

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

Lecture - 50
Web visualizations to GPU accelerated crossfiltering part 1

(Refer Slide Time: 00:14)



Hey, hi everyone this is Saurav Agarwal. I am senior solutions architect at NVIDIA. So, today I am going to present to you about Accelerating Data Science Workloads on GPUs using RAPIDS. Particularly, I will focus today on visualization and NV tabular part. So, how you can create visualizations required for exploratory data analysis or for business intelligence or analytics or simple you know descriptive analysis of data using the RAPIDS visualization ecosystem.

And also, I will also explain you how to do data pre processing using another framework apart from pdf or dask called NV tabular. So, I will focus on these two frameworks of RAPIDS today.

(Refer Slide Time: 01:04)

AGENDA

- Intro to RAPIDS
- cuXFilter
- Plotly
- NVTabular
- Demo

So, so, after introduction to RAPIDS, I will start explaining about cuXfilter which is the visualization library. Another visualization library which is integrated tightly with RAPIDS is plotly. And then I will explain about NV tabular and then we will also do hands on of the same.

(Refer Slide Time: 01:25)

END-TO-END ML WORKFLOW

The Average Data Scientist Spends 80+% of Their Time in ETL as Opposed to Training Models

CPU POWERED WORKFLOW

GPU POWERED WORKFLOW

LEGEND

- dataset collection
- analysis
- ETL
- train
- inference

So, let me start with brief recap of GPU accelerated data science on why it is important and what is the benefit of doing it as compared to the CPU power data science ok. So, we all know that majority of the time effort of a machine learning or a data science project is

spent on the ETL part of it as opposed to the model training or any other part right. So, most of this ETL part 80 percentage of the project time is spent because of few things, that is if you see on the left hand side the CPU power workflow.

So, here we have the data sets being downloaded overnight that is fine that is equal for CPU and GPU downloading you cannot accelerate the downloading part. But configuring the ETL workflow ok there is some lead time needed of coding and configuring the ETL workflow.

But apart from that if you see running the ETL workflow this green part this is a huge part hence majority of the time of a data science project is spent on this green part whether it is running the ETL workflow for the first time or if you think that you forgot to add a feature and you have to restart the ETL workflow or you found the unexpected null values and again you have to restart the ideal workflow.

So, it is like it is more of a stop gap approach where you are fixing thing and then restarting fixing. So, fixing and restarting is not a problem that is expected as a part of the project because data behaves in different ways for different projects. Hence you have to unless until you run the ETL workflow you will not get to know that what is the exact error; however, waiting for the ETL workflow to complete is the one major aspect where you get a bottleneck in your productivity.

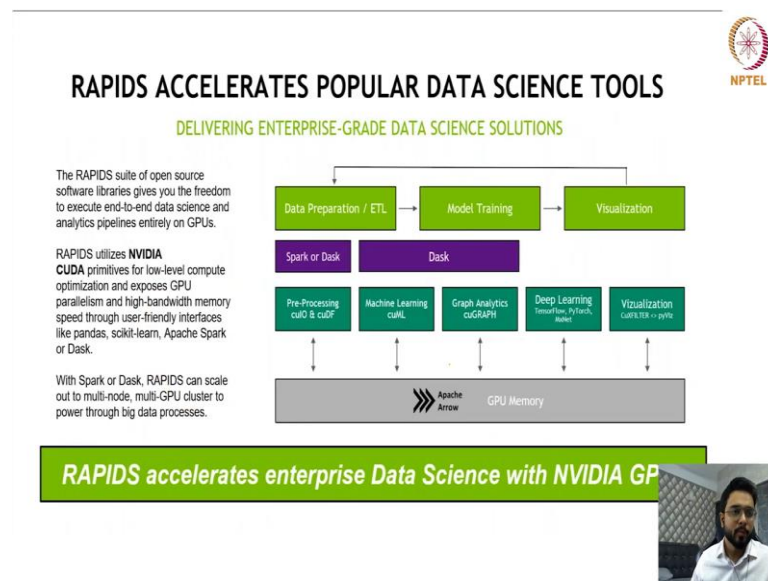
So, you do nothing, because your entire system or server is occupied running the workflow, you cannot run anything in parallel if you run parallel then the workflow will again slow down. Hence, you just have to wait for it and you cannot do any productive work at that particular point of time; hence your entire time is wasted ok.

So, here comes the GPU power workflow where we not only accelerate the model training part. So, that is something which we definitely accelerate, but we also accelerate at in media using rapid suite of libraries the running of the ETL workflow. So, whether no matter how many times you restart it, the running of the workflow will be. So, fast that you would not feel that ok my majority of the time is going in the ETL that is extract transform and loading of the data.

Hence, your effort and concentration will be more on optimizing the model and repeating the optimizations rather than waiting for the ETL workflow to complete again and again.

So, that is the biggest thing which you can do using GPU power workflow as a data scientist that because the acceleration happens end to end. So, your entire day as a data scientist is optimized for running the project in a more optimized manner as well as accurate manner that because you are getting more and more time for model optimization rather than just waiting for the data process to complete.

(Refer Slide Time: 05:06)



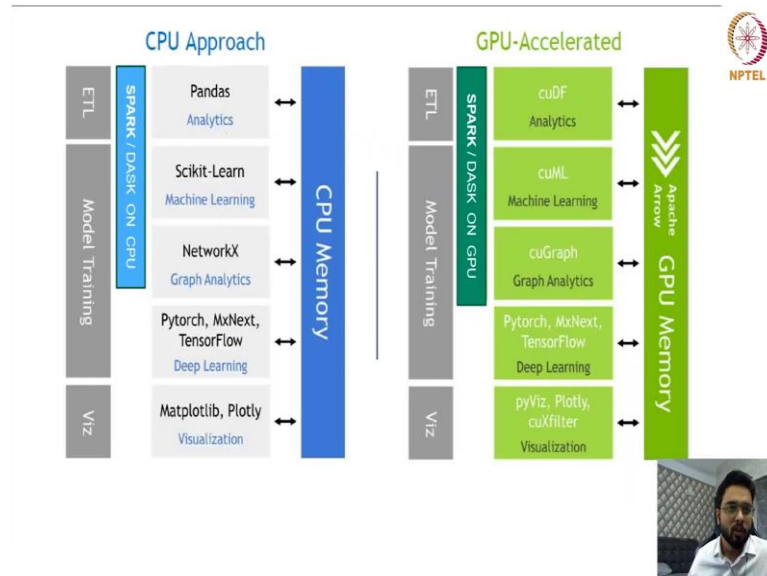
I was just giving a recap. So, how does it happen using the RAPID suite of libraries. So, RAPIDS is open source software library suit which gives us the freedom to execute end to end data science and analytics pipeline entirely on GPUs which utilizes NVIDIA CUDA primitives at the low level.

So, at the back end of RAPIDS, we have the CUDA primitives and the Apache arrow open source project which helps to do transfer of the data seamlessly from one application to other application entirely on GPUs rather than involving any host or CPU in between. So, if you see on the diagram we have data preparation, model training, and visualization; these are the main three things.

So, today definitely we will be focusing on visualization ah; however, just to give you a brief recap that ok for data preparation we have cuIO, cuDF, machine learning for model training we have cuML, for graph analytics we have cuGraph, for deep learning model training we have the integration to tensorflow pytorch MxNet, etc. And for visualization

we have cuXfilter ok all this can be also made multi node and scalable using either spark or Dask framework.

(Refer Slide Time: 06:33)



So, if you remember that the counterpart to pandas is cuDF. So, whatever suppose you write `import pandas`, `import pandas SPD` you can change that `import cuDF` as `SPD` and rest of the import almost remains the same if not exactly the same, it is almost 95 percent the same. Then we have this Scikit-Learn counterpart which is cuML then networkX counterpart which is cuGraph. So, this is the CPU approach versus this is the GPU approach.

So, though we have not integrated that ok you just add a GPU parameter. So, sometimes for frameworks like spark or frameworks like dask, we just have one parameter change that ok you add that GPU parameter it is done, but sometimes for example, for pandas or Scikit learn we have the import statement change. So, either you do the import statement change or you do the parameter change. So, by these two options you will be able to run the ETL workflow or the ml workflow or the visualization entirely on GPUs ok.

So, today we will be focusing on the visualization part. So, how you can migrate the CPU based visualization of matplotlib or plotly on GPUs using cuXfilter and plotly ecosystem.

(Refer Slide Time: 08:01)

The slide features the title "CUXFILTER" in bold black text, with "(coo-cross-filter)" in green text below it. In the top right corner is the NPTEL logo. A list of bullet points is on the left, with yellow handwritten marks underlining "cuxfilter", "Efficient GPU-backed in-browser visualization engine", "Bokeh and Datashader charts", and "Multiple charts with cross-filtering widgets". The text of the first bullet point is: "cuxfilter (ku-cross-filter) is a RAPIDS framework to connect web visualizations to GPU accelerated crossfiltering. Inspired by the javascript version of the original, it enables interactive and super fast multi-dimensional filtering of 100 million+ row tabular datasets via cuDF." In the bottom right corner of the slide is a small portrait of a man with glasses and a white shirt.

So, now introducing cuXfilter. So, cu of cuXfilter is definitely coming from the CUDA primitives perspective. So, it involves everything involves CUDA here. So, in the back end though it has a very simple easy to use Python interface for data scientists because they are not expected to learn CUDA programming or C++ programming nonetheless we just wanted to clarify that yes it is based on CUDA at the back end and that is the reason why it is accelerated and this can only work on NVIDIA GPUs.

So, here cuXfilter is a one of the libraries in the entire suite of libraries of rapids which can help us to connect web based visualizations to GPU. So, you can create small charts you can do cross filtering on top of it you can do dash boarding on top of it and anything and everything. So, inspired by the javascript version of the original that is nothing but datashader and Bokeh libraries it enables interactive and super fast multi dimensional filtering of hundred plus row tabular data sets through cuDF.

So, at the back end of cuXfilter is nothing, but cuDF. So, if you have any questions do let me know in the qna box in case you have any questions please do feel free. So, the middle of the slides if I get time I will definitely try and answer all these questions alright. So, yes coming back to cuXfilter. So, here efficient GPU based in browser visualization engine it is built on pyViz ecosystem tools pyViz ecosystem means that again.

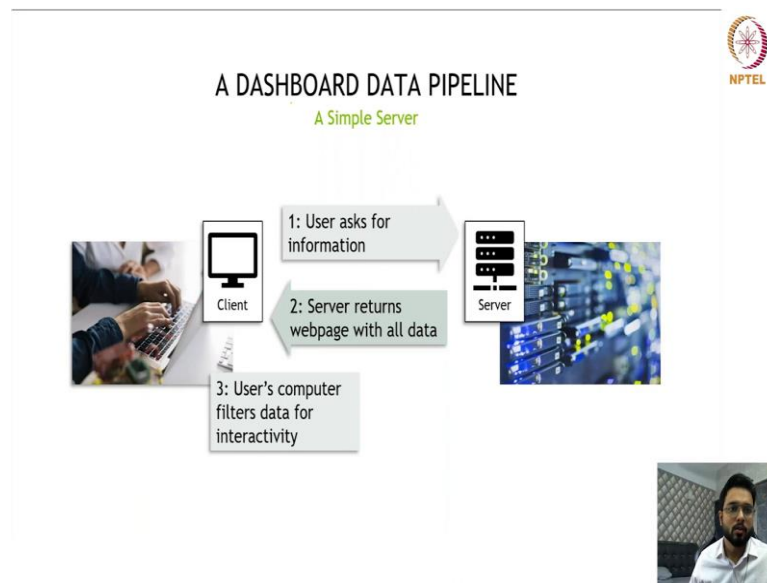
So, the back end of cuXfilter or the interface is based on Bokeh and datashader and very popular to use charts though it does not use any interface of mat plotlib, but other than that Bokeh datashader or plotly everything is being used. So, multiple charts can be created. So, you can create multiple chart; that means, you can create a web based dashboard as well.

So, you can also add cross filtering widgets that ok if you change the year the entire four to five charts inside the dashboard the year gets changed the data for that particular year which you have chosen will be reflected in the browser.

(Refer Slide Time: 10:35)



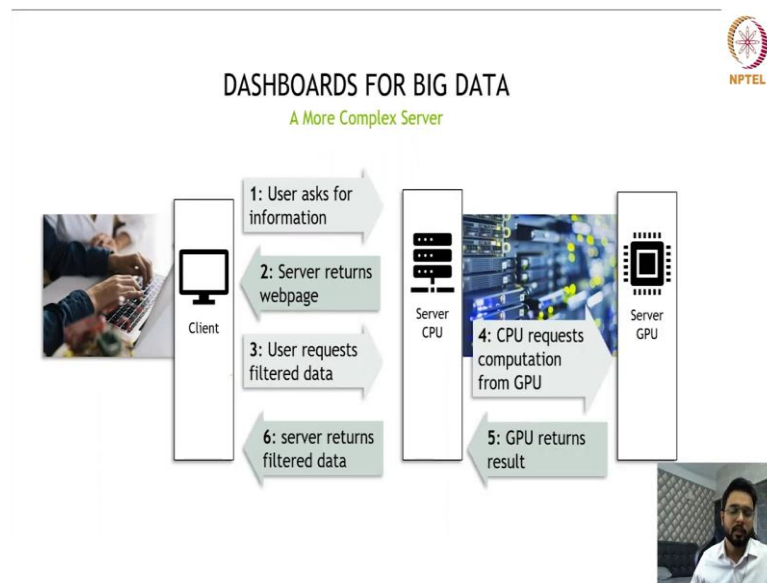
(Refer Slide Time: 10:40)



Now, let us start with how to create interactive dashboards. So, if you see a simple from the perspective of data transfer or the performance of interactive dashboards if you do a simple CPU based dash boarding you will see that user how it happens suppose you are the client; that means, you are requesting to use the visualization.

So, you ask for some information and in the server it returns the web page with all the data and then user. So, all the data is being returned into your particular browser and then it is stored in your local and then you do all the filters and interactive. Suppose, you have to change the year you have to change the region you have to multi select you have to drill down you have to slice and dice the data. So, all these happens from your end at the level of your particular system.

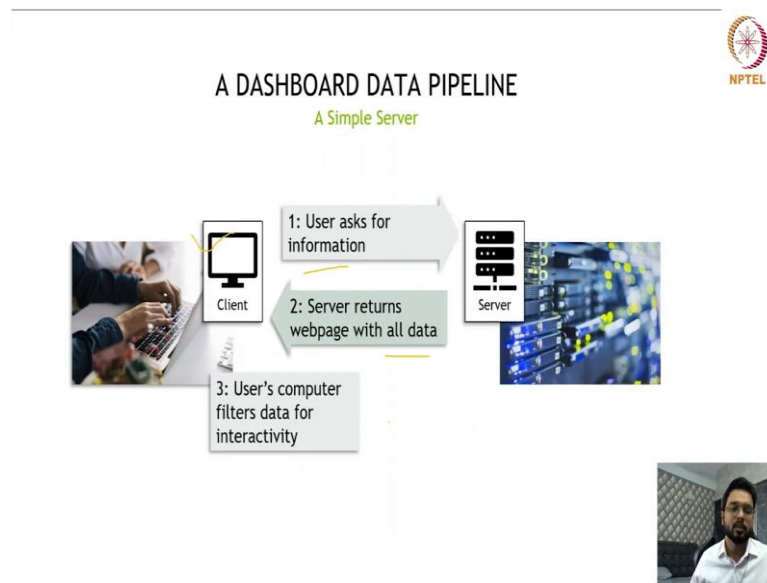
(Refer Slide Time: 11:33)



So, dashboards for big data how it works. So, big data is bit different it is more optimized hence we have included the GPU part here as well. So, you ask for information then we return the web page ok. So, that is not the end of the process. So, if you request for filter data. So, it does not filter on your CPU. So, it does not use your CPU cycles or the server CPU cycles it requires the GPU cycles to filter the data and why it will be faster because when you filter the data it is more of aggregated operations or you filtered operations.

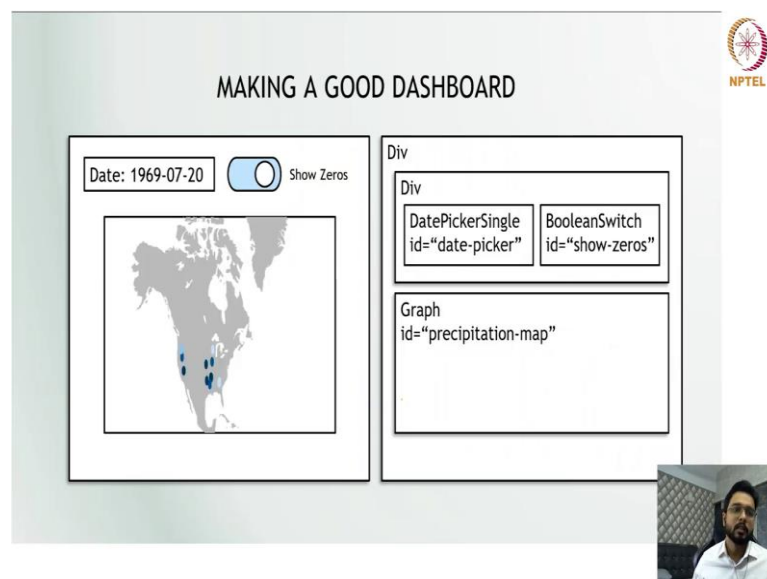
So, a bulk of data is being is being filtered or is being projected. So, because of this bulk being filter projected because of the highly parallelizable CUDA cores of GPU you can filter cross filter and do huge amount of data processing in the backend very fast on the GPU hence after GPU returns the results you see the filter data in a very fast manner.

(Refer Slide Time: 12:51)



Because if you see a simple server. So, here suppose 1 GB data, 2 GB data is fine. But when it comes to terabytes of data then it will falter for sure. So, it is not recommended that you use a simple approach for large datasets. So, you have to use the GPU approach or some other optimized approach ok. So, yes this is the way GPU rendering of the dashboards work.

(Refer Slide Time: 13:22)



So, here if you see how can you make a good dashboard. So, you have to add a chart, you have to show various colors based on your APIs and you have to add some filtering

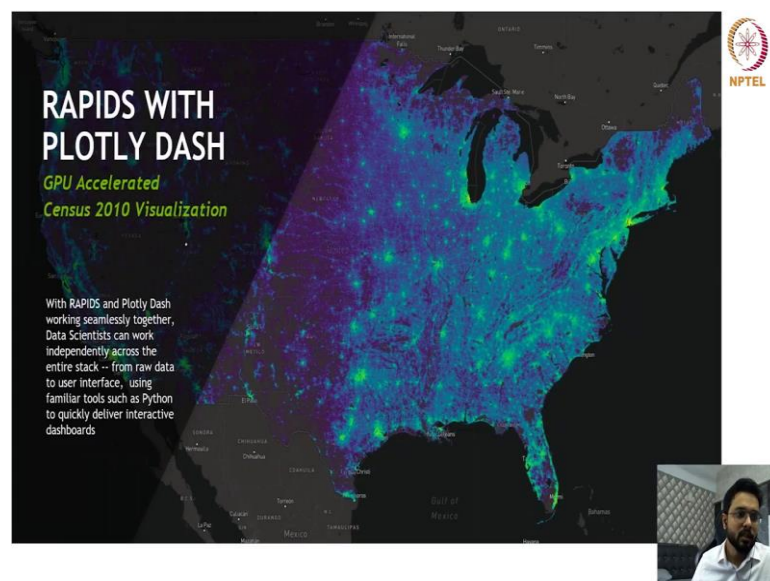
and then there is show zeros; that means, if you want to show null characters or zero value data based on your business requirement you can have one filter that ok you want or you do not want ok.

(Refer Slide Time: 13:51)



So, for that particular thing we have multiple types of charts one was which I already showed cuXfilter you can create that, but apart from that you can also use another framework called plotly dash.

(Refer Slide Time: 14:11)



So, what is plotly dash? So, plotly dash and RAPIDS work together though plotly dash cannot directly work on GPUs. So, plotly dash is something as a framework which you can create very good looking easy to use dashboards on the browsers using a simple python based code. But at the end of the day it is also accelerated using GPU.

So, whatever filtering happens will be GPU accelerated though at the end of the day the plotly dash directly does not consume as you saw in the diagram this particular diagram. So, this is how plotly works that ok it first goes to CPU and request computation from the GPU and then comes back to the CPU. So, that is why plotly does not itself work. But in the back end it uses cuDF and dask kind of frameworks. So, that is accelerated by GPUs.

(Refer Slide Time: 15:12)

PLOTLY DASH
The front end for ML and data science models

- ▶ **plotly.py** is an interactive, open-source, and browser-based graphing library for Python
- ▶ Supports line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

<https://github.com/plotly/plotly.py>

So, plotly is an interactive open source and min browser based graphing library for python which supports all the majority type of charts like plots scatter plots area charts bar charts etc, etc; all type of charts you can see in the right hand side. So, all like the possibilities are innumerable.

(Refer Slide Time: 15:41)

MAKING A GOOD DASHBOARD



```
app.layout = html.Div([
  html.Div([
    dcc.DatePickerSingle(
      id='date-picker',
      min_date_allowed=date_min,
      max_date_allowed=date_max,
      initial_visible_month=initial_date,
      date=initial_date
    ),
    daq.BooleanSwitch(
      id='show-zeros',
      on=True,
      label='Show Zeros',
      style={'display': 'inline-block'}
    )
  ]),
  dcc.Graph(id='precipitation-map')
])
```

Div

Div

DatePickerSingle id="date-picker" BooleanSwitch id="show-zeros"

Graph id="precipitation-map"



So, how do we create that good dashboard which I was talking about using plotly? So, you know the web browser or the web page is divided nowadays using div functions of html like div tags of html. So, this is the outer div this is the inner div and then this is the graph ok. So, for example, you have to create one precipitation map. So, here the graph id is equal to precipitation map and here the graph is rendered the chart is rendered, but again you have the date picker.

So, this is called the date picker and this is the Boolean switch which is nothing, but you have to show zeros or not. So, for this you can see that you have to use the elements date picker single id Boolean switch single id equal to show zero this kind of html elements. So, why this because these are the ids which will be linked to the Python based code in the back end. So, that is why I am showing you this. So, here it is the code sample.

So, you can create app dot layout where you can create initialize the first div the outer div and then we have the inner div the inner div contains the date picker where you can put a minimum data allowed to a maximum data allowed. So, this can be static fields suppose minimum date it can be 1 January 2001 maximum can be today's date.

And then the initial date what should be the default date we can put whatever today's date as the default date and then for Boolean switch you can put id is equal to show zeros on equal to true; that means, by default it will be on then the label show zeros and then in



line the block this is some design parameter. So, to mix your knowledge of html a bit and python a bit to create these kind of dashboards.

(Refer Slide Time: 17:38)

MAKING A GOOD DASHBOARD

```
app.layout = html.Div([
  html.Div([
    dcc.DatePickerSingle(
      id='date-picker',
      min_date_allowed=date_min,
      max_date_allowed=date_max,
      initial_visible_month=initial_date,
      date=initial_date
    ),
    dcc.BooleanSwitch(
      id='show-zeros',
      on=True,
      label='Show Zeros',
      style={'display': 'inline-block'}
    )
  ]),
  dcc.Graph(id='precipitation-map')
])
```

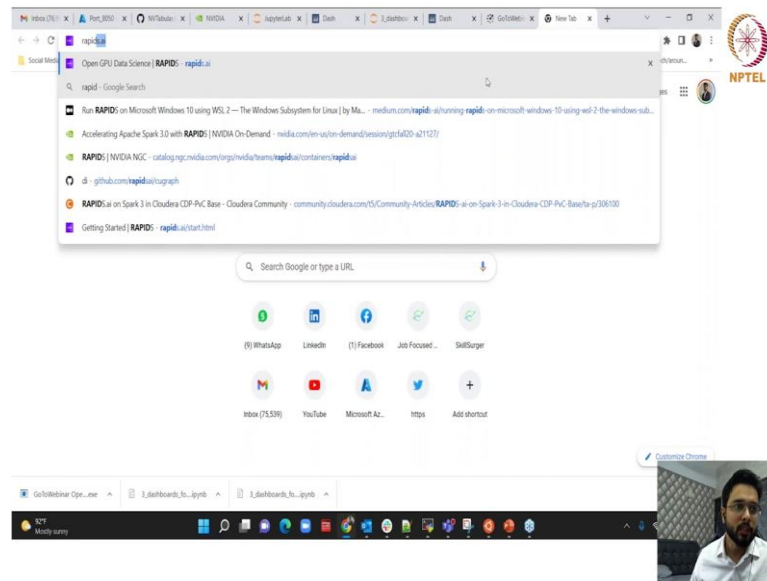
```
@app.callback(
  Output('precipitation-map', 'figure'),
  [Input('date-picker', 'date'), # 1st input
   Input('show-zeros', 'on')] # 2nd input
  def make_graph(first_input, second_input):
  --
  regular python function stuff
  --
  return fig # Must match Output data type
```



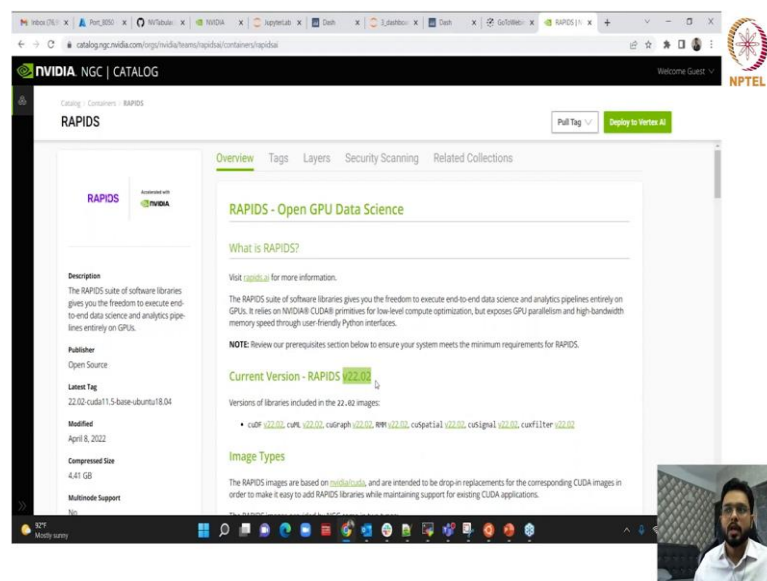
And then we have the callback; that means, how to show that in the browser. So, yes you have to create a callback where you have to show output precipitation map input date picker show zeroes on and then make graph and then so on and so forth. So, this is how you can render that particular created dashboards on the browser.

So, this is how cuXfilter and plotly works. So, these are the two main interfaces or libraries in the rapid ecosystem or suite of libraries which will help you in visualization. So, before I go to NV tabular I wanted to show some demo of how you can do it. So, let me start with the demo for this and then we go to NV tabular and then NV tabular can be shown ok.

(Refer Slide Time: 18:50)



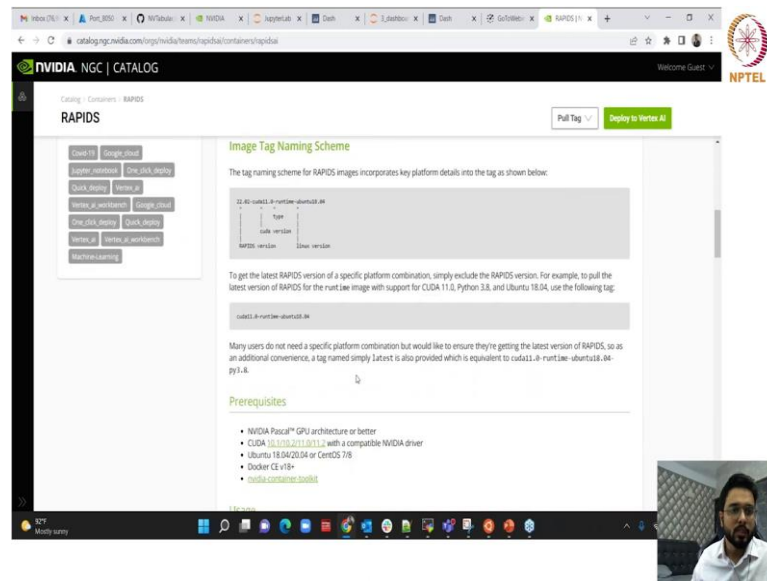
(Refer Slide Time: 19:03)



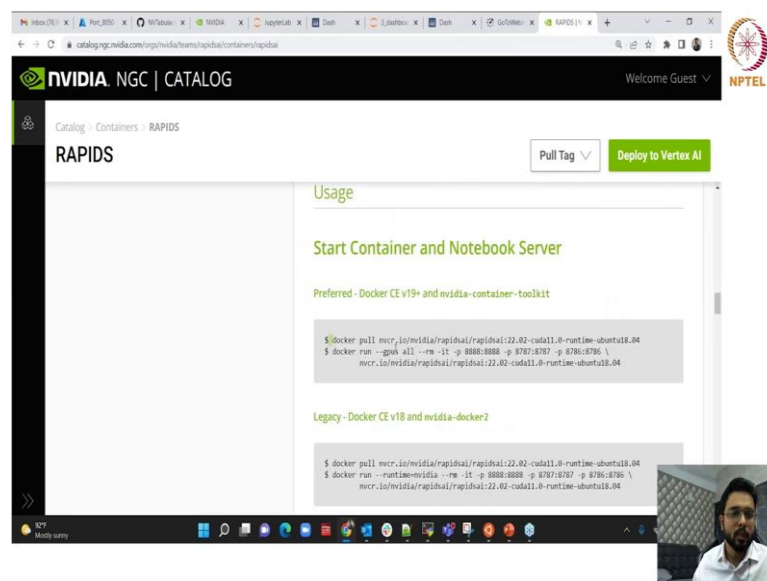
So, before showing you how you can exactly do the visualizations I would like to show you how you can install and use RAPIDS ok. So, we have something called NGC. So, NGC is nothing, but NVIDIA GPU cloud which is a combination of Docker containers or hem charts or pre trained models. So, for all these things we have the central container repository called NGC.

So, in this NGC we have this RAPIDS container also present. So, the latest version of RAPIDS is 22.02 and here we can have RAPIDS present as well.

(Refer Slide Time: 19:36)



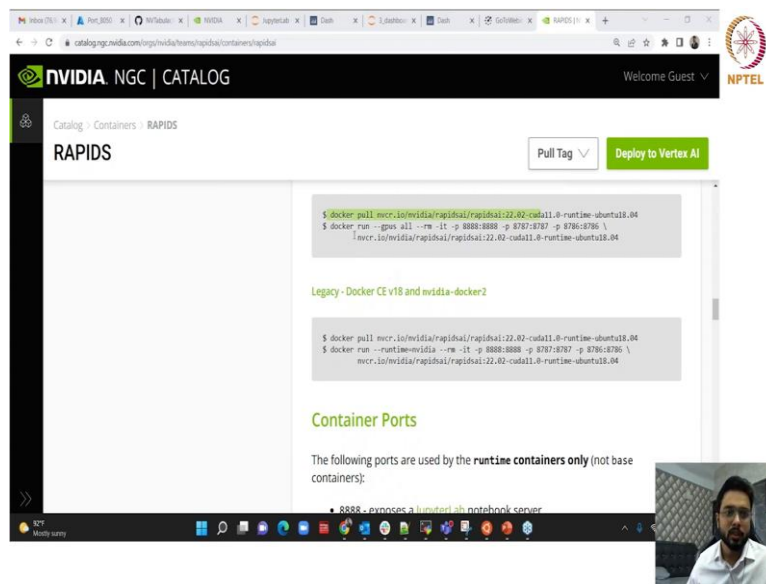
(Refer Slide Time: 19:40)



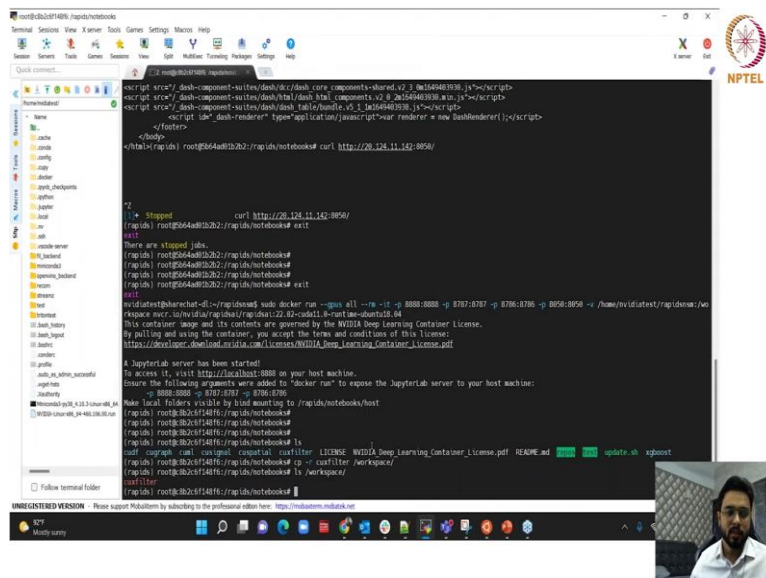
So, what you can do is you can do a Docker pool. So, if you want to see the command let me show you. So, here is the command Docker pull NVCR IO and then you can do that. So, there is one question that you are unable to run RAPIDS in colab the GPU allocated is t yes. So, the problem with colab is you will get a random GPU if you use a free version of colab and if you use that particular random GPU if it is not a tesla version or higher or sorry pascal version or higher then it will not work.

So, basically RAPIDS the minimum requirement is pascal architecture of GPUs. So, at least you should be allocated a P 100 kind of GPU or T4 or a P 100 these two GPUs it would not work in kat because kat is one of the oldest GPU of NVIDIA. So, either you have to restart your environment again and again. So, you get allocated the T 4 or the pascal P 100 GPU to use RAPIDS or you have to use the paid version or you can use the on premise GPU ok.

(Refer Slide Time: 20:58)



(Refer Slide Time: 21:07)



So, either if you use local then you can go to local host 8888, but if like me if you are using cloud then you have to go to that ip address or post name colon 8888. And here you will see that by default many folders are there with the examples of RAPIDS.