Comparative Analysis for an Optimized Data Driven System

Chinmay Pophale, Ankit Dani, Aditya Gutte, Brijesh Choudhary and Vandana Jagtap

March 11, 2022

Comparative Analysis For An Optimized Data Driven System

C.Pophale[1], A. Dani[1], A. Gutte[1], B. Choudhary[1] , V.S. Jagtap[1]
Department of Computer Engineering
Maharashtra Institute of Technology, Pune.
chinmay2997@gmail.com, ankitdani1997@gmail.com, adityagutte@gmail.com, brijesh.choudhary7@gmail.com, vandana.jagtap@mitpune.edu.in

Abstract. While designing huge systems based on data access, storage and retrieval, there a lot of problems faced by developers including ensuring portability, cross platform support, scalable design, secure platforms and reduced latency and redundancy. These problems and need to discussed and tackled in order to increase performance and efficiency for the benefit of the end user. This study makes use of one such data driven problem involving huge quantities of astronomical data to demonstrate the advantages of using umpteen efficient technologies and services in order to achieve significant improvement in results. Further sections compare and analyse the pros and cons of different technologies and protocols in use in order to derive fruitful conclusions.

Keywords: Django, Flask, MySQL, JSON, XML, Volley, Retrofit, SOAP, Web-Services, Laravel.

1 Introduction

This paper presents a deep insight into the development and management of large data based systems through the use of various technologies in the market and also suggests the implementation of methodologies through a case study.

The performance metrics of any data driven system with an interface depend on the selection of the database to be used which further is affected by the type of data to be handled by the system i.e. structured or unstructured. The capacity and features of the web frameworks currently in market including Django, Laravel and Flask amongst others play an important role and so does their use alongside communication protocols like REST and SOAP.

A real life application based on above mentioned techniques is discussed. Comparative analysis with considerable improvement over existing systems is depicted.

2 Literature Survey

The analysis of the various relevant frameworks and technologies leads to determine that each tool handles files of diverse settings, in the same way it could be observed that each of the tools offers different types of useful features to facilitate the programmer, in his work environment.

The criteria for the selection of a database was studied thoroughly pointing out the features of various available databases[1]. Django, Flask and Laravel were the identified frameworks suitable for handling web applications of large data systems[2]. A critical review of most widely used web communication protocols like REST and SOAP was performed[3]. For providing requests and response functionality, latest networking services like Google Volley and Retrofit were compared[4].

3 Comparative Analysis

3.1 Database

For any system architecture the database is it's backbone. The database performance is the most important metrics to be considered. So selection of the database which suits your system is crucial for it's efficient working. Various parameters to be considered while choosing a database are Schemas, Structuring, Data Access time etc. The detailed study of many data bases have led to the following conclusions about the same.

SQL and No SQL

SQL(Structured Query Language) is the basis of RDBMS (Relational Database Management System). Few database based on SQL are MySQL, Oracle, MS SQL Server, IBM DB2, Microsoft Access etc. SQL has an upperhand when high speeds are required for data retrieval. Large amount of data which is in a standard format is best suited for SQL databases[5]. Also not much of coding is required while handling the data to and fro from the database. This makes database management and indexing process way simpler.

NoSQL are databases are unstructred databases with no fixed schemas attached to them. These are more scalable, more flexible and have a great extent of agility[6].

These provide unrestricted management of structured, semi-structured and unstructured data [7].

3.2 Web Frameworks

In any large scale data driven project, a web framework plays a pivotal role in the development of different web applications like web resources, web APIs and web services. A web framework in simple terms facilitates a reliable method to build and deploy these applications on the World Wide Web.

Django, Flask and Laravel

Django is a high-level web framework based on Python that encourages rapid development of websites. It handles much of the perks of web development, so there is more focus on writing the application without any need to reinvent the wheel[8].

On the other side, Flask is a lightweight and minimalist web framework. It lacks some of the predefined features provided by Django. But it helps developers to keep the core of a web application simple and extensible. Unlike Django, Flask is more difficult for users to handle administrative tasks as the former gives a ready to use admin functionality. Django also contains a bootstrapping tool called django-admin. Django-admin enables users to start applications without any external requirements. Django also provides a built in ORM (Object-Relational Mapping) system to facilitate multiple kinds of database connections which is lacking in Flask [9].

Laravel is a PHP developed web framework based on Model View Controller (MVC) compared to Django's Model View Template (MVT) design[10]. Laravel functions on significantly slower speeds than Django since Python is a fast language compared to PHP. Django also aids developers avoid the mistakes of web development and implement efficient security measures. While Laravel also covers security but it doesn’t come close to Django’s measures. That’s the reason why, for example, NASA uses Django for their web portals and applications.

3.3 Web API Services: REST and SOAP

REST (Representational State Transfer) and SOAP (Simple Object Access Protocol) are primarily used communication protocols[11]. REST functions through a solitary interface to access resources while SOAP exposes components of application as services instead of data. REST allows more variety of data formats and SOAP only works with XML[12]. Compared to SOAP, REST is significantly faster and uses less bandwidth. It also offers better support for browser clients[13].

3.4 Networking Services: Volley and Retrofit

Volley is a networking library which inculcates helpful features like synchronous \& asynchronous requests, priority handling, multiple \& ordered requests, JSON parsing and of course caching advantages[14]. Retrofit is a REST client library for Android, through which a user can make easy to handle interfaces. Retrofit performs async and sync requests with automatic JSON parsing. Unlike Volley, it does not support caching. Also, Volley allows the retrying of requests along with modified timeouts automatically which Retrofit does not support. Volley supports inbuilt image loading features while Retrofit needs third party libraries[15]. Retrofit, as an advantage, supports more varied output formats than Volley.

4 Case Study

The following case study highlights the development of an optimized architecture for ASTROSAT Data Quality Reports (DQRs). ASTROSAT is India's first dedicated multi-wavelength space observatory which was launched on a PSLV-XL on 28th Sep 2015. The architecture works as a backend for an Android application which was created to fetch and display the above mentioned DQRs..

4.1 MySQL Database

The project requires handling of large sets of structured astronomical data for which an SQL database (MySQL) proved to be the best for use when it comes providing effective data insights for data analysis, minimal latency in response time and facilitates simpler database management.

4.2 Django Framework

The architecture described in the case study makes use of the distributed system functionality of Django web framework which involves the creation of a
separate "app" for each feature of the app. Use of Django significantly enhances the scalability, ease of use, robustness and security characteristics of the proposed architecture.

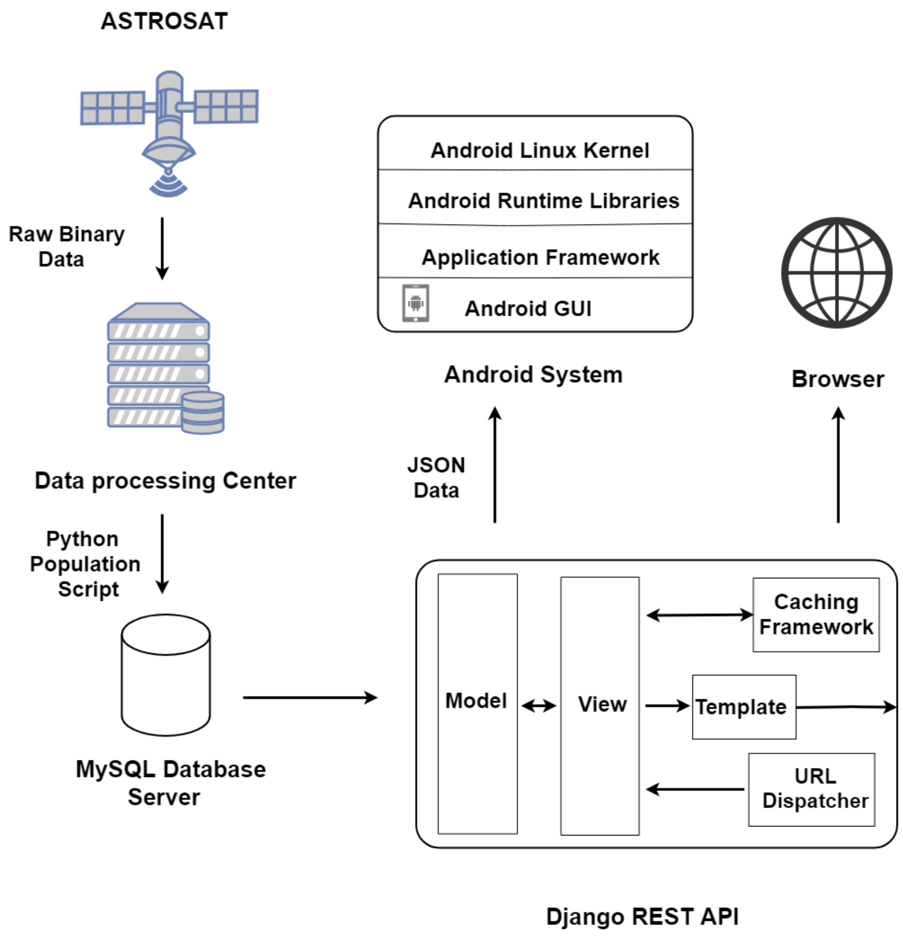

Fig. 1. Proposed System for ASTROSAT DQRs featuring a database server with an efficient Django REST API.

4.3 REST API

In the implemented RESTful API, endpoints (URLs) define the structure of the API and how end users access data from the app. REST framework is used as a base for Serialization which allows complex data such as query sets and model instances to be converted to native Python data types that are then easily rendered into JSON.

4.4 Google Volley

Google Volley adds some powerful options and is a ton quicker than other alternatives like AsyncTask and Retrofit. In this architecture, instead of creating a new instance of RequestQueue every time, the singleton pattern of creating a single instance of RequestQueue throughout the application is followed. This results in faster JSON retrieval speed. Caching capabilities of the Volley framework helps the app to pre-fetch data for faster loading.

4.5 Results

The following representation proves that the correct combination of technologies compared and analysed above is beneficial towards the effective development of any large scale data driven system. This case study implementation was compared with an existing system which lacked the use of an application server, web framework, services and data caching. The new proposed system achieves 97.2\% efficiency over the existing system. The above graph clearly demonstrates the improvement over the existing architecture shows exponential increase in response time whereas the proposed architecture gives us a linear minuscule increase. The proposed system tries to minimize the scaling time factor of the directly proportional relation.

5 Conclusion and Future Scope

In this paper, an architecture for the ASTROSAT DQRs employing a web service based on Django web framework and RESTful API; Application Server consisting of a relational SQL database; A client-side user interface comprising of a mobile interface is propositioned that optimizes the storage and fetching mechanism of the DQRs. It enhances the speed of the system significantly. The load balancing and dynamic data fetching process employed in the Django and RESTful API allows the mobile application to maintain a lightweight architecture. It also makes the data considerably ubiquitous compared to the existing system owing to the mobile user interface it provides to its users. These implementations have resulted in establishment of a highly scalable and robust system architecture.

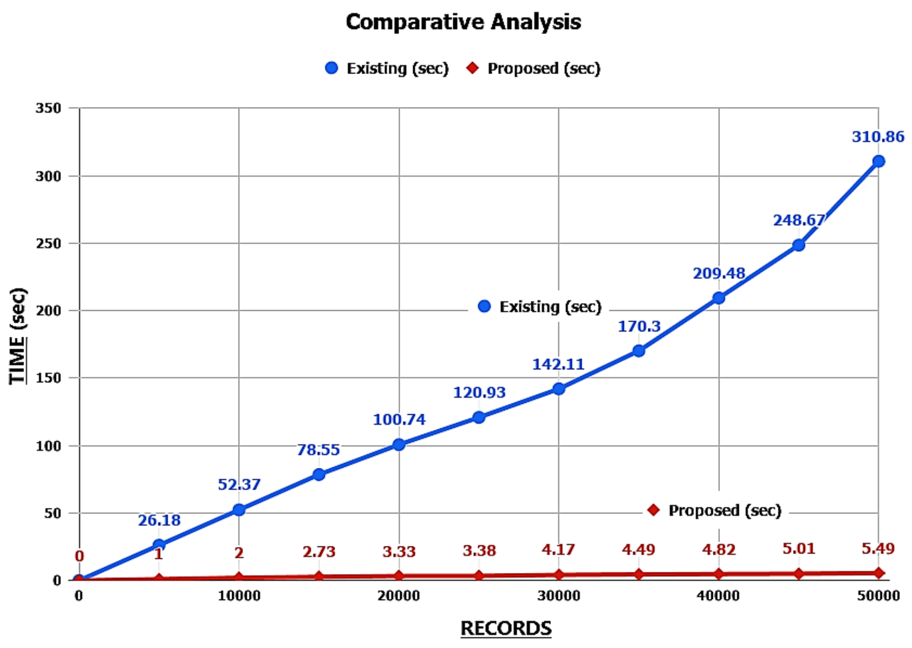

Fig. 2. Comparative Analysis of Existing System and Proposed System.

References

1. J. C. French and A. L. Powell, ``Metrics for evaluating database selection techniques" Proceedings. Tenth International Workshop on Database and Expert Systems Applications. DEXA 99, Florence,Italy, 1999, pp. 726-730.
2. R. Valarezo and T. Guarda, ``Comparative analysis of the laravel and codeigniter frameworks: For the implementation of the management system of merit and opposition competitions in the State University Península de Santa Elena," 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), Caceres, 2018, pp. 1-6.
3. Huijie Su, Bo Cheng, Tong Wu and Xiaofeng Li, ``Mashup service release based on SOAP and REST," Proceedings of 2011 International Conference on Computer Science and Network Technology, Harbin, 2011, pp. 1091-1095
4. Y. Shulin and H. Jieping, ``Research and implementation of Web Services in Android network communication framework Volley," 2014 11th International Conference on Service Systems and Service Management (ICSSSM), Beijing, 2014, pp. 1-3.
5. M. M. Patil, A. Hanni, C. H. Tejeshwar and P. Patil, "A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retrieval operations using a web/android application to explore load balancing — Sharding in MongoDB and its

advantages," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, 2017, pp. 325-330.

6. C. Győrödi, R. Győrödi, G. Pecherle and A. Olah, "A comparative study: MongoDB vs. MySQL," 2015 13th International Conference on Engineering of Modern Electric Systems (EMES), Oradea, 2015, pp. 1-6.
7. S. Rautmare and D. M. Bhalerao, "MySQL and NoSQL database comparison for IoT application," 2016 IEEE International Conference on Advances in Computer Applications (ICACA), Coimbatore, 2016, pp. 235-238.
8. Plekhanova, Julia. "Evaluating web development frameworks: Django, Ruby on Rails and CakePHP." Institute for Business and Information Technology (2009).
9. Forcier, J., Bissex, P. and Chun, W.J., 2008. Python web development with Django. Addison-Wesley Professional.
10. Chou, J., Chen, L., Ding, H., Tu, J., \& Xu, B. (2013). A Method of Optimizing Django Based on Greedy Strategy. 2013 10th Web Information System and Application Conference.
11. Rubio D. (2017) REST Services with Django. In: Beginning Django. Apress, Berkeley, CA
12. L. Li and W. Chou, "Designing Large Scale REST APIs Based on REST Chart," 2015 IEEE International Conference on Web Services, New York, NY, 2015, pp. 631-638.
13. L. Li, W. Chou, W. Zhou and M. Luo, "Design Patterns and Extensibility of REST API for Networking Applications," in IEEE Transactions on Network and Service Management, vol. 13, no. 1, pp. 154-167, March 2016.
14. M. Lachgar, H. Benouda and S. Elfirdoussi, "Android REST APIs: Volley vs Retrofit," 2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), Rabat, Morocco, 2018, pp. 1-6.
15. Y. Shulin and H. Jieping, "Research and implementation of Web Services in Android network communication framework Volley," 2014 11th International Conference on Service Systems and Service Management (ICSSSM), Beijing, 2014, pp. 1-3.