not need *a priori* knowledge of events or their relative importance; all insights are provided using black-box model comparisons. *Attendees will pose as analysts using Sentinel* to investigate performance issues by comparing PostgreSQL and benchmark client behaviour in these scenarios to that of a known good (baseline) configuration (TPC-W *ordering mix*). Two such scenarios are described next. A video of these demonstration scenarios is available online.

### 3.1 Lock Contention

Our first demonstration provides an overview of Sentinel's features and shows how Sentinel can be used to detect and investigate the common case of database lock contention. We induce lock contention by inserting a short `sleep()` in the BuyConfirm transaction while holding database locks. We compare this poorly-performing (current) workload's behaviour to that of the baseline configuration to determine the source of the problem.

Using Sentinel's reports, we determine that there are far fewer benchmark client events of *all* types in the current workload compared to the baseline, indicating widespread performance degradation (Figure 3b). The largest CDF difference in client behaviour highlights a large increase in BuyConfirm transaction execution time. We observe that the largest differences in database event frequencies and CDFs of transition time pinpoint increased lock wait events and lock wait times.

Given this investigation, the analyst concludes that lock contention is the primary cause of the performance degradation in the current workload. This contention affects the BuyConfirm transaction while hampering the performance of the workload as a whole.

### 3.2 TPC-W Transaction Mixes

Workload access patterns often change [9]; our second demonstration investigates complex behavioural differences in such a case by supposing the transaction mix has changed from the baseline *ordering* mix to the TPC-W *browsing* mix. As analysts, we observe that performance has degraded and must determine what has changed.

The largest event differences in client behaviour pinpoint differences in transaction popularity between the two workloads. Read-only transactions like the NewProduct and BestSellers transactions are more popular in the browsing mix than in the ordering mix, enabling the analyst to conclude that the browsing mix is more read-heavy than the ordering mix. The transition time CDFs for these popular read-only transactions indicate that they take longer to execute than the update transactions, which affirms that these more expensive queries are impacting system throughput.

The largest behaviour differences in the database reveal that the browsing mix has lower query throughput than the baseline ordering mix despite the baseline workload exhibiting more lock contention and disk writes. Sentinel also highlights increased checkpoint throttling behaviour in the browsing mix as fewer dirty pages are written to disk. Thus, Sentinel indicates that the read-only queries in the current workload are more complex, resulting in lower overall performance. Given these insights, the analyst optimizes the highlighted, popular, long-running transactions in the current workload to improve performance.

## REFERENCES


[1] 2020. Chart.js Github. https://www.chartjs.org/. (2020).

[2] Peter Bailis, Edward Gan, et al. 2016. MacroBase: Prioritizing Attention in Fast Data. (2016), 541–556.

[3] Apache Software Foundation. 2020. Apache Log4j 2. https://logging.apache.org/log4j/2.x/.

[4] Max Franz et al. 2015. Cytoscape.js: a graph theory library for visualisation and analysis. *Bioinformatics* 32, 2 (09 2015), 309–311.

[5] Zhen Ming Jiang, Ahmed E. Hassan, Gilbert Hamann, and Parminder Flora. 2008. An Automated Approach for Abstracting Execution Logs to Execution Events. *J. Softw. Maint. Evol.* 20, 4 (July 2008), 249–267.

[6] Z. M. Jiang, A. E. Hassan, G. Hamann, and P. Flora. 2009. Automated performance analysis of load tests. In *2009 IEEE International Conference on Software Maintenance*. 125–134.

[7] D. A. Keim. 2002. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics* 8, 1 (Jan 2002), 1–8.

[8] Kamdem Kengne et al. 2013. Efficiently Rewriting Large Multimedia Application Execution Traces with Few Event Sequences. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. 1348–1356.

[9] Lin Ma et al. 2018. Query-based Workload Forecasting for Self-Driving Database Management Systems *(SIGMOD '18)*. ACM, New York, NY, USA, 631–645.

[10] Ashraf Mahgoub, Paul Wood, et al. 2017. Rafiki: A Middleware for Parameter Tuning of NoSQL Datastores for Dynamic Metagenomics Workloads *(Middleware '17)*. ACM, New York, NY, USA, 28–40.

[11] Karthik Nagaraj, Charles Killian, and Jennifer Neville. 2012. Structured Comparative Analysis of Systems Logs to Diagnose Performance Problems. *Nsdi* (2012), 353–366.

[12] Ofir Pele and Michael Werman. 2009. Fast and robust earth mover's distances. In *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 460–467.

[13] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 2000. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision* 40, 2 (01 Nov 2000), 99–121.

[14] TPC. 2000. TPC Benchmark W (Web Commerce). http://www.tpc.org/tpcw.

[15] Dana Van Aken, Andrew Pavlo, et al. 2017. Automatic Database Management System Tuning Through Large-scale Machine Learning *(SIGMOD '17)*. ACM, New York, NY, USA, 1009–1024.

[16] Dong Young Yoon, Ning Niu, and Barzan Mozafari. 2016. DBSherlock: A Performance Diagnostic Tool for Transactional Databases *(SIGMOD '16)*. ACM, New York, NY, USA, 1599–1614.

[17] Xu Zhao, Kirk Rodrigues, et al. 2017. Log20: Fully Automated Optimal Placement of Log Printing Statements Under Specified Overhead Threshold *(SOSP '17)*. ACM, New York, NY, USA, 565–581.