HW4 COL331

Shivanshu Verma
2015CH70186

Spin locking-

Q1. Why kernel panicked?

The kernel panicked because by enabling interrupts after acquiring the lock, interrupt can occur within a critical section and interrupt handler could access the shared data (which was also being accessed within the critical section) and mutual exclusion will get violated.

Q2 Why kernel did not panic? Why different behaviour in file-table-lock and ide_lock?

The difference is, in case of ide-lock, spinlock was implemented inside the kernel whereas in case of file-table lock, spin lock was implemented in user mode. In user mode, spin locks don't need to disable/enable interrupts as kernel's interrupt handlers are not expected to touch user data. That is why, ~~kernel~~ kernel didn't panic in case of file-table-lock.

Q Why release() clear lk→pcs[0] & lk→cpu before clearing lk→locked?

⇒ if we release lock before clearing cpu and pcs, another thread could have taken the lock which leads to 2 concurrent accesses to the cpu & pcs variable
⇒ The RACE condition

Uniprocessing Locking —

Implementation 1: No, it won't work. Suppose some process P1 acquires the lock, and just after it enables interrupts, an interrupt occurs, now another process wants to lock(L), then it will go inside L and disable interrupts and so it will spin forever as the P1 lock is not released and P2 wants lock, but interrupts are disable.

⇒ sti( ) should not be in acquire or Lock(L).

Implementation 2:

It is also the same implementation as the previous one. ~~Here~~ also, some P1 acquire lock and enable interrupts, then some P2 might acquire the same lock, disable interrupts & ~~so~~ spins forever. ⇒ This is also not valid.

Sleep & Wakeup

Yes, this is correct. because producer and consumer need to be ~~interrelated~~ inter-linked. as they share the same queue..

It means, if $q = 0$, read (consumer) should sleep and producer(write) should wake up. Using different channels will require some more conditions and will complicate things.

When producer calls wakeup(q), ~~all the~~ both producer and consumer ~~will wa~~ on the same channel will wakeup. but as $q \rightarrow ptr != 0$, producer will again go to sleep whereas the ~~rea~~ consumer will wake up. So, only read function will execute.

Some unrelated part of code can wake up consumer thread but it won't be a problem from correctness standpoint cause if it's irrelevant, it will again go to sleep. However, this is not desirable from performance point of view.

Assignment : ~~xxxxx~~ xv6 file system

$ echo > a

Printed output: log-write 34 → fn ialloc( )
log-write 34 → fn iupdate( )
log-write 59 → fn writei( )

In first disk write, an inode is allocated on device dev for a. and written to inode region
In second, size and status of inode is being update which is also written to inode region
In third, dictionary - entry record is added to parent's directory's data blocks, which is written to data block region.

$ echo x > a

Printed Output: log-write 58 → fn balloc
log-write 571 → fn bzero.
log-write 571 → fn writei
log-write 34 → fn iupdate
log-write 571 → fn writei
log-write 34 → fn iupdate
~~log-write 571~~

first write: Allocating a disk block and written to block bitmap region.

second write: zeroing out block contents, written to data block region

third write: write x to block, written to data block region

fourth write: update a's inode, written to inode region.

fifth write: writing new line character to the block, written to data block region.

sixth write: update a's inode, written to inode region.

Why writes twice?

-> As it tried for more characters e.g. abc, I could locate 4 writes and after opening file, found out that there was a new line. So, at second time as mentioned above, it writes new line character

---

~~echo~~ $\boxed{\$ \text{ rm a}}$

printed output:

log-write 59 -> In writei
log-write 34 -> In iupdate
log-write 58 -> In bfree
log-write 34 -> In iupdate
log-write 34 -> In iupdate

first: write zero to a's directory-entry record in parent's directory's data blocks, written to data block region.

Second: update parent directory's inode (written to inode region)

third: ~~updat~~ free the disk block (written to block bitmap region)

fourth: update a's inode to mark it free (written to inode region)

fifth: update free block bitmap to mark data blocks free (written to inode region)

Assignment : ZCAV

Assignment : ZCAV

Please find the tests on USB, SSD and HDD on the next page.

Laptop Specifications

Model Name	:	Mac Book Air
Model number	:	MQD32HN/A A1466
Manufacturing Year	:	2017
Disk characterstics	:	APPLE SSD SM0128G
		Throughput : 143 MiB/s

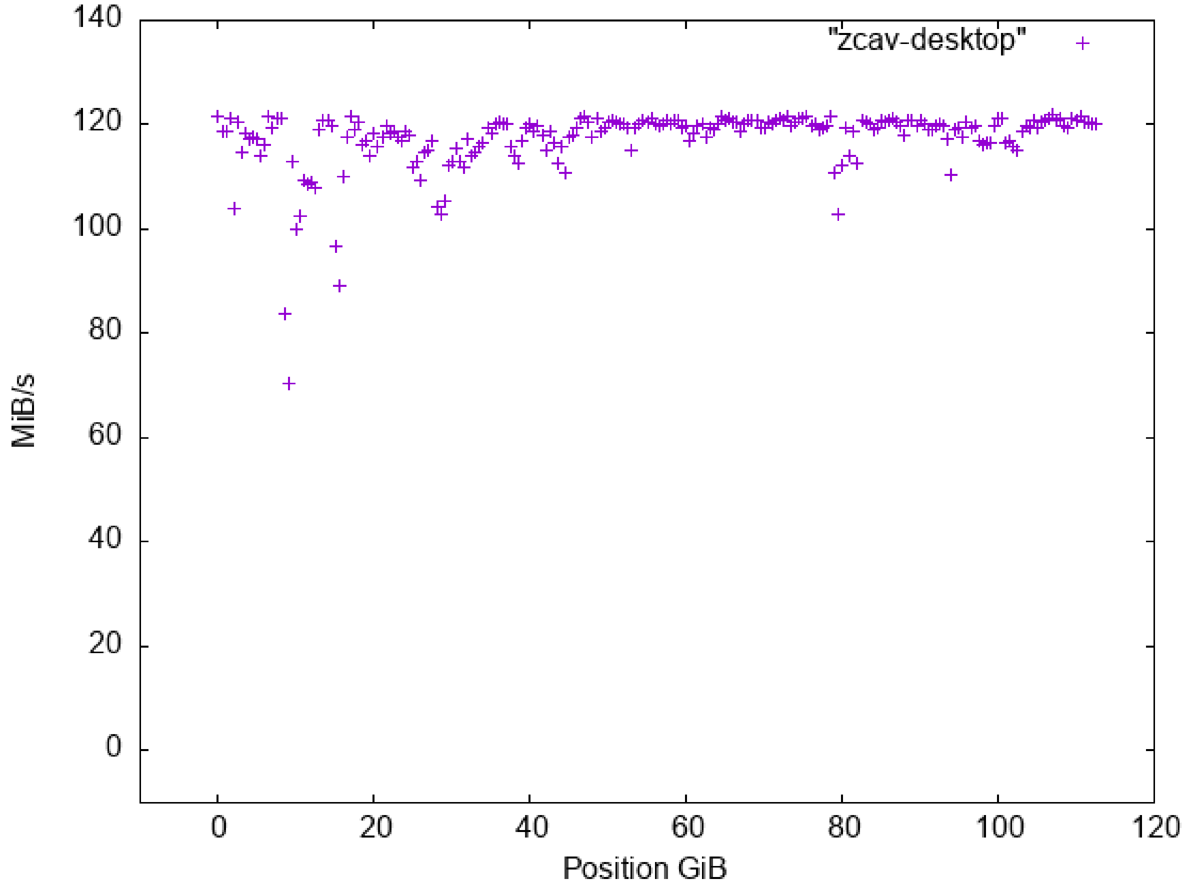

Figure 1 : ZCAV results on MacBook Air SSD 128 GB

The results of ZCAV on Macbook Air 128 GB SSD are shown in figure 1. As the characterstics of SSD are different than HDD, we could not see zones here and almost equal performance everywhere. Usually HDD has moving objects, magnetically coated platters and rotation but SSDs have no moving objects in it.

Theoritically, as per the specifications, the SSD could give a maximum performance upto 150 MiB/s but here using ZCAV, I found it to be 120 MiB/s on an average.

Out of curiosity, I borrowed my cousin's laptop to check the results on HDD as well and below are the specifications of the laptop and disk.

Laptop Specifications

Model Name	:	Asus ROG
Model number	:	GL553VE-FY040T
Manufacturing Year	:	2017 - 04
Disk characterstics	:	1 TB
		Model no : ST1000LM035-1RK172
		Seek Time : track to track → 1.5 ms
		Average → 13 ms
		Latency : 5.6 ms
		Throughput : 140 MB/s

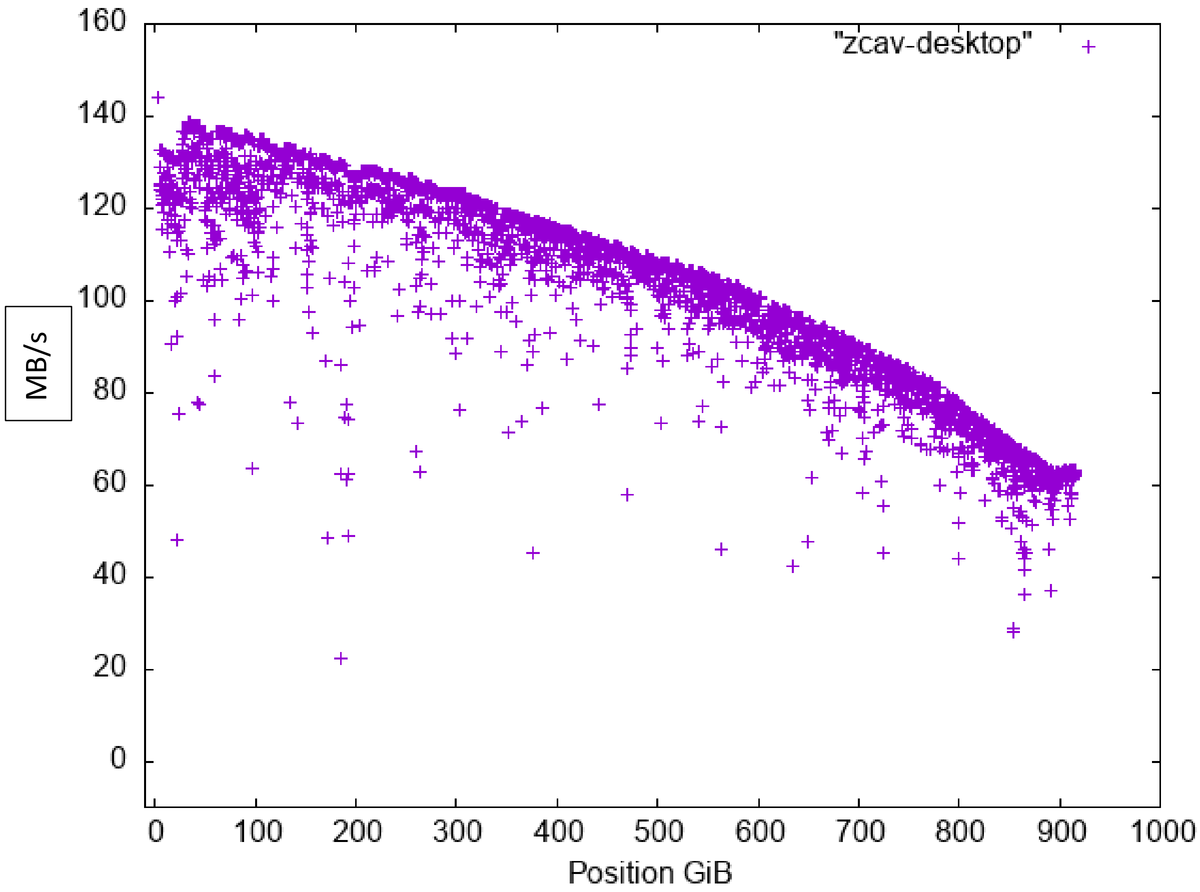

Figure 2 : Asus HDD 1TB Results

As we can see in figure 2, the graph is decreasing with the position in the hard disk. Maximum throughput of this hard disk was 140 MB/s which was nearly achieved near the postion 0 but after that it keeps on decreasing.

No of zones observed : It is nearly impossible to locate no of zones, but if we zoom in, we can find many zones, not limited to 3 or 4 only.

The mapping between sector no and physical disk layout is such that the outer tracks contain the sectors with lower addresses, therfore, the first partition allocated on a disk is significantly faster.

USB Specifications

Model Name	:	Sandisk Cruzer Switch
Model number	:	SDCZ52-008G
Manufacturing Year	:	2017
USB Port	:	USB type A 3.0
Throughput	:	20 MB/s

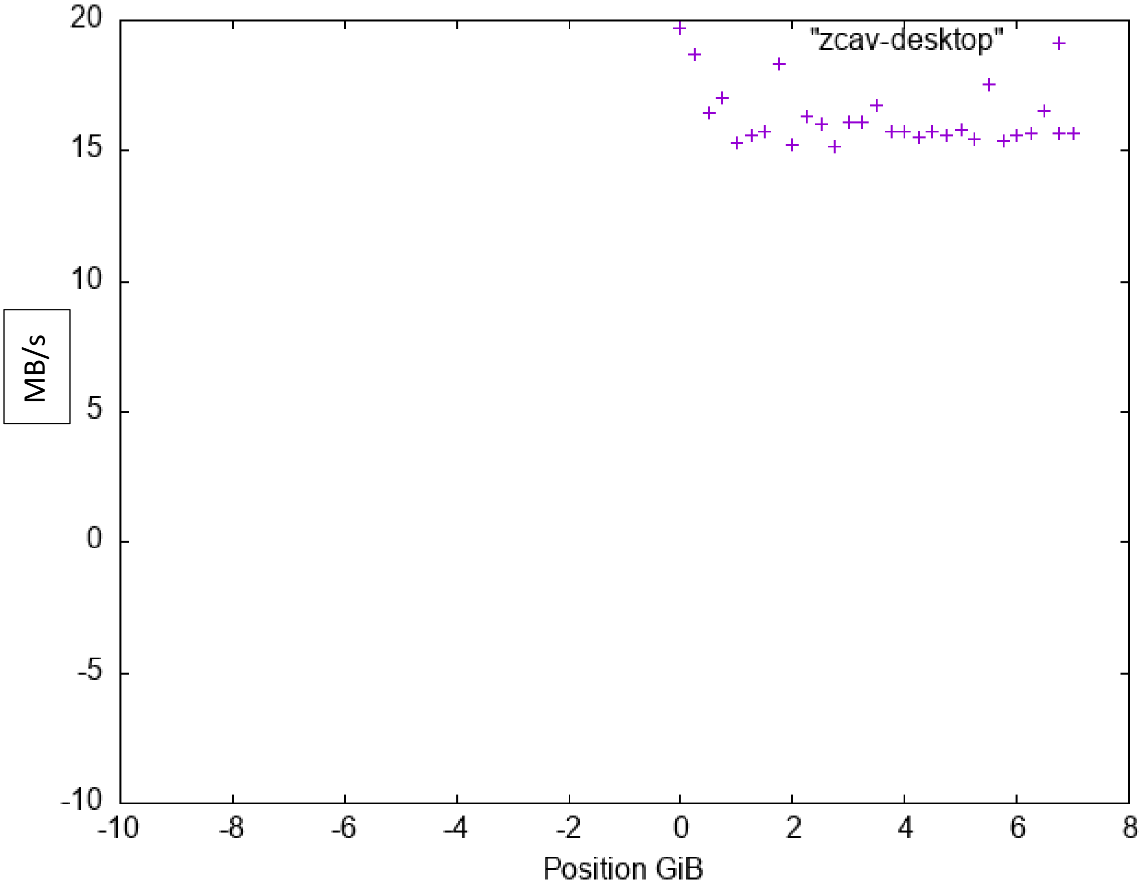

Figure 3 : Results on USB SanDisk 8GB

As we can see in figure 3, the graph is almost constant as it is a flash memory and has no moving parts in to so no zones were observed. Maximum throughput of this pen drive was 20 MB/s which was achieved sometimes but on an average the throughput was around 15 MB/s.

As ZCAV itself stands for zone circular angular velocity, this was built to find different zones of magnetic disks having rotations. That is the reason, that I borrowed my cousin's laptop despite of me having my own but with SSD. As expected, the results were almost constant for flash memories while for magnetic disks, we were able to observe a decrease in performance with position. Though we could not see the separate zones, but by zooming the figure in a smaller range, I was able to locate different zones.

The minimum speed of 1TB HDD was around 60 MB/s while some external usb hard drives also give performance upto that. Definitely, SSDs are better in performance giving around 1010 MB/s of average throughput while the maximum througput of HDD was 140 MB/s which is around 7 times lesses. But talking about HDDs only, its performance has varied from 140 to 60 MB/s which is less than 50 %.