14-Oct-2021 - Google Jamboard

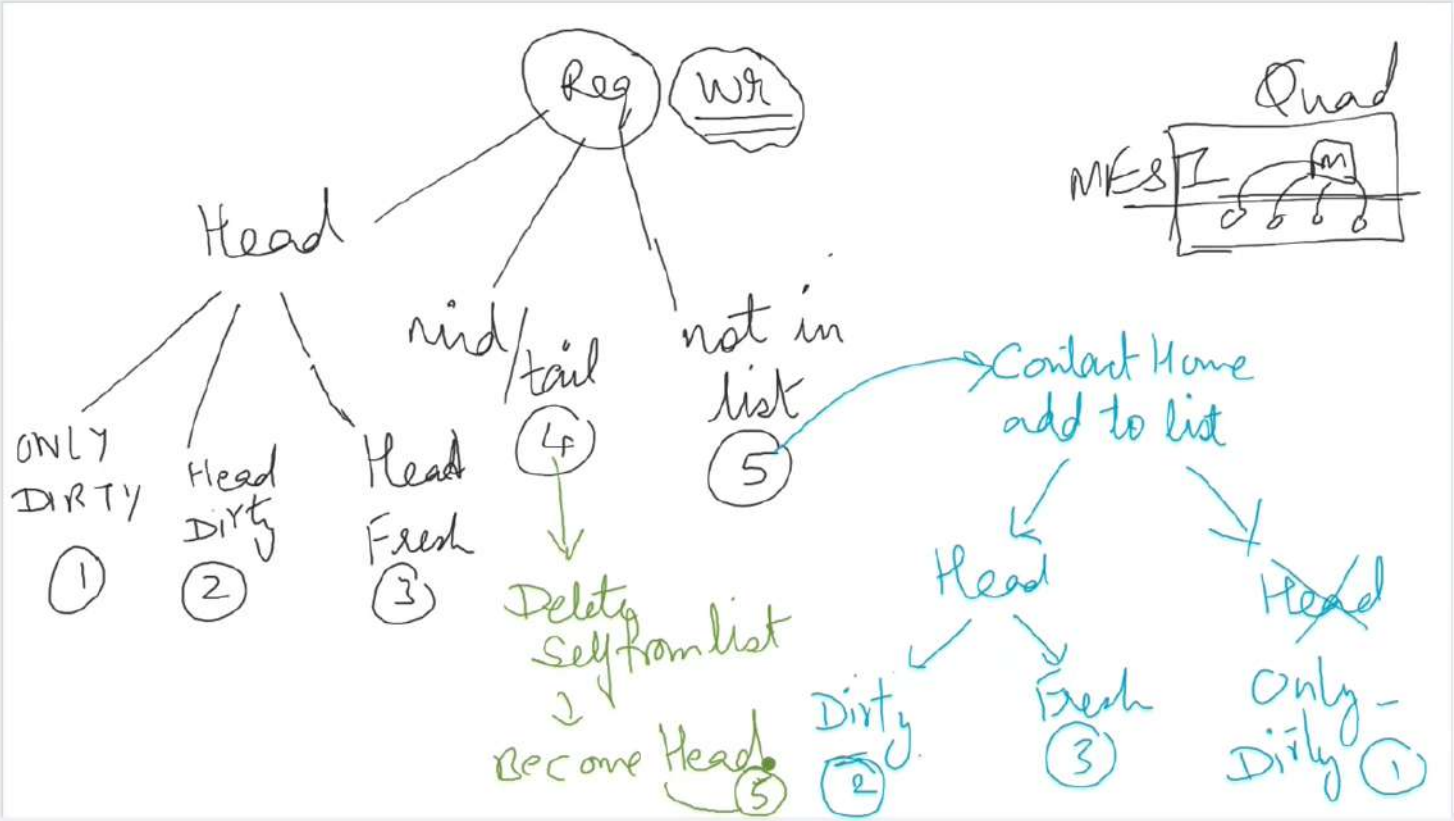
jamboard.google.com/d/1UiwQG_BvZIIGQyFfOoSRqaRIsWEb6KuLtlXtVHfRuS0/viewer?f=4

14-Oct-2021

5/5

Share

Set background | Clear frame



Syam Sankar | NALABOLU SAN... | SWATI UPADHYAY | IMIJUNGLA LON... | ADITYA KUMAR S... | SARASWATULA P... | YOGESH KUMAR | GALI JAYA PRAKA... | TANVISH | Hemangee Kalpe...

77% (2:10) Sat, October 16, 11:10

14-Oct-2021 - Google Jamboard

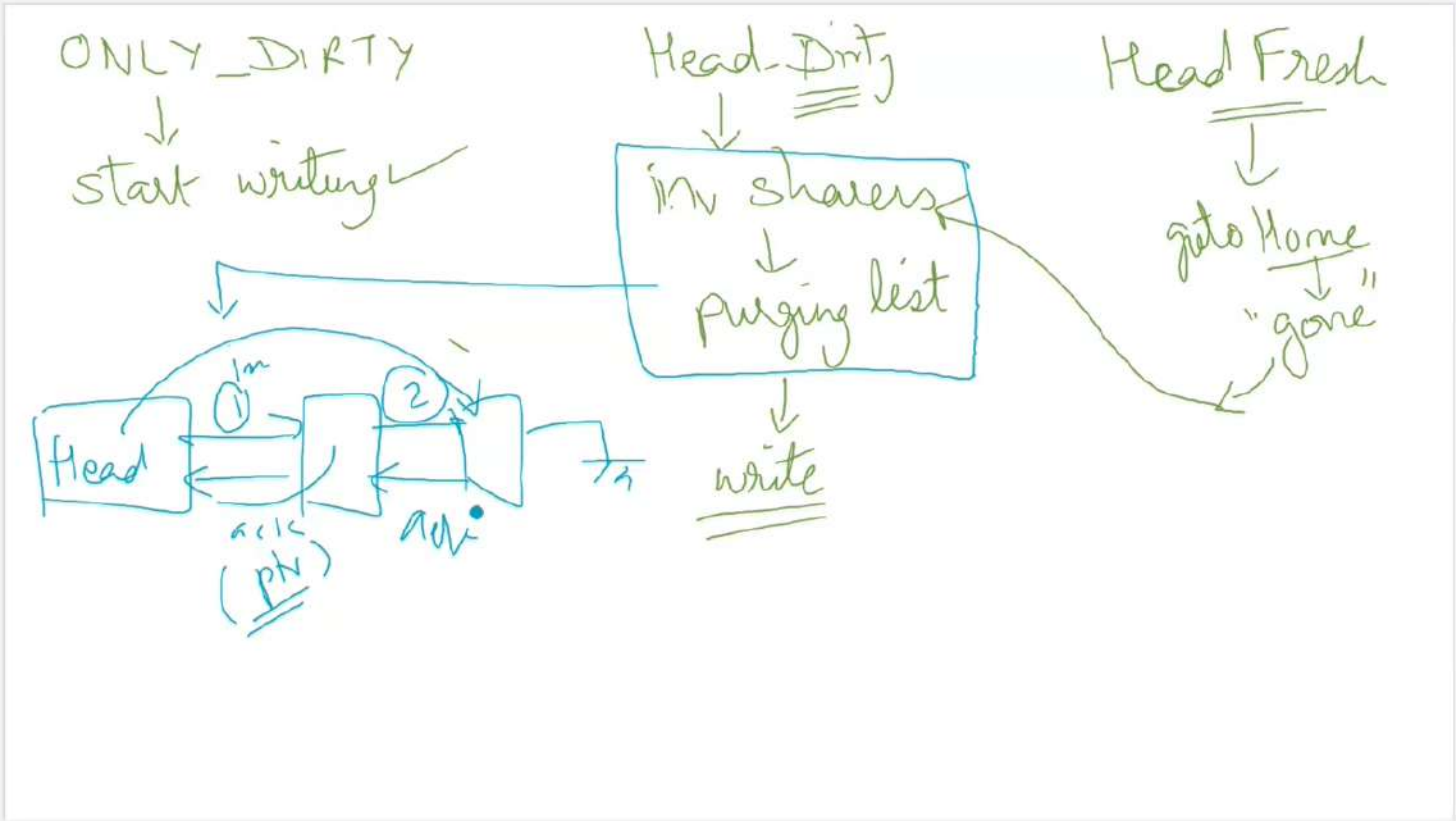
jamboard.google.com/d/1UiwQG_BvZIIGQyFfOoSRqaRIsWEb6KuLtlXtVHfRuS0/viewer?f=5

14-Oct-2021

6/6

Share

Set background | Clear frame



Syam Sankar
NALABOLU SAN...
SWATI UPADHYAY
IMUJUNGLA LON...
ADITYA KUMAR S...
SARASWATULA P...
YOGESH KUMAR
GALI JAYA PRAKA...
TANVISH
RAHUL KUMAR
VARHADE AMEY...
Hemangee Kalpe...

75% (2:16) Sat, October 16, 11:14

Handling Write Requests

- Requestor has following possibilities
  - Is at Head of sharing list: (1) ONLY_DIRTY, (2) HEAD_DIRTY, (3) HEAD_FRESH
  - (4) Not in sharing list, (5) In sharing list but not at Head
- (1) ONLY_DIRTY : OK can start writing immediately
- (2) HEAD_DIRTY and is at Head of sharing list. Sequentially invalidate all sharers then write
- (3) HEAD_FRESH
  - Request sent to home to change state to GONE
  - Make state to HEAD_DIRTY. Then as above
- (4) Not in sharing list
  - Allocate entry in (local) cache
  - Become Head by communicating with Home. Then do as above
- (5) In sharing list, but not Head
  - Remove self from list. Request/Response with neighbours (details later)
  - Add self to Head (as above). Then do as above

Purging the list

- Block is HEAD_DIRTY: Purge the sharing list
- Purge in serialised manner
- Send inv request to next-node, which rolls itself out from the list and sends back to head the next-pointer
- The head sends similar request to this next pointer and so on ...
- Head node stays in pending state till purging completes. Therefore new attempts to add to the list go in the pending list
- OPTION: to reduce the number of network transactions in the cirtical path, each node can forward the inv to next-node and ACK the previous
- Correct. But not part of SCI standard. This way it distributes the state of invalidation progress and hence complicates the protocol-level recovery from errors

Purging the list

- Writer is HEAD_FRESH
  - Goes to pending state ; goes to Home; Home changes from FRESH to GONE adn replies to writer
  - Writes goes to another pending state and start purging of the list
- Race condition:
- HEAD_FRESH goes to home to request change state to GONE
- But finds home state != FRESH but points to a newly queued node (which got there in the meantime) and this new node is about to contact the writer
- Home sends some kind of NACK to writer
- Writer will receive request from new-node to link with it
- The writer will then delete itself from list and later try to re-attach at Head to request write-operation

14-Oct-2021 - Google Jamboard

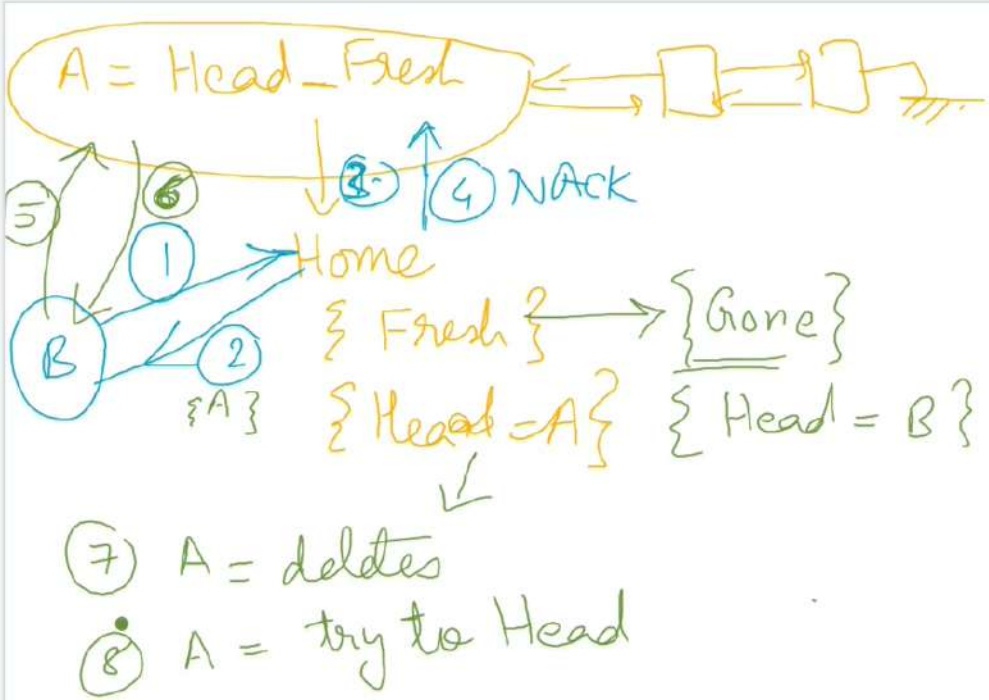
jamboard.google.com/d/1UiwQG_BvZIIGQyFfOoSRqaRIsWEb6KuLtlXtVHfRuS0/viewer?f=6

14-Oct-2021

7/7

Share

Set background | Clear frame



Syam Sarkar
NALABOLU SAN...
SWATI UPADHYAY
ADITYA KUMAR S...
SARASWATULA P...
YOGESH KUMAR
GALI JAYA PRAKA...
TANVISH
RAHUL KUMAR
VARHADE AMEY...
AMIT
IMLIJUNGLA LON...
Hemangee Kalpe...

70% (2:31) Sat, October 16, 11:25

Wr-backs/replacement: Mid

- Updating memory with data + delete self from sharing list (=roll-out) : Mid/Tail - or - Head
- MID
  - Set state to pending
  - Send request to neighbours to update pointers
  - What if neighbour is pending?
    - i.e. trying to delete itself => priority given to tail-side node
  - The block is marked “invalid” in the cache
  - Other nodes in the list do not have any change. Except if there are 2 nodes: then the head becomes ONLY_DIRTY or ONLY_FRESH as appropriate

14-Oct-2021 - Google Jamboard

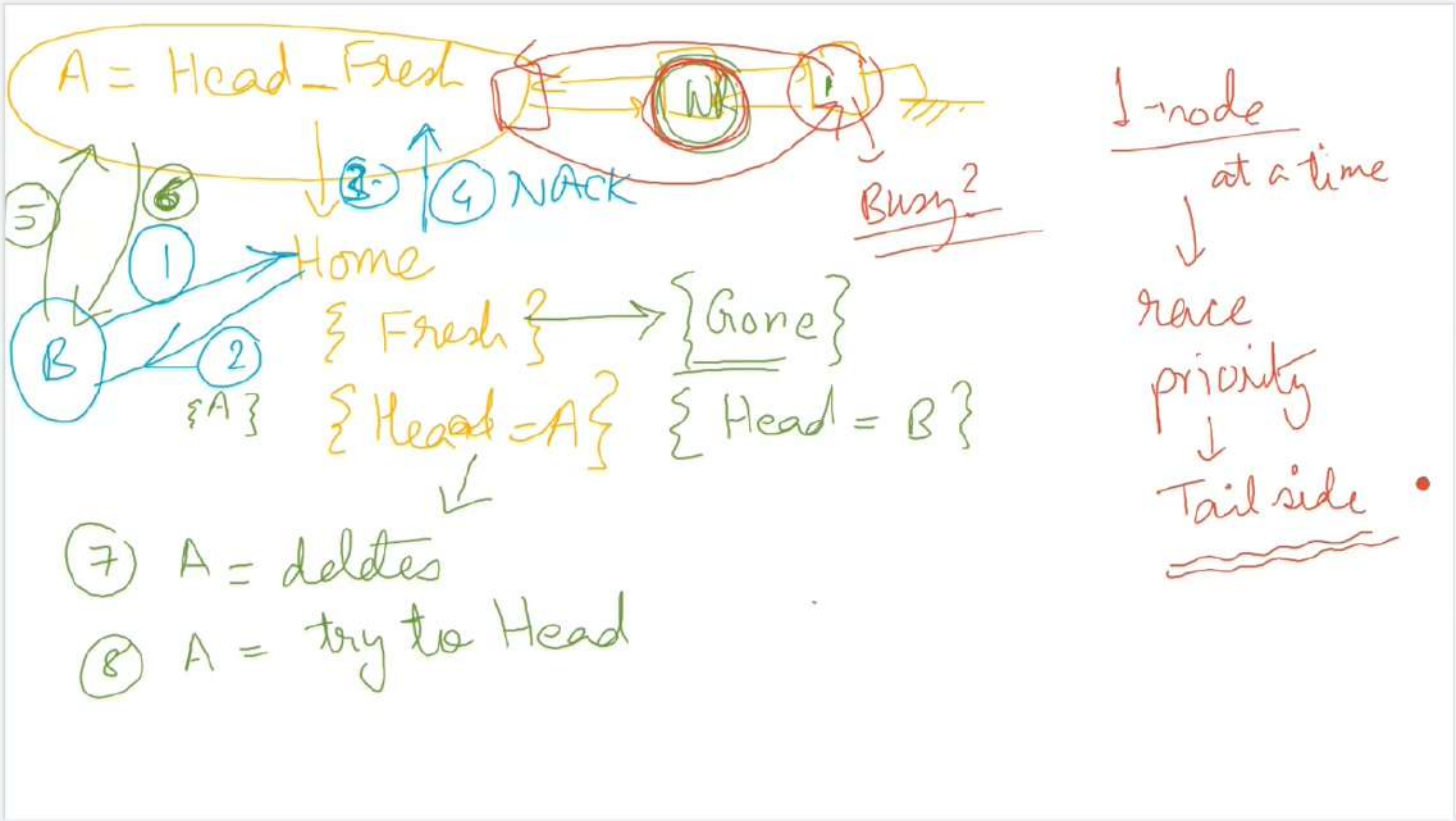
jamboard.google.com/d/1UiwQG_BvZIIGQyFfOoSRqaRIsWEb6KuLtlXtVHfRuS0/viewer?f=6

14-Oct-2021

7 / 7

Share

Set background | Clear frame



Syam Sankar
NALABOLU SAN...
SWATI UPADHYAY
ADITYA KUMAR S...
SARASWATULA P...
YOGESH KUMAR
TANVISH
RAHUL KUMAR
VARHADE AMEY...
AMIT
IMLIJUNGLA LON...
Hemangee Kalpe...

68% (1:57) Sat, October 16, 11:28

write-back/replacement: Head

- Entry dirty then write-back / Entry is clean then replacement
- Head puts itself in pending state and send message to downstream node
- Downstream node puts backward pointer to Home node and changes state appropriately and replies to replacer
  - TAIL_VALID => HEAD_DIRTY
  - MID_VALID => HEAD_DIRTY
  - MID_FRESH => HEAD_FRESH (depends on state of original Head node)
- Replacer (old head) sends transaction to Home telling to link to new Head
  - Home updates head pointer to the new Head
  - State change is not needed at Home node
  - Home replies to replacer
- After receiving reply form Home, the replacer sets its block to Invalid state
- If replacer is the **only node** in the list
  - It only has to communicate with Home node

14-Oct-2021 - Google Jamboard

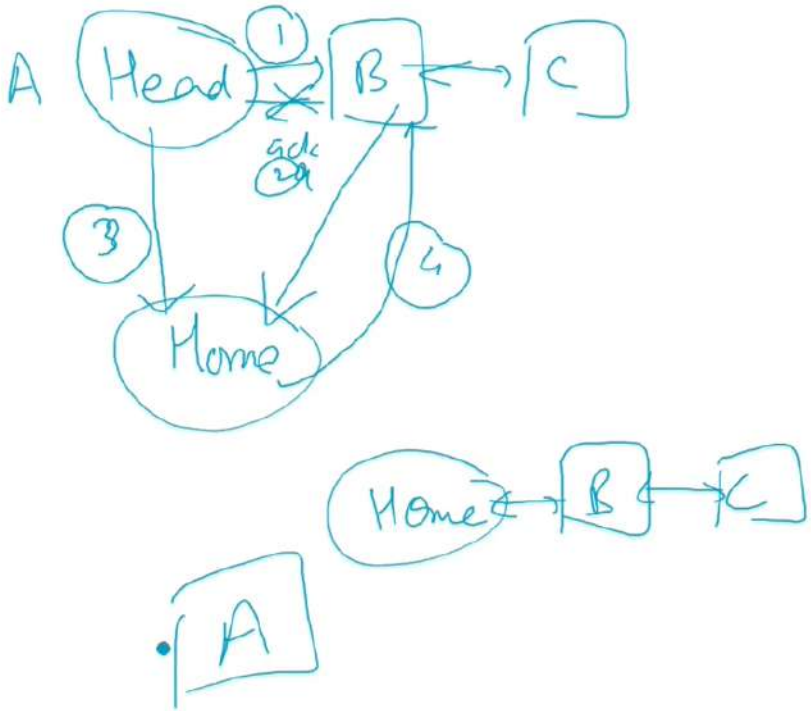
jamboard.google.com/d/1UiwQG_BvZIIGQyFfOoSRqaRIsWEb6KuLtlXtVHfRuS0/viewer?f=7

14-Oct-2021

8 / 8

Share

Set background | Clear frame

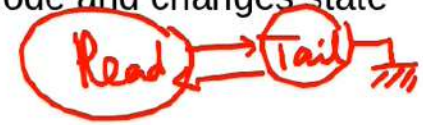


Syam Sankar
NALABOLU SAN...
SWATI UPADHYAY
ADITYA KUMAR S...
SARASWATULA P...
YOGESH KUMAR
TANVISH
RAHUL KUMAR
VARHADE AMEY...
AMIT
IMLIJUNGLA LON...
Hemangee Kalpe...

65% (1:53) Sat, October 16, 11:34

write-back/replacement: Head

- Entry dirty then write-back / Entry is clean then replacement
- Head puts itself in pending state and send message to downstream node
- Downstream node puts backward pointer to Home node and changes state appropriately and replies to replacer
  - TAIL_VALID => HEAD_DIRTY
  - MID_VALID => HEAD_DIRTY
  - MID_FRESH => HEAD_FRESH (depends on state of original Head node)
- Replacer (old head) sends transaction to Home telling to link to new Head
  - Home updates head pointer to the new Head
  - State change is not needed at Home node
  - Home replies to replacer
- After receiving reply form Home, the replacer sets its block to Invalid state
- If replacer is the **only node** in the list
  - It only has to communicate with Home node



Race condition

- When it reaches home-node, home state has changed and therefore home send response like a NACK
- Eventually Head will get the request from the new-node which tries to link with head
- Head then responds to new-head by linking it with next neighbour and thus deletes itself

- To serve write-miss: replace old block + load new
  - In bus based load new first and put old in wr-buffer
  - In SCI due to sharing-list roll-out first then load new-block, as block being deleted has data and also pointers to nodes in the sharing-list

14-Oct-2021 - Google Jamboard

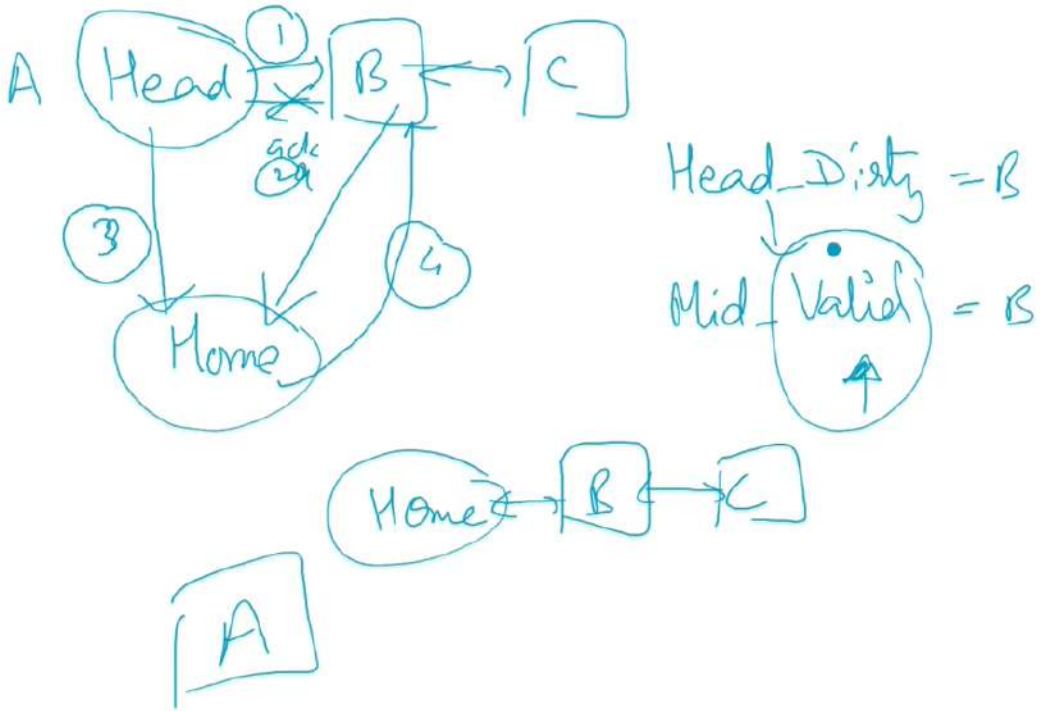
jamboard.google.com/d/1UiwQG_BvZIIGQyFfOoSRqaRIsWEb6KuLtlXtVHfRuS0/viewer?f=7

14-Oct-2021

8 / 8

Share

Set background | Clear frame



Syam Sankar
NALABOLU SAN...
SWATI UPADHYAY
ADITYA KUMAR S...
SARASWATULA P...
YOGESH KUMAR
TANVISH
RAHUL KUMAR
VARHADE AMEY...
AMIT
IMLIJUNGLA LON...
Hemangee Kalpe...

63% (2:01) Sat, October 16, 11:38

14-Oct-2021 - Google Jamboard

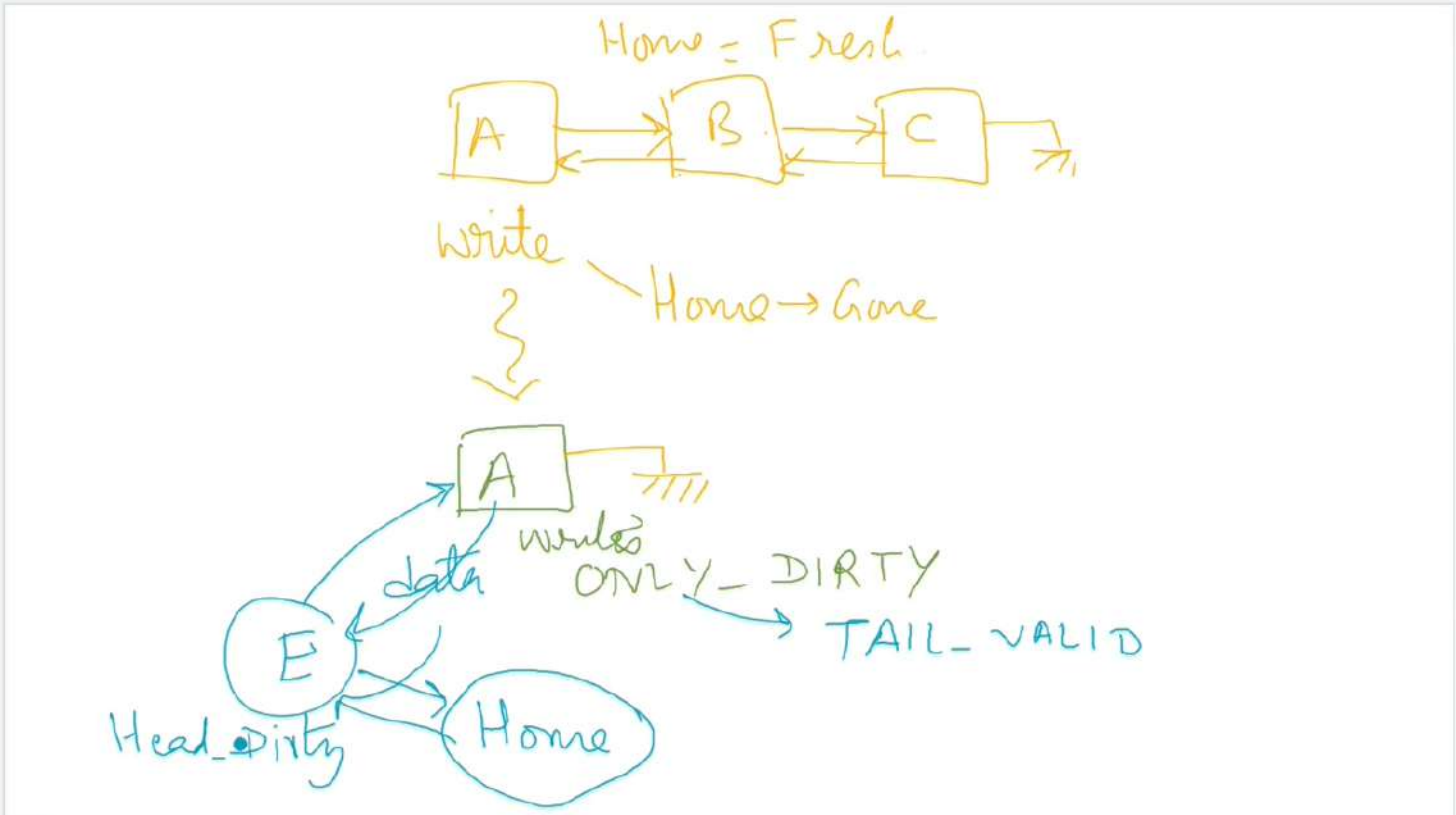
jamboard.google.com/d/1UiwQG_BvZIIGQyFfOoSRqaRIsWEb6KuLtlXtVHfRuS0/viewer?f=8

14-Oct-2021

9/9

Share

Set background | Clear frame



TANVISH
Syam Sankar
NALABOLU SAN...
SWATI UPADHYAY
ADITYA KUMAR S...
SARASWATULA P...
YOGESH KUMAR
VARHADE AMEY...
AMIT
IMLIJUNGLA LON...
RATHOD SAINATH
Hemangee Kalpe...

60% (2:04) Sat, October 16, 11:45

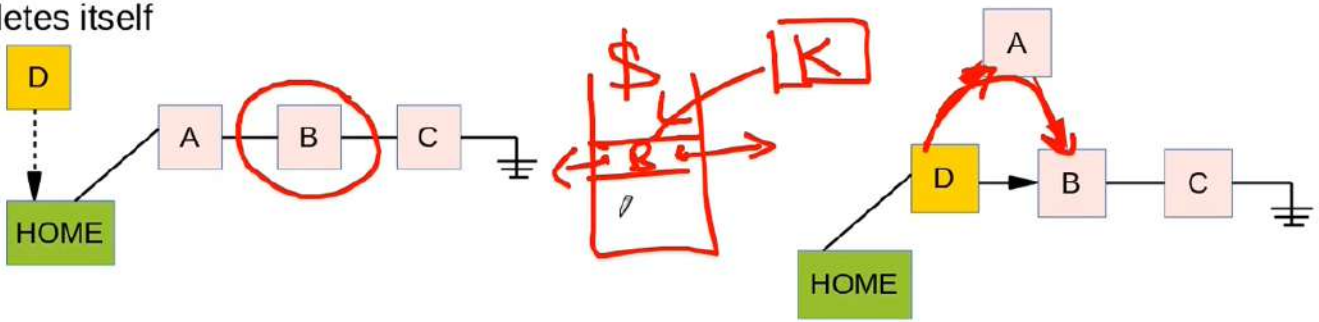
Race condition

- When it reaches home-node, home state has changed and therefore home send response like a NACK
- Eventually Head will get the request from the new-node which tries to link with head
- Head then responds to new-head by linking it with next neighbour and thus deletes itself

- To serve write-miss: replace old block + load new
  - In bus based load new first and put old in wr-buffer
  - In SCI due to sharing-list roll-out first then load new-block, as block being deleted has data and also pointers to nodes in the sharing-list

Race condition

- When it reaches home-node, home state has changed and therefore home send response like a NACK
- Eventually Head will get the request from the new-node which tries to link with head
- Head then responds to new-head by linking it with next neighbour and thus deletes itself



- To serve write-miss: replace old block + load new
  - In bus based load new first and put old in wr-buffer
  - In SCI due to sharing-list roll-out first then load new-block, as block being deleted has data and also pointers to nodes in the sharing-list

Serialisation

- Home node determines the order in which cache misses to a block are serialised
- There is no busy-state at home
- Home accepts each request and either
  - It serves itself ... Satisfies request wholly by itself, or
  - Directs to the node that it sees as Head: either true-head or pending-head
- Before re-directing request, it changes head pointer to new requestor. So that all subsequent requests will go to the new-head (or pending head!)
- If a request is not satisfied, the node remains in the pending list and awaits its turn
- Nodes in pending list obtain access to the block in FIFO order, ensuring that the order in which they complete is indeed the same as that in which they first reached the home
- In case Home NACKs a request, this request never succeeds in the current form and will send a new request. Old request is cancelled and new request is serialised

Deadlock, Livelock, Starvation

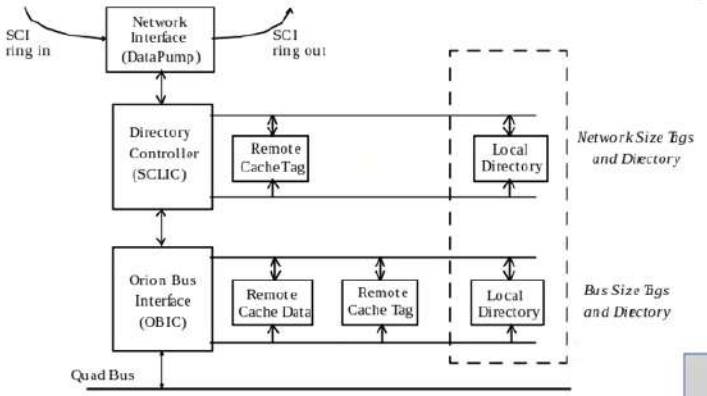
- Distributed pending list holds waiting requests at the requesting cache itself
  - No separate hardware queue at home node
  - No limited buffer

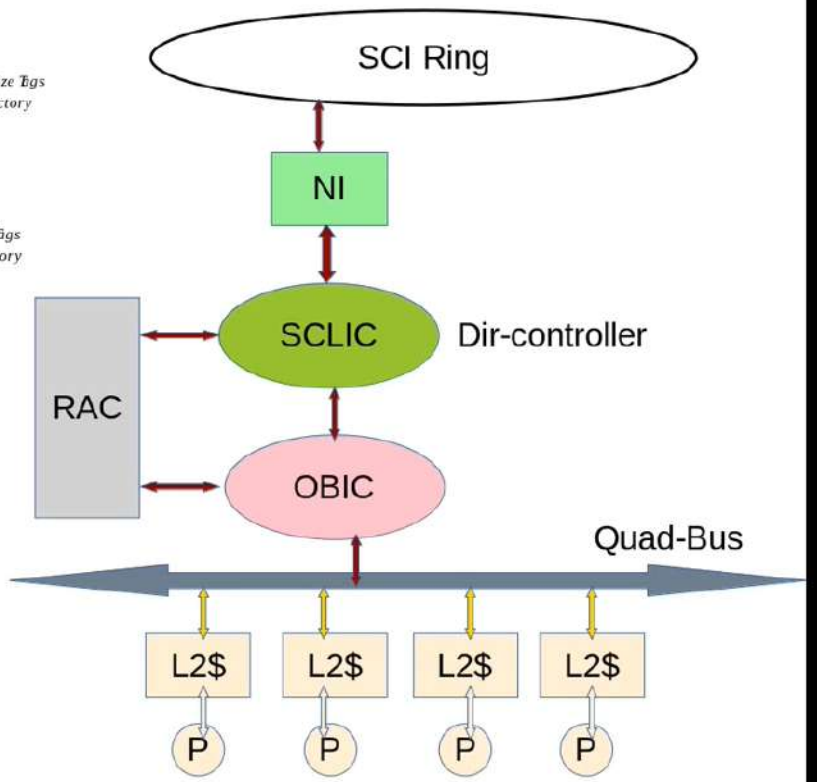
    => No deadlock
- Strict req-response
  - As no NACK there is No contention due to race condition. They simply join the pending queue and all requests make progress

    => No Livelock
- List mechanism ensures that request are ordered in FIFO order (as they reach home)

  => No Starvation
- A node can be part of number of pending lists. This number is equal to the number of outstanding requests. The space for the pointers is already with the cache, hence no extra storage overheads.
  - Pending node cannot replace until it becomes normal node

Protocol Interactions with SMP-node





- How to combine the RAC-level SCI protocol with intra-cluster protocol (on the Quad bus)

Interactions for Read-request

- Processor level read-miss in its L2-cache goes on Quad-bus
- All caches snoop bus and OBIC bus controller also snoops Quad-bus
- If request can be satisfied by RAC, i.e. RAC has the cache block and Dir-state is OK, then serve request locally
- The local memory or another local-cache inside the cluster can also serve the request.
  - There is provision to wait for snoop-result delays. Snoop results come after fixed delay of 4 cycles. If not ready then asserts a stall of another 2 cycles and this wait of 2 cycles continues until results are ready.
  - All local caches follow MESI
  - Quad-bus = in-order response

Interactions for Read-request

- If request must be propagated off-node:
  - OBIC detects and intervenes by sending a "deferred response" signal on Quad bus
  - Upon which Quad-bus leaves the request-pending and leaves in-order response for this request and starts serving new-(internal)-requests
- OBIC passes request to SCLIC to start the directory protocol
- When remote response comes back, SCLIC transfers it to OBIC
  - Then OBIC places response on Quad-bus and completes the deferred transaction
- NOTE: to have a bus-system to become part of a larger system it must be split-transaction-bus for performance + correctness
- Else
  - Bus held up for entire duration of remote transaction
  - Therefore not allowing local misses and not serving incoming requests
    - Which may result in potential deadlock

Interactions for Write-request

- Same set of actions as read
- OBIC snoops the state of block in the RAC
  - Block is owned by local quad or
  - Request must be sent outside
- Node has to be put at head of sharing list, all nodes invalidated and then the write can be performed. All this is taken care by the SCLIC
- When SCLIC is done, it puts the response on the quad bus (via the OBIC) which completes the operation

Interactions for Write-request

- Limitations
- A locally allocated block, i.e. its address maps to local memory and it is cached remotely and modified there
- When we bring this block to cluster to serve read/write-miss, the response is put on the bus.
  - Here, the block must also be updated in local memory
- But local memory design cannot accept data from deferred transactions, as it is not designed that way to understand deferred responses
  - OBIC has to use special action to update local memory before releasing the bus
- If two read-exclusive request for same block come on Quad-bus back-to-back
  - One goes to SCLIC and other is NACKed .... it must retry
  - Ideally the second request should have been buffered and the response of the first could have been used by the second request (in an appropriate state)

Serialisation

- (1) For locally allocated blocks
- Earlier we said serialisation is done by home-node in the order of requests accepted
- Therefore home has to serialise incoming transactions and also transactions from local processor on Quad-bus
- e.g. home-dir-state = HOME
  - No remote node has cached the block, BUT the block may be cached and even modified in some local-cache (inside the Quad)
  - To serve an incoming request for a block that is dirty in local cache we need to go on the Quad bus and cannot be handled entirely by SCLIC and OBIC
  - Similarly, an incoming request that makes home state (Home or FRESH) => GONE must be put on the Quad bus so that copies in the processor caches are invalidated
  - Since both incoming requests and local misses to data at the home appear on the quad bus, it is natural to let this bus be the actual serialising agent
  - **The Quad bus at the Home node is the serialising agent**

14-Oct-2021 - Google Jamboard

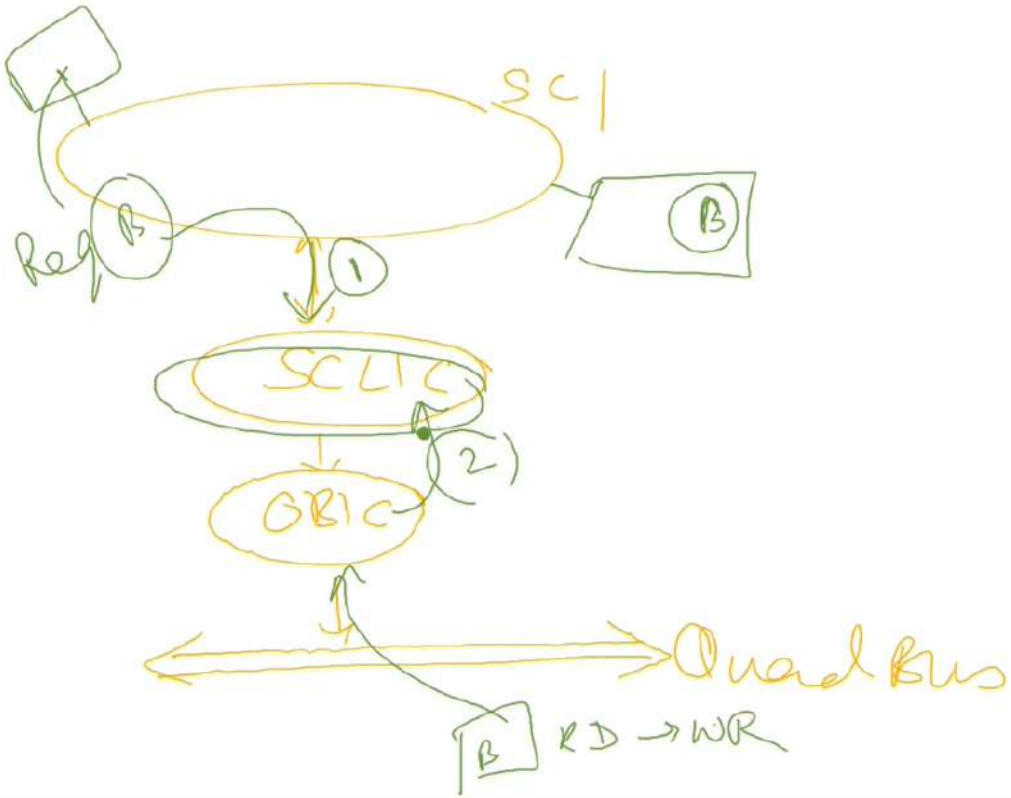
jamboard.google.com/d/1UiwQG_BvZIIGQyFfOoSRqaRIsWEb6KuLtlXtVHfRuS0/viewer?f=9

14-Oct-2021

10 / 10

Share

Set background | Clear frame



TANVISH
Syam Sankar
SU
SWATI UPADHYAY
AS
ADITYA KUMAR S...
YK
YOGESH KUMAR
IMIJUNGLA LON...
RATHOD SAINATH
VA
VARHADE AMEY...
SK
SHIVANGI KUMAR
NS
NALABOLU SAN...
Hemangee Kalpe...

45% (1:25) Sat, October 16, 12:17

Serialisation

- (2) Issues of serialisation of the requesting quad for remotely allocated blocks
- Activities within a Quad relating to remotely allocated blocks are serialised at the local SCLIC and not at the bus
- (i) Request from local processor for a block in the remote cache AND (ii) incoming requests from the SCI interconnect for the same block are serialised at local SCLIC
- Also, SCLIC (locally) serialises outstanding invalidations at a requestor and new incoming requests

14-Oct-2021 - Google Jamboard

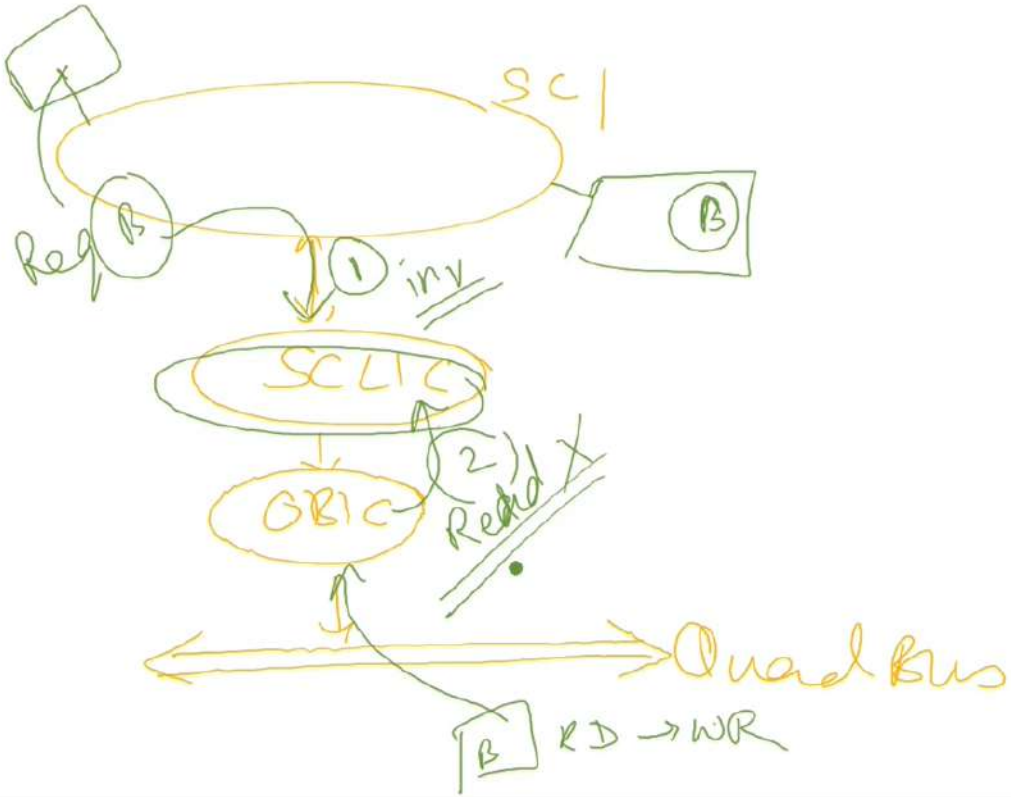
jamboard.google.com/d/1UiwQG_BvZIIGQyFfOoSRqaRIsWEb6KuLtlXtVHfRuS0/viewer?f=9

14-Oct-2021

10 / 10

Share

Set background | Clear frame



TANVISH
Syam Sankar
SU
SWATI UPADHYAY
AS
ADITYA KUMAR S...
YK
YOGESH KUMAR
IMIJUNGLA LON...
RATHOD SAINATH
VA
VARHADE AMEY...
SK
SHIVANGI KUMAR
NS
NALABOLU SAN...
Hemangee Kalpe...

45% (1:25) Sat, October 16, 12:17

14-Oct-2021 - Google Jamboard

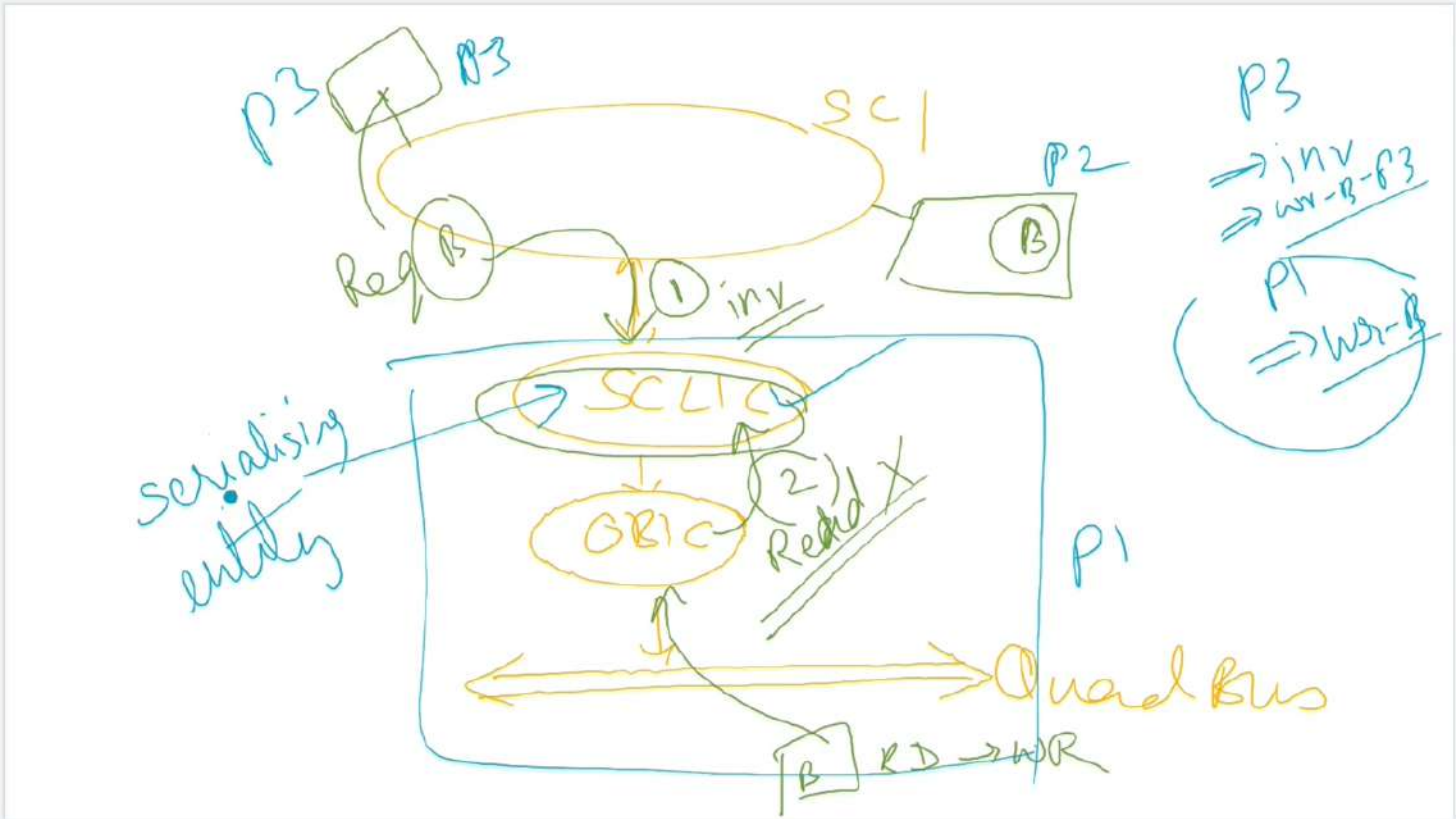
jamboard.google.com/d/1UiwQG_BvZIIGQyFfOoSRqaRIsWEb6KuLtlXtVHfRuS0/viewer?f=9

14-Oct-2021

10 / 10

Share

Set background | Clear frame



TANVISH
Syam Sankar
SWATI UPADHYAY
ADITYA KUMAR S...
YOGESH KUMAR
IMIJUNGLA LON...
RATHOD SAINATH
VARHADE AMEY...
SHIVANGI KUMAR
NALABOLU SAN...
Hemangee Kalpe...

44% (1:26) Sat, October 16, 12:19