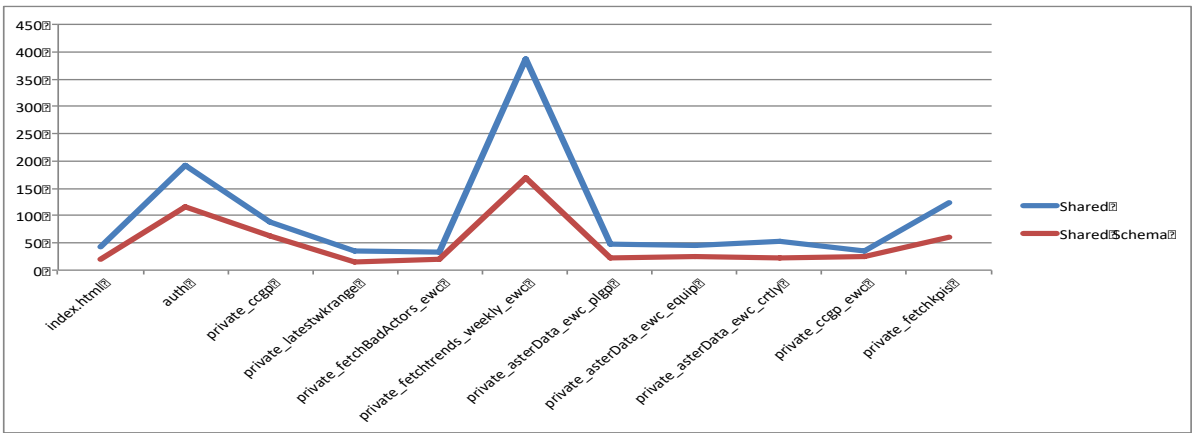

Figure 5. Latency across all endpoints

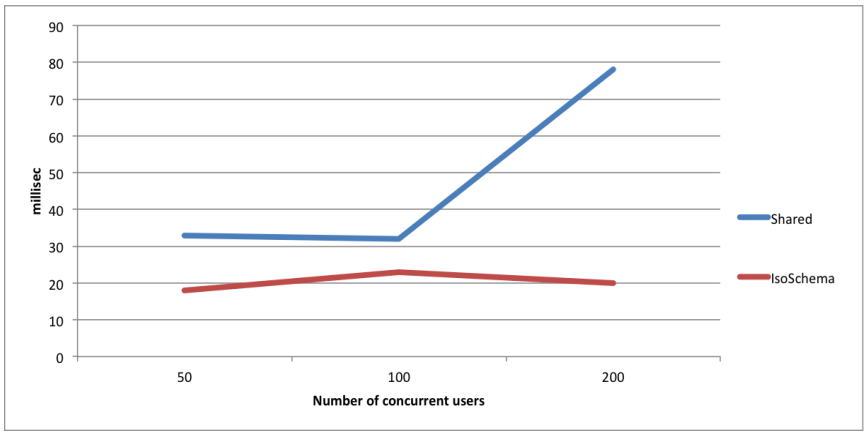

Figure. 6. Performance latency below 200 Users

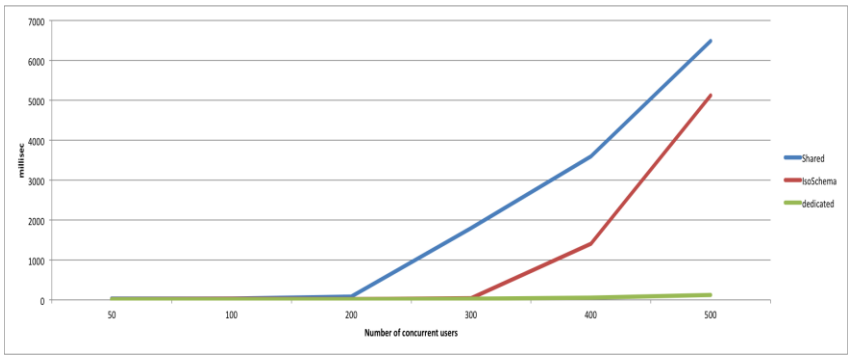

Figure 7. Performance latency above 200 Concurrent users

These tests were carried out to perform the base test on the restful endpoints provided by our case study application (AimHi). One variable is varied while the other was held constant to further understand how this variable affects the behaviour of the system. The number of tenants used in this test is three and the number of restful endpoints tested is ten. The test only covered a read operation both at the application level and data level, other types of operations such as delete, write /put will be carried out in our further research which is not covered in this paper.

5. Results

As stated in the last section, the application is configured to handle three tenants with varying number of concurrent users targeting various restful endpoints. The first set of tests for three models were performed using the same number of endpoints and dataset. Please see the results as shown in Figure 5.
The diagram above shows comparison between the shared component, isolated schema and dedicated component. The result shows that across all the endpoints tested, the shared model experience high latency compared to the isolated schema and dedicated. In some endpoints where calculations over large dataset are required, for example weekly fetch trends, the difference in latency is in magnitude of 2. However, the behaviour of the three models are consistent across all the endpoints as seen in the Figure 5.

The dedicated latency was lower compared to isolated schema model, the difference is not noticeable in this test however as the number of tenants and processing increases, dedicated will tend to perform better in terms of latency because of dedicated resources without any interference from other tenants. Figure 6 shows tests across the same number of endpoints using the three models when the user base is varied.

The figure shows up to 200 concurrent users, the dedicated and isolated schema performed reasonable well compared to shared component. This might be due to the records/rows selection process at the data layer when retrieving records belonging to a tenant or context switch between tenants in the shared component. However, there is a big leap in the latency when the number of user base hit 200 for the shared model. This occurs as more requests were queued and this causes delay in the processing.

6. Discussion

Another interesting result emerged when the three models were processed over increasing number of users as seen in figure 7. It was observed that although shared model is resource efficient, it reaches is maximum capacity around 200 concurrent users while isolated shared component reaches its maximum capacity at about 300 concurrent users, and dedicated was able to process the same dataset without any delay or errors. As the number of concurrent users increases in shared and isolated, there is exponential increase in latency across all endpoints.

Our presumption about the exponential increase in the two models might be as a result of saturated connection pooling size, which was set to 2000 and has to be shared among many tenants. Please note in this test, all form of data caching was disabled in order to show the models behaviour with any enhance algorithms.

Our empirical results are in consonant with the assertion made by [12] that there is a specific fraction of concurrent users a system can handle before the system starts queuing request or, in extreme cases, request can be dropped. Universal Scalability law introduced by [9] and Amdahl's law introduced by [6] both detailed functions to determine capacity of a system in terms of user load. [12] Asserted that resident time (latency time) of a system will start growing exponentially when the capacity is less than the concurrent users it can handle which is what is observed in both cases of multi- tenancy models experimented.

7. Conclusion

This research work systematically evaluates the performance of different multi-tenancy models at the data tier layer. Three (shared, isolated schema, and dedicated component) types of multi-tenancy models were presented. A case study approach was adopted to examine the performance of two the listed multi-tenancy models. The empirical experiments compared shared model against isolated schema using the same case study scenario and under the same circumstance. Our results show that the performance in terms of latency of isolated schema is better than the shared model. However, both models reach their full capacity at different concurrent number of users.

Shared model reaches maximum capacity under a lower concurrent user loads compared to isolated schema model. However when designing SaaS application such as relational database service where the number of tenants are considerable large, the use of partial isolated schema that creates a meta table that describe each tenant data structure and store shared data together for all the tenants data might be best options for ease of maintainability and economic standpoint. In conclusion, the number of concurrent user loads, the size of tenants, the duration of queries performed and the size of data store should be critical element when choosing which multi-tenancy model to adopt when designing an application. Our research has shown insight how the performance of

SaaS application can be impacted by the choice of the multi-tenancy model employed at the data tier layer. Our future works will explore how to best configure a multi-tenant application using Meta data pattern such as to increase isolation and customization.

8. Future Works

This research work has open up further works to better capture the performance of multi-tenancy under different scenario. Our future works will look into performance and interference that may occur when writing and deleting tenant records in a shared model. Another interesting area is hierarchical multi-tenancy; many organisations are structured in hierarchical way with various business rules and data governance. Our next work will study how multi-tenancy can implemented in this kind of organisation and provide challenges when developing hierarchical multi-tenant application. This will be extended into containerized technologies for portability across various platforms.

9. References

[1] Mohemed Almorsy and John Grundy. "SMURF: Supporting Multi-tenancy Using Re-Aspect Framework". In: 17th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2012) (2012).

[2] Michael Armbrust et al. Above the Clouds: A Berkeley View of Cloud Computing. Tech. rep. 2009.

[3] Cor-Paul Bezemer and Andy Zaidman. "Multi-Tenant SaaS Applications: Maintenance Dream or Nightmare?" In: In Proceeding of 4th International Join ERCIM/IWPSE Symposium on Software Evolution (IWPSE-EVOL) (2010), pp. 88–92.

[4] Ngo Huy Bien and Tran Dan Thu. "Hierarchical Multi- Tenant Pattern". In: 2014 International Conference on Computing, Management and Telecommunications (ComMan Tel) (2014), pp. 88–92.

[5] Christoph Alexander Fehling et al. Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud. The Science of Microfabrication. Springer, 2014.

[6] Eduardo B. Fernandez. "Patterns for operating system access control". In: In Proc. of PLoP 2002 (2002).

[7] Amdahl G.M. "Validity of the single processor approach to achieving large scale computing capabilities". In: In American Federation of Information Processing societies: Proceedings of the AFIP '67 Spring joint Computer Conference (1967).

[8] Chang Jie Guo et al. "A framework for native multitenancy application development and management". In: In Proc. Int. Conf. on E-Commerce Technology (CEC) and Int. Conf. on Enterprise Computing (2007), pp. 551 –558.

[9] Changjie Guo et al. "A Framework for Native Multi- Tenancy Application Development and Management." In: CEC/EEE. IEEE Computer Society, 2007, pp. 551– 558. ISBN: 0-7695-2913-5. URL: http://dblp.uni- trier. de/db/conf/wecwis/cec2007.html#GuoSHWG07.

[10] Rouven krebs, Alexander Wert, and Samuel Kounev. "Multi-tenancy performance benchmark for web application platforms". In: ICWE 13 Proceedings of the 13th International conference on web Engineering 7977 (3 2013), pp. 424–438.

[11] Guthaus M.R et al. "Mibench: a free, commercially representative embedded benchmark suite". In: In Proceeding of International Workshop on Workload Characterization (2001), pp. 3–14.

[12] Laud Charles Ochei, Andrei Petrovski, and Julian M. Bass. "Degree of Multitenancy Isolation for Cloudhosted Software Services: Synthesis of findings from three Case Studies". In: International Journal of Intelligent Computing Research (IJIR) 6 (3 2015).

[13] Laud Charles Ochei, Andrei Petrovski, and Julian M. Bass. "Evaluating degrees of tenant isolation in multitenancy patterns: a case study of cloud-hosted version control system (VCS)". In: International conference on information society (i-society) (2015), pp. 59–66.

[14] Davy Preuveneers et al. "Systemic scalability assessment for feature oriented multi-tenant services". In: The Journal of Systems and Software 116 (2016), pp. 162– 176.

[15] Antonio Rico et al. "Extending multi-tenant architectures: a database model for a multi-target support in SaaS applications". In: Enterprise Information Systems 10.4 (2016), pp. 400–421.

[16] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. "A Taxonomy and Survey of Cloud Computing Systems". In: Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC. NCM '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 44–51. ISBN: 978-0-7695-3769-6. DOI: 10. 1109 / NCM.2009.218. URL: http://dx.doi.org/10.1109/NCM. 2009.218.

[17] SalesForce.com. The Force.com Multitenant Architecture Understanding the Design of Salesforce.com Internet Application Development Platform. White Paper. Salesforce.com, 2001.

[18] Mark Turner, David Budgen, and Pearl Brereton. “Turning Software into a Service”. In: Computer 36.10 (Oct. 2003), pp. 38–44. ISSN: 0018-9162. DOI: 10.1109/MC. 2003.1236470. URL: http://dx.doi.org/10.1109/MC. 2003.1236470.

[19] Stefan Walraven, Eddy Truyen, and Wouter Joosen. “A Middleware Layer for Flexible and Cost-efficient Multi-tenant Applications”. In: Proceedings of the 12th ACM/IFIP/USENIX International Conference on Middleware. Middleware’11. Lisbon, Portugal: Springer- Verlag, 2011, pp. 370–389. ISBN: 978-3-642-25820-6. DOI: 10. 1007 / 978 - 3 - 642 - 25821 - 3 19. URL: http: //dx.doi.org/10.1007/978-3-642-25821-3 19.

[20] R.K. Yin. “Case Study Research: Design and Methods”. In: 3rd edition. SAGE Publications (2003).

