Questions for week 4 (Total 20 Points)

Due Date: Wednesday, 22. December 2024

We discussed the parallelization of the hash join. Perhaps read the slides and readings about the intermediate variant, where the hashing phase is parallelized.
Which answer is correct concerning the hashing phase? (3 Points)

a) A sequential scan over the larger join relation is performed while probing into the hash map
b) Each thread builds in the parallelization phase the hash table by running completely through the smaller join relation and creates an intermediate hash table
c) Each hash thread scans its part of the input table, hashes the values and writes its results into a small local hash table. When the local hash table of a thread reaches a predefined size limit and does not fit into the cache anymore, it is written back to a shared buffer in main memory

1) **What is the disadvantage when the probing phase of a join algorithm is parallelized, and the hash phase is executed sequentially? (1 Point)**

   a) The sequential hash phase will run slower due to the large resource utilization of the parallel probing phase
   b) The algorithm still has a large sequential part that limits its potential to scale
   c) The table has to be split into smaller parts, so that every core, which performs the probing, can finish
   d) Sequential performing the hashing phase introduces inconsistencies in the produced hash values

2) **Why should intermediate hash tables in a parallel hash-join be optimized? (1 Point)**

   a) to avoid sequential accesses

   b) to avoid cache misses for random lookup in the hash table

   c) because merging of smaller tables is easier

   d) because the hash table performance decreases logarithmically with the table size

3) **How many threads will be used during the hash phase(!) of parallel aggregation when the table is split into 20 junks and the GROUP BY attribute has 8 distinct values? Take as an example the aggregate statement below and assume the country table has 8 distinct values.**
   SELECT country COUNT (*)
   FROM world_population
   GROUP BY country

   (2 Point)

   a) 10 thread, if the CPU has 10 cores

   b) 20 threads

   c) 8 threads

4) **If a column is very rarely used to identify or look-up specific tuples... (1 Point)**

[✓ a) … we probably don’t need an index on top of it, but use a full column scan]
b) … an index on top of it is a "strong advise"
c) … we cannot use a full column scan
d) ... we should delete it

5) **What is the scan performance of reading dictionary encoded column “city” of our World population table with 8 billions records with 50 byte of data in column city and with a CPU with 20 cores and a scan speed of 4 MB/msec/core.**
In a *row store* table layout with stride access of column city.
We have 1 mio cities in the world.
1 kb = 1.000 byte
(2 Points)

a) 25,6 sec
[✓ b) 0,348 sec]
c) 200 sec
d) 6,4 sec

6) **What is the scan performance of reading column dictionary encoded column “city” of our World population table with 8 billions records with 50 byte of data in column city and with a CPU with 10 cores and a scan speed of 4 MB/msec/core.**
In a *column store* table layout.
We have 1 mio cities in the world.
1 kb = 1.000 byte
(2 Points)

[✓ a) 0,5 sec]
b) 2,5 sec
c) 5 sec
d) 0,25 sec

7) **When one SELETC statement is executed in parallel and was executed before sequentially… (1 Point)**

a) it’s query plan is not changed
b) it’s query plan becomes much simpler compared to sequential execution
[✓ c) it’s query plan gets adapted accordingly]

8) **Give an example where an index would make sense (2 correct answers) (2 Points)**

a) All men born after 1.1.2010 in our world population table
[✓ b) All material movements for a material in an SAP ERP system]
[✓ c) For all entries for a customer in a general ledger in an SAP ERP system]

9) Analytical queries typically are… (1 Point)

a) Short running with soft time constraints
b) Long running with soft time constraints
c) Short running with strict time constraints
d) Long running with strict time constraints

10) Query response times… (1 Point)

a) have no impact on an users work behavior
b) should never be decreased as users are unfamiliar with such user behavior and can become frustrated
c) have to be as short as possible, so the user stays focused, at the task at hand
d) can be increased so the user can do as many tasks as possible because context switches are cheap

11) Regarding the hash join, which statement is correct? (1 Point)

a) Hashing introduces additional complexity since we cannot guarantee the absence of collisions
b) The hash-join has a higher run-time complexity than the nested-loop join
c) A sorted dictionary structure slows down the performance of the hash-join
d) Hashing of values is only possible when these are stored within a columnar layout

12) Workload Management tries to optimize the computer's resources such as processing power, CPU, memory, or network bandwidth to the running queries. (2 Points)

Concerning CPU scheduling, you have different options. For example:
a) First Come First Server (FCFS), which has the advantage to be simple and easy to understand
or
b) Shortest Job First (SJF), which has the advantage that the shortest jobs are favored
or
c) Round Robin (RR), which has the advantage that every process gets an equal share of the resources
or
d) Priority based (PB), which has the advantage that it provides a good mechanism where relative importance of each process may be precisely defined
or
e) Multilevel Queue Scheduling (MLS), which has the advantage that the application of separate scheduling for various kind of processes is possible for example scheduling for System Processes (FCFS), Interactive Processes (SJF), Batch Processes (RR), Student Processes (PB). The System Process has the highest, and the Student Process has lowest priority in the OS.
or
f) Multilevel Feedback Queue Scheduling (MLQS), which has the advantage of low scheduling overhead and therefore allows aging, thus no starvation.
Aging means: a process with low priority, gets after a certain time a higher priority. So sooner or later the process has high enough priority to get into execution.
Starvation means: a query gets postponed again and again, so it's never been executed, but MLQS) is on the other hand the most complex.

Now looking at the scheduling approach in SAP HANA, which one of the mentioned before comes close to the one in SAP HANA? Only one answer.
If you want you want to give also an explanation for your choice.
a)
b)
c)
d)
e) Multilevel Queue Scheduling (MLS)

f)

If you want to read more about it, look here: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-various-cpu-scheduling-algorithms/