Enabling Faster NGS Analysis on Optane-based Heterogeneous Memory

Jiaolin Luo
jluo38@ucmerced.edu
University of California, Merced

Luanzheng Guo
lguo4@ucmerced.edu
University of California, Merced

Jie Ren, Kai Wu
{jren6,kwu42}@ucmerced.edu
University of California, Merced

Dong Li
dli35@ucmerced.edu
University of California, Merced

ABSTRACT

Next-Generation Sequencing (NGS) analysis technologies are a pioneer approach for genome sequencing. The computation of NGS analysis exhibits a unique pattern, in which the execution requests a high density of small I/Os in the process *de novo* genome assembly. The small I/Os can have a huge impact on performance and delineate the sequencing performance. To solve the problem caused by small I/Os, we leverage the byte-addressable feature of emerging persistent memory, which has largely transformed the computation architectures of HPC. We first conduct experiments to study the impact of persistent memory on NGS analysis. Furthermore, we propose an optimization mechanism, which converts POSIX read/write calls to pure memory LOAD/STORE instructions at runtime, to significantly advance the I/O efficiency. Our evaluation demonstrates the effectiveness of the optimization mechanism, in which we achieve a performance improvement by 31%.

1 INTRODUCTION

NGS analysis, e.g., the state-of-the-art Sentieon software [1], drives scientific researches in multifarious disciplines, in particular for biology and clinical medicine researches. In the process *de novo* assembly of NGS analysis such as WGS and WES, the execution issues a large number of consecutive and small I/Os with the sizes of a few cache lines (as illustrated in Figure 1), in which the I/O performance poses a bottleneck of the execution of NGS analysis pipelines.

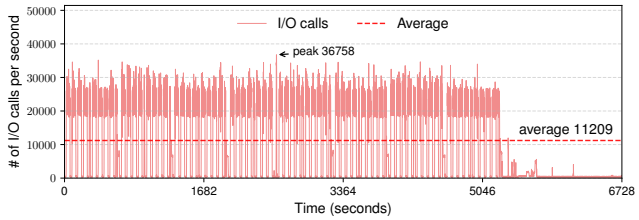

Figure 1: The number of read and write IO calls in NGS analysis.

One fundamental design assumption of Linux I/O stack – consisting of page cache, vfs layer, file systems, block layer and block drivers – is that the backing storage devices are block-addressable, in which the I/O granularity is known as sector and the data accesses are through the DMA mechanism transferring sectors to or from page cache. Such assumption is incorrect if the backing devices are byte-addressable and without the need of transferring data through DMA mechanism. In order to request DMA I/Os from or to devices and maintain file system metadata, existing Linux file I/O mechanisms (e.g., POSIX APIs *read* and *write*) involve in switching processor ring level from user context to kernel context, which can further incur extra overhead on I/O path. As a result, existing I/O stack will obscure the advantages of persistent memory and poses a bottleneck while employing persistent memory into HPC architectures.

Emerging persistent memory has profound impact on established HPC architectures and have revolutionized the design of HPC software from multiple aspects. We seek to leverage the benefits of emerging persistent memory (e.g., Intel's Optane DC) to optimize the efficiency of I/O stack in NGS analysis. First, persistent memory is byte-addressable and can transfer data without the involvement of DMA operations. Second, in terms of per GB price, the price of Intel's Optane DC is merely 50% of server level DRAM memory. Third, the available capacity of persistent memory can easily reach up to 3TB per server. Fourth, the memory access latency of persistent memory still remains comparable to DRAM memory.

The benefits of persistent memory enable HPC applications to run with larger and cheaper extended memory. Nevertheless, the performance improvement of NGS analysis from simply configuring persistent memory as faster local storage is infinitesimal, in which the execution is I/O intensive (as illustrated in Figure 1). The overhead of the I/O stack in operating system hides the benefits of persistent memory.

We first conduct experiments to answer the following concerns in NGS analysis:

- Q1: Is I/O a bottleneck?
- Q2: What is the impact of persistent memory?
- Q3: Why I/O is slow on persistent memory?

In addition, we exploit the byte-addressable feature of persistent memory to shorten the length of I/O stack in NGS analysis through converting POSIX read/write calls to memory LOAD/STORE instructions at runtime.

Table 1: Experiment Platform Specifications

Software	Ubuntu 18.04 with Linux kernel version 5.4.24
Sentieon	Version 201911
WGS fastq reads	Project RM8398 of NA12878
WGS reference genome	HG38
Processor	Intel® Xeon® Gold 6252N CPU @ 2.30GHz
DRAM	six 16-GB DDR4 DIMMs × 2 sockets (192 GB in total)
PM	six 128-GB Optane DC NVDIMMs × 2 sockets (1.5 TB in total)
Storage	Intel® D3-S4510 SSD 480GB

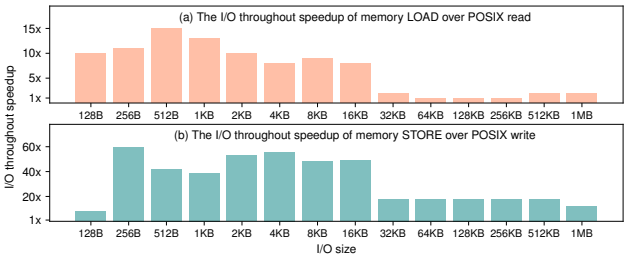

Figure 2: The I/O throughout speedup of memory LOAD/STORE instructions over POSIX read/write with respect to different IO sizes.

2 CHARACTERIZATION

We conduct experiments – denoted as A, B and C – to answer the three concerns (Q1-Q3) of Section 1, in which the experiment setup is illustrated in Table 1.

Experiment A: Is I/O a bottleneck? We use Linux *perf* to collect the statistics of I/O system calls in state-of-the-art NGS analysis software Sentieon.

Figure 1 shows the trends of the number of POSIX read/write calls across the execution. The dense I/O spikes range from 20000 to 30000 calls per second and approach to the maximum random small IOPS of installed SSD.

The high I/O numbers in Figure 1 and analysis demonstrate the I/O stack poses a bottleneck in the execution of NGS analysis.

Experiment B: The impact of Optane on execution time. To mitigate the impact from I/O stack as shown in experiment A, will that make any difference if we simply store files on faster storage such as DRAM memory disk or Optane memory disk? The answer is negative.

We evaluate the Sentieon performance on DRAM-only and Optane-only respectively, where DRAM and Optane are configured as main memory accordingly. In addition, we also evaluate the Sentieon performance while sequencing data and intermediate files are on local Solid State Disk (SSD), DRAM memory disk and Optane memory disk respectively.

Table 2 shows the impact of persistent memory Intel's Optane DC on the execution of NGS analysis. We have two findings: (1) The execution of Optane as main memory is 2.7 times slower than the execution of DRAM as main memory; (2) The execution time of Optane as local storage, DRAM as local storage and SSD as local storage has no patent performance difference.

One reasonable explanation might justify the findings is that the Linux I/O stack is inefficient as we have analyzed in Section 1. We will further explore the causes in next experiment.

Experiment C: Why I/O is slow on persistent memory? We devise a micro benchmark to compare the I/O throughput speedup of pure memory LOAD/STORE instructions over traditional POSIX read/write calls.

Table 2: The impact of Optane on the performance of Sentieon.

Main memory	Data storage	Execution time (minutes)
DRAM	SSD	120
DRAM	DRAM memory disk	120
DRAM	Optane memory disk	120
Optane	SSD	320
Optane	DRAM memory disk	320
Optane	Optane memory disk	320

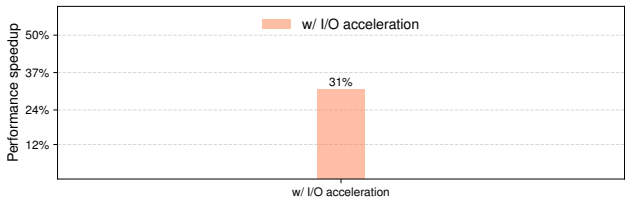

Figure 3: The performance speedup with Optane based I/O acceleration, in which the baseline is the performance without Optane based I/O acceleration.

Figure 2 shows the I/O throughput speedups of pure memory LOAD/STORE instructions over traditional POSIX read/write. The I/O throughput speedups are evaluated with I/O sizes from $128B$ to $1MB$. We have two observations: (1) For read, the I/O throughput speedup is up to 15 times; (2) For write, the I/O throughput speedup is up to 60 times.

The results suggest the inefficient I/O stack will hide the benefits of persistent memory and justify the findings in Experiment B.

3 OPTIMIZATION

As we have known from Section 1 and Section 2, the overhead of operating system I/O stack hides the benefits of persistent memory. To shorten the I/O stack length, we introduce an approach of converting POSIX read/write calls to pure memory LOAD/STORE instructions at runtime.

In the design, file-related calls (e.g., *open*, *close*, *lseek*, *read* and *write*) are re-interposed via the technique LD_PRELOAD and handled in our optimization library at runtime. When file I/O calls are from those files stored on persistent memory, the dynamic library will remap those files to the address space of processes and convert POSIX read/write calls to memory LOAD/STORE instructions.

4 EVALUATION

The evaluation environment is shown in Table 1. We compare the performance speedup of the execution time of Sentieon sequencing pipeline with I/O acceleration, in which the baseline is the execution time without I/O acceleration.

Figure 3 demonstrates the effectiveness of our Optane-based I/O acceleration technique with a performance speedup by 31%.

5 CONCLUSIONS

We first explore the impact of persistent memory (Intel's Optane DC) on I/O intensive NGS analysis software (Sentieon) and understand its performance characterization via experiments. Our study reveals the inefficiency of Linux I/O stack can pose a bottleneck in NGS analysis.

In addition, we design an optimization method by shortening the length of Linux I/O stack via converting POSIX read/write calls to pure memory LOAD/STORE instructions. The evaluation demonstrates such optimization can make a huge performance improvement in NGS analysis.

REFERENCES

[1] K. I. Kendig, S. Baheti, M. A. Bockol, T. M. Drucker, S. N. Hart, J. R. Heldenbrand, M. Hernaez, M. E. Hudson, M. T. Kalmbach, E. W. Klee, et al. Sentieon dnaseq variant calling workflow demonstrates strong computational performance and accuracy. Frontiers in genetics, 10, 2019.

Enabling Faster NGS Analysis on Optane-based Heterogeneous Memory

Jiaolin Luo, Luanzheng Guo, Jie Ren, Kai Wu & Dong Li
University of California, Merced

Motivation

The computation of Next-Generation Sequencing (NGS) analysis exhibits a unique pattern, in which a large number of consecutive and dense small I/Os is issued. The small I/Os can have huge impact on performance. We have the following motivations to explore the niche of persistent memory in NGS analysis and solve its I/O problem:

- **Large capacity**.
  The available capacity of DRAM as main memory is limited to few hundreds of GB, while the available capacity of persistent memory Intel's Optane DC can reach up to $1.5-3$TB per server.
- **Affordable price**.
  The price of per GB persistent memory is a half of DRAM, in future, the price will be even cheaper.
- **Byte-addressable**.
  The byte-addressable feature can enable faster I/O performance skipping the Linux I/O stack.

We seek to leverage the benefits of persistent memory. We conduct a scientific research to answer the following concerns:

- **Q1: Is I/O a bottleneck in NGS analysis?**
- **Q2: What is the impact of persistent memory in NGS analysis?**
- **Q3: Why I/O is slow on persistent memory?**

Furthermore, we design an optimization technique to solve the I/O problem in NGS analysis by shortening the length of I/O stack in persistent memory. Our evaluation demonstrates the effectiveness of our solution, in which we achieve a performance improvement by 31%.

Characterization

Experiment A: Is I/O a bottleneck?. Figure 1 shows the trends of the number of POSIX *read/write* calls. The result implies I/O is a bottleneck in NGS analysis.

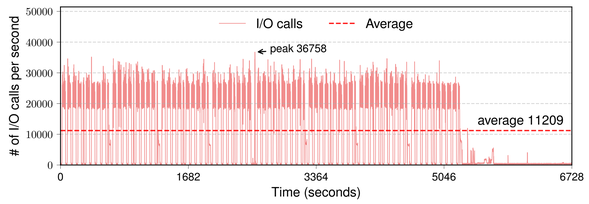

Figure 1: The number of read and write I/O calls in NGS analysis.

Experiment B: What is the impact of persistent memory?. Table 1 shows the impact of persistent memory on the execution of NGS analysis. We have two findings: (1) The execution time of Optane as main memory is 2.7 times slower than of DRAM as main memory; (2) Optane has no improvement at I/O performance if simply configured as local storage.

Table 1: The impact of Optane on the performance of Sentieon.

Execution memory	Data storage	Execution time (minutes)
DRAM	SSD	120
DRAM	DRAM memory disk	120
DRAM	Optane memory disk	120
Optane	SSD	320
Optane	DRAM memory disk	320
Optane	Optane memory disk	320

Experiment C: Why I/O is slow on persistent memory?. Figure 2 shows the results of the I/O throughput speedup of memory LOAD/STORE instructios over POSIX read/write calls.

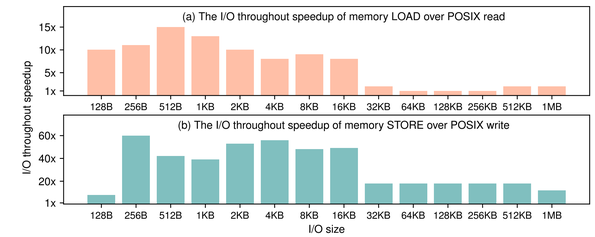

Figure 2: The I/O throughput speedup of pure memory LOAD/STORE instructions over traditional POSIX *read/write* calls.

Optimization

Small dense I/Os poses a performance bottleneck of NGS analysis and simply configuring Optane as either main memory or local storage makes no difference.

We devise an optimization method by shortening the length of I/O stack via converting POSIX read/write calls to pure memory instructions.

Evaluation

Figure 3 demonstrates the performance speedup of Optane-based I/O optimization technique. The result shows the execution speedup is up to 31% in NGS analysis.

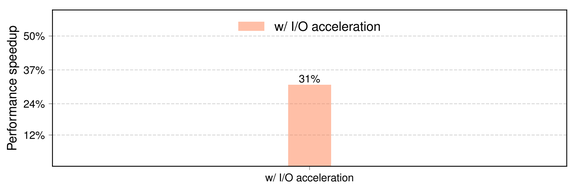

Figure 3: The normalized execution time with and without Optane-based I/O optimization technique.

Conclusions

We have the following conclusions:

- The execution of NGS analysis is bound to small and dense I/Os.
- Simply configuring Optane as either main memory or local storage has no performance improvements.
- The overhead of Linux I/O stack hides the benefits of persistent memory.
- Our design of converting POSIX read/write calls to pure memory LOAD/STORE instructions at runtime can have huge performance improvement by up to 31%.