ESWEEK: G: Real-time, Portable and Lightweight Nanopore DNA Sequence Analysis using System-on-Chip

Hasindu Gamaarachchi[1,2,*]
Martin A. Smith[2,1] (supervisor), Sri Parameswaran[1] (supervisor)
*hasindu@unsw.edu.au
[1]University of New South Wales, Sydney, Australia
[2]Garvan Institute of Medical Research, Sydney, Australia

1 PROBLEM & MOTIVATION

The future of healthcare is decidedly dependent upon precision medicine. Precision medicine takes into account the genetic makeup of an individual to develop customised medicines and doses that are effective and safe (Fig. 1). The key to precision medicine is DNA sequence analysis. DNA sequence analysis is also beneficial in fields such as epidemiology, virology, forensics and evolutionary biology. Over the last two decades, DNA sequencing machines have evolved from >500kg machines to pocket-sized devices such as the 87g Oxford Nanopore MinION (Fig. 2). However, software tools that analyse the terabytes of data produced by sequencing machines are still dependent on high-performance or cloud computers, which limits the utility of portable sequencers (Fig. 2).

The ultra-portable MinION sequencer is not constrained to laboratory environments and scientists have performed in-the-field sequencing in exotic locations, for instance, in West Africa during the 2013–2016 Ebola virus outbreak [21], at rural locations in Brazil during the Zika virus outbreak [4], in the middle of jungles, in the arctic [9] and even on the International space station [2]. During the ongoing Novel Coronavirus outbreak (COVID-19), the portable MinION sequencers have proven beneficial especially for small decentralised laboratories around the world [18]. However, the true potential of this portable DNA sequencer is currently limited due to the reliance on high-performance computers for analysis. For instance, the data generated at remote West Africa during Ebola virus sequencing had to be transferred to high-performance computers in Europe for analysis. Data had to be transferred through a mobile Internet connection which was expensive and time-consuming due to connectivity issues [21]. Technologies to perform the analysis in-the-field would have been valuable in such circumstances, not only to reduce cost but also to obtain results quickly, allowing for faster treatment and potentially saving lives.

The major causes behind DNA sequence analysis being performed on high-performance computers are as follows:

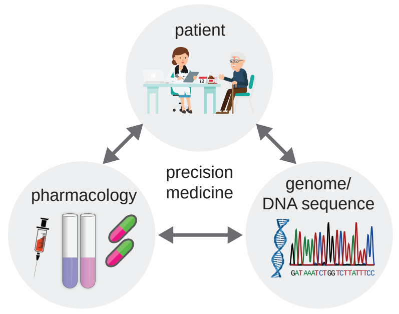


Figure 1: Precision (personalised) medicine. The amalgamation of pharmacology (study of drugs) and genomics (study of DNA sequence) to develop customised treatments that are tailored to the genetic makeup of an individual.

.

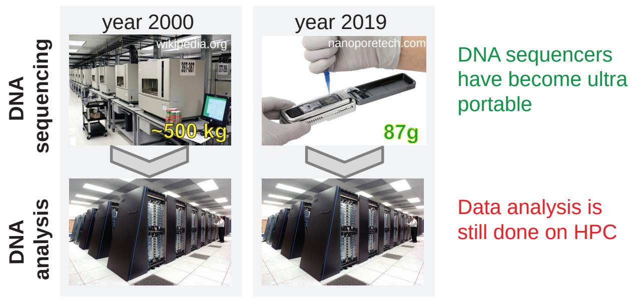


Figure 2: Current state of DNA sequencing and DNA sequence analysis. DNA sequencing has become ultra-portable, however, DNA sequence analysis is still done on supercomputers.

(1) State-of-the-art DNA sequence analysis software tools are typically designed and developed by biologists. Most biologists working in the area of DNA sequence analysis, have access to near-unlimited computational and memory resources (i.e. terabytes of RAM) in their research facilities. Consequently, those software tools are not optimised in terms of efficient resource utilisation.
(2) DNA sequence analysis workflows are extremely complex. A single workflow is a pipeline of a number of extremely complex software tools which are run sequentially. Each tool is a collection of dozens of algorithms, containing numerous heuristically determined parameters. A subtle change in a parameter during an optimisation effort by a computer scientist may severely affect the accuracy, rendering it unacceptable by biologists.
(3) The very few previous research attempts on accelerating and optimising such tools have focused only on sub-components, which are just a fraction of a tool (i.e. Smith-Waterman algorithm). While significant speedups have been recorded for some of these optimised sub-components, those optimisations have rarely been integrated into an actual software tool/workflow, most probably due to a limited benefit to the global efficiency when integrated.

For the first time, we optimise a complete nanopore DNA sequence analysis workflow to execute on portable and lightweight embedded systems. Our work performs the full DNA sequence analysis with terabytes of data on portable devices such as laptops, tablets or mobile phones. Importantly there is no impact on the accuracy of the results.

2 BACKGROUND & RELATED WORK

2.1 DNA Sequencing and Analysis

Deoxyribonucleic acid (DNA) is the blueprint of life which encodes instructions and data for the development and function of a living being. DNA is a molecule made of a long chain of millions of building blocks called *nucleotide bases* (or simply called ***bases***). There are four types of DNA nucleotide bases: adenine (A), cytosine (C), guanine (G) and thymine (T). Every cell of a human being has 23 pairs of DNA molecules (46 in total) which are known as ***chromosomes***. The full nucleotide base sequence of all chromosomes is called the ***genome***. The human genome is around 3.2 gigabases (Gbases). The process of reading fragments of contiguous DNA bases is called ***sequencing***, and the resulting strings of bases are called ***reads***. The machines that perform DNA sequencing are called *sequencers*. Third-generation sequencers are the latest and portable nanopore MinION sequencers belong to this third-generation. Third-generation sequencers produce reads that are of 10-20 kilobases (Kbases) lengths on average (could vary from 1000 bases to >1M), and these reads are known as ***long reads***.

Computational analysis is performed on reads output from the sequencers to discover information of clinical importance. The reads are aligned to a reference genome, a representative example of the genome of the species, and the differences in the reads to the reference (genetic variants) are of interest. Unfortunately, the sequencing machines produce errors (5%-10% in nanopore sequencers [27]) and the subsequent analysis step called ***polishing*** corrects the errors up to 99.8% [11] and identifies genetic variants (referred to as ***variant calling***) or base modifications such as methylated bases (referred to as ***methylation calling***) amongst sequencing errors. The polishing step requires multiple reads covering every position of the reference genome (more than 20X coverage) and the raw sensor output of sequencer (called ***raw signal***) for every read. The data volume of all the reads and associated raw signals for a human genome sample typically exceeds a terabyte.

2.2 An Example DNA Sequence Analysis Workflow

The example nanopore DNA sequence analysis workflow for methylation calling shown in Fig. 3 is a pipeline of five different software tools. Out of those five tools, *Minimap2 alignment* [13] (software 2) and *Nanopolish methylation calling* [24] (software 5) are the two computationally challenging steps (based on our analysis in Section 3.2). The background of those two tools is elaborated in this subsection.

Minimap2 is the gold standard tool amongst the genomics community for aligning long reads to a reference genome. *Minimap2* constructs a hash table (called the ***index***) out of the reference genome which is subsequently accessed very frequently to determine potential locations that a query read maps on the reference genome. Then the read is aligned base-by-base to those identified regions on the reference genome through the application of the dynamic programming alignment algorithm called Suzuki–Kasahara formulation [25]. The alignment scores from the alignment are used in conjunction with many parameters (i.e. the fraction of the read that contains repetitive DNA sequences) to determine the best

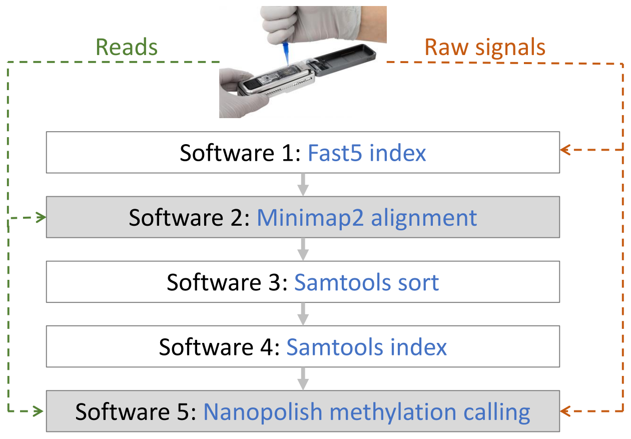

Reads
Raw signals
Software 1: Fast5 index
Software 2: Minimap2 alignment
Software 3: Samtools sort
Software 4: Samtools index
Software 5: Nanopolish methylation calling

Figure 3: An example nanopore DNA sequence analysis workflow for methylation calling.

alignment and a mapping quality score that represents the probability of the alignment accuracy. The hash table data structure itself consumes about 8 GB of memory (RAM). The typically RAM consumption is around 12 GB on average when memory is allocated for internal data structures (i.e. dynamic programming tables). However, the peak RAM can occasionally exceed 16 GB depending on the characteristics of data such as the length of the reads.

Nanopolish is the most popular software package amongst the nanopore community for the polishing step, which is methylation calling in this example. *Nanopolish methylation calling* tool takes the reads, their alignments to the reference genome and the raw signal of each read as the input. Initially, the raw signal is segmented in the time domain based on sudden jumps in the signal and these segments are known as ***events***. The events are then aligned to a hypothetical signal model using an algorithm called *Adaptive Banded Event Alignment (**ABEA**)*. The output of ABEA and alignment details of reads to the reference genome are sent through a Hidden Markov Model (HMM) to detect methylated bases.

2.3 Related Work

Third-generation nanopore DNA sequence analysis is a relatively new field (first nanopore MinION emerged in 2015). Despite the short time, many software tools have been rapidly developed by biologists for nanopore DNA sequence analysis. However, works on accelerating those software or even those exploring computational bottlenecks are very limited. The open source *Clara Genomics* library from NVIDIA [19] and work from Zonghao Feng Et Al. [5] are two examples. *Clara Genomics* accelerates a few sub-components in nanopore DNA sequence analysis workflows (i.e. all-vs-all read mapping and partial order alignment) for *denovo assembly*, the process of assembling a genome from the scratch for a novel species when a high-quality reference is unavailable. Work by Feng Et Al. [5] accelerates core components of Minimap2 alignment software with the simultaneous use of a GPU and an Intel Xeon Phi co-processor, however, the source code is not openly available.

Second-generation DNA sequencing being relatively old compared to third-generation (available for more than a decade), several acceleration efforts have been attempted on associated software. Some examples are: GPU acceleration of the core algorithm called

Smith-Waterman using GPU [15, 17]; GPU accelerated aligners such as SOAP3 [14], BarraCUDA [12], MUMmerGPU [23]; CPU accelerated variant calling tools such as BALSA [16]; FPGA based acceleration of sub-components of tools [1, 10, 20]; and, FPGA based commercial accelerators such as DeCypher [26] and [3]. Nevertheless, tools and characteristic of data related to second-generation DNA sequence analysis are significantly different from third-generation DNA sequence analysis.

3 UNIQUENESS OF THE APPROACH

3.1 Overview of our Method

We optimise a complete nanopore DNA sequence analysis workflow to run on embedded systems/SoC, in contrast to previous work that optimises tiny sub-components. For optimisations, we simultaneously exploit in-depth knowledge of: computer systems (memory hierarchy, interfacing/buses, multi-cores, instruction set architecture, threads, disk caches, virtual memory, etc); and, biology (DNA, chromosomes, genome, repeat regions, read lengths, sequencing errors, sequence alignment, variant calling, methylation calling, etc.); to ensure efficient computational resource utilisation and at the same time the accuracy of the results.

The overview of our method is in Fig. 4. We analyse the workflow and identify the nature of the workloads (CPU intensive, memory-intensive, I/O intensive) in different portions of the workflow (Fig. 4). Then we systematically re-structure the software and optimise bottlenecks to execute on lightweight System-on-Chip equipped with embedded GPU (Fig. 4). Throughout the steps, we synergistically use the characteristics of biological data, associated algorithms, and computer software and hardware architecture for re-structuring and optimising (Fig. 4). Major bottlenecks are resolved via CPU optimisations, parallelisation for GPU architectures, GPU optimisations (exploiting data access patterns for better cache usage and memory coalescing), heterogeneous CPU-GPU work-load balancing, etc. Importantly, our re-structuring and optimisations do not alter the accuracy of the results.

3.2 Applying to an Example Workflow

In this subsection, we demonstrate how the above methodology was applied to the example workflow discussed in section 2.2. Out of the five software tools, our analysis revealed that *Minimap2 alignment* and *Nanopolish methylation calling* are the computationally challenging steps, former being memory intensive (i.e. consuming 12 GB of RAM unavailable on an embedded system) and latter being computationally intensive (i.e. very high runtime). Optimisations we performed to circumvent these challenges are briefly explained below.

As stated in section 2.2, around 8GB of RAM in *Minimap2* is consumed by the index for the reference genome (hash table). We employed a divide and conquer strategy where the index was split into several partitions (number of partitions is decided upon available RAM) and reads were repeatedly mapped to each partition. Then, we proposed and implemented a merging strategy to combine the results from each partition. Note that, naively splitting the index reduce the accuracy to an unacceptable level (i.e. spurious mappings and incorrect mappings qualities). Our unique strategy is based on a careful investigation of causes behind accuracy loss and circumventing those identified causes to achieve the same accuracy as the original software. Explaining those causes and remedies require a detailed background of characteristics of data such as repeat regions and chimeric reads and are not discussed here. The findings have been published at [7].

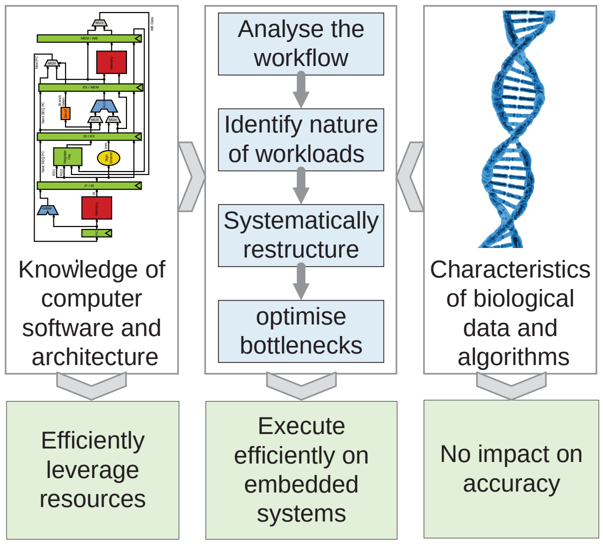

Knowledge of computer software and architecture
Analyse the workflow
Identify nature of workloads
Systematically restructure
optimise bottlenecks
Characteristics of biological data and algorithms
Efficiently leverage resources
Execute efficiently on embedded systems
No impact on accuracy

Figure 4: Simplified overview of methodology. Modern computer systems and DNA sequence analysis workflows are both very complex. The domain knowledge from both is exploited to efficiently utilise computing resources without any impact on accuracy.

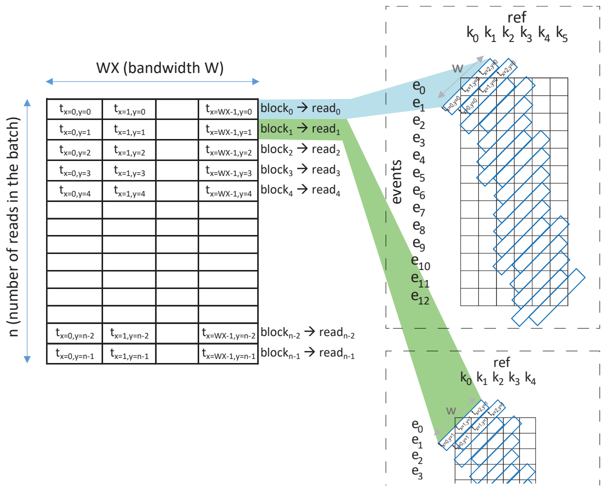

WX (bandwidth W)
n (number of reads in the batch)
$t_{x=0,y=0}$ $t_{x=1,y=0}$ $t_{x=WX-1,y=0}$ $block_0 \rightarrow read_0$
$t_{x=0,y=1}$ $t_{x=1,y=1}$ $t_{x=WX-1,y=1}$ $block_1 \rightarrow read_1$
$t_{x=0,y=2}$ $t_{x=1,y=2}$ $t_{x=WX-1,y=2}$ $block_2 \rightarrow read_2$
$t_{x=0,y=3}$ $t_{x=1,y=3}$ $t_{x=WX-1,y=3}$ $block_3 \rightarrow read_3$
$t_{x=0,y=4}$ $t_{x=1,y=4}$ $t_{x=WX-1,y=4}$ $block_4 \rightarrow read_4$
$t_{x=0,y=n-2}$ $t_{x=1,y=n-2}$ $t_{x=WX-1,y=n-2}$ $block_{n-2} \rightarrow read_{n-2}$
$t_{x=0,y=n-1}$ $t_{x=1,y=n-1}$ $t_{x=WX-1,y=n-1}$ $block_{n-1} \rightarrow read_{n-1}$
ref
k_0 k_1 k_2 k_3 k_4 k_5
W
events
e_0 e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_8 e_9 e_{10} e_{11} e_{12}
ref
k_0 k_1 k_2 k_3 k_4
W
e_0 e_1 e_2 e_3

Figure 5: GPU parallelisation of ABEA algorithm. The GPU *thread grid* (left) and dynamic programming tables of two example reads (right). The GPU processes multiple reads in parallel with a *thread block* assigned to each read. Individual threads in a thread block compute the cell scores (of a band in the dynamic programming table) in parallel.

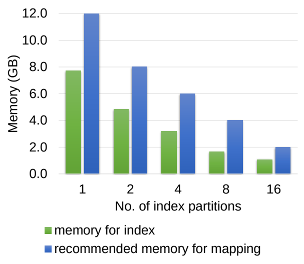
Nanopolish had a very large execution time and our profiling revealed that ~70% of execution time is for the ABEA algorithm.

We parallelised ABEA to run on GPU. ABEA is not an embarrassingly parallel algorithm, however, we carefully analysed data dependencies and restructured the algorithm to expose parallelism to sufficiently occupy GPU cores (Fig. 5). We further improved the GPU performance by studying data access patterns and laying out data for better cache usage and memory coalescing. We also used shared memory (programmer managed cache in GPU) to place frequently accessed data blocks. When aforementioned optimisations led to higher GPU core utilisation, GPU memory management (i.e. *cudamalloc* provided by NVIDIA CUDA runtime) became the next bottleneck, which was remedied by writing a custom lightweight memory manager. The overall performance of ABEA was further enhanced through a heterogeneous CPU-GPU work-load balancing strategy capable of determining the suitability of a given read for CPU or GPU and assigning it appropriately during execution. Next, I/O became a bottleneck and was minimised by interleaving I/O with processing. Note that, original *Nanopolish* was unsuitable for GPU programming paradigm and we had to completely re-engineer the tool, which resulted in separate software tool called *f5c*. More details are in our pre-print [6].

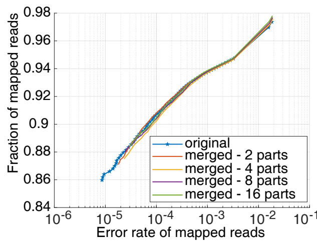
4 RESULTS & CONTRIBUTIONS

4.1 Results of Individual Tool Optimisations

The amount of RAM required in the system to run the original *Minimap2* was ~12GB. This value was reduced to ~8GB by splitting the reference index into two partitions using the strategy presented in section 3.2. By splitting the reference into 4, 8 and 16 parts, RAM requirement could further be reduced to ~6GB, ~4GB and ~2GB respectively (Fig. 6a). Our merging strategy ensured that the accuracy is unaffected despite the number of index partitions, demonstrated by accuracy curves lying on top of the curve for original *Minimap2* in Fig. 6b.



(a) Peak memory reduction

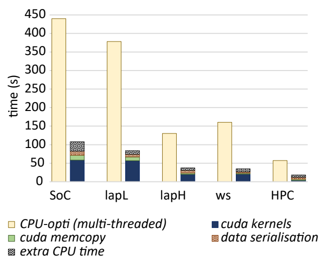


(b) No Impact on accuracy

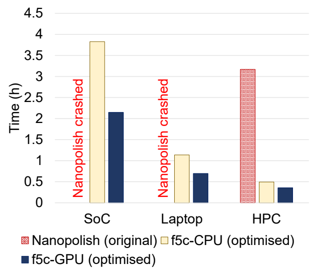
Figure 6: *Minimap2* optimisation results

The computationally expensive ABEA algorithm when parallelised and optimised for the GPU ran ~3-5X faster compared to a heavily optimised multi-threaded CPU version (Fig. 7a). This ~3-5X speedup includes all overheads such as CPU-GPU data transfer, kernels invocations etc. The breakdown of the speedup is shown in Fig. 7a. Note that the CPU implementation of ABEA in Fig. 7a is after it was heavily optimised for efficient multi-core CPU utilisation. The original implementation in *Nanopolish* was not optimised for efficient multi-core execution.

Re-engineered version of *Nanopolish* which includes all our optimisations (termed *f5c*) could successfully run on the Jetson TX2 SoC (8GB RAM) and the laptop (16 GB RAM), while original *Nanopolish* crashed during runtime due to high memory usage (Fig. 7b). In addition to SoC and laptops, the above optimisations equally benefitted HPC where *f5c* was ~9X faster with ~6X peak RAM reduction (Fig. 7b).



(a) Speedup of ABEA



(b) Overall Speedup

Figure 7: *Nanopolish methylation calling* optimisation results. *Soc, lapL, lapH, ws* and *HPC* refer to Jetson TX2, low-end laptop, high-end laptop, workstation and server respectively.

4.2 Results of Overall Workflow

After our optimisations, methylation calling workflow can now be executed on lightweight and inexpensive System-on-Chips (e.g. NVIDIA Jetson Nano module, mobile phone, tablet etc. Note that without our re-structuring and optimisations, it is impossible to run the analysis workflow on such a limited memory system due to peak memory consumption being more than 16 GB (Fig. 8).

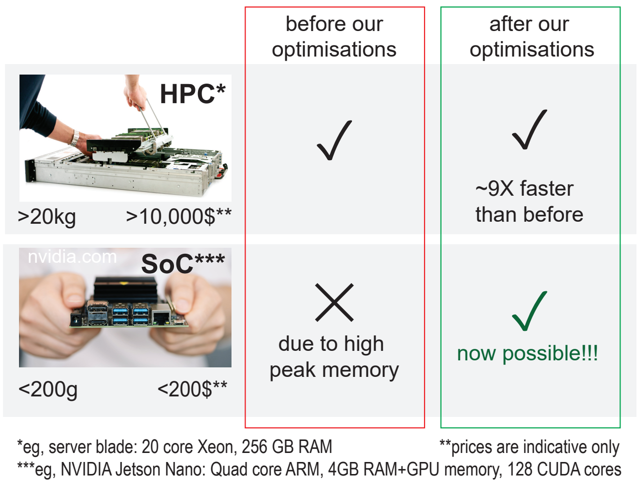


Figure 8: DNA sequence analysis on a SoC is now possible!!

In addition to the execution, it is also demonstrated that the presented strategy allows real-time processing of data output from the nanopore sequencer, on a tiny SoC equipped with an embedded GPU (Fig. 9). That is, optimised system is capable of keeping up with a Nanopore sequencer, processing the data fast enough to produce results by the end of a DNA sequencing run (time that the sequencer operates). On a Jetson Nano SoC (4 GB RAM), the complete methylation workflow on a complete Nanopore MinION dataset executed in less than 35 hours (Fig. 9). This is less than the

48-hour runtime taken by the MinION sequencer to produce the data for use by our system. Thus, if the processing is performed while the sequencer is operating, we could get the full results just after the sequencing run completes.

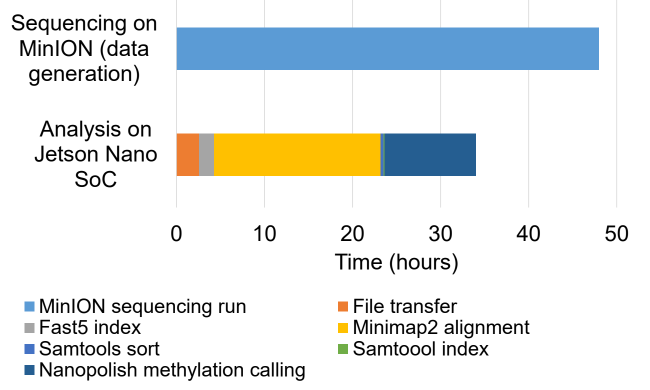


Figure 9: Real-time processing capability

4.3 Contributions and Impact

Our work realises performing a complete Nanopore DNA sequence analysis workflow on an embedded system or an SoC. Such a light-weight and portable, yet accurate embedded DNA analysis device, when connected to a portable Nanopore DNA sequencer, can enable full DNA sequence analysis in-the-field (such as in remote locations without network connectivity) and point-of-care. Embedded systems we designed are fully functional prototypes that they are being actively used at Garvan Institute of Medical Research in Sydney for methylation calling on nanopore samples and have won the best poster award at Australasian Genomic Technologies Association Conference 2019, a major bioinformatics conference. Also, our work has been retweeted by multiple times amongst the bioinformatics community, including Oxford Nanopore and NVIDIA embedded.

Minimap2 is the most popular state-of-the-art long-read aligner and our partitioning and merging based optimisations to *Minimap2* have been integrated into the original GitHub repository. Our strategy has been published in Nature Scientific Reports [8] and has received an Altmetric attention score of 89 and has been featured in 9 news articles. *Nanopolish* is very popular amongst the nanopore community and many CPU optimisations we did have been integrated into the original repository. Our optimised GPU implementation available as open-source at https://github.com/hasindu2008/f5c has received compliments from users who were pleasantly surprised by the 10X performance improvement. These optimisations are in our pre-print at [6]. Also, this work set the foundations to the design and development of the Android Application for nanopore data processing for which the pre-print is available at [22].

REFERENCES

[1] Khaled Benkrid, Ying Liu, and AbdSamad Benkrid. 2009. A highly parameterized and efficient FPGA-based skeleton for pairwise biological sequence alignment. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 17, 4 (2009), 561–570.

[2] Sarah L Castro-Wallace, Charles Y Chiu, Kristen K John, Sarah E Stahl, Kathleen H Rubins, Alexa B R McIntyre, Jason P Dworkin, Mark L Lupisella, David J Smith, Douglas J Botkin, and Others. 2017. Nanopore DNA sequencing and genome assembly on the International Space Station. *Scientific reports* 7, 1 (2017), 18022.

[3] EdicoGenome. [n.d.]. The DRAGEN Engine. http://www.edicogenome.com/dragen-bioit-platform/the-dragen-engine-2/.

[4] Nuno Rodrigues Faria, Ester C Sabino, Marcio R T Nunes, Luiz Carlos Junior Alcantara, Nicholas J Loman, and Oliver G Pybus. 2016. Mobile real-time surveillance of Zika virus in Brazil. *Genome Medicine* 8, 1 (sep 2016), 97.

[5] Zonghao Feng, Shuang Qiu, Lipeng Wang, and Qiong Luo. 2019. Accelerating Long Read Alignment on Three Processors. In *Proceedings of the 48th International Conference on Parallel Processing*. ACM, 71.

[6] Hasindu Gamaarachchi, Chun Wai Lam, Gihan Jayatilaka, Hiruna Samarakoon, Jared T Simpson, Martin A Smith, and Sri Parameswaran. 2019. GPU Accelerated Adaptive Banded Event Alignment for Rapid Comparative Nanopore Signal Analysis. *bioRxiv* (2019). https://doi.org/10.1101/756122

[7] Hasindu Gamaarachchi, Sri Parameswaran, and Martin A Smith. 2019. Featherweight long read alignment using partitioned reference indexes. *Scientific Reports* 9, 1 (2019), 4318.

[8] Hasindu Gamaarachchi, Sri Parameswaran, and Martin A Smith. 2019. Featherweight long read alignment using partitioned reference indexes. *Scientific reports* 9, 1 (2019), 4318. https://doi.org/10.1038/s41598-019-40739-8

[9] Jacqueline Goordial, Ianina Altshuler, Katherine Hindson, Kelly Chan-Yam, Evangelos Marcolefas, and Lyle G Whyte. 2017. In situ field sequencing and life detection in remote (79 26' N) Canadian high arctic permafrost ice wedge microbial communities. *Frontiers in microbiology* 8 (2017), 2594.

[10] Sitao Huang, Gowthami Jayashri Manikandan, Anand Ramachandran, Kyle Rupnow, Wen-mei W Hwu, and Deming Chen. 2017. Hardware Acceleration of the Pair-HMM Algorithm for DNA Variant Calling. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 275–284.

[11] Miten Jain, Sergey Koren, Karen H Miga, Josh Quick, Arthur C Rand, Thomas A Sasani, John R Tyson, Andrew D Beggs, Alexander T Dilthey, Ian T Fiddes, and Others. 2018. Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature biotechnology* 36, 4 (2018), 338.

[12] Petr Klus, Simon Lam, Dag Lyberg, Ming Sin Cheung, Graham Pullan, Ian McFarlane, Giles S H Yeo, and Brian Y H Lam. 2012. BarraCUDA-a fast short read sequence aligner using graphics processing units. *BMC research notes* 5, 1 (2012), 27.

[13] Heng Li. 2018. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* (2018), bty191. https://doi.org/10.1093/bioinformatics/bty191

[14] Chi-Man Liu, Thomas Wong, Edward Wu, Ruibang Luo, Siu-Ming Yiu, Yingrui Li, Bingqiang Wang, Chang Yu, Xiaowen Chu, Kaiyong Zhao, and Others. 2012. SOAP3: ultra-fast GPU-based parallel alignment tool for short reads. *Bioinformatics* 28, 6 (2012), 878–879.

[15] Yongchao Liu, Douglas L Maskell, and Bertil Schmidt. 2009. CUDASW++: optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units. *BMC Research Notes* 2, 1 (may 2009), 73. https://doi.org/10.1186/1756-0500-2-73

[16] Ruibang Luo, Yiu-Lun Wong, Wai-Chun Law, Lap-Kei Lee, Jeanno Cheung, Chi-Man Liu, and Tak-Wah Lam. 2014. BALSA: integrated secondary analysis for whole-genome and whole-exome sequencing, accelerated by GPU. *PeerJ* 2 (2014), e421.

[17] Svetlin A Manavski and Giorgio Valle. 2008. CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment. *BMC bioinformatics* 9, 2 (2008), S10.

[18] NanoporeTech. 2020. Novel Coronavirus (COVID-19): information and updates. https://nanoporetech.com/about-us/news/novel-coronavirus-covid-19-information-and-updates

[19] NVIDIA. 2020. Clara Genomics. https://developer.nvidia.com/Clara-Genomics

[20] Corey B Olson, Maria Kim, Cooper Clauson, Boris Kogon, Carl Ebeling, Scott Hauck, and Walter L Ruzzo. 2012. Hardware acceleration of short read mapping. In *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*. IEEE, 161–168.

[21] Joshua Quick, Nicholas J Loman, Sophie Duraffour, Jared T Simpson, Ettore Severi, Lauren Cowley, Joseph Akoi Bore, Raymond Koundouno, Gytis Dudas, Amy Mikhail, and Others. 2016. Real-time, portable genome sequencing for Ebola surveillance. *Nature* 530, 7589 (2016), 228.

[22] Hiruna Samarakoon, Sanoj Punchihewa, Anjana Senanayake, Roshan Ragel, and Hasindu Gamaarachchi. 2020. F5N: Nanopore Sequence Analysis Toolkit for Android Smartphones. *bioRxiv* (2020). https://doi.org/10.1101/2020.03.22.002030

[23] Michael C Schatz, Cole Trapnell, Arthur L Delcher, and Amitabh Varshney. 2007. High-throughput sequence alignment using Graphics Processing Units. *BMC bioinformatics* 8, 1 (2007), 474.

[24] Jared T Simpson, Rachael E Workman, P C Zuzarte, Matei David, L J Dursi, and Winston Timp. 2017. Detecting DNA cytosine methylation using nanopore sequencing. *Nature methods* 14, 4 (2017), 407.

[25] Hajime Suzuki and Masahiro Kasahara. 2018. Introducing difference recurrence relations for faster semi-global alignment of long sequences. *BMC bioinformatics* 19, 1 (2018), 45.

[26] Timelogic. [n.d.]. DeCypher FPGA Biocomputing Systems. http://www.timelogic.com/catalog/752/biocomputing-platforms.

[27] Ryan R Wick, Louise M Judd, and Kathryn E Holt. 2019. Performance of neural network basecalling tools for Oxford Nanopore sequencing. *Genome biology* 20, 1 (2019), 129.