In understanding how text aware products are produced, it seems that they are not easily developed because of they can be in the future. For example, in our text, *Applied Text Analysis*, it discusses a questions and answer base system for a product. In the beginning, the system will only have those questions but it will continue to gain questions turning this system into a larger application that will require more computational resources for it to distribute the workload. Therefore, how does one make this system scalable and efficient before it becomes a large beast aboard a runaway train or a massive entity that would require all hands on deck to decipher? Throughout the text, it explains how there are two different options based off the concept of parallelism. The first being task parallelism which is when different independent operations run simultaneously on the same data, and data parallelism which the same operation being applied to many different inputs simultaneously. (Chakraborty, 2019) With these two different forms of parallelism, they have a few key differences between them. (Chakraborty, 2019)

|  |  |
| --- | --- |
| Data Parallelism | Task Parallelism |
| Same task is performed on different subsets of same data | Different tasks are performed on the same or different data |
| Synchronous computation is performed | Asynchronous computation is performed |
| As there is only one execution thread operating on all sets of data, so the speedup is more | As each processor will execute a different thread or process on the same or different set of data, so speedup is less |
| Amount of parallelization is proportional to the input size | Amount of parallelization is proportional to the number of independent tasks is performed |
| It is designed for optimum load balance on multiprocessor system | Here, load balancing depends upon the availability of the hardware and scheduling algorithms like static and dynamic scheduling |

When looking to the difference of these two methods for distributing the workload, it seems that it depends on the time you have for the process and the resources that you have available to spend on a particular project. Looking towards the options that are available for multi-processing, Python looks like it is not the viable option for distributing the payload and optimizing the execution of several threads. The programming languages that have the best chance for this would be C, Java, and GO since they can take advantage of operating system threads to provide concurrency and in the multicore case, parallelism, from within a single program. (Bengfort, B., Bilbro, R., & Ojeda, T., 2018) Since Python can only access one core and it is interpreted, it does not allow the language to spread out and take hold of the various threads like the other languages.

References:

Bengfort, B., Bilbro, R., & Ojeda, T. (2018). Applied text analysis with Python: enabling language-aware data products with machine learning. Beijing: OReilly.

Chakraborty, A. (2019, October 11). Data parallelism vs Task parallelism. Retrieved May 5, 2020, from <https://www.tutorialspoint.com/data-parallelism-vs-task-parallelism>

Raschka, S. (2014, June 20). An introduction to parallel programming using Python's multiprocessing module. Retrieved May 5, 2020, from <https://sebastianraschka.com/Articles/2014_multiprocessing.html>