Scheduling the I/O of HPC applications under congestion

Ana Gainaru, Guillaume Aupy, Anne Benoit,
Yves Robert, Franck Cappello & Marc Snir

IPDPS - May 2015

Interconnect technologies: A major challenge

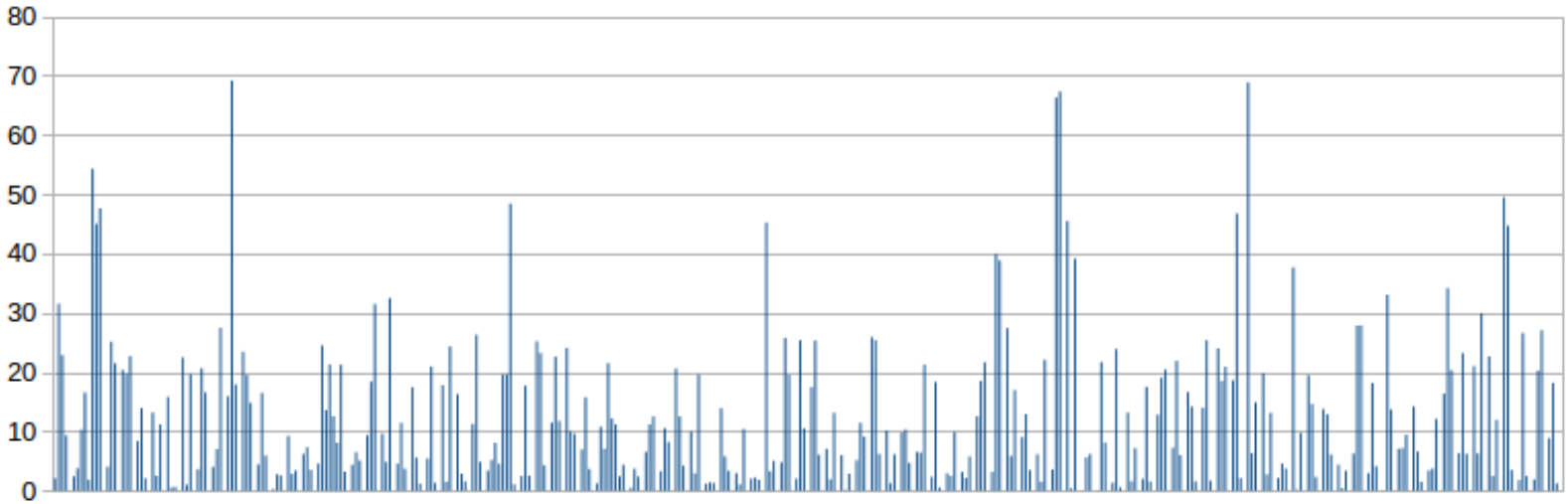
Application interference - Performance degradation due to applications accessing a shared resource (PFS)

Some app observe up to 14x decrease of performance due to interference

The challenge:

Flops are "free", we need to optimize data-movement!

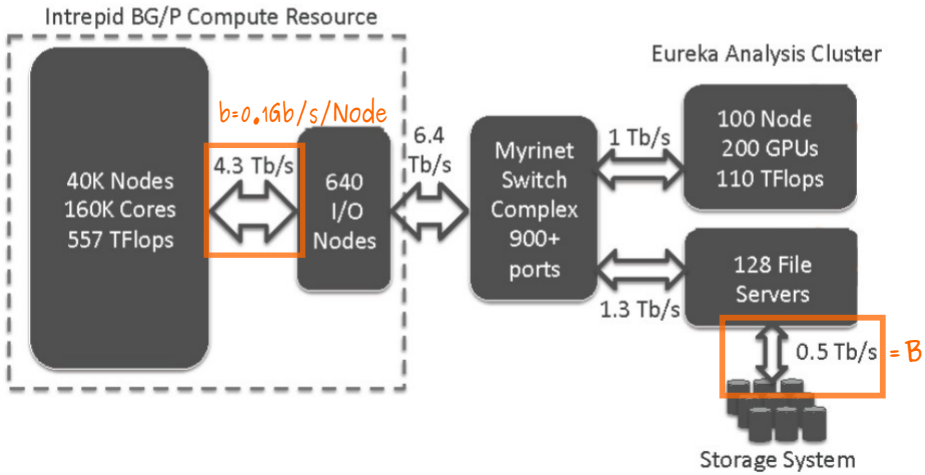
Interconnect technologies: A major challenge



Analysis of the Intrepid system @Argonne: I/O throughput decrease (percentage per application, over 400 applications).

Platform

- N unit-speed processors, equipped with an I/O card of bandwidth b
- Centralized I/O system with total bandwidth B



Model instantiation for the Intrepid platform.

Applications

K applications competing for I/O. For application $\text{App}^{(k)}$:

- Released at time r_k;
- Executed on $\beta^{(k)}$ procs;
- $n_{\text{tot}}^{(k)}$ *instances*: $\mathcal{I}_i^{(k)}$ consists of $w^{(k,i)}$ units of computation followed by the transfer of a volume $\text{vol}_{\text{io}}^{(k,i)}$;
- The minimum time to execute $\text{vol}_{\text{io}}^{(k,i)}$ is:

$$\text{time}_{\text{io}}^{(k,i)} = \frac{\text{vol}_{\text{io}}^{(k,i)}}{\min(\beta^{(k)} b, B)};$$

- Last instance finishes at time d_k.

Applications

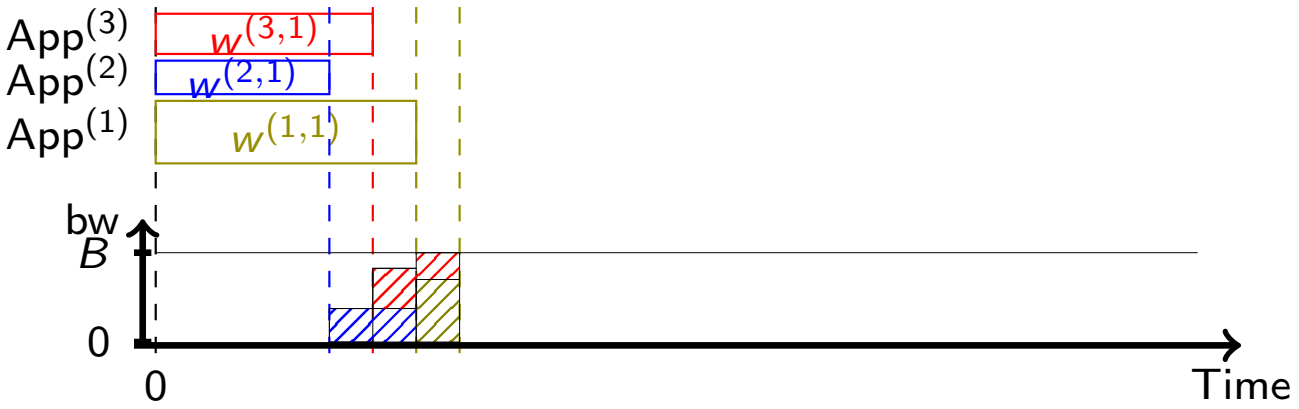
Applications

Applications

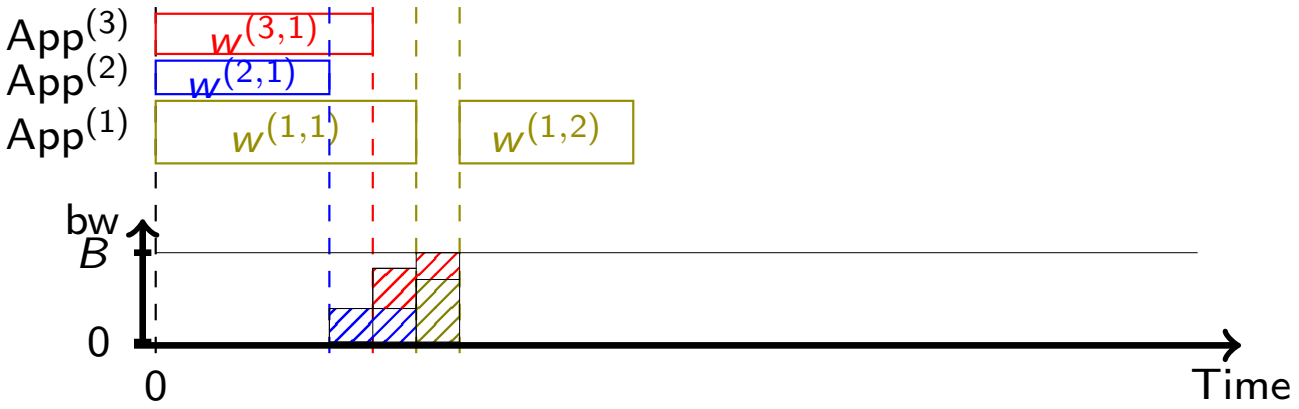
Applications

Applications

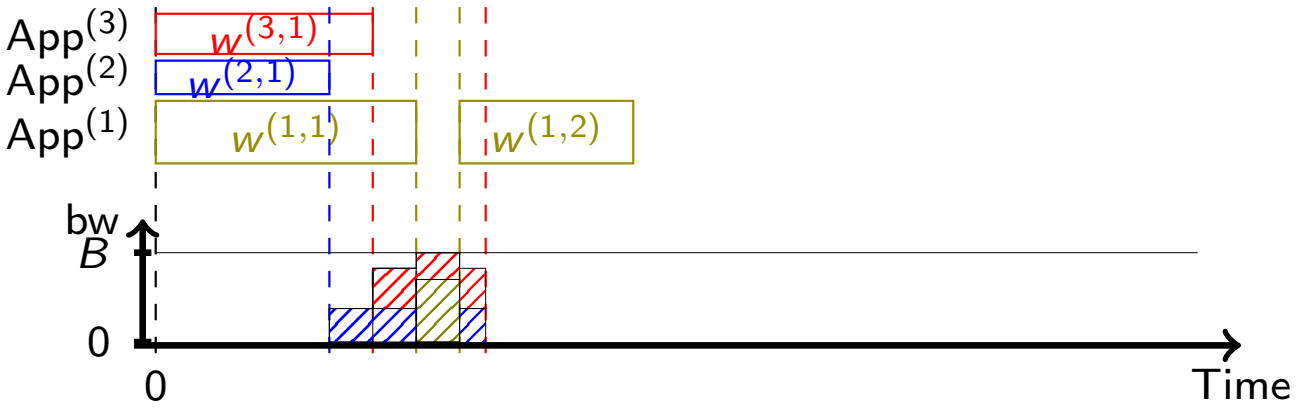
Applications



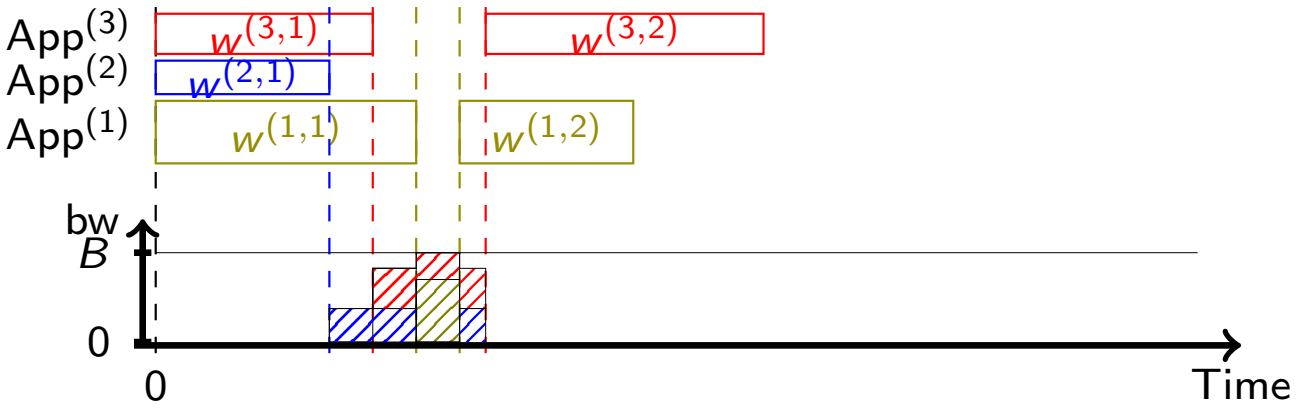
Applications



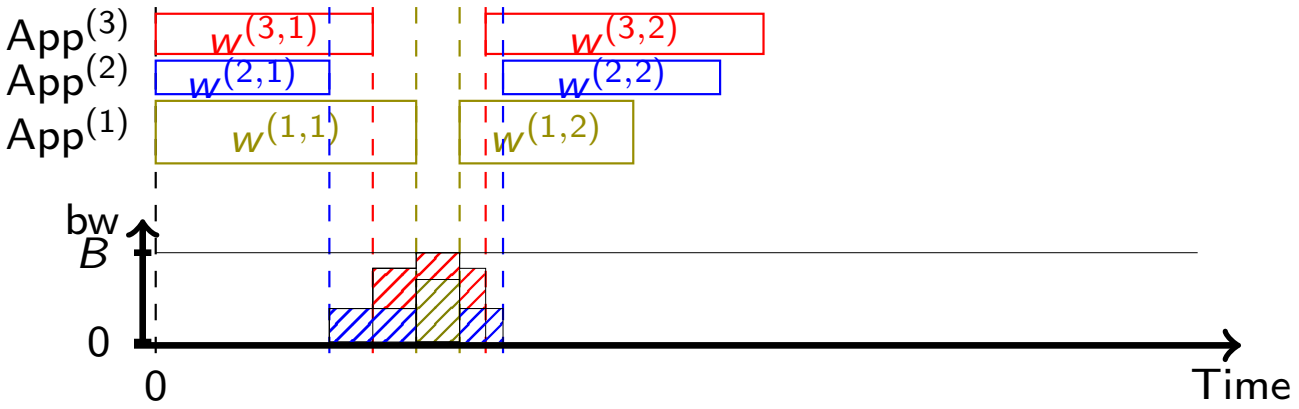
Applications



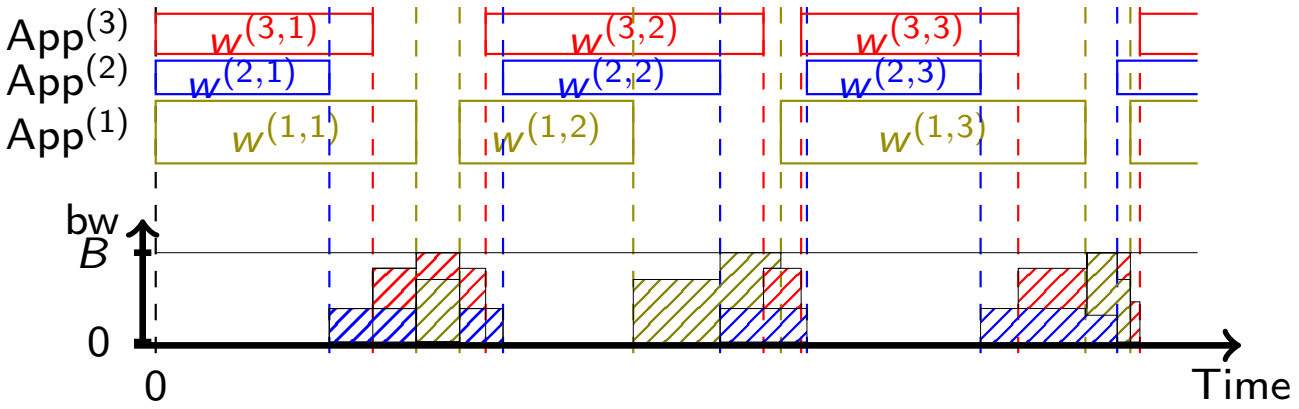
Applications



Applications



Applications



Objectives

Definition (Application efficiency)

$$\tilde{\rho}^{(k)}(t) = \frac{\sum_{i \leq n^{(k)}(t)} w^{(k,i)}}{t - r_k},$$

where $n^{(k)}(t)$ is the number of instances of $\mathrm{App}^{(k)}$ executed at time t.

Objectives

Definition (Application efficiency)

$$\tilde{\rho}^{(k)}(t) = \frac{\sum_{i \leq n^{(k)}(t)} w^{(k,i)}}{t - r_k},$$

where $n^{(k)}(t)$ is the number of instances of $\text{App}^{(k)}$ executed at time t.

Obviously: $t - r_k \geq \sum_{i \leq n^{(k)}(t)} \left(w^{(k,i)} + \text{time}_{\text{io}}^{(k,i)} \right)$.

Hence:

$$\tilde{\rho}^{(k)}(t) \leq \rho^{(k)}(t) = \frac{\sum_{i \leq n^{(k)}(t)} w^{(k,i)}}{\sum_{i \leq n^{(k)}(t)} \left(w^{(k,i)} + \text{time}_{\text{io}}^{(k,i)} \right)}.$$

Objectives

- SysEfficiency:

$$\text{maximize } \frac{1}{N} \sum_{k=1}^{K} \beta^{(k)} \tilde{\rho}^{(k)}(d_k).$$

- Dilation:

$$\text{minimize } \max_{k=1..K} \frac{\rho^{(k)}(d_k)}{\tilde{\rho}^{(k)}(d_k)}.$$

Scheduler

The scheduler monitors the stream of I/O calls; decides on the fly which applications can perform I/O.

- At each time step, it has access to the state of the system (each application efficiency, $\tilde{\rho}^{(k)}$).
- Based on a given strategy, chooses a subset of applications that are allowed to perform I/O.

Scheduler

The scheduler monitors the stream of I/O calls; decides on the fly which applications can perform I/O.

- At each time step, it has access to the state of the system (each application efficiency, $\tilde{\rho}^{(k)}$).
- Based on a given strategy, chooses a subset of applications that are allowed to perform I/O.

When a strategy *favors* $\text{App}^{(k)}$, it means that $\text{App}^{(k)}$ is executed as fast as possible ($\min\left(b\beta^{(k)}, \text{bw}_{\text{avail}}\right)$).

Different strategies

- RoundRobin: Similar to the current scheduler in HPC systems. Applications are served following the "First-Come, First Served" principle.

Different strategies

- RoundRobin: Similar to the current scheduler in HPC systems. Applications are served following the "First-Come, First Served" principle.
- MinDilation: favors applications with high values of $\frac{\rho^{(k)}(t)}{\tilde{\rho}^{(k)}(t)}$.

Different strategies

- RoundRobin: Similar to the current scheduler in HPC systems. Applications are served following the "First-Come, First Served" principle.
- MinDilation: favors applications with high values of $\frac{\rho^{(k)}(t)}{\tilde{\rho}^{(k)}(t)}$.
- MaxSysEff: favors applications with low values of $\beta^{(k)}\tilde{\rho}^{(k)}(t)$.

Different strategies

- RoundRobin: Similar to the current scheduler in HPC systems. Applications are served following the "First-Come, First Served" principle.
- MinDilation: favors applications with high values of $\frac{\rho^{(k)}(t)}{\tilde{\rho}^{(k)}(t)}$.
- MaxSysEff: favors applications with low values of $\beta^{(k)}\tilde{\rho}^{(k)}(t)$.
- MinMax-γ: same as MaxSysEff, unless there exists an applications with $\frac{\tilde{\rho}^{(k)}(t)}{\rho^{(k)}(t)}$ below a threshold γ. In that case, switches to MinDilation.

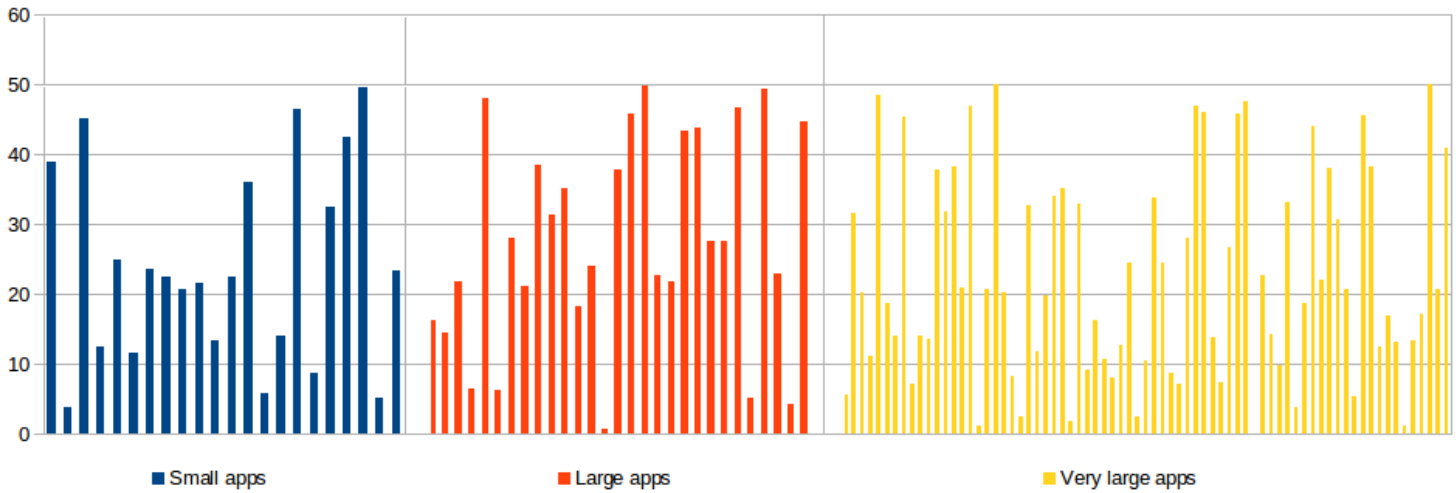
Different strategies

- RoundRobin: Similar to the current scheduler in HPC systems. Applications are served following the "First-Come, First Served" principle.
- MinDilation: favors applications with high values of $\frac{\rho^{(k)}(t)}{\tilde{\rho}^{(k)}(t)}$.
- MaxSysEff: favors applications with low values of $\beta^{(k)}\tilde{\rho}^{(k)}(t)$.
- MinMax-γ: same as MaxSysEff, unless there exists an applications with $\frac{\tilde{\rho}^{(k)}(t)}{\rho^{(k)}(t)}$ below a threshold γ. In that case, switches to MinDilation.

Priority variant: if an application has started to do some I/O, then it is prioritized.

Applications

We use Darshan to capture the behavior of applications that ran on Intrepid (2013).

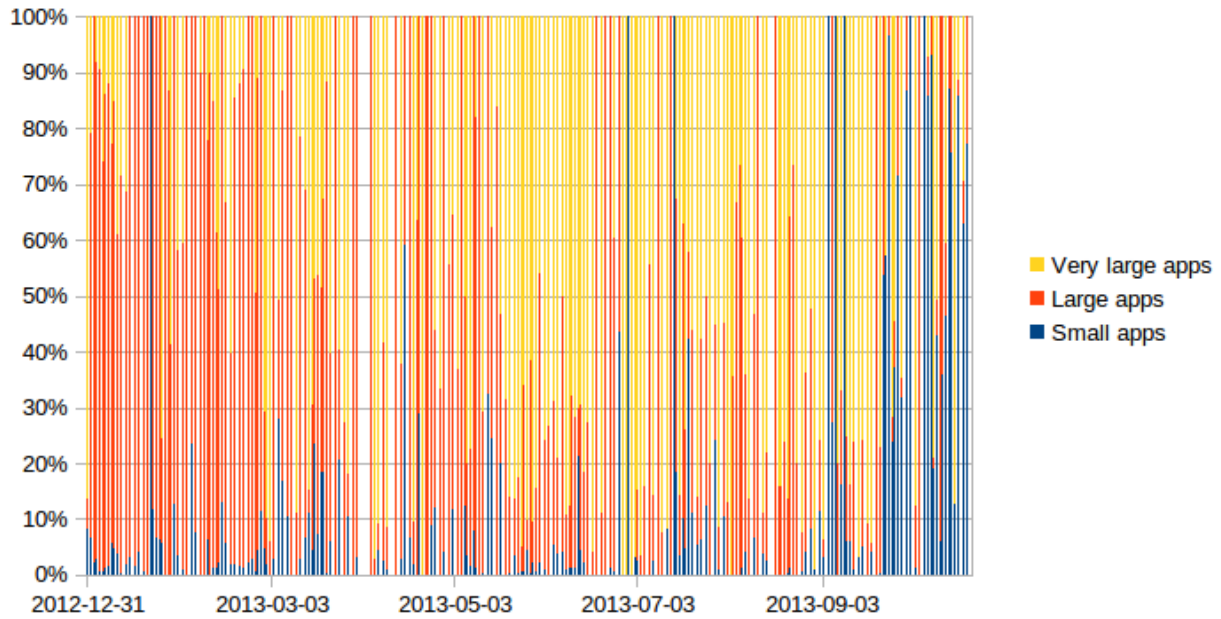


($\leq 1,284$ nodes) ($\geq 1,285$ nodes) ($\geq 4,584$ nodes)

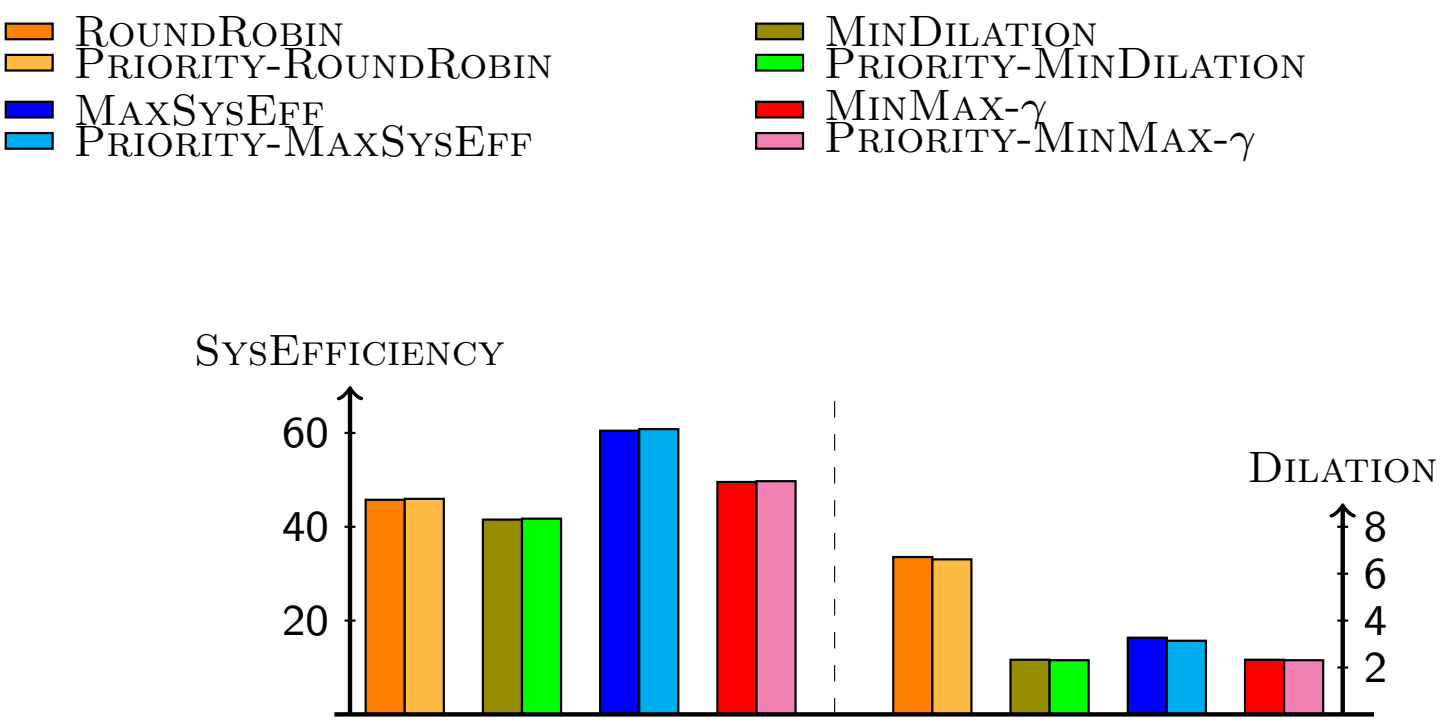
Percentage time spent doing I/O per application type.

Applications

We use Darshan to capture the behavior of applications that ran on Intrepid (2013).

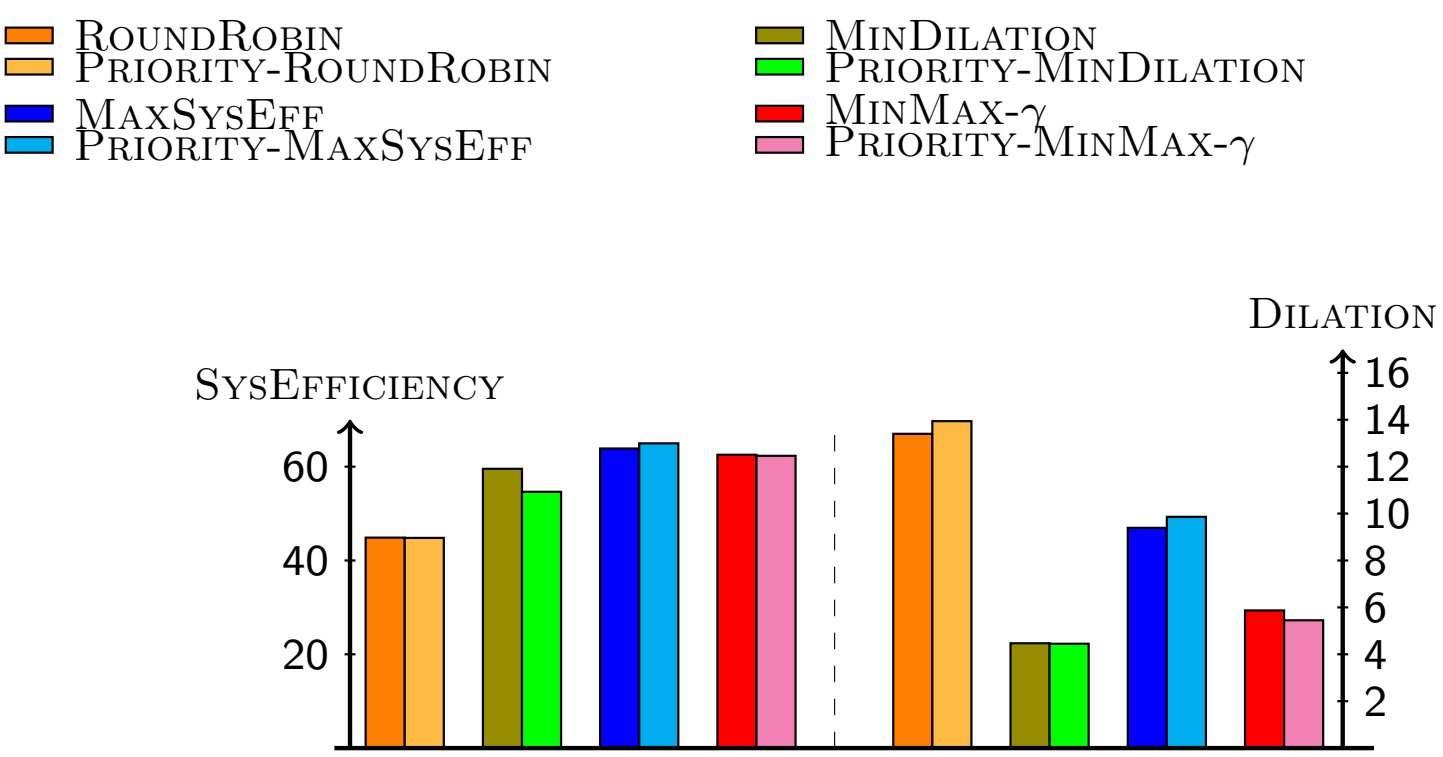


System usage per day for each application type



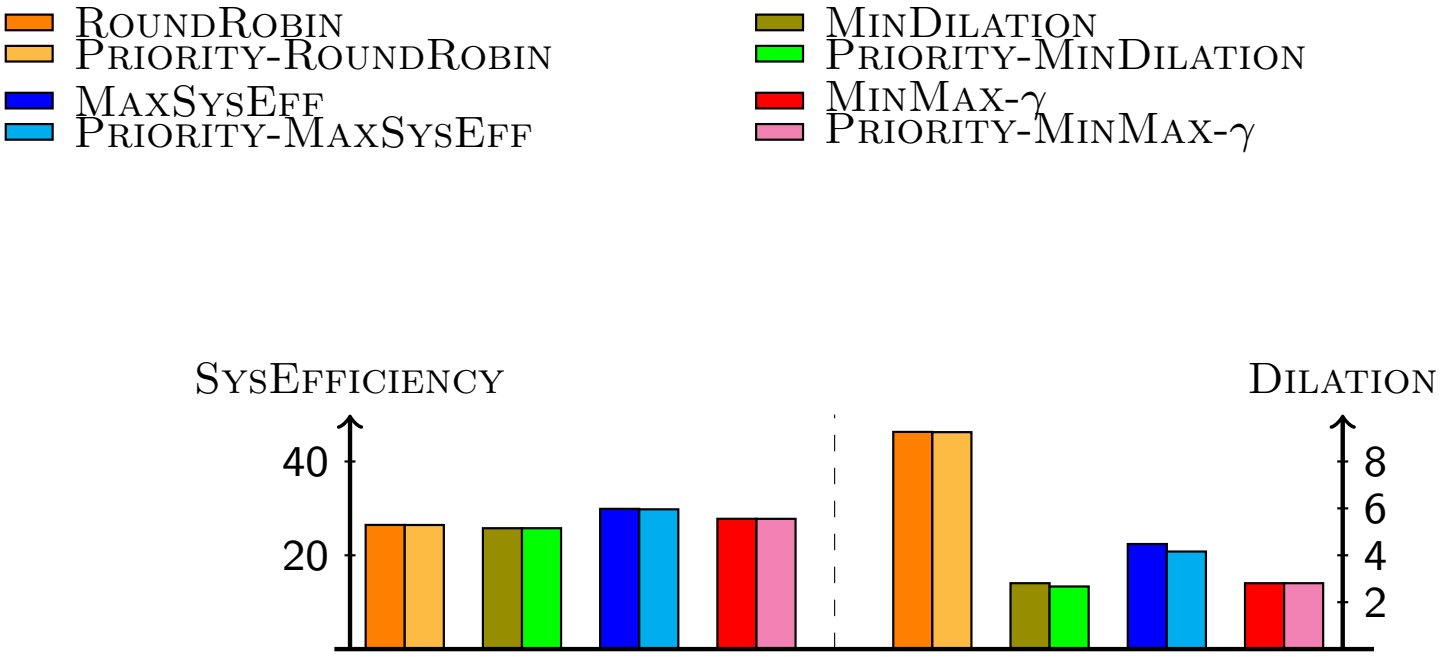
(a) 10 large applications, ratio of 20%

Objectives for different mixes of applications and I/O computation ratios.



(b) 50 small and 5 large applications, ratio of 20%

Objectives for different mixes of applications and I/O computation ratios.



(c) 50 small and 5 large applications, ratio of 35%

Objectives for different mixes of applications and I/O computation ratios.

Comparison of the heuristics on current platforms

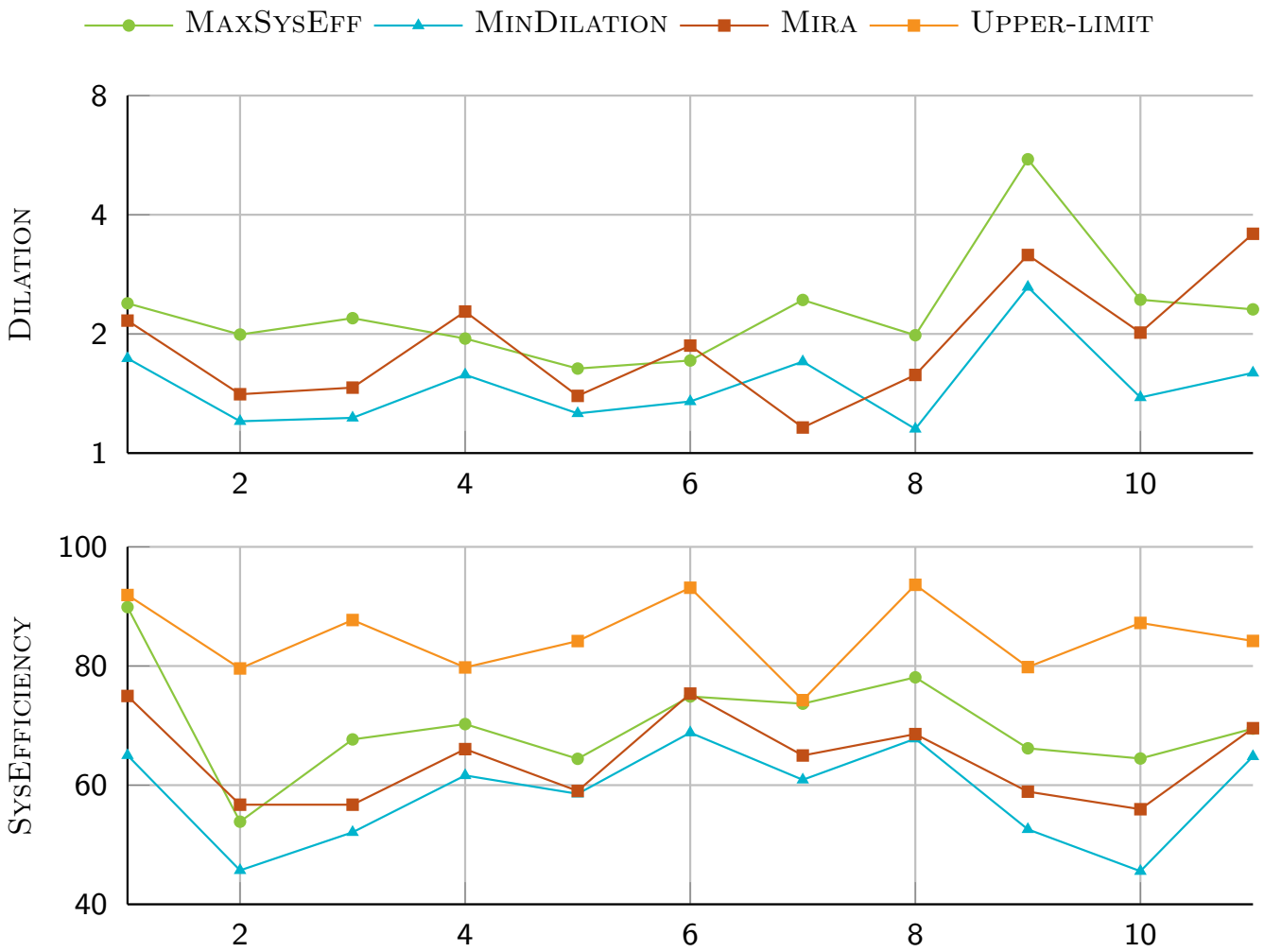
We then compared our results with the Intrepid and Mira scheduler when congestion occurs.

Note that Intrepid and Mira use an architectural enhancement to improve the behavior of applications with large bursts of I/O: *Burst Buffers*.

Comparison of the heuristics on current platforms

	Dilation (minimize)	SysEfficiency (maximize)
MaxSysEff	2.46	85.35
Priority variant	3.13	82.98
MinMax-0.25	2.33	83.08
Priority variant	2.93	80.31
MinMax-0.5	1.99	77.2
Priority variant	2.43	72.96
MinMax-0.75	1.69	71.66
Priority variant	2.03	66.94
MinDilation	1.63	70.45
Priority variant	1.96	65.64
Intrepid	2.55	71.12
Upper-limit	-	91.59

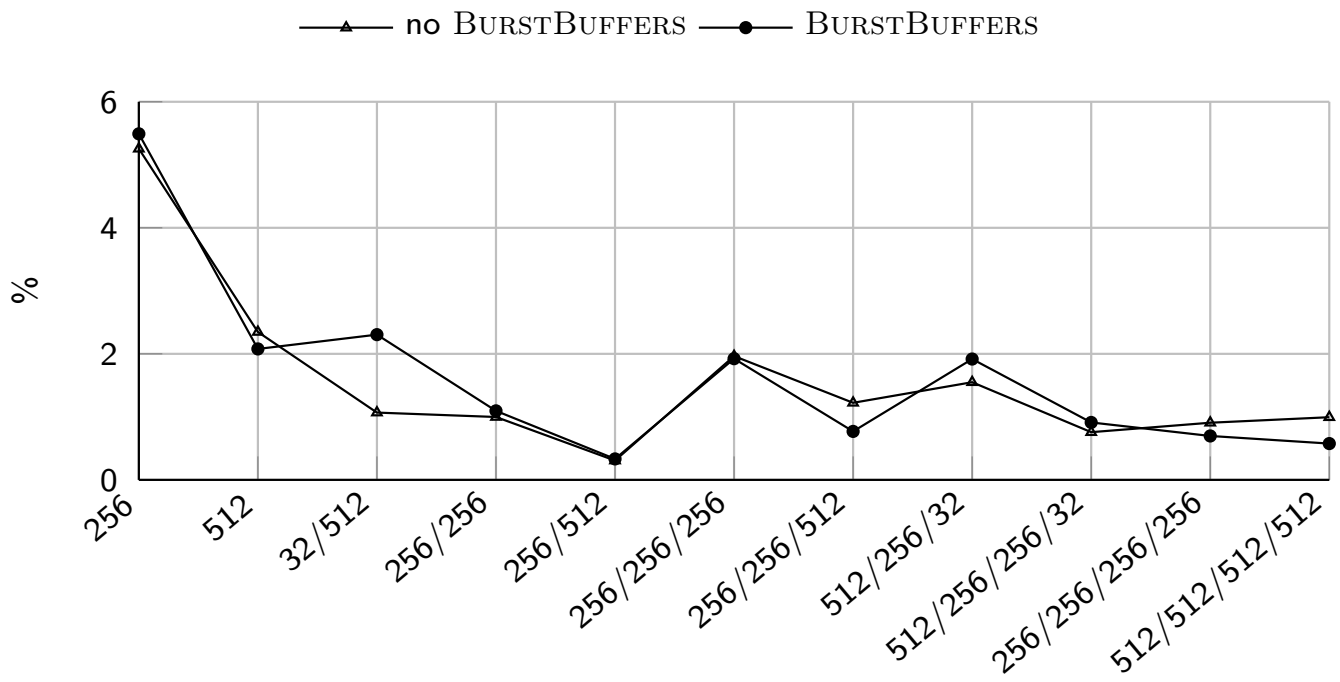
Table: Averages over 56 different congested moments on Intrepid.



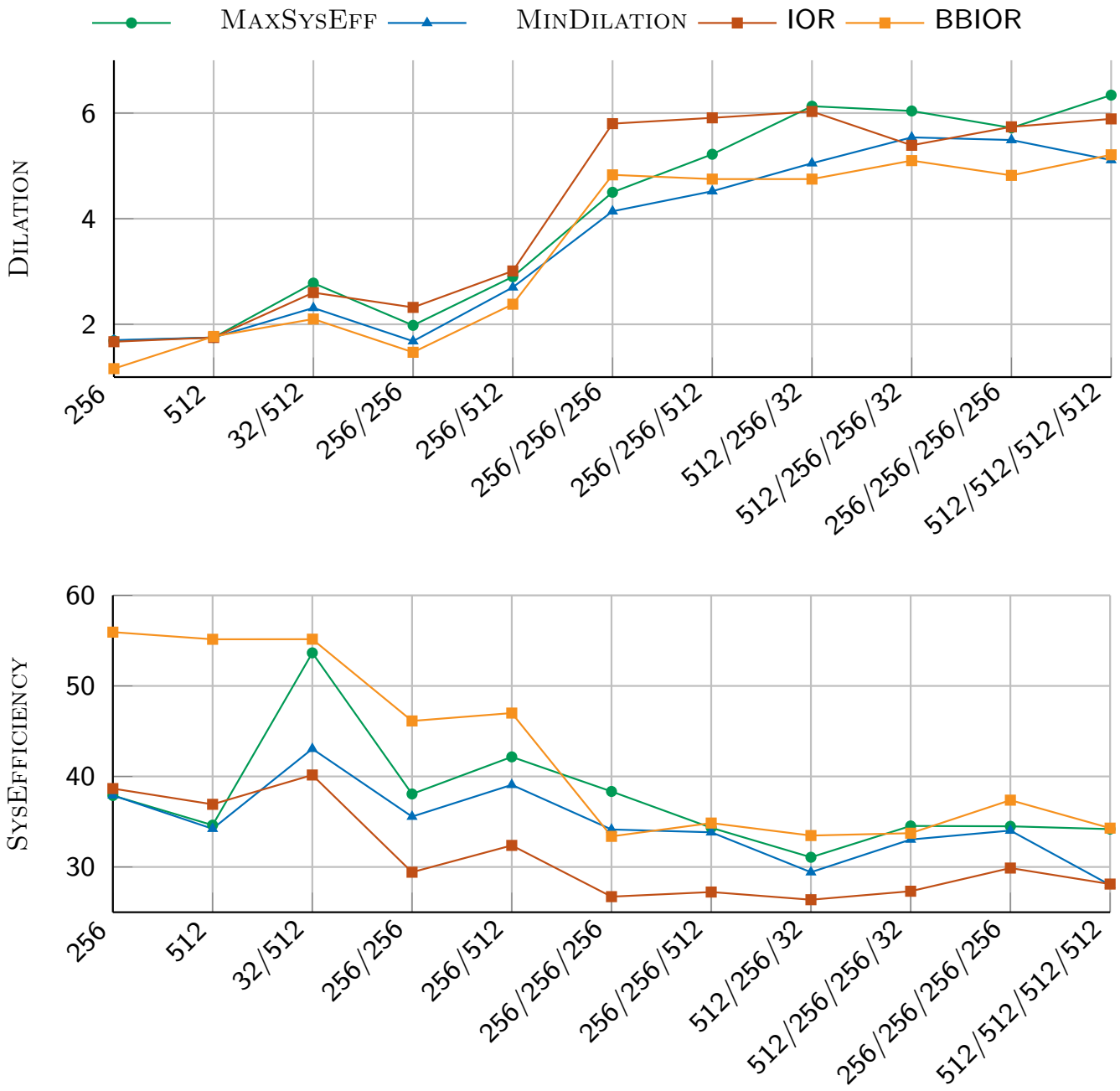
Comparison of the Priority heuristics over the current Dilation and SysEfficiency of Mira.

- Experiments on Vesta (development platform for Mira)
- Vesta is using hard disks and is affected by locality: we only used the PRIORITY variant of heuristics
- We implemented the heuristics as an additional layer on top of Vesta I/O scheduler

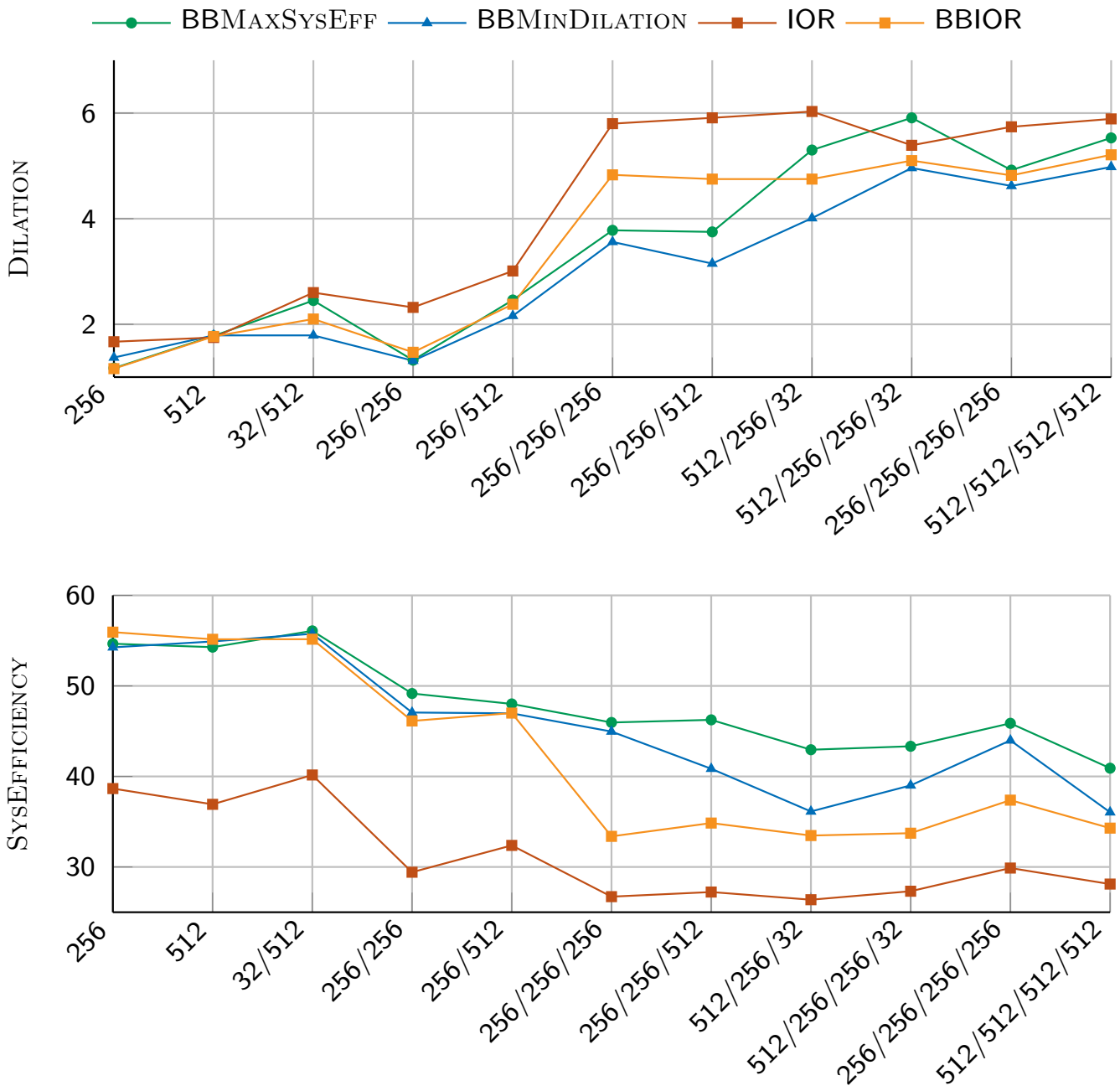
- Experiments on Vesta (development platform for Mira)
- Vesta is using hard disks and is affected by locality: we only used the PRIORITY variant of heuristics
- We implemented the heuristics as an additional layer on top of Vesta I/O scheduler



Execution time overhead of our implementation of the IOR benchmark.



System efficiency and dilation for different scenarios on Vesta.



System efficiency and dilation for different scenarios on Vesta.

Conclusion

- New I/O scheduler taking global view of system into account
- Outperforms current scheduler
- More experiments needed on larger application sets
- Window-based schedules for periodic applications?