CSE 303: Quiz #3

Due October 2nd, 2019 at 1:35 PM

Please use the remainder of this page to provide your answer. To submit your answer, create a pdf whose name is exactly your user Id and the "pdf" extension (e.g., abc123.pdf) and email it to spear@lehigh.edu before the deadline.

Servers based on the node.js framework are single-threaded, but make heavy use of concurrency in the form of asynchronous I/O. Servers based on the Java Spark framework are multithreaded. Ignoring the fact that one of these frameworks is implemented in JavaScript and the other in Java, discuss the merits of each approach (you need not do "pros and cons", since the "pros" of one are likely the "cons" of the other). You may want to consider workloads where one or the other ought to perform better, programmability of each framework, or other factors.

Node.js is single-threaded but uses asynchronous I/O to maximize the efficiency of that single thread. Node was built as an experiment to see if async could beat out the typical multi-threaded (one thread per request) sort of approach. As it turns out I/O and network calls seem to be one of the biggest bottlenecks in a program's efficiency. Typically, when you make a call to the I/O in a synchronous setting, the thread handling the request will initiate the call and then block until it receives a response. This blocking is where node excels because it has completely removed the need for blocking by using async and allowing the thread to continue doing operations while it is waiting for the response from the I/O call. Along with this saved processing time, it has all the benefits of being only a single thread. These benefits include scalability and ease in development. The single thread approach is also much more scalable when your program is doing many requests. It is much more resource intensive to create many new threads to each handle only one request; in addition, if each thread gets blocked on I/O operations it creates a lot of wasted CPU time which can have costs. The other benefit of a single thread is removing the difficulties associated with multi-threaded development. When dealing with multiple threads the developer needs to be aware of and handle problems such as race conditions and deadlocks, which are not issues that arise in a single thread.

On the other hand, java spark takes advantage of multi-threading which allows multiple operations to be done all at the same time. The biggest benefit of this is its ability to handle CPU intensive tasks by splitting up the work into multiple different processes. As time goes on, CPUs are starting to take advantage of having multiple cores. By multi-threading, the server can take advantage of all available recourses that come with modern processors. Multithreading can also excel when dealing with many small requests. Since small requests don't block for long, the program can still be efficient with the thread and be able to do its own computations free of the other requests being done. Finally, multithreaded programs could also help save resources if the CPU has multiple cores because it allows an intensive program be done with ease across more cores.

In conclusion, there are uses for both frameworks and each can be efficient depending on what needs to be done. If many I/O operations are to be expected, then node.js will be the better choice since node can remove the blocks associated with I/O. However, if a program is very CPU intensive or dealing with only small requests it would be best to take advantage of the multiple cores available in modern processors to handle the program.