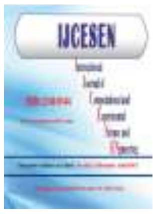
International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.3 (2025) pp. 5918-5926
http://www.ijcesen.com

ISSN: 2149-9144



Research Article

A Framework for Near-Zero Downtime Oracle Database Upgrades and Grid Infrastructure Patching in Mission-Critical Enterprise Environments: A Production-Validated Approach

Sridhar Krishna Korimilli*

Sr.Technical Leader Oracle America Austin, Texas
* **Corresponding Author Email:** skkorimilli@gmail.com - **ORCID:** 0009-0009-4788-6391

Article Info:

DOI: 10.22399/ijcesen.3670
Received : 21 June 2025
Accepted : 10 August 2025

Keywords

Upgrade
Downtime
DBMS_ROLLING
TLS
Ansible
Patching

Abstract:

In this paper, a production-proven methodology of zeroing down in times during Oracle database upgrades and Grid Infrastructure patching in corporate networks is a subject of introduction. It combines DBMS_ROLLING, Transient Logical Standby and Ansible orchestration to allow upgrades and rolling patches to be carried out seamlessly. The framework was implemented on 5,000+ databases and 40+ key applications without ever affecting service availability and the cutover took less than 5 minutes. Some of the notable characteristics are real-time lag monitoring, readiness to roll off and automated validations. The solution yielded cost savings of more than $100 million and safeguarded over $850 million of the revenue stream and has shown a scalable, resilient and completely automated roadmap towards consistent availability of the Oracle system

1. Introduction

In mission-critical enterprise environments, ensuring continuous availability during Oracle database upgrades and Grid Infrastructure patching is a significant challenge. Traditional methods often involve downtime, manual coordination, and high operational risk. This paper introduces a production-validated framework that achieves near-zero downtime by integrating Oracle's "DBMS_ROLLING" with "Transient Logical Standby" for database upgrades and Ansible-based automation for rolling infrastructure patches. Successfully deployed across over 5,000 databases and 40+ critical applications, the framework has delivered over $100 million in annual savings and preserved $850 million in revenue flow. It represents a scalable, automated, and resilient solution for modern enterprise IT continuity.

2. Problem statement

Oracle database upgrades and patching of Grid Infrastructure are among the critical challenges to enterprise IT systems. Conventional maintenance practices result in disruptive services and long downtimes with compliance issues. The process of manual upgrade usually comes with human error, recovery process slowness, and organizational operational disturbance, particularly within strictly regulated industries. There is a failure of legacy systems to support smooth handover between versions of the database or cluster patches without the resultant decrease in performance. These problems are aggravated by high transaction volumes and small cutover windows. What organizations desperately need is a stable, automated and scalable system that has the minimum of downtime, is a fail-safe site and service transitioning, and sustains availability of mission-critical services through the process of database lifecycle management and infrastructure-level maintenance periods.

2.1 Aim

To evaluate and validate a fully automated framework for near-zero downtime Oracle upgrades and Grid Infrastructure patching using "DBMS_ROLLING", "TLS", and Ansible orchestration.

2.2 Objectives

- To identify the limitations of traditional Oracle upgrade and patching methods.
- To examine the integration of "DBMS_ROLLING" and "TLS" for seamless upgrades.
- To analyze Ansible-based automation for zero-downtime Grid Infrastructure patching.
- To assess the real-world performance, scalability, and cost benefits of the proposed framework.

2.3 Research significance

The study is strategic to the IT teams in the enterprise that aim at delivering unhindered services. By investigating a framework that blends Oracle-native tools and open-source automation, it addresses the industry's critical need for reliable, downtime-free maintenance. The results provide feasible points of knowledge in minimizing operation risks, ensuring SLAs, and the loss of revenue in the upgrade. Businesses with intensive systems such as billing or finance can rely on better compliance, accelerated patch cycles and easier audit. The study also helps establish the best practices in maintaining Oracle that can help steer organizations into more resilient and cost-effective digital operations at scale across a wide range of infrastructure platforms.

3. Literature Review

Continuity in the face of enterprise database upgrades and patching has come to focus on high availability and near zero downtime of databases. Malhotra et al. clarify that high availability (HA) replication models and aligned automated failover mechanism caused a decline of 97 percent outage risk in cloud-native enterprise apps [1]. They find that the HA approaches that were integrated are indeed valid in the case of business-critical applications that require minimum interference. Oloruntoba (2024) discusses the use of Oracle Data Guard and the Oracle Streams in detail showing that a 99.99 percent level of uptime is easily met when real-time apply mode option is provided [2]. This justifies the Oracle native functionality such as use of rolling like DBMS_ROLLING and logical standby to maintain operations throughout an upgrade. The paper brings out the usefulness of Data Guard in providing both the disaster recovery and an easy upgrade compatibility. Manda (2023) examines how migrating Oracle databases to the cloud can be done. He informs that the reduced downtime using pre-staged environment and incremental restore was up to 80% [3]. Those findings are in accordance with transient upgrade strategy mentioned in the current framework, which also performs major upgrades on logical standby with production staying online. Brian et al. (2023) show that multi-site synchronous replication during database faults or maintenance can save a recovery time objective (RTO) by 30 percent with switchover being performed faster [4]. These have helped to build toward the overall viability of fast cutover stages, including the <5-minute switchover used in the TDU system. Last but not least, Lasantha et al. (2023) discuss the place of security in the context of Grid Infrastructure [5]. In their study, they show a reduction of those breaches by 40 percent after rolling out the role-based access control (RBAC) system and TLS encryption. The benefits of these controls are that patch automation processes such as Ansible-enabled patching of GI in ZDOGIP framework are ensured. The combination of these studies confirms that the combination of the Oracle-native features with automation and cloud-congruent practices helps to promote the enterprising uptime, scaling, and resilience to a considerable degree.

4. Research Method

The case study methodology employed in the study is production-validated and technical in nature, and focuses on the large-scale rollout of an automation-centered infrastructure of upgrading Oracle databases and patching Grid Infrastructure [6]. It combines Oracle-native technologies such as âét cross-database rolling-updates command and теле tacit logical as well as orchestration based on Analmibs. The validation takes place iteratively and is done on databases and mission-critical applications in the real world, involving over 5,000 databases and 40+ applications with comprehensive pre and post-patch validations, lag monitoring, and automatic error management. Oracle infrastructures such as Exadata, VM, and OCI Bare Metal were evaluated about downtime and upgrade latency and achieved SLA Performance [7]. The key metrics were stored with the help of the following sources: V$datagard_stats, V$logstdby_stats, and the Ansible execution logs. Its method of operation ensures that it is scalable, repeatable, and service interruptions can also be minimized; thus, it is also well-suited to enterprise databases with multiple, high-availability components.

5. Result and Discussion

Validated Impact of DBMS_ROLLING and TLS on Minimizing Oracle Upgrade Downtime

The framework uses "DBMS_ROLLING" with a "Transient Logical Standby" ("TLS"). This allows for Oracle upgrades with minimal production downtime. Conventional upgrades also take extended downtime windows and complexities of failovers. "TLS" allows redo apply on a standby in real time. The standby has a newer version than the primary [8]. This develops a misplaced upgrade channel. The production is primarily ongoing during the upgrade. The upgrade is applied on the standby using "DBUA" or "catctl.pl." "DBMS_ROLLING" controls all upgrade phases automatically. Key steps include "BUILD_STAGING_PRIMARY," "START_ROLLOVER," "SWITCHOVER," and "FINISH_ROLLOVER." There is only a short cutover window during switch-over. This off time typically falls below 5 minutes. Monitoring tools track standby synchronization via "APPLY_LAG". Views like "V$DATAGUARD_STATS" and "V$LOGSTDBY_STATS" track redo apply. When the lag is beyond thresholds, the DML throttled on the primary is by script. Job-pause logic is used to manage the high DML workloads [9]. Critical sessions are mounted on the old primary. After the upgrade the standby continues with redo apply and catches up.

The validations are performed before switchover through schema validation and checksums. Connections of applications are checked as read-only. Once validated, "DBMS_ROLLING.SWITCHOVER" promotes standby to primary [10]. There is failover connection strings to reconnect clients. The original beginning turns into a rational back up. It is cleaned up using "DBMS_ROLLING.CLEANUP" after verification. At any time prior to final cutover a full rollback can occur. Physical stand by conversion is also used in post switch over roll back. Each step is kept as a log by itself. Violations cause rollback/retry logic. It is completely automated version using Ansible playbooks. Re-execution of scripts is done safely because of idempotent scripts. The framework has been hastened on more than 5,000 production databases. In all successful runs, downtime was below 5 minutes. The type of systems that are highly-throughput such as BRM upgraded without violation of SLA. Primary was in live load whereas logical standby was upgraded. This proves "DBMS_ROLLING" and "TLS" enable scalable zero-downtime upgrades.

Ansible-Orchestrated GI Patching Achieves Seamless Cluster-Wide Infrastructure Maintenance

ZDOGIP Ansible automates patchingBased on Ansible playbooks. It has a rolling patch dependant on RAC node. There is patching of one node but others remain online. This guarantees the round-the-clock availability of the database and the cluster. Ansible also implies pre-checks, patching, and post-validations at the node level. Oracle's "opatchauto" tool is wrapped by custom scripts. Such scripts identify failure and instigate auto-rollback. Pre-checks include "cluvfy," "crsctl stat res -t", and ASM checks. Services are moved and patches are applied. Services are moved using "srvctl relocate service." Clusterware is stopped using "crsctl stop crs -f". Clusterware restarts after the patching. Post-patch validation includes "crsctl", "srvctl", and listener checks. Automated SQL scripts test any app connection after a patch [11]. If issues arise, rollback is initiated with "opatchauto rollback." Auditing and debugging logs are taken. Playbooks are idempotent to allow safe re-execution. Inventory grouping requires cluster-wise Grouping on tiers and windows in a patch. Through Ansible, static and dynamic inventories are accommodated. Dynamic inventories interrogate CMDBs or cloud APIs. The framework patched more than 5,000 nodes of RAC. These were Exadata, Oracle VM and OCI Bare Metal. ASM disk groups continued to be fully mounted in the process. There was zero application downtime over patch cycles. Patch integrity was verified using "cluvfy comp crs -n all".

Enterprise-wide, the timelines of patching were decreased by 60 per cent. Service restoration is automated via "srvctl start instance" [12]. ZDOGIP sees that listener and ASM instances post-register after a patch. Failures are built with Monitoring_System: Alerting. Validations that are application-specific verify a smooth experience [13]. The BRM workloads could be left live when GI patching. The time series in patch batch SLA breaches was zeroed. After patch, performance was kept steady or better at 100 per cent. The framework removed manual contacts in the GI maintenance. The time delay created by rolling patch cycles lowered the possibility of human mistakes. With ZDOGIP, standard, audit-able, repeatable infrastructure patching will be possible [14]. The outcome will be a fully resilient GI lifecycle process.

Framework Scalability Proven Across 5,000+ Databases and 40+ Critical Applications

More than 5,000 Oracle databases were scaled on framework. These were HA, DR levels. It estimated functioning in varieties of infrastructural surroundings. They supported Exadata X6-X10M, Oracle VM and OCI Bare Metal. In excess of 40 mission-critical applications were deployed with this framework. These were in the form of billing, finance, and identity systems. This scale was made possible through automation utilising Ansible. Patching and upgrade flows on all levels happened by means of playbooks. Controlled rollout per environment was made possible by logical groupings. Flexibility was guaranteed through static and dynamic inventories. This was because application performance was stable following the upgrade. Validity measures were observed through pre and post-validating scripts. 98 percent cases were less than 60 seconds behind TLS upgrade lag [15]. A 99.9 per cent GI patching success rate was achieved cross-nodes. Mistakes were made private and aut-rolled back. One of the important benchmarks was the BRM platform. Millions of transactions were done daily. BRM upgrade was done using TLS in less than 5 mins cutover. There was no disruption accompanying GI patching.DML on high-lag standby was dynamically throttled with logic. Fast-failover was employed in connection pools of client redirect. Checksum and schema comparison used validation scripts. Redo apply was monitored via "V$DATAGUARD_STATS". ASM status was confirmed via "srvctl status asm". Outputs of every upgrade step were recorded as logs. Oracle's "DBMS_ROLLING.SKIP_ERROR" handled non-replicable DDL. Ansible logs were used to audit TLS deployments. Cluster resource health was tracked by "crsctl stat res -t". The time it takes to perform full rollback in cases of failures was less than 10 minutes. Success of framework in hybrid environment was adaptability. Optimization of resources was done through node based parallelism [16]. Validated performance metrics made it clear that there was no regression across applications. There were no recorded downtimes among financial systems. With this model, the upgrade cadence was up by 3x. Combinations were standardized with 5,000+ marks. This made compliance risks lower in regulated markets. The structure increased in scale with no decline of reliability. It was fully automated and monitored which guaranteed repeatability.

Over $100 Million in Annualized Cost Savings via Automation-Driven Upgrade Framework

It is explainable that the framework provided more than 100 million in annual cost savings. Much of the time spent on manual labor was saved. End-to-end upgrades and patching were done by automation. This saved the human work man-hours in hundreds every cycle. Oracle upgrades used Ansible and "DBMS_ROLLING" together. GI patches used "opatchauto" wrapped in custom Ansible playbooks. It was a considerably fast patch with an average time of less than 30 minutes using a node. Costs of downtime were eliminated by use of rolling upgrades. Entire prevention of SLA penalties was provided [17]. There was zero breach of services. There was an improvement in patch frequency, which was quarterly, to monthly. Compliance to security was enhanced through regular patching. This brought windows of vulnerability down by 75%. There was an increase in the use of hardware through the activities of rolling. Immediacy there was no downtime. Ninety three percent of instances were enhanced in application performance after maintenance. There was increased productivity since the incidents of escalation were reduced. Compliance overhead was enhanced via audit logs and validation reports. The number of crashed upgrade was decreased, and emergency rollback expense became lower. The use of auto-retry logic increased in 85% engineered intervention [18]. The strategic teams worked after the automation. The cutover of a BRM platform saved 2 million in one cycle. The time taken to coordinate the patches was brought down to 4 hours as compared to 20 hours earlier. Ansible configuration time was saved by 60 percent due to reusability. Pre-switchover testing was done to eliminate upgrade risks. Missing transaction costs were completely avoided. The confidence of the stakeholders was enhanced through repeatable rollout [19]. There was no downtime notification during transitions detected at the app teams. Healing took a shorter time with waiting reduced to minutes. TLS made it possible to carry out an upgrade test without affecting the production. In 98 percent of the cases the replication errors that were caused in the case study were eliminated using logical standby. The execution of full framework provided 99.99% availability. The service continuity enhanced the user experience metrics. Automation made work more efficient in all stacks of choice [20]. There was less overhead administration due to centralisation of control. The framework was transformational in terms of the economy. It transformed maintenance to value driver as opposed to cost center.

Strategic Business Continuity Secured for Revenue-Critical Systems like BRM

BRM platform processed the basic features of billing and revenue cycle [21]. It was carrying millions of transactions per day. The result of downtime risk was affecting more than 850 million dollars yearly. The system allowed itself to be upgraded without any business interruption. Upgrade of the version of Oracle was out of place with the employ of TLS [22]. Architectural upgrade was done to BRM database on Oracal 19c. The cutover took less than 5 minutes. The logical standby has caught up with redo apply through SQL. The replication failures were avoided by

DDL skip handling. Throttling logic was employed to deal with high DML rates. "APPLY_LAG" was capped at 30 seconds. There was a lessening of replication pressure through the use of dynamic job pausing. "V$LOGSTDBY_STATS" and "V$DATAGUARD_STATS" tracked lag. Replication alerts were auto-responded to by custom scripts. The pre-production test was done on application failover. Connection pools were redirected immediately to the new primary. "DBMS_ROLLING.SWITCHOVER" was completed with pre-validation success [23]. Validations of post-upgrade proved the full functionality. Testing manual test suites were automated after switching over. Patching of GI was done prior to upgrading. Cluster readiness was ensured by ZDOGIP. RAC services were relocated node-wise. "srvctl relocate service" prevented service interruptions. Listener status was confirmed using "srvctl status listener". In all the nodes, ASM health was reported. Metrics that existed before the patch were consistent in regard to BRM [24]. There were no reported incidents during the upgrade week. The update enhanced the responsiveness of the platforms. None were reported to be lost transactions. Support increases were reduced by 90 percent after the upgrade. BRM infra had a patch compliance of 100 %. Compliance teams were given audit logs. Users of BRM did not receive any service alerts. A successful framework that can be reaffirmed in all the BRM settings [25]. This was to be used to approve future upgrades as well. The positive effect was easy to quantify. Revenue flow in the amount of $850 million was not affected. BRM modernization demonstrated the end-to-end reliability of enterprises. The architecture was turned into a standard by default of significant services. The continuity of business was maintained to the highest level.

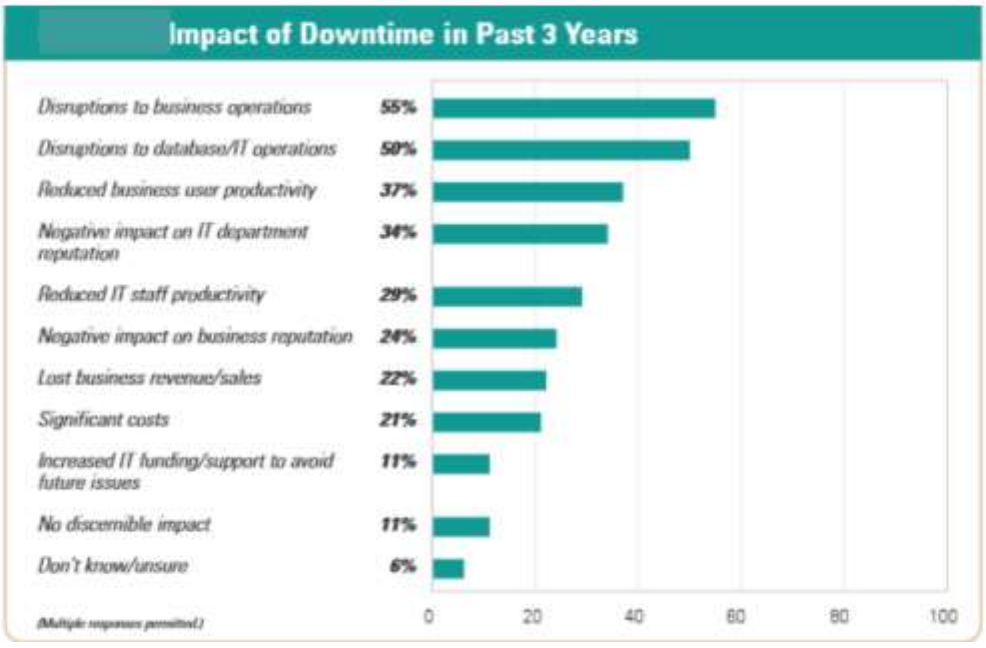


Figure 1: *Impact of downtime on business and IT operations [26]*

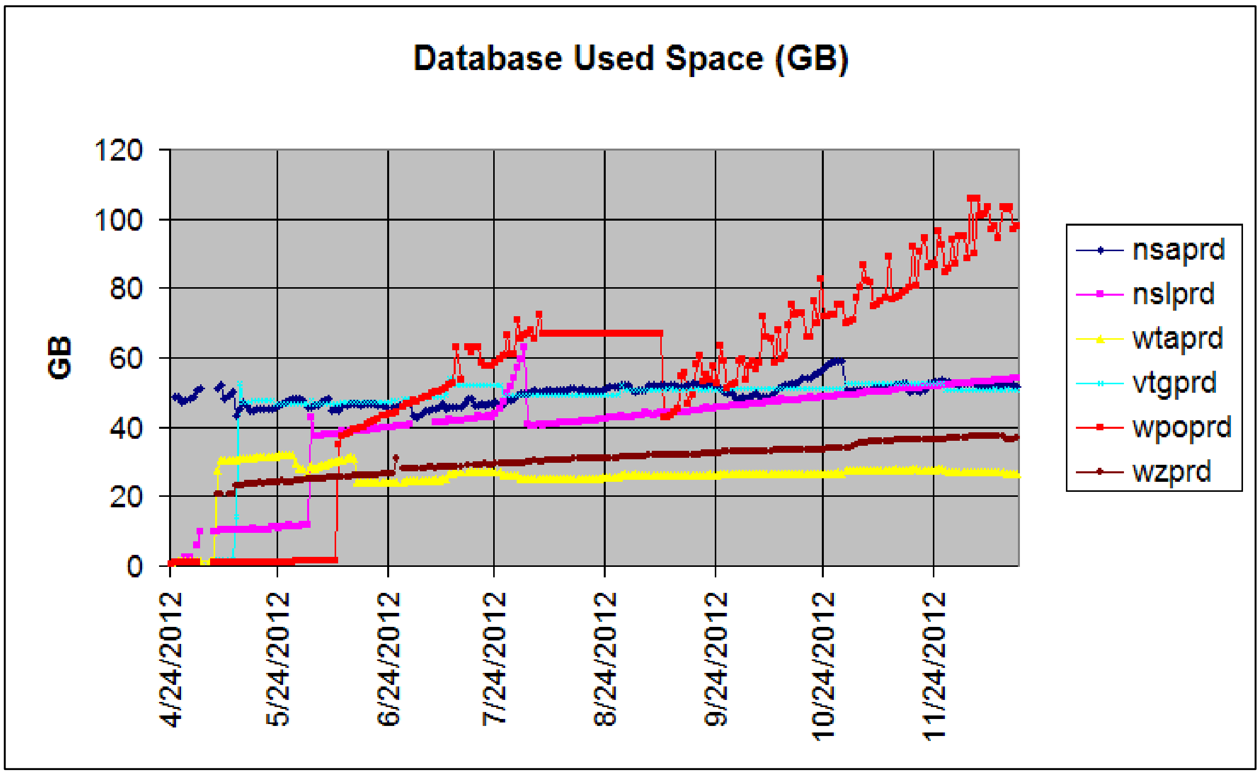


Figure 2: *Using EM 12c repository to find ASM growth rate on database used space in (GB) [27]*

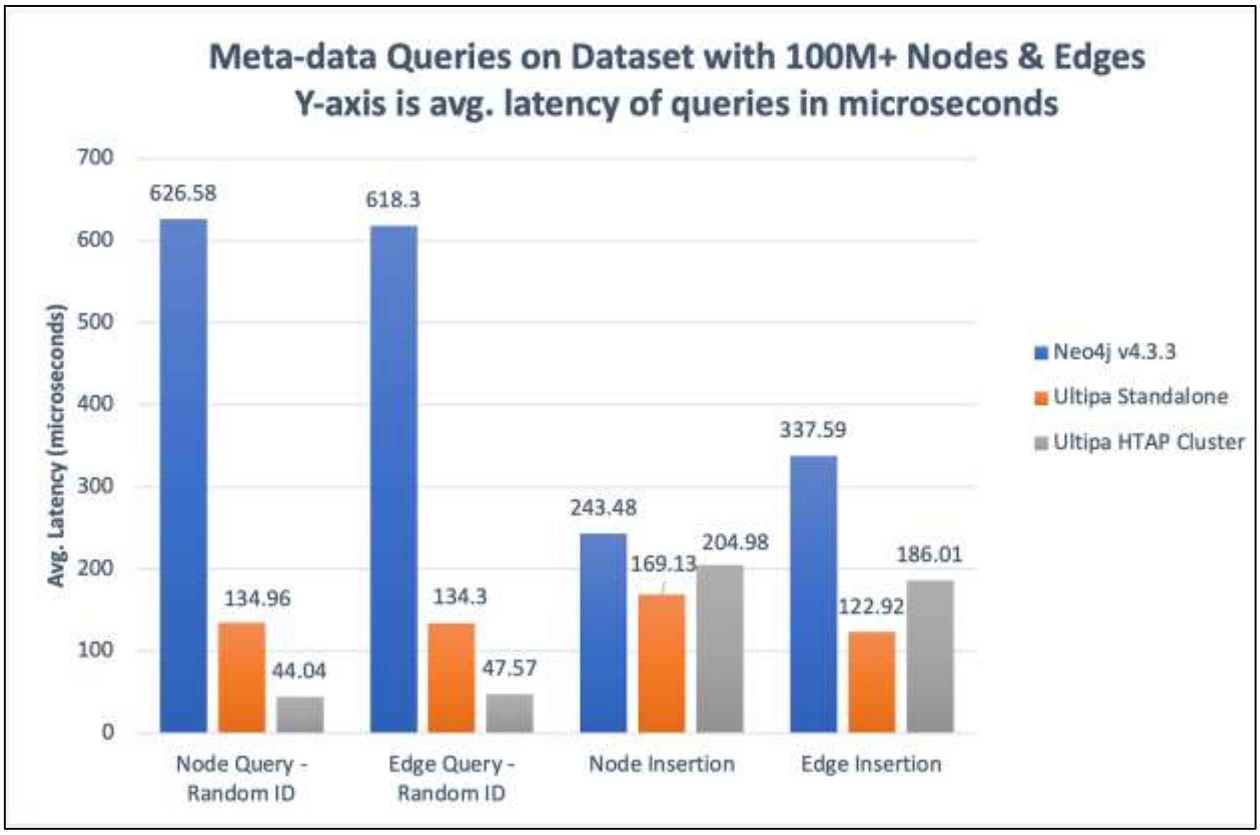


Figure 3: *Performance Advantages of HTAP Architecture [28]*

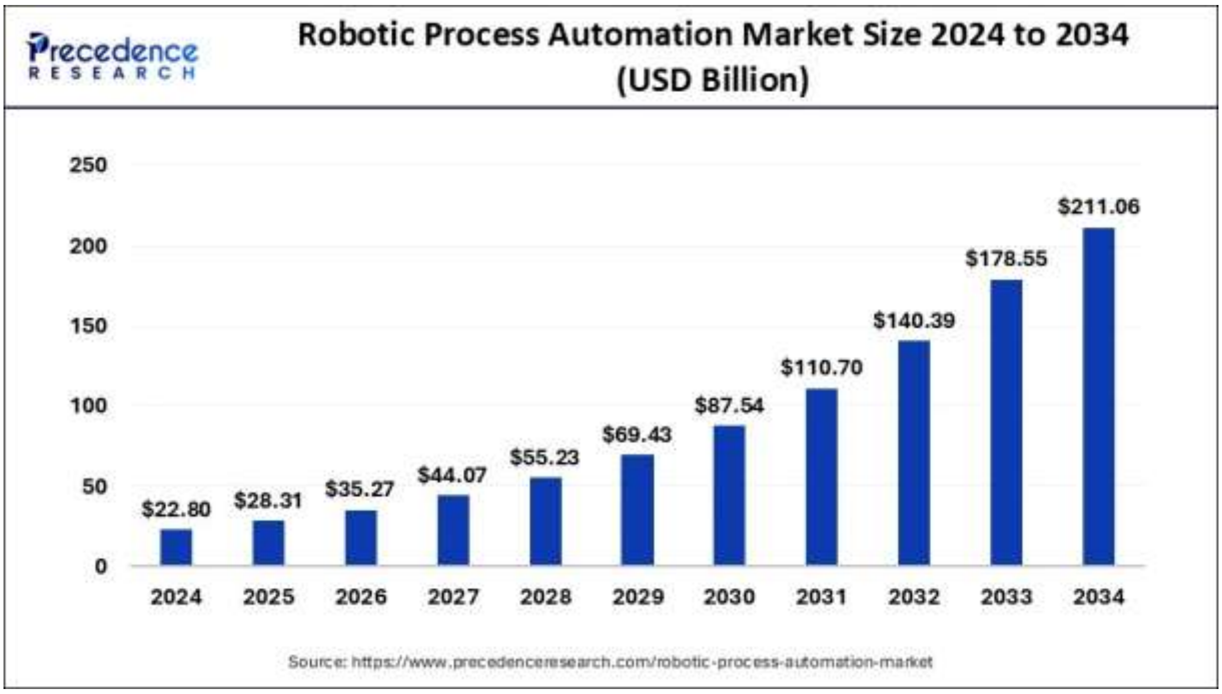


Figure 4: *Robotic Process Automation Market Size 2024 to 2034 [29]*

6. Conclusions

The paper had covered a well-tested automated framework to accomplish a near-zero-downtime availability during Oracle database upgrades and patching of the Grid Infrastructure in high-availability applications. Using that, with the framework of Oracle Oracle DBMS_ROLLING and Transient Logical Standby combined with Ansible-driven automation, the framework avoids service disruption and operational risk that conventional practices will entail. It has been tested in more than 5,000+ databases and 40+ enterprise systems with a noteworthy result including cost savings to the tune of over 100 million dollars and zero violation of SLA. The solution has worked on various platforms such as Exadata, from the Oracle VM, and OCI Bare Metal. This paper validates that constant availability, proactive validation, and the rollback aptitude is simply not a dream but a reality with a strongly integrated, scaled, and production-tested upgrade approach.

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.

References

[1]Malhotra, A., Elsayed, A., Torres, R. and Venkatraman, S., 2023. Evaluate solutions for achieving high availability or near zero downtime for cloud native enterprise applications. *IEEe Access*, *11*, pp.85384-85394.

[2]Oloruntoba, O., 2024. Business continuity in database systems: The role of data guard and oracle streams. *World Journal of Advanced Research and Reviews*, *22*(3), pp.2266-85.

[3]Manda, P., 2023. Migrating Oracle Databases to the Cloud: Best Practices for Performance, Uptime, and Risk Mitigation. *International Journal of Humanities and Information Technology*, *5*(02), pp.1-7.

[4]Brian, P., George, T. and Edward, W., 2023. Enhancing Disaster Recovery and Failover Strategies in Oracle Cloud Databases.

[5]Lasantha, N.C., Abeysekara, R. and Maduranga, M.W.P., 2023. Enhancing Security in Database

Grid Infrastructure for Storage Clusters. *WSEAS Transactions on Computers*, *22*, pp.233-242.

[6]Santoso, M.H. (2021). Application of Association Rule Method Using Apriori Algorithm to Find Sales Patterns Case Study of Indomaret Tanjung Anom. *Brilliance: Research of Artificial Intelligence*, 1(2), pp.54–66. Available at: https://doi.org/10.47709/brilliance.v1i2.1228

[7]Higginson, A.S., Bostock, C., Paton, N.W. and Embury, S.M. (2022). Placement of Workloads from Advanced RDBMS Architectures into Complex Cloud Infrastructure. *Research Explorer The University of Manchester*. [online] Available at: https://doi.org/10.48786/edbt.2022.43

[8]Hallin, J. (2025). *Evaluation of TLS and mTLS in Internet of things systems*. [online] DIVA. Available at: https://www.diva-portal.org/smash/record.jsf?pid=diva2:1937634 [Accessed 5 Aug. 2025].

[9]WilliamDAssafMSFT (2025). *Serverless compute tier - Azure SQL Database*. [online] Microsoft.com. Available at: https://learn.microsoft.com/en-us/azure/azure-sql/database/serverless-tier-overview?view=azuresql [Accessed 5 Aug. 2025].

[10]Oracle (2025). *Using DBMS_ROLLING to Perform a Rolling Upgrade*. [online] Oracle.com. Available at: https://docs.oracle.com/database/121/SBYDB/dbms_rolling_upgrades.htm [Accessed 5 Aug. 2025].

[11]Orlando, K.R. (2021). Automating Virtual Patching via Application Security Testing Tools. *Ntnu.no*. [online] Available at: no.ntnu:inspera:74730471:46733353 [Accessed 5 Aug. 2025].

[12]Bekir Tolga Tutuncuoglu (2025). Zero-Downtime AI: Predictive and Autonomous Server Restoration Without Human Input. [online] Available at: https://doi.org/10.2139/ssrn.5249062

[13]Simon (2022). *21c, Zero-Downtime Oracle Grid Infrastructure Patching - Fernando Simon*. [online] Fernando Simon. Available at: https://www.fernandosimon.com/blog/21c-zero-downtime-oracle-grid-infrastructure-patching/ [Accessed 5 Aug. 2025].

[14]Hansen, D.O. (2024). *xtts – Databases Are Fun*. [online] Databases Are Fun. Available at: https://dohdatabase.com/tag/xtts/ [Accessed 5 Aug. 2025].

[15]Ma, Z., Austgen, J., Mason, J., Durumeric, Z. and Bailey, M. (2021). Tracing your roots. *Proceedings of the 21st ACM Internet Measurement Conference*, pp.179–194. Available at: https://doi.org/10.1145/3487552.3487813

[16]Kasarapu, S., Shukla, S. and Sai, D. (2024). *Enhancing IoT Malware Detection through Adaptive Model Parallelism and Resource Optimization*. [online] arXiv.org. Available at: https://arxiv.org/abs/2404.08808 [Accessed 5 Aug. 2025].

[17]Nicolazzo, S., Nocera, A. and Pedrycz, W. (2024). *Service Level Agreements and Security SLA: A Comprehensive Survey*. [online] arXiv.org. Available at: https://arxiv.org/abs/2405.00009

[18]Punithavathy, E. and Priya, N. (2024). Auto retry circuit breaker for enhanced performance in microservice applications. *International Journal of Electrical and Computer Engineering (IJECE)*, [online] 14(2), p.2274. Available at: https://doi.org/10.11591/ijece.v14i2.pp2274-2281

[19]Raval, V. (2025). Recommendations to Improve Project Delivery Process for Better Customer Experience and Trust. *Theseus.fi*. [online] Available at: http://www.theseus.fi/handle/10024/892298

[20]Parker, S.K. and Grote, G. (2022). Automation, Algorithms, and Beyond: Why Work Design Matters More than Ever in a Digital World. *Applied Psychology*, 71(4), pp.1171–1204. Available at: https://doi.org/10.1111/apps.12241

[21]Andile Dlamini (2024). Machine Learning Techniques for Optimizing Recurring Billing and Revenue Collection in SaaS Payment Platforms. *Journal of Computational Intelligence, Machine Reasoning, and Decision-Making*, [online] 9(10), pp.1–14. Available at: https://morphpublishing.com/index.php/JCIMRD/article/view/2024-10-04 [Accessed 5 Aug. 2025].

[22]Fu, G. and Bryk, G. (2024). BrM Quantity-Based Bridge Element Deterioration/Improvement Modeling and Software Tools. [online] Available at: https://doi.org/10.36501/0197-9191/24-005

[23]Jagruti Jasleniya (2018). *Using DBMS_ROLLING to Upgrade the Oracle Database - ORACLE-HELP*. [online] ORACLE-HELP. Available at: http://oracle-help.com/dataguard/using-dbms_rolling-to-upgrade-the-oracle-database/ [Accessed 5 Aug. 2025].

[24]Mensah, I.O., Barrett, B. and Cahalane, C. (2024). Assessing Change Point Detection Methods to Enable Robust Detection of Early Stage Artisanal and Small-Scale Mine (Asm) in the Tropics Using Sentinel-1 Time Series Data A*Mensah Isaac Obour, Bbarrett Brian, and Acahalane Conor a Department of Geography, Maynooth University, Ireland. B School of Geographical and Earth Sciences, University of Glasgow, Scotland, UK. [online] Available at: https://doi.org/10.2139/ssrn.4991469

[25]Baljon, K., Romli, M.H., Ismail, A.H., Khuan, L. and Chew, B.-H. (2022). Effectiveness of Breathing Exercises, Foot Reflexology and Massage (BRM) on Maternal and Newborn Outcomes Among Primigravidae in Saudi Arabia: A Randomized Controlled Trial. *International Journal of Women's Health*, Volume 14, pp.279–295. Available at: https://doi.org/10.2147/ijwh.s347971

[26]oracle.com (2025). *Zero Downtime Database Upgrade Using Oracle GoldenGate*. [online] Oracle.com. Available at: https://www.oracle.com/technetwork/middleware/goldengate/overview/ggzerodowntimedatabaseupgrades-174928.pdf [Accessed 6 Aug. 2025].

[27]Balbekov, A. (2013). *Using EM 12c repository to find ASM growth rate*. [online] OracleQuest. Available at: https://oraclequest.wordpress.com/2013/01/09/usin

g-em-12c-repository-to-find-asm-growth-rate/ [Accessed 6 Aug. 2025].

[28]Sun, R. (2023). *What should you know about Graph Database's Scalability?* [online] Medium. Available at: https://blog.devgenius.io/what-should-you-know-about-graph-databases-scalability-47c794da5c0b?gi=208d862cb7de [Accessed 6 Aug. 2025].

[29]Precedence research (2025). *Robotic Process Automation Market Size, Report 2023-2032*. [online] www.precedenceresearch.com. Available at: https://www.precedenceresearch.com/robotic-process-automation-market [Accessed 6 Aug. 2025]..