RC25439 (MRL1401-001) January 14, 2014
Computer Science

IBM Research Report

Resolving Ambiguity in Genome Assembly Using High Performance Computing

Mahtab Mirmomeni, Tom Conway
IBM Research Division
Level 5
204 Lygon Street
Carlton, VIC 3053
Australia

Matthias Reumann
IBM Research – Zurich
8803 Rüschlikon
Switzerland

Justin Zobel
The University of Melbourne

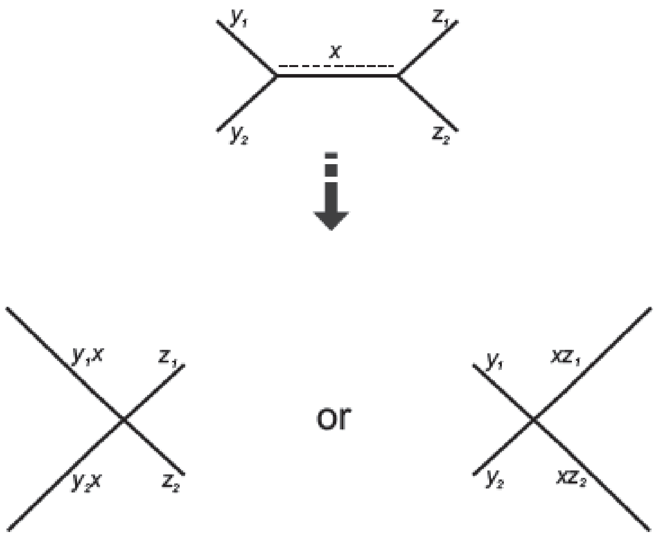
Resolving ambiguity in genome assembly using High Performance Computing

Summary:
DNA sequencing has revolutionised medicine and biology by providing insight into the nature of living organisms. High-throughput shotgun sequencing creates massive numbers of reads in a short period of time and *de novo* assembly attempts to reconstruct the original sequence, as closely as possible, using these reads. Longer pieces reconstructed by assemblies, shed more light on the underlying organism's biology. Repetitive sequences in the DNA, create ambiguities in the assembly which result in shorter fragments. In this project, we explore the search space of the assembly graph construction using the high performance computing capability of an IBM Blue Gene/Q and develop an algorithm that improves assembly quality through deeper search for valid longer sequences around repeat areas. Our results show that we can increase N50 of contigs by 4% and the number of contigs over 1000bp by up to 7%, however, this extension comes at the cost of using a great deal of computing power.

Introduction:
Genome sequencing has become an indispensable part of biomedical research. It enables researchers to understand the structure of genomes and the relationship between species. It also enables comparisons of individuals and analysis of variations in the population. High throughput sequencing technologies produce large numbers of reads from the DNA, in a short period of time. These reads need to be merged to rebuild the original DNA sequence. This process is called assembly; *de novo* assembly is a type of assembly where the reads are the only information used in this reconstruction. The programs that perform assembly are called assemblers. Many assemblers create a graph (such as the de Bruijn graph) from the reads and traverse the graph to output unambiguous fragments called contigs. To construct a de Bruijn graph for assembly, reads are first decomposed into *k*-mers, which are strings of *k* nucleotides where *k* is a positive integer, which would become the nodes of the graph. Two nodes are connected together if the suffix of the source node shares an exact match of length *k - 1* with the prefix of the destination node. Each read induces a path in the graph and the process of assembly will be mapped to finding an Eulerian path in the graph.

Assemblers have limitations when dealing with repetitive regions of DNA, since it is not be clear how many copies of a repetitive sequence exist and how the graph should be traversed. In ambiguous situations, the assemblers rely on heuristics or break the contigs at points of uncertainty. In this project, we explore the opportunity of extending the length of contigs by further searching in the assembly graph for valid sequences to produce longer, more meaningful assemblies.

Description:
In order to investigate the effect of an extensive search in the assembly graph around repetitive regions, on the length of the contigs reported by the assembler, we studied the assembly graph produced by Gossamer [1] using the human genome reads published by Gnerre el al. [2]. The assembly graph in Gossamer, called the supergraph, is the result of identifying the Eulerian paths in the de Bruijn graph. The edges of the supergraph are the contigs, which are the output of our assembly. Repetitive elements, create complexities: for example, a repeated sequence presented by contig *x* can be preceded by contigs *y1* and *y2* and followed by contigs *z1* and *z2*. These situations are called *tangles*. When Gossamer encounters a tangle, it outputs *x, y1, y2, z1,* and *z2* separately. Given that *x* can be followed with either *z1* or *z2,* both *x.z1* (*x* followed by *z1*) and *x.z2* are correct sequences in the underlying genome. Similarly *y1.x* and *y2.x* are both correct sequences. Reporting all four combinations can result in over-estimating the number of instances of *x* in the genome. We thus '*expand' x,* either from left and report *y1.x*, *y2.x, z1*, and *z2* as contigs or from right and report *y1, y2, x.z1* and *x.z2*.



Given that the assembly supergraph in our human Gnerre dataset contains over 86 million contigs, we estimate that the amount of memory required for our Gnerre dataset is over 87GB. In addition, over 13 million tangles have to be expanded. To tackle this problem, we have divided the supergraph into smaller partitions and used high performance computing (HPC) to process each partition in parallel, in a reasonable amount of time. The cost of an exhaustive search in the supergraph to expand all tangles is exponential, and therefore requires an infeasible amount of computing power. Thus, instead of an exhaustive search to find the best set of tangle expansions in a partition of the supergraph, we have implemented a heuristic search, randomly expanding the tangles in that partition a number of rounds and recording the lengths of the produced contigs. Our algorithm ran on 512 CPUs for 50 hours. Our results show that it is possible to create longer contigs, however, we used around 8 times additional computing power to the assembly algorithm, to gain this improvement.

Conclusion:

In this project, we explored the possibility of producing longer, more meaningful contigs by extending contigs around repeat regions instead of breaking them into separate contigs. The repeat regions create complex structures in our assembly supergraph called tangles. We used the structure of the graph and searched more deeply in the assembly supergraph produced by Gossamer [1] to find the best set of expansions for the tangles. Because of the size of the Gnerre dataset, we had to partition it's supergraph and use high performance computing to process different parts of the graph concurrently.

References:

[1] Conway, T., Wazny, J., Bromage, A., Zobel, J. and Beresford-Smith, B. Gossamer -- a resource-efficient de novo assembler. *Bioinformatics (Oxford, England)*, 28, 14 2012), 1937-1938.

[2] Gnerre, S., MacCallum, I., Przybylski, D., Ribeiro, F. J., Burton, J. N., Walker, B. J., Sharpe, T., Hall, G., Shea, T. P., Sykes, S., Berlin, A. M., Aird, D., Costello, M., Daza, R., Williams, L., Nicol, R., Gnirke, A., Nusbaum, C., Lander, E. S. and Jaffe, D. B. High-quality draft assemblies of mammalian genomes from massively parallel sequence data.