# CATIOS: Time-Resolved I/O-Aware Job Scheduling for HPC Systems

YuTsen Tseng
Graduate School of Information Sciences,
Tohoku University
6-6-01 Aramaki-aza-aoba, Aoba, Sendai 980-8579, Japan
tseng.yu.tsen.s4@dc.tohoku.ac.jp

Masatoshi Kawai
Graduate School of Information Sciences,
Tohoku University
6-6-01 Aramaki-aza-aoba, Aoba, Sendai 980-8579, Japan
m.kawai@tohoku.ac.jp

Keichi Takahashi
D3 Center
University of Osaka
Osaka, Japan
takahashi.d3c@osaka-u.ac.jp

Hiroyuki Takizawa
Graduate School of Information Sciences,
Tohoku University
6-6-01 Aramaki-aza-aoba, Aoba, Sendai 980-8579, Japan
takizawa@tohoku.ac.jp

## 1 Introduction

High-performance computing (HPC) systems support large-scale compute-intensive workloads, but as modern jobs become increasingly data-intensive, I/O bandwidth has become as critical as compute resources. This prompts research of I/O-aware job scheduling [1, 6], which limits the aggregate I/O demand of jobs within the maximum bandwidth of the file systems, unlike traditional compute-centric methods. However, most existing approaches rely on static or average-based bandwidth estimates, failing to capture bursty workloads with alternating compute and I/O phases. Analysis of the Blue Waters datasets [3–5] reveals that I/O bursts during operations often coincide across concurrent jobs. Average-based estimation obscures I/O bursts, causing I/O contention and runtime unpredictability when demand is underestimated, and idle resources when overestimated.

In this work, *Contention-Avoiding Temporal I/O-aware job Scheduling (CATIOS)* is proposed to address these issues by incorporating time-resolved contention-aware simulation into scheduling decisions, requiring only job scripts as input. By anticipating the timing and magnitude of bursts, CATIOS coordinates job execution to improve throughput and predictability beyond average-based approaches.

## 2 The CATIOS Framework

CATIOS models each job as a sequence of time-ordered I/O intervals, enabling the scheduler to anticipate and avoid overlapping bursts through simulation. Since incoming job profiles are unknown upon submission, CATIOS first infers them by matching them against historical records. The job's command line is embedded with a Sentence-BERT model and compared semantically to historical embeddings to identify candidate jobs. Among these, the one with the closest processor count is selected to ensure a comparable execution scale, and its job profile is used as a proxy.

Jobs are then entered into a pending queue and prioritized under one of three policies: FCFS (First-Come, First-Served) preserves submission order to ensure fairness; LJF (Longest Job First) prioritizes longest runtime jobs to improve throughput; CATIOS-Mix assigns each job a weighted score derived from normalized wait time and predicted runtime, balancing fairness and throughput.

Finally, pending jobs are evaluated sequentially through time-resolved simulation, tracking cumulative I/O activity in one-second intervals. A job is admitted only if its execution does not cause bandwidth usage to exceed system limits in any interval; otherwise, it is deferred, and the next priority job is considered. Meanwhile, to mitigate the influence of outliers, individual per-second demands are clipped at 75% of maximum system bandwidth in the simulation. The bandwidth timeline is updated as jobs are admitted or completed, ensuring accurate forecasts. By combining semantic profile inference, flexible prioritization, and predictive admission control, CATIOS orchestrates job timing to avoid contention while maintaining high utilization.

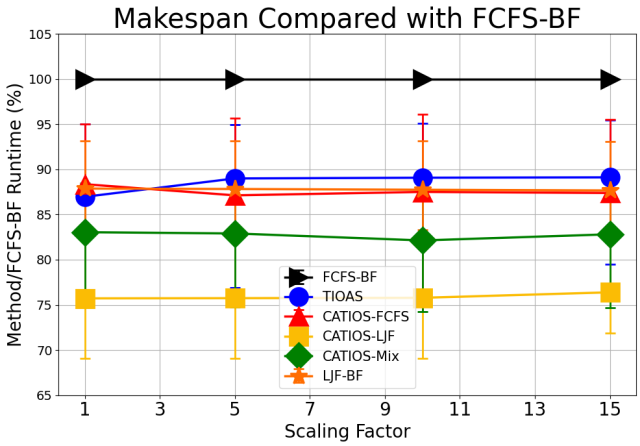
## 3 Evaluation and Discussion

CATIOS is evaluated through simulations extended from the SimGrid [2] platform. The simulation models a shared file system with a shared buffer and global bandwidth, using workloads from the Blue Waters dataset reproduced via Darshan logs and converted into per-second I/O demand profiles. Each run includes five batches of 500 jobs with diverse runtimes, scales, and I/O characteristics. Jobs are mapped to homogeneous compute nodes, with I/O behavior derived from historical traces and submitted at five-second intervals.

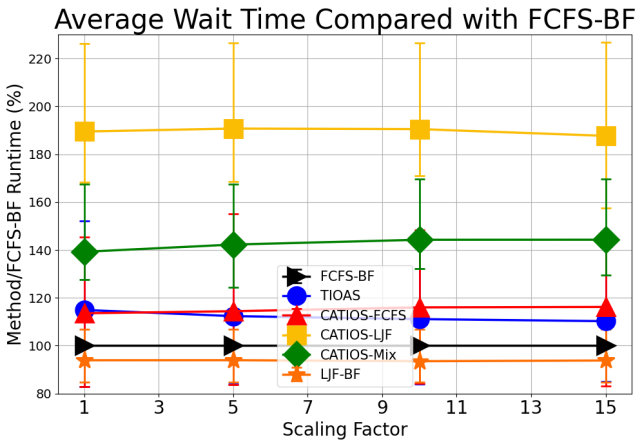
Performance is measured by makespan (time from first job start to last completion) and average wait time (mean queue delay), normalized against an I/O-unaware FCFS with backfilling (FCFS-BF) baseline. Robustness is tested by proportionally scaling all jobs' I/O demand. As the scaling factor increases, a decline in normalized values indicates slower degradation relative to FCFS-BF, an increase indicates faster degradation, and a stable trend suggests a degradation rate comparable to the baseline.

The comparison includes FCFS and LJF with backfilling (FCFS-BF and LJF-BF), a traditional I/O-aware job scheduling (TIOAS), and three CATIOS variants: CATIOS-FCFS, CATIOS-LJF, and CATIOS-Mix. TIOAS represents a class of existing static I/O-aware job scheduling [1, 6] that preserves submission order during scheduling, and admits jobs when the aggregated average I/O demand doesn't exceed system capacity.

The result shows that CATIOS-FCFS sustains a stable makespan even under heavy I/O load, while TIOAS degrades. Meanwhile,




**(a) Makespan**




**(b) Average Wait Time**

**Figure 1: Job Scheduling Methods Performance Comparison.**

CATIOS-LJF achieves the most considerable makespan reduction, though at the cost of markedly higher wait times for short jobs. On the other hand, CATIOS-Mix balances throughput improvement with limited wait time growth. Across all load levels, CATIOS prevents underutilization and slowdowns caused by overlapping bursts, ensuring predictable system performance. Quantitatively, CATIOS achieved a trade-off ratio of approximately 1:1.3 between makespan reduction and average wait time increase, demonstrating an effective balance between fairness and throughput.

## 4 Conclusion

This work presents the CATIOS framework, which integrates command-based job matching, configurable scheduling policies, and time-resolved contention-aware simulation, enabling proactive scheduling that avoids overlapping bursts.

Simulation results demonstrate that CATIOS reduces makespan, sustains stable average wait times under high load, and outperforms baselines. These findings reveal its capability to enhance data-intensive HPC systems and its ability to balance fairness and throughput as objectives evolve. Future work includes addressing multi-resource contention and improving profile matching accuracy.

## References


[1] Guillaume Aupy, Ana Gainaru, and Valentin Le Fèvre. 2021. Periodic I/O scheduling for supercomputers. In *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '21)*. ACM, 141–152. https://doi.org/10.1145/3431379.3460653

[2] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter. 2014. Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms. *J. Parallel and Distrib. Comput.* 74, 10 (2014), 2899–2917. https://doi.org/10.1016/j.jpdc.2014.06.008

[3] National Center for Supercomputing Applications. 2013. Blue Waters Supercomputer Project. https://bluewaters.ncsa.illinois.edu.

[4] Arnab K. Paul, Olaf Faaland, Adam Moody, Elsa Gonsiorowski, Kathryn Mohror, and Ali R. Butt. 2020. Understanding HPC Application I/O Behavior Using System Level Statistics. In *27th IEEE International Conference on High Performance Computing, Data, and Analytics (HiPC 2020)*. 202–211. https://doi.org/10.1109/HiPC50609.2020.00034

[5] Ehsan Saeedizade, Roya Taheri, and Engin Arslan. 2023. I/O Burst Prediction for HPC Clusters using Darshan Logs. https://doi.org/10.48550/arXiv.2308.10311

[6] Michela Taufer. 2021. AI4IO: A Suite of Ai-Based Tools for IO-Aware HPC Resource Management. In *2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC)*. 1–1. https://doi.org/10.1109/HiPC53243.2021.00012