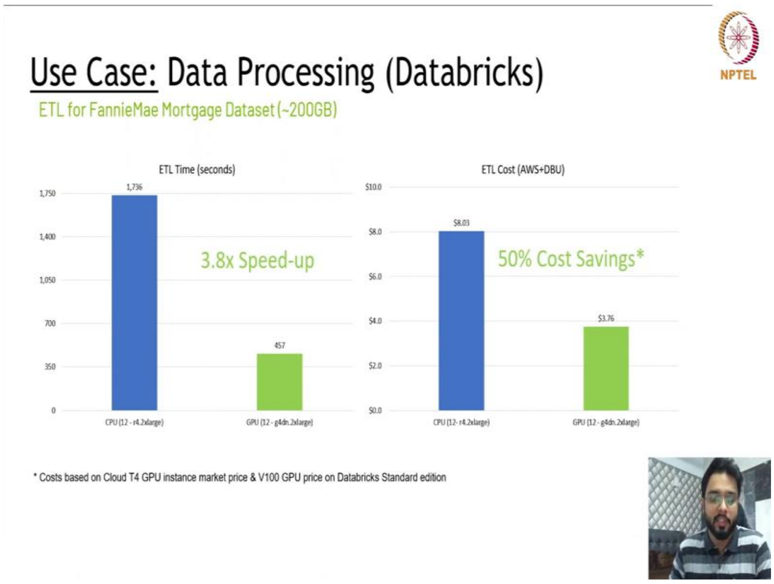
Applied Accelerated Artificial Intelligence
Prof. Saurav Agarwal
Department of Computer Science and Engineering
Indian Institute of Technology, Palakkad

Lecture - 53
Accelerated ETL Pipeline with SPARK part 2

(Refer Slide Time: 00:14)

Ok. So, I have been telling about that there is a lot of performance improvement and all that stuff; but let us understand how much is the performance improvement.

(Refer Slide Time: 00:27)



So, we picked up a use case, where we had data processing from data we did data processing on based on the public dataset of 200GB CSV which is called the FannieMae Mortgage Dataset and you can find out, I will show you the links and all in some time; everything for notebook and all available on the internet.

So, there you will find that ok; you will be seeing that ok data in the data bricks environment, if you use 12 node cluster of CPUs with r4.2xlarge GPU type on the T4 instances versus the T4 instances of cloud where you use in data AWS theg4dn to its large environment.

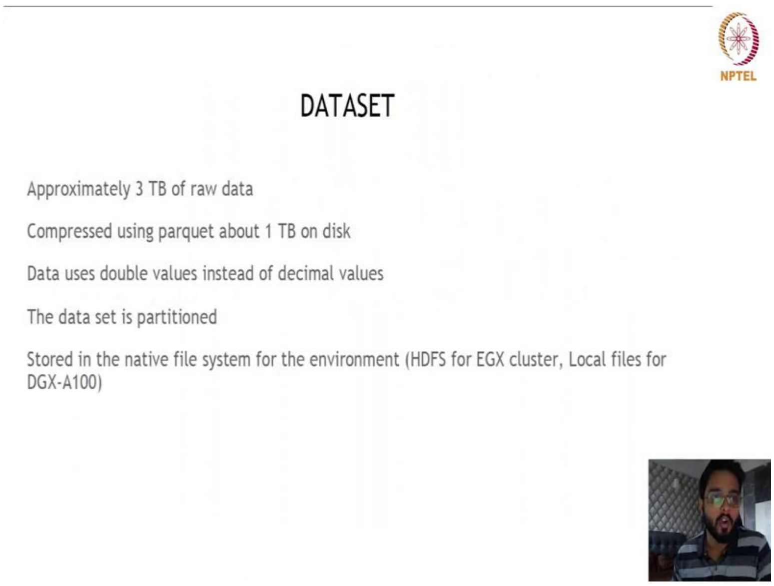
So, you will see that the same ETL which is reading a CSV file, converting into parquet format and then, reading it, again doing some aggregations and transformations and then, writing it back to the parquet file is doing the same job 4 times, almost 4 times faster than the CPU counterpart. We call it is 4 times faster; the dollar cloud bill of CPU is 8 dollars and the cloud bill of a AWS of GPU is 3.76 dollars. Since we can see that we are saving 50 percent cost as well, in addition to accelerating it at 4x performance.

(Refer Slide Time: 01:56)

So, this was a simple dataset and simple used case. But when we go to enterprises, and industry; so, we do use something called a decision support. Ah. So, decision support is something which is a industry accepted benchmarks which is cross like maintained by an independent organization. So, that any kind of processing benchmarks you need to test, we can go through this at the TPC-DS benchmark and we can compare it. Suppose if you want to compare spark with map reduce or map reduce spark 2 versus spark 3.

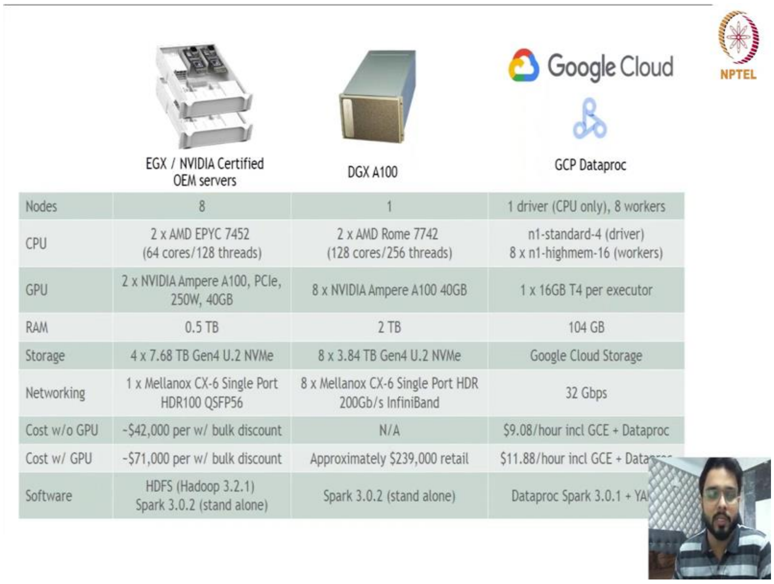
So, everyone uses this TPC-DS as a benchmark tool for doing that kind of testing. Hence, we thought that using this particular benchmark would be more preferable so that we can prove our point to the larger customers as well.

(Refer Slide Time: 02:51)



So, we use 3 terabyte of raw data; yes, we compressed the data of 3 terabyte of CSV into 1 terabyte on disk and then, we use double values instead of decimal values and the data was partitioned, stored in the native file system for the environment using the EGX cluster and the DGX A100.

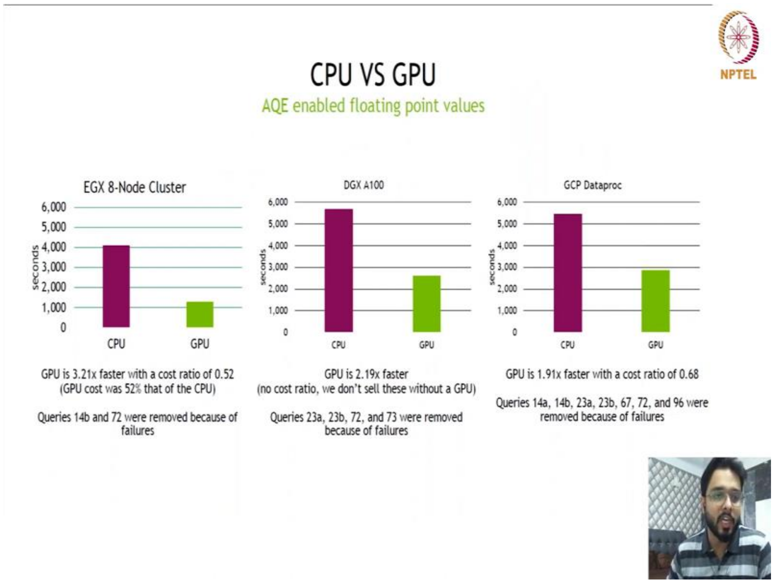
(Refer Slide Time: 03:17)



	EGX / NVIDIA Certified OEM servers	DGX A100	GCP Dataproc
Nodes	8	1	1 driver (CPU only), 8 workers
CPU	2 x AMD EPYC 7452 (64 cores/128 threads)	2 x AMD Rome 7742 (128 cores/256 threads)	n1-standard-4 (driver) 8 x n1-highmem-16 (workers)
GPU	2 x NVIDIA Ampere A100, PCIe, 250W, 40GB	8 x NVIDIA Ampere A100 40GB	1 x 16GB T4 per executor
RAM	0.5 TB	2 TB	104 GB
Storage	4 x 7.68 TB Gen4 U.2 NVMe	8 x 3.84 TB Gen4 U.2 NVMe	Google Cloud Storage
Networking	1 x Mellanox CX-6 Single Port HDR100 QSFP56	8 x Mellanox CX-6 Single Port HDR 200Gb/s InfiniBand	32 Gbps
Cost w/o GPU	~$42,000 per w/ bulk discount	N/A	$9.08/hour incl GCE + Dataproc
Cost w/ GPU	~$71,000 per w/ bulk discount	Approximately $239,000 retail	$11.88/hour incl GCE + Data
Software	HDFS (Hadoop 3.2.1) Spark 3.0.2 (stand alone)	Spark 3.0.2 (stand alone)	Dataproc Spark 3.0.1 + YA

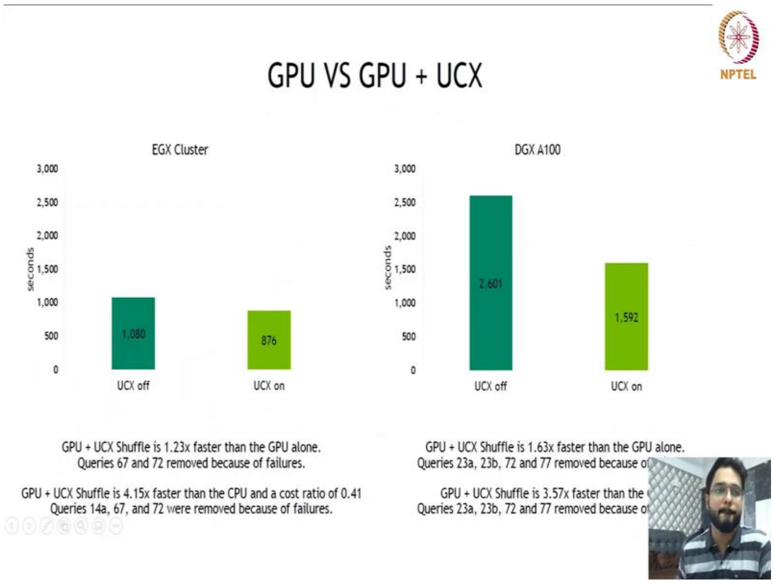
So, what we did is if you see this diagram, so the hardware we compared against Google Data Proc, DGX A100 which is combination of 8 A100 GPUs and finally, EGX which is combination of 8 servers with 2 GPUs each ok. So, we did this benchmarking on these three kind of hardware cloud on premise and two type of on premise; node based and specialized machine based.

(Refer Slide Time: 03:46)



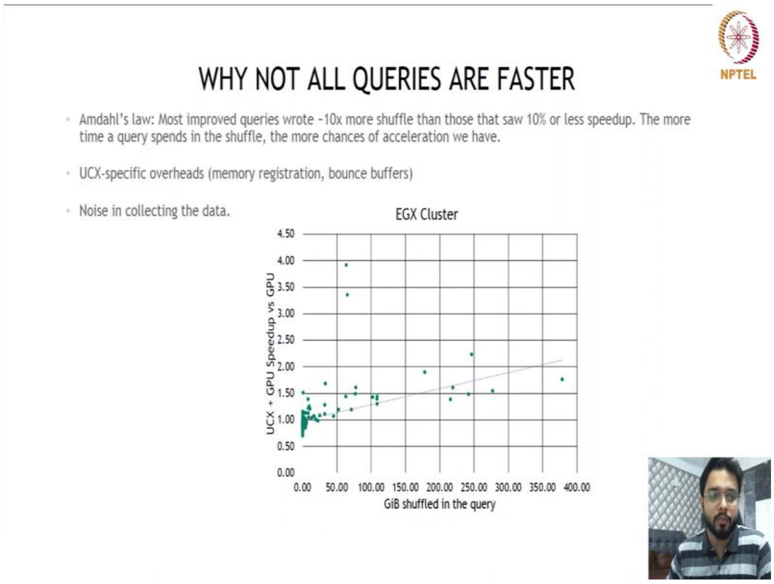
So, we found that whether it is EGX, DGX or cloud, we are definitely faster. Though, we are then fastest in the EGX kind of system, where we have 2 GPUs each in the each server and multiple such servers. DGX was faster, but it was around 2.19x faster and then, on cloud, it was almost 2x faster. The latest benchmark we have does not have even failures all the queries are running as usual.

(Refer Slide Time: 04:16)



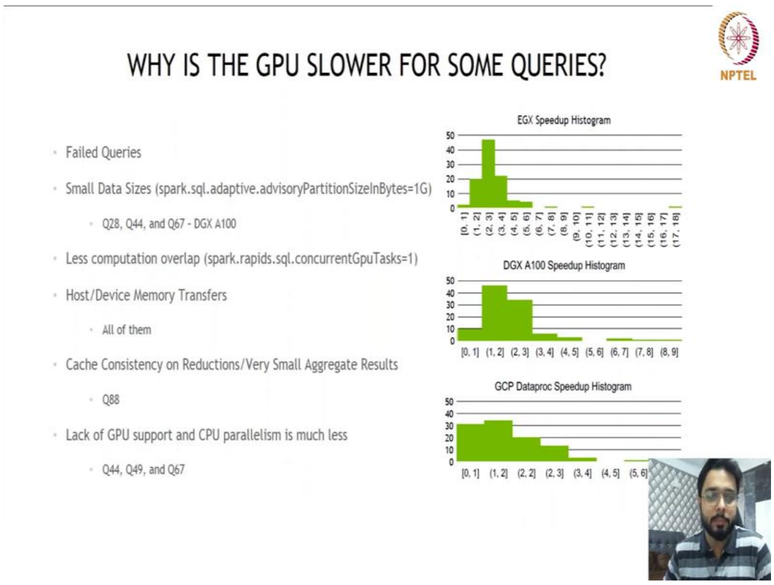
Ah So, if we add the UCX, so how much faster it is? The accelerated spark shuffle, if we add that on top of the normal GPU acceleration, we further get 1.63 X faster than GPU alone and if we compare it with CPU, we get 3.5x faster. So, hence, we see 1.5x to 2x faster for the networking acceleration as well.

(Refer Slide Time: 04:47)



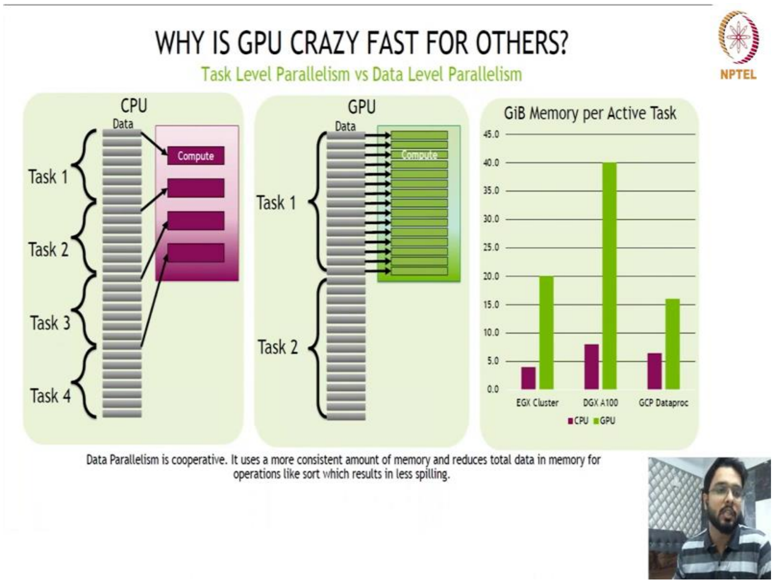
All type of queries are not faster. So, there is no one stop solution as of now that you can have to run all the queries faster. So, here if you see UCX plus GPU speed up versus GPU. So, the amount of the amount of data, we are shuffling is directly proportional to the acceleration we are getting. If we are not shuffling a lot of data, the acceleration is not that much. If we are shuffling a lot of data, then we have a lot of acceleration.

(Refer Slide Time: 05:23)



So, why GPUs are slower? There are if the data size is small, if the data queries are failing, if there is a lot of CPU usage, if there is not lack of GPU support, these kind of queries we are seeing to be slower.

(Refer Slide Time: 05:45)

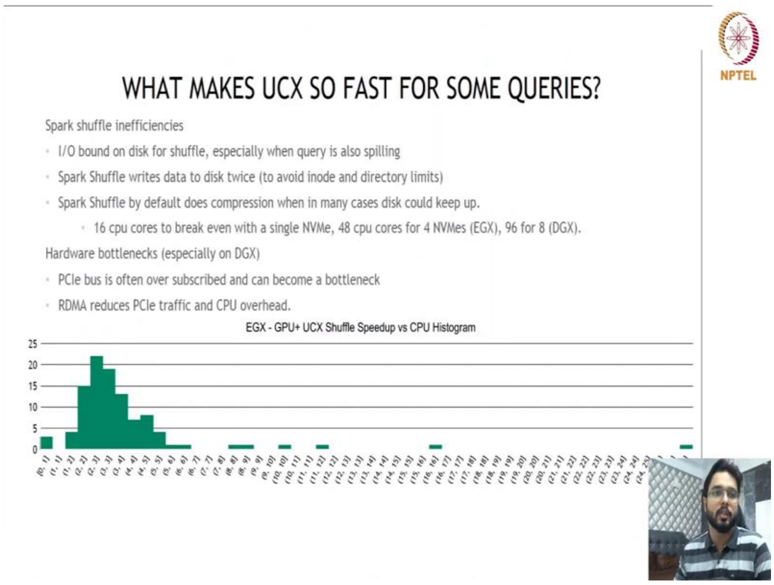


But most of the queries are faster, they are crazy fast because we are able to achieve task level parallelization. So, if you see here, the amount of data we are processing or the number of parallel tasks we are running; task 1, 2, 3, 4 amount of data we have. So, the task 1 has a very less amount of data as compared to the task 1 of GPU. So, overall throughput in addition to the parallelization is increased in processing using the GPU base system. Hence, we are seeing the fastness.

(Refer Slide Time: 06:23)

Cloud is a bit slow because it does not have networking interconnects faster; the collocation of the nodes are not there sometimes. It has shared nodes, data proc specially has external shuffle for stability, data locality is missing there. So, if you are sitting in Bangalore and if you are running cluster in US, then the data locality issue is also there. Hence it is slower.

(Refer Slide Time: 06:51)

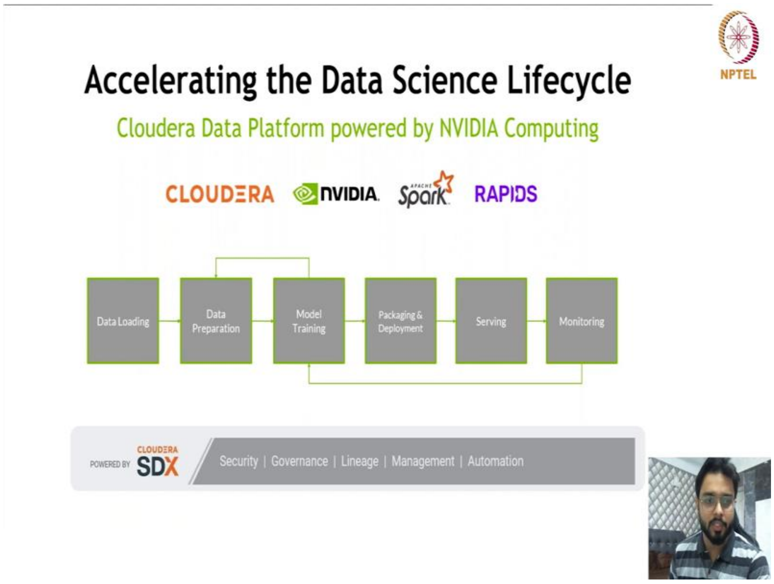


Some of the queries are crazy fast on networking interconnect because sparks of shuffle are inefficient for I/O bound or disk for shuffle which writes a lot of data to the disk and many times to the disk; sometimes there are hardware bottlenecks, where PCIe bus is involved as I was showing in the diagram. So, for all those scenarios, the thus UCX is fast and slow respectively ok.

(Refer Slide Time: 07:22)

Let us move to the Cloudera section now. So, we accelerate rapids on cloudera as well.

(Refer Slide Time: 07:36)

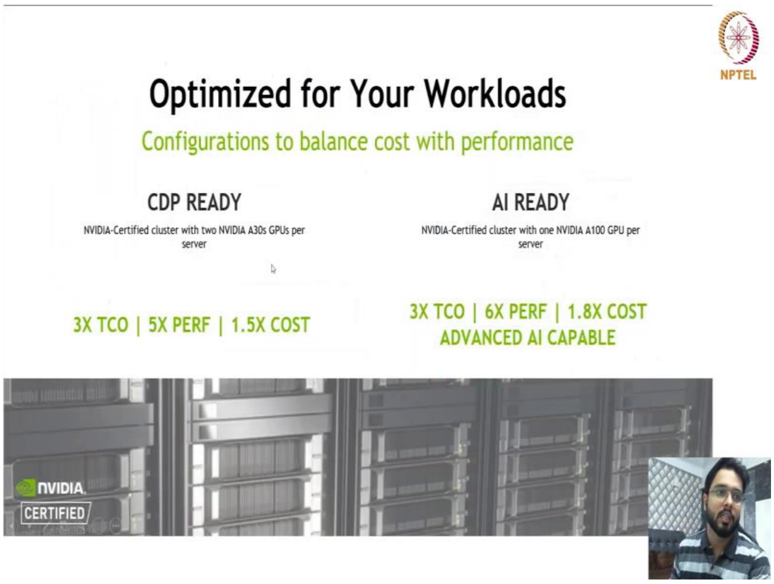


So, where we if you see a typical data science workflow which starts with data loading, data preparation, model training, packaging deployment, serving and monitoring. So, all this based on cloudera data science, data platform is made available to the users and we, at media are accelerating each part of it, if not all. ah

(Refer Slide Time: 07:59)

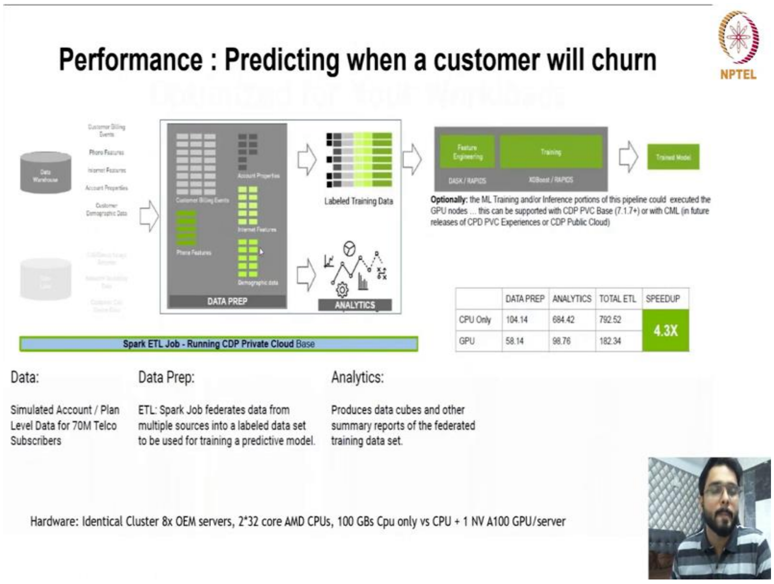
So, we know that spark, the latest version of cloudera called cloudera data platform has spark 3 enabled capability where we can enables spark 3.

(Refer Slide Time: 08:13)



And we can optimize for our workloads as well based on GPU acceleration. So, we have seen that 5X performance acceleration, where we are just saving 3X of the total cost because of the 5X acceleration. Based on the type of GPUs, it is more or less. So, if we use A30; two A30 per node, then we see that ok it is 5X faster; if we use one A100 per node, it is 6X faster.

(Refer Slide Time: 08:47)

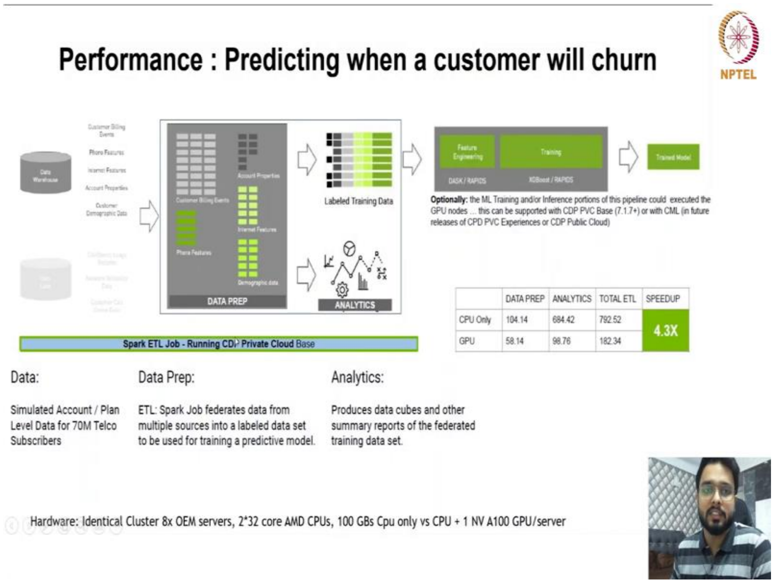


So, we did some benchmarking specifically on cloudera data platform as well, where we used identical cluster of CPU as a GPU, where we use 8 servers with 232 core AMD CPUs and 100 GBs CPU versus the same CPU plus 1 A100 GPU per server.

So, there we saw another in our data analytics pipeline, where we had data from customer billing, phone features, internet features, account properties, customer demographic data and so on and so forth. We saw that where we doing data preparation like finding out the account properties, internet features, customer billing events, phone features, demographic data and then, doing some analytics on top of it was faster.

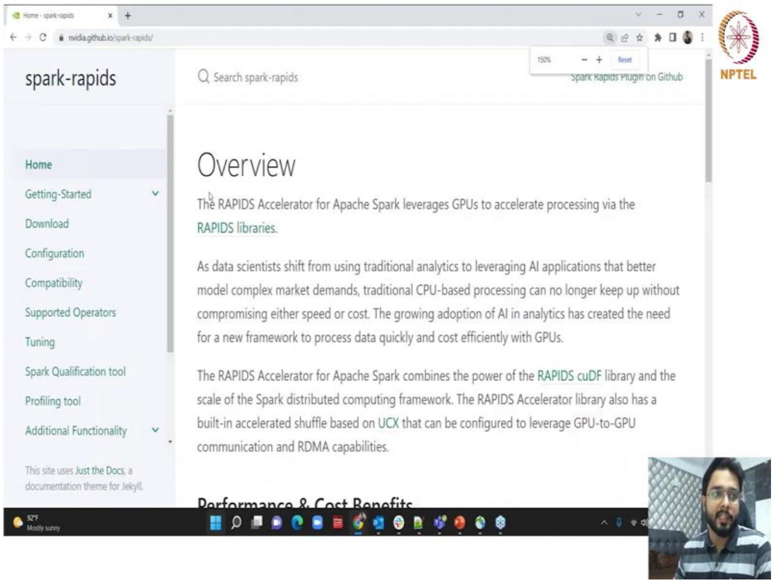
So, how much faster it was? So, it was around 5X faster; 4.3X faster. Now, this 3X faster; 4.3X faster is coming from the entire pipeline on an average, where we are combining data preparation, analytics both together. So, analytics also involve the data science portion, where we are also doing feature engineering model training and so on and so forth ok.

(Refer Slide Time: 10:11)



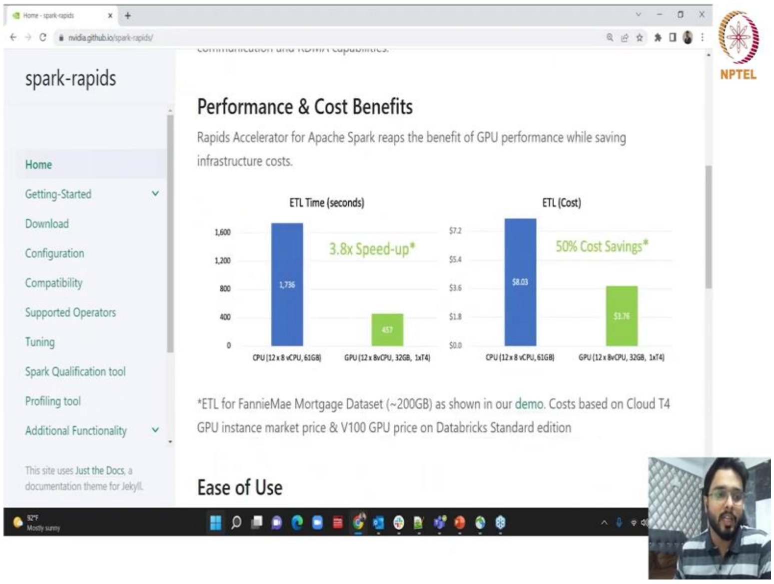
So, that is the from the theory perspective. Let us let me show you some hands on as well.

(Refer Slide Time: 10:28)

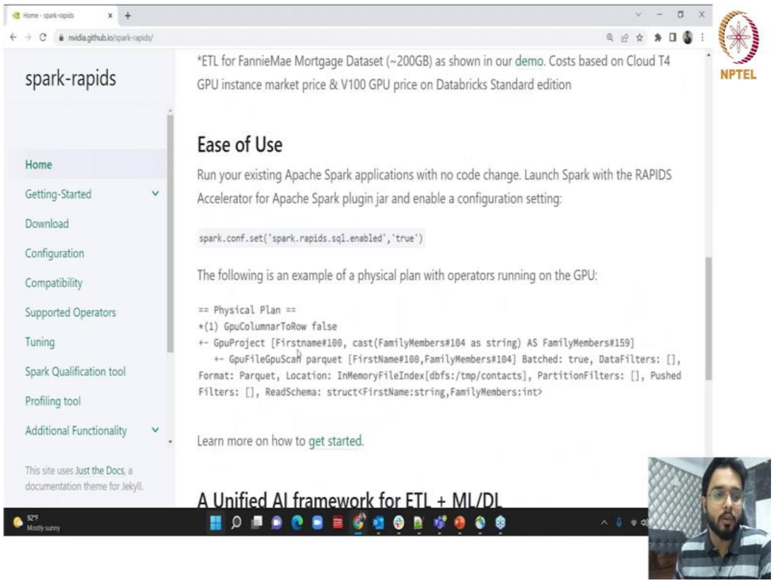


So, first of all, I will like to show this particular website, where we have the entire details about the spark on GPU acceleration. So, this is called spark nvidia.github.io/spark-rapids.

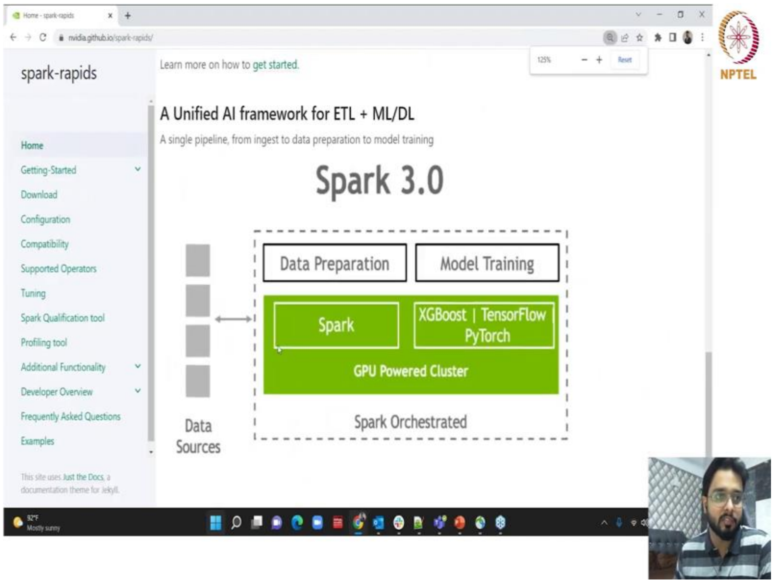
(Refer Slide Time: 10:45)



(Refer Slide Time: 10:45)

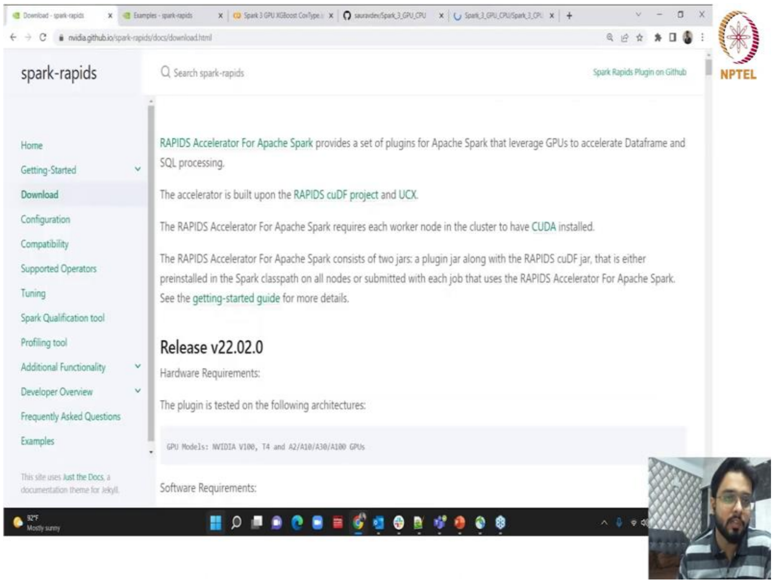


(Refer Slide Time: 10:46)

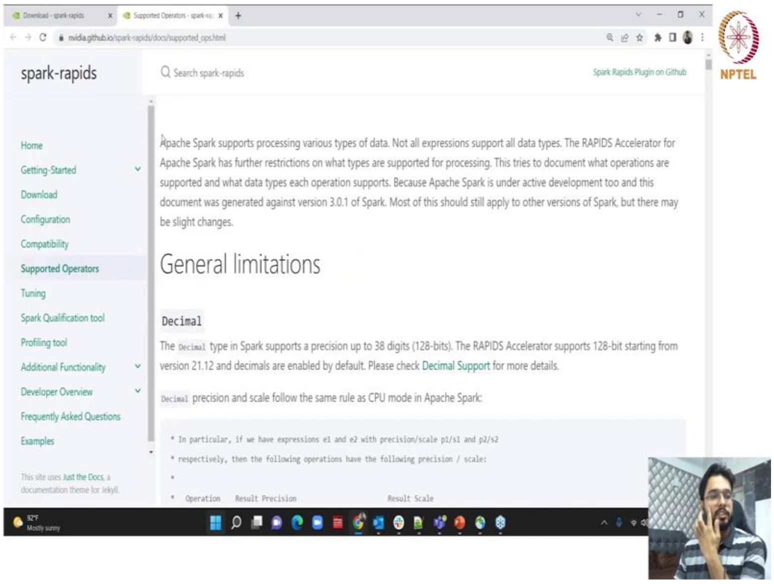


There you will see what kind of acceleration, we give how to use at a high level and all the details which I will went through today. Then, it will also give ok how to configure it On-Prem, EMR, Databricks etc etc; how to download those jars and configurate; how it to check the compatibility, what are the operative just a put it.

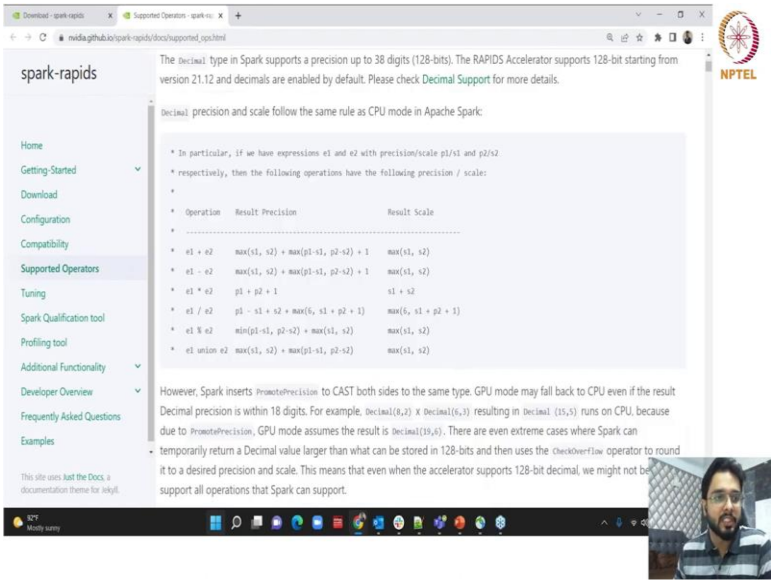
(Refer Slide Time: 11:00)



(Refer Slide Time: 11:06)

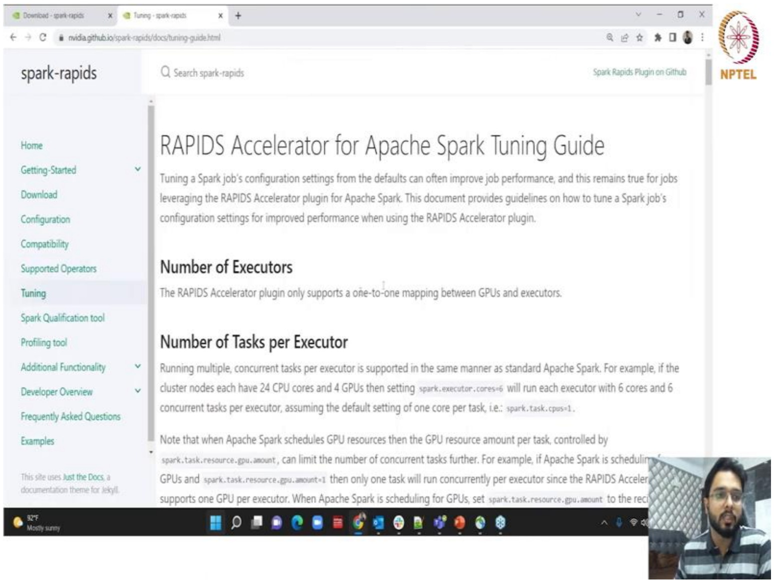


(Refer Slide Time: 11:10)



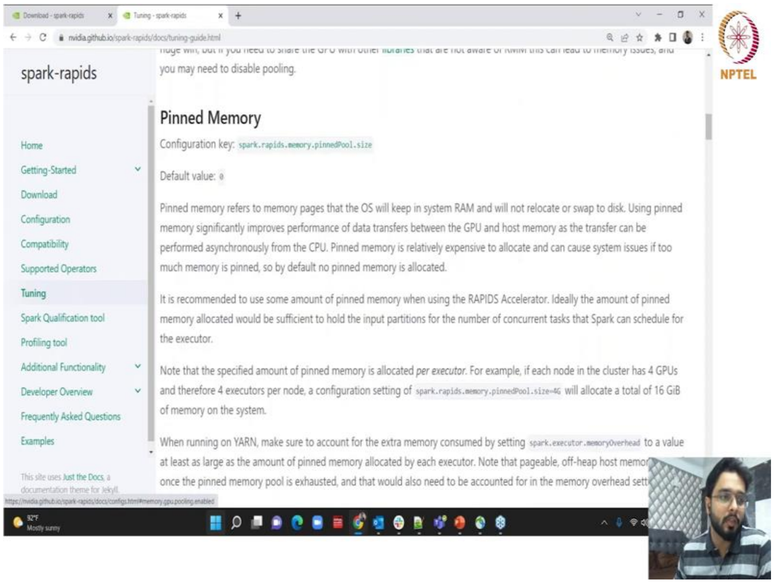
As I was saying that not everything is supported; decimal some limitation is there, timestamp calendar intervals and so on so forth.

(Refer Slide Time: 11:15)



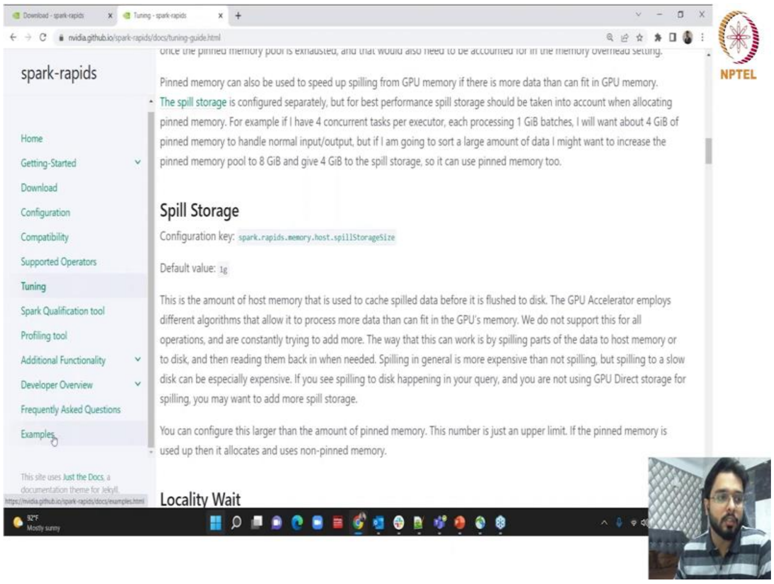
How to tune the accelerator for spark tuning?

(Refer Slide Time: 11:21)

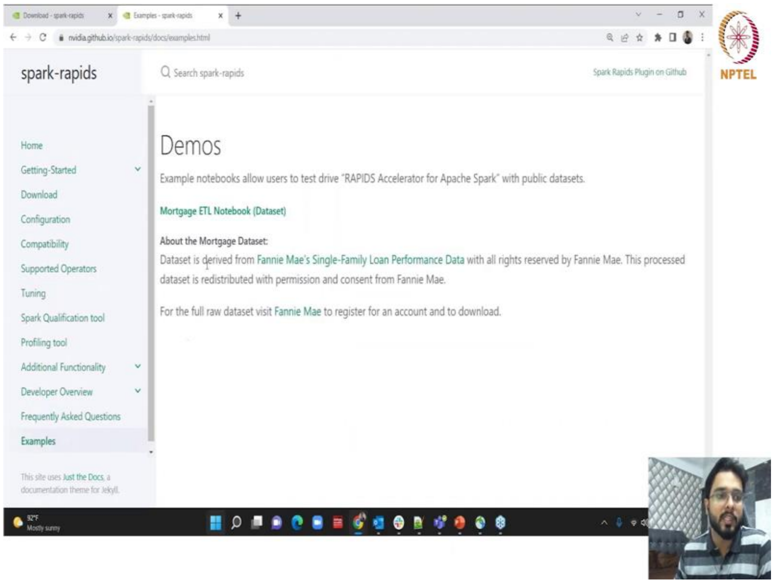


How to set the number of executor, pooled memory, print, pinned memory; all these details will find here.

(Refer Slide Time: 11:25)

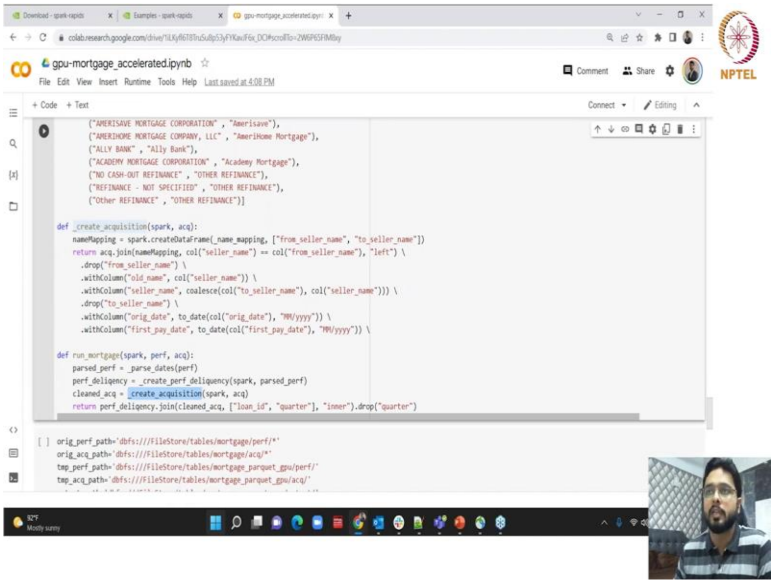


(Refer Slide Time: 11:31)

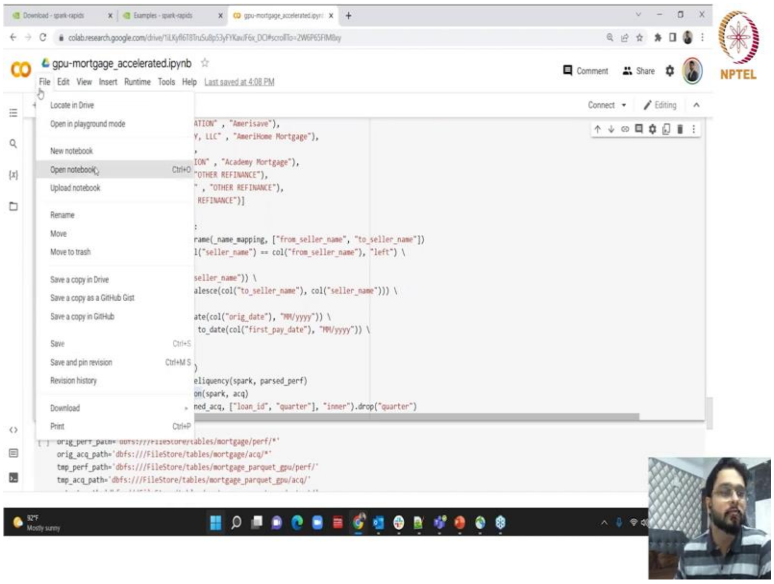


And also, if you want to do your hands on some examples of the ETL notebook, you will find. So, one such notebook apart from this is the mortgage dataset we have.

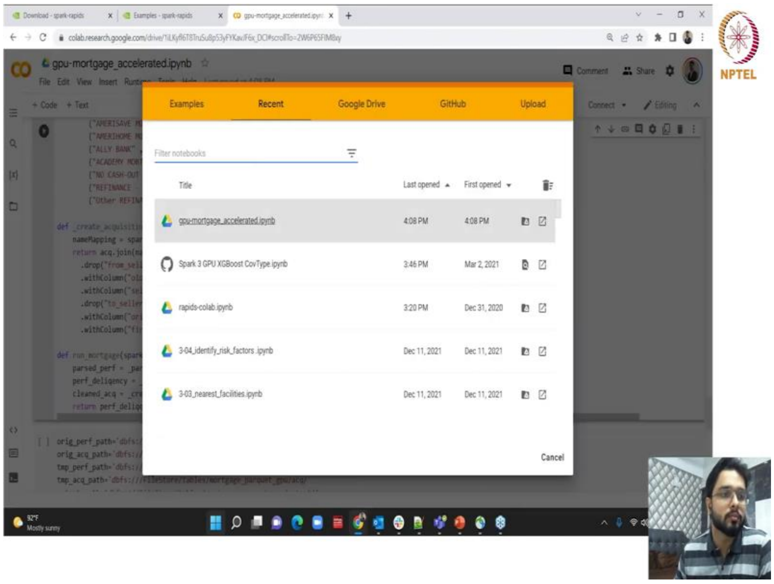
(Refer Slide Time: 11:44)



(Refer Slide Time: 11:49)

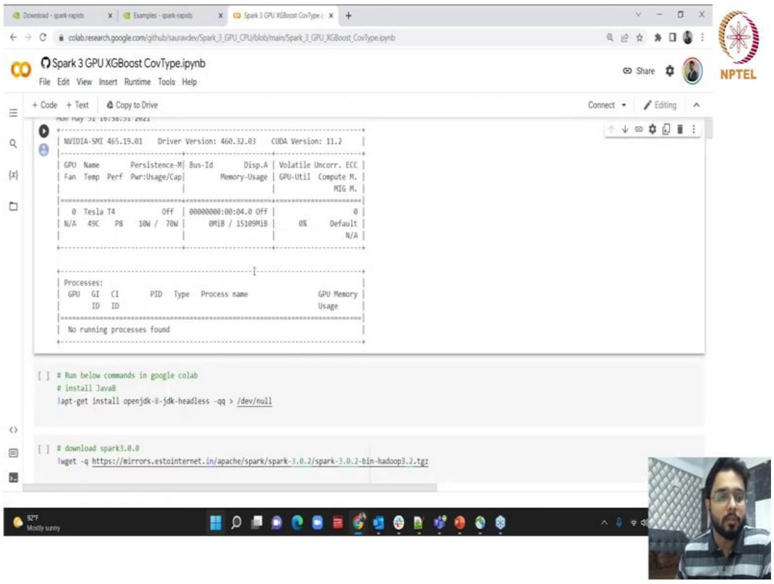


(Refer Slide Time: 11:54)



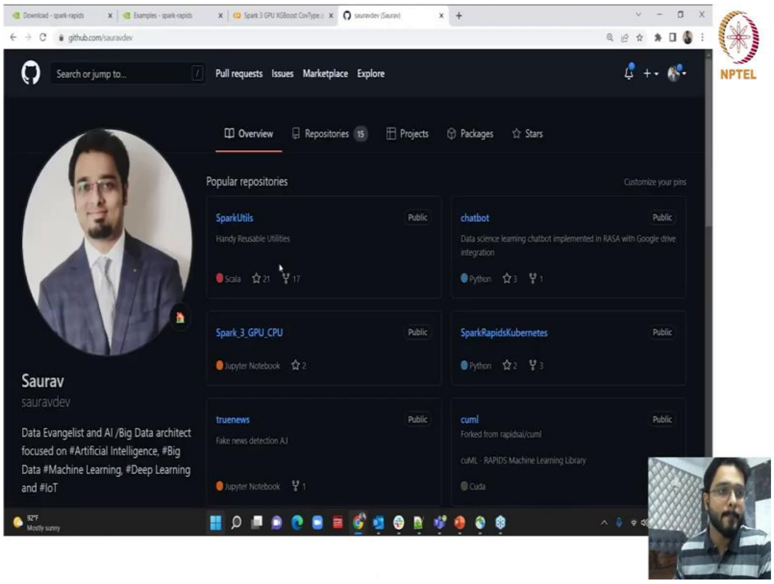
And another notebook is on the and just open that XGBooster.

(Refer Slide Time: 11:56)

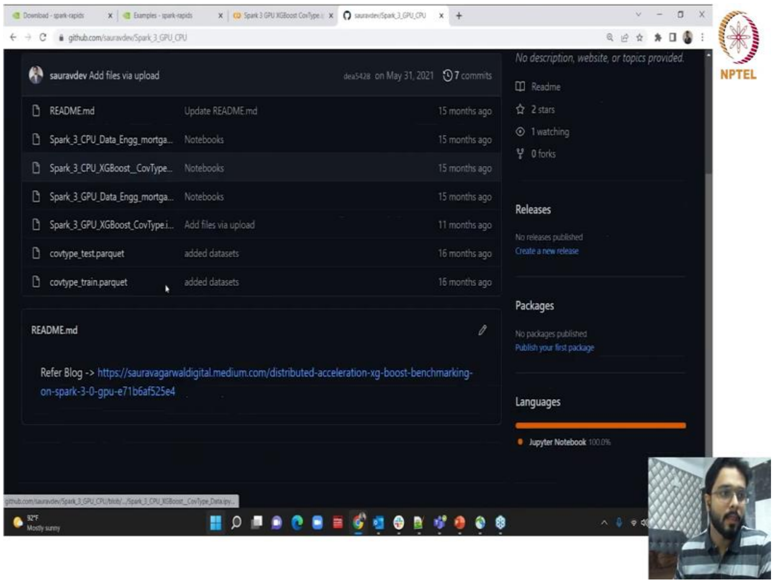


So, this is a simple use case, where I am comparing CPU versus GPU on using colab; Google colab ah. So, you can find these notebooks in my github as well which is again in public website.

(Refer Slide Time: 12:11)

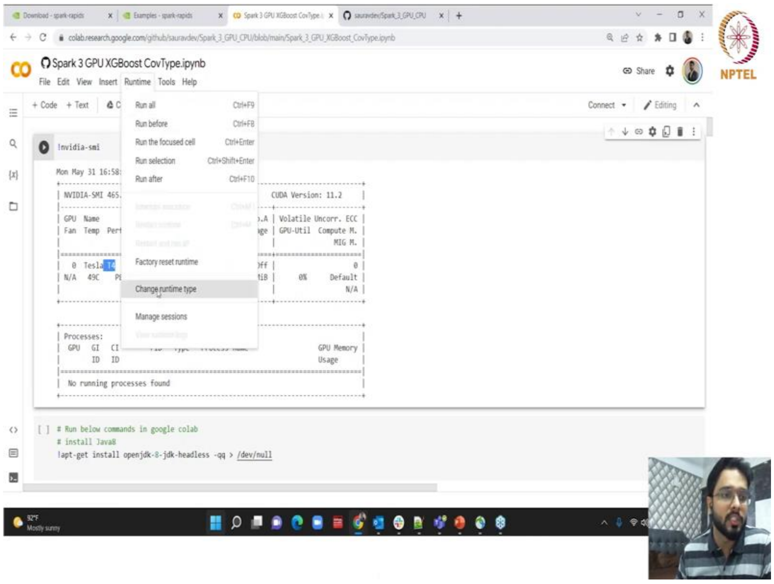


(Refer Slide Time: 12:23)

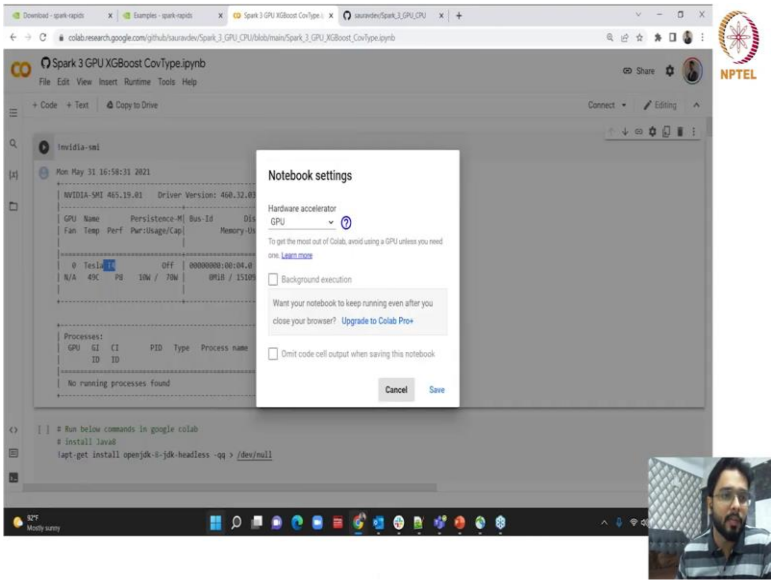


So, here the CPU versions are there and here, the GPU versions are there. So, I just use the same notebook here and import it in colab and then, I made sure that I am allocated at T4 GPU.

(Refer Slide Time: 12:38)

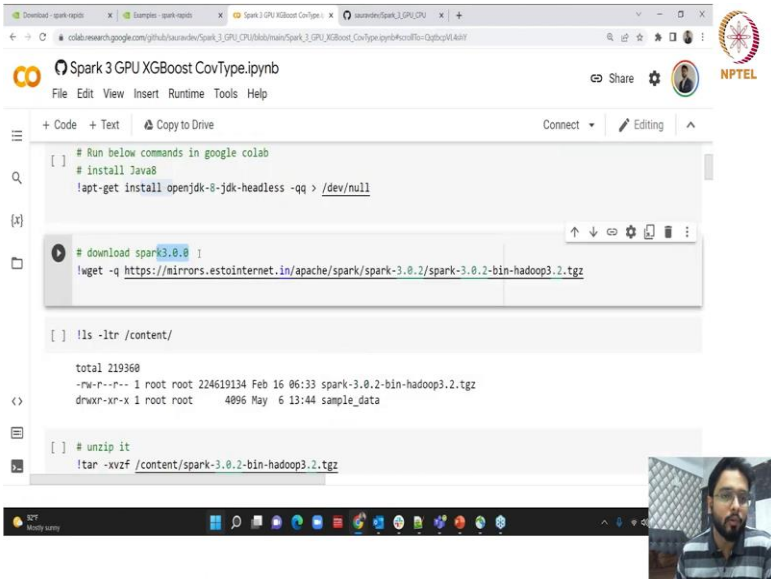


(Refer Slide Time: 12:40)



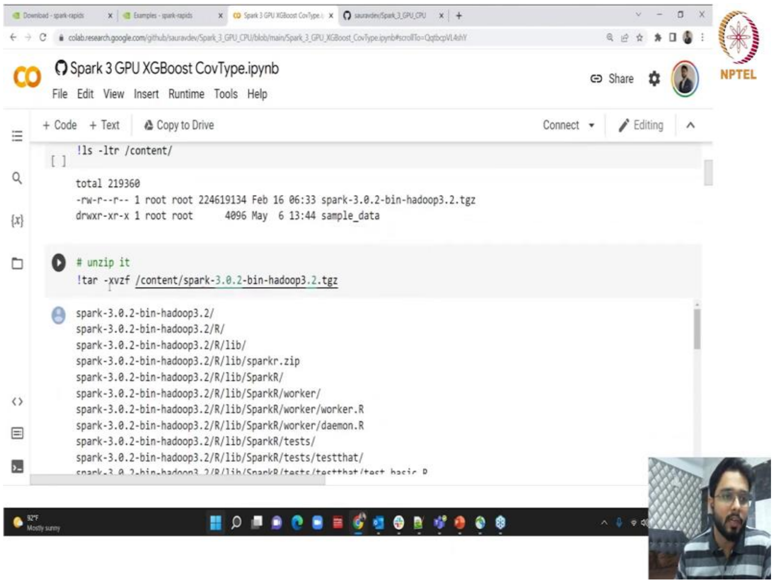
So, if you are not allocated, then you have to go to the runtime, change the runtime type to GPU and then, ok. Again, if you do not see the nvidia-smi command giving tesla T4, then you can do a factory reset and try to run it again and again. So, maybe 5 to 10 times you have to do; but you will definitely get allocated one GPU for sure.

(Refer Slide Time: 12:58)

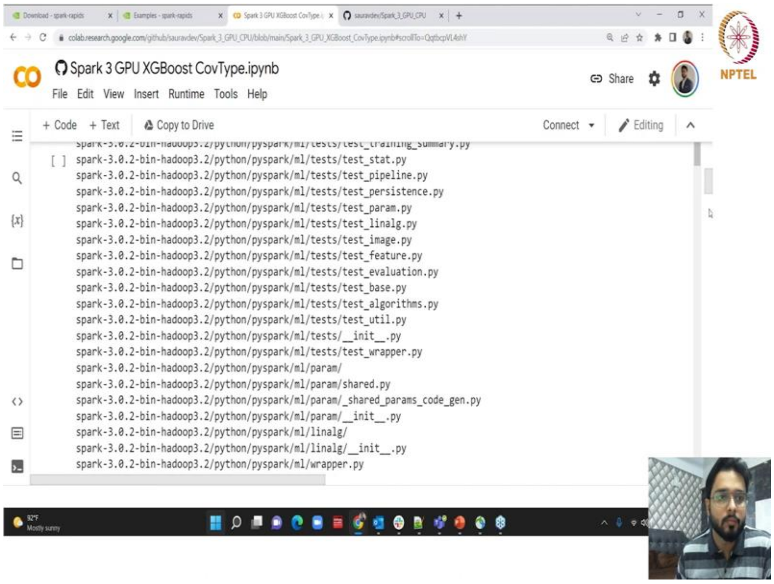


Once you are allocated the GPU, you can follow the notebook. I will just magnify and explain at a high level. So, here if you see first, I am running downloading Java8 because it needs Java, then I am downloading spark 3.

(Refer Slide Time: 13:14)

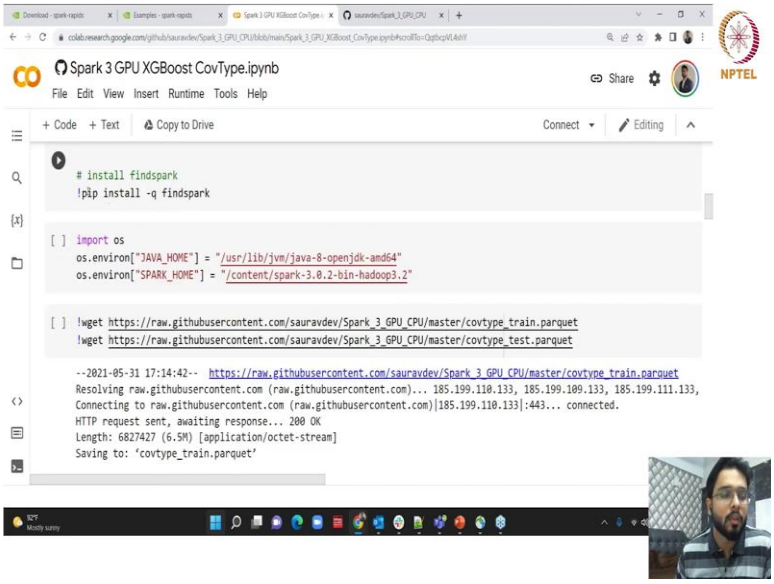


(Refer Slide Time: 13:21)



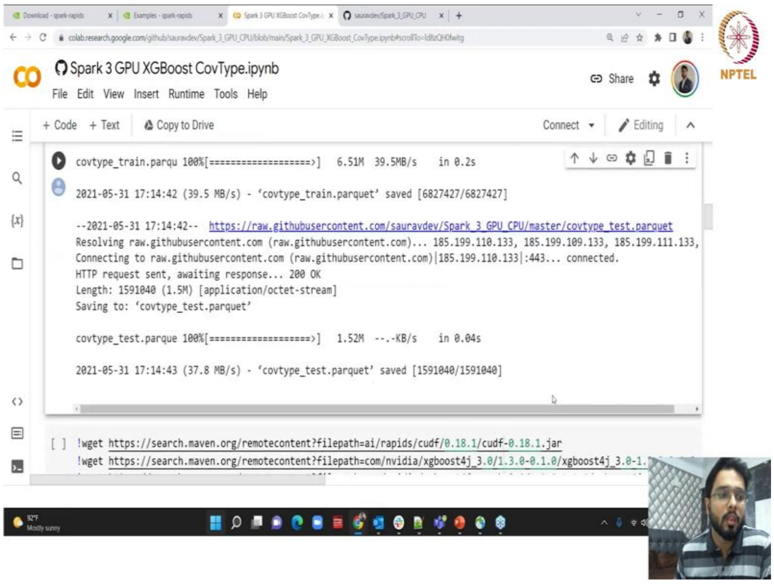
Then, I am downloading unpacking the spark 3; open source for this.

(Refer Slide Time: 13:24)

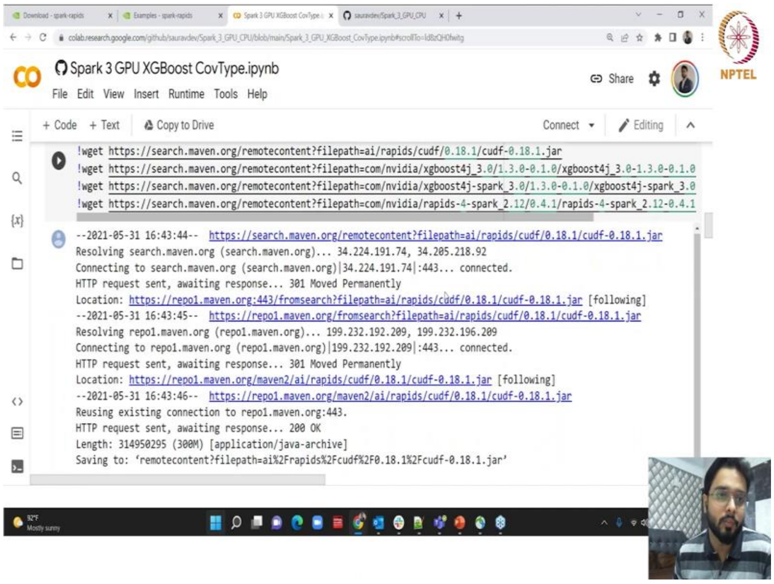


Then, I am installing some library called find spark, where which will be able to use and you know initialize spark; setting the JAVA HOME, SPARK HOME, downloading that data the forest cover data in parquet format, then setting up the libraries.

(Refer Slide Time: 13:45)

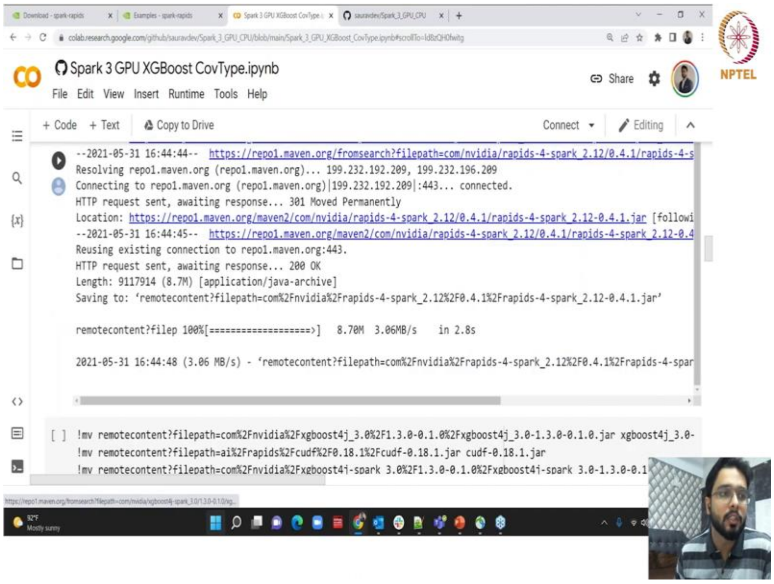


(Refer Slide Time: 13:47)

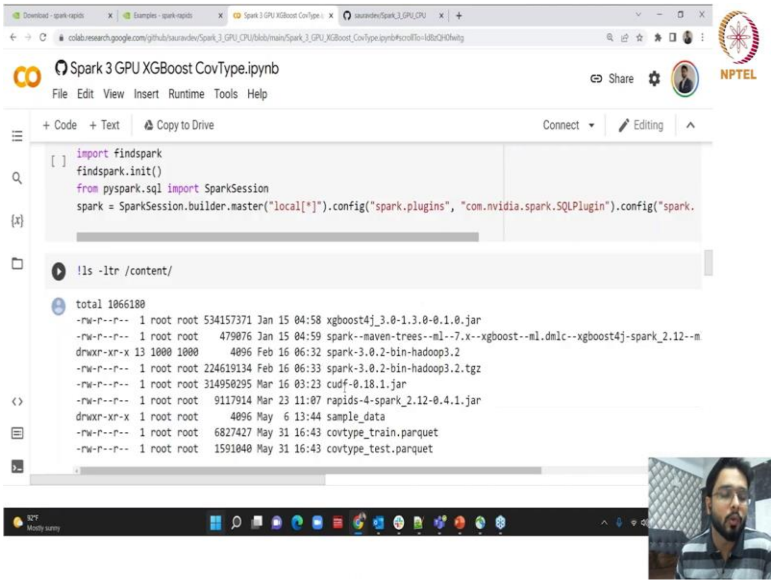


So, we need the cuDF library, we need the RAPIDS Spark library, the jars which I was talking about, we need the XGBoost because we are running XGBoost model for GPU. So, all those jars will be downloaded from the respective maven repositories into the local.

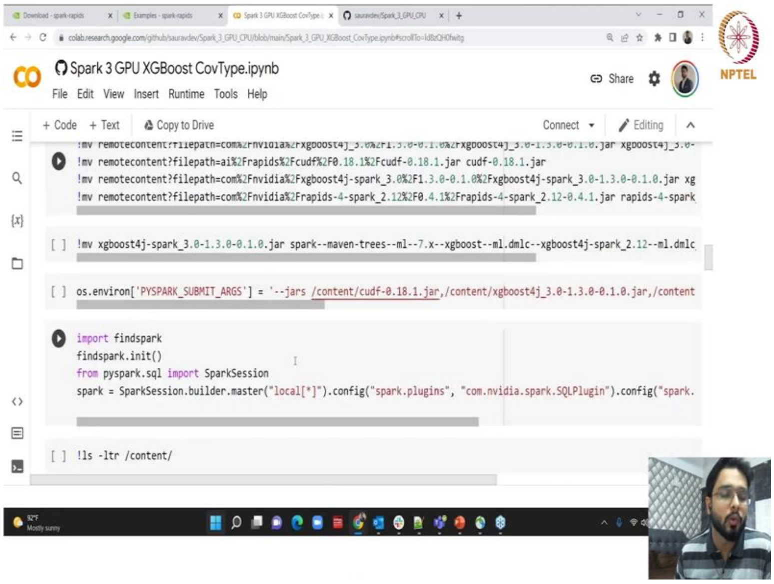
(Refer Slide Time: 14:04)



(Refer Slide Time: 14:07)

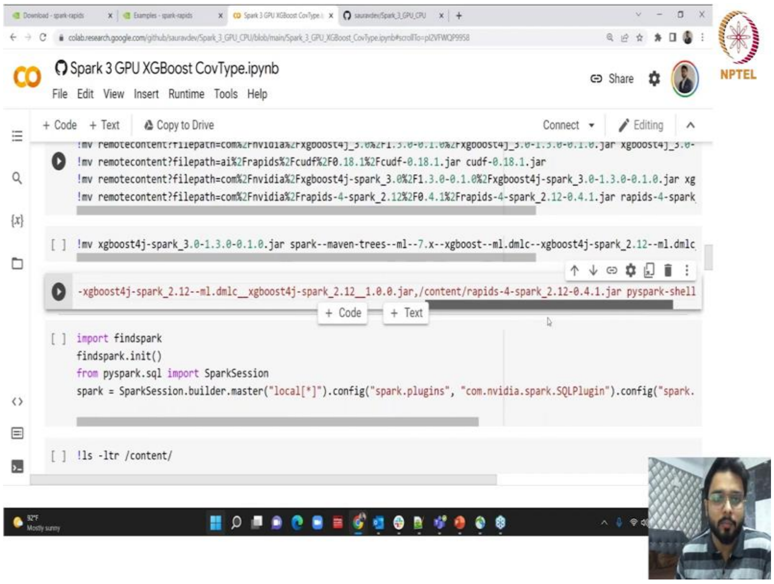


(Refer Slide Time: 14:08)



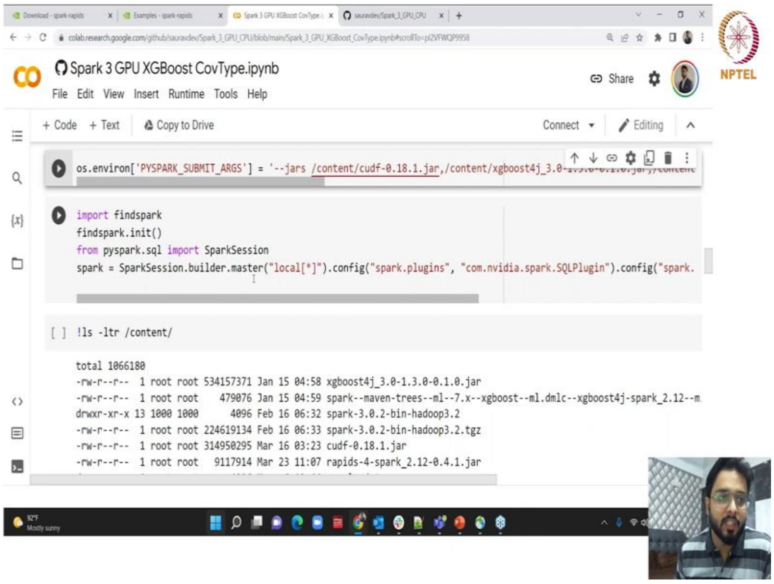
And then, we can move it to the respective location.

(Refer Slide Time: 14:11)



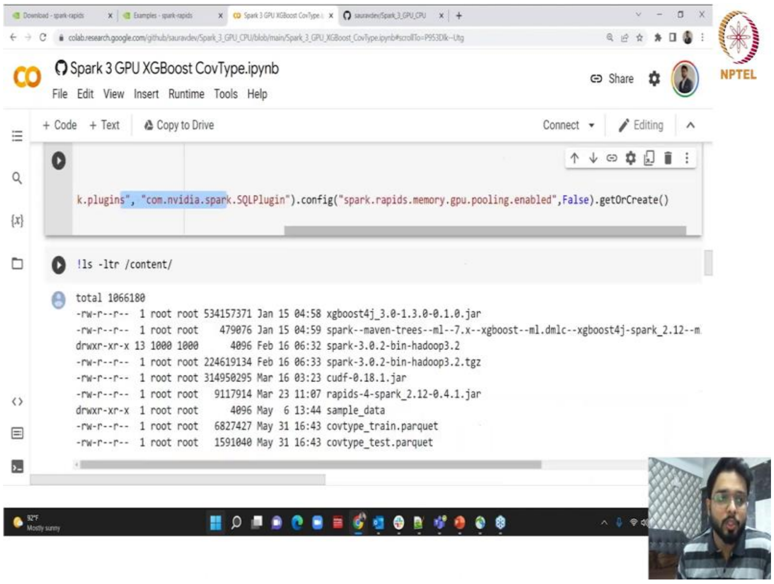
And then, set up the form like set up the jars in the PYSPARK shell. So, that we are able to execute spark code now.

(Refer Slide Time: 14:23)



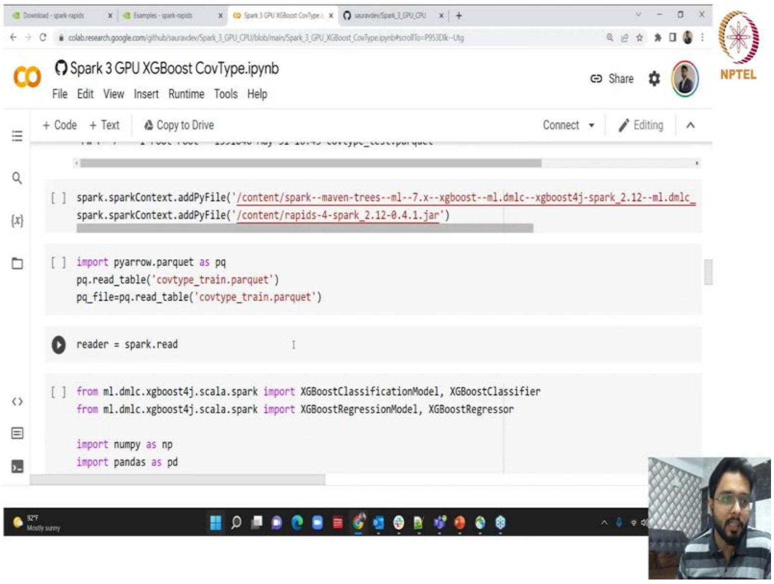
So, we can configure the local; that means, it will use the only one node cluster.

(Refer Slide Time: 14:28)



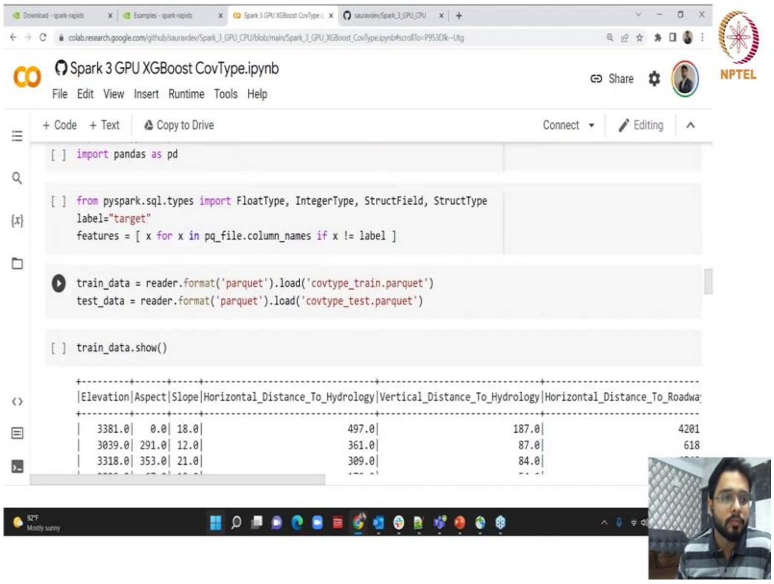
This is the configuration for enabling the spark rapid GPU plugin and we do not want GPU pooling because we only have single GPU of now. So, I am disabled

(Refer Slide Time: 14:40)

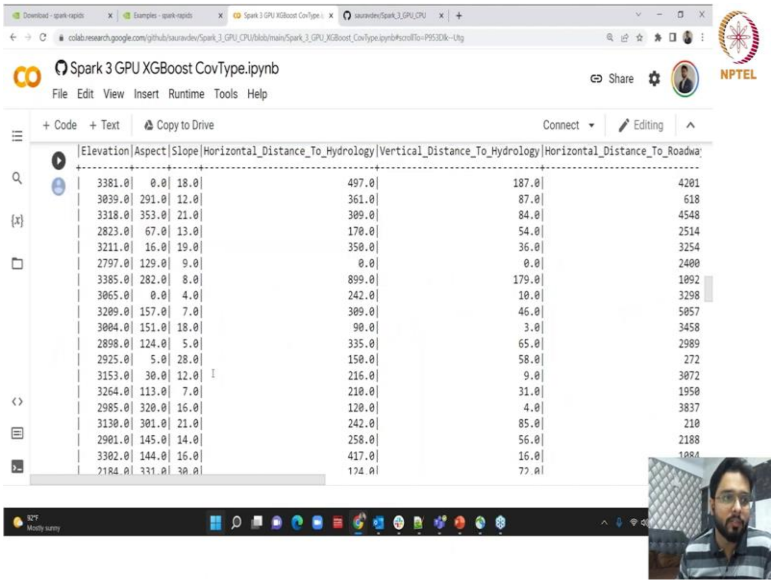


Then, we can add all the machine learning XGBoost jars reading the data; creating data spark data frame; training and testing data.

(Refer Slide Time: 14:52)

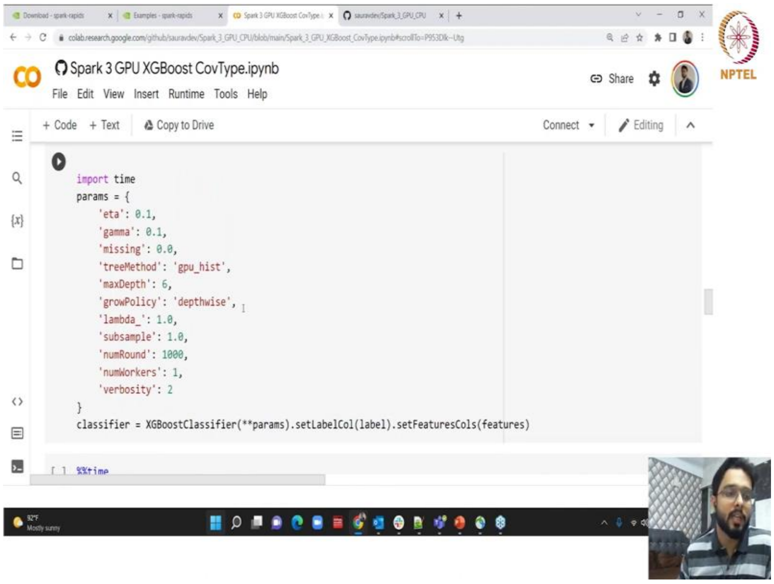


(Refer Slide Time: 14:55)



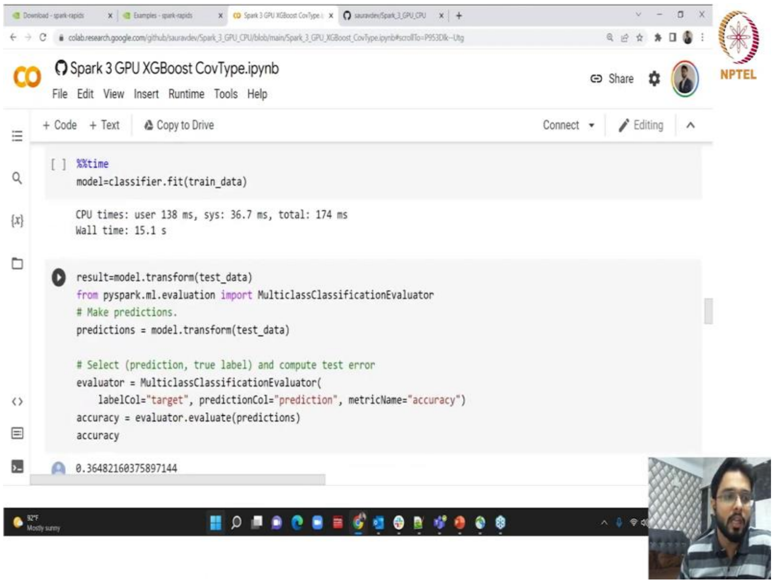
So, this is the data frame forest tree cover. So, where we have elevation, aspects, slope, horizontal distance, to hydrology, vertical distance, horizontal distance, to roadways and so on and so forth.

(Refer Slide Time: 15:10)



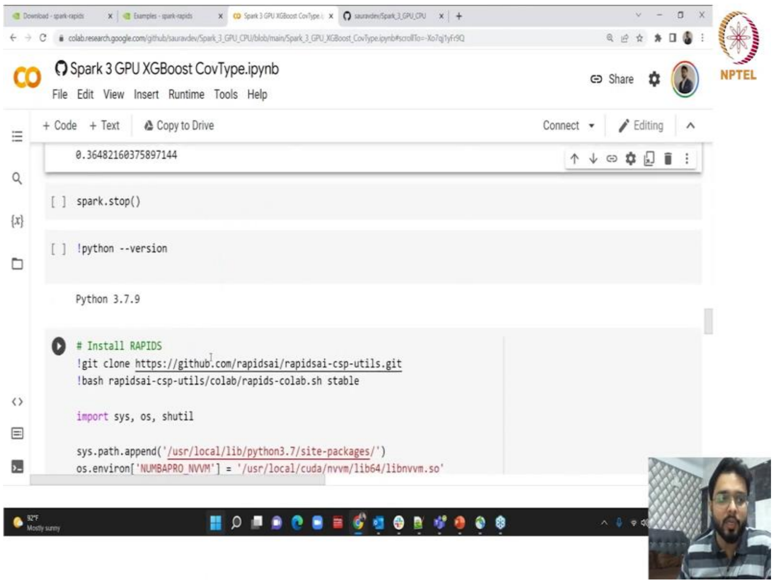
So, we are using this data.

(Refer Slide Time: 15:17)



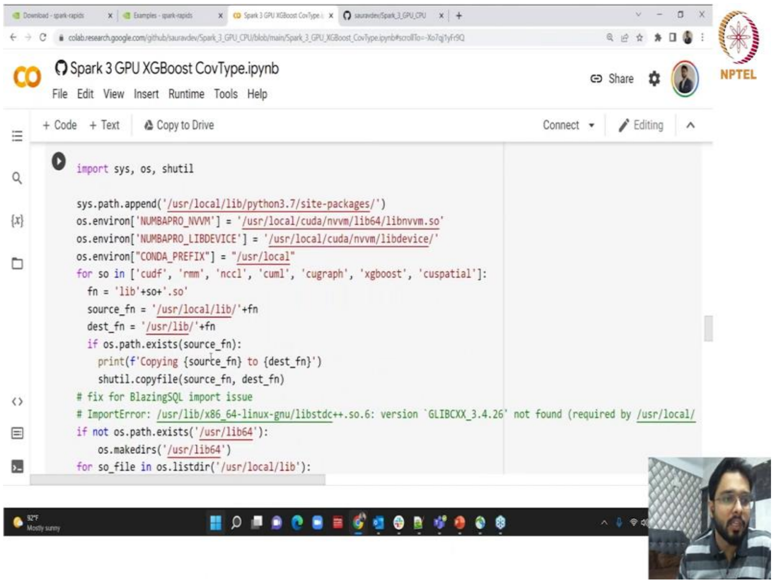
Passing the parameters, classifier, setting the label, feature columns, classifier.fit train the data and then, making the predictions.

(Refer Slide Time: 15:23)

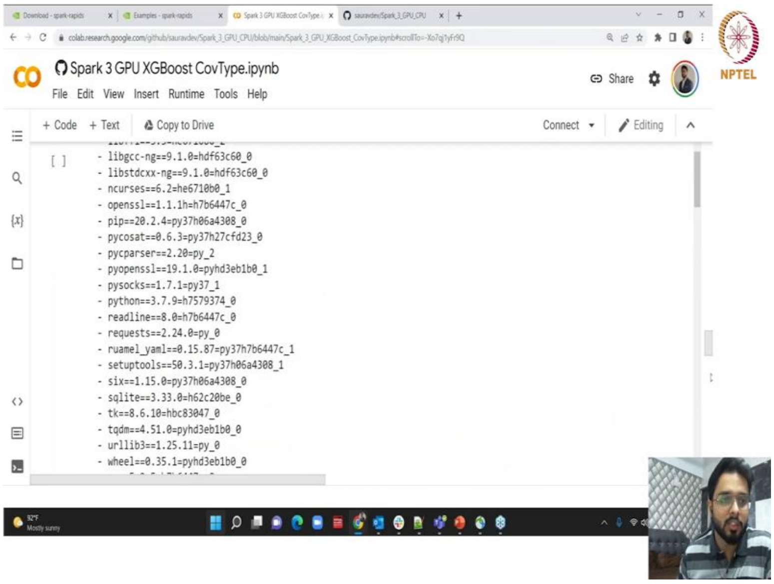


Ah Selecting the and computing the test error; so, here we are using the ml library basically multi classed evaluation for understanding the accuracy, though the accuracy is bad; but definitely, it will help you to understand the flow at least.

(Refer Slide Time: 15:42)

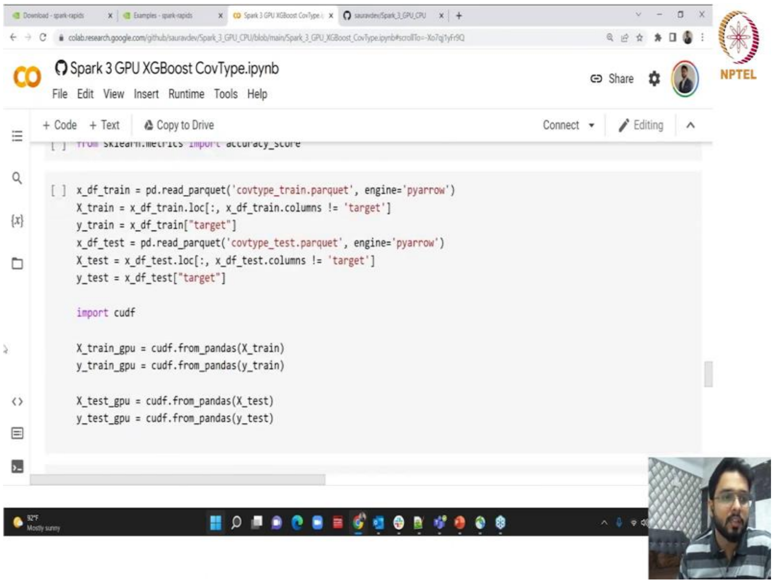


(Refer Slide Time: 15:46)

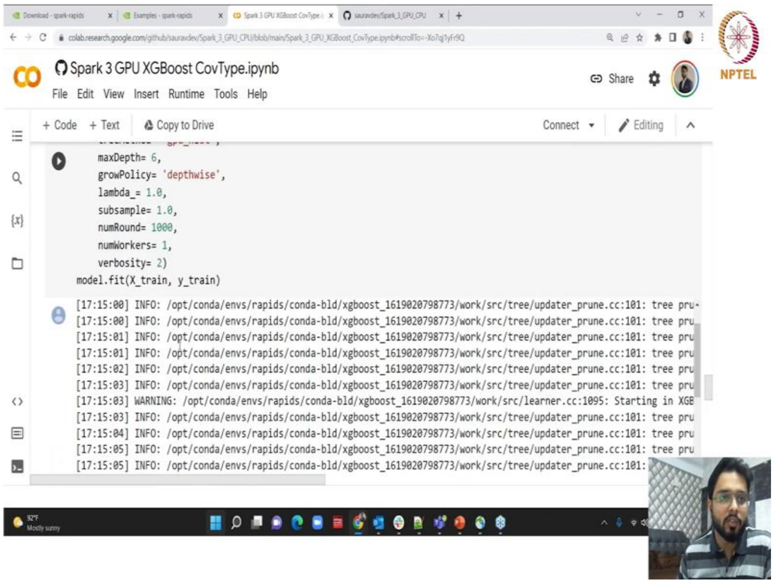


So, the same thing, we are doing for CPU versus GPU and we are finding ok for CPU versus GPU training and testing data, converting from pandas to GPU format, running by XGB format.

(Refer Slide Time: 15:54)

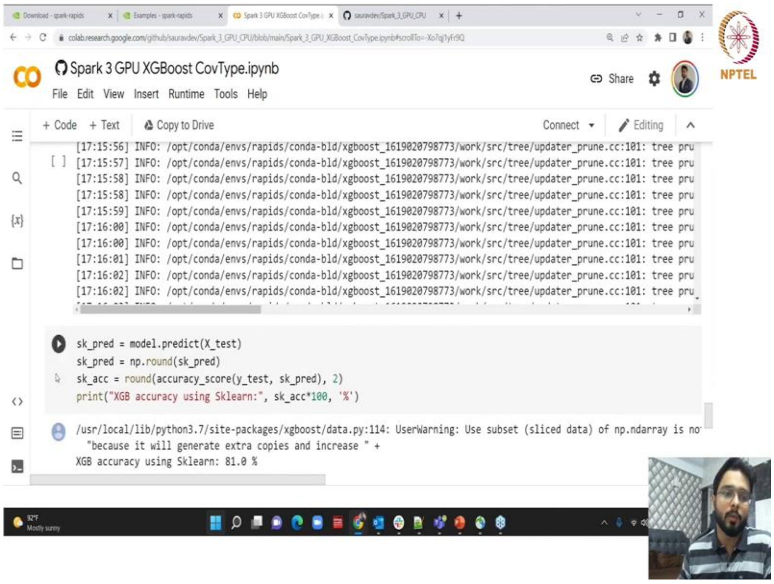


(Refer Slide Time: 16:01)



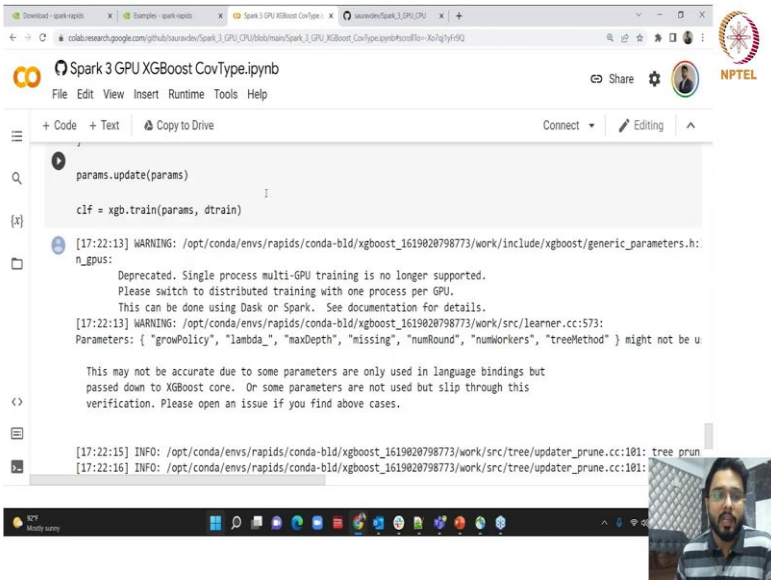
So, we are comparing against a normal cudf as well.

(Refer Slide Time: 16:06)



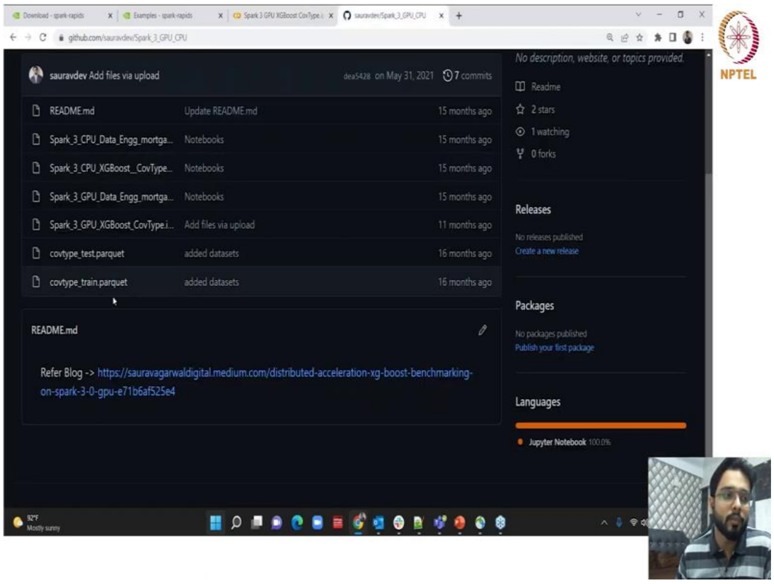
So, cuDF might have heard about that this alternative to pandas on GPU. So, even spark gets faster than cuDF.

(Refer Slide Time: 16:20)



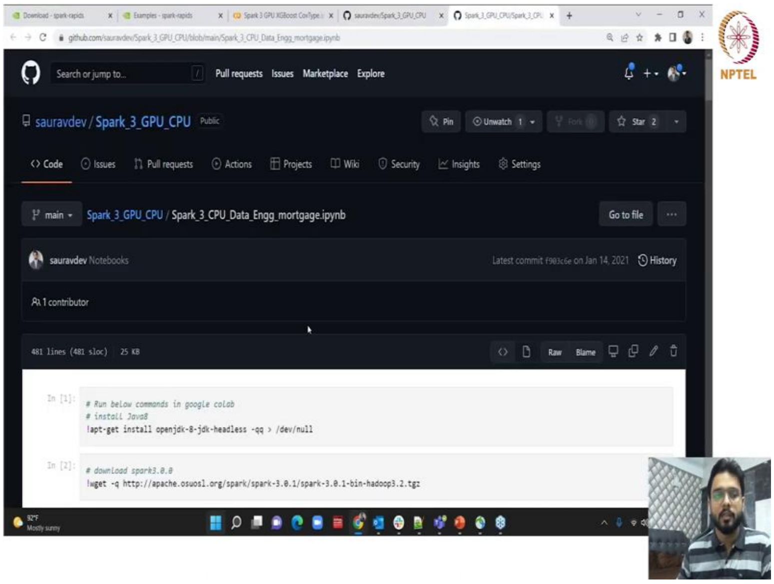
So, this is a simple classification kind of data set. ah

(Refer Slide Time: 16:25)

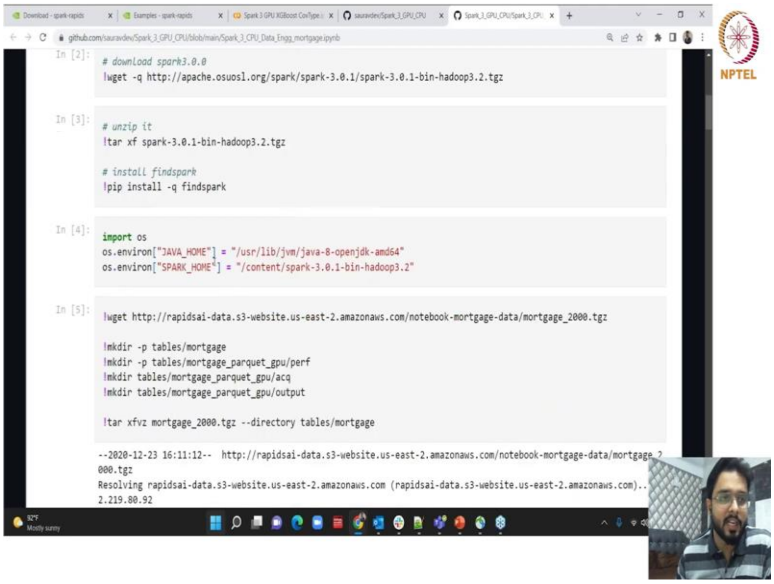


Though you can find more and more examples, where the FannieMae Mortgage Dataset, which I was talking about you can keep that as well here.

(Refer Slide Time: 16:31)

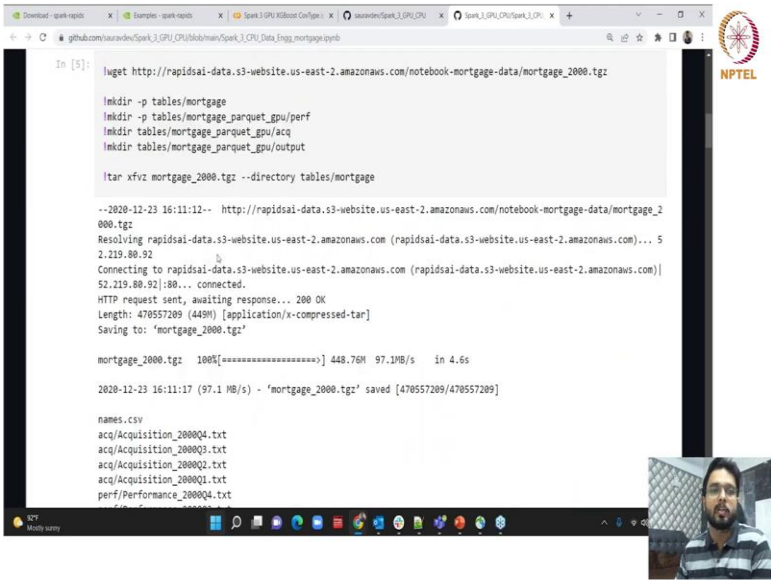


(Refer Slide Time: 16:36)



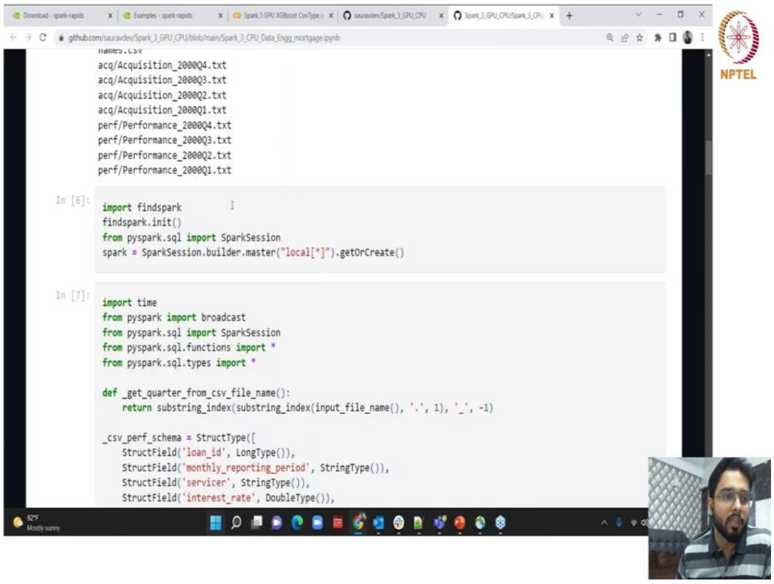
So, if I can go through; so, here if you see the fanniemae mortgage data set, the installation part is exactly the same; just that the data download is different.

(Refer Slide Time: 16:45)



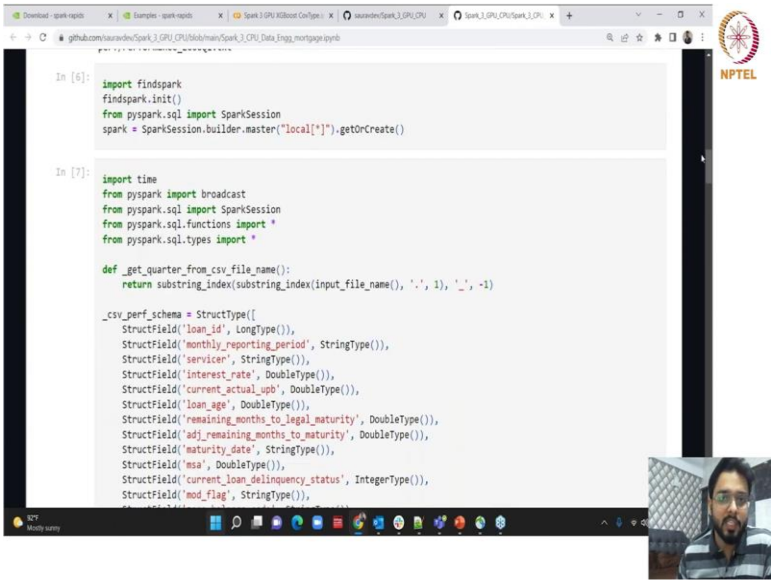
Because you are downloading a new data set of mortgage data of 200 GB csv.

(Refer Slide Time: 16:54)



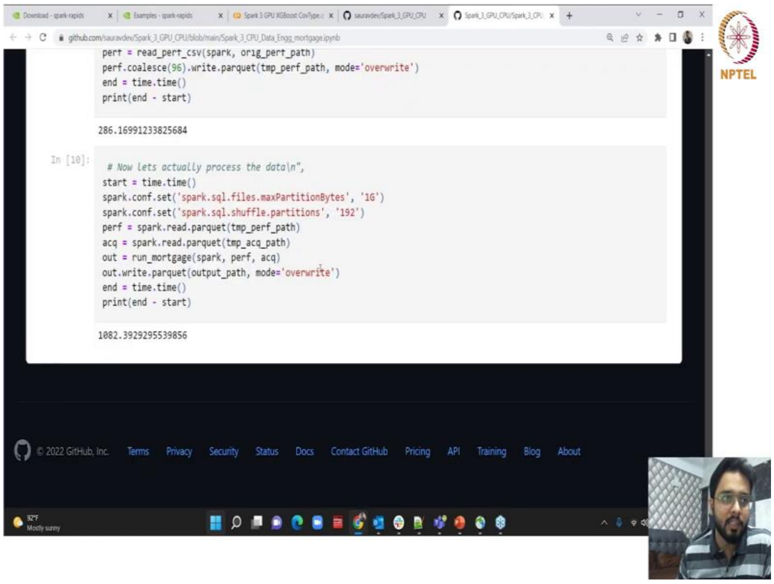
Then, we are reading the data. ah

(Refer Slide Time: 16:56)



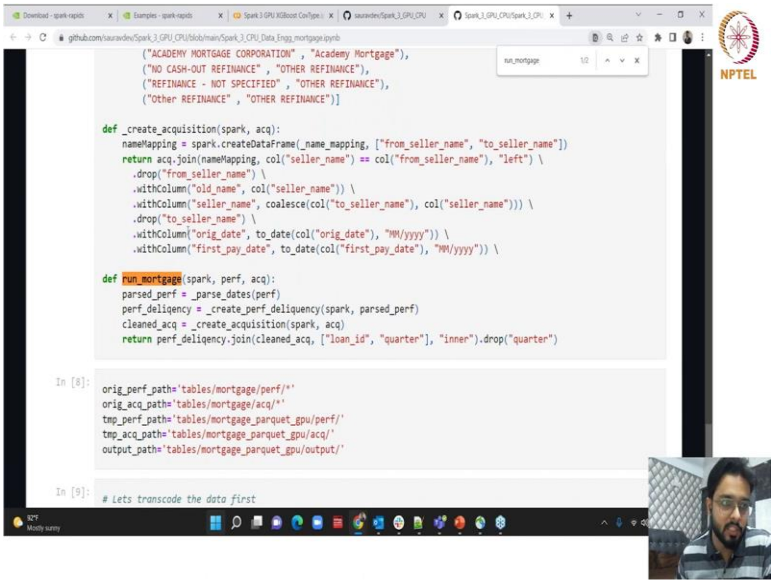
So, the base function, the main function is at the end, where which is calling the read parquet.

(Refer Slide Time: 17:00)



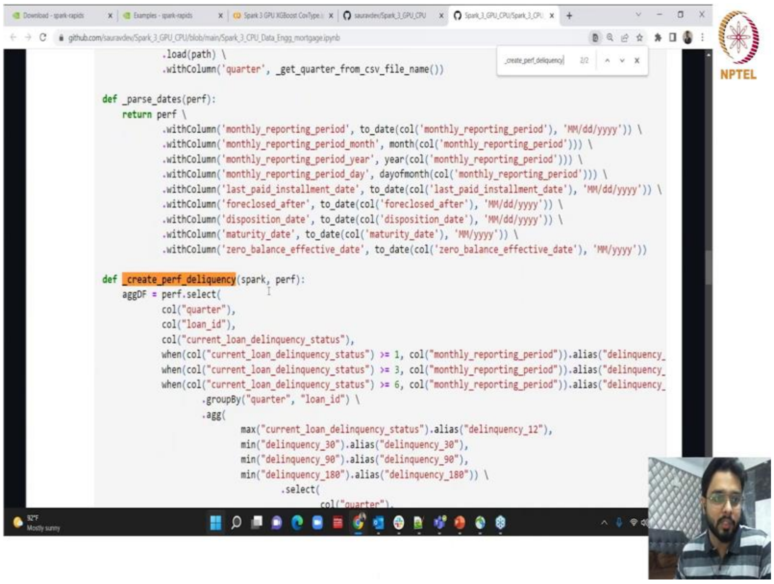
Then, it is converting that parquet into read csv and converting to parquet.

(Refer Slide Time: 17:15)

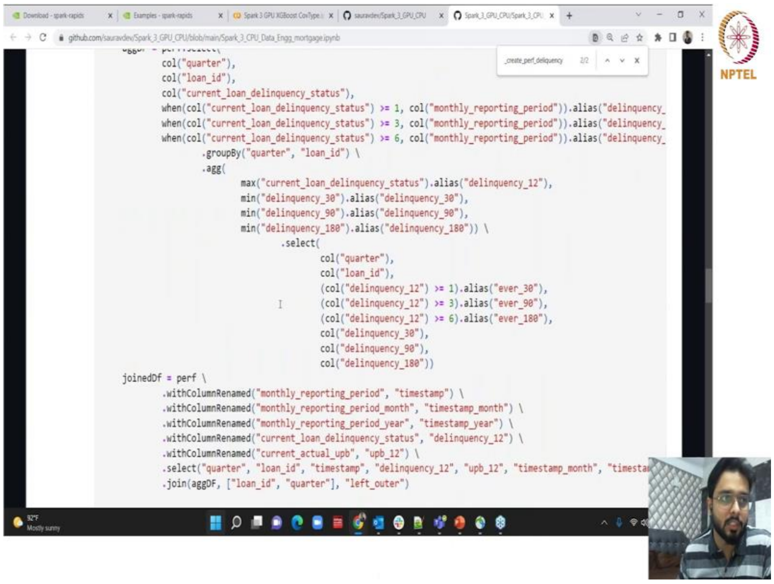


Then, reading the parquet and running this notebook function called run mortgage, which is basically running the two more functions called create of delinquency and create acquisition. So, create perf delinquency is one where we do the maximum transformation, where you can see we have case one statements, we have group by aggregates, where we have max, min, select, joint joining of data frames.

(Refer Slide Time: 17:27)

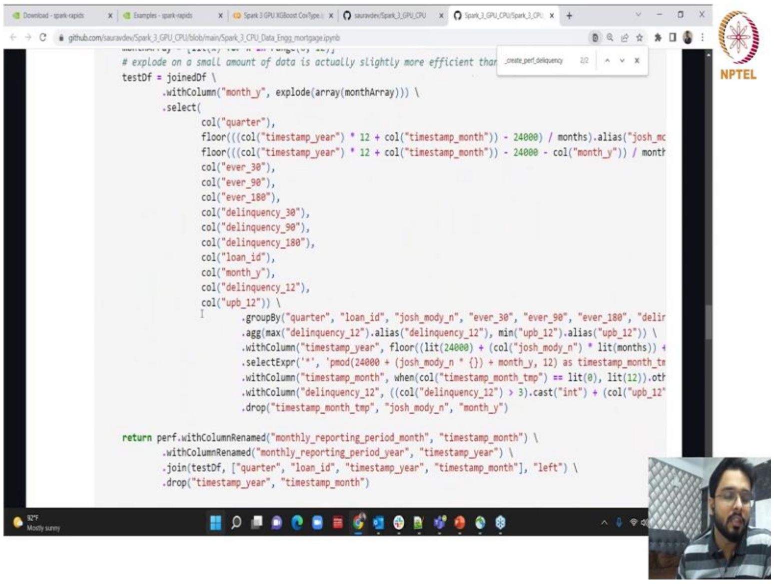


(Refer Slide Time: 17:29)



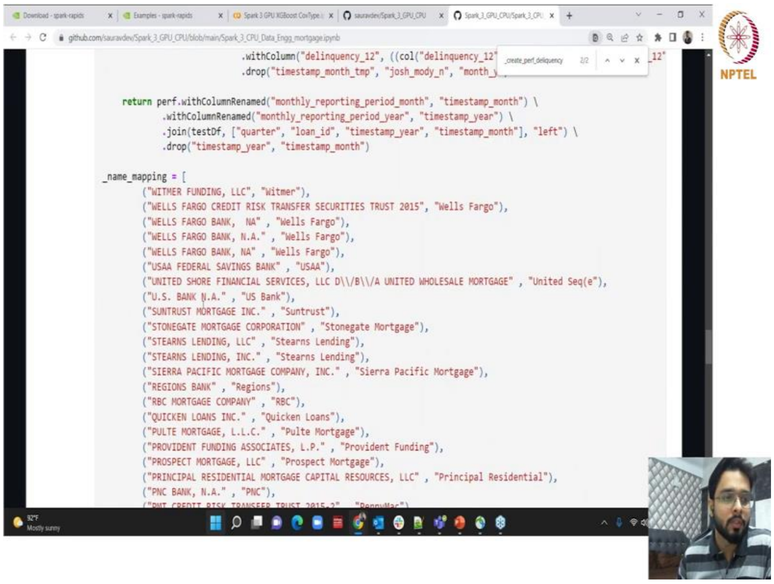
hm.

(Refer Slide Time: 17:41)

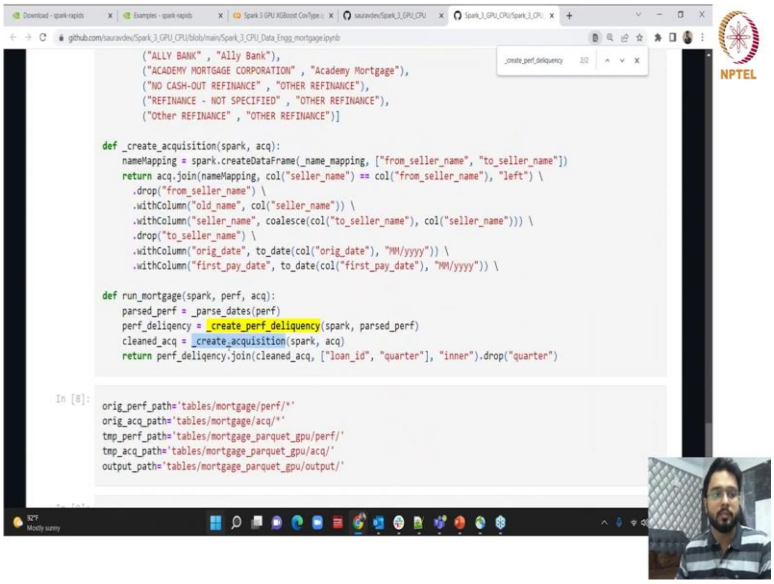


And then, with column exploding selecting and so on and so forth.

(Refer Slide Time: 17:45)

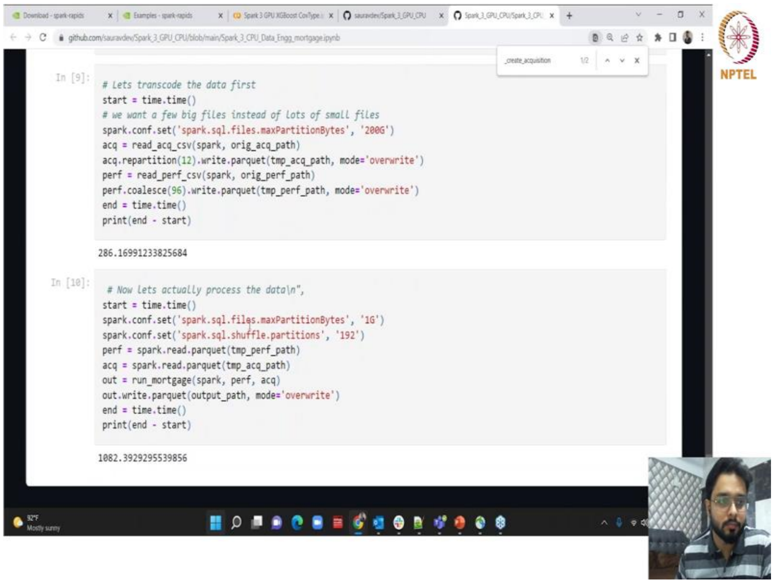


(Refer Slide Time: 17:50)



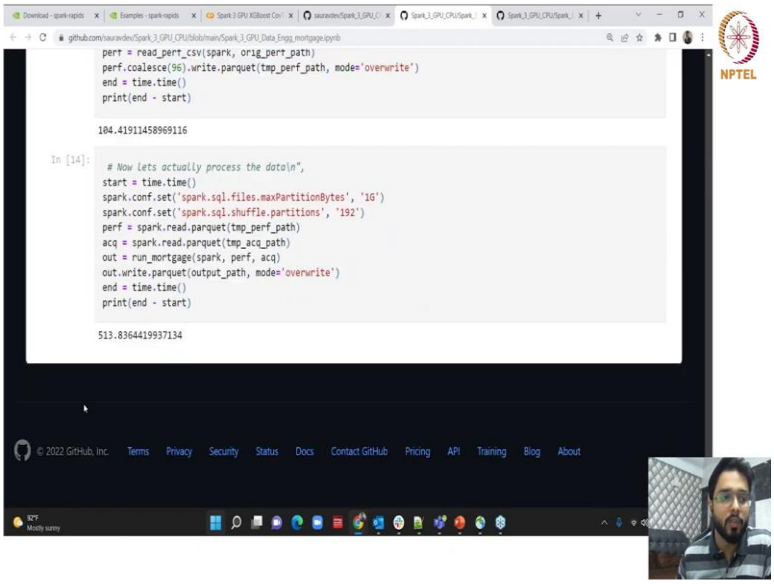
And then, we have creating acquisition a function, where we have creating data frame, joining the data frame, dropping, adding more columns. So, you might have if you even do not know spark, it is very similar to what you can do with pandas or similar tools. So, it is a data frame based processing, a logic which you definitely might know about.

(Refer Slide Time: 18:13)



So, here also you can see that there is a performance improvement for sure. So, here if you see we are running in 1082 seconds. Ah. So, if I compare it with GPU, so this was the CPU part, where we are taking 1082 seconds.

(Refer Slide Time: 18:54)



But the GPU the same thing on the GPU will just run in 513 seconds. So, you will see that ok you are even gaining 2x on the Google colab environment which is free to use as well. Because it does not have that great GPU, it is a very cheap GPU called T4. So, you will just see 2 to 2.5x acceleration. But if you use better GPUs like A30, A100, then definitely you will find more and more acceleration.