MONGODB INDEX PERFORMANCE STUDY

1. COLLECTION “user”
   1. Username

The first field in which we study the possibility of indexing is the *username* one in the *user* collection. A username is a REQUIRED and UNIQUE field of each user, and it is his/her mnemonic id inside the application.

The field username is involved in the following queries:

W1-)Insert a new username at registration time of an arbitrary user

W2-)Remove a username when an admin delete’s a user from the system

R1-)Check uniqueness of a username at registration time

R2-)Check user’s credential at login time

R3-)Find a user by username when a new follow request is submitted

Considering these operations, we try to estimate reads and writes’ loads on the Database considering realistic hypothesis on the frequency of each query.

Assuming that a registered user will play the game for about 100 days before “getting bored”, we can state that the number of logins-per-day will be 100 times the number of registrations-per-day: this means that the queries R1+R2 are submitted 101 times more than query W1.

Moreover, we can assert that query W2 will be very rare, while R3 is a popular query among the network structure of the application, say 30 times the number of registered users: we find out that read operations on this field are about 130 times the number of write operations.

Now consider MongoDb performances with and without using an index on the *username* field, in a Database populated by 250k users.

db.user.find({username:”eee”}, {username:1}).explain(“executionStats”)

Immagine che contiene testo, monitor, screenshot, elettronico

Descrizione generata automaticamenteIn the picture on the right is reported the output of the query when we do not use an index. Execution time is huge due to the very high number of docs examined.

Immagine che contiene testo, screenshot, elettronico, computer

Descrizione generata automaticamenteOn the contrary when we index “username” field, the same query need an execution time almost 100 times lower, and of course thanks to the index, DBMS only need to examinate one document.

Considering the very high speed-up ratio of the indexing and the high frequency of this kind of queries w.r.t. the write operations (as explained before), a UNIQUE INDEX on *username* has been created

* 1. Country

As seen before, starting from the application queries we demonstrate the benefits of an index in the field *country.*

W1-)Insert the country data at registration time

W2-)Remove all the user’s data if a user is banned by an admin

W3-)Changing of settings after a user changes residence’s country

R1-)Rank all users by country

R2-)Rank countries with the highest logins-per-day ratios

Let x be the number of registrations-per-day (W1), w.r.t this number W2 and W3 are very rare operations. Indeed, even though we can expect mischievous behaviors from some user, the number of country changes will never be comparable with x.

On the other hand, in order to guarantee a read-your-own-write eventual consistency on ranking R1, this query is recomputed every time a user asks to see the ranking itself. Thus, since the gameplay is highly based on rankings, we can estimate that R1 frequency will be about 400x.

Furthermore we have to consider R2. Despite the fact that this query is executed just once per day (so frequency(R2)<<x), it is an asynchronous procedure sensitive to execution time since it needs to lock the entire collection, make it unavailable to users for a while.

As seen before, let us compare DBMS performances with and without a *country* index.

db.user.find({country:"Italy"}).explain("executionStats")

Immagine che contiene testo, monitor, screenshot, elettronico

Descrizione generata automaticamenteConsidering again about 250k users, without an index we need to scan the whole database, which means a medium-high execution time for each request

Immagine che contiene testo, screenshot, monitor, elettronico

Descrizione generata automaticamente

On the contrary, we have a very high increase of performances introducing and index on *country*.

To summarize, considering the difference in frequency between reads and writes and the high decrease of execution time, an index on *country* has been introduced