Towards a de novo short read assembler for large genomes using cloud computing

Michael C. Schatz (mschatz@umd.edu)

Center for Bioinformatics and Computational Biology, University of Maryland, College Park, MD, USA

Abstract

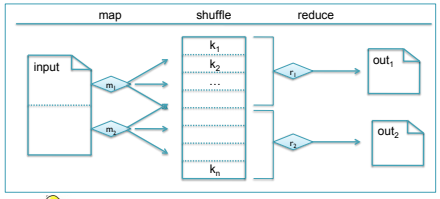
The massive volume of data and short read lengths from next generation DNA sequencing machines has spurred development of a new class of short read genome assemblers. Several of the new assemblers, such as Velvet and Euler-USR, model the assembly problem as constructing, simplifying, and traversing the de Bruijn graph of the read sequences, where nodes in the graph represent k-mers in the reads, with edges between nodes for consecutive k-mers. This approach has many advantages for these data, such as efficient computation of overlapping reads and robust handling of sequencing errors, and has demonstrated success for assembling small to moderately sized genomes. However, this approach is computationally challenging to scale to mammalian-sized genomes because it requires constructing and manipulating a graph far larger than can fit into memory.

MapReduce was developed at Google™ for parallel computation on their extremely large data sets, including their database of more than 1 trillion web pages. Computation in MapReduce is structured into 2 main phases: the map phase and the reduce phase, which act together to construct a large distributed hash table of key-value pairs in a map phase, and then evaluate a function on each bucket of the hash table in the reduce phase. The power of MapReduce is dozens or hundreds of map and reduce instances can execute in parallel, enabling efficient computation even on terabyte and petabyte sized data sets.

Drawing on the success of CloudBurst, a MapReduce-based short read mapping algorithm capable of mapping millions of reads to the human genome with high sensitivity, we have developed a MapReduce-based short read assembler that shows tremendous potential for enabling de novo assembly of mammalian-sized genomes. The deBruijn graph is constructed with MapReduce by emitting and then grouping key-value pairs (k_i,k_j) between successive k-mers in the read sequences. After construction, MapReduce is used again to remove spurious nodes and edges from the graph caused by sequencing error in the reads, and to compress simple chains of nodes into long sequence nodes representing the unambiguous regions of the genome between repeat boundaries. The resulting graph is a small fraction of the size of the original deBruijn graph, and is output in a format compatible with other short read assemblers for additional analysis.

Cloud Computing and MapReduce

Cloud computing is an emerging model for remote large-scale computing, where compute resources are accessed generically and rented as needed, especially to augment local resources for time critical or large computations. Several companies, including Amazon, Google, and Microsoft now offer tens of thousands of machines in their clouds. Machines are rented for as little as 10¢ per hour per machine, making it an attractive platform for large scale computation without the expense of purchasing or maintaining a large infrastructure.



http://hadoop.apache.org

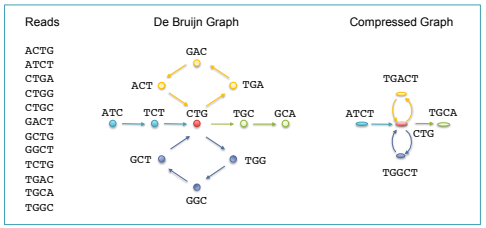
Programming multiple computers for very large data problems requires efficient methods to distribute work, monitor and restart tasks, and collect results. As such, Google invented MapReduce to automatically provide these common services within a very simple programming model. Application developers focus on just 2 functions called *map* and *reduce*, and the system efficiently scales them to large clusters. The *map* function emits key-value pairs representing local partial results from each input. The key-value pairs are then automatically shuffled so all values with the same key are collected into a single list. The *reduce* function then executes on each list to provide the final result. Conceptually, MapReduce constructs and analyzes a large distributed hash-table, and is thus applicable to many problems, including distributed search and sort, machine learning, and many graph algorithms. Hadoop is a leading open-source implementation of MapReduce, and is used on large production compute clouds, analyzing petabytes of data.

Large Genome Assembly with MapReduce

Recent studies of individual human genomes analyzed 3 - 4 billion short reads stored in 110+ GB of compressed sequence data. Consequently, *de novo* assembly of these data on a single machine with a short read assembler such as Velvet, EULER-USR, or ALLPATHS is not feasible. However, MapReduce enables the de Bruijn graph assembly algorithms to scale to these large datasets as outlined below.

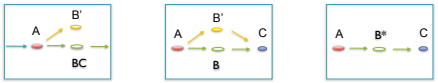
1. De Bruijn Graph Construction and Compression

Construction of the de Bruijn graph is naturally implemented in MapReduce. The map function emits key value pairs (k_i, k_{i+1}) for consecutive k-mers in the reads, which are then globally shuffled and reduced to build an adjacency list for all k-mers in the reads. Regions of the genome between repeat boundaries form non-branching simple paths of up to tens of thousands of nodes in the human genome. Compression of these paths is necessary to simplify the representation but adjacent nodes of the graph will be stored on physically separate machines. Nevertheless, MapReduce can efficiently compress simple chains of length S in $O(log(S))$ rounds using a parallel list ranking algorithm.



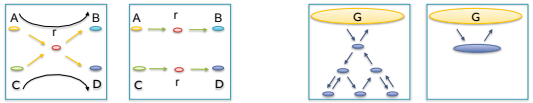
2. Error Correction

Errors in the reads distort the graph structure creating dead-ends (left) or bubbles (middle). These graph structures are recognized and resolved in a single MapReduce cycle creating additional simple paths (right). Bubbles with good support for both sequences are caused by heterozygote positions in the genome, and are tracked in the nodes (B*).



3. Graph contraction

Additional simplification techniques such as x-cut (left) and cycle tree compaction (right) further simplify the graph structure, and create more opportunities for simple path compression.



4. Scaffolding

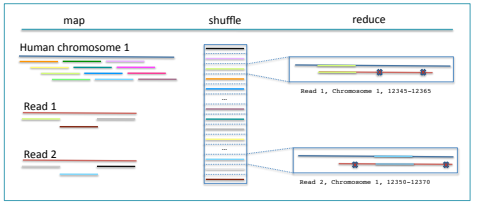
Finally mate-pairs, if available, are analyzed to further resolve ambiguities. MapReduce is used to identify the connected components of the assembly graph, which are then separately but concurrently analyzed using the scaffolding component of an assembler such as Velvet or the Celera Assembler. The final sequences resolves larger regions of the genome, revealing new biology not accessible through purely comparative techniques.

CloudBurst: Highly Sensitive Short Read Mapping

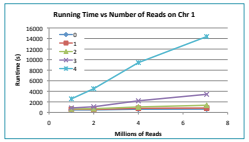
CloudBurst is a read mapping algorithm optimized for mapping next-generation sequence data to the human genome and other reference genomes, for use in a variety of biological analyses including SNP discovery, genotyping, and personal genomics. It is modeled after the short read mapping program RMAP, and reports either all alignments or the unambiguous best alignment for each read with any number of mismatches or differences.

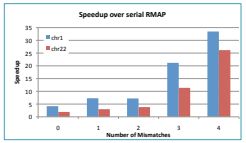
CloudBurst uses well-known seed-and-extend algorithms to map reads to a reference genome. It can map reads with any number of differences or mismatches using the observation that for an r bp read to align to the reference with at most k differences, the alignment must have a region of length $s=r/k+1$ called a seed that exactly matches the reference. Given an exact seed, CloudBurst attempts to extend the alignment into an end-to-end alignment with at most k mismatches or k differences by either counting mismatches between the two sequences, or with a dynamic programming algorithm to allow for gaps.

This level of sensitivity could be prohibitively time consuming, but CloudBurst uses Hadoop to parallelize execution across multiple compute nodes. In the map phase, the map function emits all length-s k-mers from the reference sequences, and all non-overlapping length-s kmers from the reads. In the shuffle phase, read and reference kmers are collected together. In the reduce phase, the seeds are extended into end-to-end alignments. The power of CloudBurst is the functions run in parallel over dozens or hundreds of computers.

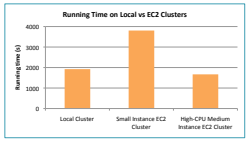


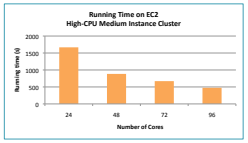
CloudBurst's running time scales linearly with the number of reads mapped, and with near linear speedup as the number of processors increases. In a 24-processor core configuration, CloudBurst is up to 30 times faster than RMAP executing on a single core, while computing an identical set of alignments.





In a large remote compute cloud with 96 cores at Amazon EC2, CloudBurst reduces the running time from hours to mere minutes for typical jobs involving mapping of millions of short reads to the human genome. CloudBurst is available open-source as a model for parallelizing other bioinformatics algorithms with MapReduce.





Schatz, MC (2009) CloudBurst: Highly Sensitive Read Mapping with MapReduce. *Bioinformatics*. http://cbcb.umd.edu/software/cloudburst