Developing with HyperLogLog in an Active-Active database

HyperLogLog is an algorithm that addresses the count-distinct problem. To do this it approximates the numbers of items in a set. Determining the exact cardinality of a set requires memory according to the cardinality of the set. Because it estimates the cardinality by probability, the HyperLogLog algorithm can run with more reasonable memory requirements.

HyperLogLog in Redis

Open source Redis implements HyperLogLog (HLL) as a native data-structure. It supports adding elements (PFADD) to an HLL, counting elements (PFCOUNT) of HLLs, and merging of (PFMERGE) HLLs.

Here is an example of a simple write case:

<table>
<thead>
<tr>
<th>Time</th>
<th>Replica 1</th>
<th>Replica 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>PFADD hll x</td>
<td></td>
</tr>
<tr>
<td>t2</td>
<td></td>
<td>— sync —</td>
</tr>
<tr>
<td>t3</td>
<td></td>
<td>PFADD hll y</td>
</tr>
<tr>
<td>t4</td>
<td></td>
<td>— sync —</td>
</tr>
<tr>
<td>t5</td>
<td>PFCOUNT hll -&gt; 2</td>
<td>PFCOUNT hll -&gt; 2</td>
</tr>
</tbody>
</table>

Here is an example of a concurrent add case:

<table>
<thead>
<tr>
<th>Time</th>
<th>Replica 1</th>
<th>Replica 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>PFADD hll x</td>
<td>PFADD hll y</td>
</tr>
<tr>
<td>t2</td>
<td>PFCOUNT hll -&gt; 1</td>
<td>PFCOUNT hll -&gt; 1</td>
</tr>
<tr>
<td>t3</td>
<td></td>
<td>— sync —</td>
</tr>
<tr>
<td>t4</td>
<td>PFCOUNT hll -&gt; 2</td>
<td>PFCOUNT hll -&gt; 2</td>
</tr>
</tbody>
</table>

The DEL-wins approach

Other collections in the Redis-CRDT implementation use the observed remove method to resolve conflicts. The CRDT-HLL uses the DEL-wins method. If a DEL request is received at the same time as any other request (ADD/MERGE/EXPRIE) on the HLL-key the replicas consistently converge to delete key. In the observed remove method used by other collections (sets, lists, sorted-sets and hashes), only the replica that received the DEL request removes the elements, but elements added concurrently in other replicas exist in the consistently converged collection. We chose to use the DEL-wins method for the CRDT-HLL to maintain the original time and space complexity of the HLL in open source Redis.

Here is an example of a DEL-wins case:

| HLL | | Set |
|-----| |-----|
|     | |     |
| Time | Replica 1 | Replica 2 | | Time | Replica 1 | Replica 2 |
HLL in Active-Active databases versus HLL in Open Source Redis

In Active-Active databases, we implemented HLL within the CRDT on the basis of the Redis implementation with a few exceptions:

- Redis keeps the HLL data structure as an encoded string object such that you can potentially run any string request can on a key that contains an HLL. In CRDT, only get and set are supported for HLL.

- In CRDT, if you do SET on a key that contains a value encoded as an HLL, then the value will remain an HLL. If the value is not encoded as HLL, then it will be a register.

Updated: July 7, 2021