Use cases for DOM lock retrieving

Several use-cases of DOM optimizations are based on DOM lock being returned early to the client, so we avoid extra RPC. Finally it should be paired with LAYOUT lock to be used for an IO request. At the same time a DOM lock bit itself doesn't provide any IO without corresponding cl_object attached in l_ast_data, so technically it is safe to have it without valid layout, e.g. if layout is not yet received or being invalidated.

DOM and LAYOUT lock bits returning strategies

Consider possible strategies how DOM lock can be returned to the client and used lately.

				Client has no LAYOUT lock					Other client has		Comments
OPEN enq				**LAYOUT enq**			**DOM enq**	**TOTAL RPCs**	LAYOUT lock	DOM lock	
#	ibits	available with lock_try	mode	ibits	available with lock_try	mode					
1	-	DOM + LAYOUT	PW/PR	-	-		-	1 (OPEN)	X	X	Works only if no other clients have LAYOUT or DOM lock. Note that LAYOUT has strict PR/PW mode due to DOM so another open for the same file will fail on lock_try() for these bits and fall to cases #3-6
2	-	LAYOUT	PW/PR	-	-		DOM	2 (OPEN + DOM)	depends on mode	with BL AST	very common case if other client uses the same file, it has compatible LAYOUT lock but blocking DOM lock. Also this is case when OPEN lock in taken, because DOM is not compatible with it and only LAYOUT bit is trying
3	-	DOM	PW/PR	LAYOUT	-	CR[CW]	-	2 (OPEN + LAYOUT)		X	Quite likely this is cases #9-11 otherwise it is possible also if layout is being changed. This is still safe and it is very unlikely that DOM component will disappear after that so DOM lock will be still used after all. With lock convert it can be also the case when client has DOM+LAYOUT lock and DOM bit was canceled. So this new OPEN tries DOM bits successfully but fails to get LAYOUT.
4	-	-		LAYOUT+DOM	-	PW/PR??	-	2 (OPEN + LAYOUT)	can be BL AST	with BL AST	All three cases here and below are about other client has lock already and it is combined LAYOUT + DOM too, so OPEN failed to try both. This particular case cause BL AST on LAYOUT enqueue. It will cancel lock on other client, which will be canceled anyway after all, but we spend 2 RPCs for that. The problem with that case and case #5 is that it is not clear what mode to use for DOM lock, it must be PR/PW but unlike OPEN there is no either information about file open mode or future IO mode. Probably it can be get from CLIO though. Meanwhile it is not the only problem with this case.
5	-	-		LAYOUT	DOM	PW/PR??	-	2 (OPEN + LAYOUT)	can be BL AST	X	For the above case we may try DOM during LAYOUT enqueue. This is unlikely after open because we just failed to try it at OPEN.
6	-	-		LAYOUT	-	CR[CW]	DOM	3 (OPEN + LAYOUT + DOM)		with BL_AST	The worst case when each lock takes own RPCs. Note that after all we just finished with the same blocking AST to the client with DOM+LAYOUT lock but spend 3 RPCs for this.
7	DOM	LAYOUT	PW/PR	-	-		-	1 (OPEN)	depends on mode	with BL AST	Considering that DOM lock covers whole component we may ask it mandatory during OPEN. In case of other client has DOM lock it will be revoked and it takes 1 RPC, instead of 2-3 RPCs in cases #4 - 6. It looks also better than case #2. The idea behind mandatory DOM lock at OPEN is that if other client has DOM lock then it will be canceled in any case, but with lock_try() we never succeeded and just delay its canceling on the last DOM enqueue. SO to achieve the same result we will spend 2-3 RPCs while with mandatory DOM bit on open it is 1 RPCs in most cases.
8	DOM	-	PW/PR	LAYOUT	-		-	2 (OPEN + LAYOUT)		with BL AST	Variant of case #7 if other client has LAYOUT lock in conflict mode.
				Client has LAYOUT lock							

	OPEN enq			LAYOUT enq		DOM enq				
	ibits	lock_try		ibits	trying					
9	-	DOM		client has LAYOUT lock		-	1 (OPEN)		X	Asking about trybits on server we don't know if client has LAYOUT lock or not, so returning just DOM bit can be exactly what is needed to safe RPCs
10	-	-		client has LAYOUT lock		DOM	2 (OPEN + DOM)		with BL AST	Some client has DOM lock too, separate RPCs will cancel it. Can be avoided by case #11
11	DOM	-		client has LAYOUT lock		-	1 (OPEN)		with BL AST	The same as case #9 but works better because no need it case #10 as the worst case. So from all cases when client has LAYOUT lock already, this one is the optimal and preferred.

Note 1: when we get DOM lock at open it uses open mode to set lock mode PW or PR, when DOM enqueue is issued, it uses real IO mode READ or WRITE. With LAYOUT enqueue we have no such information about which mode to use that is why these rows are yellow.

Use Cases

We have several quite common situations to handle.

New file is created

We may use cases #1 and #7, in both we ends up with 1 RPCs.

Other client with DOM+LAYOUT lock on the same file

The other client has DOM + LAYOUT (PW/PR), so lock_try() will fail at open and it is cases #4, #6 and #8(7). Case #6 is the worst one and we can avoid it with #4 or #8. They both provides 2 RPCs but have significant differences:

- Case #4: final lock is DOM+LAYOUT (PW/PR) mode, the other client lock is canceled when we are getting layout. This resulting clients to have DOM+LAYOUT almost always together for DOM files. While this case work similar to #8 it is difficult to implement due to two reasons. 1) we have no idea about what lock mode to use PW or PR. 2) LAYOUT enqueue has no `mdt_body` to return DOM attributes along with lock. Both reasons can be solved in one or another way but it worths to do only if these cases are really better than others.
- Case #8: final lock is DOM(PW/PR) and LAYOUT(CR), the other client lock is canceled during open phase. Considering drawback of strategy #4, this one looks more preferable.

Other client with separate DOM and LAYOUT lock on the same file

In that case OPEN will fail to try DOM lock while be able to return LAYOUT lock, so it is cases #2 and #7 as main cases and may have cases #4, #6 and #8 if LAYOUT mode is CW for some reason (layout change). The #2 is 2 RPCs but #7 is 1 RPC.

- Case #2: as result we will have again separate LAYOUT and DOM locks, DOM is taken only when needed.
- Case #7: the result is combined DOM + LAYOUT lock.

Client has LAYOUT lock

Obviously we should consider cases when client has LAYOUT lock already. Having LAYOUT lock for some previously used file is a common condition considering it has CR mode. So file can lost DOM lock due to other client activity but LAYOUT lock will stay.

In that case the better choice is case #11 as shown in a table, it provides always 1 PRCs. While #10 is 2 RPCs case.

Main strategies for DOM lock returning.

Considering troubles with returning the DOM lock along with LAYOUT enqueue, there are 2 possible strategies remains.

1. Optional DOM+LAYOUT at OPEN, separate LAYOUT enqueue and DOM enqueue

While this one is simple approach but it give us less benefits. It work only with new files, for all conflict cases with the same file being used on other client it would produce 2-3 RPCs.

Consider the case of parallel work on the same file:

1. client #1: opens new file, DOM+LAYOUT lock (PW), do write (1 RPC)

2. client #2: open the same file - case #6 in table - get finally LAYOUT lock (CR) and DOM lock (PR/PW), do IO. Both locks are lost by client #1 (3 RPCs + 1 BL AST)
3. client #1: Now we have two scenarios, a) when file is kept opened and b) open/closed each time:
    a. OPENED: gets LAYOUT(CR) separately and new DOM (PW) (2 RPCs + 1 BL AST)
    b. CLOSED: OPEN returns LAYOUT(PW/PR) but without DOM and new DOM (PW) (2 RPCs + 1 BL AST)
4. And then client #2:
    a. OPENED: takes DOM lock (PR/PW) only, after that step clients will just ping-pong an IO lock (1 RPCs + 1 BL AST)
    b. CLOSED: the same but with OPEN RPC (2 RPCs + 1 BL AST)

The first conflict with other client causes the worst case so in general this strategy works bad in cases when one client writes something in files and other reads that, especially when this is new file each time.

So the first 4 iterations give us 7 enqueue RPCs if file is being opened always and 8 enqueue RPCs if it is also closed after each access.

2. Mandatory DOM at OPEN

This is also quite simple and good strategy with 1 - 2 PRCs in all cases, the only drawback is DOM lock cancel for concurrent client at open stage.

The same case as above gives us always not more than 2 RPCs:

1. client #1: opens new file, DOM+LAYOUT lock (PW) (1 RPC)
2. client #2: open the same file - case #8 in table - get DOM lock (PR/PW) and separate LAYOUT(CR). Both locks are lost by client #1 (2 RPCs + 1 BL AST)
3. client #1:
    a. OPENED: get new LAYOUT lock (CR) and DOM lock (2 RPCs + 1 BL AST)
    b. CLOSED: gets DOM+LAYOUT (PR/PW) at open because LAYOUT bit is compatible with CR mode on other client and DOM is taken mandatory (1 RPCs + 1 BL AST)
4. client #2:
    a. OPENED: issues new DOM lock only (1 RPC + 1 BL AST)
    b. CLOSED: open with DOM lock and separate LAYOUT(CR) because LAYOUT lock fails lock_try() due to lock mode. (2 RPCs + 1 BL AST). Since that step clien #1 will always take DOM+LAYOUT and client #2 separate DOM lock.

Therefore this strategy issues 6 enqeue RPCs always for the same 4 first iterations. This strategy works better with shared files accesses. Meanwhile it may cause unnecessary lock cancels if files are opened but not used for any IO or opened with RW mode just for READ (remark **1**). E.g. if many clients read the same file but opens it with RW mode, each open will cause lock cancel on each client with PR DOM lock. Basically, this will lead to the only one client has a lock at the moment, since it is PW lock. Though this is wrong application behavior, it should be taken into account.

Short READ along with lock getting

Another case for Data-on-MDT optimization is the data prefetch from server with DOM lock during OPEN and other RPCs. It is very important to note that unlike DOM lock only, the data cannot be read without both DOM and LAYOUT locks (remark **2**). If we have both locks we can do READ on OPEN and return data in reply OR we can do the same on DOM lock enqueue and also return data fit into reply buffer (remark **3**). In the latest case more data can be returned. When data is returned by OPEN then llite places it into page cache where they will found by CLIO immediately. When data is returned along with DOM lock the CLIO itself may handle it.

Consider above strategies for DOM lock plus short READ, so the table of possible cases is much less now because cases without LAYOUT lock are removed

	Strategy #1 (optional DOM + LAYOUT on OPEN and optional DOM at LAYOUT enq)							**Comments**	
	OPEN enq		**LAYOUT enq**			**DOM enq**	**TOTAL RPCs**		
#	ibits	lock_try	ibits	lock_try	mode				
1	-	DOM + LAYOUT **READ**	-	-		-	1 (OPEN/READ)	Works only for new open.	
2	-	LAYOUT	-	-		DOM **READ**	2 (OPEN + DOM/READ)	very common case if other client uses the same file and have LAYOUT lock with CR mode. In that case READ can be done during DOM lock enqueue.	
5	-	-	LAYOUT **READ**	DOM	PW/PR	-	2 (OPEN + LAYOUT/READ)	It would be troublesome to return data with LAYOUT lock it uses LVB for layout, so data attributes can't be transferred nether in LVB nor in mdt_body which doesn't exist with LAYOUT enqueue. This could be added though, but this particular case needs also lock convert to prevent data flush due to layout lock revocation.	
6	-	-	LAYOUT	-	CR/CW	DOM **READ**	3 (OPEN + LAYOUT + DOM/READ)	The worst case for lock getting due to numbers of RPCs but still we can read on DOM lock getting.	

9	-	DOM **READ**	client has LAYOUT lock	client has LAYOUT lock		-	1 (OPEN/READ)	While this is valid READ but it is unknown on server that client has LAYOUT already. It can be done though by searching for such lock in server queue.	
10	-	-	client has LAYOUT lock	client has LAYOUT lock		DOM **READ**	2 (OPEN + DOM/READ)	most frequent case for READ in case of shared file	
	Strategy #2 (mandatory DOM lock at OPEN)								
7	DOM	LAYOUT **READ**	-	-		-	1 (OPEN/READ)	Should be quite common case if other client has separate locks. But unfortunately the paired case with DOM on open without LAYOUT cannot be used because data shouldn't be read without layout, even the in case below, when client has LAYOUT lock already it would require extra work.	
11	DOM **READ**	-	client has LAYOUT lock	client has LAYOUT lock		-	1 (OPEN/READ)	The same restriction here - we don't know on server if client has layout. We can search for that lock though.	

As it seen from table, the strategy #1 (optional DOM + LAYOUT) is better for READ prefetch. The main issue with stategy #2 (mandatory DOM lock at open) is that we may have no LAYOUT lock and even if client has it already, that is not known now, though can be done.

NOTE: Strategy #2 may work much better if PR lock is taken always on OPEN no matter which opened mode is used. In that case all DOM+LAYOUT (especially LAYOUT) are compatible and will be taken most likely and READ can be done as well. If client needs WRITE lock it will be issues separately when needed.

Conclusion

From all above it can be said that while strategy #2 works better for getting DOM locks, it is not suitable for data prefetch. Therefore the preferable way is to stay with strategy #1 though it provides less benefits. Strategy #2 could be used if it known that READ prefetch is not needed, e.g. it is disabled as feature on MDT or if data doesn't fit into reply buffer. The mdt_object_open_lock() may implement both strategies and choose one or another depending on use case and known data. The default strategy is #1 when read_prefetch is ON, and strategy #2 is used otherwise. It worth to keep both strategies for some time at least to try them on real setups and application workloads. One of another strategy could be selected by set_param.

If there will be no significant differences, then #1 can be kept as the only strategy for simplicity, otherwise if strategy #2 has real benefits, it can use always PR mode for READ prefetch or check on server if client has LAYOUT lock.

Remarks

1. Andreas Dilger: This happens with Fortran applications. They always open files with RW mode.

2. Jinshan Xiong: If we add a restriction that the client has to acquire DoM lock before migrating data then we could probably release limitation, just to support fetch data_version with WRITE mode from DoM component.

3. Jinshan Xiong: Please notice that we should use bulk to transfer data back for large data.