r/apachespark • 4 yr. ago
Javaguy44

Spark performance: local faster than cluster

I posted to stack overflow here:

https://stackoverflow.com/questions/63834379/spark-performance-local-faster-than-cluster

Was wondering if someone could help me out / advise on why my standalone local[*] is faster than a cluster when my cluster has 2 reasonably beefy and identical machines.

Thanks v much

6 16 Share

Add a comment

Sort by: Best Search Comments

ed_elliott_ • 4y ago •

You need to look at the spark ui, probably you are getting shuffles on the cluster so more data copies between nodes but there are quite a few things to look at.

11 Reply

nieuweyork • 4y ago •

If it's faster locally, you don't need spark.

If you can provision your cluster with only 2 nodes...you probably don't need spark.

11 Reply

everything else is the same, if I double the amount of RAM and CPU in the cluster, my expectation is that the time time should reduce. I'm not expecting the time to cut in half but there should be some performance gain

0 Reply

[deleted] • 4y ago •

dub-dub-dub • 4y ago •

It really depends. How many partitions are you using? How much data are we talking about?

1 Reply

[deleted] • 4y ago •

So, if you are working with a small dataset- which it sounds like you are- spark will probably be slower than straight python, and using parallelism will likely be slower as well since there is overhead involved, while parallelism benefits are negligible.

Follow up- have you set the property "spark.sql.shuffle.partitions" anywhere? For such a small dataset as this, I would recommend setting spark.sql.shuffle.partitions= num executors * num cores. That is because this gives you the maximum number of tasks which can run in parallel. If you use the default number of shuffle partitions, which is 200, and only have 16 tasks running at once (2 executors, 8 cores each)- that's going to take a lot longer! So if you're using the initial setup in our stack overflow q, try setting that param to 16.

1 Reply

Javaguy44 OP • 4y ago • Edited 4y ago •

Thanks I'll try the shuffle! That's a good tip and hopefully works

Re small dataset - I guess perhaps with the data you might work with it's small. But 30 min to process that dataset on a stand-alone with really good hardware seems like it's a reasonable size test imho + seems like that could be optimised by adding a same spec machine to the cluster

In EC2 terms my Mac instance size would be equiv to a single c5d.2xlarge taking 30 min to process this dataset. Adding a 2nd c5d.2xlarge to the cluster should hypothetically result in performance gain. Can appreciate if getting this performance gain might not be low hanging fruit hence after a couple of days of tuning and changing spark submit parameters I posted

1 Reply

[deleted] • 4y ago •

What's the size of the dataset unzipped on disk actually? I only saw the zipped size initially.

1 Reply

Javaguy44 OP • 4y ago •

spark.*conf*.set("**spark.sql.shuffle.partitions**", 100)

Thx!

1 Reply

[deleted] • 4y ago •

Gotcha- so then I would still say that the dataset is too small to see a performance improvement by adding another machine. The tasks are small enough that splitting them up hurts you more than it helps. You also have to consider that when all operations are running on a single machine, you don't have data needing to be shuffled across the network. So by adding more machines, you will take additional time to shuffle between the two. Of course, with a larger data set this is well worth it! If I were running this, I would start by changing the sql.shuffle.partitions to 16 (because again, with this param set to 100, you are running only 16 of those 100 tasks at a time), and change your repartition statement in your code to repartition into (spark.sql.shuffle.partitions) blocks instead of 100 as well. If you see a performance benefit from there, you could try decreasing the number of cores & shuffle partitions even further. The optimal data block size for each spark task falls between 128MB and 1GB depending on who you ask, so 16 still feels a little high.

3 Reply

Javaguy44 OP • 4y ago • Edited 4y ago •

Thanks - understood re: shuffling of data.

I've changed the sql.shuffle.partitions to 16 and am re-running.

Re: data size, for my base case, we usually deal in sample data sets of 250MB to 500MB in size. Would you say then that a spark cluster provides no benefit for this base case? Because we would still benefit greatly from reducing that time from 30 min for a 265MB file to something sub 10 min (get a coffee and back)□. Thats ultimately what I'm trying to evaluate. And its too slow as well in straight python (otherwise I wouldn't be doing this R&D). thx

1 Reply

[deleted] • 4y ago •

Was that intended to be 500 GB or did you mean 500MB? If you meant 500 GB- you would definitely see improvements! For the latter, I would try to do it with multi threading in python on a single machine.

1 Reply

Javaguy44 OP • 4y ago •

just multi threading in python / Scala is best. Its no problem to multi thread in Scala for us - again I was just hoping to see what Spark could help with

1 Reply

thepinkbunnyboy • 4y ago •

At the end of the day, 500MB of data can be read and written by python or Scala in seconds, so if you're seeing slowness it's due to what you're doing to process it. Check to make sure your algorithms are efficient, that you're not doing any n^2 algorithms that can be done in nlogn, for example.

You might also be using Spark wrong. Are you using head or collect anywhere (except, perhaps, the very end of your processing)? A common beginner misstep is not understanding how to use the dataframe API correctly and using operators that actually bring the whole dataset onto the driver before it's necessary to.

1 Reply

Javaguy44 OP • 4y ago •

Thanks. It's actually not my code it's from a udemy course. I was just trying to do an apples to apples comparison test between spark standalone vs spark cluster and simplistically assumed that adding an identical beefy machine to the cluster would give me some type of performance gain (which sadly it has not)

1 Reply

tal_franji • 4y ago •

shuffle and task/container dispatching is a huge overhead. Spark starts to be better than local or database only on really big data. If you think your data will really grow - starting with local spark is a good way to write code that will allow scale later.

1 Reply