Eventual Consistency and Deterministic Dataflow Programming

A Case Study of Integrating Derflow with the Riak Data Store

Christopher Meiklejohn

Basho Technologies, Inc.

cmeiklejohn@basho.com

Abstract

Even with the upcoming 2.0 release, queryability of Riak [1], an open source distributed database from Basho Technologies, remains an area for improvement. As of this release, Riak provides three main mechanisms for executing queries across values stored in the database: secondary indexing (2i), a MapReduce-like [5] framework, and Yokozuna. However, all three have significant drawbacks in terms of scalability and flexibility.

Secondary indexing offers the ability to tag objects as they are written into the database with key-value pairs that can be used as the basis for queries. However, the entire set of tags needs to be specified every time the object is written, and tags are restricted to range and equalities over strings and integers. In addition, there is no mechanism for providing ad-hoc conjunctions or disjunctions.

Riak's MapReduce-like framework, provided through an application called $riak\_pipe$, provides the ability to do on-demand, scatter-gather queries, but requires re-evaluation of the whole input set even though there may be no changes for a phase, causing increased cluster load.

Finally, Yokozuna provides an abstraction over distributed Solr, but relies on a glue layer on top of Riak to interface with a JVM running on each node, executing Solr queries across the cluster. As of writing, it is still unclear how far this mechanism can be scaled, and what penalty exists at scale when moving data between the Java Virtual Machine and the Erlang runtime system.

Given these drawbacks, we have identified a series of desirable properties for a future query mechanism for Riak. Specifically, these are:

- The ability for a user to submit a computation to the Riak cluster, and have it performed in a highly-available, fault-tolerant manner across the entire cluster.
- An execution mechanism that can re-use and incrementally update partial results, thereby alleviating the need to re-execute the entire query across the cluster on repeated executions.
- A query mechanism that reduces harvest while maintaining yield [3] during failure conditions.

Recently, there has been a series of research efforts surrounding the use of bounded-join semilattices, a generalization of state-based conflict-free replicated data types (CRDTs) [9], as data structures in new programming models to provide deterministic execution in distributed scenarios. Two examples of this are LVars [8], providing deterministic execution across multiple threads in Haskell, and Bloom [4], which provides deterministic execution across multiple instances of the Ruby virtual machine. In both these cases, properties of the bounded-join semilattice, combined with monotone functions, assist in ensuring determinism, specifically handling cases of repeated updates and out-of-order updates.

More recently, as part of the SyncFree project in the European Seventh Framework Programme of which Basho is a participant, there has been further addition to these distributed deterministic programming models named Derflow [10]. Derflow provides a similar programming model, but is built using the Erlang-based, Dynamo-inspired [6], distributed systems toolkit, $riak\_core$. [2]

We explore the process of developing the reference implementation of Derflow while simultaneously integrating the reference implementation with Riak to provide a new prototype query mechanism. We expose the ability for users to submit deterministic computations to the Riak data store, which are executed as values are written, providing the user the ability to retrieve the computed results through a query API, similar to a materialized view mechanism as exposed by other commercial databases, such as CouchDB. [7]

Our integration exploits the following properties of Riak and Derflow:

- Computations and their results are partitioned and replicated along with their input data in the data store. This allows us to provide highly-available results, which sacrifice harvest during failure conditions. [1]
- The partial results of computations can be combined deterministically, given the merge properties of state-based conflict-free replicated data types.

The main contribution of this talk is an experience report from the Basho engineering team that details the following:

- Assisting in the design and development of the Derflow library on top of $riak\_core$.
- Adapting the research concept and reference implementation of Derflow into Riak.
- Contributing changes made to Derflow for use inside of Riak back to the reference implementation of Derflow.

Acknowledgments

This work was partially funded by the SyncFree project in the European Seventh Framework Programme (FP7/2007-2013) under Grant Agreement n^{o} 609551.

References

[1] Basho Technologies Inc. Riak source code repository. `http://github.com/basho/riak`, .

[2] Basho Technologies Inc. Riak core source code repository. `http://github.com/basho/riak_core`, .

[1] Specifically, as input values to the computation become unavailable, the computations that resulted in those inputs also become unavailable.

[3] E. A. Brewer. Lessons from giant-scale services. *IEEE Internet Computing*, 5(4):46–55, July 2001. ISSN 1089-7801. . URL http://dx.doi.org/10.1109/4236.939450.

[4] N. Conway, W. Marczak, P. Alvaro, J. M. Hellerstein, and D. Maier. Logic and lattices for distributed programming. Technical Report UCB/EECS-2012-167, EECS Department, University of California, Berkeley, Jun 2012. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-167.html.

[5] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008. ISSN 0001-0782. . URL http://doi.acm.org/10.1145/1327452.1327492.

[6] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 205–220, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-591-5. . URL http://doi.acm.org/10.1145/1294261.1294281.

[7] B. Holt. *Writing and Querying MapReduce Views in CouchDB*. O'Reilly Media, Inc., 1st edition, 2011. ISBN 1449303129, 9781449303129.

[8] L. Kuper and R. R. Newton. Lvars: Lattice-based data structures for deterministic parallelism. In *Proceedings of the 2Nd ACM SIGPLAN Workshop on Functional High-performance Computing*, FHPC '13, pages 71–84, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2381-9. . URL http://doi.acm.org/10.1145/2502323.2502326.

[9] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. A comprehensive study of Convergent and Commutative Replicated Data Types. Rapport de recherche RR-7506, INRIA, Jan. 2011. URL http://hal.inria.fr/inria-00555588.

[10] SyncFree. Derflow source code repository. http://github.com/syncfree/derflow.