02/28/2022

Today's discussion was about Column Store, basically stores the data in the form of columns and has several capabilities such as compression which is quite helpful for retrieving the values very easily and efficiently. While dealing with the real-world data situations where we have assumptions such as the data received will not have any updates, no deletions. Thus, in such scenarios it is very feasible to store logs and history of transactions using column store. With the previous idea to use R_A(RID, AttributeValue), we can go with the implicit representation for append only data, is an array of attribute values and indexed on top of the RID. One such implantation is Monet DB, problem associated with it involves processing the large arrays. Several questions come into mind, does it fit into the memory? Possibly we can. How does the data move? With large movement?

The implementation of Monet DB uses the columns as arrays which turns to be large. When needed to perform the vectorized operations we can utilize SSE, AVX instructions. It chunks the big column arrays such that the immediate vectors can fit into L2 cache. The 1K size is large enough to reduce the overhead of calling functions. Store the info by column in turn solves the cache line utilization problem. The ideal way to utilize them by performing the row store processing for column store access and process them simultaneously. Datapath & GrokIt, uses chunked column access adaptive execution. The minimum time for execution is at least 1-2 seconds resulting in a faster execution. The process of generating code and compiling on the fly involves LLM with around 100 – 2000 ms, and FPGAs for compiling and code generation using modern architecture.

03/02/2022

Based on the previous sections, we wanted to enhance the performance of the large sized data loads using the parallel DB workloads. With the concepts of modern architecture in mind, we need to re-think can we come up with an understanding to change the existing DB engines to perform good on high-core machine. Basically, for Analytical purposes we use 1 large query engine, and it consists of many small queries. A universal solution for the same is to make use of postgres as a unit and run multiple instances per machine. Several such implementations include Netezza, AsterData, GreenPlum, RedShift. Main idea behind it is to virtualize the CPU and memory and disk. Divides one large problem into smaller parts and execute on independent postgres instances and combines the result back to one. Communication between them is done via TCP-IP protocol and also sharding of the data is done in the database. This helps us to achieve the distributed databases and perform the operations across them for effective and faster results. In order to parallelize the operations across multiple databases, we need to use lock-free data-structures and doesn't scale beyond 4 cores. Reducing the implicit synchronization can help us to reduce the burden and locking makes the lease utilization of the resources even when we have the processing power.

03/04/2022

In the last class with the implementation of the distributed databases, we can think of the warmup sorting which is not as friendly with many-cores but can sort the data fast and can fit in L2 cache which is problematic with the large amount of data. Considering them, we can ask ourselves that when can the cores work independently with each other under what scenarios? Solution to it is when the data fits in L2 cache and when data is not shred across them. When we perform the linear scans on the data from the same NUMA nodes. We can group the data under small groups that can fit in L2 or below thus makes it to access all of them very easily, also performing the aggregate operations make it easier to process with less read operations concerning to the same group as adding and counting. Considering this, we can come up

with user defined aggregations pertaining to the scenario. Understanding the mechanics of how aggregations are performed is by using API. Some such examples are finding the Top K elements out of the given input and performing the Average of the elements, Obtaining the Max and Min of the data with the implementation of the basic operations of addition, division and multiplication.