CS632 Midsem, Spring 2011, Feb 26, 2011, 5.30-7.30 PM

Total Marks: 60

NOTE: Answer all subparts of a question together, do not split them up.

1. Volcano/MQO:

   (a) In Volcano, a goal with a required physical property is satisfied by either an algorithm node guaranteeing the property, or by an enforcer from the null property. This ensures that the physical equivalence DAG is actually a DAG, i.e. it does not have cycles. Explain using a small example with (fake but not unreasonable) costs, why in the context of MQO, it may make sense to allow derivations in the other direction, from a physical equivalence node with a stronger property to one with a weaker property. ...5

   (b) Consider the problem of index selection for a given set of queries and updates, i.e. what indices should be created to handle the given set of queries/updates at minimum cost.

   Explain how the MQO greedy algorithm can be applied to solve this problem. In particular, make sure you explain clearly how to compute the benefit of a particular index. ...5

2. Explain very briefly (a) how the split-delta idea helps minimize random IO when updating an index, and (b) why in addition to sorting on keys, a secondary sort on the operation (insert/delete) is required. ...4

3. Eddies/POP:

   (a) In terms of benefit to reoptimization, compare the approaches of LCEM and ECB. ...2

   (b) Explain, using an example, why the failure of the independence assumption can not only lead to a bad join order, but can also result in POP repeatedly finding bad plans. ...4

   (c) Explain with an example (using qualitative arguments, no need for exact numerical values) why it is possible for Eddies to use (in effect) a join order that is better than any static join order on the same set of relations. ...4

4. BigTable/PNUTS/Asynchronous View Maintenance

   (a) Both BigTable and PNUTS use replication and logging to handle failure, but the approaches differ significantly. What are the key differences in their approaches to (a) replication and (b) logging. ...2+2

   (b) The test-and-set-write feature of PNUTS can be useful to implement concurrency control without locking. Consider a case of two transactions, with two data items A and B. (a) Give an example of a non-serializable execution that would be prevented by the above feature, and (b) give an example of a non-serializable execution that would be allowed using the same feature. ...4

   (c) Megastore (which you may not have read about) uses an idea similar to test-and-set-write at the level of an entity-group, which is a set of related records, for which it maintains a single version number. Show that if we allow transactions to access only one entity group, serializability would be guaranteed. (Hint: the ordering is actually serial). ...5

   (d) In PNUTS, what is the need for a tablet master, given that each record specifies which site is its master? ...3

   (e) Although preaggregation (also known as combiner in Hadoop) could be used to reduce network traffic when maintaining an aggregate view, the Agrawal et al paper on Asynchronous view maintenance does not use it. Explain why. ...4

5. Hyracks:

   (a) Explain the difference between how Hyracks handles failures and how Hadoop handles failures. ...3

   (b) Based on the above, outline a situation where Hyracks will not make any progress, but Hadoop can complete the given task. ...2

   (c) Explain why the Compat mode performs better than Hadoop, while Native mode performs better than Compat. ...3

6. Consider the following task on a graph:

   - In the first step, node 0 (a special node not part of the graph really) sends, an initial score $v[i]$ to each node $S[i]$ (ignore the issue of how to read the arrays, assume they are available). All other nodes set their own value to 0 and vote to halt.
   - In subsequent steps, each node sums up the values it has received in the previous step, adds a fraction d of the sum it has computed to its own value, and spreads the remaining $1 - d$ fraction amongst all its out neighbours (assume even distribution). However, if the value to be spread is less than some value ϵ it votes to halt.

   (a) Write Pregel code to implement the above task. ...4

   (b) Now suppose at the end, each node also sends its value to an aggregator; the job of the aggregator is to output the top-k values. Outline how to implement such an aggregator; don't worry about exact code since the paper does not give the API for aggregators, just explain in words how you would implement such an aggregator. ...4

Total Marks = 60