Performance analysis of joins in reactive queries

Mitar Milutinović (mitar@tnode.com), Amy Pavel (amypavel@berkeley.edu)

Motivation

NoSQL databases like MongoDB remove relations between documents and leave it to users to resolve relations on their own. Resolving relations on the client side results in round trip delays.

Modern web frameworks like Meteor provide an API to create reactive queries to the MongoDB: after initial query results, query is kept open and any update to the results set is streamed to the client. But if a query contains joins, constructing reactive queries by hand and resolving relations becomes hard. Moreover, consequence of multiple round trip delays accumulates.

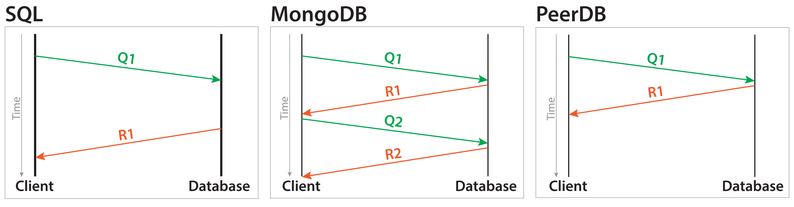
PeerDB offers a way to embed useful fields from related documents and keep them updated, reducing read time on queries with joins, while allowing easy construction of reactive queries as well.

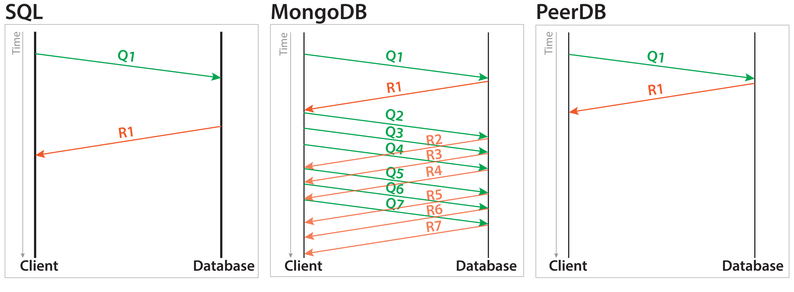
MongoDB

```
{
      "_id": "k7cgWtxQpPQ3gLgxa",
      "body": "I'm a post about rabbits!",
      "author": {
            "_id": "frqejWeGWjDTPMj7P"
      },
      "tags": [{
            "_id": "KMYNwr7TsZvEboXCw"
      }, {
            "_id": "tMgj8mF2zF3gjCftS"
            }]
}
```

MongoDB with PeerDB

```
{
      "_id": "k7cgWtxQpPQ3gLgxa",
      "body": "I'm a post about rabbits!",
      "author": {
            "_id": "frqejWeGWjDTPMj7P",
            "name": "Henry",
            "picture": "http://img.com/img.jpg"
      },
      "tags": [{
            "_id": "KMYNwr7TsZvEboXCw",
            "name": "animals"
      }, {
            "_id": "tMgj8mF2zF3gjCftS",
            "name": "information"
            }]
}
```

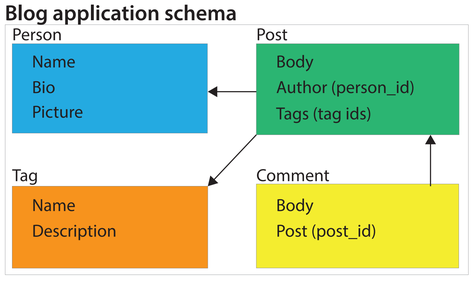




Goals

Evaluate PeerDB and recommend design guidelines for embedding fields. In particular, we will compare performance of the following variations using a blog application:

- PeerDB (high-level Meteor and low-level queries)
- MongoDB (high-level Meteor and low-level queries)
- Postgres (high-level Django and low-level queries)

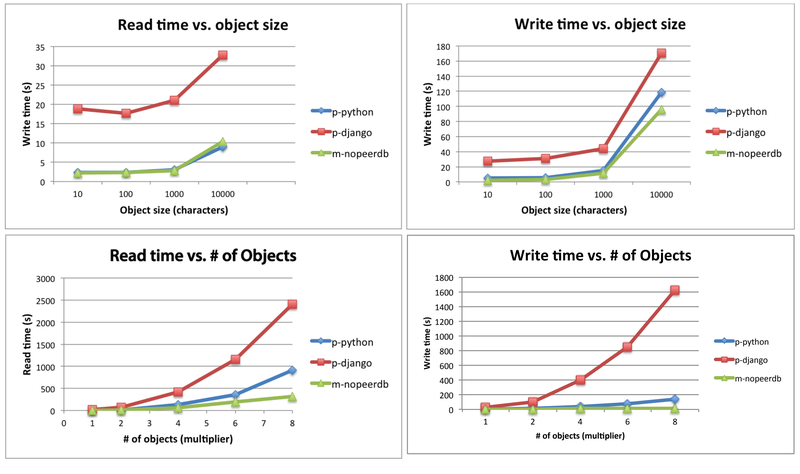


We will offer design guidelines on when to use PeerDB embedded fields versus other methods by varying these parameters:

- Size of fields in related objects
- Number of of objects
- Number of database instances

Comparison results

We found that high performance reads of PeerDB come at the expense of slow writes (figure to the right). Based on our evaluation we recommend using postgres for write-heavy applications.



Design guidelines

Based on our evaluation and use of PeerDB in PeerLibrary we came up with the following design guidelines for picking fields to embed. These design considerations can be automated or handled during schema specification.

- Fields to embed are small in size
- Values of all embedded fields are shared across all queries
- Reads on documents using embedded fields are much more common than writes

Reactive queries with joins

In addition to making static queries to evaluate PeerDB performance we evaluated reactive queries with joins as well.

Related work

- Materialized views
- Column stores and database cracking
- Customizing NoSQL databases

Github links

PeerDB: github.com/peerlibrary/meteor-peerdb
Benchmarking: github.com/mitar/peerdb-benchmark