PathFS: A Streaming File System for Edge Environments

www.cs.toronto.edu/~kianoosh/projects/pathfs

Progress Report

As described in the project proposal, the baseline version of PathFS was a mere client-side application of PathStore [1]. We implemented that application as our first step in the project. Since PathStore is built on top of Cassandra, its applications are CQL files defining the application schema. Therefore, in this client-side application, we introduced three CQL tables for storing the data. The corresponding tables are 'file,' 'directory,' and 'filedata.' The file and directory tables store the metadata of the corresponding entries, including the full path, name, parent directory, permissions, and the owner user. We used the md5 hashing of the entry's full path as the primary key as each entry has a unique full path. The filedata table, on the other hand, stores the various data chunks of a file. Since PathFS' primary motivation is to support streaming files, we store every streaming part as a separate row in that table. Even though these rows can be arbitrarily large, we ran an empirical study about the optimal file size and realized that 16KB file chunks are the most optimal for the underlying Cassandra replica. Moreover, we implemented various command-line interface utilities such as creating a file or directory, reading a file, and appending to a file.

Next, we implemented the proposed event-driven approach. Unlike the traditional periodic updates, we used the eager update approach so that the rows get propagated sooner. This change has been done to the PathStore core so that every incoming insert query can be propagated immediately. The eager update method entails upward and downward propagations. In the upward propagation algorithm, each node forwards the insert statement and its decoded values to its parent. The parent then executes the query on its own database replica, calls the upward propagation function on its parent and propagates the query down to its children (other than the original sender). The downward propagation algorithm is slightly more complicated as the nodes have to decide which children to send the data to. We used an existing infrastructure of PathStore called interests. Each interest is essentially a select query that indicates that a node has run the corresponding query, hence is propagated. Interests were used in PathStore to pull updates from the parents periodically. To implement the downward propagation algorithm, we kept track of the child nodes that request an interest, and then match the incoming insert query to the registered interests. Therefore, we could determine which child nodes to forward the query. The matching phase was done by applying the insert and the corresponding select statement on a small, temporary table, where the insert gets deleted soon.

We evaluated the event-driven system against the original PathStore implementation, as well as IPFS [2] as a real-world baseline. The results showed at least a 60% improvement in the latency as we replaced periodic updates with immediate propagation. Moreover, it showed a 6% improvement in the bandwidth consumption as it reduced the need for periodic pull queries.

Future plans

The multi-parent approach presented in the proposal no longer seems to be feasible. We realized that in most of the real-world use cases of this system, the depth of the tree is at most 3 or 4, while each node can have many direct children. So the topology is a shallow, wide tree, and the multi-parent system would not be a good fit. For the future, we plan to integrate the system with the FUSE [3] library and implement the remaining functionalities such as removals and symbolic links so that PathFS would be a full-fledged file system.

References

[1] S. H. Mortazavi *et al*, "Pathstore, a data storage layer for the edge," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services,* 2018, .

[2] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv Preprint arXiv:1407.3561,* 2014.

[3] B. K. R. Vangoor, V. Tarasov and E. Zadok, "To $\$FUSE$\$ or not to $\$FUSE$\$: Performance of user-space file systems," in *15th $\{$USENIX$\}$ Conference on File and Storage Technologies ($\{$FAST$\}$ 17),* 2017, .