

SOFTWARE

Open Access

ReadDB Provides Efficient Storage for Mapped Short Reads

P Alexander Rolfe^{1,2} and David K Gifford^{1*}

Abstract

Background: The advent of high-throughput sequencing has enabled sequencing based measurements of cellular function, with an individual measurement potentially consisting of more than 10^8 reads. While tools are available for aligning sets of reads to genomes and interpreting the results, fewer tools have been developed to address the storage and retrieval requirements of large collections of aligned datasets. We present ReadDB, a network accessible column store database system for aligned high-throughput read datasets.

Results: ReadDB stores collections of aligned read positions and provides a client interface to support visualization and analysis. ReadDB is implemented as a network server that responds to queries on genomic intervals in an experiment with either the set of contained reads or a histogram based interval summary. Tests on datasets ranging from 10^5 to 10^8 reads demonstrate that ReadDB performance is generally within a factor of two of local-storage based methods and often three to five times better than other network-based methods.

Conclusions: ReadDB is a high-performance foundation for ChIP-Seq and RNA-Seq analysis. The client-server model provides convenient access to compute cluster nodes or desktop visualization software without requiring a shared network filesystem or large amounts of local storage. The client code provides a simple interface for fast data access to visualization or analysis. ReadDB provides a new way to store genome-aligned reads for use in applications where read sequence and alignment mismatches are not needed.

Background

Next generation DNA sequencing technology [1] has enabled the use of sequencing to query biological function using methods such as ChIP Seq [2] and RNA Seq [3]. The cost of sequencing is rapidly declining, and as a consequence large repositories of sequencing data have arisen from key biological experiments. Early experiments used 25 bp tags for ChIP-Seq and generated a few million reads per sample. Recently available long reads, paired reads, and read counts over 10^8 reads per sample have enabled RNA-Seq [3,4] experiments that detect splice form variants. Deep sequencing also permits detection of SNPs and other variants across populations and between samples and reference sequences.

The number of public, potentially useful datasets is vast. For example, the ENCODE project ([http://www.](http://www.genome.gov/10005107)

[genome.gov/10005107](http://www.genome.gov/10005107), [5]) has produced hundreds sequencing datasets in human cell lines for histone modifications and transcription factor binding. Analyses incorporating these datasets might query the genome in kilobase windows and perform millions of queries against each dataset in an analysis. The full alignment output for a dataset might be hundreds of megabytes to several gigabytes (see Table 1), requiring over a terabyte of space for the complete ENCODE datasets.

The data requirements for analysis algorithms vary by application. Assembly and SNP detection algorithms require access to all bases of the read and the quality score of each base. In contrast, some ChIP-Seq analysis algorithms can ignore individual reads and operate only on the histogram of read depth at each base. Common formats currently include BAM [6] and WIG/BigWig [7]. BAM stores the full alignment output, including the read sequence, read quality, hit position, and mismatch and indel information. WIG (and its binary version BigWig) is a histogram format that stores positions and values; each position corresponds to a single histogram

* Correspondence: dkg@mit.edu

¹Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA

Full list of author information is available at the end of the article

Table 1 Storage space required for each format

	small	medium	large
read set size	536 k	13.9 m	175 m
BigWig	989 kB	23 MB	83 MB
ReadDB	3.3 MB	126 MB	463 MB
BAM	25 MB	532 MB	5.1 GB

Space used by ReadDB, BigWig files, and BAM files (including the index files for each format).

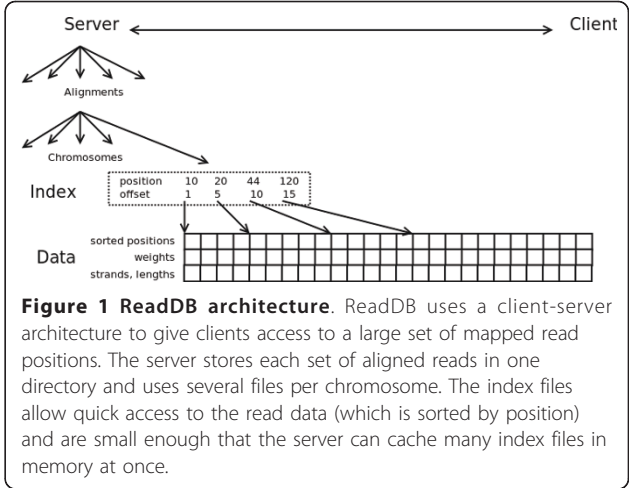
bin and the value to the number of reads starting in or crossing that bin.

Our software, ReadDB, aims to support ChIP-Seq, RNA-Seq, DNA methylation, and DNase hypersensitivity analysis and visualization applications by providing efficient access to mapped read positions for single and paired end reads. ReadDB operates as a network server process such that data can be curated by one or a few people for a large group and so that clients do not need access to a shared network filesystem or large amounts of local disk space. The server can respond to queries either with the positions of individual reads or with a histogram generated with arbitrary bin size. The provided client code can then provide quick network access to thousands of datasets. However, ReadDB does not implement analysis algorithms (eg, ChIP-Seq peak calling or RNA-Seq transcript detection) or a visualizer itself. Rather, we use and expect others to use ReadDB as the backend storage for these applications.

Implementation
Architecture

ReadDB implements an abstraction called an *alignment* that describes where reads from a read set map to a reference genome. A *read set* may be a collection of reads from a single experiment, or it may combine reads from multiple experiments that are replicates. Queries on an alignment are performed with respect to the coordinate space of the reference genome used for the alignment. Note that a read set can be aligned to different genomes, or different alignment methods or parameters can be used to align a read set to the same genome; each alignment variant for a given read set is stored as a unique alignment.

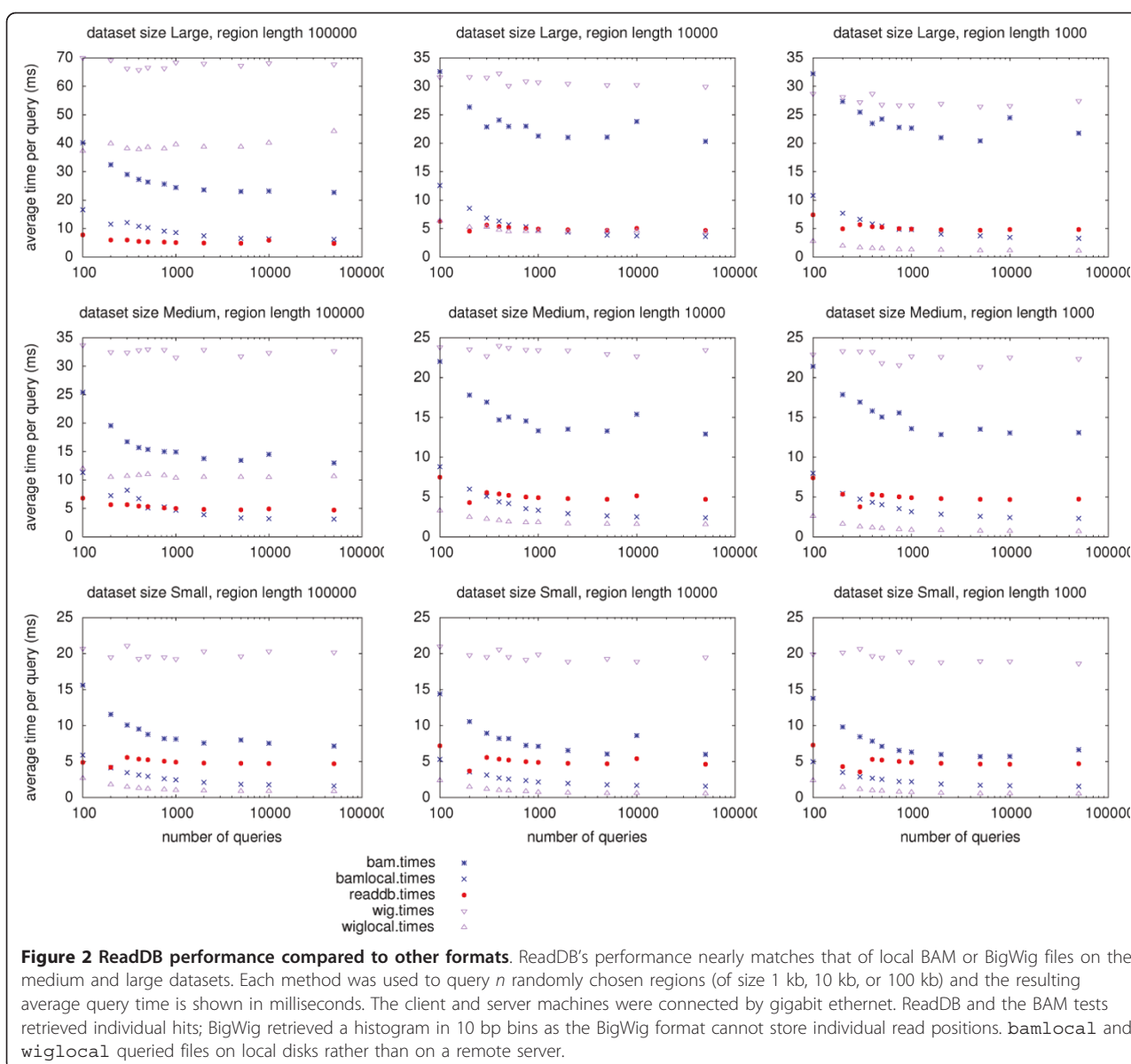
An alignment has distinct data structures for each chromosome. Three files make up the column store and hold the hits (a hit is a genomic position to which a read is mapped) sorted by position (5' coordinate of the aligned read, a 32 bit int), the strand and hit length (one bit strand, 15 bit hit length), and weight (32 bit float, typically the inverse of the number of alignments for the read). A fourth file indexes the sorted hits. Figure 1 provides an overview of the ReadDB architecture.



ReadDB’s key feature is the index of the sorted hits for a chromosome. The index file is similar to a single block of a B-tree index and points into the main data files every 4-16 kb (disk reads smaller than this won’t be significantly faster so a denser index doesn’t yield a performance improvement). At four bytes per record in the data file and eight bytes per entry in the index, a 64 kb index file is sufficient for eight million $((4 \text{ kb}/4) * (64 \text{ kb}/8))$ to thirty two million records. As most experiments contain fewer than ten million hits per chromosome, most index files are smaller than 64 kb and ReadDB can cache hundreds of index files in a reasonable amount of memory. The index provides $O(\log(n))$ accesses into the list of hits, meaning that ReadDB query operations are $O(\log(n) + m)$ where n is the number of hits stored and m is the number of hits returned.

For compactness, ReadDB stores chromosomes as integers. Client code may either use this directly to store chromosome numbers (with some alternate scheme to handle X, Y, mt, 5_random, etc) or may use an external database that maps chromosomes to numeric identifiers. In the former case, the alignment name ought to indicate the genome assembly, eg “GM12878 CTCF against mm8.” In the latter case, a system such as GSE [8] maps the combined genome build and chromosome name to an identifier and maintains metadata about each alignment. In either case, client code can unambiguously specify which genome should be used and could even query both genomes by executing queries against both chromosome identifiers.

While ReadDB can return individual read positions, many applications benefit from server generated histograms. Condensing a potentially large number of hits into a much smaller number of bin positions and counts



generates substantially less network traffic between the client and server. The server can also answer aggregate queries such as the total number of reads or the sum of hit weights in a region, chromosome, or dataset.

ReadDB also supports paired-end read sets and allows queries based on either end of the read. The query specifies, for example, a genomic range for which to query the "left" reads and all read pairs for which the left read maps to that range are returned. Paired-end storage is implemented by adding columns for the mate's position, strand, and length and by storing each read twice: once keyed by the "left" read and once keyed by the "right" read. The duplication allows for quick access by either side at a minimal cost of storage space since the data

stored per read is only slightly larger than the corresponding key (chromosome, position) and pointer.

Interface

The ReadDB server accepts queries as text and may be queried from any programming language. The interface, described in full in additional file 1, includes methods to

- store single-end and paired-end hits to an alignment. Both hit types may be stored to an alignment
- delete hits from an alignment (single or paired-end)
- retrieve a list of chromosomes to which reads were mapped in an alignment

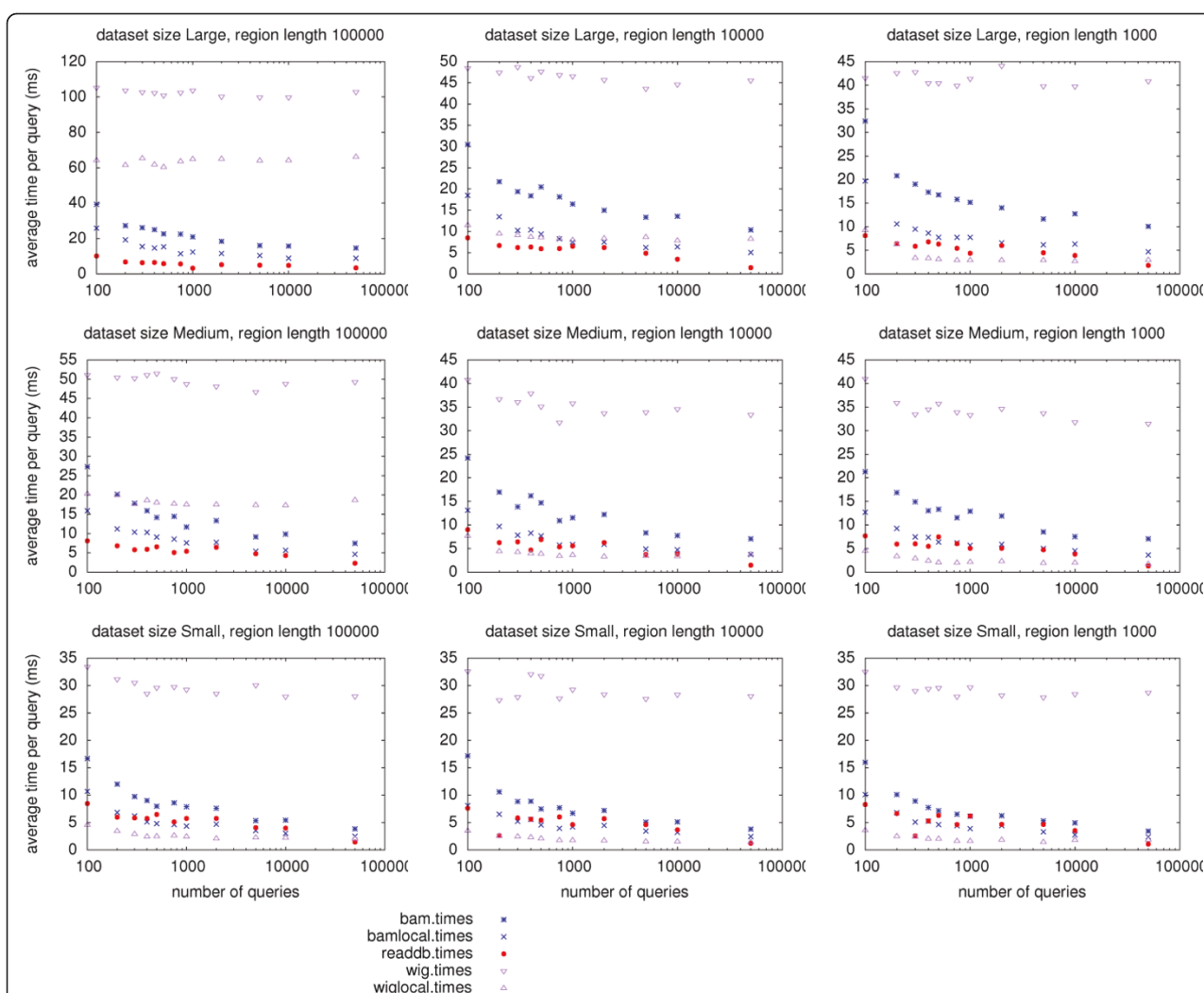


Figure 3 Performance comparison with no network latency. Comparison of runtimes with the client software and server on the same machine. To separate the effects of network throughput and latency from those of the disk and CPU, we ran the test suite from the server machine. The ReadDB test still went through the ReadDB server and the BAM and WIG files were read by the httpd process. These tests indicate that ReadDB's fast queries compared to remote BAM are not the result of BAM queries passing more information across the network; the ReadDB queries through the ReadDB server are faster than either BAM through httpd or BAM read directly from local disk.

- retrieve the number of hits present in an alignment or in a particular chromosomal region of an alignment
- retrieve the hits present in an alignment or in a particular chromosomal region of an alignment. The interface allows only positions or weights to be retrieved in addition to the full hit information.
- retrieve a histogram either of hit counts or hit weight sums across a chromosomal region. The interface allows the caller to specify the histogram bin width.

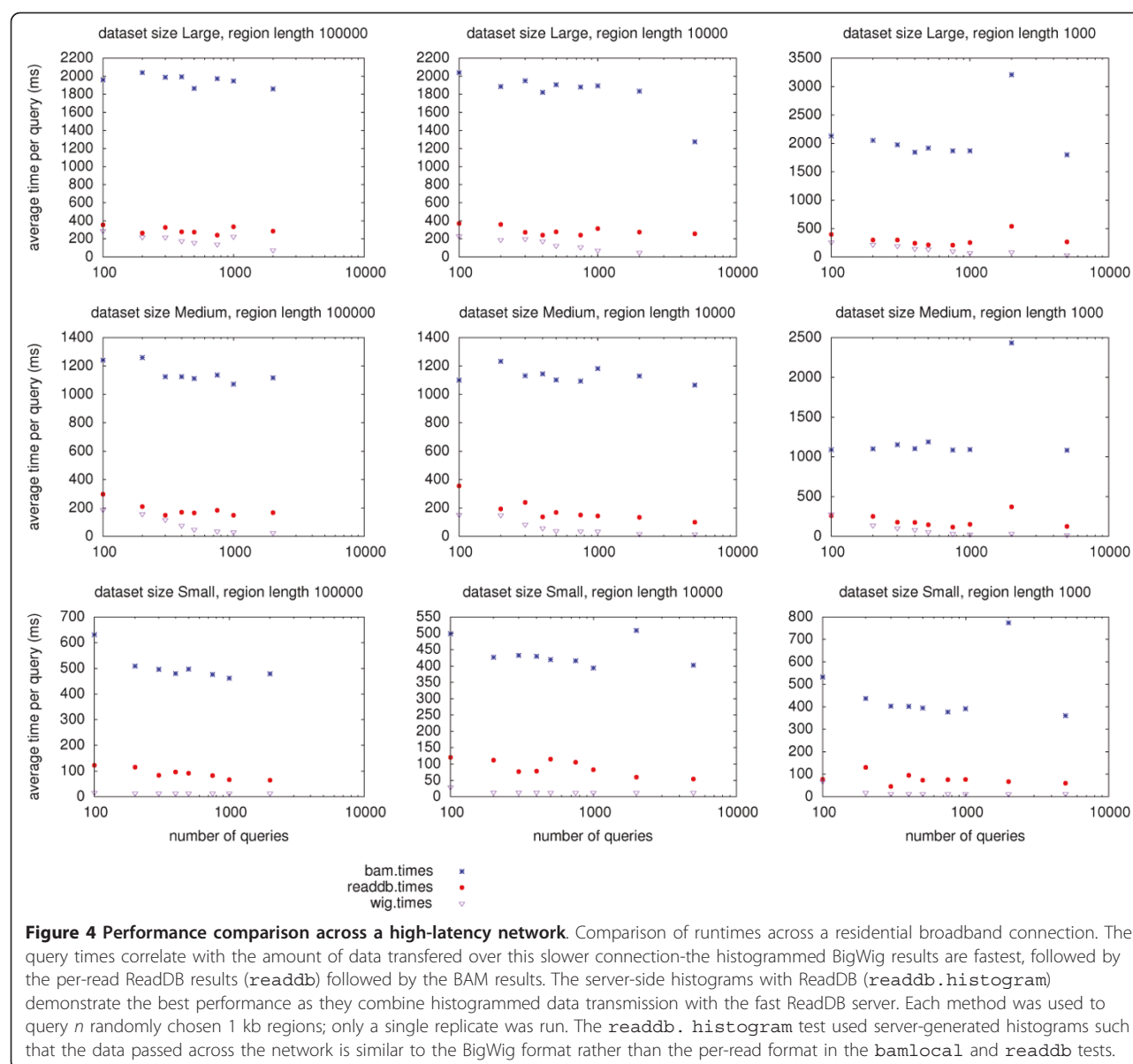
All queries may be filtered by hit strand, paired-ness, and weight.

Results

We tested a Java ReadDB client with a remote ReadDB server against the following alternatives:

- Remote BAM (HTTP access) and remote BAM index (HTTP access)
- BAM and index on local disk
- Remote BigWig (HTTP access)
- BigWig on local disk

ReadDB and BAM files were queried to retrieve individual read positions. BigWig files only support queries for histograms and were tested in this mode.



For each setup, we tested results from aligning three read files to the hg19 assembly: a small set of 513,870 Pol2 ChIP-Seq reads [9], a medium set of 13,901,600 CTCF ChIP-Seq reads (ENCODE project, Crawford Lab at Duke University, <ftp://hgdownload.cse.ucsc.edu/goldenPath/hg18/encodeDCC/wgEncodeChromatinMap/wgEncodeUtaChIPseqRawDataRep3K562Ctcf.fastq.gz>), and a large set of 175,128,655 reads DNase hypersensitivity reads (ENCODE project, Stam/Uw Lab at the University of Washington, <http://hgdownload.cse.ucsc.edu/goldenPath/hg18/encodeDCC/wgEncodeUwDnaseSeq/wgEncodeUwDnaseSeqRawDataRep2Gm12878.fastq.gz>). Table 1 shows the number of reads in each dataset as well as the storage space required for each method.

Each test consisted of retrieving all hit positions within some number of regions n (ranging from $n = 10$ to $n = 100,000$) of size 1 kb, 10 kb, or 100 kb. ReadDB was queried with our Java code. BAM files were queried with code using the Picard <http://picard.sourceforge.net> library and BigWig files were queried with Perl code using the Bio::BigFile module (<http://search.cpan.org/~lds/Bio-BigFile-1.06>; we were unable to find a Java interface for the BigWig file format). Our ReadDB server machine also provided HTTP access to the BAM and BigWig files off the same filesystem that provides ReadDB storage using Apache 2.2.12.

Over a local area gigabit network, remote ReadDB performs similarly to local-disk access and three to five

times faster than remote BAM or BigWig files. As shown in Figure 2, ReadDB also outperforms remote BAM and remote WIG files for all dataset sizes and query region sizes. ReadDB also outperforms a local BAM file in some tests against the large dataset. As expected, the local WIG file often provides good performance as one would expect from a local data source that is a pre-computed histogram.

Tests performed through the network server (ReadDB or HTTP) from the server machine itself help separate the effect of file format from the effects of network throughput and latency. As shown in Figure 3, ReadDB access in this situation is as at least as fast as local BAM access indicating that ReadDB's on-disk format provides a performance advantage over the BAM format as the dataset size increases. The difference between local BAM performance and BAM via HTTP in this test indicates the extra overhead incurred by the HTTP server and avoided by ReadDB's server.

We also performed a subset of tests from a residential broadband connection (10 Mb/s down, 1 Mb/s up, roughly 20 ms latency to the server) to determine the effects of higher latency and lower bandwidth. As seen in Figure 4, runtimes increased for all methods. The difference between ReadDB and BAM increased substantially to five to seven fold. WIG performed very well in this test as its relatively small size incurred the least network transmission time.

Conclusions

ReadDB's query performance bests that of remote BAM and BigWig files on all but the smallest datasets. Since ReadDB's theoretical query time is $O(\log(n) + m)$ (where n is the number of hits stored and m is the number of hits returned), ReadDB should scale to datasets that are many times larger than those evaluated here. Our tests demonstrate that the expected behavior holds over two orders of magnitude in dataset size.

ReadDB provides fast and compact access to aligned short-read datasets in situations where mismatch information and quality scores are unnecessary. In particular, we have found that ReadDB provides an excellent back end for visualization and analysis of ChIP-Seq, DNA methylation, DNase hypersensitivity, and RNA-Seq datasets. We currently store over four thousand alignments covering over two thousand lanes of sequencing and enjoy performance nearing that of local disk without incurring the local storage overhead of BAM or BigWig files.

Availability and Requirements

readdb.jar, provided as additional file 2, contains the Java class files, source files, and a HOWTO file describing how to setup the ReadDB server. ReadDB requires Java 1.6 to run. The source code is provided under the

GPL version 3. Updated versions of the jar file are available at <http://cgs.csail.mit.edu/readdb>.

Additional material

Additional file 1: A description of the ReadDB interface and the method calls it contains.

Additional file 2: The Java class files, source files, and a HOWTO file describing how to setup the ReadDB server. Updated versions of the jar file are available at <http://cgs.csail.mit.edu/readdb>.

Acknowledgements

Thanks to Shaun Mahony and Yuchun Guo for being early adopters of ReadDB and assisting with bug reporting. We also appreciate Dr. Mahony's feedback on the manuscript. PAR was supported by the NIH Common Fund grant 5TL1EB008540 and by 5R01HG002668, P01-NS055923-01, and 1-UL1-RR024920 to DKG.

Author details

¹Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA. ²Pathogenica, Inc, 18th Floor, 245 First Street, Cambridge, MA 02142, USA.

Authors' contributions

AR designed and wrote the software and performed the comparisons to other storage methods. DKG provided design suggestions and helped designed the comparison methods. AR and DKG wrote the paper and read and approved the final version.

Competing interests

The authors declare that they have no competing interests.

Received: 17 November 2010 Accepted: 7 July 2011

Published: 7 July 2011

References

- Hillier LW, Marth GT, Quinlan AR, Dooling D, Fewell G, Barnett D, Fox P, Glasscock JI, Hickenbotham M, Huang W, Magrini VJ, Richt RJ, Sander SN, Stewart DA, Stromberg M, Tsung EF, Wylie T, Schedl T, Wilson RK, Mardis ER: **Whole-genome sequencing and variant discovery in *C. elegans***. *Nat Meth* 2008, **5**(2):183-188.
- Robertson G, Hirst M, Bainbridge M, Bilenky M, Zhao Y, Zeng T, Euskirchen G, Bernier B, Varhol R, Delaney A, Thiessen N, Griffith OL, He A, Marra M, Snyder M, Jones S: **Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing**. *Nat Meth* 2007, **4**(8):651-657.
- Nagalakshmi U, Wang Z, Waern K, Shou C, Raha D, Gerstein M, Snyder M: **The Transcriptional Landscape of the Yeast Genome Defined by RNA Sequencing**. *Science* 2008, **320**(5881):1344-1349 [<http://www.sciencemag.org/cgi/content/abstract/320/5881/1344>].
- Trapnell C, Williams BA, Pertea G, Mortazavi A, Kwan G, van Baren MJ, Salzberg SL, Wold BJ, Pachter L: **Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation**. *Nat Biotech* 2010, **28**(5):511-515.
- The ENCODE (ENCyclopedia Of DNA Elements) Project. *Science (New York, NY)* 2004, **306**(5696):636-640 [<http://www.ncbi.nlm.nih.gov/pubmed/15499007>], [PMID: 15499007].
- Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, Subgroup GPD: **The Sequence Alignment/Map format and SAMtools**. *Bioinformatics* 2009, **25**(16):2078-2079 [<http://bioinformatics.oxfordjournals.org/content/25/16/2078.abstract>].
- Kent WJ, Zweig AS, Barber G, Hinrichs AS, Karolchik D: **BigWig and BigBed: enabling browsing of large distributed datasets**. *Bioinformatics* 2010, **26**(17):2204-2207 [<http://bioinformatics.oxfordjournals.org/content/26/17/2204.abstract>].

8. Danford T, Rolfe A, Gifford D: **GSE: a comprehensive database system for the representation, retrieval, and analysis of microarray data.** *Pac Symp Biocomput* 2008, 539-50.
9. Barski A, Cuddapah S, Cui K, Roh T, Schones DE, Wang Z, Wei G, Chepelev I, Zhao K: **High-Resolution Profiling of Histone Methylations in the Human Genome.** *Cell* 2007, **129**(4):823-837 [<http://www.sciencedirect.com/science/article/pii/S0092867407006009>].

doi:10.1186/1471-2105-12-278

Cite this article as: Rolfe and Gifford: ReadDB Provides Efficient Storage for Mapped Short Reads. *BMC Bioinformatics* 2011 **12**:278.

**Submit your next manuscript to BioMed Central
and take full advantage of:**

- Convenient online submission
- Thorough peer review
- No space constraints or color figure charges
- Immediate publication on acceptance
- Inclusion in PubMed, CAS, Scopus and Google Scholar
- Research which is freely available for redistribution

Submit your manuscript at
www.biomedcentral.com/submit

