WEB SUPPLEMENTARY MATERIAL FOR "IO-SETS: Simple and efficient approaches for I/O bandwidth management"

Francieli Boito, Guillaume Pallez, Luan Teylo and Nicolas Vidal

A. DESCRIPTION OF VARIABLES

Table 1 presents an overview of the variables used in this paper and their meaning.

Table 1: Description of Variables

Variable	Description
w_{iter}	Characteristic time representing the average time between the beginning of two consecutive I/O phases of an application.
α_{io}	The I/O ratio of a job.
μ	The mean value used to draw the characteristic time w_{iter} for each job. It is selected on an experiment basis.
σ	The standard deviation used to draw the characteristic time w_{iter} for each job. It is selected on an experiment basis.
b	The bound parameter representing machine variability for processing and I/O performance. It is used to draw the variables $\gamma_{\text{cpu}}^{(i)}$ and $\gamma_{\text{io}}^{(i)}$.
h	The time horizon for which applications are executed. All applications execute for a time that is long enough (a horizon h) to reach the steady state evaluation of the system.
ω	The target average I/O stress for each workload scenario. It represents the average I/O load and takes values of 0.8 for a saturated system and 0.2 for a light I/O stress.
N	The number of parallel applications in each workload.
Γ	The release time for the first complete iteration of each job. It represents the amount of work left from the previous iteration and is drawn uniformly at random in the interval $[0, w_{\text{iter}}]$.
t_{cpu}	The average compute phase length for a job.
t_{io}	The average I/O phase length for a job.
$\gamma_{\text{cpu}}^{(i)}$	The noise parameter representing machine variability in processing performance for the i-th iteration of a job.
$\gamma_{\text{io}}^{(i)}$	The noise parameter representing machine variability in I/O performance for the i-th iteration of a job.
$t_{\text{cpu}}^{(i)}$	The compute phase length for the i-th iteration of a job, considering machine variability. It is calculated as $(1+\gamma_{\text{cpu}}^{(i)}) \cdot t_{\text{cpu}}$, where t_{cpu} is the average compute phase length.
$t_{\text{io}}^{(i)}$	The I/O phase length for the i-th iteration of a job, considering machine variability. It is calculated as $(1+\gamma_{\text{io}}^{(i)}) \cdot t_{\text{io}}$, where t_{io} is the average I/O phase length.
T_{begin}	The beginning time of the measurement interval.
T_{end}	The end time of the measurement interval.
e_j	The effective completed work for task J_j. It is calculated as the sum of the lengths of all compute and I/O phases executed by the task within the measurement interval, represented as $e_j = e_j^{cpu} + e_j^{io}$.
e_j^{iter}	The effective iterations performed by task J_j within the measurement interval. It represents the number of iterations executed by the task inside the time frame.
n_H	The number of high-frequency jobs. These jobs have a characteristic time w_{iter} drawn from a normal distribution $\mathcal{N}(10, 1)$.
n_M	The number of medium-frequency jobs. These jobs have a characteristic time w_{iter} drawn from a normal distribution $\mathcal{N}(100, 10)$.
n_L	The number of low-frequency jobs. These jobs have a characteristic time w_{iter} drawn from a normal distribution $\mathcal{N}(1000, 100)$.

B. I/O TRACES ANALYSIS

To evaluate the applicability of IO-SETS considering real data, we utilized traces collected from PlaFRIM, the experimental cluster at INRIA Bordeaux. These traces cover a one-year period from May 2022 to May 2023 and were obtained by combining information about jobs — from the resource manager (Slurm) — and about instantaneous usage of the BeeGFS file system, given by beegfs-ctl. Our traces contain, for each job, the number of requests and bandwidth of read and write operations for every second of its execution.

After analyzing the collected data, we identified a total of 53,413 jobs[1]. To characterize the I/O behavior of these applications, we applied the FTIO method [21], which provides information such as the number of I/O phases, the mean time between phases, and the amount of data transferred per phase. By utilizing FTIO, we are able to define the w_{iter}

1. We only consider jobs that appear on beegfs-ctl. Therefore, the total number of jobs only includes applications that performed I/O operations on the PFS.

- *FB and LT are with Univ. Bordeaux, CNRS, Bordeaux INP, INRIA and LaBRI. GP is with Inria. NV is with Oak Ridge National Laboratory.*
- *e-mail: {francieli.zanon-boito, guillaume.pallez, luan.teylo}@inria.fr, vidaln@ornl.gov*

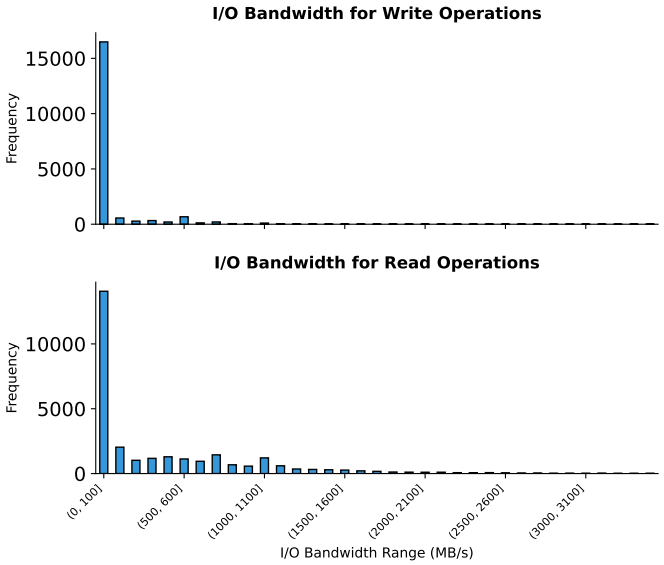

Figure 1: Distribution of I/O bandwidth in the job traces collected from PlaFRIM, considering read and write operations. The figure shows that the vast majority of jobs in our dataset had an I/O bandwidth of less than 100 MB/s.

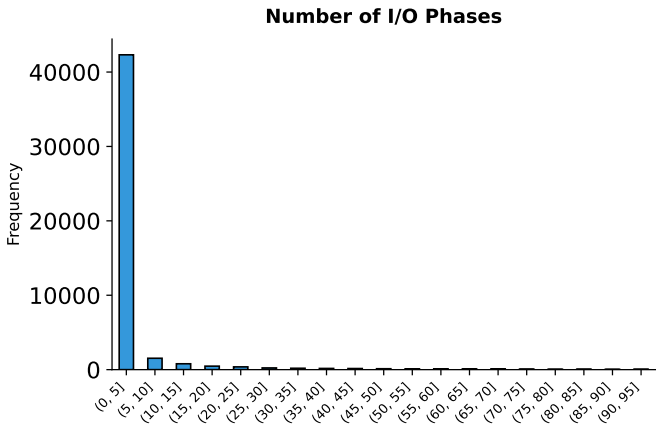

Figure 2: Histogram of the number of I/O phases in the job traces collected from PlaFRIM. The figure shows that the vast majority of jobs in our dataset have up to 5 I/O phases.

parameter, which is used to determine the applications' sets (see Section 3.2).

We applied filters to exclude jobs that exhibited an I/O bandwidth greater than 16 GBps, a value significantly exceeding our platform's capacities [20]. Sometimes a very short I/O phase may affect the capacity of beegfs-ctl to estimate the correct I/O bandwidth. Additionally, we also excluded all jobs originating from our team, as they primarily consisted of I/O benchmark tools (most with periodic behaviors) and may not accurately represent the rest of the workload. After implementing these filters, 48,582 jobs remained. Considering these, we aim to answer the following questions:

- What are the characteristics of our dataset in terms of I/O?
- Do the applications in our dataset belong to distinct sets as defined by Equation (3)?
- Among applications executing at the same period of time, how many of them belong to distinct sets?
- Can we find scenarios from the traces that are relevant as test cases for the proposed method?

I/O workload characterization

Firstly, we analyzed the average application's I/O bandwidth. Figure 1 presents a histogram of the bandwidth in our dataset, considering both write and read operations. As shown, the majority of the applications had an I/O bandwidth below 100 MB/s (62.88%). In fact, 30.2% of the I/O operations in the traces have a bandwidth below 10 MB/s. These results indicate that the majority of jobs on our platform are not I/O intensive. This aligns with our expectations since PlaFRIM is predominantly used by CPU-bound applications. Moreover, its storage infrastructure largely exceeds its I/O speed needs.

Next, we analyzed the number of I/O phases in the jobs. As shown in Figure 2, 42,316 jobs (87.10%) in our dataset have up to 5 I/O phases. A closer look at the number of phases reveals that 75.36% of the applications executed in PlaFRIM over the course of one year had only one I/O phase.

To proceed with the analysis, we chose to consider only applications with an I/O bandwidth greater than 100 MB/s and with more than one I/O phase, a total of 4,088 jobs. Among these applications, we observed that 1,860 of them have more than 10 I/O phases, indicating a pattern of periodic I/O behavior: in average, these I/O-intensive jobs has 52.97 phases, with an average I/O volume of 366.60 MB per phase. These observations support our assumption that I/O-bound applications tend to exhibit periodic behavior.

Do we have multiple sets?

We then classified our 4,088 jobs in sets according to their w_{iter}. The results were presented in Section 8. It is important to notice we excluded the jobs for which FTIO detected a single I/O phase because they do not necessarily all have a single I/O phase. Indeed, by Nyquist, with a sampling period of 1 second we cannot detect I/O phases that happen more often than once every 2 seconds. Therefore, these one-phase jobs could either be in higher sets (when w_{iter} is the execution time, and this time is long) or in the first set (for a $w_{\text{iter}} < 2$). Because we have no way of knowing how many of them correspond to each situation, we decided to not count them. Still, our data shows that jobs cover multiple sets.

Next, we aim to answer the question: "Do jobs executing at a given period of time belong to distinct sets?". To achieve this, we extracted what we refer to as "execution cases" from the traces. These cases represent instances where at least two of our 4,088 jobs were executed within a specific time frame. We identified 652 execution cases using a time window of 5 hours. The prevalence of multiple sets in there execution cases was discussed in Section 8.

The search for test cases

Considering the execution cases, we analyzed their I/O stress, as defined in Equation (2). Figure 3 presents a box plot summarizing the I/O stress of all 652 execution cases considered in this section. The box plot reveals a median value of 0.07, indicating a tendency towards very low I/O stress in PlaFRIM. Furthermore, the cases with stress levels around 0.6 are considered outliers, suggesting that high I/O stress is not the norm in PlaFRIM. In fact, only 5.89% of the execution cases (192 cases) exhibit an I/O stress greater than 0.6.

It is important to notice even this low I/O stress is somewhat overestimated: we define the execution cases by taking jobs that executed within the same 5-hour time window, but they are not necessarily always concurrent. For an example, see Figure 4.

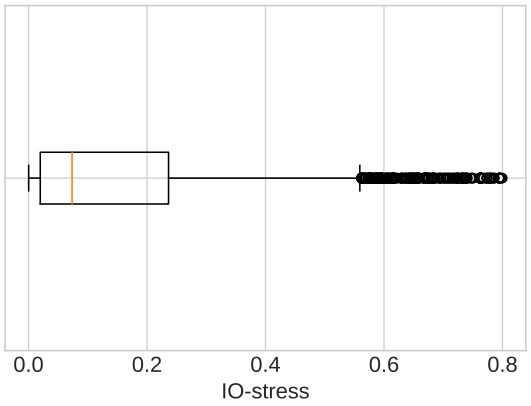

Figure 3: The distribution of the I/O stress in the 652 execution cases found in the traces considering execution periods of 5-hours.

These data demonstrate that when multiple applications run simultaneously on PlaFRIM, they do not cause significant I/O stress on the platform. In other words, I/O congestion does not occur on our platform. Therefore, these traces are not suitable for evaluating I/O scheduler techniques since there is no room for improving I/O performance based on this data.

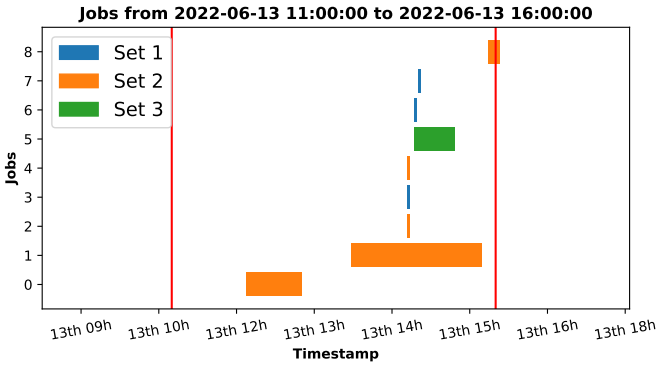

Figure 4: An example of an execution case considering a 5-hour time window (represented by the red lines). In this example, 9 jobs with distinct durations were found. The colors represent the set each job belongs to.

Synthetic simulations

Because the collected data is not adequate for a performance evaluation, we aimed to created a synthetic scenario with enough I/O congestion, but that still represents the PlaFRIM workload. For that, we conducted tests similar to those described in Section 6, but using the set distribution observed in PlaFRIM (presented in Section 8).

In order to generate the applications, we followed the protocol described in Section 5.2, with a horizon h = 30,000s, an I/O stress $\omega = 0.8$, and the metrics were computed using a time frame of 8,000s ($T_{begin} = 12,000$ and $T_{end} = 20,0000$). We considered 100 jobs, and since we are considering the six sets presented in our dataset (from set 0 to 5), we define the characteristic time (w_{iter}) using the following job profiles:

- S_0 has 8 jobs with a characteristic time $w_{\text{iter}} \sim \mathcal{N}(1, 0.1)$;
- S_1 has 45 jobs with a characteristic time $w_{\text{iter}} \sim \mathcal{N}(10, 1)$;
- S_2 has 33 jobs with a characteristic time $w_{\text{iter}} \sim \mathcal{N}(100, 10)$;
- S_3 has 11 jobs with a characteristic time $w_{\text{iter}} \sim \mathcal{N}(1000, 100)$;
- S_4 has 2 jobs with a characteristic time $w_{\text{iter}} \sim \mathcal{N}(10000, 1000)$;
- S_5 has 1 job with a characteristic time $w_{\text{iter}} \sim \mathcal{N}(100000, 10000)$.

Figure 5 presents the results considering the relative `IO-slowdown`. As can be seen, SET-10 outperforms FAIR-SHARE and EXCLUSIVE-FCFS, achieving an improvement of around 20% compared to FAIR-SHARE. These results are similar to the ones presented in Section 6.2. Therefore, SET-10 achieves better results even when considering execution cases with more jobs and more sets.

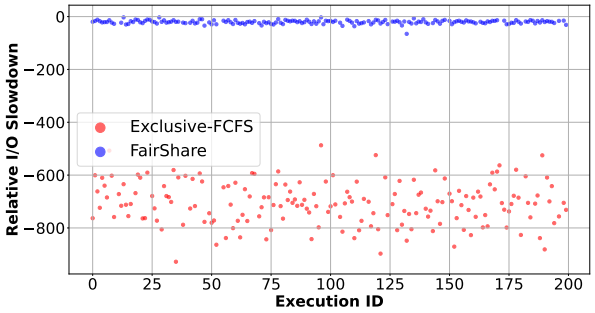

Figure 5: Relative `IO-slowdown` (the higher, the better) showing the relative loss of FAIR-SHARE and EXCLUSIVE-FCFS to SET-10 considering synthetic execution cases.

C. ADDITIONAL RESULTS

This appendix extends the results presented in Section 6 of the paper. In Section 1, we present the graphs that were suppressed due to the lack of space in the submitted version of the paper. Moreover, we also present and discuss additional tests that were conducted during the work. Section 2 present results for a workload with 15 tasks ($N = 15$), while Section 3 shown the tests considering a light I/O scenario with ω =0.2.

1 EXTRA GRAPHS FOR WORKLOAD WITH $N = 60$ AND HEAVY I/O LOAD (ω =0.8)

As we pointed out in Section 6.2.1, in the study of the impact of the set mapping and the priority assignment, the differences in terms of `Utilization` and `IO-slowdown` between the heuristic are not evident as the `Max Stretch`. We see in Figure 6 that the importance of the sets and the bandwidth priority are still apparent. However, the actual contribution of the sets, i.e., application fairness, cannot be seen with the other metrics. In any way, all metrics confirm that both parts of SET-10 are complementary and necessary for the results.

Figure 7 presents all metrics related to the study where introduced variability in the phases' duration (reported in Figure 8). We see that the variability in the iteration length shows no impact in any of the heuristics, independently of the considered metric.

Figure 8 shows the results for the cases where the sets are not well-defined. Once again, the graphs present a stable behaviour independently of the metric. We also see that in terms of `Max Stretch`, the difference between SET-10 and FAIR-SHARE narrows. These results show that some applications are impacted when the sets are not well-defined. Still, when we look at the `Utilization` and `IO-slowdown`, we see that, on average, SET-10 presents better results.

Figure 9 shows the results considering all metrics for the comparison of the strategies with the addition of an extra application profile. It reinforces the conclusions presented in the paper: the mapping function used by SET-10 is robust and behaves consistently in all cases.

2 WORKLOADS WITH 15 APPLICATIONS AND HEAVY I/O (ω =0.8)

We also evaluate the algorithms considering workloads with 15 jobs. We set $n_H + n_M + n_L = 15$ and conducted the same studies, but for $n_M = 5$, $n_H \in \{0, \ldots, 10\}$, and $n_L = 5 - n_H$. Once again, we use $b = 0.1$ and consider the same three job profiles used in Section 6.

Figure 10 show that with 15 applications, in general, SET-10 has the same behavior as before: it is, on average, the best strategy, principally when we consider the `IO-slowdown`. However, in terms of `Max Stretch`, the difference between SET-10 and FAIR-SHARE narrows. That occurs because, with 15 applications, less concurrency happens during the I/O phases, which is particularly good for FAIR-SHARE. But we still see the benefits for SET-10, especially for the I/O performance. For the noise evaluation, with $n_H = n_M = n_L = 5$, present in Figure 11, we see the same results as before: w_{iter} is robust enough independent of the workload size.

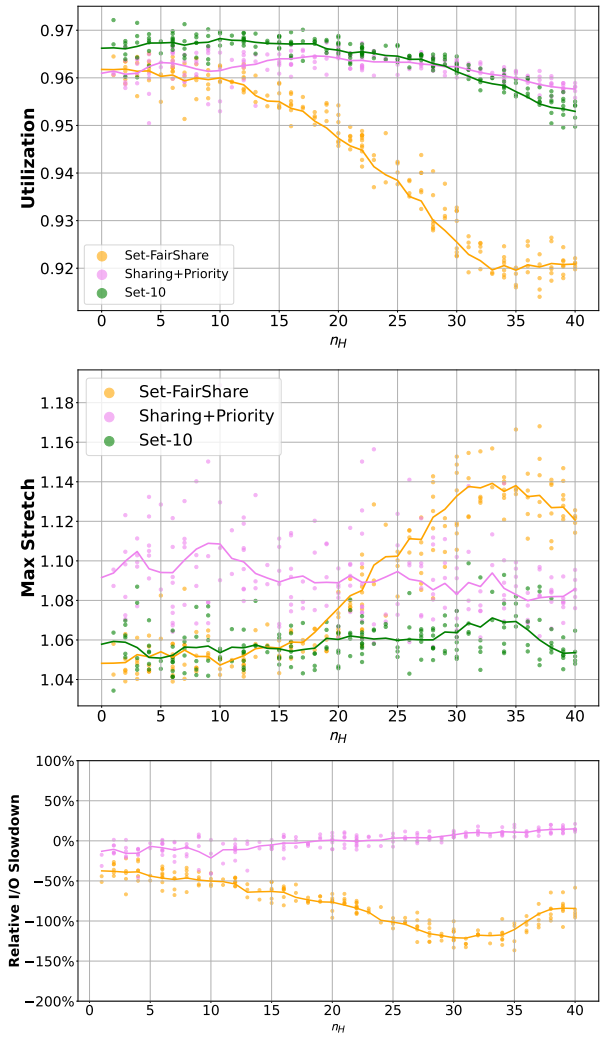

Figure 6: All results of the study of the set mapping and priority assignment parts of SET-10 presented in Figure 7. The relative `IO-slowdown` shows the relative loss of SET-FAIRSHARE and SHARE+PRIORITY to SET-10.

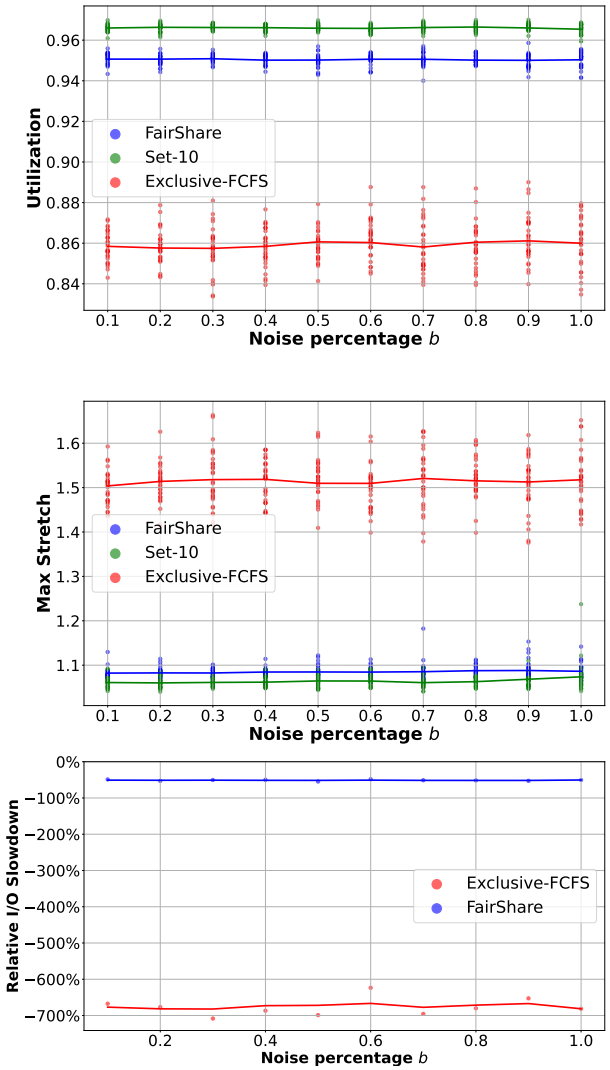

Figure 7: `Utilization`, `Max Stretch` and `IO-slowdown` according to the percentage of noise introduced in phases' length for the evaluation presented in Figure 8. The relative `IO-slowdown` shows the relative loss of FAIR-SHARE and EXCLUSIVE-FCFS to SET-10.

3 WORKLOAD WITH $N = 60$ AND LIGHT I/O (ω =0.2)

We also evaluated a light I/O scenario with an I/O load of 0.2. Figure 12 presents the results for all the evaluated metrics. Unlike the cases with heavy I/O (ω =0.8), there is almost no difference between SET-10 and FAIR-SHARE in a scenario with almost no I/O. Therefore, even in cases where there are no advantages for SET-10 (due to the lack of I/O), the heuristic is still as good as FAIR-SHARE.

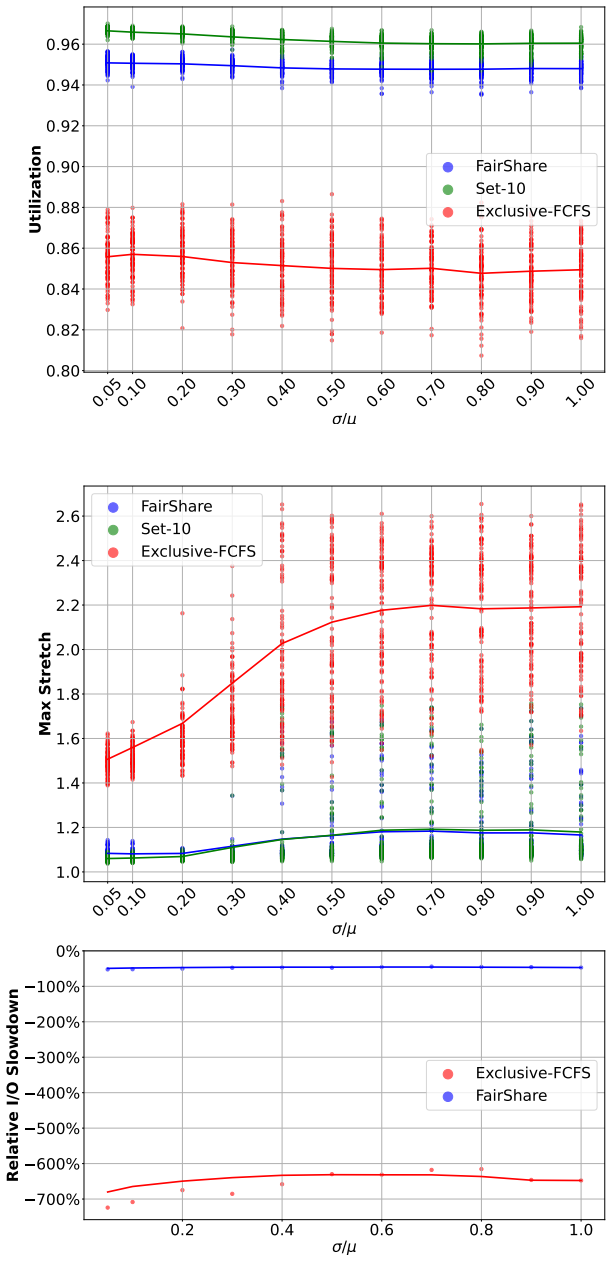

Figure 8: Evaluation of SET-10 for the cases where sets are not well-defined. The graph shows `Utilization`, `Max Stretch` and `IO-slowdown` according to the increment of the standard deviation used for the workloads. The graph complements the results presented in Figure 9. The relative `IO-slowdown` shows the relative loss of FAIR-SHARE and EXCLUSIVE-FCFS to SET-10.

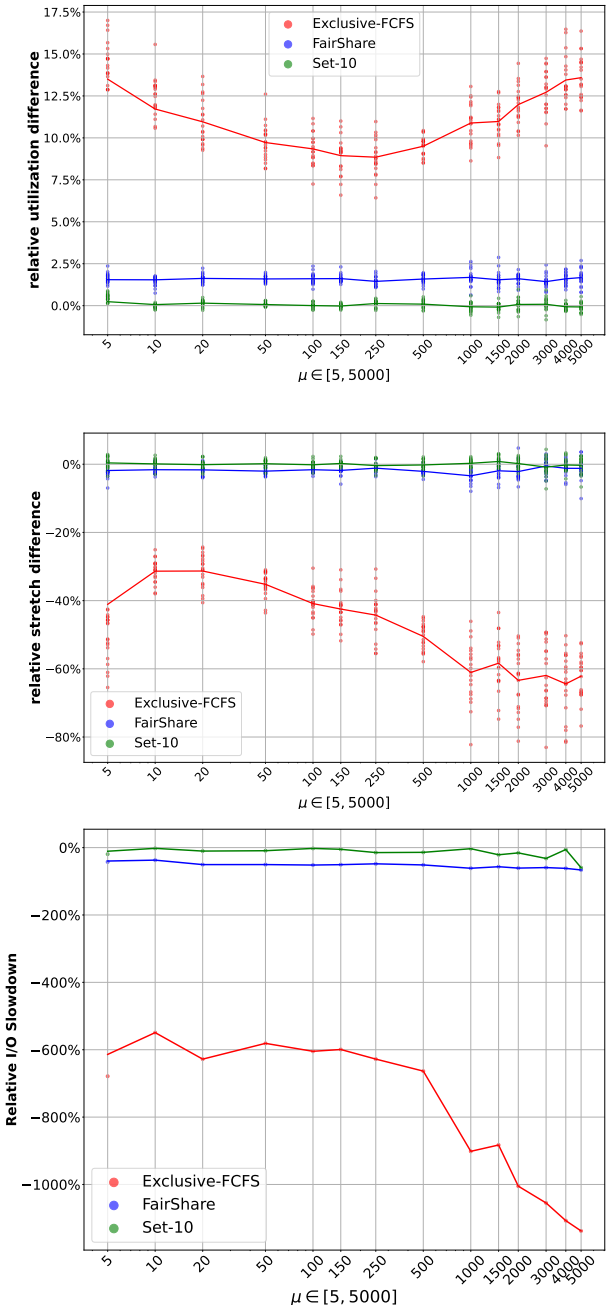

Figure 9: Comparison of the strategies with the addition of a fourth application profile (x-axis). Results are normalized by the CLAIRVOYANT ones. The graphs present the `Utilization`, `Max Stretch` and `IO-slowdown`, complementing the results shown in Figure 10

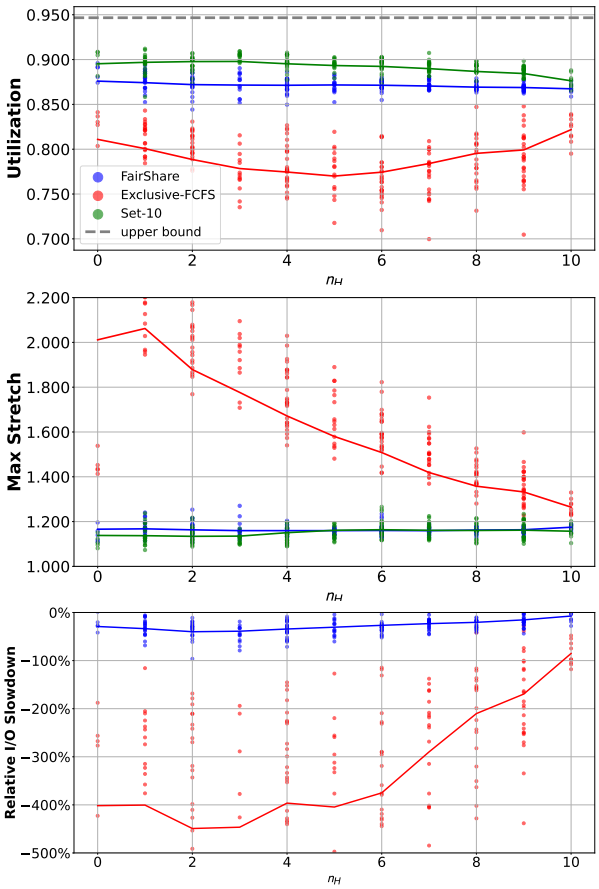

Figure 10: Comparison of the policies over the studied objectives considering the workload with $N = 15$. The relative `IO-slowdown` shows the relative loss of FAIR-SHARE and EXCLUSIVE-FCFS to SET-10.

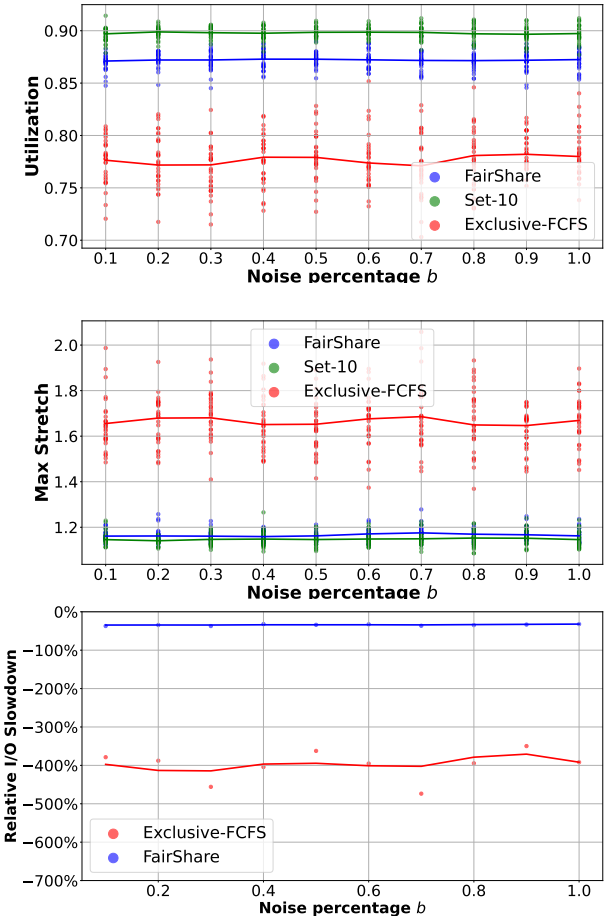

Figure 11: `Utilization`, `Max Stretch` and `IO-slowdown` according to the percentage of noise introduced in phases' length for $N = 15$. The relative `IO-slowdown` shows the relative loss of FAIR-SHARE and EXCLUSIVE-FCFS to SET-10.

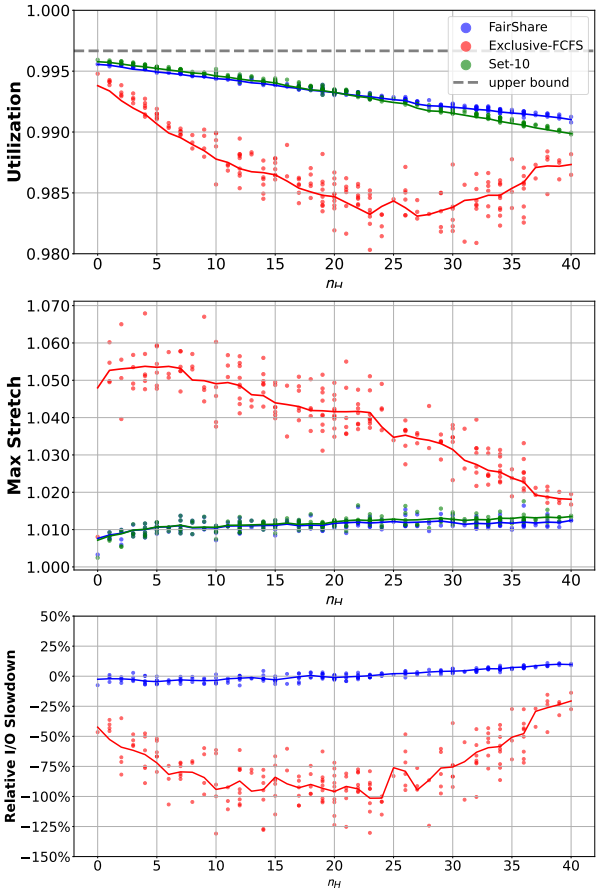

Figure 12: Comparison of the policies over the studied objectives considering the workload with $N = 60$ and light I/O load (ω =0.2). The relative `IO-slowdown` shows the relative loss of FAIR-SHARE and EXCLUSIVE-FCFS to SET-10.