SC20: International Conference for High Performance Computing, Networking, Storage and Analysis | 978-1-7281-9998-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/SC41405.2020.00088

Job Characteristics on Large-Scale Systems: Long-Term Analysis, Quantification, and Implications[*]

Tirthak Patel
Northeastern University

Zhengchun Liu, Raj Kettimuthu
Argonne National Laboratory

Paul Rich, William Allcock
Argonne National Laboratory

Devesh Tiwari
Northeastern University

***Abstract*—HPC workload analysis and resource consumption characteristics are the key to driving better operation practices, system procurement decisions, and designing effective resource management techniques. Unfortunately, the HPC community does not have easy accessibility to long-term introspective workload analysis and characterization for production-scale HPC systems. This study bridges this gap by providing detailed long-term quantification, characterization, and analysis of job characteristics on two supercomputers: Intrepid and Mira. This study is one of the largest of its kind – covering trends and characteristics for over three billion compute hours, 750 thousand jobs, and spanning a decade. We confirm several long-held conventional wisdom, and identify many previously undiscovered trends and its implications. We also introduce a learning based technique to predict the resource requirement of future jobs with high accuracy, using features available prior to the job submission and without requiring any application-specific tracing or application-intrusive instrumentation.**

***Index Terms*—High Performance Computing, Large-Scale Systems, Monitoring, Queueing Analysis, Statistical Analysis**

I. Introduction

Understanding job characteristics and their impact on the system is critical to efficiently managing and operating a large-scale system. Various communities, from enterprise computing to cloud infrastructure providers to (High-Performance Computing) HPC centers, invest significant resources in procuring and planning for the next generation of systems. Large-scale system workload characterization drive the decision making for new system procurement and the research efforts that design and evaluate new techniques to improve HPC resource management. Unfortunately, the lack of frequent and periodic characterization studies leads to research efforts over-fitting their technique to old characteristics. As a result, these efforts are unable to show effectiveness in fast-changing workload and computing environments. As a side-effect, system operators are reluctant to incorporate new research studies because they are unsure of the effectiveness of these techniques on the new systems.

Multiple studies perform workload analysis for a given system, but often consider only a small time period due to the infeasibility of collecting data over long periods. These studies are therefore limited to only small- or medium- scale systems [7, 45, 46, 49]. Further, these studies do not provide detailed long-term trends [4, 8, 18, 25, 28, 36, 36, 37, 39, 44, 49, 50, 56]. One of the latest and richest workload analysis appeared in USENIX ATC 2018 which covers job characteristics from 1600-node LANL Mustang cluster for five years and 9000-node LANL Trinity supercomputer for only three months [6]. Unfortunately, while very useful for comparison between HPC and enterprise computing workload characteristics, this study only covers certain aspects: job size, job length, job submission rate, and job outcomes. This study does not cover a detailed analysis of the different changes in job characteristics over time, queue wait time trends, similarity among jobs being submitted from the same user, and across users. Even several recent studies and traces released from cloud computing providers and data center operators including Google, Microsoft, Alibaba do not cover these aspects [13, 20, 24, 26, 38, 43, 52–54].

In general, the HPC community does not have easy accessibility to introspective workload analysis and characterization. This study bridges this gap by providing insights from long-term workload analysis of two large-scale production HPC systems, which includes characteristics beyond traditional job characteristics covered by the latest studies [6, 13, 20, 24, 26, 38, 43, 52–54]. This study paves the way for researchers to incorporate these new trends and aspects as they develop novel resource management strategies. **Contributions of this paper include:**

- We perform a detailed and careful analysis of job characteristics on two leadership-class systems, Intrepid and Mira, at Argonne National Laboratory. Our in-depth characterization study is the largest study of its kind, analyzing 750 thousand jobs from 1300 users over 3 billion compute core hours from among the largest supercomputers in the world. We explain our findings with the context of operational and scheduling policies.

[*]A part of this work was performed during an unprecedented time – around the peak of the COVID-19 pandemic. We were able to conduct this scientific study only because first-line responders and essential workers worked tirelessly to keep our community safe and functioning – we are eternally grateful to them. This paper is dedicated to the memories of all the first-line responders and essential workers who sacrificed their lives trying to keep ours safe.

- Our study confirms several long-held conventional wisdom and expectations that continue to be true.

  Such findings include: *HPC system utilization is always high without any distinguishable periodic peaks, HPC users often overestimate the run time of their jobs by a large margin, and a small fraction of users consume almost all of the system's node-hours.*

- Our study discovers and explains many previously undiscovered and non-quantified insights about how HPC job characteristics have evolved. We briefly summarize some notable examples.

  HPC jobs are becoming longer and larger, but the total number of core hours delivered by the system is not necessarily dominated by large jobs. Instead, medium-sized, long-running jobs account for most core hours delivered by these machines.

  The queue wait times for resources are increasing rapidly and are often more than 7× the run time, especially for large jobs. We observed that users have reacted to this trend by opting for relatively-longer running, medium-sized jobs that account for most of the compute core hours delivered.

  While users often overestimate the run time of their jobs, we found that the characteristics of these over-estimations are often specific to the queue and have remained similar over time. Our study also discovers that a high degree of similarity exists in the resource consumption of the jobs belonging to the same user and across users. We leverage this finding to develop a prediction scheme that predicts the job resource usage prediction accurately (less than 15% median estimation error) even before the job starts.

  Our analysis reveals that HPC resource management strategies may have "unintentional" unfairness side-effects. For example, the total queue wait time may not be correlated with the total resource consumption of similar users and jobs.

- This paper also introduces a simple learning based technique that can predict the resource requirement (execution time and number of cores) of future jobs from a user with high accuracy, using basic features (i.e., user Id, project Id, and job queue Id) that are available before job submission and without requiring any application-specific tracing and application-intrusive instrumentation. This kind of prediction can be used by the job scheduler to anticipate the demand for system's resources and optimize resource management.

Out dataset and analysis tools are open-sourced and available at `https://doi.org/10.5281/zenodo.3957897`.

II. Background & Methodology

TABLE I. System description and comparison of two ALCF supercomputing systems: Intrepid and Mira.

	Intrepid	**Mira**
System Architecture	IBM Blue Gene/P	IBM Blue Gene/Q
Node Architecture	IBM PowerPC 450	IBM PowerPC A2
Interconnect Config.	3D Torus	5D Torus
Number of Racks	40	48
Number of Nodes	40,960	49,152
Processor Speed	850 MHz	1.6 GHz
Number of Cores	163,840	786,432
Total Memory	80 TB	760 TB
Peak Performance	557 TFlops	10 PFlops
Began Full Production on	February 2009	January 2014
Retired on	December 2013	December 2019

A. System Description

Overview of Systems: We consider two leadership-class systems at the Argonne Leadership Computing Facility (ALCF) at Argonne National Laboratory (ANL): *Intrepid and Mira*.

Intrepid was a 557 TFlops IBM Blue Gene/P system that entered full production in February 2009. It consisted of 40,960 850 MHz IBM Power PC 450 nodes, each with four cores and 2 GB memory. Each node on Intrepid was connected to separate networks for I/O and inter-node communication. The inter-node communication interconnect was set up with a 3D Torus configuration. The system was decommissioned in December 2013, and replaced by its successor, Mira [12]. Table I provides more Intrepid details.

Mira was a 10 PFlops IBM Blue Gene/Q system, providing about 20x performance of Intrepid. Mira consisted of 49,152 1.6 GHz IBM PowerPC A2 nodes, each with 16 cores and 16 GB memory. The nodes on Mira were connected using 5D Torus topology. Mira was the primary leadership-class supercomputing system at ALCF until its decommissioning in 2019. Mira entered production in April 2013; however, it began handling full ALCF workload in January 2014 (after Intrepid was decommissioned) [12]. Detailed specifications about Mira are summarized in Table I.

Applications: Both of these ALCF supercomputers have served a variety of scientific domains over the years. These include materials science (QMCPACK and CPMD), biology (NAMD), transportation infrastructure, energy production (MADNESS and MPQC), nuclear combustion (FLASH and GFMC), cosmology (HACC), quantum (MILC, Chroma, and CPS) and plasma (GTC and GTS) physics, and chemistry and climate modeling (HIRAM and CM4+) [11, 55]. These workloads display a variety of resource usage requirements.

Resource Allocation: Both systems support jobs with run times of up to 24 hours. Both systems allow node allocations in pre-configured arrangements that support the Mesh or Torus topologies. Therefore, the number of nodes that a job can be allocated cannot be arbitrarily selected by the users; instead, it is chosen from one of the allowed job-size buckets (512, 1024, 2048, etc.). The minimum number of nodes that a job on Intrepid and Mira can run on is 512, as is enforced by policy. Jobs requesting the number of nodes that do not belong to one of the job-size buckets have to wait for the next largest node configuration to be available [2, 4]. For example, if a job on Mira requests 3000 nodes, the scheduler will wait until a partition of 4096 nodes connected using Torus topology becomes available. It will then schedule the job on that partition, allocating all 4096 nodes to the job.

Job Queue Policies: Different queue partitions are available for scheduling jobs of different sizes, expected run times, and priorities. Both, Intrepid and Mira have four primary queues: *backfill*, *prod-capability*, *prod-long*, and *prod-short*. Mira has additional queues that support specific Torus configurations for 1K and 32K jobs (these jobs get allocated a partly Mesh topology in regular queues). However, these queues do not observe significant traffic. The *prod** queues support production jobs of different sizes (e.g, **-short* and **-long* for jobs requiring less than or equal to 4K nodes and **-capability* for jobs requiring more) and lengths (e.g., **-short* for jobs less than 6 hours and **-long* for jobs between 6 and 12 hours). Jobs in *prod** queues receive higher priority over jobs in *backfill* queue. The *backfill* queue supports jobs of all sizes up to 6 hours long. The queue is only available to projects which have surpassed their allocations, and it only schedules jobs when no other job is available to use the system resources [1].

B. Description of Dataset

The analysis dataset contains entries for each job submitted to the system; each entry includes information such as anonymized user Id, project Id, job Id, queue partition, submission time, start time, end time, number of nodes, number of cores, wall time requested, actual run time, etc. The analysis is not performed on application Id or name, nor the configurations and/or input parameters of the applications to ensure anonymity. Information about the jobs' CPU utilization, memory usage, and network behavior is also not measured at per job granularity. We use the Intrepid datasets from 2009 to 2013, and Mira datasets from 2014 to 2018. Note that while Intrepid was active in 2008 and Mira was active in 2013, both did not enter full production until 2009 and 2014, respectively. Therefore, we do not use the Intrepid dataset from 2008 and the Mira dataset from 2013. Our analysis does not include 2019 for Mira as the system was only recently shutdown (a few weeks before the paper submission deadline), and logs have not been completely sanitized for an error-free analysis.

C. Summary of Terminology

Job: A job is a single instance of an executable that is run on the system. **User:** A user has the permission to execute one or more jobs on the system. **Wait time:** A job's wait time is the total amount of time it spends in the queue from the moment it is submitted to the moment it starts running. Note that a job might not be eligible to run when it is submitted (e.g., a user might have other jobs running in parallel, filling up the quota), in which case, *wait time is calculated from the moment it becomes eligible*. **Run time:** A job's run time is the total amount of time it takes to execute from the moment it starts running to the moment it finishes running. **Number of nodes:** A job's number of nodes is the number of compute nodes it executes on. **Wall time:** A job's wall time is the amount of time it requests the node resources for. Note that this time is not necessarily equal to the run time. It is typically much longer than the run time as users tend to overestimate the run times of their jobs [17, 36, 57]. **Core hours:** Core hours is a metric used to characterize the total resource utilization of the job. It is equal to the number of cores that the job is running on multiplied by its total run time. Generally, node hours is also used as a metric to characterize total resource utilization. However, we commonly use core hours as the number of cores per node is different for Intrepid and Mira.

Note that for a large part of our analysis, we use the "median" of a distribution (e.g., the median number of core hours per job, the median run time of jobs, etc.) and not the "mean" of a distribution. This is due to two reasons: (1) the analysis is empirical and driven by data; therefore, the distributions are not guaranteed to be symmetrical (typically, mean is used for symmetrical distributions), and (2) outlier value (e.g., excessively long wait times of some jobs) can skew the mean considerably. However, such data points do not have any impact on the median.

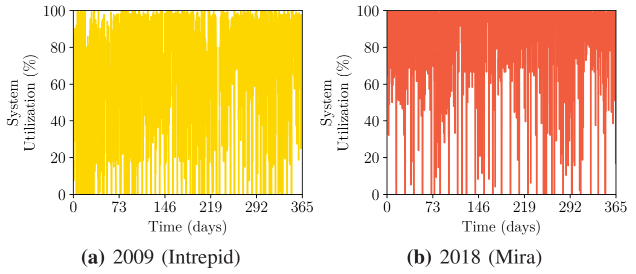


Fig. 1. System utilization during the years 2009 and 2018.

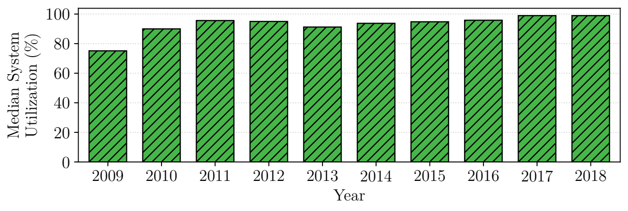


Fig. 2. Median system utilization remains high over the years.

III. Overall System Utilization Trends

We begin this study by analyzing trends in the overall system utilization over the last decade from Intrepid to Mira.

Fig. 1 shows the system utilization for the two years at both of the ends of the period considered in this study (2009 and 2018). First, we observe that in general the system utilization for these capability-based supercomputers are typically fairly

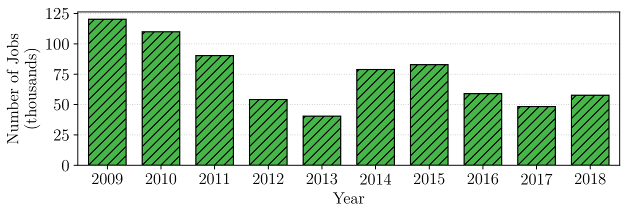


Fig. 3. The total number of jobs served by the leadership-class systems has decreased by 50% from 2009 to 2018.

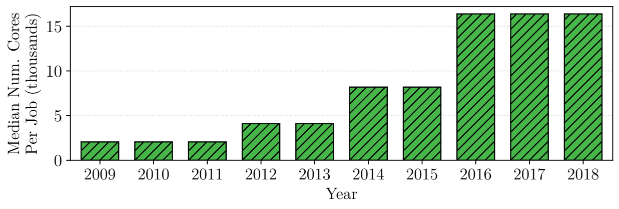


Fig. 4. Median number of cores used per job has increased from about 2K cores in 2009 to about 16K cores in 2018.

high throughout the year. Fig. 2 shows that median system utilization for all years from 2009 to 2018 remains consistently high over the decade. This trend aligns with the expectation since leadership-class HPC systems serve long-running computationally-intensive scientific workloads, resulting in high utilization at all times. To better understand the scope of these trends, we performed utilization analysis at the week and month granularity. However, seasonal peaks in utilization or distinct periodic temporal patterns in the utilization behavior were not found. These observations hold across all the years considered in this study. These findings are in stark contrast to enterprise computing systems, which typically have very low utilization (less than 30% on a median) and exhibit bursty behavior in utilization during peak periods [58, 60, 61].

Finally, we calculated that these leadership class systems (Mira and Intrepid) have served increasingly more number of total core hours over the years – a key metric for the effectiveness of these systems. The total number of core hours served by these systems has increased from 1 billion core hours to over 6 billion core hours, which is an increase of 6× (not shown for brevity).

Summary. *Over the past decade, resource utilization of the leadership class systems at ALCF (Intrepid and Mira supercomputers) has increased significantly – almost nearing full utilization most of the time. These systems experienced a dramatic increase in the demand for core hours (6× increase). This increase in demand for core hours is significant primarily because the processors have also become faster by almost 2×.*

IV. Investigating Trends in Job Characteristics

In this section, we characterize and analyze the resource consumption characteristics of jobs. First, we plot the total number of jobs executed per year from 2009 to 2018 (Fig. 3). The results reveal an interesting and somewhat counter-intuitive trend. The number of jobs run per year has been on a significant decline in the last ten years, experiencing an almost 50% decrease. In 2009, Intrepid executed nearly 120,000 jobs, but Mira ran less than 60,000 jobs in 2018. We note that during the years 2014 and 2015, there was an increase in the number of jobs executed, this is because Mira went online for users and early adopters of the new system might have submitted more jobs to fully move to a new system.

Unfortunately, the decrease in the total number of jobs executed does not trivially align with our observation that more core hours were served by these systems (Sec. III). To investigate this further, we analyzed the characteristics of jobs (e.g., job size and run time) to explain these observations consistently. Our results indicate that the median number of cores per job has increased by almost 7× from 2009 to 2018 (Fig. 4). To investigate this finding further, we plotted the histogram of job size for years 2009 and 2018 (Fig. 5). Consistent with our observed trend in Fig. 4, Fig. 5 shows that the fraction of jobs running on larger nodes has increased significantly. For example, in 2009, 75% of all jobs ran on 512 nodes, but in 2018 more than 50% of all jobs ran on more than 512 nodes. An increase in job size is due to an increase in emphasis on parallelizing and scaling out jobs. It would have been particularly alarming if the median job size was on a significant decline for a capability-based computing system such as Mira.

Motivated by the finding that jobs have generally become larger, we investigate how other job-related characteristics (median run time and core hours consumed per job) have changed over time. Fig. 6 shows that the median run time

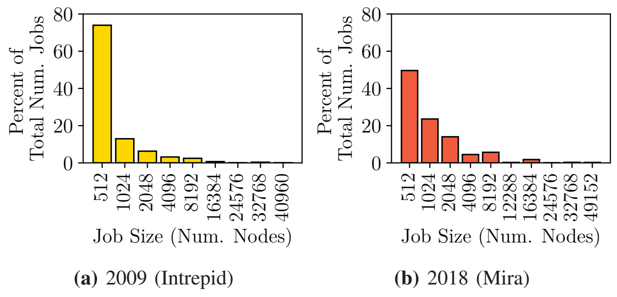


Fig. 5. The fraction of total jobs running on larger node counts has increased from 2009 to 2018.

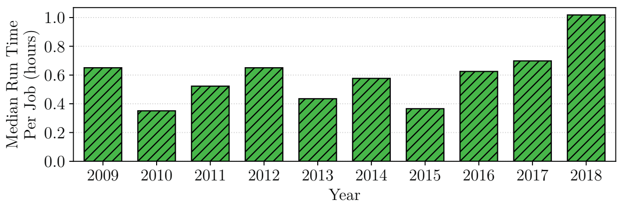


Fig. 6. Median run time of HPC jobs has increased from about 0.6 hours in 2009 to over 1 hour in 2018.

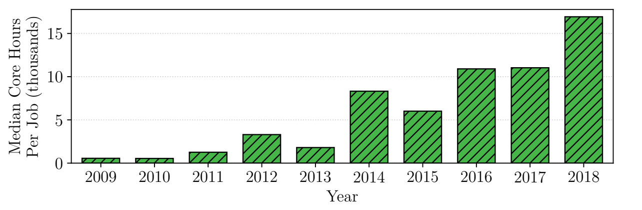
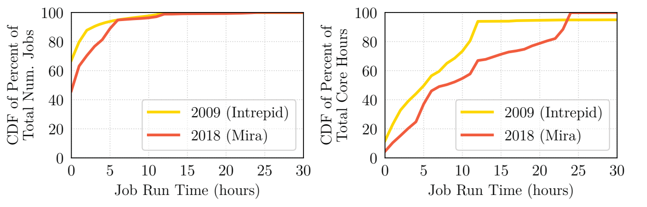


Fig. 7. There has been an increase in the median number of core hours per job from 2009 to 2018. The median number of core hours consumed by each job has increased by over 18×.



(a) CDF of Num. Jobs **(b)** CDF of Node Hours

Fig. 8. The CDFs of job run times show that the median run time of jobs has increased from 2009 to 2018.

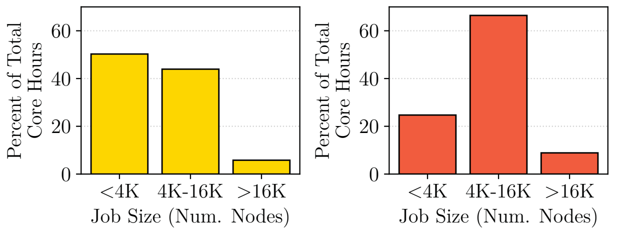
per job has increased from roughly 0.65 hours to over 1 hour (≈60% increase). Interestingly, the median core hours per job have also increased significantly from 2009 to 2018, as shown in Fig. 7. The median number of core hours has slowly grown from less than 1,000 core hours per job to more than 18,000 core hours per job (18× increase). This is essentially the combined effect of the increase in both median job size and median run time per job. The switch can explain the noticeable increase in the median job size and median run time per job in 2014 from Intrepid to Mira.

Summary. *Over the past decade, HPC jobs on leadership-class systems at ALCF have experienced interesting changes in their job characteristics. While the number of jobs executed has decreased, the median job size and run time per job have increased significantly. Consequently, the median number of core hours consumed per job has also increased by a factor of almost 18×. These findings have important implications for HPC resource management for future systems. These trends will exacerbate system-level challenges since long-running jobs are more likely to experience system failures [10, 14, 16, 30, 47, 51] and larger jobs are likely to suffer from manufacturing variability among nodes and locality issues in MPI communication [3, 19, 23, 42].*

Next, we analyze how the increase in job characteristics such as run time and size affect the composition of the total number of jobs and compute core hours delivered by the system over the years from 2009 to 2018.

Fig. 8(a) shows that in 2009, 80% of the jobs ran for less than 1.5 hours, but in 2018 only 60% of the jobs ran for 1.5 hours or less. Additionally, there is a significant change in the jobs that consume the most number of core hours. In recent years, longer jobs are consuming a large fraction of the core hours (Fig. 8(b)). In 2018, jobs running for more than 15 hours were responsible for consuming 25% of the total core hours. In contrast, in 2009, such jobs consumed only 5% of the core hours. *In summary, the number of jobs running for a longer time has increased and longer jobs also now consume a large fraction of total core hours delivered by the system.*

Naturally, this leads us to investigate how jobs of different sizes contribute toward the system core hours. Fig. 9 shows the total number of core hours consumed by jobs of various sizes: small (<4K nodes; less than 10% of the machine), medium (4K-



(a) 2009 (Intrepid) **(b)** 2018 (Mira)

Fig. 9. For Mira, jobs of medium size contribute the most toward the the total core hours delivered.

16K node; 10%-30% of the machine), and large jobs (>16K node; >30% of the machine). We observe that on Intrepid, most of the core hours were consumed by both small- and medium-sized jobs. However, on Mira, this trend is no longer true. Medium-sized jobs consume most of the core hours. Usually, one can expect most core hours to be consumed by the large jobs for capability-focused systems. But, our analysis reveals this is not necessarily true. This is because the median run time of medium-sized jobs has increased significantly compared to jobs of other sizes (Fig. 10). Users are now submitting medium-sized jobs because the waits time for larger sizes tends to be longer, as shown in the Sec. V. Smaller-size jobs do not exploit enough parallelism. Hence, this trend shows a trade-off between wait time and work done per job.

Summary. *Our analyses reveal that the total number of core hours delivered by the system is not necessarily dominated by large jobs as reported by other studies [29, 36, 45]. Instead, the total number of core hours provided by the system can broadly be governed by long-running, but medium-sized jobs.*

V. Analyzing Job Queue Wait Time Trends

First, our results show that the median queue wait time for jobs has increased drastically over the past decade (Fig. 11). The median wait time was just 0.1 hours in 2009, increased to over 1.2 hours in 2018, and was exceptionally high in 2017 (over 2.5 hours).

When the system is newly operational, the wait times tend to be smaller since users are slowly beginning to use the system. But, the wait times increase as more and more users adopt the new system. This is why we observe the drop from 2013 to

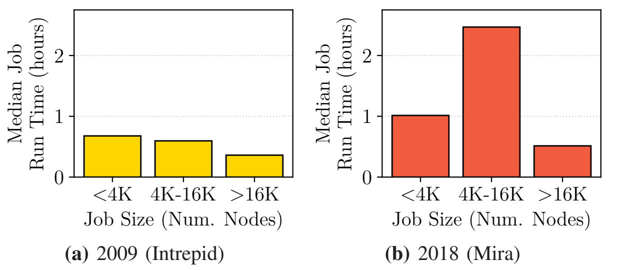


Fig. 10. Jobs of medium size have the highest median run times. Comparatively, small jobs and especially large jobs generally run for fewer hours.

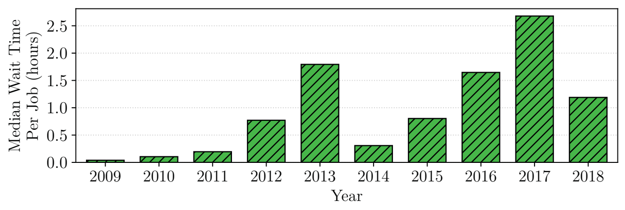


Fig. 11. The median wait time of jobs submitted to the large-scale HPC system has increased by almost 10×.

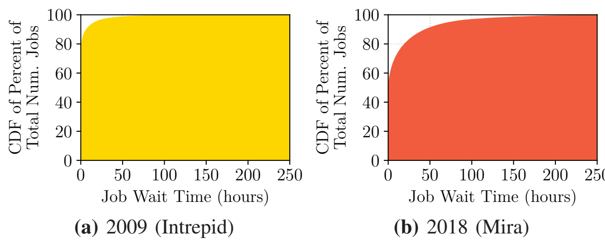


Fig. 12. Wait times are much longer in 2018 and 2009. In 2009, more than 80% of the jobs wait for 1 hour or less. In 2018, only 50% of the jobs wait for 1 hour or less.

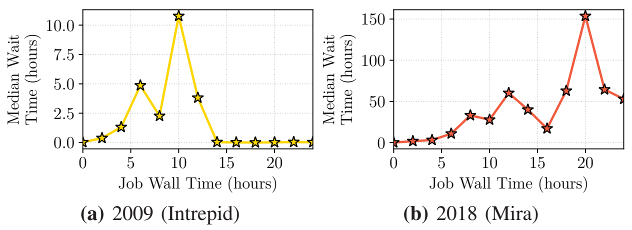


Fig. 13. Generally, jobs with longer wall times have higher wait times. On Intrepid, jobs with very long wall times tend to be high-priority and have small wait times.

2014 as users transitioned to the new system (Mira). Fig. 12(a) and (b) compare the wait-time distributions for all jobs in year 2009 and 2018. Not only the median but the distribution of job wait times has changed significantly too. In 2009, more than 80% of the jobs waited for one hour or less, but in 2018, more than 50% of the jobs waited for one hour or more, thus, suggesting a large increase in wait times.

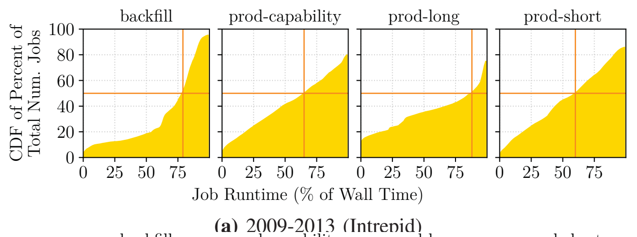
We analyze trends in job wait times further by looking at how job wait time changes with a job's requested run time or wall time (Fig. 13). We observe that the wait time increases for jobs with longer wall times – this is expected since it is harder to reserve a resource for a more extended period. However, interestingly, the wait times for jobs longer than 10 hours is almost 50 hours or more in 2018. In 2009, even jobs with wall time of 10 hours had to wait for a little over 10 hours. This trend generally gets worse as the job wall time increases. We note that jobs with very long wall time observe a relatively less wait time, especially on Intrepid. This is because these are likely to be mission-critical scientific applications that are provided a higher priority on a case-by-case basis (e.g., Gordon Bell runs). On Mira, even jobs with longer wall times have higher wait times because, as we observed in Sec. IV, Mira has many more jobs (even medium-sized jobs) with long run times than Intrepid.

To understand the reasoning behind the trend of increasing wait times and study how this can be mitigated, we explored the relationship between the requested time (wall time) and the run time (actual time taken to run). For example, if a job's run time is far less than its wall time, this could create sources of inefficiency for the job scheduler and specific jobs might have to wait longer, because the scheduler could not plan optimally due to misleading estimations about the job run times from the users.

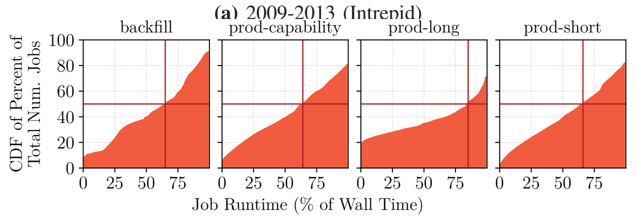
Fig. 14 quantifies the inaccurate estimations for Intrepid and Mira over their lifetimes for all four major queues. We make two major observations. First, we find that users indeed provide significantly inaccurate estimations about run times across different queues and on both the systems. Fig. 14 shows that most jobs run for much less time than the wall time they request. This finding is not new by itself as multiple other studies have observed this user behavior [17, 36, 57]. However, surprisingly this trend has not improved over the years despite much research and efforts. Fig. 14(a) shows that in the *prod-capability* queue on Intrepid, 50% jobs consumed 65% or less time than the requested wall time. Similar behavior can be seen for jobs in the *prod-capability* queue on Mira.

Second, we observe that the "shape" of inaccurate estimation is specific to the queue type, and it remains consistent over time. In other words, this behavior of overestimation is different for jobs submitted to different queues but similar across the two systems. For instance, 50% of the jobs submitted to the *prod-short* queue on Mira run for over 85% of requested wall time, indicating better estimation than jobs submitted to *prod-capability*. This shows that overestimation correction techniques should be aware of the queue-level effects, and not just learn about the user-specific behavior [17, 57]. The second implication is that if the overestimation correction technique is ported from one queue to another, it is likely to not yield the same benefits.

To further demonstrate the severity of job wait time trends, we analyzed the median wait time, requested wall time, and

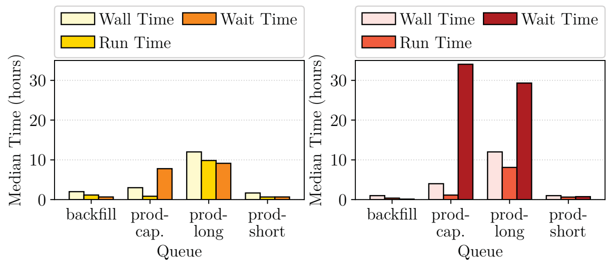


(a) 2009-2013 (Intrepid)



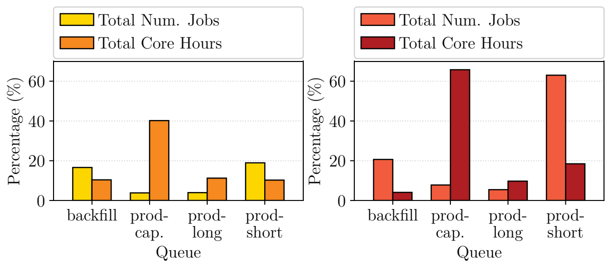
(b) 2014-2018 (Mira)

Fig. 14. Across different queues, most of the jobs run for much less time than the wall time they request. This is true for both the systems. Moreover, different queues have different overestimation behaviors ("shape" of the CDF is different across queues but remains similar for the same queue on both the systems).



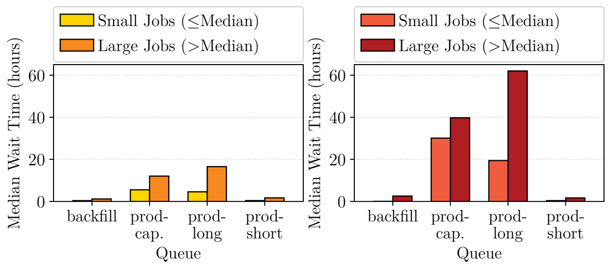
(a) 2009-2013 (Intrepid) **(b)** 2014-2018 (Mira)

Fig. 15. Job wait times have increased dramatically (they are much longer than job run times and even wall times), especially for *prod-capability* and *prod-long* queues.



(a) 2009-2013 (Intrepid) **(b)** 2014-2018 (Mira)

Fig. 16. Jobs in the *prod-capability* queue consume the biggest fraction of core hours (40% on Intrepid and 70% on Mira). But *prod-short* queue runs the most number of jobs.



(a) 2009-2013 (Intrepid) **(b)** 2014-2018 (Mira)

Fig. 17. Larger jobs generally spend more time in the queue. The median queue time has increased drastically from 2009 to 2018.

run time for different job queues on both the systems (analysis performed across their lifetimes). Fig. 15 shows that the median wait time is much longer than the median run time and median wall time submitted to all queues on the Mira supercomputer. This includes the *prod-long* queue which runs jobs with size of 4096 nodes or less [1]. On Intrepid, the median wait time was, in fact, less than the median job run time for the *prod-long* queue. Fig. 16 shows that *prod-long* receives less than 10% of all jobs, and the jobs which run on it consume less than 10% of all core hours. However, it still has a median wait time of almost 30 hours. On the other hand, *prod-short* is the busiest queue on Mira, running 65% of all jobs, and still has a short median wait time. Note that *prod-capability* does not serve a lot of jobs, but jobs that run on it consume 40% of all core hours on Intrepid and 70% of all core hours on Mira. In this queue, the median wait time is more than 7× of even the wall time in many cases. This result also shows that this severe wait time trend is not limited to only specific queues.

Next, we also analyze the relationship between job size and wait time. Fig. 17 shows that the median wait time is typically longer for large jobs. This is an expected trend since it takes a longer time to reserve more nodes. However, surprisingly, the difference in wait time between small and large jobs is less drastic on Mira than on Intrepid (even though overall, the wait times are much worse). For example, on Intrepid, for the *prod-capability* queue, small jobs have a median wait time of 5.5 hours, and long jobs have a median wait time of 12 hours, which is a degradation of 2.2×. On Mira, for the *prod-capability* queue, small jobs have a median wait time of 30.1 hours, and long jobs have a median wait time of 39.8 hours, which is only 1.3× worse (30% more wait time).

This finding can be explained in conjunction with our previous result, where we observed an increase in the medium-sized jobs on Mira and their corresponding run time (Sec. IV). Users on Intrepid observed significant wait time difference in small vs. large jobs (5.5 hours vs. 12 hours). Hence, when they transitioned to Mira, they tended to request for a more modest number of nodes to reduce the wait time, but increase the wall time in hopes to perform a similar amount of work. This resulted in a relatively less gap in the wait times (2.2× on Intrepid vs. 1.3× on Mira). But, it increased the number of medium-sized jobs and their median run times.

Lastly, one more factor apart from wall time and job size, which can affect the wait time of a job is the time of submission of the job. Jobs submitted during times when the system

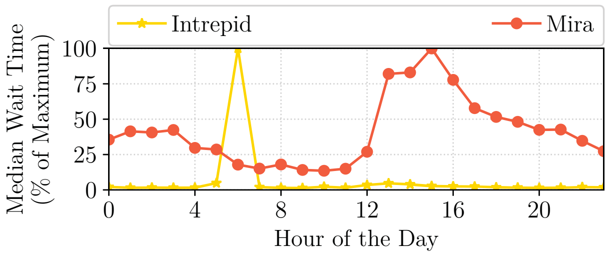


Fig. 18. Jobs run during the afternoon hours (1pm to 5pm) experience over 4× the amount of median wait time as jobs run during the morning hours (6am to 11am).

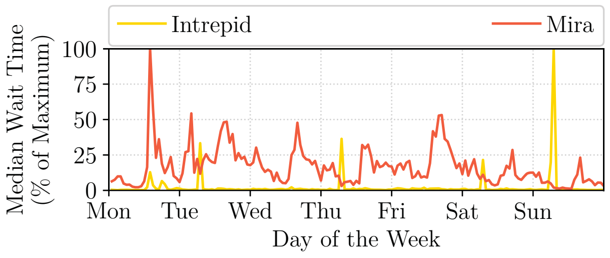


Fig. 19. Wait times are higher during the afternoon hours of all days. In fact, wait times are much higher during the beginning of the week and they get smaller as the week progresses.

utilization is high and/or when there are many jobs ahead of them in the queue may experience much longer wait times. Fig. 18 shows the relative wait times of jobs submitted at different hours during the day on the two systems. Jobs run during the afternoon hours on Mira (1 pm to 5 pm) experience over 4× the amount of median wait time as jobs run during the morning hours (6 am to 11 am). This indicates a lack of effort to amortize jobs throughout the day. In fact, when we look at wait times for hours during a week (Fig. 19, we see similar trends of increased wait times during the afternoon hours of all days throughout the week on Mira. In fact, wait times are much higher during the beginning of the week (Monday), and they get smaller as the week progresses toward the weekend. This again indicates that leadership-class facilities can benefit from encouraging users to submit jobs at pre-determined times to amortize job traffic throughout the day and week. Note that the trend is similar for Intrepid; however, it is not visible due to the sharp increase in wait times at 6 am of Sundays, which corresponds to management runs, which held up user jobs in the queue for a long time.

Summary. *The most alarming trend in HPC resource management is the drastic increase in job wait times. The median job wait time has increased by almost a factor of 10 in the last ten years. An implication of this finding could be careful and deterministic co-location of jobs that are less interference-sensitive, on the same node. Our study shows that a wait-time mitigation technique should take into account: (1) inaccurate estimations of the requested wall time, and (2) a job's queue, and (3) the time of the day during which a job is submitted.*

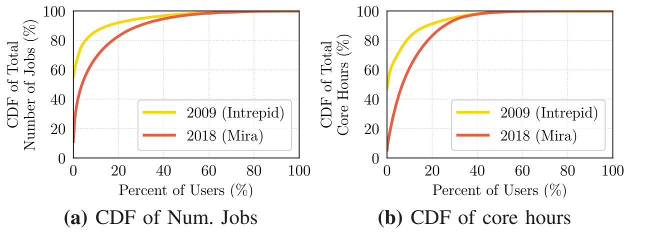


Fig. 20. The percent of users that executed most of the jobs executed increased from 2009 to 2018.

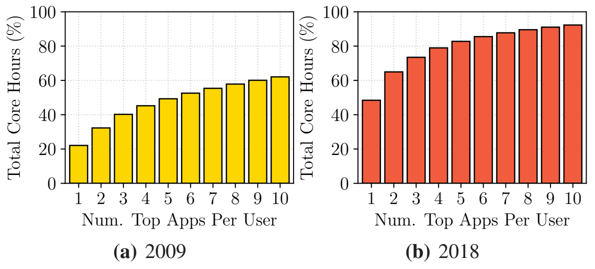


Fig. 21. The plot shows the percent of total core hours consumed by top n (highest core hours consuming) job-groups per user. Users tend to execute jobs with similar "resource-configuration" characteristics.

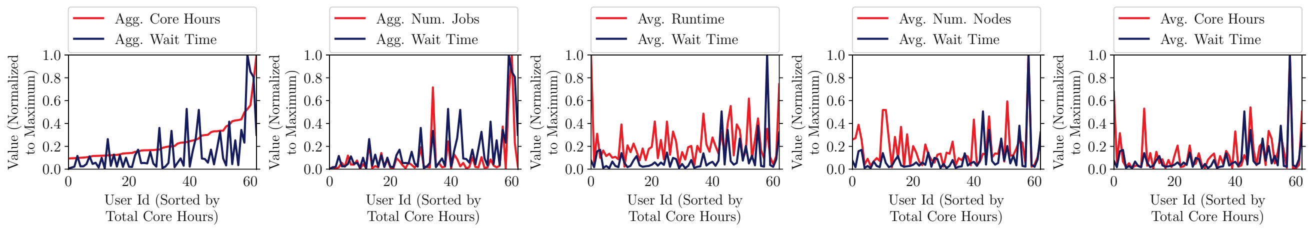
VI. Finding Trends in User-level Behavior

Next, we investigate how user characteristics and their corresponding job behaviors have changed over the years.

First, we observe that the number of users that consume most of the resources is still relatively small, less than 20% (Fig. 20). This finding has also been observed by other researchers. However, the distribution of such users has changed significantly over the years. As Fig. 20 shows, the curve is getting less steep. In 2009, the top 10% of the users consumed roughly 80% of the core hours, but in 2018 top 10% of the users consumed only 60% of the total core hours. We observe similar trends for the total number of jobs executed.

Next, we dive deeper into the job characteristics of every user to identify important job "resource-configurations" and explore the implications that these implications may have on overall system operations and resource management. We define job "resource-configuration" as a pair of <number of nodes, run time>. Note that jobs from different users may have the same job resource-configuration, and the same user may have multiple different job resource-configurations. Note that we consider run time instead of requested wall time to avoid bias as numerous users may ask for the same wall clock time (generally, the maximum allowed by a queue) even for jobs with very different run times.

We perform grouping of the jobs coming from the *same* user who have "similar" job resource-configuration. For two jobs from a user to belong to the same group, they have to have the same number of nodes and run times within 10% of their mean run time. This helps classify jobs per user based on their



(a) Aggregate Core Hours vs. Aggregate Wait Time **(b)** Aggregate Number of Jobs vs. Aggregate Wait Time **(c)** Average Run Time vs. Average Wait Time **(d)** Average Number of Nodes vs. Average Wait Time **(e)** Average Core Hours vs. Average Wait Time

Fig. 22. The figures show the wait times of jobs are related to various user statistics for the top 20% core hours consuming users. The aggregate wait time is the sum of wait times of all jobs submitted by the user, while the average wait time is the average wait time of all the jobs. Wait times are not strongly correlated with any of the characteristics for top 20% of the users.

resource usage characteristics. *Our first interesting finding is that each user has only 24 such job-groups on average, even though each user runs 185 jobs on average. Each job group refers to a unique job resource configuration (pair of <number of nodes, run time>).* This finding discovers that jobs executed by a user often follow similar resource usage patterns.

Next, we proceed to investigate how dominant these job-groups are in terms of the core hours consumed. *For example, are a few job-groups from each user consuming most of the core hours delivered by the system? If so, this has important implications for managing and operating the system. For example, one could focus on these few job-groups, which are the most dominant (each job group has a unique job resource configuration), from each user.*

We discover that only considering the top few job-groups per user can account for a large majority of core hours consumed on the system (Fig. 21). The "top" groups are defined as groups that consume the largest core hours for that user. In 2009, considering the top one group per user accounted for 30% of the nodes hours, and considering the top ten groups accounted for 80% of the core hours. But, this trend has only become stronger over the years. In 2018, only considering the top one job-group per user accounted for 60% of total core hours, and considering the top ten groups per user accounted for 98% of total core hours. Our finding can enable new practices in how future HPC systems will provision and manage resources – by focusing on top job configurations from each user.

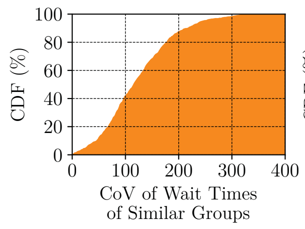
Next, we relax our analysis to focus only on unique job-groups without tagging them with user IDs. We found that in both 2009 and 2018, considering only the top 1% of all job-groups overall (without imposing the condition that each user has at least one group in consideration) can account for 90% of all core hours consumed. This shows that different users tend to execute jobs with similar resource usage patterns, and these jobs account for most of the compute core hours. However, such efforts might not be coordinated or intentional. However, this finding creates opportunities for friendly co-location of jobs with similar resource usage from different users.

Summary. *Our user-level analysis reveals that users exhibit highly repetitive patterns – their jobs can be categorized in a small number of groups where jobs within each group run on the same number of nodes and execute for a similar amount of time. A few groups per user account for a large fraction of the total number of core hours. Additionally, our analysis uncovers that different users tend to execute jobs with similar resource usage patterns, and these jobs account for most of the compute core hours. Future job scheduling practices can leverage such analysis to perform better proactive job scheduling, co-location of jobs from the same and different users, and resource allocation and management.*

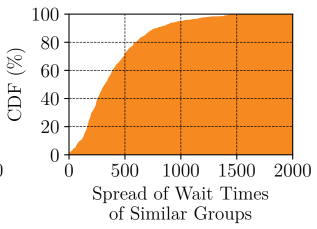
Next, we study the fairness in wait time among users in terms of resource usage, an emerging concern in HPC systems [9, 34]. As our analysis revealed earlier (Sec. V), wait times have become an increasing concern over time and is correlated with job characteristics such as job run time and job size. We explore the same path by investigating the wait time characteristics from user-level characteristics. *In particular, we ask: "are wait times fairly distributed w.r.t users and their respective resource consumption?"* For example, it is reasonable to expect that users running jobs with a high number of cores per job are likely to experience higher wait times. But is it actually the case?

We follow a systematic methodology to answer such questions. We study the correlation between wait times and different user-level attributes (number of jobs, core hours, run time, and job size). We categorize the results into two classes: aggregate correlations and average correlations. Total wait time is correlated with users' aggregate behavior (the number of core hours and jobs). Average wait time per job is correlated with users' average job characteristics (run time per job, job size, core hours per job).

Table II summarizes the correlation between different users' characteristics and their wait times on Mira supercomputer in 2018. We make this comparison for all users, which were active in 2018 and the top 20% core hours consuming users (top 20% of the users consume 80% of the total system core hours (Fig. 20)).



(a) CoV of Wait Times



(b) Spread of Wait Times

Fig. 23. CDF of CoV (standard deviation as percent of mean) and spread (difference between maximum and minimum as percent of mean) of wait times of similar groups across users. The wait time varies for similar groups for different users.

TABLE II. Correlation between different user-level metrics and the user's aggregate wait time and average wait time per job.

		All Users	Top 20% Users
Statistic 1	Statistic 2	Spearman Correlation	
Agg. Wait Time	Agg. Core Hours	0.91	0.52
Agg. Wait Time	Agg. Num. Jobs	0.81	0.63
Avg. Wait Time	Avg. Run Time	0.71	0.56
Avg. Wait Time	Avg. Num. Nodes	0.61	0.43
Avg. Wait Time	Avg. Core Hours	0.77	0.60

These results show that user-level characteristics such as aggregate core hours, number of jobs, job size, run time are highly correlated with the wait time. The correlation coefficient is often higher than 0.6. This matches our expectations. However, these correlation coefficients drop significantly when we only focus on the top 20% of the users. This does not align with intuition and indicates that some users might be waiting for longer than their proportional usages. To explain this trend in more detail, we plot how the wait times compare to different statistics of individual users in the top 20% bracket. Fig. 22 shows weak correlation trend visually. *One may hypothesize that some users who use the system heavily might have been given a welcome break in wait time by increasing the priority of their jobs. However, Fig. 22 shows many cases where even users with very high core hours consumption end up experiencing higher wait times than others with similar aggregate resource consumption and average job behavior.*

To investigate these trends further, we compare the wait times of different users who fall in the same job-group as defined earlier. Recall that for each user, jobs with the same number of nodes and similar run times (within 10% of their mean run time) are grouped in order to study characteristics of jobs with similar resource consumption. We now take these job-groups from individual users and compare their wait times to "similar" job-groups across users. Two groups are considered similar if their jobs have the same number of nodes, and their average run times (mean run time of all jobs belonging to the group) are within 10% of each other. We expect that similar groups from different users should experience similar wait times as the job configurations, in terms of the resources allocated, are the same. However, we discover that the results suggest otherwise. Fig. 23 shows the CDFs of the coefficient of variation (standard deviation as the percentage of mean) and spread (the difference between maximum and minimum as a percent of the mean) of wait times of similar groups across users. A high coefficient of variation and spread indicates that users observe a high variation in wait times despite being in a similar group. Ideally, we would like them to be near zero. But, we find that the mean coefficient of variation (CoV) is 125%, and the mean spread is 408%. Only 5% of similar groups have CoV of wait times less than 50%, and over 90% of similar groups have spread of wait times more than 100%, which is very considerably large.

These results show the high variability in wait times of jobs coming from different users but consuming similar resources. While it is reasonable to anticipate that the timing of job arrival may influence and induce some variability, the degree of the variability is alarming. While the system does not inherently implement unfair policies, our analysis reveals that it may suffer from "unintentional fairness" and hence, large-scale HPC systems can track and embed fairness in their resource management strategies to avoid such side-effects.

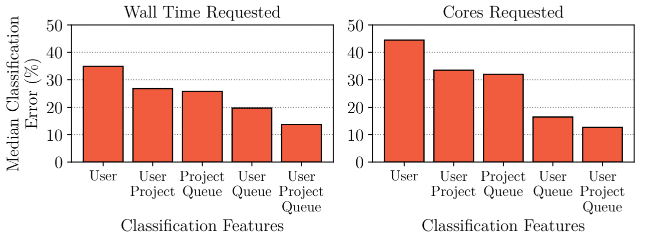


Fig. 24. A job's wall time and number of cores requested can be accurately estimated by developing a classifier using only features available before the job's execution (submission).

Summary. *Our study uncovers an interesting and new insight into the relationship between users' resource consumption and their own wait times. We find that the correlation between resource consumption and user job characteristics is weak, especially for users who are top consumers of the resources. Some users might experience "unintentional unfairness" in terms of the higher wait times that they observe. This implication of this finding is to begin tracking and embedding fairness into HPC resource management strategies.*

Now that we have established that user jobs can be grouped together in terms of their resource consumption, it would be helpful *if this property can be leveraged to predict a user's job characteristics (namely, its requested wall time and the requested number of cores) before it is submitted. This would allow the job scheduler to perform more optimal resource management by anticipating job characteristics and requirements and system constraints.*

To perform such a prediction, we looked at different job features that are known before a job is executed. These include the user submitting the job, the project associated with the job (i.e., the scientific project which the job serves and whose allocation budget the job will consume resources from), and the queue on which the job is submitted. Because there are only four primary queues, a user generally tends to submit to the same queue, and the queue can be used as a classification feature. Then, we classified different jobs on Mira using different combinations of these three features. Jobs with the same features were classified into the same cluster. For example, if the two features used are user Id and project Id, all jobs with the same user Id and project Id are classified as belonging to the same cluster of jobs.

We then calculate the prediction classification error for each job's resource requests relative to its cluster's mean. For instance, if we are trying to predict the wall time that will be requested by a job, we look at a group which has the same user Id and project Id as the job (continuing from the above example), and we predict that the job will have a wall time which is the mean of all previously run jobs in that group. The classification error is then calculated as the absolute difference between the actual wall time and the group mean as a percentage of the group mean.

Fig. 24 shows the median classification error for requested wall times and the number of cores using different features. Right away we observe that when all three of the available features (user, project, and queue id) are used, the error is minimal (less than 15% in both cases), which suggests that job characteristics can be predicted with high accuracy, especially when compared to the 50% median overestimation of run time by the users when requesting wall time (Sec. V). In fact, owing to the error of 15%, the job scheduler can simply predict that the job will consume 15% more resources than the prediction and *thus, accurately predict the requirement of most jobs before the job is submitted.*

We also observe that the classification error is small when only the user and queue Id are used for classification. In fact, when only the project and queue Id are used, the classification error is even better than using user and queue Id as features. This suggests that projects have a large impact on job characteristics and that jobs belonging to the same project, even when submitted by different users, have similar characteristics in terms of their resource demand. Using just user Id as a classification feature has the worst performance, suggesting that queue Id is an important feature.

Summary. *Readily-available user characteristics such as user Id, project Id, and queue Id can be used to estimate the requested wall time and the number of requested cores of a job before its submission with a median estimation error of less than 15%. This estimation can enable the job scheduler to perform more optimal resource management by anticipating job characteristics and requirements, and scheduling these jobs according to temporal system constraints.*

VII. Discussion

Challenges and additional benefits of collecting more fine-grained data. Performing application-specific hardware instrumentation, including CPU utilization, IPC, memory bandwidth, cache usage, etc., would make the analysis richer. However, the overhead of such instrumentation makes it less attractive for production HPC systems. These hardware measurements have to be instrumented, collected, and processed frequently (e.g., every few hundred milliseconds) to be able to extract meaningful information (e.g., maximum memory bandwidth, rate of change of CPU utilization, etc.), which would interfere with the application execution. However, such an approach can be adopted with sparse sampling or by limiting the number of collected metrics. On the other hand, collecting the application name and input parameters can be readily more useful with near-zero overhead. However, it is very challenging to perform accurate analysis of such available information, since application name and input parameters can be easily reused, even across applications with different execution characteristics.

Changes in the INCITE/ALCC awards. Changes in the INCITE/ALCC awards can affect job characteristics. These changes are implicitly reflected in our findings and temporal trends, but cannot be easily examined without explicitly performing controlled characterization of specific codes.

Effect of an opportunistic queue such as backfill. In general, backfilling increases utilization and decreases wait time, but it can result in unfairness if misused. On our systems, jobs running in backfill queue account for less than 7% of all the core hours consumed. This is because the queue is only available to projects which have surpassed their allocations. This also means that jobs submitted to backfill resemble jobs submitted to other queues as the same projects that submit jobs to other queues also submit jobs to backfill when they run out of allocated resource units.

Hero-sized jobs. In our system, hero-sized jobs (defined as jobs larger than 32k nodes) constitute less than 1% of all jobs and consume approximately 10% of all node hours consumed during the period of our study. In our analysis, we found that the characteristics that these jobs exhibit are similar to the characteristics that all other jobs exhibit in terms of the increase in run times over the years, increase in wait times over the years, and even poor estimation of wall times. For example, the median run time of hero jobs is about 0.4 hours, while the median requested wall time of hero jobs is about one hour, which is an over-estimation by over a factor of two. Therefore, hero-sized jobs were neither removed nor presented separately in our analysis.

VIII. Scope and Threats to Validity

While this study sheds new light on leadership-class HPC job characteristics and their evolution, we recognize and acknowledge that our findings are specific to the systems we are studying and are influenced by the workloads that are run on these systems. We hope that this study motivates other HPC centers to perform similar studies and share findings. This will help us improve the state of practice of all HPC centers collectively. We also recognize other threats to the validity of our findings, including (1) impact of I/O and interconnect congestion on resource utilization, (2) memory utilization, and (3) program phase-based performance analysis and MPI communication patterns. Similarly, the impact of machine-learning and statistical tools for fault-tolerance, I/O contention mitigation and power-efficiency on job characteristics can not be isolated without controlled experiments or heavy instrumentation [15, 21, 22, 27, 31, 33–35, 48, 59].Unfortunately, capturing most of these factors requires high overhead instrumentation not suitable for production systems. Nevertheless, we have attempted to carefully scope our analysis findings concerning what has been measured.

IX. Related Work

Prior works have studied various system, user and job characteristics of HPC systems [4–7, 25, 36, 40, 41, 45, 46, 49, 56]. On the system side, Simakov et al. [45] have shown that Texas Advanced Computing Center (TACC) systems have high utilization with 23% of the projects utilizing 73% of the system. This is similar to our findings. Alvarez et al. [5] have studied diversity in resource allocation configurations of jobs and how it affects their wait times. We have shown that even when jobs have similar resource allocation configurations, they can still experience dramatically different wait times. Allcock et al. [4] have analyzed some resource allocation characteristics of Mira from 2013 until early 2017, such as system and queue utilization, but have not provided a detailed analysis of long-term (decade-long) trends.

On the user side, Wolter et al. [56] have shown that different HPC users submit separate sets of jobs with different sizes and runtimes. They find that while wait times are different for these jobs of various sizes, large jobs generally receive higher priority. However, we show that larger jobs usually have had much longer wait times from 2009 to 2018. Schlagkamp et al. [40] provide analysis based on an extensive user survey about how knowledgeable users are about scheduling policies. The work finds that most users are not aware of how the choice of requested resources (number of nodes and wall time) and queue partition affect the wait time. This is in line with our findings, which suggest that users request more wall time than their jobs require without being aware that this could potentially increase the wait time of the jobs.

On the job side, Rodrigo et al. [36] have examined characteristics of jobs submitted on three National Energy Research Scientific Computing Center (NERSC) systems (Carver, Edison, and Hopper) from 2010 to 2014. The work draws some high-level conclusions about characteristics such as job runtime, job wait time, and job core hours. Some of these conclusions, such as wait times being long, match our findings. However, other findings, such as most jobs run on ten nodes or fewer and inter-arrival times are less than two minutes for 90% of the jobs, are specific to the NSERC clusters and do not match our findings. Similarly, Simakov et al. [45] also find that the average job size has reduced. However, this is in contrast to ALCF, where we find that jobs have become longer and larger, and inter-arrival times have increased over the last decade.

Lastly, QBETS [32], published in 2007, aimed to predict queue times of jobs to provide users with an estimate of job completion times. QBETS notes that in many cases users experience much longer wait times than the run times of their jobs. We show that the trend continues and only got worse from 2009 to 2018 as increased wait times for jobs. QBETS categorizes jobs based on queue name, job size, and run time to predict queue wait times for different jobs for a given user. In contrast, we observed that on Intrepid and Mira, the wait times spanned a wide range even when they were grouped based on user, job size, and requested wall time.

X. Conclusion

In this study, we performed a detailed analysis of job characteristics and their resource consumption trends on two leadership-class systems: Intrepid and Mira. While our research confirms several long-held conventional wisdom and expectations that continue to be true, it discovers many new and sometimes surprising insights:

- Utilization of HPC systems has only increased over the past decade, inching closer to full utilization at all times.
- HPC jobs have become larger and longer, with a median increase of 18× in the number of core hours per job. Long-running and medium-sized jobs consume the most number of core hours delivered.
- Wait times of HPC jobs have become much worse over the years; yet, users continue to over-estimate the run times of jobs and request long wall times.
- An HPC user submits jobs with very similar characteristics requesting similar resources. But, users with similar resource consumption might experience different wait times for their jobs.
- User characteristics such as user Id, project Id, and queue Id can be used to predict the requested wall time and the number of requested cores of a job before its submission.

Our analysis about how HPC job characteristics have evolved over the last decade at Argonne National Laboratory can be used to drive impreovement in HPC resource management.

Acknowledgment: We are thankful to anonymous reviewers to provided thoughtful feedback despite the pandemic challenges. We are thankful for the support from Northeastern University and the Massachusetts Green High Performance Computing Center (MGHPCC) facility. The work is also supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research and Basic Energy Sciences, under Contract DE-AC02-06CH11357. We gratefully acknowledge the computing resources provided by the Argonne Leadership Computing Facility, which is a U.S. Department of Energy, Office of Science User Facility.

References

[1] Job Scheduling Policy for BG/Q Systems. *Argonne Leadership Computing Facility*, 2019. https://www.alcf.anl.gov/user-guides/job-scheduling-policy-bgq-systems.

[2] Machine Partitions on BG/Q. *Argonne Leadership Computing Facility*, 2019. https://www.alcf.anl.gov/user-guides/machine-partitions-bgq.

[3] Anthony Agelastos, Benjamin Allan, Jim Brandt, Ann Gentile, Sophia Lefantzi, Steve Monk, Jeff Ogden, Mahesh Rajan, and Joel Stevenson. Continuous Whole-System Monitoring Toward Rapid Understanding of Production HPC Applications and Systems. *Parallel Computing*, 58:90–106, 2016.

[4] William Allcock, Paul Rich, Yuping Fan, and Zhiling Lan. Experience and Practice of Batch Scheduling on Leadership Supercomputers at Argonne. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 1–24. Springer, 2017.

[5] Gonzalo Pedro Rodrigo Alvarez, Per-Olov Östberg, Erik Elmroth, Katie Antypas, Richard Gerber, and Lavanya Ramakrishnan. Towards Understanding Job Heterogeneity in HPC: A NERSC Case Study. In *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 521–526. IEEE, 2016.

[6] George Amvrosiadis, Jun Woo Park, Gregory R Ganger, Garth A Gibson, Elisabeth Baseman, and Nathan DeBardeleben. On the Diversity of Cluster Workloads and its Impact on Research Results. In *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, pages 533–546, 2018.

[7] Norbert Attig, Paul Gibbon, and Th Lippert. Trends in Supercomputing: The European Path to Exascale. *Computer Physics Communications*, 182(9):2041–2046, 2011.

[8] Guillaume Aupy, Ana Gainaru, Valentin Honoré, Padma Raghavan, Yves Robert, and Hongyang Sun. Reservation strategies for stochastic jobs. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 166–175. IEEE, 2019.

[9] Frédéric Azevedo, Dalibor Klusáček, and Frédéric Suter. Improving fairness in a large scale htc system through workload analysis and simulation. In *European Conference on Parallel Processing*, pages 129–141. Springer, 2019.

[10] Leonardo Bautista-Gomez, Ana Gainaru, Swann Perarnau, Devesh Tiwari, Saurabh Gupta, Christian Engelmann, Franck Cappello, and Marc Snir. Reducing waste in extreme scale systems through introspective analysis. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 212–221. IEEE, 2016.

[11] Susan Coghlan, Kalyan Kumaran, Raymond M Loy, Paul Messina, V Morozov, James C Osborn, Scott Parker, KM Riley, Nichols A Romero, and Timothy J Williams. Argonne Applications for the IBM Blue Gene/Q, Mira. *IBM Journal of Research and Development*, 57(1/2):12–1, 2013.

[12] Jim Collins. Passing the Torch from Intrepid to Mira, Feb 2014.

[13] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 153–167, 2017.

[14] David L Darrington, Matthew W Markland, Philip James Sanders, and Richard Michael Shok. Scheduling Work in a Multi-Node Computer System based on Checkpoint Characteristics, August 16 2016. US Patent 9,417,909.

[15] Anwesha Das, Frank Mueller, Charles Siegel, and Abhinav Vishnu. Desh: deep learning for system health prediction of lead times to failure in hpc. In *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, pages 40–51, 2018.

[16] James Elliott, Kishor Kharbas, David Fiala, Frank Mueller, Kurt Ferreira, and Christian Engelmann. Combining Partial Redundancy and Checkpointing for HPC. In *2012 IEEE 32nd International Conference on Distributed Computing Systems*, pages 615–626. IEEE, 2012.

[17] Yuping Fan, Paul Rich, William E Allcock, Michael E Papka, and Zhiling Lan. Trade-off Between Prediction Accuracy and Underestimation Rate in Job Runtime Estimates. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 530–540. IEEE, 2017.

[18] Jinghua Feng, Guangming Liu, Jian Zhang, Zhiwei Zhang, Jie Yu, and Zhaoning Zhang. Workload characterization and evolutionary analyses of tianhe-1a supercomputer. In *International Conference on Computational Science*, pages 578–585. Springer, 2018.

[19] Neha Gholkar et al. On the Management of Power Constraints for High Performance Systems. 2018.

[20] Jing Guo, Zihao Chang, Sa Wang, Haiyang Ding, Yihui Feng, Liang Mao, and Yungang Bao. Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces. In *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2019.

[21] Luanzheng Guo, Dong Li, and Ignacio Laguna. Paris: Predicting application resilience using machine learning. *arXiv preprint arXiv:1812.02944*, 2018.

[22] Saurabh Gupta, Devesh Tiwari, Christopher Jantzi, James Rogers, and Don Maxwell. Understanding and exploiting spatial properties of system failures on extreme-scale hpc systems. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 37–44. IEEE, 2015.

[23] Nikhil Jain, Abhinav Bhatele, Xiang Ni, Todd Gamblin, and Laxmikant V Kale. Partitioning Low-Diameter Networks to Eliminate Inter-Job Interference. In *2017*

IEEE International Parallel and Distributed Processing Symposium (IPDPS), pages 439–448. IEEE, 2017.

[24] Congfeng Jiang, Guangjie Han, Jiangbin Lin, Gangyong Jia, Weisong Shi, and Jian Wan. Characteristics of co-allocated online services and batch jobs in internet data centers: a case study from alibaba cloud. *IEEE Access*, 7:22495–22508, 2019.

[25] Wayne Joubert and Shi-Quan Su. An Analysis of Computational Workloads for the ORNL Jaguar System. In *Proceedings of the 26th ACM international conference on Supercomputing*, pages 247–256. ACM, 2012.

[26] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S Gunawi, Cody Hammock, et al. Lessons learned from the chameleon testbed. In *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pages 219–233, 2020.

[27] Feng Liu and Jon B Weissman. Elastic job bundling: An adaptive resource request strategy for large-scale parallel applications. In *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12. IEEE, 2015.

[28] Andre Merzky, Mark Santcroos, Matteo Turilli, and Shantenu Jha. Executing dynamic and heterogeneous workloads on super computers, 2016.

[29] Bin Nie, Devesh Tiwari, Saurabh Gupta, Evgenia Smirni, and James H Rogers. A large-scale study of soft-errors on gpus in the field. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 519–530. IEEE, 2016.

[30] Bin Nie, Ji Xue, Saurabh Gupta, Christian Engelmann, Evgenia Smirni, and Devesh Tiwari. Characterizing temperature, power, and soft-error behaviors in data center systems: Insights, challenges, and opportunities. In *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 22–31. IEEE, 2017.

[31] Bin Nie, Ji Xue, Saurabh Gupta, Tirthak Patel, Christian Engelmann, Evgenia Smirni, and Devesh Tiwari. Machine learning models for gpu error prediction in a large scale hpc system. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 95–106. IEEE, 2018.

[32] Daniel Nurmi, John Brevik, and Rich Wolski. QBETS: Queue Bounds Estimation from Time Series. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 76–101. Springer, 2007.

[33] Tirthak Patel, Suren Byna, Glenn K Lockwood, and Devesh Tiwari. Revisiting i/o behavior in large-scale storage systems: the expected and the unexpected. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–13, 2019.

[34] Tirthak Patel and Devesh Tiwari. Perq: Fair and efficient power management of power-constrained large-scale computing systems. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pages 171–182, 2019.

[35] Tirthak Patel, Adam Wagenhäuser, Christopher Eibel, Timo Hönig, Thomas Zeiser, and Devesh Tiwari. What does power consumption behavior of hpc jobs reveal?: Demystifying, quantifying, and predicting power consumption characteristics. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 799–809. IEEE, 2020.

[36] Gonzalo P Rodrigo, P-O Östberg, Erik Elmroth, Katie Antypas, Richard Gerber, and Lavanya Ramakrishnan. Towards understanding hpc users and systems: a nersc case study. *Journal of Parallel and Distributed Computing*, 111:206–221, 2018.

[37] Gonzalo Pedro Rodrigo Álvarez, Per-Olov Östberg, Erik Elmroth, Katie Antypas, Richard Gerber, and Lavanya Ramakrishnan. Hpc system lifetime story: Workload characterization and evolutionary analyses on nersc systems. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, pages 57–60, 2015.

[38] Krzysztof Rzadca, Pawel Findeisen, Jacek Swiderski, Przemyslaw Zych, Przemyslaw Broniek, Jarek Kusmierek, Pawel Nowak, Beata Strack, Piotr Witusowski, Steven Hand, et al. Autopilot: workload autoscaling at google. In *Proceedings of the Fifteenth European Conference on Computer Systems*, pages 1–16, 2020.

[39] Stephan Schlagkamp, Rafael Ferreira Da Silva, Ewa Deelman, and Uwe Schwiegelshohn. Understanding user behavior: from hpc to htc. *Procedia Computer Science*, 80:2241–2245, 2016.

[40] Stephan Schlagkamp, Rafael Ferreira da Silva, Johanna Renker, and Gerhard Rinkenauer. Analyzing Users in Parallel Computing: A User-Oriented Study. In *2016 International Conference on High Performance Computing & Simulation (HPCS)*, pages 395–402. IEEE, 2016.

[41] Stephan Schlagkamp, Rafael Ferreira da Silva, William Allcock, Ewa Deelman, and Uwe Schwiegelshohn. Consecutive Job Submission Behavior at Mira Supercomputer. In *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*, pages 93–96, 2016.

[42] Thomas Scogland, Jonathan Azose, David Rohr, Suzanne Rivoire, Natalie Bates, and Daniel Hackenberg. Node variability in large-scale power measurements: perspectives from the green500, top500 and eehpcwg. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11, 2015.

[43] Siqi Shen, Vincent van Beek, and Alexandru Iosup. Statistical characterization of business-critical workloads hosted in cloud datacenters. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 465–474. IEEE, 2015.

[44] Pavel Shvets, Vadim Voevodin, and Dmitry Nikitenko.

Approach to workload analysis of large hpc centers. In *International Conference on Parallel Computational Technologies*, pages 16–30. Springer, 2020.

[45] Nikolay A Simakov, Joseph P White, Robert L DeLeon, Steven M Gallo, Matthew D Jones, Jeffrey T Palmer, Benjamin Plessinger, and Thomas R Furlani. A Workload Analysis of NSF's Innovative HPC Resources Using XDMoD. *arXiv preprint arXiv:1801.04306*, 2018.

[46] Vladimir Vladimirovich Stegailov and Henri Edgarovich Norman. Challenges to the Supercomputer Development in Russia: a HPC User Perspective. *Program Systems: Theory and Applications*, 5(1):111–152, 2014.

[47] Kun Tang, Devesh Tiwari, Saurabh Gupta, Ping Huang, Qiqi Lu, Christian Engelmann, and Xubin He. Power-capping aware checkpointing: On the interplay among power-capping, temperature, reliability, performance, and energy. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 311–322. IEEE, 2016.

[48] Devesh Tiwari, Saurabh Gupta, and Sudharshan S Vazhkudai. Lazy checkpointing: Exploiting temporal locality in failures to mitigate checkpointing overheads on extreme-scale systems. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 25–36. IEEE, 2014.

[49] Ramses van Zon, Marcelo Ponce, Erik Spence, and Daniel Gruner. Trends in Demand, Growth, and Breadth in Scientific Computing Training Delivered by a High-Performance Computing Center. *arXiv preprint arXiv:1901.05520*, 2019.

[50] Laurens Versluis, Roland Mathá, Sacheendra Talluri, Tim Hegeman, Radu Prodan, Ewa Deelman, and Alexandru Iosup. The workflow trace archive: Open-access data from public and private computing infrastructures. *IEEE Transactions on Parallel and Distributed Systems*, 31(9):2170–2184, 2020.

[51] Chao Wang, Frank Mueller, Christian Engelmann, and Stephen L Scott. A Job Pause Service under LAM/MPI+ BLCR for Transparent Fault Tolerance. In *2007 IEEE International Parallel and Distributed Processing Symposium*, pages 1–10. IEEE, 2007.

[52] John Wilkes. More Google cluster data. Google research blog, November 2011. http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.

[53] John Wilkes. Google cluster-usage traces v3. Technical report, Google Inc., Mountain View, CA, USA, April 2020. https://github.com/google/cluster-data/blob/master/ClusterData2019.md.

[54] John Wilkes. Yet more Google compute cluster trace data. Google research blog, April 2020. https://ai.googleblog.com/2020/04/yet-more-google-compute-cluster-trace.html.

[55] Timothy J Williams. Preparing Applications for Mira, a 10 PetaFLOPS IBM Blue Gene/Q System, Nov 2011.

[56] Nicole Wolter, Michael O McCracken, Allan Snavely, Lorin Hochstein, Taiga Nakamura, and Victor Basili. What's Working in HPC: Investigating HPC User Behavior and Productivity. *CTWatch Quarterly*, 2(4A):9–17, 2006.

[57] Michael R Wyatt II, Stephen Herbein, Todd Gamblin, Adam Moody, Dong H Ahn, and Michela Taufer. PRIONN: Predicting Runtime and IO using Neural Networks. In *Proceedings of the 47th International Conference on Parallel Processing*, page 46. ACM, 2018.

[58] Anas A Youssef and Diwakar Krishnamurthy. Burstiness-Aware Service Level Planning for Enterprise Application Clouds. *Journal of Cloud Computing*, 6(1):17, 2017.

[59] Di Zhang, Dong Dai, Youbiao He, and Forrest Sheng Bao. Rlscheduler: Learn to schedule hpc batch jobs using deep reinforcement learning. *arXiv preprint arXiv:1910.08925*, 2019.

[60] Sheng Zhang, Zhuzhong Qian, Zhaoyi Luo, Jie Wu, and Sanglu Lu. Burstiness-Aware Resource Reservation for Server Consolidation in Computing Clouds. *IEEE Transactions on Parallel and Distributed Systems*, 27(4):964–977, 2016.

[61] Xinqian Zhang, Tingming Wu, Mingsong Chen, Tongquan Wei, Junlong Zhou, Shiyan Hu, and Rajkumar Buyya. Energy-Aware Virtual Machine Allocation for Cloud with Resource Reservation. *Journal of Systems and Software*, 147:147–161, 2019.

Appendix: Artifact Description/Artifact Evaluation

SUMMARY OF THE EXPERIMENTS REPORTED

Analysis Overview

The analysis is performed on resource consumption logs from the ALCF supercomputer Intrepid and Mira over the past decade. The analysis dataset contains entries for each job submitted to the system; each entry includes information such as anonymized user id, job id, queue partition, queue time, start time, end time, number of nodes, wall time requested, actual run time, etc. Note that in the interest of anonymity, the analysis is not performed on application id or name, nor the configurations and/or input parameters of the applications. Information about jobs' CPU utilization, memory usage, and network behavior is also not measured at per job granularity. We use the Intrepid datasets from 2009 to 2013, and Mira datasets from 2014 to 2018. Note that while Intrepid was active in 2008 and Mira was active in 2013, both did not enter full production until 2009 and 2014, respectively. Therefore, we do not use the Intrepid dataset from 2008 and the Mira dataset from 2013. The analysis of the dataset was performed on MacOS Catalina 10.15 (Darwin Kernel Version 19.6.0 x86_64) using Python 3.7.4 (libraries: numpy 1.18.4, scipy 1.1.4 (dependency: numpy >= 1.13.3), csv, json, time, and datetime). Refer to Section 2 for additional methodological details.

Testbed System Description

Intrepid was a 557 TFlops IBM BlueGene/P system that entered full production in February 2009. It consisted of 40,960 850 MHz IBM Power PC 450 nodes, each with4 cores, and 2 GB memory. Each node on Intrepid was connected to separate networks for I/O and inter-node communication. The inter-node communication interconnect was set up with a 3D Torus configuration. The system was decommissioned in December 2013, and replaced by its successor, the Mira system.

Mira entered production in April 2013; however, it began handling full ALCF workload in January 2014 (after Intrepid was decommissioned). Mira was a 10 PFlops IBM Blue Gene/Q system, providing about 20x performance of Intrepid. Mira consists of 49,1521.6 GHz IBM PowerPC A2 nodes, each with 16 cores and 16 GB memory. The nodes on Mira are connected using 5D Torus topology.

Note that the system software environment on Intrepid and Mira were periodically updated during their operational production period, however, the details of those updates are not required to perform the operations conducted in this study.

Artifact Description

(1) **Dataset:** The job trace dataset is hierarchically categorized according to the two systems and the years of trace collection. The reported parameters include:
- User ID
- Project ID
- Queue name
- Number of nodes requested
- Number of cores requested
- Wall time requested
- Job queued timestamp
- Job start timestamp
- Job end timestamp
- Queue time
- Run time
- Node seconds used
- Core seconds used

(2) **Analysis Codes:** Tools to parse and analyze the dataset:
- **system_utilizasystem_utilization_over_time.py:** Utilization for different years (Figure 1 in the paper).
- **median_statistics_per_year.py:** Yearly median trends (Figure 2, 3, 4, 6, 7, and 11).
- **job_size_frequencies_over_the_years.py:** Empirical PDF of job sizes (Figure 5).
- **job_run_times_frequencies_over_the_years.py:** Empirical PDF of job run times (Figure 8).
- **total_core_hours_by_job_size.py:** Total core hours consumed by jobs of different sizes (Figure 9).
- **median_run_times_by_job_size.py:** Median run times of jobs of different sizes (Figure 10).
- **job_wall_times_frequencies_over_the_years.py:** Empirical PDF of requested job wall times (Figure 12).
- **job_wait_times_frequencies_over_the_years.py:** Empirical PDF of job wait times (Figure 13).
- **queue_based_analysis.py:** Characteristics (run time, wait time, wall time, job size, and core hours) of jobs submitted to different queues (Figure 14, 15, 16, and 17).
- **temporal_analysis.py:** Wait time during different hours of the day and days of the week (Figure 18 and 19).
- **core_hours_by_user.py:** Number of jobs and core hours by top users (Figure 20).
- **app_grouping_per_user.py:** Heirarchical clustering-based grouping of applications submitted by different users (Figure 21, 22, and 23).

- **classification_of_jobs_by_user_project_queue.py:** Job classification using grouping for prediction of wall time and number of cores (Figure 24).

Key Machine-Generated Environment Features

- **machdep.cpu.vendor:** GenuineIntel
- **machdep.cpu.brand_string:** Intel(R) Core(TM) i5-8210Y CPU @ 1.60GHz
- **machdep.cpu.family:** 6
- **machdep.cpu.model:** 142
- **machdep.cpu.extmodel:** 8
- **machdep.cpu.extfamily:** 0
- **machdep.cpu.features:** FPU VME DE PSE TSC MSR PAE MCE CX8 APIC SEP MTRR PGE MCA CMOV PAT PSE36 CLFSH DS ACPI MMX FXSR SSE SSE2 SS HTT TM PBE SSE3 PCLMULQDQ DTES64 MON DSCPL VMX EST TM2 SSSE3 FMA CX16 TPR PDCM SSE4.1 SSE4.2 x2APIC MOVBE POPCNT AES PCID XSAVE OSXSAVE SEGLIM64 TSCTMR AVX1.0 RDRAND F16C
- **machdep.cpu.extfeatures:** SYSCALL XD 1GBPAGE EM64T LAHF LZCNT PREFETCHW RDTSCP TSCI

ARTIFACT AVAILABILITY

Software Artifact Availability: All author-created software artifacts are maintained in a public repository under an OSI-approved license.

Hardware Artifact Availability: There are no author-created hardware artifacts.

Data Artifact Availability: All author-created data artifacts are maintained in a public repository under an OSI-approved license.

Proprietary Artifacts: No author-created artifacts are proprietary.

Author-Created or Modified Artifacts:

```
Persistent ID: https://doi.org/10.5281/zenodo.3957897
Artifact name: Job Trace Data and Analysis Tools
```

BASELINE EXPERIMENTAL SETUP, AND MODIFICATIONS MADE FOR THE PAPER

Relevant hardware details: Mira and Intrepid Supercomputers

Operating systems and versions: Analysis: MacOS Catalina 10.15 (Darwin Kernel Version 19.6.0 x86_64)

Compilers and versions: Python 3.7.4

Applications and versions: N/A

Libraries and versions: Analysis: Python 3.7.4 (libraries: numpy 1.18.4, scipy 1.1.4 (dependency: numpy >= 1.13.3), csv, json, time, and datetime)

Key algorithms: Hierarchical Clustering, Group-based Classification

Input datasets and versions: N/A