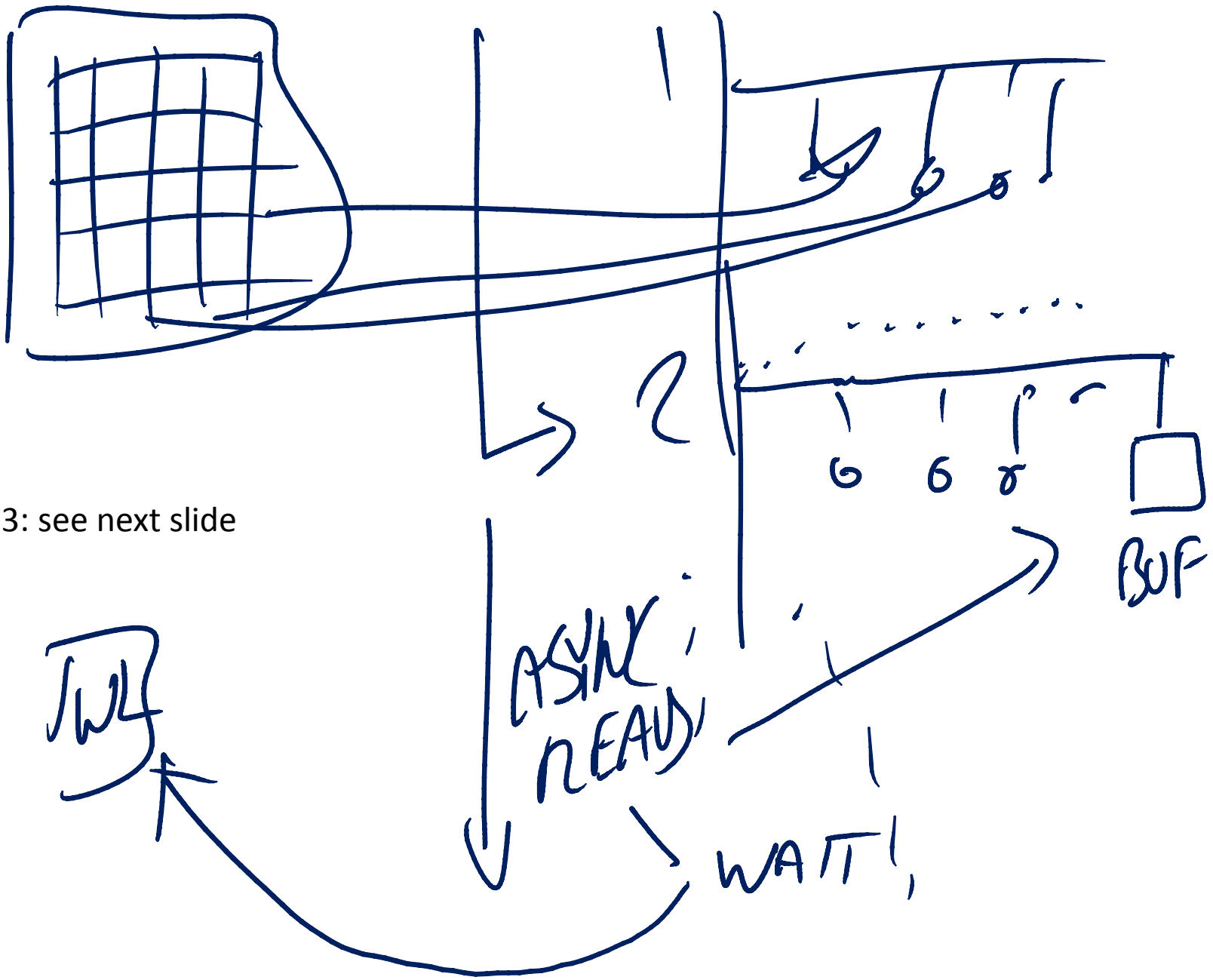


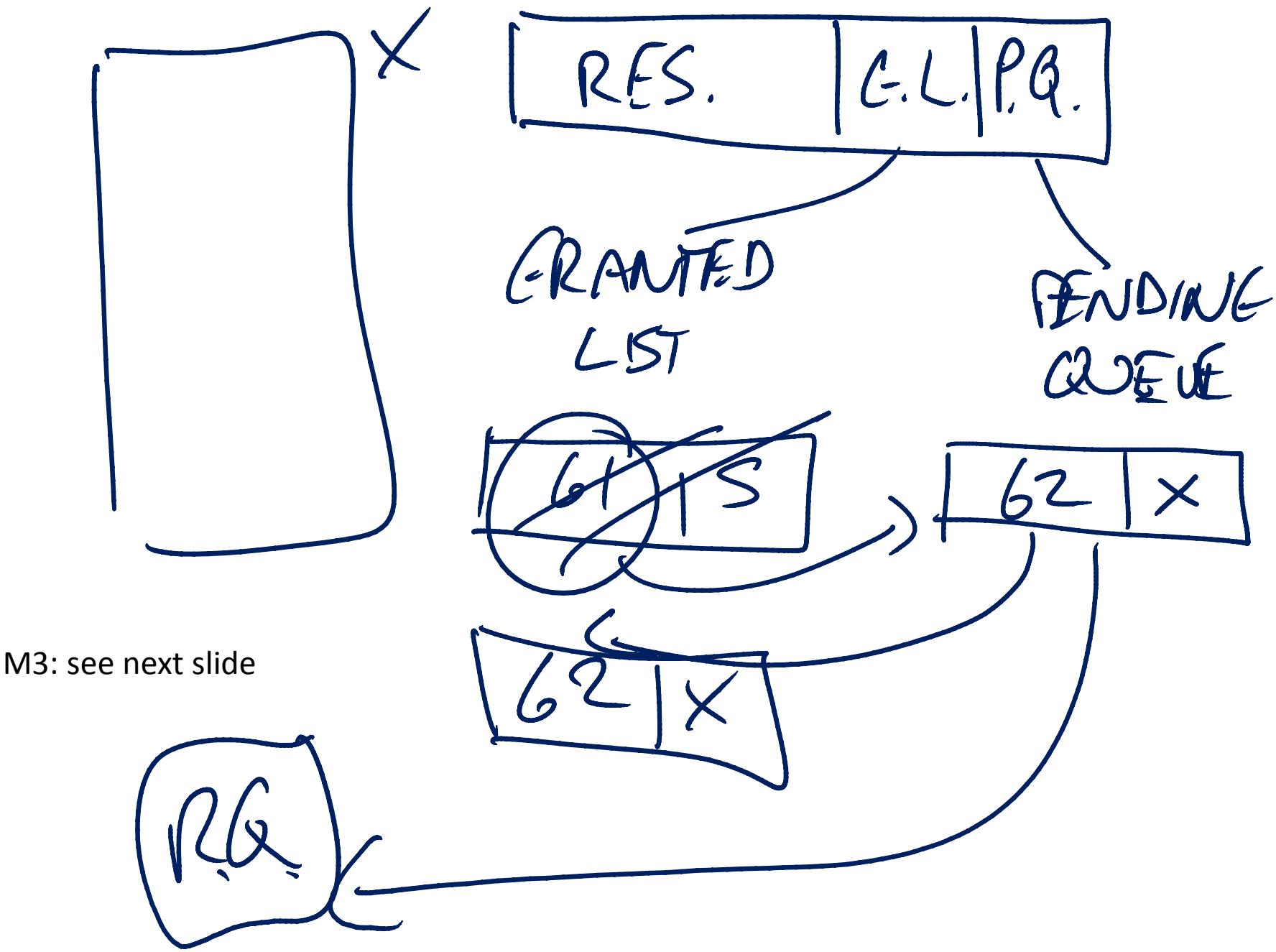
M3: see next slide

M3 Explaining how a physical I/O works

Thread looks in the hash lists for the BUF structure for the page it wants.
It doesn't find it.
It creates the BUF structure and latches it EX (this prevents any other threads who also want that page from trying to read it from disk).
It starts the async physical read.
It puts itself on the I/O waiter list for the scheduler.
It comes off the processor and is now suspended.

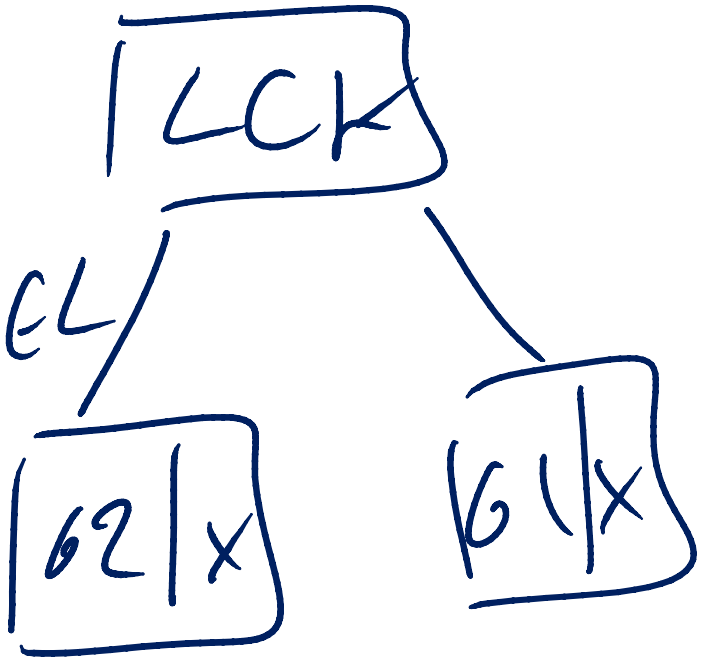
Some other thread on that scheduler, when it's about to get suspended, will check to see whether any of the I/Os for threads on that scheduler have completed. If so, whatever thread is waiting for the completed I/O is moved to the runnable queue.

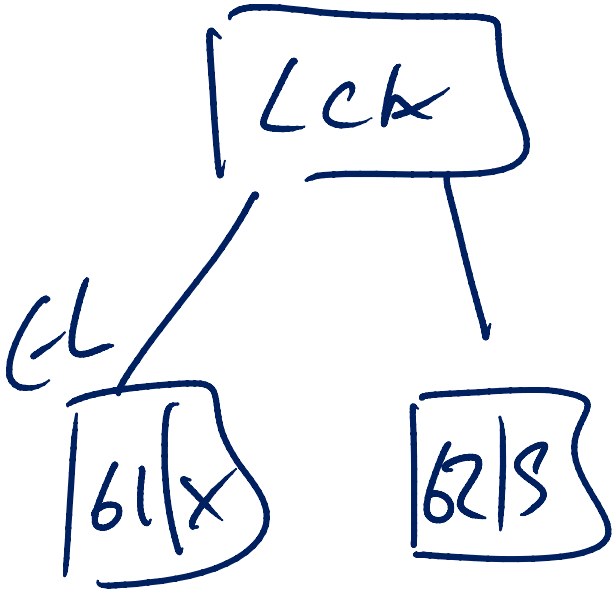
Remember: threads schedule themselves cooperatively.



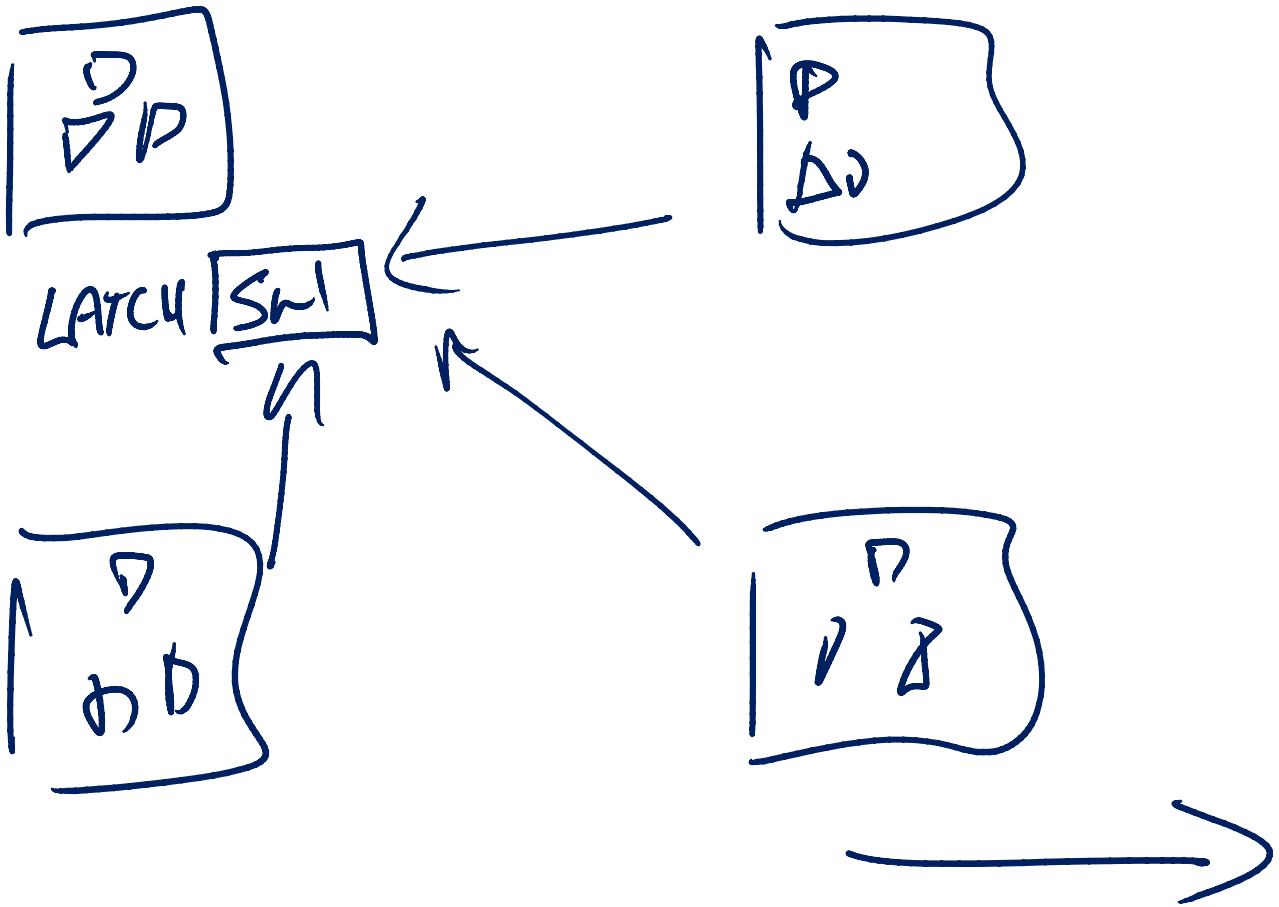
M3: see next slide

M3 Explaining how the lock pending queue works.
At time:
X: Thread 61 holds the lock in S mode
X+1: Thread 62 tries to acquire the lock in IX and can’t. It put itself on the pending queue in the lock value block, which is the waiter list for this resource. It then suspends itself, and moves off the CPU.
X+2: Thread 61 releases the lock. It checks the pending queue for locks that can now be granted ownership of the lock (taking into account any other threads still holding the lock). In this case, thread 62 is granted the lock in IX mode. Thread 61 then moves thread 62 to the runnable queue of the scheduler where thread 62 was running.



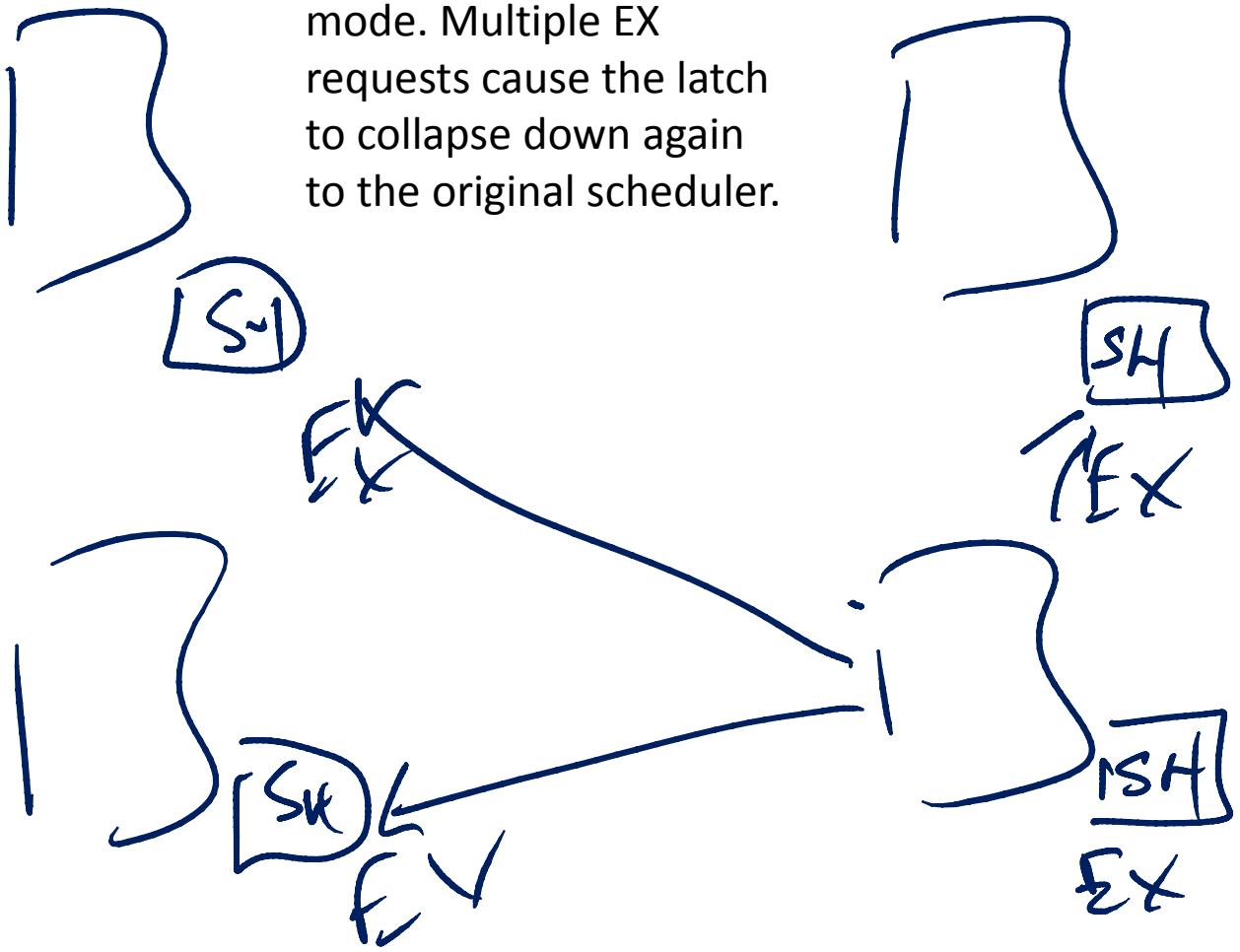


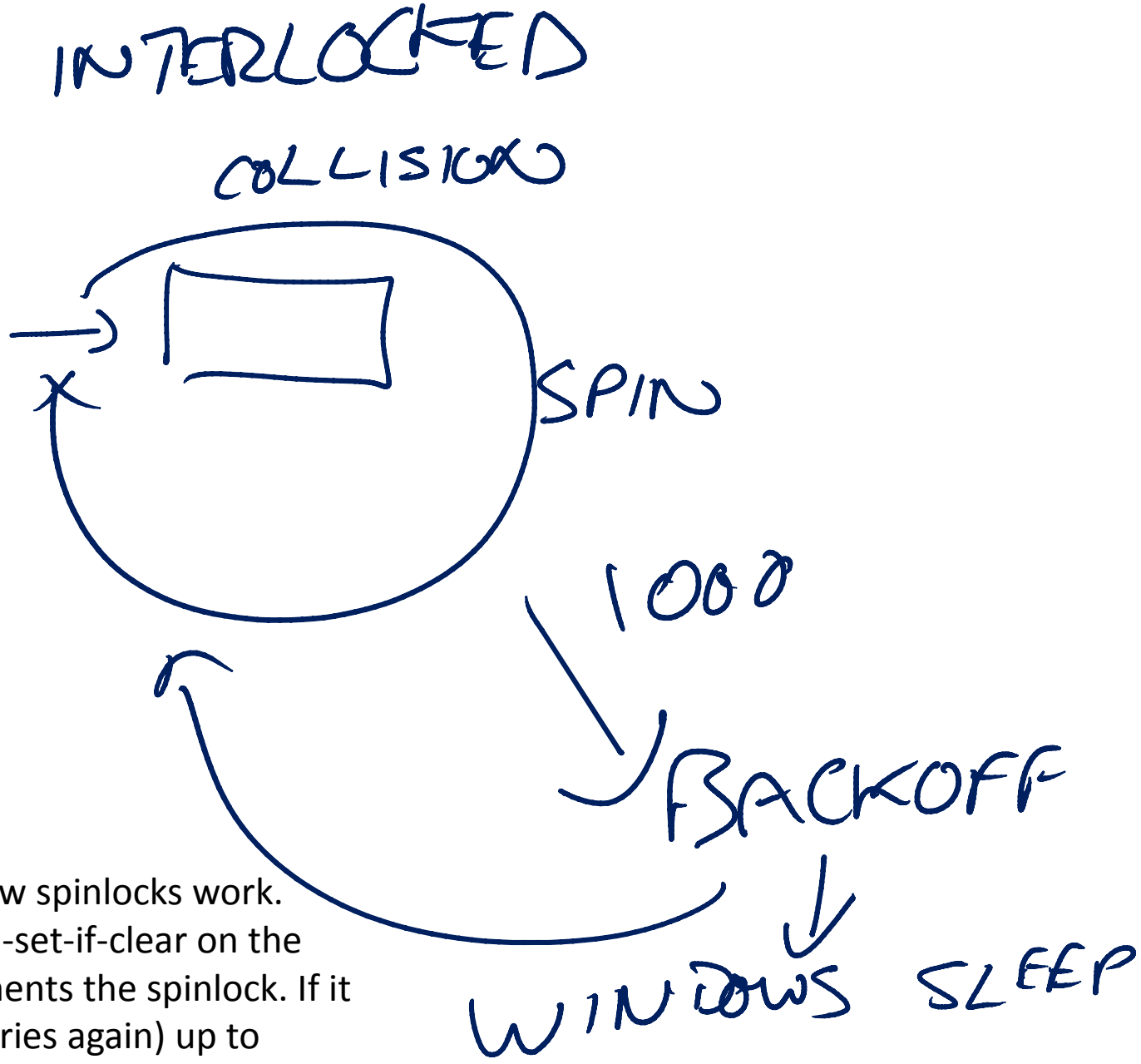
Deadlock scenario: Spid 62 X locks the first page. Spid 61 X locks the second page. (GL = Granted List of locks for that resource.) Spid 61 then tries to X lock the first page. Spid 62 then tries to S lock the second page. That's a deadlock. The deadlock monitor notices and picks a victim, based on who's done the least amount of work. Let's say it's 62. Thread 62 (waiting for the S lock) is signaled (moves to the runnable queue) and the deadlock monitor has set a bit in Thread 62's memory that indicates it's been chosen as a deadlock victim. Thread 62 then initiates rolling back the transaction that it's part of.



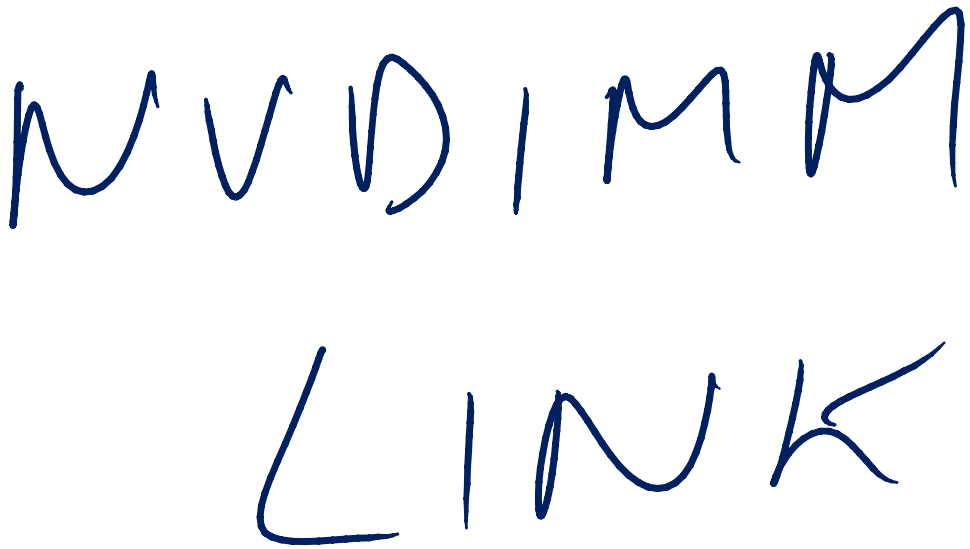
M3S38 Explaining about superlatches. This is when there's only one latch and it's owned by one scheduler (big boxes are scheduler)…

M3S38 Explaining about superlatches. This is when there's one latch per scheduler, in SH mode. Multiple EX requests cause the latch to collapse down again to the original scheduler.





M3S43 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 (generally) 1000 times and then backs off.



M3S73 The link to the NVDIMM blog post is
https://sqlskills.com/p/038

M3S80s Keeping MAXDOP set to size of a NUMA node helps to avoid foreign memory accesses, which are expensive. Check in Buffer Node perfmon object. Big boxes are NUMA nodes. Small boxes are buffer pool partitions.