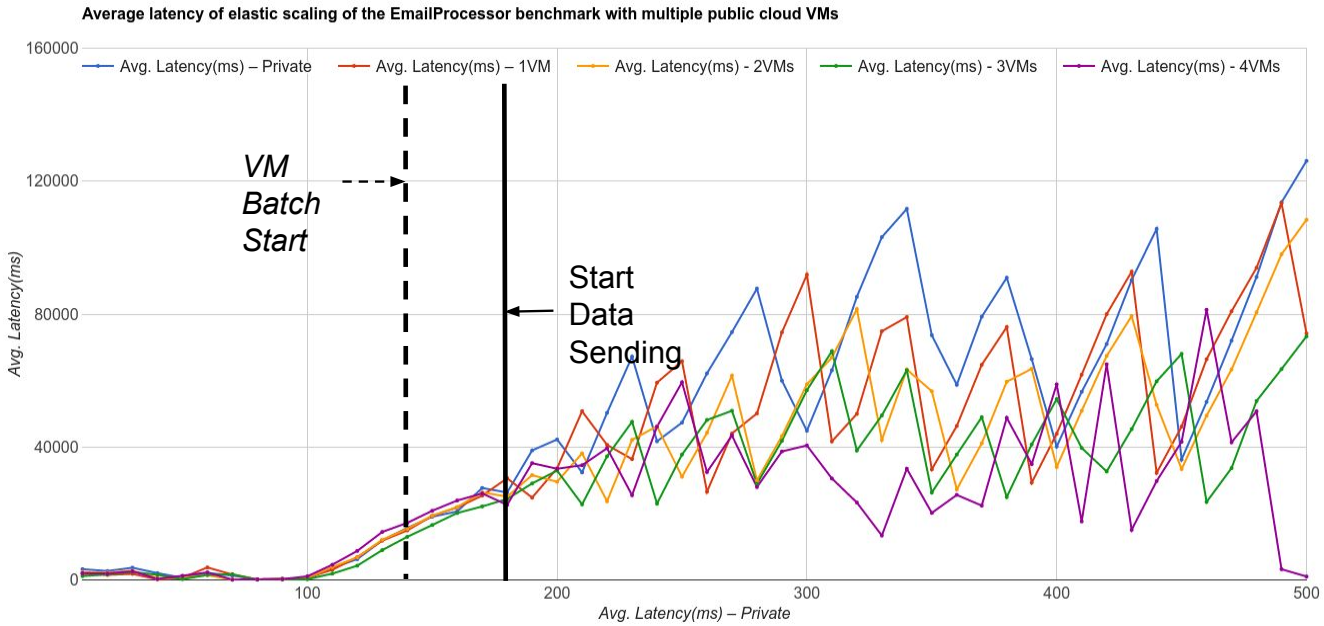

Figure 11: Average latency of elastic scaling of the EmailProcessor benchmark with multiple public cloud VMs

gle VM. The naive private cloud only average latency was 129.32 seconds. When a single VM on public cloud was started by the ESM the per event latency was further reduced by 53.33 seconds. This was a 15.1% average latency improvement compared to private cloud only deployment. However, when the ESM was configured to use maximum 2 VMs the overall average latency was reduced to 110.78 seconds which is a reduction of 18.54 seconds (22.8%) compared to the private cloud only deployment. Introduction of 3rd and 4th public cloud VMs resulted in further reductions in average latency by 36% and 47% compared to the private cloud only mode. Compared to the single public cloud VM, the 4 public cloud VMs further reduce the latency by 37.55%. These results indicate that ESM is capable of elastically scaling to multiple public cloud VMs and Scaling introduces significant performance gains. Note that the results shown on Figure 11 does not include the scenarios of VM shutdown since the motivation of the experiment was to indicate that the proposed ESM can reduce the overall latency further to considerable extent when additional VMs were added in the public cloud.

7.6 Discussion

The above five different experiments indicate the effectiveness of our elastic stream processing mechanism in lowering the average latency of data stream processing. In this paper our focus was on stream processing jobs which get deployed in public cloud completely during such elastic scaling scenario. Elastic scaling of a portion of the stream processing job is an important area. For example, the last three query operators of the EmailProcessor benchmark (Q_3, Q_4, and Q_5) can be transferred to public cloud while the first two operators (Q_1 and Q_2) could still remain in the private cloud.

We have demonstrated how effective it is to use compression along with elastic switching to handle out-of-order events which gets produced due to the use of public cloud. However, a limitation of our current implementation is that out-of-order event handling implementation could not turn off the VM once the switching to public cloud happens. This is due to the added latency of the buffer based re-ordering. We are working on to fix this issue.

We have used average latency as the user specified QoS metric in the current implementation because latency is one of the most important performance metrics in an elastic stream processing mechanism. However, we plan to investigate on use of other metrics such as throughput, system load average, etc. as parameters for switching process in future. We need to consider higher percentiles when there are a large number of outlier values in a dataset. We see two potential uses of higher percentile latencies in relation to ESM. First, for the switching function where the decision for switching needs to be taken. Because there are less number of events with outlier latencies in the private cloud mode of operation there is no significant difference in the use of average latency or a percentile such as 95th percentile for taking the switching decision. The switching decision is influenced only by the latency of the private cloud. Hence we opt to use average latency to make the switching decision. The second place where higher latencies would matter is measuring the latency benefit of elastic scaling. Switching to public cloud inherently introduces events with outlier latencies to the event stream. However, the benefit of ESM realizes when we consider the overall picture using the average latency. In hybrid cloud (public+private cloud) operation, although there are few outlier events with large latencies, their effect is subdued by the overall average latency reduction achieved by elastic switching. Hence, for measuring the elastic scaling benefit of using ESM we used the reduction of average latency but not the reduction of higher percentiles such as 95th percentile.

It should be noted that the ESM presented in this paper reorders events only at the output of the streaming pipeline. This assumes that the operations conducted within the elastic stream processing pipeline (i.e., the public cloud portion of the stream processing

pipeline) are not sensitive for the order of the events.

We used two application benchmarks which we believe are good example applications for stream processing. However, there are other different types of applications which involve complex operations which we plan to conduct in future. Event pattern matching, event sequence matching, sliding window operations, etc. are some examples for such applications.

8. CONCLUSIONS

In this paper we present an elastic switching mechanism (ESM) for elastic scaling of data stream processing systems. ESM leverages public cloud to augment a private cloud when the private cloud is overloaded. We implement the proposed approach on WSO2 Complex Event Processsor (WSO2 CEP). We have tested the feasibility of our approach by using two application benchmarks (i.e., queries) called EmailProcessor and SNB2016. We observe that in both data switching and query switching elastic scaling scenarios our elastic switching mechanism provides performance benefits in terms of latency reductions. In the case of the data switching scenario with single public cloud VM instance we obtained 16.7% improvement of average latency compared to private cloud only operation. The proposed compressed stream processing approach achieves 8.86% performance gain compared to naive elastic switching. In another experiment which involves query switching we obtained 7% improvement of overall average latency. Furthermore, we demonstrated that our data field compression based ESM could effectively produce lesser out-of-order events by 7% compared to a scheme which does not involve data stream compression. Moreover, when presented the option of scaling EmailProcessor with four public cloud VMs ESM further reduced the average latency by 37.55% compared to the single public cloud VM. These performance figures indicate that our elastic scaling mechanism can effectively reduce the average elapsed time spent on stream data processing.

Currently we do switch entire query to public cloud in the case of query switching. In future we plan to migrate only portions of the queries which will lead for better control of the elastic switching process. We also plan to investigate and implement Fully Homomorphic Encryption based ESM which will be useful for implementing stream processing applications which involve sensitive data such as processing health records. Furthermore, we plan to add microbenchmarks for different components in the ESM as well as use more elaborate performance models which may consider multiple aspects such as immediate saturation of the latencies. While a network emulator has been used to simulate latency between two distant data centers, we plan to conduct experiments in real cloud environments in future.

9. REFERENCES

[1] M. Blount, M. Ebling, J. Eklund, A. James, C. McGregor, N. Percival, K. Smith, and D. Sow. Real-time analysis for intensive care: Development and deployment of the artemis analytic system. *Engineering in Medicine and Biology Magazine, IEEE*, 29(2):110–118, March 2010.

[2] J. Cervino, E. Kalyvianaki, J. Salvachua, and P. Pietzuch. Adaptive provisioning of stream processing systems in the cloud. In *Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on*, pages 295–301, April 2012.

[3] R. Cocci, T. Tran, Y. Diao, and P. Shenoy. Efficient data interpretation and compression over rfid streams. In *2008 IEEE 24th International Conference on Data Engineering*, pages 1445–1447, April 2008.

[4] A. Cuzzocrea and S. Chakravarthy. Event-based lossy compression for effective and efficient {OLAP} over data streams. *Data & Knowledge Engineering*, 69(7):678 – 708, 2010. Advanced Knowledge-based Systems.

[5] Datapath. Datapath.io. URL: http://console.datapath.io/map, 2016.

[6] M. Dayarathna and T. Suzumura. Automatic optimization of stream programs via source program operator graph transformations. *Distributed and Parallel Databases*, 31(4):543–599, 2013.

[7] M. Dayarathna and T. Suzumura. *A Mechanism for Stream Program Performance Recovery in Resource Limited Compute Clusters*, pages 164–178. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[8] N. Dindar, Ã. Balkesen, K. Kromwijk, and N. Tatbul. Event processing support for cross-reality environments. *IEEE Pervasive Computing*, 8(3):34–41, July 2009.

[9] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter. *Cloud Computing Fundamentals*, pages 21–78. Springer Vienna, Vienna, 2014.

[10] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[11] S. Halevi and V. Shoup. *Algorithms in HElib*, pages 554–571. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[12] J. Hazra, K. Das, D. P. Seetharam, and A. Singhee. Stream computing based synchrophasor application for power grids. In *Proceedings of the First International Workshop on High Performance Computing, Networking and Analytics for the Power Grid*, HiPCNA-PG '11, pages 43–50, New York, NY, USA, 2011. ACM.

[13] S. Hemminger. Network emulation with netem. 2005.

[14] W. Hummer, B. Satzger, and S. Dustdar. Elastic stream processing in the cloud. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 3(5):333–345, 2013.

[15] T. Hunter, T. Das, M. Zaharia, P. Abbeel, and A. Bayen. Large-scale estimation in cyberphysical systems using streaming data: A case study with arterial traffic estimation. *Automation Science and Engineering, IEEE Transactions on*, 10(4):884–898, Oct 2013.

[16] S. Jayasekara, S. Perera, M. Dayarathna, and S. Suhothayan. Continuous analytics on geospatial data streams with wso2 complex event processor. In *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems*, DEBS '15, pages 277–284, New York, NY, USA, 2015. ACM.

[17] M. Jayasinghe, A. Jayawardena, B. Rupasinghe, M. Dayarathna, S. Perera, S. Suhothayan, and I. Perera. Continuous analytics on graph data streams using wso2 complex event processor. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, DEBS '16, pages 301–308, New York, NY, USA, 2016. ACM.

[18] S. R. Jeffery, M. J. Franklin, and M. Garofalakis. An adaptive rfid middleware for supporting metaphysical data independence. *The VLDB Journal*, 17(2):265–289, 2008.

[19] W. Kleiminger, E. Kalyvianaki, and P. Pietzuch. Balancing load in stream processing with the cloud. In *Data Engineering Workshops (ICDEW), 2011 IEEE 27th International Conference on*, pages 16–21, April 2011.

[20] B. Klimt and Y. Yang. Introducing the enron corpus. In *CEAS 2004 - First Conference on Email and Anti-Spam, July 30-31, 2004, Mountain View, California, USA*, 2004.

[21] S. Loesing, M. Hentschel, T. Kraska, and D. Kossmann. Stormy: An elastic and highly available streaming service in the cloud. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, EDBT-ICDT '12, pages 55–60, New York, NY, USA, 2012. ACM.

[22] C. Mutschler and M. Philippsen. Distributed low-latency out-of-order event processing for high data rate sensor streams. In *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 1133–1144, May 2013.

[23] Z. Nabi, E. Bouillet, A. Bainbridge, and C. Thomas. Of streams and storms. *IBM White Paper*, 2014.

[24] Y. Nie, R. Cocci, Z. Cao, Y. Diao, and P. Shenoy. Spire: Efficient data inference and compression over rfid streams. *IEEE Transactions on Knowledge and Data Engineering*, 24(1):141–155, Jan 2012.

[25] A. Page, O. Kocabas, S. Ames, M. Venkitasubramaniam, and T. Soyata. Cloud-based secure health monitoring: Optimizing fully-homomorphic encryption for streaming algorithms. In *2014 IEEE Globecom Workshops (GC Wkshps)*, pages 48–52, Dec 2014.

[26] B. Theeten, I. Bedini, P. Cogan, A. Sala, and T. Cucinotta. Towards the optimization of a parallel streaming engine for telco applications. *Bell Labs Technical Journal*, 18(4):181–197, 2014.

[27] M. Thompson, D. Farley, M. Barker, P. Gee, and A. Stewart. High performance alternative to bounded queues for exchanging data between concurrent threads. *technical paper, LMAX Exchange*, 2011.

[28] WSO2. Wso2 complex event processor. URL: http://wso2.com/products/complex-event-processor/, 2016.