Memory: The Dominant Bottleneck in Genomic Workloads

Meysam Taassori, Anirban Nag, Keeton Hodgson, Ali Shafiee†, Rajeev Balasubramonian
University of Utah, Samsung Semiconductor Inc†

Abstract

Precision Medicine promises a new wave of advancements in healthcare. Genome sequencing is at the heart of many Precision Medicine approaches. A patient's genome carries the code for protein production, vulnerability to various ailments, and potential treatments. Therefore, genome sequencing is a vital first step in newborn screening, rare disease diagnosis, preemptive treatments, and targeted cancer therapies. Sequencing devices have advanced at rates far better than Moore's Law; modern sequencing devices can produce 1,250 Mbases/minute. However, sequence alignment algorithms on traditional CPUs can process this high volume of information at a meager throughput of 2 Mbases/minute. Software and hardware advancements are therefore required for nearly free and accurate sequence alignment and variant calling algorithms. While many works have attempted to accelerate these computations, we make an argument that a balanced sequence alignment pipeline is almost entirely constrained by memory bottlenecks. Genomic workloads that execute on the cloud have to also be concerned with privacy. Algorithms for privacy typically incur large memory overheads, further amplifying the memory bottleneck. Therefore, this position paper calls for more investment in memory systems research.

1. Introduction

Better understanding of the human genome can lead to significant advancements in healthcare [16]. Genetic analysis can be performed during newborn screening, it can be used to identify risk of disease (and preemptive treatments), and it can be used to create therapies targeted at say specific cancer cells. Just as cheap computation has created a variety of applications/devices and transformed society, nearly free genome analysis will enable a variety of future medical procedures.

Sequencing devices have improved at a rate far greater than Moore's Law [25], with modern Next Generation Sequencing devices producing as many as 1,250 Mbases/minute [13]. The outputs of these devices are then fed to sequence alignment algorithms that consume 20 hours on traditional CPUs, i.e., a throughput of only 2.5 MBases/minute [17]. Some efforts have achieved low-latency sequence alignment by modifying algorithms with various heuristics (that may introduce false negatives) and by employing FPGAs to accelerate key computations [17].

Several studies [26, 19, 11, 3] have pursued similar approaches, using FPGAs and vector instructions to accelerate key computations. While such efforts may be useful in some settings, we may be fast approaching a point where acceleration with better computational devices may yield diminishing returns. Some works [15] reduce the memory bandwidth by moving computation closer to the memory. In our preliminary analysis of the sequence alignment pipeline, we show that for a well-designed alignment pipeline, most of the execution time is spent in stages that perform few computations. These stages are limited by their access to memory; reducing their execution time will therefore require innovative memory systems.

This memory bottleneck is expected to be even more severe when sequence analysis algorithms are executed on shared cloud infrastructures. Genomic data is sensitive and cloud servers will be required to use privacy best practices when handling such data. A state-of-the-art secure system based on Intel's SGX incurs high slowdowns because of the overheads introduced when accessing memory [5, 18]. In addition, as we show here, an SGX-based system does not guarantee privacy and there exist other information side channels. Future systems may be required to close these side channels, further adding to the memory access overhead.

Thus, with the preliminary data in this position paper, we make the case that sequence analysis will be a memory-bound workload, and computational accelerators and FPGAs will have relatively little impact.

2. Background

2.1. Sequence Alignment

The sequence alignment process compares short reads from the sequencers against a reference genome to figure out their mapping locations. It has two different approaches based on how the reference is stored: suffix array or hash table. In our analysis here, we focus on the hash table based pipeline.

Hash-based aligners rely on the pigeon hole principle – if a read has at most e errors, and it is split into $e+1$ non-overlapping regions, then at least one of the regions will match exactly with the reference genome. Accordingly, a read is divided into $e+1$ *seeds* of size k, each of which look up a hash table that stores locations in the reference genome where the seed can be found. These candidate locations are then sent to an edit distance calculation step, where the read is compared against the reference genome fragment at each candidate location. This calculation uses dynamic programming and is computationally expensive.

To reduce invocations of the expensive edit distance calculation step, seeds must be selected such that few candidate locations are passed to the next stage. In addition, these candidate locations are filtered with heuristics that can quickly determine if bases around the seed in the read and in the ref-

erence genome location are dissimilar. Both seed selection and filtering require many memory accesses and relatively few computations.

The sequence alignment pipeline is therefore composed of three stages: (i) Seed Selection, which can be performed with various algorithms – Naive, Hobbes, Optimal [28], (ii) Filtering, which can be performed with algorithms like FastHash [27], SHD [26], and Magnet [4], and (iii) Edit Distance Calculation, performed with the Smith-Waterman algorithm (SWA) [21].

2.2. Variant Calling

Variant calling is a step that analyzes the results of the sequence alignment process. In Variant Calling, (i) machine-introduced errors are cleaned out, and (ii) the genome is compared to a cohort of reference genomes (multiple individuals) to identify disease-causing mutations. This is done with a combination of De novo assembly and Bayesian analysis. At high-variation sites where errors/mutations are suspected, De novo assembly is carried out at small scale amongst those reads. The results of the De novo analysis are fitted into a Bayesian model to figure the probability of an error or mutation at each site.

While our understanding about the relation between different variants and corresponding diseases is restricted, there are at least nine syndromes that can be associated with deletion regions in the genome. For example, the role of APOE $\varepsilon 4$ allele located at chromosome 19 boosts the risk of Alzheimer's [22].

Both sequence alignment and variant calling touch specific regions of the involved data structures with varying numbers of memory accesses. Leaking this information can therefore compromise patient privacy.

2.3. Intel SGX

Intel's Software Guard eXtensions provide basic primitives to protect an application from a variety of attacks, including those from a compromised OS. A sensitive application places its data in an *enclave* and the underlying hardware guarantees confidentiality, integrity, and authentication for the enclave. These guarantees are expensive [1] because sensitive data must be first moved into an Enclave Page Cache (EPC) before access, and EPC look-ups convert a single memory access into multiple memory accesses [5, 18, 8].

However, in spite of these measures, SGX does not close every information side channel. A co-scheduled application can measure latencies to access shared resources like the memory system and estimate the extent of memory activity in other co-scheduled applications (a timing channel). A malicious OS can also swap out an application's pages to detect when the application swaps the page back, thus revealing the trace of pages being touched. Finally, an attacker with physical access to the hardware can monitor the memory bus and identify the exact trace of individual memory blocks being touched. Attacks have been demonstrated for each of these side channels [24, 23, 7, 20, 29, 6, 14].

3. Preliminary Results

While prior works typically focus on one stage of the sequence alignment pipeline, we examine the execution time of the entire pipeline by combining various algorithms for each stage. Figure 1 shows execution time when these packages run on one core of an Intel Xeon at 2.9 GHz with four DDR3 channels and 64 GB of memory. The reference genome was obtained from the Genome Reference Consortium (Build 38) [9] and 10,000 reads were obtained from the 1000 genome project [12].

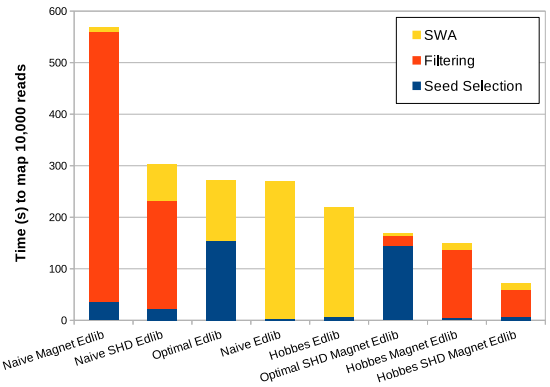

Figure 1: A design space exploration with various algorithms for the different stages of hash-based sequence alignment. Seed selection can use Naive/Hobbes/Optimal, Filtering can use nothing/SHD/Magnet, and SWA uses Edlib.

We observe that bottlenecks shift in these different designs. In particular, performing more work in early stages is helpful because it reduces the burden on the computationally-intensive edit distance calculation. Thus, in a well-balanced pipeline (the one that uses Hobbes for seed selection, SHD and Magnet for filtering), the memory accesses involved in seed selection and filtering are a more significant bottleneck than the computations in the edit distance stage. In other words, an FPGA-based accelerator for edit distance calculation will have a small impact on this optimized baseline.

Next, we show that side channels can leak significant information in variant calling. We perform an experiment to show that the access pattern of a variant calling application can reveal the chromosomes being examined, and therefore potential ailments in the patient. Variant calling is performed with SAMTools [2] on a 16-core Xeon with 24 GB memory and Linux Mint OS. Intel's Pin tool is used to track virtual addresses being touched. Figure 2 shows that the application footprint is very different for each chromosome being analyzed. Thus, a malicious OS that can track pages being touchead or a hardware attacker that can monitor the memory bus can extract patient information with high probability.

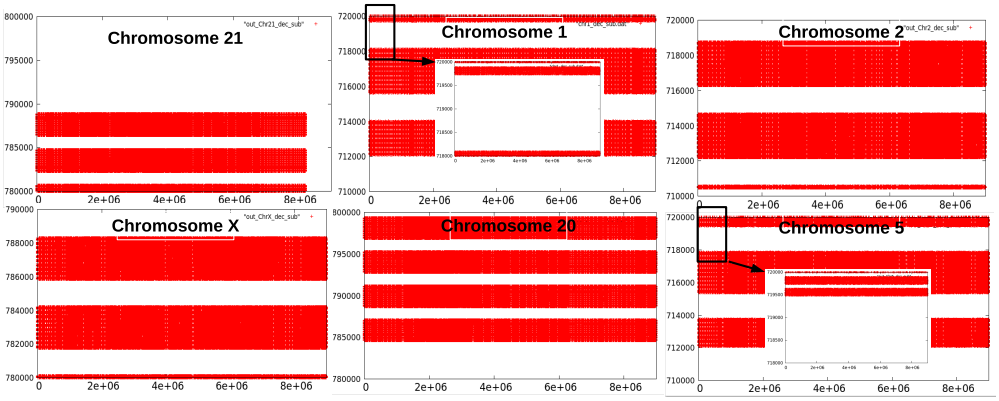

Figure 2: Memory trace of SAMTools when the variant calling tool executes on different chromosomes. In these graphs, the X-axis shows time and the Y-axis shows the accessed virtual address.

To close these side channels, we must rely on techniques that make the access pattern *indistinguishable*. Traditionally, this has been done with ORAM-based solutions [10], which are known to increase memory overheads by orders of magnitude.

4. Conclusions

While accelerators are in vogue today, our preliminary analysis shows that for sequence alignment, the optimal combination of known algorithms yields a pipeline that spends more time in seed selection and filtering – phases that are limited by memory accesses. Further, in a cloud setting, these genomic workloads must be augmented with constructs that preserve privacy – these constructs are known to increase memory overheads by orders of magnitude. To emphasize the need for privacy constructs, we demonstrate leakage through data access patterns in variant calling. Thus, our results highlight that the memory system will be the dominant bottleneck in genomic workloads.

References

[1] "iDASH Privacy and security workshop," http://www.humangenomeprivacy.org/2017/competition-tasks.html.

[2] "SAMTools," http://samtools.sourceforge.net/, 2017.

[3] N. Ahmed, V. Sima, E. Houtgast, K. Bertels, and Z. Al-Ars, "Heterogeneous Hardware/Software Acceleration of the BWA-MEM DNA Alignment Algorithm," in *Proceedings of ICCAD*, 2015.

[4] M. Alser, O. Mutlu, and C. Alkan, "Magnet: Understanding and Improving the Accuracy of Genome Pre-Alignment Filtering," *arXiv preprint arXiv:1707.01631*, 2017.

[5] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. Stillwell *et al.*, "SCONE: Secure Linux Containers with Intel SGX." in *OSDI*, 2016, pp. 689–703.

[6] F. Brasser, U. Muller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A. Sadeghi, "Software Grand Exposure: SGX Cache Attacks Are Practical," *arXiv preprint arXiv:1702.07521*, 2017.

[7] J. Bulck, N. Weichbrodt, R. Kapitza, F. Piessens, and R. Strackx, "Telling Your Secrets without Page Faults: Stealthy Page Table-Based Attacks on Enclaved Execution," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1041–1056.

[8] V. Costan and S. Devadas, "Intel SGX Explained," 2016, https://eprint.iacr.org/2016/086.pdf.

[9] Genome Reference Consortium, "Human Build 38," (retrieved) 2017, https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.37.

[10] O. Goldreich, "Towards a Theory of Software Protection and Simulation by Oblivious RAMs," in *Proceedings of STOC*, 1987.

[11] E. J. Houtgast, V. Sima, K. Bertels, and Z. Al-Ars, "An FPGA-Based Systolic Array to Accelerate the BWA-MEM Genomic Mapping Algorithm," in *International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, 2015.

[12] IGSR, "The International Genome Sample Resource," (retrieved) 2017, http://www.internationalgenome.org/data/.

[13] Illumina, "Illumina Introduces the NovaSeq Series: A New Architecture Designed to Usher in the $100 Genome," https://www.illumina.com/company/news-center/press-releases/press-release-details.html?newsid=2236383, 2017.

[14] M. Islam, M. Kuzu, and M. Kantarcioglu, "Access Pattern Disclosure on Searchable Encryption: Ramification, Attack, and Mitigation," in *Proceedings of NDSS*, 2012.

[15] R. Kaplan, L. Yavits, R. Ginosar, and U. Weiser, "A Resistive CAM Processing-in-Storage Architecture for DNA Sequence Alignment," *IEEE Micro*, vol. 37, no. 4, pp. 20–28, 2017.

[16] M. Might, "An Algorithm for Precision Medicine," http://matt.might.net/articles/algorithm-for-precision-medicine/, 2016.

[17] N. A. Miller, E. G. Farrow, M. Gibson, L. K. Willig, G. Twist, B. Yoo, T. Marrs, S. Corder, L. Krivohlavek, A. Walter *et al.*, "A 26-Hour System of Highly Sensitive Whole Genome Sequencing for Emergency Management of Genetic Diseases," *Genome Medicine - 7*, 2015.

[18] M. Orenbach, P. Lifshits, M. Minkin, and M. Silberstein, "Eleos: Exit-Less OS Services for SGX Enclaves," in *EuroSys*, 2017, pp. 238–253.

[19] D.-H. Park, J. Beaumont, and T. Mudge, "Accelerating Smith-Waterman Alignment Workload with Scalable Vector Computing," in *IEEE International Conference on Cluster Computing*, 2017.

[20] S. Shinde, Z. L. Chua, V. Narayanan, and P. Saxena, "Preventing Page Faults from Telling Your Secrets," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ACM, 2016, pp. 317–328.

[21] T. Smith and M. Waterman, "Identification of Common Molecular Subsequences," *Journal of Molecular Biology*, vol. 147, 1981.

[22] R. E. Tanzi, "The genetics of Alzheimer disease," *Cold Spring Harbor perspectives in medicine*, vol. 2, no. 10, p. a006296, 2012.

[23] W. Wang, G. Chen, X. Pan, Y. Zhang, X. Wang, V. Bindschaedler, H. Tang, and C. Gunter, "Leaky Cauldron on the Dark Land: Understanding Memory Side-Channel Hazards in SGX," *arXiv preprint arXiv:1705.07289*, 2017.

[24] Y. Wang, A. Ferraiuolo, and G. E. Suh, "Timing Channel Protection for a Shared Memory Controller," in *Proceedings of HPCA*, 2014.

[25] K. Wetterstrand, "DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP)," (retrieved) 2017, http://www.genome.gov/sequencingcostsdata.

[26] H. Xin, J. Greth, J. Emmons, G. Pekhimenko, C. Kingsford, C. Alkan, and O. Mutlu, "Shifted Hamming Distance: A Fast and Accurate SIMD-Friendly Filter to Accelerate Alignment Verification in Read Mapping," *Bioinformatics*, vol. 31(10), 2015.

[27] H. Xin, D. Lee, F. Hormozdiari, S. Yedkar, O. Mutlu, and C. Alkan, "Accelerating Read Mapping with FastHASH," *BMC genomics*, 2013.

[28] H. Xin, S. Nahar, R. Zhu, J. Emmons, G. Pekhimenko, C. Kingsford, C. Alkan, and O. Mutlu, "Optimal Seed Solver: Optimizing Seed Selection in Read Mapping," *Bioinformatics - 32*, 2015.

[29] Y. Xu, W. Cui, and M. Peinado, "Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems," in *Proceedings of IEEE Symp. on Security and Privacy (S&P Oakland)*, 2015.