Methodology for Evaluating the Potential of Disaggregated Memory Systems

Nan Ding, Samuel Williams, Hai Ah Nam, Taylor Groves, Muaaz Gul Awan
LeAnn Lindsey, Christopher Daley, Oguz Selvitopi, Leonid Oliker, Nicholas Wright
Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
{nanding, swwilliams, hnam, tgroves, mgawan, lmlindsey, csdaley, roselvitopi, loliker, njwright}@lbl.gov

***Abstract*—Tightly-coupled HPC systems have rigid memory allocation and can result in expensive memory resource underutilization. As novel memory and network technologies mature, disaggregated memory systems are becoming a promising solution for future HPC systems. It allows workloads to use the available memory of the entire system. In this paper, we propose a design framework to explore the disaggregated memory system design space. The framework incorporates memory capacity, network bandwidth, and local and remote memory access ratio, and provides an intuitive approach to guide machine configurations based on technology trends and workload characteristics. We apply our framework to analyze eleven workloads from five computational scenarios, including AI training, data analysis, genomics, protein, and traditional HPC. We demonstrate the ability of our methodology to understand the potential and pitfalls on a disaggregated memory system and motivate machine configurations. Our methodology shows that the 10 out of our 11 applications/workflows can leverage disaggregated memory without affecting performance.**

***Index Terms*—Memory disaggregation, system architecture design space**

I. Introduction

The last five decades have seen compute and high-capacity memory performance continue to diverge. In response, computer architects have added more and more levels of caching within the memory hierarchy to mitigate this performance discrepancy for the applications that exhibit sufficient temporal and spatial locality. Today, GPU-accelerated systems rely on a hierarchy of progressively faster and smaller memories — CPU-attached DDR high capacity memory, GPU-attached HBM high performance memory, and multiple levels of very fast, on-GPU, SRAM cache memories. Future systems (e.g. NVIDIA's Hopper H100 GPU [1]) will continue to leverage this template, but do so in a more tightly integrated and performant form factor. Integration will ultimately enable single-chip CPU-GPU architectures — the Accelerated Processing Unit (APU).

Although the trend towards integration can improve performance, it can result in large, expensive, monolithic computing nodes whose resource utilization can vary greatly from one application to another within a workload. Examples include the observation that only 15% of the scientific workloads on NERSC's Cori supercomputer use over 75% of the available memory per node [2] and that 90% of jobs utilize less than 15% of the node memory capacity on the HPC clusters at Lawrence Livermore National Laboratory [3]. Additionally, up to 83% of memory can be underutilized on these tightly-coupled resources that are over-provisioned for workloads with the greatest demands [4]. Similarly, in the commercial cloud sector, it has been shown that memory is the dominate cost of servers while most VMs use less than half of their available memory [5]. As such, HPC system architects are often forced to either overprovision systems for the long tail of memory requirements (incurring substantial additional cost), deploy a variety of node architectures (incurring the complexity and inefficiency of job scheduling), or demand those applications restructure themselves to fit in half or a quarter of their nominal memory footprint.

Concurrently, many applications and workflows leverage high-performance distributed file systems for rather mundane tasks — holding read-only or private files — resulting in an overprovisioning of file system performance and degrading QoS for the applications that truly need a high-performance distributed file system.

Recent improvements in interconnect technology [5] have reinvigorated memory disaggregation as a viable solution to both the memory and file system stranded resource problems. Memory disaggregation decouples compute and memory resources. Compute nodes would contain only a limited amount of local memory, but could access a large pool of remote memory available via the network. This design enables HPC systems to easily scale memory capacity and allocate memory more flexibly. Physically, this large pool of memory will be partitioned among a number of smaller "memory nodes" containing DRAM and a NIC in order to maximize bandwidth, capacity, reliability, etc...

In this paper, we explore the value of adding disaggregated memory to an APU-only HPC system. To that end, it is imperative one provide a methodology for analyzing how technology and system architecture constrain application performance. In our work, we:

1) Develop a methodology for evaluating and visualizing application performance on disaggregated systems.
2) Characterize and analyze eleven applications and workflows using two extensions of the Roofline model.

This material is based upon work supported by the Advanced Scientific Computing Research Program in the U.S. Department of Energy, Office of Science, under Award Number DE-AC02-05CH11231 and used resources of the National Energy Research Scientific Computing Center (NERSC) which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. Thanks Khaled Ibrahim and Tan Nguyen for answering my questions on the AI workloads. Thanks John Wu and Bin Dong for providing their expertise on DASSA.

3) Provide insights and guidance to vendors developing technologies that support disaggregated memory.
4) Discuss how HPC system architects should use our methodology to evaluate different disaggregated memory system architectures for their respective workloads (a time- and space-weighted set of applications).

Our results demonstrate that 10 of our 11 examined applications do not suffer a performance penalty using a disaggregated system with a significantly reduced memory balance.

II. Related Work

Recent studies highlight how the current approach of allocating resources to jobs on statically-configured compute nodes wastes memory and NIC resources. Utilization analysis for HPC systems report the average memory utilization of a job can be as small as 11.9% and 74.63% of individual jobs never use more than 50% of on-node memory. Approximately three-quarters of the time, each compute node uses only 0.3% of memory bandwidth and 0.5% of available NIC bandwidth [4]. Often resources are idle, since HPC system node design is based on the peak usage, i.e., the maximum memory usage. It is worth mentioning that DRAM consumes static power even when idle, so unused memory still contributes to the HPC system operating cost [3]. It has become a common state of the practice for memory resources on HPC systems to be over-provisioned and have low resource utilization.

The problem of low memory utilization is fundamentally not solvable with current static allocations on tightly-coupled HPC systems. Network-attached memory disaggregation has been proposed to improve resource utilization for nearly a decade and is seeing a resurgence of interest to improve memory utilization in commercial cloud data centers [4, 6–12]. The growing interest and maturity of Compute Express Link (CXL) [13–16], a standardized protocol for memory pooling, has been contributing to this renewed interest in memory disaggregation. CXL provides memory coherency and semantics over the PCIe physical layer. A practical CXL based disaggregated solution for a production cloud deployment projected that the memory pooling approach would incur only a performance loss between 1-5%, but could achieve a 9-10% reduction in overall system DRAM required, which represents hundreds of millions of dollars in cost savings for a large cloud provider [5]. Recently, disaggregated remote memory has been proposed for HPC systems. Peng et al. further designed a user-space remote paging library to allow applications to explore the potential of throughput scaling on disaggregated memory [3].

Previous studies focus on exploring the potential benefits of memory disaggregation and the limitations using current HPC systems [3, 4, 9, 10, 17]. The main concern for implementing memory disaggregation is the bandwidth and latency penalty over the network which would degrade application performance [18]. However, there is no structured analytical method that demonstrates which applications are performance constrained on the disaggregated memory system, how much the network performance penalty affects performance, and

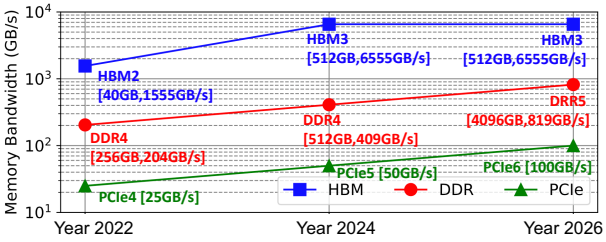


Fig. 1: Memory bandwidth trends for HBM, DDR and PCIe from years 2022 to 2026. Relative bandwidth improvements remain constant. The PCIe is the performance bottleneck for disaggregated systems.

what are important application metrics to reason the performance on a disaggregated memory system. Missing from past work is a practical and intuitive approach to assess how much disaggregation is needed or viable given the technology trend and the impacts to the diverse workload. In this paper, we aim to provide a structured system design model to explore the architecture design space and its capabilities. We provide several methods to visualize the design space and a methodology that could be adapted for a broader range of users, e.g., vendors and application developers, to help to design a new architecture or purchase future systems.

III. System Architecture Design

It is necessary to understand the emerging technology trends and capabilities in a future disaggregated memory system to assess the potential benefits and pitfalls. Fig. 1 charts the memory bandwidth trends of HBM, DDR, and PCIe from today to the year 2026. We assume HBM3 is built with eight 16-Hi stacks and each stack has a capacity of 64 GB. We use the maximum capacity and bandwidth per DIMM (DDR4: 32 GB/DIMM and 25.6 GB/s/DIMM; DDR5: 256 GB/DIMM and 51.2 GB/s/DIMM) with a total of 16 DIMMs for DRAM memory. Both CXL devices and network interface cards are limited by the performance of the physical PCIe interconnect that they connect through. As such, PCIe would eventually be the performance bottleneck on a disaggregated system since data needs to be loaded from DDR via the network.

Fig. 2 presents a schematic of a basic network-attached disaggregated memory system. We consider that such a system could have C compute nodes and M memory nodes. Each compute node consists of an accelerated processing unit (APU) with its own local high-bandwidth memory (HBM). An APU combines a CPU with a GPU onto a single silicon die, and both

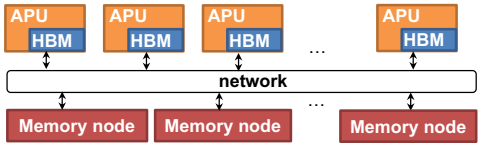
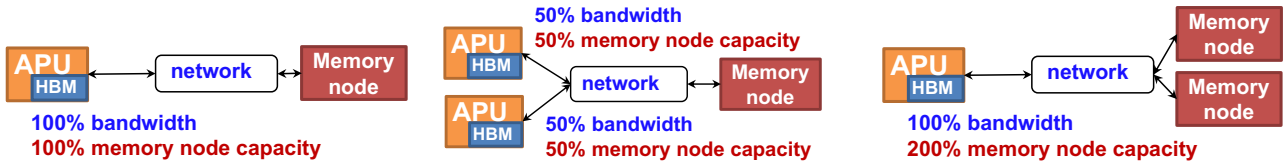


Fig. 2: Conceptual disaggregated memory system architecture.



(a) $\frac{C}{M} = \frac{1}{1}$: 100% of one memory node capacity and 100% of remote memory bandwidth (b) $\frac{C}{M} = \frac{2}{1}$: 50% of one memory node capacity and 50% of remote memory bandwidth (c) $\frac{C}{M} = \frac{1}{2}$: 200% of one memory node capacity and 100% of remote memory bandwidth

Fig. 3: Three use cases for network-attached disaggregated memory. Each varies the ratio of compute nodes (C) to memory nodes (M) and gives the available remote memory bandwidth (relative to one NIC) and remote memory capacity (relative to one memory node) available to each compute node. The flexibility of disaggregated memory allows systems to realize all three configurations simultaneously for different applications.

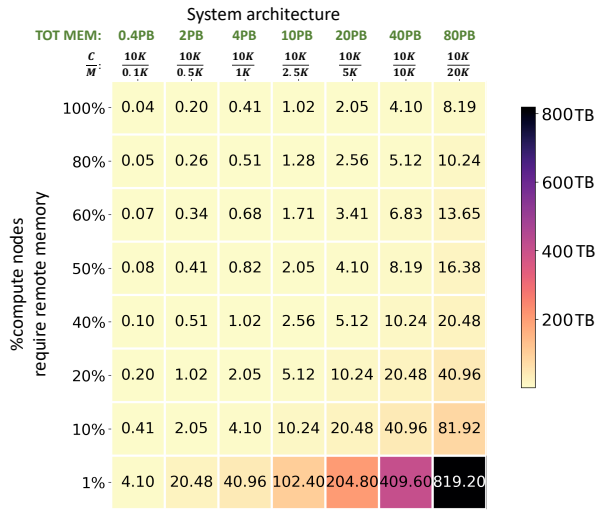
CPU and GPU share a common path to the remote memory. Future processor trends favor the APU because it addresses the bottleneck of data transfers between CPU and GPU [19, 20]. Each memory node is equipped with DDR memory as the remote memory. The compute nodes and memory nodes are connected via a network and is assumed to have one PCIe NIC. Using these assumptions, we propose a structured system architecture design methodology to explore the potential and pitfalls of disaggregated memory.

A. *Remote memory resources versus utilization*

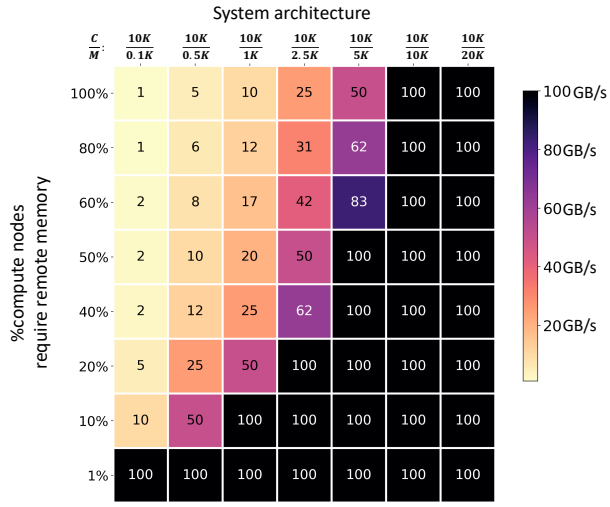
Fig.3 visualizes three use cases that highlight the impact of system architecture on available remote memory capacity and memory bandwidth. Fig. 3a highlights the simplest case ($\frac{C}{M} = \frac{1}{1}$), where each compute node is in a job is paired with one memory node. Each compute node would theoretically have access the the memory's nodes full capacity and 100% of the NIC's bandwidth as remote memory bandwidth. Unsurprisingly, in the case of $\frac{C}{M} = \frac{2}{1}$ in Fig. 3b, each compute node has haf the capacity and half the remote memory bandwidth. Interestingly, if $\frac{C}{M} = \frac{1}{2}$ as in Fig. 3c, each compute node could access 200% of a memory node's capacity, but still only attain 100% of the NIC bandwidth as remote memory bandwidth as bandwidth is constrained by the APU's NIC rather than the two memory node NICs.

Following this method, we can build a design space with various ratios to describe the hardware capabilities in terms of memory capacity and available remote memory bandwidth. We scale the building blocks to a modern-day HPC system and assume that we have 10K compute nodes. The heat maps in Fig. 4 present the (a) available remote memory capacity and (b) available remote memory bandwidth per compute node under different compute and memory node ratios, assuming one memory node capacity of 4TB. For the fixed number of compute nodes, Fig. 4a shows the available DDR5 remote memory capacity (TB) to the compute nodes with growing numbers of memory nodes (100 to 20K). The vertical axis is binned into the percentage of compute nodes that will require more resources than the local HBM memory and use the remote DDR memory, a value that will be specific to HPC systems and their workload. The available remote memory capacity per compute node becomes larger as we increase the number of memory nodes (moving left to right of Fig. 4a). That is to say, there is less contention as we increase the number of memory nodes. Similarly, we can also reduce contention as the number of compute nodes that require remote memory decreases (moving top to bottom of Fig. 4a). For example, the first row represents the scenario where all the compute nodes require remote memory. Therefore, if we have 10K DDR5 memory nodes ($\frac{C}{M} = \frac{1}{1}$), each compute node can access one memory node's capacity of 4TB. When decreasing the demand, only 5K compute nodes (50% of the total 10K) require the remote memory of 10K memory nodes, then each compute node can then access 8TB of remote memory, which equals the capacity of two memory nodes. Correspondingly, Fig. 4b presents the available remote memory bandwidth for the cases in Fig. 4a. Unlike memory capacity in Fig. 4a, memory bandwidth in Fig. 4b will saturate at the compute node's peak NIC bandwidth as one moves decreases $\frac{C}{M}$ (right) or one decreases the fraction of compute nodes requiring remote memory (down).

Determining an optimal system configuration relies on multiple factors specific to the HPC system workload (demand) and available budget (supply of memory nodes). Fig. 1 suggests in the 2026 time frame, HBM3 could provide 0.5TB of local memory. Thus, in planning for the next machine, as a guiding principle, there should be enough memory nodes to provide more remote memory capacity per node than local memory capacity. As such, configurations in the upper left region of Fig. 4 where memory node capacity is smaller than 0.5TB are wasteful architectures. Conversely, configurations on the right of the figure can be quite expensive as there are as many or more memory nodes than compute nodes (the network has 2-3$\times$ more endpoints). Finally, although configurations in the bottom right provide 100s of TB per compute node, they can only access it at 100GB/s. As such, it will take minutes to hours to read all of remote memory once. Such architectural configurations may become impractical given the number of times an application might desire to read memory coupled with finite job run time limits.



(a) Available remote memory capacity per compute node.



(b) Available remote memory bandwidth per compute node.

Fig. 4: Disaggregated memory system design space assuming fixed 10K (C) compute nodes and varying the number of (M) memory nodes, each with 4TB of DDR5 remote memory accessed through a PCIe6 NIC. Contention is reduced from left to right and from top to bottom. The y-axis shows the demand from compute nodes for remote memory and the x-axis shows the supply of memory nodes available.

B. Remote memory access patterns

Disaggregated memory promises to improve system-wide memory utilization, but individual application performance is of equal concern. Prior work argues that disaggregation comes with substantial bandwidth and latency penalties to applications [3]. However, such conclusions are derived assuming current technologies and lack consideration of emerging technologies in the future. To analyze the impact to individual applications in the near future, we introduce the local to remote memory access ratio (L:R) metric to characterize application performance on a disaggregated memory system. We then correlate the metrics using a memory Roofline plot to provide a generalized framework to evaluate and visualize the performance bottlenecks of applications running on a disaggregated memory system.

The traditional Roofline model [21] characterizes an application's performance (GFLOP/s) as a function of its arithmetic intensity (FLOPs executed per Byte moved). It provides a quick visual comparison of the application performance compared against the bounds set by the peak compute performance (GFLOP/s) and the peak memory bandwidth of the target architecture (GB/s) to determine what is limiting performance: memory or compute.

Following the methodology of the traditional Roofline model, our new memory Roofline model characterizes an application's sustained memory performance (GB/s) as a function of its local and remote memory access ratio (L:R), the peak local memory bandwidth, and the peak remote memory bandwidth. An application's L:R on a disaggregated memory system could be considered as the ratio of HBM data movement (local) to the DDR data movement (remote over PCIe) or even the HBM to file size ratio when examining applications using memory nodes as a private file system. Applications with a L:R data movement ratio greater than the system's local:remote bandwidth ratio can effectively hide the slow remote (disaggregated) memory bandwidth behind a multitude of fast, local memory accesses.

Fig. 5 presents the memory Roofline model using future HBM (local) and PCIe (remote) bandwidths. One quickly observes the visual similarity to the traditional Roofline model with local bandwidth replacing the traditional peak GFLOP/s plateau and remote bandwidths replacing the traditional memory diagonals. We observe an HBM3:PCIe6 machine balance of 65.5 — the ratio of data movement that results in equal time for local and remote transfers. This ratio is very close to today's HBM2:PCIe4 machine balance of 62.2. This suggests future hardware trends will not detract from the efficacy of disaggregated memory.

Applications like ADEPT with a L:R ratio of nearly 500 (far greater than 65.5) are insensitive to memory disaggregation, dominated by on-node performance, and will use less than 14% of the available PCIe bandwidth (green diagonal line). Conversely, applications like STREAM with an theoretical L:R ratio of 2 will see their performance limited and degraded by disaggregated memory bandwidth. Ultimately, increases in NIC bandwidth shift the machine balance to the left (decreasing the number of applications penalized by disaggregation) while increases in HBM bandwidth shift the machine balance to the right (increasing the number of applications penalized by disaggregation).

IV. Application Case Studies

In this section, we apply our system architecture design methodology to select a disaggregated system configuration and evaluate the potential performance using eleven applications across five computational scenarios.

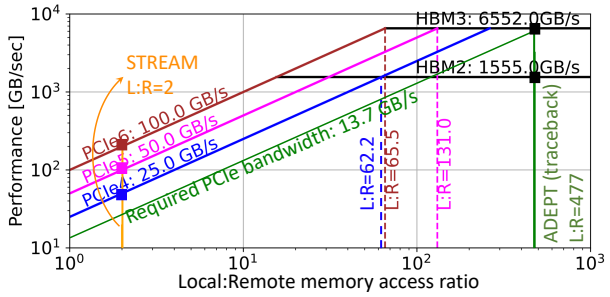


Fig. 5: Memory Roofline model characterizing an application's memory access performance (GB/s) as a function of its local to remote memory access ratio (L:R). A high L:R ration is critical in mitigating the performance penalties of disaggregated memory. Observe ADEPT is insensitive to disaggregated memory while STREAM is penalized by it.

A. Disaggregated System Configurations

To select a machine configuration, recall the HPC system described in Section III. A with 10,000 compute nodes with 512 GB of HBM3 local memory capacity, accessing DDR5 remote memory nodes via PCIe6-connected NICs. As previous studies showed only 15% of the workloads use 75% of the node memory [2], we conservatively assume that at any instant, 10% of the compute nodes will require remote memory for our machine configuration. Referencing Fig 4a, at 10%, we could choose 500 memory nodes or more with DDR5 memory (x-axis) to ensure each compute node has access to remote memory greater than the local HMB3 memory. Including the memory bandwidth information from Fig 4b, the maximum memory bandwidth per compute node peaks at 1000 memory nodes. Purchasing more memory nodes would only add additional capacity and cost, not additional memory bandwidth. For the configuration of 10,000 compute nodes accessing an aggregate four petabytes of DDR5 memory on 1000 memory nodes, we see from Fig.4 that each of the compute nodes requiring remote memory can access, on average, four terabytes of remote memory with a peak remote memory bandwidth of 100 GB/s.

B. Application Characteristics

Due to the diversity of applications, the case studies we examined required a variety of approaches to measure or estimate the local and remote memory accesses (L:R). This section summaries the high-level methods of calculating L:R for each workload. Note that throughout the paper, we assume each application will preserve its current conceptual approach to exploiting data locality and expressing data movement when ported to a disaggregated memory architecture even if the mechanisms are syntactically different.

Artificial intelligence (AI) training workloads. AI is an area of increasing scientific interest with growing computational demands [2]. It is a driver for future DOE investments in HPC platforms [22]. We focus on training workloads, which is more computationally expensive and requires a larger memory capacity than inference. We demonstrate the benefit of a disaggregated memory system using three AI training workloads: CosmoFlow [23] and DeepCAM [24] from the MLPerf HPC benchmark suite [25], and a well established image classification model, ResNet-50 [26] from the MLPerf Training benchmark suite [27]. The actual computation and memory characteristics of the three AI training workloads come from Ibrahim *et al.* [28] and listed in Table I. The local:remote memory ratio is characterized by FLOP:sample Byte/Flop:HBM Byte. All the numbers reported in Table I refer to the memory per job.

TABLE I: Computation and Memory Characteristics

	ResNet-50	DeepCAM	CosmoFlow
Training set size [28]	0.15 TB	8.8 TB	5.1 TB
FLOP:HBM Byte [28]	55.35	55.5	38.6
FLOP:sample Byte [28]	221,000	107,000	15,400
Local:Remote memory access ratio	3993	1927	399

Data analysis workloads. Data analysis applications are a growing workload in HPC facilities [2]. We use two data analysis software frameworks, DASSA [29] and TOAST [30], to showcase disaggregated memory benefits. DASSA [29] is a distributed acoustic sensing (DAS) data storage and analysis framework for geophysicists to perform DAS data analysis on HPC systems. We use a real DAS data analysis case for earthquake detection via local similarity. We use analytical modeling to estimate the L:R and refer its input file size as the remote memory capacity requirement.

TOAST [30] is a software framework designed for simulation and reduction of data from telescope receivers which acquire time streams of individual detector responses. Here we use a satellite telescope benchmark as an example to show the implication of memory disaggregation. The core computation in the satellite telescope benchmark is the PCG solver. We profile its DRAM data movement using Intel Vtune on one Cori Haswell [31] node as its local memory accesses and refer its input file size as the remote memory capacity requirement.

Genomics workloads. With the rapid development of genome sequencing technologies, it is now possible to sample and study genomes at an unprecedented scale. MetaHipMer [32] is a large-scale metagenome assembler that can leverage the large memory and compute capacities of supercomputers to co-assemble terabase-scale datasets. We use three important kernels in MetaHipMer, ADEPT [33] w/wo traceback and EXTENSION [34] to understand the their potential on a disaggregated memory system. We use analytical modeling to calculate the L:R of ADEPT w/wo traceback kernels. We use NVIDIA NSight compute [35] to collect the HBM data movement for single extension on Cori GPU [36], and then multiply that with 45 million extensions as its local memory access. We use analytical modeling to estimate the remote memory capacities for all three kernels.

Protein similarity search workloads. Bioinformatics applications have been increasingly turning to HPC solutions for solving big problems with reasonable time-to-solution.

Especially in metagenomics research, the scale of the data often requires memory and compute resources that are beyond what serial systems can provide. An important task that forms the backbone of many bioinformatics workflows is the alignment of a set of given sequences against a reference database. PASTIS [37] is a distributed-memory many-against-many search tool specifically developed for protein sequences. This search requires a lot of memory and its memory complexity grows quadratically with the number of sequences while being compute-intensive. For batch pairwise alignments required by the protein similarity search, PASTIS uses SeqAn [38] for CPUs and ADEPT [33] for GPUs. We use NVIDIA NSight compute [35] to collect the HBM data movement as the local memory access and use analytical modeling to estimate the remote memory capacities.

Traditional HPC Workload Bookends. Traditional HPC workloads are designed for distributed-memory systems. They can sometimes scale to thousands of and even millions of cores [39–41]. They can distribute the memory footprint, and can fit in the 512 GB HBM3 in a 2026 disaggregated system which is larger than the currently provided node-local DDR (256 GB DDR on a 2021 HPC system). We use GEMM [42] and STREAM [43] as two representative benchmarks to show the implications as the data size grows. We use analytical modeling to estimate the L:R for GEMM and STREAM. Note that STREAM can be a proxy for giant AI=O(1) linear solvers (stencil/sparse) without any multiphysics/AMR.

C. Application Analysis

Fig. 6 visualizes a summary of all the tested applications in this section. It combines the two critical metrics, the local and remote memory access ratio (L:R) from Fig. 5 and the per-node memory capacity to provide an intuitive way to visual the performance of applications on a future disaggregated memory system and assess individual application potential and pitfalls. Our system assumes 2026 memory technologies with each compute node having 512 GB HBM3 local memory which is two times larger than a 2021 machine's node-local DDR capacity [44]. Thus, applications that can fit in 2021 machine's node-local DDR can undoubtedly fit in future local memory. We characterize the applications into three categories. **Blue:** required memory footprint can fit in local HBM memory. Thus, applications in this region would be HBM bound, e.g., ResNet-50. **Orange:** required memory footprint can not fit in local memory and there will be a performance penalty from disaggregation due to the low L:R ratio (smaller than 65.5), e.g., STREAM (>512GB). **Green:** required memory footprint does not fit in local memory but applications could achieve HBM3 bandwidth due a high L:R ratio (larger than 65.5), and would thus not incur a performance penalty from disaggregation, e.g., DeepCAM. Note, applications in the green region are ultimately HBM bound, but can still be impacted by the PCIe NIC bandwidth due to inefficient data movement and bandwidth contention forcing them to fall into the orange zone.

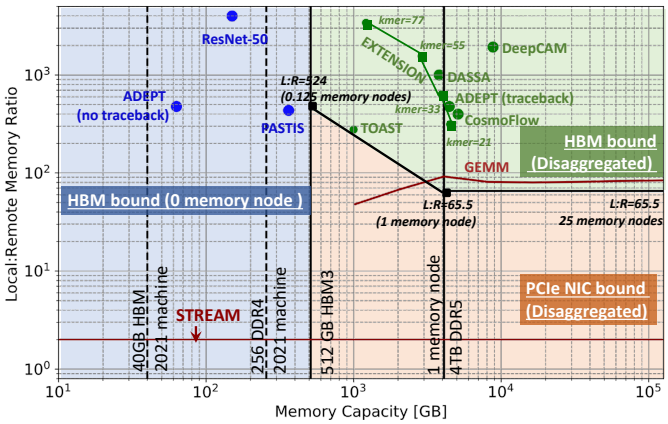


Fig. 6: Capacity and bottleneck visualization of applications on a disaggregated memory system. The L:R of PASTIS and ResNet-50 are profiled from Cori GPU [36] at NERSC. ADEPT w/o traceback is modeled based on its implementation on Cori GPU.

The antidiagonal line connecting L:R=524 to L:R=65.5 in Fig. 6 shows the implications of network contention. If the memory capacity the application needs is between 512 GB and 4 TB, there are two design possibilities. The first is to use one memory node per compute node (L:R=65.5, one memory node) which guarantees all 100 GB/s of the PCIe6 NIC bandwidth but wastes memory capacity. The other possibility is to share memory nodes across compute nodes (upper left region in Fig. 4a and Fig. 4b). In this case, memory is not wasted, but compute nodes must contend for memory node NIC bandwidth (L:R=524, 0.125 memory node). Such contention leads to an antidiagonal boundary between the green and orange zones. Following the same methodology, one could imagine that disaggregating today's system can shrink the blue region to the 40GB HBM vertical dotted line. The green and orange zones expand to the left accordingly. The antidiagonal boundary moves to lower left, and the L:R boundary moves down a little to 62.2.

ResNet-50: The ResNet-50 v1.5 is a 50-layer deep convolutional neural network. ResNet-50 has been implemented in both TensorFlow and PyTorch with numerous implementations and optimizations that prevent direct comparisons of system performance. The actual computation and memory characteristics of ResNet-50 come from Ibrahim *et al.* [28]. ResNet-50 on Imagenet data requires 0.15 TB memory to store the training data set, and its L:R ratio is 3993. On the selected system configuration, the L:R ratio has no impact because the training data can easily fit into local memory.

DeepCAM: The DeepCAM climate benchmark is based on the 2018 work of Kurth *et al.* [24] which was awarded the ACM Gordon Bell Prize. It uses deep learning to identify extreme weather phenomena from background images. Unlike ResNet-50, DeepCAM has a large memory requirement, 8.8 TB, to store the training size [28]. It requires 2.2 memory nodes using our selected system configuration. As its L:R is

1927, which is higher than 65.5 (on the left of the orange dotted wall in Fig. 5), DeepCAM can operate at HBM3 speed on a disaggregated system that uses HBM3 as local memory and PCIe6 NIC for the network.

CosmoFlow: CosmoFlow uses a 3D convolutional neural network with five convolutional layers and three fully connected layers. For the training run in Table I, we can replicate the 5.1TB training data over 1.25 memory nodes per APU [28] in the disaggregated memory system. It has an L:R of 399.

As the AI model size grows exponentially [45], it pushes the AI training workloads to have even larger memory requirements in the future. Therefore, AI training workloads with dense activation layers will result in a high L:R ratio and benefit from memory disaggregation (green zone). Alternatively, AI training workloads with shallow networks will pay the network bandwidth performance penalty on a disaggregated memory system (orange zone).

DASSA: The local similarity method is a time-domain data analysis algorithm developed to detect earthquakes in array seismic datasets [46]. Each input file contains a 2D array (30,000 time samples and 11,648 channels). For each cell in that 2D array, it calculates two correlations and each correlation needs to refer the other cells in a different channel in the window. With a typical window size of five hundred cells, one cell needs to access one thousand cells for its computation. Thus, the number of local memory accesses per cell is one thousand cells. The remote memory access is to stream the input data to the local memory once. As such, the number of remote memory accesses equals the total number of cells. This leads to an L:R ratio of 1,000.

TOAST: The L:R ratio is calculated by profiling the DRAM data movement using the Intel Vtune on one Cori Haswell [31] node and dividing the input size. Thus, the L:R ratio is 278 and the required memory capacity is 1TB.

ADEPT (no-traceback): The core computation of ADEPT is to perform Smith-Waterman (SW) alignments [33]. SW is a dynamic programming algorithm that constructs an $m \times n$ matrix A given two sequences of lengths m and n. The matrix A is used to find the optimal local alignment between the two sequences by listing all possible alignments. When operating in no-traceback mode, ADEPT is able to discard most of the matrix A except the cells needed for the next iteration. When computing the matrix A, the score of any element $A(i,j)$ depends on elements $A(i,j-1)$, $A(i-1,j)$ and $A(i-1,j-1)$. The whole score matrix $(m \cdot n)$ is maintained in the local memory. This leads to the number of local memory access to $3 \cdot m \cdot n$, and the number of remote memory access to $m+n$. In this paper, we use a data set of about 31 million DNA reads and corresponding reference pairs with upper limits of $m = 200$ and $n = 780$ leading to the 63 GB total remote memory requirements with a L:R ratio of 477.

ADEPT (traceback): When operating in traceback mode, ADEPT kernel traces a path connecting the matrix cell with the highest score $(i,j), i > i_0, j > j_0$ back to the starting point (i_0, j_0) in the matrix. For this, the full matrix must be available in the GPU's HBM therefore, this along with increasing total HBM requirements also requires additional memory accesses for traceback. The traceback step usually follows pointers starting from the highest scoring cell and ending when a cell with zero score is found, leading to additional l memory accesses where l represents the longest possible alignment. The longest alignment can at most be of the length 200, i.e. longest possible read in the dataset. This also leads to an approximate L:R ratio of 477.

EXTENSION: MetaHipMer's [32] local assembly phase performs local extensions on sections of DNA with the help of DNA reads. We use the Arctic synth data set [47] with four typical kmer size: 21, 33, 55 and 77 with 45 million extensions for four application runs. As such, one can observe that the L:R ratios vary from 314 to 3,402 with increasing kmer sizes.

PASTIS: We use a dataset that performs around 840 million pairwise alignments. This results in 158TB of local and 363GB of remote memory data movement (an L:R of 435).

GEMM: The general matrix multiplication (GEMM) kernel is defined as the operation $C = A \cdot B$, with A and B as matrix inputs and C as the output. We assume all three matrices are square double-precision matrices (N $\times$ N) with N being the maximum dimension that fits in DDR memory. Using estimates derived from the Holder-Brascamp-Leib (HBL) inequality [48], we may estimate the data movement (R) to/from remote memory as $\frac{2 \cdot N^3}{\sqrt{M}} + N^2 - 3 \cdot M$ elements where M is the local memory (cache) capacity in elements (64G). We apply this model recursively to estimate the local data movement per local GEMM using a 512GB memory and a 40MB cache [49]. This number is scaled by the requisite number of local GEMMs $((\frac{DDR}{HBM})^{\frac{3}{2}})$ to produce the L:R ratio which we observe varies from about 50 to 90.

STREAM: STREAM TRIAD is a benchmark that measures sustainable memory bandwidth. It computes a vector operation $C(i) = A(i) + \alpha \cdot B(i)$. This operation involves two loads ($A(i)$ and $B(i)$) and one store ($C(i)$) in the remote memory. Reads from (writes to) remote memory incur writes(reads) in local memory on top of nominal reads/writes in local memory. Thus, the L:R equals 2.

D. Application Analysis Summary

The case studies represent a diverse array of memory access patterns and memory capacity needs across multiple domains. This trend will likely continue as scientific workloads evolve over time. For the exemplar disaggregated memory system configuration consisting of 10,000 compute nodes and 1,000 memory nodes, we see that ten out of eleven workloads fall into the blue and green zones, that will not suffer penalties from disaggregation. Only the STREAM (>512GB) fall into the orange zone and could see a penalty from disaggregated memory. Although we proxied future L:R ratios using today's problem sizes, we believe future L:R ratios will be at least as big as surface to volume ratios never shrink. Ultimately, these applications are unlikely to see any performance loss from disaggregated memory over the existing state of the practice.

Assuming these applications represent a workload and the applications falling into the green and orange zones constitute

less than 10% of the total workload node hours, then the disaggregate system discussed here could save the cost of 40PB of node-local DDR memory (10K compute nodes×4TB) of memory at the expense of 1000 memory nodes of 4TB each without hurting performance.

V. Discussion and Conclusions

In this paper, we focused on architecture, bottlenecks, and characterization of applications running on disaggregated memory system architectures in the 2024-2026 time frame. As visualized in Fig. 6, 10 out of 11 of the applications we examined either have sufficiently low memory requirements that they can comfortably fit in a future APU's HBM memory or have a sufficiently high local:remote data movement ratio that the architected local:remote bandwidth tapering will not impede performance. Nevertheless, it is imperative HPC system architects and vendors follow a few design principles lest the potential remote bandwidth be underutilized.

Memory Extension: System architects must decide whether remote memory is exposed as a second NUMA node with data movement affected via RDMA (e.g. SHMEM put/get) or uncacheable load/store instructions – or – whether HBM should be viewed as either a hardware-controlled line cache or OS-controlled page cache. The nuance arises in whether applications and processors can express concurrency greater than remote memory's latency-bandwidth product [50] assuming a latency comparable to the 2us observed on a 2021 HPC system and bandwidth varying from PCIe4 to PCIe6.

Inspired by the Roofline model [21], Figure 7 plots the impact of Little's Law on memory bandwidth for varying access quanta (diagonals) and concurrency (vertical lines). System architects must choose a quanta that attains available bandwidth at an application concurrency less than the processor/system concurrency upper bound. For example, an OS cache sustaining only one outstanding page fault (concurrency$\leq$1) will never be able to sustain even PCIe4 bandwidth with 4KB pages. Similarly, an A100 GPU has insufficient LDST concurrency to sustain PCIe5 bandwidth using coalesced 32B cache lines. Ultimately, vendors must provide either larger pages (e.g $\geq$256KB), $\geq$64B cache lines, twice as many LDST units as an A100 GPU, or demand applications continually initiate hundreds of KB-sized asynchronous RDMAs (spread across multiple processes).

Lustre Replacement: Lustre is superfluous for applications requiring either private or read-only file access (e.g. AI training data). Vendors wishing to leverage remote memory nodes as a replacement for distributed file systems must provide a file system interface and guarantee durability for the life of a job (more than an individual executable). Our Roofline-inspired Little's Law analysis (Fig. 7) still applies. That is, assuming file system software overhead is far less than network latency, applications must both continually read/write 256KB blocks in order to sustain PCIe6 bandwidths whilst ensuring the local to file IO data movement ratio exceeds 65.

Inter-Process Communication: Whereas memory and inter-process communication (e.g. MPI) were traditionally architected with dedicated bandwidths, in disaggregated systems, they both contend for finite PCIe bandwidth. Applications with a inter-process to remote memory ratio may see PCIe bandwidth emerge as a bottleneck.

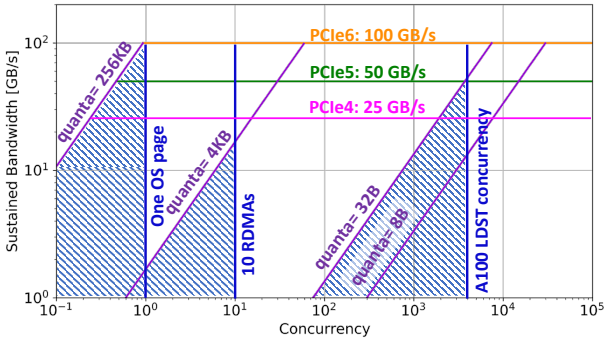


Fig. 7: The Roofline-inspired visualization of Little's Law (Concurrency Roofline) informs architects of viable combinations of network bandwidth, memory access quanta, and concurrency. When the intercept of quanta and concurrency is less than a PCIe bandwidth, sustained bandwidth is impeded (e.g. 4KB OS paging or 32B caching on the A100).

Future Portents: Historically, latency lags bandwidth [51], and to a lesser degree, one expects remote bandwidth to lag local bandwidth. As such, beyond 2026 we expect the latency-bandwidth product (requisite concurrency) to increase nearly as fast as remote bandwidth. Systems in that time frame with require even larger pages, even more concurrent RDMAs (easily realized with more processes per node), or GPUs with even more concurrency (almost guaranteed). Similarly, the hardware local:remote ratio will increase slowly implying some applications may become remote memory-limited. As such, memory disaggregation will likely continue to be a viable approach so long as network bandwidth increases.

Workload Analysis: Whereas this paper focused on analyzing individual applications in a system with disaggregated memory, the ultimate efficacy of such a system is premised on the workload on the system. Practitioners wishing to leverage our methodology should characterize their applications along the lines of Fig. 6 and ratio their compute to memory nodes as the sum of the node hours of all the applications falling into the blue region divided by the sum of the node hours of all the applications falling into the green and orange regions (scaled by memory capacity/4TB). If the scaled node hours of the green and orange regions dominate, then there is little value in disaggregation as such a ratio will demand more memory nodes than compute nodes. Similarly, if the scaled node hours of the orange region dominates, the workload is better served with node-local DDR unencumbered by limited PCIe bandwidths.

For amenable workloads and collaborative vendors, memory disaggregation will provide a cost-effective means of mitigating the dynamic and highly variable memory requirements found in HPC centers.

References

[1] A. C. Elster and T. A. Haugdahl, "Nvidia hopper gpu and grace cpu highlights," *Computing in Science & Engineering*, vol. 24, no. 2, pp. 95–100, 2022.

[2] "NERSC-10 workload analysis," https://portal.nersc.gov/project/m888/nersc10/workload/N10_Workload_Analysis.latest.pdf.

[3] I. Peng, R. Pearce, and M. Gokhale, "On the memory underutilization: Exploring disaggregated memory on hpc systems," in *2020 IEEE 32nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, 2020, pp. 183–190.

[4] G. Michelogiannakis, B. Klenk, B. Cook, M. Y. Teh, M. Glick, L. Dennison, K. Bergman, and J. Shalf, "A case for intra-rack resource disaggregation in hpc," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 19, no. 2, pp. 1–26, 2022.

[5] H. Li, D. S. Berger, S. Novakovic, L. Hsu, D. Ernst, P. Zardoshti, M. Shah, I. Agarwal, M. Hill, M. Fontoura *et al.*, "First-generation memory disaggregation for cloud platforms," *arXiv preprint arXiv:2203.00241*, 2022.

[6] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, "Disaggregated memory for expansion and sharing in blade servers," *ACM SIGARCH computer architecture news*, vol. 37, no. 3, pp. 267–278, 2009.

[7] M. K. Aguilera, N. Amit, I. Calciu, X. Deguillard, J. Gandhi, P. Subrahmanyam, L. Suresh, K. Tati, R. Venkatasubramanian, and M. Wei, "Remote memory in the age of fast networks," in *Proceedings of the 2017 Symposium on Cloud Computing*, 2017, pp. 121–127.

[8] J. Gu, Y. Lee, Y. Zhang, M. Chowdhury, and K. G. Shin, "Efficient memory disaggregation with infiniswap," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017, pp. 649–667.

[9] P. X. Gao, A. Narayan, S. Karandikar, J. Carreira, S. Han, R. Agarwal, S. Ratnasamy, and S. Shenker, "Network requirements for resource disaggregation," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 249–264.

[10] F. V. Zacarias, P. Carpenter, and V. Petrucci, "Memory demands in disaggregated hpc: How accurate do we need to be?" in *2021 International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. IEEE, 2021, pp. 1–6.

[11] K. Lim, Y. Turner, J. R. Santos, A. AuYoung, J. Chang, P. Ranganathan, and T. F. Wenisch, "System-level implications of disaggregated memory," in *IEEE International Symposium on High-Performance Comp Architecture*. IEEE, 2012, pp. 1–12.

[12] C. Wang, H. Ma, S. Liu, Y. Li, Z. Ruan, K. Nguyen, M. D. Bond, R. Netravali, M. Kim, and G. H. Xu, "Semeru: A Memory-Disaggregated Managed Runtime," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 261–280.

[13] "Compute Express Link," https://docs.wixstatic.com/ugd/0c1418_d9878707bbb7427786b70c3c91d5fbd1.pdf.

[14] "Compute Express Link 2.0 White Paper," https://b373eaf2-67af-4a29-b28c-3aae9e644f30.filesusr.com/ugd/0c1418_14c5283e7f3e40f9b2955c7d0f60bebe.pdf.

[15] "PCIE 6.0 vs 5.0 – All you need to know," https://www.rambus.com/blogs/pcie-6/.

[16] S. Van Doren, "Hoti 2019: Compute express link," in *2019 IEEE Symposium on High-Performance Interconnects (HOTI)*. IEEE, 2019, pp. 18–18.

[17] C. Pinto, D. Syrivelis, M. Gazzetti, P. Koutsovasilis, A. Reale, K. Katrinis, and H. P. Hofstee, "Thymesisflow: a software-defined, hw/sw co-designed interconnect stack for rack-scale memory disaggregation," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 868–880.

[18] "Memory Bandwidth and System Balance in HPC Systems: 2021 Update," https://hpc.fau.de/files/2021/12/memorybw_systembalance_slides_2021-12-15.pdf.

[19] M. Daga, A. M. Aji, and W.-c. Feng, "On the efficacy of a fused cpu+ gpu processor (or apu) for parallel computing," in *2011 Symposium on Application Accelerators in High-Performance Computing*. IEEE, 2011, pp. 141–149.

[20] A. Branover, D. Foley, and M. Steinman, "Amd fusion apu: Llano," *Ieee Micro*, vol. 32, no. 2, pp. 28–37, 2012.

[21] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, 2009.

[22] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild *et al.*, "Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence," USDOE Office of Science (SC), Washington, DC (United States), Tech. Rep., 2019.

[23] A. Mathuriya, D. Bard, P. Mendygral, L. Meadows, J. Arnemann, L. Shao, S. He, T. Kärnä, D. Moise, S. J. Pennycook *et al.*, "Cosmoflow: Using deep learning to learn the universe at scale," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2018, pp. 819–829.

[24] T. Kurth, S. Treichler, J. Romero, M. Mudigonda, N. Luehr, E. Phillips, A. Mahesh, M. Matheson, J. Deslippe, M. Fatica *et al.*, "Exascale deep learning for climate analytics," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2018, pp. 649–660.

[25] "MLPerf Training: HPC." https://mlcommons.org/en/training-hpc-07/.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[27] P. Mattson, C. Cheng, G. Diamos, C. Coleman, P. Mi-

cikevicius, D. Patterson, H. Tang, G.-Y. Wei, P. Bailis, V. Bittorf *et al.*, "Mlperf training benchmark," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 336–349, 2020.

[28] K. Z. Ibrahim, T. Nguyen, H. A. Nam, W. Bhimji, S. Farrell, L. Oliker, M. Rowan, N. J. Wright, and S. Williams, "Architectural requirements for deep learning workloads in hpc environments," in *2021 International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. IEEE, 2021, pp. 7–17.

[29] B. Dong, V. R. Tribaldos, X. Xing, S. Byna, J. Ajo-Franklin, and K. Wu, "Dassa: Parallel das data storage and analysis for subsurface event detection." in *IPDPS*, 2020, pp. 254–263.

[30] "TOAST," https://toast-cmb.readthedocs.io/en/toast3/.

[31] "Cori-NERSC Documentation," https://docs.nersc.gov/systems/cori/.

[32] S. Hofmeyr, R. Egan, E. Georganas, A. C. Copeland, R. Riley, A. Clum, E. Eloe-Fadrosh, S. Roux, E. Goltsman, A. Buluç *et al.*, "Terabase-scale metagenome coassembly with metahipmer," *Scientific reports*, vol. 10, no. 1, pp. 1–11, 2020.

[33] M. G. Awan, J. Deslippe, A. Buluc, O. Selvitopi, S. Hofmeyr, L. Oliker, and K. Yelick, "Adept: a domain independent sequence alignment strategy for gpu architectures," *BMC bioinformatics*, vol. 21, no. 1, pp. 1–29, 2020.

[34] M. G. Awan, S. Hofmeyr, R. Egan, N. Ding, A. Buluc, J. Deslippe, L. Oliker, and K. Yelick, "Accelerating large scale de novo metagenome assembly using gpus," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–11.

[35] "Nsight Compute Command Line Interface." https://docs.nvidia.com/nsight-compute/pdf/NsightComputeCli.pdf.

[36] "Cori GPU Nodes," https://docs-dev.nersc.gov/cgpu/.

[37] O. Selvitopi, S. Ekanayake, G. Guidi, G. A. Pavlopoulos, A. Azad, and A. Buluç, "Distributed many-to-many protein sequence alignment using sparse matrices," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020, pp. 1–14.

[38] A. Döring, D. Weese, T. Rausch, and K. Reinert, "Seqan an efficient, generic c++ library for sequence analysis," *BMC Bioinformatics*, vol. 9, no. 1, p. 11, Jan 2008.

[39] N. Ding, Y. Liu, S. Williams, and X. S. Li, "A message-driven, multi-gpu parallel sparse triangular solver," in *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA21)*. SIAM, 2021, pp. 147–159.

[40] N. Ding, S. Williams, Y. Liu, and X. S. Li, "Leveraging one-sided communication for sparse triangular solvers," in *Proceedings of the 2020 SIAM Conference on Parallel Processing for Scientific Computing*. SIAM, 2020, pp. 93–105.

[41] H. Fu, J. Liao, N. Ding, X. Duan, L. Gan, Y. Liang, X. Wang, J. Yang, Y. Zheng, W. Liu *et al.*, "Redesigning cam-se for peta-scale climate modeling performance and ultra-high resolution on sunway taihulight," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[42] "General Matrix Multiplication," https://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms#Level_3.

[43] "STREAM: Sustainable Memory Bandwidth in High Performance Computers," https://www.cs.virginia.edu/stream/.

[44] "Perlmutter," https://docs.nersc.gov/systems/perlmutter/system_details/.

[45] "NVIDIA Hopper Architecture In-Depth," https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/.

[46] Z. Li, Z. Peng, D. Hollis, L. Zhu, and J. McClellan, "High-resolution seismic event detection using local similarity for large-n arrays," *Scientific reports*, vol. 8, no. 1, pp. 1–10, 2018.

[47] S. Hofmeyr, R. Egan, E. Georganas, A. C. Copeland, R. Riley, A. Clum, E. Eloe-Fadrosh, S. Roux, E. Goltsman, A. Buluç *et al.*, "Terabase-scale metagenome coassembly with metahipmer," *Scientific reports*, vol. 10, no. 1, pp. 1–11, 2020.

[48] T. M. Smith, B. Lowery, J. Langou, and R. A. van de Geijn, "A tight i/o lower bound for matrix multiplication," *arXiv preprint arXiv:1702.02017*, 2017.

[49] N. Nvidia, "A100 tensor core gpu architecture," 2020.

[50] D. Bailey, "Little's law and high performance computing," in *RNR Technical Report*, 1997.

[51] D. A. Patterson, "Latency lags bandwith," *Communications of the ACM*, vol. 47, no. 10, pp. 71–75, 2004.

TABLE II: Approaches for characterizing the local and remote memory accesses (L:R) and remote memory capacities

	Workloads	Local memory accesses	Remote memory accesses	Remote memory capacity
AI training	ResNet-50 DeepCAM CosmoFlow	memory usage of the activation layer	sample batch size	Training set size
Genomics workloads	ADEPT (w/wo traceback)	analytical modeling	analytical modeling	input size
	EXTENSION	NVIDIA NSight compute		
Protein workloads	PASTIS	NVIDIA NSight compute	analytical modeling	
Data analysis workloads	DASSA	analytical modeling	analytical modeling	
	TOAST	Intel VTune		
Traditional HPC workloads	GEMM	analytical modeling	analytical modeling	analytical modeling
	STREAM			
NVIDIA NSight compute Metrics		32B*(dram__sectors_read.sum + dram__sectors_write.sum)		
Intel VTune Metrics		64B*sum(UNC_M_CAS_COUNT.WR[UNIT0-7],UNC_M_CAS_COUNT.RD[UNIT0-7])		

APPENDIX
ARTIFACT DESCRIPTION

A. Abstract

The key contribution of this paper is the methodology for evaluating system architecture with disaggregated memory, and evaluating potential and pitfalls for workloads on such a system. The hardware and software environment used in this paper are all publicly available as described below.

B. Description

Approaches: Case study results presented in this paper were obtained using different approaches due to the diversity of the workloads. Table II lists the approaches to measure or estimate the local and remote memory accesses (L:R). Note that all the numbers reported in refer to the memory per job.

The profiling and plotting scripts can be found https://github.com/nanding0701/memory_disaggregation.git. The repo includes the plotting scripts for *disaggregated memory system design space (Fig. 4)* and *capacity and bottleneck visualization of applications on a disaggregated memory system (Fig. 6)*. With the metric table above, these scripts, readers should be able to apply our methodology and analysis to their own kernels or applications.

Machines: TOAST was profiled on the Haswell nodes on Cori at NERSC. EXTENSION and PASTIS's local memory accesses results were obtained on the GPU-accelerated partition on Cori (EXTENSION and Cori-GPU) at NERSC.

Applications: The three profiled workloads in the paper are described below.

1) EXTENSION: the source code can be found https://github.com/mgawan/mhm2_staging. Note that it is the complete MetaHipmer2 application source code with EXTENSION kernel integrated. We profiled single extension for vary kmer sizes and then multiply that with 45 million extensions.
2) PASTIS: the source code can be found https://github.com/PASSIONLab/PASTIS. We run PASTIS using 5 million sequences.
3) TOAST: the source code can be found https://github.com/hpc4cmb/toast/tree/toast3. We run the toast_benchmark_satellite benchmark with a medium case.