Distributed databases

Jens Kristian R. Nielsen, Thomas D. Vinther

1. maj 2019

1 Abstract

In part 1 we consider how banks should handle moving money, and the perils of trying to move more money than exists on a given account. We find that a careful approach should be taken with low chance of failure, since it is a high stakes situation. So we should first attempt to remove the money from the sender, and only if successful attempt to add it to the receiver. The situation is much more complicated in reality, for instance there are policies on allowing ad hoc loans instead of denying negative account balance.
In part 2 we see that 2 phase commit handles aborts the same way as 3 phase commit, and write a diagram of an abort situation. In part 3 we consider optimizing the location of local copies of a given table, with regards to look ups and updates. We arrive at a brute force method that calculates all the costs with this formula
$\text{cost}(I) := \sum_{i \in N \setminus I} \left(10c \cdot q_i + 10d \cdot |I| \cdot u_i\right) + \sum_{i \in I} \left(c \cdot q_i + (10d \cdot (|I| - 1) + d) \cdot u_i\right)$ And simply chooses the index set I such that this cost function is minimized. In the example we cover we could believe that a relatively small number of copies is optimal, but we suspect that this varies greatly depending on the situation one finds oneself in.
Finally in part 4 we outline a method using MapReduce to look through a set of documents and count the occurrences of all words within all documents.

2 Solution

2.1

Consider an SQL query issued at bank A that moves 500kr from an account at A to an account at bank B.
a. How would the corresponding transaction as a transparent distributed query be decomposed for the sites in the distributed databases, assuming that A and B are a site each?

Let t be the transaction, then it is decomposed to t_A and t_B, and t_A is handled at site A and removes 500kr. t_B is handled at site B and adds 500kr, if t_A is successful.

b. What should the local databases at sites A and B do?

At A we try to remove 500kr from the corresponding account, if successful we send a message to B that we are ready to *transfer* and B tries to add 500kr to the account in question, if successful we send a message back to A and from there back to the client.

c. What can go wrong if there is less than 500kr in the account at A?

If your account balance is unable to sustain a loss of 500kr then we should throw an error and stop the transaction, unless some policies are in place allowing for negative balance. If however the transaction at B does not wait for a conformation and just adds 500kr to the account we obviously have a problem.

2.2

Consider a distributed database where site 0 is the coordinator, but not otherwise involved in query execution, and two regular sites with local databases that we call sites 1 and 2.
a. Provide a possible list of messages that would be sent in the system (by any site or by the coordinator) if site 1 wants to commit and site 2 wants to abort.

2PC:
1 -> 0: commit?
Coordinator decides to attempt commit.
0 -> 1: prepare
0 -> 2: prepare
1 -> 0: ready
2 -> 0: don't commit
0 -> 1: abort
0 -> 2: abort

Note that there is no use of 3PC because the voting failed, and phase 2 aborts, before the pre-commit stage.

2.3

We study a distributed database of 3 sites that stores a single table T. Our workload is that site 1 issues 30 queries to T per hour, site 2 issues 25, and site 3 issues 10. Similarly, site 1 issues 2 updates to T per hour, site 2 28, and site 3 90. Executing a query locally at a site costs 1, executing remotely if there is no local copy costs 10. Similarly, the cost of locally updating is 5, and 50 for each remote site.

a. Which sites should replicate T? Describe how you arrived at this conclusion.
An overview of the situation

	querys	updates
1	30	2
2	25	28
3	10	90

Now to see where a copy should be placed consider the following table, where the calculation has been included for clarity.

Copy at site	query cost	update cost	total cost
1	$30 \cdot 1 + 25 \cdot 10 + 10 \cdot 10$	$2 \cdot 5 + 28 \cdot 50 + 90 \cdot 50$	6290
2	$30 \cdot 10 + 25 \cdot 1 + 10 \cdot 10$	$2 \cdot 50 + 28 \cdot 5 + 90 \cdot 50$	5165
3	$30 \cdot 10 + 25 \cdot 10 + 10 \cdot 1$	$2 \cdot 50 + 28 \cdot 50 + 90 \cdot 5$	2510
1&2	$30 \cdot 1 + 25 \cdot 1 + 10 \cdot 10$	$2 \cdot 55 + 28 \cdot 55 + 90 \cdot 100$	10808
2&3	$30 \cdot 10 + 25 \cdot 1 + 10 \cdot 1$	$2 \cdot 100 + 28 \cdot 55 + 90 \cdot 55$	7025
1&3	$30 \cdot 1 + 25 \cdot 10 + 10 \cdot 1$	$2 \cdot 55 + 28 \cdot 100 + 90 \cdot 55$	8150
1&2&3	$30 \cdot 1 + 25 \cdot 1 + 10 \cdot 1$	$2 \cdot 105 + 28 \cdot 105 + 90 \cdot 105$	12665

We see that the cost is minimized if we keep a copy at site 3 and only site 3.

b. Now, generalize this problem as follows: we study a distributed database of n sites that stores a single table T. Our workload is that each of the sites i issues q_i queries and u_i updates to T per hour. Executing a query locally at a site costs c, executing remotely if there is no local copy costs 10c. Similarly, the cost of locally updating is d, and 10d for each remote site. Describe how to choose the sites at which to replicate T with respect to c and d.

Let $N = \{1, \ldots, n\}$ be the index set of our sites, we then wish to find the subset of these that minimizes the cost function.

$$I \in \mathcal{P}(N) : \emptyset \neq I \wedge \forall J \in \mathcal{P}(N)(\emptyset \neq J \Rightarrow \text{cost}(I) \leq \text{cost}(J))$$
$$\text{cost}(I) := \sum_{i \in N \setminus I} \left(10c \cdot q_i + 10d \cdot |I| \cdot u_i\right) + \sum_{i \in I} \left(c \cdot q_i + (10d \cdot (|I| - 1) + d) \cdot u_i\right)$$

We choose the non empty index set I that has the property that the cost function evaluates to a value smaller than all other non empty index sets. The cost function is the sum of two sums, the first calculates the cost of site operations of sites that does not hold a copy, and the second of those that do. A site that does not hold a copy costs 10c pr local query, 10d pr copy pr local update and here |I| is the number of copies. In a site that contains a copy we pay c pr local query and 10d pr copy except this one, which we only pay d for pr local update.

2.4

Describe how you can use MapReduce to count the number of documents in which each word appears.

a. Clearly outline what is emitted in each step, and what the input is. (hint: there is a similar problem studied and described in the second Distributed Databases lecture on MapReduce and you are encouraged to follow this example).

We wish to use MapReduce to count the number of documents in which each word appears, and we follow the hint given in the assignment.
As seen in the lectures on slide 31, we first use a Map Function that parses each document, and emits (word, document_id) pairs.

E.g. (Trump, 123), (tweets, 123), (Trump,123), (Trump, 124), (watches, 124), (TV, 124) ,...
The Reduce function takes all pairs for a given word, removes duplicates, counts the remaining pairs for the given word and emits (word,count) pair.

E.g. (Trump, 2),(tweets,1),(watches,1),(TV,1),...
In this way we first find all the instances of a word in all the documents. Reduce then only keeps one instance of a word for each document, and then counts all of the surviving instances.

3 Summary

In this assignment we looked at implementation and application of Databases. First off, we looked at SQL queries issued from one bank to another, finding this to be quite straightforward, as the query could easily be decomposed into one query for each bank. However this could also easily go wrong if one transaction failed, and so we put the restraint to the receiving bank, that it could not receive the money before they had been successfully "removed"from the first bank. Perhaps some more restraints should be applied in real life as well as proper recovery, but as we only looked at how a query would be decomposed we didn't really look too much into it, but we did mention some of the things that could go wrong.
We then looked at a distributed database with a central coordinator and provided a possible list of messages that would be sent in the system. We initially did both a 2PC and a 3PC, but as the result was the same we only provided the 2PC and mentioned 3PC. One thing we were perhaps not 100 percent sure of, was when the coordinator would send the prepare message. Would it be prompted by site 1 that wants to commit, or would it just try to commit at specific time intervals or some third option.
After looking at this distributed database we looked at the workload of a distributed database, simply calculating all possible combinations and choosing the least expensive. This could be generalized to a database with n sites, either once again calculating all combinations and taking the smallest.
Lastly we did a simply MapReduce following the example from the lectures, first mapping all the words, then using Reduce to delete duplicates and count the instances as described.
One thing we would perhaps like to learn more of is how to actually implement the MapReduce as the example wasn't very clear on the implementation.