Jahresbericht 1.5.2015 bis 30.4.2016

für die

Alexander von Humboldt Stiftung

Prof. Dr. rer. pol.

Hans-Arno Jacobsen

Alexander von Humboldt-Professor

Technische Universität München
Fakultät für Informatik

Lehrstuhl für Anwendungs- und Middleware-Systeme/
Application and Middleware Systems Research Group

Boltzmannstraße 3
85748 Garching bei München
Tel. +49 89 289 19452
Fax +49 89 289 19466
jacobsen@in.tum.de
www.i13.in.tum.de

Inhaltsverzeichnis

0.0.1 Dynamic Scalable View Maintenance in Key-Value Stores

Motivation — During the 2014 Soccer World Cup, a peak of 580K Twitter messages per minute was recorded; the Facebook warehouse is reported to grow 600 TB each day. To process and store these volumes of data, a new breed of data bases, the distributed key-value stores (KV-stores) have been invented (such as Google's BigTable, Amazon's Dynamo, Yahoo's PNUTS or Apache's HBase). While KV-stores scale to an almost infinite number of nodes and, thereby, allow the creation and management of large storage spaces, the challenge becomes the evaluation and analysis of this data. New ways have to be found to efficiently compute and update the results of analytical expressions such that they are available on demand.

Problem statement — To sustain the high volume of processing, distributed KV-stores spread the data over multiple network nodes; they sacrifice standard database functionalities such as powerful query languages or transactional guarantees. Instead, they offer a simple API, comprising set, get and delete operations. While they provide efficient access to single row entries, the processing of more complex queries (e.g. aggregation or table join), require costly application-level operations; required query capabilities are missing in the systems. Further, the update of query results – as it is needed to obtain the most recent figures – forces repeated scans over large data chunks.

As data selection, data projection, aggregation, and join processing are so commonplace and freshness of analytical computations is a necessity today, we suggest introducing mechanisms for the materialization and maintenance of views (i.e. query results) in KV-stores. The challenge, then, is not the repeated scanning of base data any more; it is the creation and incremental update of view tables. These tables hold the result of any SQL-like query expression and can be accessed as any standard table in the system. Thus, materialized views can replace data-warehouse functionality and offer results with almost real-time currentness.

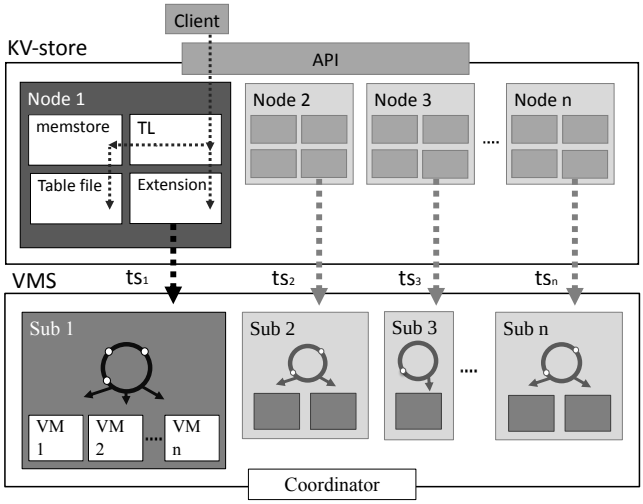

Abbildung 1: System Design

Approach — We describe and evaluate the design of a scalable view maintenance system (VMS) [1], that integrates with the widely used Apache HBase KV-store, as it is shown on top of Figure 0.0.1. As described before, the KV-store is an inherently distributed database. It runs a variable number of n data nodes. The incoming client request (i.e. the incoming data) are distributed over these nodes. Node 1 in the figure depicts a magnification of the processes that are triggered during one client operation on a data node: the operation is written to a transaction log (TL), then it is inserted into a memstore and finally it is pushed to the table file on disk. The extension component is an additional component. It reads all new client operations from the TL and forwards them to our VMS.

On the bottom of the figure, the VMS is depicted. Queries can be issued to the VMS in a SQL-like query language. The VMS, then, computes and maintains the result of the queries in view tables (which are simple tables in KV-store); its computations are based on a stream of client operations that it receives, in a distributed fashion (i.e. one stream ts per KV-node), from KV-store. The VMS, like KV-store, is a scalable system. A variable number of n sub systems, contain a variable number of view managers (VMs) – which are doing the actual work of computing and applying the client operations to the view tables.

The VMS offers an approach that scales in view update load and number of views maintained. The design does not interfere with read/write processing of tables in the KV-store, thus, leaving base table processing latencies unaffected. It integrates "naturally"with KV-store. Thus, our approach inherits the availability, reliability, and scalability properties of KV-store itself. View tables can be split across servers, distributed across the network, replicated across the underlying file system, and made available to increasing number of clients.

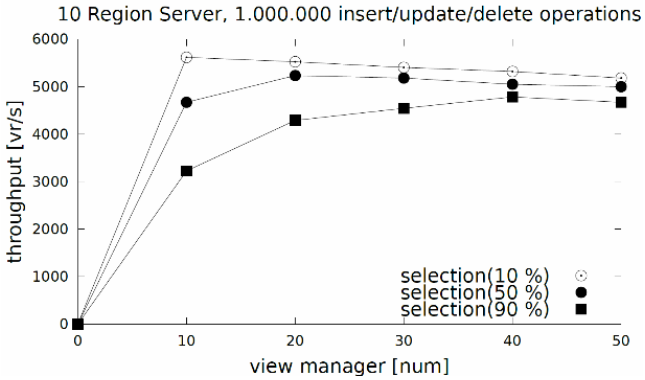

Abbildung 2: Scaling the number of VMs for selection views

Implementation and results — We have implemented the VMS as a separate framework and integrated it with Apache HBase (Version 0.98.1.6) [1]. The VMS has been evaluated on a cluster of 40 nodes (cf. Fig. 0.0.1 & 0.0.1). We demonstrated that it is possible to efficiently maintain the different view types (e.g., selection, projection, aggregation and join views), as well as combinations of complexer query constructions. The maintenance system is horizontally scalable and achieves a maximum throughput on our cluster of 6.000 view records

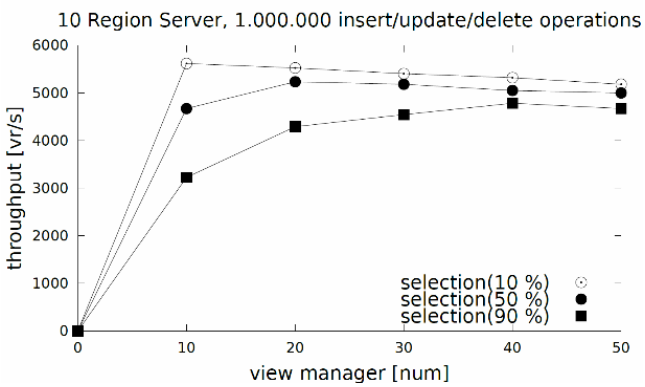

Abbildung 3: Scaling the number of VMs for aggregation views

per second. As the number of computed views grows, more view manager processes can be deployed for the VMS; the VMS itself adapts to varying load by performing an internal load balancing.

Next steps — As next steps, we plan to investigate how to efficiently manage multi-view processing, i.e., to share and reuse computations and reduce the overall maintenance cost for the concurrent materialization of many individual view expressions, which may even depend on one another. We build a model of n directed acyclic graphs (i.e. DAGs), each corresponding to the maintenance plan of one of our queries. Then, we run an optimization algorithm, based on a cost model, to determine a global maintenance plan; the algorithm strives for sharing a maximum of intermediate results and reduces the global processing effort, as well as the overall storage cost.

Also, we plan to investigate how to integrate our incremental view maintenance approach with batch processing to enable the creation of new view expressions on the fly, i.e., when existing base tables views depend on are already available.

Publications — Jan Adler, Martin Jergler, Hans-Arno Jacobsen **Dynamic Scalable view maintenance in KV-stores** *2016 USENIX Annual Technical Conference* (submitted for publication, March 2016).

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[1] J. Adler, M. Jergler, and H.-A. Jacobsen. Dynamic scalable view maintenance in kv-stores. In *Submitted to Proceedings of the 2016 USENIX Annual Technical Conference*, 2016.

[2] V. del Razo, T. Riihonen, F. Gregorio, S. Werner, and R. Wichman. Nonlinear amplifier distortion in cooperative amplify-and-forward ofdm systems. In *Proceedings of the 2009 IEEE Conference on Wireless Communications & Networking Conference*, WCNC'09, pages 352–356, Piscataway, NJ, USA, 2009. IEEE Press.

[3] M. Jergler, C. Doblander, M. Najafi, and H. Jacobsen. Grand challenge: real-time soccer analytics leveraging low-latency complex event processing. In *The 7th ACM International Conference on Distributed Event-Based Systems, DEBS '13, Arlington, TX, USA - June 29 - July 03, 2013*, pages 307–312, 2013.

[4] M. Jergler, M. Sadoghi, and H. Jacobsen. D2WORM: A management infrastructure for distributed data-centric workflows. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 1427–1432, 2015.

[5] M. Jergler, M. Sadoghi, and H.-A. Jacobsen. D2worm: A management infrastructure for distributed data-centric workflows. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1427–1432, New York, NY, USA, 2015. ACM.

[6] M. Najafi, M. Sadoghi, and H. Jacobsen. Configurable hardware-based streaming architecture using online programmable-blocks. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 819–830, 2015.

[7] J. Rivera, M. Jergler, A. Stoimenov, C. Goebel, and H.-A. Jacobsen. Using publish/subscribe middleware for distributed ev charging optimization. *Computer Science - Research and Development*, pages 1–8, 2014.

[8] S. Rusitschka, C. Doblander, C. Goebel, and H.-A. Jacobsen. Adaptive middleware for real-time prescriptive analytics in large scale power systems. In *Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference*, Middleware Industry '13, pages 5:1–5:6, New York, NY, USA, 2013. ACM.

[9] M. Sadoghi, M. Jergler, H. Jacobsen, R. Hull, and R. Vaculín. Safe distribution and parallel execution of data-centric workflows over the publish/subscribe abstraction. *IEEE Trans. Knowl. Data Eng.*, 27(10):2824–2838, 2015.

[10] A. Veit, C. Goebel, R. Tidke, C. Doblander, and H.-A. Jacobsen. Household electricity demand forecasting: Benchmarking state-of-the-art methods. In *Proceedings of the 5th International Conference on Future Energy Systems*, e-Energy '14, pages 233–234, New York, NY, USA, 2014. ACM.

[11] H. Ziekow, C. Goebel, J. Strüker, and H. Jacobsen. The potential of smart home sensors in forecasting household electricity demand. In *IEEE Fourth International Conference on Smart Grid Communications, SmartGridComm 2013, Vancouver, BC, Canada, October 21-24, 2013*, pages 229–234, 2013.