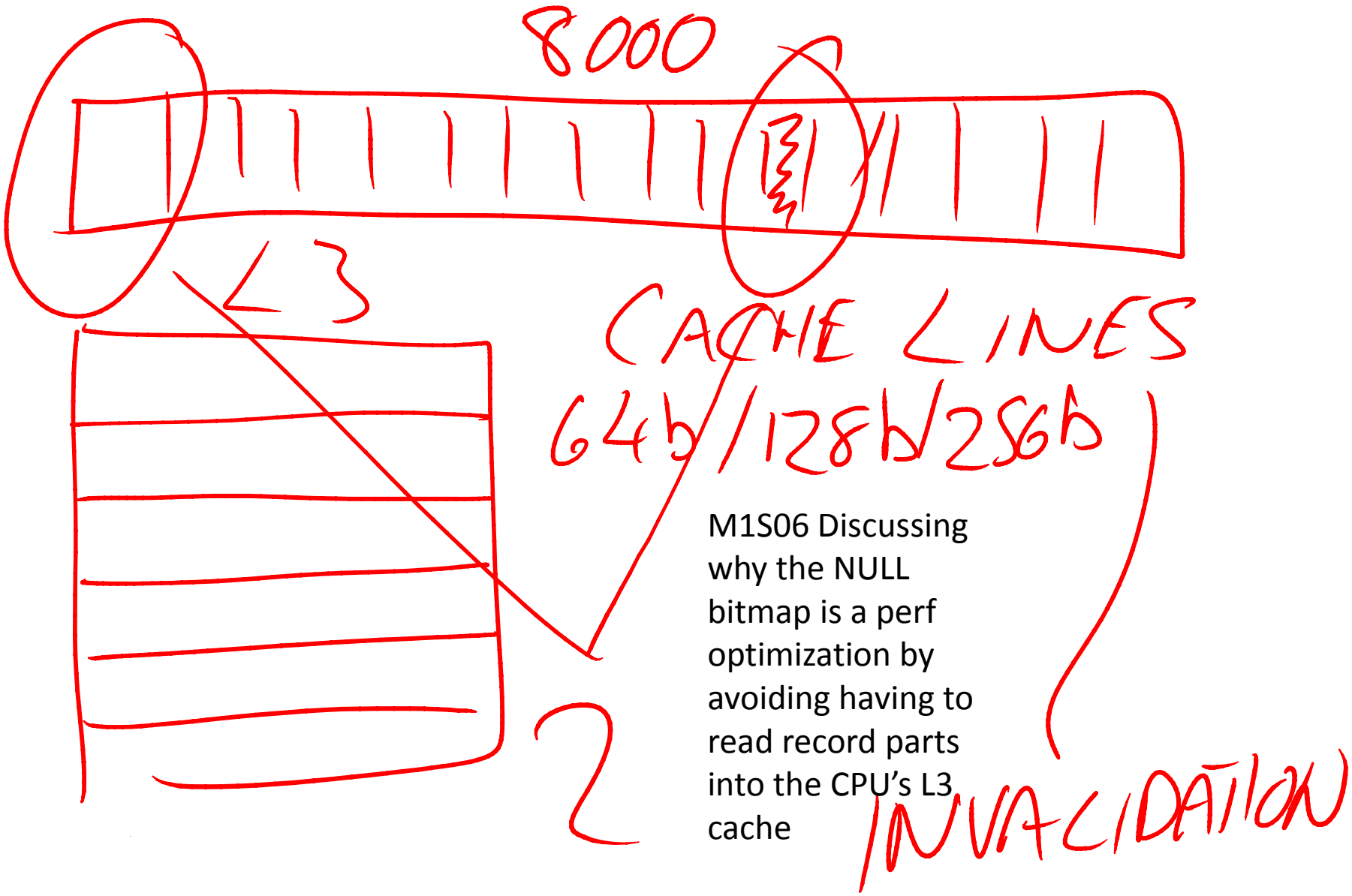
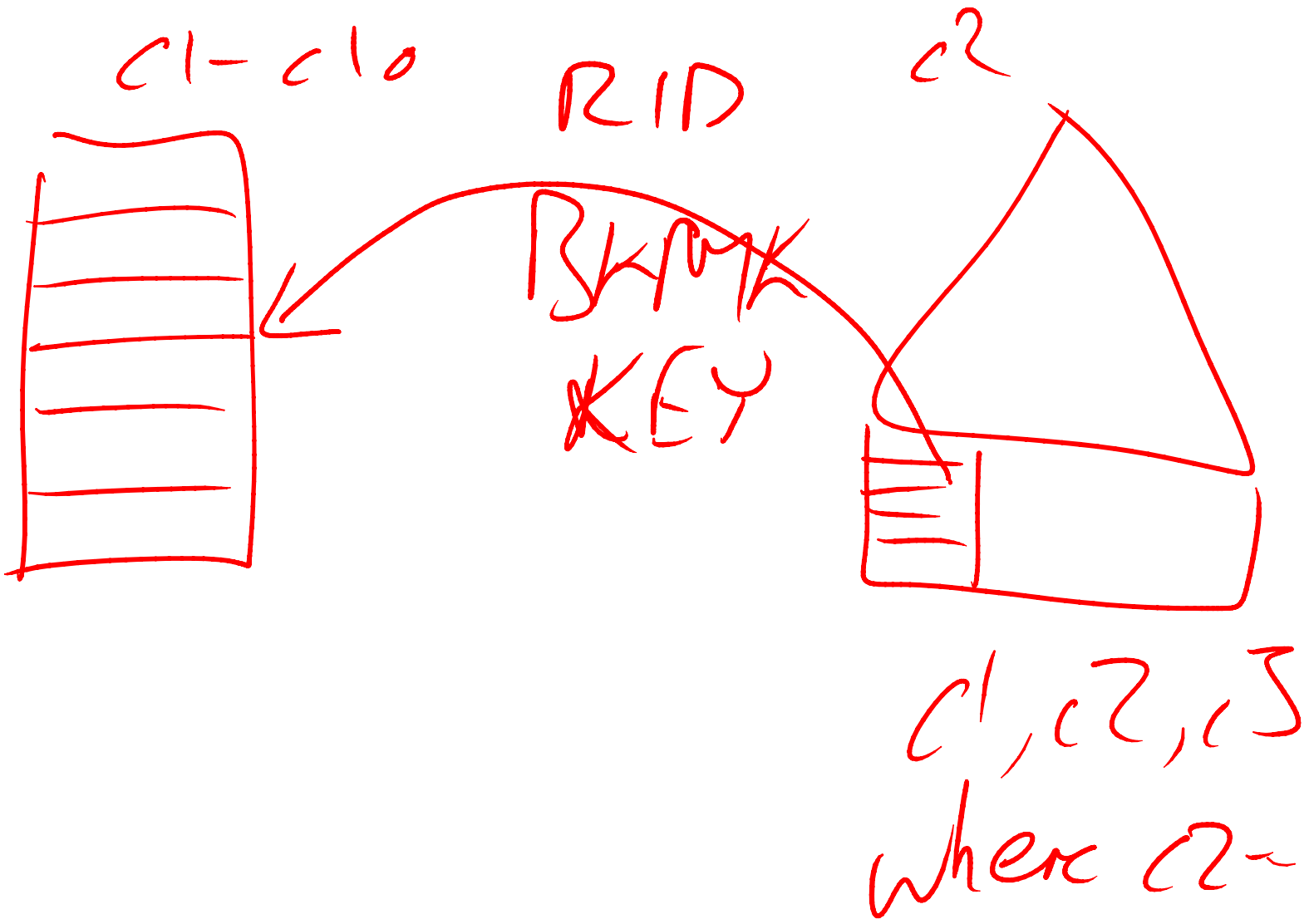
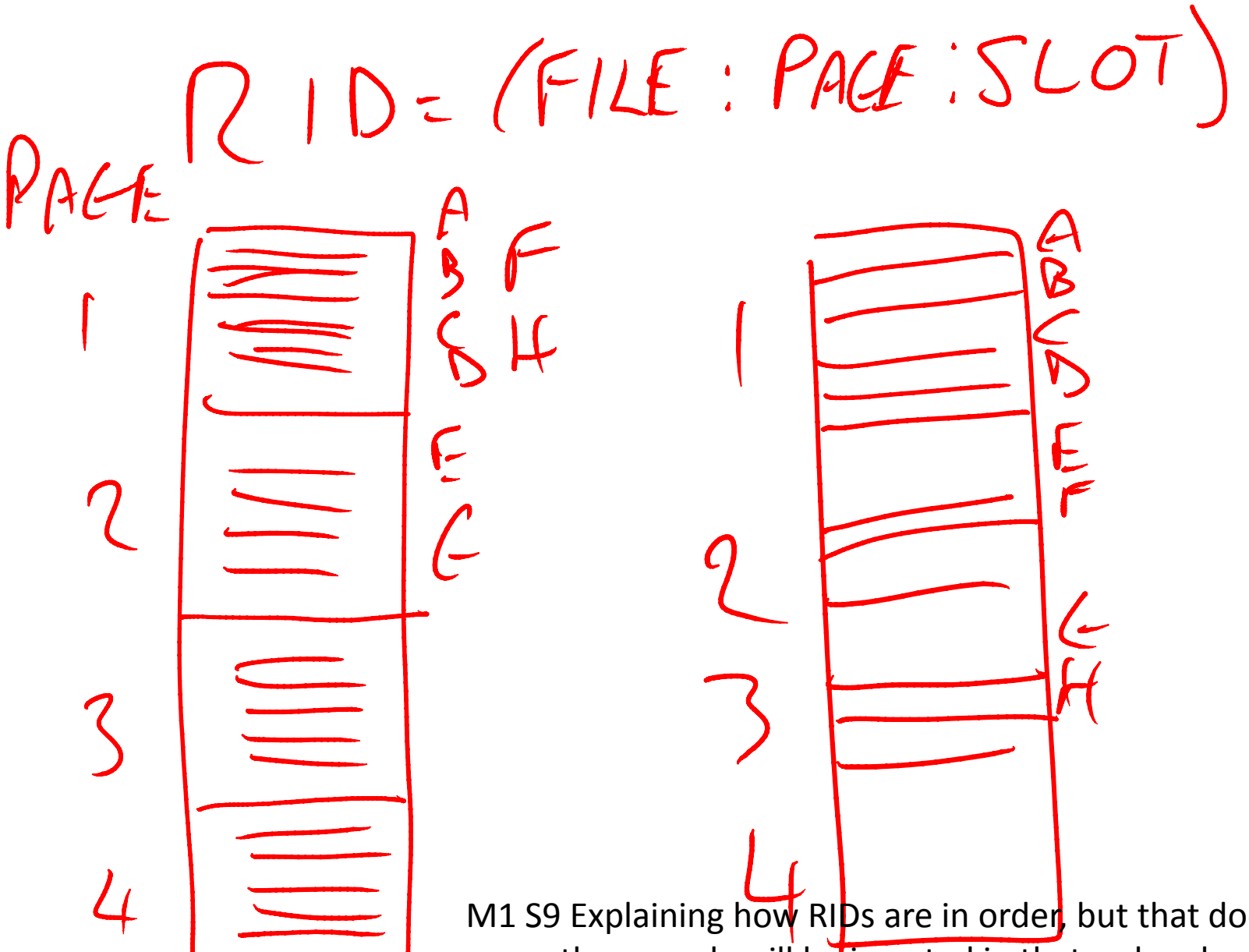

#splhelp



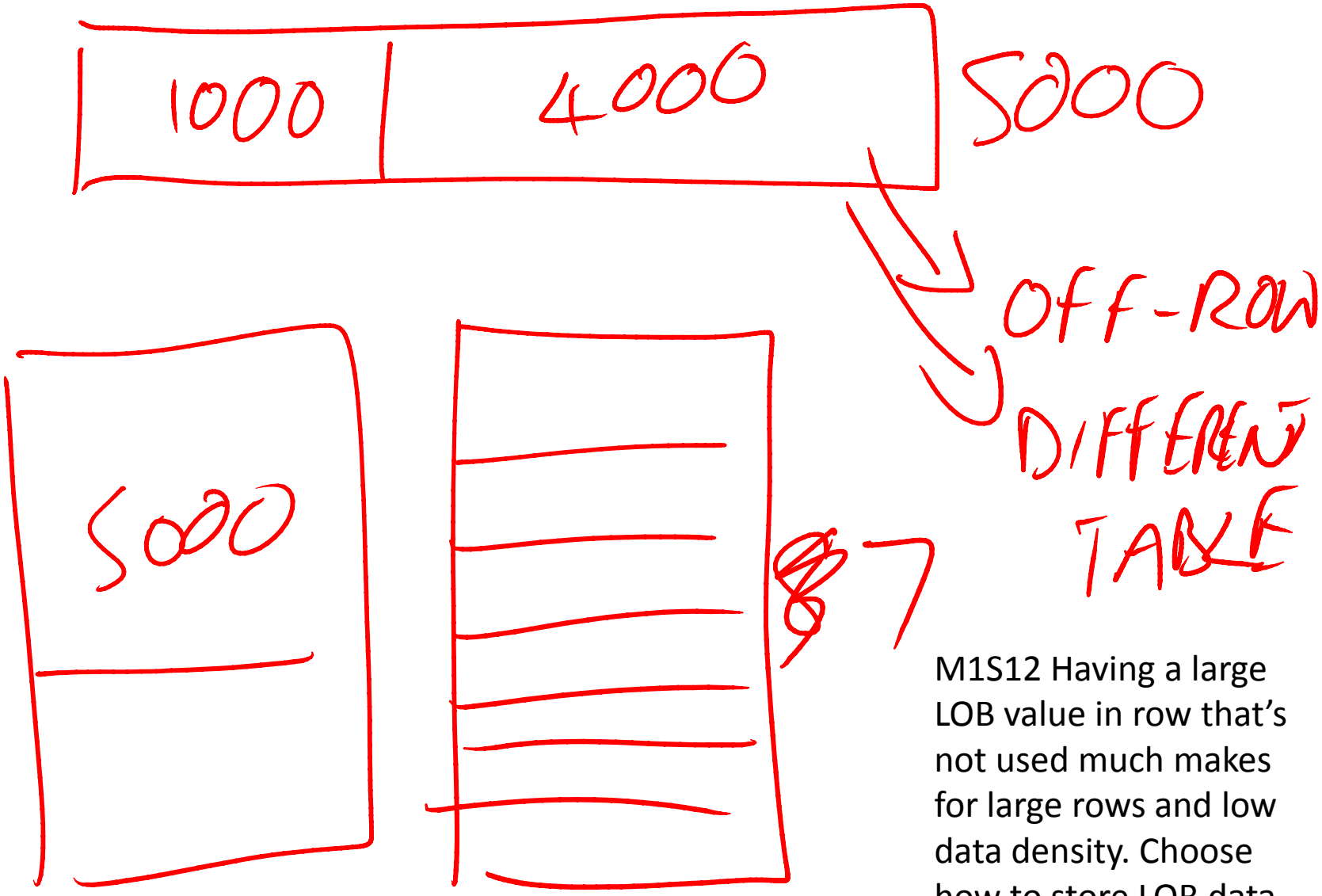
M1S06 Discussing why the NULL bitmap is a perf optimization by avoiding having to read record parts into the CPU's L3 cache



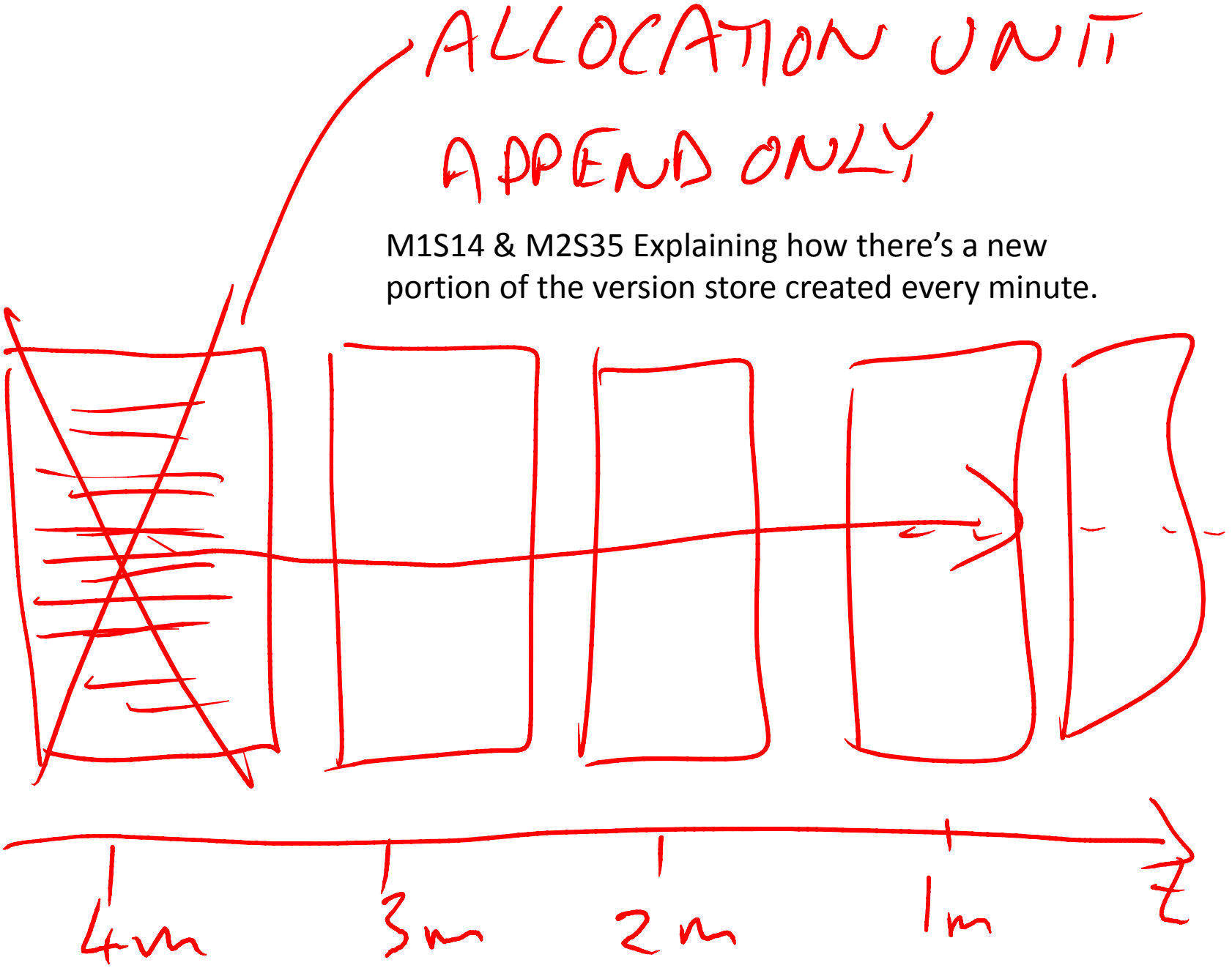
M1S09 Explaining NC index back-links to the data records as part of describing why forwarding/forwarded records exist. Without them, moving a heap row would entail changing all NC index records with a back-link to that row.



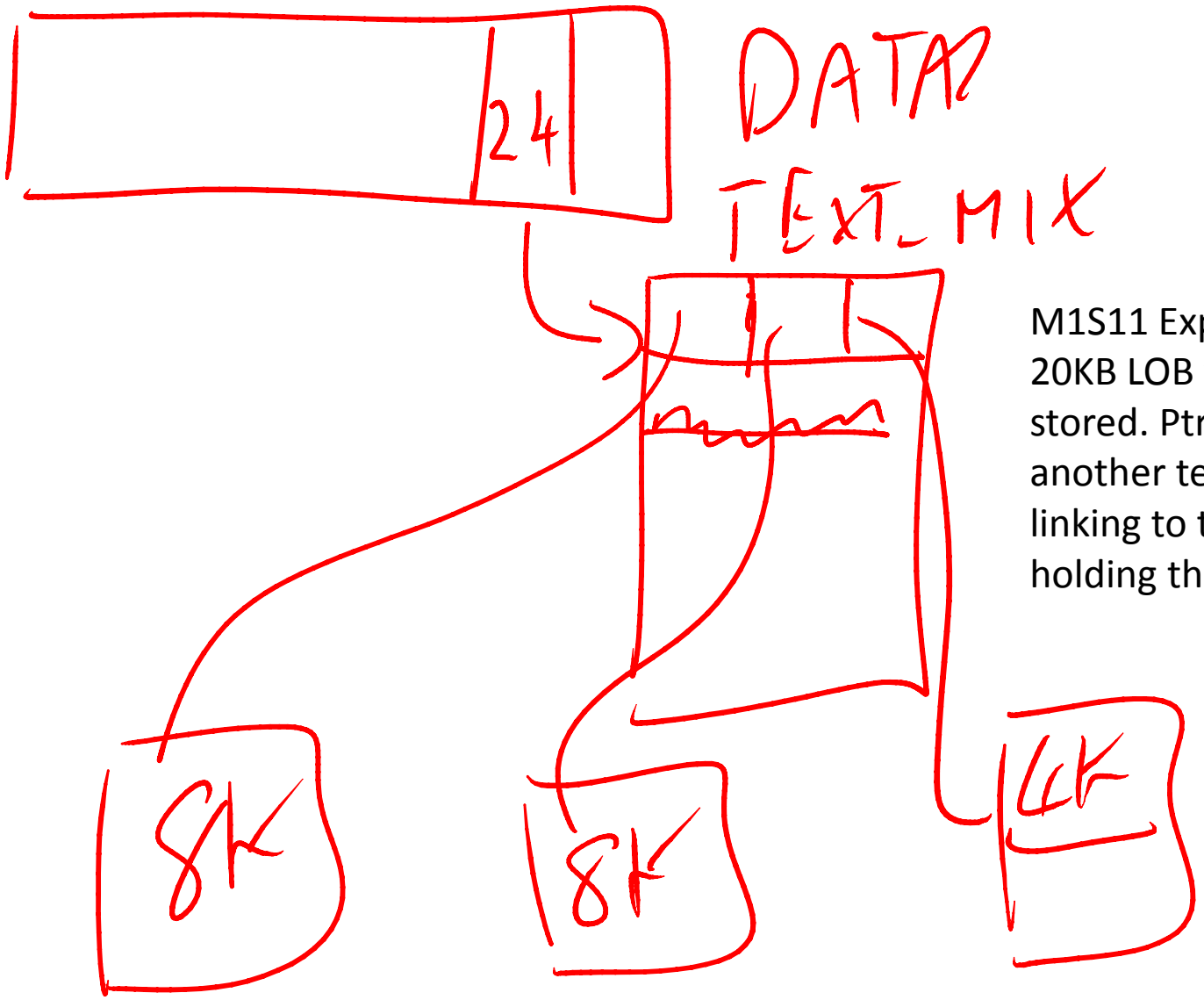
M1 S9 Explaining how RIDs are in order, but that doesn't mean the records will be inserted in that order, depending on their size and free-space searches.



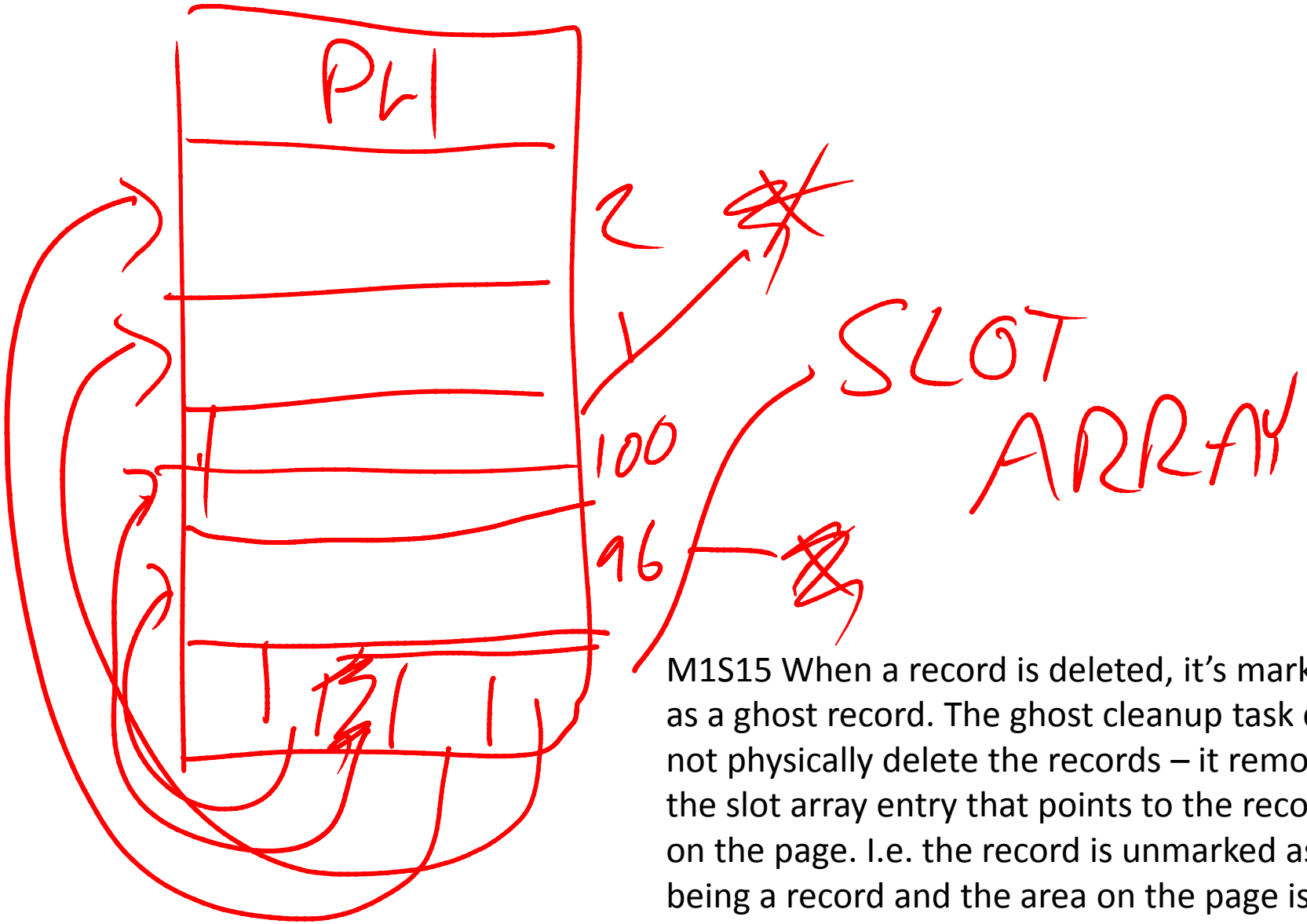
M1S12 Having a large LOB value in row that's not used much makes for large rows and low data density. Choose how to store LOB data wisely.



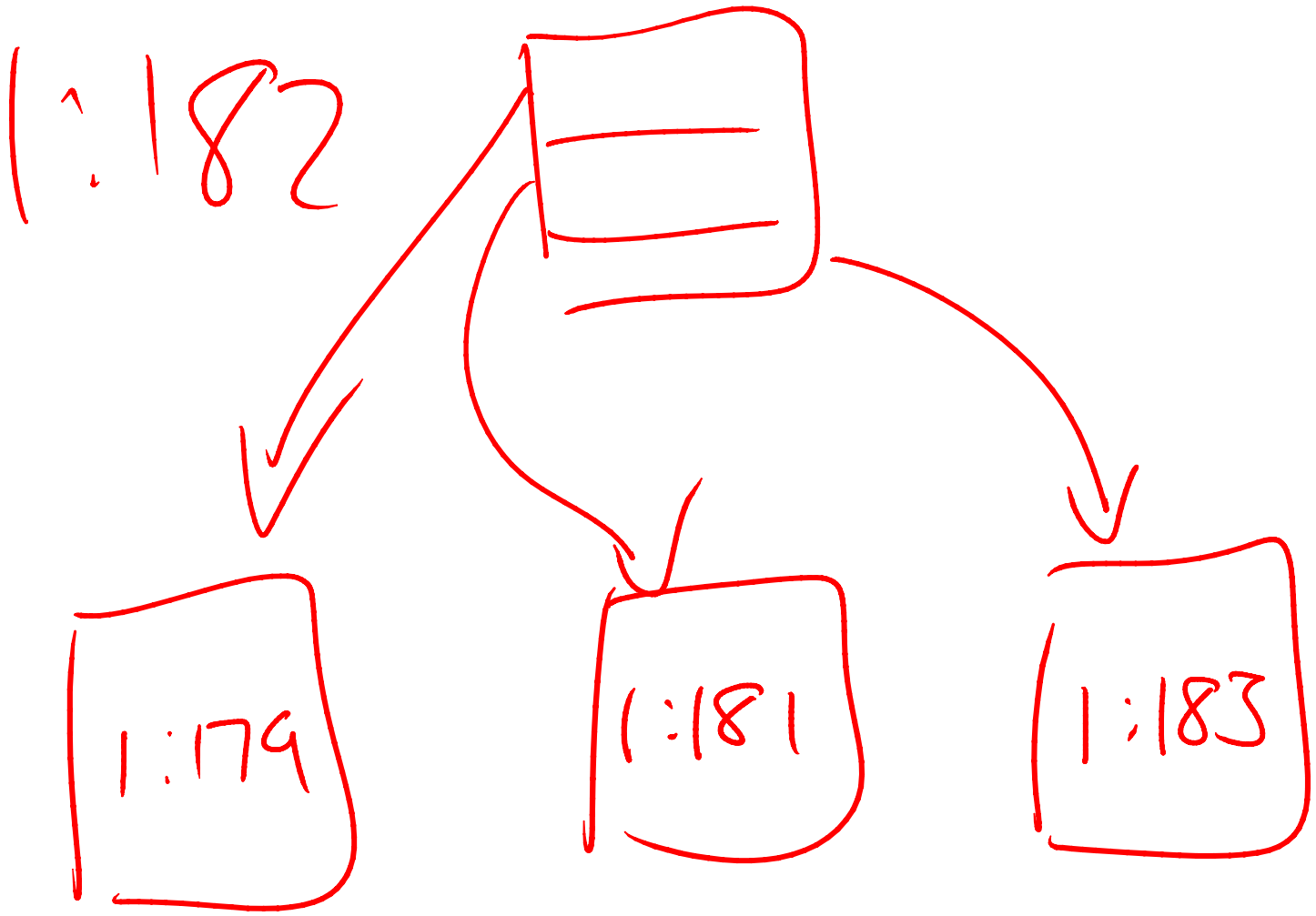
M1S14 & M2S35 Explaining how there's a new portion of the version store created every minute.



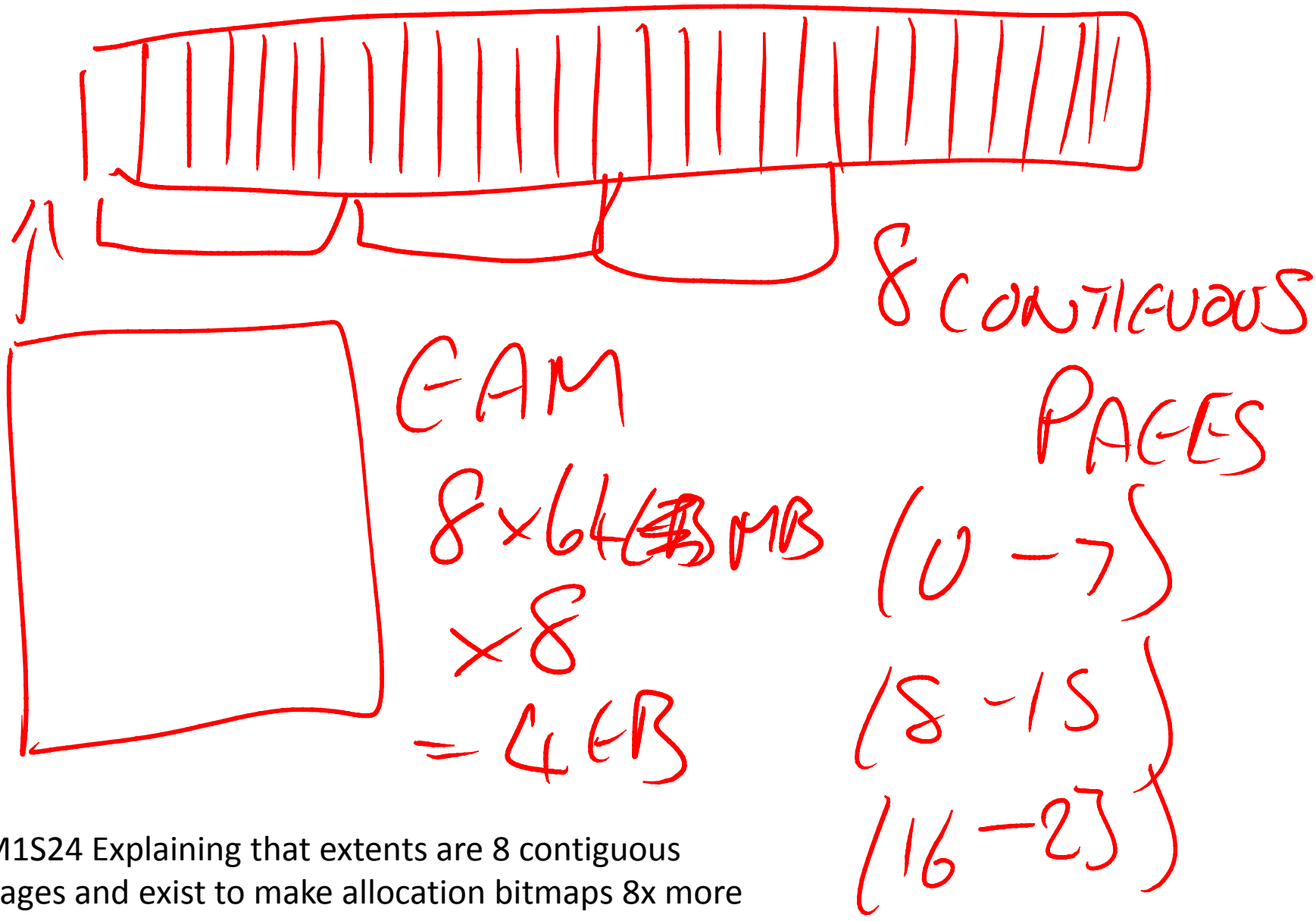
M1S11 Explaining how a 20KB LOB value will be stored. Ptr in the row to another text record, linking to three records holding the data.



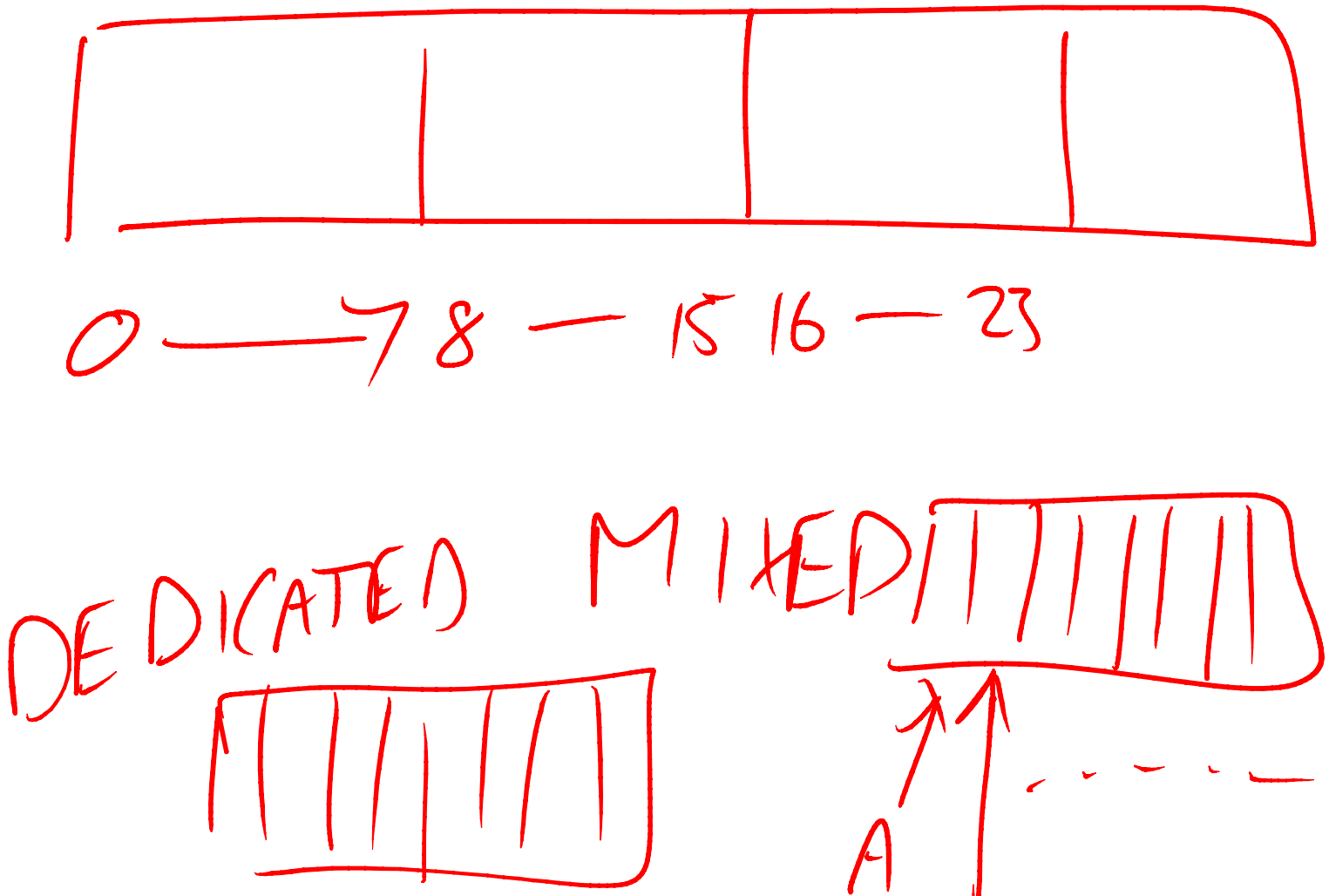
M1S15 When a record is deleted, it's marked as a ghost record. The ghost cleanup task does not physically delete the records – it removes the slot array entry that points to the record on the page. I.e. the record is unmarked as being a record and the area on the page is now free space – that just happens to have bits and bytes that used to be a valid record.



M1S22 Showing the clustered index structure generated by the page/record demo.



M1S24 Explaining that extents are 8 contiguous pages and exist to make allocation bitmaps 8x more dense and efficient.

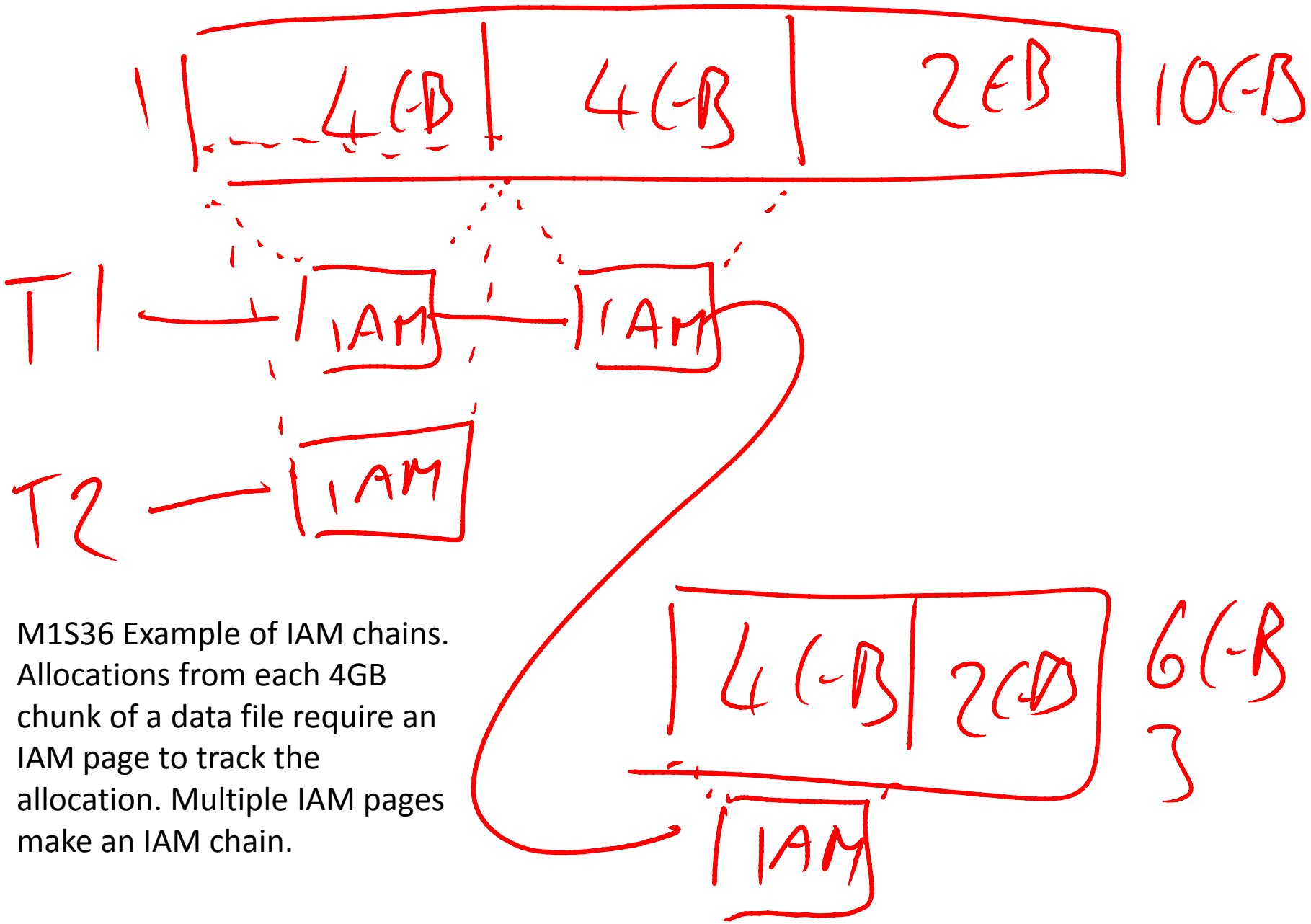


M1S24 Explaining the difference between mixed and dedicated extents. Mixed extent is one in which the 8 pages could be allocated to 8 different objects. Dedicated extent is one where all 8 pages are reserved for exclusive use of a single object.

Later in the deck I refine ‘object’ to really be ‘allocation unit’.

0 – FH
1 – PFS
2 – GAM
3 – SGAM
4 – UNUSED
5 – UNUSED
6 – DIFF
7 – ML

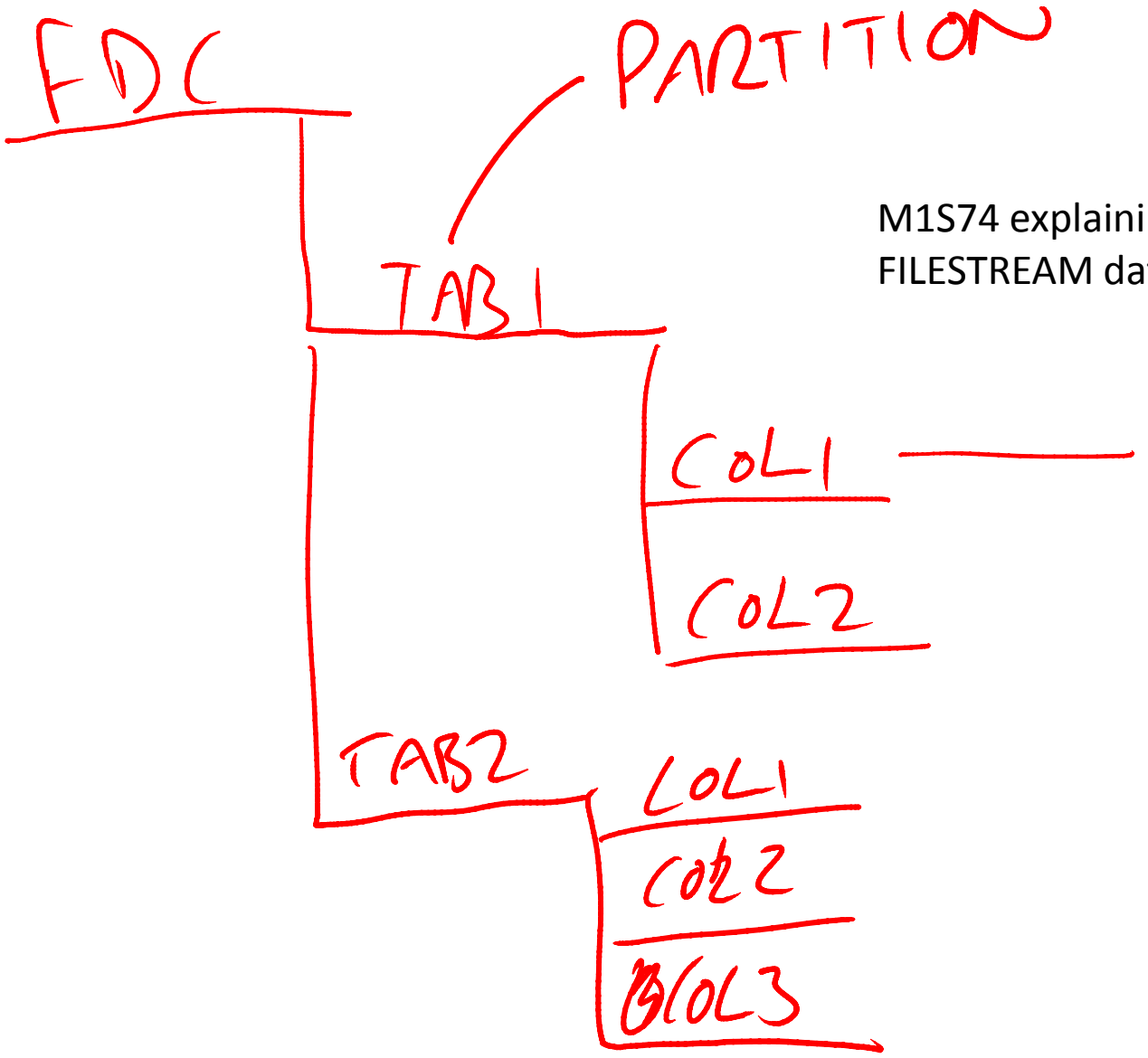
M1 The page types in the first extent in a data file



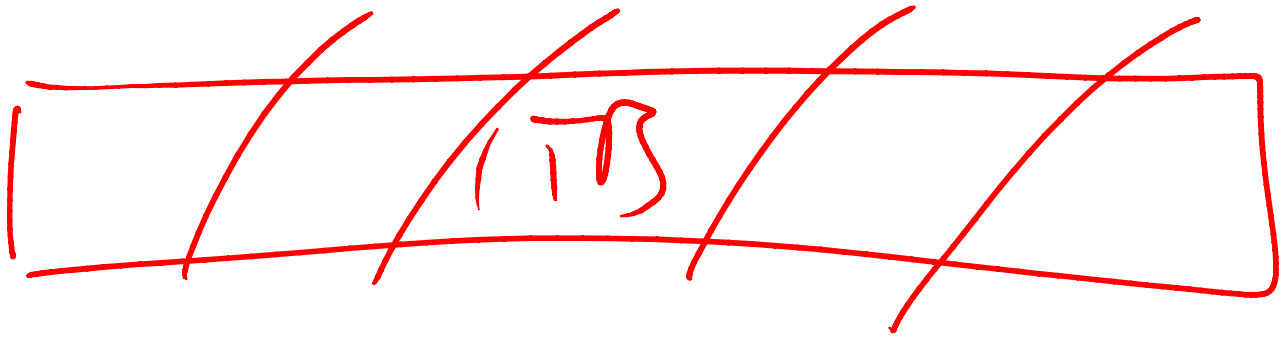
M1S36 Example of IAM chains. Allocations from each 4GB chunk of a data file require an IAM page to track the allocation. Multiple IAM pages make an IAM chain.

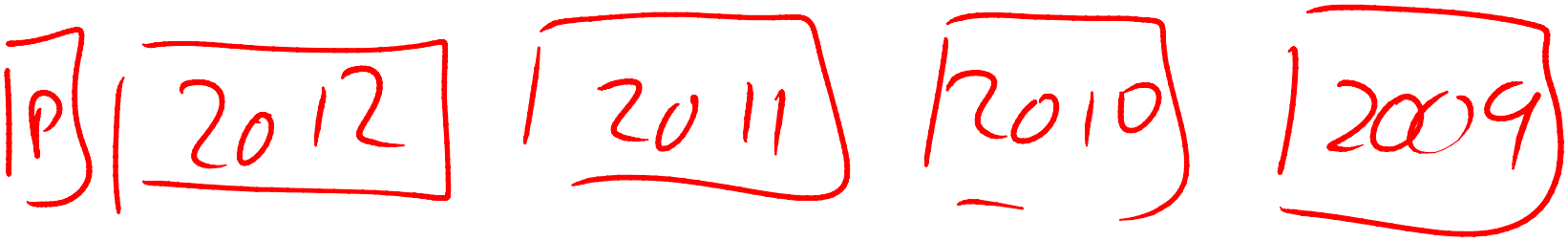
2008 SYSTEM CATALOG VIEWS POSTER

bit.ly/g4ATKt

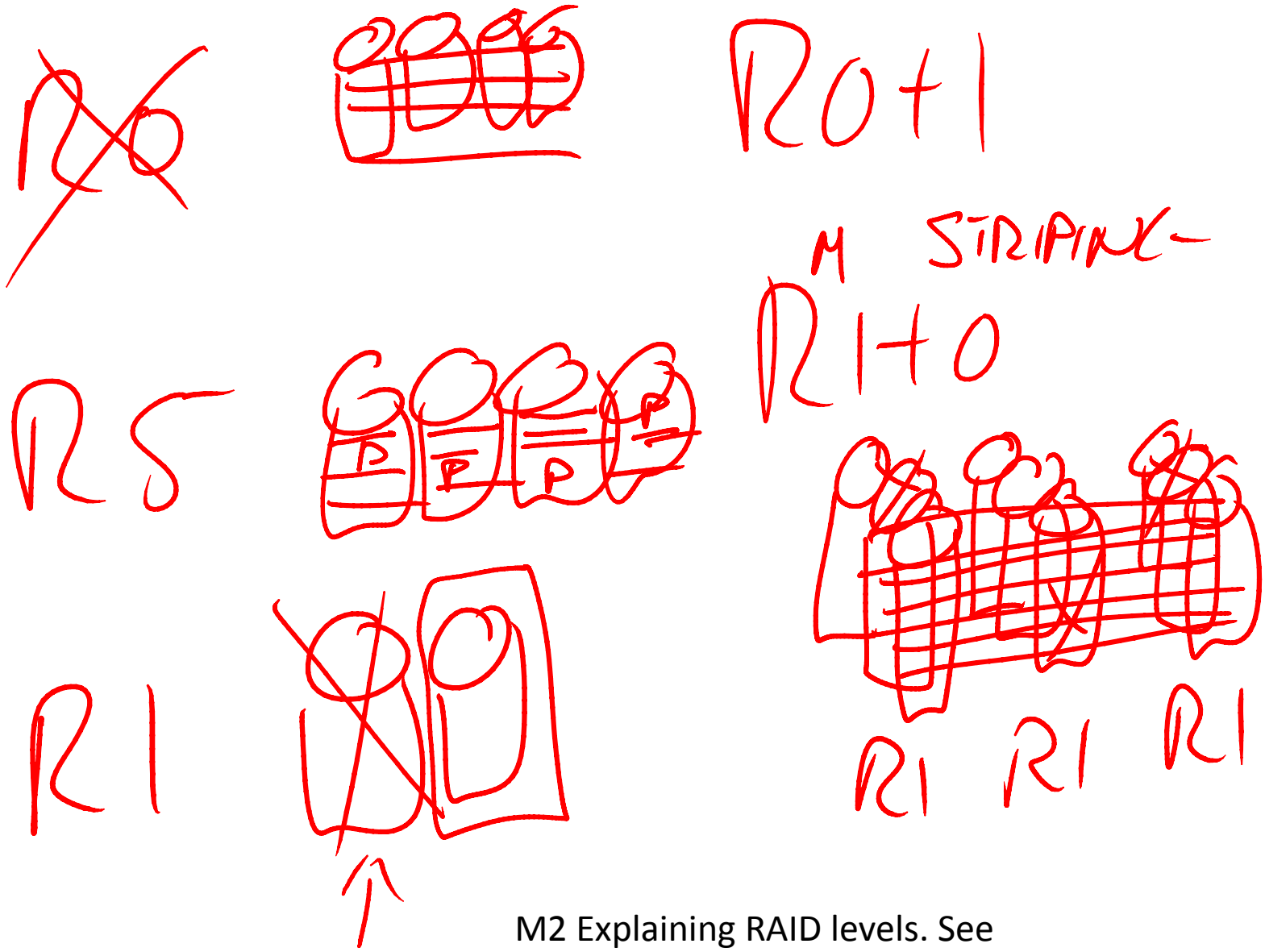


M1S74 explaining the NTFS structure of a FILESTREAM data container





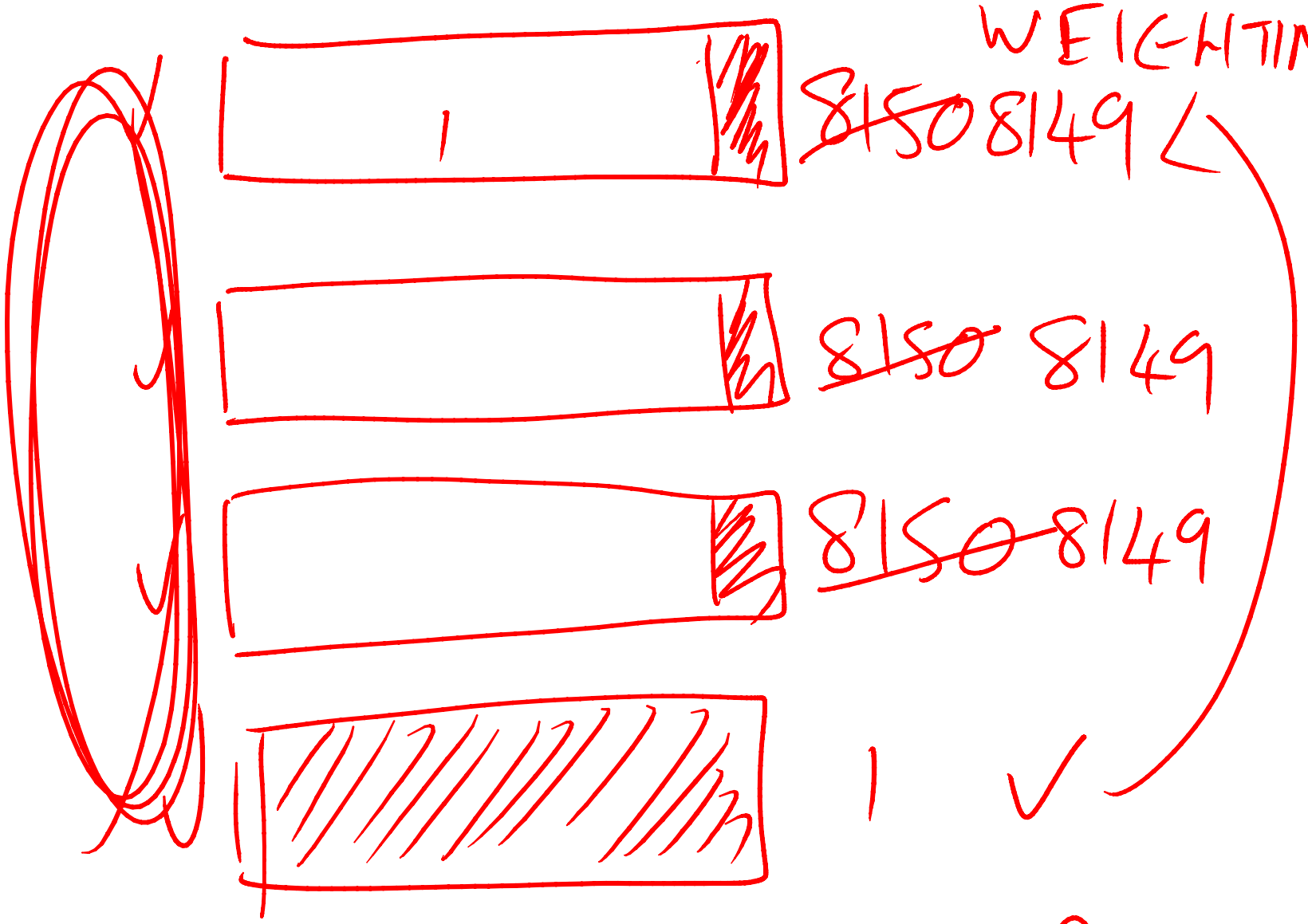
M2S7 Splitting a large database up into multiple filegroups can enable you to do small, targeted restores or even a partial restore from scratch – in the event that the database is damaged or destroyed.



M2 Explaining RAID levels. See
http://en.wikipedia.org/wiki/RAID

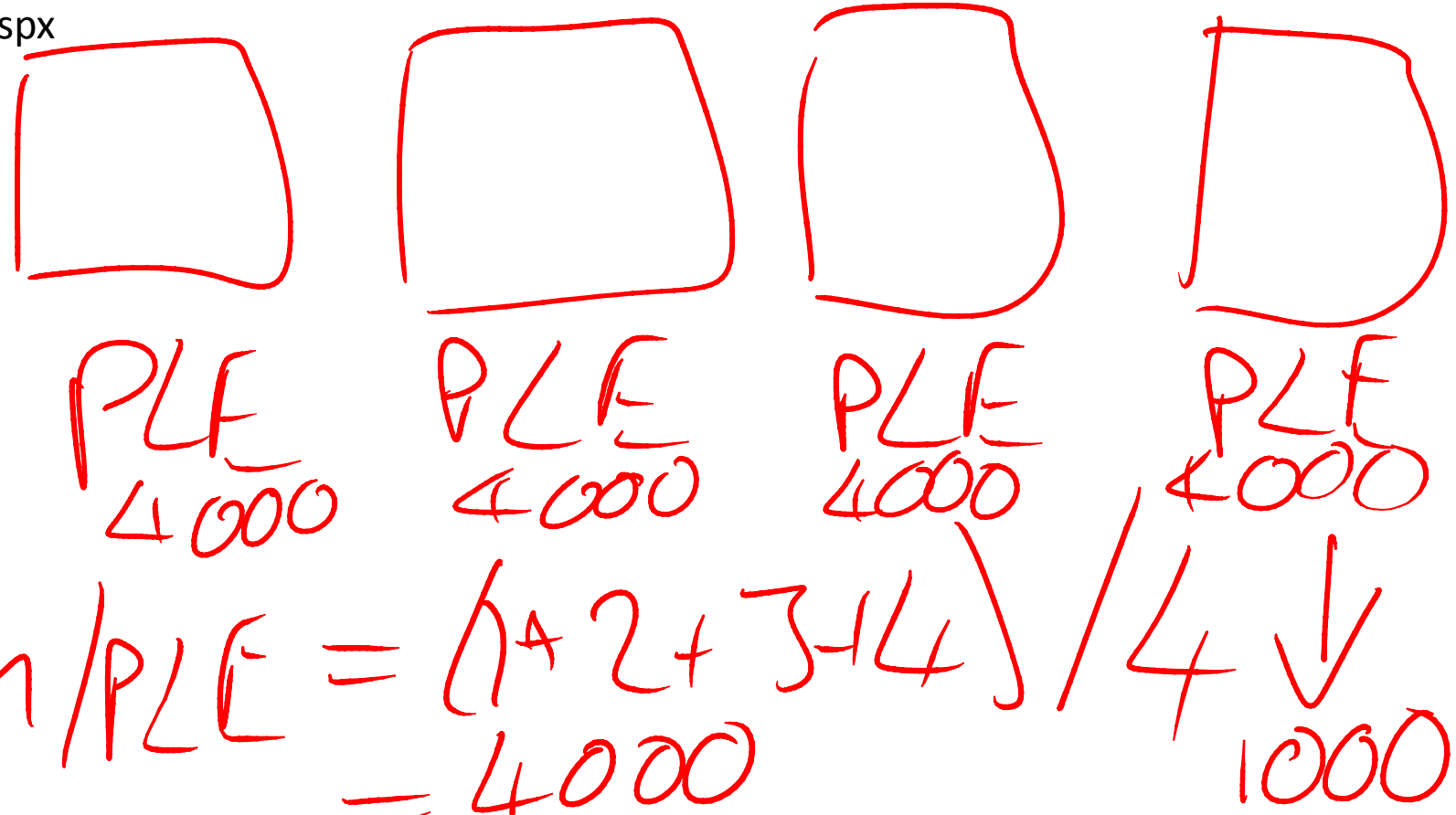
sys.dm_io_virtual_stats

M2 sys.dm_io_virtual_file_stats is used to identify I/O hotspots from within SQL Server. See http://bit.ly/XrIYYG

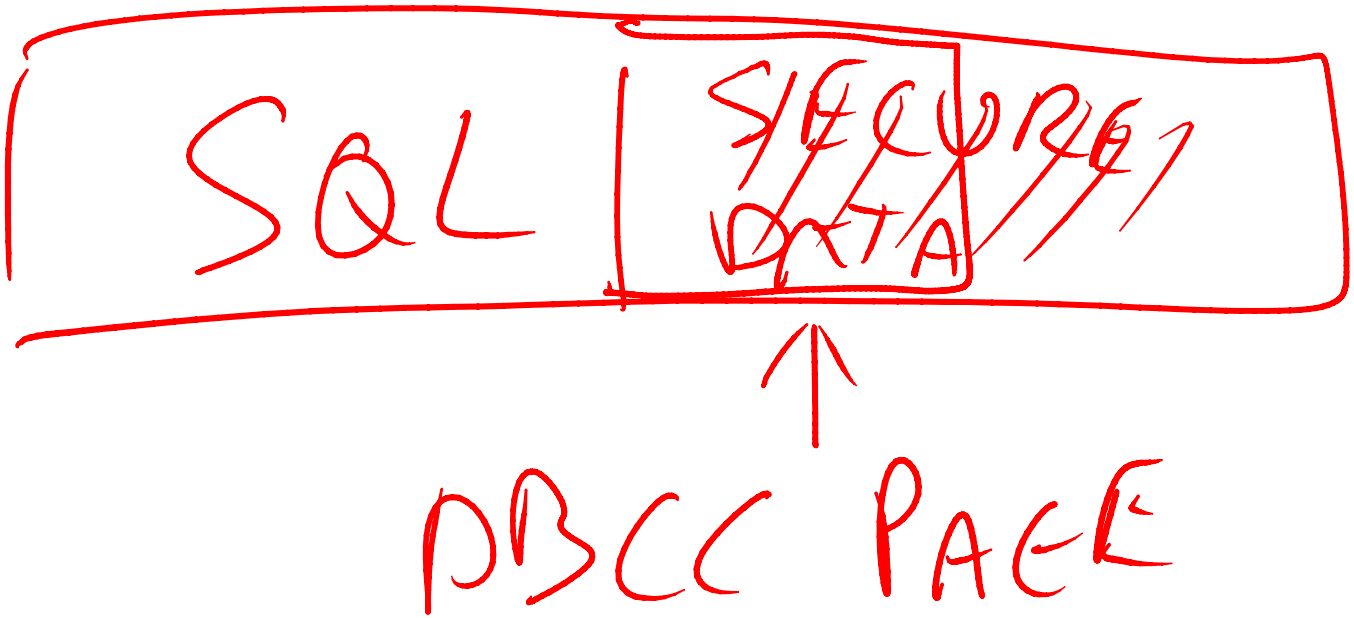


M2S19 Explaining round-robin allocation and proportional fill

See http://www.sqlskills.com/BLOGS/PAUL/post/Page-Life-Expectancy-isnt-what-you-think.aspx

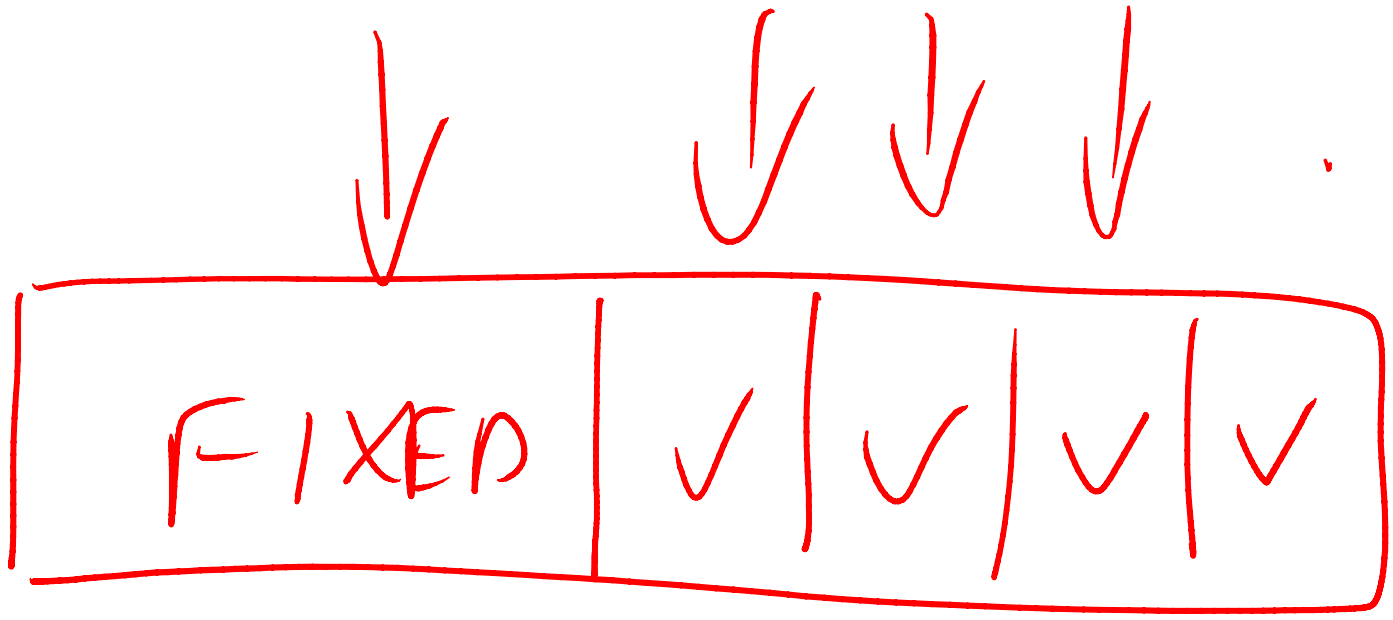


This should be Buffer Node instead of Buffer Partition – my mistake.

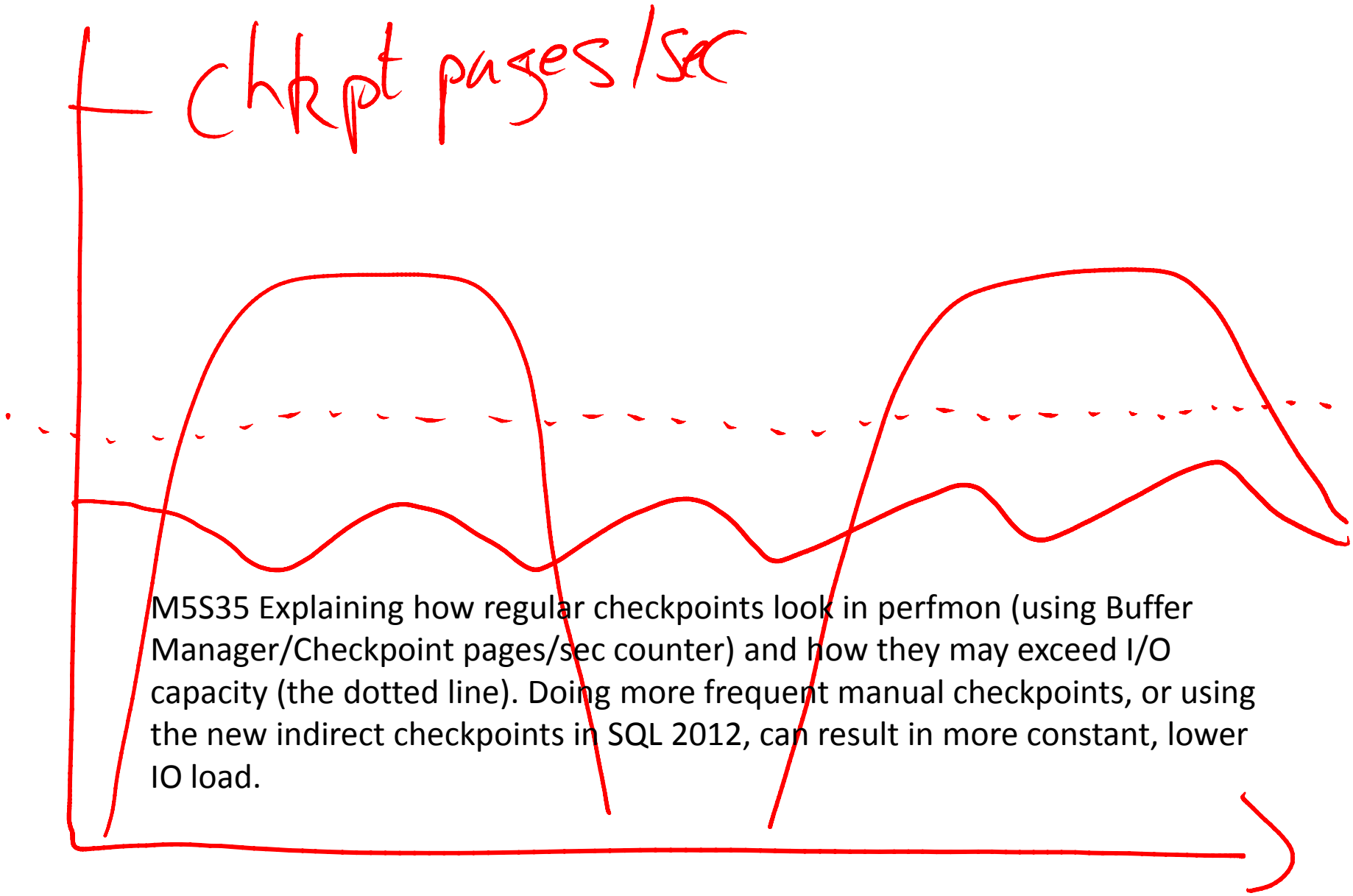


M2S24 Showing the security risk with instant file initialization

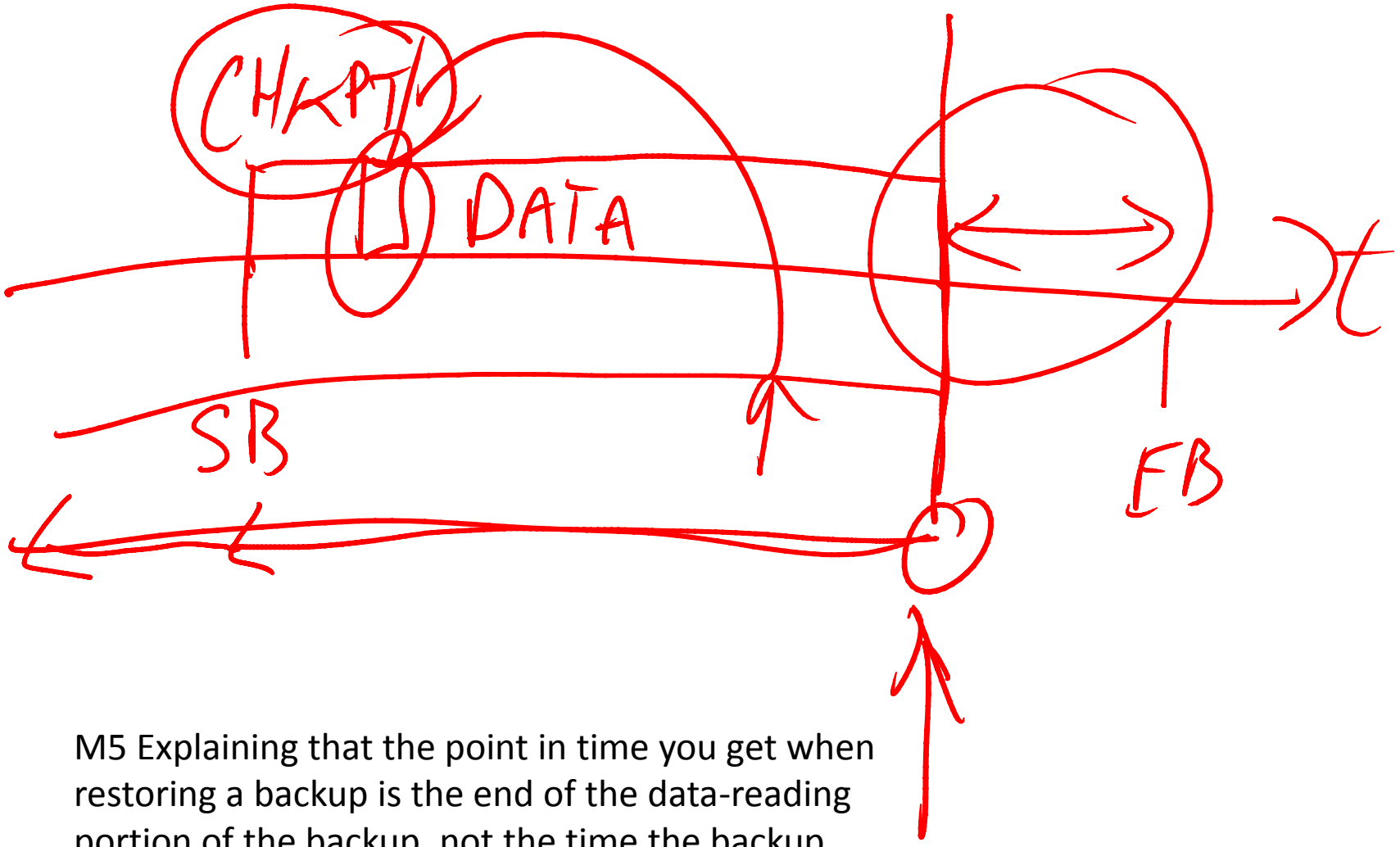
M2S40 Explaining PFS and SGAM contention in tempdb and how adding multiple files spreads the contention over multiple files, thus reducing it.



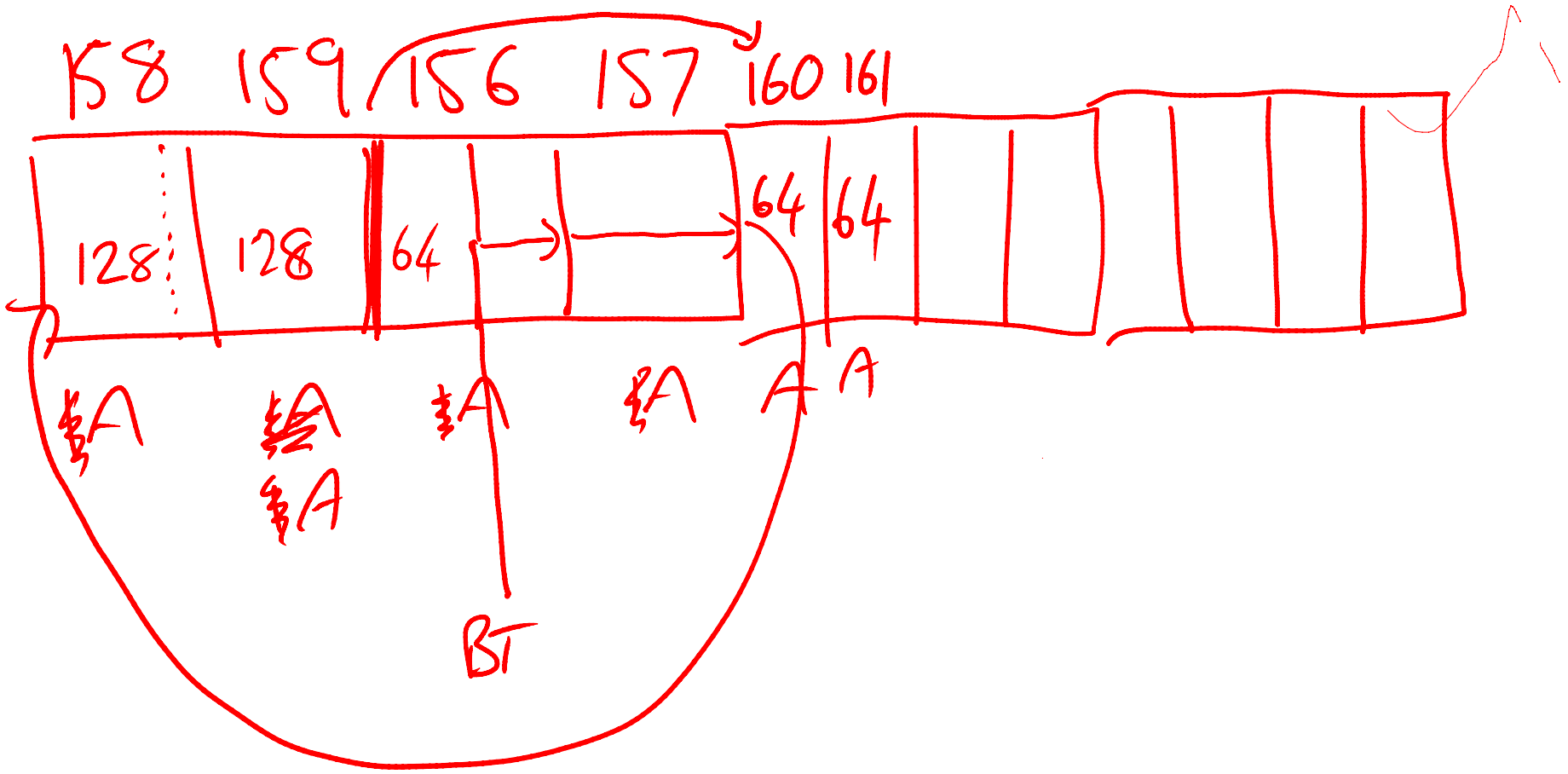
M5S19 Explaining how the Access Methods determines how to log UPDATEs efficiently by splitting the logging by portion of the record. Each variable length column is a portion, as is the fixed-width portion (up to 16 bytes in each log record)



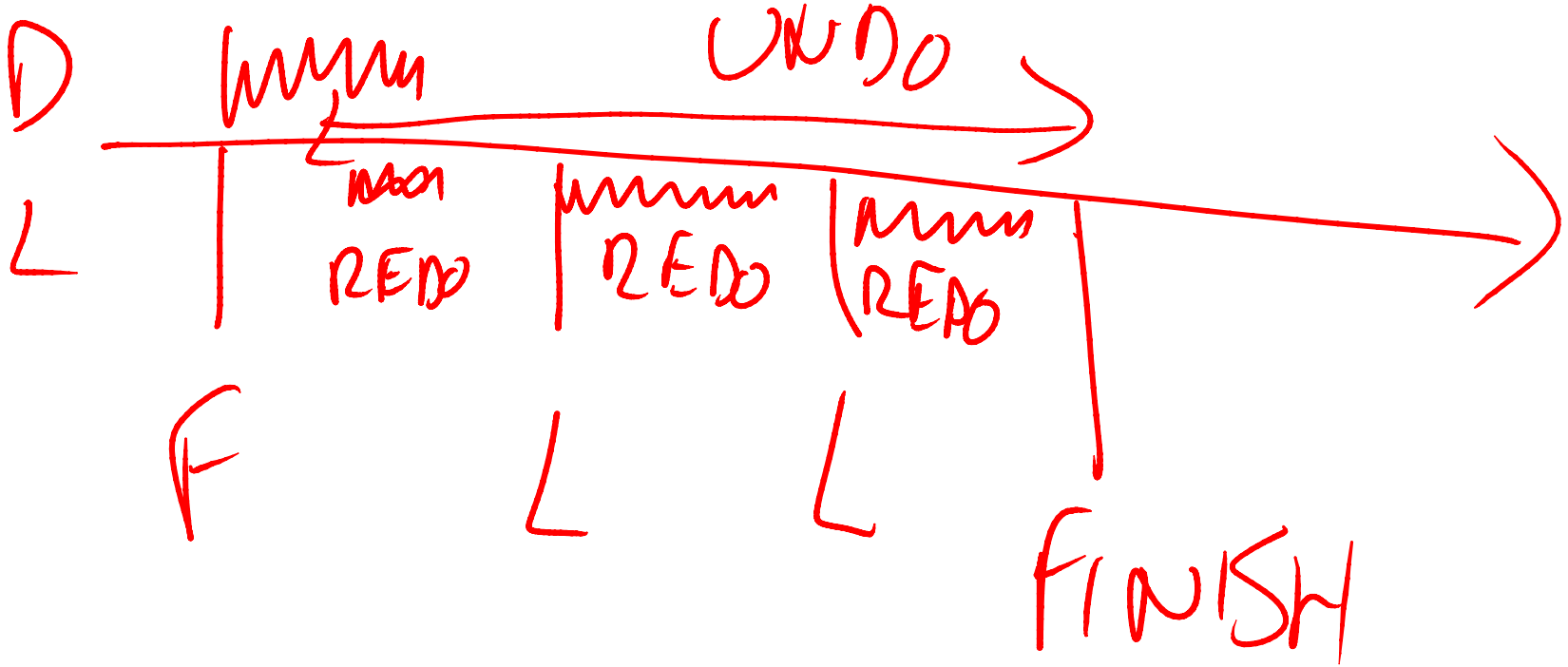
M5S35 Explaining how regular checkpoints look in perfmon (using Buffer Manager/Checkpoint pages/sec counter) and how they may exceed I/O capacity (the dotted line). Doing more frequent manual checkpoints, or using the new indirect checkpoints in SQL 2012, can result in more constant, lower IO load.



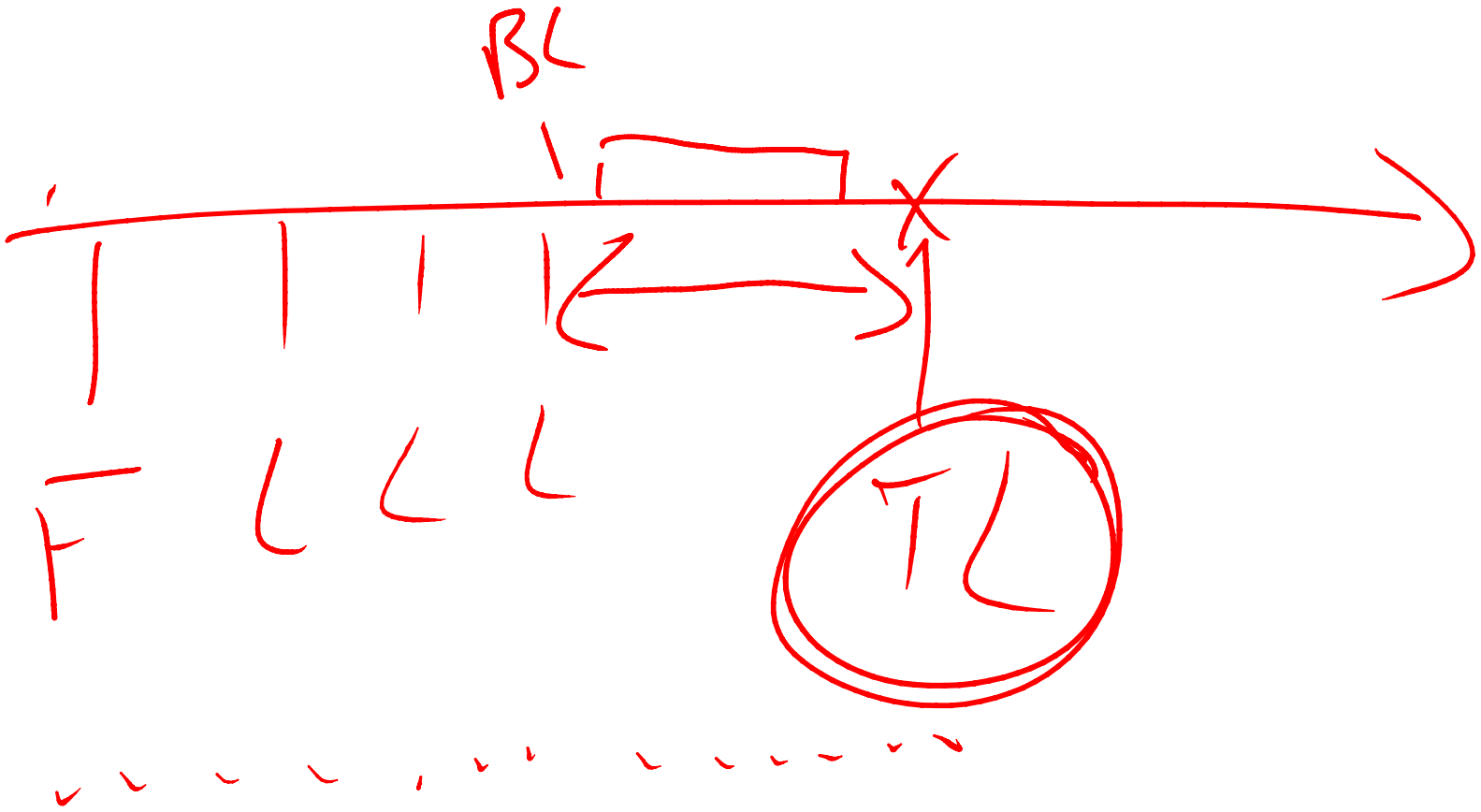
M5 Explaining that the point in time you get when restoring a backup is the end of the data-reading portion of the backup, not the time the backup ends. The double-ended arrow in the circle above right is the time it takes to read all the log back to the checkpoint at the beginning of the backup, or to the oldest active transaction when the backup began. SB = start backup, EB = end of backup.



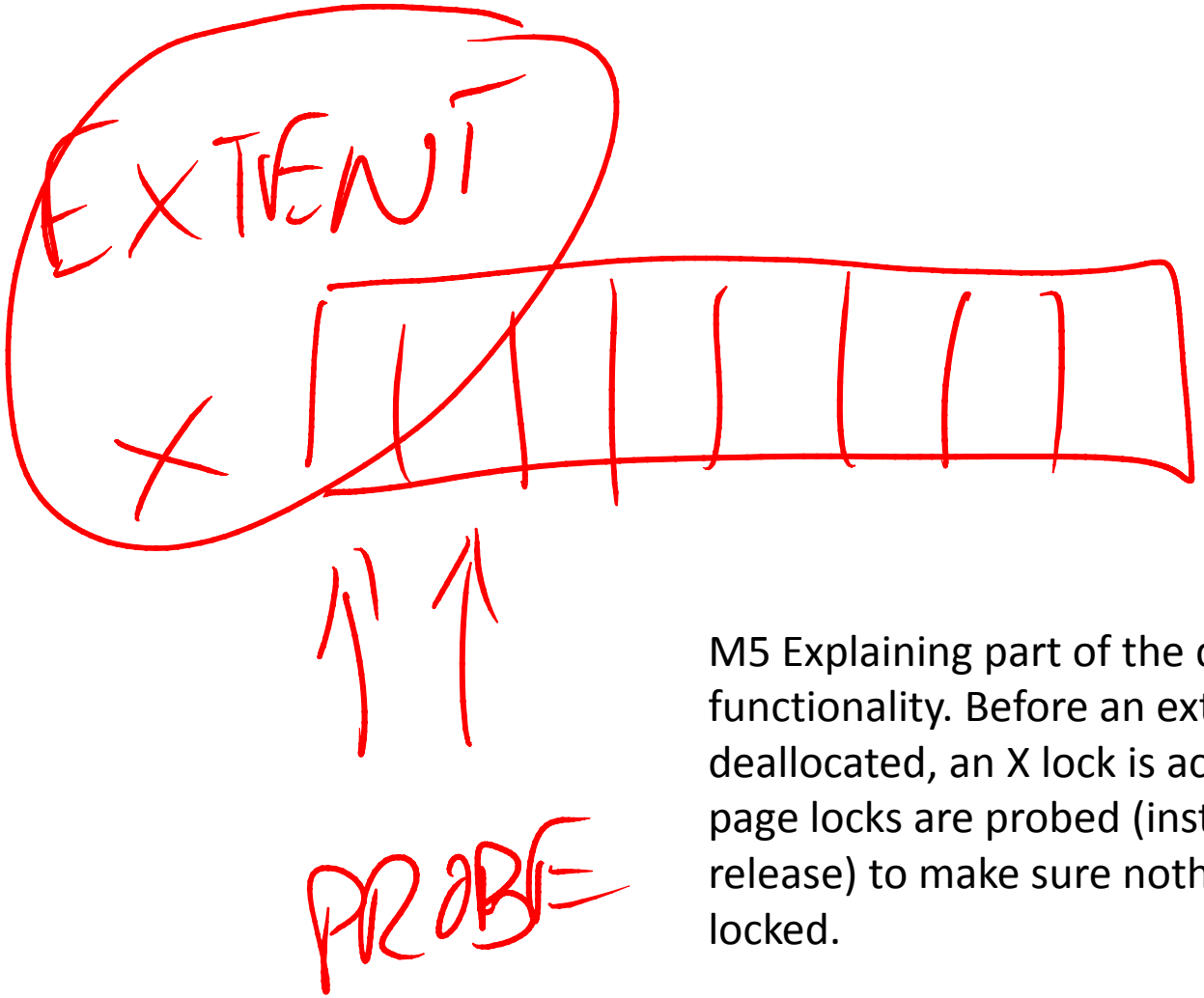
M5S48 Explaining the circular log demo. We discussed log space reservation that happens to ensure the active transactions can roll back without having to grow the log – this accounts for extra log growths. Space reserved in the log does not make a VLF become active – as there are no log records written out, just empty space reserved.



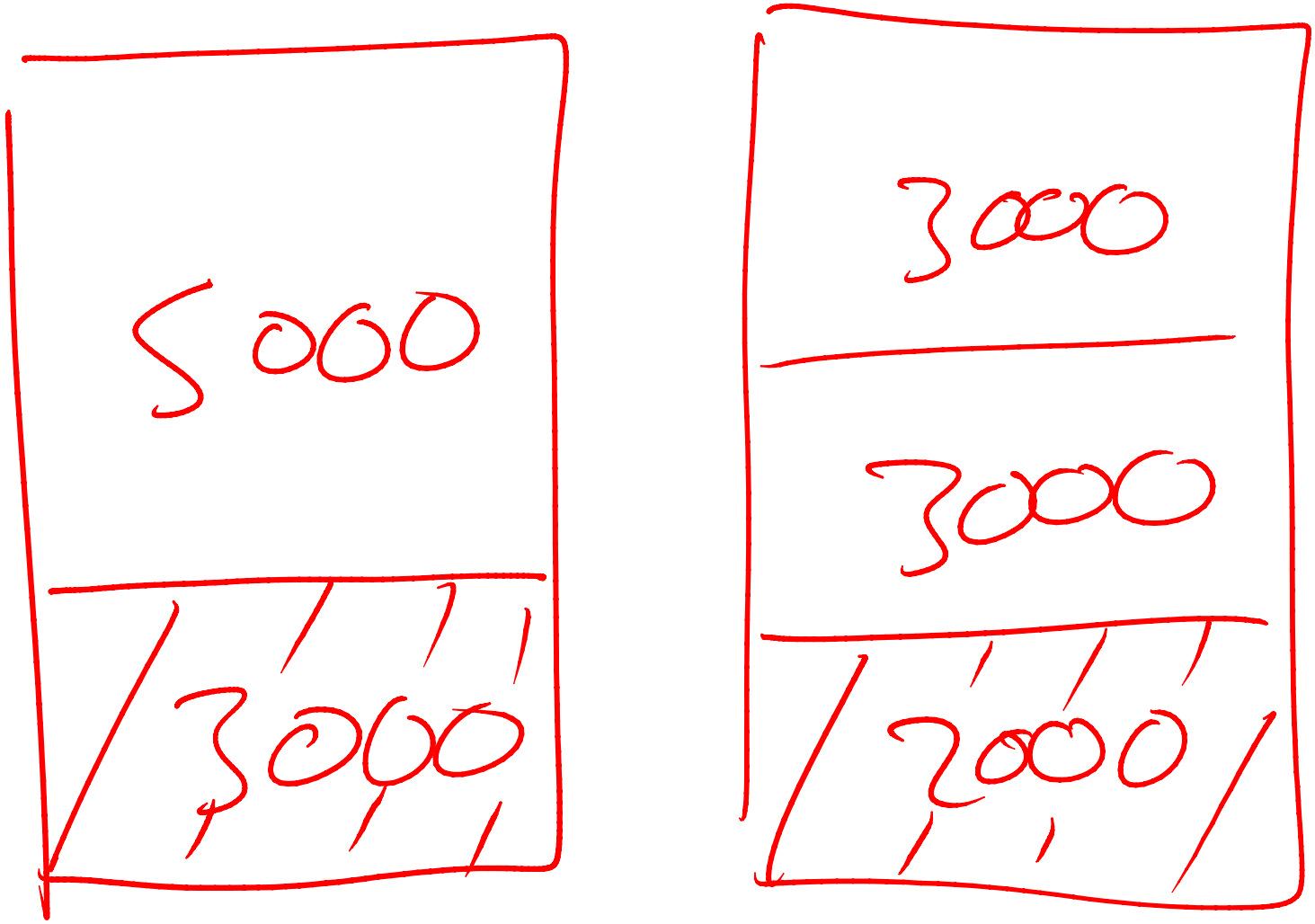
M5 Explaining that in a multi-backup restore sequence, REDO is run when each backup is restored. UNDO is run at the end. It does NOT leave both REDO and UNDO until the end.



M5S60 Explaining that if a minimally-logged operation is present in a log backup, you cannot do a point-in-time restore to any time covered by that backup (except the start or end of that time period). You also cannot do a tail-of-the-log backup after a crash that destroys the data files if there has been a minimally-logged operation since the last log backup. Well, you can, but the log backup is invalid.

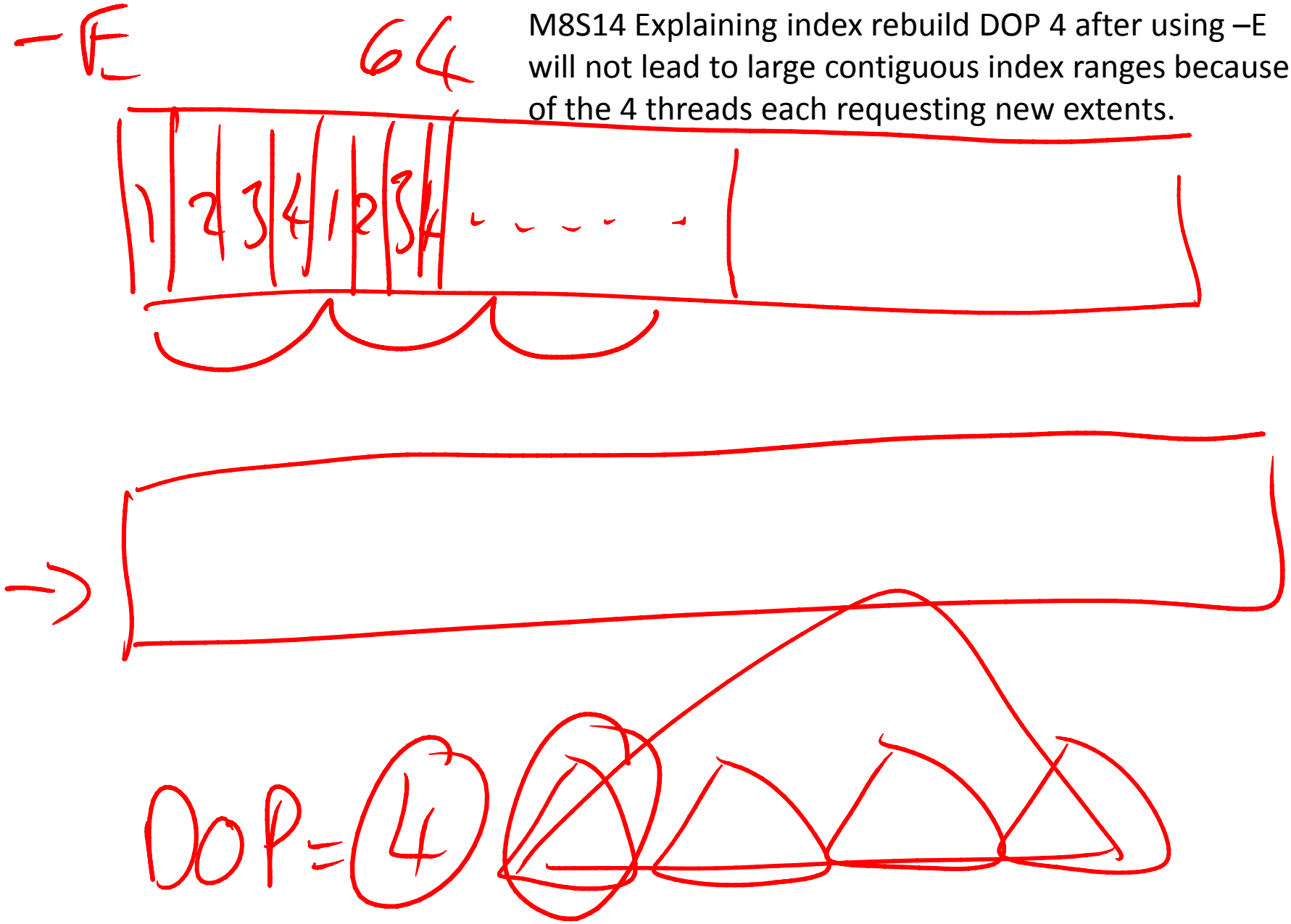


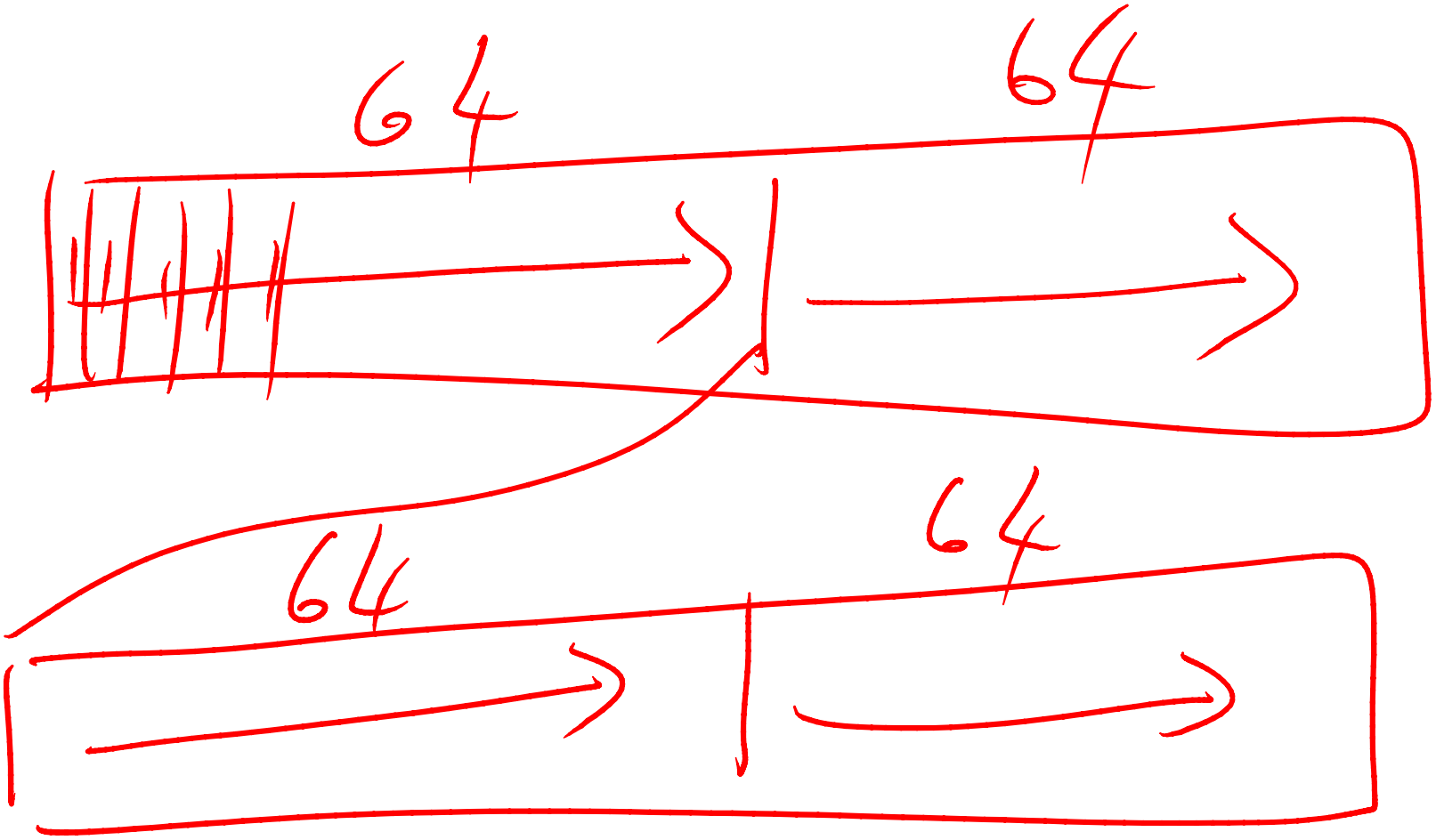
M5 Explaining part of the deferred-drop functionality. Before an extent can be deallocated, an X lock is acquired on it and all 8 page locks are probed (instant acquire X lock and release) to make sure nothing else has them locked.



M8S13 Explaining how fixed-size large rows leads to low page density and wasted space.

M8S14 Explaining index rebuild DOP 4 after using –E will not lead to large contiguous index ranges because of the 4 threads each requesting new extents.



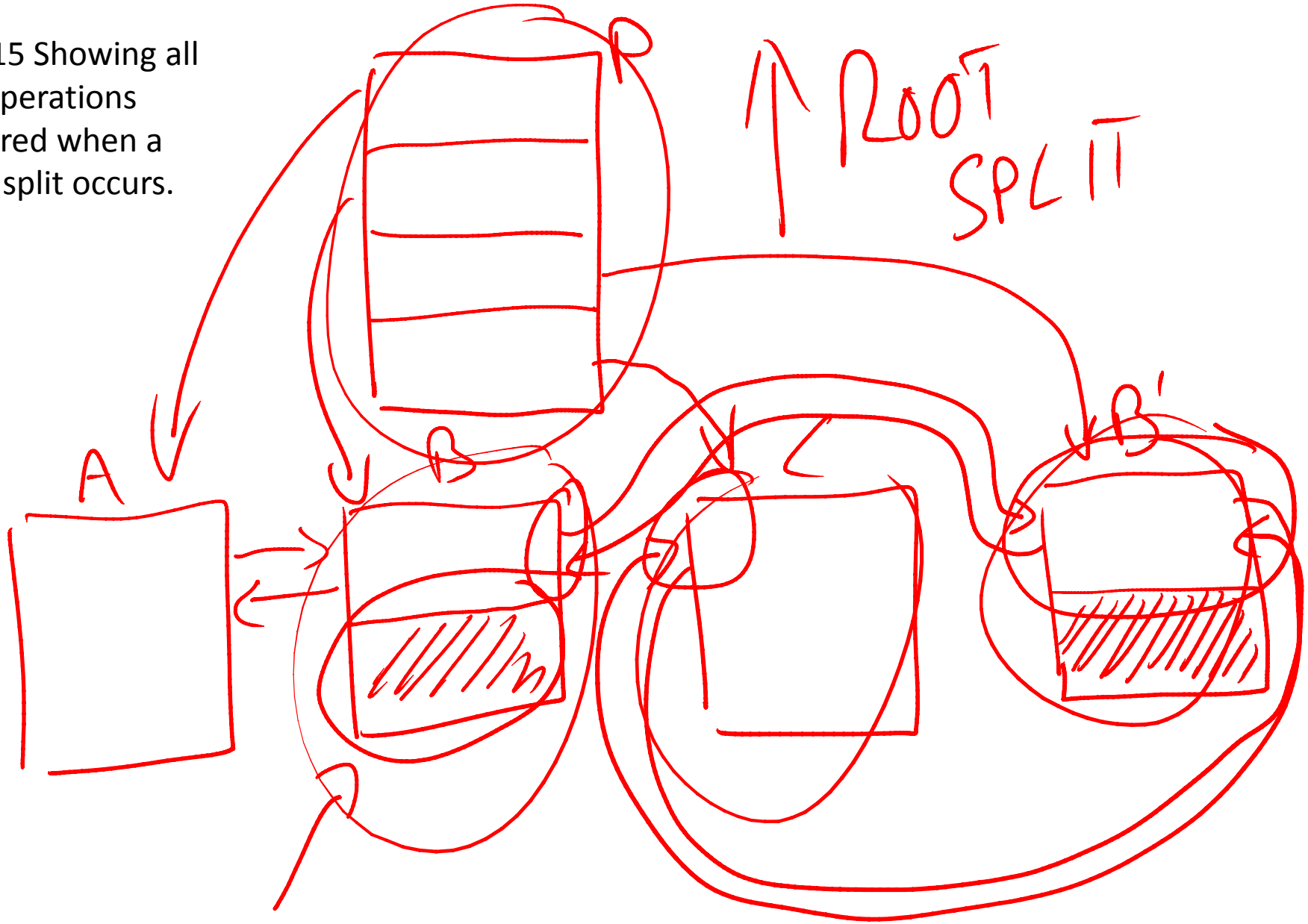


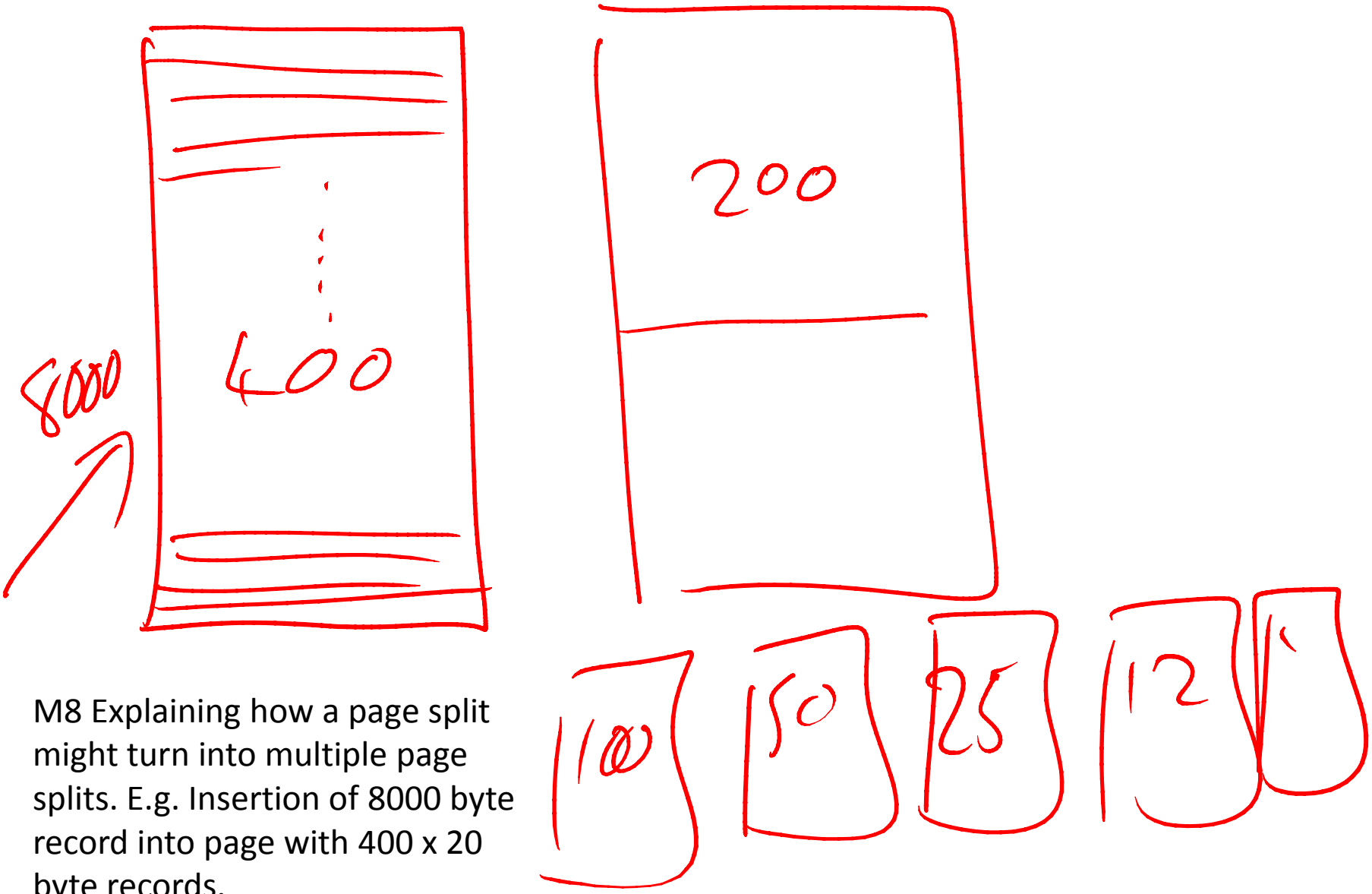
DOP=1

M8S14 Showing a DOP1 index rebuild with two files and –E. It will produce contiguous runs of 64 extents in each file.

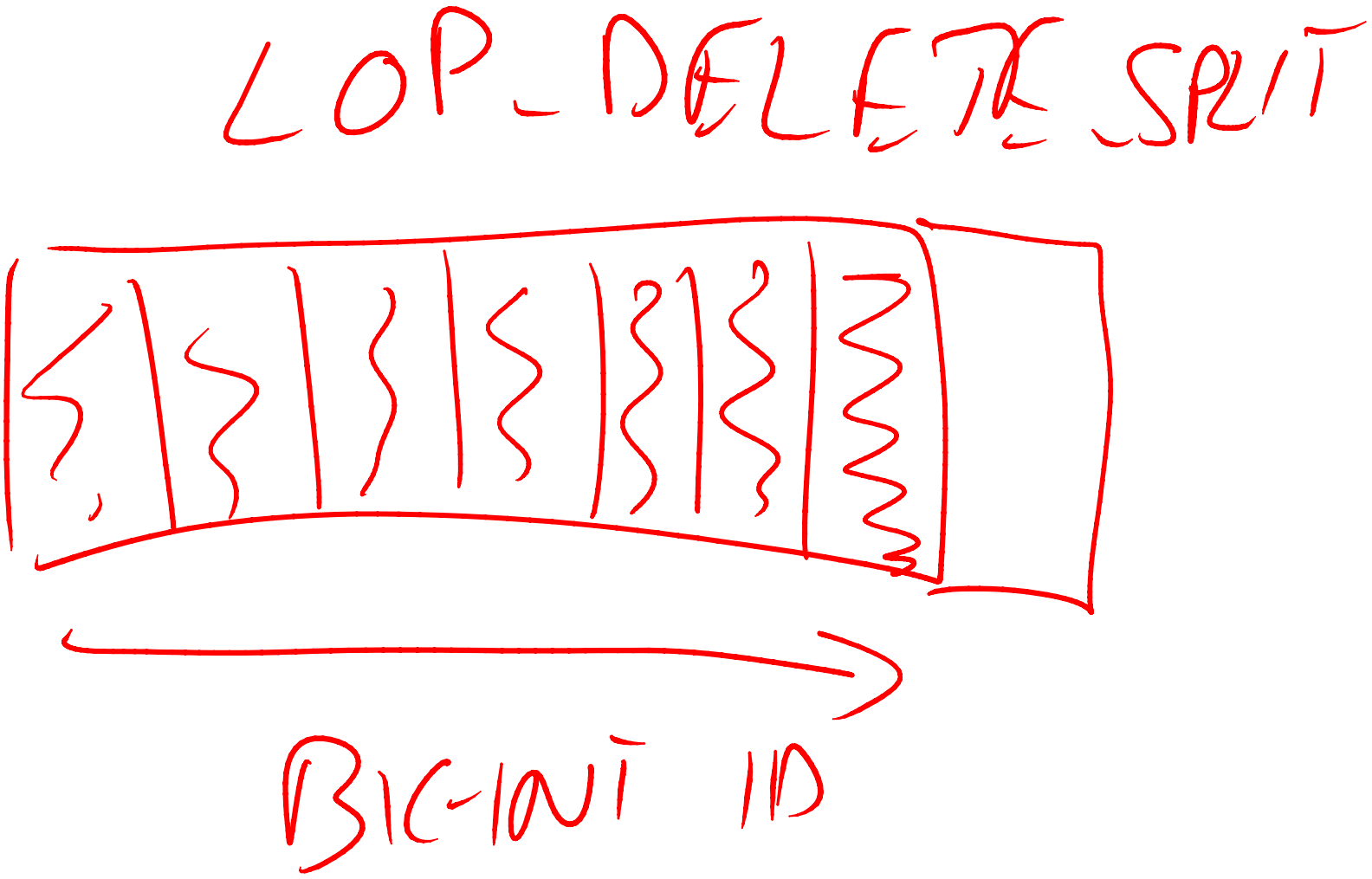
M8S15 A page split usually creates two pages with around 50% free space (i.e. low page density)

M8S15 Showing all the operations required when a page split occurs.





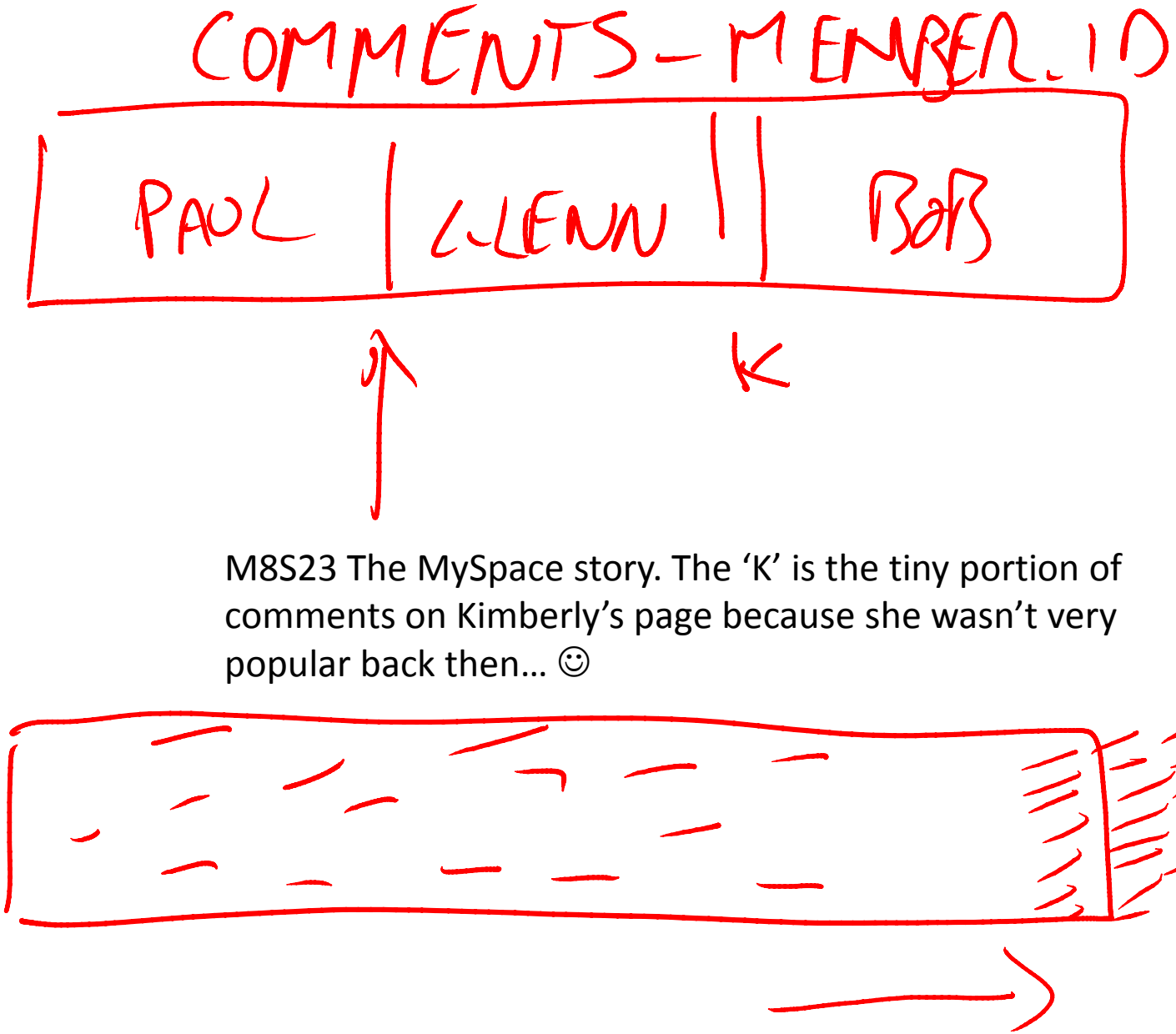
M8 Explaining how a page split might turn into multiple page splits. E.g. Insertion of 8000 byte record into page with 400 x 20 byte records.



M8S18 Append-only insert pattern fills pages on right-hand side of index. New page allocations in this case are called page splits too – making all methods of tracking page splits largely useless (until 2012 with Xevents). You can look for LOP_DELETE_SPLIT log records to see splits occurring.

SIZE	NO SPLIT	SPLIT	X
1000	1244	6772	x5
100	344	5964	x18
10	256	10364	x45

M8S22 The numbers from the page split logging cost demo.



M8S23 The MySpace story. The ‘K’ is the tiny portion of comments on Kimberly’s page because she wasn’t very popular back then… ☺

M8S37 Explaining staggered index maintenance with database mirroring. See http://bit.ly/IS04W5 for more explanation