Q1 Parallel Databases
30 Points

This online assignment is optional extra credit for HW 7. Any points you get on this assignment will be applied to your final grade after the curve.

We'll refer to our familiar schema of track meets and contestants:

```
Meet(id, name, year)
Event(id, mid, name)
Team(id, name, organization)
Contestant(id, tid, name, birthyear)
Performance(eid, cid, time, won) // Time in seconds. Won is 0 if lost, 1 if won.
```

As reminders:

- Meet.id, Event.id, Team.id, Contestant.id are primary keys in their respective tables.
- Performance (eid, cid) is the primary key of the Performance table. Performance links Events to Contestants on their primary keys.
- Event.mid is a foreign key to the Meet table.
- Contestant.tid is a foreign key to the Team table.

Furthermore, assume for this problem that we are storing this data in a parallel relational database with more than 10 nodes, and that all the tables are too large to be stored on a single node. Thus we must choose a partitioning scheme for each table.

Q1.1 Block Partitioning
5 Points

What is the benefit of block partitioning our tables? Explain in a few sentences.

Block partitioning is beneficial because it ensures an even distribution of data among the nodes. In general, this prevents nodes from becoming bottlenecks. Now our multiple processors can operate simultaneously and

hopefully reduce the runtime of our queries.

Q1.2 Hash Partioning
5 Points

What is the benefit of hash partitioning? Give one example of a table we should consider hash partitioning and what attribute to choose.

Hash partitioning is beneficial because it provides a wide distribution of the data (however one should beware the Justin Bieber Effect and act accordingly). Modding the hash value by the number of nodes evenly distributes the data among the nodes, ideally. Moreover, since the hash function is consistent attributes with the same value are sent to the same node - I think this can be beneficial because it will facilitate grouping by a certain attribute. For example, we could hash on the Team table with team id - since there are a bunch of teams (hopefully) we can avoid the JB effect and members of the same team will be sent to the same node. I think. :)

Q1.3 Joins and Partitions
10 Points

Assume that by far the most common query in our parallel system is the following:

```
SELECT C.id, COUNT(*)
FROM Performance P, Event E, Contestant C
WHERE P.eid = E.id AND
      P.cid = C.id
GROUP BY C.id
```

Assume all three tables are block partitioned to begin with. Describe what shuffles we would need to do in the parallel join query plan.

We need to use the "[block] partitioned hash equijoin algorithm." There will be multiple shuffles and we shuffle on the join attributes. So in this case, I believe a possible first course of action would be to shuffle on event id between the performance and even table. Then we can shuffle on that intermediate table and the Contestant table on contestant id in order to do the parallel query. :]

Q1.4
10 Points

Is there a way to partition our data differently such that we would need to do less shuffles? What is the best way to do this and how many shuffles would this query require in that case.

(Only consider partitioning, replication is not possible because of the size of the tables.)

I believe so. I think we could hash partition on event id. Then each event will have its data contained on one node, which reduces the number of shuffles by 1 (to a total of 1 I think). :}

Q2 NoSQL
10 Points

Our track meet database was stored in a simple parallel relational database with SQL as the query language. If we had access to more resources, we could use a larger cluster and store our database in a Spark cluster.

Q2.1
5 Points

What might be one **benefit** to using Spark for this database?

One benefit of using spark would be fault tolerance. Because it keeps a lineage/history of actions taken by nodes, recovery will be easy and efficient. And if there will be a lot of shuffling, spark will be faster because everything is computed in memory and data is not written to/ read from disk.

Q2.2
5 Points

Aside from the extra cost for the extra nodes we need to add, what is one **downside** to using Spark for this database?

One downside to using spark is its large memory usage, since it does not use disk storage to read/write data (if this is what is meant by the "extra cost for extra nodes to add" then I apologize for my idiocy). thank u for fun quarter.

HW7-Optional-Extra-Credit

● Graded

Select each question to review feedback and grading details.

Student

Andrew William Garwood

Total Points

39 / 40 pts

Question 1

Parallel Databases — **29** / 30 pts

1.1 **Block Partitioning** — **5** / 5 pts

✔ **+ 5 pts** Correct (many valid points, "guaranteed even distribution of tuples" is the simplest)

1.2 Hash Partioning — **5** / 5 pts

1.3 Joins and Partitions — **10** / 10 pts

1.4 (no title) — **9** / 10 pts

Question 2

NoSQL — **10** / 10 pts

2.1 (no title) — **5** / 5 pts

2.2 (no title) — **5** / 5 pts