Exercise Sheet 4: Asynchronous Parallelism

Commit Protocols

1. In the two-phase commit protocol, describe what happens in each of the following scenarios :

   (a) The coordinator fails before sending `Prepare` messages to the subordinates.

   Solution:
   Subordinates don't find any `Prepare` message in their logs and abort the transaction. The coordinator, after it recovers, also doesn't find anything in the log and aborts.

   (b) The coordinator sends `Prepare` messages and the link between itself and a subordinate S fails.

   Solution:
   The coordinator aborts the transaction assuming that S failed. S also aborts the transaction assuming that the coordinator failed. Other subordinates abort the transaction upon receiving `Abort` from the coordinator.

   (c) One of the subordinates S receives a `Prepare` message and crashes.

   Solution:
   The coordinator aborts the transaction. S, after recovering, also aborts the transaction as nothing has been logged.

   (d) A subordinate S receives the `Prepare` message, logs it and then crashes.

   Solution:
   The coordinator aborts the transaction. S, after recovering sees the `Prepare` message in the log and contacts the coordinator to know the status of the transaction. The coordinator responds with an abort and S aborts the transaction as well.

   (e) A subordinate S receives the `Prepare` message, writes `Abort` to its log and crashes

   Solution:

   The coordinator aborts the transaction. After recovering, S finds the `Abort` message in its log and aborts the transaction.

   (f) A subordinate S receives the `Prepare`, writes `Prepare` to its log and the link between itself and the coordinator fails.

   Solution:
   The coordinator aborts the transaction. S keeps pinging the coordinator for status periodically, and when the link is re-established, the coordinator tells S to abort the transaction.

   (g) A subordinate S receives the `Prepare` message, writes `Abort` to its log and the link between itself and the coordinator fails.

   Solution:
   Both the coordinator and S abort the transaction.

   (h) The coordinator fails before collecting all the responses to the `Prepare` message.

   Solution:
   All subordinates that voted `Yes` keep pinging the coordinator periodically, and waits for its response. The coordinator after recovering finds nothing in its log and aborts.

   (i) The coordinator fails after writing `Commit` to its log.

   Solution:

All subordinates keep pinging the coordinator periodically and wait for its response. The coordinator after recovering finds the `Commit` in its log and sends `Commit` to all subordinates.

(j) The coordinator fails after writing `Abort` to its log.

Solution:
All subordinates that voted `Yes` keep pinging the coordinator periodically and wait for its response. The coordinator after recovering finds the `Abort` in its log and sends `Abort` to all subordinates.

(k) The link between the coordinator and a subordinate S fails after the former writes `Commit` to its log.

Solution:
The subordinate must have voted `Yes` and therefore waits until it hears from the coordinator. When the link is restored, S asks the coordinator about the status of the transaction and gets a `Commit` message.

(l) The link between the coordinator and a subordinate S fails after the former writes `Abort` to its log.

Solution:
If the subordinate had voted `Yes` it waits to hear from the coordinator. When the link is restored, S asks the coordinator about the status of the transaction and gets an `Abort` message. If the subordinate had voted `No`, it aborts without waiting.

2. In the two-phase commit protocol, explain why subordinates need to force-write a prepared log record before sending a `Yes` vote using an example failure scenario. Show what happens if a subordinate does **not** force-write the prepared log record, then show what happens if the subordinate does force-write the prepared log record.

Solution:
Let's assume that all nodes respond to the coordinator with a `Yes` vote. In this case, the coordinator will force-write a commit log record and will send `Commit` messages to all the subordinates. At this point, the transaction is considered to have committed. It should thus be durable.

Let's first consider the case when a subordinate does not force write a prepared log record and crashes after sending a `Yes` vote. In this scenario, upon restarting, the recovery process will find no commit log records for the transaction. It will abort the transaction and "forget" about it. The system will then be in an inconsistent state. The transaction should have committed at all sites. Instead, one site aborted it.

Let's now consider the case when a subordinate does force write a prepared log record and crashes after sending a `Yes` vote. In this scenario, upon restarting, the recovery process will find that it is in the prepared state for the transaction. It will periodically try to contact the coordinator site to find out how the transaction should be resolved. In our scenario, the outcome of the transaction is a commit. Because the coordinator cannot forget about a committed transaction until it receives final `Ack`s from all nodes, it will correctly respond with a `Commit` message to the inquiry. The subordinate will then be able to commit the transaction properly.

3. Illustrate with examples how 2PC trades liveliness for safety while 3PC does the opposite.

Solution:
Case 1: The coordinator sends `Prepare` , receives `Yes` from all subordinates and starts sending `Commit/ PreCommit` messages. After sending it to one subordinate, the coordinator crashes. In 3PC, the subordinates communicate with each other and find the `PreCommit` message received by one, and they all commit. In 2PC, one subordinate commits, and the others block until coordinator recovers and then commit. Here 3PC is more live than 2PC.

Case 2: Same as above, except that there is also a network partition occurs that isolates that one subordinate from others after the coordinator crashes. In 3PC, the isolated subordinate commits after the timeout, and the rest abort. In 2PC, the isolated subordinate commits, and the others block until coordinator recovers and then commit. Here 2PC is still safe, but 3PC is inconsistent.

4. Among the following propositions, which one(s) describe an honest two-phase commit system?

(a) The TA decides to cancel the exercise session. He asks the students to vote either "yes" or "no", but gets no reply as most of them crashed reading about 2PC. The exercise session is maintained.

(b) The TA asks the students to vote to either cancel or maintain the exercise session and cancels it anyway.

(c) The TA decides to cancel the exercise session. He first asks the professor, who then requests the students to vote either “yes” or “no”. Based on the votes, the professor takes a decision.

(d) The TA wants to cancel the exercise session, asks the professor, who in turn asks the students to vote. Students take too much time to vote, and the exercise session is canceled.

(e) People decide independently to come or not to the exercise session.

Solution:
(a) and (c)

Consensus

1. In the Paxos algorithm, when is the earliest time we can say that a value V has been finalized?

   (a) When leader receives a majority prepare-ok and proposes V.

   (b) When a majority nodes accept V.

   (c) When the leader receives a majority accept-ok for value V.

   Note: The question is **not** asking when does the algorithm terminate and choose a value. Instead, it asks about a stage in the algorithm after which no matter what happens, when the algorithm eventually terminates, the value chosen would be V.

   Solution:
   (b)

2. Consider a Paxos system with three acceptors A, B, and C. Also assume that each proposer (less than 10) adds a fractional value denoting its id to the proposal number so that timestamps are unique and can be ordered.

   (a) Describe what happens (in terms of state updates and messages) when a proposer P_1 sends `Propose(1)`.

   Solution:
   All three acceptors set their highest proposal number as 1.1 and reply with `Promise(1, null)`

   (b) Suppose P_1 fails after sending all the `Propose(1)` messages and then recovers. Describe what happens when both P_1 and another proposer P_2 send `Propose(2)`. Consider the scenario where C crashes, and both A and B receive message from P_1 first.

   Solution:
   Both A and B first set their highest proposal number to 2.1 and reply to P_1 and then to 2.2 and reply to P_2. Both P_1 and P_2 gets promises from majority and proceed to send `Accept` messages.

   (c) In the scenario described above, what happens if B receives the message from P_2 first?

   Solution:
   In this case B sets its highest proposal number directly to 2.2 and rejects P_1's proposal. Only P_2 gets promises from a majority and proceeds to send `Accept` messages.

   (d) Now, suppose C has recovered. P_2 sends `Accept(2,`v_2`)` to A and B and crashes before sending it to C. Has the value been finalized?

   Solution:
   Both A and B have not received a proposal with a number greater than 2.2 and therefore accepts it. Yes, the value has been finalized.

(e) Suppose instead that P_2 sends `Accept(2,`v_2`)` to only A and crashes before sending it to B and C . Has value been finalized?

Solution:
A has not received a proposal with a number greater than 2.2 and therefore accepts it. However, the value has not been finalized as a majority of acceptors has not accepted it.

(f) Consider the scenario in 2d after which A fails. Describe what happens when P_1 sends `Propose(3)` and the algorithm is run until termination. (A does not recover)

Solution:
When P_1 sends `Propose(3)`, it is the highest proposal seen by B as well as C and they set it to 3.1. B replies with `Promise(3, 2:`v_2`)` and C replies with `Promise(3, null)`. Since B has accepted a value, P_1 chooses that value instead of its own, and then sends `Accept(3, `v_2`)`. Both B and C accept it and the algorithm terminates with v_2 as the chosen value.

(g) Consider the scenario in 2e after which A fails. Describe what happens when P_1 sends `Propose(3)` and the algorithm is run until termination. (A does not recover)

Solution:
When P_1 sends `Propose(3)`, it is the highest proposal seen by B as well as C and they set it to 3.1. Unlike the previous scenario, B has not accepted any value so far and therefore both B and C reply with `Promise(3, null)`. P_1 chooses its own value, say v_1, and then sends `Accept(3, `v_1`)`. Both B and C accept it and the algorithm terminates with v_1 as the chosen value.

CAP Theorem

1. What is the CAP theorem? To what kind of system does it apply?

   Solution:
   The CAP theorem states that for distributed data stores, it is possible to only provide two out of the Consistency, Availability and Partition tolerance guarantees.

2. What is the definition of consistency in the CAP theorem?

   Solution:
   Reads return the value of the most recent write or an error.

3. Describe the simplest system you can think of that provides consistency and partition tolerance, in a distributed setting.

   Solution:
   All reads return errors. We sacrifice availability.

4. Describe the simplest system you can think of that provides availability and partition tolerance, in a distributed setting.

   Solution:
   Each server maintains its own data store and they lazily synchronize whenever they want. They answer queries from the client based on their knowledge of the state of the store.

5. Among the following propositions, identify which of the CAP theorem guarantees is/are violated.

   (a) You matched on Tinder with your soul mate. Whenever you try to send a message, the application tells you to retry.

   Solution:
   Availability

(b) You create a slack channel to discuss about the next PUBG gaming session you will have with your Swiss and American friends. After a while, you only see messages from your Swiss friends. Worried, you call Bob who lives in the US, and learn that he only sees messages from your American friends.

Solution:
Consistency

(c) You open Facebook messenger on both your phone and computer and start talking with Mark Zuckerberg about privacy issues. You do not see the same content on your phone and on your laptop, making the conversation really confusing. All your private data is sold to the Cambridge analytica company.

Solution:
Consistency

(d) You want to find a torrent of the latest smurf movie. The website keeps telling you that there is no matching result. The same thing happens for any movie you try to find. Time to get a Netflix account...

Solution:
Availability

(e) You play an online video game. Suddenly, everybody freezes, and none of the players is able to move. Apparently, a cat played with the plug of one of the servers in the data-center, disconnecting it from the other machines...

Solution:
Availability

Serializability

1. Identify the isolation anomalies in each of the following scenarios. Assume that every operation by a single entity is one transaction and different transactions by different entities run concurrently.

(a) Pathé Flon is in the process of updating its schedule for Monday and adds a tentative screening for the Captain Marvel movie at 19:30. Alice looks at showtimes for the movie at various theatres online and buys a ticket from Pathé Flon. However, Pathé Flon changes their mind and decides not to show the movie.

Solution:
Dirty Read

(b) Alice and Bob buy tickets for the same seat for the same show in Pathé Flon from an online website.

Solution:
Lost Update (Overwriting uncommitted data)

(c) Alice decides to watch the movie only if the ticket price is less than 20 CHF. She finds a cheap ticket for 15 CHF and clicks on buy and is charged 25 CHF because Pathé Flon increased the prices in between.

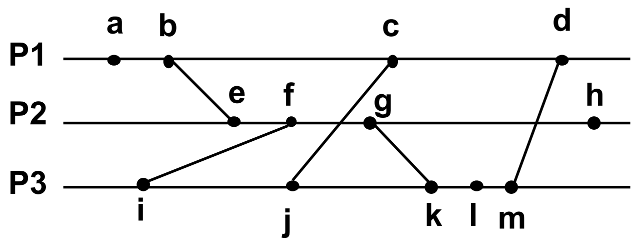
Solution:
Unrepeatable reads

(d) Pathé Flon introduces a special ticket for a group of friends to watch the movie together in five consecutive seats. Alice decides to watch the movie with 4 of her friends and looks at the list of available seats and buys the special ticket for seats starting from 21. Meanwhile, someone else buys a (normal) ticket for seat 23.

Solution:
Phantom read, Lost update

Weak Consistency

1. Assign timestamps to each of the events in the figure if each processor is following the vector clock algorithm. Also construct a table that indicates causal relationship between all pairs of events ($<$, $>$, or $\|$).



Solution:

a = [1,0,0]	b = [2,0,0]	c = [3,0,2]	d = [4,3,5]	
e = [2,1,0]	f = [2,2,1]	g = [2,3,1]	h = [2,4,1]	
i = [0,0,1]	j = [0,0,2]	k = [2,3,3]	l = [2,3,4]	m = [2,3,5]

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**
a		$<$	$<$	$<$	$<$	$<$	$<$	$<$	$\|$	$\|$	$<$	$<$	$<$
b	$>$		$<$	$<$	$<$	$<$	$<$	$<$	$\|$	$\|$	$<$	$<$	$<$
c	$>$	$>$		$<$	$\|$	$\|$	$\|$	$\|$	$>$	$>$	$\|$	$\|$	$\|$
d	$>$	$>$	$>$		$>$	$>$	$>$	$\|$	$>$	$>$	$>$	$>$	$>$
e	$>$	$>$	$\|$	$<$		$<$	$<$	$<$	$\|$	$\|$	$<$	$<$	$<$
f	$>$	$>$	$\|$	$<$	$>$		$<$	$<$	$>$	$\|$	$<$	$<$	$<$
g	$>$	$>$	$\|$	$<$	$>$	$>$		$<$	$>$	$\|$	$<$	$<$	$<$
h	$>$	$>$	$\|$	$\|$	$>$	$>$	$>$		$>$	$\|$	$\|$	$\|$	$\|$
i	$\|$	$\|$	$<$	$<$	$\|$	$<$	$<$	$<$		$<$	$<$	$<$	$<$
j	$\|$	$\|$	$<$	$<$	$\|$	$\|$	$\|$	$\|$	$>$		$<$	$<$	$<$
k	$>$	$>$	$\|$	$<$	$>$	$>$	$>$	$\|$	$>$	$>$		$<$	$<$
l	$>$	$>$	$\|$	$<$	$>$	$>$	$>$	$\|$	$>$	$>$	$>$		$<$
m	$>$	$>$	$\|$	$<$	$>$	$>$	$>$	$\|$	$>$	$>$	$>$	$>$	

2. Consider the following pairs of versions of data that a client obtains from two servers. Which pairs do not conflict?

   (a) Data 1: ([S_X, 1])
   Data 2: ([S_Y, 1])

   (b) Data 1: ([S_X, 2])
   Data 2: ([S_X, 1], [S_Y, 1])

   (c) Data 1: ([S_X, 2])
   Data 2: ([S_X, 2], [S_Y, 1])

   Solution:
   (c)

3. Consider the following pairs of versions of data that a client obtains from two servers. Which pairs do not conflict?

   (a) Data 1: ([S_X, 10], [S_Y, 20], [S_Z, 3])
   Data 2: ([S_X, 10], [S_Y, 10], [S_Z, 60])

(b) Data 1: ([S_X, 5], [S_Y, 10])
Data 2: ([S_X, 10], [S_Z, 30])

(c) Data 1: ([S_Y, 10])
Data 2: ([S_Y, 1])

(d) Data 1: ([S_X, 5], [S_Y, 10])
Data 2: ([S_X, 10], [S_Y, 20], [S_Z, 30])

Solution:
(c) and (d)