Experimental Survey on Power Dissipation of *k*-mer-Handling Data Structures for Mobile Bioinformatics

Franco Milicchio
Department of Engineering
Roma Tre University
Rome, Italy
franco.milicchio@uniroma3.it

Mattia Prosperi
Department of Epidemiology
University of Florida
Gainesville, FL, USA
m.prosperi@ufl.edu

2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) | 978-1-6654-0126-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/BIBM52615.2021.9669768

***Abstract*— Mobile sequencing technologies, including Oxford Nanopore's MinION, Mk1C, and SmidgION, are bringing genomics in the palm of a hand, opening unprecedented new opportunities in clinical and ecological research and translational applications. While sequencers now need only a USB outlet and provide on-board preprocessing (e.g., base calling), the main data analysis phases are tied to an available broadband Internet connection and cloud computing. Yet the ubiquity of tablets and smartphones, along with their increase in computational power, makes them a perfect candidate for enabling mobile/edge mobile bioinformatics analytics. Also, in on site experimental settings tablets and smartphones are preferable to standard computers due to resilience to humidity or spills, and ease of sterilization. We here present an experimental study on power dissipation, aiming at reducing the battery consumption that currently impedes the execution of intensive bioinformatics analytics pipelines. In particular, we investigated the effects of assorted data structures (including hash tables, vectors, balanced trees, tries) employed in some of the most common tasks of a bioinformatics pipeline, the *k*-mer representation and counting. By employing a thermal camera, we show how different *k*-mer-handling data structures impact the power dissipation on a smartphone, finding that a cache-oblivious data structure reduces power dissipation (up to 26% better than others). In conclusion, the choice of data structures in mobile bioinformatics must consider not only computing efficiency (e.g., succinct data structures to reduce RAM usage), but also power consumption of mobile devices that heavily rely on batteries in order to function.**

Keywords—data structures, power dissipation, cache, k-mer, mobile bioinformatics, edge computing

I. Introduction

Mobile sequencing technologies are bringing genomics in the palm of a hand, opening unprecedented new opportunities in clinical and ecological research and translational applications [1]. The chemistry behind is 'nanopore' sequencing, a technology employing nanometer-sized holes on a membrane through which a DNA molecule is passed and recognized. Nanopore sequencing enables very long DNA strands to be recognized (up to hundred thousands of bases), compared to shorter fragments deriving from high-throughput sequencers, e.g., Illumina (https://www.illumina.com/). Currently, Oxford Nanopore Technologies ltd (https://nanoporetech.com/) is the sole relevant market supplier with several devices: the MinION sequencer, Mk1C, GridION, PromethION, and SmidgION. The MinION sequencer weighs 90g and measures 10×3×2cm, making it the smallest sequencing device currently available on the market. The maximum throughput is within the tens of gigabyte range: sequence reads have a median length of a few kilobases, but can reach hundreds of kilobases [2]. However, the long read throughput of these devices is affected by a high error rate, sensibly higher than other technologies, up to 10% for less recent chemistry, but decreased over the years [3].

In the same manner, mobile and edge computing have risen thanks to a strong smartphone/tablet market penetration, which rendered mobile devices an ubiquitous form of computing employed in different science domains, e.g., sociology [4-6], biomedicine, and human-computer interaction [7, 8]. However, the employment of mobile computing architectures is still limited within bioinformatics, where software tools are bound to desktop, sever, or cloud architectures, usually based on Intel-compatible processors, thus rendering portable analytics still unfeasible and impractical [9].

Nanopore's MinION output format is called FAST5, and it is based on the standard Hierarchical Data Format 5 (HDF5, http://www.hdfgroup.org/HDF5/), allowing for metadata content. Since the reads exhibit high error rates and have a peculiar length distribution, there has been a substantial development of ad-hoc algorithms and data structures for MinION data analytics, with further development into data interoperable libraries, integrated data processing pipelines [10], and web-based tools at the consumer's grade, e.g., the Metrichor (https://metrichor.com).

Many sequencing analytics tools, e.g., de novo assembly, metagenomics taxonomy, functional classification, rely on data structures that exploit *k*-mers, i.e., strings of fixed length *k*.

De novo assembly refers to the reconstruction of an organism's genome by merging all the sequenced reads, and often relies on the de Bruijn graph data structure built upon *k*-mers [11]. In the de Bruijn graph, nodes are *k*-mers, and two *k*-mers are connected by an arc if they overlap exactly by (*k*-1) characters, by aligning (or sliding) them starting from the end of the first *k*-mer and the head of the second. For instance, the 7-mers GATTACA and ATTACAT overlap by ATTAC. Once the graph has been built, the genome is reconstructed by applying an algorithm for Eulerian path discovery [12]. A number of efficient data structures and algorithmic solutions for setting up the de Bruijn graph and searching for Eulerian paths have been devised and implemented into usable tools [13-14].

Assembly tools, as well as metagenomics classification tools, must deal with sequencing error rates, generating

Supported in part by NSF SCH #2013998.

erroneous *k*-mers, which can be filtered through different approaches, based on overall frequency and/or consistency of paths in the de Bruijn graph [15-17]. Bottom line, the parsing and counting of all *k*-mers from reads, creating a table of all *k*-mers and their frequencies, called the *k*-mer spectrum, is the prerequisite of most downstream analysis. Calculating a *k*-mer spectrum is an onerous task that has no trivial solution. Given the size of sequencers' output files, in the tens and hundreds of gigabytes, a glut of diverse approaches has been devised [18-24].

Several tools employ out-of-core approaches, i.e., utilizing the disk in order to overcome the lack of RAM due to the size of the genomic sequences [18]. Additionally, binary trees and hashing techniques have been employed to speed-up the computation of the *k*-mer spectrum, either correctly or approximately [22-23], with genomes being represented implicitly or explicitly by means of integers or strings [19, 24].

As the current trend in computing shows, *"cache is the new RAM."* Modern computers are not represented by the classic von Neumann architectures, with a progressive gap between theory and practice when comparing performances [13]. In fact, the concept of one single memory has been superseded by a hierarchy of memories with different performances [25]. From CPU registers to memory caches of various levels (L1 to L3), from RAM to disk and to the network, the performances degrade progressively and severely. Merely counting the number of machine instructions as a measure of algorithmic runtime does not account the cache levels involved in the computation, whereas the cache oblivious model [27] as opposed to the classic one, allows to focus on a two-layered memory model, extending its results to architectures with different memory hierarchies: results obtained in the cache oblivious model hold in architectures with three or more memory layers.

Contribution. In this work we address the feasibility of truly portable *k*-mer representation and counting on mobile devices, by assessing the power consumption of diverse, commonly used *k*-mer-handling data structures. Additionally, we developed a cache-oblivious Van Emde Boas tree [26] to assess the cache efficiency usage on the power dissipation. By employing a thermal camera, we retrieved the thermal field on a smartphone running the counting algorithm, and we were able to compute the overall power dissipation. Results show that employing a cache-oblivious data structure can reduce the total power dissipated by 26% as compared to other data structures.

Overview. This paper is organized as follows. Section II will introduce the principal data structures, and issues regarding them, employed in the bioinformatics field to tackle the *k*-mer representation and counting problem, motivating its importance. Section III will illustrate the experimental setup employed in the computation of the power dissipation. Section IV will illustrate the results, discuss them and draw conclusions.

II. *K*-MER REPRESENTATION AND COUNTING

The necessary steps to enable metagenomics, de novo assembly and other related analytics on mobile devices include initial parsing and processing read sequences into *k*-mers. Such data processing must be executable in reasonable turnaround time and with minimal side-effects, e.g., overheating, or excessive battery consumption. The analysis of high-throughput sequencing data often requires a considerable amount of memory and CPU resources, even when compressed data structures are used [22-23]. Therefore, it is highly likely that power efficient data structures are necessary.

The first issue with *k*-mer counting relies in the representation itself of a *k*-mer. Counters parse a textual file containing a string of 8-bit ASCII characters representing individual nucleotide bases, namely adenosine ('A'), cytosine ('C'), thymine ('T'), and guanine ('G'), to say nothing of ambiguous bases, quality scores, and misplaced characters. The most recent introduction of binary files, such as the FAST5 file format, still rely on the ASCII description of the genetic sequence, output of a portable sequencer. A *k*-mer counter therefore should choose whether to store the string-based representation of the string [29], or to transcode the sequence into a more memory efficient form [20, 30]. As DNA is based on four nucleotides, one solution is to represent *k*-mers by employing 2 bytes per nucleotide, for instance indicating with 00 ('A'), 01 ('C'), 11 ('T'), and 10 ('G'): such bitwise compacted representation requires less memory when compared with a string one [28], with no impact on performance. Hence, the bitwise is the representation of choice of most of *k*-mer counters.

The second issue is to efficiently count the *k*-mers from files that can be very large. Usually, two different approaches are sought: in-core and out-of-core, i.e., relying on the main memory and utilizing the external mass storage, respectively. In either case, appropriate data structures must be employed. Popular data structures employed to store the *k*-mer spectrum include hash-based and tree-based structures, exact (lossless, collision-free) or probabilistic (i.e. lossy).

Hash tables are notoriously fast for item access, with O(1) theoretical computational complexity, and usually implemented by starting from a vector, or a contiguous portion of RAM. However, such approach has several drawbacks. Several items could be assigned the same hash value, thus rendering the memory continuity layout impossible to maintain, requiring either rehashing or collision lists. As the concurrent access to a hash table is required in order to speedup computations, several tools—see, for instance, [18]—employ minimizers, or shorter hashes requiring less bits, in order to minimize the impact of concurrent access.

Other approaches employ trees or a variation of such data structure. For instance, in [23] the authors implemented a trie, i.e., a non-binary tree whose nodes represent single nucleotide, and a path from the root a genomic sequence. Vectors and arrays were employed [24] in lieu of hashing with significant performance improvement, especially in the context of parallel processing of sequences reads. Such structures offer inherent cache-efficient memory handling, and some trees also may be laid out in memory to exploit the benefits offered by cache memory layers.

III. POWER DISSIPATION

In order to perform a comprehensive assessment of the power dissipation involved in the *k*-mer representation and counting processes, we tested the major data structures

employed by state-of-the-art software tools, namely: (i) a resizable sorted vector; (ii) a red-black self-balancing tree; (iii) and a hash table. To be consistent across tests, we used the ISO C++14 programming language. Also, C++ has been proven one of the top performing, portable, energy efficient programming languages [31], a highly desirable property in the context of application development for mobile devices [34].

Among the chosen data structures, only the sorted vector is cache-efficient, as items are stored in a contiguous portion of the main memory. Hash tables, in case of no collisions, store items in a single memory block, however, owing to the hashing properties, memory addresses are accessed randomly, thus invalidating the spatial locality required by the cache. Trees are node-based data structures with values and pointers, with each node allocated at runtime in a non-necessarily contiguous fashion, thus with negated cache benefits. However, tree-enabling cache-oblivious data structures can be used, as explained in the next subsection.

A. The Van Emde-Boas Tree

The use of cache-oblivious data structures has been recently proposed in the bioinformatics field [32]. Such structures are cache optimized without any prior knowledge of the cache implementation [25]. In this experimental setup, we implemented a binary tree employing the Van Emde Boas memory layout [26].

Given a binary search tree, where each node has a constant number of children, a recursive mapping from the tree nodes to memory locations is made as follows, and pictured in Figure 1. The tree is subdivided in half with respect to the height. This operation leads to portions of the tree containing $O(\sqrt{n})$ nodes, being n the total number of nodes in the tree: the top containing the root, and each sub-tree of the top-portion's leaves. Then, each fragment of the tree is assigned to a contiguous vector, in order: the top, and each sub-tree starting from the leftmost one. Then, recursively, each sub-tree is subdivided in the same fashion until the base case is reached, i.e., a subtree consisting of only three nodes, a root and two leaves. This recursive mapping implements an associative array with m-bit integer keys [26] storing a large number of elements and performing operations in $O(\log m)$ time, or equally in $O(\log \log n)$ time. We refer the reader to [27] for a review of cache-oblivious binary trees.

B. Experimental Setup

Our experimental setup has the objective of recording the power dissipation in a common smartphone executing the k-mer parsing, representation and counting. In order to measure the power dissipation, we employed a Teledyne FLIR model A310 (cf. https://www.flir.com/products/a310/). This particular model is capable of recording accurate temperature fields ranging from -20 °C to +120 °C, with a ±2% accuracy.

The smartphone of choice was an Apple iPhone 6, an ARM-based smartphone, chosen because it has been the most successful smartphone ever produced, accounting 222 million units sold to date [33]. The experimental setup is depicted in Figure 2, with the smartphone and thermal FLIR A310 camera attached to the data processing computer, running the Microsoft Windows operating system.

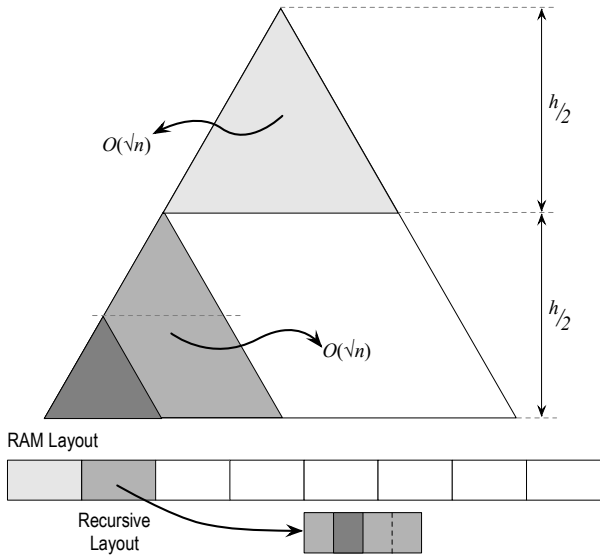

Fig. 1 The Van Emde Boas binary tree recursive layout.

IV. Results

We conducted the k-mer processing tests on the smartphone by considering the k-mer spectrum of the publicly available Escherichia Coli genome (NCBI accession SRR14875861 / SRX11193265). All k-mers were encoded as 64-bits integers by transcoding the ASCII sequence into a string of 2-bits per nucleotide as described in Section 2.

Due to the limitations of RAM and permanent storage allowed on the device, we limited the number of k-mers to fit the maximum allowed size each of the considered data structures, as managed by the operating system.

In order to be able to record the temperature field without any interference, we developed an application in C++ and Objective-C in Apple Xcode IDE, allowing the user to tap on a button and starting the k-mer count with a delay sufficient for the operator to place a distance from the thermal camera and cause no thermal interference on the results.

We recorded the execution and temperature field of the smartphone from the start of the execution up to 1,200 seconds from the beginning. The thermal plots of the experiments and the smartphone's thermal field for each data structure are depicted in Figure 3.

The last piece of information needed to compute the power dissipation of each data structure, is the thermal conductivity coefficient of the smartphone, being the power dissipation determined as $P = c_T \Delta T$, with c_T being the thermal conductivity coefficient, while ΔT being the change in temperature. In order to retrieve c_T we conducted an additional experiment in a controlled environment following the specifications from the manufacturer. The total battery energy is 6.55 Wh (cf. Apple, Inc. technical specifications), and we allowed the device to navigate the web uninterrupted for 2 hours, reaching a temperature of 37 °C, with an ambient temperature of 24 °C, identical to the initial temperature of the device at rest. With the energy data, time, and temperature increase we were able to

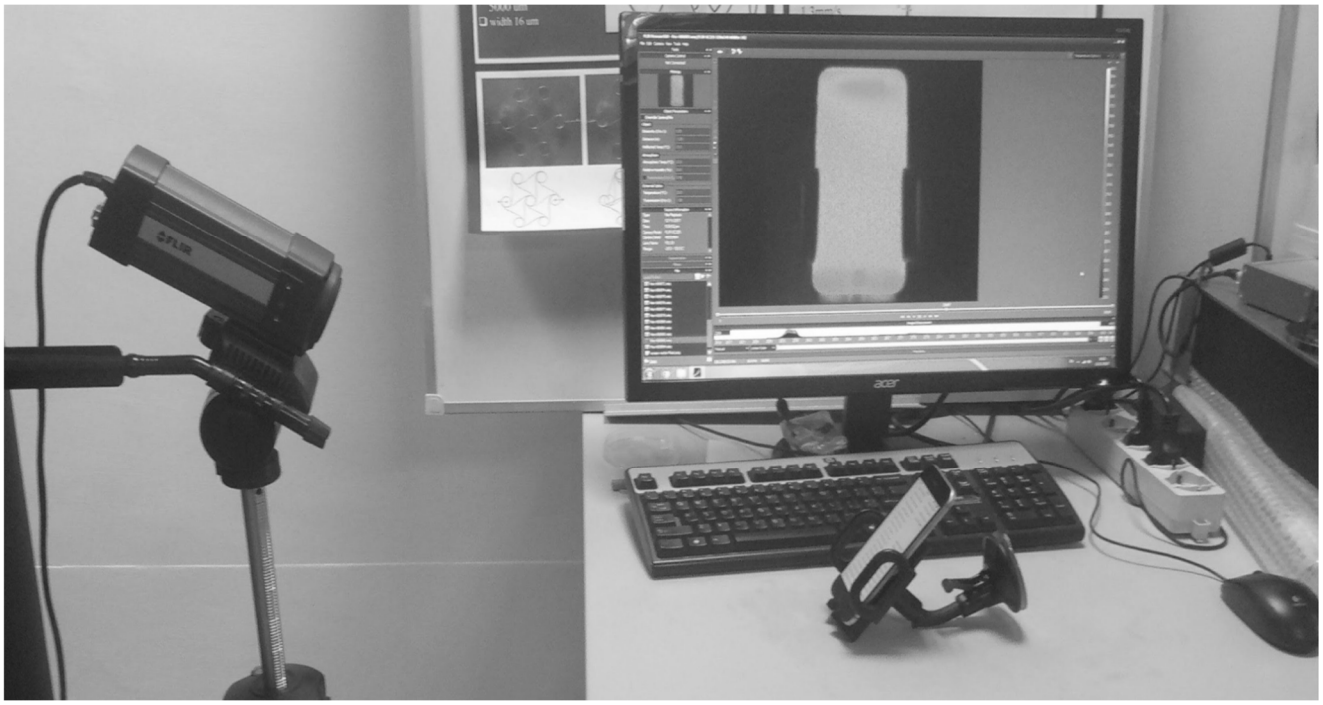

Fig. 2 The experimental setup with the thermal camera, smartphone, and data processing workstation. On the monitor screen the thermal field is visible.

compute the thermal conductivity coefficient as $c_T = 0.25$ W/°C. With the thermal coefficient determined we calculated the total power dissipation for the four data structures under scrutiny, and detailed in Table 1 (results are averaged over 10 independent replications).

As one can see, of the four data structures, the most power efficient ones are the sorted vector and the Van Emde Boas tree, owing to their cache optimization, consuming on average 2.0 W and 1.7 W, respectively. When compared to the most commonly employed data structures, trees and hash tables that consume 2.3 W, we can conclude that, by implementing a *k*-mer counter with a cache-oblivious data structure, the power dissipation can be reduced by 26%, and thus the strain on the battery.

TABLE I. EXPERIMENTAL RESULTS OVER 10 REPLICATIONS

Data Structure	***Final Temperature °C mean (st.dev)***	***Power Dissipation (W) mean (st.dev)***
Sorted vector	32.5 (2.3)	2.0 (0.5)
Red-Black tree	33.3 (2.8)	2.3 (0.7)
Hash table	33.5 (2.8)	2.3 (0.5)
Van Emde Boas tree	31.2 (2.4)	1.7 (0.5)

V. CONCLUSIONS

We conducted an experimental survey aimed at determining the quantitative effects in terms of power dissipation for different data structures used commonly by bioinformatics analytics tools. Our tests were focused on *k*-mer-handling processes, which are common to many analytics procedures, from error correction, to genome assembly, to metagenomics classification. We implemented three commonly used data structures—a sorted vector, a tree, a hash map—and additionally developed a cache-oblivious data structure, the Van Emde Boas tree. The thermal camera recording allowed us to measure the power dissipation for the data structures, and showed how the cache-oblivious Van Emde Boas tree achieves a 26% reduction in power dissipation with respect to the state-of-the-art data structures, i.e., trees and hash tables.

In conclusion, cache-oblivious data structures should be considered as an advantageous and realistic option for the development of mobile bioinformatics tools.

ACKNOWLEDGMENTS

The work and Dr. Prosperi were supported in part by US Federal grant NSF SCH #2013998.

REFERENCES

[1] Lin B, Hui J, Mao H. Nanopore Technology and Its Applications in Gene Sequencing. Biosensors (Basel). 2021 Jun 30;11(7):214. doi: 10.3390/bios11070214.

[2] Midha MK, Wu M, Chiu KP. Long-read sequencing in deciphering human genetics to a greater depth. Hum Genet. 2019 Dec;138(11-12):1201-1215. doi: 10.1007/s00439-019-02064-y.

[3] Rang FJ, Kloosterman WP, de Ridder J. From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy. Genome Biol. 2018 Jul 13;19(1):90. doi: 10.1186/s13059-018-1462-9.

[4] N. Bitman, and N.A. John, "*Deaf and Hard of Hearing Smartphone Users: Intersectionality and the Penetration of Ableist Communication Norms*", Journal of Computer-Mediated Communication, Vol. 24, No. 2, pp. 56-72, 2019.

[5] P. Grenfell, N. Tilouche, J. Shawe, R.S. French, "*Fertility and digital technology: narratives of using smartphone app 'Natural Cycles' while trying to conceive*", Sociol Health Illn., Vol. 43, No. 1, pp. 116-132, 2021, https://doi.org/10.1111/1467-9566.13199. Epub 2020 Nov 4. PMID: 33147647; PMCID: PMC7894554.

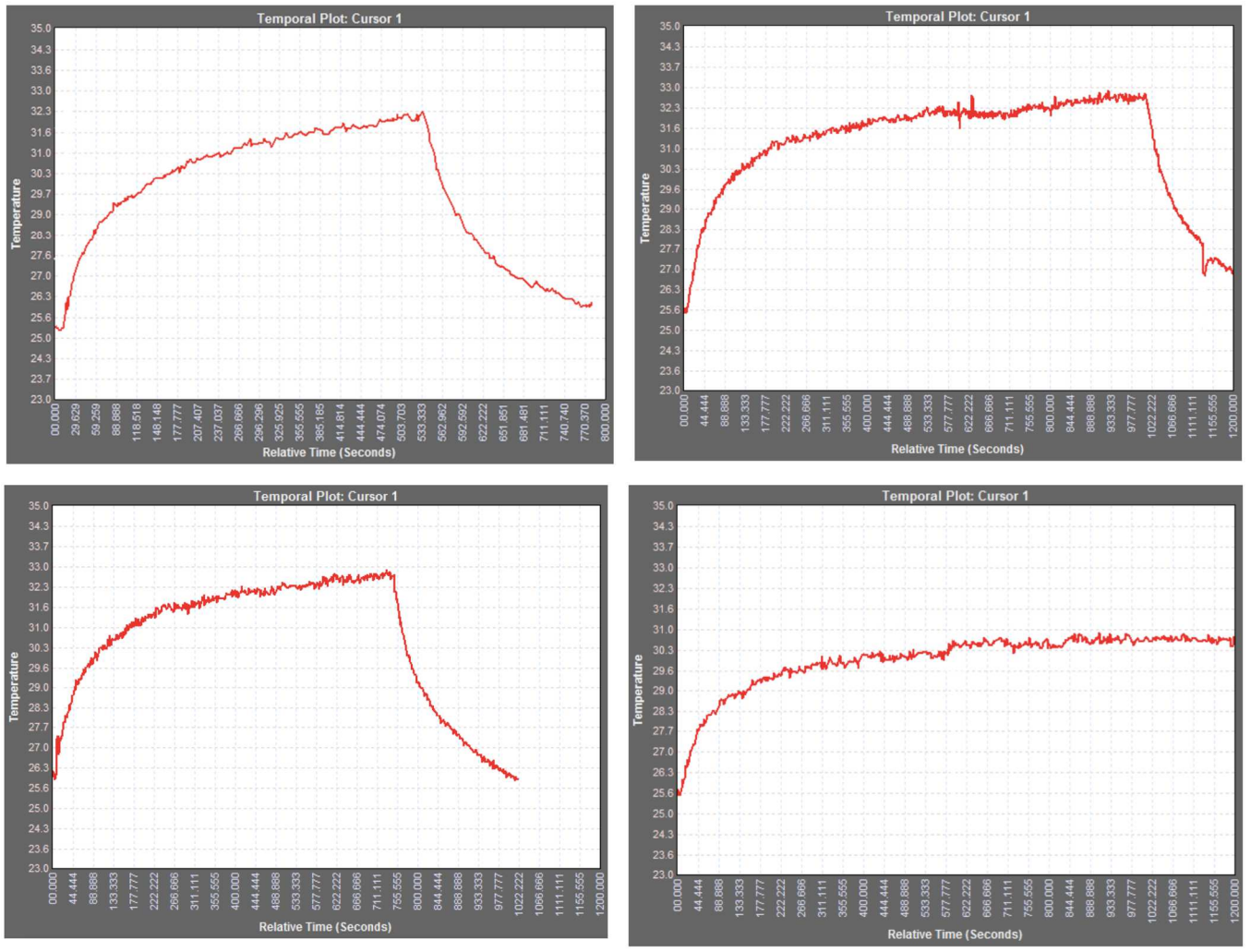

Fig. 3. Top panels: the temperature temporal plots of sorted vector (left) and red-black tree (right). Bottom panels: the temperature temporal plot of the hash table (left) and the Van Emde Boas tree (right).

[6] F. Milicchio and M.C.F. Prosperi, "Accessible Tourism for the Deaf via Mobile Apps", Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments, June 2016 No. 23, pp. 1-7, https://doi.org/10.1145/2910674.2910694.

[7] V. Selamneni, P. Barya, N. Deshpande and P. Sahatiya, "*Low-Cost, Disposable, Flexible, and Smartphone Enabled Pressure Sensor for Monitoring Drug Dosage in Smart Medicine Applications*", IEEE Sensors Journal, Vol. 19, No. 23, pp. 11255-11261, 2019, https://doi.org/10.1109/JSEN.2019.2935383.

[8] A. Bottaro, et al. "Visual Programming of Location-Based Services". In: Human Interface, Part I, HCII 2011, (2011).

[9] M. Oliva, et al., "*Portable nanopore analytics: are we there yet?*", Bioinformatics, Vol. 36, No. 16, pp. 4399-4405, 2020, https://doi.org/10.1093/bioinformatics/btaa237. PMID: 32277811; PMCID: PMC7828464.

[10] M. Jain, H.E. Olsen, B. Paten, M. Akeson, "*The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community*", Genome Biol., Vol. 17, No. 1, 2016, https://doi.org/10.1186/s13059-016-1103-0. Erratum in: Genome Biol. 2016 Dec 13;17(1):256. PMID: 27887629; PMCID: PMC5124260.

[11] Nagarajan, N., Pop, M. Sequence assembly demystified. Nat Rev Genet 14, 157–167 (2013). https://doi.org/10.1038/nrg3367.

[12] M.J. Chaisson, P.A. Pevzner, "*Short read fragment assembly of bacterial genomes*", Genome Res., Vol. 18, No. 2, pp. 324-30, 2008, https://doi.org/10.1101/gr.7088808. Epub 2007 Dec 14. PMID: 18083777; PMCID: PMC2203630.

[13] J.A. Chapman, I. Ho, S. Sunkara, S. Luo, G.P. Schroth, D.S. Rokhsar, "*Meraculous: de novo genome assembly with short paired-end reads*", PLoS One, Vol. 6, No. 8, e23501, 2011, https://doi.org/10.1371/journal.pone.0023501. Epub 2011 Aug 18. PMID: 21876754; PMCID: PMC3158087.

[14] D. Li, R. Luo, C.M. Liu, C.M. Leung, H.F. Ting, et al., "*MEGAHIT v1.0: A fast and scalable metagenome assembler driven by advanced methodologies and community practices*", Methods, Vol. 102, pp. 3-11, 2016, https://doi.org/10.1016/j.ymeth.2016.02.020. Epub 2016 Mar 21. PMID: 27012178.

[15] F. Milicchio and M.C.F. Prosperi, "*HErCoOl: High-Throughput Error Correction by Oligomers*", IEEE International Symposium on Computer-Based Medical Systems, New York, NY, USA, 2014

[16] I. Akogwu, N. Wang, C. Zhang, P. Gong, "*A comparative study of k-spectrum-based error correction methods for next-generation sequencing data analysis*", Hum. Genomics, Vol. 10, Suppl. 2 (Suppl 2):20, 2016, https://doi.org/10.1186/s40246-016-0068-0. PMID: 27461106; PMCID: PMC4965716.

[17] F. Milicchio, I. E. Buchan and M. C. F. Prosperi, "A* fast and scalable high-throughput sequencing data error correction via oligomers," 2016

IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2016, pp. 1-9, doi: 10.1109/CIBCB.2016.7758117.

[18] M. Kokot, M. Dlugosz, S. Deorowicz, "*KMC 3: counting and manipulating k-mer statistics*", Bioinformatics, Vol. 33, No. 17, pp. 2759-2761, 2017, https://doi.org/10.1093/bioinformatics/btx304. PMID: 28472236.

[19] P. Pandey, M.A. Bender, R. Johnson, R. Patro, B. Berger, "*Squeakr: an exact and approximate k-mer counting system*", Bioinformatics, Vol. 34, No. 4, pp. 568-575, 2018, https://doi.org/10.1093/bioinformatics/btx636. PMID: 29444235.

[20] P. Jiang, J. Luo, Y. Wang, P. Deng, B. Schmidt, et al., "*kmcEx: memory-frugal and retrieval-efficient encoding of counted k-mers*", Bioinformatics, Vol. 35, No. 23, pp. 4871-4878, 2019, https://doi.org/10.1093/bioinformatics/btz299.

[21] A. Rahman and P. Medevedev, "*Representation of k-Mer Sets Using Spectrum-Preserving String Sets*", Journal of Computational Biology, Vol. 28, No. 4, pp. 381-394, 2021, https://doi.org/10.1089/cmb.2020.0431.

[22] P. Melsted, J.K. Pritchard, "*Efficient counting of **k**-mers in DNA sequences using a bloom filter*", BMC Bioinformatics, Vol. 12, 2011, https://doi.org/10.1186/1471-2105-12-333.

[23] A.A. Mamun, S. Pal, S. Rajasekaran, "*KCMBT: a k-mer Counter based on Multiple Burst Trees*", Bioinformatics, Vol. 32, No. 18, pp. 2783-90, 2016, https://doi.org/10.1093/bioinformatics/btw345. Epub 2016 Jun 9. PMID: 27283950; PMCID: PMC5939891.

[24] R.S. Roy, D. Bhattacharya, A. Schliep, "*Turtle: identifying frequent k-mers with cache-efficient algorithms*", Bioinformatics, Vol. 30, No. 14, pp. 1950-7, 2014, https://doi.org/10.1093/bioinformatics/btu132. Epub 2014 Mar 10. PMID: 24618471.

[25] M. Frigo, C.E. Leiserson, H. Prokop, and S. Ramachandran, "*Cache-Oblivious Algorithms*", ACM Transaction on Algorithms, Vol. 8, No. 1, Article 4, 2012, https://doi.org/10.1145/2071379.2071383.

[26] P. Van Emde Boas, "*Preserving Order in a Forest in Less Than Logarithmic Time and Linear Space*", *Inf. Process. Lett*., Vol. 6, pp. 80-82, 1977.

[27] M.A. Bender, E.D. Demaine, and M. Farach-Colton, "*Cache-Oblivious B-Trees*", SIAM Journal on Computing, Vol. 35, No. 2, pp. 341–358, 2006, https://doi.org/10.1137/S0097539701389956.

[28] S. Deorowicz, and S. Grabowski, "*Data compression for sequencing data*", Algorithms Mol Biol., Vol. 8, No. 1, 2013, https://doi.org/10.1186/1748-7188-8-25.

[29] A. Döring, D. Weese, T. Rausch, K. Reinert, "*SeqAn An efficient, generic C++ library for sequence analysis*", BMC Bioinformatics Vol. 9, No. 11, 2008, https://doi.org/10.1186/1471-2105-9-11.

[30] F. Milicchio, and M.C.F. Prosperi, "*Efficient data structures for mobile de novo genome assembly by third-generation sequencing*", International Conference on Mobile Systems and Pervasive Computing, MobiSPC, 2017, Leuven, Belgium.

[31] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, et al., "*Energy efficiency across programming languages: how do energy, time, and memory relate?*", ACM SIGPLAN International Conference on Software Language Engineering, 2017. https://doi.org/10.1145/3136014.3136031.

[32] F. Milicchio, G. Tradigo, P. Veltri, and M.C.F. Prosperi, "*High-performance data structures for de novo assembly of genomes: cache oblivious generic programming*", ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, 2016, https://doi.org/10.1145/2975167.2985691.

[33] B.S. Yun, S.G. Lee, and Y. Aoshima, "*An analysis of the trilemma phenomenon for Apple iPhone and Samsung Galaxy*", Serv. Bus., Vol. 13, pp. 779–812, 2019, https://doi.org/10.1007/s11628-019-00405-5.

[34] R. De Virgilio, and F. Milicchio, "*Physical design for distributed RFID-based supply chain management*", Distributed. and Parallel Databases, Vol. 34, pp. 3-32, 2015.