Welcome to ACS TA session 4

Julian, Leonardo, Li, Nikolaus, Rodrigo, Tilman, Yijian

Department of computer science, University of Copenhagen

Academic year 2021-2022, Block 2

Agenda for today

- **Revisit ARIES**
    - ARIES properties, approach, principles, data structures, phases
    - Exercises on the ARIES algorithm

Revisit ARIES: principles and properties

Properties

- **Atomicity**: undo the transactions that do not commit
- **Durability**: ensure all actions of committed transactions survive system

crashes and media failures

Approach

- **Steal**: pages are written to disk in yet uncommitted transactions
- **No-force**: when a transaction commits, its pages are not forced to disk

Principles

- **Write-ahead logging**: log the operation before executing it
- **Repeat history**: re-bring system to its state when it crashed and then fix
- **Log the undo**: to fully repeat the history, including the undo operations.

BUT we never undo the undo operations.

ARIES log record data structure

Log record

- **Log**: chronologic sequence of log entries
- **Log tail**: the portion of the log in main memory (not forced yet)
- **Log sequence numbers** (LSN): strictly increasing IDs for log records

Log record types and fields

- ***All**: prevLSN, transID, type*
- **Update**: pageID, length, offset, before-image, after-image
- **Commit**, **Abort**, **End**
- **Compensation** (CLR): undoNextLSN

ARIES additional data structures

Dirty pages table

- One entry per page not written to disk yet
- **Fields**: pageID, recLSN

Transactions table

- One entry per transaction
- **Fields**: transID, lastLSN, status
- Entries with status **committed** or **aborted** are removed from the

Table when the corresponding transaction reaches the **end** state

ARIES phases

Analysis

- Identifies dirty pages and active transactions

Redo

- Repeats all actions from safe point to moment of crash
- Leaves the data structures in the latest state prior to the crash

Undo

- Undoes the actions of uncommitted transactions reverse-chronologically

Exercise 1

1)If we can guarantee that uncommitted data is never written to disk, is **undo** still necessary?

2) What about **redo**?

3) If updates are always forced to disk when a transaction commits, is **undo** still necessary?

4) What about **redo**?

Exercise 1

1)If we can guarantee that uncommitted data is never written to disk, is **undo** still necessary?

2) What about **redo**?

No Steal – No Undo

3) If updates are always forced to disk when a transaction commits, is **undo** still necessary?

4) What about **redo**?

Exercise 1

1)If we can guarantee that uncommitted data is never written to disk, is **undo** still necessary?

No Steal – No Undo

2) What about **redo**?

3) If updates are always forced to disk when a transaction commits, is **undo** still necessary?

Force – No Redo

4) What about **redo**?

ARIES Questions

After a crash failure, where in the log ...

1. should the **analysis** phase start?

2. should the **redo** phase start?

3. should the **undo** phase start?

ARIES Questions

After a crash failure, where in the log ...

1. should the **analysis** phase start?

   most recent checkpoint

2. should the **redo** phase start?

3. should the **undo** phase start?

ARIES Questions

After a crash failure, where in the log ...

1. should the **analysis** phase start?

   most recent checkpoint

2. should the **redo** phase start?

   smallest recLSN in dirty page table

3. should the **undo** phase start?

ARIES Questions

After a crash failure, where in the log ...

1. should the **analysis** phase start?

   most recent checkpoint

2. should the **redo** phase start?

   smallest recLSN in dirty page table

3. should the **undo** phase start?

   largest lastLSN in transaction table

Exercise 2

Apply the ARIES recovery algorithm to the next scenario. Show:

1. the state of the transaction and dirty page tables after the **analysis** phase
2. the sets of winner and loser transactions
3. the values for the LSNs where **redo** starts and **undo** ends

How far back into the log must ARIES scan during **redo** and **undo**?

What are:

4. the set of log records that may cause pages to be rewritten during redo?
5. the set of log records undone during undo?
6. the contents of the log after the recovery procedure completes?

Exercise 2

Xact table

transID	status	lastLSN
T2	running	2
T1	committed	3

Dirty page table

pageID	recLSN
C	1
B	2

LSN	prevLSN	transID	type	pageID
1	null	T1	update	C
2	null	T2	update	B
3	1	T1	commit	

4
5
			begin checkpoint
			end checkpoint	

6	3	T1	end	
7	null	T3	update	A
8	2	T2	update	C
9	8	T2	commit	
10	9	T2	end	
CRASH!!!				

Exercise 2

1.the state of the transaction and dirty page tables after the **analysis** phase

Xact table

transID	status	lastLSN
T2	running	2
T1	committed	3

Dirty page table

pageID	recLSN
C	1
B	2

LSN	prevLSN	transID	type	pageID
1	null	T1	update	C
2	null	T2	update	B
3	1	T1	commit	

4
5
			begin checkpoint
			end checkpoint	

6	3	T1	end	
7	null	T3	update	A
8	2	T2	update	C
9	8	T2	commit	
10	9	T2	end	
		CRASH!!!		

Exercise 2

1.the state of the transaction and dirty page tables after the **analysis** phase

Xact table

transID	status	lastLSN
T3	running	7

Dirty page table

pageID	recLSN
C	1
B	2
A	7

LSN	prevLSN	transID	type	pageID
1	null	T1	update	C
2	null	T2	update	B
3	1	T1	commit	

4
5
			begin checkpoint
			end checkpoint	

6	3	T1	end	
7	null	T3	update	A
8	2	T2	update	C
9	8	T2	commit	
10	9	T2	end	
CRASH!!!				

Exercise 2

2.the sets of winner and loser transactions

Xact table

transID	status	lastLSN
T3	running	7

Dirty page table

pageID	recLSN
C	1
B	2
A	7

LSN	prevLSN	transID	type	pageID
1	null	T1	update	C
2	null	T2	update	B
3	1	T1	commit	

4
5
			begin checkpoint
			end checkpoint	

6	3	T1	end	
7	null	T3	update	A
8	2	T2	update	C
9	8	T2	commit	
10	9	T2	end	
CRASH!!!				

Exercise 2

2.the sets of winner and loser transactions

Xact table

transID	status	lastLSN
T3	running	7

Winner:T1, T2

Loser: T3

Dirty page table

pageID	recLSN
C	1
B	2
A	7

LSN	prevLSN	transID	type	pageID
1	null	T1	update	C
2	null	T2	update	B
3	1	T1	commit	

4
5
			begin checkpoint
			end checkpoint	

6	3	T1	end	
7	null	T3	update	A
8	2	T2	update	C
9	8	T2	commit	
10	9	T2	end	
CRASH!!!				

Exercise 2

3.the values for the LSNs where **redo** starts and **undo** ends

Xact table

transID	status	lastLSN
T3	running	7

Dirty page table

pageID	recLSN
C	1
B	2
A	7

LSN	prevLSN	transID	type	pageID
1	null	T1	update	C
2	null	T2	update	B
3	1	T1	commit	

4
5
			begin checkpoint
			end checkpoint	

6	3	T1	end	
7	null	T3	update	A
8	2	T2	update	C
9	8	T2	commit	
10	9	T2	end	
CRASH!!!				

Exercise 2

3.the values for the LSNs where **redo** starts and **undo** ends

Xact table

transID	status	lastLSN
T3	running	7

Dirty page table

pageID	recLSN
C	1
B	2
A	7

	LSN	prevLSN	transID	type	pageID
Redo start	1	null	T1	update	C
	2	null	T2	update	B
	3	1	T1	commit	

	4
	5
				begin checkpoint
				end checkpoint	

	6	3	T1	end	
Undo ends	7	null	T3	update	A
	8	2	T2	update	C
	9	8	T2	commit	
	10	9	T2	end	

CRASH!!!

Exercise 2

4.the set of log records that may cause pages to be rewritten during redo?

Xact table

transID	status	lastLSN
T3	running	7

Dirty page table

pageID	recLSN
C	1
B	2
A	7

Redo start

LSN	prevLSN	transID	type	pageID
1	null	T1	update	C
2	null	T2	update	B
3	1	T1	commit	

4
5
			begin checkpoint
			end checkpoint	

6	3	T1	end	
7	null	T3	update	A
8	2	T2	update	C
9	8	T2	commit	
10	9	T2	end	
CRASH!!!				

Exercise 2

4.the set of log records that may cause pages to be rewritten during redo?

Xact table

transID	status	lastLSN
T3	running	7

Dirty page table

pageID	recLSN
C	1
B	2
A	7

Redo start

LSN	prevLSN	transID	type	pageID
1	null	T1	update	C
2	null	T2	update	B
3	1	T1	commit	

4
5
			begin checkpoint
			end checkpoint	

6	3	T1	end	
7	null	T3	update	A
8	2	T2	update	C
9	8	T2	commit	
10	9	T2	end	
CRASH!!!				

Exercise 2

5.the set of log records undone during undo?

Xact table

transID	status	lastLSN
T3	running	7

Dirty page table

pageID	recLSN
C	1
B	2
A	7

Undo ends

LSN	prevLSN	transID	type	pageID
1	null	T1	update	C
2	null	T2	update	B
3	1	T1	commit	

4
5
			begin checkpoint
			end checkpoint	

6	3	T1	end	
7	null	T3	update	A
8	2	T2	update	C
9	8	T2	commit	
10	9	T2	end	

CRASH!!!

Exercise 2

5.the set of log records undone during undo?

Xact table

transID	status	lastLSN
T3	running	7

Dirty page table

pageID	recLSN
C	1
B	2
A	7

Undo ends

LSN	prevLSN	transID	type	pageID
1	null	T1	update	C
2	null	T2	update	B
3	1	T1	commit	

4
5
			begin checkpoint
			end checkpoint	

6	3	T1	end	
7	null	T3	update	A
8	2	T2	update	C
9	8	T2	commit	
10	9	T2	end	
CRASH!!!				

Exercise 2

6.the contents of the log after the recovery procedure completes?

Xact table

transID	status	lastLSN
T3	running	7

Dirty page table

pageID	recLSN
C	1
B	2
A	7

ToUndo:

LSN	prevLSN	transID	type	pageID
1	null	T1	update	C
2	null	T2	update	B
3	1	T1	commit	

4
5
			begin checkpoint
			end checkpoint	

6	3	T1	end	
7	null	T3	update	A
8	2	T2	update	C
9	8	T2	commit	
10	9	T2	end	
CRASH!!!				

Recovery: The UNDO Phase

ToUndo={ *lsn* | *lsn* a lastLSN of a “loser” Xact}

Repeat:

- Choose largest LSN among ToUndo.
- If this LSN is a CLR and undonextLSN==NULL
  - Write an End record for this Xact.
- If this LSN is a CLR, and undonextLSN != NULL
  - Add undonextLSN to ToUndo
- Else this LSN is an update. Undo the update, write a CLR, add prevLSN to ToUndo.

Until ToUndo is empty.

Source: Ramakrishnan & Gehrke (partial)

Exercise 2

6.the contents of the log after the recovery procedure completes?

Xact table

transID	status	lastLSN
T3	running	7

Dirty page table

pageID	recLSN
C	1
B	2
A	7

ToUndo: 7

LSN	prevLSN	transID	type	pageID
1	null	T1	update	C
2	null	T2	update	B
3	1	T1	commit	

4
5
			begin checkpoint
			end checkpoint	

6	3	T1	end	
7	null	T3	update	A
8	2	T2	update	C
9	8	T2	commit	
10	9	T2	end	
CRASH!!!				
11	7	T3	abort	
12	11	T3	CLR:Undo LSN 7	
13	12	T3	end	

Exercise 3

The next scenario depicts a situation where the system crashes during recovery. Apply the ARIES algorithm after:

1. **Crash 1** that occurred during normal execution
2. **Crash 2** that occurred during recovery

Exercise 3

LSN	prevLSN	transID	Type	undoNextLSN	pageID
01 05			Begin checkpoint End checkpoint		
10		T1	Update		P5
20		T2	Update		P3
30	10	T1	Abort		
40 45	30 40	T1	CLR:Undo LSN 10 End		
50		T3	Update		P1
60	20	T2	Update		P5
CRASH 1 !!!					
70	60	T2	Abort		
80	50	T3	Abort		
90	70	T2	CLR:Undo LSN 60	20	
100 105	80 100	T3	CLR:Undo LSN 50 End		
CRASH 2 !!!					

Thank you!

Julian, Leonardo, Li, Nikolaus, Rodrigo, Tilman, Yijian

Department of computer science, University of Copenhagen

Academic year 2021-2022, Block 2