameworks is equally crisp in the other direction. Kafka Streams and Flink keep the general case: joins across streams, dynamic topologies, large or pointer-rich state, evolving logic, millisecond-tolerant SLO – and they compose with IQPS-F, consuming the same Layer-1 archive for everything that is not the hot fold. Spark remains the substrate for retrospective analytics over the identical event history.

The system-design consequence is pleasant: IQPS-F does not replace the streaming estate; it extracts the small deterministic kernel where physics (per-key serialism, tail floors, fan-out bandwidth) rather than throughput accounting decides, and returns everything else to software.

Two honest caveats from M3 deserve emphasis. First, against genuinely LMAX-class software the FPGA never wins on cloud dollars per throughput inside one card's ingress bound; its wins there are the 42-fold hot-key ceiling, the deterministic tail, the egress cap, and (on premises) energy. Second, NRE is the controlling economic variable below ~10^8 events per second: the HLS-first strategy halves the frontier, and the fixed-schema/CRA discipline of Section 4 is precisely what keeps the register-transfer-level design small enough for a four-person-month HLS budget to be plausible.

2. Threats to validity

Model risk. M/D/1 and M/G/1 with a two-component service mixture are deliberate simplifications; real software tails mix garbage collection, page faults, non-uniform memory access, and coordinated omission, and real FPGA ingress jitters at the MAC. The queueing contrast (deterministic against heavy-mixture service) is robust to these details; absolute percentiles are not. Poisson arrivals understate burstiness; all capacity results are stated at $\rho \leq 0.7$ partly for this reason, and the renewal argument of Proposition 4 specifically requires Poisson input — bursty inputs will shift the age-of-information constants (the egress cap $M_{sub}Q$ holds regardless, by construction). **Parameter risk.** Software tiers span a 60-fold range (10^5–6×10^6); conclusions that depend on the tier are always reported per tier. Prices are on-demand snapshots of July 2026; the break-evens scale linearly with them. NRE estimates are industry folklore ranges, not measurements; the frontier plot exposes their leverage. **Scope risk.** We model one card and one partition group; multi-card sharding inherits the Layer-1 analysis of IQPS unchanged, but cross-card key migration and skew re-balancing are future work. Numbers such as 0.8 µs wire-to-wire and 250 MHz at II = 1 are supported by the cited literature for workloads obeying A1-A4, but any specific δ/ω datapath must re-earn them in timing closure – the CRA discipline is what makes that outcome the expected case rather than a hope. **Verification of the models.** All 26 analytic-versus-simulation checks pass (the worst exact-formula error is 1.3 %; queueing means are within the 95 % confidence intervals; the checkpoint optimum lies within the flat region); the residual risk is the fidelity of the models to deployments, not internal inconsistency.

CONCLUSIONS AND FUTURE WORK

Moving the second layer of IQPS into an FPGA is not a blanket accelerator play; it is a contract: bounded state, copyless constant-depth updates, per-key ordering, fixed serialization, last-write-wins notifications, and upstream-resolved nondeterminism (attributes A1-A6). Where the contract holds, the payoffs are structural, not incremental: a per-key serial ceiling 42-2,500 times above software (the one axis scale-out cannot buy), flat five-nines microsecond tails, an egress hard cap $M_{sub}Q$ with provably unchanged observer semantics and staleness $\approx 1/(2Q)$, checkpointing whose cost rounds to zero so that N is set purely by the recovery-time objective, and – beyond about 3.1 million events per second in today's cloud – a lower bill than framework-class software. Where the contract fails, Section 4.4 says so, and the frameworks keep the workload.

The scientific novelty of the obtained results consists in the first formal applicability framework (A1-A6, D1-D4) for a hardware projection layer; the first exact closed form for the egress rate and the receiver age of information of a rate-limited last-value cache under Poisson updates, formulas (2)-(4); the improvement of the two-layer IQPS architecture by a hardware core with a fixed-offset serialization contract; and the further development of the quantitative models of two-layer processing for the hardware substrate, all validated by simulation.

Future work: (i) implementing the reference core against the existing auction prototype and validating M1/M2 on hardware; (ii) compiling the quorum vote itself into fabric [17], completing a hardware IQPS instance; (iii) extending Proposition 4 to bursty (Markovian-arrival) inputs and to

delta-conflation (mergeable increments rather than last-write-wins values); (iv) a compiler from CRA descriptions to VHDL/HLS that makes A2 a checked property rather than a design guideline; (v) partial reconfiguration for online δ/ω upgrades under the replay-equivalence gate.

Conflict of interests: the authors declare that they do not have a conflict of interests, in particular financial, personal, copyright or of any other nature, which could have influenced the research, as well as the results published in this article

Financing: research was carried out without financial support

Data availability: all data are available in digital or graphic form in the main text of the manuscript; the implementations of the models, the parameters of the experiments, and the generated graphic materials are available from the corresponding author upon reasonable request

Use of artificial intelligence tools: during the preparation of the manuscript the authors used the artificial-intelligence tool Claude (Anthropic) for grammatical and stylistic checks of the text and for verification of the formatting. All fragments processed by the tool were checked and edited by the authors. Artificial intelligence was not used to generate the scientific provisions, models, formulas, results or conclusions; the scientific provisions, results and conclusions are the authors' own intellectual contribution

REFERENCES

1. "MDP 3.0 – Conflation Processing – BrokerTec and EBS Markets Examples". *CME Group Client Systems Wiki*. 2026. – Available from: https://cmegroupclientsite.atlassian.net/wiki/spaces/EPICSANDBOX/pages/457572870. –[Accessed: 2026.05.15].

2. "The conflation mechanism in RTO: 3 updates per second". *Global Directory*. 2022. – Available from: https://community.developers.lseg.com/discussion/89652/the-conflation-mechanism-in-rto-3-updates-per-second. – [Accessed: 2026.05.17]

3. Carbone, P., Ewen, S., Fóra, G., Haridi, S., Richter, S. & Tzoumas, K. "State management in Apache Flink: consistent stateful distributed stream processing". *Proceedings of the VLDB Endowment*. 2017; 10 (12): 1718–1729, https://www.scopus.com/pages/publications/85028448839. DOI: https://doi.org/10.14778/3137628.3137651.

4. Armbrust, M., Das, T., Torres, J., Yavuz, B., Zhu, S., Xin, R., Ghodsi, A., Stoica, I. & Zaharia, M. "Structured Streaming: a declarative API for real-time applications in Apache Spark". *Proceedings of SIGMOD*. 2018. p. 601–613. DOI: https://doi.org/10.1145/3183713.3190664.

5. Henning, S., Vogel, A., Leichtfried, M., Ertl, O. & Rabiser, R. "ShuffleBench: a benchmark for large-scale data shuffling operations with distributed stream processing frameworks". *Proceedings of the ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2024. p. 2–13. DOI: https://doi.org/10.1145/3629526.3645036.

6. Karimov, J., Rabl, T., Katsifodimos, A., Samarev, R., Heiskanen, H. & Markl, V. "Benchmarking distributed stream data processing systems". *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 2018. p. 1507–1518. DOI: https://doi.org/10.1109/ICDE.2018.00169.

7. Henning, S. & Hasselbring, W. "Theodolite: scalability benchmarking of distributed stream processing engines in microservice architectures". *Big Data Research*. 2021; 25: 100209. DOI: https://doi.org/10.1016/j.bdr.2021.100209.

8. Kanev, S., Darago, J. P., Hazelwood, K., Ranganathan, P., Moseley, T., Wei, G.-Y. & Brooks, D. "Profiling a warehouse-scale computer". *Proceedings of the International Symposium on Computer Architecture (ISCA)*. 2015. p. 158–169. DOI: https://doi.org/10.1145/2749469.2750392.

9. Fowler, M. "The LMAX architecture". *martinfowler.com*. 2011. – Available from: https://martinfowler.com/articles/lmax.html. – [Accessed: 2024.05.12].

10. Leber, C., Geib, B. & Litz, H. "High frequency trading acceleration using FPGAs". *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL)*. 2011. p. 317–322. DOI: https://doi.org/10.1109/FPL.2011.64.

11. Mueller, R., Teubner, J. & Alonso, G. "Streams on wires: a query compiler for FPGAs". *Proceedings of the VLDB Endowment*. 2009; 2 (1): 229–240. DOI: https://doi.org/10.14778/1687627.1687730.

12. Thomas, J., Hanrahan, P. & Zaharia, M. “Fleet: a framework for massively parallel streaming on FPGAs”. *Proceedings of ASPLOS*. 2020. p. 639–651. DOI: https://doi.org/10.1145/3373376.3378495.

13. Geethakumari, P. R. & Sourdis, I. “Enthuse: efficient adaptable high-throughput streaming aggregation engines”. *arXiv*. 2024. DOI: https://doi.org/10.48550/arXiv.2405.18168.

14. Li, B., Ruan, Z., Xiao, W., Lu, Y., Xiong, Y., Putnam, A., Chen, E. & Zhang, L. “KV-Direct: high-performance in-memory key-value store with programmable NIC”. *Proceedings of SOSP*. 2017. p. 137–152. DOI: https://doi.org/10.1145/3132747.3132766.

15. Firestone, D., et al. “Azure accelerated networking: SmartNICs in the public cloud”. *Proceedings of NSDI*. 2018. p. 51–66. – Available from: https://www.usenix.org/conference/nsdi18/presentation/firestone.

16. István, Z., Sidler, D., Alonso, G. & Vukolić, M. “Consensus in a Box: inexpensive coordination in hardware”. *Proceedings of NSDI.* 2016. p. 425–438, https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/istvan.

17. Mamouras, K., Raghothaman, M., Alur, R., Ives, Z. G. & Khanna, S. “StreamQRE: modular specification and efficient evaluation of quantitative queries over streaming data”. *Proceedings of PLDI.* 2017. p. 693–708. DOI: https://doi.org/10.1145/3062341.3062354.

18. Shapiro, M., Preguiça, N., Baquero, C. & Zawirski, M. “Conflict-free replicated data types”. *Proceedings of the Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*. 2011. p. 386–400.

19. Yates, R. D., Sun, Y., Brown, D. R., Kaul, S. K., Modiano, E. & Ulukus, S. “Age of Information: an introduction and survey”. *IEEE Journal on Selected Areas in Communications*. 2021; 39 (5): 1183–1210. DOI: https://doi.org/10.1109/JSAC.2021.3065072.

20. Daly, J. T. “A higher order estimate of the optimum checkpoint interval for restart dumps”. *Future Generation Computer Systems*. 2006; 22 (3): 303–312. DOI: https://doi.org/10.1016/j.future.2004.11.016.

21. Barroso, L., Marty, M., Patterson, D. & Ranganathan, P. “Attack of the killer microseconds”. *Communications of the ACM*. 2017; 60 (4): 48–54. DOI: https://doi.org/10.1145/3015146.

22. AMD. “Alveo U55C data center accelerator card data sheet (DS978)”. 2026. – Available from: https://docs.amd.com/r/en-US/ds978-u55c. – [Accessed: 2026.07.17].

23. “Aeron: high-performance messaging and archival”. *Real Logic*. 2026. – Available from: https://github.com/real-logic/aeron. – [Accessed: 2026.07.17].

24. “Amazon EC2 F2 instances”. *AWS*. 2026. – Available from: https://aws.amazon.com/ec2/instance-types/f2. – [Accessed: 2026-07-17].

25. “Amazon EC2 C7i instances”. *AWS*. 2026. – Available from: https://aws.amazon.com/ec2/instance-types/c7i. – [Accessed: 2026.07.17].

Received: 20.07.2026
Received after revision: 27.08.2026
Accepted: 04.09.2026

DOI: https://doi.org/10.15276/ict.03.2026.35
УДК 004.41:004.75

Умови застосовності, архітектура та кількісні моделі апаратного другого рівня у дворівневій розподіленій обробці подій

Купріянов Олександр Миколайович[1)]
ORCID: https://orcid.org/0009-0003-8155-1321; kupriyanov.a062@stud.op.edu.ua. Scopus Author ID: 60718473000
Максименко Андрій Олександрович[1)]
ORCID: https://orcid.org/0000-0003-3117-1657; maksymenko.a.o@op.edu.ua. Scopus Author ID: 57215216446
[1)] Національний університет «Одеська політехніка», просп. Шевченка, 1. Одеса, 65044, Україна

АНОТАЦІЯ

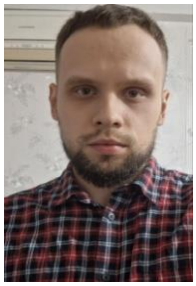
Актуальність. Події, що мають спільний ідентифікатор агрегата, є функціонально залежними: їх слід застосовувати до стану проєкції строго послідовно, тому пропускна здатність одного агрегата обмежена одним послідовним виконавцем, горизонтальне масштабування не підвищує цю межу, а програмні виконавці додатково демонструють «хвости» затримок, спричинені паузами. **Метою статті** є визначення формальних умов, за яких другий (проєкційний) рівень дворівневої розподіленої архітектури обробки подій доцільно обчислювати в реконфігурованій апаратурі, а не в програмному забезпеченні, а також побудова й валідація кількісних моделей такої реалізації. **Завдання:** формалізація проєкції як поключового автомата-перетворювача станів; виведення атрибутів застосовності та умов прийняття рішень; специфікація еталонної архітектури рівня серіалізації, апаратного ядра та супровідного програмного забезпечення хоста; побудова і валідація моделей пропускної здатності, затримок, вартості, ущільненого (конфльованого) вихідного потоку та контрольних точок. **Методи:** теорія автоматів із регістрами вартості та об'єднаних напівграток, теорія масового обслуговування, теорія відновлення, модель контрольних точок Янга–Делі, імітаційне моделювання дискретних подій. **Наукова новизна:** уперше запропоновано формальний апарат застосовності — шість атрибутів зі сполучними твердженнями та чотири умови прийняття рішень, — який точно визначає, які алгоритми обробки подій виграють від апаратного проєкційного рівня; уперше отримано точний замкнений вираз для інтенсивності вихідного потоку та середнього віку інформації кеша останнього значення з обмеженням темпу, що живиться пуассонівським потоком оновлень; удосконалено дворівневу архітектуру з обробкою в оперативній пам'яті шляхом перенесення обчислення проєкції в апаратне ядро з контрактом серіалізації з фіксованими зміщеннями; дістали подальшого розвитку кількісні моделі дворівневої обробки подій для апаратної основи. **Практична значимість:** поверхні прийняття рішень і криві беззбитковості, що дають змогу практикам обґрунтовано обирати між апаратним рівнем, оптимізованим програмним забезпеченням і потоковими каркасами, а також принципи реалізації та протокол верифікації, які знижують ризики апаратного шляху **Результати**: окремий апаратний прискорювач забезпечує обробку великого обсягу подій із короткими пакетами за секунду, суттєво перевершуючи звичайні ядра процесорів та стандартні показники послідовної обробки програмним забезпеченням; він підтримує наднизькі та передбачувані затримки хвоста, повністю усуваючи піки зупинок, притаманні програмному забезпеченню; досягає високої економічної ефективності порівняно з програмними альтернативами в масштабі за сучасних цін у хмарних середовищах; а також суттєво зменшує вихідний трафік розсилки аукціонів за збереження обмеженого середнього показника застарілості даних. **Висновки:** перенесення другого рівня в апаратуру є контрактом із шести атрибутів; там, де контракт виконується, виграш є структурним, а не поступовим, а там, де він порушується, правильною основою залишаються потокові каркаси. Усі аналітичні вирази підтверджено результатами моделювання з нехтовно малою відносною похибкою у найгіршому випадку.

Ключові слова: обробка потоків; програмована логічна інтегральна схема; мова опису апаратури; журналювання подій; розділення відповідальності команд і запитів; конфляція; вік інформації; автомати з регістрами вартості; контрольні точки; Kafka Streams; ринкові дані

ABOUT THE AUTHORS

Oleksandr M. Kupriianov – Graduate Student, Department of Computer Systems. Odesa Polytechnic National University, 1, Shevchenko Av. Odesa, 65044, Ukraine
ORCID: https://orcid.org/0009-0003-8155-1321; kupriyanov.a062@stud.op.edu.ua. Scopus Author ID: 60718473000
***Research field*:** distributed event-processing systems; reconfigurable computing

Купріянов Олександр Миколайович – аспірант кафедри Комп'ютерних систем Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна\
Галузь досліджень: розподілені системи обробки подій; реконфігуровані обчислювальні системи



Andrii O. Maksymenko – PhD, Department of Computer Systems. Odesa Polytechnic National University, 1, Shevchenko Av. Odesa, 65044, Ukraine
ORCID: https://orcid.org/0000-0003-3117-1657;maksymenko.a.o@op.edu.ua. Scopus Author ID: 57215216446
Research field: distributed systems; stream processing architectures

Максименко Андрій Олександрович – доктор філософії (PhD), кафедра Комп'ютерних систем Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна
Галузь досліджень: розподілені системи; архітектури обробки потоків даних