Rapid Parallel Genome Indexing with MapReduce (Summary)

The MapReduce parallel programming model is improved by accelerating the suffix array (SA) and the Burrows-Wheeler Transform (BWT) constructions of a DNA sequence for alignment purposes. The former index leads to rapid binary search algorithms for matching query sequences of any length while the latter one reduces the space requirements of the SA – from 12GB to 3GB – by recording as an index a (reversible) permutation of the string based on the ordering of the SA. The MapReduce algorithm uses the data processing capabilities of MapReduce to divide the suffix array construction into multiple independent ranges that are then independently solved (in parallel) in order to reduce computation time.

The suffix array is an index consisting of the lexicographically sorted list of all suffixes in a genome sequence. It enables fast, variable-length lookups for any substring in the reference genome thus accelerating sequence alignment computations. It also supports inexact alignment algorithms that allow for some differences between the reference and query sequences by using a seed-and-extend technique which finds relatively short exact matches using the suffix array to anchor the search for longer potentially in-exact matches. In practice the suffix array records only the list of suffix offsets for lookup as opposed to explicitly storing each suffix as otherwise the memory requirement would be intractable. The closely related BWT index structure reduces the space requirement more by using a permutation of the sequence as an index. It is the last column of the Burrows-Wheeler Matrix (BWM), a lexicographically sorted matrix of all of the cyclic permutations of the string.

MapReduce can distribute computation across a cluster with hundreds or thousands of computers, each analysing a portion of the dataset stored locally on the compute node. After an initial round of independent parallel computation, the machines efficiently exchange intermediate results, from which the final results are computed in parallel. The parallel computation is split in three phases – map, shuffle, and reduce; map scans the input dataset and emits key-value pairs representing some relevant information of the data tagged by the key, shuffle distributes and shuffles all values associated with a given key to collect them in a single list (a parallel distributed merge sort of the key-values pairs) while reduce processes these lists to compute the final results.

The MapReduce implementation of suffix array and BWT partitions the suffixes into non-overlapping batches with lexicographically similar values, and then sorts the batches on different machines across the cluster. The suffixes are independently assigned to partitions in parallel according to their prefix and the batches can be independently sorted.

In order to improve load balance among the reducers, a partitioner which samples the genome to select the boundaries of the batches based on the true sequence distribution was implemented. In order to improve the batch sorting performance, an optimized recursive bucket sort that pre-computes and determines problematic repeats on-the-fly was also implemented. Optimizing single and multiple character repeats accelerates the recursive sort. Also, by increasing the number of reducers, the number of suffixes per reducer is decreased along with the peak memory usage and the runtime is improved.

The algorithm was substantially limited by the Hadoop overhead (approximately half the runtime is used by it alone). There is an approximately linear relationship between the genome size and the runtime meaning the algorithm performs well in the presence of complicated repeats.