about GrapheneDb and alternatives

5 posts by 4 authors

Mustafa Duman Apr 24

Hello,

I wonder about your thoughts and experiences on hosting neo4j in production environment.

For GrapheneDb, the idea of making http calls for DB access sounds bad for me for performance. What do you think?

What do you think about self-hosting?

Thanks

Michael Hunger Apr 24

The Neo4j Server uses HTTP APIs as well, as you just send a query that is executed **within** the server and returns data the protocol doesn't matter so much.

Depends on your use-cases if that performance is enough for your SLAs or not.

There is also the embedded option and we'll be working on a binary remoting protocol too.

Michael

- show quoted text -
- show quoted text -

--

Hans Uhlig Apr 24

Any word when the binary protocol will be ready. I am looking to build a very very large graph that will be under realtime updating and dealing

with a stateless non websocket HttpAPI will be the death of the ETL.
- show quoted text -

Michael Hunger Apr 24

Hi Hans,

I can't give any dates right now there will be updates later this quarter.

What makes you say that? What are the numbers you're looking for?

I tested Neo4j updates via http and for instance created 100M nodes + 100M relationships concurrently and transactionally in 220 seconds via http & cypher. (1M tx with 100 each).

Cheers, Michael
- show quoted text -

Alberto Perdomo Apr 26

Hello Mustafa,

Neo4j has two modes: server and embedded, each with its own pros and cons.

In the embedded mode, the database sits together with your application. This has some upside (performance!!) but also some downside as well (maintainability, scalability as app and database are tied together).
In the server mode, for now the client makes HTTP calls, adding some network latency. This will change once the binary protocol is introduced.

As to wether to use GrapheneDB or not, to avoid HTTP calls, I think the question should rather be **if you should use server or embedded mode**. If you run in server mode, you'll always be doing HTTP calls, regardless of where it's hosted (GrapheneDB or on your own).

If you have a specific feature that is **performance-critical** (complex traversals or certain queries with very high throughput) I would encourage you to look at using server extensions.
We've seen plenty of customers succeed with this approach:

- Maintainability of the graph database is great because it's not tied to the application server. It can also be outsourced to 3rd party hosting services like GrapheneDB (custom extensions are supported in the Developer plan and higher).
- You can use **Cypher for most of your implementation**.
- You can **implement time-critical pieces on top of the Java API**using extensions, achieving almost as performant results as in embedded mode.

Once the binary protocol is released, you can take advantage of the drastic performance improvements, without having to change much in your application. Probably just the way the Cypher queries are sent to the database, which is implemented in the driver.

When working with server mode, it's really important to use persistent connections (TLS negotiation phase, HTTP latency to establish a new

connection).
If your use case allows for it, it's possible to wrap thousands of operations into single transactions and reduce the number of HTTP requests.
There are a number of other things to take into account to achieve best performance results with server mode, but I wanted to mention these as they are directly related to the HTTP calls performance topic.

Cheers!
Alberto.