DEPARTMENT OF COMPUTER SCIENCE | LUNDS TEKNISKA HÖGSKOLA | PRESENTED 2024-01-30

MASTER THESIS Cache replacement policies and their impact on graph database operations
STUDENTS Tora Elding Larsson, Lukas Gustavsson
SUPERVISOR Jonas Skeppstedt (LTH), Anton Klarén (Neo4j)
EXAMINER Michael Doggett (LTH)

What to remember and what to forget

POPULAR SCIENCE SUMMARY **Tora Elding Larsson, Lukas Gustavsson**

Imagine that you are working on an essay at your desk. Instead of having all your books on a bookshelf, you have the books you need for the essay at your desk so that you don't have to go back and forth all the time. Unfortunately, all books don't fit on your desk, so sometimes you need to replace one. The question is which one?

In this thesis, the bookshelf is a long-term memory and your desk is a cache memory. For Neo4j, all data must be stored long-term, but you still want some data in the cache memory so that you can find it more quickly. Whenever the cache is full, something needs to be replaced. Imagine the bookshelf again. If you replace a book that you need soon after it has been replaced, you would have to go back and forth to the bookshelf two times in a very short period. A better idea would be to replace something that you don't need in the future, or at least not for a very long time. In the cache, the decision of what to replace is made by an algorithm called the eviction policy. The focus of this thesis has been to investigate the policy currently used by Neo4j and to try to improve it.

The algorithm currently used decides what to replace based on when it was used. If we go back to the book example it would mean that the book that was used the longest time ago is replaced, since we assume that a book that was recently used will be used again shortly. In Neo4j the cache is also threaded, meaning several threads use the cache at the same time. Imagine that several people worked on the essay at the same time at the same desk. Then you wouldn't want a person to replace a book that you might need soon.

We tried many approaches to improve the algorithm, based on previous research. Some approaches were to tweak the current algorithm, and some was to implement entirely new algorithms. We discovered that the current algorithm was working very well. However, tweaking some parameters in it and changing the structure for how the different threads were organized, improved the cache.