BIOINFORMATICS APPLICATIONS NOTE Vol. 27 no. 5 2011, pages 718–719
doi:10.1093/bioinformatics/btq671

Sequence analysis Advance Access publication January 5, 2011

Tabix: fast retrieval of sequence features from generic TAB-delimited files

Heng Li

Program in Medical Population Genetics, The Broad Institute of Harvard and MIT, Cambridge, MA 02142, USA

Associate Editor: Dmitrij Frishman

ABSTRACT

Summary: Tabix is the first generic tool that indexes position sorted files in TAB-delimited formats such as GFF, BED, PSL, SAM and SQL export, and quickly retrieves features overlapping specified regions. Tabix features include few seek function calls per query, data compression with gzip compatibility and direct FTP/HTTP access. Tabix is implemented as a free command-line tool as well as a library in C, Java, Perl and Python. It is particularly useful for manually examining local genomic features on the command line and enables genome viewers to support huge data files and remote custom tracks over networks.

Availability and Implementation: http://samtools.sourceforge.net.

Contact: hengli@broadinstitute.org

Received on October 14, 2010; revised and accepted on December 1, 2010

1 INTRODUCTION

When we examine local genomic features on the command line or via a genome viewer, we frequently need to perform *interval queries*, retrieving features overlapping specified regions. We may read through the entire data file if we only perform interval queries a few times, or preload the file in memory if interval queries are frequently performed (e.g. by genome viewers). However, reading the entire file makes both strategies inefficient given huge datasets. A solution to the efficiency problem is to build a database for the data file (Kent *et al*., 2002; Stein *et al*., 2002), but this is not optimal, either, because generic database indexing algorithms are not specialized for biological data (Alekseyenko and Lee, 2007). The technical complexity of setting up databases and designing schemas also hampers the adoption of the database approach for an average end user.

Under the circumstances, a few specialized binary formats including bigBed/bigWig (Kent *et al*., 2010) and BAM (Li *et al*., 2009) have been developed very recently to achieve efficient random access to huge datasets while supporting data compression and remote file access, which greatly helps routine data processing and data visualization. At present, these advanced indexing techniques are only applied to BED, Wiggle and BAM. Nonetheless, as most TAB-delimited biological data formats (e.g. PSL, GFF, SAM, VCF and many UCSC database dumps) contain chromosomal positions, one can imagine that a generic tool that indexes for all these formats is feasible. And this tool is Tabix. In this article, I will explain the technical advances in Tabix indexing, describe the algorithm and evaluate its performance on biological data.

2 METHODS

Tabix indexing is a generalization of BAM indexing for generic TAB-delimited files. It inherits all the advantages of BAM indexing, including data compression and efficient random access in terms of few seek function calls per query.

2.1 Sorting and BGZF compression

Before being indexed, the data file needs to be sorted first by sequence name and then by leftmost coordinate, which can be done with the standard Unix `sort`. The sorted file should then be compressed with the `bgzip` program that comes with the Tabix package. `Bgzip` compresses the data file in the BGZF format, which is the concatenation of a series of gzip blocks with each block holding at most 2^{16} bytes of uncompressed data. In the compressed file, each uncompressed byte in the text data file is assigned a unique 64-bit *virtual file offset* where the higher 48 bits keep the real file offset of the start of the gzip block the byte falls in, and the lower 16 bits store the offset of the byte inside the gzip block. Given a virtual file offset, one can directly seek to the start of the gzip block using the higher 48 bits, decompress the block and retrieve the byte pointed by the lower 16 bits of the virtual offset. Random access can thus be achieved without the help of additional index structures. As `gzip` works with concatenated gzip files, it can also seamlessly decompress a BGZF file. The detailed description of the BGZF format is described in the SAM specification.

2.2 Coupled binning and linear indices

Tabix builds two types of indices for a data file: a binning index and a linear index. We can actually achieve fast retrieval with only one of them. However, using the binning index alone may incur many unnecessary seek calls, while using the linear index alone has bad worst-case performance (when some records span very long distances). Using them together avoids their weakness.

2.2.1 The binning index The basic idea of binning is to cluster records into large intervals, called *bins*. A record is assigned to bin k if k is the bin of the smallest size that fully contains the record. For each bin, we keep in the index file the virtual file offsets of all records assigned to the bin. When we search for records overlapping a query interval, we first collect bins overlapping the interval and then test each record in the collected bins for overlaps.

In principle, bins can be selected freely as long as each record can be assigned to a bin. In the Tabix binning index, we adopt a multilevel binning scheme where bins at same level are non-overlapping and of the same size. In Tabix, each bin k, $0 \le k \le 37\,449$, represents a half-close-half-open interval $[(k-o_l)s_l, (k-o_l+1)s_l)$, where $l = \lfloor \log_2(7k+1)/3 \rfloor$ is the *level* of the bin, $s_l = 2^{29-3l}$ is the size of the bin at level l and $o_l = (2^{3l}-1)/7$ is the offset at l. In this scheme, bin 0 spans 512 Mb, 1–8 span 64 Mb, 9–72 8 Mb, 73–584 1 Mb, 585–4680 128 kb and 4681–37449 span 16 kb intervals. The scheme is very similar to the UCSC binning (Kent *et al*., 2002) except that in UCSC, $0 \le k \le 4681$ and therefore the smallest bin size is 128 kb.

Table 1. Performance of Tabix

	EST[a]	Short reads[b]
Original file size (GB)	1.23	50.00
`Bgzip` compressed file size (GB)	0.35	11.20
Index file size (MB)	1.39	0.36
Indexing time (CPU in s)	15.34	509.72
Time on 1000 queries[c] (CPU s)	0.71	17.97
Bytes read from disk (MB)[d]	30.41	446.26
Data retrieved in plain text (MB)	3.50	113.24
No. of lseek calls per query[d]	1.06	1.01
No. of lseek calls per query without linear index[d]	6.15	13.17

[a]All human EST alignments from UCSC hg19 (Rhead *et al*., 2010), sorted by the Unix `sort` utility.
[b]NA18507 chr1 40X short-read alignment from Bentley *et al*. (2008).
[c]1000 random intervals ranged from 1–1000 bp; index loaded once.
[d]Measured by the `strace` Linux command.

When the data files are sorted by coordinates, records assigned to the same bin tend to be adjacent. Thus in the index file, we do not need to keep the virtual file offset of each record, but only to keep the start offset of a chunk of records assigned to the same bin.

2.2.2 The linear index With the binning index alone, we frequently need to seek to records assigned to bins at lower levels, in particular bin 0, the bin spanning the entire sequence. However, frequently records at lower levels do not overlap the query interval, which leads to unsuccessful seeks and hurts performance especially when data are transferred over network. The linear index is to reduce unnecessary seek calls in this case (Table 1).

In the linear index, we keep for each tiling 16 kb window the virtual file offset of the leftmost record (i.e. having the smallest start coordinate) that overlaps the window. When we search for records overlapping a query interval, we will know from the index the leftmost record that possibly overlaps the query interval. Records having smaller coordinates than this leftmost record can be skipped and unsuccessful seek calls can be saved.

3 RESULTS

The Tabix algorithm is implemented in C and Java with Perl and Python bindings to the C library. When the input file name is a URL starting with 'http://' or 'ftp://', Tabix will retrieve data directly from the remote file. The Tabix package is distributed under the MIT/X11 license, free to both academic and commercial uses.

To evaluate the performance of Tabix, I applied it to two datasets: all human EST alignments and 40X short-read alignments (Table 1). On both datasets, the `bgzip` compression reduces file sizes by a factor of 3–5 and Tabix only invokes one seek function call for each interval query. Tabix is slower on the short-read alignments because each 16 kb interval contains more data. Nonetheless, Tabix is still fast enough for a genome viewer. Data retrieval, especially retrieval over network, is I/O bounded rather than CPU bounded. Invoking fewer seek calls and reading fewer data from disk/network have a bigger impact on the practical performance.

4 DISCUSSIONS

Tabix is a stand-alone, lightweight (compact disk space and low memory usage) and efficient (few seek function calls) software package to achieve random access to a generic TAB-delimited file located locally or on a remote FTP/HTTP site. It is a valuable tool for researchers who frequently perform interval queries, for graphical viewer developers who want to display large data files in limited memory, and for genome database developers who intend to support large remote custom tracks.

Another closely related file format is bigBed (Kent *et al*., 2010). Both Tabix and bigBed support data compression and direct FTP/HTTP access. While bigBed further has the standing advantages of HTTPS support, explicit caching, integrated index and data file and partial index loading, Tabix works with many more text formats. It provides the first generic solution for common TAB-delimited format. Tabix also invokes fewer seek function calls partly as a result of loading the full index into memory, though at the cost of larger memory footprint. On the BED file converted from human EST alignments (Table 1), the Bio::DB::BigBed Perl module requires 4.19 seeks per interval query, as is opposed to 1.04 seeks per query by the Tabix Perl module. In addition, another advantage of Tabix is it does not change the text format and the compression is gzip compatible. It may potentially turn a FTP/HTTP file server into a minimal database. The 1000 genomes project (http://1000genomes.org) is providing SNPs in the Tabix indexed Variant Call Format.

ACKNOWLEDGEMENTS

I thank Bob Handsaker for the proposal of adding the linear index and the initial implementation of the C BGZF library, Petr Danecek for various bug fixes, Lincoln Stein for the suggestion of direct FTP/HTTP access and Jim Kent, James Bonfield and Richard Durbin for their helpful discussions on general indexing techniques.

Funding: NIH 1U01HG005208-01.

Conflict of Interest: none declared.

REFERENCES

Alekseyenko,A.V. and Lee,C.J. (2007) Nested containment list (NCList): a new algorithm for accelerating interval query of genome alignment and interval databases. *Bioinformatics*, **23**, 1386–1393.

Bentley,D.R. *et al*. (2008) Accurate whole human genome sequencing using reversible terminator chemistry. *Nature*, **456**, 53–59.

Kent,W.J. *et al*. (2002) The human genome browser at UCSC. *Genome Res.*, **12**, 996–1006.

Kent,W.J. *et al*. (2010) BigWig and BigBed: enabling browsing of large distributed datasets. *Bioinformatics*, **26**, 2204–2207.

Li,H. *et al*. (2009) The sequence alignment/map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Rhead,B. *et al*. (2010) The UCSC Genome Browser database: update 2010. *Nucleic Acids Res.*, **38**, D613–D619.

Stein,L.D. *et al*. (2002) The generic genome browser: a building block for a model organism system database. *Genome Res.*, **12**, 1599–1610.