

Checkpoints

Overview

In this video, we discuss checkpoints. A few advantages of these: faster recovery and less space used for the log-file

Simple Checkpointing (for undo logging)

Idea: *checkpoint* the log periodically

- Every m min., after t transactions since last checkpoint, ...
- No need to undo transactions before the checkpoint

Part before checkpoint can therefore be discarded from log

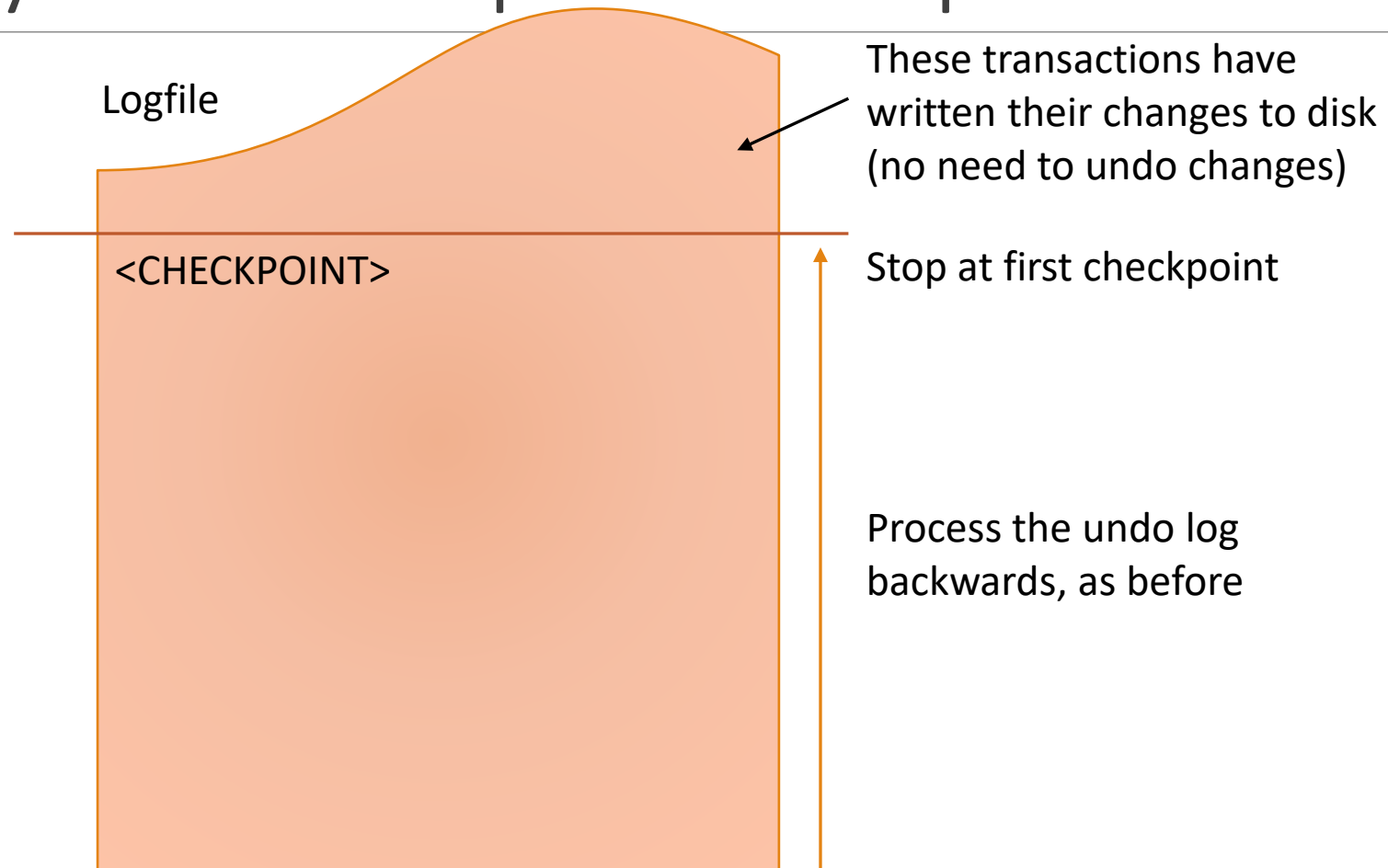
Procedure:

1. Stop accepting new transactions
2. Wait until all active transactions finish and have written COMMIT or ABORT record to the log.
3. Flush the log to disk.
4. Write a log record **<CHECKPOINT>**.
5. Flush the log to disk again.
6. Resume accepting transactions.

There are variants of checkpointing that avoid this
See later!

Recovery via Simple Checkpoints

Recovery With Simple Checkpoints



Transaction 3 can then proceed after we wrote <CHECKPOINT>

Transaction 3 will have to wait until we are done with our checkpoint

Checkpointing starts

Transaction T₃ is submitted

Write <CHECKPOINT> in log and flush

Time	Transaction T ₁	Transaction T ₂	Log (buffer)	Log (disk)
0			<START T ₁ >	
1	read(X)			
2	X := X * 2			
3	write(X)		<T ₁ , X, 1>	
4			<START T ₂ >	
5		read(X)		
6	read(Y)			
7		X := X * 3		
8		write(X)	<T ₂ , X, 2>	
9	Y := X + Y			
10	write(Y)		<T ₁ , Y, 2>	
11	flush_log			
12	output(X)			
13	output(Y)			
14			<COMMIT T ₁ >	
15	flush_log			
16		flush_log		
17		output(X)		
18			<COMMIT T ₂ >	
19		flush_log		

ARIES Checkpoints

Requirements:

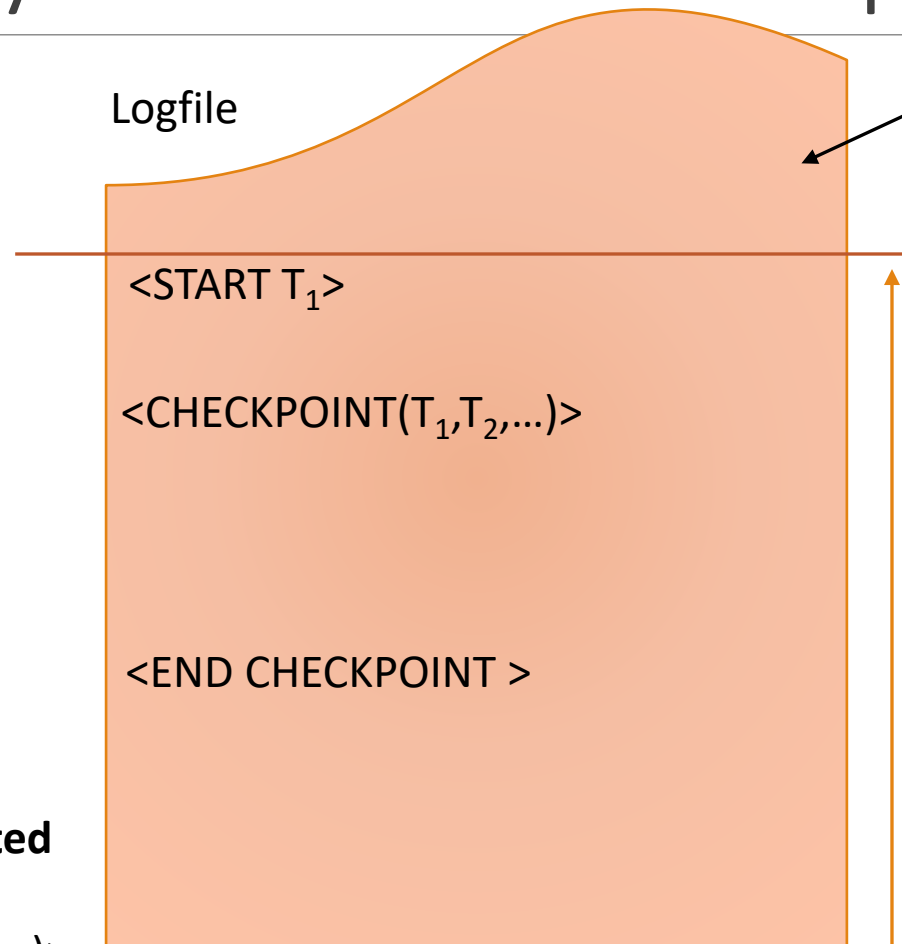
- Undo/Redo logging
- Transactions do not write to buffers(!) before they are sure they want to commit

Procedure:

- Write $\langle \text{CHECKPOINT}(T_1, T_2, \dots) \rangle$ in log and flush it
 - T_1, T_2, \dots are the transaction in progress (i.e. not committed and not aborted)
- Write the content of the **buffer** to disk (i.e. **output**)
- Write $\langle \text{END CHECKPOINT} \rangle$ in log and flush it

Recovery via ARIES Checkpoints

Process the undo/redo log,
as before EXCEPT:
only redo part of
committed transactions in
 T_1, T_2, \dots after
<CHECKPOINT(T_1, T_2, \dots)>
AND then undo all of **uncommitted**
transactions in T_1, T_2, \dots
incl. before <CHECKPOINT(T_1, T_2, \dots)>



These transactions have
written their changes to disk
(no need to do changes)

Stop after having found a
<CHECKPOINT(T_1, T_2, \dots)>
with corresponding
<END CHECKPOINT>
AND <START T_i > for all
mentioned transactions
that are **uncommitted**

Summary

Checkpoints can be used to speed up recovery and use less space on the log files

We saw Simple Checkpoints which simply prevents new transactions from starting until all are done

We saw ARIES Checkpoints that were more advanced and required undo/redo and you can't write to the buffer before being sure you want to commit