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 observations of the execution of Next-Generation Sequencing (NGS) analysis:

- **The execution is I/O intensive (a huge number of small and consecutive I/Os)**.
- **The memory footprint is not large (from few GBs to tens of GBs)**.
- **Using memory disk does not have performance improvement.**.

The benefits of persistent memory on HPC architectures:

- **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 existing 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?**
- **Q4: How do we optimize NGS analysis with persistent memory?**

Optimization

We harness the byte-addressable feature of persistent to optimize the I/O efficiency of existing Linux I/O stack:

- **Re-interposing POSIX operations through LD_PRELOAD technique**.
  POSIX calls, e.g., *read*, *write*, *lseek*, *open* and *close*, will be handled in our library at runtime.
- **Converting POSIX read/write calls to pure memory LOAD/STORE instructions**.
  In this case, the existing Linux I/O, which is designed for block-addressable devices, will be skipped.

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.

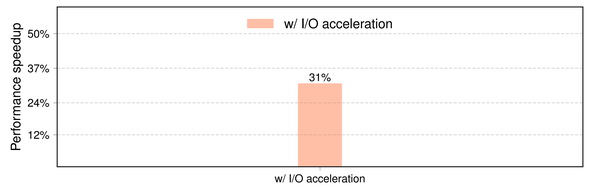

Figure 3: The normalized execution time with and without Optane-based I/O optimization mechanism.

Characterization

Experiment A: Is I/O a bottleneck?. Figure 1 shows the trends of the number of POSIX read/write calls, which approaches the maximum IOPS of local storage. The result suggests I/O efficiency poses a bottleneck in NGS analysis.

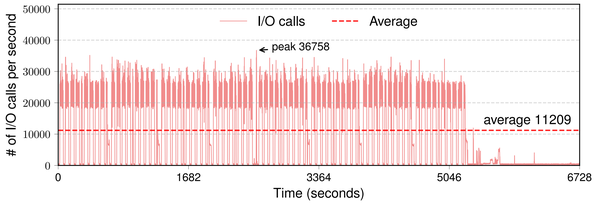

Figure 1: The number of read and write I/O calls in NGS analysis.

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 I/O performance improvement if simply configured as local storage.

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

Execution memory	Data storage	Average 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 speedups of memory LOAD/STORE instructios over POSIX read/write calls with respect to various I/O sizes.

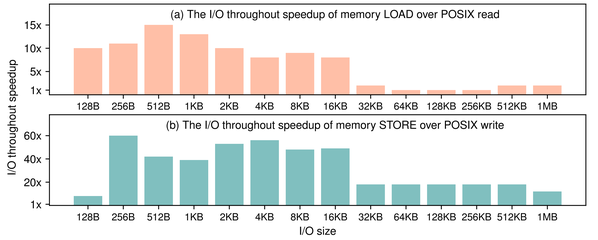

Figure 2: The I/O throughput speedup of pure memory LOAD/STORE instructions over traditional POSIX read/write calls.

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 existing Linux I/O stack hides the benefits of persistent memory**.
- **The optimization mechanism of converting POSIX read/write calls to pure memory LOAD/STORE instructions is proved to be effective**.
- **Our design of converting POSIX read/write calls to pure memory LOAD/STORE instructions at runtime can have a huge performance improvement by up to** 31%.

We need further study to explore the best use of heterogeneous memory in NGS analysis. For instance, how we can cache hot I/Os on DRAM memory while cold I/Os on persistent memory.