Variant Calling Parallelization on Processor-in-Memory Architecture

Dominique Lavenier, Remy Cimadomo, Romaric Jodin

Variant Calling Parallelization on Processor-in-Memory Architecture

Dominique Lavenier
Univ. Rennes, IRISA / CNRS, INRIA
Rennes - France
lavenier@irisa.fr

Remy Cimadomo, Romaric Jodin
UPMEM
Grenoble - France
rcimadomo,rjodin@upmem.com

***Abstract* - This paper introduces a new combination of software and hardware PIM (Process-in-Memory) architecture to accelerate the variant calling genomic process. PIM translates into bringing data intensive calculations directly where the data is: within the DRAM, enhanced with thousands of processing units. The energy consumption, in large part due to data movement, is significantly lowered at a marginal additional hardware cost. Such design allows an unprecedented level of parallelism to process billions of short reads. Experiments on real PIM devices developed by the UPMEM company show significant speed-up compared to pure software implementation. The PIM solution also compared nicely to FPGA or GPU based acceleration bringing similar to twice the processing speed but most importantly being 5 to 8 times cheaper to deploy with up to 6 times less power consumption.**

Keywords – variant calling; processing-in-memory; PIM; bioinformatics, genomic; I/O disk bandwidth; hardware accelerator; power consumption

I. INTRODUCTION

Variant calling is a fundamental genomic analysis. It consists in detecting, at a DNA level, small differences between two genomes. More precisely, from a pool of short DNA fragments (reads) coming from a specific individual, and obtained using high throughput sequencing, the objective is to locate the place, in a reference genome, where short DNA strings (< 50 bp) differ.

Many software, such as GATK [1], Strelka2 [2], Varscan2 [3], SOAPsnp [4] or DeepVariant [5] have been developed for that purpose. Although these tools have some advantages and disadvantages, they are daily used to identify a large number of specific variations in many health or agronomic application domaines.

Several methods have been proposed to accelerate variant calling by the means of parallel and distributed computing techniques: HugeSeq [6], MegaSeq [7], Churchill [8] and Halvade [9] support variant calling pipelines related to GATK [10]. These parallel implementations exploit the fact that the alignment of one read is independent of the alignment of the others, while the call of variants is independent from one region to another. Other parallel pipelines for variant calling include SpeedSeq [11] and ADAM [12].

Other strategies are based on hardware accelerators. FPGA technologies are particularly well suited to hardwire DNA computation intensive algorithms such as sequence alignments or read mapping. Among recent FPGA systems dedicated to genomic data analysis, the following platforms demonstrate significant speed-up compared to standard GATK software: the Illumina DRAGEN-Bio-IT platform [13] and the WASAI Lightning platform [14]. These FPGA architectures associate both reconfigurable computing resources and memory chips. They provide nice speed-up ranging from 10 to 50 on variant calling applications.

GPU devices offer another alternative to reduce genomic analysis runtime, especially for read mapping which is an important step in the variant calling process [15][16][17][18]. More recently the parallelization of GATK on the NVIDIA Clara Parabricks pipelines [19] achieves a 35-50X acceleration.

This paper explores another way of speeding up the variant calling process using a Processing-in-Memory (PIM) architecture. We present an original parallelization based on new PIM chips developed by the UPMEM company. Actually, PIM architecture is not a new concept. In the past, various research projects have investigated the possibilities to close data and computation. The Berkley IRAM project [20] probably pioneers this kind of architecture to limit the Von Neumann bottleneck between the memory and the CPU. The PIM project of the University of Notre Dame [21] was also an attempt to solve this problem by combining processors and memories on a single chip.

The variant calling task perfectly illustrates how such time-consuming applications can benefit from the PIM architecture. The mapping step, which represents a large part of the overall computation time, is particularly well suited, as fine grained parallelization can be efficiently executed to perform multiple independent alignments along the whole reference genome. Deporting this activity directly to the PIM-DRAM module, and parallelizing the whole process to hundreds of PIM cores, avoids a lot of CPU-memory transactions compared to a standard multithreaded solution.

The objective of the research work presented here, is to precisely evaluate the potentialities of a PIM architecture composed of a bunch of UPMEM DIMM modules, coupled to

the main computer memory bus, on a critical genomic treatment. A generic variant calling algorithm, called upVC, has been implemented as a testbed on real PIM components to provide exact measurement and fair comparison with existing systems in terms of speed-up, energy consumption and cost. From a quality point of view, the upVC implementation is not intended to immediately compete with mature software such as GATK.

II. UPMEM Architecture Overview

UPMEM's PIM technology consists of thousands of parallel coprocessors (called DPU) within the main memory of a host CPU (e.g., x86, ARM64, or Power9). Standard and UPMEM DIMMS can coexist on a server to operate both regular processing and PIM. The CPU provides programming instructions to DPUs, and collects their results as they operate individually. This design relieves the CPU from a memory bottleneck and greatly reduces the energy hungry data movement.

The UPMEM DDR4 2400 DIMM (dual ranks) comprises 16 PIM enabled chips totaling 128 DPU. A DPU is a 32-bit processor running at 500MHz. Up to 20 UPMEM DIMMs can be plugged into a x86 platform, keeping 2 slots per socket for traditional DRAM. The solution scales with the ability to increase the number of DPU in a system and can reach 5120 DPUs in a quadri socket platform with 40 PIM DIMMs .

A PIM memory chip contains 8 DPUs. Each DPU is associated with 64MB of DRAM shared with the host CPU. The calculations happen on chip and within each unit with a memory bandwidth of 1GB/s. This is why PIM imposes data locality and parallelism with the consequence to alleviate the need for important data movements.

A DPU is a 24 threads, 32-bit RISC processor – with 64-bit capabilities – working at 500Mhz with an ISA close to traditional ARM or RISC-V equivalent processors, making it easily programmable. DPUs have a 64KB of WRAM (Working RAM) and a 24KB instruction memory, called IRAM, that can hold up to 4,096 48-bit encoded instructions. DPUs are independent from each other and run asynchronously.

Every DPU can be programmed individually or in groups while orchestrated through the host code. The PIM architecture sits on an efficient toolchain centered around a LLVM based C-compiler using LLVM v10.0.0 and with Linux drivers for x86 servers. It also contains a full-featured runtime library for the DPU, management and communication libraries for host to DPUs operations and a LLDB based debugger. This experiment has been achieved using the SDK v2020.3.0 [27].

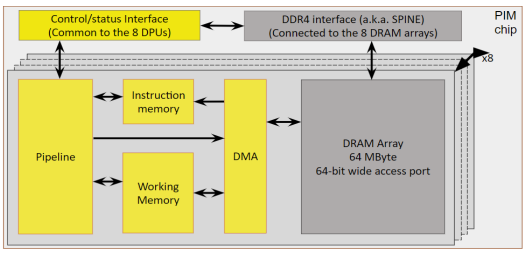

Figure 2: UPMEM PIM chip diagram

III. Experiment

The upVC software has been tested on a 1280 DPU prototype system running at 266 MHz on Human Genome. Details of the implementation is given in a companion paper [28].

In order to have a ground truth to validate the correct functionality of upVC, a simulated sequencing dataset has been generated using the HG38 Human reference genome from which 3,153,377 small variants have been inserted. Paternal and maternal chromosomes have been generated using the vcf2diploid tool [22]. Short Illumina paired-end reads have been generated with the ART read simulator [23] with the following parameters: insert size = 400; standard deviation = 50; read length = 150bp; coverage = 30X. The resulting dataset contains 586 x 10^6 reads split into two fastq files. With reads of length 150bp, the index size for the human genome is equal to 120 GBytes.

To estimate the variant calling quality, we measure the following metrics: True Positive (TP), i.e. existing variants found by upVC; False Negative (FN), i.e. variants not found by upVC; False Positive (FP), i.e. non existing variants found by upVC; The following table summarizes the quality.

	substitution		insertion		deletion	
	upVC	**GATK**	**upVC**	**GATK**	**upVC**	**GATK**
TP %	99.77	100	99.10	99.69	99.57	99.98
FN %	0.50	0.23	0.91	0.74	0.57	0.37
FP %	0.23	0	0.90	0.31	0.43	0.02

GATK has been run with standard parameters. Compared to upVC, the quality is clearly better. However, upVC provides excellent results and legitimates our variant calling implementation on PIM architecture, knowing that the current code has plenty of room for improvement.

The experiments have been done on an Intel® Xeon® Silver 4110 CPU @ 2.1 Ghz, 8 cores with 64 GBytes of RAM, equipped with 10 additional UPMEM double-rank DIMM devices with DPU running at 266 MHz. Of a total of 1280 available DPUs (10 x 128), only 1024 full operational DPUs have been used. The available PIM memory size is thus equal to 64 GBytes (64 Mbytes per DPU).

The complete execution time to perform the variant calling is equal to 11048 sec. (~184 minutes). Due to its limit amount of memory, it requires 3 passes, each one working on a subset of the bank index.

Extrapolating the real results obtained from this prototype to a 5120 system running at 600 MHz, the target frequency of the next generation of DPUs gives an idea of tomorrow PIM potentiality. Schematically, with 4 times more DPUs running at a frequency 2.3 times higher, the execution time will be divided by 9 (4 x 2.25), leading to decrease the total execution time to 20 minutes.

IV. COMPARISON WITH ALTERNATIVE SYSTEMS

We compare the performances of the upVC PIM implementation with two other hardware accelerators: Illumina DRAGEN using proprietary software on 8 FPGA Xilinx UltraScale Plus 16 nm FPGA [13], and Nvidia Parabricks using BWA-GATK4 on 8 NVIDIA®Tesla®V100 GPUs [19].

To provide a fair comparison, we extend performance results to different PIM configurations with increased density and DPU frequency. At the time of writing, the reference platform available at UPMEM is a 2*Xeon Silver 4108 with 128GB RAM and 160 GB of PIM memory with 2560 DPUs clocked at 400MHz. Servers with higher PIM DIMM density such as the Cooper Lake 4* Xeon Gold 6328H with 5120 DPUs and the AMD ARM Epyc with 3584 DPUs are in the process of qualification while DPUs clocked at 500MHz and more are under development.

Figure 4 reports the execution time of the different systems to process a typical variant calling operation on a 30X human genome dataset. The loading of the reference genome in MRAM is not considered as part of the computation time if enough PIM memory in a system allows single batch runs of upVC. In this case, the reference genome loading process only happens at the start of the server and can be used for all subsequent sample analysis.

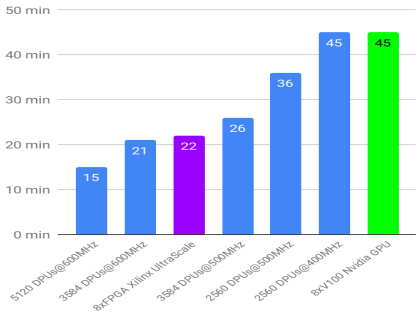

Figure 4: Execution time based on 30X human genome dataset on FPGA, GPU and PIM

A 2560 DPUs configuration does not allow enough space in MRAM to load the entire genome and simultaneously retain enough space to save the reads in MRAM. One DPU has 45,5 MB available to store reference neighbors, which totalizes 116,5 GB with 2560 memory banks, slightly less than the 120 GB of the reference genome. To avoid the need for 2 batches and consequently load 2 halves of the reference genome with long HDD transfers we divide the input read buffer by 2. This way we free enough space for the reference genome but still has for consequence to double the DPU processing time on this configuration. Naturally we observe a gain in performance once the memory space issue is alleviated in 3000+ DPUs configurations.

Figure 5 gives the power consumption of FPGA, GPU and PIM systems. The consumption of an FPGA board depends on its configuration. For this workload it is estimated to be used near maximum capacity at 320W per board, 90% of its TDP. The consumption of a V100 GPU is provided by Nvidia and reaches around 300W in full use. UPMEM provides precise measurements of a DPU power consumption and depends on its version and clocking. At 400Mhz, a DPU in current version v1.2 consumes 160 mW, while it consumes 190 mW at 500MHz. DPUs at 600MHz are benefiting from energy reduction designs and are expected at around 120mW. The overall consumption accounts that the charge of DPUs can hardly go over 90% during the entire execution and that every PIM module consumes 3W. PIM based configurations are in average 6x less energy consuming than the considered alternative accelerators.

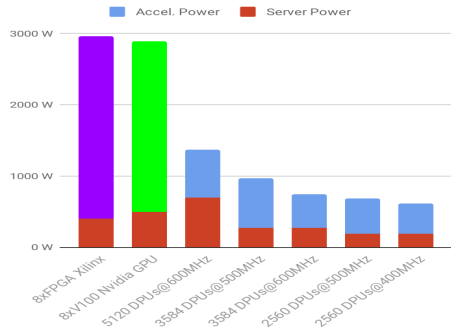

Figure 5: Power consumption (hardware accelerator + server) of PIM, GPU (8 NVIDIA®Tesla®V100*) and FPGA (8* FPGA Xilinx UltraScale*) systems.*

The consumptions of the server for PIM configurations is based on the 2*Xeon Silver 4110 with 128GB RAM. The TDP is given at 190W. An AMD 2*Epyc has a TDP around 280W and a Cooper Lake with 4* Xeon Gold has a TDP of 700W. We consider a 2*Xeon Gold 6328H server base for alternative accelerators to ensure efficient orchestration at TDP of 400W. 3W per 8 GB of memory accounts for the DRAM consumption.

Server and infrastructure costs follow the comprehensive AWS TCO cost estimator [24]. The estimator accounts for an annual maintenance of 15% of the hardware cost. The same logic if applied to each of the considered accelerator's hardware. The consumption evaluation is based on previous energy considerations for a full 3 years and accounts for a cooling and infrastructure overhead (additional 70% of the hardware consumption). The cost of electricity is based on median US commercial price [25]: $0,1/kWh.

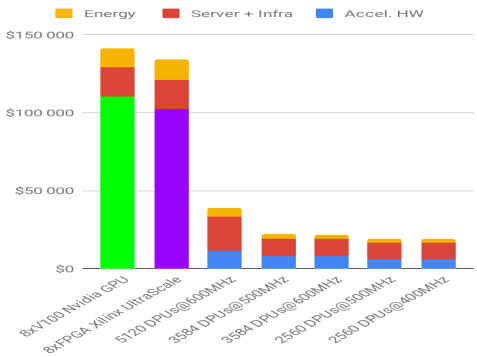

Figure 6: years TCO Comparison between PIM configurations against GPU and FPGA based accelerators for Mapping+Variant calling.

A full 3 years on premise FPGA based solution's TCO nears $135,000, which accounts for FPGA board at around $8,800 per unit. It is about 5 times more expensive than UPMEM PIM solution at identical throughput for a 4096 DPUs at 500MHz configuration. Note that if we were to consider the software cost for running Illumina Dragen, an additional $572,000 would be required over a period of 3 years, multiplying the cost reduction made by UPMEM solution by yet again a factor 5.

Nvidia quotes its V100 GPU at $9,500 per unit resulting in a full 3 years TCO of Nvidia Parabrick estimation over $140,000. In terms of algorithms, they are identical to BWA-GATK4, using DNA-Bricks to port them over GPU architectures and do not represent an overhead cost to use the solution. At equivalent throughput it is about 8 times more costly than UPMEM PIM FASTQ to VCF using 4096 DPUs at 500MHz.

Thus, UPMEM technology offers a drastic financial and environmental gain compared to both Nvidia and Illumina solutions. Though it does not reach an as high accuracy, development efforts on upVC are progressively narrowing the gap.

V. CONCLUSION

This implementation demonstrates the performance of the PIM architecture when dedicated to a large scale and highly parallel task in genomics: every DPU independently computes read mapping against his fragment of the reference genome while the variant calling is pipelined on the host.

The algorithm works well within the confines of the experiment but still remains a long way from a real-world application with a holistic alignment strategy. It is a prototype that verifies the capabilities of a PIM architecture in the context of mapping and variant calling. The low CPU usage of this implementation allows additional CPU based functions to complete the pre-variant calling workflow that would pave the road towards a commercial application.

In comparison to existing accelerators, the PIM solution promises to deliver equal to better performances but with massive energy reduction and TCO gains. It is a crucial advantage in sight of the prominent place that genomics is about to occupy in the world of data computing and for its accessibility by medical institutions across the globe. A configuration with 3584 DPUs at 600MHz has the best TCO profile and could bring the cost of human genome analysis near $0,34/genome.

PIM is a promising technology that shows a great potential to solve some of the challenges of genomics in terms of actionable computing power, programmability and cost.

REFERENCES

[1] Van der Auwera, G. A. *et al*. From FastQ data to high confidence variant calls: the Genome Analysis Toolkit best practices pipeline. *Curr Protoc Bioinformatics* **43**(11), 10.1–33 (2013).

[2] Kim, S. *et al*. Strelka2: fast and accurate calling of germline and somatic variants. *Nat Methods* **15**, 591–594 (2018).

[3] Koboldt, D. C., Larson, D. E. & Wilson, R. K. Using VarScan 2 for Germline Variant Calling and Somatic Mutation Detection. *Curr Protoc Bioinformatics* **44**(15.4), 1–17 (2013).

[4] Li R, Li Y, Fang X, Yang H, Wang J, Kristiansen K. SNP detection for massively parallel whole-genome resequencing. Genome Res. 2009;19(6):1124–1132.

[5] Ryan Poplin, Pi-Chuan Chang, David Alexander, Scott Schwartz, Thomas Colthurst, Alexander Ku, Dan Newburger, Jojo Dijamco, Nam Nguyen, Pegah T. Afshar, Sam S. Gross, Lizzie Dorfman, Cory Y. McLean, and Mark A. DePristo. A universal SNP and small-indel variant caller using deep neural networks. Nature Biotechnology 36, 983–987 (2018).

[6] Lam HYK, Pan C, Clark MJ, Lacroute P, Chen R, Haraksingh R, et al. Detecting and annotating genetic variations using the HugeSeq pipeline. Nature Biotechnology. 2012 Mar;30(3):226–229

[7] Puckelwartz MJ, Pesce LL, Nelakuditi V, Dellefave-Castillo L, Golbus JR, Day SM, et al. Supercomputing for the parallelization of whole genome analysis. Bioinformatics. 2014 Jun;30(11):1508–1513

[8] Kelly BJ, Fitch JR, Hu Y, Corsmeier DJ, Zhong H, Wetzel AN, et al. Churchill: an ultra-fast, deterministic, highly scalable and balanced parallelization strategy for the discovery of human genetic variation in clinical and population-scale genomics. Genome biology. 2015 Jan;16(1)

[9] Decap D, Reumers J, Herzeel C, Costanza P, Fostier J. Halvade: scalable sequence analysis with MapReduce. Bioinformatics. 2015 Mar;31(15):2482–2488

[10] McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, et al. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome research. 2010 Sep;20(9):1297–1303

[11] Chiang C, Layer RM, Faust GG, Lindberg MR, Rose DB, Garrison EP, et al. SpeedSeq: ultra-fast personal genome analysis and interpretation. Nature Methods. 2015 Aug;12(10):966–968.

[12] Nothaft F. Scalable Genome Resequencing with ADAM and avocado. UC Berkeley; 2015. Technical Report no UCB/EECS-20IS-6S.

[13] Illumina DRAGEN Bio-IT Platform v3.2.8. User Guide. 2019

[14] https://www.wasaitech.com/genomics

[15] Y. Liu, B. Schmidt, D. Maskell: CUSHAW: a CUDA compatible short read aligner to large genomes based on the Burrows-Wheeler transform, Bioinformatics, (2012) 28(14): 1830-1837

[16] Y. Liu, B. Schmidt: CUSHAW2-GPU: empowering faster gapped short-read alignment using GPU computing. IEEE Design & Test of Computers 31(1):31-39, 2014

[17] Klus P, Lam S, Lyberg D, Cheung MS, Pullan G, McFarlane I, Yeo GSH, Lam BY. (2012) BarraCUDA - a fast short read sequence aligner using graphics processing units. BMC Research Notes, 5:27.

[18] Langdon WB, Lam BY, Petke J, Harman M. (2015) Improving CUDA DNA Analysis Software with Genetic Programming. Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation - GECCO '15

[19] https://www.nvidia.com/en-us/docs/parabricks/

[20] Patterson, D., Anderson, T., Cardwell, N., Fromm, R., Keeton, K., Kozyrakis, C., Thomas, R., and Yelick, K. (1997). "A Case for Intelligent RAM: IRAM," *IEEE Micro,* **17** (2), pp. 34–44

[21] https://sdk.upmem.com/

[22] Kogge, P. M., T. Sunaga and e. a. E. Retter (1995). Combined DRAM and Logic Chip for Massively Parallel Applications. 16th IEEE Conf. on Advanced Research in VLSI, Raleigh, NC

[23] Rozowsky J. et al. AlleleSeq: analysis of allele-specific expression and bin ding in a network framework. Mol Syst Biol. 2011

[24] Huang, Weichun et al. "ART: a next-generation sequencing read simulator." *Bioinformatics (Oxford, England)* vol. 28,4 (2012): 593-4. doi:10.1093/bioinformatics/btr708

[25] https://awstcocalculator.com/

[26] U.S. Energy Information Administration's Electric Power Monthly report

[27] https://sdk.upmem.com/2020.3.0/

[28] D. Lavenier, E. Jodin, R. Cimadomo, Variant Calling Parallelization on Processor-in-Memory Architecture, bioRxiv, https://www.biorxiv.org/content/10.1101/2020.11.03.366237v1