Ponte Vecchio Across the Atlantic: Single-Node Benchmarking of Two Intel GPU Systems

Thomas Applencourt
Argonne Leadership Computing Facility
Argonne National Laboratory
Lemont, IL, USA
tapplencourt@anl.gov

Aditya Sadawarte
School of Computer Science
University of Bristol
Bristol, UK
aditya.sadawarte@bristol.ac.uk

Servesh Muralidharan
Argonne Leadership Computing Facility
Argonne National Laboratory
Lemont, IL, USA
servesh@anl.gov

Colleen Bertoni
Argonne Leadership Computing Facility
Argonne National Laboratory
Lemont, IL, USA
bertoni@anl.gov

JaeHyuk Kwack
Argonne Leadership Computing Facility
Argonne National Laboratory
Lemont, IL, USA
jkwack@anl.gov

Ye Luo
Argonne Leadership Computing Facility
Argonne National Laboratory
Lemont, IL, USA
yeluo@anl.gov

Esteban Rangel
Argonne Leadership Computing Facility
Argonne National Laboratory
Lemont, IL, USA
erangel@anl.gov

John Tramm
Argonne Leadership Computing Facility
Argonne National Laboratory
Lemont, IL, USA
jtramm@anl.gov

Yasaman Ghadar
Argonne Leadership Computing Facility
Argonne National Laboratory
Lemont, IL, USA
ghadar@anl.gov

Arjen Tamerus
University of Cambridge
Cambridge, UK
at748@cam.ac.uk

Chris Edsall
University of Cambridge
Cambridge, UK
cje57@cam.ac.uk

Tom Deakin
School of Computer Science
University of Bristol
Bristol, UK
tom.deakin@bristol.ac.uk

***Abstract*—Intel Data Center GPU Max 1550, known as Ponte Vecchio (PVC), is a new Intel GPU architecture for high-performance computing. It is the basis of two systems on the June 2024 Top 500 list, Dawn (#51) and Aurora (#2).**

This work provides micro-benchmarking data on PVCs from which application developers may benefit, shows how the micro-benchmarking results are indicative of mini-app performance on PVC, and demonstrates real applications on large-scale Intel GPU systems.

We quantify the obtainable performance from PVC systems through micro-benchmarking fundamental architectural properties. We evaluate the performance of four mini-apps with known performance characteristics, and two full applications, comparing performance on a node of Aurora and Dawn with a node of NVIDIA H100 GPUs and a node of AMD MI250 GPUs. We show the figure-of-merit of the mini-apps on a single PVC ranges from 0.6–1.8X the performance of an H100, and 0.8–7.5X of a MI250.

I. Introduction

Aurora and Dawn are two recent GPU-based supercomputers at the Argonne Leadership Computing Facility and the University of Cambridge, respectively. They are unique in that while the majority of GPU-based supercomputers utilize Nvidia or AMD GPUs [1], Aurora and Dawn are based on the Intel Data Center GPU Max 1550, a new GPU architecture from Intel. We refer to these GPUs by its architectural codename Ponte Vecchio (PVC) for the remainder of this paper. Nvidia GPUs have been involved in high-performance computing for over a decade (e.g. Titan entered the Top500 list [2] back in 2012). They have been extensively benchmarked [3] [4] [5] forming an excellent resource for application developers. In comparison, the Intel GPUs are new, and very few benchmarking results are available as references. Recent results from Dawn and Aurora in the Top500 provide a glimpse into their potential through the LINPACK and HPCG benchmarks [6], [7]. However, these results are from large machine runs and are not always useful for application optimizations, as discussed by Siefert et al. [8]. Since the architecture behind PVC is new, it is important for application developers targeting these machines to have a reference for their performance characteristics. Using Dawn and Aurora, we present microbenchmarking results so developers can relate them to their own application's performance. Additionally, for users running on Aurora and Dawn, it is important to understand how node-level design differences in dense GPU

See Acknowledgement for funding sources.

systems manifest in performance even while utilizing the same GPU architecture.

To discover the *achievable* upper bounds of PVC on Dawn and Aurora, we present results from a set of microbenchmarks to characterize single node performance. We predominantly focus on single-GPU and single-node performance. Since the configuration of the node, such as the CPUs available, can affect the performance of the GPU, running benchmarks on a full node of the system is important for comparison. The benchmarks were selected to capture the bounds of what applications may hit, including flop-rate, memory bandwidths, communication latency, PCIe bandwidth, general matrix multiplication (GEMM), fast Fourier transform (FFT), and memory access latency. Further, they are written in high level languages (OpenMP, SYCL, MPI) to provide useful information for application developers, rather than lower-level languages whose performance application developers using those high-level languages may be unable to achieve. Our microbenchmarks therefore provide indicative performance measurements of key architectural features on Intel discrete GPU systems.

In addition, we evaluate single-GPU and single-node performance of four mini-apps on Dawn, Aurora, a single-nodes of NVIDIA H100 and AMD MI250 GPUs. Since the mini-apps have known bottlenecks, we can evaluate the performance between the four architectures in terms of the aforementioned microbenchmarks to compare performance between the four systems. We also compare the performance of two full applications across the four architectures on a single node.

To the best of our knowledge, this is the first paper to report an extensive set of high-level microbenchmarks of the PVC on the Aurora and Dawn nodes.

We make the following contributions:

- Quantification of obtainable fundamental performance limits of PVC-based nodes through microbenchmarking using high-level languages in the manner applications would be developed;
- Evaluation of the performance of mini-apps with known characteristics, relating their performance to the results of the microbenchmarks;
- Demonstration of functionality and performance of mini-apps and applications on PVC hardware;
- Performance comparison of two science applications across a single node of four different systems with varying CPU and GPU architectures (an Aurora node (six PVC), a Dawn node (four PVC), a node of four NVIDIA H100s, and a node of four AMD MI250s;
- Discussion of how microbenchmarking can provide insight into application and mini-application performance as well as its limitations.

This paper is organized as follows: we first present related work in Section I-A, then an overview of the PVC architecture in Section II, followed by an overview of the Dawn, Aurora, NVIDIA H100, and AMD MI250 systems employed in Section III. In Section IV we describe the microbenchmarks used and the single-GPU and intra-node scalability on Dawn and Aurora, and in Section V we present the mini-apps used and their performance results. Section VI discusses the applications and results. Section VII concludes.

This paper walks a fine line between a deep dive into the PVC hardware and a thorough analysis of selected applications. It's not meant to be a deep-dive on PVC hardware. The H100 and MI250 performance comparison should be seen as points of reference, and are here to demonstrate that PVC performance is on par with other GPUs.

A. Related work

Although there have been many papers showing the performance of mini-apps or applications on PVC hardware [9]–[12], to the best of our knowledge there has not been a paper providing extensive single-node benchmarking for PVC-based systems. Single node performance papers have been done for other architectures, for example Ref. [13] discusses microbenchmarking of Frontier [14] nodes.

The literature has many examples where architecture comparisons are made directly or indirectly whereby parallel programming models, domain specific languages, abstraction layers, etc, are compared across different GPUs and CPUs. As this is a ripe field, we omit specific references.

II. Intel Data Center GPU Max 1550 Series

In this section we discuss the architecture and nomenclature of the Intel Data Center GPU Max 1550 Series, previously codenamed Ponte Vecchio (PVC) [15]–[17]. More details about the lower-level architecture are discussed in Ref. [16] The PVC GPU is structured in a hierarchical manner, similar to the previous generations of low-power integrated Intel GPUs such as Gen9 [18]. The basic element of the PVC is the Xe-Core. The Xe-Core is composed of eight vector and matrix engines, with a total register file of 512 KB. The vector unit is 512-bits wide (8-wide for double precision) and supports FMAs. The matrix unit is 4096-bits wide and supports only lower precision operations. The register file can be partitioned among hardware threads in two different ways: with 8 active hardware threads with 128 registers each, or 4 active hardware threads with 256 registers each. Since each vector engine is able to perform two double precision FMAs per clock, together all the vector engines in each Xe-Core can perform 256 double precision floating point operations per clock (8 vector engines/Xe-Core $\times$ 8 (512 bits SIMD)$\times$2 (FMA)$\times$2). The maximum GPU clock speed is 1.6 GHz, but this can downclock due to TDP (Thermal Design Power) constraints.

Sixteen Xe-Cores are grouped into a Xe-Slice. Four Xe-Slices are grouped into a Xe-Stack. The Xe-Stack contains a shared L2 (LLC) cache, HBM2e memory controllers, high speed Stack-to-Stack interconnect within the same silicon substrate and Xe-links high-speed coherent fabric for remote GPU to GPU communication. The Stack is similar in concept to a AMD GCD. Each Xe-Stack contains 192MiB LLC and also connects to its own HBM2e memory stacks and is physically tagged to its local last level cache. Then, two Xe-Stacks (incl. its local HBM2e memory stacks) are bundled together into an Intel Data Center GPU Max 1550 card, for a total of 128

Xe-Cores and 32,768 double precision and single precision floating point operations per clock. Only the first Xe-Stack contains the PCIe link to connect the card to the host. Data movement from the second Xe-Stack needs to go via the high-speed Stack-to-Stack interconnect before reaching the host via the first Xe-Stack's PCIe link to the host. The Intel Data Center GPU Max 1550 card supports PCIe Gen5 link speed for the interface to the host.

Throughout this paper, we will benchmark the PVC device in a number of ways, including as a single Xe-Stack (”One Stack”), both stacks on a single PCIe card (”One PVC”), as well as multiple GPUs in the node. We typically run one MPI process per Stack, known as ”explicit scaling” [19].

III. Systems Employed

Four different systems were used in this work: two PVC systems: Dawn and Aurora; one H100, and one MI250. We used the H100 node and MI250 node of the **Joint Laboratory System for System Evaluation** (JLSE). JLSE is a testbed system at the Argonne Leadership Computing Facility which is composed of nodes of a variety of architectures [20]. The Aurora work was done on a pre-production supercomputer with early versions of the Aurora software development kit.

Dawn is composed of 256 nodes, each with two 48-core Intel Xeon Platinum 8468 CPUs with a total of 1024GB DDR and four PVC with 128 GB HBM each [21]. As an operational setting, each PVC card is power-capped to 600W. The GPUs are connected all-to-all with Xe-Link. The default software used was Intel oneAPI 2024.1 public release.

Aurora contains roughly forty times the number of nodes as Dawn, with 10,624 total nodes [22]. Each node has two 52-core (104-thread) Intel Xeon Gold 5320 CPUs with 64GB HBM and 512GB DDR5 each [23]. Unlike Dawn's 4 PVC per node, Aurora has 6 PVC (with 128 GB HBM each) per node. However, while all 64 Xe-Cores per Xe-Stack on the Dawn PVC are active, on Aurora each Xe-Stack has only 56 Xe-Core activated. Like on Dawn, all PVC are connected all-to-all inside a node. The topology will be discussed more in the peer-to-peer MPI micro-benchmark section. Another difference from Dawn is that each PVC is set to have an idle frequency of 1.6 GHz and each PVC on Aurora is power capped to 500W. The default software used was also the Intel oneAPI 2024.1 public release.

JLSE H100 node The JLSE-H100 node is composed of two Intel Xeon Platinum 8468 CPUs (48 cores/96 threads) with 512GB DDR5 total and four NVIDIA H100 SXM5 80GB GPUs. The default software used was NVHPC 24.1 and CUDA 12.3.0.

JLSE MI250 node The JLSE-MI250 node is composed of two AMD EPYC 7713 64c CPUs (64 cores/128 threads) with 512 GB DDR4 and four AMD Instinct MI250 GPUs. The default software used was ROCm 6.1.0.

IV. microbenchmarks

The goal of the microbenchmarks in this paper is to measure the upper bounds of achievable performance for applications running on PVCs. To achieve this, we implement the benchmarks in high-level language programming models (OpenMP, SYCL, and MPI) rather than lower-level programming models, such as assembly code, in order to give an “unlikely-but-still-achievable” upper bound on the performance of applications. The benchmarks are available on GitHub [24] for comparison and further development.

The focus of this paper is to illustrate the performance of PVC through these microbenchmarks. To properly compare the reasons why these microbenchmarks differ in performance to other vendor GPUs requires a much lower-level design discussion on the microarchitecture of PVC so it's beyond the scope of this paper. However, to present where PVC stands in comparison, we provide flop-rates and memory and PCIe bandwidths in Table IV from official AMD and Nvidia sources [25], [26] and Frontier [13].

Our microbenchmark codes are new ports of industry-standard algorithms used for benchmarking (stream triad, chain of FMAs, data-transfert). We ported these codes to SYCL and/or OpenMP. The benchmark source codes are available on github at Ref. [24].

A. Microbenchmark Evaluation Framework

A summary of the microbenchmarks evaluated is presented in Table I. Each microbenchmark is executed multiple times and the best performance number is presented. This avoids run-to-run variations and any other intermittent artifacts. In addition, binding the MPI ranks to the CPU closest to the GPU ensures data transfer doesn't happen between CPU sockets. For example, Aurora uses CPU cores 0 and 52 (the first core from each CPU socket) for OS kernel threads. Therefore, rank 0 is bound to CPU core 1 and PVC 0 Stack 0. Each Stack is mapped to one MPI rank. The `ZE_AFFINITY_MASK` environment variable (similar to `CUDA_VISIBLE_DEVICES`) is used to control the visibility of each Stack to each rank.

1) Peak Floating Point Operations per Second (FLOPs): This OpenMP microbenchmark performs a chain of Fused Multiple Add instructions (similar to clpeak [27]). Each kernel performs 16×128 FMA operations using single and double precision floating point values.

2) Device Memory Bandwidth: We measure bandwidth to/from the device local High Bandwidth Memory (HBM) though a simple triad (two loads, one store) kernel in OpenMP loading 805 MB (192 *1024*1024 Bytes (LLC per Stack) * 4 (STREAM factor)) of double precision values per array.

3) Host to Device Transfer Bandwidth: This benchmark measures the time to transfer data over the PCIe bus, 500 MB in the case of host-to-device, device-to-host, or a total of 1 GB when transferred simultaneously in both directions. We use `sycl::malloc_host()` for the host memory. This is internally implemented by a call to `ze_malloc_host()` [28] an equivalent to Nvidia pinned memory.

4) Device to Device Transfer Bandwidth: The time taken to transfer data from one device to another is a key performance metric. As described in Section II, the design of the PVC consists of dual Stack architecture; therefore, we can have two scenarios of data transfer between Stacks. The first case is when the Stack is part of the same PVC then we measure the Stack to Stack bandwidth. The second case is when the Stack is part of a remote PVC then we measure the bandwidth achieved over Xe-link. In order to test these scenarios using a high-level benchmark, we use MPICH with Level Zero support [29] that can transfer GPU buffers using the MPI routines. Nonblocking routines such as `MPI_Isend()` and `MPI_IRecv()` are used to transfer messages of 500 MB in size to ensure sufficient overlap of send and receive communication calls.

An important caveat to note here is that, at a high level, it can be seen that all Stacks are connected in an all-to-all manner via Xe-link; however, due to the physical placement of the GPUs, certain Stacks even if they correspond to a similar location between GPUs, could require an extra hop to reach. Let's assume the notation of `GPU_ID.STACK_ID` to refer to each PVC and its Stack uniquely. At the hardware level, each Stack belongs to one of two planes. If we look at the connectivity pattern on Aurora, the two planes consist of 0.0, 1.1, 2.0, 3.0, 4.0, 5.1 for the first plane and 0.1, 1.0, 2.1, 3.1, 4.1, 5.0 for the second. Even though 0.0 and 1.1 Stack are in different positions, since they are physically close to each other, they are connected in a single plane. In such cases, to transfer data from 0.0 to 1.0, the driver can use one of two possible paths: $0.0 \rightarrow 1.1 \rightarrow 1.0$ or $0.0 \rightarrow 0.1 \rightarrow 1.0$.

5) General Matrix Multiplication (GEMM): GEMM is used to measure floating-point (FP64, FP32, FP8, BF16, and TF32) and small integer (I8) operation throughput. We use a square $N \times N$ matrix of size $N = 20480$. The matrix size is large enough such that even the smallest data size (I8) still saturates the PVC's compute throughput. The GEMMs are implemented using the oneMKL library and the SYCL programming language. A total of $2 * N^3$ floating point operations is expected to be performed.

6) Fast Fourier Transform (FFT): We test Forward and Backward FFTs using a size of 4096 and 20,000 for 1D FFTs, and 10,000 for 2D FFTs. We use the standard Cooley-Tukey FFT of $5 \times N \times \log_2 N$ number of flops for complex transform and $2.5 \times N \times \log_2 N$ for real. Similar to the GEMM benchmark, FFTs are implemented using oneMKL and in the SYCL programming model.

7) Memory Latency: The `lats` [30] benchmark measures the memory access latency by chasing pointers on arrays of various lengths to determine the different levels of the memory hierarchy. It was originally designed to chase the pointers in a ring; in a similar fashion on both the CPU and GPU architectures (i.e., on a single thread). We modified this benchmark to perform the same operation simultaneously on one sub-group or warp (Coalesced Access) with 16 work-items, reflecting the memory access patterns on modern GPUs [31].

B. Microbenchmark Results

The microbenchmarks were run on Aurora and Dawn. In both cases, the benchmarks were run on a single Stack, a single PVC (two Stacks), and all PVCs on a single node (four on Dawn and six on Aurora).

1) Stack Linear Scaling: As expected, flops and memory bandwidth scales linearly with the number of Stacks (recall each Stack has its own local HBM memory).

For flops on Aurora, we observe $97\% = 33/(17 \times 2)$ scaling efficiency for two Stacks, and $95\% = 33/(17 \times 12)$ for the full node (we see 92% and 88% scaling efficiency, respectively, on Dawn). 17 Tflop/s is 99% of the expected theoretical number: 1.2 GHz $\times 448$ (vector engines per Stack) $\times 8$ (512-bit SIMD) $\times 2$ (FMA) $\times 2 = 17$ TFlop/s. The third row of Table II shows perfect scaling of main memory bandwidth with Stack count on Aurora and Dawn.

2) Peak Flops performance: As per design specifications, PVC is expected to have the same throughput for both FP32 and FP64 computations [17]. However, we observe the ratio between single and double precision Flops is 1.3x (23/17) on a single Stack on Aurora. This can be explained by the GPU running at a lower frequency during FP64 FMA computations due to the TDP design of the platform. To confirm this, we measure the PVC operating frequency. We saw that the PVC operated at $\sim$1.2GHz for FP64 and $\sim$1.6GHz for FP32 FMA operations. The ratio of flops on Aurora and Dawn roughly follows the ratio of the number of Xe-Cores per Stack: $17/20 = 0.85 \approx 56/64 = 0.875$.

Full node scaling efficiency was good (95% $(195/(17 \times 12))$ on Aurora, 87% $(140/(20 \times 8))$ for Dawn). The FP64 flop-rate on one PVC card (two Stacks) is roughly equivalent to the double-precision theoretical performance on an H100 (30–34 TFlop/s), while the FP32 flop-rate is similar to an MI250 (45 TFlops); see Table IV.

3) Device Memory Bandwidth: The stream HBM bandwidth (1 TB/s) is low compared to the HBM2e spec and official Intel specification for PVC (3 TB/s [15]). In comparison, MI250x on Frontier reach 1.3 TB/s per GCD (Table IV), matching the expected 80% of the theoretical peak.

4) Host to Device Transfer Bandwidth: The PCIe bandwidth between the host CPU and the GPU scales poorly for the full node, $40\% = 264/(53 \times 12)$, suggesting some contention on the host side. PCIe is a full duplex protocol, so the bi-directional bandwidth is expected to be twice that of uni-directional. However, we observe only 1.4x bandwidth for bi- vs. uni-directional. Overall, since the PVC operates at PCIe Gen5, it does achieve a higher bandwidth in comparison to MI250x, which uses PCIe Gen4 (reaching 25 GB/s uni-direction, Table IV).

5) General Matrix Multiplication (GEMM): SGEMM reaches nearly 95% of the peak, and DGEMM reaches nearly 80% of the measured peak. The relative drop of DGEMM efficiency is currently unexplained. It could be possible that the hardware operates at a lower frequency for DGEMM due to TDP or a potential limitation of the hardware for the double-precision pipeline. The investigation is beyond the scope of this paper. In comparison to Frontier, on one GCD of MI250x GEMM reaches 24.1 TFlop/s for double precision and 33.8 TFlop/s for single precision (Table IV). Thus, the

TABLE I: Summary of microbenchmark results

	Programming Model	Description
Peak Compute	OpenMP	Chain of FMA to measure FLOPS
Device Memory Bandwidth	OpenMP	Triad used for HBM bandwidth
Host to Device Transfer Bandwidth	SYCL	Compute the Bandwidth of the PCIe datatransfer
Device to Device Transfer Bandwidth	SYCL	Measure the Bandwidth between 2 Ranks (Stacks on the GPU & between GPUs)
General Matrix Multiplication (GEMM)	SYCL	DGEMM, SGEMM, ...
Fast Fourier Transform (FFT)	SYCL	Backward and forward
Lats	SYCL, CUDA, HIP	Measure the access latency of different levels of the memory hierarchy

TABLE II: Microbenchmark Results except Point to Point

	Aurora (PVC)			Dawn (PVC)		
	One Stack	One PVC	Six PVC	One Stack	One PVC	Four PVC
Double Precision Peak Flops	17 TFlop/s	33 TFLop/s	195 TFlop/s	20 TFlop/s	37 TFlop/s	140 TFlop/s
Single Precision Peak Flops	23 TFlop/s	45 TFlop/s	268 TFlop/s	26 TFlop/s	52 TFlop/s	207 TFlop/s
Memory Bandwidth (triad)	1 TB/s	2 TB/s	12 TB/s	1 TB/s	2 TB/s	8 TB/s
PCIe Unidirectional Bandwidth (H2D)	54 GB/s	55 GB/s	329 GB/s	53 GB/s	54 GB/s	218 GB/s
PCIe Unidirectional Bandwidth (D2H)	53 GB/s	56 GB/s	264 GB/s	51 GB/s	53 GB/s	212 GB/s
PCIe Bidirectional Bandwidth	76 GB/s	77 GB/s	350 GB/s	72 GB/s	72 GB/s	285 GB/s
DGEMM	13 TFlop/s	26 TFlop/s	151 TFlop/s	17 TFlop/s	30 TFlop/s	120 TFlop/s
SGEMM	21 TFlop/s	42 TFlop/s	242 TFlop/s	25 TFlop/s	48 TFlop/s	188 TFlop/s
HGEMM	207 TFlop/s	411 TFlop/s	2.3 PFlop/s	246 TFlop/s	509 TFlop/s	1.9 PFlop/s
BF16GEMM	216 TFlop/s	434 TFlop/s	2.4 PFlop/s	254 TFlop/s	501 TFlop/s	2.0 PFlop/s
TF32GEMM	107 TFlop/s	208 TFlop/s	1.2 PFlop/s	118 TFlop/s	200 TFlop/s	850 TFlop/s
I8GEMM	448 TIop/s	864 TIop/s	5.0 PIop/s	525 TIop/s	1.1 PIop/s	4.1 PIop/s
Single-precision FFT C2C 1D	3.1 TFlop/s	5.9 TFlop/s	33 Tflop/s	3.6 TFlop/s	6.6 TFlop/s	26 TFlop/s
Single-precision FFT C2C 2D	3.4 TFlop/s	6.0 TFlop/s	34 Tflop/s	3.6 TFlop/s	6.5 TFlop/s	25 TFlop/s

TABLE III: Microbenchmark Results for Stack to Stack Point to Point Communication

	Aurora (PVC)		Dawn (PVC)	
	One Stack-Pair	Six Stack-Pairs	One Stack-Pair	Four Stack-Pairs
Local Stack Unidirectional Bandwidth	197 GB/s	1129 GB/s	196 GB/s	786 GB/s
Local Stack Bidirectional Bandwidth	284 GB/s	1661 GB/s	287 GB/s	1145 GB/s
Remote Stack Unidirectional Bandwidth	15 GB/s	95 GB/s	-	-
Remote Stack Bidirectional Bandwidth	23 GB/s	142 GB/s	-	-

TABLE IV: Performance characteristic of Nvidia H100 [25] AMD MI250 [26] and AMD MI250x GPUs [32] . H100 and MI250 are theoretical, MI250x are measured [13]

	H100	MI250	1x GCD MI250x
FP32 peak	67.0 Tflop/s	45.3 Tflop/s	-
FP64 peak	34.0 Tflop/s	45.3 Tflop/s	-
SGEMM	-	-	33.8 TFlop/s
DGEMM	-	-	24.1 TFlop/s
Memory BW	3.4 GB/s	3.2 TB/s	1.3 TB/s
PCIe BW	128.0 GB/s	64.0 GB/s	25.0 GB/s
GCD to GCD	-	-	37.0 GB/s

GEMMs on one GCD of MI250x is ∼50% faster than a PVC Stack. However, the MI250X GEMM makes use of the matrix core units [13], which have twice the peak of the non-matrix cores. If we compare with MI250x's theoretical peak double precision matrix performance (48 Tflop/s per GCD [32]), the efficiency is lower (50% versus GEMM on PVC is 80%).

6) Memory Latency: The memory latency measured in cycles is shown in Figure 1. Notice how the latency significantly increases with each hierarchy in the cache. On all cache levels, both Dawn and Aurora consistently perform within 1–2% of each other, as expected, since it's the same architecture.

The graph shows that the Xe-Core on Dawn and Aurora has a L1 cache of 512KiB (2^{19} bytes), which matches the specification of the hardware. The figure highlights that this is larger than the other GPUs in this study. The L1 cache has 90% higher latency than the H100 GPU and about 51% lower than the MI250. Likewise, the L2 is also larger, with a latency of 50% and 78% higher than the H100 and MI250, respectively. Understanding the cache occupancy and footprint of data in non-synthetic applications is therefore crucial in understanding their expected performance based on the data shown here. The HBM2e on PVC shows 23% and 44% higher access latency than HBM3 on the Nvidia GPUs and HBM2e on AMD GPUs, respectively.

7) Device to Device Transfer Bandwidth: The local Stack to Stack bi-directional bandwidth (shown in Table III) reaches 55% efficiency compared to the $2 * 197$ that is expected. The parallel efficiency is scaling linearly as expected with the number of pairs (95% parallel efficiency). The Xe-Link (the link connecting a Stack to a remote Stack) bandwidth is much

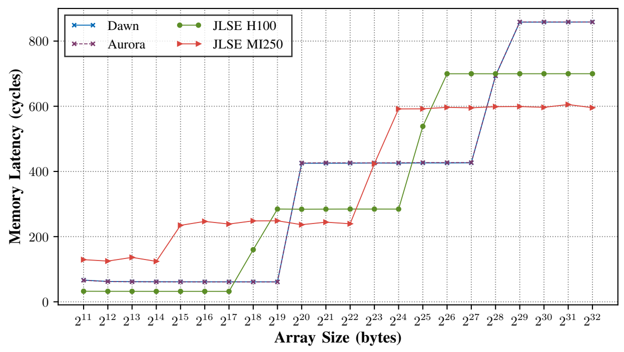

Fig. 1: Memory Latency

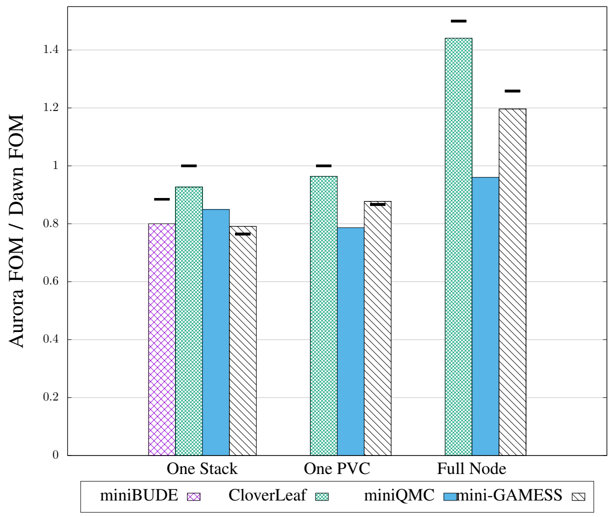

Fig. 2: Figure of Merits on Aurora Relative to Dawn. The black bars are the expected relative performance based on the microbenchmarking results.

slower. They are in fact slower than PCIe, and they reach 55% efficiency in each direction. In comparison, MI250x in Frontier nodes reach 37GB/s for GCD to GCD communication shown in Table IV.

V. Mini-Apps

We evaluate mini-apps across the four systems described in Section III. If a mini-app used a different software configuration than mentioned previously, it is noted below. The build and run scripts for the mini-apps are available in Ref. [24].

A. Mini-Apps Descriptions

The main characteristics of the mini-apps and their Figure-of-Merits (FOMs) are summarized in Table V. In this section the details of the mini-apps and how they were run is provided.

1) miniBUDE: miniBUDE [33] is a mini-app for BUDE, the Bristol University Docking Engine. BUDE is a GPU-accelerated program for performing *in silico* molecular docking, a computational technique used to predict the structure of a complex formed between two molecules and gauge the strength of their interaction [34]. miniBUDE performs virtual screening on the NDM-1 protein by repeatedly evaluating the energy of a single generation of poses for a number of iterations, rendering it compute bound.

In this study, we use an input deck of 2672 ligands, 2672 proteins and 983040 poses. This is run with a combination of poses per work-item (ppwi) and work-group sizes to find the fastest result. The number of interactions (in Billion Interactions/s) associated with this result is the FOM. On Intel PVC, the SYCL port of miniBUDE is used, and on NVIDIA H100 and AMD MI250 we use CUDA and HIP respectively.

2) CloverLeaf: Cloverleaf [35] is a Lagrangian-Eulerian hydrodynamics benchmark, which represents a memory-bandwidth-bound workload. Originally written in Fortran, the mini-app computes the solution of compressible Euler equations; a system of four partial differential equations representing the conservation of energy, density, and momentum. This study uses a variant of the Cloverleaf benchmark written in C++ [36] and ported to various parallel programming models.

A grid of size 15360 ($\approx$ 47GB) is solved on each rank, and the results are weakly scaled up to a full node. This large problem size has been selected to minimise the overhead incurred by MPI communication. The number of cells divided by the total runtime represents the Figure of Merit.

3) miniQMC: miniQMC contains a simplified but computationally accurate implementation of the real space quantum Monte Carlo algorithms implemented in the full production QMCPACK application [37], [38]. The mini-app is designed to enable tests and benchmarks of different programming methodologies, optimizations and algorithms. The OpenMP offload branch is used in this paper for evaluating GPU performance. The mixed precision feature is turned on to exercise FP32 performance. The FOM is defined as $N_{\text{walkers}} \times N^3_{\text{elec}}/T_{\text{diffusion}}$ and the simulation uses a 2x2x1 cell and 320 walkers per GPU. The computation is weak scaled with MPI on every Stack.

4) GAMESS RI-MP2 mini-app (mini-GAMESS): GAMESS (General Atomic and Molecular Electronic Structure Software) [39] is a freely-available software package which implements many quantum chemistry methods. To help explore offloading GAMESS to GPUs, a mini-app for the RI-MP2 method was developed, and it implements the computation of the perturbative correction. The main portion of the mini-app is a call to DGEMM and a reduction, which has been offloaded to GPUs using OpenMP, as discussed in [40]. In this study, we used the code from [41] with branch $SC24\_PMBS$. For the performance comparison, the FOM is defined by $1/\text{walltime}(h)$, and a single input (i.e., W90.rand, an artificial input with the same data structure of 90 water clusters) was used on a single stack of GPU, a single GPU with two stacks, and a single node with multiple GPUs.

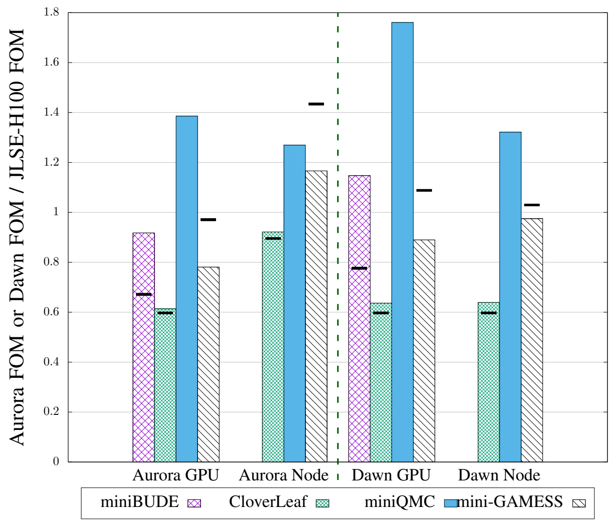

Fig. 3: Figure of Merits on Aurora and Dawn Relative to JLSE-H100. The black bars are the expected relative performance based on the microbenchmarking results and theoretical peaks.

B. Mini-Apps Results

The results of running the mini-apps are summarized in Table VI.

1) Comparing Aurora to Dawn: The Figure of Merits on Aurora relative to Dawn for one Stack, one PVC, and full nodes are shown in Fig. 2. The black bars above each column designate the expected relative performance of the mini-app based on the bounds of each mini-app (Table V) and the microbenchmarks in Table II. For example, miniBUDE is a single precision (FP32) flop-rate bound mini-app, and thus the expected relative performance is the ratio of the peak single precision performance on Aurora to that on Dawn, 0.88X (23 Tflops/s/26 Tflop/s). Note that miniBUDE is not an MPI application, so we run only one rank and target one Stack. We see that in general the black expected performance bars are close to the columns, and thus mini-apps are performing as expected from Dawn to Aurora. miniQMC does not have the expected performance bars in Figure 2, since it is affected by CPU congestion and GPU instruction throughput (discussed more below), and this is not captured by the microbenchmarks.

Unlike the other mini-apps, the FOM of miniQMC on six GPUs on Aurora is less than that on four GPUs on Dawn. This is caused by the fact that resources on each CPU socket are shared by more GPUs attached to it on Aurora. Due to some remaining computation on the CPU and CPU-GPU data transfers, shared DDR and PCIe transfer buses further penalize the intra-node weak scaling performance on Aurora. The high GPU to CPU ratio doesn't benefit miniQMC. In this case, none of the microbenchmarks represented the CPU congestion bottleneck in this mini-app. This is a limitation of microbenchmarking single features whereby all the effects of the full node configuration for certain codes are not captured.

2) Comparing a Dawn and Aurora node to an Nvidia node: Figure 3 shows the FOMs relative to JLSE-H100 for one PVC to one H100 and one Aurora or Dawn node to one JLSE-H100 node. The performance of a single PVC on Aurora and Dawn relative to an H100 ranges from 0.6x and 1.8x, with Cloverleaf showing the lowest relative performance, and miniQMC showing the highest. For a single node, the lowest relative performance is 0.6x (Cloverleaf) and the highest is 1.3x (miniQMC). If intra-node scaling was perfect, we would expect the ratios to stay the same. However miniQMC has lower intra-node scaling on the Aurora and Dawn nodes than the H100 node so the ratio is lower for the full node vs. the single GPU. Note that for miniBUDE, since the application is not MPI, we doubled the single-Stack value to get a full PVC value.

The black bars are the expected relative performance, computed from the ratios of the measured microbenchmarks in Table II and the theoretical peaks for H100 in Table IV. For example, for Cloverleaf (bound by memory bandwidth) on a single GPU, the measured memory bandwidth on a PVC on Aurora and Dawn is 2 TB/s, while for H100 the memory bandwidth used is 3.35 TB/s (from Table IV). Thus the expected ratio is 0.59, which is where the black bar is in the figure. If the values are close the to black bars, then the mini-apps are performing as expected. If the value is above (below) the black bar, then it is performing better (worse) on Dawn/Aurora than expected. Since we use the theoretical value for H100 instead of the measured values, the black bars are a lower bound since the measured values are likely lower than the theoretical ones. In general, the mini-apps are performing as expected, but we see miniBUDE performing better than expected, and mini-GAMESS performing worse than expected.

For miniBUDE, the results on Aurora and Dawn place them around 45% and 49% of their peak single precision flops, which is close to the expected performance ($\sim$50% peak). On the other hand, H100 reaches 30% of its peak, which is much lower than the PVC GPUs. We also performed a similar test on an A100, which reached 62% of its peak. Hence, we expect a deeper evaluation is necessary for miniBUDE on the newer platforms to identify the source of this decreased floating point efficiency.

3) Comparing a Dawn and Aurora node to an AMD node: Figure 4 shows the FOMs relative to JLSE-MI250. Relative to a MI250 GCD, the performance of a single Stack on Aurora and Dawn range from 0.8x to 7.5x, with again Cloverleaf as the lowest and miniQMC as the highest. For a single node, the performance relative to JLSE-MI250 for Aurora and Dawn ranges from 0.8x to 18x. Here, miniQMC scales better on Aurora and Dawn than on JLSE-MI250. For miniQMC, H100 performance is on par with a single PVC Stack while MI250 is significantly penalized by software inefficiency (an order of magnitude slower). The mini-GAMESS MI250 FOM results are absent since it failed to build with the AMD Fortran compiler. As in the previous figures, the black bars in Figure 4 indicate the expected relative performance based on the ratios

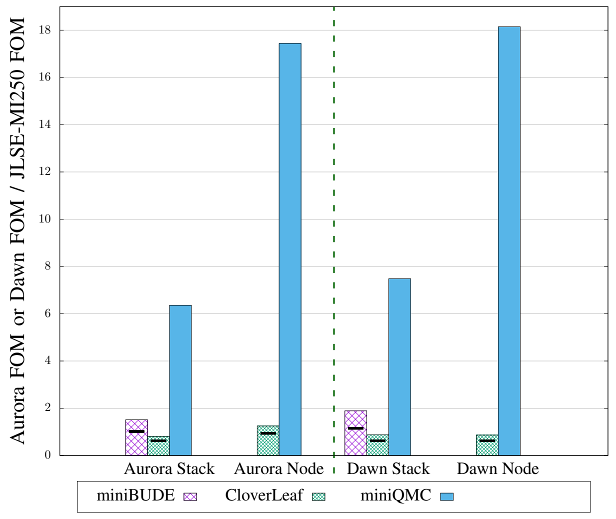

Fig. 4: Figure of Merits on Aurora and Dawn Relative to JLSE-MI250. The black bars are the expected relative performance based on the microbenchmarking results and theoretical peaks.

of the microbenchmarks and the MI250 peaks. We see that for miniBUDE and Cloverleaf, the relative performance on Aurora and Dawn compared to MI250 is close or above what we expect. Similar to on H100, miniBUDE reached about 26% of single-precision floating point peak, while we expect it to reach about 50% of peak. As with H100, we expect a deeper evaluation is necessary.

4) Miniapp conclusion: In general, the relative performance of the mini-apps performed as expected based on the ratios of the microbenchmarks, especially for comparing Aurora to Dawn. However, using the results of the microbenchmarks to predict mini-app performance had limitations. In particular, as discussed above, running miniQMC resulted in a different bottleneck than expected (CPU congestion) which was not captured by the microbenchmarks.

Although (as mentioned in Section IV), the peak performance of MI250x is 50% higher in Flop/s for GEMMs and the memory bandwidth 30% higher, we see that for the mini-apps, performance is competitive with or better than MI250, which has a similar specification to MI250x. We plan to investigate the performance difference in a future paper.

VI. Applications

A. Science Application Descriptions

The main characteristics of the applications evaluated in this paper are summarized in Table V. In this section the details of the applications and how they were run is provided.

1) OpenMC: OpenMC is a Monte Carlo neutral particle transport code capable of neutron and photon transport [42]. It is used in a variety of scientific and engineering fields, in particular in support of both fission and fusion energy system design and analysis. The Monte Carlo method employed by OpenMC is the “gold standard” for fidelity in particle transport, as it uses the bare minimum of physical approximations. While Monte Carlo is highly accurate and can be used in a general purpose manner for nearly any transport problem, its primary downside is its high computational cost. In this light, recent years have seen a significant focus on development of GPU offloading capabilities with OpenMC [43]. OpenMC uses the OpenMP target offloading model, allowing it to run and scale efficiently on Intel, NVIDIA, and AMD GPU architectures [44].

2) CRK-HACC: The Hardware/Hybrid Accelerated Cosmology Code (HACC) [45] is an N-body simulation code designed for large-scale structure formation studies. Originally developed for gravity-only simulations, CRK-HACC [46] now incorporates gas hydrodynamics using a modern smoothed-particle hydrodynamics (SPH) approach called conservative reproducing kernel SPH (CRKSPH). With this new approach, some discrepancies with grid-based hydrodynamic schemes have been resolved, while maintaining the scaling and performance of a particle-based scheme.

B. Science Application Results

The results of running the applications on full nodes of Aurora, Dawn, JLSE-H100, and JLSE-MI250 are summarized in Table VI. Here we discuss the results of the runs.

1) OpenMC: We assess the performance of OpenMC on a small modular reactor (SMR) benchmark problem featuring depleted fuel, as defined in [44]. The figure of merit is derived from the rate of execution of the program when in the “active” phase of the simulation that involves highly complex tallying operations, and is measured in units of thousands of particles per second. The results of OpenMC in Table VI demonstrate the excellent performance of OpenMC on the Aurora PVC architecture as compared to the NVIDIA and AMD nodes, with the Aurora 6× PVC node design offering 1.7× the performance of the JLSE 4× H100 node design.

2) CRK-HACC: We evaluate the performance using two cosmological adiabatic N-body simulations: 2×480^3 particles for a 12 rank configuration and 2×400^3 particles for 8 ranks. The typical CRK-HACC simulation assigns one MPI rank per accelerator device and requires at least 8 MPI ranks to run correctly. On Aurora, 12 MPI ranks map to each of the PVC stacks; on Dawn and JLSE-MI250 nodes, 8 MPI ranks map to each logical device (stack/GCD). On JLSE-H100 we assign 2 ranks per GPU. OpenMP is used for host-side CPU operations, evenly dividing the available cores on each system. The SYCL implementation of CRK-HACC [47] is used on Dawn and Aurora, and the CUDA and HIP implementations are used on JLSE-H100 and JLSE-MI250.

When comparing Dawn and Aurora nodes to the JLSE-H100 and JLSE-MI250 systems, we calculate the scaled performance of the GPUs. Single precision theoretical peak multiplied by the aggregate GPU time is used. Relative to the JLSE-H100 node, the Dawn and Aurora single-GPU performance is 0.954 and 0.947; relative to the JLSE-MI250 node, performance is 0.987 and 0.981. The FOM results in

TABLE V: Mini-App and Application Descriptions (See Mini-app and Application Descriptions in the text for more details)

	Science Domain	Language	Programming model	Characteristic	Scaling	Figure-of-Merit
miniBUDE	BioChemistry	C++	SYCL, HIP, CUDA	FP32 flop-rate bound	N/A	$\frac{\textit{Billion Interactions}}{time(s)}$
CloverLeaf	Computational Fluid Dynamics	C++	SYCL, HIP, CUDA	Memory bandwidth bound	Weak	$\frac{N_{cells}}{time(s)}$
miniQMC	Material Science	C++	OpenMP	Compute/Memory BW bound CPU congestion bound	Weak	$\frac{N_w N_e 10^{-11}}{\textit{diffusion time}(s)}$
GAMESS RI-MP2 mini-app	Quantum Chemistry	Fortran	OpenMP	DGEMM bound	Strong	$\frac{1}{time(h)}$
OpenMC	Particle Transport	C++	OpenMP	Memory latency/bandwidth bound	Weak	$\frac{\textit{Thousand particles}}{time(s)}$
HACC	Cosmology	C++	SYCL, HIP, CUDA	CPU memory BW bound, GPU FP32 flop-rate bound	Weak	$\frac{N_p N_{steps}}{time(s)}$

TABLE VI: Mini-App and Application Figure-of-Merits Across Aurora, Dawn, and JLSE (Units for FOM are in Table V)

	Aurora (PVC)			Dawn (PVC)			JLSE (H100)		JLSE (MI250)	
	One Stack	One GPU	Six GPU	One Stack	One GPU	Four GPU	One GPU	Four GPU	One GCD	Four GPU
miniBUDE	293.02	-	-	366.17	-	-	638.40	-	193.66	-
CloverLeaf	20.82	40.41	240.89	22.46	41.92	167.15	65.87	261.37	25.71	192.68
miniQMC	3.16	5.39	15.64	3.72	6.85	16.28	3.89	12.32	0.50	0.90
mini-GAMESS	19.44	38.50	197.08	24.57	43.88	164.71	49.30	168.97	-	-
OpenMC	-	-	2039	-	-	-	-	1191	-	720
HACC	-	-	13.81	-	-	12.26	-	12.46	-	10.70

Table VI reflect the differences in GPU compute capabilities along with the available CPU threads and bandwidth.

VII. Conclusions and Future Work

In this work we presented benchmark results for the Intel Data Center GPU Max Series (known as Ponte Vecchio (PVC)) for a range of microbenchmarks, mini-apps and real applications.

We measured the obtainable fundamental performance limits of PVC-based nodes through microbenchmarking using high-level languages in the manner applications would be developed. This can be used as reference for future work.

By comparing the two different node-level architectures in PVC systems in Dawn and Aurora, we highlight through the microbenchmarks the potential performance impacts of multi-GPU systems, typically as a result of the TDP considerations available to the node at large. We saw that the microbenchmark performance was as expected: for single-GPU results, the compute-bound microbenchmarks on Aurora performed about 0.875x (the ratio of compute units) as on Dawn and the memory-bound ones performed the same on both systems.

We were able to run and characterize applications written in multiple language (C++, Fortran) and programming models (SYCL, OpenMP) on multiple PVC-based systems, demonstrating the robustness of the PVC software stack. The two real-world applications tested in this study showed performance on PVC which was competitive with H100 and MI250.

Additionally, in Section V we saw that generally the mini-apps performed as expected based on the microbenchmarks, especially for comparing Aurora to Dawn. However, in some cases, like in miniQMC, a different bottleneck arose during running (GPU congestion) which was not captured by the microbenchmarks. This shows a limitation of using this set of microbenchmarks as not all possible bottlenecks are captured.

As this paper was mostly focused on characterizing Dawn and Aurora nodes, in future work we plan to further compare mini-apps and applications on other supercomputing systems such as Frontier against Dawn and Aurora results. Future work should also include study of machine learning and sparse data applications.

This work provides a starting point for more in-depth benchmarking of Intel GPUs at a micro-architectural level in the future, and offers a useful resource for application-level studies with which to discuss performance.

Acknowledgment

This work was performed using resources provided by Cambridge Dawn AI Service as part of the UK AI Research Resource operated by the University of Cambridge Research Computing Service (www.hpc.cam.ac.uk), funded UKRI, DSIT, Dell EMC and Intel. This research also used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357. We also gratefully acknowledge the computing resources provided and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory. The University of Bristol is a oneAPI Center of Excellence which supported this work.

References

[1] T. Lists, "Top500 list," 2023. [Online]. Available: https://www.top500.org/

[2] ORNL, "Titan," https://web.archive.org/web/20130226194137/http://www.olcf.ornl.gov/wp-content/themes/olcf/titan/Titan_Debuts.pdf.

[3] Z. Jia, M. Maggioni, B. Staiger, and D. P. Scarpazza, "Dissecting the nvidia volta gpu architecture via microbenchmarking," *arXiv preprint arXiv:1804.06826*, 2018.

[4] H. Wong, M.-M. Papadopoulou, M. Sadooghi-Alvandi, and A. Moshovos, "Demystifying gpu microarchitecture through microbenchmarking," in *2010 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS)*. IEEE, 2010, pp. 235–246.

[5] W. Luo, R. Fan, Z. Li, D. Du, Q. Wang, and X. Chu, "Benchmarking and dissecting the nvidia hopper gpu architecture," *arXiv preprint arXiv:2402.13499*, 2024.

[6] T. linpack, "Top500 linpack," 2023. [Online]. Available: https://top500.org/project/linpack/

[7] J. Dongarra, M. A. Heroux, and P. Luszczek, "Hpcg benchmark: a new metric for ranking high performance computing systems," *Knoxville, Tennessee*, vol. 42, 2015.

[8] C. M. Siefert, C. Pearson, S. L. Olivier, A. Prokopenko, J. Hu, and T. J. Fuller, "Latency and bandwidth microbenchmarks of us department of energy systems in the june 2023 top 500 list," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1298–1305. [Online]. Available: https://doi.org/10.1145/3624062.3624203

[9] F. Salvadore, G. Rossi, S. Sathyanarayana, and M. Bernardini, "Openmp offload toward the exascale using intel® gpu max 1550: evaluation of streams compressible solver," *The Journal of Supercomputing*, pp. 1–34, 2024.

[10] M. Zubair, A. Walden, G. Nastac, E. Nielsen, C. Bauinger, and X. Zhu, "Optimization of ported cfd kernels on intel data center gpu max 1550 using oneapi esimd," in *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, 2023, pp. 1705–1712.

[11] K. Yuan, C. Bauinger, X. Zhang, P. Baehr, M. Kirchhart, D. Dabert, A. Tousnakhoff, P. Boudier, and M. Paulitsch, "Fully-fused multi-layer perceptrons on intel data center gpus," *arXiv preprint arXiv:2403.17607*, 2024.

[12] P. Nguyen, P. Nayak, and H. Anzt, "Porting batched iterative solvers onto intel gpus with sycl," in *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, 2023, pp. 1048–1058.

[13] S. Atchley, C. Zimmer, J. Lange, D. Bernholdt, V. Melesse Vergara, T. Beck, M. Brim, R. Budiardja, S. Chandrasekaran, M. Eisenbach *et al.*, "Frontier: exploring exascale," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2023, pp. 1–16.

[14] "Frontier," https://www.olcf.ornl.gov/frontier/. [Online]. Available: https://www.olcf.ornl.gov/frontier/

[15] "Intel(R) Data Center GPU Max 1500," https://www.intel.com/content/www/us/en/products/sku/232873/intel-data-center-gpu-max-1550/specifications.html. [Online]. Available: https://www.intel.com/content/www/us/en/products/sku/232873/intel-data-center-gpu-max-1550/specifications.html

[16] W. Gomes, A. Koker, P. Stover, D. Ingerly, S. Siers, S. Venkataraman, C. Pelto, T. Shah, A. Rao, F. O'Mahony, E. Karl, L. Cheney, I. Rajwani, H. Jain, R. Cortez, A. Chandrasekhar, B. Kanthi, and R. Koduri, "Ponte vecchio: A multi-tile 3d stacked processor for exascale computing," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 42–44.

[17] "Intel (r) xe gpu architecture," https://www.intel.com/content/www/us/en/docs/oneapi/optimization-guide-gpu/2024-2/intel-xe-gpu-architecture.html. [Online]. Available: https://www.intel.com/content/www/us/en/docs/oneapi/optimization-guide-gpu/2024-2/intel-xe-gpu-architecture.html

[18] "The compute architecture of intel processor graphics gen9." [Online]. Available: https://cdrdv2-public.intel.com/774710/the-compute-architecture-of-intel-processor-graphics-gen9-v1d0-166010.pdf

[19] J. R. Reinders, "Options for using a GPU Tile Hierarchy," https://www.intel.com/content/www/us/en/developer/articles/technical/flattening-gpu-tile-hierarchy.html, 2023.

[20] "Joint Laboratory for System Evaluation," https://www.jlse.anl.gov/. [Online]. Available: https://www.jlse.anl.gov/

[21] "Dawn - Intel GPU (PVC) Nodes," https://docs.hpc.cam.ac.uk/hpc/user-guide/pvc.html. [Online]. Available: https://docs.hpc.cam.ac.uk/hpc/user-guide/pvc.html

[22] "Aurora," https://www.alcf.anl.gov/aurora. [Online]. Available: https://www.alcf.anl.gov/aurora

[23] "Intel® xeon® gold 5320 processor," https://www.intel.com/content/www/us/en/products/sku/215285/intel-xeon-gold-5320-processor-39m-cache-2-20-ghz/specifications.html. [Online]. Available: https://www.intel.com/content/www/us/en/products/sku/215285/intel-xeon-gold-5320-processor-39m-cache-2-20-ghz/specifications.html

[24] "Benchmarking Scripts for Reproducibility," https://github.com/UoB-HPC/abc-pvc-deepdive. [Online]. Available: https://github.com/UoB-HPC/abc-pvc-deepdive

[25] "Nvidia h100 tensor core gpu," https://resources.nvidia.com/en-us-tensor-core/nvidia-tensor-core-gpu-datasheet. [Online]. Available: https://resources.nvidia.com/en-us-tensor-core/nvidia-tensor-core-gpu-datasheet

[26] "Amd instinct™ mi250 accelerators," https://www.amd.com/en/products/accelerators/instinct/mi200/mi250.html. [Online]. Available: https://www.amd.com/en/products/accelerators/instinct/mi200/mi250.html

[27] K. Bhat, "clpeak v1.1.2," https://github.com/krrishnarraj/clpeak, 2022.

[28] oneAPI Level Zero Memory Specification. [Online]. Available: https://spec.oneapi.io/level-zero/latest/core/PROG.html#types

[29] M. team, "MPICH README," https://github.com/pmodels/mpich.

[30] M. Martineau, P. Atkinson, and S. McIntosh-Smith, "Benchmarking the NVIDIA V100 GPU and Tensor Cores," in *Euro-Par 2018: Parallel Processing Workshops*, ser. Lecture Notes in Computer Science, G. Mencagli, D. B. Heras, V. Cardellini, E. Casalicchio, E. Jeannot, F. Wolf, A. Salis, C. Schifanella, R. R. Manumachu, L. Ricci, M. Beccuti, L. Antonelli, J. D. Garcia Sanchez, and S. L. Scott, Eds. Cham: Springer International Publishing, 2019, pp. 444–455.

[31] M. Fang, J. Fang, W. Zhang, H. Zhou, J. Liao, and Y. Wang, "Benchmarking the GPU memory at the warp level," *Parallel Computing*, vol. 71, pp. 23–41, Jan. 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167819117301825

[32] "Amd instinct™ mi250x accelerators," https://www.amd.com/en/products/accelerators/instinct/mi200/mi250x.html. [Online]. Available: https://www.amd.com/en/products/accelerators/instinct/mi200/mi250x.html

[33] A. Poenaru, W.-C. Lin, and S. McIntosh-Smith, "A performance analysis of modern parallel programming models using a compute-bound application," in *High Performance Computing*, B. L. Chamberlain, A.-L. Varbanescu, H. Ltaief, and P. Luszczek, Eds. Cham: Springer International Publishing, 2021, pp. 332–350.

[34] S. McIntosh-Smith, J. Price, R. B. Sessions, and A. A. Ibarra, "High performance in silico virtual drug screening on many-core processors," *The International Journal of High Performance Computing Applications*, vol. 29, no. 2, pp. 119–134, 2015, pMID: 25972727. [Online]. Available: https://doi.org/10.1177/1094342014528252

[35] A. Mallinson, D. Beckingsale, W. Gaudin, J. Herdman, J. Levesque, and S. Jarvis, "Cloverleaf: Preparing hydrodynamics codes for exascale," in *Cray User Group*, Napa Valley, California, USA, May 2013.

[36] Cloverleaf mini-app GitHub Repository. [Online]. Available: https://github.com/UoB-HPC/CloverLeaf/tree/pvc-performance-portability

[37] J. Kim, A. D. Baczewski, T. D. Beaudet, A. Benali, M. C. Bennett, M. A. Berrill, N. S. Blunt, E. J. L. Borda, M. Casula, D. M. Ceperley, S. Chiesa, B. K. Clark, R. C. Clay, K. T. Delaney, M. Dewing, K. P. Esler, H. Hao, O. Heinonen, P. R. C. Kent, J. T. Krogel, I. Kylänpää, Y. W. Li, M. G. Lopez, Y. Luo, F. D. Malone, R. M. Martin, A. Mathuriya, J. McMinis, C. A. Melton, L. Mitas, M. A. Morales, E. Neuscamman, W. D. Parker, S. D. P. Flores, N. A. Romero, B. M. Rubenstein, J. A. R. Shea, H. Shin, L. Shulenburger, A. F. Tillack, J. P. Townsend, N. M. Tubman, B. V. D. Goetz, J. E. Vincent, D. C. Yang, Y. Yang, S. Zhang, and L. Zhao, "QMCPACK: an open source ab initio quantum monte carlo package for the electronic structure of atoms, molecules and solids," *Journal of Physics: Condensed Matter*, vol. 30, no. 19, p. 195901, Apr. 2018.

[38] P. R. C. Kent, A. Annaberdiyev, A. Benali, M. C. Bennett, E. J. Landinez Borda, P. Doak, H. Hao, K. D. Jordan, J. T. Krogel, I. Kylänpää, J. Lee, Y. Luo, F. D. Malone, C. A. Melton, L. Mitas, M. A. Morales, E. Neuscamman, F. A. Reboredo, B. Rubenstein, K. Saritas, S. Upadhyay, G. Wang, S. Zhang, and L. Zhao, "QMCPACK: Advances in the development, efficiency, and application of auxiliary field and real-space variational and diffusion quantum Monte Carlo," *The Journal of Chemical Physics*, vol. 152, no. 17, p. 174105, May 2020, publisher: American Institute of Physics.

[39] G. M. J. Barca, C. Bertoni, L. Carrington, D. Datta, N. De Silva, J. E. Deustua, D. G. Fedorov, J. R. Gour, A. O. Gunina, E. Guidez, T. Harville, S. Irle, J. Ivanic, K. Kowalski, S. S. Leang, H. Li, W. Li, J. J. Lutz, I. Magoulas, J. Mato, V. Mironov, H. Nakata, B. Q. Pham, P. Piecuch, D. Poole, S. R. Pruitt, A. P. Rendell, L. B. Roskop, K. Ruedenberg, T. Sattasathuchana, M. W. Schmidt, J. Shen, L. Slipchenko, M. Sosonkina, V. Sundriyal, A. Tiwari, J. L. Galvez Vallejo, B. Westheimer, M. Włoch, P. Xu, F. Zahariev, and M. S. Gordon, "Recent developments in the general atomic and molecular electronic structure system," *The Journal of Chemical Physics*, vol. 152, no. 15, p. 154102, 2020. [Online]. Available: https://doi.org/10.1063/5.0005188

[40] J. Kwack, C. Bertoni, B. Pham, and J. Larkin, "Performance of the RI-MP2 Fortran Kernel of GAMESS on GPUs via Directive-based Offloading with Math Libraries," *Sixth Workshop on Accelerator Programming Using Directives*, 2019.

[41] GAMESS RI-MP2 mini-app GitHub Repository. [Online]. Available: https://github.com/jkwack/GAMESS_RI-MP2_MiniApp

[42] P. K. Romano, N. E. Horelik, B. R. Herman, A. G. Nelson, and B. Forget, "OpenMC: A state-of-the-art Monte Carlo code for research and development," *Ann. Nucl. Energy*, vol. 82, pp. 90–97, 2015, https://doi.org/10.1016/j.anucene.2014.07.048.

[43] J. R. Tramm, P. K. Romano, J. Doerfert, A. L. Lund, P. C. Shriwise, A. R. Siegel, G. Ridley, and A. Pastrello, "Toward portable GPU acceleration of the OpenMC Monte Carlo particle transport code," in *PHYSOR 2022 - International Conference on Physics of Reactors*, 2022. [Online]. Available: https://www.researchgate.net/publication/360792320_Toward_Portable_GPU_Acceleration_of_the_OpenMC_Monte_Carlo_Particle_Transport_Code

[44] J. R. Tramm, P. K. Romano, P. C. Shriwise, A. L. Lund, J. Doerfert, P. Steinbrecher, A. R. Siegel, and G. Ridley, "Performance portable Monte Carlo particle transport on Intel, NVIDIA, and AMD GPUs," in *SNA + MC 2024: Joint International Conference on Supercomputing in Nuclear Applications + Monte Carlo*, Paris, France, Oct. 2024, accepted. Pre-print available at https://arxiv.org/abs/2403.12345.

[45] S. Habib, A. Pope, H. Finkel, N. Frontiere, K. Heitmann, D. Daniel, P. Fasel, V. Morozov, G. Zagaris, T. Peterka, V. Vishwanath, Z. Lukić, S. Sehrish, and W.-k. Liao, "HACC: Simulating sky surveys on state-of-the-art supercomputing architectures," *New Astronomy*, vol. 42, pp. 49–65, Jan. 2016.

[46] N. Frontiere, J. D. Emberson, M. Buehlmann, J. Adamo, S. Habib, K. Heitmann, and C.-A. Faucher-Giguère, "Simulating Hydrodynamics in Cosmology with CRK-HACC," *The Astrophysical Journal Supplement Series*, vol. 264, no. 2, p. 34, Feb. 2023.

[47] E. M. Rangel, S. J. Pennycook, A. Pope, N. Frontiere, Z. Ma, and V. Madananth, "A performance-portable sycl implementation of crk-hacc for exascale," in *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, 2023, pp. 1114–1125.

Appendix: Artifact Description/Artifact Evaluation

Artifact Description (AD)

I. Overview of Contributions and Artifacts

A. *Paper's Main Contributions*

- C_1 Quantification of obtainable fundamental performance limits of PVC-based nodes through microbenchmarking using high-level languages in the manner applications would be developed
- C_2 Evaluation of the performance of mini-apps with known characteristics, relating their performance to the results of the microbenchmarks
- C_3 Demonstration of functionality and performance of mini-apps and applications on PVC hardware
- C_4 Performance comparison of two science applications across a single node of four different systems with varying CPU and GPU architectures (an Aurora node (six PVC), a Dawn node (four PVC), a node of four NVIDIA H100s, and a node of four AMD MI250s
- C_5 Discussion of how microbenchmarking can provide insight into application and mini-application performance as well as its limitations.

B. *Computational Artifacts*

A_1 https://doi.org/10.5281/zenodo.13843869

Artifact ID	Contributions Supported	Related Paper Elements
A_1	C_1-C_5	Tables II,III,V,VI Figure 1-4

II. Artifact Identification

A. *Computational Artifact A_1*

Relation To Contributions

Computational Artifact A_1 is the source code and run scripts for the microbenchmarks as well as the instructions to reproduce the mini-apps results. This supports C_1 as it contains the high-level implementations needed to reproduce the microbenchmarking results. It supports C_2, C_3, and C_4 as it contains the compile and run instructions for the mini-apps on Intel, Nvidia, and AMD hardware. C_5 is supported as the results from running the microbenchmarks and mini-apps are analyzed in the text and the connection between the expected results of the mini-apps from the microbenchmarking and the actual results from the mini-apps is discussed.

Expected Results

We expected the microbenchmark code to build and run correctly, and running it on Dawn or Aurora should be able to confirm C_1. As discussed in the paper, the compute-bound microbenchmarks on Aurora should be 0.875x the results on Dawn and the memory-bound ones should be equal on both systems. The mini-apps and applications we expect to be able to compile and run correctly on Aurora, Dawn, JLSE-H100, and JLSE-MI250 (except for the GAMESS RI-MP2 code on JLSE-MI250 as discussed in the text), which demonstrates C_3. We generally expect the ratios of the results on the different systems to match the expected ratios based on the microbenchmarks and the known bounds of the mini-apps, which demonstrates C_2. In certain cases (like miniQMC) we encountered a different bottleneck than expected which contributes to the discussion for C_5.

Expected Reproduction Time (in Minutes)

We expect the runtime for all codes to be less than 2 hours. Running the microbenchmark script on PVC should take less than 30 min.

Artifact Setup (incl. Inputs)

Hardware: The systems used are listed below:

- Dawn: Each node is two 48-core Intel Xeon Platinum 8468 CPUs with a total of 1024GB DDR and four Intel Data Center GPU Max 1550 Series, previously codenamed Ponte Vecchio (PVC)
- Aurora: Each node is two 52-core (104-thread) Intel Xeon Gold 5320 CPUs with 64GB HBM and 512GB DDR5 each and six Intel Data Center GPU Max 1550 Series, previously codenamed Ponte Vecchio (PVC)
- JLSE-H100: Each node is two Intel Xeon Platinum 8468 CPUs (48 cores/96 threads) with 512GB DDR5 total and four NVIDIA H100 SXM5 80GB GPUs
- JLSE-MI250: Each node is two AMD EPYC 7713 64c CPUs (64 cores/128 threads) with 512 GB DDR4 and four AMD Instinct MI250 GPUs

Software: The microbenchmark code and scripts needed to build and download the mini-apps and applications are available at https://doi.org/10.5281/zenodo.13843869. The needed modules and environment for Dawn, Aurora, JLSE-H100, and JLSE-MI250 are located at https://doi.org/10.5281/zenodo.13843869 in $./environment$. The compilers that are used on each system are:

- Dawn: The default software used was also the Intel oneAPI 2024.1 public release.
- Aurora: The default software used was also the Intel oneAPI 2024.1 public release.
- JLSE-H100: The default software used was NVHPC 24.1 and CUDA 12.3.0.
- JLSE-MI250: The default software used was ROCm 6.1.0.

Datasets / Inputs: The microbenchmarks (results in Table I-III and Fig. 1) did not have separate input files. The inputs for the mini-apps (results in Table VI and Fig. 2-4) are contained within the scripts to run them in https://doi.org/10.5281/zenodo.13843869 in $./mini-apps$.

Installation and Deployment: To install the microbenchmark and mini-app codes evaluated in the paper, the source code from https://doi.org/10.5281/zenodo.13843869 should be obtained. Inside there are three main subfolders: microbenchmarks, mini-apps, and applications. In the microbenchmarks and mini-apps case, the script that runs the code also builds and installs the code so the instructions are covered in the $ArtifactExecution$ section below.

Artifact Execution

The workflow to run the inputs to generate the Tables and Figures in the paper is as follows:

For Table II, and III:

- Move to the 'microbenchmark' subdirectory. Run the script $./microbenchmark/run\_table.sh$ on Dawn and Aurora. This will generate the results for the microbenchmarking tables in the paper.

For Figure 1:

- Move to the 'microbenchmark' subdirectory. Run the script $./microbenchmark/run\_lats.sh$ on Dawn and Aurora. This will generate the results for the memory latency figure in the paper.

For Table VI:

- For miniBUDE: Move to the 'mini-apps/miniBUDE' subdirectory. The input needs to be fetched from cschpc/epmhpcgpu and copied to the $minibude/data$ directory. Run the script $./run.sh$ on Dawn, Aurora, JLSE-H100, and JLSE-MI250. This compiles and runs the miniBUDE code with the input used in the paper.
- For cloverleaf: Move to the 'mini-apps/cloverleaf' subdirectory. Run the script $./run.sh$ on Dawn, Aurora, JLSE-H100, and JLSE-MI250. This compiles and runs the cloverleaf code with the input used in the paper.
- For mini-GAMESS: Move to the 'mini-apps/gamess_rimp2_mini_app' subdirectory. Follow the steps in the $./mini - app - reproducibility.sh$ script to run the code on Dawn, Aurora, JLSE-H100, and JLSE-MI250.
- For miniQMC: Move to the 'mini-apps/miniQMC' subdirectory. Move into the subfolder for each system (for example, 'aurora_pvc' for Aurora) and run $./mini - app - reproducibility.sh$

For Figure 2:

Figure 2 plots the Figures-of-Merit on Aurora relative to Dawn. This can be computed by taking the Aurora results from Table VI and dividing them by the Dawn results in Table VI. The black bars designate the expected relative performance (i.e. Aurora peak / Dawn peak) of the mini-app based on the bounds of each mini-app, which are listed in Table V, and the microbenchmark results in Table II. For example, miniBUDE is listed in Table V as bound by the single precision (FP32) flop-rate. Thus the expected relative performance is the ratio of the peak single precision performance on Aurora to that on Dawn, 0.88X (23 Tflops/s$/$26 Tflop/s), where the peak values come from Table II.

For Figure 3:

Figure 3 plots the Figures-of-Merit on Aurora and Dawn relative to the JLSE-H100 Figures-of-Merit. This can be computed by taking the Aurora and Dawn results from Table VI and dividing them by the JLSE-H100 results in Table VI. The black bars designate the expected relative performance (i.e. Aurora peak or Dawn peak / JLSE-H100 peak) of the mini-app based on the bounds of each mini-app, which are listed in Table V, the microbenchmark results in Table II, and the bounds of JLSE-H100 listed in Table IV. For example for one GPU, Cloverleaf is listed in Table V as bound by the memory bandwidth. Thus the expected relative performance is the ratio of the peak memory bandwidth on Aurora or Dawn to that on JLSE-H100, 0.59X (2 TB/s$/3.35$ TB/s) for Aurora and Dawn (it's the same since they have the same memory bandwidth per GPU). The peak values for Aurora and Dawn come from Tables I and for JLSE-H100 come from Table II.

For Figure 4:

Figure 4 plots the Figures-of-Merit on Aurora and Dawn relative to the JLSE-MI250 Figures-of-Merit. This can be computed by taking the Aurora and Dawn results from Table VI and dividing them by the JLSE-MI250 results in Table VI. The black bars designate the expected relative performance (i.e. Aurora peak or Dawn peak / JLSE-MI250 peak) of the mini-app based on the bounds of each mini-app, which are listed in Table V, the microbenchmark results in Table II, and the bounds of JLSE-MI250 listed in Table IV. For example for one PVC Stack / one AMD GCD, miniBUDE is listed in Table V as bound by the single precision (FP32) flop-rate. Thus the expected relative performance is the ratio of the peak FP32 flop-rate on Aurora or Dawn to that on JLSE-MI250. For Aurora it's 1.0X (23 Tflops/s$/(45.3/2)$ Tflop/s) and for Dawn it's Aurora 1.1X (26 Tflops/s$/(45.3/2)$ Tflop/s). The peak values for Aurora and Dawn come from Tables I and for JLSE-MI250 come from Table II (where the JLSE-MI250 number is divided by two since it's run on a single GCD instead of two.

Artifact Analysis (incl. Outputs)

The output of running the codes above for the microbenchmarks and mini-apps are available in A_1, and the values in the table can be read directly to generate the tables or figures in the paper.