2025 IEEE 25th International Symposium on Cluster, Cloud and Internet Computing (CCGrid) | 979-8-3315-0934-7/25/$31.00 ©2025 IEEE | DOI: 10.1109/CCGRID64434.2025.00025

Scaling Laws for the Workload Throughput of Emerging Heterogeneous Clusters

Akhil Alasandagutti*, Joshua Suetterlein†, Jesun Firoz†, Stephen Young†, Joseph Manzano†, Jason R. Stewart*, Patrick G. Bridges*, Trilce Estrada*, Kevin Barker†
*University of New Mexico, †Pacific Northwest National Laboratory
*{aalasand1, jastewart, patrickb, trilce}@unm.edu
†{{first name}.{last name}}@pnnl.gov

***Abstract*—Next-generation HPC clusters are evolving into highly heterogeneous systems that integrate traditional computing resources with emerging accelerator technologies such as quantum processors, neuromorphic units, dataflow architectures, and specialized AI accelerators within a unified infrastructure. These advanced systems enable workloads to dynamically utilize different accelerators during various computation phases, creating complex execution patterns. The performance of the workloads can therefore be impacted by many factors, including how the accelerators are shared, their utilization, and their placement within the system. Moreover, effects such as the system and network state due to the overall system load can significantly impact the job completion rate. Understanding, identifying, and quantifying the impact of the most critical factors (e.g., the number of allocated accelerators) will help decide the investment decisions for accelerator acquisition and deployment that can improve the overall system throughput. This paper extensively studies these complex interactions among advanced accelerators within an HPC cluster and various workloads. We introduce a novel analytical model which predicts the speedup of a workload given an accelerator/system configuration. This model can be used to quantify the effect of augmenting additional accelerators on job performance running on an HPC cluster. We validate the model using both simulated and real environments.**

I. Introduction

In the post-Moore law era, the rapid evolution of high-performance computing (HPC) architectures is increasingly driven by the emergence of diverse and complex workloads that comprise a mixture of computationally intensive and communication-bound applications, including, but not limited to advanced simulation, machine learning (ML) and graph analytics [1]. As transistor scaling slows, today's HPC systems incorporate heterogeneous components, such as graphics processing units (GPU) and field-programmable gate arrays (FPGAs), to continue to deliver performance improvements [2]. The next generation of emerging HPC systems will likely integrate a broader array of specialized accelerators such as quantum processors (e.g. IBM's Quantum Heron [3]), neuromorphic accelerators (e.g. Intel's Loihi [4]), Dataflow architectures (e.g. SambaNova [5]), and specialized ML accelerators (e.g. Cerebras [6]), co-located within the same HPC cluster [7]. [1] For instance, recent research has demonstrated the potential of combining quantum and classical computing with co-located quantum processors generating noisy samples to be processed by the Fugaku supercomputer to estimate ground-state energies in chemistry [8].

To maximize their potential, the next-generation of HPC systems must effectively integrate diverse accelerators with classical computing while addressing job scheduling, system utilization, resource allocation, and system provisioning. At the heart of these challenges is machine balance and utilization. Successful clusters will balance their hardware configuration based on job scalability and throughput to reduce the Total Cost of Ownership (TCO). While low-level simulation targeting an individual type of accelerator provides a detailed analysis of a single application, it is ill-suited for a broader systems-level analysis that spans many cluster configurations and applications (some of which may not yet exist) needed to address these questions. Instead, a high-level model exploring future applications' broad characteristics and accelerator usage will enable a quicker and more intuitive means to explore cluster configurations. To this end, we explore workload and accelerator abstractions to better understand the impacts of job scheduling and cluster configuration on systems running applications that tightly couple classical and novel computations.

Subsequently, this work proposes a new analytical model for understanding the performance of workloads on an HPC cluster incorporating multiple co-located accelerators. Our model projects the improvement in throughput as accelerators are added to the system, taking into account the following factors: whether the accelerators are shared or used exclusively, the amount of time a job spends on classical computing units and accelerators, and the job startup time before running on an accelerator.

This paper makes the following **contributions**:

- *An in-depth study on how accelerator access modes (multi-tenant vs exclusive), network topology, accelerator placement, mixture of communication- & computation-intensive workloads, and scheduling strategies impact workload performance and system throughput.*
- *An analytical model for projecting how system throughput will scale as the number of accelerators increases in the next-generation HPC systems with multiple accelerators.* The model considers several key factors, such as accelerator access modes, job composition (distribution of workload on

[1] In the subsequent discussion, we will refer to such specialized hardware as *accelerators*.

2993-2114/25/$31.00 ©2025 IEEE
DOI 10.1109/CCGRID64434.2025.00025

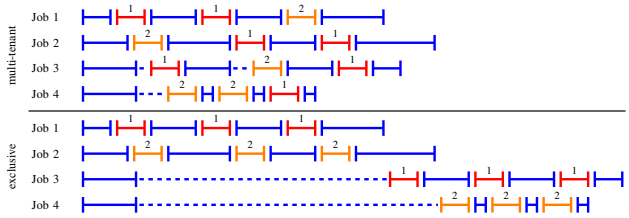


Fig. 1. Illustration of multi-tenant versus exclusive scheduling for a workload consisting of four jobs sharing two accelerators. Each individual job alternates between classical tasks (blue) and accelerated tasks (red/orange) with inactive time denoted by a dashed blue line. The color of the interval and the number above the interval both indicate which of the two accelerators are being used to resolve the given task in the job.

traditional processors and accelerator), and queue time for resource allocation.

- *Validation of our analytical model with simulation (extending the Structural Simulation Toolkit Macroscale Element Library (`SST/macro`) [9]) and on a real cluster.*

II. Background

A. Scheduling/Access Modes

In this study, we consider two modes of scheduling through which jobs get assigned to the accelerators. Figure 1 illustrates the two scheduling modes described below.

Multi-Tenant Scheduling Mode. In multi-tenant scheduling, each job reserves an accelerator only for a particular phase of the workload execution. Once the phase completes execution, the accelerator is made available to other jobs. In this way, multi-tenancy may improve the overall system throughput by providing finer access granularity to the accelerators and reducing the wait time in the job queue.

Exclusive Scheduling Mode. Exclusive scheduling, where jobs reserve an accelerator for the entire duration of their workload execution, has its drawbacks. While it ensures dedicated resources and more predictable execution times, it can lead to under-utilization of accelerators in cases where the workloads consist of a significant amount of classical computation that the accelerators depend on. This model is widely used in current HPC systems, with popular schedulers such as Slurm [10], LSF [11], and PBS [12]. In our work, we evaluated a variation of exclusive scheduling where a job is permitted to start classical compute prior to an accelerator being made available (i.e., job 3 in Figure 1). Such an alteration is motivated by the tight integration of heterogeneous applications where all jobs are expected to use an accelerator. In general, this evaluation of exclusive scheduling is more optimistic.

B. `SST/macro` Simulation Framework

`SST/macro` [9] is an open-source, discrete event simulator that models communication and computation in high-performance computing (HPC) systems. The simulator intercepts distributed communication interface calls, for example, Message Passing Interface (MPI) calls, and simulates message flow through the network layers such as network interface cards (NICs), switches, and links. `SST/macro` is highly customizable through its configuration files. These files enable users to adjust hardware parameters, including topology configuration, buffer sizes, and minimum message transmission units. In our experiments, we chose the detailed packet-based model with Quality-of-Service guarantee within `SST/macro`, namely the Simulator Network for Adaptive Priority Packet Routing (`SNAPPR`) model [13] to ensure the most accurate execution of the network flow and interactions. Several research studies [14]–[16] have utilized SST/macro to explore next-generation HPC network topologies and evaluate hardware designs for exascale computing systems.

III. Effects of Different Factors on Jobs Running on a System with Multiple Accelerators

This section presents a comprehensive evaluation of how various system factors affect job performance in emerging HPC clusters with multiple accelerator types. With simulation, we analyze the impact of several key parameters: *accelerator task time (ATT)* which represents the duration each job phase utilizes accelerators, accelerator access modes (multi-tenant vs. exclusive), network topology configurations (Dragonfly and Fat Tree), application types (physics, machine learning, and graph analytics), and workload composition within the system. Our analysis provides insights into how these factors collectively influence system performance and resource utilization as the number of accelerators scales.

A. Experimental Methodology

Simulation Infrastructure. In our experimental setup, we extend the `SST/macro` simulation framework to simulate an HPC system with multiple accelerators at the `MPI` application level. We implement a driver application that reserves a single `MPI` rank for each accelerator and wraps all of the instances of each target app (i.e., physics, ML, or graph application). A `MPI` communicator is created within the driver, including all ranks for each instance of a specific target app and one accelerator. Each node/accelerator is assigned a single rank in the cluster configuration.

Next, all target apps are modified with an asynchronous gather from each rank to the accelerator in its subcommunicator. To simulate a tightly coupled application, we place this communication in the main loop of each target app. During execution, each accelerator polls for incoming data. When all the data from a target app is received, it simulates servicing the application by performing computations for a set duration, after which it broadcasts the results back to all the ranks of the served app. The target app proceeds to the next iteration upon receiving the computed value.

The composition of target apps is randomly selected, and their execution schedule is passed on via a configuration file. Further, the mapping of target apps to accelerators is also determined statically a priori. We explore two approaches, the nearest accelerator (from the network perspective) and load balanced, but only report results for load balanced as it consistently outperforms the nearest accelerator approach.

Dragonfly	a		g		h		IG Conn.		Conc	Nodes
	4		64		16		Circulant		2	512
Fat Tree	**Ports**	**Pods**	**Sw.**	**E.Sw.**	**HxP**	**LB**	**AB**	**CB**	**Conc.**	**Nodes**
	14	14	245	98	49	1x	2x	2x	7	686

TABLE I. SST/MACRO CONFIGURATION FOR DRAGONFLY AND FAT TREE WHERE a = # OF ROUTERS IN A GROUP, g = # OF GROUPS, h = # OF GLOBAL LINKS, Sw. = # OF SWITCHES, E.Sw. = # OF EDGE SWITCHES, HxP = HOSTS PER POD, LB = LEAF BANDWIDTH MULTIPLIER, AB = AGGREGATE BANDWIDTH MULTIPLIER, CB IS CORE BANDWIDTH MULTIPLIER, AND CONC. = CONCENTRATION

App Type	Apps	Processes
Graph Analytics	MiniVite (5 inputs)	2,4,6,8,16, 32,64,128
Machine Learning	ResNet-152, Cosmoflow, DLRM	2,4,6,8,16, 32,64,128
Physics	Sweep3D, Halo3D, FFT	2,4,6,8,16, 32,64,128

TABLE II. APPLICATION AND PROCESS COUNT CHOICES FOR EACH APPLICATION TYPE.

Application	Inputs
MiniVite	**Graphs:** High Energy Physics, Phenomenology, Wikipedia Requests for Adminship, Amazon Product Co-Purchasing, Network, Gnutella P2P File Sharing Network, Epinions Social Network
ResNet-152	100,000 (Allreduce size)
Cosmoflow	1000 (size multiplier)
DLRM	1000 (size multiplier)
Sweep3D	*(pex, pey)*: (8,2), (8,4),(8,8),(8,16)
Halo3D and FFT	*(pex, pey, pez)*: (4,4,1), (4,4,2), (4,4,4), (4,4,8)

TABLE III. INPUT CHOICES FOR EACH APPLICATION.

Network topology configurations. In the network topology specification, we reserve a switch (and all endpoints associated with it) in a group within a Dragonfly topology for an accelerator and a switch in a pod in the Fat Tree topology. Table I shows the specific simulation parameters used for both topologies in our experimental setup.

Applications. We consider three types of representative applications: physics, graph analytics, and machine learning, with diverse communication-computation patterns. These patterns, found in many HPC jobs, coupled with interleaved phases of accelerator usage for certain ATTs, constitutes our simulation premise. For physics applications, we consider Sweep3D, FFT, and Halo3D proxy applications from SST/macro's skeletonized examples [17]. For machine learning applications, we consider ResNet-152, CosmoFlow, and the Deep Learning Recommendation Model (DLRM) proxy applications from [18]. For graph analytics, we consider a community detection algorithm from ExaGraph, MiniVite [19]. Graph applications are generally communication-bound. In contrast, machine learning applications strike a balance between moderate computational tasks and significant bursts of data communication. On the other hand, physics applications are compute-bound and have pre-defined communication phases.

Job Composition. We now describe our methodology for combining various applications to generate comprehensive system workloads. Initially, we determine the key parameters for our chosen network topology: the quantity of accelerators, the scheduling mode, and an ATT for a chosen topology. Next, we generate a workload characterized by a fixed distribution of application types. This distribution determines the fraction of the total workload processes that each application type can occupy. For a given distribution, we iteratively randomly pick eligible applications with eligible random process counts and a randomly selected input and add it to the workload mix until the selected system configuration is fully occupied. As the applications are added to the workload mix, they are assigned to accelerators in a round-robin fashion to ensure a load-balanced setup. Tables II and III display the application classes, their types, process counts, and inputs to each application.

We generate 50 random replications of each workload to ensure that they capture statistically significant variability and report the mean of the observed performance for the workload. According to the Central Limit Theorem, running samples of this size result in an approximately normal distribution.

B. Performance Analysis Across System Configurations

We evaluate the impact on *jobs* with different accelerator access modes as the number of accelerators increases.

Experimental Design and Performance Evaluation Methodology. We evaluated the effect on runtime by increasing the number of accelerators for two topologies: Dragonfly and Fat Tree. The results are presented in Figures 2 and 3. Our experimental design encompasses two scenarios: homogeneous workloads, where we run graph analytics, ML, or physics applications exclusively, and heterogeneous workloads, which consist of a mix of these application types. We have experimented with two distribution patterns for the heterogeneous workloads: a balanced composition with equal representation from each application type and an asymmetric distribution of workloads. We evaluate these configurations over a range of ATTs: 10 to 10 million microseconds (μs). We divide the results into shorter ATTs (10μs-100,000μs) and longer ATTs (1 million μs to 10 million μs). Our evaluation was conducted for both exclusive and multi-tenant modes. To quantify performance improvement, the relative speedup is computed by comparing it against the same workload/system configuration runtime but with a single accelerator scheduled in exclusive mode.

Impact on homogeneous and heterogeneous workloads with multi-tenant vs exclusive scheduling on Dragonfly topology. From Figure 2, we make the following observations for jobs running on an HPC system with the Dragonfly topology. For shorter ATTs, utilizing accelerators in multi-tenant mode generally results in improved job performance as the number of accelerators increases. However, the performance gains become less significant beyond the seven accelerators mark. When considering particularly shorter ATTs, the speedup curves tend to flatten after five to seven accelerators are added, depending on the application type. These facts suggest that five to seven accelerators with multi-tenancy

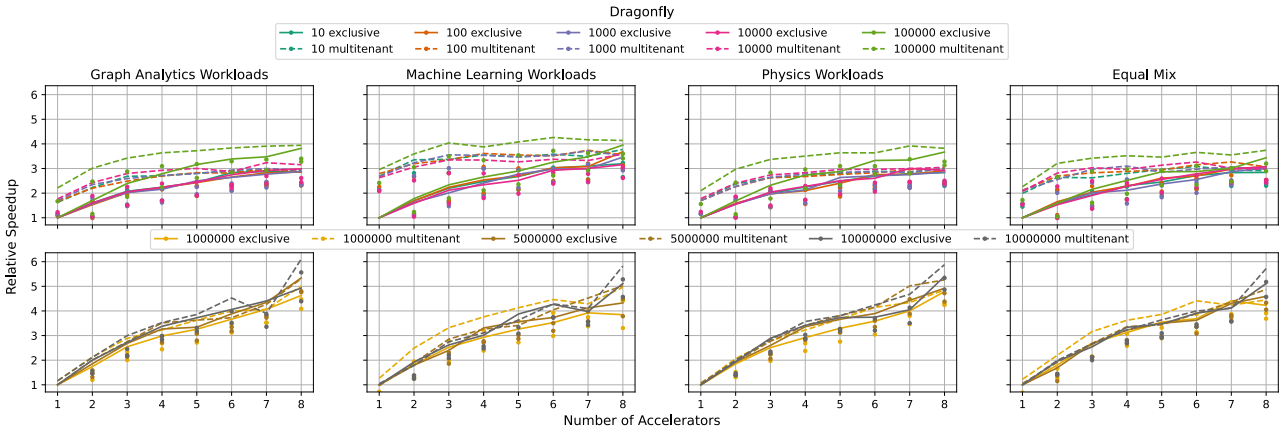


Fig. 2. Relative Speedups of workloads with increasing number of accelerators and various ATTs, running on a system with the Dragonfly topology.

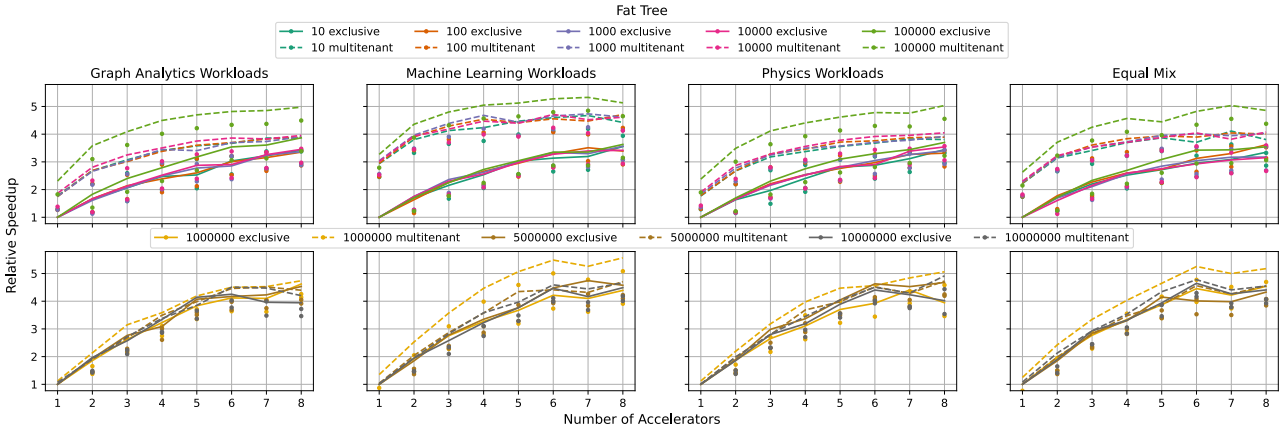


Fig. 3. Relative Speedups of workloads with increasing number of accelerators and various ATTs, running on a system with the Fat Tree topology.

may offer a good balance between performance gain and provisioning cost for a system of this particular size. The performance gap between the multi-tenant and exclusive mode was less pronounced for longer ATTs, with fewer accelerators (up to five). However, this gap widens with additional accelerators. This continued improvement of the job performance trend suggests that jobs with longer ATT could benefit from additional accelerators if the budget allows such expansion.

In terms of the impact on workloads, adding more accelerators yields a more significant positive impact (speedup) for graph analytics and physics applications than machine learning workloads. Both homogeneous and equally distributed heterogeneous workloads demonstrate similar performance improvement trends as the number of accelerators increases. Figure 4 shows the same performance, but on skewed workloads, where one of the three application types has a majority ($\geq 50\%$) of the total number of processes on the system. We observe a similar trend as that of homogeneous workloads for lower ATTs, however, the larger ATTs for these workloads do not attain relative speedups as the homogeneous workloads, peaking at roughly 5x, compared to 7x of the homogeneous.

Impact on homogeneous and heterogeneous workloads with multi-tenant vs exclusive scheduling on Fat tree topology. From Figure 3, for Fat Tree topology, we make the following observations. For shorter ATTs, compared to the Dragonfly topology, there is an apparent dichotomy of performance improvement between multi-tenancy and exclusive mode in the Fat Tree case, with shorter ATT, performance plateaus beyond six accelerators with multi-tenancy for graph analytics and physics workloads (except for 100k ATT). For machine learning applications, the plateau occurs around four accelerators. Fat Tree often shows continued improvement beyond six accelerators for larger ATT values, similar to Dragonfly. Overall, the performance improvement is better on the Dragonfly topology (Figure 2) compared to the Fat tree topology (Figure 3). However, the Fat Tree topology is larger than Dragonfly (686 vs 512 nodes) and can accommodate more jobs per accelerator than the latter. The increased speedup for large ATTs on Dragonfly is likely due to relatively lower amounts of work per accelerator.

Overall Utilization of Accelerators. From the system's perspective, we also assess the overall utilization of the accelera-

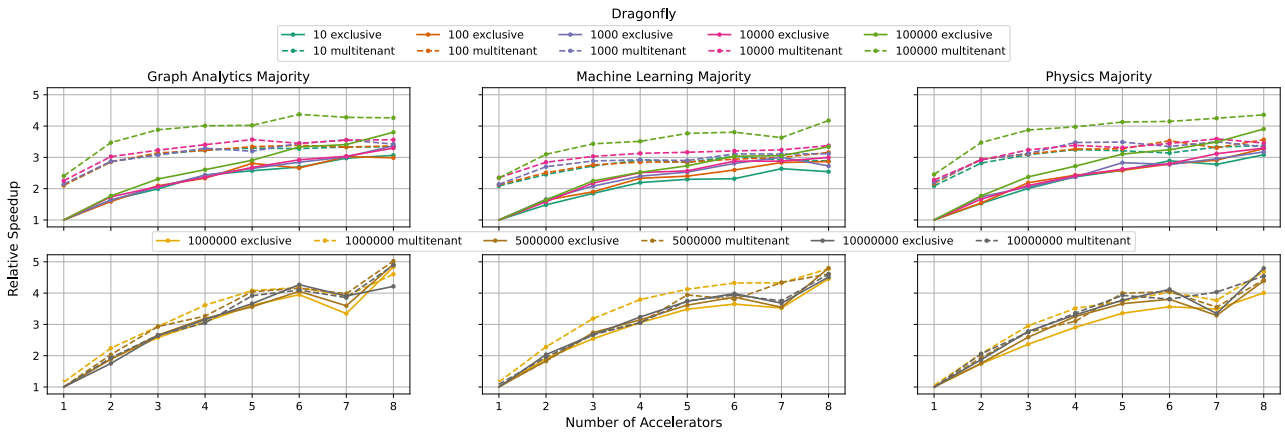


Fig. 4. Relative Speedups of skewed workloads with increasing number of accelerators and various ATTs, running on a system with the Dragonfly topology.

tors. The *cumulative accelerator utilization* is computed using the following formula

$$100 * \frac{\text{\# Jobs} \times \text{\# Iterations} \times \text{ATT}}{\text{Total Runtime}}$$

We report the results for the Dragonfly topology in Figure 5. We noticed a similar trend with the Fat Tree topology.

We observe that shorter ATTs generally result in lower utilization (1-25%). In contrast, longer ATTs achieve higher cumulative accelerator utilization (50-600%) overall. For longer ATTs, increasing the number of accelerators in the system leads to higher overall cumulative accelerator utilization.

Takeaway 1 : For jobs where classical compute dominates (low ATT), clusters can achieve up to 2x improvement in throughput with multitenant scheduling while using a fraction of the accelerators. When accelerator compute dominates, the schedule does not have a significant impact.

IV. Analytical Performance Model for Accelerator-Augmented HPC Systems

We now present analytical models to estimate the speedup or slowdown of workloads in future heterogeneous HPC systems as the number of accelerators varies. These models offer insights into the trade-offs between total cost of ownership (TCO) for multi-tenant and exclusive scheduling modes, taking into account the composition of the workload. We develop distinct models for two scheduling scenarios, detailed in Section II-A, which can be used to predict performance changes as the accelerator counts on network changes. By modeling these dynamics, we provide intuition on how workload characteristics impact the TCO and performance under different workloads and system configurations.

Now, we develop a queuing-theory-style model of the overall drain timefor a collection of jobs under both the multitenant and exclusive frameworks. As the trade-offs for classical parallelism are orthogonal to the accelerator usage issues we are investigating, we will assume that each job has been appropriately parallelized, and there are sufficient classical resources to handle all classical work simultaneously. Thus, each job in the total workload can be parameterized by a triplet of values;

$s \equiv$ classical startup time before first accelerator usage,

$a \equiv$ total time spent on accelerators, and

$c \equiv$ total (non-startup) time for classical computation.

Thus the total runtime of a job on an empty system will be $s + a + c$.

In order to simplify notation, and focus on the essential issues of accelerator usage, we will only consider the case where there is a single global queue from which individual tasks are dispatched to the accelerators[2]. Other load-balancing schemes will yield a qualitatively similar form for the overall clearing time (perhaps with different parameters). For example, if the load-balancing scheme divides the jobs among different collections of accelerators, the analysis which follows can be applied to each collection individually and then the maximum taken among all collections, yielding a similar form of the overall drain time.

Job drain time with exclusive scheduling. We first consider the drain time, T_e, of a collection of jobs $\{(s_j, a_j, c_j)\}$ running on a system with k accelerators using `slurm`-style exclusive scheduling. In this case, the time for the first k jobs i.e., those jobs that are "first-in-line" for the accelerators, will be given by $s_j + a_j + c_j$ while subsequent jobs can perform the initial start-up work of s_j while waiting for an available accelerator, yielding a total effective runtime of $a_j + c_j$.

We note that (other than pre-processing that subsequent jobs can do) identifying the minimum total run time for the jobs is an instance of minimum makespan problem for scheduling identical machines (in this case the accelerators).[3] Recall that,

[2]In our simulated and physical validation experiments (see Section V), we use a greedy load-balancing scheme based on the size of the individual jobs

[3]Again, we note that considering multiple accelerator types does not change the qualitative nature of our results, but instead yields a similar functional form for each accelerator type individually.

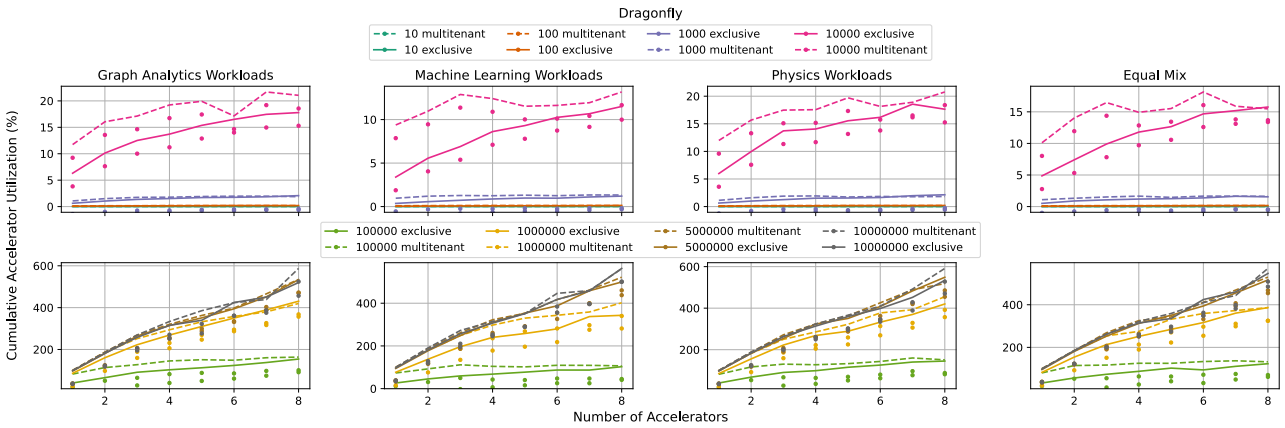


Fig. 5. Cumulative accelerator utilization on Dragonfly. Percentages larger than 100% are indicative of usage saturating multiple accelerators.

given a collection of jobs $J_1, \ldots, J_n$, the *minimum makespan problem* for identical machines (also known as the *job shop scheduling problem*) seeks to find an assignment of jobs to one of k machines, so as to minimize the maximum total runtime of any machine, i.e., the clearing time of the workload. In this context, an *arbitration* scheme decides which jobs get assigned to which machines and in what order. Finding the true optimum solution for the minimum makespan problem is $\mathcal{NP}$-complete [20], [21]. One of the well-known arbitration approaches, *list scheduling*, takes an ordered list of jobs as input, iteratively assigns each job to the machine with the currently lowest load, and continues until all jobs are assigned. In [22], Graham showed that any arbitrary list scheduling solution can deviate from the true optimum by a factor of at most $2 - \frac{1}{m}$, where m denotes the total number of machines. Thus, if we ignore the details of the arbitration mechanism and fix a set K of k jobs that receive first priority on the accelerators, the job drain time with exclusive scheduling can be bounded as follows.

$$\left(\frac{1}{k}\sum_{j\in K} s_j + \frac{1}{k}\sum_j (a_j + c_j)\right) \leq T_e, \text{ and}$$

$$T_e \leq \left(2 - \frac{1}{k}\right)\left(\frac{1}{k}\sum_{j\in K} s_j + \frac{1}{k}\sum_j (a_j + c_j)\right).$$

In general, the sum $\sum_{j\in K} s_j$ will scale with k (independent of the arbitration scheme), and thus $\frac{1}{k}\sum_{j\in K} s_j$ is effectively constant and independent of the number of accelerators. Thus, T_e can be modeled as $\alpha_e + \frac{1}{k}\beta_e$ where $\alpha_e = \frac{1}{k}\sum_{j\in K} s_j$ and $\beta_e = \frac{1}{k}\sum_j (a_j + c_j)$. α and β are parameters dependent on the workload, but crucially, not on the number of accelerators.

Job drain time with multi-tenant scheduling. We now turn to the drain time in the multi-tenant scenario, denoted by T_m. The multi-tenant case differs from the exclusive setting as the classical portion of the job can run in parallel, with minimal delays for the queuing of the accelerated portion of the workload, for example, see Figure 1. Thus the true runtime is heavily dependent on the sequence of the accelerator job arrivals at each of the distinct accelerators. If we denote the total time that job j spends waiting in the accelerator queue by q_j, then $T_m = \max\{s_j + a_j + c_j + q_j\}$. However, we note that this can be upper bounded by separating out the "classical computation" and the accelerator costs, that is

$$T_m \leq \max\{s_j + c_j\} + \max\{a_j + q_j\}.$$

We note that queue times, q_j, are maximized by the (unrealistic) scenario where all accelerator requests are placed in the common queue immediately. Again applying the result of Graham, this implies that

$$\frac{1}{k}\sum_j a_j \leq \max\{a_j + q_j\} \leq \left(2 - \frac{1}{k}\right)\frac{1}{k}\sum_j a_j.$$

As a consequence, we can also model T_m as $\alpha_m + \frac{1}{k}\beta_m$. Here, α_m represents the maximum classical computation time $\max\{s_j + c_j\}$ and $\frac{1}{k}\beta_m$ represents the accelerator and queue time component. α_m and β_m depend on the precise workload (and load-balancing scheme) and are independent of the number of accelerators.

Overall speedup with k accelerators. As a consequence, the speedup for using k accelerators instead of a single accelerator can be given by

$$\frac{\alpha + \beta}{\alpha + \frac{1}{k}\beta} = \frac{1}{1 - \gamma + \frac{1}{k}\gamma} \tag{1}$$

where $\gamma = \beta$ (and $\alpha = 1 - \gamma$) is a some workload and scheduler dependent parameters that quantifies the proportion of the overall workload which benefits from accelerator parallelism.

We note that this speed-up formalism can used to identify the point of diminishing returns in terms adding accelerators. Specifically, adding a new accelerator will increase speedup by less than a factor of $(1 + \epsilon)$ if the number of accelerators, k, satisfies

$$\frac{1 - \gamma + \frac{1}{k}\gamma}{1 - \gamma + \frac{1}{k+1}\gamma} < (1 + \epsilon),$$

We can solve this equation algebraically (by treating the inequality as an equation at the balance point) to find the exact number of accelerators k that represents this threshold, which gives us

$$k = \frac{-1 + \sqrt{1 + 4^{\frac{\gamma(1-\gamma)}{\epsilon}}}}{2(1-\gamma)}. \quad (2)$$

Takeaway 2 : Using eq. (2), we offer a method to provision a cluster with the right number of accelerators for achieving a target performance level. Additionally, optimization problems can be formulated to identify the point of diminishing returns, as illustrated in fig. 5.

Relationship to existing speedup laws. We note that although the speedup equation with k accelerators, eq. (1), has the same functional form as Amdahl's law [23], it differs fundamentally in both purpose and interpretation. Traditional speedup laws focus on the parallelism of a single program. In contrast, our model (i) is used to reason about the throughput of jobs (i.e., the speedup of multiple applications scheduled for execution); (ii) addresses system-level effects rather than just program-level parallelism and overhead – multiple concurrent jobs' interactions and resource sharing, queue waiting times, network topology impacts, and different accelerator access modes (multi-tenant vs. exclusive).

Our parameters α, β, and γ capture job interaction patterns rather than program parallelism. **In particular, in neither scheduling scenario does γ correspond to the portion of the workload which is accelerated**. Specifically, in the exclusive scenario $1 - \gamma$ is the mean time before first accelerator call compared to the all other compute and accelerator time. While in the multi-tenant scenario, γ is more complicated to interpret, but can be thought of as measuring the relationship between the total accelerator time for the workload and the maximum classical work done by any job.

V. Validation of Our Analytical Model

This section compares our analytical model results against our simulation results and real system runs to ensure the validity of our models. We use a combination of trend lines and Z-scores to ensure that our model prediction are in acceptable error margins.

Experimental Setup. We present projections from our model for three representative classes of experiments. We present simulated runs with the median ATTs of $100,000\mu s$ and the largest ATTs of $10,000,000\mu s$.

Methodology. We fit our analytical speedup models to predict speedup from acceleration for a given configuration of topology, workload distribution, ATT, and scheduling mode. For **homogeneous workload distributions**, for example, if we use two data points – the first with a single accelerator and the second with two accelerators, we can compute the speedup gained from the addition of a second accelerator as follows: $S = \frac{T_1}{T_2}$, where T_1 is the execution time of a workload with one accelerator, and T_2 is the execution time of the same workload with two accelerators. S can then be plugged into our speedup equation with $k = 2$, and then solved for γ, which gives us:

$$\gamma = \frac{2S - 2}{S}$$

For **heterogeneous workload distributions**, we use the γ values obtained from their homogeneous constituents as a simple weighted sum:

$\gamma = frac_{graph}(\gamma_{graph}) + frac_{ml}(\gamma_{ml}) + frac_{physics}(\gamma_{physics})$

where $frac_x$ is the fraction of the application type x in the total workload, and γ_x is the γ value obtained from application type x's homogeneous workload runs. We can compute a γ value for any workload distribution if their homogeneous constituents' γ values are known.

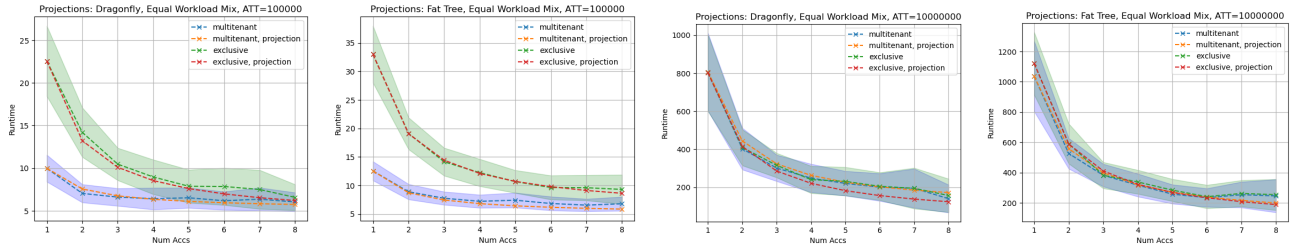
After obtaining γ and single accelerator execution time for a configuration of topology, workload distribution, ATT, and scheduling mode, we can use our speedup formulation given by eq. (1) to compute the speedup S for k accelerators. To obtain the execution time for k accelerators, we divide the single accelerator execution time by S.

As one of our validation metrics, we used Z-Score to calculate the variability of our experimental sets. The Z-score metric is calculated by the formula $z = (x_{observation} - \mu)/\sigma$, and it presents how many standard deviations the observations are away from the population mean. Thus, a score closer to zero tells us that a set of experiments is being predicted well despite the variability presented across other sets of experiments. Finally, a score can be positive or negative if it is below or above the mean. This method compares experimental setups with intrinsically different statistical properties as long as they follow normal distributions (as is our case with all our different experimental combinations). We use the score to compare the actual experimental mean against our predicted observation mean when one experimental dimension is fixed.

In our experiments, we also evaluated other aspects, such as accelerator and job placement, nearest accelerator scheduling, and the effects of changing the amount of communication between the accelerators and target apps. However, all of these configurations, except the nearest accelerator scheduling, resulted in little variance, leading to their ommision in our results. Further, we only report our load-balanced schedule since it outperformed the nearest-accelerator scheduling.

A. Simulated Runs

Model Accuracy. Figure 6a and Figure 6b visualize the performance estimations of our analytical models on simulated Dragonfly and Fat Tree topologies respectively for an ATT of 100K μs. The green and blue areas on the plots represent the confidence intervals of the observed runs within one standard deviation of the observed lines, which are means. We observe a clear performance separation between multi-tenant and exclusive scheduling modes until the very end of the plot on Fat Tree, but there seems to be a saturation point at six accelerators on Dragonfly, after which there



(a) Dragonfly, ATT = 100000 μs (b) Fat Tree, ATT = 100000 μs (c) Dragonfly, ATT = 10,000,000 μs (d) Fat Tree, ATT = 10,000,000 μs

Fig. 6. Model projections vs actual trend for simulated runs for moderate (100,000 μs) and large (10,000,000 μs) Accelerator Task Times (ATT).

is a significant overlap in performance with an increase in accelerators. Our models track the real performance well, with the most significant errors remaining under the one standard deviation confidence interval.

Similarly, Figure 6c and Figure 6d visualize the same information as above, but with large ATTs of $10,000,000 \mu s$. There is no clear separation between multi-tenant and exclusive scheduling modes in these plots, with a significant overlap between their distributions. There is also no observable trend difference between Dragonfly and Fat Tree runs. When workloads are accelerator-bound, multi-tenancy does not provide any speedup. However, similar to the median ATT projections, our model's errors remain well under the confidence intervals.

Impact of experimental features on variability. Tables IV, V, VI (sim rows), VIIa, and VIII (sim rows) represent the mean Z-scores for different experimental parameters (i.e., number of accelerators, workload distribution, topology, accelerator intensity, and mode of execution), showing how well the analytical model predicts performance across various system configurations and workload characteristics for simulated runs. This setup lets us see how the different features affect the overall experimental variability by fixing one of the parameters and seeing how the variability changes.

Dragonfly and Fat Tree configurations show similar prediction accuracy (table IV), with mean Z-scores of -0.05 and -0.07 respectively, indicating consistent model performance across network architectures. The impact of ATT duration (table V) shows interesting patterns, with relatively stable predictions (mean Z-scores between -0.01 and -0.04) for shorter ATTs up to 100,000μs, but slightly increased variability for longer ATTs of 1-10 million μs (mean Z-scores of -0.11 to -0.13). Table VI shows how the score changes when the number of accelerators is fixed for the experiments. We start with perfect predictions with slight increases in error as we increase the number of accelerators (although with small magnitudes). The scheduling mode analysis (table VIIa) demonstrates that exclusive scheduling predictions (mean Z-score -0.03) are marginally more accurate than multi-tenant predictions (mean Z-score -0.09). Across different workload distributions, the model maintains consistent accuracy (table VIII) with mean Z-scores around -0.06, with slightly higher variability for machine learning workloads (-0.08). **The overall trend in both the trend lines and the Z-scores showcase a great fit of the modeling data to the simulation results. Even when variability is high, the absolute variability among all dimensions is still relatively low across all datasets.**

TABLE IV. SIMULATION: MEAN OF Z SCORES GROUPED BY TOPOLOGY

	dragonfly	fattree
μ	-0.05	-0.07
σ	0.14	0.12

TABLE V. SIMULATION: MEAN OF Z SCORES GROUPED BY ATT

	1e+01	1e+02	1e+03	1e+04	1e+05	1e+06	5e+06	1e+07
μ	-0.04	-0.01	-0.02	-0.02	-0.03	-0.13	-0.11	-0.11
σ	0.07	0.07	0.08	0.07	0.07	0.19	0.16	0.19

TABLE VI. SIMULATION VS. REAL: MEAN OF Z SCORES GROUPED BY NUMBER OF ACCELERATORS

		1	2	3	4	5	6	7	8
Sim.	μ	0.00	-0.00	-0.01	-0.04	-0.06	-0.09	-0.15	-0.14
	σ	0.00	0.03	0.05	0.07	0.10	0.13	0.20	0.20
Real	μ	0.00	-0.02	-0.05	-0.05	-0.06	-0.07	-0.06	-0.06
	σ	0.00	0.06	0.09	0.08	0.11	0.11	0.12	0.13

TABLE VII. SIMULATION VS. REAL: MEAN OF Z SCORES GROUPED BY ACCELERATOR MODE

(a) Simulation

	exclusive	multitenant
μ	-0.03	-0.09
σ	0.12	0.14

(b) Real

	exclusive	multitenant
μ	-0.07	-0.02
σ	0.12	0.06

TABLE VIII. SIMULATION VS. REAL: MEAN OF Z SCORES GROUPED BY WORKLOAD DISTRIBUTION

		Physics	ML	Graph	M. Phys	M. ML	M. Graph	Equal
Sim.	μ	-0.06	-0.06	-0.06	-0.06	-0.08	-0.06	-0.06
	σ	0.14	0.14	0.14	0.13	0.12	0.13	0.11
Real	μ	-0.03	-0.12	0.01	-0.05	-0.14	-0.02	0.03
	σ	0.02	0.16	0.06	0.04	0.08	0.07	0.05

B. Real System Experiments

We conducted our next validation experiments on a system comprising 48 nodes, each equipped with an Intel(R) Xeon(R) Gold 6126 CPU running at 2.60GHz and 192 GB of RAM, with a Fat Tree network topology, interconnected via an EDR 100G InfiniBand fabric. For these experiments, we used the same applications as in our simulations, with $100 \mu s$ ATT. Note

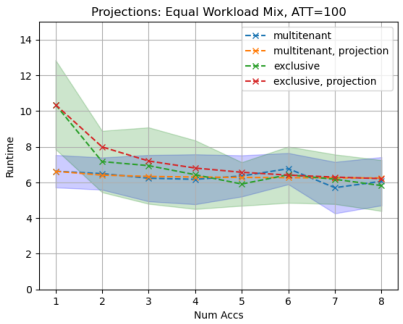


Fig. 7. Model Projections vs Actual Trend on a real system (ATT = 100 μs)

that we use simulated network accelerators in this setup by designating some nodes as accelerators.

Model Accuracy. Figure 7 illustrates performance projections versus actual trends on the real system using up to eight accelerators. These experiments were conducted on 32 nodes, compared to 512 and 686 nodes for the simulated Dragonfly and Fat Tree experiments. We observe that, there is a clear separation in performance until four accelerators, beyond which a significant overlap in performance occurs. The performance of multi-tenant mode saturates at a count of one accelerator, with no noticeable improvements in performance after that. Here, it also shows a much higher variability in runtime compared to the simulated versions of these runs, likely owing to network interference and other effects on real systems. Similar to the simulated runs, our models perform just as well in estimating performance on real system runs.

Impact of experimental features on variability. As with the simulated results in Section V-A, we collected the z-scores across all the experiments to test their variability (Tables VIIb, VIII, and VI (rows/columns annotated with "real")). While maintaining similar magnitude of prediction errors, the real system shows distinct behavioral patterns across different parameters. The workload distribution analysis (Table VIII) shows varying accuracy levels, with graph analytics achieving the best predictions (mean Z-score 0.01) and machine learning workloads showing higher variability (mean Z-score -0.12). Multi-tenant scheduling performs better than exclusive scheduling in real environments, with mean Z-scores of -0.02 and -0.07 respectively (Table VIIb). The accelerator scaling pattern (Table VI) differs from simulations, showing more stable predictions with increasing accelerator counts (Z-scores ranging from 0 to -0.07). **The increased variability in real system results can be attributed to network interference and system effects not present in simulations, though the overall trends and magnitudes remain comparable to simulation results.**

VI. Discussion

While the results from Section III highlight the benefits of multi-tenancy, the resulting equations from Section IV make such observations actionable. In particular, they provide a way to quantitatively reason about the trade-offs in performance across the scheduling paradigms for different machine configurations given the typical properties of a cluster's workload (i.e., maximum and average times for startup, ATT, and classical compute time). Given a workload and scheduling strategy, these results can be used to better understand the ideal number of accelerators for a cluster and to compare the average slowdown per job with the change in throughput. Further, by approximating γ, our model becomes pseudo-analytical, integrating runtime, hardware, and job characteristics into our equations, providing more accurate estimates for performance. While our analytical model is critical during the planning phase before standing up a cluster, the pseudo-analytical model provides a path for upgrading an existing cluster or bootstrapping existing job schedulers. We believe these types of analysis will help to explore the impacts of workloads and job scheduling on TCO, cluster configuration, and resource provisioning, among other key metrics.

VII. Related Work

Previous studies [24], [25] examined multi-tenancy for quantum accelerators with a focus on a single workflow. In contrast, our approach takes a more abstract and generalized view of accelerators, emphasizing the modeling of system throughput for jobs that do not require detailed simulations at the accelerator level.

The research presented in [26] provides a detailed examination of performance variability in a Cray XC Aries dragonfly network, and use this data to develop a machine learning-based performance forecasting model.. While our study shares some similarities with their approach, we distinguish ourselves by considering both exclusive and multi-tenant modes, thereby exploring the advantages of multi-tenancy. Furthermore, our predictive model is analytical in nature, offering a more computationally efficient alternative to the attention-based models employed in [26].

The study in [27] showcases the impact of multi-job workloads in Fat Tree networks. They use the simulation infrastructure Trace-CODES to explore design decisions (such as link bandwidth, number of rails, and number of planes, among others). They discovered that increasing bandwidth has limited gains when considering the computation length. They conclude that performance is affected by workload composition and that dual rail networks help with the degradation due to inter-job interference. This research aims to characterize and explore design decisions from the hardware perspective; we focus on the scheduling and prediction of performance in the face of heterogeneity and multitenancy from a system perspective.

Researchers in [28] created ML-based models based on gradient-boosted trees and forests of highly randomized trees to predict communication-heavy workloads' execution time. Moreover, their consequent analysis allowed them to identify the most relevant features and hardware components affecting network congestion and runtime. This analysis relied on post-mortem logs (not simulated results), and they do not consider the multi-job, heterogeneity, or multi-tenancy, as we do.

[29] shows an in-depth study of how parallel jobs are affected by job placement and message routing strategies. They

build a novel model to predict the traffic of individual links for different routing strategies. Their model provides a detailed view of link usage, and their experimentation focuses on Dragonfly and routing algorithms, plus job placement. However, our analysis and model considers the whole (heterogeneous) system view focusing on workloads and their interactions.

VIII. Conclusion

Emerging high-performance computing (HPC) systems are increasingly incorporating a variety of accelerators beyond traditional GPUs and FPGAs. To address the modeling needs for system-level cluster analysis, we present a novel analytical model and validation that projects the performance workloads with cluster configurations. We conducted a study with three classes of applications focusing on interleaved execution phases for a specific average turnaround time (ATT) and explores the benefits of multi-tenancy on system utilization.

To our knowledge, this is the first study to propose an analytical model for emerging multi-accelerator systems, demonstrating how various factors influence job performance. Our model aligns well with both simulation data and real-world experiments, showing slightly higher variance in real scenarios. This work paves the way for future research, including efforts to reduce prediction errors in the model and expand the testbed scale, enhancing our understanding of workload optimization and cluster balance.

Acknowledgment

We thank the reviewers for their helpful feedback. This research was supported in part by an award from the National Science Foundation under grant numbers OAC-2103510 and OAC-1807563, and by the U.S. Department of Energy's National Nuclear Security Administration (NNSA) under the Predictive Science Academic Alliance Program (PSAAP-III), Award DE-NA0003966. This research used funding and resources supported by the U.S. DOE Office of Science, Office of Advanced Scientific Computing Research, under award 66150: "CENATE - Center for Advanced Architecture Evaluation". The Pacific Northwest National Laboratory is operated by Battelle for the U.S. Department of Energy under contract DE-AC05-76RL01830. PNNL information release number: PNNL-SA-208649.

References

[1] D. P. Khatri, G. Song, and T. Zhu, "Heterogeneous computing systems," 2022. [Online]. Available: https://arxiv.org/abs/2212.14418

[2] T. Boku, "How fpga can contribute to hpc?" in *2022 International Symposium on VLSI Design, Automation and Test*, 2022, pp. 1–1.

[3] IBM, "Ibm debuts next-generation quantum processor, ibm quantum system two, extends roadmap to advance era of quantum utility," *IBM News Room*, 2023. [Online]. Available: https://newsroom.ibm.com/2023-12-04-IBM-Debuts-Next-Generation-Quantum-Processor-IBM-Quantum-System-Two,-Extends-Roadmap-to-Advance-Era-of-Quantum-Utility

[4] "A new brain-based supercomputer could revolutionize scientific discovery," *Sandia National Laboratories News Releases*, 2023.

[5] R. Prabhakar *et al.*, "Sambanova sn40l: Scaling the ai memory wall with dataflow and composition of experts," *arXiv:2405.07518*, 2024.

[6] S. Lie, "Cerebras architecture deep dive: First look inside the hw/sw co-design for deep learning: Cerebras systems," in *2022 IEEE Hot Chips 34 Symposium (HCS)*. IEEE Computer Society, 2022, pp. 1–34.

[7] RIKEN, "Riken selects ibm's next-generation quantum system to be integrated with the supercomputer fugaku," 2023. [Online]. Available: https://newsroom.ibm.com/2023-12-14-IBM-announces-collaboration-with-RIKEN-to-integrate-IBM-quantum-system-two-with-supercomputer-fugaku

[8] J. Robledo-Moreno *et al.*, "Chemistry beyond exact solutions on a quantum-centric supercomputer," *arXiv arXiv:2405.05068*, 2024.

[9] J. Wilke *et al.*, "Structural Simulation Toolkit (SST) Macroscale Element Library," https://github.com/sstsimulator/sst-macro, 2023, accessed: 2023-01-05.

[10] M. Jette, C. Dunlap, J. Garlick, and M. Grondona, "Slurm: Simple linux utility for resource management," *OSTI.gov*, 7 2002.

[11] IBM, "IBM Spectrum LSF Session Scheduler," https://www.ibm.com/docs/en/spectrum-lsf/10.1.0?topic=lsf-session-scheduler, 2024, modified : 2024-02-29.

[12] B. Nitzberg, J. M. Schopf, and J. P. Jones, *PBS Pro: Grid computing and scheduling attributes*. USA: Kluwer Academic Publishers, 2004, p. 183–190.

[13] J. J. Wilke and J. P. Kenny, "Opportunities and limitations of quality-of-service in message passing applications on adaptively routed dragonfly and fat tree networks," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, 2020, pp. 109–118.

[14] S. Young, S. Aksoy, J. Firoz *et al.*, "Spectralfly: Ramanujan graphs as flexible and efficient interconnection networks," in *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, IEEE. New Jersey, USA: IEEE, 2022, pp. 1040–1050.

[15] K. Wen *et al.*, "Flexfly: Enabling a reconfigurable dragonfly through silicon photonics," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE. New Jersey, USA: IEEE, 2016, pp. 166–177.

[16] K. S. Hemmert *et al.*, "Evaluating trade-offs in potential exascale interconnect technologies," Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States ..., Tech. Rep., 2020.

[17] J. Wilke *et al.*, "SST Macroscale Element Library Skeletons," https://github.com/sstsimulator/sst-macro/tree/master/sstmac/skeletons, 2023, accessed: 2024-06-20.

[18] S. Li, "DNN-cpp-proxies," https://github.com/spcl/DNN-cpp-proxies, 2022, accessed: 2024-06-20.

[19] R. Z. Sayan Ghosh *et al.*, "miniVite," https://github.com/ECP-ExaGraph/miniVite, 2021, accessed: 2024-06-20.

[20] J. Lenstra, A. Rinnooy Kan, and P. Brucker, "Complexity of machine scheduling problems," in *Studies in Integer Programming*, ser. Annals of Discrete Mathematics, P. Hammer, E. Johnson, B. Korte, and G. Nemhauser, Eds. Elsevier, 1977, vol. 1, pp. 343–362.

[21] C.-Y. Lee, "Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint," *Operations Research Letters*, vol. 20, no. 3, pp. 129–139, 1997.

[22] R. L. Graham, "Bounds for certain multiprocessing anomalies," *The Bell System Technical Journal*, vol. 45, no. 9, pp. 1563–1581, 1966.

[23] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, ser. AFIPS '67 (Spring). New York, NY, USA: ACM, 1967, p. 483–485.

[24] A. D'Onofrio, A. Hossain, L. Santana, N. Machlovi, S. Stein, J. Liu, A. Li, and Y. Mao, "Distributed quantum learning with co-management in a multi-tenant quantum system," in *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 2023, pp. 221–228.

[25] S. Upadhyay and S. Ghosh, "Stealthy swaps: Adversarial swap injection in multi-tenant quantum computing," in *2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID)*. IEEE, 2024, pp. 474–479.

[26] A. Bhatele, J. J. Thiagarajan, T. Groves, R. Anirudh, S. A. Smith, B. Cook, and D. K. Lowenthal, "The case of performance variability on dragonfly-based systems," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2020, pp. 896–905.

[27] N. Jain *et al.*, "Predicting the performance impact of different fat-tree configurations," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17, New York, NY, USA, 2017.

[28] A. Bhatele, A. R. Titus, J. J. Thiagarajan, N. Jain, T. Gamblin, P.-T. Bremer, M. Schulz, and L. V. Kale, "Identifying the culprits behind network congestion," in *2015 IEEE International Parallel and Distributed Processing Symposium*, 2015, pp. 113–122.

[29] N. Jain, A. Bhatele, X. Ni, N. J. Wright, and L. V. Kale, "Maximizing throughput on a dragonfly network," in *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014, pp. 336–347.