4.2.0 Speed, Scalability and Large Collections

Hi, today we're going to be talking about speed, scalability and large collections.

So, in the modern day, there are new data types constantly being created. If you start thinking about things like the Internet of Things, the industrial control system suddenly being connected and monitored, we've got a whole new data types just constantly being created, and an ever-increasing amount of data.

So, the data sets that we're trying to analyse are also trying to grow constantly. systems that exist for dealing with these on an isolated scale, or small data functions just don't scale when it comes to dealing with all of this data in one go.

One of the big reasons for this is because high level languages, like for argument's sake, Python, or PHP, or Java, they are designed to be as user friendly as possible as a language decoding and it means that they've got some predefined limits to the inputs that they can handle. So, looking at some of the older languages, they had limits of things like 255 characters on a string input, for argument's sake, and right now we're creating objects and nested objects, and, you know, arrays, and associative arrays, and all of this data that just, it's too big to put into some of the coding languages that exist, especially the high level languages and you take all of that as a problem.

You combine all of those problems together, and it is exacerbated by the fact that the world is used to and wants instant access to the information that they feel they need at that given time. So, we've got more data types to deal with, we've got more data within those data types to deal with. We don't have the functions in our prebuilt systems to handle it and the world wants the answers right now. So, these are some of the problems that we've got when we start looking at speed and scalability of any solution that looks at big data analytics.

So, what are some of the solutions? Or how can we optimise what we're doing as big data analysts to ensure that we can service these needs and we can meet these problems head on.

One of the big things that we can do is when we're designing our systems, when we're building our systems, we can ensure that we only have to run the data through our system once. Looping through the data multiple times takes a long time to do and is redundant. You're wasting effort. If you have the information in front of you or running through the process at the time, you should do all the checks that you need to do on that data as and when it's there, as opposed to constantly looping back through if possible.

We're going to utilise things like random access. Now, random access is an interesting principle. It is the ability to build into most languages. And it's the ability to go directly to a part of a file that we want to access. So, if we assume, we have for argument's sake, a text file, so we have a text file that has got two or three thousand lines of text in it. But we know that we are only after the last 500 records that were in there, we can skip the first couple of

thousand lines and jump straight to the ones that we want. So, we don't have to parse the entire collection, we don't have to parse the whole set of data and bring it in line by line and look at it, we can start where we want to do and that's called random access.

We should also use things like time-based functions to monitor how our processing is going. There's a lot of systems over the years that have fallen apart, not got off the ground, and been underutilised because they weren't able to pick up effectively where they left off. They weren't monitoring how far they got. They weren't assigning timestamps to the things that they'd already analysed, and they weren't cueing things that were coming in as new data into that time-based process so that it can be picked back up later down the line. This is something that will save a huge amount of processing power and a huge amount of repetitive analysis later down the line.

We want to make sure that our analysis uses simple logic operators as possible. So, things like add, multiplied and XOR functions or XOR functions are things that are built-in logic operators, things that are built into the computer that the computer knows how to do, and can do very, very quickly.

And the key here for a lot of systems is to only analyse what's needed to the point that it's useful. So, having a huge amount of data, there is a tendency to say “Okay, what’s all of the answers that I can get from this data, I'm going to analyse it, I'm going to get all possible answers out of this data and I'm going to I'm going to build my system so that it's going to do absolutely everything and nobody will ever have to touch this data again, because I will have pulled all of the answers out of it in one go”. It's an it's a nice thought, but does it serve the purpose to which the analysis is being done?

If you only need a certain answer to come out of the other end of your analysis, then only perform the things needed for it. One of the things that can help with speed and certainly scalability, is pre-processing. So, this is reviewing the data and doing sort of a pre analysis of the data and there's a couple of methods used for this.

So, one is correlation. This is looking at a large data set and seeing if we can put any of the data values in groups, do any of the data values already correlate and give us some indication of how we should maybe treat the analysis. So does one large data set once put through a correlation algorithm something like a dot product, end up with two or three correlated areas that we can then apply two or three processes to. So, we can break our one big analysis down into two or three more specific analysis, which will run faster and more optimised than trying to analyse everything in a large process in a large data set.

The next one is clustering and classifying. So, these are ways of dealing with unknown or uncorrelated data. So, something that when you run through a dot product just brings you back these things don't relate to each other, these things are not related in any way, shape, or form.

What we can say is okay, they may not be related to us, but if we apply something like a cluster algorithm, such as k-means, we can select x number or k number of random data points and then try to correlate or try and cluster the rest of the data points to that random

selection, which will allow us to see how far apart, what the variance actually looks like in the data set. And understanding this variance in these clusters and how far away from our random points they are, will allow us to be able to get some insight into how we should proceed and help us to plan what our analysis is going to need to look like.

Another way of doing this is with classifiers. And the difference between clusters and classifiers is that classifiers are user defined known classes. So, we predefined a class, so for argument's sake, if we were looking at data around vehicles, we could have one class that is a car, one class that is a motorbike, one class that is a van and one class that is a lorry. And then we can then apply our data set or our data points to those classes. And we're saying, Okay, how close is this vehicle to being a car? How close is this vehicle to being a motorbike? And at that point, we've now got broken things down into classes. And we can then treat them accordingly based on their proximity to known classes, or whether they sit evenly between two.

The final part of part sort of pre-processing your data is to look at how persistence might work with that data. So, it's very, very easy to bring data into a programme, have that programme, perform logic on it, give you the result, and then forget the processing that it did and forget all of the steps it took along the way.

When we look at persistence, we look at how can we record everything that happened. How can we save data values? And, how can we save important steps and stages along the way to make sure that we can pick things back up if we need to go back to something, but also so that we can get others to verify our results, or even build on the analysis that we did in a second phase. And this can be done persistence can be handled using file outputs, databases, objects, you know, hashing, you know, hashing your results of particular analysis, and then comparing those hashes. And there's a number of other ways as well.

Once we've got an idea of some of our clusters of data, and we know what classifiers we're using, and we've got some correlation, and we've maybe got our persistence in place, we're then going to move on to how do we gain understanding what's going to be our yardstick in effect for some of this, and can we do any other pre-processing that will give us insight into the data before we go ahead and do our main analysis.

So, one great technique here is the use of denominators, if I said to you that “I had created a new software, I built this software, I've released it onto the market, I've given my software to 1000 people, and 1000 people have come back to me and said that that software is the best software they've ever used. And I come to you and I say I've got 1000 people and this 1000 people say this is the best software that they've ever used”. Sounds impressive, it sounds amazing, sounds like it's the best software that has ever been made by man.

But, what about if I added a denominator? And a denominator is a value that gives meaning to another value. So, me saying 1000 people who have used my software think it's amazing, is not as meaningful as me saying 1000, out of the million people that I actually gave the software to thought it was amazing. Adding the denominator of 1 million, makes my results a lot less impressive.

In order to apply denominators, it usually requires that you've normalised the data, and there's some sort of categorization has happened prior to the analysis, so that you can break it back down.

Another technique here is to use frequency analysis. So, we can look at when we're looking at qualitative data, we can look at things like Pareto's principle, where it's also known as the 80/20 rule. So, 80% of the information will be contained in 20% of the values. So when you analyse something like a book, if you broke it down, and you counted up all of the usages of every word in that book, you'll actually find that 20% of the words will take up 80% of the actual text. And this applies in lots of other areas as well, the 80/20 rule, the Pareto's principle is pretty well known. Schools are a good example of this, where a about 20% of the children in the school will cause 80% of the problems. This also happens in work, if you're going into work, you'll probably find that they'll be a minority within the office or within your working environment that seem to cause a lot of the problems and seem to be the root cause or involved in a lot of the problems and it'll roughly work out at 20%.

We talked about last time, looking at the maximum and minimum values and some of the insights that that can lead to, but actually outliers within a data set, I mean true outliers, so outliers that are results that may at first look anomalous, they look like they don't belong, but actually, they do adhere to the same rules as all of the other results. So, they are valid results.

And a good example of this use of outliers to gain a further understanding would be Isaac Newton and his friend Edmund Halley, who of course had a comet named after him. Edmund Halley posed a question to Isaac Newton around the orbital problem. At the time, they couldn't figure out how orbits worked and it was a big question mark as to how everything orbited around each other and how things were held together and why everything didn't collide and why everything didn't just fly off into space. Now, as it turns out, Isaac Newton had given this a bit of a thought and he explained his theory to Edmund Halley. And eventually, this theory was written down and became Newton's theory of everything, it became the theory that explained gravity, explained how things relate to other things. This theory came out of Edmund Halley as posing a question based on some outlying results that nobody could make sense of. They understood how things worked on a small scale, but when they made it a big scale they couldn't grasp it, they didn't have a theory to explain it and Isaac Newton was able to create his theory of everything based on that outlier.

Another technique that's used here, quite a lot is back of envelope or in the UK, we tend to refer to this as back of a fag packet. You know, this is, can I do some high-level estimated calculations based on the initial data, that will give me some guidance on what I'm expecting to see when I do a full analysis.

So, a good example here is the impact of a pandemic on total deaths that are happening. So, if we wanted to meaningfully be able to say exactly how many deaths were related to a pandemic, we would have to wait for the pandemic to be over, we'd have to wait for every autopsy of all patients that died during that pandemic to be concluded and we'd have to allow for all of the potential disorders that could be or could not be related to that

pandemic, or could or could not have been exacerbated by that pandemic. And we'd have to wait for all of these results to come out before we can conduct a thorough analysis and come up with an exact number of deaths due to a pandemic.

The back of an envelope way to do this, which can give us great results that we can act on during the pandemic, would be, let's take last year's data, let's take the average of the last 10 years data the last 20 years data for how many deaths occurred during the months that the pandemic was going on for.

If we can say, in April, last year, we had x number of deaths. And in April, this year, during a pandemic, we had y number of deaths, then we can surmise from that, that the increased level of deaths between x and y is the deaths relating to the pandemic. We can make this a little bit more robust by going back for a decade, two decades, and taking an average of deaths in April and then compare them to this year. That's not something that requires a huge amount of processing. It's not something that requires us building a bespoke big data analytics tool, but it can give us results that we can act on right now.

So, this concludes our session on scalability, speed and large collections, and how we can deal with them.

Next, we're going to be talking about random numbers and they use in analysis