Time-Coherent Analytics at Scale

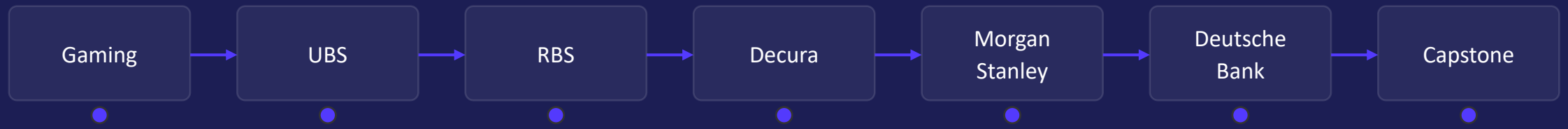
Snapshot barriers, deltas, and compiled valuation plans

Fabien Huré

STAC Summit, London · 29 April 2026

Fabien Huré

Low-latency systems and real-time optimisation, from gaming, through the dotcom era, into finance.



Building valuation infrastructure and real-time systems across asset classes for over 30 years.

contact@fabienhure.com • linkedin.com/in/fabien-huré

THE THESIS

Most ticks do not deserve a rebuild

1 Systems discard coherent state on every cycle

Most analytics stacks treat each valuation cycle as a full invalidation of the entire book. Graph traversal, object rebinding and cache invalidation all fire regardless of actual change.

2 Positions are largely stable intraday

A discretionary trading book at a tier-1 bank holds millions of positions. Proportionally very few change on any given day: new bookings, amends and lifecycle events.

3 Wasted capacity and unstable cycle times

Cycle completion becomes unpredictable. Intra-cycle variance grows under load as more of the graph is touched. The cost is proportional to book size, not to what changed.

FULL REBUILD PER CYCLE

2,000,000

positions on the book

every position repriced in full on every cycle

ACTUALLY CHANGED TODAY

~ 3,000

positions touched today

new bookings + amends + lifecycle events

`0.15% of the book`

THE PROBLEM

The graph has no model of what is stable

1 Traversal and invalidation fire on every cycle

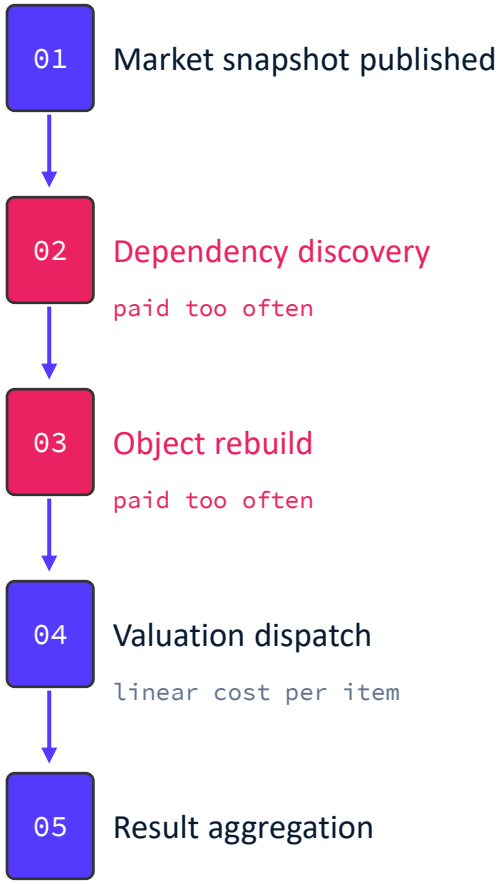
A market data change triggers synchronisation across the entire dependent subgraph. Every node that shares that curve or surface must be invalidated to ensure coherence.

2 Distributed cache makes cycle time unpredictable

Intermediate objects live across the network. Under load, bandwidth and and congestion drive tail cycles: the cost is not bounded and is very difficult to difficult to predict or diagnose.

3 Lazy evaluation doesn't change the economics

A dependency graph can in theory skip unchanged subgraphs. In practice, a single practice, a single curve tick propagates broadly enough that discovery cost cost dominates. Most of the graph is touched anyway.



THE SOLUTION

Represent the coherence the book already has

Graphs belong at compile time and on the slow path. This talk is about removing them from the hot path, not from the architecture.

01 SNAPSHOT BARRIERS

Coherent as-of states

Ingest market data continuously, publish to consumers atomically. Every reader values against a stable, versioned snapshot. Partial updates are never visible mid-cycle.

02 DELTA SETS

Explicit change representation

Carry forward precisely what changed between snapshots. Never Never rediscover at runtime which positions, or market items moved. moved. Represent it once and propagate it explicitly.

03 TWO-SPEED PATH

Optimise for the unchanged majority

The stable book follows a fast path, reusing precomputed structure. New structure. New bookings, amendments and lifecycle events go through a through a controlled slow path, isolated from the fast path cost.

04 COMPILED VALUATION PLANS

Deterministic execution

Resolve product structure and model recipe into a fixed execution plan, execution plan, compiled at booking and reused across every position position that share structure. Runtime executes the plan; it does not not traverse or negotiate.

MECHANISM 01

Snapshot barriers: the EOD → SOD transition

1 Close the book, freeze the world

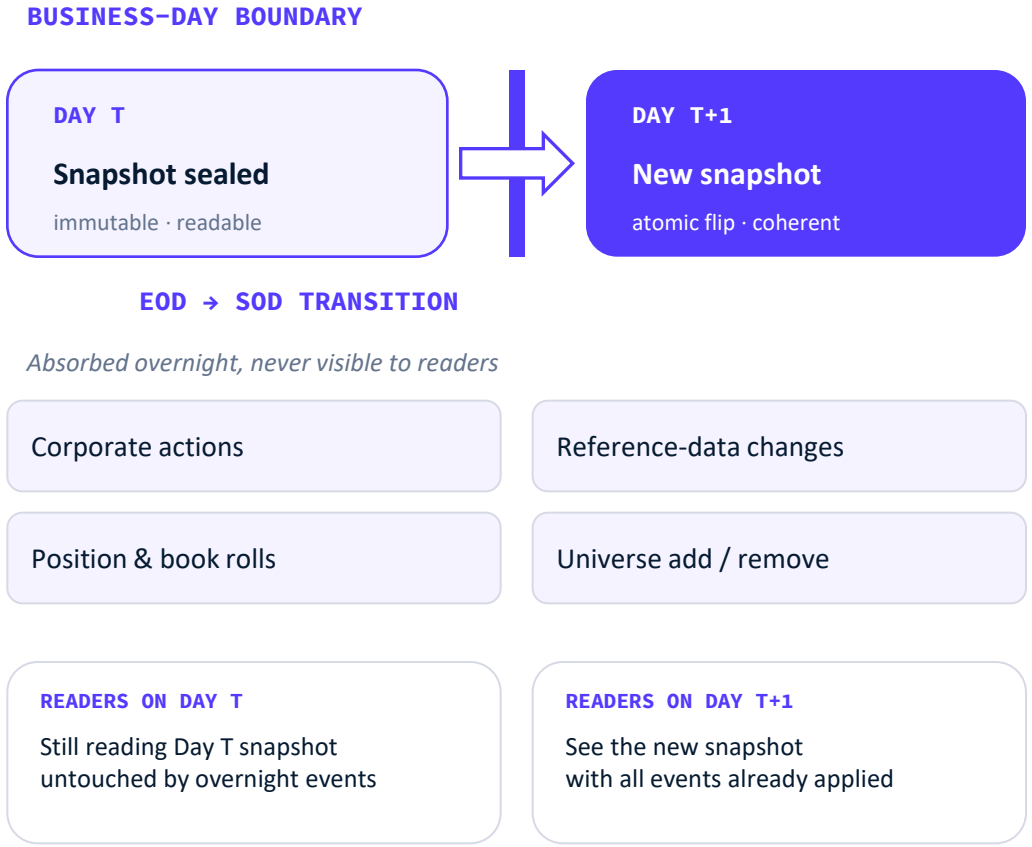
At EOD, Day T's state is sealed as an immutable snapshot. Downstream Downstream consumers keep reading it without interference.

2 Apply lifecycle events overnight

Corporate actions, reference-data changes, book rolls and universe edits are absorbed behind the barrier: never visible mid-day.

3 Publish the new SOD atomically

At cutover, the Day T+1 snapshot becomes visible in one coherent flip. No flip. No half-processed corporate action is ever readable.



MECHANISM 02 + 03

Explicit deltas and precomputed stable state

1 Deltas carry precisely what changed

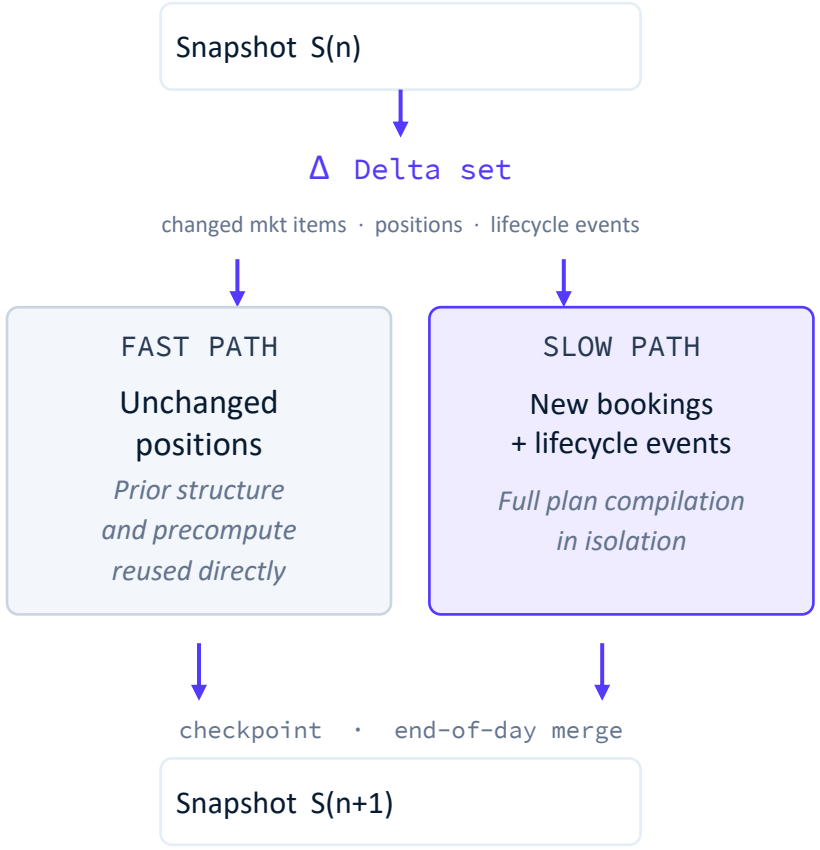
Computed once at snapshot transition, which modelling decision changed, decision changed, which positions was amended, which lifecycle events events fired. Every consumer reads from the same explicit set.

2 Fast path for unchanged, slow path for new

Prior compiled structure and precomputed values are reused directly. New bookings and lifecycle events go to the slow path. Full plan compilation in isolation.

3 Stable layers precomputed before the cycle

Instrument representation, cashflow schedules and model recipes do not recipes do not change on every cycle. Build them once, version them, reuse them, reuse them. Only genuinely market-sensitive work runs on the hot the hot path.



MECHANISM 04

Compiled plans, not runtime graph traversal

1 Compile the plan from product and model recipe

Resolved once at booking and updated at lifecycle checkpoints. A single compiled plan serves every position that shares product structure and model recipe. Fixed before the cycle begins, not negotiated at runtime.

2 Runtime executes a deterministic sequence

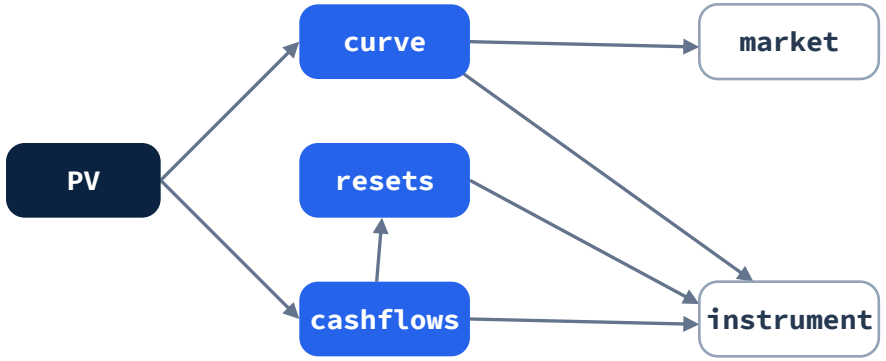
Inputs are fixed, action sequence is fixed, outputs are explicit. The valuation engine runs, it does not traverse, discover or decide.

3 Cycle time becomes bounded and predictable

Bottlenecks are immediately visible rather than emerging unpredictably unpredictably under load. Worst-case cycle time is determined by slow-path slow-path volume, which is explicit.

RUNTIME GRAPH

dependency discovery on every cycle



COMPILED PLAN · IRS EXAMPLE

1	Load snapshot S(n)	LIVE
2	Fixed leg cashflows	PRECOMP
3	Float fixing dates	PRECOMP
4	Apply forecast curves	LIVE
5	Discount + aggregate	LIVE

LIMITATIONS

When this does not help

01 FLUID BOOKS

No overnight coherence to build from

The architecture assumes most of today's book was on the book book yesterday. Intraday-focused books with little overnight overnight warehousing, or books that turn over almost entirely entirely between cycles, offer no stable compiled baseline to carry to carry forward. The fast path is empty. The slow path carries carries everything.

02 CONFIG CHURN

Constant model or configuration change

Compiled plans are stable when product structure and model model recipes are stable. During intensive benchmarking, model model validation, or broad recalibration across product classes, classes, recompilation fires constantly. The architecture is optimised for settled production valuation, not for exploratory exploratory workflows where the model itself is the variable. variable.

This architecture trades higher setup cost at lifecycle events for large steady-state gains on the unchanged majority. unchanged majority. Know which regime your book is in.

Take this with you

01 Publish coherent snapshots atomically

Never expose a partially-built state. Readers always value against a stable, versioned snapshot.

02 Carry explicit deltas, precompute stable layers: keep the hot path thin

Do not rediscover what changed at runtime. Do expensive work at booking and lifecycle checkpoints. The cycle executes only what is only what is genuinely market-sensitive.

03 Compile valuation plans once, execute them brute-force

Plans are compiled once and reused across all positions that share structure. Millions of bookings collapse to a few hundred plans. The cycle runs them brute-force against the current snapshot. No graph walk, no dependency discovery, no "what changed?" at runtime.

These patterns do not add new constraints. Time-coherence is an inherent property of the book between events; the architecture simply stops discarding it.