Using coroutines for reducing memory stalls in a bioinformatics tool

BWA (Burrows-Wheeler Aligner) is a very common tool for mapping genome reads to a reference genome. In genomic sequencing, short “reads” of tens to hundreds of base pairs (“letters”) are recorded by the sequencing machines. The process of mapping consists of putting those pieces together, based on the idea that most parts of the genome are identical, so the read is aligned to the most similar part of a known reference genome, in this case using the Burrows-Wheeler data structure.

However, especially for short reads, it can be shown that BWA will stall on memory accesses. That is not unsurprising; with an optimal data structure each access to the index can be essentially random. Common techniques to combat such memory stalls include loop unrolling, making sure that other useful work is available to the CPU while it is awaiting the results from a high-latency access to main memory. The index accesses in BWA occur within a separate function that is not amenable to unrolling.

BWA is written in C. From C++20, there is support for coroutines in the core language. Using coroutines, one can implement cooperative multitasking with limited code changes. Thus, the existing serial code for index accesses could be turned into coroutines, where several index accesses would be launched in parallel. To accomplish this, relevant parts of the code should be changed to C++ compilation and adapted for coroutines using the CppCoro library. Benchmarking should be done on extreme short read aDNA data (ancient DNA, archeological samples) to quantify what performance gains might be possible.

Contact: carl.nettelblad@it.uu.se