A Near-Storage Framework for Boosted Data Preprocessing of Mass Spectrum Clustering

Weihong Xu Jaeyoung Kang Tajana Rosing
University of California, San Diego
{wexu,j5kang,tajana}@ucsd.edu

ABSTRACT

Mass spectrometry (MS) has been a key to proteomics and metabolomics due to its unique ability to identify and analyze protein structures. Modern MS equipment generates massive amount of tandem mass spectra with high redundancy, making spectral analysis the major bottleneck in design of new medicines. Mass spectrum clustering is one promising solution as it greatly reduces data redundancy and boosts protein identification. However, state-of-the-art MS tools take many hours to run spectrum clustering. Spectra loading and preprocessing consumes average 82% execution time and energy during clustering. We propose a near-storage framework, MSAS, to speed up spectrum preprocessing. Instead of loading data into host memory and CPU, MSAS processes spectra near storage, thus reducing the expensive cost of data movement. We present two types of accelerators that leverage internal bandwidth at two storage levels: SSD and channel. The accelerators are optimized to match the data rate at each storage level with negligible overhead. Our results demonstrate that the channel-level design yields the best performance improvement for preprocessing - it is up to 187× and 1.8× faster than the CPU and the state-of-the-art in-storage computing solution, INSIDER, respectively. After integrating channel-level MSAS into existing MS clustering tools, we measure system level improvements in speed of 3.5× to 9.8× with 2.8× to 11.9× better energy efficiency.

CCS CONCEPTS

• **Hardware → Emerging architectures**; **Memory and dense storage**; **Application specific integrated circuits**; • **Computer systems organization → Parallel architectures**.

KEYWORDS

Near-storage computing, Mass spectrometry, domain-specific acceleration

ACM Reference Format:
Weihong Xu, Jaeyoung Kang, and Tajana Rosing. 2022. A Near-Storage Framework for Boosted Data Preprocessing of Mass Spectrum Clustering. In *Proceedings of Proceedings of ACM Conference (DAC '22).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3489517.3530449

DAC '22, July 10–14, 2022, San Francisco, CA, USA

ACM ISBN 978-1-4503-9142-9/22/07.
https://doi.org/10.1145/3489517.3530449

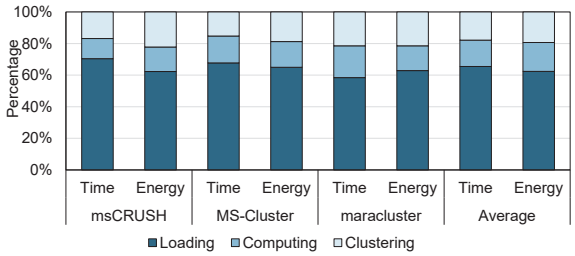

Figure 1: Execution time and energy breakdown for various mass spectrum clustering tools, msCRUSH [29], MS-Cluster [11], and *MaRaCluster* [26]. Preprocessing = Loading + Computing.

1 INTRODUCTION

Four key "omics" disciplines drive precision medicine today: genomics, transcriptomics, proteomics, and metabolomics. While genomics and transcriptomics rely on sequence alignment (DNA and RNA sequences), proteomics and metabolomics depend on mass spectrometry (MS). Due to its powerful capabilities of identifying molecular structures, an increasing number of fields exploit MS-based proteomics and metabolomics to unravel the components and structures underlying proteins and cells [10, 12, 18, 23].

A popular approach to obtain spectra data is shotgun proteomics [18], where mass spectrometers analyze samples and acquire millions of fragment spectra in hours. Processing spectra data is the most time-consuming part during experiments [26] since the same molecules may be scanned by mass spectrometer many times, creating many similar and redundant spectra fragments. The high data redundancy severely degrades the efficiency of the MS analysis pipeline, such as open search [10]. Various clustering algorithms and tools [11, 13, 26, 29] have been developed to reduce the data redundancy through clustering similar spectra and selecting representatives of each cluster for protein and peptide identification. The clustering step not only decreases the overall analysis time but also improves the identification quality [11, 29].

However, state-of-the-art MS clustering tools are too slow to tackle the exponential growth of MS data. *MS-Cluster* [11] and *spectra-cluster* [13] take nearly 30 hours to cluster a dataset with 25M spectra [6], far behind the over-gigabyte hourly data generation speed of modern mass spectrometers [26]. The number of spectra data submission to one of the largest public mass spectra datasets, PRIDE [20], has increased over 10× during the past ten years. The MassIVE [27] database has stored 4×10^9 publicly accessible spectra with over 300 terabytes (TB). Due to the enormous size of these datasets, clustering is done only a few times a year, resulting in less accurate search results. To this end, new tools are essential for both clustering and search of mass spectrometry data

to accelerate the identification of proteins, which are critical for the development of new medicines.

Our profiling in Fig. 1 shows that the spectra preprocessing step (includes loading and computing) is the major bottlenecks that account on average 82% of execution time and energy of the clustering pipeline. The speed of spectra loading and computing is restricted by the costly data movement and limited bandwidth between host memory and storage. MS tools need to fetch the bulky spectra data from storage devices to the host memory before performing the computing step on the CPU. Even with PCIe and NVMe [3] techniques, the peak read speed of commercial SSD storage [1] (3.2GB/s) is insufficient compared to spectra data over tens of GBs. The other issue is that only a tiny portion of spectra are preserved after preprocessing. The current pipeline still loads the entire unprocessed dataset into memory, leading to unnecessary time and data movement overhead. Unfortunately, existing DRAM-based accelerators, such as MEDAL [15] and RAPID [14], are not suitable for MS preprocessing because they are optimized for reducing data movement cost near DRAM. They are unsuitable to efficiently process spectra and remove the inherent redundancy before fetching data from the storage without dramatically increasing the DRAM capacity and cost. Thus, MS preprocessing should be accelerated in storage as it has the low cost and high capacity needed to prepare the MS data.

In this work, we propose a near-storage architecture, MSAS, to accelerate MS preprocessing. The key contributions in this work can be summarized as follows:

- To the best of our knowledge, MSAS is the first near-storage design to boost MS spectra preprocessing. We exploit the internal bandwidth of SSD by conducting a design space exploration at the SSD level and channel level. The results indicate that the channel-level acceleration yields the best hardware and energy efficiency.
- We develop a fully pipelined accelerator with scalable performance for each storage level. We identify the top-k selector as the bottleneck of MSAS and develop an efficient top-k selector based on modified Bitonic algorithms to match the internal bandwidths.
- We extensively compare MSAS with state-of-the-art MS clustering tools on various datasets. We obtain up to 187× speedup on spectra preprocessing tasks. Furthermore, as compared to the state-of-the-art in-storage computing prototype [24], MSAS is up to 1.8× faster. After integrating MSAS into state-of-the-art clustering tools, 2.8× to 11.9× energy efficiency and 3.5× to 9.8× speedup are achieved on clustering workloads.

2 BACKGROUND ON MS AND SSD

2.1 Mass Spectrometry

A wide variety of fields, such as analytical chemistry and large-scale proteomics, have adopted MS as a powerful tool to identify biological structures or chemical compounds. A mass spectrometer is used to generate spectra data containing the mass-to-charge ratio (m/z) and ion signal intensity of molecules. As depicted in Fig. 2 (b), a mass spectrum can be considered as a plot of ion signal intensity and mass-to-charge ratio [30], where the ion strength is expressed as its intensity in y-axis against their m/z in the x-axis. Each peak in a spectrum stands for a component with unique m/z in the tested sample. Throughout this work, we adopt the commonly used Mascot generic format (MGF) [22] as the storage format of spectra.

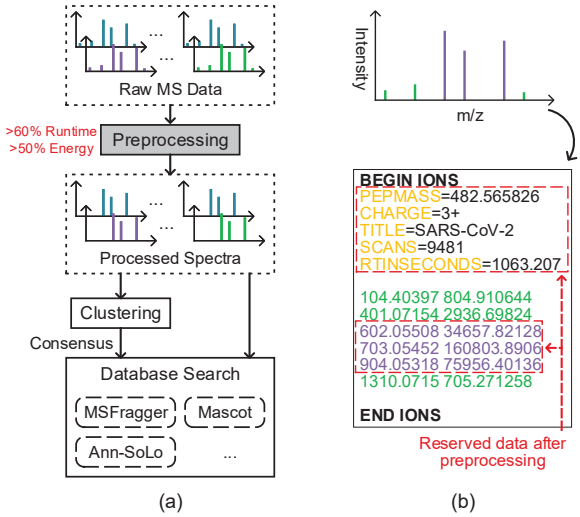

Figure 2: (a) Pipeline of data analysis for MS, (b) A spectrum example in MGF format.

MGF consists of metadata and spectra data, such that metadata record the precursor m/z, peptide charge, query title, and other information. In contrast, the spectra data we use consists of the peak intensity and m/z pairs. Fig. 2 (b) shows a spectrum with total six intensity and m/z pairs.

Fig. 2 (a) illustrates the data analysis pipeline for MS. Researchers can leverage the spectra data to identify molecules and analyze inherent properties through matching the discovered spectra against all the peptides in sequence databases using search engines, like Mascot [22]. Preprocessing step, including filtering, intensity selection, and data normalization, is essential for the subsequent processes since it improves the quality of the results. The filtering step filters the precursor-related peaks and peaks out of the target range, thus reducing noise interference. After the filtration, the intensity selection step finds and preserves the k most intensive peaks. It further reduces the impact of trivial peaks. The typical k value is from 30 to 50 [11, 29]. Data normalization is the final step to lessen the dominant effects of excessive values through additional transformations. The spectra preprocessing is critical for the final analysis quality as demonstrated in [32]. The above three steps are widely used in clustering and search tools [11, 13, 29]. Moreover, the preprocessing compresses spectra size and reduces data redundancy.

2.2 Modern SSD

The architecture of modern SSDs is shown in Fig. 3 [1, 8, 19]. The internal SSD is organized into multiple-level hierarchies, such as channels, chips, planes, to name a few. The most frequently used non-volatile storage elements in SSDs are NAND flash memory [31]. The NAND flash chips are organized into 4 to 32 channels, and each channel operates independently and simultaneously [8]. The flash memory controller (FMC) is implemented to perform dedicated data access and error correction for each channel. Several NAND flash chips share one channel, and these NAND flash chips can issue I/O requests independently. Normally 4 to 8 chips are connected to single-channel [19]. Each NAND chip consists of multiple dies, where each die contains multiple planes, blocks, and pages [8].

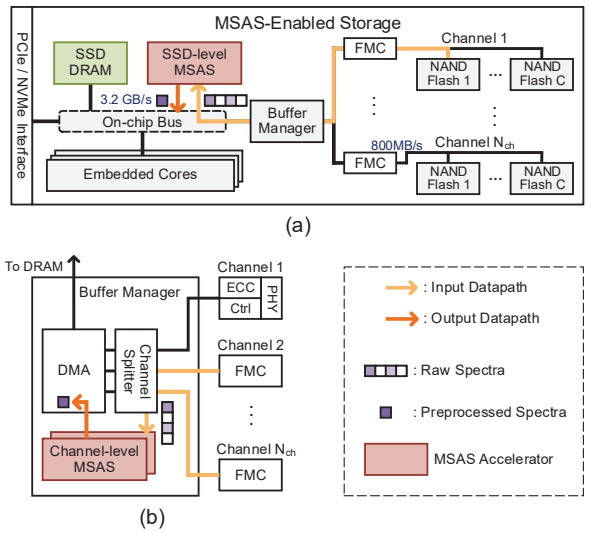

Figure 3: Overall diagram of MSAS accelerators embedded in regular SSD, including two types of designs in different storage levels: (a) SSD-level design, (b) Channel-level design in the buffer manager.

Page with 4 to 16KB size is the smallest unit that can be accessed and indexed in SSDs. The embedded cores are responsible for SSD control, including issuing I/O commands from the external interface and data scheduling. Modern SSDs normally adopt the high-speed NVMe [3] interface to ensure over GB/s external data rate. However, the internal multiple-level hierarchy provides significantly higher data parallelism [4, 8].

3 MSAS ARCHITECTURE

3.1 Overview

Fig. 3 illustrates the overall architecture of MSAS in generic SSDs. MSAS-enabled SSD storage preserves the regular SSD datapath and can additionally boost MS data processing. The limited external bandwidth of SSD may become the bottleneck for data-intensive workloads. However, the highly parallelized SSD internal architecture provides opportunities to alleviate the bandwidth bottleneck [8]. To this end, we create and evaluate two independent designs, namely SSD-level and channel-level MSAS accelerators, to exploit the internal bandwidth of SSD. Accelerators in different storage levels use scalable hardware configurations to provide sufficient throughput at the cost of reasonable overhead. For the two designs in Fig. 3, they follow a similar execution flow: the raw spectra are fetched from NAND flash through the input datapath and then computed in MSAS accelerators. The preprocessed spectra results are temporarily cached in the buffers. When the buffers are full or ready, the output datapath transfers processed spectra in buffers through the regular SSD read datapath to host memory. The difference between the two designs is they are exposed to different internal bandwidths and physical address space.

SSD-level Design: Fig. 3 (a) shows the topmost SSD-level MSAS accelerator that resides in the SSD. The SSD-level accelerator is implemented using CMOS technology on the same die of SSD's embedded cores. It is connected to the global on-chip bus and fetches data from the NAND flashes through the regular SSD datapath.

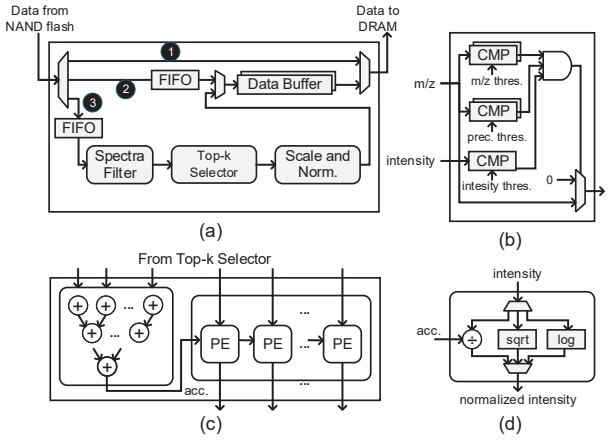

Figure 4: (a) Architectures of MSAS accelerator, ❶: regular SSD read datapath, ❷: metadata loading, ❸ datapath for *m/z* and intensity preprocessing, (b) Spectra filter, (c) Scale and normalization module, (d) Processing element (PE).

Thus it can access the entire physical address space of back-end NAND flashes and enjoy a peak bandwidth that equals to the external SSD bandwidth (*e.g.* 3.2GB/s [1]). The SSD-level accelerator helps to save over 95% SSD's external bandwidth through the NVMe interface; over 95% of data after preprocessing step are discarded due to the redundancy of spectra data.

Channel-level Design: To improve the overall efficiency, the channel-level accelerator in MSAS aggregates the bandwidth of multiple channels as shown in Fig. 3 (b). In contrast, DeepStore [19] extends the internal bandwidth by implementing accelerators in each channel. However, this approach has two defects. First, only 800MB/s rate can be exposed to the accelerator in each channel. Second, implementing accelerators in every channel incurs a large hardware overhead. Our channel-level design resides between FMC and DMA engine, where each channel has 800MB/s bandwidth [2]. In generic SSD architecture, the buffer manager is connected to all N_{ch} channels, and it is responsible for transporting data from channel bus to DRAM buffer [4]. We add a channel splitter [9] between FMC and DMA engine to multiplex the channel data bus. The channel-level MSAS accelerator is implemented within the buffer manager, and it receives data from the channel splitter and sends processed results to DMA. We can tune the internal bandwidth exposed to the channel-level design by choosing different splitting factors of the channel splitter. Assume each accelerator shares C_{share} channel buses, the highest available bandwidth to each accelerator is $(800 \cdot C_{\text{share}})$MB/s. In this case, total $(N_{\text{ch}}/C_{\text{share}})$ channel-level accelerators need to be implemented. Meanwhile, the physical address space of shared C_{share} channels is accessible for each accelerator. Section 4.1 gives the chosen parameter C_{share}.

3.2 MSAS Accelerator

Fig. 4 (a) gives the architecture of MSAS accelerator for performing spectra loading and preprocessing. Throughout this work, we use 32-bit single floating-point as the spectra data format processed by MSAS accelerators. The original data reading of SSD is preserved in datapath ❶. Datapath ❷ and datapath ❸ are used for spectra loading and preprocessing. When a new spectrum is coming, the

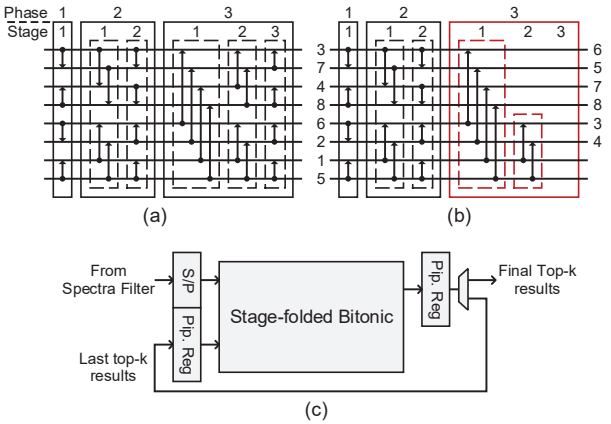

Figure 5: (a) Full Bitonic sorting network with $N = 8$, (b) Simplified Bitonic network for Top-k selection ($N = 8$ and $k = 6$), (c) Iterative Top-k selector for streaming data.

metadata is first cached into the data buffer through datapath ❷. The processing for *m/z* and intensity are performed in datapath ❸ after the metadata loading is finished. Three processing modules, including spectra filter, top-k selector, and scale and normalization module, are implemented over ❸. These modules are executed in a fully pipelined manner to ensure high throughput.

Data Buffer: Multiple data buffers are implemented in each MSAS accelerator, and each data buffer has the same size as SSD's page (8KB). After the preprocessing steps, data buffers work as plane registers to transfer data to SSD DRAM. The page-size data buffers align with SSD's physical addresses since the page is the minimum data chunk that can be indexed. Moreover, we implement double data buffers and interleave the data access of data buffers to avoid clock stalls caused by pipelining.

Spectra Filter: *m/z* and intensity pairs are fed into the spectra filter in Fig. 4(b). The spectra filter requires only a single clock cycle to perform a total of five comparisons. The first two comparators determine whether the given *m/z* value is located within the range of *m/z* threshold. The third and fourth comparisons with the precursor threshold are performed to discard those spectra near the precursor peptide. The last comparison is for filtering the spectra with peak intensity less than the intensity threshold. Finally, the *m/z* and intensity pairs that satisfy all the comparison criteria are allowed to pass to the top-k selector.

Top-k Selector: The top-k selector finds the k most intensive peaks of the streaming *m/z* and intensity pairs from the spectra filter. Classic sorting algorithms like Bitonic sort and quick sort can solve the top-k problem by (1) sorting all the data and then (2) selecting the k largest values. However, the straightforward Bitonic or quick sort is inefficient and superfluous in practical use since the top-k selector does not require strict sorting. Moreover, the number of streaming intensities could be over hundreds, thus finding the top-k values in such data volume would incur huge hardware overhead.

We simplify the original Bitonic algorithms for efficient top-k selection. Fig. 5(a) illustrates an example of 8-point Bitonic sorting network composed of $\log_2 N$ phases where the i-th phase contains i stages. The last phase is called *merge phase*. There are $N/2$ compare and swap (CS) in each stage. The required number of CS units is $O(N \log^2 N)$ with $O(\log^2 N)$ latency. We would obtain sorted results after the data pass through the whole network. The original Bitonic network can be simplified to efficiently support top-k selection [25] as shown in Fig. 5(b). The basic idea is the remove those CS units in the merge phase that will not impact the top-k results. Assuming a top-k problem $k = 6$ with Bitonic network $N = 8$, the last stage together with the upper part of the second stage in the merge phase can be removed to obtain correct top-k results. In this case, the latency is reduced from 6 to 5 and the required CS units decrease from 24 to 18.

Using a fully parallel Bitonic network would introduce prohibitive hardware overhead. Instead, we construct a stage-folded iterative Top-k selector in Fig. 5(c). A single Bitonic stage is reused, and the top-k results are computed iteratively. Specifically, the generated top-k results will be latched in the output pipeline register. Then k inputs are reserved for the last top-k results while the rest $N - k$ inputs are used to receive new input from the spectra filter. The last top-k results and the new data from the spectra filter are sent to the next round of selection. During the period of top-k computation, the serial to parallel register (S/P) is continuously caching new *m/z* and intensity pairs. The new top-k results are sent to the next computation with new cached data in the S/P register. Considering the typical k value is 30 up to 100 for MS preprocessing [6, 11], we choose $N = 128$ for channel-level and SSD-level accelerators to support these k values. This configuration delivers a peak throughput of 5.8GB/s for $k = 100$.

Scale and Normalization Module: The k most intensive peaks from the top-k selector need to be scaled and normalized to eliminate the dominant effect of large intensity. The scale and normalization module supports three commonly used functions, including log scaling, square root scaling, and unit normalization. For unit normalization, the k peak intensities are scaled to $(0, 1]$ with the accumulation value of peaks. The accumulation of all k intensities are computed in the stage-pipelined adder tree in Fig. 4(c). In turn, the normalized values are computed by the divider in the processing elements (PE). The log and square root normalizations are computed by the PE in Fig. 4(d).

3.3 Data Mapping Scheme in MSAS

The available physical address space varies for different MSAS accelerators. There are two basic requirements for spectra data mapping to maximize bandwidth and hardware utilization. First, they need to be stored in a page-aligned manner. Second, the data of continuous spectrum should be stored in continuous space. For the SSD-level accelerator, the entire SSD address space is accessible. Thus, the generic SSD page allocation scheme can work as the SSD-level data mapping scheme. Each channel-level design is exposed to the address space of continuous C_{share} channels. Thus, the spectra data should be evenly allocated to each channel group composed of C_{share} channels.

4 EVALUATION

4.1 Methodology

Baselines: We evaluate four state-of-the-art spectrum clustering tools, including *MS-Cluster* [11], *spectra-cluster* [13], *MaRaCluster* [26] and *msCRUSH* [29]. The fragment mass tolerance and precursor mass tolerance are set to 0.05 Da and 20 ppm, respectively. The most 50 intensive peaks are preserved. We filter those peaks whose

Table 1: Spectra Datasets for Evaluation

MS Spectra Datasets					
Class	**Sample Type**	**PRIDE ID**	**# Spectra**	**Avg. Len.**	**Size**
PXD-Tiny	Kidney cell[10]	PXD001468	1.1×10^6	≈181	5.6GB
PXD-Small	Kidney cell[23]	PXD001197	1.1×10^6	≈816	25GB
PXD-Medium	HeLa proteins[7]	PXD003258	4.1×10^6	≈509	54GB
PXD-Large	HEK293 cell[12]	PXD001511	4.2×10^6	≈798	87GB

m/z is out of range 200 to 2000. *MS-Cluster* is set to run three rounds of clustering and uses the integrated LTQ_TRYP model. *spectra-cluster* is tested under its *fast mode*. The other options and parameters are set to default. On top of the comparison to the CPU-based implementation, we show the performance improvement of MSAS over the state-of-the-art in-storage computing solution INSIDER [24].

Spectra Datasets: Table 1 summarizes the used datasets at different data scales and average lengths. We classify the datasets into four categories based on their size. The data are publicly available and downloaded from the PRIDE repository [28]. All raw data are converted into MGF format using ThermoRawFileParser [16] with release version 1.3.4.

Area and Power Modeling: The baseline MS clustering tools were evaluated on a server with Intel Xeon E5-2680 CPU, 64GB DDR4-2133MHz memory, and a 2TB commercial PCIe-based SSD. The maximum read speed is 3.2GB/s. The energy consumption of CPU baselines is measured using CPU Energy Meter [5]. MSAS accelerators are implemented using *Verilog* and synthesized by *Synopsys Design Compiler* on TSMC 28nm process. The clock frequency is set to 800MHz. The area and energy consumption of FIFO and buffer are estimated using CACTI [21].

Table 2: MSAS Implementation and Area Breakdown

Design	**SSD-Level**	**Channel-Level**
Frequency	800MHz	
Bitonic Top-K	$N = 128$ (0.09mm^2)	
Norm. Scale	0.27mm^2	0.27mm^2
FIFO	32b×64	
Buffer	256KB (0.36mm^2)	64KB (0.09mm^2)
Area	0.72mm^2	0.45mm^2
Number	1	4
Total Area	0.72mm^2	1.81mm^2
Average Power	2.22W	8.06W

MSAS is evaluated on 1TB Intel DC P4500 SSD, providing a sequential read bandwidth of 3.2GB/s and sequential write bandwidth of 600MB/s [1]. The active power under sequential access mode is 11W and 9.6W for write and read, respectively. We assume each flash array has a read latency of 64us, 16 channels, 4 flash chips per channel, 2 dies per chip, 2 planes per die, 512 blocks per plane, and 1024 pages per block. Each flash page is 8KB, and each channel using ONFI [2] has a bandwidth of 800MB/s. We combine performance data obtained via SSD simulation with the SSD power model in [17] to estimate the energy consumption. The configurations and area breakdown of MSAS accelerator in three storage levels are summarized in Table 2. We let each channel-level accelerator share $C_{\text{share}} = 4$ channels, requiring $16/4 = 4$ channel-level accelerators. The flash-level design contains a total of $16 \times 4 = 64$ accelerators.

4.2 Performance and Energy Evaluation

Comparison of SSD and channel-level designs We first compare the preprocessing speed of MSAS accelerators at different storage levels. The configurations of SSD-level and channel-level designs are shown in Table 2. The spectra data are pre-stored in SSD before the preprocessing starts. Fig. 6 illustrates the execution time on four MS datasets, where the channel-level execution time is normalized to 1. Channel-level design is 3.3× to 3.8× faster vs. SSD-level design. The gain comes from two aspects. First, spectra preprocessing is data-intensive rather than computation-intensive workload. In this case, higher bandwidth brings about more significant speedup. Second, each channel-level accelerator is connected to 4 channel buses and enjoys 3.2GB/s bandwidth. As discussed in Section 3.2, each fully pipelined channel-level accelerator can fulfill 4.8GB/s data rate. The four channel-level accelerators are able to fully exploit the aggregate 12.8GB/s internal bandwidth of 16 channels. In comparison, the performance of SSD-level design is restricted by the limited 3.2GB/s bandwidth of the on-chip bus.

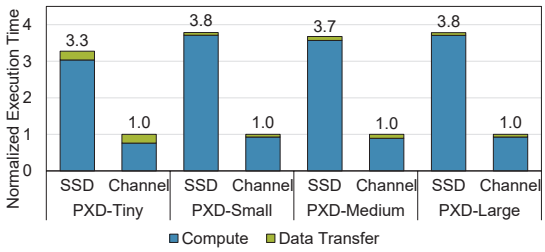

Figure 6: Execution time comparison for SSD-level and channel-level designs.

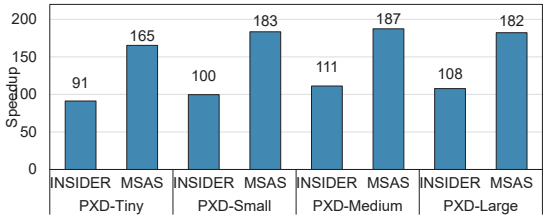

Figure 7: Preprocessing speedup of INSIDER [24] and MSAS over CPU baselines.

Spectra preprocessing speedup In Fig. 7, we compare the performance of channel-level design and the in-storage computing prototype, INSIDER [24], over CPU on spectra preprocessing workload. We use *msCRUSH* as the CPU baselines as it yields the fastest speed. INSIDER uses 8-lane PCIe, which delivers 8GB/s peak bandwidth, and the spectra preprocessing is computed using FPGA in SSD. The channel-level MSAS achieves 165× to 187× speedup over CPU. Moreover, MSAS is 1.7× to 1.8× faster than INSIDER. The speedup is derived from two aspects. First, INSIDER needs to load raw spectrum data from SSD into FPGA and then transfer processed data back to SSD, incurring redundant data movement. MSAS only fetches data from NAND flash once. Second, the internal bandwidth of MSAS is 60% higher than INSIDER's PCIe bandwidth.

System-level improvements after integration We integrate the channel-level design into the existing MS clustering tools (*msCRUSH*, *MS-Cluster*, and *MaRaCluster*) by replacing the preprocessing part with MSAS framework. This offloads the preprocessing into MSAS-enabled SSD. The clustering part is executed on the CPU. For CPU baselines, the entire preprocessing and clustering process is computed by CPU. Fig. 8 gives the speedup and energy efficiency of MSAS over CPU baselines after integration. The channel-level

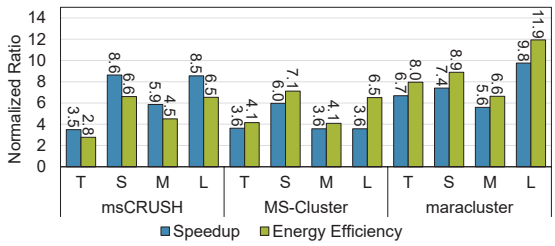

Figure 8: Speedup and energy efficiency over CPU after integrating MSAS into clustering tools.

MSAS achieves 3.5× to 9.8× speedup and 2.8× to 11.9× energy efficiency on four datasets. The preprocessing steps are dominant the runtime and energy of clustering workloads, as shown in Fig. 1. Accelerating preprocessing provides significant overall speedup for clustering tools.

Comparing the performance gain within each clustering tool, we observe that the gain is more significant on datasets with longer average spectrum length (*e.g.* PXD-Small and PXD-Large). This is due to the fact that the average spectra length of PXD-Tiny is close to the top-k value ($k = 50$). The data size of preprocessed spectra is not greatly reduced compared to the raw spectra, leading to 3.5× to 6.7× with 2.8× to 8.0× lower energy consumption. The difference in performance gain between the three clustering tools is mainly benefitted from the code optimization.

4.3 Overhead Analysis

The channel-level design consumes the largest area among the proposed designs. Considering that the TLC NAND flash chip [31] has a die size over 100mm^2, the largest channel-level design only incurs 0.03% area overhead, which is negligible. SSD's 50W power budget [19] is more than sufficient for the added MSAS accelerators. The channel-level design, with the highest 8.06W power dissipation, meets the power supply constraints.

5 CONCLUSION

In this work, we propose MSAS , the near-storage computing framework that efficiently accelerates the mass spectrum data preprocessing. Based on the observation that preprocessing takes nearly 82% of the overall execution time and energy consumption for MS analysis, MSAS tackles the challenge by performing the preprocessing in SSD. We present two types of accelerator designs to exploit the internal storage bandwidth at different levels of the storage hierarchy. Then we design scalable and energy-efficient accelerators to satisfy the data rate for each storage level. The channel-level MSAS generates the best efficiency with 1.81mm^2 area and 8.06W power. The experiments show that the channel-level MSAS is able to boost spectra preprocessing by up to 187× compared to the fastest MS analysis tool. Moreover, MSAS reduces the execution time up to 1.8× compared to in-storage computing prototype, INSIDER. We show that the proposed solution can improve the clustering speed and energy efficiency of the overall MS clustering pipeline by 3.5× to 9.8× and 2.8× to 11.9×, respectively.

ACKNOWLEDGMENTS

This work was supported in part by CRISP, one of six centers in JUMP (an SRC program sponsored by DARPA), SRC Global Research Collaboration (GRC) grant, and NSF grants #1826967, #1911095, #2052809, #2112665, #2112167, and #2100237.

REFERENCES

[1] 2017. Intel SSD DC P4500 Series. https://ark.intel.com/content/www/us/en/ark/products/series/96935/intel-ssd-dc-p4500-series.html.
[2] 2021. Open NAND Flash Interface Specification. http://www.onfi.org/specifications.
[3] 2022. NVM Express Base Specification 2.0. https://nvmexpress.org/developers/nvme-specification/.
[4] Nitin Agrawal et al. 2008. Design tradeoffs for SSD performance.. In *USENIX ATC*, Vol. 57.
[5] Dirk Beyer and Philipp Wendler. 2020. CPU Energy Meter: A tool for energy-aware algorithms engineering. *Tools and Algorithms for the Construction and Analysis of Systems* 12079 (2020), 126.
[6] Wout Bittremieux et al. 2021. Large-scale tandem mass spectrum clustering using fast nearest neighbor searching. *bioRxiv* (2021).
[7] François-Michel Boisvert et al. 2012. A quantitative spatial proteomics analysis of proteome turnover in human cells. *Molecular & Cellular Proteomics* 11, 3 (2012).
[8] Feng Chen et al. 2016. Internal parallelism of flash memory-based solid-state drives. *ACM TOS* 12, 3 (2016), 1–39.
[9] Wooseong Cheong et al. 2018. A flash memory controller for 15μs ultra-low-latency ssd using high-speed 3d nand flash with 3μs read time. In *ISSCC*. 338–340.
[10] Joel M Chick et al. 2015. A mass-tolerant database search identifies a large proportion of unassigned spectra in shotgun proteomics as modified peptides. *Nature Biotechnology* 33, 7 (2015), 743–749.
[11] Ari M Frank et al. 2008. Clustering millions of tandem mass spectra. *Journal of Proteome Research* 7, 01 (2008), 113–122.
[12] Timo Glatter et al. 2015. Comparison of different sample preparation protocols reveals lysis buffer-specific extraction biases in gram-negative bacteria and human cells. *Journal of Proteome Research* 14, 11 (2015), 4472–4485.
[13] Johannes Griss et al. 2016. Recognizing millions of consistently unidentified spectra across hundreds of shotgun proteomics datasets. *Nature Methods* 13, 8 (2016), 651–656.
[14] Saransh Gupta et al. 2019. RAPID: A ReRAM processing in-memory architecture for DNA sequence alignment. In *ISLPED*. 1–6.
[15] Wenqin Huangfu et al. 2019. Medal: Scalable dimm based near data processing accelerator for dna seeding algorithm. In *MICRO*. 587–599.
[16] Niels Hulstaert et al. 2019. ThermoRawFileParser: modular, scalable, and cross-platform RAW file conversion. *Journal of Proteome Research* 19, 1 (2019), 537–542.
[17] Myoungsoo Jung et al. 2016. NANDFlashSim: High-fidelity, microarchitecture-aware NAND flash memory simulation. *ACM TOS* 12, 2 (2016), 1–32.
[18] Min-Sik Kim et al. 2014. A draft map of the human proteome. *Nature* 509, 7502 (2014), 575–581.
[19] Vikram Sharma Mailthody et al. 2019. Deepstore: In-storage acceleration for intelligent queries. In *MICRO*. 224–238.
[20] Lennart Martens et al. 2005. PRIDE: the proteomics identifications database. *Proteomics* 5, 13 (2005), 3537–3545.
[21] Naveen Muralimanohar et al. 2007. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0. In *MICRO*. 3–14.
[22] David N Perkins et al. 1999. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis* 20, 18 (1999), 3551–3567.
[23] Andreas Roos et al. 2016. Cellular signature of SIL1 depletion: disease pathogenesis due to alterations in protein composition beyond the ER machinery. *Molecular Neurobiology* 53, 8 (2016), 5527–5541.
[24] Zhenyuan Ruan, Tong He, and Jason Cong. 2019. INSIDER: Designing in-storage computing system for emerging high-performance drive. In *2019 USENIX ATC*. 379–394.
[25] Anil Shanbhag et al. 2018. Efficient top-k query processing on massively parallel hardware. In *International Conference on Management of Data*. 1557–1570.
[26] Matthew The and Lukas Kall. 2016. MaRaCluster: A fragment rarity metric for clustering fragment spectra in shotgun proteomics. *Journal of Proteome Research* 15, 3 (2016), 713–720.
[27] UCSD. 2022. MassIVE: Mass Spectrometry Interactive Virtual Environment. https://massive.ucsd.edu/.
[28] Juan A Vizcaíno et al. 2014. ProteomeXchange provides globally coordinated proteomics data submission and dissemination. *Nature Biotechnology* 32, 3 (2014), 223–226.
[29] Lei Wang et al. 2018. msCRUSH: fast tandem mass spectral clustering using locality sensitive hashing. *Journal of proteome research* 18, 1 (2018), 147–158.
[30] Wikipedia. 2022. Mass Spectrometry. https://en.wikipedia.org/wiki/Mass_spectrometry.
[31] Ryuji Yamashita et al. 2017. A 512Gb 3b/cell flash memory on 64-word-line-layer BiCS technology. In *ISSCC*. 196–197.
[32] Şule Yilmaz et al. 2017. Methods to calculate spectrum similarity. In *Proteome bioinformatics*. 75–100.