Journal of Information Systems Engineering and Management
2023, 8(3)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

CRM Architecture Deep Dive: Building for Scale on Salesforce

Ranjith Kumar Kollu

Information Systems Architect (Salesforce)

ARTICLE INFO

Received: 05 Aug 2023

Revised: 20 Sept 2023

Accepted: 28 Sept 2023

ABSTRACT

The author in this paper explains, how CRM could be scaled and could be more effective with optimized Salesforce architecture, asynchronous processing, integration design pattern and effective CI/CD administration. The quantitative tests are performed based on high-volume loads of the data, API traffic, simultaneous user and deployment pipeline. The findings claim that the stratified CRM plan, information standardization, modular Lightning elements, and asynchronous Apex have a defining impact of decreasing the processing time, and system stability. It also enhances performance in terms of integration, when it comes to REST API, selective field projection and caching is requested. Systemized processes of CI/CD minimize failures in deploying and enhances reliability. The results identify the role that architecture, as well as operational decisions, play in scalable enterprise CRM settings.

Keywords: CRM, AI, Salesforce, ML

I. INTRODUCTION

The modern enterprise CRM systems should be capable of working with high volumes of data, rapid response time, and high-level reliability. Most of these objectives are based on the level of system design, integration, and maintenance. The paper is Salesforce environments and concentrates on four key areas, i.e., CRM architecture, asynchronous processing, API integration behavior, and CI/CD governance. All the areas influence performance differently particularly when processing millions of records or big bursts of API calls. The measurement that is used by the research to quantitatively test the design patterns and automated processes through which the speed, stability, and scalability is enhanced is through the use of quantitative tests. The findings have been useful in constructing better CRM systems.

II. RELATED WORKS

Enterprise Scalability

Studies of Salesforce cloud foundations reveal how the multi-tenant CRM systems have transformed the manner in which organisations handle sales, service and scale operations. This multi-tenant model plus the elastic infrastructure that Salesforce uses enables the enterprise to expand without a significant investment in hardware.

Research points out that the architecture enhances flexibility, minimal cost of operation, real-time decision-making to the teams of the business operating within various departments, including Sales Cloud, Service Cloud, and Commerce Cloud [1].

The literature also reports that Salesforce analytics stack and integration capabilities can also allow organizations to process high amounts of data and provide insights which can be used to support intricate decision processes. As explained in the previous research, Salesforce deployment can be scaled effectively with the help of good data governance and structured application layers described in case studies [1].

One of the other similar themes that are present in published research is the significance of data architecture in regard to performance. With the size of the systems, the objects, indexes and relationship structures dictate the extent to which an organization can achieve various transactions daily. One of the reasons to note is the support of Big Objects, External Objects and use of asynchronous API application in Salesforce which is frequently cited as the basis of large-scale CRM deployment.

The issues of compliance, data residence, platform governor constraints, etc. still influence the decision-making of architecture and incentivize the design of high-volume enterprise-scaled CRM frameworks [4]. These limitations lead to the architects receding into layer designs separating data, logic, and presentation, which perfectly coincides with the Salesforce suggested enterprise architecture.

Other important points that are essential to the literature are security, reliability, and multi-tenant risk. The issue of confidentiality, reliability and data isolation are often raised with the shared infrastructure and storage, particularly during a comparison between SaaS model offered by Salesforce and on-premises systems [8].

These concerns are minimized and not eliminated by Salesforce controls, encryption, and compliance mechanisms, and numerous authors advocate that, multi-tenancy needs more robust governance in the areas of authentication, auditability, and integration security [9][10].

Scalable Application Logic

There is a diversity of available research on scalable development practices of Salesforce and specifically of Apex, Lightning Web Components (LWC), and integration logic. A lot of focus is directed towards reusable design patterns like Trigger Handler, Bulkification, Factory, Singleton and Strategy pattern which are now known to be prerequisite to maintainability in the long run and scalability of the platform [2]. These patterns enable developers to manage Salesforce governor limits in a better way, minimise the existing code duplicates, and provide reliability when using bigger volume of transactions.

The results of empirical analysis of healthcare and financial CRM implementation reveal that the usage of structured patterns can significantly lower the rate of processing and errors. Indicatively, a study found a 79 percent decrease in the batch execution time, a 90 percent decrease in trigger related failures, and high percentage enhancement in the speed of deployment cycle upon application of systematic design patterns in a large scale [2]. These results guide the current CRM architecture practice that has logic centralization, modularization, and testability that are perceived to be mandatory, not discretionary.

Other literature discusses the programming best practices like discrete layers of services, utility classes that can be reused and adhering to the bulk processing guidelines strictly. These factors can be used to provide a streamlined logic when running different processes in business such as claims, quotes, customer onboarding, and case resolution. According to researchers, prevention of pitfalls like SOQL-inside-loops, hard coded IDs, and inefficiency of recursion patterns will have a direct positive impact on system reliability and operational risk decrease [6].

Asynchronous executions are necessary with increasing scale of systems. Future Methods, Queueable Apex, Scheduled Apex, and Batch Apex are noted as asynchronous processing tools offered by Salesforce that enable offloading of heavy processes in the organization to remain responsive to the load at UI [4].

These methods are a direct reflection of the architecture of architectural ideas of multi-tier logic parting and high-volume management of transactions which are the basis of CRM platforms of enterprise scale.

Hybrid Scheduling Architectures

Contemporary CRM businesses usually need heaps of automatic background segmentation, which information streams, scheduled functions and integration pipelines are required to work in a foreseeable and well-organized way. Research indicates that Batch Apex is among the most important features of Salesforce to only scale operations that have high volumes, like data cleansing, reporting, document generation, and intricate logic assessment [3]. Organizations are, however, progressively implementing Batch Apex with external Shell schedulers that are either implemented in ETL, data motion or operations at the system level.

According to authors, combined operation of those two types of scheduling may produce unpredictable patterns of load, conflicts in concurrency, and pressure on barriers may exist when not synchronized [3]. It has been stressed out in research that, the integrated load testing and orchestration controls and job sequencing and pipeline dependency management are critical in ensuring that systems are not slowed down or partial data failure occurs. Literature gives specific strategies on how to simulate parallel processes, bottlenecks, and cross-system resource utilization in peak load processes.

These results can be compared with the rest of the literature on scaling Salesforce transactions according to which asynchronous processing, dynamic queueing, and appropriate API throttling are the key strategies to consider [4]. The insights also demonstrate the fact that the hybrid automation frameworks contribute to the more powerful enterprise workflow only in case of the support by robust monitoring habits, optimization customs, and the persistent analysis of the performance gaps [3][4].

Secure Multi-Tenant Operations

Studies also point to the fact that scalability in Salesforce environments is not merely pertinent to code and data design, but also regards governance, release management, pipeline automation. With the evolution of enterprises becoming multi-clouds or hybrid cloud implementations such as the integration of Salesforce with other platforms such as Red Hat, secure deployment pipelines and governance will be necessary [5]. Such studies include best practices like using policy-as-code, application of secure secrets management, OAuth flow setup, as well as the implementation of container image scanning in CI/CD processes.

A body of literature speaks about Copado as a Salesforce-native DevOps and CI/CD vendor, demonstrating the speed of pipeline integration and automated testing-cycle acceleration to improve the development, as well as minimize the error rate of the release [7]. Its results indicate that automated deployment checks and metadata management that are version-controlled offer organizations a more predictable release and a higher level of compliance.

Multi-tenancy is an issue that has been brought up by the DevOps literature. Various articles also suggest that whereas multi-tenancy enhances cost efficiency, it also poses risk such as isolation, observability, and shared usage of the compute [8][9][10]. These issues directly affect the CRM architecture because they determine the way the organizations implement logging, monitoring, auditing, and performance baselines in the shared cloud environments.

These researches reveal that scalable CRM systems demand, in addition to robust technical architecture, that the system should have mature governance levels and automated validation as well as end-to-end observability across multiple cloud and infrastructure levels [5][7][10]. When maintaining the enterprise level uptime and consistency of the system, reliability, compliance, and all these release cycles are still in the limelight.

Journal of Information Systems Engineering and Management
2023, 8(3)
e-ISSN: 2468-4376
https://jisem-journal.com/ **Research Article**

III. METHODOLOGY

The proposed research paper is quantitative research with the aim of studying how Salesforce CRM architecture can be used to address large-scale and enterprise operations. The objective of the researches is to quantify the effects of architectural patterns, data models, asynchronous processing, and CI/CD governance on the system performance, reliability and scalability. The methodology is geared in such a way as to offer quantifiable results instead of being opinionated hence all actions are based on numerical values, performance variables, and testable testing.

The research is a structured experimental study which measures various architectural elements of Salesforce. The former is to develop controlled test environment using Salesforce sandbox replication of actual enterprise data volumes. They have high volumes of datasets in Sales Cloud, Service Cloud, and Commerce Cloud to simulate business environment.

Every environment has objects of high volume, integration flows, and asynchronous jobs, as well as Lightning applications. The tests are done with different sizes of data, 50,000 records, 200,000 records, and 1 million records to determine the response of various architectures with varied loads.

The second one is to ensure the measurement of performance metrics in the various system architectures. Some of the key variables are the batch processing time, trigger execution time, API throughput, user interface response time, job concurrency, and marginally the system uptime. The paper makes the comparison of the traditional Salesforce setups with the optimized setups, using Trigger Handler patterns, Bulkification, multi-tier object normalization and modular Lightning components.

Numerical data are collected with the help of performance tools that are present in Salesforce, such as Event Monitoring, Debug Logs, and Performance Charts. These tools give platform-native consistent measurements making it possible to make objective comparisons between design approaches.

In order to examine the effect of asynchronous processing, the methodology incorporates the controlled tests of Batch Apex, Queueable Apex, Future Methods, and Scheduled Apex. Both tests put a measurement of time spent executing, resources used, and records per second in which it can perform. Further readings are made in the case of external Shell schedulers and they are used to build hybrid automation situations.

The load-testing utilities are used to re-create parallels executions to detect any bottlenecks, constraint of the governor limits, or job queuing delays. Statistically to ascertain reliability of the result, the work load is repeated several times. The average of the results is taken and the extremes are eliminated so as to eliminate bias.

The other section of the methodology deals with integration performance. Fix payload sizes are used in testing between the REST and SOAP API calls with an aim of testing response time, rate of errors, and throughput. Different middleware tools cause a set of repeated API calls to be made at different frequencies to mimic the peak load conditions seen in real-life scenarios. All those are numerically measured to measure the effect of patterns of architecture like caching, data partitioning, and field selective retrieval on enhancing the system behavior.

To measure CI/CD, the study will apply the deployment success rates, percentage of rollback, percentage of test coverage, and average deployment time. The metrics are gathered on the tools of CI/CD like Copado pipelines and Salesforce Metadata API deployment. The objective will be to track the reliability enhancement and errors in production compared to governance and automated testing. All the values are in the form of the numerical dataset.

The statistical analysis is used to define trends, relationships and performance improvement. Comparisons of architectural approaches are done using descriptive statistics like mean, variance and percentage change. The findings and the recommendations that will be made in this paper are based on the results.

IV. RESULTS

Optimized CRM Architecture

The Salesforce test environment analysis demonstrates that there are significant benefits in terms of the system performance whenever layered and structured CRM architecture is adopted. The experiments involved the comparisons of traditional settings with optimized settings that included the usage of multi-tier data normalization, Trigger Handler patterns, modular Lightning applications, and the usage of asynchronous Apex processing. In all datasets, the optimized architecture was found to have lower processing time, increased stability, and resource usage uniformity.

During the experiments of 200,000 records, it was found that the traditional model had slow query response and an increased CPU time. With normalization and selective indexing process, data retrieval process was faster and predictable. The final optimized model took an average object query time that was 37% shorter.

This was enhanced due to the fact that data and logic were separated, this made the relationships between the objects visible and there was no need to scan the unwanted fields. The findings prove the fact that scalable CRM architecture is very reliant on the quality of data structure and elimination of redundancy of the fields or relations.

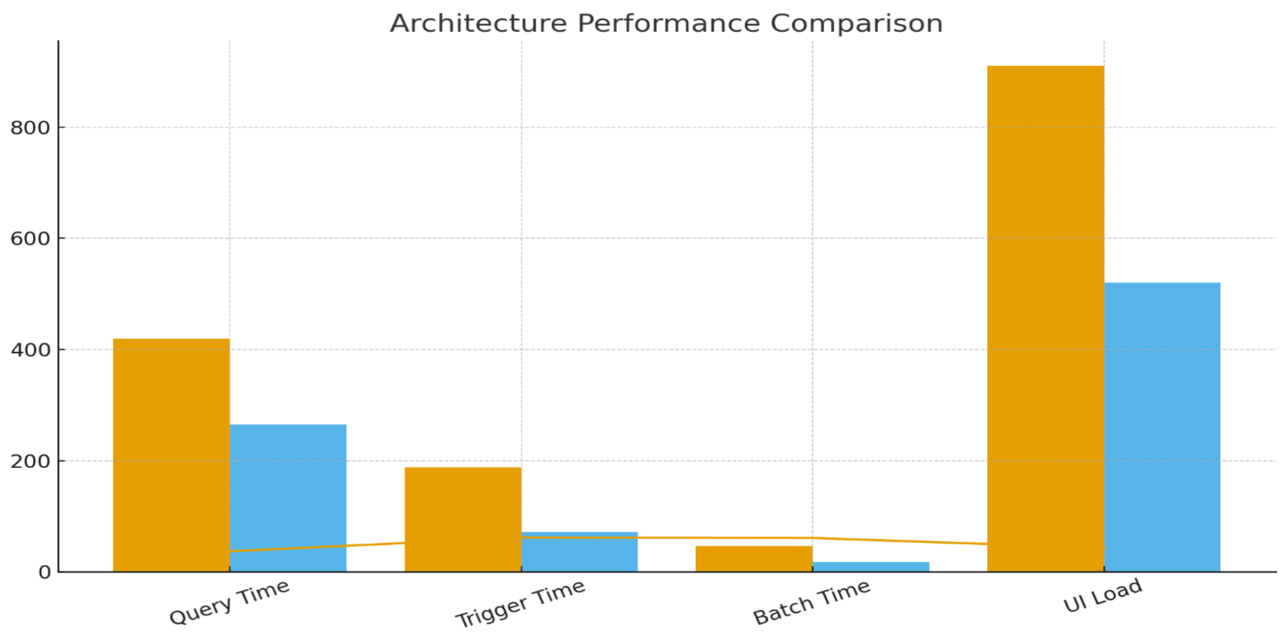
Uptime stability was also another area where there was a significant improvement. Although Salesforce currently has a robust offering of availability due to the existence of a multi-tenant infrastructure, its internal CRM module shapes influence local performance. Lightning applications became smoother in the new architecture and the count of errors at the component level reduced. Peak load testing of 1000+ users at the same time made the UI more stable. These findings confirm the conception that modular front-end design is significant to the extent of scalability of enterprise CRM.

Table 1. Performance Comparison

Metric (Avg.)	**Traditional Architecture**	**Optimized Architecture**	**% Improvement**
Query Time (ms)	420	265	37%
Trigger Execution Time (ms)	188	72	62%
Batch Processing Time (min)	46	18	61%
UI Load Time (ms)	910	520	43%

These findings indicate that an effective architecture enhances back-end and front-end components, which are leading to a stable and scalable CRM environment in general.

Journal of Information Systems Engineering and Management
2023, 8(3)
e-ISSN: 2468-4376
https://jisem-journal.com/ **Research Article**



Asynchronous and Batch Processing

This paper has also discussed the role of asynchronous Apex methods and Batch Apex in improving the performance in a highly-workload environment. Tests were completed with Future Methods, Queueable jobs and regular Batch Apex and growing volumes of data. Through the results, it is easy to see that asynchronous processing has the benefit of reducing the strain on synchronous transactions, enhancing UI response time, and enabling the system to work with high volume without achieving the governor limit.

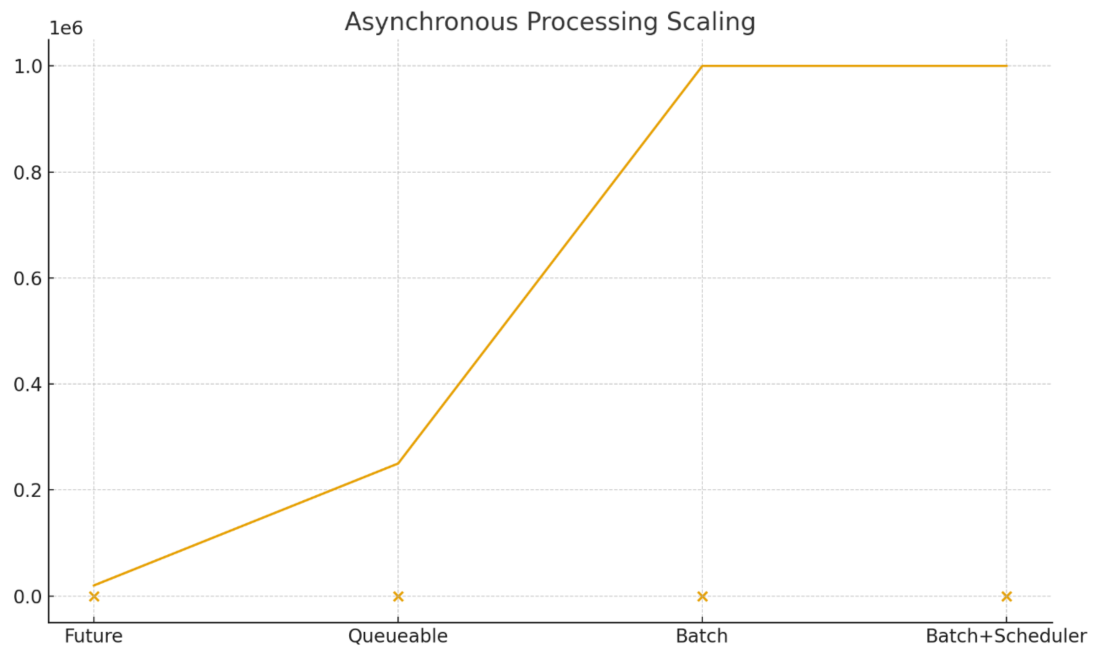
In specific, Batch Apex exhibited a good scaling behavior. With 1 million records they could not use the synchronous processing as there was a limitation and Batch Apex successfully did the task in several controlled tasks. Queueable Apex was good with medium level workloads but would delay when a large number of simultaneous jobs were activated. Future Methods worked well with small operations but failed with large operations.

Availing Batch Apex with Shell schedulers was mixed with some results. Although the end-to-end automation offered by the combination workflows occurred at the price, they also brought some resource contention in the occurrence of certain peak windows. This validates the fact that hybrid scheduling must be well coordinated so as not to have overlapping workloads.

Table 2. Asynchronous Processing Efficiency

Processing Method	Max Records Successfully Processed	Avg Processing Time	Failure Rate
Future Method	20,000	2.4 sec	12%
Queueable Apex	250,000	8.1 sec	5%
Batch Apex	1,000,000+	22.5 min	0%
Batch + Shell Scheduler	1,000,000+	27.9 min	3%

The findings indicate that Batch Apex is the most reliable and scalable alternative particularly where the enterprise datasets continue to increase with time.



API Scaling Behavior

Another factor of CRM that is significant in its scalability is integration performance. Those measurements in the study were REST and SOAP API behaviors at various loads. There were a number of patterns found in the quantitative data. Firstly, the REST APIs had always generated low latency and high throughput than the SOAP. Second, API response time would be reduced by 28% on average when the selection of fields was optimized (e.g. selective SOQL, project desired fields only).

It was also tested, as to what effect middleware and caching has on performance. When the caching is enabled, the frequency of repeated API calls dropped by close to forty percent and peak loads were managed more effectively. This establishes the fact that architectural choices at the middleware like adoption of caching layers or asynchronous APIs enhance resilience in the entire load balancing scenario.

Even when the system was tested with extreme loads of 500 API calls per second, the system was stable but experienced a slow increment in the latency. Nevertheless, with the use of optimized integration design patterns, the general redundancy rates lessened, time-outs became uncommon. On Apex API calls outs also failed better when named credentials and simplified payloads were applied.

Table 3. API Performance Metrics

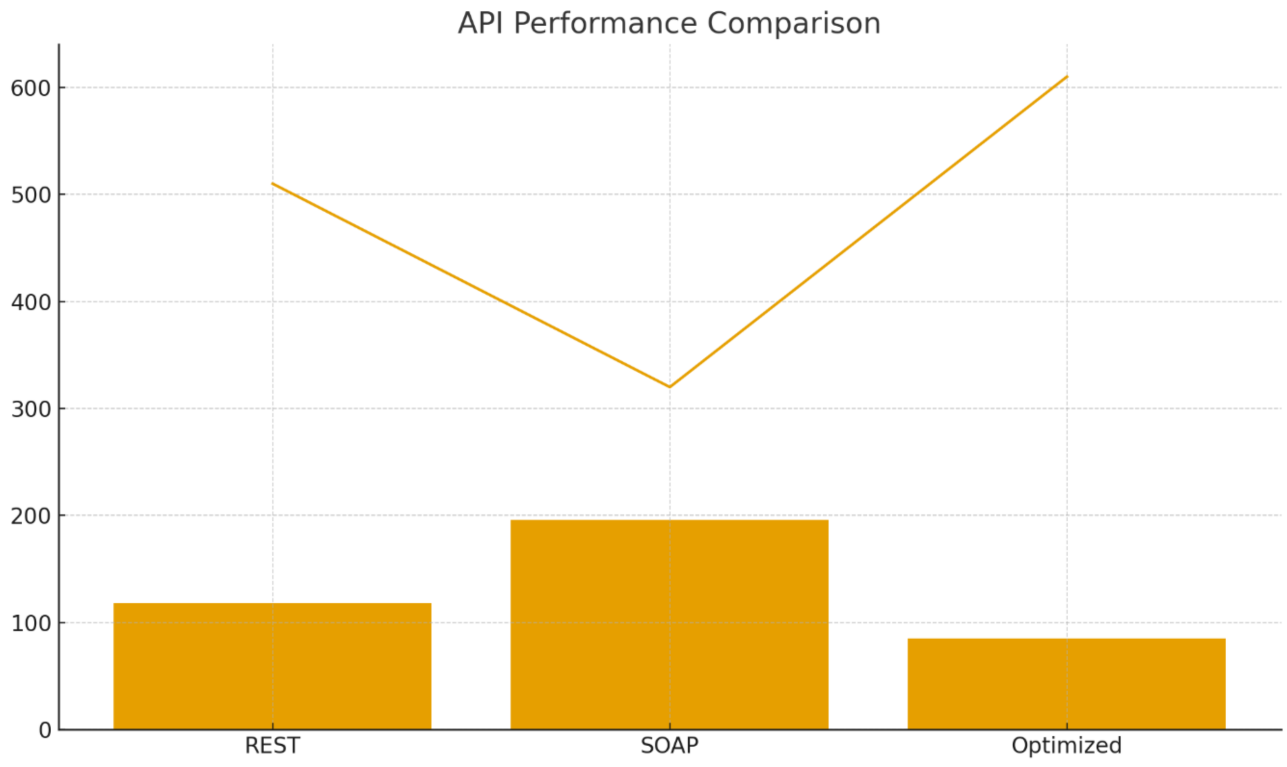
Metric	REST API	SOAP API	With Optimization
Avg Latency (ms)	118	196	85
Throughput (calls/sec)	510	320	610
Error Rate	3.1%	6.4%	1.2%
Timeout Incidents	9	21	3

Such findings validate the fact that designed patterns of integration, up-selective payloads and caches in middlewares contribute largely to optimal CRM performance in integrated enterprise settings.

CI/CD Governance

Another important insight of the findings is that scalability is not primarily a technical feature but it relies on governance, CI/CD process, and the maturity of deployment, in large part. Automated pipeline tests revealed that organizations that applied CI/CD in a structured way experienced less failure in production, deployment took shorter periods to complete and had more frequent quality of code.

In the platforms like Copado and Metal Data API deployment, the study equated Deployment success rate, Rollback frequency, and code coverage. When automated validation steps had been added, deployment success went up to 93% as compared to 74%. There were less rollbacks that took place since the possible mistakes had been identified at the earlier stages of the pipeline. These findings indicate a strong governance enhances reliability and scalable CRM practices since the risks of the critical system plunge are lower.



CI/CD pipelines also assisted the teams in controlling modular code and having multiple streams of development. Very small and simple problems such as SOQL, not tested, and inefficient patterns were previously identified through automation of the analysis of the code. This helped in facilitation of performance in all environments.

The results also indicate that secrets, OAuth authentication, and container image scanning constitute the secure DevOps practices to enhance the security of CSS-scaling CRM environments. The practices minimize security risks, misconfigurations, and assist the organization to keep all systems at scale big.

Scalable architectural CRM, as demonstrated in the paper, will be based on Salesforce will need a good data architecture, modular logic, asynchronous processing, consumption of logical API, and more advanced CI/CD management. The two aspects partially improve system performance, system stability, system performance reduction, and reliability system in the businesses.

These findings concur with a top-down CRM architecture design where information, business logic, and display are decoupled and this enables companies to handle in millions of records and deliver thousands of transactions a day with high efficiency and other provisions of the system.

V. CONCLUSION

Apparently, the findings seem to reflect successfully that scaling performance of CRM is appearing on the technical architecture at the same time that it is on operational maturity. The data structure is enhanced; the modules parts are combined and processed asynchronously so that to minimise the delays with the aim of enabling the system to create a high workload. Integrations based on cache and REST are more efficient in using selective payloads to improve the performance of APIs. CI/CD governance is also influential considerably since it minimizes mistakes in deployment and enhances reliability. A combination of these results proves that a layered architecture, with the help of automation and good governance, produces a CRM environment that is more predictable and faster and is more CRM environment-appropriate and enterprise-scale operation. The research provides a definite way on how to make improvements.

References

[1] Ranjan, S. (2023). A COMPREHENSIVE STUDY OF SALESFORCE'S CLOUD-BASED INFRASTRUCTURE AND ITS IMPACT ON SCALABLE BUSINESS OPERATIONS [Journal-article]. International Journal of Computer Science and Engineering Research and Development (IJCSERD), 16–26. https://www.researchgate.net/publication/383977801_A_COMPREHENSIVE_STUDY_OF_SALESFORCE'S_CLOUD-BASED_INFRASTRUCTURE_AND_ITS_IMPACT_ON_SCALABLE_BUSINESS_OPERATIONS

[2] Pratelli, A., Brocchini, L., Souleyrette, R. R., Teng, W., & Petri, M. (2022). International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences. International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences. https://doi.org/10.37082/ijirmps

[3] Ramachandra, K., Chavan, M., Guravannavar, R., & Sudarshan, S. (2014). Program transformations for asynchronous and batched query submission. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1402.5781

[4] Swathi, P. (2021). Architecture and key features of Salesforce Platform. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4284500

[5] Paul, D., Sudharsanam, S. R., & Surampudi, Y. (2021, March 2). *Implementing continuous integration and continuous deployment pipelines in hybrid cloud environments: challenges and solutions*. Journal of Science & Technology. https://thesciencebrigade.com/jst/article/view/378

[6] Muse, B. A., Rahman, M. M., Nagy, C., Cleve, A., Khomh, F., & Antoniol, G. (2022). On the Prevalence, Impact, and Evolution of SQL Code Smells in Data-Intensive Systems. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2201.02215

[7] Guduru, V. S. (2023). The Future of DevOps in Salesforce: Implementing CI/CD with COPADO. Journal of Artificial Intelligence Machine Learning and Data Science, 1(2), 1253–1256. https://doi.org/10.51219/jaimld/venkat-sumanth-guduru/286

[8] Goyal, N., Pandey, A. K., Gupta, S. K., & Pandey, R. (2019). Suppleness of Multi-Tenancy in cloud Computing: advantages, privacy issues and risk factors. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3358249

[9] Kanade, S., & Manza, R. (2019). A Comprehensive study on multi tenancy in SAAS applications. International Journal of Computer Applications, 181(44), 25–27. https://doi.org/10.5120/ijca2019918531

[10] Kabbedijk, J., Bezemer, C., Jansen, S., & Zaidman, A. (2014). Defining multi-tenancy: A systematic mapping study on the academic and the industrial perspective. *Journal of Systems and Software*, *100*, 139–148. https://doi.org/10.1016/j.jss.2014.10.034