03/21/2022

Today's discussion started with understanding the GLAs properties and one among them is understanding the adaptive execution. With the ease of various executions there arises a problem of handling the large states of GLAs and these are handled in a natural fashion. In the case of computational unit availability, a new GLA will be started including the initialization and state is built. In these scenarios where the GLAs work in-parallel, and data will be scanned from them and are fed into various GLA states. Using different states makes them achieve no sharing among them and attain the perfect parallel execution without any dependency. This involves merging the states in the GLA to form an adaptive tree merge strategy without having any conflicts between them. For parallel executions, each of the worker unit grabs any of the unmerged states and works on it to merge and saves the state to the pool to avoid repetitive working on the same. This has lot of overhead and eventually slows down to the end. For example, grouping several millions of records involves lot of time for the last states to take a bit of more time than any other states, another problem with this is if all the available GLAs have the final state demanding for more memory for all the states than the required.

A more general approach to deal with the large states is to have a large single such GLA and in parallel we need to process the data for which we get and need to build an average sized GLA such that it can have one chunk for each of it to handle. We need to free up the occupancy of the big GLA by putting a lock on it to be reserved or hung up with other tasks or else this will be a bottleneck. The process of segmentation involves hashing the segment from a large portion of it basically into 128 segments. Parallelism is used just like the above approach to merge them to produce the tuples in parallel using segments. If needed, we can utilize the 2 staged hash function to reassign any of the same assigned tasks. For situations where the # of groups are larger than the main memory we go with the approach to use sliding windows to avoid the overhead.

03/23/2022

Discussion started with the understanding on the previous lecture and a question on whether GLAs can run on multiple machines, to be able to compute a state per machine and make the merge tree spanning to the machines and be able to merge to the final master machine and be effective just like there are no conflicts to the performed tasks. To answer it is a little bit tricky when considering the memory allocation as it ends up taking lots of small memory allocations which eats up the memory. A better way to deal with it by using the right approach of reusing the TALLOC instead of MALLOC version for best version of assigning the memory. Avoid the locks on threads and switch gears to thread-only memory and can avoid allocation of memory. Usage of the custom-made allocators for much faster allocation of memory. Also, several swapping like usage of unordered_map and usage of oniguruma improves the process to handle with ease. On generalizing the operation on the GLAs as filtering to selection, transformer to projection, input to data loading.

03/25/2022

Today's discussion started with understanding the waypoints typically refers to a point of reference that is used to identify the well-known location and favorably used to navigate to that particular location. A similar kind of waypoint in Grokit is discussed, as it contains several such kind of them, where some of them are special grade and mainly relates to joins where there are inevitable when dealing with the large data. They can be related to something like pointer but are described in a generic fashion respective to the designated function/ task. They allow the communication from top

to bottom or bottom to top in a way to collaborate if needed in the Query planning which is really useful when we wanted to deal with specific things to be dealt with. One such waypoint we discussed is JoinWaypoint version, it is complicated to begin with. GLAWay Point performs the delay on the query that needs to happen, the strategy that it uses is the individual tasks creates a state and is used to manage it. The next discussion will be to understand some of the transactional systems and map-reduce.