Commodity Computing in Genomics Research

Michael Schatz and Mihai Pop

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

http://www.cbcb.umd.edu/research/cloud/

Abstract

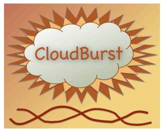
Recent advances in DNA sequencing technology from Illumina, 454 Life Sciences, ABI, and Helicos, are enabling next generation sequencing instruments to sequence the equivalent of the human genome (~3 billion bp) in few days and at low cost. In contrast, the sequencing for the human genome project of the late 90's and early '00s required years of work on hundreds of machines with sequencing costs measured in hundreds of millions of dollars. This dramatic increase in efficiency has spurred tremendous growth in applications for DNA sequencing.

For example, whereas the human genome project sought to sequence the genome of a small group of individuals, the 1000 genomes project aims to catalog the genomes of 1000 individuals from all regions of the globe in just three years. Related projects aim to catalog all of the biologically active transcribed regions of the genome over a wide variety of environmental and disease conditions. Similar studies are also underway for model organisms such as mouse, rat, chicken, rice, and yeast, and other organisms of interest.

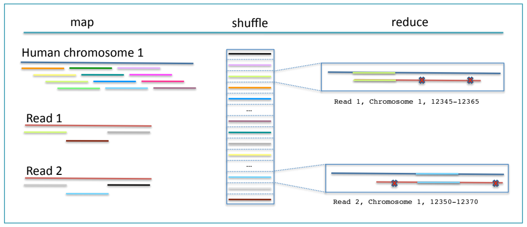
Cheap and fast sequencing technologies are also providing scientists with the tools to analyze the largely unknown microbial biosphere. The majority of microbes inhabiting our world and our bodies are unknown and cannot be easily manipulated in the laboratory. In recent years a new scientific field has emerged - metagenomics - that aims to characterize entire microbial communities by sequencing the DNA directly extracted from an environment. Several studies have already targeted a range of natural environments (ocean, soil, mine drainage) as well as the commensal microbes inhabiting the bodies of humans and other animals and insects. The latter are the target of a new NIH initiative – the Human Microbiome Project - an effort to characterize the diversity of human-associated microbial communities and to understand their contributions to human health.

The raw data generated by the new sequencing instruments often exceed 1 terabyte and are straining the computational infrastructure typically available in an average research lab. Furthermore, biological datasets are only increasing in size, as data for more individuals and more environments are collected, further complicating computational analyses. Even seemingly simple tasks, such as mapping a collection of sequencing reads to one of the human reference genome, can require days of computation, and *de novo* assembly of an entire human genome using new generation sequence data has only been possible with specialized compute resources. The only long-term solution to the challenges posed by the massive biological data-sets being generated is to combine computational biology research with advances from high performance computing. Here we explore the use of commodity computing within a cloud computing paradigm to tackle these problems.

CloudBurst: Highly Sensitive Short Read Mapping



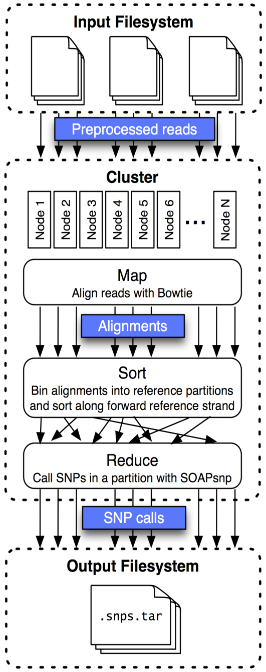
CloudBurst is a new open-source read-mapping algorithm, for use in a variety of biological analyses including SNP discovery, genotyping, and personal genomics. It is modeled after the short read mapping program RMAP, but uses Hadoop to compute alignments in parallel. CloudBurst's running time scales linearly with the number of reads mapped, and with near linear speedup as the number of processors increases. In a large remote compute cloud with 96 cores, CloudBurst achieves over a 100-fold speedup over a serial execution of RMAP, reducing the running time from hours to mere minutes to map millions of short reads to the human genome.


map
shuffle
reduce
Human chromosome 1
Read 1
Read 2
...
...
Read 1, Chromosome 1, 12345-12365
Read 2, Chromosome 1, 12350-12370

Schatz, MC (2009) **CloudBurst: Highly Sensitive Read Mapping with MapReduce**. *Bioinformatics.* 25(11):1363-1369. http://www.cbcb.umd.edu/software/cloudburst

Crossbow: Searching for SNPs with Cloud Computing

Crossbow is a cloud-computing software system that combines the speed of the short read aligner *Bowtie* and the accuracy of the *SOAPsnp* consensus and SNP caller. Executing these tools in parallel with the Hadoop implementation of MapReduce, Crossbow aligns reads and makes SNP calls with >99% accuracy from a dataset comprising 38-fold coverage of the human genome in less than one day on a cluster of 40 computer cores, and in less than three hours using a 320-core cluster rented from a commercial cloud computing service. Crossbow's ability to run in the clouds means that users need not own or operate an expensive computer cluster to run Crossbow.



The user first uploads reads to a filesystem visible to the Hadoop cluster. If the Hadoop cluster is in EC2, the filesystem might be an S3 bucket. If the Hadoop cluster is local, the filesystem might be an NFS share.

A cluster may consist of any number of nodes. *Hadoop handles the details of routing data, distributing and invoking programs, providing fault tolerance, etc.*

Map step is short read alignment. Many instances of **Bowtie** run in parallel across the cluster. Input tuples are reads and output tuples are alignments.

Sort step bins alignments according to primary key (genome chromosome) and sorts according to a secondary key (offset into chromosome). *This is handled efficiently by Hadoop.*

Reduce step calls SNPs for each reference partition. Many instances of **SOAPsnp** run in parallel across the cluster. Input tuples are sorted alignments for a partition and output tuples are SNP calls.

Simulated	Data	Crossbow sensitivity	Crossbow precision	Nodes	Time / Cost
Human Chr. 22	40 M reads 1.8 GB	99.01%	99.14%	1 master + 2 worker	0h 30m $2.40
Human Chr. X	172 M reads 5.6 GB	98.97%	99.64%	1 master + 8 workers	0h 26m $7.20
Genuine	**Data**	**Autosomal agreement**	**Chr. X agreement**	**Nodes**	**Time / Cost**
Whole human, versus Illumina 1M BeadChip	2.7 B reads 103 GB	99.5%	99.6%	1 master + 40 workers	2h 53m $98.40

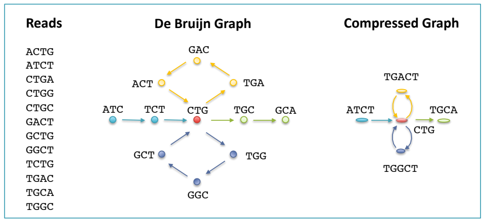
Langmead, B, Schatz, MC, Lin, J, Pop, M, Salzberg, SL (2009) **Searching for SNPs with Cloud Computing**. *Submitted for publication.* http://bowtie-bio.sf.net/crossbow

Genome Assembly with MapReduce

De *novo* assembly of human-sized genomes from short reads on a single machine is not feasible with a current assembler such as Velvet, EULER-USR, or ALLPATHS, as those algorithms would require >10TB of RAM to execute. However, MapReduce enables the de Bruijn graph assembly algorithms to scale to these large datasets as outlined below.

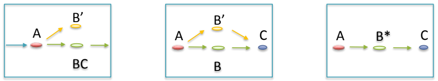
1. De Bruijn Graph Construction and Compression

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, and are efficiently compressed in O(*log(S)*) MapReduce rounds using a parallel list ranking algorithm.



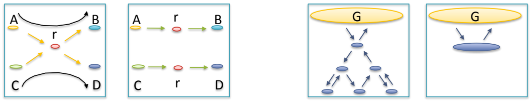
2. Error Correction

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).



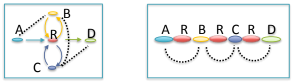
3. Graph contraction

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.



Acknowledgements

We would like to thank UMD students and faculty Ben Langmead, Jimmy Lin, Steven Salzberg, and Dan Sommer for their contributions.

This work is supported by the NSF Cluster Exploratory (CluE) program grant IIS-0844494.