Apache Spark Fundamentals

Day 1 Lab

Spark + Iceberg - Memory Tuning, Joins, and Partitions

Transcript:

48:29
going to be the big thing and then you want to go to the data engineer handbook and you go boot camp materials spark
48:35
fundamentals and then in here you'll see there's a spark fundamentals and advanced spark setup so there's two
48:40
options here you can either run makeup or Docker compose up so in here I'm just going to show what Docker compose Docker
48:46
compose up now this is uh setting up and uh
48:51
running all of the docker stuff that it needs to download and install and do all
48:56
the uh stuff to like set up Docker you'll see this needs to set up four containers we need one for Iceberg we
49:03
need one for minio we need one for MC and we need one for spark Iceberg those are going to be the three big things
49:09
that are going to be part of this right and then ultimately we are going to be accessing this from Local Host
49:15
8888 that will be uh where this is going make sure you have Docker set up otherwise this is not going to work
49:21
right and this is going to take a little bit of time because these images are really big right so uh we're just going
49:27
to have to Vibe here for just a little bit while this is uh installing and then we will uh we'll go from there should be
49:35
pretty exciting time here as we uh get to the um actual setup here because I know
49:42
y'all are really excited to get ready to test out some spark stuff should be pretty

49:48
great so I think if we go here is it set up okay there we go so that actually set
49:54
up then we can go in the notebooks and we can go here right and then this
49:59
should uh run so you'll see we have our data pulled in here this is all of the
50:04
data that we have and so if you can get this up and running uh you're good to go for the lab one of the things I want youall to try is um just like running
50:12
this first cell and you should get um this should uh actually show you um some
50:19
data here it should give you back uh a good good data set that will show you
50:24
pretty much what you would want to see here around like URL your browser family
50:30
device family host all all that kind of stuff so one of the things that this is
50:38
probably pretty pretty straightforward so far we're just reading in a table it's reading in a CSV file that is the
50:45
one CSV file that's in the repo so and then we're adding one column here which is the event date and the reason why
50:52
we're adding the event date column is because we want to uh use that later on
50:57
in iceberg but I actually learned that like you don't even need this column for Iceberg like iceberg is actually really
51:03
really smart but um so I just want to kind of go over each one of these lines
51:09
real quick because some of y'all might not have a ton of experience this line actually we don't need
51:14
it so spark is managed by this thing called the spark session so you want to build a spark session so that's what
51:21
spark session. Builder does and then usually you create some sort of app name
51:26
can call we call it Jupiter here which is your Jupiter notebook and then you do do get or create one of the things that
51:33

you might notice here about this syntax is that it's actually uh not pythonic
51:39
because if this was like a pythonic way right this would be get or create right it' be like that and uh one of the
51:46
reasons why it's not that and it's it's actually this it's camel case and not snake case is because spark is a jvm
51:54
library and so even when you're using python uh P spark you're going to still
52:02
a lot of the libraries and functions and stuff that you're going to call are going to be um jvm same like with this
52:09
with column it's with column not with column like that right so
52:15
um most of the style like this is where like ppar can be a little bit funky because like you actually do use Java
52:22
styling but you have um python code and and the main reason for that is that P
52:28
py Spark all it does is it wraps um it wraps the um python it wraps um the
52:35
spark libraries in Python so that then you can just use Python to do stuff right so one of the things I wanted to
52:41
show real quick here though is you see there's do show I wanted to show dot collect um this line uh this this might
52:48
actually out of memory let's let's try it oh yeah I forget oh yeah python is so
52:54
annoying you have to put the back slashes
52:59
okay okay so you see how now we're getting a stage it actually gave us a stage and uh it's um that actually gave
53:06
us we're actually like running spark there okay so there we go wow it's because there's not tons and tons of
53:12
data so generally speaking if this is a very big data set like if it was actual Big Data I think there's like half a
53:18
million rows here but um like that is going to be the idea right is that you

53:24
can also kind of out of memory and crash Spark by calling this DF do collect
53:30
right so I kind of want to show that just because I think it's important to illustrate right because you can do like
53:36
a join here or like here let's uh let's do it this way we're going to say um
53:42
dojin DF and then in this case what we're going to say is um DF or we're
53:48
going to say can we just say lit I think say lit one equals equals lit one so
53:54
like we're essentially going to just like like blow up the data set here by
53:59
like uh doing essentially like every row matching with every other row so this is going to make it like a lot more data
54:05
like and so I wanted to show you how this is going to probably crash the data set right okay there we go so now we're
54:11
saying um okay perfect perfect so you see here um how what we're doing here is
54:20
we have this Java Heap space problem where it's like out of memory because we
54:26
tried to bring too much data back right so what we could do right is if we say
54:32
um instead of collect say we do like do take five now this should actually run
54:38
like I'm pretty sure this will run okay hold up one second let me see because if I try to run collect here again it might
54:44
have just died okay so I I straight up killed the colel colonel here so we got to like we got to like stop him and the
54:50
colel um we got to re we got to restart the colel here cuz I straight up killed the colonel because I uh
54:57
um we did a nice little o there right so uh which is a common thing right so this
55:03
is a great example I just my whole point here is that like why like this is a bad practice that I've seen many times in
55:09

many spark jobs is uh if you call do collect like this make sure that it's
55:14
filtered down you don't ever want to pull in the entire data set into the driver ever it's just not something
55:21
that's not something that we do so I'm pretty sure though if we do take five this should still run see see how it's
55:26
actually like thinking about it this time and it's not just like barfing because I'm not trying to bring back
55:32
like because that's 400k times 400k I think that's like 1.6 Bill think so this
55:39
is still going to take a little bit because we're doing this like crazy cross join right but the thing is is
55:45
this is now like whenever you're bringing data back to the driver it should always be limited or like or
55:54
another way to think about it is like this should be aggregated or limited or anything like that there should be
55:59
something that happens that uh is the colonel like even running it says the colonel is Idle did I like straight up
56:05
kill the colonel right now okay let me try and run this again because this should be okay there we go now we're
56:11
it's alive now I just had to refresh so this is going to be running now and this will actually I'm pretty sure this is
56:17
not going to O because um see boom didn't oom even though we did a crazy
56:22
cross joint here right so be aware of that bad practice because
56:27
I've seen that bad practice many many many many many many many times in my career right and so uh you want to like
56:33
filter this down like either like uh so generally speaking the way I think about it is like you want to be using take and
56:40
show and not really using collect collect is the only time you want to use

56:45
collect is if you know for sure that the data set that you have here is like
56:50
aggregated down and you're uh you can use that as your data set so that's
56:55
something thing that I um just am cautioning you about because I think it's very important and just kind of
57:02
showing you how like hey you can kill spark you see how I killed spark there I like it's great I love killing spark
57:07
because it's like um it's it's a fun thing to um troubleshoot so anyways I'm going to get rid of this cross joint
57:13
line because it's going to make everything really slow later down but um uh let's talk about um some more um
57:21
lines of code here that are going to be uh different and kind of things here
57:26
that are that like I want to show you about how these lines work and how we're
57:32
going to split things up and we're going to figure things out but uh here's the idea behind what this is going to do and
57:39
like how this works right so um you'll see DF do repartition so what this is
57:45
going to do is however many partitions we had before we're now going to have 10 partitions and we're going to split it
57:52
up based on event date that's going to be uh the column that we're looking at and that's how we're going to split data
57:58
up um and then we have sort within partitions so I want to show you the difference between a couple different
58:04
things here so we have uh so you see how we have this one I want to copy this line here and I want to um we're going
58:12
to create another line here but you'll see you can also just do do sort uh and
58:17
this is going to be um we're going to call this sorted to and then we're going to say uh sorted to.show

58:24
so um because these two are actually
58:30
um uh the same in some regard but they're also very different so let's see
58:36
if this actually works okay so you'll see that uh one of the things you'll
58:42
notice between these two is they actually pull different data like you'll
58:47
see that um in this case we have data all the way back to
58:53
um uh 2019 the 12th and then here we have data back to the 14th right so it's
59:00
like why is that the case like what is the difference between these two right so that's one of the things that I think
59:06
is important to talk about because um these both are different and they
59:13
actually are very different at scale so I want to just really emphasize this
59:20
with y'all because what I'm about to show you is another very powerful thing but you want to make sure that you
59:26
understand these apis clearly to understand why why these are different at scale so let's talk about what so it
59:35
says sort within partitions and this is sort so if you do sort within
59:41
partitions it will just look at it will take every like the 10 the 10 partitions
59:47
that you have it will look at all the data in those partitions and sort all of them kind of locally within that
59:53
partition and in this case we're going to sort on event date host and browser family and um the difference between
1:00:00
that and sort is you'll see here um with do sort we actually get data all the way
1:00:06
back to um like this literally does what's called a global sort so this is
1:00:12
sorting data as far back as you can go and like it but it it doesn't just sort the data within the partitions it does a

1:00:19
global sort of all the data and so um that GL mobile sort like that is
1:00:27
actually um very slow uh so these two um because the data
1:00:33
isn't very big these two like in this data set are going to be about the same um I think we can uh can we do uh I
1:00:42
think we can do this I think we can put a do explain in here and I think this will give us what we're looking for
1:00:49
because we want to check out do explain okay
1:00:54
so oh it's cuz Dot show doesn't exist so we'll just get rid of do show you just got to do um just do
1:01:01
explain okay so what we want to do here is let's look at the difference between
1:01:06
these two um plans just so we can kind of understand how they are different
1:01:13
because they are significantly different here and um I hope that y'all can
1:01:19
understand what's going on here and because this is where using do expain can make a very big difference uh in
1:01:26
understanding what is happening with like the difference between this sort and this sort so in this case okay like
1:01:33
we see so the way these explained plans work is you go to the the most indented thing first and this is going to be the
1:01:40
first step which is we read a CSV file makes sense right and then we select our
1:01:46
columns right which is going to be that's what project means project and select are the same thing right and then
1:01:52
we do an exchange so this Exchange in this case is going to be uh uh that's
1:01:58
this repartition line right that's what that's going to do exchange and then we have our sort here and uh that's going
1:02:05
to be the next line and there's this uh you'll see there's a difference here this false and this true and uh that's

1:02:11
going to be based on whether or not they are doing a global sort or not and so
1:02:17
that's going to be the next kind of line and then you have after that you have the pro project right and then this is
1:02:23
the final select where you're select out the columns again so you see that like we ended up selecting these same columns
1:02:30
from here and here those are the same like list of columns so how is that different from this one and you'll see
1:02:38
that this one has one more line in it though cuz you see here's the hash partitioning where we're where we
1:02:45
repartition um into 10 but then you'll see this um you see this exchange here
1:02:50
this exchange range partitioning you see this guy uh you see how this line is is
1:02:56
not in this line see this uh range partitioning here this is the line that
1:03:02
is going to be painful at scale this line is going to be very very painful at
1:03:07
scale so that's how you can tell when you look at these partitioning plans like so what this is saying is okay
1:03:13
we're going to shuffle once here and then we're going to shuffle again that's what this uh every time you see the word
1:03:20
Exchange in a query plan think exchange means Shuffle Shuffle and exchange those
1:03:27
are like uh the same sort of words that's exactly like they're synonyms when you're looking at the plan and when
1:03:32
you are running the job so you'll see when you use the do sort here uh you get
1:03:38
freaking it's nasty it's real nasty because you get this double Shuffle right but if you use sort within
1:03:44
partitions you only get one like you just have one but this like honestly if
1:03:49
you like if you remove that repartition all right let's run this again and you'll see now in this case

1:03:56
right there's just uh we have the file scan we have the project and we have the sort no Shuffle right because we I got
1:04:03
like just deleted the repartition but like in this case even without this repartition so if we just have do sort
1:04:11
here and we run so we get rid of both repartitions you'll see we still are going to have this range exchange range
1:04:18
here and this is like especially if you have more than like I don't know like 20
1:04:24
gigs of data this this line is going to be very slow because the thing is is like when you do a global sort you have
1:04:30
to pass all the data through one executor because that's the only way you can guarantee that the data is globally
1:04:35
sorted it's the only way so um my main point with this is at scale
1:04:43
you want to use sort within partitions and not sort you should almost never use do sort sort is like I don't like do
1:04:50
sort I think it's whack like I don't like you should not think about using do sort so um this is this concept right
1:04:59
here can make uh your pipeline way more efficient as well because knowing how to
1:05:05
sort stuff when you write it out makes a big difference and we're going to talk about that um in here in just a second
1:05:12
okay great now what I want to talk about now uh to shift gears here a little bit
1:05:18
is around um we're going to talk a little bit about paret talk a little bit about iceberg might go a little bit over
1:05:24
today but but um so we talked about sorting super important I want to run this sorted line again because we're
1:05:30
going to be using this sorted line so what I want y'all to do is also try to run these other kind of boot camp squl
1:05:36

lines cuz uh they're going to be important here so um okay so let's go to
1:05:43
this line where we're creating this um Iceberg table so you'll see in here we can uh you can partition your data
1:05:52
however you want in iceberg and it's cool cuz like you can actually uh like in this case you can say years like or
1:05:58
you can say days you can even say hours here or years or days right and then
1:06:04
that will actually look at that time stamp or date field and it will just split it there but you can also just do event date I'm just going to do I'm
1:06:10
actually going to do event date it doesn't matter you can keep it as years it doesn't matter that much because we're going to be running all the data
1:06:15
together so then we have that and then um I'm also going to put the partitioned here partitioned by um event date in
1:06:24
this one GNA run this did I oh there's a
1:06:31
semicolon so hopefully y'all were able to create your tables here and um that
1:06:38
is working and uh and you have all of that working you can have the
1:06:43
partitioning there or not it doesn't matter it doesn't matter that much cuz like we will be able to still illustrate the point of this so you saw how like we
1:06:51
had our um we talked about this like sorted DF but we're going to go back to just DF here and we're going to um
1:06:57
essentially do a couple things here one is like we're going to repartition we're going to have four partitions here
1:07:02
because we want to talk more about like when we write our data write our data out so with this first one here we're
1:07:09
going to do a sort and we're going to do sort within partitions and we're just going to sort on um uh like we're going
1:07:17
to sort on event date browser family and host we're going to do all three and then uh this second one here we're just

1:07:24
just going to sort on a event date those are the only two things that we're going to do differently here and um I don't
1:07:30
think we need this line I don't think this is needed because we have start DF okay so actually no this one is not even
1:07:37
needed at all because we're not writing here so we're we're comparing these two right so one of these has um this like
1:07:43
event time right and then it's not sorted and then the second one is sorted
1:07:49
one second I want to drop these real quick so let say drop table can we drop I want to drop these
1:07:56
cuz I want the I want the number of partitions here to be like based on the
1:08:01
date so can we like run this oh you got to you can only run one
1:08:07
command at a time it's so
1:08:12
annoying okay because you you guys don't need to do this CU like you already like have it working so let me cut this
1:08:20
guy run this and then run this run this
1:08:26
okay so now we have it all partitioned up and we're going to run this repartition line again write everything
1:08:32
out great now in this line line 21 we can uh we can look at the file sizes
1:08:37
again um well this didn't actually finish I guess like that doesn't make sense that it's still only four
1:08:45
files because now they should be partitioned on more because this is boot
1:08:51
camp that event sorted how does that work why is there only four files interesting um because there should be I
1:08:58
guess dates here and uh because that's interesting but okay
1:09:04

anyways like I want to talk more about uh how this works because I think this query is probably really crazy to y'all

1:09:10

because this is one of the ways that you can actually look at the sizes of your data sets right so in iceberg after you

1:09:16

write your tables out you can then um query them like you can query like so you see this is like demo this like and

1:09:23

then you have your this is the highest level then you have boot camp. events sorted this is the table and then files

1:09:30

this is the metadata of that table and like for example let's uh let's just like run um something here that is a

1:09:38

little bit simpler like kind of like a select star here and you'll see okay so

1:09:43

this is going to be um the files that we have written out and this is going to be all of the um kind of data sets oh

1:09:51

interesting it's because it's saying event date is like null is it because I don't even have event

1:09:57

date up here oh yeah okay event date's right there should be there okay well um

1:10:05

apparently it's saying out there it's always none but is that the case then hold up because if we do a select star

1:10:12

is there like a bug in this second because it's like no okay event date is definitely there it's right

1:10:19

there it's totally there okay it's fine it's just like weird that it's not

1:10:24

writing it out out the way that you would expect cuz like there should be more there should be a lot more files there should be one file like per

1:10:30

partition cuz see the partition here is saying that like they're all like none and oh because did the table not

1:10:37

actually get dropped I think the table must have like not gotten dropped or something like that so what I want to try here I'm just going to try something

1:10:43

real quick I'm going to put V2s on all these and just make brand new tables because Iceberg can be weird sometimes

1:10:49
okay so we made brand new tables and then we're going to put V2 on all of these and then we will uh see if that
1:10:56
um does the thing I'm looking for so now if we run this line of code
1:11:02
we should get way more files
1:11:08
what that doesn't even make sense like there should be way more files here than just like the okay but see this actually
1:11:14
gives a lot of like metrics about the sorting and stuff like that but one of
1:11:19
the things I wanted to go over with this is um you'll see there's like a couple things in here you see file size and
1:11:24
bites so say some file size in bytes and then um oh it's cuz I was not quering V2
1:11:32
that's why Wow Wow rookie move rookie
1:11:38
move these are V2 okay apparently that it's not actually like
1:11:46
there should be more uh partitioning here than just like none it should be like uh it should be based on like the
1:11:51
actual like unique values of the date because if this is days right okay
1:11:57
anyways I don't want to like get caught up in that like what I want to show is cu even with this example like it still
1:12:03
has what I'm looking for and we can still look at it file file size in bytes
1:12:09
and we want to say count one and then I want to say and this is as num files this is as
1:12:17
size and let's uh go with this we want to say um oops I want to say Union all
1:12:24
and then we have events unsorted okay so you'll see here one of
1:12:32
the things that you get here is you see how the file size is significantly
1:12:37
different like you get like 10 or 15% just by sorting and the main reason for
1:12:45

that is this comes from run length en coding where if you have um all of the low cardinality things together then
1:12:53
you're going to get better um better partitioning that way so um for
1:12:58
example like let me show you like if we instead we only we're only going to sort
1:13:04
on date and not on we're not going to also sort on browser family and host because I think this is going to give
1:13:09
you a kind of an example of how this works so okay that ran okay so now you see um you see how
1:13:18
sorting on date was like it helped like just a little bit just a tiny tiny tiny
1:13:24
bit right it's like the difference between the sizes here is like the same for the most part so um one of the
1:13:31
things that you can do here is you really want like like when you're writing your data out you want your data
1:13:38
to be written out from uh your um your lowest cardinality first all the way
1:13:44
down like to your highest cardinality like when you're sorting data so in this case event date is going to be the
1:13:50
lowest cardinality because there's only going to be like a couple hundred records there and then um then you have
1:13:55
a browser after that and then probably host there's other ones in here that you could probably put as well there's like
1:14:01
uh like OS family other things that could possibly like you can kind of play around with this stuff and this is
1:14:07
something that that you should definitely do when you are kind of like working with writing your data out and you'll see okay so you see but in that
1:14:14
case um OS family was not nearly as good as host when you're writing out right
1:14:19
cuz it was like what it's a 2 million 2.9 Million and then what is it here
1:14:25

see how this is like 2.8 million you get like a significantly bigger size reduction when you're sorting and
1:14:31
writing data out with um uh start DF and like you get a better compression from
1:14:37
the Run length en coding and you can kind of shrink your data down this way and like if you care about stuff like
1:14:43
this as a data engineer like you're going to make way more efficient pipelines awesome well everyone uh
1:14:48
thanks for uh thanks for coming to today's uh lab congrats on making it to the end of spark Basics make sure to like comment and subscribe and keep
1:14:55
crushing it I know you're going to get through the rest of this boot camp [Music]