# Full Lifecycle Data Analysis on a Large-scale and Leadership Supercomputer: What Can We Learn from It?

Bin Yang[*§], Hao Wei[*], Wenhao Zhu[†§], YuHao Zhang[*], Weiguo Liu[†], and Wei Xue[*§]

[*]Tsinghua University
[†]Shandong University
[§]National Supercomputer Center in Wuxi

## Abstract

The system architecture of contemporary supercomputers is growing increasingly intricate with the ongoing evolution of system-wide network and storage technologies, making it challenging for application developers and system administrators to manage and utilize the escalating complexity of supercomputers effectively. Moreover, the limited experience of application developers and system administrators in conducting insightful analyses of diverse High-Performance Computing (HPC) workloads and the resulting array of resource utilization characteristics exacerbate the challenge. To address this issue, we undertake a comprehensive analysis of six years' worth of 40 TB data (comprising I/O performance data and job running information) from Sunway TaihuLight, boasting 41508 nodes and currently ranked as the world's 11th-fastest supercomputer. Our study provides valuable insights into operational management strategies for HPC systems (i.e., job hanging caused by heavy-load benchmark testing, job starvation caused by aggressive scheduling policies) and I/O workload characteristics (i.e., getattr operations spiking caused by massive access to grid files, a large number of files accessed by many applications in a short period), shedding light on both challenges and opportunities for improvements in the HPC environment. This paper delineates our methodology, findings, and the significance of this study. Additionally, we discuss the potential of our research for future studies and practice within this domain.

## 1 Introduction

As high-performance computing (HPC) technology advances rapidly, supercomputers have made tremendous leaps in computational power, ushering in a new era. However, the accompanying complexity in their architecture [21, 54, 83] is undeniable. Moreover, HPC systems are inherently highly shared environments, often accommodating numerous applications running together. HPC systems' complexity, applications' diversity, and workloads' dynamic nature collectively pose substantial challenges for application developers and system administrators to analyze and understand HPC workloads and resource utilization characteristics. This knowledge is pivotal for enhancing parallel system and software design and optimization [50], and the absence of such knowledge not only hinders applications from obtaining optimal performance but also acts as a significant barrier to effectively managing and utilizing supercomputers' precious resources.

To this end, numerous efforts have been made in the past to address the issue from many aspects, such as studying job running behaviors [50, 58], analyzing applications' I/O characteristics [40, 41, 49], performing controlled experiments on large-scale supercomputers [42, 73], and incorporating probes into storage systems to predict system load trends [34, 38]. Nonetheless, a knowledge gap remains due to the continuous development of HPC applications and systems. For example, accurately analyzing I/O data to comprehend the characteristics of applications and systems is still challenging [9, 59]. The above factors make it difficult to utilize supercomputer resources efficiently to serve various HPC applications.

Recently, more efforts have advanced in scalable data collection [4, 55, 77], making it possible to observe the production-run large-scale supercomputers from a multi-source and holistic perspective, which supports identifying the relationships between applications and the back-end storage systems and uncovering the performance bottlenecks and management issues. For example, Shah et al. [61] explored the I/O interference caused by file-access patterns, and Gunawi et al. [24] uncovered the fail-slow factors for distributed storage systems. Nevertheless, a comprehensive and enduring perspective on the operations of large-scale supercomputers and their storage systems remains absence.

To address this challenge, we perform a measurement and systematic analysis of around six years' worth of I/O activity data and job running information from TaihuLight [21], currently the world's No.11 supercomputer (formerly first-ranked from June 2016 to November 2017). The data, collected by the Beacon monitoring system [76] from April 2017 to December 2022, encompasses information from 40,960 compute nodes (over ten million cores), 240 forwarding nodes, 304 storage nodes, 3 metadata nodes, and 1 job scheduling node, amassing around 40 TB data. This extensive data set empowers us to investigate temporal and spatial patterns from both application and system perspectives, facilitating the identification of nuanced relationships between diverse HPC applications' characteristics and system components over time. This work's

[*]Wei Xue is the corresponding author. Email: xuewei@tsinghua.edu.cn

primary contributions are as follows:

- We conduct a comprehensive and detailed analysis of the system and job characteristics on a cutting-edge computing system from the National Supercomputing Center in Wuxi, TaihuLight. To our best knowledge, our in-depth characterization study is the largest and longest of its kind, analyzing nearly 6-year system and job data on the current world's No. 11 supercomputer, including more than 6 million jobs from 1077 users that consume 4 billion compute core hours. We explain our findings from multiple perspectives, such as operation and maintenance, system scheduling, job I/O characteristics, etc.
- Our study uncovers and explains many previously undiscovered and unquantified insights into how system and application characteristics evolve, revealing previously unnoticed problems, such as job hanging caused by heavy-load benchmark testing, job starvation caused by overly aggressive scheduling policies, *getatrr* operations spiking caused by massive access to grid files and Lustre prefetching mechanism, etc. Notably, our study confirms some previous work and conjectures, such as small-scale jobs consuming less than 1% of total core hours and the average job waiting time being usually long.
- We suggest optimizations or improvements based on our experience and experimental testing, such as improving the default job scheduling strategy, incorporating application hang detection into administrators' periodic benchmark testing and conducting it during the low-load period to relieve job hanging issue, employing Symbolic Links instead of deep directory structures to accelerate the index, etc.

## 2 Background

### 2.1 Sunway TaihuLight's architecture

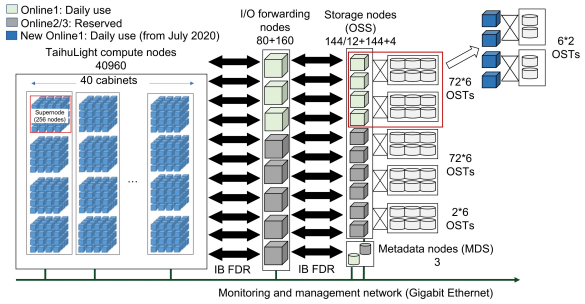



Figure 1: Overview of TaihuLight

To begin, we introduce TaihuLight, currently ranked as No.11 according to the Top 500 list of November 2023, which serves as this study's primary platform for data collection and analysis. Figure 1 illustrates the current architecture of TaihuLight, consisting of three layers: compute nodes, I/O forwarding nodes, and storage nodes The left side of the figure shows 40,960 compute nodes, totaling an impressive 10,649,600 cores. The theoretical peak performance of these nodes can reach up to 125 PFlop/s. Notably, every 256 compute nodes are grouped to form a supernode. Within each supernode, the nodes are fully connected, and they are further linked to a quarter-cropped Fat-tree network via dual-rail FDR InfiniBand (IB) connections to establish communication with other supernodes. Additionally, each supernode maintains a connection to the storage system (Icefish) through an IB line.

The back end of Icefish is a popular shared file system, Lustre [7] Icefish is configured and deployed as three independent and non-overlapping file systems: Online1, Online2, and Online3. Each file system is configured with two MDSs in a master-standby strategy (one MDS is in use, and the other MDS is on standby). Online1, the default file system for regular users, has 144 Lustre Object Storage Servers (OSSs) and 432 Lustre Object Storage Targets (OSTs). However, due to the failure of a RAID disk array in July 2020, the administrator upgraded Online1 by replacing the original Sungon disk arrays with the newly purchased DDN disk arrays. The new disk array reconfigured a new Lustre file system with 12 OSSs and 12 OSTs, which we refer to as the "New Online1". Online2 and Online3 are the reserved file systems for "VIP" users. There are 144 OSSs and 432 OSTs for Online2 back-end storage and 4 OSSs and 12 OSTs for Online3 back-end storage. The OSSs currently run the Lustre parallel file system version 2.10, 2.5, and 2.12 for Online1, Online2, and Online3, respectively. Moreover, to keep from slowing the overall system performance when creating and opening these files, Icefish are configured to use a 1 MB stripe size and a stripe count of 1, just like other systems [37, 39].

Between the compute node and the storage node is a global-shared layer, consisting of 240 I/O forwarding nodes and is positioned in the middle layer to connect the front-end compute nodes to the back-end storage nodes. Each I/O forwarding node plays a dual role, both as a LWFS server [16] to the compute nodes and client to the Lustre back end, and is responsible for forwarding I/O requests from the compute node to the back-end Luster. Eighty forwarding nodes are used for daily service, while the remaining 160 nodes act as backup systems. The compute nodes are statically mapped to the forwarding nodes in a 512:1 ratio. Note that an I/O forwarding node can provide a bandwidth of 2.5 GB/s, aggregating to 600 GB/s for the entire forwarding layer. However, limited by the back-end service capabilities and static system configuration policies, Icefish has been empirically measured to provide approximately 200 GB/s of aggregated bandwidth for both read and write operations.

### 2.2 Data collection tool

TaihuLight embraces an end-to-end monitoring tool, termed as Beacon [76], to collect multi-layer performance data. Figure 2 shows the current deployments of Beacon (The second

version of Beacon, we also call it *Beacon*$^{+}$) on the Taihu-Light supercomputer and its main components, including the monitoring, storage, and analysis components.

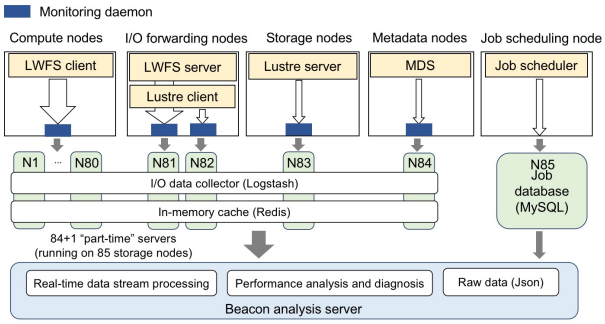



Figure 2: Beacon deployments and its main components

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| LWFS client | Node IP | Timestamp | Operation | File descriptor | Offset | Size | Count |
| LWFS server | Fwd IP | Timestamp | Operation | Latency | Execution time | Count | Queue length |
| Lustre client | Fwd IP | OST ID | Read RPC Count | Write RPC Count | Pages per RPC | Snapshot Time | |
| Lustre server | OST ID | Read RPC Count | Write RPC Count | Pages per RPC | Snapshot Time | | |
| MDS | Client IP | Operation1 Count | Opearation2 Count | ... | OperationN Count | Snapshot Time | |
| Job scheduler | Job ID | Status | Nodelist | Job Name | Start time | End Time | Comd. Path |

Figure 3: Detailed data format collected by Beacon

Beacon performs I/O monitoring at six different parts of TaihuLight, and the detailed data format can be seen in Figure 3. For the LWFS client, data was collected on the compute nodes to reconstruct the fine-grained application I/O behavior. For the forwarding layer, data were collected simultaneously from the LWFS server and the Lustre client to describe the load status of the forwarding layer. For the storage nodes, data was gathered from the Lustre OSTs to describe the application's I/O behavior on the back end. Metadata request information was collected from the metadata nodes to describe the overall metadata load. Finally, job running information was collected from the job scheduling node to analyze the running status of jobs on the HPC system.

Beacon's storage component is primarily deployed on 85 part-time storage nodes, each equipped with a Logstash [67] and a Redis [56] for temporary data buffering. Real-time data analysis is conducted on a dedicated analysis node, delivering instantaneous diagnostic analysis to application developers and system administrators. Simultaneously, the raw data is preserved in JSON [8] format files for permanent storage. Notably, Beacon's overhead is very low, with less than 1% impact on the application and less than 0.1% CPU usage by the collection daemons, so it has been deployed on TaihuLight since April 2017, collecting data for six years.

## 3 Dataset overview

Since its deployment on TaihuLight, Beacon has diligently collected vast amounts of I/O performance data and job running information, amassing approximately 40 TB of data from April 2017 to December 2022. This dataset encompasses 6,256,017 jobs, consuming around 4 billion compute core hours. Among them, 2,793,938 jobs exhibited a parallel scale exceeding 16 processes, consuming over 99% of the total core hours, aligning with prior findings on the NSERC clusters [58]. Consequently, the main focus of this paper is on jobs with a parallel scale greater than 16 processes. Besides, 662,405 jobs featuring non-trivial I/O (I/O volume over 200MB) have been analyzed to reveal HPC applications' I/O workload, constituting 23.7% of jobs exceeding 16 processes. In the following, we analyze the dataset from three perspectives: long-term overall system workload analysis, job running information analysis, and job I/O characteristics analysis.

## 4 Long-term overall workload analysis

### 4.1 Compute node utilization analysis

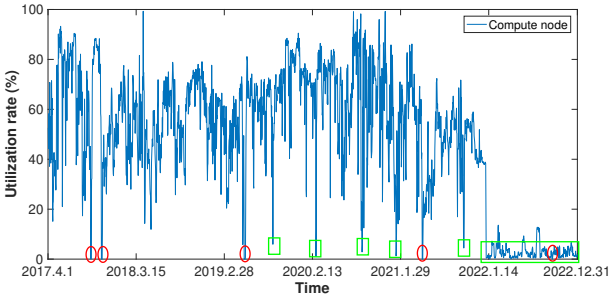


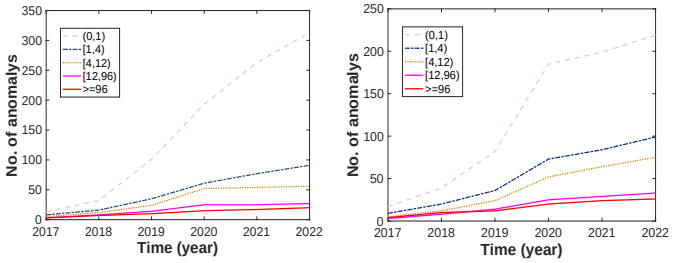

Figure 4: Utilization rate of the compute nodes from April 2017 to December 2022

First, we conduct a six-year statistical analysis of load variations on TaihuLight's compute nodes, with Figure 4 depicting the utilization rates. The utilization rate denotes the actual core-hour consumption of compute nodes within a day relative to the maximum core hours consumed by all compute nodes in that day. Notably, the red ellipses in the figure represent instances of zero utilization, occurring five times, aligning with scheduled downtimes for maintenance lasting from two to four days. Meanwhile, the green boxes highlight periods characterized by exceptionally low utilization.

***Observation* 1.** *Between 2019 and 2021, low-utilization periods exhibited a periodicity, recurring approximately every 6-7 months. In particular, the fifth instance of low utilization in October 2021 also occurred after a similar interval since the fourth scheduled maintenance downtime.*

We further analyze this phenomenon and identify system anomalies as the primary cause. Instances such as back-end system node anomalies or network switch failures can lead to the simultaneous abnormal termination of numerous jobs. Moreover, these anomalies or failures frequently last for tens of hours, resulting in prolonged idle periods for the compute nodes and, thus, a significant drop in utilization. Subsequently, we examine the system anomalies reported by Beacon, with

Figure 5 showing anomalies on TaihuLight's I/O forwarding and storage nodes. Different colors represent the different recovery intervals after the abnormality occurs.




(a) On the I/O forwarding nodes (b) On the storage nodes

Figure 5: Changes in cumulative values of system anomalies identified by Beacon(i.e., the number of anomalies in 2018 includes the number of anomalies in 2017)

The number of anomalies exhibits a clear upward trend, with potential causes including node performance degradation, RAID reconstruction, abnormal shutdowns due to failures, etc. Of course, administrators have invested a considerable effort in system fault tolerance, which is evident in the system's ability to recover from numerous short-term anomalies swiftly, minimizing their impact on overall utilization. However, persistent issues like fail-slow (i.e., nodes with performance degradation) can stay for an extended period, posing challenges for detection. Although these instances might not significantly affect overall utilization, they can degrade the quality of service, adversely impacting application performance. Notably, the increased rate of OST anomalies observed post-2020 has considerably slowed down, mainly due to the replacement of new storage systems, as mentioned in Section 2.1.

**Lessons.** System anomalies pose challenges not only at the user level but also at the system level, impacting overall utilization and quality of service. Our findings also demonstrate prior research that abnormal rates increase with both prolonged run time and larger scale [44, 60]. Notably, current anomaly detection methods, particularly for fail-slow issues [24], heavily rely on administrators utilizing benchmark testing offline, which is expensive and may disrupt running applications. With the growing scale of post-E-class era machines, traditional operational methods, including regular maintenance and benchmark detection, are proving insufficient for administrators. To address this, additional strategies such as online anomaly detection with low overhead, fault prediction, and migration become essential for mitigating the impact of anomalies and enhancing system utilization.

***Observation* 2.** *2022's utilization rate dropped significantly.*

We attribute the observed trend to the emergence of the new-generation Sunway supercomputer, prompting the migration of multiple applications. To delve into this, we analyze the top 10 primary applications spanning from 2017 to 2022, including CESM [31], WRF [52], GKUA [36], FBA [30], Incompact3d [5], GRAPES [15], VASP [25], KRPs [29], WW3 [66], and COAWST [71]. Here, primary applications refer to those utilizing significant compute resources on TaihuLight, constituting approximately 75% of the total. As anticipated, core hours consumed by most primary applications (CESM, WRF, GKUA, FBA, Incompact3d, GRAPES, VASP) decreased by over 90% in 2022 compared to the period from 2017 to 2021, signaling a substantial decline in TaihuLight's utilization in 2022 due to application migrations. Nevertheless, the resource utilization of a few applications has not decreased significantly, prompting an analysis of the primary applications' running status. Table 1 presents the results, where *Common parallelism* indicates the most frequently used running scale, *Successful rate* denotes the probability of applications completing normally, and *Performance fluctuation* reflects the variance in I/O performance across multiple runs under typical configurations.

Table 1: 10 primary applications from 2017 to 2022

| Application | Common parallelism | Successful rate | Performance fluctuation |
|---|---|---|---|
| CESM | 1024 | 40.3% | 22.84 |
| WRF | 1024 | 54.0% | 52.5 |
| GKUA | 512 | 15.4% | 17.9 |
| FBA | 2000 | 32.0% | 43.4 |
| Incompact3d | 1024 | 43.6% | 40.1 |
| GRAPES | 256 | 43.0% | 18.7 |
| VASP | 200 | 41.0% | 28.1 |
| KRPS | 1000 | 81.1% | 4.3 |
| WW3 | 500 | 76.7% | 10.4 |
| COAWST | 1024 | 78.7% | 2.9 |

***Observation* 3.** *Most applications run on a relatively modest parallel scale in daily business operations. Pursuing larger parallel scales or higher computational power may not be a major factor in application migration. Furthermore, jobs with high successful rates and relatively small performance fluctuations tend not to change runtime platforms easily.*

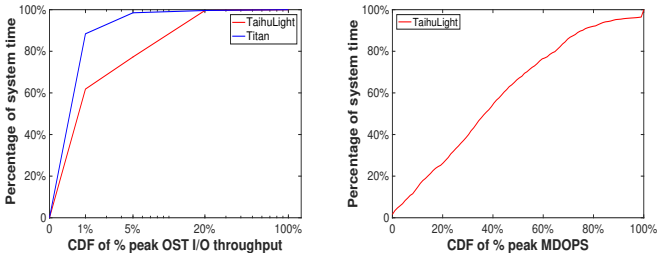
We investigate the factors contributing to the notably high success rates and relatively low-performance fluctuations observed in applications like KRPs, WW3, and COAWST and find that these applications run with relatively modest I/O loads and often utilize exclusive supernodes and I/O forwarding nodes, contributing to the stability of their operational environments.

**Lessons.** With the advent of new supercomputers, especially those with similar architectures, established supercomputing centers face the risk of user attrition. On the one hand, migrating applications to supercomputers with similar architectures only entails little cost. On the other hand, applications can achieve better performance. However, our analysis indicates that not all applications prioritize larger scales or stronger computational power. A crucial factor influencing user retention is the stability of the environment provided by the su-

percomputing center. Instances of high running interruptions, low successful completion rates, and significant performance fluctuations can lead to user dissatisfaction and attrition. For a supercomputing center aspiring to long-term operation, machine upgrades are one facet of maintaining user engagement. Equally important is the provision of a stable environment. We recommend that administrators implement proactive measures, including interrupt detection, resource isolation, and dynamic configuration, to ensure user applications' consistent and stable operation. These steps are paramount for fortifying user loyalty and sustaining the development of supercomputing centers in the long run.

## 4.2 Storage system utilization analysis

Next, we analyze the long-term load of the back-end storage system over nearly six years. Figure 6 shows the Cumulative Distribution Function (CDF) of Lustre OST and MDS usage.




(a) OST usage (b) MDS usage

Figure 6: CDF of storage system usage

***Observation* 4.** *In contrast to the compute nodes, the load on the back-end storage nodes remained consistently low throughout most of the observed period.*

This observation aligns with findings from another supercomputer, Titan [3,37] (marked as the blue line in Figure 6(a)). However, low storage system load does not necessarily translate to easily achieving good application performance. As demonstrated in many previous works [26,48,77], the I/O wall problem plagues many HPC applications. The contradiction between low system load and low application performance is a critical issue in current HPC systems, and *load imbalance* is one of the main reasons [81]. As anticipated, we also find an obvious load imbalance on TaihuLight.

Figure 7 shows the Coefficient of Variation (CoV), calculated as the standard deviation of load divided by the average load, for all OSTs in three months surrounding July 2020. A CoV closer to zero indicates a more balanced load distribution. Before July 2020, TaihuLight featured 384 OSTs, revealing a significant load imbalance and ineffective utilization of back-end storage nodes. Post-July 2020, the CoV of OSTs significantly decreased, potentially attributed to changes in the daily storage system, reducing the number of commonly used OSTs to 12. Additionally, performance optimization tools proposed by Yang et al. [78] contribute to load balancing among system nodes. Figure 8 shows the load status of sampled random I/O forwarding nodes, presenting hourly peak loads on each node. The 100 I/O forwarding nodes, including 80 commonly used ones, exhibit varying darkness levels, representing the achieved maximum bandwidth during that hour. "High," "medium," "low," and "idle" labels correspond to the maximum values in the >60%, 30-60%, 10-30%, and 0-10% intervals, respectively, relative to each forwarding node's peak bandwidth baseline.

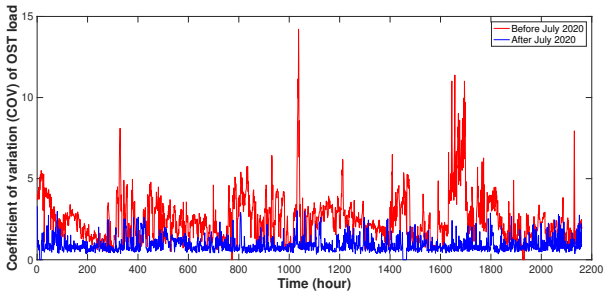



Figure 7: CoV of OST load surrounding July 2020

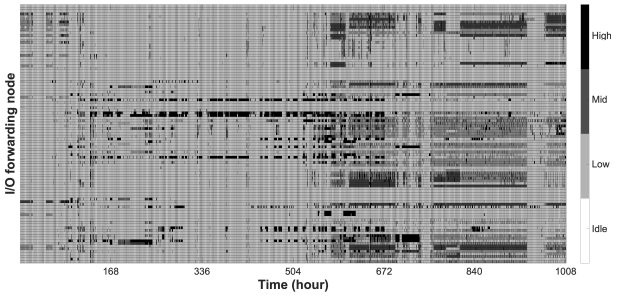



Figure 8: Sample TaihuLight 6-week load summary in 2021, showing the peak load level by the hour across sampled 100 I/O forwarding nodes, including the 80 nodes for daily use

***Observation* 5.** *Load imbalance may occur not only on the back-end storage nodes but also on nodes of other storage tiers, such as I/O forwarding nodes.*

Ji and Yang's research [26,76] underscores that applications frequently suffer I/O interference due to resource competition along the I/O paths, while some applications require additional I/O resources to enhance their performance. This highlights the inadequacy of existing resource allocation strategies in meeting diverse application requirements, contributing significantly to the suboptimal utilization and load imbalance of I/O nodes.

**Lessons.** Load imbalance remains a critical issue in current HPC systems and will occur on all storage tiers [45,51]. Commonly used static or fixed resource allocation policies in the absence of pertinent system operational data exacerbate this issue. Our observed data highlights that these policies, while effective in specific scenarios, prove inadequate in accommodating dynamic load changes within real-world su-

Table 2: Job running information statistics on TaihuLight from April 2017 to December 2022, with a parallel scale greater than 16 processes

| Type | 2017 | 2018 | 2019 | 2020 | 2021 | 2022 |
|---|---|---|---|---|---|---|
| Number of jobs | 308202 | 231153 | 118040 | **1971152** | 139708 | 25677 |
| Avg. core hours | 2107.0 | 3457.8 | 7273.8 | **486.7** | 4732.1 | 1603.8 |
| Avg. parallelism | 1432.8 | 1764.9 | 1724.8 | 2864.5 | 1299.6 | 1175.2 |
| Avg. running time | 21185.9 | 27203.5 | 17392.9 | **728.3** | 17647.9 | 8620.3 |
| Avg. waiting time | 639.1 | **3869.1** | 524.6 | 61.6 | 279.4 | 118.3 |

percomputing production-run environments. This inadequacy results in a substantial misallocation of system resources. Despite many efforts to mitigate this problem, limitations persist, such as only working well on newly created files [69] and lacking a real-time global view to adapt to dynamically changing loads [35]. Therefore, the future system design should embrace a more dynamic and flexible allocation strategy tailored to specific application scenarios coupled with real-time performance monitoring tools.

***Observation* 6.** *Metadata usage has a different story: the MDS load significantly surpasses that of OSTs, with the MDS load consistently exceeding 40% for approximately 50% of the total time. Notably, there are instances, accounting for 4% of the time, where the metadata load exceeds 98%.*

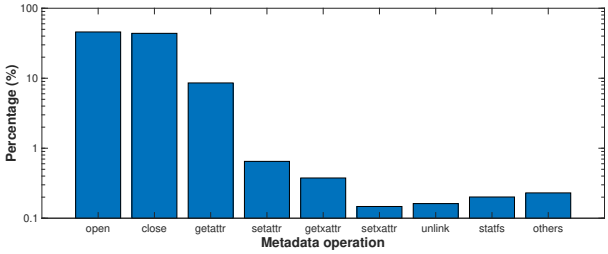



Figure 9: Statistics on metadata operations

Extremely high metadata loads underscore the potential bottleneck caused by metadata access in modern storage systems. An in-depth analysis of the distribution of various metadata operations reveals insightful findings, as depicted in Fig 9. As anticipated, *open* and *close* operations constitute around 90% of the total metadata requests, as applications operate files that must open and close them. Surprisingly, *getattr* commands contribute nearly 9% to the overall metadata requests. Examination of periods with heightened Lustre client *getattr* requests exposes that applications that access many grid files (e.g., via NetCDF) exhibit a propensity for invoking *sys_stat*, resulting in many *getattr* requests, was a significant contributor to this phenomenon. Additionally, Lustre's readahead mechanism will prefetch many files at high file concurrency, also significantly contributing to this phenomenon.

**Lessons.** Currently, HPC applications exhibit numerous metadata demands, making the capabilities of metadata services a pivotal metric for evaluating storage systems. In light of this, we recommend that storage system designs take more into account applications with high metadata requirements and their metadata access habits (i.e., more open/close/getattr access). Our in-depth analysis of diverse metadata access behaviors of applications is expounded upon in Section 6. This exploration aims to provide comprehensive insights for designing storage systems that effectively cater to the distinctive needs posed by high-metadata-demand applications.

# 5 Job running information analysis

## 5.1 Job scheduling strategy analysis

In this section, we conduct a comprehensive analysis of the job running information on TaihuLight over the past six years and statistic key metrics such as the number of job submissions, average job parallelism (indicative of the job's parallel scale), average job running time, and average job waiting time for each year spanning from 2017 to 2022. The summarized results are presented in Table 2.

***Observation* 7.** *The year 2020 witnessed the highest volume of job submissions, but the average core hours consumed by jobs and the average job running time are very small.*

We then analyze the jobs submitted in 2020 and find two primary contributing factors. Firstly, a notable proportion, approximately 10%, are abnormally terminated jobs. Here, "abnormally terminated job" refers to jobs that exit unexpectedly due to various reasons. Secondly, a few users, later identified as administrators, submitted over a million jobs, seemingly for testing and debugging purposes (with most jobs concluding in less than 200 seconds). Given the upgrade in the storage system of TaihuLight in July 2020, this submission surge could be attributed to administrators conducting tests on the new storage system using specific benchmarks.

***Observation* 8.** *The average job waiting time per year is consistently high, typically measured in hundreds of seconds. Notably, in 2018, the average job waiting time significantly surpassed that of other years, reaching 3869.1 seconds.*

We then further analyze the waiting time of all jobs from April 2017 to December 2022 and find that 2.8% of the jobs experienced exceptionally long waiting times, with some surpassing 80,000 seconds. Here, we reveal a problem, *job starvation*, which indicates that jobs experience prolonged periods without scheduling. Upon excluding these abnormal jobs, we re-evaluated the waiting time for the remaining jobs. The re-

sults, illustrated in Figure 10, indicate that the waiting time for most jobs is within 100 seconds. To delve deeper into the factors contributing to the extended waiting times of certain jobs, we scrutinize a subset of 3,620 jobs running within the same job queue between March 1, 2018, and March 7, 2018. Employing job scheduling simulations, we evaluate the impact of four distinct scheduling policies on waiting times:

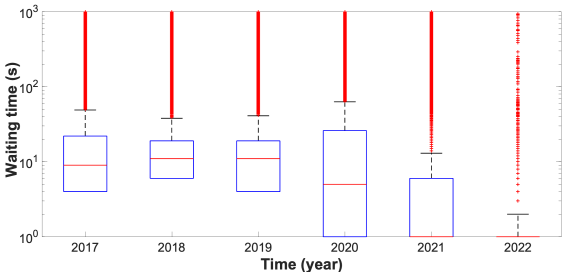
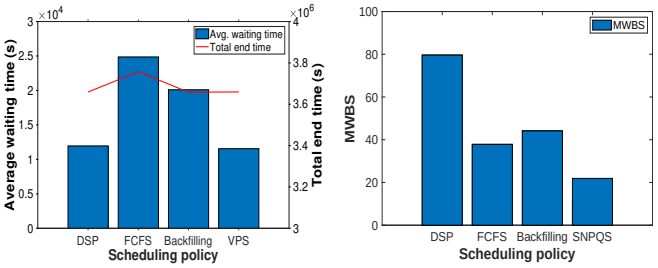



Figure 10: Distribution of job waitting time

- DSP (Default Scheduling Policy), also known as "small-scale job priority scheduling," is the inherent scheduling policy adopted by the TaihuLight scheduler and other supercomputers [80]. This policy assigns higher priority to jobs with fewer nodes.
- FCFS (First-Come, First-Served) [43] is the simplest job scheduling policy, which schedules jobs in the order they are submitted.
- The Backfilling policy [20] allows subsequent jobs to occupy the gaps created by the first blocked job in the waiting queue, as long as they do not cause delays in the expected start times of the blocked jobs in the queue. This approach enhances system utilization by filling these gaps while preventing starvation by allocating reservations to the blocked jobs.
- VPS (Variable Priority Scheduling) is our designed incremental scheduling policy based on DSP. Its key idea is to place small jobs (jobs that use fewer node resources) at the front of the queue for priority execution. However, if a job's wait time exceeds a predetermined threshold, the system will promote the job to the top of the queue to ensure the job can be run as quickly as possible.

Figure 11(a) shows the average waiting time of jobs across four different scheduling policies, accompanied by the end time of the last job. Note that the smaller the total end time, the higher the resource utilization. DSP, VPS, and Backfilling exhibit superior performance by optimizing the order of job submissions, leading to reduced resource idle time and shorter job completion periods. In contrast, FCFS, constrained by its inability to alter job execution order, yields comparatively inferior results. Noteworthy is that the DSP results presented in Figure 11(a) emanate from simulation, closely matching the real collected job running information. This congruence attests to the accuracy of our simulation methodology. However, the above evaluation methods cannot reveal the problems caused by job starvation. We additionally use MWBS (Mean Weighted Bounded Slowdown) to evaluate these scheduling policies, as shown in Figure 11(b). MWBS are used to evaluate user-aware performance in parallel job scheduling [82]. Formulas 1 and 2 show the calculation flow of the MWBS.




(a) Avg. waiting time and total end time (b) Mean weighted bounded slowdown

Figure 11: Evaluation on four scheduling policies

$$Slowdown = \frac{Waitingtime + \max(Runtime, X)}{\max(Runtime, X)} \quad (1)$$

$$MWBS = \frac{\sum_{j=1}^{N}(Slowdown_j \cdot Parallelsim_j)}{\sum_{j=1}^{N} Parallelism_j} \quad (2)$$

In HPC systems, we pay more attention to large-scale parallel jobs, so we use MWBS and regard jobs' parallelism as a weight. Moreover, we set X to 10 to minimize the impact of jobs that immediately exit due to some errors after submission. The experimental results show that although DSP guarantees resource utilization and has advantages in terms of average job waiting time and total end time, it has the highest MWBS and is prone to starvation, reducing users' satisfaction. In contrast, VPS retains the idea of prioritizing small-scale jobs and ensures that large-scale jobs do not have to wait long, so it performed well in all of our tests.

**Lessons.** Numerous supercomputers commonly implement the small-scale job priority scheduling policy as their default choice, aiming to enhance resource utilization. Our findings, however, reveal a potential issue, *job starvation*, associated with this policy. Job starvation can increase user-aware performance. For example, jobs usually running for a day may be extended to three or more days due to scheduling delays. We suggest that administrators should pay attention to this problem because the prolonged waiting time diminishes user satisfaction and poses the risk of user attrition, adversely affecting the supercomputing center. To mitigate this problem, we propose a novel scheduling algorithm designed to minimize the occurrence of starved jobs while concurrently maximizing compute node utilization.

## 5.2 Users' job submission habits analysis

In this section, we analyze job submissions from the user-level perspective, counting the number of job submissions and the average job core hourly consumption at an hourly granularity (each time period records all job submissions in

the corresponding time period on each day for nearly six years). Figure 12 shows the results. The blue bar represents the number of job submissions for different time periods, and the red line represents the average core hour consumption per submission for different time periods.

***Observation* 9.** *User activity is notably concentrated during the on-duty period, with the highest volume of job submissions occurring between 9 a.m. and 5 p.m., excluding the noon-hour break (12 p.m. and 1 p.m.). Interestingly, many job submissions persist during the off-duty period, from 6 p.m. to 8 a.m. the following day.*

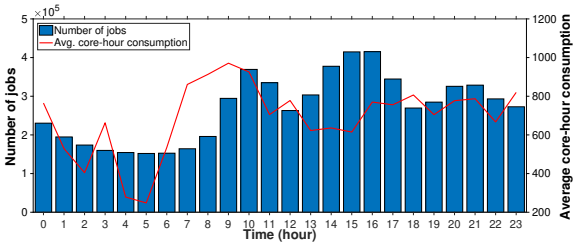



Figure 12: Number of job submissions and the average core-hour consumption per submission at different time periods

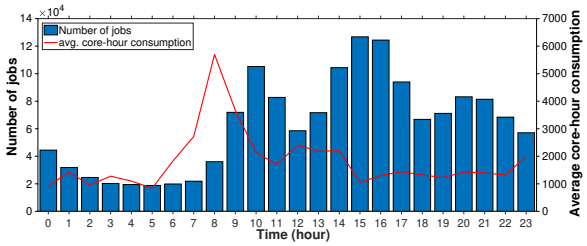
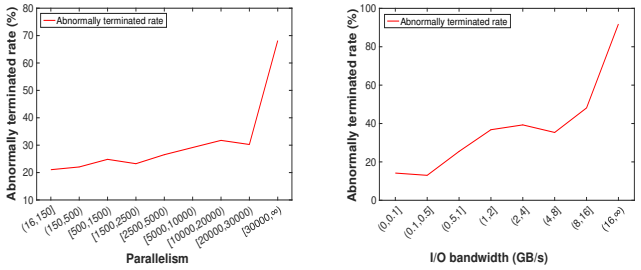



Figure 13: Number of abnormally terminated jobs and the average core-hour consumption per job at different time periods

More jobs submitted during the on-duty period align with our expectations, so we will focus on jobs that run during the off-duty period. Figure 13 shows the number of abnormally terminated jobs during different time periods.

***Observation* 10.** *A large proportion of jobs are abnormally terminated, and large-scale and long-running jobs tend to exit abnormally between 8 a.m. and 9 a.m.*

This observation reveals a problem in current supercomputers: *job hanging*, where a job, due to factors like storage system stalls, network disruptions, or node connectivity problems, becomes stuck and unable to progress in its computation but still occupies compute nodes. Users typically check the status of their jobs submitted the previous day upon returning to work and kill any abnormal jobs. Notably, large-scale and long-running jobs tend to terminate abnormally between 8 a.m. and 9 a.m. Jobs with abnormal terminations during this timeframe consumed a substantial 469,537,688.3 core hours, constituting around 11.6% of the total core hours expended by all applications over six years on TaihuLight. This underscores the critical nature of job hanging in HPC, leading to the wastage of valuable compute resources. Besides, the cumulative core hours of all abnormally terminated jobs surpass that of normally completed jobs, prompting us to conduct a deeper analysis. We explore the correlation between the abnormally terminated rate and some job characteristics and find that jobs' parallelism and I/O bandwidth are the two most relevant parameters. Figure 14 presents the results.




(a) Jobs' parallelism (b) Jobs' average I/O bandwidth

Figure 14: Relationship between the abnormally terminated rate and jobs' parallelism and I/O load

The trend in the figure reveals that large-scale jobs and those with high I/O loads exhibit significantly higher rates of abnormal termination compared to others with smaller-scale and lower I/O loads, aligning with our conjectures. Remarkably, the abnormal termination rates soar to 70% and 90% for jobs with processes exceeding 30,000 and an average I/O bandwidth surpassing 16 GB/s, respectively. Further analysis suggests that this anomaly might be attributed to high-load benchmarks executed by system administrators. Administrators routinely employ such benchmarks during presumed low system loads, typically during off-duty periods, to detect anomalies in some nodes or system components (Also mentioned in Section 4.1). These benchmarks, characterized by high load (e.g., network or storage system detection benchmarks), can amplify and capture abnormal system issues such as unstable network links or storage system performance degradation. These problems, in turn, significantly impact large-scale, high-load jobs submitted by users during off-hours, leading to job hanging.

**Lessons.** Job hanging is a critical problem in HPC [19, 63], leading to a significant waste of valuable compute resources and necessitating more attention. This issue is particularly pronounced during off-duty periods, where large-scale and long-running job hangs can severely impede supercomputer utilization. While regular benchmark testing by administrators serves to detect system issues, it can inadvertently exacerbate problems, leading to application hangs. We advise incorporating application hang detection into periodic benchmark testing to mitigate this issue. It is crucial to note that effective job hanging detection often requires tight integration with applications, such as periodic logging by applications. We encourage users to bolster their applications' fault tolerance mechanisms. Simultaneously, we are actively engaged in detection efforts at the system level. This involves routine

sampling of applications' communication and I/O behavior through Beacon. Applications with prolonged periods of inactivity will trigger alerts, enabling timely communication with users and taking measures to address potential issues.

Furthermore, we also analyze the relationship between users' job submission habits and storage system load, and Figure 15 shows the average OST load analysis in hourly granularity.

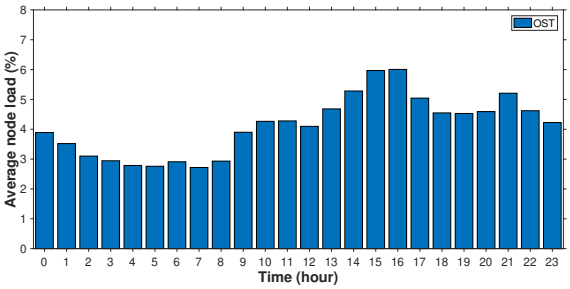



Figure 15: Average OST load per hour

***Observation* 11.** *Bars in Figure 12 and Figure 15 have similar trends, implying that the average load of the storage system is directly proportional to the number of job submissions during the same time period.*

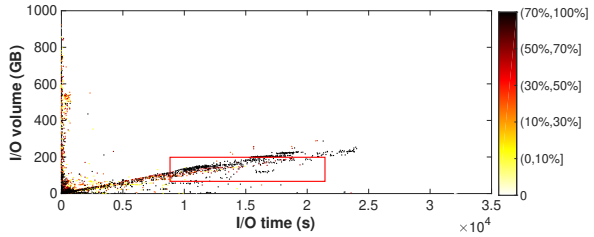
We speculate that the observed phenomenon is linked to the program execution process, where compute nodes must read the executable file from the storage node. In the case of large-scale applications, this operation involves reading substantial data, potentially reaching hundreds of gigabytes. To illustrate, if Application1 runs with a parallel scale of 10,000 processes and the executable file is 10MB, the compute nodes must read around 100GB of data before program execution. This substantial load, arising from retrieving extensive executable files, contributes to the rise in average back-end storage load with an increasing number of job submissions.

Further investigation into the sizes of applications' executables on TaihuLight reveals a noteworthy trend—they are often huge. This can be attributed to the inherent complexity of scientific computing applications, which frequently involve intricate multi-mode coupling, resulting in a substantial amount of code. A case in point is CESM [31], boasting millions of lines of code and consequently yielding excessively large executable files. Additionally, it is noteworthy that applications on TaihuLight tend to use static linking. While this practice aids in minimizing link library overhead during runtime and enhancing overall speed, it concurrently leads to increased disk usage and memory overhead due to the nature of static linking.

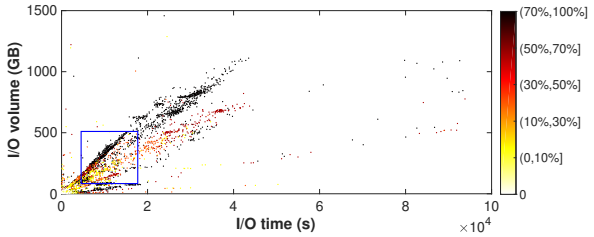
**Lessons.** Static linking, while beneficial for faster running speed, can introduce challenges in handling substantial data reads from the back-end storage system during the execution of large-scale applications, causing a significant impact on the file system and increasing application startup overhead. Dynamic linking, while slightly sacrificing time, offers enhanced space efficiency and flexibility. In light of these considerations, we advocate for a judicious evaluation of the trade-offs by application developers. Specifically, we recommend using dynamic linking for large applications where space efficiency is critical and adopting static linking for smaller applications where the emphasis is on speed optimization and the impact on storage and startup overhead is comparatively low.

## 6 Job I/O characteristics analysis

In this section, we perform a statistical analysis for the I/O characteristics of jobs that feature non-trivial I/O. Figure 16 shows the I/O volume and I/O time for read-dominant and write-dominant jobs, respectively, and each dot in the figures represents a job. The colors of the different points represent different jobs' I/O ratios (I/O ratio refers to the percentage of I/O time to the total running time of a job ), and the darker the color, the higher the I/O ratio. We can see that many jobs have a high I/O ratio, both for read-dominant and write-dominant jobs. However, the average I/O bandwidth is relatively low, less than 100 MB/s. Combined with the storage system load analysis in Figure 6(a), this shows that I/O is still a problem even when the storage system load is very low, confirming the current "I/O wall" problem in HPC.




(a) Read-dominant jobs



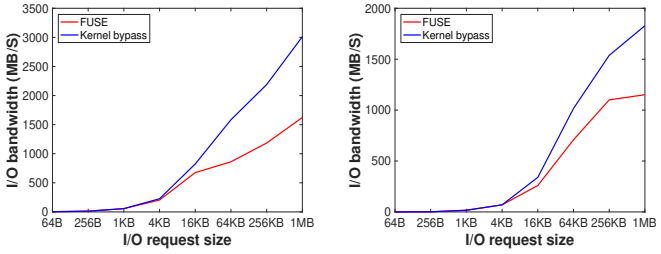

(b) Write-dominant jobs

Figure 16: The I/O volume and I/O time of jobs with non-trivial I/O from April 2017 to December 2022

***Observation* 12.** *Most jobs are gathered on several straight lines of the coordinate axis.*

By calculating the ratio of I/O volume to I/O time, we can get the slope of these lines, which also represents the average I/O bandwidth of these jobs. We further analyze the jobs on the same line. In conjunction with the average IOPS of the jobs, we find that for read, the average I/O request size of the jobs marked by the red boxes in Figure 16(a) is

128KB. For write, the average I/O request size of the jobs marked by the blue boxes in Figure 16(b) is 512KB, which also matches the previous work finding [76], mainly due to the limitation of the FUSE file system (As mentioned above, the storage system on TaihuLight consists of fuse-based LWFS and Lustre). To improve the performance of these applications with large I/O requests, system developers introduced the kernel bypass method instead of the original FUSE file system. However, there are many other jobs with small I/O volumes but very long I/O times, and most have small I/O requests. According to our statistics, jobs' average I/O requests below 4KB accounted for more than 16%. Here, our experiments demonstrate that kernel bypass is not always optimal and that we sometimes need kernel back.




(a) Read I/O bandwidth (b) Write I/O bandwidth

Figure 17: Performance comparison with or without FUSE

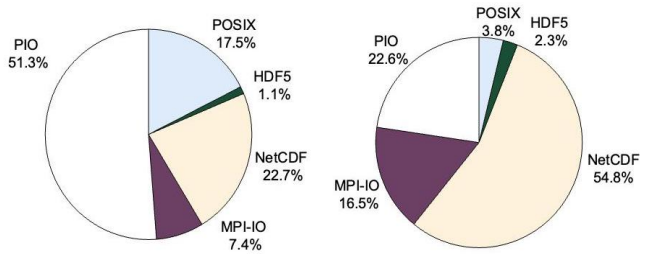
We evaluate the performance of FUSE and kernel bypass in handling various I/O request sizes with the MPI-IO benchmark of 16 computing nodes (64 processes). Figure 17 shows the results. When the I/O request size is less than 4KB, due to the kernel cache mechanism, the performance of FUSE will be better than that of kernel bypass. However, when the I/O request size increases, kernel bypass will show extremely high-performance scalability.

**Lessons.** FUSE and kernel bypass represent two prevalent methods adopted by numerous file systems; each has its own applicable scenarios. Our real-world data analysis and experimental tests reveal that, despite FUSE often facing criticism for its lower performance, it can exhibit commendable results in certain circumstances. Particularly, when the supercomputing system is oriented to applications with mostly read and write small blocks, FUSE can deliver satisfactory performance. Coupled with its simple and convenient development process, it still has good use value. Owing to space limits and the primary focus of this paper, we refrain from conducting more intricate tests on the FUSE and kernel bypass methods.

We then analyze the primary I/O libraries utilized on TaihuLight, with applications relying on these libraries constituting a dominant percentage, surpassing 90%. Figure 18 presents the outcomes of this analysis, including PIO [18], NetCDF [57], POSIX [10], HDF5 [33], and MPI-IO [65].

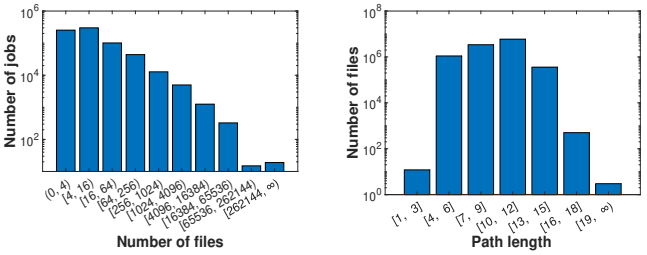
***Observation* 13.** *Traditional scientific computing applications and the self-describing format I/O libraries they use still occupy a major position.*

When statistics in terms of core hours, applications employing PIO constitute over 50% of the total core hours because PIO-dependent applications like CESM [31] consume lots of compute resources on TaihuLight. However, when statistics are in units of I/O volume, the proportion of NetCDF experiences a notable increase. This shift is attributed to NetCDF-based production applications such as WRF [52] and GRAPES [15], contributing significantly to the overall I/O volume.

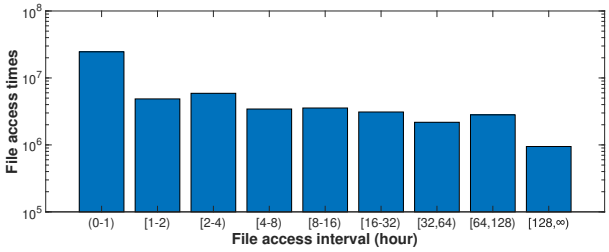



(a) In occupied core hours (b) In I/O volume

Figure 18: Distribution of commonly used I/O libraries




(a) Number of files accessed by jobs (b) Files' path length




(c) Files' access interval

Figure 19: Statistics on files accessed by jobs

**Lessons.** Traditional scientific computing applications still occupy a major position on TaihuLight. However, the traditional I/O libraries they use have been challenging to meet the needs of application development, resulting in I/O overhead gradually increasing, and cannot effectively exert the storage system's performance. Therefore, for I/O optimization of scientific computing applications, especially application-level optimization methods, to promote the optimization method, the compatibility of API interfaces is essential.

Subsequently, we focus on the file access characteristics of jobs, delving into the analysis of more than 700,000 jobs with

over 10 million files. Notably, 50% of these files are shared resources accessed by multiple distinct jobs. Figure 19(a) shows the number of files accessed by various jobs.

***Observation* 14.** *Most jobs only access less than 1024 files, accounting for around 90%, but a minority of jobs can access an extensive number of files, even exceeding 262,144.*

We focus on jobs that access a significant number of files because extensive file operations, especially for small files, can easily cause application performance bottlenecks. Further analysis reveals that most jobs belong to artificial intelligence applications, which frequently operate small files. Given the growing prevalence of AI, it is anticipated that the proportion of AI applications will continue to rise, intensifying the demand for metadata. This trend has prompted various supercomputers, including TaihuLight, equipped with only one metadata server, to undergo upgrades [1, 2]. Additionally, we analyze file path lengths and file access intervals by jobs, with the results presented in Figure 19(b) and Figure 19(c).

***Observation* 15.** *Most files' path lengths range from 10 to 12, accounting for 50%, but still, a notable portion of files have lengths surpassing 12, extending up to 19. In addition, files with access intervals within 1 hour occupy a majority.*

**Lessons.** The I/O characteristics of applications play a crucial role in the design, management, and optimization of HPC storage systems. Our findings offer valuable insights for system designers and administrators, enabling them to comprehend application I/O characteristics and trends. This understanding facilitates designing better systems or taking measures to improve application I/O performance and overall system utilization. For instance, addressing long path lengths is essential to prevent performance issues and boost file system efficiency. Administrators can advise users to routinely clean up unnecessary files, merge small files to reduce the overall count or employ Symbolic Links instead of deep directory structures that map certain deep-level directory structures to shallower ones through symbolic links to simplify paths. Moreover, the frequent simultaneous access of shared files by multiple jobs within a short timeframe presents an opportunity to leverage emerging storage technologies, such as Burst Buffer [27, 64].

## 7 Discussion

It should be acknowledged that the conclusions of this paper are partially but not fully applicable to other supercomputing systems due to the special architecture and other reasons. However, the analytical methodology employed and the key findings presented herein can offer valuable insights applicable to diverse systems. For example, our analysis of scheduling policies, job hanging, and system anomalies can serve as benchmarks for other systems to assess the presence of similar issues and implement corresponding improvements. Our in-depth analysis of applications' I/O behaviors, such as metadata access patterns, file access patterns, and I/O trends, provides valuable guidance for the HPC community in designing and improving storage systems to accommodate diverse application needs better. Below, we give more discussions.

*Suggestions for supercomputing operations and maintenance.* Administrators typically perform system management holistically from the perspective of the entire system, which may inadvertently neglect the impact on individual jobs. Our research has uncovered many issues caused by operational and scheduling strategies, such as job hangings and job starvation, which can substantially waste valuable compute resources and diminish user satisfaction. Notably, fault tolerance remains an important issue that is difficult to address at the user or administrator level alone, necessitating collaborative efforts between users and administrators. Moreover, we propose proactive solutions to address these challenges and hope this research can help HPC system administrators improve the overall quality of operations and maintenance practices.

*Suggestions for performance optimization.* Performance optimization methods can be broadly categorized into application-level and system-level strategies. For application-level optimization, our study confirms that current HPC applications still extensively use legacy I/O libraries. If these I/O libraries are not compatible, it is not easy to generalize the optimization approach. For system-level optimization, designers and administrators confront many issues that impact performances, such as load imbalance and interference. To address these issues, system parameters and scheduling strategies adjustment should both be based on the real-world load analysis results. Moreover, optimizing performance in current specific HPC scenarios, such as deep directory tree access, extensive small file operations, and shared file access (especially when numerous files are accessed frequently within short intervals), also requires very detailed I/O characteristics of applications and systems. Our study provides valuable insights for system designers and administrators and can offer guidance for optimizing performance in diverse HPC scenarios.

*Challenges and opportunities for future monitoring tools.* Using data collected by Beacon, encompassing multi-layer I/O data and job running information, we have unearthed various I/O performance issues and shed light on operational and management challenges. However, it is crucial to note that certain issues require additional information to investigate their root causes comprehensively. For instance, a precise understanding of job hangings requires insights into network status, CPU load, disk information, etc. As supercomputing systems evolve in scale and complexity, identifying and diagnosing issues are anticipated to become more critical and intricate. In conclusion, adopting finer granularity and multi-field data collection can enhance issue discovery and facilitate understanding of their underlying causes. Nevertheless, it is imperative to recognize that such data collection strategies may incur additional costs, and balancing the granularity of data collection across different fields remains challenging.

## 8 Related work

Due to the essential nature of data analysis for understanding the operational status of the system, identifying I/O access patterns of applications, predicting future load changes, optimizing I/O performance for applications, and solving other critical tasks, in recent years, many researchers have proposed different methods for analyzing the I/O characteristics of HPC applications, storage system load characteristics, and other information.

**I/O monitoring tool.** Various techniques and tools have been proposed to monitor and analyze the I/O performance in HPC applications and systems. Generally, they can be divided into three categories according to the position of the collection point. The first category is application-layer collection tools, which mainly start from the application side and obtain I/O information by instrumenting high-level I/O libraries, such as Darshan [14], Reflector [4], RIOT [72], etc. Application-layer collection tools are close to the application and are suitable for fine-grained analysis of application I/O behavior. The second category of tools is system-level collection tools aimed at the backend file system. They mainly describe the system load by collecting file system status information, such as Luster RPC and OST information, and typical tools include LIOProf [75], LMT [22], LustreDu [13] and so on. System-level collection tools have low overhead and are easy to deploy on a large scale, but it is challenging to analyze application behavior effectively. The third category is cross-layer monitoring tools, which combine the advantages of application-layer and system-level collection tools, such as TOKIO [6], Beacon [76], GUIDE [68], and so on. Overall, due to the widespread deployment of advanced monitoring, some previously undiscovered problems have gradually been revealed, effectively supporting application and system performance analysis and optimization.

**Application-side analysis.** Some works focus on the analysis of job running information. For example, Rodrigo and Patel et al. [50, 58] used job running information on several supercomputers, including running time, core hours, and other information, and analyzed the primary parallelism of jobs and job submissions in different time periods, etc. Simakov et al. [62] also analyzed job running information and uncovered trends in job parallelism that have reduced. Other works focus on the application's I/O burst analysis. For example, Yang et al. [79] analyzed the I/O burst of jobs, found that I/O bursts widely exist in HPC applications, and found that applications with similar job names often have similar I/O behaviors. Some work [17, 37, 47] clustered the jobs with similar features and studied similar periodic I/O burst behaviors. In addition, many works analyzed I/O interference between applications, performance variation, and other performance issues, such as [11, 12, 28, 53]. Unlike the previous works, this paper analyzes job running information and I/O characteristics on TaihuLight from a long-term perspective. Besides, we focus on exploring the fine-grained I/O characteristics, such as the I/O request size and file access offsets. Our analysis reveals the actual needs of the current application and can provide data support for system designers and administrators to service applications well.

**System-side analysis.** Many works analyzed the system load, characterized the storage in detail, displayed periods of idle times, and studied the correlation of I/O bursts among OSTs [23, 32, 46, 48, 49, 70]. Then, optimization suggestions are put forward for the resource allocation strategy and system parameter configuration based on the storage system analysis. For example, Oral et al. [46] used benchmark suites to analyze different types and configurations of file and storage systems and then provided recommendations. Wadhwa et al. [69] analyzed the Lustre OST load and proposed an optimization method for load balance based on the lightweight I/O monitoring system. Lockwood et al. [38] used probes to test the file system's load, but the probes' overhead was high, and due to resource allocation problems, it was difficult to reflect the actual state of the system, which would affect the analysis results. Xie et al. [74] proposed a statistical benchmarking methodology to measure write performance across I/O configurations, hardware settings, and system conditions. They analyzed the I/O write behaviors of the Titan supercomputer and its Lustre parallel file stores under production load and gave several recommendations for optimization. Unlike the previous works, we analyze the load of Sunway TaihuLight's multi-level nodes from multiple aspects in terms of long-term and spatial scales, reveal some problems suffered during its long-term operation, and propose some optimization suggestions. Our experience can be applied by the HPC community to design better supercomputing systems or operate and maintain better supercomputing systems to improve user experience and meet user application needs.

## 9 Conclusion

We perform a comprehensive and detailed analysis of system and job characteristics and their trends on the current world's No.11 supercomputer, TaihuLight. Our study confirms some previous findings and conjectures, uncovers and explains many previously undiscovered questions and unquantified insights, and proposes some suggestions for the identified problems. For instance, application performance fluctuations and multi-layer system node load imbalances persist and demand careful consideration; the waste of resources due to job hanging, and both users and system administrators need to pay attention to fault tolerance; the design and optimization of storage systems need to consider some hot scenarios, such as access to a large number of small files, a large number of files being accessed by many applications in a short period, etc. We believe that the HPC community can apply our experience to design, optimize, and manage scalable HPC systems to meet the needs of their users.

## References


[1] Sequoia supercomputer. https://computation.llnl.gov/computers/sequoia.

[2] Sierra supercomputer. https://hpc.llnl.gov/hardware/platforms/sierra.

[3] Titan supercomputer. https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/.

[4] Abdullah Al-Mamun, Jialin Liu, Tonglin Li, Quincey Koziol, Zhongyi Zhai, Junyan Qian, Haoting Shen, and Dongfang Zhao. Reflector: a fine-grained i/o tracker for hpc systems. In *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 427–428, 2020.

[5] Paul Bartholomew, Georgios Deskos, Ricardo AS Frantz, Felipe N Schuch, Eric Lamballais, and Sylvain Laizet. Xcompact3d: An open-source framework for solving turbulence problems on a cartesian mesh. *SoftwareX*, 12:100550, 2020.

[6] Lawrence Berkeley and ANL. TOKIO:Total knowledge of I/O, 2017. http://www.nersc.gov/research-and-development/tokio.

[7] Peter Braam. The lustre storage architecture. *arXiv preprint arXiv:1903.01955*, 2019.

[8] Tim Bray. The javascript object notation (json) data interchange format. Technical report, 2014.

[9] A Brinkmann, K Mohror, and W Yu. Challenges and opportunities of user-level file systemsfor hpc. 2017.

[10] David R Butenhof. *Programming with POSIX threads*. Addison-Wesley Professional, 1997.

[11] Kirk W Cameron, Ali Anwar, Yue Cheng, Li Xu, Bo Li, Uday Ananth, Jon Bernard, Chandler Jearls, Thomas Lux, Yili Hong, et al. Moana: Modeling and analyzing i/o variability in parallel system experimental design. *IEEE Transactions on Parallel and Distributed Systems*, 30(8):1843–1856, 2019.

[12] Zhen Cao, Vasily Tarasov, Hari Prasath Raman, Dean Hildebrand, and Erez Zadok. On the performance variation in modern storage stacks. In *FAST*, pages 329–344, 2017.

[13] Adam G Carlyle, Ross G Miller, Dustin B Leverman, William A Renaud, and Don E Maxwell. Practical support solutions for a workflow-oriented cray environment. In *Proceedings of Cray User Group Conference (CUG 2012)*, 2012.

[14] Philip Carns, Robert Latham, Robert Ross, Kamil Iskra, Samuel Lang, and Katherine Riley. 24/7 characterization of petascale i/o workloads. In *International Conference on Cluster Computing and Workshops*, pages 1–10, New Orleans, 2009. IEEE.

[15] DeHui Chen, JiShan Xue, XueSheng Yang, HongLiang Zhang, XueShun Shen, JiangLin Hu, Yu Wang, LiRen Ji, and JiaBin Chen. New generation of multi-scale nwp system (grapes): general scientific design. *Chinese Science Bulletin*, 53(22):3433–3445, 2008.

[16] Qi Chen, Kang Chen, Zuo-Ning Chen, Wei Xue, Xu Ji, and Bin Yang. Lessons learned from optimizing the sunway storage system for higher application i/o performance. *Journal of Computer Science and Technology*, 35:47–60, 2020.

[17] Emily Costa, Tirthak Patel, Benjamin Schwaller, Jim M Brandt, and Devesh Tiwari. Systematically inferring i/o performance variability by examining repetitive job behavior. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2021.

[18] John M Dennis, Jim Edwards, Ray Loy, Robert Jacob, Arthur A Mirin, Anthony P Craig, and Mariana Vertenstein. An application-level parallel i/o library for earth system models. *The International Journal of High Performance Computing Applications*, 26(1):43–53, 2012.

[19] Catello Di Martino, William Kramer, Zbigniew Kalbarczyk, and Ravishankar Iyer. Measuring and understanding extreme-scale application resilience: A field study of 5,000,000 hpc application runs. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 25–36. IEEE, 2015.

[20] Dror G Feitelson. Experimental analysis of the root causes of performance evaluation results: a backfilling case study. *IEEE Transactions on Parallel and Distributed Systems*, 16(2):175–182, 2005.

[21] Haohuan Fu, Junfeng Liao, Jinzhe Yang, Lanning Wang, Zhenya Song, Xiaomeng Huang, Chao Yang, Wei Xue, Fangfang Liu, Fangli Qiao, et al. The sunway taihulight supercomputer: system and applications. *Science China Information Sciences*, 59:1–16, 2016.

[22] Jim Garlick. Lustre monitoring tool, 2010. https://github.com/LLNL/lmt.

[23] Raghul Gunasekaran, Sarp Oral, Jason Hill, Ross Miller, Feiyi Wang, and Dustin Leverman. Comparative i/o workload characterization of two leadership class storage clusters. In *Proceedings of the 10th Parallel Data Storage Workshop*, pages 31–36, 2015.

[24] Haryadi S Gunawi, Riza O Suminto, Russell Sears, Casey Golliher, Swaminathan Sundararaman, Xing Lin, Tim Emami, Weiguang Sheng, Nematollah Bidokhti, Caitie McCaffrey, et al. Fail-slow at scale: Evidence of hardware performance faults in large production systems. *ACM Transactions on Storage (TOS)*, 14(3):1–26, 2018.

[25] Jürgen Hafner. Materials simulations using vasp—a quantum perspective to materials science. *Computer physics communications*, 177(1-2):6–13, 2007.

[26] Xu Ji, Bin Yang, Tianyu Zhang, Xiaosong Ma, Xiupeng Zhu, Xiyang Wang, Nosayba El-Sayed, Jidong Zhai, Weiguo Liu, and Wei Xue. Automatic, application-aware i/o forwarding resource allocation. In *Proceedings of the 17th USENIX Conference on File and Storage Technologies*, pages 265–279, 2019.

[27] chen Kang, Wu Yongwei, and zheng Weiming. Madfs: a high performance burst buffer file system. *Big Data*, 7(3):150, 2021.

[28] Yao Kang, Xin Wang, Neil McGlohon, Misbah Mubarak, Sudheer Chunduri, and Zhiling Lan. Modeling and analysis of application interference on dragonfly+. In *Proceedings of the 2019 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pages 161–172, 2019.

[29] Go Kato, Mikio Fujiwara, and Toyohiro Tsurumaru. Advantage of the key relay protocol over secure network coding. *IEEE Transactions on Quantum Engineering*, 4:1–17, 2023.

[30] Kenneth J Kauffman, Purusharth Prakash, and Jeremy S Edwards. Advances in flux balance analysis. *Current opinion in biotechnology*, 14(5):491–496, 2003.

[31] Jennifer E Kay, Clara Deser, A Phillips, A Mai, Cecile Hannay, Gary Strand, Julie Michelle Arblaster, SC Bates, Gokhan Danabasoglu, James Edwards, et al. The community earth system model (cesm) large ensemble project: A community resource for studying climate change in the presence of internal climate variability. *Bulletin of the American Meteorological Society*, 96(8):1333–1349, 2015.

[32] Youngjae Kim and Raghul Gunasekaran. Understanding i/o workload characteristics of a peta-scale storage system. *The Journal of Supercomputing*, 71:761–780, 2015.

[33] Sandeep Koranne and Sandeep Koranne. Hierarchical data format 5: Hdf5. *Handbook of open source tools*, pages 191–200, 2011.

[34] Julian Kunkel and Eugen Betke. Tracking user-perceived i/o slowdown via probing. In *High Performance Computing: ISC High Performance 2019 International Workshops, Frankfurt, Germany, June 16-20, 2019, Revised Selected Papers 34*, pages 169–182. Springer, 2019.

[35] Xiuqiao Li, Limin Xiao, Meikang Qiu, Bin Dong, and Li Ruan. Enabling dynamic file i/o path selection at runtime for parallel file system. *The Journal of Supercomputing*, 68:996–1021, 2014.

[36] Zhi-Hui Li, Ao-Ping Peng, Qiang Ma, Lei-Ning Dang, Xiao-Wei Tang, and Xue-Zhou Sun. Gas-kinetic unified algorithm for computable modeling of boltzmann equation and application to aerothermodynamics for falling disintegration of uncontrolled tiangong-no. 1 spacecraft. *Advances in Aerodynamics*, 1(1):1–21, 2019.

[37] Yang Liu, Raghul Gunasekaran, Xiaosong Ma, and Sudharshan S Vazhkudai. Server-side log data analytics for i/o workload characterization and coordination on large shared storage systems. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 819–829. IEEE, 2016.

[38] Glenn K Lockwood, Shane Snyder, Teng Wang, Suren Byna, Philip Carns, and Nicholas J Wright. A year in the life of a parallel file system. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 931–943. IEEE, 2018.

[39] Jay Lofstead, Ivo Jimenez, Carlos Maltzahn, Quincey Koziol, John Bent, and Eric Barton. Daos and friends: a proposal for an exascale storage system. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 585–596. IEEE, 2016.

[40] Uri Lublin and Dror G Feitelson. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105–1122, 2003.

[41] Jakob Lüttgau, Shane Snyder, Philip Carns, Justin M Wozniak, Julian Kunkel, and Thomas Ludwig. Toward understanding i/o behavior in hpc workflows. In *2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS)*, pages 64–75. IEEE, 2018.

[42] Sandeep Madireddy, Prasanna Balaprakash, Philip Carns, Robert Latham, Robert Ross, Shane Snyder, and Stefan M Wild. Analysis and correlation of application i/o performance and system-wide i/o activity. In *2017 International Conference on Networking, Architecture, and Storage (NAS)*, pages 1–10. IEEE, 2017.

[43] Shikharesh Majumdar, Derek L Eager, and Richard B Bunt. Scheduling in multiprogrammed parallel systems. *ACM SIGMETRICS Performance Evaluation Review*, 16(1):104–113, 1988.

[44] Bashir Mohammed, Irfan Awan, Hassan Ugail, and Muhammad Younas. Failure prediction using machine learning in a virtualised hpc system and application. *Cluster Computing*, 22:471–485, 2019.

[45] Sarah Neuwirth, Feiyi Wang, Sarp Oral, and Ulrich Bruening. Automatic and transparent resource contention mitigation for improving large-scale parallel file system performance. In *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*, pages 604–613. IEEE, 2017.

[46] Sarp Oral, James Simmons, Jason Hill, Dustin Leverman, Feiyi Wang, Matt Ezell, Ross Miller, Douglas Fuller, Raghul Gunasekaran, Youngjae Kim, et al. Best practices and lessons learned from deploying and operating large-scale data-centric parallel file systems. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 217–228. IEEE, 2014.

[47] Sarp Oral, Sudharshan S Vazhkudai, Feiyi Wang, Christopher Zimmer, Christopher Brumgard, Jesse Hanley, George Markomanolis, Ross Miller, Dustin Leverman, Scott Atchley, et al. End-to-end i/o portfolio for the summit supercomputing ecosystem. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14, 2019.

[48] Tirthak Patel and Suren Byna. Uncovering access, reuse, and sharing characteristics of i/o-intensive files on large-scale production hpc systems. In *Proceedings of the 18th USENIX Conference on File and Storage Technologies, 2020*, 2020.

[49] 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.

[50] Tirthak Patel, Zhengchun Liu, Raj Kettimuthu, Paul Rich, William Allcock, and Devesh Tiwari. Job characteristics on large-scale systems: long-term analysis, quantification, and implications. In *SC20: International conference for high performance computing, networking, storage and analysis*, pages 1–17. IEEE, 2020.

[51] Arnab K. Paul, Arpit Goyal, Feiyi Wang, Sarp Oral, Ali R. Butt, Michael J. Brim, and Sangeetha B. Srinivasa. I/o load balancing for big data hpc applications. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 233–242, 2017.

[52] Jordan G Powers, Joseph B Klemp, William C Skamarock, Christopher A Davis, Jimy Dudhia, David O Gill, Janice L Coen, David J Gochis, Ravan Ahmadov, Steven E Peckham, et al. The weather research and forecasting model: Overview, system efforts, and future directions. *Bulletin of the American Meteorological Society*, 98(8):1717–1737, 2017.

[53] Zhenbo Qiao, Qing Liu, Norbert Podhorszki, Scott Klasky, and Jieyang Chen. Taming i/o variation on qos-less hpc storage: What can applications do? In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–13. IEEE, 2020.

[54] V Rajaraman. Frontier—world's first exaflops supercomputer. *Resonance*, 28(4):567–576, 2023.

[55] Neeraj Rajesh, Hariharan Devarajan, Jaime Cernuda Garcia, Keith Bateman, Luke Logan, Jie Ye, Anthony Kougkas, and Xian-He Sun. Apollo: An ml-assisted real-time storage resource observer. In *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing*, pages 147–159, 2021.

[56] WG Redis. Redis, 2016. http://redis.io/topics/faqAccessedNovember.

[57] Russ Rew and Glenn Davis. Netcdf: an interface for scientific data access. *IEEE computer graphics and applications*, 10(4):76–82, 1990.

[58] 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.

[59] Robert Ross, Lee Ward, Philip Carns, Gary Grider, Scott Klasky, Quincey Koziol, Glenn K Lockwood, Kathryn Mohror, Bradley Settlemyer, and Matthew Wolf. Storage systems and input/output: Organizing, storing, and accessing data for scientific discovery. report for the doe ascr workshop on storage systems and i/o.[full workshop report]. Technical report, USDOE Office of Science (SC)(United States), 2018.

[60] Bianca Schroeder and Garth A Gibson. A large-scale study of failures in high-performance computing systems. *IEEE transactions on Dependable and Secure Computing*, 7(4):337–350, 2009.

[61] Aamer Shah, Chih-Song Kuo, Akihiro Nomura, Satoshi Matsuoka, and Felix Wolf. How file-access patterns influence the degree of i/o interference between cluster applications. *Supercomputing Frontiers and Innovations*, 6(2):29–55, 2019.

[62] 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.

[63] Luka Stanisic and Klaus Reuter. Mpcdf hpc performance monitoring system: Enabling insight via job-specific analysis. In *Euro-Par 2019: Parallel Processing Workshops: Euro-Par 2019 International Workshops, Göttingen, Germany, August 26–30, 2019, Revised Selected Papers 25*, pages 613–625. Springer, 2020.

[64] Osamu Tatebe, Shukuko Moriwake, and Yoshihiro Oyama. Gfarm/bb — gfarm file system for node-local burst buffer. *Journal of Computer Science and Technology*, 35:61–71, 2020.

[65] Rajeev Thakur, William Gropp, and Ewing Lusk. Data sieving and collective i/o in romio. In *Proceedings. Frontiers' 99. Seventh Symposium on the Frontiers of Massively Parallel Computation*, pages 182–189. IEEE, 1999.

[66] Hendrik L Tolman et al. User manual and system documentation of wavewatch iii tm version 3.14. *Technical note, MMAB Contribution*, 276(220), 2009.

[67] James Turnbull. *The Logstash Book*. James Turnbull, 2013.

[68] Sudharshan S Vazhkudai, Ross Miller, Devesh Tiwari, Christopher Zimmer, Feiyi Wang, Sarp Oral, Raghul Gunasekaran, and Deryl Steinert. Guide: a scalable information directory service to collect, federate, and analyze logs for operational insights into a leadership hpc facility. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, Denver, 2017. IEEE.

[69] Bharti Wadhwa, Arnab K Paul, Sarah Neuwirth, Feiyi Wang, Sarp Oral, Ali R Butt, Jon Bernard, and Kirk W Cameron. iez: Resource contention aware load balancing for large-scale parallel file systems. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 610–620. IEEE, 2019.

[70] Feiyi Wang, Sarp Oral, Saurabh Gupta, Devesh Tiwari, and Sudharshan S Vazhkudai. Improving large-scale storage system performance via topology-aware and balanced data placement. In *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pages 656–663. IEEE, 2014.

[71] John C Warner, Brandy Armstrong, Ruoying He, and Joseph B Zambon. Development of a coupled ocean–atmosphere–wave–sediment transport (coawst) modeling system. *Ocean modelling*, 35(3):230–244, 2010.

[72] Steven A Wright, Simon D Hammond, Simon J Pennycook, Robert F Bird, JA Herdman, Ian Miller, Ash Vadgama, Abhir Bhalerao, and Stephen A Jarvis. Parallel file system analysis through application i/o tracing. *The Computer Journal*, 56(2):141–155, 2013.

[73] Bing Xie, Jeffrey Chase, David Dillow, Oleg Drokin, Scott Klasky, Sarp Oral, and Norbert Podhorszki. Characterizing output bottlenecks in a supercomputer. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE, 2012.

[74] Bing Xie, Sarp Oral, Christopher Zimmer, Jong Youl Choi, David Dillow, Scott Klasky, Jay Lofstead, Norbert Podhorszki, and Jeffrey S Chase. Characterizing output bottlenecks of a production supercomputer: Analysis and implications. *ACM Transactions on Storage (TOS)*, 15(4):1–39, 2020.

[75] Cong Xu, Suren Byna, Vishwanath Venkatesan, Robert Sisneros, Omkar Kulkarni, Mohamad Chaarawi, and Kalyana Chadalavada. Lioprof: Exposing lustre file system behavior for I/O middleware. In *Cray User Group Meeting*, pages 1–9, London, 2016. Cray.

[76] Bin Yang, Xu Ji, Xiaosong Ma, Xiyang Wang, Tianyu Zhang, et al. End-to-end i/o monitoring on a leading supercomputer. In *16th USENIX Symposium on Networked Systems Design and Implementation*, 2019.

[77] Bin Yang, Wei Xue, Tianyu Zhang, Shichao Liu, Xiaosong Ma, Xiyang Wang, and Weiguo Liu. End-to-end i/o monitoring on leading supercomputers. *ACM Transactions on Storage*, 19(1):1–35, 2023.

[78] Bin Yang, Yanliang Zou, Weiguo Liu, and Wei Xue. An end-to-end and adaptive i/o optimization tool for modern hpc storage systems. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1294–1304. IEEE, 2022.

[79] Wenxiang Yang, Xiangke Liao, Dezun Dong, and Jie Yu. A quantitative study of the spatiotemporal i/o burstiness of hpc application. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1349–1359. IEEE, 2022.

[80] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60. Springer, 2003.

[81] Jie Yu, Guangming Liu, Wenrui Dong, Xiaoyong Li, Jian Zhang, and Fuxing Sun. On the load imbalance problem of i/o forwarding layer in hpc systems. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pages 2424–2428. IEEE, 2017.

[82] Yulai Yuan, Guangwen Yang, Yongwei Wu, and Weimin Zheng. Pv-easy: a strict fairness guaranteed and prediction enabled scheduler in parallel job scheduling. In *proceedings of the 19th ACM International symposium on high performance distributed computing*, pages 240–251, 2010.

[83] Chris Zimmer. Summit burst buffer, 2018.