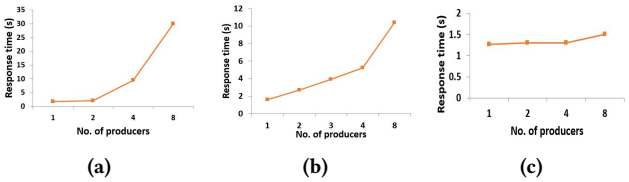

Figure 6: Response time of recommendation system using instacart benchmark on three different instances with (a) low network band width (5Gb/s) (b) low to mid network bandwidth (10 Gb/s) (c) High network bandwidth (25Gb/s)

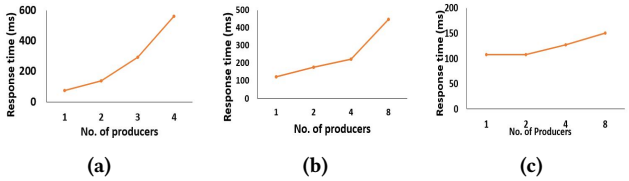

Figure 7: Response time of recommendation system using PAKDD benchmark on three different instances with (a) low network band width (5Gb/s) (b) low to mid network bandwidth 10 Gb/s) (c) High network bandwidth (25Gb/s)

the effect of instance configuration on performance and cost of recommendation system. In this section we present results of some the experiments.

Figures 3a and 3b show throughput for on-premise and sagemaker based implementation of iPrescribe using instacart and PAKDD benchmarks respectively in batch mode. We observed better throughput with AWS SageMaker which further improved as the number of active cores and workload is increased. Higher throughput is attributed to better optimizations and scaling available in the SageMaker.

In our next experiment we used m-family instances of AWS sagemake. We first ran our benchmarks on one ml.m5.2xlarge (8 cores) followed by two instances of ml.m5.xlarge (4 cores each) instance and lastly four ml.m5.large (2 cores each) instances. In these three runs total number of cores used (8 cores) and total cost (USD) is same but number and type of instances is different. As shown in Figure 4, throughput of the recommendation system increases as we increase the number of instances while keeping the total number of cores same. Maximum throughput was observed with 4 instances of ml.m5.large. This is due to the fact that size of mini batch in SageMaker that can be allocated to an instance at a time is limited by parallelism and size of records. Hence more instances result in more number of mini batches executed in parallel on instances available for inference.

Figures 5a and 5b show throughput and cost per inference of instacart benchmark on five SageMaker instances differing in number of cores, available memory and their usage cost. As shown in Figure 5b, cost per inference increases significantly by using an instance with large number of cores and higher costs but relatively smaller increase in the throughput is observed for the same instance as shown in Figure 5a. Hence prompting for a trade-off in cost and throughput.

Figures 6 and 7 show the effect of available network bandwidth on response time of benchmarks in real time. We used three types of AWS instances with different network bandwidths. We see a significant improvement in the response time by choosing instances with larger network bandwidth. For a high available network bandwidth we see that response time is almost constant even with increasing workload. We can conclude that not only the cores available but also network bandwidth affect the response time of the recommendation system.

8 CONCLUSION

We have successfully migrated an in-house recommendation system called iPrescribe from on-premise deployment to AWS cloud using automated ML workflow called SageMaker. In this work, we have studied performance metrics of iPrescribe including throughput, response time and cost per inference. Based on experimental results, we can conclude that by using cloud resources judiciously and appropriately, we can achieve our objective of better performance and cost-effective deployment of a recommendation system.

REFERENCES

[1] 2017. KaggleInstacartChallenge. https://www.kaggle.com/c/instacart-marketbasket-analysis.
[2] 2017. KaggleInstacartChallenge. http://www.recobell.com/rb/main.php?menu=pakdd2017..
[3] https://mxnet.apache.org/, 2019. [Online;Accessed 01 January 2019].
[4] nginx[enginex].http://nginx.org/en/, 2019. [Online;Accessed 25 December2019].
[5] Tensorflow serving. https://www.tensorflow.org/, 2019. [Online;Accessed 20 January 2019].
[6] Uber michelangelo. https://eng.uber.com/michelangelo/, 2019. [Online;Accessed 20 January 2019].
[7] XGBoost - Extreme Gradient Boosting. https://xgboost.readthedocs.io/en/latest/tutorials/model.html, 2019. [Online;Accessed 25 December2019].
[8] Crankshaw, D., Wang, X., Zhou, G., Franklin, M. J., Gonzalez, J. E., and Stoica, I. Clipper: A low-latency online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)* (Boston, MA, Mar. 2017), USENIX Association, pp. 613–627.
[9] Deak, R. M., and Morra, J. H. Aloha: A machine learning framework for engineers. In *Proceedings of the SysML Conference* (2018).
[10] Gaille, B. Average Web Page Load Times By Industry. http://www.byreputation.com/Average-Web-Page-Load-Times_a/452.htm, 2015. [Online;Accessed 28 June 2015].
[11] Ge, W. A continuous dataflow pipeline for low latency recommendations. Master's thesis, KTH, School of Information and Communication Technology (ICT), 2016.
[12] Hazelwood, K., Bird, S., Brooks, D., Chintala, S., Diril, U., Dzhulgakov, D., Fawzy, M., Jia, B., Jia, Y., Kalro, A., et al. Applied machine learning at facebook: A datacenter infrastructure perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (2018), IEEE, pp. 620–629.
[13] Modi, A. N., Koo, C. Y., Foo, C. Y., Mewald, C., Baylor, D. M., Breck, E., Cheng, H.-T., Wilkiewicz, J., Koc, L., Lew, L., Zinkevich, M. A., Wicke, M., Ispir, M., Polyzotis, N., Fiedel, N., Haykal, S. E., Whang, S., Roy, S., Ramesh, S., Jain, V., Zhang, X., and Haque, Z. Tfx: A tensorflow-based production-scale machine learning platform. In *KDD 2017* (2017).
[14] Singhal, R., Shroff, G., Kumar, M., Choudhury, S. R., Kadarkar, S., Virk, R., Verma, S., and Tewari, V. Fast online'next best offers' using deep learning. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data* (2019), ACM, pp. 217–223.
[15] Yun, J.-M., He, Y., Elnikety, S., and Ren, S. Optimal aggregation policy for reducing tail latency of web search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2015), SIGIR '15, Association for Computing Machinery, p. 63–72.
[16] Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., et al. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull. 41*, 4 (2018), 39–45.
[17] Zhang, C., Yu, M., Wang, W., and Yan, F. Mark: Exploiting cloud services for cost-effective, slo-aware machine learning inference serving. In *2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19)* (2019).