How Task-Based and Concurrent Algorithms are currently used to great affect within industry

Anthony Sturdy

Task-Based Software Engineering

Staffordshire University

Concurrent Programming is a method of developing an application that allows multiple tasks to be executed in parallel. True parallelism cannot be performed on a single core processor, the CPU will switch between tasks, but they won’t be processed concurrently unless they are being executed on a multi-core CPU (Clifton, 2008). The advantage of parallel processing is that more of the CPU cores are utilised at once meaning tasks are processed faster.



Although parallel processing can increase an application’s speed, it can also be equal or slower than sequential processing if done wrong. Sometimes the overhead of starting a task on a new thread can cause the task to execute slower than sequential processing. Another common problem with parallel programming is when race conditions occur, this can happen when two separate threads operate on the same object at the same time (Paul, 2017) which can lead to the application crashing (Rouse, 2019).

Most modern processors have multiple cores and multiple threads. According to the Steam March 2020 Hardware and Software Survey (Steam, 2020), only 0.41% of Steam’s user base use single core processors on Windows, and 1.32% on Linux. This means the utilisation of parallel processing is important to software developers as it could greatly increase the speed of applications over most devices. Parallel programming is most important when quick processing time is vital, for example, when working with big data. If the purpose of an application is to filter through or process 100,000,000 pieces of data, it could take a long time sequentially, whereas if it were processed concurrently it may be sped up by a considerable amount.

Another example where parallel programming is important is within video games. Modern video games often aim to render 60 frames per second, meaning everything in that frame needs to be processed in under 0.016 seconds. That’s all the input, AI, audio, networking, graphics, etc. in under 16 milliseconds. This would be a lot of work for just a CPU, but with a dedicated graphics card, graphics can be rendered a lot quicker. Graphics cards are designed to process in parallel (Hruska, 2018), which is why it’s able to render 2,073,600 pixels for a 1080p render in such a short amount of time. Snippets of code, called Shaders, are sent to the graphics card, along with data to be processed. When pixels are being rasterised (by what is known as a Fragment Shader), the code can’t access the data from neighbouring pixels due to it being ran parallel.

Plenty of industries use parallel programming (Gossett, 2019): Astronomers need to simulate objects in space due to how slowly events happen in space. Computers use parallel processing to quickly and accurately simulate these events; Banks use GPUs for things such as fraud detection to credit scoring. Crypto mining also caused a shortage of graphics cards in 2018 causing GPU prices; the medical industry uses GPUs for high computation and high bandwidth tasks which has led to better speed and accuracy for many aspects of the medical industry; these are just a few examples but there are many more instances where parallel programming can be useful.

Microsoft offer multiple tools for parallel programming, such as F#, the Task Parallel Library and LINQ/PLINQ (Clifton, 2008). F# itself isn’t a language for parallel programming, but it does support asynchronous workflows which “can be used to write applications that have UIs that remain responsive to users as the application performs other work” (Microsoft, 2016). The Task Parallel Library brings functionality to .NET languages which simplify a parallel programming workflow. Instead of the user having to create functionality to start new threads and tasks, TPL provides most of the functionality already. Developers can easily Parallelise a for loop by using Parallel.For instead of just a regular for. LINQ (Language-integrated Query) or the Parallel version, PLINQ, can be used to query data similarly to SQL. Parameters are set on the query, and any objects which meet the parameters will be returned in a new collection.