INTERNATIONAL MULTI DISCIPLINARY JOURNAL FOR RESEARCH & DEVELOPMENT
eISSN 2394-6334 Impact factor: 7,854 Volume 12, issue 10 (2025)

A Theoretical and Practical Synthesis for Minimizing Cache Fusion Waits in Oracle RAC via Access-Driven Block Allocation

John R. Whitaker
University of Edinburgh

Abstract: This article presents a comprehensive, publication-ready study synthesizing theoretical cache-management frameworks with practical techniques for optimizing Oracle Real Application Clusters (RAC) performance. Drawing exclusively from the provided references, it constructs an integrated argument that instance-specific block allocation (ISBA) combined with access-time–aware caching policies, hierarchical TTL-based cache design principles, and careful I/O subsystem tuning can materially reduce cache-fusion wait events, improve scalability, and stabilize application response times in multi-instance database clusters. The abstracted methodology describes a layered approach: first, conceptual mapping of RAC's Cache Fusion mechanism into canonical operating-system and caching models; second, formulation of allocation and cache-coherence heuristics inspired by knapsack optimization and utility-maximization frameworks; third, a stepwise deployment and monitoring plan that aligns with Oracle recommended design patterns. Results are presented as descriptive, theory-grounded findings drawn from cross-domain analogies and documented empirical reports in vendor literature and case studies. The discussion explores trade-offs, limitations, and extensions—contrasting centralized cache-control vs. instance-local optimizations, analyzing the cost of increased local allocation on interconnect traffic, and showing how hierarchical cache theory can frame multi-tier RAC deployments. The paper concludes by offering a set of practically applicable guidelines for DBAs and system architects, as well as a research agenda that bridges theoretical caching literature with future empirical RAC investigations. (Keywords: Oracle RAC, cache fusion, instance-specific allocation, hierarchical caching, disk I/O optimization)

Keywords: Oracle RAC, Cache Fusion, Instance-Specific Block Allocation, Hierarchical Caching, I/O Throughput, Performance Optimization

INTRODUCTION

The architecture of modern enterprise database systems increasingly relies on distributed cluster designs to meet demands for high availability, seamless scalability, and predictable performance. Among these, Oracle Real Application Clusters (RAC) has been a prominent commercial solution, enabling multiple database instances to access a single database storage while presenting a unified service to applications (Oracle Corporation, 2023). RAC's Cache Fusion mechanism, which allows instances to share data blocks without round-trips to disk by transferring blocks over the interconnect, is foundational to the cluster's ability to maintain coherent views of database state while delivering scale-out behavior (Oracle Corporation, 2023). Yet the very mechanism that enables coherence—on-demand transfer and coordination of block ownership—can become a primary source of contention and latency when workloads produce high inter-instance sharing, leading to cache-fusion wait events that degrade throughput and elevate response-time variability (Oracle Support, 2020).

This article posits that bridging the gap between theoretical caching models and concrete RAC deployment techniques offers a pathway to substantial performance improvements. The central problem addressed here is: how can instance-specific block allocation strategies, guided by formal caching and allocation theory, be designed and operationalized to reduce Cache Fusion waits and I/O contention while preserving coherence guarantees and application correctness? The problem is both practical—DBAs and architects must tune

operational systems—and theoretical—how to map RAC behavior to canonical caching paradigms such as hierarchical caching, utility optimization, and knapsack-style allocation.

The literature presents several relevant strands. Vendor documentation and white papers provide authoritative descriptions and recommendations for RAC design and deployment, including guidelines to reduce cache-fusion wait events and to use hardware features (Oracle Corporation, 2023; Oracle Support, 2020; Oracle SuperCluster, 2014). Practitioner-focused treatments summarize optimization strategies and real-world case studies that illustrate recurring performance patterns and remedial measures (Smith, 2021; I. M. Review, 2021). Foundational academic work in operating systems, caching theory, and I/O subsystem modeling supplies formal tools for reasoning about allocation, trade-offs, and resource optimization (Coffman & Denning, 1973; Kellerer et al., 2004; Che et al., 2002; Fofack et al., 2014; Dehghan et al., 2016; Garetto et al., 2016; Neglia et al., 2016). Empirical study of disk I/O and multi-disk throughput remains an essential complement to cache-focused analyses; models of disk behavior and bus-level contention continue to inform realistic performance expectations (Barve et al., 1999; Ng, 1998).

By synthesizing these domains, this paper identifies ISBA as a promising tactic: allocate blocks or buffer cache residency preferentially on the instance that most frequently accesses those blocks, thereby reducing remote block transfer events and the associated cache-fusion waits. The idea intersects with hierarchical cache system design—emphasizing access awareness and local optimizations—and with utility-maximization frameworks that formalize the value of cache space for different content classes (Che et al., 2002; Dehghan et al., 2016; Neglia et al., 2016). This study argues that ISBA should not be treated as a single heuristic but as a controlled, tunable policy embedded in a broader architecture of monitoring, I/O tuning, and coordinated workload placement consistent with Oracle deployment guidance (Oracle Corporation, 2023; Oracle Support, 2020; Smith, 2021).

The contribution of this paper is primarily synthetic and theoretical with operational guidance: (1) a detailed conceptual mapping of RAC Cache Fusion into canonical caching and operating-system models; (2) an elaboration of allocation heuristics inspired by knapsack and utility-optimization principles that can govern instance-local block residency decisions; (3) a descriptive methodology for deploying, measuring, and iterating ISBA in production RAC environments that minimizes risk while maximizing reductions in cache-fusion waits; and (4) a critical discussion of limitations, trade-offs, and directions for empirical validation. Where vendor documentation and case studies provide concrete recommendations and outcomes, these are woven into the theoretical narrative to produce practical takeaways for system architects and DBAs (Oracle Support, 2020; Smith, 2021; I. M. Review, 2021).

METHODOLOGY

This work follows a conceptual-analytical methodology rather than empirical experimentation based on new measurements. The methodological approach consists of four interlinked stages: conceptual mapping, theoretical formulation, operational design, and evaluative reasoning. Each stage synthesizes evidence from the provided references and infers logically consistent recommendations, ensuring every major claim is grounded in cited literature.

Conceptual mapping begins by situating Oracle RAC’s Cache Fusion within canonical operating-system models of shared-memory coherence and buffer cache management. Cache Fusion effectively forms a distributed buffer cache where block ownership and coherence semantics must be negotiated across instances—this is analogous to shared-memory coherence protocols discussed in operating systems theory and informs the interpretation of transfer costs and wait events (Coffman & Denning, 1973; Oracle

Corporation, 2023). The mapping identifies primitives: block residency (which instance holds a dirty or clean block), request patterns (read-only vs. read-write), ownership transfer cost (interconnect latency and processing), and fallback to physical I/O (disk access).

Theoretical formulation applies models from caching theory and optimization to reason about allocation. Hierarchical and TTL-based caching studies provide a template for multi-level cache reasoning: content may be preferentially placed in local ("instance") caches to minimize remote hits and transfers, while higher-level caches (shared storage or SSD tiers) handle durability and cold data (Che et al., 2002; Fofack et al., 2014). Utility optimization frameworks and access-time–aware algorithms inform a policy that quantifies the benefit of retaining a block locally versus releasing it, incorporating an estimate of access frequency, cost of remote transfer, and cost of eventual disk fetch (Dehghan et al., 2016; Neglia et al., 2016; Garetto et al., 2016). Knapsack problem analogies offer a combinatorial lens for constrained cache-space allocation—selecting which blocks to hold locally within limited buffer cache capacity to maximize cumulative access utility (Kellerer et al., 2004). Classical I/O throughput models constrain achievable improvements by bounding the physical I/O baseline and demonstrating scenarios where avoiding remote transfers nevertheless increases disk contention or creates imbalanced I/O flows (Barve et al., 1999; Ng, 1998).

Operational design draws directly from Oracle's deployment guidance and practitioner resources to ensure feasibility. Oracle's own documentation provides design techniques and administrative controls for RAC that are compatible with instance-level optimizations and prescribe monitoring metrics relevant to cache-fusion wait events (Oracle Corporation, 2023; Oracle Support, 2020). Practitioner texts and case studies provide context for real-world workload characteristics, latency thresholds, and DBMS configuration knobs that influence cache residency decisions, including buffer cache sizing, instance parameterization, and application affinity strategies (Smith, 2021; I. M. Review, 2021; Oracle SuperCluster, 2014). Synthesis yields a sequence of deployment steps: instrumentation and baseline measurement; incremental ISBA policy rollout under canary workloads; iterative tuning informed by response-time distributions and wait-event decomposition; and fallback safety mechanisms to preserve correctness and durability.

Evaluative reasoning eschews fresh experiments but instead constructs plausible result narratives grounded in vendor reports and theoretical expectations. For each claimed benefit of ISBA (reduction in cache-fusion waits, improved tail latency, improved throughput), the discussion ties the effect to specific mechanisms—local residency reduces cross-instance transfers (Oracle Support, 2020), access-time–aware caching reduces miss rates in local caches (Neglia et al., 2016), and knapsack-like allocation ensures that limited local cache capacity is used for the most valuable blocks (Kellerer et al., 2004). Contraints from I/O subsystem analysis temper claims by observing cases where localized caching leads to concentrated I/O demand or when workloads exhibit high write-sharing that necessarily requires frequent ownership transfers (Barve et al., 1999; Ng, 1998).

Throughout the methodological exposition, the paper provides detailed textual explanations of algorithmic choices and administrative operations rather than pseudocode or formulas, respecting the instruction not to include mathematical notation while preserving strong logical clarity.

RESULTS

Because this article is an integrative theoretical and operational synthesis rather than a new experimental report, the "results" are descriptive and inferential: they are distilled outcomes that emerge from applying the methodological combination of theoretical frameworks to RAC operational realities, supported by vendor documentation and case-study reports referenced in the literature.

Reduction in Cache-Fusion Wait Events through ISBA: When blocks are preferentially allocated or retained on the instance that most frequently accesses them, remote block requests diminish because many subsequent accesses hit the local buffer cache instead of requiring ownership transfer or block transfer across the interconnect. The Oracle Support white paper explicitly identifies cache fusion wait events as a dominant cause of latency in shared workload patterns and recommends measures to localize access patterns and reduce remote transfers (Oracle Support, 2020). The conceptual mapping to hierarchical caching underpins the mechanism: local cache residency acts as the first-level cache in a multi-tier system, reducing expensive cross-instance accesses (Che et al., 2002; Fofack et al., 2014). Thus, ISBA produces an expected reduction in average and tail cache-fusion waits, particularly for workloads with identifiable instance affinity.

Improved Predictability and Tail Latency: The literature on cache behavior and utility optimization indicates that reducing variance in cache access paths yields improved predictability in application response times (Dehghan et al., 2016; Garetto et al., 2016). In RAC, unpredictable ownership transfers and the resulting serialization of certain operations can create heavy tails in response-time distributions. ISBA's localized residency reduces the probability of a given request incurring the transfer-and-grant handshake and thus truncates the tail of the latency distribution. Practitioner case reports emphasize the importance of predictability for SLA compliance and indicate that measures which reduce inter-instance sharing often correlate with improvements in worst-case latency (Smith, 2021; I. M. Review, 2021).

Workload Characterization Matters: The degree to which ISBA yields benefits depends strongly on workload sharing characteristics. Read-mostly workloads with stable instance affinity are the most favorable: frequent reads of shared blocks can be satisfied locally if those blocks are allocated or pinned on the serving instance. If workloads involve frequent write-sharing—many instances updating the same blocks—ISBA cannot avoid transfers that enforce serializability; in such cases, the cost of trying to localize blocks may simply shift contention without reducing overall transfer counts (Oracle Support, 2020; Barve et al., 1999). This is a nuanced result: ISBA helps when access locality exists, but ill-applied localization can be neutral or harmful under strong write-sharing workloads.

Trade-offs with I/O and Buffer Cache Utilization: The knapsack and utility optimization analogies suggest that local buffer caches must be used for the most valuable blocks under capacity constraints (Kellerer et al., 2004; Dehghan et al., 2016). ISBA that indiscriminately pins blocks locally risks crowding out other valuable local content and might cause increased disk I/O due to evictions. The theoretical synthesis thus recommends an allocation policy that ranks blocks by expected access utility (combining access frequency, cost of remote fetch, and cost of disk fetch) and allocates local cache space accordingly. Oracle guidance to size buffer caches appropriately and monitor evictions aligns with this conclusion; likewise, analyses of disk I/O show that optimizing transfers without attention to physical throughput can produce secondary bottlenecks (Barve et al., 1999; Ng, 1998).

Alignment with Hierarchical Cache Principles: The TTL-based and hierarchical caching literature shows that multi-level cache systems benefit from coordinated policies that account for content freshness and cost of retrieval from lower tiers (Che et al., 2002; Fofack et al., 2014). ISBA can be interpreted as introducing an instance-level first tier that—when paired with appropriate eviction and TTL-like staleness controls—reduces expensive cross-instance transfers while ensuring consistent revalidation from shared storage when necessary. This conceptual alignment strengthens ISBA's theoretical justification and suggests practical mechanisms such as adaptive TTLs, probabilistic admission, or class-based retention policies that echo successful hierarchical cache strategies (Neglia et al., 2016; Garetto et al., 2016).

Case Study Corroboration and Vendor Validation: Practitioner reports and vendor documentation present corroborative evidence that strategies centered on localizing access and tuning buffer caches can reduce cache-fusion waits and improve scalability (Oracle Support, 2020; Smith, 2021; I. M. Review, 2021). Vendor white papers describing Oracle SuperCluster and related hardware deployments emphasize that careful alignment between workload placement, buffer caching, and faster network interconnects produces substantial throughput gains for core banking and other demanding domains (Oracle SuperCluster, 2014). While these reports do not typically present the formal ISBA construct as articulated here, their recommended practices—affinity-aware workload placement, buffer cache sizing, and interconnect optimization—are consistent with ISBA's operational prerequisites.

Synthesis: Expected Aggregate Effects: Taken together, the theoretical framework and documented practices imply several aggregate results when ISBA is applied judiciously: reduction in cache-fusion wait events for locality-prone workloads, improved tail latency, potentially higher transaction throughput due to fewer cross-instance serializations, and a requirement for careful buffer cache and I/O tuning to avoid shifting bottlenecks to physical disks. Oracle documentation and practitioner reports provide an empirical plausibility cushion for these expectations, albeit not a statistical proof; deeper field experiments remain necessary to quantify effect sizes under diverse real-world workloads (Oracle Corporation, 2023; Smith, 2021; I. M. Review, 2021).

DISCUSSION

The above results, while theoretically grounded and operationally plausible, raise numerous interpretive questions, limitations, and directions for future work. This section unpacks the implications, enumerates counter-arguments, and sketches a research agenda to empirically test and refine ISBA.

Interpretation of Theory-Practice Synthesis: The conceptual mapping of RAC's Cache Fusion to hierarchical caching and operating-system coherence models is illuminating because it reframes inter-instance contention in language familiar to caching researchers—access locality, miss penalties, and cache allocation trade-offs (Coffman & Denning, 1973; Che et al., 2002). This reframing suggests that canonical optimization techniques—such as access-time–aware caching and utility-maximization—can be ported into DBMS cluster tuning with careful adaptation (Dehghan et al., 2016; Neglia et al., 2016). The knapsack analogy provides a rigorous metaphor for constrained cache-space decisions, implying that the highest-utility blocks should receive residency priority when local cache is scarce (Kellerer et al., 2004).

Counter-arguments and Limits: Several important counterpoints must be considered. First, the DBMS semantics impose stricter correctness constraints than many caching systems; ownership transfers and serialization are necessary to preserve ACID properties. Any allocation strategy must preserve correctness and cannot simply ignore ownership or locking costs. This is supported by Oracle's emphasis on maintaining integrity and correctness in RAC deployments (Oracle Corporation, 2023). Second, the presence of dynamic workload shifts and ephemeral access patterns complicates static allocation or naive pinning approaches: blocks that were previously hot on one instance may become hot elsewhere, making rigid localization counterproductive. Third, hardware topology—non-uniform memory access, varying interconnect latencies, and heterogeneous storage back-ends—introduces practical considerations. Oracle SuperCluster and other vendor reference architectures highlight that hardware-induced asymmetries can materially affect the success of any cache-localization strategy and thus must be integrated into operational decision-making (Oracle SuperCluster, 2014). Finally, the knapsack-style optimization is computationally intensive if implemented naively; practical heuristics will be necessary to approximate theoretically optimal allocations in real time (Kellerer et al., 2004).

Operational Trade-offs: Localizing blocks reduces remote transfer counts but can increase local resource contention—particularly buffer cache pressure and disk I/O when evictions occur. Thus, ISBA must be paired with dynamic eviction policies that account for both local value and system-wide effects. Utility-optimization frameworks that consider the global benefit of retaining a block locally versus making it available for other instances may provide theoretical guidance, but real-time enforcement requires scalable instrumentation and lightweight analytics (Dehghan et al., 2016; Garetto et al., 2016). Monitoring is critical: Oracle-provided performance metrics and practitioners' recommendations emphasize continuous measurement of wait events, buffer cache hit ratios, and physical I/O patterns so that allocation policies can be adapted responsively (Oracle Corporation, 2023; Smith, 2021).

Methodological and Empirical Limitations: This article does not present new empirical measurements; instead, it integrates theoretical frameworks with existing vendor and practitioner literature. Consequently, effect size predictions are qualitative and plausibility-based rather than statistically quantified. To validate the claims, controlled experiments and production-scale A/B tests are required. Recommended experimental designs include synthetic benchmarks that vary sharing degrees and write intensities, and production trials with phased rollouts and instrumentation that measures cache-fusion wait times, transfer counts, and I/O metrics. The theoretical literature suggests specific measurable targets—changes in miss rates, reductions in interconnect transfer volume, and shifts in response-time percentiles—that should be captured in empirical work (Neglia et al., 2016; Dehghan et al., 2016; Barve et al., 1999).

Practical Deployment Guidelines: From the synthesis, several practical guidelines emerge for practitioners considering ISBA:

1. Instrumentation-first Deployment: Establish a comprehensive baseline by measuring cache-fusion wait events, buffer cache hit/miss ratios, and disk I/O profiles using Oracle's diagnostic tools and recommended metrics (Oracle Corporation, 2023; Oracle Support, 2020). Without a baseline, it is impossible to discern the marginal impact of any policy change.

2. Affinity-aware Workload Placement: When possible, co-locate application threads or middleware components to increase instance affinity—reducing the cross-instance sharing that gives rise to cache-fusion events (Smith, 2021; I. M. Review, 2021).

3. Adaptive Allocation Heuristics: Implement allocation policies that estimate block utility dynamically, weighing access frequency and remote-transfer cost. Use probabilistic or TTL-like admission policies to avoid over-committing local cache to transient items (Neglia et al., 2016; Fofack et al., 2014).

4. Conservative Pinning with Fallbacks: If block pinning is used, apply it conservatively and include automatic release or demotion heuristics. Ensure correctness mechanisms remain intact and provide admin overrides.

5. I/O Subsystem Co-design: Ensure that any local-residency strategy is coordinated with disk layout, RAID configurations, or SSD tiers to avoid overloading physical devices. Disk throughput models suggest balancing localized cache benefits against potential increases in disk demand (Barve et al., 1999; Ng, 1998).

6. Progressive Rollout and Measurement: Phase changes under controlled loads and measure their impacts on both mean and tail latency, as well as on system-wide transfer counts and disk I/O patterns. Use canary instances and rollbacks if adverse effects appear (Oracle Support, 2020; Smith, 2021).

Future Research Agenda: Several avenues for further research could strengthen the theoretical and empirical foundations of ISBA in RAC:

● Formal Modeling: Develop formal models that quantify the trade-off between local residency and global coherence cost under varying sharing patterns, network latencies, and bandwidth constraints. Borrow techniques from utility optimization and stochastic modeling to produce prescriptive policies (Dehghan et al., 2016; Neglia et al., 2016).

● Heuristic Evaluation: Evaluate practical heuristics—such as LRU variants enhanced with access-time weighting, or admission policies informed by predicted next-access times—in controlled experiments to determine which approximations deliver most benefit with minimal overhead (Garetto et al., 2016).

● Workload Taxonomy: Construct a taxonomy of workloads based on read/write ratios, sharing intensity, and temporal locality to identify classes where ISBA is most effective versus classes where it is ineffective or detrimental.

● Hardware-Aware Policies: Investigate ISBA variants that account explicitly for hardware topology (e.g., NUMA, heterogeneous interconnects) to avoid transferring hot ownership to instances with poor network proximity.

● Integration with Oracle Controls: Explore how ISBA-like policies could be integrated into DBMS parameterization or Oracle's own cache-coherence mechanisms in a supported and safe manner, perhaps through advisory guidance or management APIs.

CONCLUSION

This paper advances an integrated theoretical and operational perspective on reducing cache-fusion wait events and improving performance in Oracle Real Application Clusters by leveraging instance-specific block allocation and related cache-optimization techniques. By mapping RAC's Cache Fusion to canonical caching and operating-system models, applying principles from hierarchical caching and utility optimization, and grounding recommendations in vendor documentation and practitioner reports, it outlines a pragmatic pathway for DBAs and architects to explore ISBA safely and effectively. The benefits—reduced inter-instance transfers, improved predictability, and enhanced throughput—are most likely for workloads with strong instance affinity and limited write-sharing; conversely, write-intensive, widely shared workloads will see limited gains and require more nuanced measures.

The synthesis emphasized careful instrumentation, progressive rollout, co-design with the I/O subsystem, and conservative pinning heuristics to mitigate risks. It also highlighted the need for empirical validation through targeted experiments and proposes a research agenda encompassing formal modeling, heuristic evaluation, workload taxonomy development, and hardware-aware adaptations. Oracle's own documentation and white papers provide operational context and corroboration for the high-level claims (Oracle Corporation, 2023; Oracle Support, 2020; Oracle SuperCluster, 2014), while academic works on caching and I/O modeling supply the theoretical foundation necessary for principled policy design (Coffman & Denning, 1973; Kellerer et al., 2004; Che et al., 2002; Dehghan et al., 2016; Neglia et al., 2016; Garetto et al., 2016; Barve et al., 1999; Ng, 1998).

In sum, ISBA is not a panacea but a conceptually grounded tool in the optimizer's toolkit: when applied thoughtfully and measured carefully, it offers a promising route to mitigate one of RAC's most insidious performance drains—cache-fusion waits—while opening pathways for future research that more tightly

couples theory with operational practice.

REFERENCES

1. Oracle Corporation. Oracle Real Application Clusters Documentation. 2023. https://docs.oracle.com/cd/E11882_01/rac.
2. Oracle Support. Reducing Cache Fusion Wait Events in Oracle RAC. Oracle White Papers. 2020.
3. Smith, J. High-Performance Oracle RAC: Strategies for Optimization. Pearson. 2021.
4. M. Review. Case Studies in Oracle RAC Performance Optimization. 2021. https://www.itmanagementreview.com
5. Coffman, E. G., Jr., & Denning, P. J. Operating Systems Theory. Prentice Hall Professional Technical Reference. 1973.
6. Kellerer, H., Pferschy, U., & Pisinger, D. Knapsack Problems. Springer. 2004.
7. Che, H., Tung, Y., & Wang, Z. Hierarchical Web caching systems: modeling, design and experimental results. Selected Areas in Communications, IEEE Journal on, vol. 20, no. 7, pp. 1305–1314. Sep 2002.
8. Fofack, N. C., Nain, P., Neglia, G., & Towsley, D. Performance evaluation of hierarchical TTL-based cache networks. Computer Networks, vol. 65, pp. 212–231. 2014.
9. Dehghan, M., Massoulie, L., Towsley, D., Menasche, D., & Tay, Y. A Utility Optimization Approach to Network Cache Design. Proc. of IEEE INFOCOM. 2016.
10. Garetto, M., Leonardi, E., & Martina, V. A Unified Approach to the Performance Analysis of Caching Systems. ACM Trans. Model. Perform. Eval. Comput. Syst., vol. 1, no. 3, pp. 12:1–12:28. May 2016. http://doi.acm.org/10.1145/2896380
11. Neglia, G., Carra, D., Feng, M., Janardhan, V., Michiardi, P., & Tsigkari, D. Access-time aware cache algorithms. Inria, Research Report RR-8886. Mar. 2016. https://hal.inria.fr/hal01292834
12. Barve, R., Shriver, E., Gibbons, P. B., Hillyer, B. K., Matias, Y., & Vitter, J. S. Modeling and Optimizing I/O Throughput of Multiple Disks on a Bus. Proceedings of the 1999 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. SIGMETRICS '99. 1999, pp. 83–92.
13. Ng, S. W. Advances in Disk Technology: Performance Issues. IEEE Computer, vol. 31, pp. 75–81. 1998.
14. Natti, M. Reducing Oracle RAC Wait Events by Using Instance-Specific Block Allocation for Production Applications. The Eastasouth Journal of Information System and Computer Science, 1(01), 65–68. 2023. https://doi.org/10.58812/esiscs.v1i01.447
15. Oracle Corporation. Oracle Real Application Clusters Administration and Deployment Guide, 21c: Design and Deployment Techniques. https://docs.oracle.com/en/database/oracle/oracle-database/19/racad/design-anddeployment-techniques.html

16. Oracle SuperCluster. Record Performance and Seamless Scalability in Core Banking with Infosys Finacle and Oracle SuperCluster. Oracle White Paper. November 2014. https://www.oracle.com/docs/tech/infosys-finacle-wr-osc.pdf.