RESEARCH

LISA : Learned Indexes for DNA Sequence Analysis

Darryl Ho[1†], Saurabh Kalikar[2†], Sanchit Misra[2*], Jialin Ding[1], Vasimuddin Md[2], Nesime Tatbul[1,3], Heng Li[4] and Tim Kraska[1]

Abstract

Background: Next-generation sequencing (NGS) technologies have enabled affordable sequencing of billions of short DNA fragments at high throughput, paving the way for population-scale genomics. Genomics data analytics at this scale requires overcoming performance bottlenecks, such as searching for short DNA sequences over long reference sequences.

Results: In this paper, we introduce LISA (Learned Indexes for Sequence Analysis), a novel learning-based approach to DNA sequence search. We focus on accelerating two of the most essential flavors of DNA sequence search—*exact search* and *super-maximal exact match (SMEM) search*. LISA builds on and extends FM-index, which is the state-of-the-art technique widely deployed in genomics tools. Experiments with human, animal, and plant genome datasets indicate that LISA achieves up to $2.2\times$ and $13.3\times$ speedups over the state-of-the-art FM-index based implementations for *exact search* and *super-maximal exact match (SMEM) search*, respectively.

Keywords: DNA sequence search; learned indexes; sequence analysis; SMEM; exact search; architecture-aware optimizations

Background

The latest high throughput DNA sequencers can read terabases of DNA sequence data per day. For example, the Illumina NovaSeq 6000 sequencer, introduced in January 2017, can read up to 6 terabases of DNA sequence data in a 44-hour run [1]. In particular, it can sequence nearly 20 billion paired-reads, each of length 150 base-pairs, at a cost as low as $600 per genome. The sequencing throughput is increasing and the cost is decreasing at an exponential rate. The recently unveiled MGI DNBSEQ-TX sequencer, which came just 3 years after Novaseq 6000, can sequence at a rate of up to 20 terabases/day, generating reads of length 150, potentially enabling a $100 cost per genome [2]. Already today, a growing number of public and private sequencing centers with hundreds of NGS deployments are paving the way for population-level genomics. However, realizing this vision in practice heavily relies on building scalable systems for downstream genomics data analysis.

*Correspondence: sanchit.misra@intel.com

[2]Intel Labs, Bangalore, India

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

†Equal contributor

A significant portion of time during the downstream processing of DNA sequence data is spent in searching for DNA sequence queries in a database of reference DNA sequences. This is typically done by building an index of the database to accelerate the search. Recently, in the databases domain, machine learning based index structures (a.k.a., learned indexes) have been shown to accelerate database queries [3]. In this work, we explore the use of learned indexes to accelerate DNA sequence analysis, in particular, DNA sequence search and present our results for a subset of problems that can potentially benefit from using learned indexes.

Reference-guided assembly plays a critical role in downstream analysis. It is performed by piecing together the short reads (with length of a few hundred bases) by mapping each individual read to a long reference genome (e.g., the human genome consisting of 3 billion bases). Thus, a fundamental step of downstream analysis is mapping of millions of short reads (DNA query sequences) to a long reference sequence. BWA-MEM and Bowtie2 are two of the most widely used tools for sequence mapping [4, 5]. The key operation that has been shown to constitute a significant performance bottleneck during this mapping process is the search for *exact* matches of substrings of reads over the given reference sequence [5–10]. In this work, we focus on two variants of that: 1) *exact search*: search of matches of fixed length substrings of a read in the reference sequence and 2) *super-maximal exact match (SMEM) search*: for every position in the read, search of exact matches of longest substring of the read that passes through that position and still has a match in the reference sequence. Thus, SMEM search produces exact matches of variable length by definition. We specifically chose these two variants as these are the key kernels in BWA-MEM and Bowtie2.

The state-of-the-art techniques to perform DNA sequence search are based on building an FM-index over the reference sequence [11]. The FM-index implicitly represents the lexicographically sorted order of all suffixes of the indexed sequences. The key idea behind an FM-index is that, in the lexicographically sorted order of all suffixes of the reference sequence, all matches of a short DNA sequence (a.k.a., a "query") will fall in a single region matching the prefixes of contiguously located suffixes. Over the years, many improvements have been made to make the FM-index more efficient, leading to several state-of-the-art implementations that are highly cache-and processor-optimized [5, 7–10, 12–19]. Hence, it becomes increasingly more challenging to further improve this critical step in the genomics pipeline to scale with increasing data growth.

In this paper, we propose a machine learning based approach to improving the sequence search performance: LISA (Learned Indexes for Sequence Analysis). The core idea behind LISA, which enables a new machine learning enhanced algorithm for DNA sequence search, is to speed up the process of finding the right region of suffixes in the FM-index by learning the distribution of suffixes in the reference. Recent work on learned index structures has introduced the idea that indexes are essentially models that map input keys to positions and, therefore, can be replaced by other types of models, such as machine learning models [3]. For example, a B-tree index maps a given key to the position of that key in a sorted array. Kraska et al. show that using knowledge of the distribution of keys, we can produce a learned model, that outperforms B-trees in query time and memory footprint [3]. Taking a similar perspective, the FM-index can be seen as a model that maps a given query sequence to the single region matching the prefixes of contiguously located suffixes. We introduced LISA in [20] and showed some preliminary results for *exact search*. In this paper, we have

extended LISA to *SMEM search* and developed fully architecture optimized implementations for both problems using multi-threading, vectorization and efficient cache utilization.

More specifically, we make the following contributions in this paper.

- We demonstrate how *exact search* and *SMEM search* problems can be solved using learned indexes. This is the first ever work to do so.
- Since the state-of-the-art algorithms have implementations that are well tuned to the underlying architecture, for a fair comparison, we have developed a fully architecture-optimized implementation of our approach as well. We focus our efforts on the CPU as that is the most widely available architecture for DNA sequence search.
- We demonstrate the benefits of LISA on an Intel® Xeon® Platinum 8280 processor[1]. LISA achieves up to 2.2× and 13.3× speedups over the state-of-the-art implementations for *exact search* and *super-maximal exact match (SMEM) search*, respectively.

Results

We demonstrate the efficacy of LISA by comparing the throughput (million-reads/sec) with FM-Index based *exact search* and *SMEM search*. For the baseline comparison, we use Trans-Omics Acceleration Library (TAL) which provides the architecture optimized implementations for traditional FM-index *exact search* and *SMEM search* [18, 21, 22]. The optimized SMEM kernel from TAL is also used in BWA-MEM2 [21], an architecture-optimized implementation of BWA-MEM [4]. In order to establish TAL as the appropriate baseline, we first show a comparison with Sapling, a learned index based approach for *exact search* that was published just a few weeks back [19].

[1]Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex. ©Intel Corporation. Intel Xeon and Intel Xeon Phi are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Other names and brands may be claimed as the property of others.

Table 1 System Configuration

	Intel® Xeon® Platinum 8280 Processor (CLX)
Sockets × Cores × Threads	1 × 28 × 2
AVX register width (bits)	512, 256, 128
Vector Processing Units (VPU)	2/Core
Base Clock Frequency (GHz)	2.5
L1D/L2 Cache (KB)	32/1024
L3 Cache (MB) / Socket	38.5
DRAM (GB) / Socket	96
Bandwidth (GB/s) / Socket	128
Compiler Version	ICC v. 19.1.3.304

Table 2 Reference Sequences

Reference Sequence	Length (Million bases)	Version
Human	3101	human_g1k_v37
Asian Rice	387	IR64 (IRRI)
Zebra Fish	1679	GRCz11

Experimental Setup

System Configuration

We evaluate our solution on a single socket of Intel® Xeon® Platinum 8280 processor as detailed in Table 1 and referred to as CLX from here on. To force all memory allocations to one socket, we use the *numactl* utility . For multi-threaded runs, we use 2 threads per core to get the benefit of hyper threads. Optimizing file I/O is beyond the scope of this paper. Therefore, we do not include file I/O time in any of our results.

Datasets

We use three reference sequences - Human, Asian Rice, and Zebra Fish as detailed in Table 2. For each of these reference sequences, we use multiple real read datasets (H1-H5 for Human, A1-A3 for Asian rice, and

Table 3 Read Datasets

Read Datasets	Organism	Length	No. of Reads	Source (NCBI-SRA)
H1	Human	151	5 million	ERR2990063
H2	Human	151	5 million	ERR3239330
H3	Human	151	5 million	SRR7733443
H4	Human	101	5 million	SRR622461
H5	Human	101	5 million	SRR622457
A1	Asian Rice	151	5 million	SRR10724346
A2	Asian Rice	151	5 million	SRR10838753
A3	Asian Rice	151	5 million	SRR8241153
Z1	Zebra Fish	151	5 million	ERR2624531
Z2	Zebra Fish	151	5 million	ERR3333446
Z3	Zebra Fish	151	5 million	SRR10958316

Table 4 Seed Datasets

Seed Datasets	Length	No. of Seeds	Original Read Dataset
S1	21	50 million	H1
S2	21	50 million	H2
S3	21	50 million	H3
S4	21	50 million	H4
S5	21	50 million	H5
S6	21	50 million	A1
S7	21	50 million	A2
S8	21	50 million	A3
S9	21	50 million	Z1
S10	21	50 million	Z2
S11	21	50 million	Z3

Z1-Z3 for Zebra Fish) downloaded from sequence read archive [23] (Table 3). All of these read datasets consist of 5 million reads. The read datasets for Asian rice and Zebra fish have reads of length 151. For Human reference, we use two types of reads datasets: H1-H3 contain of 151 length reads and H4-H5 are of 101 length. The older sequencing technologies produce reads of length 101, so we use these 101 length datasets to show the compatibility of LISA with the older sequencing technologies. For *exact search*, we use 21 length seeds generated from the read datasets. Seeds are the small fixed-sized substrings generated from a read sequence. For generating seeds, we followed the same strategy as Bowtie2 and generated 50 million seeds for each of the read datasets [5].

Correctness

In all cases, we have verified that output of LISA based approach is identical to that of the traditional FM-index based approach.

Establishing The Baseline

Here, we compare the execution time of LISA and TAL with the recently published Sapling [19] for *exact search*. Sapling demonstrated a speedup of over 2× over Bowtie, Mummer and an optimized implementation of binary search. Therefore, we omit any comparison with Bowtie, Mummer and binary search. Moreover, Sapling is single threaded. Therefore, we compare the performance of the three implementations using only a single thread. We have used the same evaluation method as used in the Sapling paper. We use the scripts provided with Sapling source code [24] to generate 50 million seeds of length 21 for the three reference sequences. The script ensures that there is at least one match of the generated seeds.

Figure 1 shows the comparison. Note that the time reported for Sapling here is 1) more than 2× less that the time reported in the [19] potentially due to a dif-

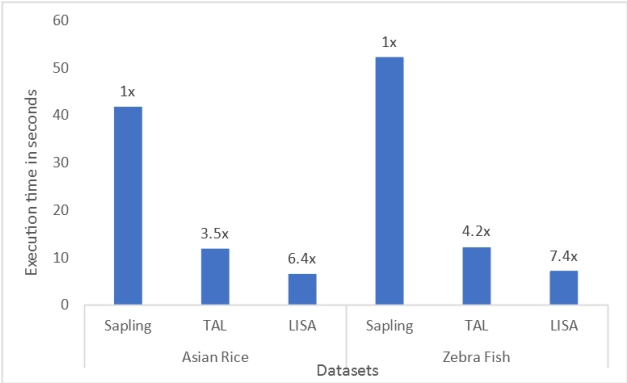

Figure 1 Performance of the *exact search* for Sapling, TAL, and LISA using a single thread.

ference in architecture and 2) the time spent in getting one position in the interval in the suffix array in which the query matches – one needs to perform a search on both sides of the position to get the interval. We could not run Sapling for human genome as it ran out of memory even when run on a different machine with 256 GB DRAM. The time reported for TAL and LISA is the time spent in getting the full interval.

Figure 1 shows that TAL is 3.5× and 4.2× faster than Sapling and LISA is 6.4× and 7.4× faster than Sapling, respectively, for Asian Rice and Zebrafish. Therefore, for the rest of the paper, we only compare LISA with TAL.

Performance Evaluation

Exact Search

Figures 2 and 3 show the throughput achieved by LISA and TAL for *exact search* on a single thread and single socket, respectively, across different reference sequences and the read datasets. The x-axis represents the reference sequence and the datasets, and the throughput is shown on the y-axis (the higher the better). Observe that LISA outperforms TAL across all datasets.

Exact search finds all end-to-end matches of 21-length seeds. Recall that the traditional FM-index based search matches one base at a time against the reference sequence and therefore takes 21 steps for end-to-end matching of 21-length seed. LISA processes a whole 21-length seed in one shot and finds its matches in a single step. As a result, LISA achieves $1.4 - 2.2\times$ higher throughput than TAL.

SMEM Search

Figures 4 and 5 show the throughput comparison for *SMEM search*. LISA achieves $4.4 - 13.3\times$ higher throughput than TAL. On a single threaded execution, LISA achieves, on an average, 8.4× speedup over TAL across all datasets. On a multithreaded execution, LISA achieves, on an average, 5.43× speedup over TAL.

Although LISA outperforms TAL across all datasets, the performance gain varies across datasets. The nature of reads and the reference sequences affect the overall performance gain. For instance, a read dataset with longer matching SMEMs is a better candidate for LISA than the one with the shorter matches. In Figures 4 and 5 the average length of the matches in H1 is 45 where as in H3, the average length is 26.

Conclusions and Future Work

Creating an index of the database appears as a motif in many key areas in computational biology including genomics, transcriptomics and proteomics. In this

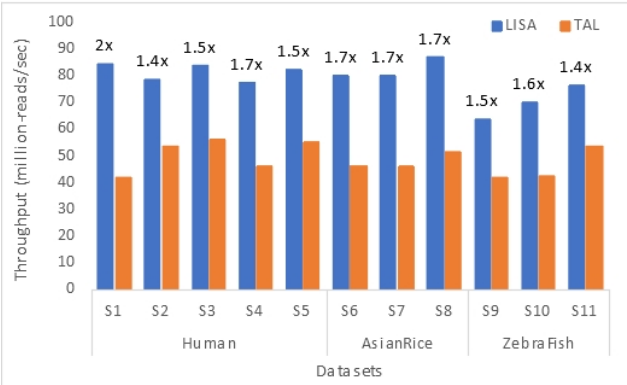

Figure 2 Performance evaluation for *exact search* problem on a single thread. Throughput gain for each dataset is shown on the top of LISA bars.

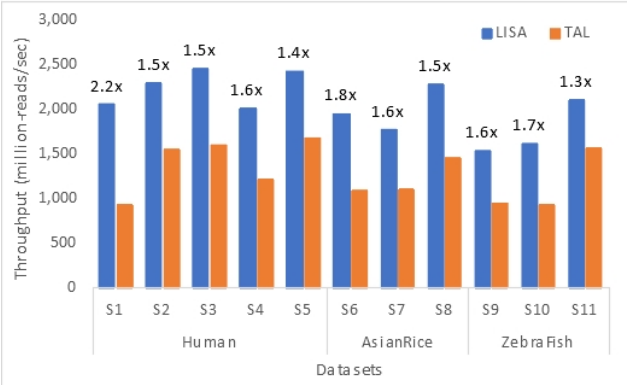

Figure 3 Performance evaluation for *exact search* problem on a single socket (28 cores). Throughput gain for each dataset is shown on the top of LISA bars.

paper, we presented LISA - a machine learning based approach to index a database of DNA sequence to accelerate DNA sequence search. We demonstrated the benefits of our approach through two specific variants of DNA sequence search - *exact search* and *SMEM search* and show up to 2.2× and 13.3× speedup, respectively. As future work, we plan to extend the ideas presented in this paper to many other problems in computational biology in which an index is created to accelerate search through a database. In particular, hash tables are a prime candidate for acceleration through the learned approach.

Methods

This section is being written as of now and will be available soon.

Acknowledgements

This research is supported by Google, Intel, and Microsoft as part of the MIT Data Systems and AI Lab (DSAIL) at MIT.

Author details

[1]Massachusetts Institute of Technology, Cambridge, USA. [2]Intel Labs, Bangalore, India. [3]Intel Labs, USA. [4]Harvard Medical School, Boston, USA.

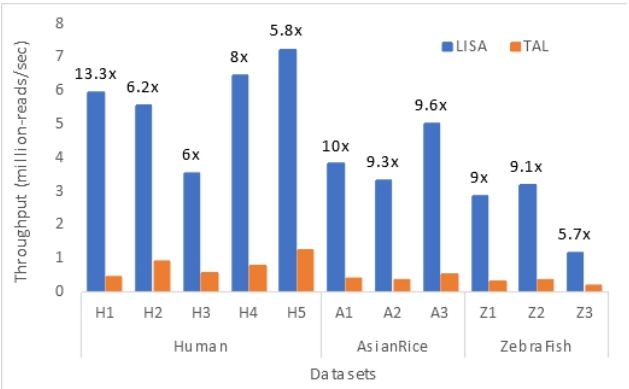

Figure 4 Performance evaluation for *SMEM search* problem on a single thread. Throughput gain for each dataset is shown on the top of LISA bars.

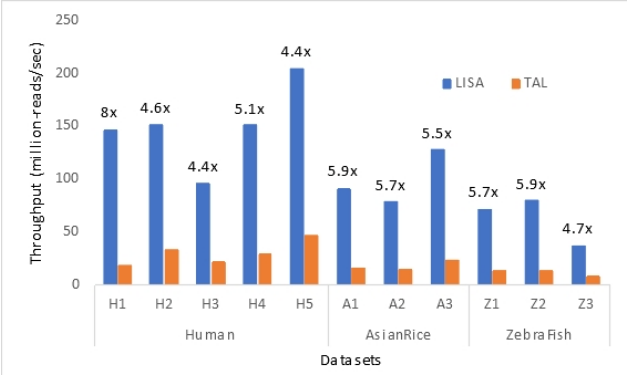

Figure 5 Performance evaluation for *SMEM search* problem on a single socket (28 cores). Throughput gain for each dataset is shown on the top of LISA bars.

References

1. Illumina Inc.: HiSeqX^{TM} Series of Sequencing Systems. url=https://www.illumina.com/content/dam/illumina-marketing/documents/products/datasheets/novaseq-6000-system-specification-sheet-770-2016-025.pdf. Accessed: November 2020

2. Julia Karow: MGI Unveils 'Extreme Throughput' Sequencing Platform at AGBT to Enable $100 Human Genome. url=https://www.genomeweb.com/sequencing/mgi-unveils-extreme-throughput-sequencing-platform-agbt-enable-100-human-genome.X6VFe4gzaUk. Accessed: November 2020 (2020)

3. Kraska, T., Beutel, A., Chi, E.H., Dean, J., Polyzotis, N.: The Case for Learned Index Structures. In: ACM International Conference on Management of Data (SIGMOD), pp. 489–504 (2018)
4. Li, H.: Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. arXiv preprint arXiv:1303.3997 (2013)
5. Langmead, B., Salzberg, S.L.: Fast Gapped-read Alignment with Bowtie 2. Nature Methods **9**(4), 357–359 (2012)
6. Md, V., Misra, S., Aluru, S.: Identification of Significant Computational Building Blocks through Comprehensive Investigation of NGS Secondary Analysis Methods. bioRXiv (2018)
7. Langmead, B., Trapnell, C., Pop, M., Salzberg, S.L.: Ultrafast and Memory-efficient Alignment of Short DNA Sequences to the Human Genome. Genome Biology **10**(3), 1 (2009)
8. Li, R., Yu, C., Li, Y., Lam, T.-W., Yiu, S.-M., Kristiansen, K., Wang, J.: SOAP2: An Improved Ultrafast Tool for Short Read Alignment. Bioinformatics **25**(15), 1966–1967 (2009)
9. Luo, R., Wong, T., Zhu, J., Liu, C.-M., Zhu, X., Wu, E., Lee, L.-K., Lin, H., Zhu, W., Cheung, D.W., *et al.*: SOAP3-dp: Fast, Accurate, and Sensitive GPU-Based Short Read Aligner. PLOS ONE **8**(5), 65632 (2013)
10. Li, H., Durbin, R.: Fast and Accurate Short Read Alignment with Burrows–Wheeler Transform. Bioinformatics **25**(14), 1754–1760 (2009)
11. Ferragina, P., Manzini, G.: Opportunistic Data Structures with

Applications. In: IEEE Annual Symposium on Foundations of Computer Science (FOCS), pp. 390–398 (2000)

12. Chacón, A., Moure, J.C., Espinosa, A., Hernández, P.: n-step FM-Index for Faster Pattern Matching. Procedia Computer Science **18**, 70–79 (2013)
13. Chacón, A., Marco-Sola, S., Espinosa, A., Ribeca, P., Moure, J.C.: Boosting the FM-Index on the GPU: Effective Techniques to Mitigate Random Memory Access. IEEE/ACM Transactions on Computational Biology and Bioinformatics **12**(5), 1048–1059 (2015)
14. Pantaleoni, J., Subtil, N.: NVBIO: A Library of Reusable Components Designed by NVIDIA Corporation to Accelerate Bioinformatics Applications using CUDA. http://nvlabs.github.io/nvbio/. Accessed: November 2017
15. Zhang, J., Lin, H., Balaji, P., Feng, W.-c.: Optimizing Burrows-Wheeler Transform-based Sequence Alignment on Multi-core Architectures. In: IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid), pp. 377–384 (2013)
16. Fernandez, E., Najjar, W., Lonardi, S.: String Matching in Hardware Using the FM-Index. In: IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, pp. 218–225 (2011)
17. Grabowski, S., Raniszewski, M., Deorowicz, S.: FM-index for Dummies. In: International Conference on Beyond Databases, Architectures, and Structures (BDAS), pp. 189–201 (2017)
18. Misra, S., Pan, T.C., Mahadik, K., Powley, G., Vaidya, P.N., Vasimuddin, M., Aluru, S.: Performance Extraction and Suitability Analysis of Multi- and Many-core Architectures for Next Generation Sequencing Secondary Analysis. In: Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques (PACT), pp. 3–1314 (2018)
19. Kirsche, M., Das, A., Schatz, M.C.: Sapling: Accelerating Suffix Array Queries with Learned Data Models. Bioinformatics (2020). doi:10.1093/bioinformatics/btaa911. btaa911. https://academic.oup.com/bioinformatics/advance-article-pdf/doi/10.1093/bioinformatics/btaa911/34042255/btaa911.pdf
20. Ho, D., Ding, J., Misra, S., Tatbul, N., Nathan, V., Md, V., Kraska, T.: LISA: Towards Learned DNA Sequence Search (2019). 1910.04728
21. Vasimuddin, M., Misra, S., Li, H., Aluru, S.: Efficient architecture-aware acceleration of bwa-mem for multicore systems. In: 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 314–324 (2019). IEEE
22. Trans-Omics-Acceleration-Library. url=https://github.com/IntelLabs/Trans-Omics-Acceleration-Library. Accessed: November 2020 (2020)
23. NCBI Sequence Read Archive. url=https://www.ncbi.nlm.nih.gov/sra/. Accessed: November 2020
24. SAPLING: Suffix Array Piecewise Linear INdex for Genomics. url=https://github.com/mkirsche/sapling. Accessed: November 2020