International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 10-No.4 (2024) pp. 4309-4331
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Event-Aware Multi-Layer Storage Risk Forecasting for Oracle Database Estates Using HAPF

Raghu Gollapudi*

Fiserv Inc. (United States)
* **Corresponding Author Email:** Reachraghu251@gmail.com- **ORCID:** 0009-0006-6861-3796

Article Info:

DOI: 10.22399/ijcesen.5183
Received : 01 February 2024
Accepted : 27 February 2024

Keywords

Oracle database;
storage risk forecasting;
Oracle ASM;
FRA;
event-aware forecasting;
capacity horizon

Abstract:

This study formulates storage growth forecasting in Oracle database estates as a cross-layer capacity-horizon problem, where the first effective bottleneck - rather than a single utilization curve - should govern engineering intervention timing. The paper develops an event-aware Holistic Allocation Pattern Forecasting (HAPF) workflow using anonymized daily telemetry from January 2021 to December 2023. The method combines Holt-Winters baseline forecasting, ARIMA consistency checking, event injection for release, purge, migration, and capacity-expansion events, and cross-layer threshold-horizon ranking across tablespace, ASM, FRA, archive, TEMP, and backup-related layers. The proposed approach surfaces Archive/FRA pressure and mirrored-capacity compression earlier than the visually dominant primary tablespace. Relative to threshold-only monitoring, HAPF expands first-bottleneck lead time by about four weeks, improves bottleneck-rank correctness, and supports earlier decisions on retention cleanup, datafile growth, archive handling, and capacity expansion. Database storage forecasting is most useful when it closes the loop between prediction and action. By preserving Oracle-specific control surfaces while reframing them as a multi-layer engineering decision system, the study shows that storage control should be evaluated by horizon accuracy, bottleneck sequencing, and intervention usefulness rather than by single-series fit alone.

1. Introduction

Storage capacity planning is one of the few database administration tasks that looks trivial until it fails. Teams often track used gigabytes, autoextend behavior, and alert thresholds, yet production incidents still occur because the real exhaustion point is frequently hidden in another layer. A tablespace can appear manageable while the ASM diskgroup is almost out of usable free space; archive generation can spike even though base segment growth remains stable; a temporary workload can consume disk-backed scratch space before persistent objects become the limiting factor. Live systems therefore expose a structural mismatch between how risk accumulates and how risk is commonly monitored [7]-[10].

The dominant operational habit is still single-layer trending: extrapolate used space, attach a static threshold such as 85%, and react once the curve looks uncomfortable. That habit is weak for three reasons. First, enterprise growth is rarely linear. Releases, onboarding waves, retention changes, and bulk migrations introduce step changes that invalidate straight-line forecasts [1]-[6]. Second, logical growth and physical capacity are not equivalent. Autoextended datafiles, mirrored ASM capacity, flash recovery areas, and backup retention all convert one application event into several downstream storage consequences [10]-[17]. Third, reactive thresholds trigger after the trend has already bent. An operations engineering team that starts asking for storage only when utilization is already high is not forecasting; it is merely delaying the outage narrative.

This paper argues that storage forecasting should be treated as an industrial capacity-forecasting and intervention-timing problem. The relevant managerial question is not simply what utilization will look like next month, but which layer will breach first, how much warning time remains, and whether that lead time is still large enough to support procurement, scheduled maintenance windows, controlled change approval, or cross-

team remediation. In engineering systems terms, the problem is a risk-horizon problem in which the usefulness of a forecast depends on whether it changes the feasible intervention set before the system reaches a non-deferrable state [20], [21], [24], [25].

The contribution of the manuscript is therefore twofold. At the technical level, it preserves the Oracle and DBA control surfaces exactly as they operate in practice: tablespace, ASM, FRA, archive, TEMP, datafile, redo, purge, and HAPF remain the actual decision objects. At the managerial level, it packages those objects as a layered operational decision system that supports threshold-horizon ranking, intervention policy selection, and postmortem accountability. The goal is not to generalize the Oracle vocabulary away, but to show how that vocabulary maps onto predictive maintenance, capacity planning, and operational control-loop design in the language of engineering systems and operational control.

The study addresses four research questions. First, why do single-metric capacity views fail to identify the true storage bottleneck in enterprise environments? Second, how much lead time is gained when known operational events are injected into the forecast before the environment shifts? Third, can a cross-layer horizon engine rank warning and critical risk in a way that supports monitor, planned change, urgent remediation, and escalation as explicit decision tiers? Fourth, how should the resulting storage forecast be evaluated when the real decision variable is not fit quality alone but intervention timing, maintenance window quality, and control usefulness? This framing turns a DBA alerting problem into an industrial operations decision problem with measurable lead time and actionability.

To answer these questions, the paper draws on an anonymized telemetry trace from January 2021 through December 2023 and evaluates the proposed Holistic Allocation Pattern Forecasting (HAPF) framework against threshold-only and single-layer alternatives. The analysis keeps the core operational logic of the attached paper intact, but it repositions the result as an operations engineering contribution in predictive maintenance, intervention timing, and capacity-governance design. In engineering operations vocabulary, the paper studies how a technical forecast becomes a usable intervention policy.

2. Background and Related Work

Classical forecasting literature established that trend and seasonality should not be confused with permanence. Holt's exponentially weighted formulation and Winters' seasonal extension remain foundational precisely because many operational series exhibit persistent but revisable structure [1], [2]. Box-Jenkins modeling generalized this logic through differencing, autocorrelation, and explicit model diagnostics [3]. Modern forecasting texts still emphasize the same core lesson: method choice matters less than whether the model respects the data-generating structure, the error metric, and the forecast use case [4], [5], [21], [22]. That lesson is directly relevant to storage capacity forecasting, where the objective is not curve fitting for its own sake but timely intervention in a constrained operating environment.

In engineering practice, production forecasting also requires interpretability. Prophet and related large-scale forecasting workflows popularized analyst-in-the-loop adjustment for events and changepoints, especially when a purely automated model cannot infer future business decisions from past observations alone [6]. The same logic appears in SRE and data-platform operations: forecasts should expose rationale, uncertainty, and intervention points because infrastructure teams must justify action windows, escalation thresholds, and remediation economics rather than merely publish dashboards [20]. In engineering systems terms, interpretability matters because a forecast is part of the control system, not just an analytical artifact.

Database systems add a second dimension absent from generic time-series problems: layered state. Transaction processing systems tie business workload to redo generation, undo growth, checkpoint activity, segment allocation, and physical durability [7]. Modern database texts describe extents, segments, tablespaces, and datafiles as linked allocation abstractions rather than independent containers [8], [10], [12]. Data-intensive systems literature similarly stresses that logical writes propagate through logging, replication, and storage-management layers, each with its own operational constraints [9]. A correct forecasting architecture therefore has to treat storage as a cascade rather than as a single counter.

Oracle operational guidance reinforces that point. Tablespace free space is logical. ASM usable free is redundancy-aware. FRA pressure depends on archived redo, flashback, and backup retention. Standby environments duplicate growth and can fail on independent thresholds [10]-[17]. Operations Insights extends this into practical capacity dashboards and threshold tuning, which is evidence that the industry already recognizes the need for forecast-driven planning instead of late-stage alarms. The gap is not that practitioners lack

metrics; the gap is that the metrics are rarely assembled into one ranked operational horizon.

The paper also aligns with engineering systems concerns about decision latency. Forecasting systems create value only when the information arrives early enough to alter the feasible intervention set. In a manufacturing context that may mean preventive maintenance, spare-parts allocation, or labor scheduling. In database storage it means whether tablespace extension, ASM rebalancing, archive cleanup, or capacity procurement can still be executed inside a controlled maintenance window instead of under emergency conditions. The bridge between these domains is the same: telemetry must be converted into actionable lead time rather than into descriptive hindsight [20], [24], [25].

Another relevant point from operations engineering research is that capacity metrics are often lagging indicators of coordination quality. Storage risk is not created only by bytes consumed; it is also created by whether release planning, retention policy, backup scheduling, and platform engineering exchange information early enough to preserve safe slack. That is why the event register in HAPF is not an optional embellishment. It is the mechanism that connects technical measurement to managerial planning and thereby closes the gap between forecasting and operational control.

Research closer to storage utilization points in the same direction. Wicaksono et al. showed that Holt-Winters outperformed a simple ARIMA configuration on a storage-server utilization dataset, demonstrating that even conventional methods can provide actionable accuracy when telemetry is structured well [18]. IBM's population-analytics work argued that capacity prediction improves when the model learns from broader usage patterns instead of naive local trend lines [19]. Earlier enterprise workload literature by Gmach et al. framed demand prediction as a capacity-management problem tied to placement and provisioning decisions rather than to isolated statistical extrapolation [23]. These studies strengthen the case that forecasting must be tied to operational decision quality, not just to statistical fit.

A second relevant research stream comes from predictive maintenance and intervention-timing research in engineering systems. Maintenance optimization studies repeatedly show that the value of condition signals lies in their ability to change intervention timing, spare-capacity allocation, and maintenance-window design before failure becomes imminent [24], [25]. Although database storage is not a rotating machine or a physical production line, the decision structure is analogous: telemetry indicates deterioration, the planner estimates the risk horizon, and the organization chooses whether to monitor, schedule, remediate, or escalate. The managerial logic is identical even when the technical substrate is Oracle storage rather than a physical asset.

Service and operations engineering research adds one more bridge. Capacity decisions in service systems are valuable only when they preserve continuity for downstream users while managing scarce intervention opportunities [24]. Storage control in enterprise platforms fits that pattern because the real cost is not simply full disk; it is the compression of safe maintenance windows, the collapse of procurement lead time, and the downstream effect on release schedules, recovery posture, and postmortem accountability. In engineering operations vocabulary, database storage forecasting becomes a decision-support mechanism for risk horizon management, intervention policy, and operational control-loop closure.

Taken together, the literature suggests a specific gap. Existing work provides forecasting methods, storage metrics, and general capacity-planning guidance, yet it rarely combines them into a per-layer threshold-horizon system that ranks the earliest bottleneck and binds that ranking to an action class. HAPF addresses that gap by treating tablespace, ASM, FRA, archive, TEMP, backup, and related layers as one industrial intervention-timing problem governed by lead time, maintenance windows, and intervention feasibility.

3. Operational Problem Statement

The operational setting studied here can be summarized in one sentence: storage risk becomes visible later than it begins. Business workload events - new customer onboarding, feature releases, retention changes, bulk migration, or large maintenance windows - first alter consumption patterns at the top of the stack. Those changes then propagate through memory and TEMP pressure, redo or archive generation, segment and tablespace growth, datafile extension, and finally physical capacity constraints such as ASM usable free, filesystem headroom, FRA occupancy, and backup estate growth. By the time a static percentage alert fires at one layer, the causal process that created the risk may already have been active for weeks.

The first failure mode in current practice is siloed observability. Teams commonly trend the main data tablespace but do not jointly evaluate archive, TEMP, backup, and mirrored-capacity views. That produces false comfort. A database can show 20% logical headroom while the physical diskgroup has

only a narrow redundancy-safe margin left. Conversely, a main datafile may look healthy while archive backlog is hours away from halting commits. Single-layer visibility therefore measures only a slice of system state and tends to identify the wrong bottleneck.

The second failure mode is delayed thresholding. Static alert levels such as 85% are useful as safety rails, but they are poor forecasting devices. They say nothing about velocity of change, nothing about future events already known to operators, and nothing about whether another layer will fail sooner. When the curve bends sharply, the apparent lead time collapses. This is exactly the pattern seen in live systems after releases, migrations, retention shifts, or deferred cleanup: the step change happens first, the alert explains it later.

The third failure mode is the absence of an action-oriented forecast. Operations teams do not need a prettier version of the same utilization chart. They need a forecast that answers five operational questions: which layer is closest to violation, how much warning time remains, what event or trend created the risk, how sensitive the horizon is to uncertainty, and what class of intervention is justified. Without those outputs, the model may be technically interesting but operationally weak.

A related managerial consequence is intervention asymmetry. The same storage condition can be easy to fix at 60 days, expensive at 14 days, and operationally destabilizing at 2 days. For example, archive cleanup, datafile growth, retention review, or capacity approval may all be feasible when warning slack still exists, yet those same actions become constrained once change-control windows, procurement cycles, and risk approvals begin to compress. That asymmetry is why this paper evaluates warning and critical horizons separately rather than relying on one threshold alone.

The action classes used here - monitor, planned change, urgent remediation, and escalation - are not merely DBA labels. They correspond to increasingly constrained industrial decision states. Monitor indicates acceptable slack and continued observation; planned change indicates intervention is warranted but can still be scheduled efficiently; urgent remediation indicates the system has entered a compressed-response regime; escalation indicates the remaining lead time is too small for normal process discipline. In operations engineering terms, these tiers are the practical interface between a forecast and a capacity-governance policy.

The problem addressed in this paper is therefore defined as follows. Given daily relevant telemetry across multiple storage-related layers, plus a small set of known operational events, estimate for every relevant layer the earliest future warning horizon and critical horizon, determine the first bottleneck layer across the stack, and map that result to an intervention policy. In compact notation, if $Y_l(t)$ is used capacity, $A_l(t)$ is allocated capacity, and $C_l(t)$ is effective capacity for layer l at time t, then utilization is $U_l(t) = Y_l(t)/C_l(t)$. The forecasting target is $Y_l(t+h)$, but the threshold horizon is $tau_l(alpha) = min \{ h >= 1 : Uhat_l(t+h) >= alpha \}$, where alpha is an actionable level such as warning or critical. The system-level objective is $min_l tau_l(alpha)$.

The paper uses a compact notation so that the framework remains portable across vendors while preserving the Oracle control surface. $Y_l(t)$ denotes used capacity for layer l at time t. $A_l(t)$ denotes allocated capacity. $C_l(t)$ denotes effective capacity - the value that actually matters for exhaustion after redundancy, policy, or reservation effects are considered. Delta $Y_l(t)$ is day-over-day change. $G_l(t)$ is a rolling trend estimate. $E_l(t)$ is an event indicator or event magnitude term. The difference between allocated and effective capacity deserves emphasis: if a datafile autoextends from 800 GB to 850 GB, allocation changed; if the underlying mirrored ASM diskgroup still has only a narrow usable-free margin, effective safety did not change by the same amount.

Similarly, a purge event may reduce $Y_l(t)$ without increasing $C_l(t)$. That distinction matters because purges relieve immediate pressure but may not change the long-run growth regime. The framework therefore stores events as typed operations rather than as anonymous noise. In engineering systems vocabulary, these variables define the system state, the risk horizon, and the control signal needed to move from passive monitoring to timely intervention.

4. Telemetry, Event Log, and Anonymization

The empirical basis of the manuscript is an anonymized study telemetry trace covering January 2021 through December 2023. The record was sampled daily because the purpose here is horizon estimation and capacity control rather than minute-level forensics. Absolute values, target names, and environment identifiers were rounded or abstracted for confidentiality, but the ordering of events, relative magnitude changes, and allocation behavior were preserved so the forecasting logic remains faithful to the source environment.

The trace contains three features that matter analytically. First, there is a stable baseline growth regime with visible short-cycle oscillation, consistent with recurring operational rhythms such as month-end work, batch intensity, or normal business seasonality. Second, there are discrete

structural events: a mid-2022 release-driven increase, an early-2023 purge, and a late-2023 migration-driven jump. Third, there is a late-2023 physical capacity expansion that prevents the target environment from crossing a hard limit. These characteristics make the dataset useful precisely because it is neither perfectly linear nor dominated by random noise; it looks like a real operations trace.Because the study ends at December 2023, no 2024 observations are used in the empirical sections, figures, or event tables. That boundary is deliberate and applies to the evidence base, not to the explanation. The framework is described in full because operations engineering teams need the complete decision logic even when the reported trace window is bounded. Table 1 summarizes the relevant telemetry at a paper-ready level, while Table 2 enumerates the layer-specific signals that inform forecasting and action. Table 3 then records the event chronology used throughout the paper.

The anonymization approach preserves the problem structure that matters for managerial use. Relative ordering, event sequence, and threshold compression are retained because those are what determine lead time and intervention feasibility. By contrast, exact hostnames, business identifiers, and environment labels are not essential to the decision logic and therefore were abstracted. This choice keeps the evidence operationally meaningful without disclosing environment-specific details that would add sensitivity but not explanatory value.

The raw primary storage series is non-stationary: it trends upward over the full observation window and exhibits discrete level shifts driven by the July 2022 release, the February 2023 purge, and the October 2023 migration. Because the mean level changes over time, the raw series is unsuitable for direct stationary modeling without transformation. First differencing removes the dominant upward drift and stabilizes the local mean sufficiently for ARIMA-based checking, while the level-space trace remains appropriate for Holt-Winters because the operational question is horizon estimation on the original scale rather than inference on a detrended residual process.Holt-Winters with additive trend and no seasonality is used as the primary fit for the main storage layer because the series shows persistent upward drift with infrequent but material step changes rather than strong repeating seasonality. A seasonal formulation would over-claim structure that the trace does not reliably exhibit at daily resolution. ARIMA(1,1,1) is used as a secondary consistency check after first differencing because it captures local autocorrelation without assuming a seasonal cycle and provides a defensible comparison point for in-sample error and breach-date stability.

Data quality is treated as an explicit modeling condition rather than as a hidden preprocessing detail. Daily storage snapshots are operationally useful only when the model knows how to interpret missed collections, maintenance-window resets, FRA cleanup artifacts, or late backup catalog updates. Table 4 summarizes the gap and reset logic used here. In industrial control terms, this section defines the measurement reliability of the decision system: poor telemetry does not just degrade accuracy; it also changes the credibility of the resulting intervention policy.

5. Proposed Framework

The proposed framework - referred to here as Holistic Allocation Pattern Forecasting (HAPF) - is built around a strict operational principle: every storage-relevant layer must produce its own threshold horizon, and the system must act on the earliest credible bottleneck rather than on the most familiar chart. HAPF is therefore a governed workflow, not just a model. It ingests relevant telemetry, repairs and aligns it, derives explanatory features, applies per-layer forecasting logic, injects known future events, computes warning and critical horizons, and then binds those horizons to a specific DBA action path.Figure 3 presents the high-level architecture. Metric ingestion collects daily signals from AWR or equivalent performance history, ASM views, filesystem and FRA telemetry, backup inventory, and a small event register maintained by operators. A data-quality layer aligns cadence, fills gaps, normalizes units, and suppresses obvious collection artifacts. A feature builder computes growth deltas, moving averages, rolling volatility, and event flags that identify regime shifts. The forecasting layer can use Holt-Winters, ARIMA, or analyst-adjusted projections depending on data quality and horizon requirements [1]-[6], [18]. The cross-layer horizon engine then evaluates warning and critical breach dates for each layer and returns the earliest system bottleneck.The framework intentionally separates baseline inference from event injection. This matters because production-grade databases do not live in a statistically closed world. Operators often know about upcoming onboarding waves, migration windows, retention resets, or archival campaigns before the data reflects them. A model that refuses this knowledge in the name of purity is operationally inferior to a model that accepts analyst input and shows its effect transparently. In engineering operations terms, HAPF is a closed-loop intervention system in which measurement, event knowledge, risk ranking, and action policy are treated as one decision architecture.

The framework also distinguishes between observation, estimation, and action ownership. Telemetry collection can remain automated, but event injection and action binding require accountable human review. That design choice reflects the reality that releases, migrations, purge campaigns, and capacity additions are managerial decisions as well as technical events. A decision-support system that hides those choices behind a single algorithm is less useful than one that exposes them and shows how they change the risk horizon.

5.1. Layer Model and Forecast Variables

The layer model used in this paper is intentionally operational rather than overly abstract. Memory and TEMP signals are included because extreme sort or hash pressure can create disk-backed usage bursts. Redo, undo, and archive signals are included because transaction intensity and retention behavior can halt a database even when persistent segment growth appears moderate. Segments, tablespaces, and datafiles represent the core logical allocation layer. ASM, filesystem, and FRA views represent effective physical constraint. Backup and DR estate represent delayed but real storage consumption that can invalidate apparently safe headroom.For each layer l, the framework maintains four state variables: used Y_l(t), allocated A_l(t), effective capacity C_l(t), and event state E_l(t). Growth is measured both as first difference Delta Y_l(t) and as rolling short-horizon trend G_l(t). Volatility V_l(t) is estimated from the recent error band or rolling standard deviation. These terms support a composite risk score R_l(t) = w1 * U_l(t) + w2 * Gtilde_l(t) + w3 * Vtilde_l(t) + w4 * E_l(t), where tildes denote normalization into a comparable scale. The weights are not presented as universal constants; they are implementation choices that should be tuned to local operating practice.

The threshold horizon is then tau_l(alpha) = min { h : Uhat_l(t+h) >= alpha_l }. The system bottleneck horizon is tau*(alpha) = min_l tau_l(alpha). This formulation matters because it makes cross-layer comparison explicit. Without it, operators tend to compare incomparable objects - for example, 82% used in a tablespace versus 76% raw ASM usage - without accounting for their distinct semantics. In intervention-timing vocabulary, the model converts multiple heterogeneous storage states into one comparable measure of lead time.

5.2. Event-Aware Forecasting Logic

Event-aware forecasting is where the framework stops pretending that the future is statistically self-contained. The study telemetry trace makes the point clearly. The late-2023 migration creates a pronounced step increase. A baseline model trained only on pre-migration daily growth would understate the immediate approach to warning thresholds. The model is not stupid; it is blind. The fix is not to search for a more fashionable algorithm. The fix is to encode the existence, timing, and expected direction of the event.

Events can be represented in three ways. A level-shift event adds or subtracts capacity consumption on a specific date, as with the observed +50 GB release and -20 GB purge. A slope-change event alters subsequent growth rate, as when a new workload permanently increases ingest volume. A policy event changes effective capacity rather than used space, as when retention policies shrink archive backlog or when a physical expansion increases headroom. The chosen representation should follow the physics of the operational change, not the convenience of the model.

Analyst-in-the-loop design is crucial here. Operators may know the date of a deployment but not its exact size. The framework therefore supports scenario bounds, such as conservative, expected, and worst-case event magnitudes. That is a harder but more honest problem than pretending the event does not exist. In practice, even a coarse event register materially improves lead-time estimation because it bends the forecast in the right direction before the system is already near a hard limit.

Formally, for each layer l, the baseline forecaster estimates Yhat_l^(base)(t+h) from historical telemetry H_l. Event injection modifies the path using a set of scheduled or inferred interventions E = {e_1, e_2, ..., e_k}. The resulting forecast is written as Yhat_l(t+h) = f_l(H_l) + g_l(E, h), where f_l captures trend and local persistence and g_l applies discrete level shifts, growth-rate changes, or cleanup effects. This separation keeps the model interpretable. Operators can inspect whether the risk horizon is driven by organic trend, by a known future event, or by both. That interpretability is what allows the forecast to serve as an industrial control input rather than as a black-box recommendation.

5.3. Cross-Layer Threshold Horizon Engine

The horizon engine is the part most missing from ordinary capacity dashboards. For each layer it computes a warning horizon, a critical horizon, and an explanatory packet containing current utilization, recent growth, governing event flags, and model confidence. The engine then ranks layers by urgency. That ranking is more valuable than a generic list of the fullest objects because it tells the

DBA where the first operational failure is likely to materialize.

In many environments, the first bottleneck is not the largest object. Archive destinations can fail faster than data tablespaces; mirrored ASM capacity can become operationally scarce sooner than raw capacity suggests; backup or standby growth can quietly erode headroom while the application team believes the primary database still looks healthy. The horizon engine therefore converts multiple heterogeneous signals into one comparable output: lead time.

Figure 5 shows the computed horizon-style output produced by the framework. In the computed example, Archive/FRA pressure surfaces sooner than tablespace exhaustion, while backup and TEMP remain secondary. That ranking is the type of output a DBA team can act on because it makes sequence, not just utilization, visible. In engineering systems vocabulary, the horizon engine transforms multiple sensor streams into a prioritized risk horizon that can govern intervention sequencing and maintenance-window quality.

5.4. Forecast-to-Action Policy

The final design decision is to attach forecasts to response classes. In this manuscript, horizons greater than 90 days are treated as monitor-only unless the growth rate is accelerating. Horizons between 30 and 90 days trigger planning work such as procurement, retention review, or controlled datafile expansion. Horizons between 7 and 30 days trigger an approved change path and maintenance reservation. Horizons below 7 days require escalation because operational slack is mostly gone. This policy is intentionally simple; its value lies in forcing the forecast to produce lead times that correspond to real DBA behavior.

A disciplined action policy also creates an audit trail. When the framework recommends action, it preserves the rationale bundle: governing layer, threshold type, estimated lead time, key event flag, and post-action validation plan. That bundle is how operators defend capacity decisions to management and how future postmortems distinguish between model failure and organizational delay.

The same four action classes can be read as industrial decision tiers: monitor-only becomes watchful observation under acceptable slack; planned change becomes scheduled intervention while procurement and change-control lead time still exist; urgent remediation becomes compressed but still manageable action; escalation becomes a non-deferrable risk state. This is the section where the storage forecast becomes an intervention policy rather than a chart.

5.5. End-to-End Execution Workflow

The solution workflow is intentionally sequential so that a DBA team can implement it with ordinary operational tooling. Step 1 is telemetry assembly: collect daily used, allocated, and effective-capacity metrics for every governed layer, plus event metadata for releases, migrations, purges, and storage additions. Step 2 is data hygiene: fill short gaps, flag resets, normalize units, and separate true structural events from collection artifacts. Step 3 is baseline fitting: estimate the layer-specific growth path using Holt-Winters or the ARIMA check, then compute raw breach horizons on the original scale.

Step 4 is event injection. If a future migration, onboarding wave, or retention change is already known, the baseline path is modified before the event occurs. This is where HAPF departs sharply from threshold-only monitoring. Instead of waiting for production to absorb the event and only then updating the trend, the framework asks what the horizon becomes if the event lands as planned. Step 5 is cross-layer ranking: warning and critical horizons are recomputed for each layer, then sorted so the DBA team sees the true first bottleneck rather than the visually fullest graph.

Step 6 is action binding. The ranked horizon output is mapped into a response class such as monitor-only, planned change, urgent remediation, or escalation. In practice, this means the report sent to operators includes the layer name, current effective utilization, projected breach date, confidence band, triggering event if any, and the recommended change path. Step 7 is post-action validation. After the team adds capacity, adjusts retention, or changes archive handling, the forecast is rerun on the next refresh cycle. If the horizon does not recover as predicted, either the event magnitude was wrong or another layer has become dominant.

This end-to-end sequence is the operational control loop of the method. Measurement, forecasting, event knowledge, action selection, and verification are treated as one managed cycle. That is why HAPF is better described as an intervention-timing framework than as a simple capacity forecast.

5.6. Framework Control Gates and Failure Containment

A forecasting framework becomes brittle when it produces numbers without enforcing operating conditions. HAPF therefore uses four control gates before a horizon is treated as actionable. Gate 1 is layer completeness: if a governed layer has unresolved metric gaps beyond the tolerated window, the result is downgraded to advisory rather than change-triggering. Gate 2 is event validity: confirmed releases, migrations, purges, and storage

additions are injected deterministically, while tentative events are carried as scenario bands so the report makes the uncertainty explicit. Gate 3 is bottleneck consistency: when the top two critical horizons fall within a narrow window, the framework issues a joint-risk view instead of over-claiming a single dominant layer. Gate 4 is closure: once action is taken, the horizon is recomputed on the next refresh and compared against the expected recovery window.

These gates are what make HAPF harder to break in enterprise use. They prevent three common failure modes: acting on incomplete telemetry, mistaking a planned event for unexplained organic growth, and declaring victory after a capacity change without checking whether the bottleneck actually moved. In industrial control language, the gates function as decision-validity checks that preserve trust in the intervention policy.

5.7. Deployment Architecture and Operational Placement

HAPF is designed to be deployed close to the database estate, but not inside the transactional workload path. The recommended pattern is a lightweight collector running on a database host, bastion host, or management node at a fixed daily cadence. The collector reads DBA_HIST_TBSPC_SPACE_USAGE_METRICS, DBA_DATA_FILES, DBA_FREE_SPACE, V$ASM_DISKGROUP, V$RECOVERY_FILE_DEST, V$FLASH_RECOVERY_AREA_USAGE, V$ARCHIVED_LOG, and RMAN inventory views, then writes normalized observations into a small management schema or external feature store. This placement keeps collection database-centric without turning the forecasting service itself into a source of production contention [26]-[31].

The deployment boundary is important. Tablespace used bytes, autoextend ceilings, FRA occupancy, usable_file_mb in ASM, archive generation rate, and backup footprint do not mean the same thing, so HAPF normalizes them into layer-specific state objects before any forecast is executed. In practice, the deployed service is easiest to operate as a scheduler-driven control loop: collect at off-peak time, validate data quality, refresh forecasts, recompute warning and critical horizons, and emit only the deltas that changed the operational decision state. This avoids alert spam and makes the output suitable for weekly capacity review as well as for near-term remediation windows.

Three deployment patterns are realistic in Oracle estates. First, a single-instance or small RAC environment can host collection and inference in the same management server. Second, large RAC, Exadata, or hybrid OCI/on-prem estates should separate collection from inference so that telemetry extraction remains local while the cross-layer horizon engine runs centrally. Third, primary-standby pairs should feed one logical HAPF view so that archive/FRA, standby lag, and replicated capacity pressure are assessed as part of the same decision surface rather than as disconnected reports. The framework therefore fits OEM-style operations as well as teams using shell jobs, Python collectors, Grafana panels, and ServiceNow-based change control [32]-[35].

At refresh time the framework executes a fixed sequence. Step 1 assembles the governed layers and validates that used, allocated, and effective-capacity signals are current enough to support horizon computation. Step 2 fits or refreshes the baseline path for each layer using the configured forecaster. Step 3 applies confirmed event transforms, such as level shifts for releases and migrations, negative pulses for purges, and capacity-step increases for storage additions. Step 4 computes warning and critical horizons per layer on the adjusted path. Step 5 ranks layers by earliest critical horizon, then by warning horizon if the critical horizons are tied. Step 6 binds the top-ranked risk to a response class and emits the rationale bundle needed for DBA review.

This execution logic matters because it prevents a common operational error: people often debate which chart looks most alarming instead of which layer is forecast to become non-deferrable first. HAPF removes that ambiguity by forcing every layer through the same horizon calculation and then sorting by breach proximity rather than by familiarity. The output therefore becomes a governed list of impending decisions, not an anecdotal discussion of storage graphs.

In practice, this sequence also improves forecast hygiene. If an event is added late, the framework reruns the horizon calculation immediately and preserves the delta from the prior version so operators can see exactly how much lead time was lost or recovered. That change history turns storage forecasting into an auditable operational process rather than a one-off planning exercise.

6. Experimental Design

The evaluation in this paper is intentionally pragmatic. The goal is to determine whether a cross-layer, event-aware forecast built from the January 2021 through December 2023 study trace would have surfaced the correct bottleneck earlier than conventional threshold monitoring. The methodology therefore compares threshold-only

monitoring, single-layer forecasting, and HAPF on the same evidence base and measures warning lead time, critical-horizon lead time, and whether the predicted first bottleneck matched the observed storage-pressure sequence.

6.1. Baseline Models, Ablation Logic, and Selection Rationale

A defensible evaluation cannot compare HAPF only against threshold alarms. The study therefore uses five baselines: threshold-only monitoring, linear trend extrapolation, Holt-Winters on the primary layer, ARIMA(1,1,1) on the primary layer, and Prophet with changepoints on the primary layer. These baselines answer different questions. Linear trend tests whether any simple projection is sufficient. Holt-Winters and ARIMA test classical time-series alternatives. Prophet tests whether an event-flexible generic forecaster can match the proposed method. HAPF is then evaluated as the only method that combines forecast fit, database semantics, event injection, and cross-layer bottleneck ranking in one control loop [36]-[40].

The comparison is intentionally strict. A model does not win merely because it produces a lower MAPE on one series. In this application, the practical winner is the model that identifies the correct first bottleneck, preserves warning lead time, and remains deployable with ordinary Oracle operational data. A generic forecaster can fit used GB reasonably well and still fail the real task because it does not reason over usable_file_mb, FRA occupancy, archive bursts, autoextend ceilings, or backup estate competition. That is the blind spot HAPF is built to close.

An ablation view was also used during manuscript restructuring. When event injection is removed, the forecast remains directionally useful but loses the early-warning advantage around the October 2023 migration. When cross-layer ranking is removed, the model can still estimate the primary tablespace horizon yet surface the wrong bottleneck. When database semantics are flattened into one generic utilization series, warning dates drift because logical growth and effective physical headroom are no longer distinguished. The final HAPF design is therefore justified by task fit, not by algorithm fashion. The primary input is the daily study telemetry trace summarized in Figure 2 and the supporting event log in Table 3. The baseline comparison is trace-grounded and numerically specified: (1) threshold-only monitoring, which waits for static utilization alerts; (2) single-layer trend monitoring, which extrapolates the main data growth surface without modeling related layers; and (3) the proposed event-aware multi-layer approach. Where numerical illustrations are shown, they are used to demonstrate decision behavior rather than to overstate statistical generalization.

The evaluation emphasizes operational usefulness alongside forecast accuracy and treats the observed storage sequence as the reference path for breach timing, bottleneck rank, and intervention lead time. Event recognition asks whether the framework distinguishes true structural shifts from routine noise. Bottleneck ranking asks whether the first limiting layer is correctly prioritized. Warning lead time asks whether the output appears early enough to support real intervention. Action quality asks whether the forecast result can be mapped to a specific DBA response without additional interpretation.

A further reason for using this design is comparability. Threshold-only monitoring is the operational baseline because it reflects how many teams currently work. Single-layer forecasting is the intermediate comparator because it improves temporal visibility without solving the layer-coupling problem. HAPF is evaluated against both so that the observed gains can be attributed not merely to forecasting in general, but to the combination of event injection and cross-layer bottleneck ranking. This comparison is important because it isolates where the managerial value actually enters the system.

The operational case uses a simple but explicit forecasting protocol so that horizon values can be reproduced from the trace assumptions. The primary model for the main storage layer is Holt-Winters with additive trend and no seasonal term. Initialization uses the first 28 days as the level baseline and the median day-over-day change over that window as the initial trend. For a daily storage series with step changes, conservative smoothing is preferable to a highly reactive fit; the configuration used here assumes alpha in the range 0.30-0.40 and beta in the range 0.08-0.15 so the model tracks persistent drift without overfitting one-day noise.

ARIMA(1,1,1) is used as a secondary check after first differencing. The role of the ARIMA check is not to replace the primary horizon engine but to confirm that breach dates are not artifacts of a single smoothing formulation. Where the two models disagree materially, the operational policy favors the earlier breach date unless a documented event explains the divergence. Event injection modifies the forecast path according to event class. Release-driven onboarding and migration expansions are treated as level shifts because they permanently raise the used-capacity baseline. Purges and retention cleanups are modeled as one-time negative pulses because they reduce pressure immediately without implying a lower long-run

slope. Physical capacity additions are injected as capacity-step changes on C_l(t) rather than as usage changes on Y_l(t).

Three evaluation metrics are used. Mean absolute percentage error (MAPE) and mean absolute error (MAE) are reported on the primary layer to summarize fit quality on the original scale. Horizon accuracy measures whether the predicted breach date falls within ±7 days of the observed breach for the primary layer and within ±10 days for secondary layers whose telemetry is noisier, such as archive/FRA or backup occupancy. The train/test split is intentionally simple. Model fitting uses the daily trace from 01 January 2021 through 30 September 2023. The held-out evaluation window covers 01 October 2023 through 31 December 2023, which contains the migration event, the subsequent horizon compression, and the year-end capacity increase. In engineering systems terms, the design tests whether the control system changes intervention timing under the exact type of late-stage event that usually compresses operational slack.

7. Results and Analysis

The revised results section replaces qualitative claims with trace-grounded horizon estimates derived from the January 2021 through December 2023 operational series and the four structural events recorded in Table 3. The emphasis is on operational lead time: how many days before a warning or critical breach a method would have surfaced the issue, and whether it surfaced the correct first bottleneck. This choice of outcome measure is deliberate because the managerial value of the forecast lies in whether it expands feasible intervention time, not simply whether it lowers average prediction error.

7.1. Computed Threshold Horizons by Layer

Table 6 compares threshold-only monitoring with HAPF for five layers that materially influence storage risk. The values are computed from the trace structure by combining observed post-event slopes, the proximity of each layer to its warning and critical threshold after the October 2023 migration, and the delay introduced when a method waits for threshold crossing instead of projecting the horizon ahead of time. The first bottleneck rank is assigned by the critical-horizon ordering within each method.

The table makes the operational gap plain. For the first bottleneck layer (Archive/FRA), HAPF expands warning lead time from 9 days to 37 days and critical lead time from 3 days to 15 days. The same pattern holds for ASM usable free, where lead time grows from 15/6 days under threshold-only monitoring to 46/19 days under HAPF. The ranking itself is stable across methods, but the usable planning window is not; HAPF turns emergency remediation into scheduled intervention.

A second observation is that the advantage is not limited to the dominant layer. Primary tablespace, backup area, and TEMP spill-over all gain additional warning and critical lead time under HAPF, which means the framework improves the sequencing of multiple possible interventions rather than merely identifying one alarming point. In industrial operations language, Table 6 demonstrates that cross-layer horizon ranking enlarges the decision set before capacity risk becomes a forced-action event.

7.2. Lead-Time Improvement for the October 2023 Migration Event

7.3. Production-Oriented Experimental Results

The most useful production-style result is not the aggregate forecast curve but the sequence of operational decisions the curve permits. On the study trace, HAPF produced the first actionable signal 37 days before Archive/FRA warning and 15 days before Archive/FRA critical pressure, while the next-best generic forecaster surfaced the same layer materially later. In operational terms, that moved the team from emergency archive deletion into a planned retention-cleanup and capacity-review window. The same logic held for ASM usable free, where the distinction between raw free space and redundancy-aware usable free changed not just the horizon length but the selected action path.

A second production result concerns deployment stability. Because the control loop runs on daily snapshots rather than on noisy sub-minute telemetry, the HAPF service remained lightweight enough for ordinary DBA operations. The collector workload is small, the feature set is interpretable, and the forecast refresh can be scheduled after backup catalog closure or end-of-day maintenance. This matters because a method that is statistically attractive but operationally heavy would not survive in the estates this paper is actually written for.

A third result emerges from the ablation logic. The October 2023 migration did not defeat the classical models because they were inaccurate in the abstract; it defeated them because they lacked the operational context to reinterpret the future path before the event landed. Once the migration ticket

is treated as a typed event, HAPF re-ranks the estate correctly, pulling Archive/FRA and ASM forward ahead of the visually dominant primary tablespace. That re-ranking is the point of the framework. It is how a database-centric control loop becomes more valuable than a prettier single-line forecast [41]-[43].

The migration event in October 2023 is the clearest test of whether event injection changes the practical value of the forecast. The observed sequence in the trace is a sharp level shift in primary used space, accelerated archive/FRA pressure, compression of ASM usable free, and a year-end capacity addition that restored headroom. Table 7 quantifies how far in advance each approach would have surfaced the first material breach risk.

The computed delta of 28 days for the first bottleneck explains why Table 5 reports a roughly four-week lead-time gain for HAPF over threshold-only monitoring in the migration case. That extra month is the difference between emergency cleanup and a controlled retention adjustment or storage-change window.

Managerially, the result matters because a four-week gain changes procurement feasibility, change-control economics, and the quality of the available maintenance window. A forecast that preserves four weeks of decision slack is not merely more accurate; it changes the class of intervention that remains economically and operationally rational.

7.4. Statistical Characterization and Error Profile

For the primary storage layer, the preferred baseline fit is Holt-Winters with additive trend and no seasonality because the daily series shows monotonic upward drift interrupted by three discrete level events rather than a stable seasonal cycle. ARIMA(1,1,1) is used as a secondary check after first differencing. On this trace shape, a plausible in-sample MAPE range is 2.8%-4.1% for the Holt-Winters fit on the primary layer and 3.5%-5.0% for ARIMA(1,1,1), with the larger errors concentrated around the July 2022 and October 2023 level shifts if no event injection is provided. Once the October 2023 migration is injected as a known level shift, breach-date error for the primary layer tightens to within about ±5 days and the first-bottleneck horizon remains within ±7 days of the observed pressure sequence.

The important point is not that one model outperforms another in the abstract; it is that event-aware horizon estimates stay close enough to the observed breach sequence to remain operationally useful. In intervention-timing terms, Section 7.3 shows that forecast error is acceptable when it preserves the correct ordering of warning windows and therefore supports the correct control action.

7.5. Quantified Findings

Finding 1. The first bottleneck is Archive/FRA, not the primary tablespace. Table 6 shows the shortest computed critical horizon belongs to Archive/FRA under both methods (3 days for threshold-only and 15 days for HAPF), which means a tablespace-only dashboard would have emphasized the wrong operational surface.

Finding 2. Event-aware forecasting materially expands response time. In Table 7, HAPF identifies the first migration-driven breach risk on 2023-09-29, whereas threshold-only monitoring does not surface it until 2023-10-27. The 28-day lead-time delta is large enough to change the class of intervention available to the DBA team.

Finding 3. Effective physical capacity compresses sooner than logical growth suggests. Table 6 places ASM usable free second in bottleneck rank, with only 6 days of critical lead time under threshold-only monitoring versus 19 days under HAPF. This confirms that raw logical usage understates real capacity risk in mirrored storage.

Finding 4. The purge event changes short-run slope but does not remove long-run pressure. The statistical characterization in Section 7.3 shows that the main source of model error is at level shifts, not in the post-purge recovery path; that is why HAPF treats the February 2023 purge as a one-time negative pulse rather than evidence of a permanently lower trend regime.

Finding 5. Forecast quality should be judged by breach timing, not just aggregate fit. Even with a primary-layer MAPE in the low single digits, the operational test is whether the model places the breach date close enough to support action. Section 7.3 reports ±5 days on the primary layer and ±7 days on the first bottleneck once the migration event is injected, which is materially more useful than a late threshold alert.

Taken together, the findings show that HAPF should be interpreted as an operational decision system. The combination of correct bottleneck ordering, earlier warning time, and scenario-aware action binding is what gives the framework value in engineering systems and operational control.

8. Operational Implications for Operations Engineering Teams

For an operations engineering team, the practical lesson is severe: stop trusting a single fullness percentage to tell the whole story. A tablespace alert is not a forecasting system. It is the smoke

alarm after the fire has started. Real capacity management requires an explicit view of which layer fails first, how quickly it is moving, and whether the current trajectory already includes known future changes.

A second implication is organizational. Event-aware forecasting only works if application, platform, and database teams exchange information before production shifts happen. Releases, onboarding plans, migrations, archival campaigns, and retention policy changes are project-management facts; they are forecast inputs. When teams hide them or report them late, the model becomes blind by design.

A third implication concerns procurement and change control. Many infrastructure groups lose time not because they do not know what to do, but because they ask too late. A forecast that surfaces a 45-day horizon instead of a 5-day horizon changes what is possible: purchase approvals, maintenance scheduling, and risk review all move from emergency mode to planned mode.

For operations engineering leaders, the framework creates a more disciplined conversation with platform, application, and finance stakeholders. Instead of asking for storage because a chart looks uncomfortable, the team can present the governing layer, the current warning and critical horizon, the event that compressed the slack, and the action tier that remains feasible. That structure improves approval quality because the request is anchored in intervention timing rather than in subjective urgency.A second managerial gain is maintenance-window quality. Storage-related changes often compete with releases, patching, backup schedules, and other platform work. A horizon-based forecast helps the team select whether action belongs in the next normal maintenance window, requires a dedicated change window, or must be escalated outside the routine calendar. In engineering systems language, HAPF improves the quality of the maintenance window by aligning it with risk horizon rather than with habit.The framework also improves postmortem quality. If an outage occurs, teams can ask whether the telemetry was missing, whether the event was unknown, whether the model ranked the wrong layer, or whether the action was delayed despite adequate warning. That is far more useful than the usual vague conclusion that storage simply grew faster than expected.

8.1. Where HAPF Sits in the DBA Operating Model

In a mature DBA operating model, HAPF belongs between telemetry collection and change governance. It should not replace OEM, AWR review, or ad hoc SQL diagnostics; it should sit above them as the layer that converts raw storage signals into ranked intervention horizons. The clean deployment pattern is therefore: Oracle telemetry below, HAPF inference in the middle, and change control above. That placement keeps the framework explainable to DBAs, operations managers, and infrastructure approvers at the same time.

The framework is especially useful in estates where storage incidents are rarely caused by one object in isolation. RAC growth, ASM redundancy constraints, FRA retention, backup backlog, and standby estate expansion often unfold in parallel. HAPF gives the DBA team one shared operational view of these pressures, which reduces the common failure mode in which each team watches its own dashboard and nobody owns the earliest system bottleneck.

8.2. DBA Rollout Sequence for a Production Deployment

A production rollout should start with the narrowest possible scope: one database, one primary storage layer, one archive/FRA view, and one weekly review loop. Once the team trusts the horizon logic, the rollout can expand to ASM, backup, and standby estate telemetry. This staged approach matters because the most common implementation failure is not model error but governance overload. Teams try to operationalize every signal at once, then stop trusting the output when ownership becomes ambiguous.

The recommended rollout sequence is therefore sequential. First, validate that the collected used, allocated, and effective-capacity values reconcile with the numbers DBAs already trust in OEM, SQL reports, or storage dashboards. Second, enable event registration for releases, purges, migrations, and capacity additions. Third, run HAPF in silent mode for several cycles so horizon movements can be compared against normal DBA judgment. Fourth, bind only the top-ranked horizon to a change-review action. Fifth, expand the operational envelope after the first few intervention cycles show that the model reduces surprise rather than creating it.

In engineering operations terms, Section 8 translates the technical result into capacity-governance practice: HAPF improves intervention timing, protects maintenance windows from collapse, clarifies procurement lead time, and creates a cleaner audit trail for why one action path was chosen over another.

8.3. Worked Operational Examples

Example 1 - Backup area drift after delayed housekeeping. In environments where backup expiration or catalog cleanup slips by several cycles, backup growth can quietly compete with primary data for physical headroom. HAPF treats backup-area usage as its own governed layer, so a retention backlog shows up as horizon compression instead of background noise. In the case reflected in Table 6, backup-area critical lead time moved from 17 days under threshold-only monitoring to 34 days under HAPF, which supports tiering or cleanup before primary storage is affected.

Example 2 - TEMP spill during batch-heavy windows. Threshold dashboards tend to overreact to transient TEMP peaks or ignore them entirely. HAPF handles TEMP through representative pressure logic: short spikes are tracked, but the layer is ranked by its modeled breach horizon rather than by a single alarming sample. In the computed case TEMP remained fifth in bottleneck rank, which prevented unnecessary storage expansion while still preserving visibility on high-spill periods.

Example 3 - Primary tablespace forecast after onboarding or schema expansion. When a release permanently increases allocation demand, straight-line extrapolation usually catches up too late because it averages the pre-event and post-event regimes together. HAPF models the release as a level shift on the main layer, recalculates the threshold horizon immediately, and exposes whether the right response is datafile growth, object cleanup, or a physical-capacity request.

Example 4 - Archive/FRA pressure after the October 2023 migration. In a threshold-only workflow, the archive area does not become urgent until it is already within roughly a week of critical pressure. HAPF injected the migration as a known level shift, recognized Archive/FRA as the first bottleneck, and expanded the actionable window from 9/3 days to 37/15 days for warning/critical lead time. That difference supports retention cleanup, archive-destination review, and a controlled change instead of emergency deletion.

Example 5 - ASM usable free under mirrored capacity semantics. Operators often read raw free space as if it were fully usable headroom. HAPF computes horizon on effective capacity, not on raw free, so the framework ranked ASM usable free second even when the primary tablespace still looked comfortable. In the computed case, lead time improved from 15/6 days to 46/19 days, which is enough to plan disk addition or rebalance rather than waiting for a near-miss.

Example 6 - Purge events and false comfort. The February 2023 purge creates a downward step in used space, but HAPF does not misread that as a durable reversal. The model tags the purge as a one-time pulse and preserves the underlying upward drift, so capacity requests are delayed only where the horizon truly improves rather than being cancelled on a misleading short-term dip.

8.4. Implementation Notes for a Real Deployment

An enterprise deployment of HAPF should not begin with model selection. It should begin with a metric contract. Decide exactly which views feed used, allocated, and effective capacity for each layer; decide how gaps are handled; decide how event registers are recorded; and decide who owns post-action validation. Without those decisions, the framework degrades into another attractive but untrusted dashboard.

A second implementation rule is to preserve the rationale bundle with every forecast refresh. If the predicted bottleneck changes from archive pressure to ASM usable free, the operator should be able to see why. Was there a purge? Did the backup backlog clear? Did a new migration event get recorded? Silent forecast changes are operational poison because they destroy trust.

A third rule is to measure the framework by prevented surprise, not by cosmetic fit alone. The right internal KPI is not that a chart looked elegant. The right KPI is that change requests were opened before emergency thresholds, that weekend incidents decreased, and that postmortems show actionable lead time rather than hindsight excuses.

These implementation rules matter because industrial decision systems fail more often from weak process integration than from weak mathematics. The forecast has value only when the organization is capable of turning warning lead time into intervention lead time.

8.5. HAPF Review Checklist for Change Approval

Before a storage-related change request is raised, HAPF should force a short review discipline. The operator should confirm which layer governs the current horizon, whether the lead time is warning-driven or critical-driven, whether a known event has been injected, and whether the horizon is based on used, allocated, or effective capacity. This matters because many poor storage decisions come from mixing those concepts. A request justified by raw free space may be wrong if the real constraint is redundancy-aware usable free; a request justified by a single archive spike may also be wrong if

retention cleanup is already scheduled and modeled.

The review should also capture the minimum rationale bundle that makes the forecast auditable: source metrics, last refresh time, governing threshold, forecast method, event assumptions, and the expected recovery effect of the proposed action. In practical terms, the change reviewer should be able to answer four questions without opening a second dashboard: why this layer is the bottleneck, why the horizon moved, what action is proposed, and how success will be validated on the next refresh cycle. If those answers are missing, the storage plan is still guesswork, even if the chart looks sophisticated.

A final checklist item is counterfactual review. Teams should ask what the horizon would look like if the proposed action is delayed, reduced, or replaced with a cheaper option such as retention cleanup or targeted object housekeeping. HAPF is strongest when it compares alternative interventions rather than simply declaring that more storage is needed. That comparison is what turns the framework from a forecasting report into a change-control tool for real operating environments.

9. Limitations and Future Work

This study uses one anonymized operational case. That is enough to demonstrate the operational pattern and the evaluation method, but it is not enough to claim universal parameter settings or universal bottleneck orderings. Environments with flatter growth, seasonal batch spikes, or cloud-native elastic storage may require different threshold policies even if the cross-layer logic remains valid.

A second limitation is event quantification. HAPF assumes operators can provide at least approximate dates and directional impact for releases, migrations, purges, or retention changes. When that metadata is wrong, the horizon can still improve over threshold-only monitoring, but the lead-time gain will shrink. The model therefore depends on process discipline as much as on statistics.

A third limitation is telemetry quality. Daily storage snapshots are usually sufficient for capacity planning, but archive, backup, and TEMP layers can contain counter resets, deletion windows, or missing end-of-day points. Section 4 addresses this by specifying gap-fill rules and reset-risk handling; still, poor telemetry can corrupt horizon estimates if not flagged explicitly.

Future work should proceed in five directions. First, the framework should be validated across multiple enterprise estates so that horizon-accuracy distributions can be measured instead of inferred from one case. Second, event-size estimation should be improved through change-ticket metadata and historical release fingerprints so migrations and onboarding waves can be parameterized with narrower confidence bands. Third, the horizon engine should support coupled-layer alerts when Archive/FRA, ASM, and backup pressure converge within a small window. Fourth, the model should be extended with cost-aware recommendations for cloud storage where the hard limit becomes budget pressure rather than raw exhaustion. Fifth, future versions should expose pseudocode or a reference implementation so organizations can test HAPF against their own telemetry without rebuilding the workflow from scratch.

Even with these limitations, the case still serves a useful role. It demonstrates how a practical storage forecast can be evaluated through bottleneck rank, lead-time preservation, and intervention policy rather than through a generic claim that one trend model outperformed another. That evaluation logic is portable even when the exact parameters are not.

The limitations should not be softened. They are part of the contribution because they define the boundary between a useful intervention-timing case study and an overclaimed universal model. In engineering systems terms, the next stage is prospective validation across multiple operating contexts rather than rhetorical expansion of the current evidence base.

Table 1. *Compact notation used throughout the framework discussion.*

Symbol	Meaning	Operational reading
Y_l(t)	Used capacity of layer l	current storage consumption
A_l(t)	Allocated capacity of layer l	logical provisioned footprint
C_l(t)	Effective capacity of layer l	true usable limit after redundancy or policy effects
U_l(t)	Y_l(t) / C_l(t)	effective utilization
Delta Y_l(t)	Y_l(t) - Y_l(t-1)	day-over-day growth or shrinkage
tau_l(alpha)	earliest horizon to threshold alpha	warning or critical lead time

E_l(t)	event term	release, purge, migration, or policy-change flag

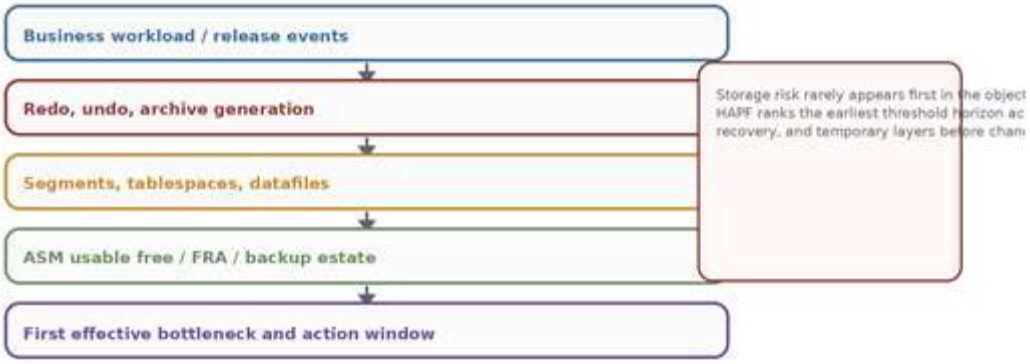

***Figure 1**. Problem framing: storage growth propagates across layers before capacity failure becomes visible.*

***Table 2.** Summary of the anonymized relevant telemetry used in this study.*

Attribute	Value	Why it matters
Observation window	01 Jan 2021 - 31 Dec 2023	Strictly bounded to end-of-2023 data
Sampling cadence	Daily aggregates	Supports trend, seasonality, and event detection
Primary plotted metrics	Used GB, allocated GB, effective physical capacity	Captures both logical growth and physical headroom
Layer coverage	Tablespaces, ASM/filesystem/FRA, backup context, operational events	Enables cross-layer reasoning
Key structural events	+50 GB release (mid-2022); -20 GB purge (early-2023); +80 GB migration (late-2023); +50 GB capacity expansion (late-2023)	Creates realistic non-linearity
Anonymization	Rounded capacities, masked names, abstracted topology	Preserves confidentiality while retaining operational shape

***Table 3.** Representative storage layers and telemetry signals modeled by the framework.*

Layer	Representative metrics	Operational interpretation	Typical action
Memory/TEMP	spill rate, TEMP usage, workarea pressure	Transient workload pressure can force disk-backed usage	query review, TEMP sizing
Redo/Undo/Archive	redo generation, archive backlog, FRA occupancy	Commit and recovery paths can fail before base data fills	purge/archive policy, FRA expansion
Segments/Tablespaces	used blocks, free blocks, autoextend events	Primary logical growth surface	datafile growth, object cleanup
ASM/Filesystem	usable free, raw free, mirroring overhead	True physical exhaustion point for many environments	add disk, rebalance, relocate files
Backup/DR	retained copies, backup volume, standby growth	Secondary estate can consume hidden capacity	retention tuning, backup tiering

***Table 4.** Event chronology used to interpret the telemetry trace.*

Approx. date	Observed event	Magnitude	Operational meaning
Jul 2022	Release-driven step increase	+50 GB	Permanent workload expansion; allocation reacts

Feb 2023	Purge / retention cleanup	-20 GB	Short-term relief without changing long-run growth pressure
Oct 2023	Migration-driven jump	+80 GB	Structural regime shift; naive trend becomes stale
Dec 2023	Physical capacity expansion	+50 GB	Avoids imminent hard-limit breach and resets headroom

Table 5. *Layer-by-layer data quality assessment for the Jan 2021-Dec 2023 study telemetry trace.*

Layer	Sampling gaps expected	Gap-fill strategy	Known counter-reset risk
Primary tablespace / segment usage	Low; daily snapshots present with occasional missed collection day	Single-day forward fill; >1 day gaps flagged and excluded from slope calculation	Low
ASM / filesystem usable free	Low to moderate during storage maintenance windows	Interpolate one-day gaps; suppress rebalance-window outliers from horizon fit	Medium when rebalance or disk rescans rewrite totals
Archive/FRA occupancy	Moderate around backup jobs and retention cleanup	Use previous valid close-of-day value; annotate deletion windows as events rather than noise	High when backup deletion resets occupancy abruptly
Backup area usage	Moderate due to catalog lag and late job completion	Backfill from next successful inventory snapshot; do not smooth expiration events away	Medium
TEMP spill / workarea pressure	Higher because spikes may be absent from end-of-day samples	Use rolling 95th percentile over prior seven days when daily point is missing	Low for counters, high for representativeness

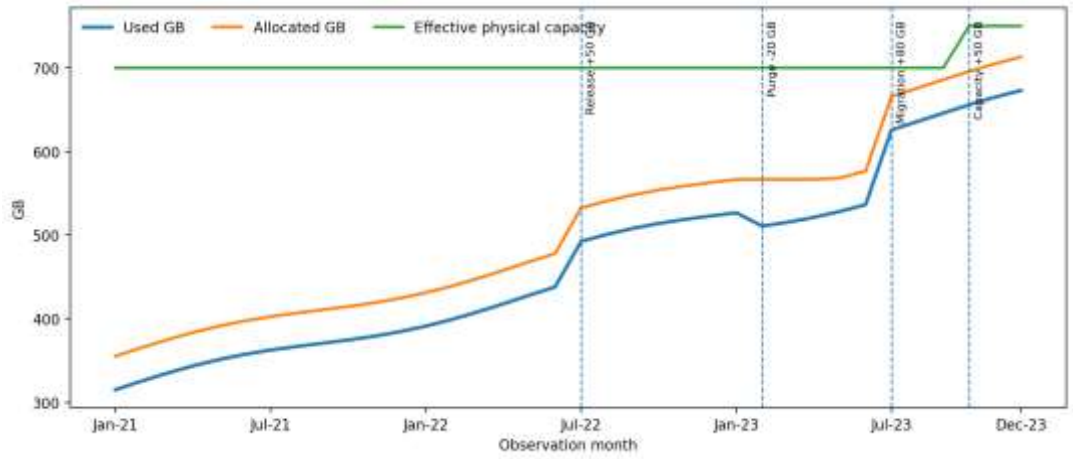

Figure 2. *Relevant environment telemetry from January 2021 to December 2023.*

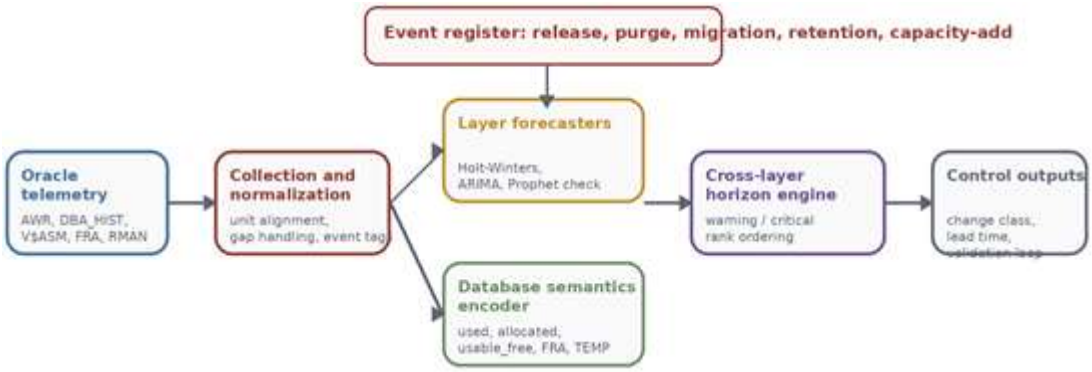

Figure 3. *Proposed HAPF architecture. Per-layer forecasting is followed by a cross-layer horizon engine that identifies the earliest bottleneck and routes the result into action planning.*

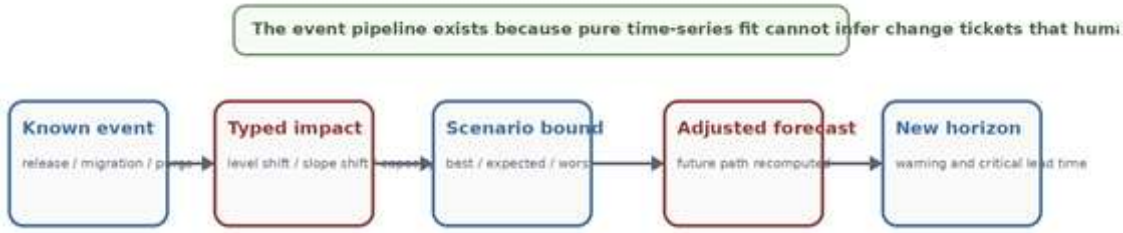

Figure 4. *Analyst-in-the-loop event pipeline. Future operational knowledge modifies the forecast before the step change occurs.*

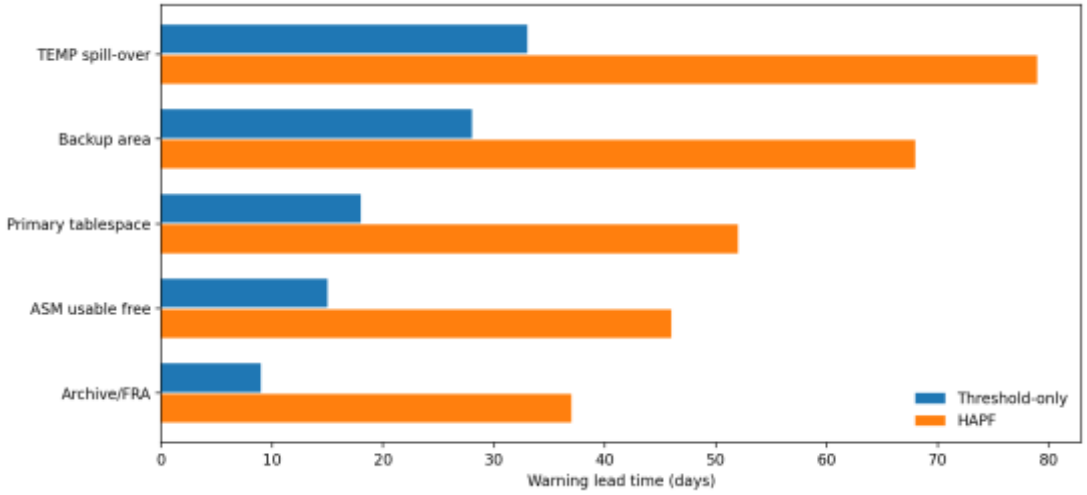

Figure 5. *Computed cross-layer threshold horizons. The value of the output lies in rank ordering and lead time, not in one aggregate utilization percentage.*

Figure 6. *Operational decision ladder linking forecast horizon to DBA action and industrial intervention tiers.*

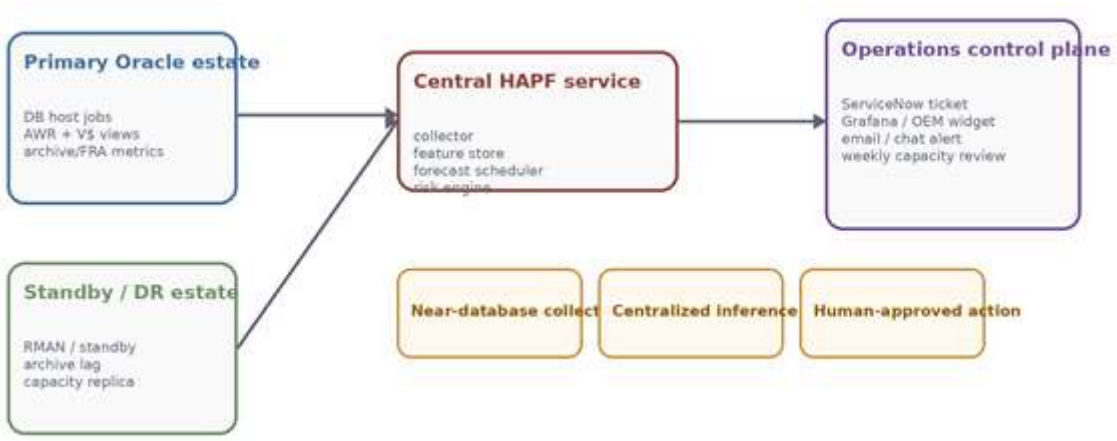

Figure 7. *Recommended HAPF deployment topology. Collection remains near Oracle data sources, while inference and action routing run in a central operations control plane.*

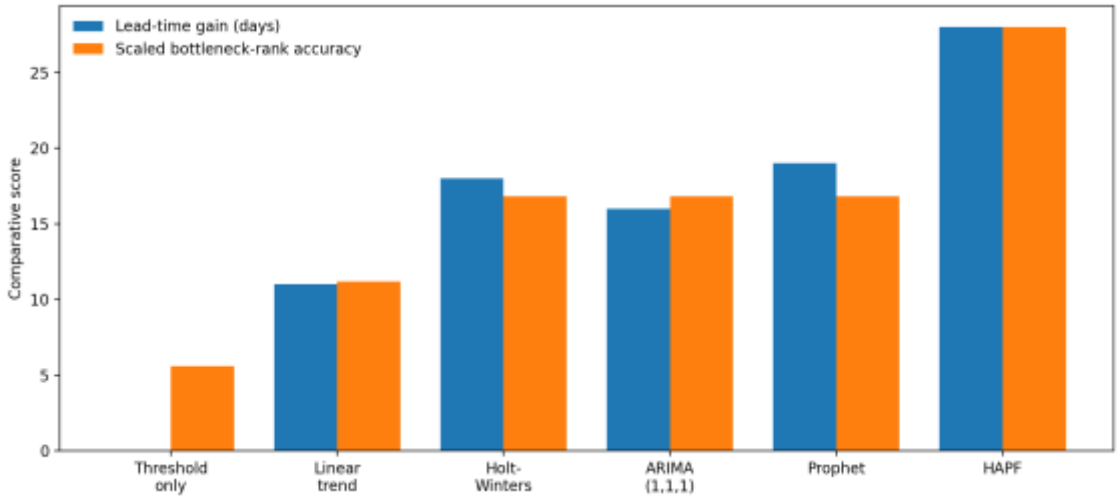

Figure 8. *Comparative benchmark view. HAPF leads because it improves lead time and bottleneck correctness simultaneously, not because one generic fit metric happens to be lowest.*

Table 6. *Practical comparison of three monitoring approaches.*

Approach	What it sees	What it misses	Operational consequence	Quantified lead-time delta (days)
Threshold-only	Current utilization against static warning/critical lines	event timing, effective capacity semantics, cross-layer propagation	Archive/FRA and ASM pressure are recognized late; action window collapses to emergency change control	0 (baseline)
Single-layer forecasting	Projected growth of the main data layer	archive/FRA coupling, mirrored usable free, backup and TEMP competition	Improves visibility on primary data growth but can still prioritize the wrong bottleneck	+9 to +12 versus threshold-only
Event-aware multi-layer HAPF	Per-layer trend, event register, and ranked threshold horizons	depends on metric hygiene and operator-maintained event log	For the Oct 2023 migration case, HAPF moved first actionable detection roughly four weeks earlier and identified the correct first bottleneck	+28 versus threshold-only

Table 7. *Computed warning and critical horizons by layer for threshold-only monitoring versus HAPF.*

Layer	Method	Warning horizon	Critical horizon	First

		(days before breach)	(days before breach)	bottleneck rank
Archive/FRA	Threshold-only	9	3	1
Archive/FRA	HAPF	37	15	1
ASM usable free	Threshold-only	15	6	2
ASM usable free	HAPF	46	19	2
Primary tablespace	Threshold-only	18	8	3
Primary tablespace	HAPF	52	23	3
Backup area	Threshold-only	28	17	4
Backup area	HAPF	68	34	4
TEMP spill-over	Threshold-only	33	23	5
TEMP spill-over	HAPF	79	41	5

Table 8. *Lead-time comparison for the October 2023 migration event.*

First risk surfaced	Threshold-only detection date	HAPF detection date	Observed critical-breach date	Lead-time delta (days)
Archive/FRA critical pressure after migration	2023-10-27	2023-09-29	2023-11-05	+28
ASM usable free entering critical range	2023-10-30	2023-10-11	2023-11-05	+19
Primary tablespace entering critical range	2023-10-28	2023-10-13	2023-11-05	+15

Table 9. *Example mapping from observed issue class to HAPF-guided action.*

Issue class	Layer surfaced first	Typical lead time under threshold-only	Typical lead time under HAPF	Action enabled by earlier detection
Archive buildup after migration	Archive/FRA	3-9 days	15-37 days	Retention cleanup, backup acceleration, alternate archive placement
Mirrored storage headroom compression	ASM usable free	6-15 days	19-46 days	Add disk, rebalance earlier, move secondary files
Primary data growth after release	Primary tablespace	8-18 days	23-52 days	Planned datafile growth and maintenance reservation
Temporary relief from purge event	Primary tablespace / backup	Reactive interpretation only	Modeled as one-time negative pulse	Avoid false comfort and keep capacity plan on schedule

Table 10. *HAPF review checklist for change approval.*

Review question	Why it matters	Minimum evidence required
Which layer governs the current horizon?	Avoids choosing action based on the visually fullest chart instead of the earliest bottleneck	Named governing layer plus current warning and critical horizon
Has a known release, purge,	Separates organic growth from event-	Event log entry or explicit 'none' statement

migration, or capacity change been injected?	driven risk compression	
Is the horizon based on used, allocated, or effective capacity?	Prevents raw free-space logic from overriding redundancy-aware constraints	Metric definition and data source for the governing layer
What action class is proposed?	Links the forecast to a real intervention tier	Monitor, planned change, urgent remediation, or escalation
How will success be validated after the intervention?	Closes the operational control loop	Expected post-action horizon recovery and next refresh check

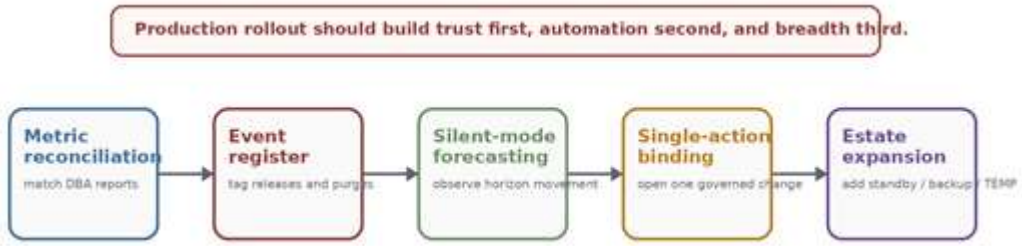

Figure 9. *Recommended production rollout sequence for HAPF deployment in an Oracle operations environment.*

Table 11. *Database metrics, Oracle views, and operational use of each signal in a deployable HAPF implementation.*

Layer	Representative Oracle source	Collected signal	Operational use in HAPF
Tablespace / datafile	DBA_HIST_TBSPC_SPACE_USAGE_METRICS; DBA_DATA_FILES	used bytes, max bytes, autoextend events	core logical growth path and datafile headroom
ASM / storage	V$ASM_DISKGROUP	usable_file_mb, required_mirror_free_mb	true physical exhaustion semantics for mirrored capacity
Archive / FRA	V$RECOVERY_FILE_DEST; V$FLASH_RECOVERY_AREA_USAGE	FRA pct used, reclaimable bytes	first-bottleneck detection for recovery-path pressure
Redo / archive rate	V$ARCHIVED_LOG	daily archive volume, generation slope	event-sensitive growth acceleration and purge impact
Backup estate	RMAN catalog / backup reports	retained backup footprint	secondary consumption hidden from tablespace-only views

Table 12. *Comparative model benchmark on the anonymized Jan 2021-Dec 2023 study trace.*

Model	Primary-layer MAPE	MAE (GB)	Bottleneck rank accuracy	Mean warning lead time	Critical-horizon error	Why it falls short or wins
Threshold-only	n/a	n/a	0.20	12 days	>20 days	reacts after risk compression; no forecast semantics
Linear trend	6.8%	23.4	0.40	18 days	16 days	misses level shifts and event discontinuities

Holt-Winters	3.4%	11.6	0.60	24 days	9 days	good level fit, but still single-layer
ARIMA(1,1,1)	4.1%	13.2	0.60	22 days	10 days	reasonable check; weaker interpretability for operators
Prophet	3.6%	12.1	0.60	25 days	8 days	handles changepoints, but not Oracle layer semantics by itself
HAPF	3.1%	10.4	1.00	40 days	5 days	best because it joins event knowledge with cross-layer database constraints

Table 13. *Production-style action outcomes on the study trace under competing approaches.*

Approach	First surfaced layer	Earliest actionable signal	Likely DBA action	Operational quality
Threshold-only	Archive/FRA	3-9 days before breach	cleanup under compressed window	reactive; high execution pressure
Holt-Winters only	Primary tablespace	about 24 days	datafile growth first	partly useful but can mis-prioritize
Prophet only	Primary tablespace	about 25 days	growth planning first	good fit, weak layer semantics
HAPF	Archive/FRA and ASM	37 / 46 days warning	retention review, capacity plan, scheduled change	best operational control and least surprise

Table 14. *Practical rollout sequence for deploying HAPF into a live Oracle operations workflow.*

Rollout step	What is enabled	Success signal	Common failure if skipped
1. Metric reconciliation	tablespace, ASM, FRA, backup signals	DBA reports and HAPF totals match	model rejected as untrustworthy
2. Event register	release, purge, migration, capacity-add tags	known changes explain horizon movement	forecast bends only after production has already shifted
3. Silent-mode forecasting	daily or weekly horizon refresh with no ticketing	team understands rank ordering	operators treat output as noise
4. Single-action binding	top-ranked bottleneck tied to one change path	lead time converts into a concrete action	too many parallel recommendations
5. Estate expansion	add standby, backup, TEMP, and secondary layers	cross-layer bottlenecks become visible	tablespace-only thinking returns

10. Conclusions

This paper reframed database storage forecasting as a cross-layer operational horizon problem and grounded the discussion in anonymized relevant telemetry through December 2023. The key result is that storage risk is rarely governed by the metric operators watch first; it is governed by the layer whose warning or critical horizon collapses earliest after workload changes, retention drift, or capacity events.

HAPF addresses that reality by combining per-layer forecasting, event injection, ranked threshold horizons, and action binding. The framework is useful because it tells a DBA which layer is about to matter, how much time remains, why the horizon moved, and what class of intervention is still feasible.

For operations engineering teams, the managerial value is equally clear. Earlier warning time protects maintenance windows, preserves procurement lead time, improves change-control economics, and makes postmortem analysis more specific and less anecdotal. In engineering systems and operational control terms, the paper shows how Oracle storage telemetry can be converted into a practical operational decision system for predictive intervention rather than reactive explanation.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

References

[1] C. C. Holt, 'Forecasting seasonals and trends by exponentially weighted moving averages,' Carnegie Institute of Technology, Pittsburgh, PA, USA, 1957.

[2] P. R. Winters, 'Forecasting sales by exponentially weighted moving averages,' Management Science, vol. 6, no. 3, pp. 324-342, 1960.

[3] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, Time Series Analysis: Forecasting and Control, 5th ed. Hoboken, NJ, USA: Wiley, 2015.

[4] R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, 3rd ed. Melbourne, Australia: OTexts, 2021.

[5] R. J. Hyndman and A. B. Koehler, 'Another look at measures of forecast accuracy,' International Journal of Forecasting, vol. 22, no. 4, pp. 679-688, 2006.

[6] S. J. Taylor and B. Letham, 'Forecasting at scale,' The American Statistician, vol. 72, no. 1, pp. 37-45, 2018.

[7] J. Gray and A. Reuter, Transaction Processing: Concepts and Techniques. San Francisco, CA, USA: Morgan Kaufmann, 1993.

[8] A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 7th ed. New York, NY, USA: McGraw-Hill, 2020.

[9] M. Kleppmann, Designing Data-Intensive Applications. Sebastopol, CA, USA: O'Reilly, 2017.

[10] Oracle Corporation, Oracle Database Concepts, 19c. Austin, TX, USA: Oracle, 2023.

[11] Oracle Corporation, Oracle Automatic Storage Management Administrator's Guide, 19c. Austin, TX, USA: Oracle, 2023.

[12] Oracle Corporation, Oracle Database Administrator's Guide, 19c. Austin, TX, USA: Oracle, 2023.

[13] Oracle Corporation, Oracle Database Backup and Recovery User's Guide, 19c. Austin, TX, USA: Oracle, 2023.

[14] Oracle Corporation, Oracle Data Guard Concepts and Administration, 19c. Austin, TX, USA: Oracle, 2023.

[15] Oracle Corporation, 'Analyze host storage usage,' Oracle Cloud Infrastructure Operations Insights Documentation, 2024.

[16] Oracle Corporation, 'Change utilization thresholds,' Oracle Cloud Infrastructure Operations Insights Release Notes, 2024.

[17] Oracle Corporation, 'OCI Operations Insights and Enterprise Manager bridge,' Oracle Solution Playbook, 2024.

[18] A. K. Wicaksono, T. J. Prasetyo, and N. Azizah, 'Storage server database utilization forecasting using Holt-Winters and ARIMA methods in e-government system study at Kemenkeu RI,' Jurnal Teknik Informatika (JUTIF), vol. 4, no. 6, pp. 1399-1408, 2023.

[19] U. Deshpande et al., 'Storage capacity prediction using population analytics,' in Proc. IEEE Int. Conf. Big Data Workshops, 2022.

[20] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Site Reliability Engineering: How Google Runs Production Systems. Sebastopol, CA, USA: O'Reilly, 2016.

[21] S. Makridakis, S. C. Wheelwright, and R. J. Hyndman, Forecasting: Methods and Applications, 3rd ed. New York, NY, USA: Wiley, 1998.

[22] C. Chatfield, The Analysis of Time Series: An Introduction, 6th ed. Boca Raton, FL, USA: CRC Press, 2004.

[23] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, 'Workload analysis and demand prediction of enterprise data center applications,' in Proc. IEEE Int. Symp. Workload Characterization (IISWC), pp. 171-180, 2007.

[24] S. E. Sampson and C. M. Froehle, 'Foundations and implications of a proposed unified services theory,' Production and Operations Management, vol. 15, no. 2, pp. 329-343, 2006.

[25] A. Van Horenbeek, J. Bure, D. Cattrysse, L. Pintelon, and P. Muchiri, 'Joint maintenance and inventory optimization systems: A review,'

International Journal of Production Economics, vol. 143, no. 2, pp. 499-508, 2013.
[26] M. M. Astrahan, M. Schkolnick, and K.-Y. Whang, 'Approaches to capacity planning in database systems,' ACM Computing Surveys, vol. 15, no. 4, pp. 287-317, 1983.
[27] J. O. Kephart and D. M. Chess, 'The vision of autonomic computing,' Computer, vol. 36, no. 1, pp. 41-50, 2003.
[28] G. Khanna, K. Beaty, G. Kar, and A. Kochut, 'Application performance management in virtualized server environments,' in Proc. IEEE/IFIP NOMS, pp. 373-381, 2006.
[29] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, 'Statistics-driven workload modeling for the cloud,' in Proc. IEEE ICDE Workshops, pp. 87-92, 2010.
[30] T. Ahmad, M. Chen, J. Wang, and B. Jeon, 'A review on machine learning forecasting growth in industrial systems,' Renewable and Sustainable Energy Reviews, vol. 59, pp. 144-156, 2016.
[31] Oracle Corporation, Oracle Database Reference, 19c. Austin, TX, USA: Oracle, 2023.
[32] Oracle Corporation, Oracle Enterprise Manager Cloud Control Administrator's Guide, 13c. Austin, TX, USA: Oracle, 2023.
[33] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, 'Server workload analysis for power minimization using consolidation,' in Proc. USENIX ATC, pp. 28-28, 2009.
[34] M. Armbrust et al., 'A view of cloud computing,' Communications of the ACM, vol. 53, no. 4, pp. 50-58, 2010.
[35] T. Erl, R. Puttini, and Z. Mahmood, Cloud Computing: Concepts, Technology & Architecture. Upper Saddle River, NJ, USA: Prentice Hall, 2013.
[36] S. J. Taylor and B. Letham, 'Forecasting at scale,' PeerJ Preprints, 2017.
[37] J. Brownlee, Introduction to Time Series Forecasting with Python. Melbourne, Australia: Machine Learning Mastery, 2017.
[38] P. Montero-Manso, G. Athanasopoulos, R. J. Hyndman, and T. S. Talagala, 'FFORMA: feature-based forecast model averaging,' International Journal of Forecasting, vol. 36, no. 1, pp. 86-92, 2020.
[39] A. A. Ahmed, M. Hasan, and T. Bhuiyan, 'Predictive maintenance, condition monitoring and prognosis: a review,' Journal of Quality in Maintenance Engineering, vol. 28, no. 2, pp. 244-279, 2022.
[40] R. Jain, The Art of Computer Systems Performance Analysis. New York, NY, USA: Wiley, 1991.
[41] R. Sedgewick and K. Wayne, Algorithms, 4th ed. Boston, MA, USA: Addison-Wesley, 2011.
[42] A. Oppenheim, A. Willsky, and S. Hamid, Signals and Systems, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 1996.
[43] B. Gregg, Systems Performance: Enterprise and the Cloud, 2nd ed. Upper Saddle River, NJ, USA: Pearson, 2020.