International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Experimentation as Infrastructure: Designing a Centralized A/B Testing Platform for 500 Million Users

Abhinav Wagle

Abstract: Building valid experimentation infrastructure at internet scale requires solving engineering problems whose solutions cannot be derived from statistical methodology alone. This article documents the architecture and design principles of a centralized Experimentation-as-a-Service (EaaS) platform supporting 500 million users, 1,000+ concurrent experiments, and sub-second assignment latencies across Yahoo, AOL, and affiliated properties following their 2015 acquisition. The platform addresses three interconnected engineering challenges: deterministic reproducible traffic assignment through a multi-layer orthogonal hash-based bucketing architecture; statistical validity assurance via the discovery and remediation of a systematic non-uniformity bias in the Fowler-Noll-Vo (FNV) hash function and the development of a pre-validated bucket system eliminating the traditional 4–5 day per-experiment A/A gating bottleneck; and dual batch and real-time event processing pipelines sustaining petabyte-scale data volumes required for high-power experiment analysis. Continuous platform health monitoring through identifier-level discrepancy detection reduced systematic bucket inconsistency from approximately 6% to below 1%, a quality improvement invisible to per-experiment validation. The pre-validated bucket system and its monitoring architecture were subsequently recognized in U.S. Patent Application Publication No. US 2019/0057108 A1 filed by Yahoo Holdings, Inc., independently confirming the engineering novelty of the contributions described here.

Keywords: *A/B testing, data pipelines, experimentation infrastructure, false discovery rate, hash-based bucketing, online controlled experiments, pre-validated data buckets, sequential testing*

1. Introduction

Controlled experimentation has become the standard methodology for evaluating causal effects of product changes at internet-scale organizations, with major technology companies running thousands of concurrent experiments annually to guide decisions affecting hundreds of millions of users [1]. The statistical framework underlying controlled experiments is well-established: random assignment of users to treatment and control groups eliminates confounding, allowing differences in observed outcomes to be attributed causally to the intervention. What the statistical framework does not specify is the engineering layer between its requirements and their realization in a production system, the hash functions, identity resolution mechanisms, event collection pipelines, and monitoring architectures that determine whether the randomization is genuinely uniform, whether outcomes are correctly attributed, and whether statistical results are actually valid [2].

This article documents the architecture of a centralized EaaS platform built to standardize controlled experimentation across the combined Yahoo-AOL ecosystem following the 2015 acquisition. The platform supported 500 million+ users, 1,000+ concurrent experiments across multiple properties and identity types, and sub-second assignment latencies on distributed infrastructure. The central engineering contribution is the identification of a systematic non-uniformity bias in the platform's FNV hash-based bucketing algorithm, the development and validation of an MD5 replacement, and the construction of a pre-validated bucket system that eliminated the traditional 4–5 day per-experiment A/A validation requirement for experiments drawing from the validated pool. Yahoo Holdings, Inc. subsequently filed a patent application formally documenting this pre-validated bucket method and its monitoring architecture as a novel invention (U.S. Patent Application Publication No. US 2019/0057108 A1, filed August 15, 2017) [3], a post-tenure filing that independently confirms the technical significance of the infrastructure described here.

The distinction between the statistical theory of experimentation and the engineering practice of it deserves explicit framing. A platform whose hash function maps user identifiers non-uniformly introduces pre-existing compositional differences between experiment arms before any treatment is applied. These differences are real statistical effects, detectable at conventional significance thresholds, but they are artifacts of the assignment mechanism

Software Engineer, Yahoo, USA
ORCID: 0009-0009-8602-4778

rather than indicators of genuine treatment effects. At a platform supporting 1,000+ concurrent experiments, a systematic hash bias that elevates A/A failure rates by even a few percentage points generates dozens of spurious validity concerns per experiment cycle, eroding organizational trust in experimental results and creating operational overhead that compounds across the full experiment portfolio.

The article proceeds as follows: Section 2 describes the three-layer EaaS architecture and the multi-layer orthogonal experiment framework; Section 3 documents the bucketing algorithm design, the FNV non-uniformity discovery, and the pre-validated bucket system; Section 4 describes the continuous platform health monitoring and discrepancy detection system; Section 5 presents the statistical infrastructure for false discovery rate control and sequential testing; Section 6 describes the dual batch and real-time event processing pipelines; and Section 7 synthesizes the contributions and their broader implications for experimentation infrastructure design.

2. Platform Architecture: Core Design Principles

2.1 The Three-Layer Experimentation Stack

Producing valid experimental results at scale requires solving three interdependent engineering problems whose failure modes are distinct and whose solutions demand independent design. The assignment layer must place every user request into an experiment arm through a mechanism that is both deterministically reproducible, the same user receives the same assignment on every request, and statistically independent of the outcome metric being measured. The event collection layer must capture impressions and outcome signals at petabyte scale with high reliability; event loss correlated with treatment assignment is statistically indistinguishable from a null treatment effect and systematically biases estimated effect sizes toward zero. The statistical analysis layer must produce valid, interpretable metrics with controlled type I error rates across a concurrent portfolio of experiments with heterogeneous user populations and continuously accumulating data [2].

Each layer was designed as an independently scalable and independently reliable subsystem with explicit service-level contracts governing availability, throughput, and data quality. Independence between layers is a correctness requirement: a failure in event collection should not corrupt assignment records; an analysis pipeline error should not affect the serving of ongoing experiments. The central artifact binding all three layers is the experiment registry, a versioned, ownership-tracked record of every active and historical experiment, containing its namespace, traffic allocation, arm definitions, target metric set, and lifecycle state. Every assignment decision, event attribution, and analysis scope computation references the registry as its sole authoritative source. Any corruption of the registry propagates immediately to the validity of all downstream results, making its integrity a first-class engineering concern.

The heterogeneous user population of the combined Yahoo-AOL ecosystem introduced identity resolution complexity at the assignment layer that single-property platforms do not face. Logged-in users were identified by persistent globally unique identifiers (GUIDs); logged-out desktop users by browser cookies (bcookies); mobile users by device identifiers. Each identity type carries distinct distributional properties that affect hash function behavior, GUIDs are pseudo-random, bcookies carry sequential temporal structure in their low-order bits, and device identifiers reflect manufacturer and region encoding patterns. The assignment layer was required to produce consistent, collision-free bucket assignments across all three identity types simultaneously, with defined fallback behavior for partial-identity sessions and explicit documentation of the cross-identity assignment independence boundary.

Table 1. EaaS Platform Specifications: Scale, Latency, and Throughput

Dimension	Specification	Notes
Total user population	500 million+	Yahoo, AOL, and affiliated properties combined
Concurrent experiment capacity	1,000+	Across all properties and namespaces
Assignment API latency (p99)	Sub-second	Distributed infrastructure, stateless serving
Daily event volume	Petabyte-scale	Impression + outcome events across all experiments

Identity types supported	GUID, bcookie, device ID	Distinct distributional properties per type
Property coverage	Yahoo, AOL, and affiliates	Post-2015 acquisition combined ecosystem

Table 1: Key operational specifications of the EaaS platform. These parameters establish the operating context for all engineering decisions in subsequent sections, the scale at which hash uniformity failures have portfolio-wide consequences, at which event collection reliability directly determines statistical power, and at which identity resolution complexity is a first-order design constraint. Data: Author's primary research data.

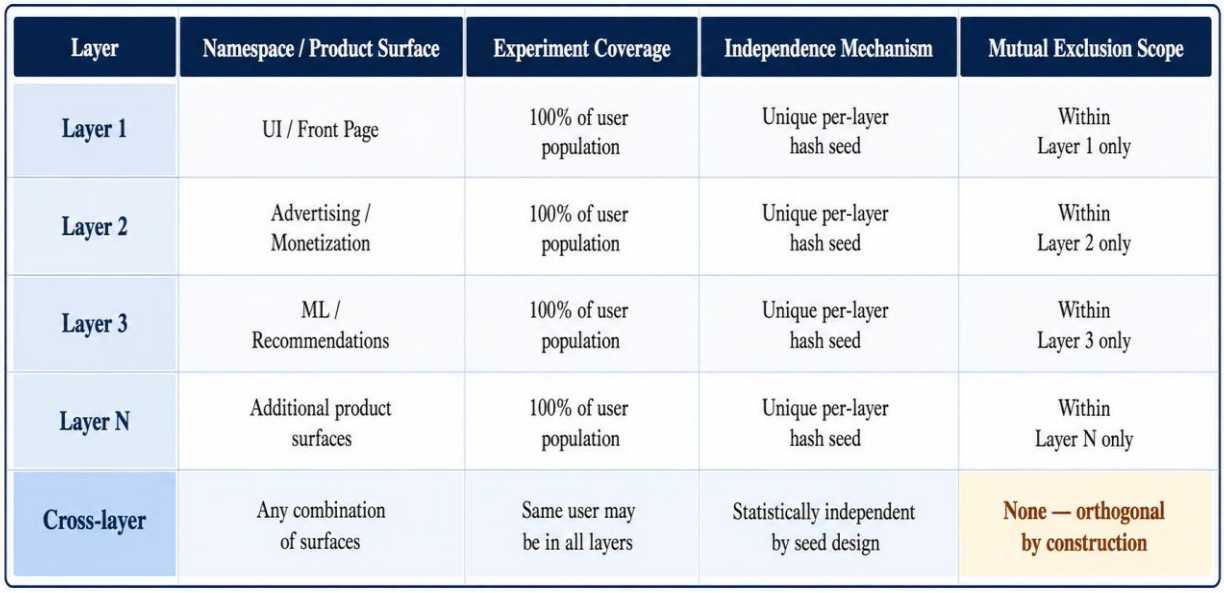
2.2 Multi-Layer Overlapping Architecture and Namespace Design

Experiment interference, where two concurrent tests modifying overlapping product features contaminate each other's results, scales linearly with the number of concurrent experiments on a single-layer platform. A naive single-layer system partitions the user population across all active experiments, so as concurrent experiment count grows, each experiment receives a progressively smaller traffic allocation and the risk of assigning users to multiple conflicting experiments increases. The multi-layer overlapping architecture resolves both problems simultaneously by partitioning parameters into N independent layers, each covering the full user population [4]. Within a layer, experiments are mutually exclusive, their bucket allocations sum to at most 100% of traffic. Across layers, experiments may run simultaneously on the same users because each layer applies the hash function with a unique per-layer random seed, ensuring that a user's bucket assignment in Layer i is statistically independent of their assignment in Layer j for all $i \neq j$.

The independence property of the multi-layer design is a mathematical consequence of per-layer seeding rather than an empirical approximation. Because the seed is incorporated into the hash input before any bucket computation, two layers processing the same user identifier with different seeds produce outputs sharing no statistical dependence under any reasonable model of hash function behavior. This guarantee was empirically verified during platform validation through pairwise correlation analysis: Pearson correlations between layer assignments across a representative user sample were statistically indistinguishable from zero at conventional significance thresholds across all tested layer pairs. The patent filed by Yahoo Holdings after completion of this work documents the architecture explicitly, confirming that each layer has a unique seed that is different and random, and that each layer covers all available traffic with each user identifier placed into at most one bucket per layer [3].

Namespace design governed the assignment of experiments to layers based on product surface and parameter ownership. Each major product surface, search, front page, advertising, mobile applications, was assigned to a designated set of layers, ensuring that experiments modifying the same product parameters were placed on the same layer (making them mutually exclusive by construction) while experiments on different surfaces occupied different layers (making them orthogonal by construction). A namespace registry tracked which product parameters were owned by which namespace, enforcing that parameter ownership conflicts surfaced at experiment registration time rather than through unexpected post-hoc result contamination.

Figure 1. Multi-Layer Overlapping Architecture: Independent Namespaces Covering Full User Population



Layer	Namespace / Product Surface	Experiment Coverage	Independence Mechanism	Mutual Exclusion Scope
Layer 1	UI / Front Page	100% of user population	Unique per-layer hash seed	Within Layer 1 only
Layer 2	Advertising / Monetization	100% of user population	Unique per-layer hash seed	Within Layer 2 only
Layer 3	ML / Recommendations	100% of user population	Unique per-layer hash seed	Within Layer 3 only
Layer N	Additional product surfaces	100% of user population	Unique per-layer hash seed	Within Layer N only
Cross-layer	Any combination of surfaces	Same user may be in all layers	Statistically independent by seed design	None — orthogonal by construction

Figure 1: Multi-layer overlapping architecture structure. Each layer covers the full user population independently, with mutual exclusion enforced within layers and statistical independence enforced across layers through unique per-layer hash seeds. Concurrent experiment capacity scales with the number of layers rather than with population size, each additional layer adds a full population's worth of experiment capacity. Data: Author's primary research data.

3. Bucketing Algorithm Design and the Pre-Validated Bucket System

3.1 Hash-Based Deterministic Assignment

Deterministic user bucketing uses a salted hash function with the assignment key constructed as hash(userID + experiment_salt + experiment_id). Per-experiment salting serves two purposes simultaneously: it decorrelates bucket assignments across experiments for the same user, knowing a user's bucket in Experiment A provides no information about their bucket in Experiment B, and it ensures that the bucket space for each experiment is independently derived even when the same user identifier participates in many concurrent experiments. The salt incorporates both a global experiment identifier and a per-experiment random component, making cross-experiment correlation vanishingly unlikely under any reasonable statistical model of hash function behavior across the full identifier space.

The uniformity requirement, that the hash function maps user identifiers to bucket indices with a distribution statistically indistinguishable from uniform random, is a non-negotiable correctness precondition for the entire assignment layer. Non-uniform hash output creates systematic compositional differences between experiment arms before any treatment is applied, producing pre-existing between-group differences that inflate A/A failure rates and, if undetected, bias effect size estimates [5]. At portfolio scale, even a modest hash bias that misallocates 2–3% of identifiers systematically generates statistically detectable pre-existing differences across many experiments simultaneously, creating an operational pattern of elevated A/A failures whose origin may be mistaken for sampling variability rather than a deterministic infrastructure defect.

The identity-specific properties of bcookie identifiers introduced a non-uniformity risk not anticipated during initial platform design. Unlike GUIDs, which are pseudo-random by construction, bcookies are assigned sequentially within generation windows, embedding temporal structure in their low-order bits. A hash function whose output distribution depends on the low-order bit pattern of its inputs will produce non-uniform bucket assignments when applied to sequentially structured identifiers, not because the hash function performs poorly on general inputs, but because the specific input distribution violates the distributional assumptions under which the function achieves uniformity. Detecting this interaction requires

empirical chi-squared testing against the actual identifier distributions in production, not theoretical analysis of the hash function in isolation.

3.2 The FNV Non-Uniformity Discovery and MD5 Migration

Systematic investigation of elevated A/A validation failure rates began with chi-squared testing of bucket assignment distributions produced by the Fowler-Noll-Vo (FNV) hash function across large bcookie samples drawn from production traffic. The chi-squared test statistic compared observed identifier counts in each of the 1,000 bucket indices against the expected count under the null hypothesis of uniform distribution. Results were unambiguous: across multiple independent bcookie samples spanning different traffic periods, FNV produced highly significant chi-squared p-values, indicating that its mapping of bcookie identifiers to bucket indices was systematically non-uniform. The non-uniformity was not random variation across samples, it was a reproducible structural property of FNV's interaction with the sequential prefix patterns present in bcookie identifiers, independently documented in comparable experimentation settings [5].

The operational chain from hash non-uniformity to experimental invalidity is direct. Non-uniform bucket assignment means that users landing in any given bucket range are not a representative sample of the full population, they are systematically enriched or depleted on whatever user characteristics correlate with the identifier prefix structure that FNV maps non-uniformly. For bcookies, which encode temporal assignment information, this translates to systematic differences in account age, device generation, and engagement vintage between bucket ranges. These compositional differences manifest as pre-existing differences in engagement metrics, exactly the pattern A/A testing is designed to detect. The elevated A/A failure rate was therefore a correct detection of a real statistical problem, not a false alarm; the problem was in the assignment mechanism rather than in the user population.

empirical chi-squared testing against the actual identifier distributions in production, not theoretical analysis of the hash function in isolation.

3.2 The FNV Non-Uniformity Discovery and MD5 Migration

Systematic investigation of elevated A/A validation failure rates began with chi-squared testing of bucket assignment distributions produced by the Fowler-Noll-Vo (FNV) hash function across large bcookie samples drawn from production traffic. The chi-squared test statistic compared observed identifier counts in each of the 1,000 bucket indices against the expected count under the null hypothesis of uniform distribution. Results were unambiguous: across multiple independent bcookie samples spanning different traffic periods, FNV produced highly significant chi-squared p-values, indicating that its mapping of bcookie identifiers to bucket indices was systematically non-uniform. The non-uniformity was not random variation across samples, it was a reproducible structural property of FNV's interaction with the sequential prefix patterns present in bcookie identifiers, independently documented in comparable experimentation settings [5].

The operational chain from hash non-uniformity to experimental invalidity is direct. Non-uniform bucket assignment means that users landing in any given bucket range are not a representative sample of the full population, they are systematically enriched or depleted on whatever user characteristics correlate with the identifier prefix structure that FNV maps non-uniformly. For bcookies, which encode temporal assignment information, this translates to systematic differences in account age, device generation, and engagement vintage between bucket ranges. These compositional differences manifest as pre-existing differences in engagement metrics, exactly the pattern A/A testing is designed to detect. The elevated A/A failure rate was therefore a correct detection of a real statistical problem, not a false alarm; the problem was in the assignment mechanism rather than in the user population.

Table 2. Hash Function Comparison: FNV vs. MD5 vs. SpookyHash on Uniformity and Production Suitability

Dimension	FNV	MD5	SpookyHash
Chi-squared test on bcookie inputs	Statistically significant non-uniformity	Uniform, passes at all sample sizes	Uniform, independent cross-validation
Chi-squared test on GUID inputs	Passes	Passes	Passes
Chi-squared test on device ID inputs	Marginal	Passes	Passes
Computational throughput (relative)	1.0× (baseline)	0.7× FNV	1.4× FNV
Collision resistance	Low, not cryptographic	Moderate, MD5 is cryptographic	High, designed for hashing
Production deployment risk	Known non-uniformity issue	Low, validated across all ID types	Low, used as cross-validation
Recommended role	Replaced, do not use for bucketing	Primary bucketing hash	Independent uniformity validation

Table 2: Comparison of FNV, MD5, and SpookyHash for production bucketing use. MD5 is the correct replacement for FNV because it achieves uniform distribution across all three identity types while maintaining adequate throughput for sub-second assignment latency. SpookyHash served as independent cross-validation confirming that the MD5 improvement reflects genuine distributional correctness rather than an MD5-specific artifact. The key finding is that hash function selection for bucketing must be validated empirically against production identifier distributions.

The MD5 replacement was validated through a staged process before any production experiment traffic was migrated. Chi-squared tests on MD5 bucket assignments across all three identity types consistently produced p-values within the expected distribution under the null hypothesis of uniformity across all tested sample sizes. SpookyHash, evaluated in parallel as an independent alternative, produced comparable uniformity results, confirming that the performance difference between FNV and MD5 reflected a genuine improvement in identifier distribution handling. Following validation, the MD5-based bucketing system was itself subjected to A/A testing before experiment traffic migration, applying the same quality verification process to the infrastructure change that all product experiments underwent.

3.3 The Pre-Validated Bucket System

The MD5 migration resolved the structural source of elevated A/A failures but did not eliminate the per-experiment A/A validation requirement that had accumulated as operational practice. The requirement existed for a sound reason: confirming that a specific experiment's bucket assignment, as configured with its particular salt and traffic allocation, had not introduced any pre-existing group differences. With MD5-based bucketing validated as uniformly distributed, the remaining source of A/A failures is natural statistical variation at the nominal alpha level, the expected fraction of correctly configured experiments whose control groups differ significantly by chance. A mechanism that could identify and quarantine the hash value ranges most likely to produce these chance failures would allow the remaining bucket space to support experiment launches without per-experiment waiting, converting the platform's reliability improvement into a direct operational acceleration.

The pre-validated bucket pipeline operates on a 24-hour refresh cycle over historical user activity data. Each of the 1,000 integer hash values in the range [0, 999] is assigned a set of user identifiers determined by the MD5 mapping. For each hash value, the pipeline computes mean values of key engagement metrics, days visited per period, page view rate, session count, and distinct identifier count, across all identifiers mapping to that value. These per-hash-value metric means are ranked across all 1,000 hash values for each metric dimension independently. Hash values falling in the top-N or bottom-N percentile on any single engagement metric dimension are flagged as statistical tails and excluded from the eligible pool. The remaining homogeneous set, hash values unremarkable across all engagement dimensions simultaneously, constitutes the pre-validated pool

available for experiment assignment without per-experiment A/A gating. This mechanism was subsequently documented in U.S. Patent Application Publication No. US 2019/0057108 A1 as a novel method for providing pre-approved A/A data buckets for online experiments [3].

The reduction in experiment cycle time produced by the pre-validated bucket system compounded across the portfolio of 1,000+ concurrent experiments. Each experiment that previously required 4–5 business days of A/A validation before treatment exposure could instead launch from the pre-validated pool within hours of registration. At portfolio scale, this acceleration represents a substantial increase in the organization's annual rate of evidence-based product decisions, the throughput of the experimentation system as a whole, not merely the latency of individual experiments (author's primary research data). The mechanism also shifted the operational model from reactive quality assurance, investigating A/A failures as they occurred, to proactive quality assurance, with tail bucket ranges identified and excluded before any experiment drew from them.

Figure 2. Pre-Validated Bucket Pipeline: From Raw User Activity to Experiment Launch

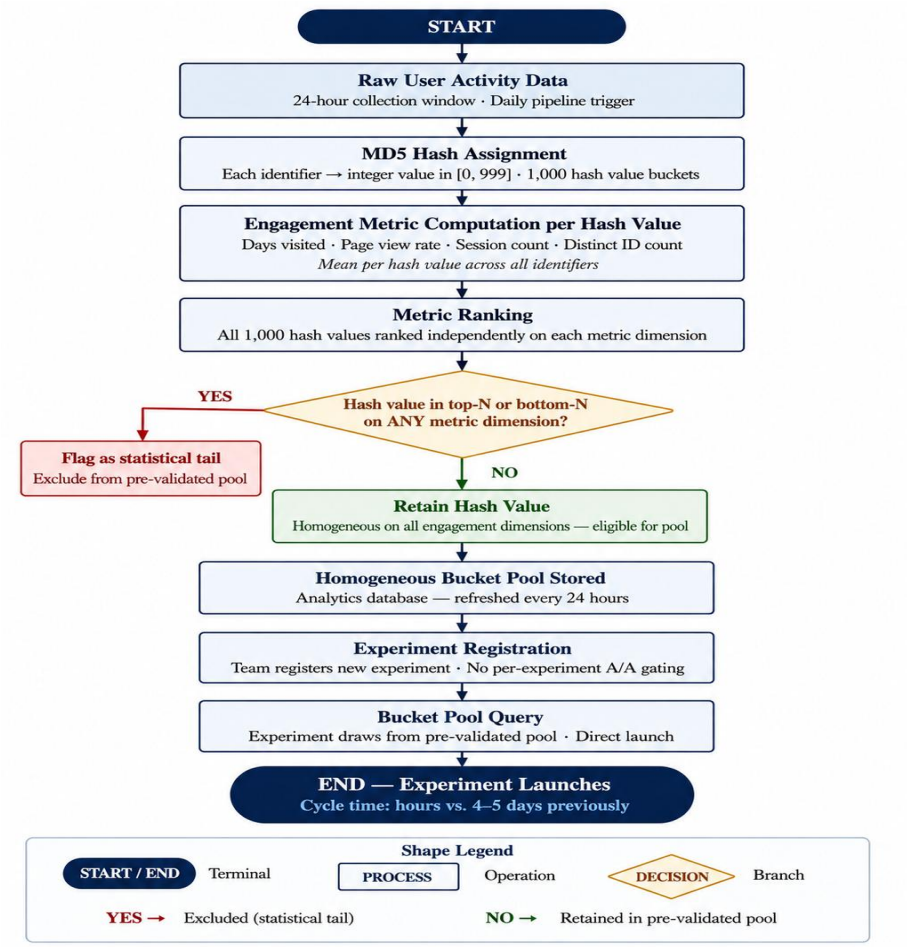


Figure 2: Pre-validated bucket generation and consumption pipeline. Tail exclusion is applied across all metric dimensions simultaneously, a hash value is excluded if it is anomalous on any single dimension. The 24-hour refresh ensures pre-validated buckets reflect current traffic composition. Shape legend: [START/END] = terminal; [PROCESS] = operation; [DECISION] = branch point; [YES/NO] = branch outcomes. Data: Author's primary research data.

4. Continuous Platform Health Monitoring and Discrepancy Detection

Eliminating per-experiment A/A gating shifted the locus of quality assurance from individual experiments to the platform itself. Per-experiment A/A validation detects assignment problems specific to individual experiment configurations but is structurally incapable of detecting systematic platform-level failures that affect all experiments uniformly. A consistent bias in the assignment

mechanism produces no detectable between-group difference in any single experiment's A/A test, yet it invalidates every experiment's results simultaneously. Continuous platform-level monitoring addresses this blind spot by treating assignment integrity as an observable, measurable engineering metric rather than an assumed precondition, enabling systematic issues to surface as quantified quality indicators rather than unexplained statistical anomalies.

The continuous monitoring architecture uses a dedicated layer within the multi-layer experimental framework, a layer whose sole purpose is quality bucket stamping rather than product experimentation. Every user identifier that successfully traverses the full bucketing and assignment pipeline receives a metadata quality stamp from the monitoring layer, creating a reference count of identifiers that should appear in experiment buckets if the assignment mechanism is operating correctly. The discrepancy detection system compares this reference population against the identifier counts actually observed in each experiment's data buckets on a daily cycle, computing gap rates, identifiers present in the reference but absent from experiment buckets, and overlap rates, identifiers appearing in multiple buckets of the same experiment, for every active experiment. Gaps indicate event collection failures or identity resolution errors; overlaps indicate bucketing logic defects that assign users to multiple arms simultaneously.

Production data documented in the Yahoo Holdings patent reveals a finding central to the case for identifier-level monitoring: apparent bucket balance at the aggregate level, where total bucket sizes appear correct, can coexist with approximately 6% identifier-level inconsistency, where specific identifiers are misattributed across buckets [3]. This inconsistency is invisible to aggregate-level quality checks because the misattributions are frequently reciprocal, an identifier incorrectly excluded from one bucket is often incorrectly included in another, leaving aggregate counts balanced while individual attributions are wrong. Implementation of the monitoring layer and systematic discrepancy remediation reduced this inconsistency rate to below 1% over time, converting each identified inconsistency source from a statistical anomaly requiring multi-day investigation into a concrete, actionable engineering task.

Phase	Identifier Inconsistency Rate	Quality Visibility	Remediation Mode	Impact on Experiment Portfolio
Pre-monitoring baseline	~6%	Not observable — invisible to aggregate checks	Reactive — post-hoc statistical anomaly investigation	Silent contamination of all concurrent experiments
Monitoring deployment phase	6% → 2%	Observable — specific inconsistency sources identified	Active — engineering tickets per identified source	Contamination rate declining per remediation
Post-monitoring steady state	<1%	Continuously monitored — deviations alert immediately	Proactive — threshold alerts trigger investigation	Minimal contamination — near-full assignment integrity

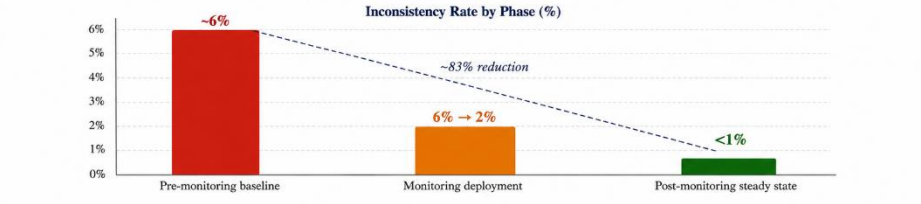


Figure 3. Identifier Inconsistency Rate Before and After Continuous Monitoring Implementation Figure 3: Identifier inconsistency rate across three phases of monitoring implementation. The transition from ~6% to <1% inconsistency represents a direct improvement in statistical validity across all concurrent experiments, a 6% inconsistency rate dilutes estimated effect sizes toward zero, reducing statistical power across the portfolio. The monitoring layer made this previously invisible quality dimension both measurable and manageable. Data: Author's primary research data (mechanism documented in U.S. Patent No. US 2019/0057108 A1 [3]).

The relationship between identifier inconsistency rate and experimental validity is quantifiable and direct. A 6% inconsistency rate means that 6% of user identifiers are misattributed across experiment arms, contaminating both treatment and control populations with users whose assignment does not reflect their actual product experience. The resulting treatment effect dilution biases estimated effect

sizes toward zero, the same direction as a null effect, meaning that genuine product improvements are systematically underestimated, and experiments that would have achieved statistical significance under correct attribution fail to do so. Reducing the inconsistency rate from 6% to below 1% through continuous monitoring directly increases the platform's statistical power across all concurrent experiments, compounding the benefit across the full portfolio of 1,000+ active experiments.

5. Statistical Infrastructure: False Discovery Rate Control and Sequential Testing

A portfolio of 1,000+ concurrent experiments creates a multiple comparisons problem whose magnitude exceeds what individual-experiment statistical controls can address. At a nominal false positive rate of 5%, an uncorrected portfolio of 1,000 independent experiments is expected to produce approximately 50 statistically significant results from purely null effects, a false discovery count operationally indistinguishable from genuine treatment effects without portfolio-level correction. The appropriate framework depends on the organizational decision objective: Bonferroni correction provides family-wise error rate control, the probability that any single false discovery occurs is bounded at alpha, but at the cost of substantial power loss across all experiments simultaneously. When the goal is to maximize the number of true discoveries while maintaining an acceptable fraction of false ones, Benjamini and Hochberg's false discovery rate (FDR) procedure is substantially more appropriate [6].

The Benjamini-Hochberg procedure controls the expected proportion of false discoveries among all declared significant results. For a portfolio of K experiments with ranked p-values $p(1) \leq p(2) \leq ... \leq p(K)$, the procedure identifies the largest rank k such that $p(k) \leq (k/K) \times$ alpha and declares all experiments at or below that threshold as significant. This approach trades a controlled and quantifiable proportion of false positives, bounded at alpha on average, for substantially higher true discovery yield relative to Bonferroni correction. The platform implemented Benjamini-Hochberg FDR control as the default framework for teams running families of related experiments or large multivariate tests, complemented by documentation explaining the trade-off between false positive protection and discovery yield to experiment owners accustomed to per-experiment alpha-level reasoning.

Sequential testing, the inflation of false positive rates when product teams monitor experiment dashboards under fixed-horizon testing assumptions, was addressed through mixture sequential probability ratio tests (mSPRT) as the platform's framework for valid early stopping. Fixed-horizon test statistics evaluated repeatedly as data accumulates inflate the realized false positive rate from a nominal 5% to 20–30% or higher depending on monitoring frequency [7]. The mSPRT produces always-valid p-values that maintain the nominal alpha guarantee regardless of when the test is stopped, because the sequential boundary accounts for all possible stopping times simultaneously. A team that monitors an mSPRT experiment daily and stops when the p-value crosses the alpha threshold will not exceed the stated false positive rate, a guarantee that monitored fixed-horizon testing cannot provide.

Table 3. Statistical Framework Comparison: Classical NHST vs. FDR Control vs. mSPRT

Dimension	Classical NHST	BH-FDR Control	mSPRT
Portfolio scale	Single experiments	Concurrent experiment families	Any scale
Early stopping validity	Invalid, inflates FPR to 20–30%	Not applicable	Valid, always-valid p-values
False positive control mode	Per-experiment alpha	Portfolio FDR at alpha	Per-experiment alpha at any stopping time
Statistical power vs. NHST	Baseline	Higher, fewer corrections applied	Slightly lower, sequential boundary cost
Recommended EaaS context	Isolated experiments, no monitoring	Related experiment families; multivariate tests	Any experiment where teams monitor dashboards

Table 3: Comparison of three statistical frameworks across deployment context, false positive control mode, and power characteristics. The frameworks address distinct statistical problems and are deployed as complementary options rather than competing alternatives. NHST is appropriate only for isolated experiments evaluated once at planned sample size; BH-FDR is appropriate for portfolio-scale discovery optimization; mSPRT is appropriate

whenever teams monitor experiment results before the planned sample size is reached. Data: Author's primary research data.

Variance reduction through pre-experiment covariate adjustment complemented the sequential testing framework by reducing sample sizes required for adequate statistical power [8]. The CUPED (Controlled-experiment Using Pre-Experiment Data) methodology regresses the outcome metric on its prior-period value, removing variance attributable to stable user-level characteristics that do not change in response to the treatment. For metrics with high baseline variance, daily active users, session duration, revenue per user, CUPED adjustments typically reduce required sample sizes by 30–50% relative to unadjusted analysis, translating directly to faster experiment cycles and higher throughput of the evidence production pipeline across the full concurrent experiment portfolio.

6. Event Processing: Dual Batch and Real-Time Pipelines

The event processing layer must sustain two qualitatively distinct workloads simultaneously: the high-throughput daily analysis of petabytes of assignment and outcome event data, and the continuous low-latency monitoring of guardrail metrics enabling regression detection within minutes of experiment launch. These workloads have fundamentally conflicting optimization objectives. Throughput and accuracy favor batch processing: large-scale MapReduce jobs process petabyte datasets reliably at the cost of latency measured in hours. Low-latency monitoring favors stream processing: event aggregation pipelines produce sub-minute metric updates at the cost of the deduplication and attribution accuracy that batch processing provides. A dual-pipeline architecture is therefore a correctness requirement rather than an optimization, neither pipeline can substitute for the other without sacrificing either statistical rigor or guardrail monitoring timeliness.

The batch processing layer used Hadoop MapReduce and Apache Hive to execute the primary experiment analysis pipeline on a daily cycle. The core computational step is an attribution join between the raw event log and the experiment assignment log, attributing each outcome event to the experiment arm responsible for the user's product experience at the time of the event. This join is executed across the full daily event volume at petabyte scale, producing per-experiment, per-arm metric summaries used as inputs to the statistical analysis layer. Pre-experiment covariate values for CUPED variance reduction were computed in the same pipeline pass using prior-period event data, eliminating a separate historical data scan. The monitoring layer's discrepancy detection computations, comparing reference identifier counts to observed counts per experiment, ran as an additional step in the same daily Hadoop job, ensuring quality audit records were co-produced with statistical results for every active experiment on every daily cycle.

The real-time pipeline, built on Apache Storm for sub-minute event aggregation and Apache Druid for low-latency analytical query serving, provided continuous experiment health visibility at timescales the batch pipeline cannot support. Guardrail metrics, application error rates, API response latency percentiles, crash rates, and revenue per session, were computed continuously from the live event stream and indexed in Druid, enabling teams to query current experiment health with sub-second latency from monitoring dashboards. A critical regression in a guardrail metric produces a detectable signal within minutes of experiment launch at 500-million-user scale, allowing intervention before significant user harm accumulates. The real-time pipeline complemented the batch pipeline without competing with it: real-time outputs were explicitly labeled as preliminary monitoring data, and final statistical conclusions were drawn exclusively from daily batch results with fully deduplicated event attribution.

Table 4. Dual Pipeline Comparison: Hadoop/Hive Batch vs. Storm/Druid Real-Time

Dimension	Batch (Hadoop/Hive)	Real-Time (Storm/Druid)
Update cadence	Daily	Sub-minute (continuous)
Data throughput capacity	Petabyte-scale, full daily event log	Current event stream, no historical scan
Query latency	Minutes (MapReduce job completion)	Sub-second (Druid analytical query)
Primary use case	Final statistical analysis; monitoring discrepancy audit	Guardrail metric monitoring; early regression detection
Statistical validity of outputs	Authoritative, fully deduplicated and attributed	Preliminary, at-least-once delivery semantics
Failure consequence	24-hour delay in final analysis	Missed early regression detection window
Output labeling in dashboard	Experimental conclusions with confidence intervals	Monitoring data, not for causal inference

Table 4: Comparison of batch and real-time pipelines across seven operational dimensions. The two pipelines serve distinct and non-interchangeable roles. The batch pipeline produces statistically authoritative results; the real-time pipeline provides early warning signals whose statistical properties are explicitly acknowledged as preliminary. Dashboard labeling enforcing this distinction prevented premature product decisions based on pre-convergence monitoring signals. Data: Author's primary research data.

The organizational benefit of separating final statistical analysis from guardrail monitoring extended beyond technical performance to decision-making discipline. When all experiment metrics appear in a single dashboard without provenance labeling, product teams naturally make decisions based on whichever metric first crosses a significance threshold, regardless of whether that metric has the statistical properties required for causal inference. The dual-pipeline architecture enforced a labeling regime: Storm/Druid outputs were explicitly marked as monitoring data, and Hadoop/Hive outputs were marked as experimental conclusions. This separation reduced premature stopping decisions based on early guardrail signals and improved the overall quality of product decisions produced by the platform beyond what any statistical methodology change could achieve independently.

Conclusion

A centralized experimentation platform at 500 million user scale is ultimately a product quality assurance system whose reliability determines the validity of every data-driven decision the organization makes. The engineering contributions documented in this article each address a specific failure mode observed in production: FNV hash non-uniformity, identified through chi-squared testing and resolved through MD5 migration with SpookyHash cross-validation; systematic bucket identifier inconsistency at 6%, detected through continuous identifier-level monitoring and reduced to below 1% through targeted remediation; multiple comparison false positive inflation, managed through Benjamini-Hochberg FDR control; sequential monitoring false positive inflation, addressed through mSPRT always-valid p-values; and the operational bottleneck of per-experiment A/A gating, eliminated through the pre-validated bucket system that converted a platform health improvement into a direct acceleration of the organization's evidence production rate.

The pre-validated bucket system represents the article's most operationally significant contribution. Its mechanism, continuous refresh of a homogeneous bucket pool from which experiments draw without per-experiment validation waiting, converted a statistical reliability improvement into an organizational capability improvement, reducing experiment cycle time from weeks to hours for experiments drawing from the pre-validated pool. The independent recognition through U.S. Patent Application Publication No. US 2019/0057108 A1 provides the strongest possible external confirmation of engineering novelty: a patent application filed after the author's tenure, by the organization that operated the infrastructure, documenting the mechanism as a novel and patentable invention.

For practitioners building experimentation infrastructure at scale, the article's central finding is that statistical validity is an engineering property requiring active maintenance rather than a precondition established once at design time. Hash

function uniformity must be validated empirically against production identifier distributions. Bucket assignment integrity must be verified at the identifier level through continuous monitoring, not inferred from aggregate bucket size correctness. Statistical framework selection must reflect how experiment teams actually monitor results, not how the frameworks assume results will be evaluated. Each of these insights derives from production failures in a system supporting 500 million users, giving them a practical authority that theoretical analysis alone cannot provide.

References

[1] Ron Kohavi, Diane Tang, and Ya Xu, "Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing," Cambridge University Press, Cambridge, 2020. Available: https://www.researchgate.net/publication/339914315_Trustworthy_Online_Controlled_Experiments_A_Practical_Guide_to_AB_Testing

[2] Ron Kohavi, et al., "Online controlled experiments at large scale," ACM Digital Library, 2013. [Online]. Available: https://doi.org/10.1145/2487575.2488217

[3] Russell Chen, et al., "Method and system for providing pre-approved A/A data buckets," U.S. Patent Application Publication No. US 2019/0057108 A1, Yahoo Holdings, Inc., filed Aug. 15, 2017, pub. Feb. 21, 2019. [Online]. Available: https://patentimages.storage.googleapis.com/60/d5/e4/f0d549241cb006/US20190057108A1.pdf

[4] Diane Tang, et al., "Overlapping experiment infrastructure: More, better, faster experimentation," ACM Digital Library, 2010. [Online]. Available: https://doi.org/10.1145/1835804.183581

[5] Zhenyu Zhao, et al., "Online experimentation diagnosis and troubleshooting beyond A/A validation," 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7796936

[6] Yoav Benjamini and Yosef Hochberg, "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing," Journal of the Royal Statistical Society: Series B (Methodological), 1995. [Online]. Available: https://academic.oup.com/jrsssb/article/57/1/289/7035855

[7] Ramesh Johari, et al., "Peeking at A/B Tests: Why it matters, and what to do about it," ACM Digital Library, 2017. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/3097983.3097992

[8] Alex Deng, et al., "Improving the sensitivity of online controlled experiments by utilizing pre-experiment data," ACm Digital Library, 2013. [Online]. Available: https://doi.org/10.1145/2433396.2433413

[9] Ya Xu, et al., "From Infrastructure to Culture: A/B Testing Challenges in Large Scale Social Networks," ACM Digital Library, 2015. [Online]. Available: https://dl.acm.org/doi/10.1145/2783258.2788602

[10] Alex Deng, et al., "Continuous monitoring of A/B tests without pain: Optional stopping in Bayesian testing," 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7796910

[11] Eytan Bakshy, Dean Eckles and Michael S. Bernstein, "Designing and deploying online field experiments," WWW '14: Proceedings of the 23rd international conference on World wide web, 2014. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/2566486.2567967

[12] Alex Deng and Xiaolin Shi, "Data-Driven Metric Development for Online Controlled Experiments: Seven Lessons Learned," KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/2939672.2939700

[13] Somit Gupta, et al., "Top challenges from the first practical online controlled experiments summit," ACM SIGKDD Explorations Newsletter, 2019. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/3331651.3331655

[14] Ron Kohavi and Roger Longbotham, "Online controlled experiments and A/B testing," in Encyclopedia of Machine Learning and Data Mining, 2017. [Online]. Available: https://link.springer.com/rwe/10.1007/978-1-4899-7687-1_891

[15] Peter Auer, Nicolò Cesa-Bianchi & Paul Fischer, "Finite-time analysis of the multiarmed bandit problem," Machine Learning, 2002. [Online]. Available: https://link.springer.com/article/10.1023/A:1013689704352