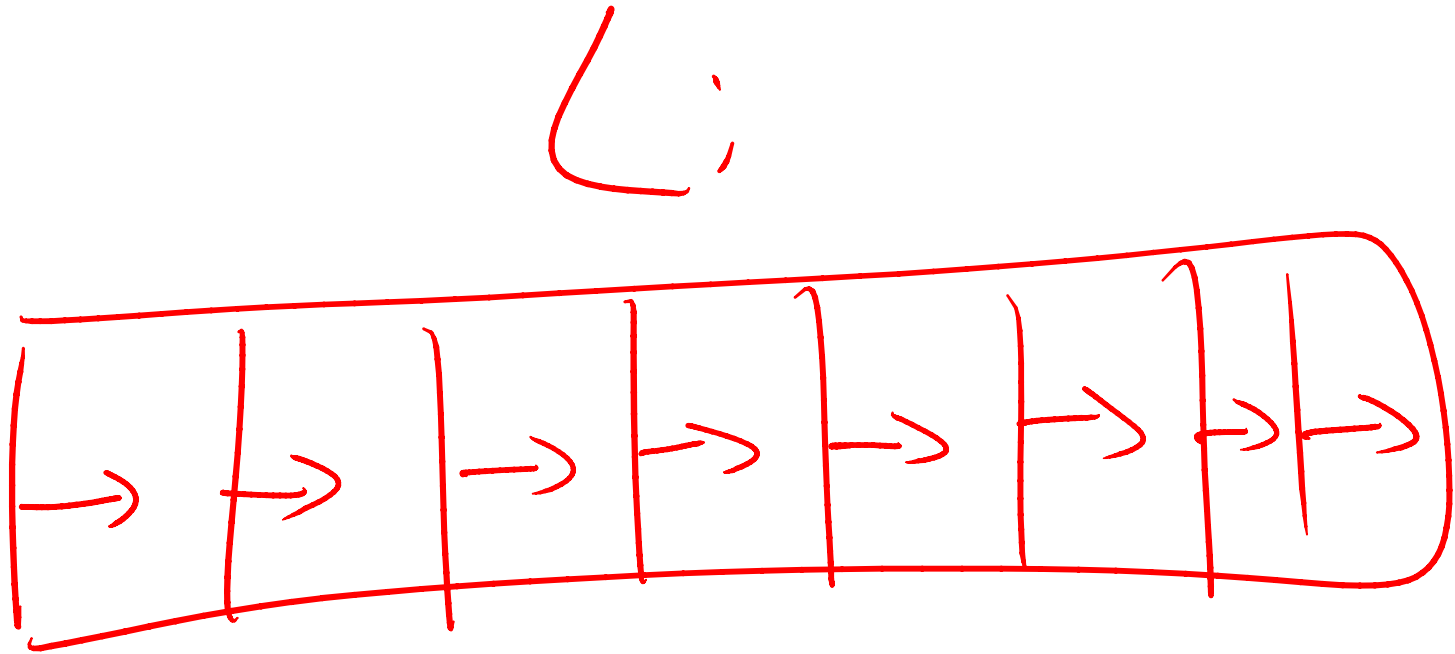




M1S7 Be careful about putting all the log files on a single volume. If they're all having activity, that creates multiple write points on the volume, making it a random I/O workload, even though each log is sequential-write only.

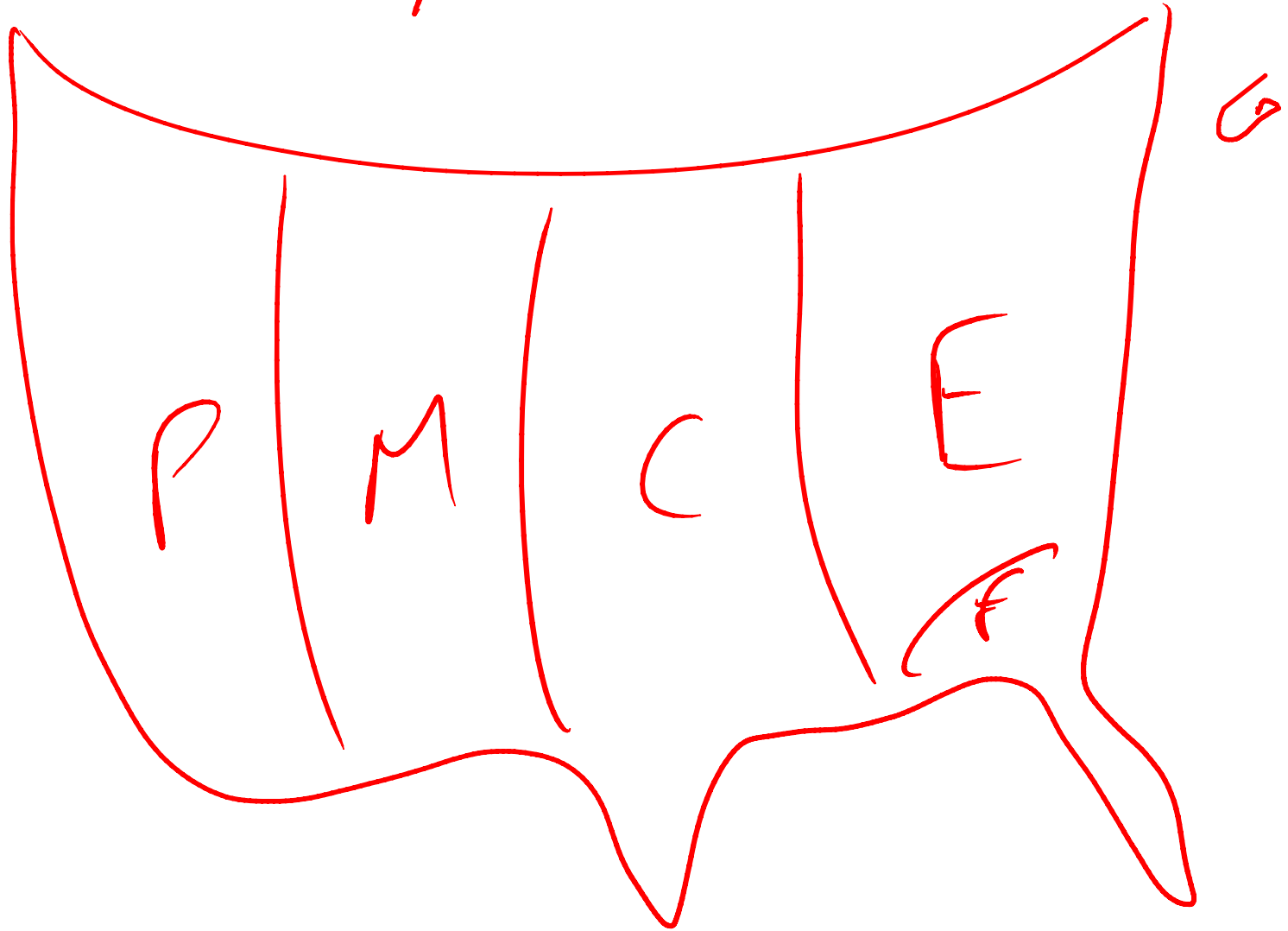


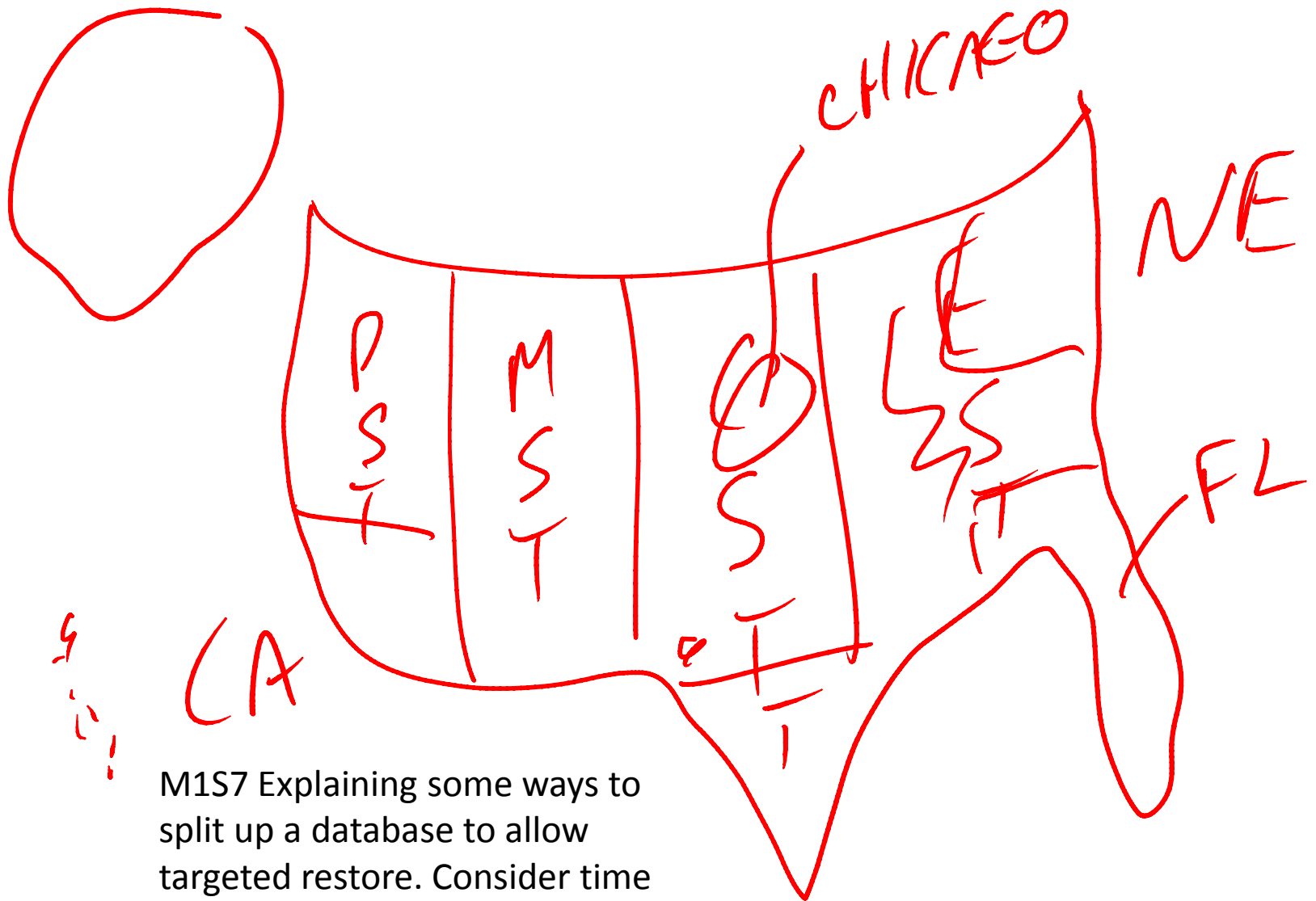
M1S7 Explaining that dividing a large database into filegroups allows targeted restores during disaster recovery, rather than having to wait for the entire single file database to restore.

PARTIAL



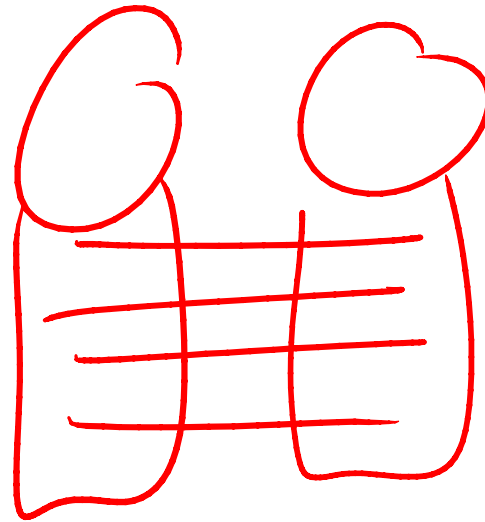
7-6





M1S7 Explaining some ways to split up a database to allow targeted restore. Consider time zone, population density, or other demographics.

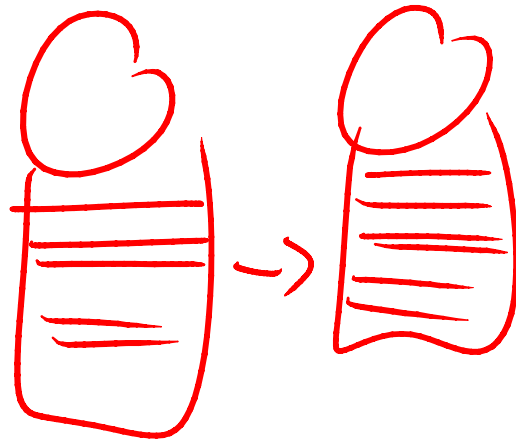
STRIPING



R0

http://en.wikipedia.org/wiki/Standard_RAID_levels

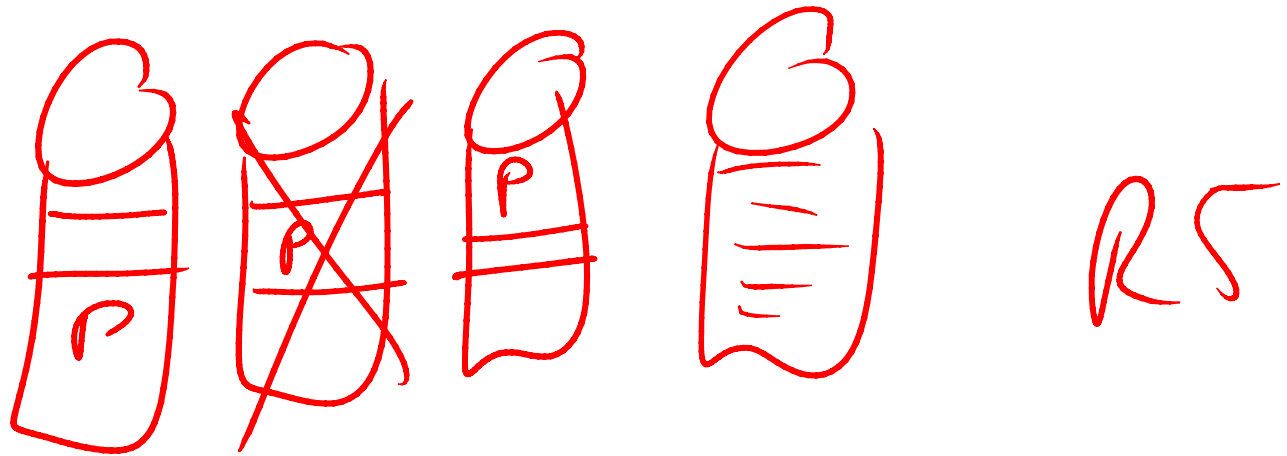
MIRRORING



RA1

http://en.wikipedia.org/wiki/Standard_RAID_levels

NOTATING PARITY

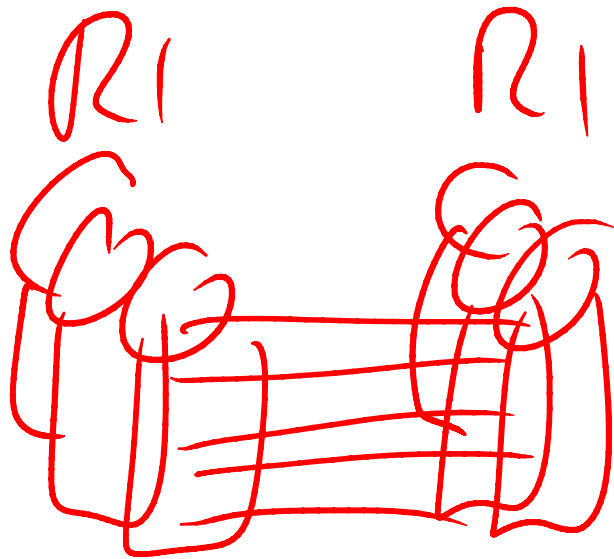


http://en.wikipedia.org/wiki/Standard_RAID_levels

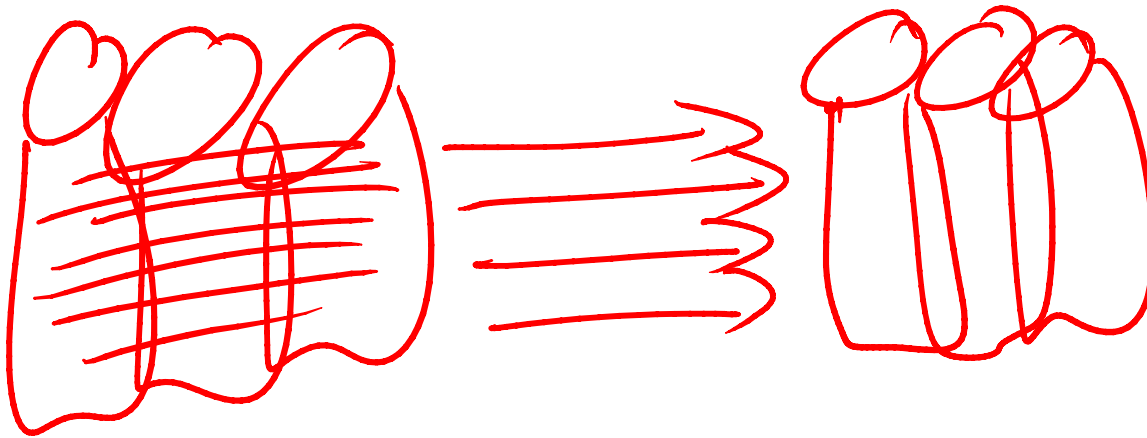
$$R10 = R1 + G$$

= MIRRORING

+ STRIPING

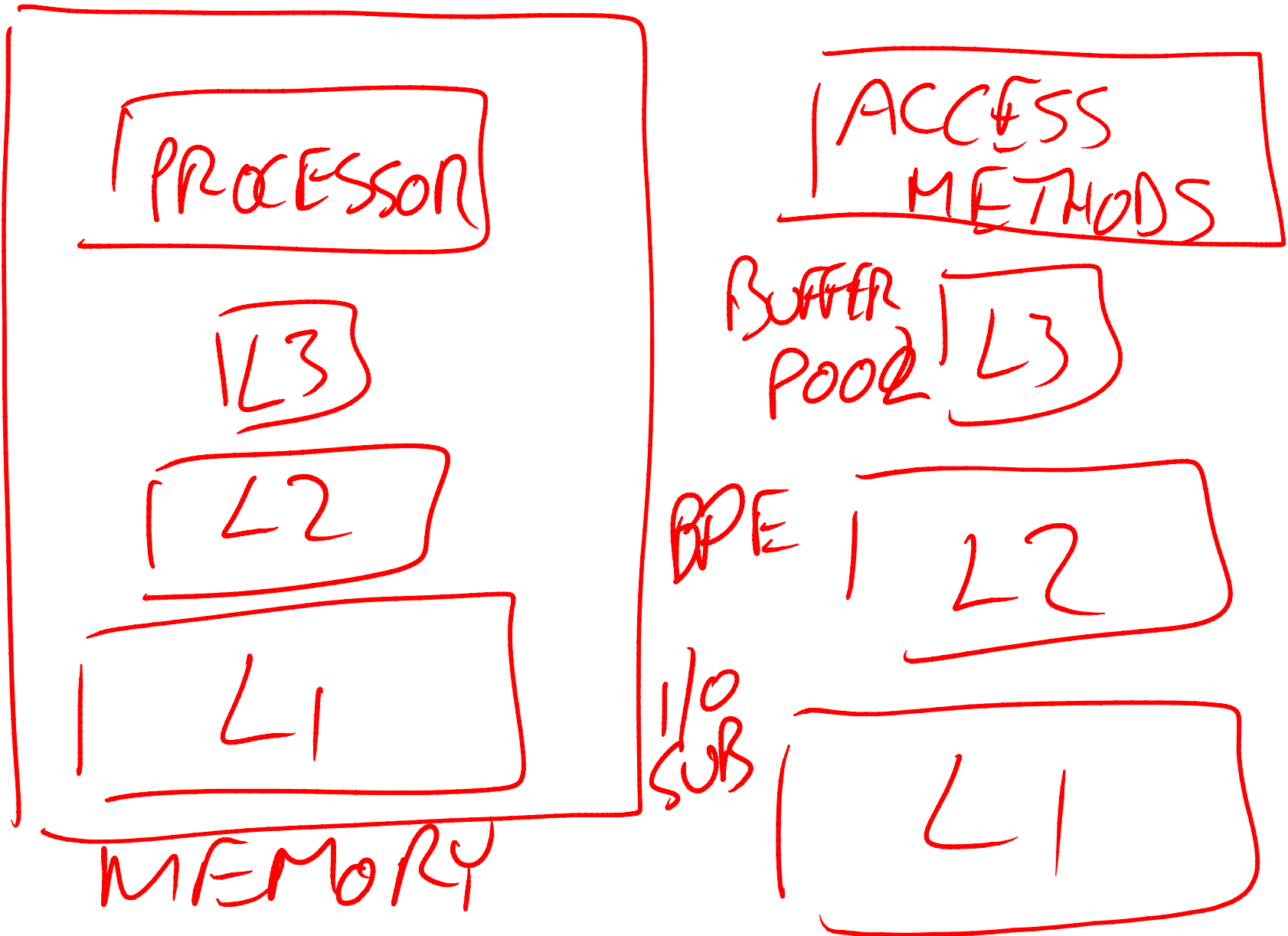


2×1

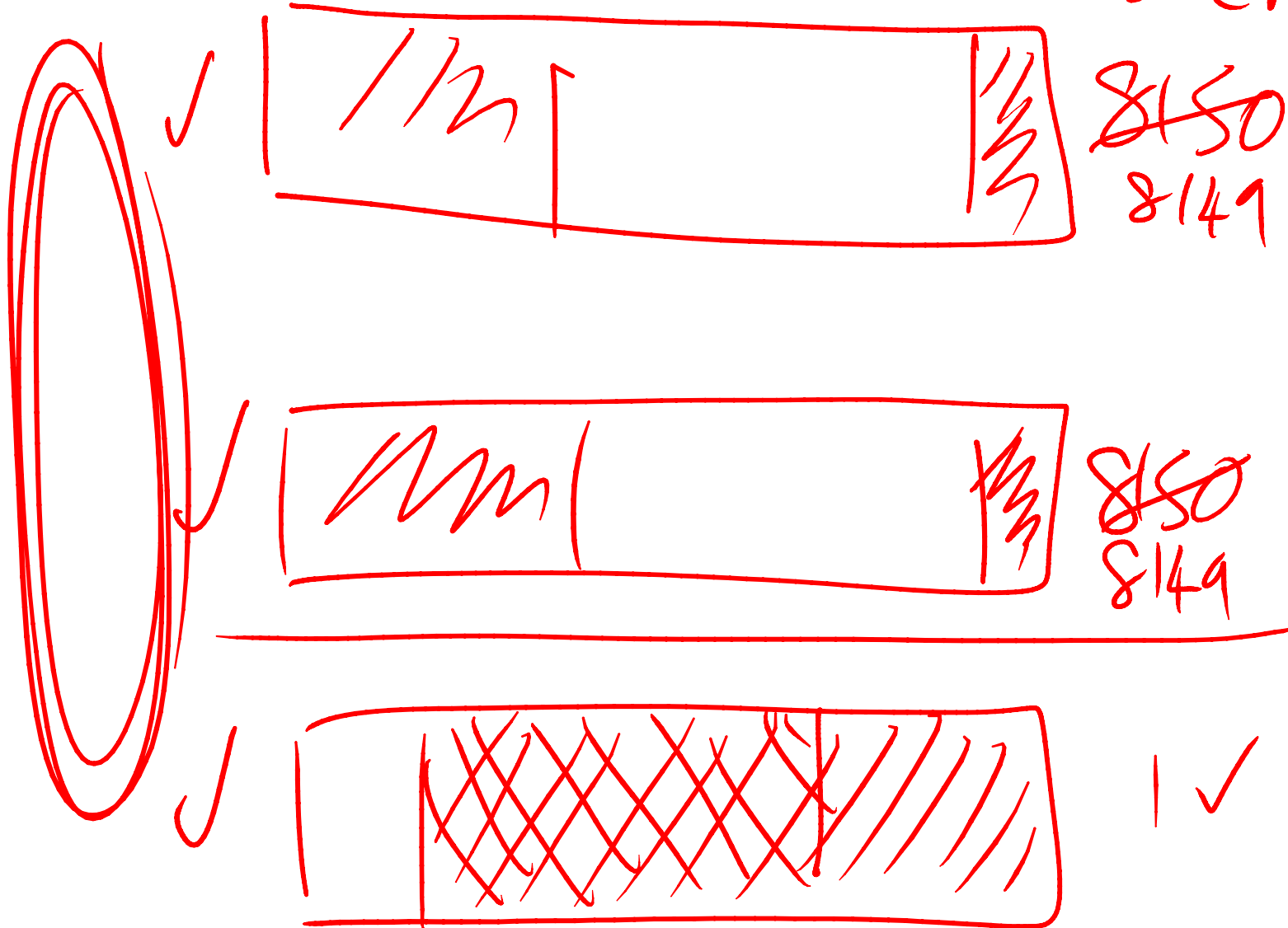


http://en.wikipedia.org/wiki/Standard_RAID_levels

M1S13 Explaining 2014 Buffer Pool Extension. It's like have L1-L3 caches for your data in memory. BPE sits in between memory and the I/O subsystem.



WEIGHTING



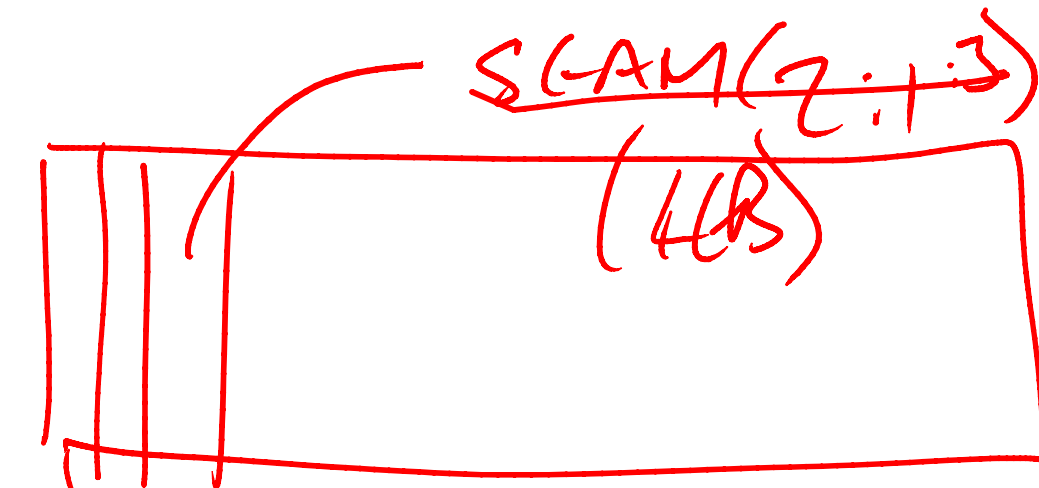
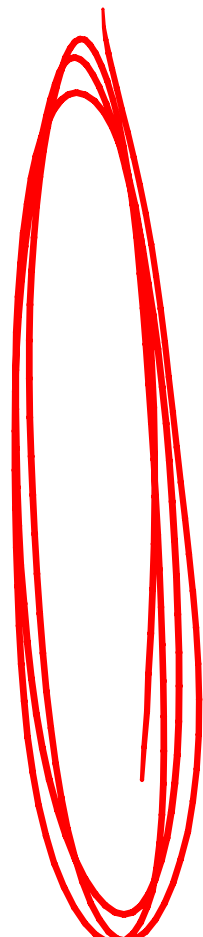
M1S30 Explaining proportional fill and round robin. Adding another file to a full filegroup does not allow for rebalancing of data.

M1S14 How to tell if you have Instant Initialization enabled if you don't have Windows access. Enable these two trace flags and create a dummy database. If you see a line in the error log about zeroing out the data file, you do NOT have Instant Init enabled. Get Windows admin to grant Perform Volume Maintenance Tasks privilege and restart SQL Server.

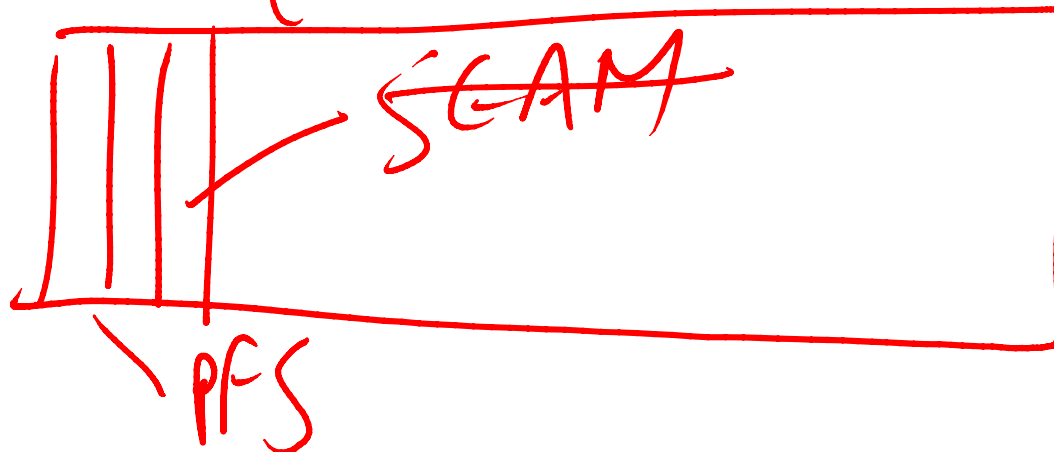
3004

3605

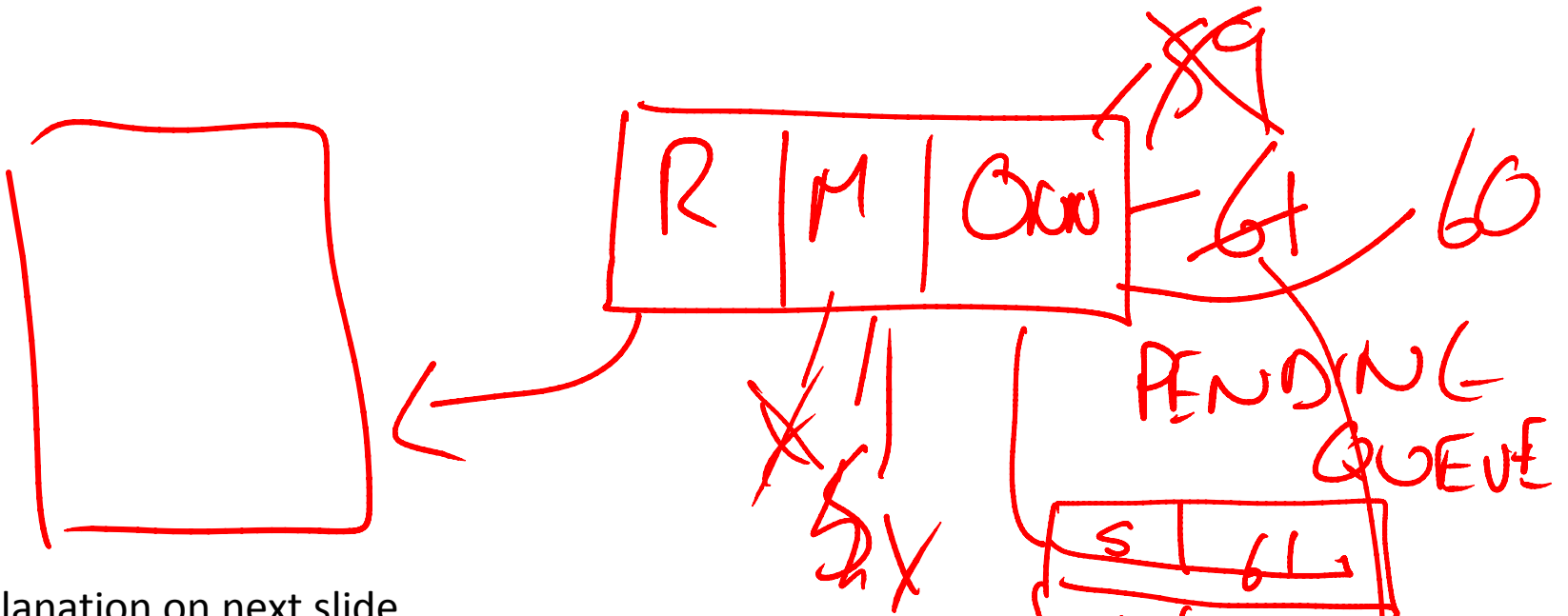
DBCC TRACEON(3004,
3605, -1)



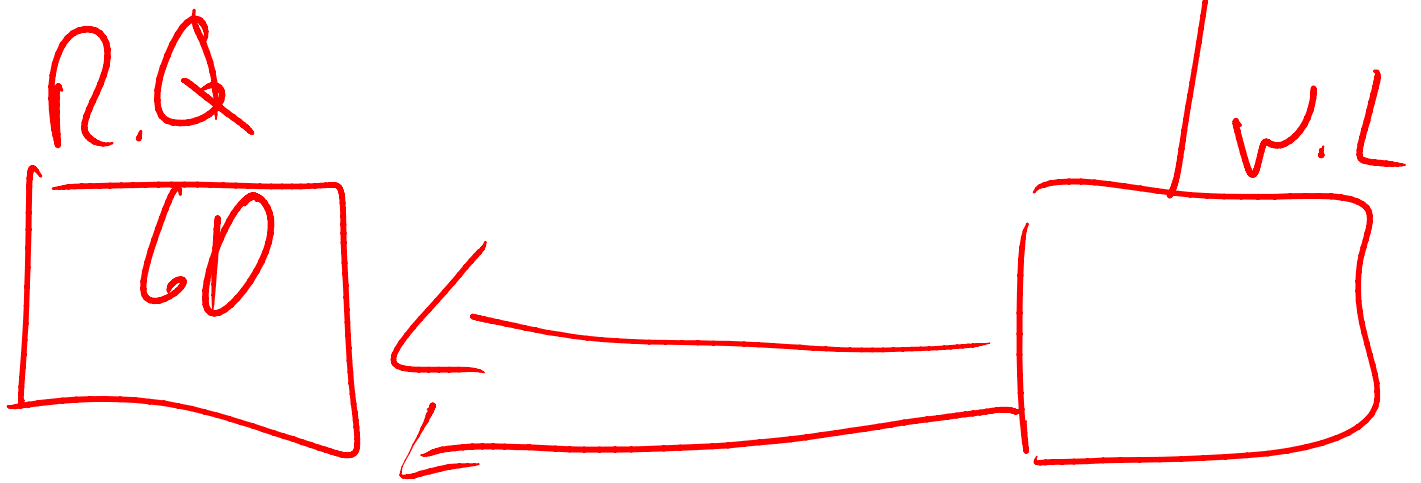
PFS(2:1:1)
(64MB)



M1S41 Explaining PFS and SGAM contention in tempdb and how adding multiple files spreads the contention over multiple files, thus reducing it.



Explanation on next slide



M7 Explaining how the lock pending queue works.

At time:

X: Thread 59 holds the lock

X+1: Thread 61 tries to acquire the lock and can't. It registers a callback routine for itself on the pending queue in the lock value block. Then it suspends itself, moves off the CPU and onto the waiter list.

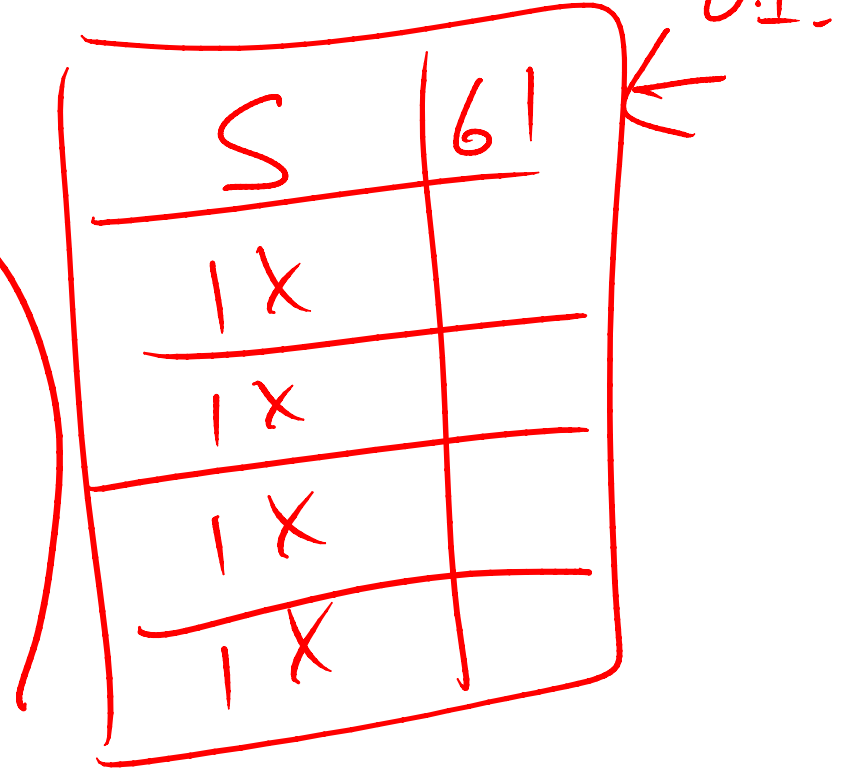
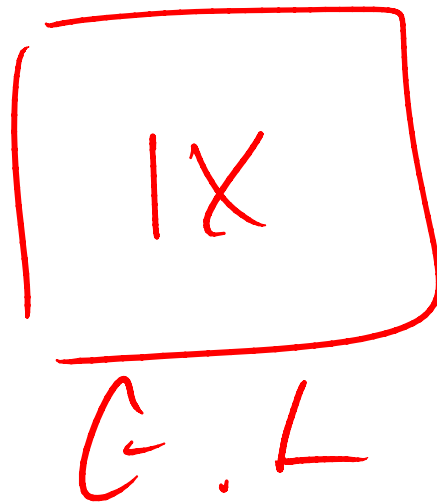
X+2: Thread 60 tries to acquire the lock and can't. It registers a callback routine etc etc, and it behind thread 61 in the pending queue.

X+3: Thread 59 releases the lock, which grants the lock to thread 61. Thread 61's callback routine is called, signaling it and allowing it to move to the runnable queue on it's scheduler. Thread 60 is now at the head of the pending queue for the lock.

X+4: Thread 61 releases the lock, which grants the lock to thread 60. Etc etc. There is now no pending queue for the lock.

2014 ONLINE

INDEX
P.Q



More on the pending queue... with 2014 WAIT_AT_LOW_PRIORITY for online index operations, the S lock for the online rebuild in the queue can be jumped over by other locks until the wait timeout expires.

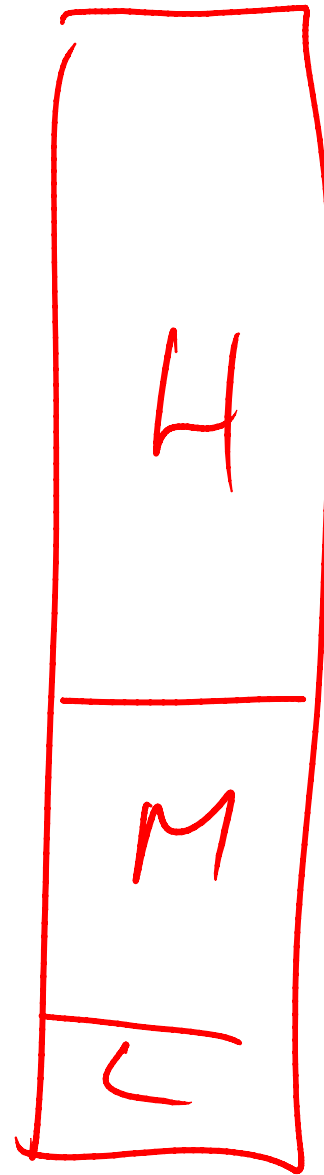
RESOURCE POOL

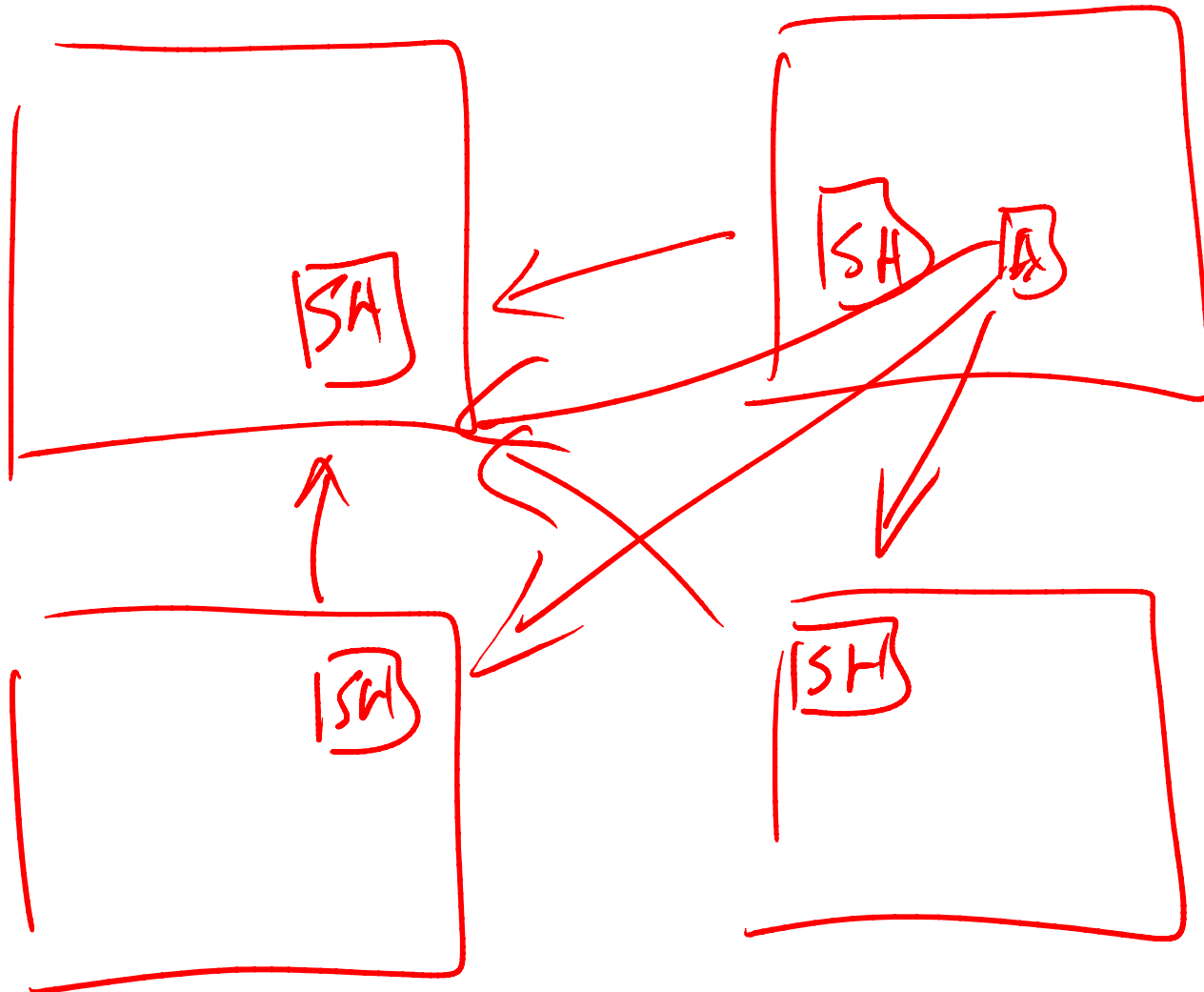
> / WORKLOAD GROUPS

H / M / L

9 : 3 : 1

M7S17 Explaining how the Runnable Queue is a true FIFO (First-In, First-Out) queue unless Resource Governor workloads groups and group priorities are being used. High-Medium-Low priority equals 9-3-1 threads through the Runnable Queue.

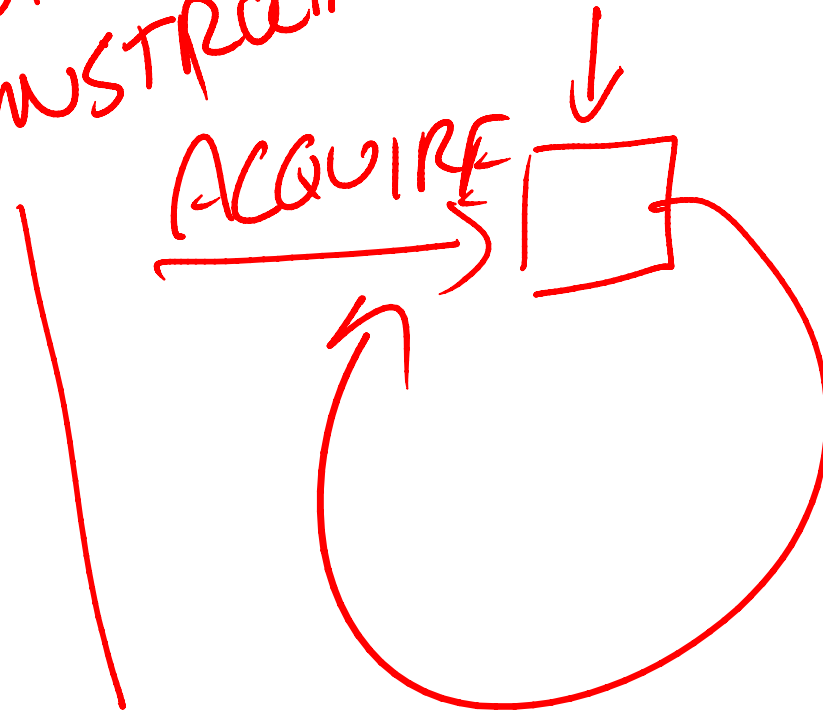




NUMA

M7S35 Explaining about
superlatches.

INTERLOCKED
INSTRUCTION SPINLOCK



SPINNING

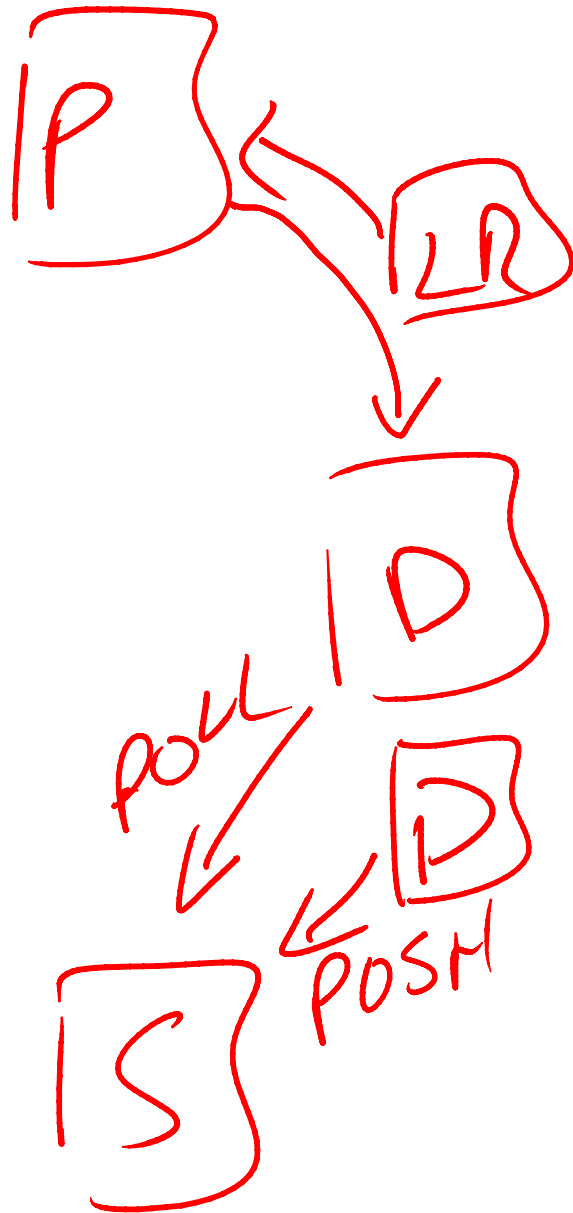
1000

COLLISION

TST_SET_IF_CLEAR

BACK OFF

M7S62 Explaining how spinlocks work. Code does a test-and-set-if-clear on the memory that implements the spinlock. If it can't get it, it spins (tries again) up to 1000 times and then backs off.

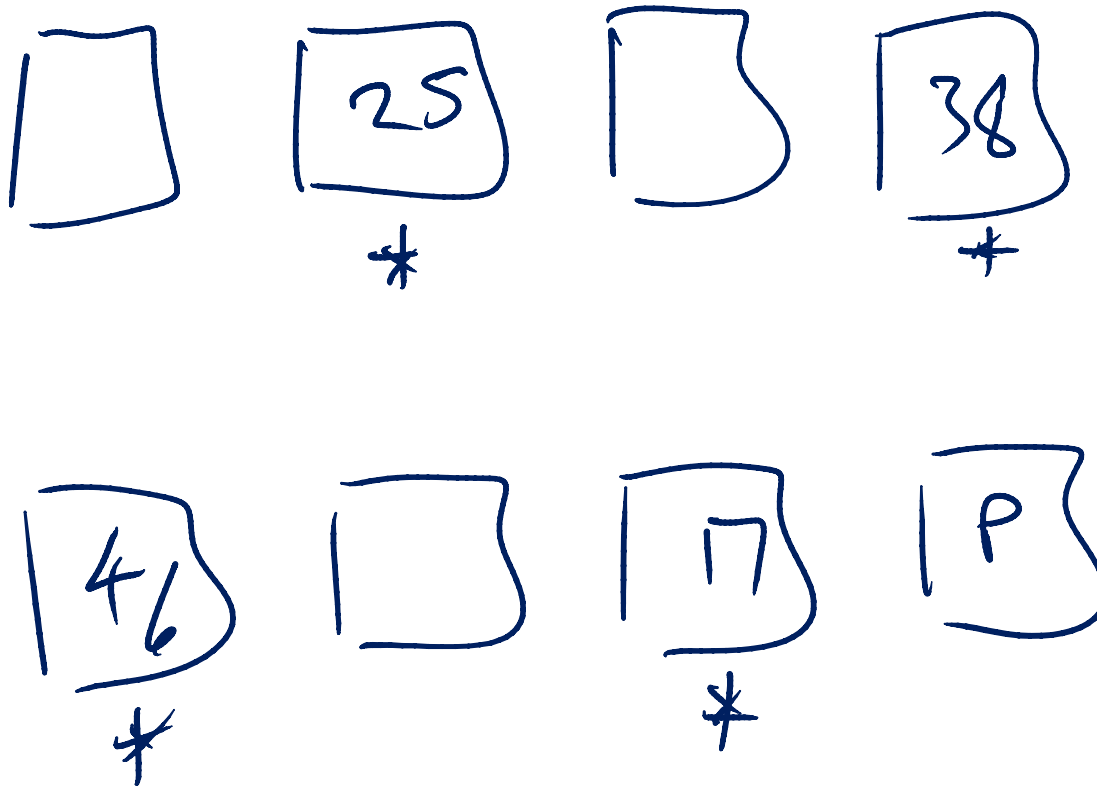


M7S52 Explaining how transactional replication works. The Log Reader Agent reads from the publication log, harvesting committed transactions and putting them into the distribution database. The Distribution Agent either pushes or pulls them to the subscription database. Pull subscriptions are more efficient and less prone to bottlenecks.

Question was about whether transactional replication can affect WRITELOG wait times. No, not unless the read workload really affects the write workload.

8 cores

DOP 4



From Erin's module: with DOP 4, a query will only use 4 schedulers, no matter how many threads it needs for any parallel operator ($\text{max} = \text{DOP} \times 2 + 1$)