Project proposal

Hashing abstracts the mapping of keys to value in O(1) time for average case. Due to the O(1), it has fast lookup responses and are used in main memory databases. However in the world of databases unlike algorithms, we do care for the constant time as well and work to improve it. There are various state of art hashing algorithms such as XXHash3, CityHash and other non-cryptographic hashing alternatives that can perform superior performance as compared to the default hashing implemented in the open-source databases. However, replacing the existing hashing scheme can be complicated. I will have to figure out how to replace the existing access path, simulate the new hash like the old hash so that any existing functionality does not break. However that seems too ambitious to work in a month timeline. I will be instead focusing on different hash functions, benchmark them independently by varying different parameters.

A: Testing the hashing scheme in isolation

- Will be testing for parameters such as MiB/sec, cycl./hash , cycl./map, size and any quality problems with the hashing functions. I will be using the SMHasher test suite for testing this properties.
- Hashing schemes are typically built for DRAM. However, it can have certain problems when it comes to persistent memory[1]. DRAM has scaling issues as seen in the Five minutes Rule[2]. Even-though Intel optane was killed, there's still hope of new persistent memory technology coming to the market. I will be profiling the hashing schemes on the persistent memory and compare its performance with respect to the traditional architecture. Since I do not have physical access to persistent memory, I will be using Quartz[3] which is a performance simulator for Persistent memory software by HP.

B: Integrating atleast one hashing scheme with an open-source database

- I will chose an open-source database system like Postgres and replacing its hash function with different hashing algorithms, such as XXHash3 and CityHash, and benchmarking its performance with existing benchmark such as TPC-C as discussed in the office hour.

C: Testing B on persistent memory

- If replacing the hashing scheme works, I will try to test the same on the Quartz persistent memory simulator and compare the different.

[1] *Pengfei Zuo, Yu Hua, and Jie Wu. 2018. Write-optimized and high-performance hashing index scheme for persistent memory. In Proceedings of the 13th USENIX conference on Operating Systems Design and Implementation (OSDI'18). USENIX*

Association, USA, 461–476.
[2] Goetz Graefe. 2008. The Five-Minute Rule 20 Years Later: and How Flash Memory Changes the Rules: The old rule continues to evolve, while flash memory adds two new rules. Queue 6, 4 (July/August 2008), 40–52. https://doi.org/10.1145/1413254.1413264
[3] Haris Volos, Guilherme Magalhaes, Ludmila Cherkasova, and Jun Li. 2015. Quartz: A Lightweight Performance Emulator for Persistent Memory Software. In Proceedings of the 16th Annual Middleware Conference (Middleware '15). Association for Computing Machinery, New York, NY, USA, 37–49. https://doi.org/10.1145/2814576.2814806