I/O Scheduling in HPC systems

Guillaume Aupy

Project page: **https://project.inria.fr/dash/**

Collab. with JT Acquaviva (DDN), O Beaumont (Inria), L Eyraud-Dubois (Inria), E Jeannot (Inria), A Gainaru (Vanderbilt), V Le Fèvre (ENS Lyon), N Vidal (Inria)

Motivation

IO congestion in HPC systems:

- HPC applications are generating lots of data for PFS.
- Idea is to use a buffer when the I/O bandwidth is fully occupied
- The buffer can be emptied at a later time.

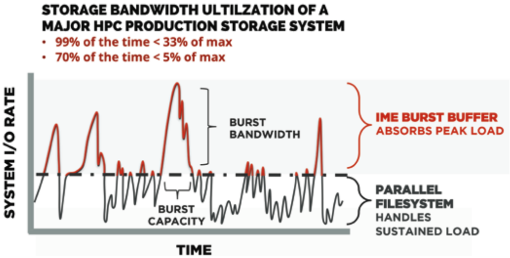


Figure: Burst-buffers to absorb IO peaks (DDN material)

IO congestion in HPC systems:

- HPC applications are generating lots of data for PFS.
- Idea is to use a buffer when the I/O bandwidth is fully occupied
- The buffer can be emptied at a later time.

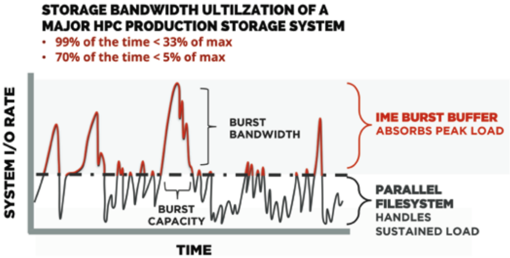


Figure: Burst-buffers to absorb IO peaks (DDN material)

"BUFFERS ARE THE ANSWER TO EVERYTHING!"
– DDN salesperson

Motivation

IO congestion in HPC systems:

- HPC applications are generating lots of data for PFS.
- Idea is to use a buffer when the I/O bandwidth is fully occupied
- The buffer can be emptied at a later time.

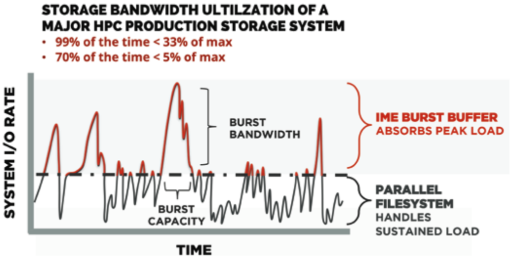
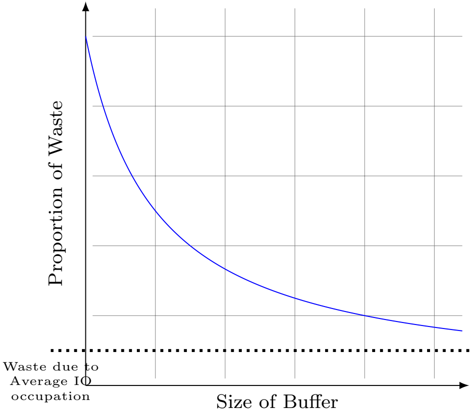


Figure: Burst-buffers to absorb IO peaks (DDN material)

"BUFFERS ARE THE ANSWER TO EVERYTHING!"
– DDN salesperson (maybe)

Finding the right size

Overall, for various parameter we obtain something that looks like this:

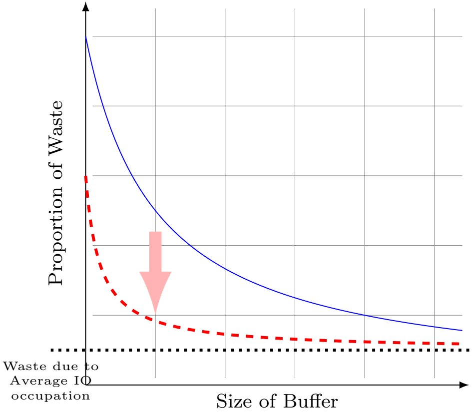


Naive strategy:

- If IO bandwidth avail.:
  → use it
- Else;
  → fill the burst buffers
- When IO bandwidth is avail. and buffers are full enough:
  → empty the burst-buffers.

Finding the right size

Overall, for various parameter we obtain something that looks like this:



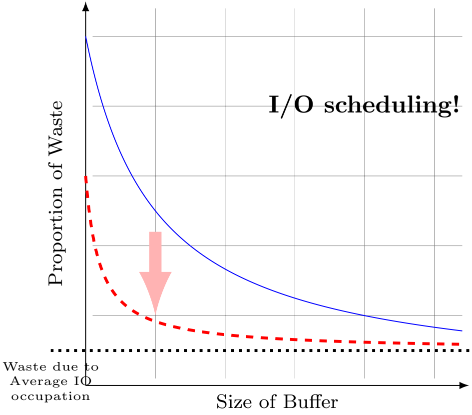
Naive strategy:

- If IO bandwidth avail.:
  → use it
- Else;
  → fill the burst buffers
- When IO bandwidth is avail. and buffers are full enough:
  → empty the burst-buffers.

How can we further reduce S for a given waste?

Finding the right size

Overall, for various parameter we obtain something that looks like this:



Naive strategy:

- If IO bandwidth avail.:
  → use it
- Else;
  → fill the burst buffers
- When IO bandwidth is avail. and buffers are full enough:
  → empty the burst-buffers.

How can we further reduce S for a given waste?

Previously in IO scheduling.

"Online" scheduling:

- When an application is ready to do I/O, it sends a message to an I/O scheduler;
- Based on the other applications running and a priority function, the I/O scheduler will give a **GO** or **NOGO** to the application.
- If the application receives a **NOGO**, it pauses until a **GO** instruction.
- Else, it performs I/O.

Gainaru, A., Benoit, Cappello, Robert, Snir,
`Scheduling HPC applications under I/O congestion`, IPDPS'15

Previously in IO scheduling.

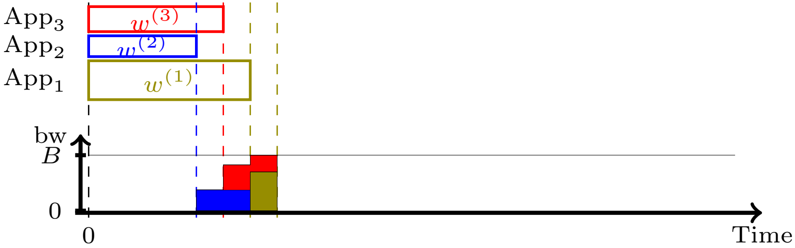
Previously in IO scheduling.

Previously in IO scheduling.

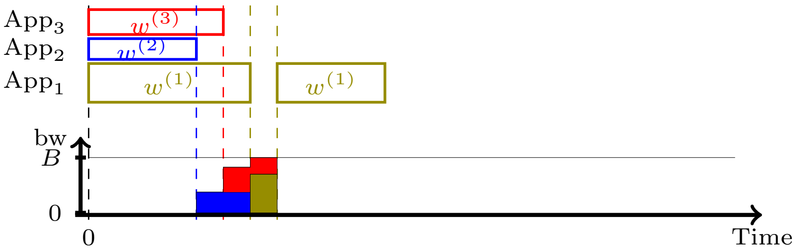
Previously in IO scheduling.

Previously in IO scheduling.

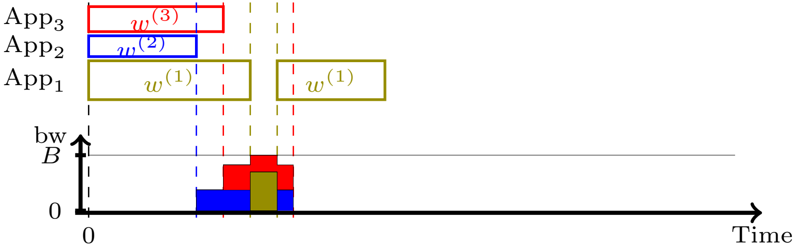
Previously in IO scheduling.



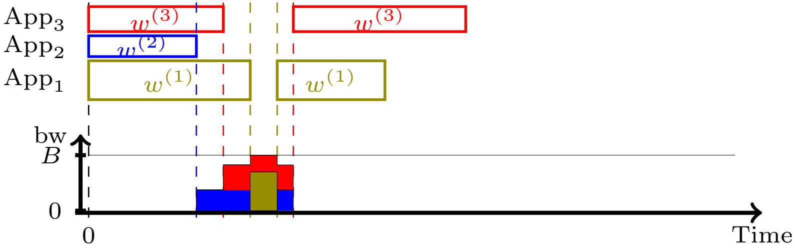
Previously in IO scheduling.



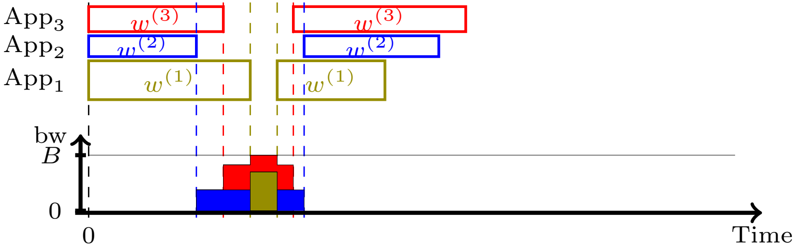
Previously in IO scheduling.



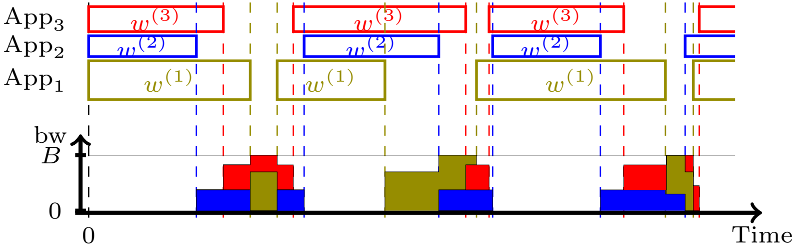
Previously in IO scheduling.



Previously in IO scheduling.



Previously in IO scheduling.



Approx 10% improvement in application performance with 5% gain in system performance on Intrepid.

Can we do better?

Assume we know applications I/O patterns:

- from historical data;
- because of periodic checkpointing;
- average filling speed of buffers

  $\Longrightarrow$ we know that every x min, $y\%$ of the buffer needs to be emptied;
- ..

Can we do better?

Assume we know applications I/O patterns:

- from historical data;
- because of periodic checkpointing;
- average filling speed of buffers

  $\Longrightarrow$ we know that every x min, $y\%$ of the buffer needs to be emptied;
- ..

Can we use this information and enforce efficient static schedules?

Spoiler: it works very well (at least it seems promising)

High-level constraints

- Applications are already scheduled on the machines: not (yet) our job to do it;

A., Gainaru, Le Fèvre, `Periodic I/O scheduling for super-computers`, PMBS'17

High-level constraints

- Applications are already scheduled on the machines:
  not (yet) our job to do it;
- We want the schedule information distributed over the applis:
  the goal is not to add a new congestion point;

A., Gainaru, Le Fèvre, `Periodic I/O scheduling for super-computers`, PMBS'17

High-level constraints

- Applications are already scheduled on the machines: not (yet) our job to do it;
- We want the schedule information distributed over the applis: the goal is not to add a new congestion point;
- Computing a full I/O schedule over all iterations of all applications would be too expensive (i) in time, (ii) in space.

A., Gainaru, Le Fèvre, `Periodic I/O scheduling for super-computers`, PMBS'17

High-level constraints

- Applications are already scheduled on the machines:
  not (yet) our job to do it;
- We want the schedule information distributed over the applis:
  the goal is not to add a new congestion point;
- Computing a full I/O schedule over all iterations of all applications would be too expensive (i) in time, (ii) in space.
- We want a minimum overhead for Applis users:
  otherwise, our guess is, users might not like it that much ☺.

A., Gainaru, Le Fèvre, `Periodic I/O scheduling for super-computers`, PMBS'17

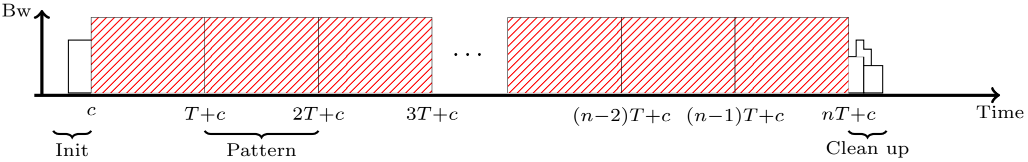
High-level constraints

- Applications are already scheduled on the machines: not (yet) our job to do it;
- We want the schedule information distributed over the applis: the goal is not to add a new congestion point;
- Computing a full I/O schedule over all iterations of all applications would be too expensive (i) in time, (ii) in space.
- We want a minimum overhead for Applis users: otherwise, our guess is, users might not like it that much ☺.

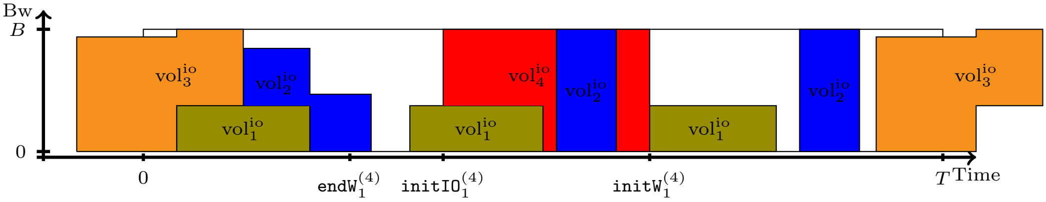
We introduce Periodic Scheduling.

A., Gainaru, Le Fèvre, `Periodic I/O scheduling for super-computers`, PMBS'17

Periodic schedules



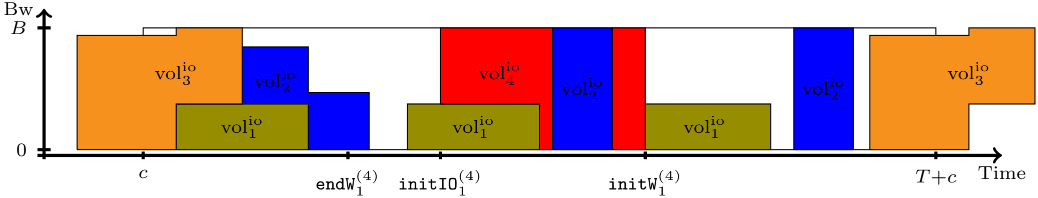
(a) Periodic schedule (phases)



(b) Detail of I/O in a period/pattern

Periodic schedules

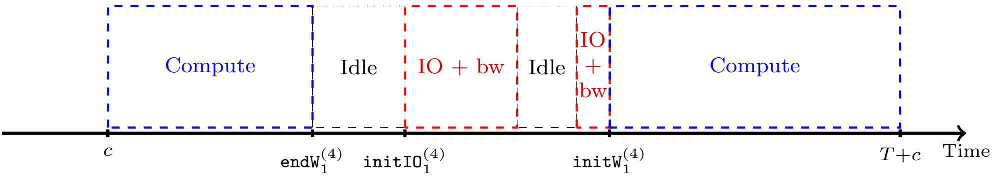
Time Schedule vs what Application 4 sees



- Distributed information
- Low complexity
- Minimum overhead

Periodic schedules

Time Schedule vs what Application 4 sees



- Distributed information ✓
- Low complexity ✓
- Minimum overhead ✓

Model validation

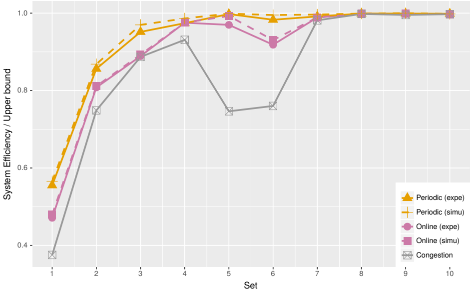
Setup:

- Applications from the literature, 10 sets with different contention.
- Comparison between simulations and a real machine (Jupiter @Mellanox: 640 cores, $b = 0.01GB/s$, $B = 3GB/s$).
- Instations with IOR benchmark (ideal world, no other communication than I/O transfers).

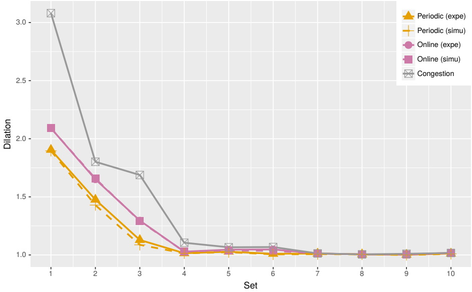
Algos:

- Periodic: our periodic algorithm;
- Online: the best performance of any online algorithm in Gainaru et al. IPDPS'15;
- Congestion: Current performance on the machine.

A., Gainaru, Le Fèvre, `Periodic I/O scheduling for super-computers`, PMBS'17



(c) SysEfficiency/Upper bound

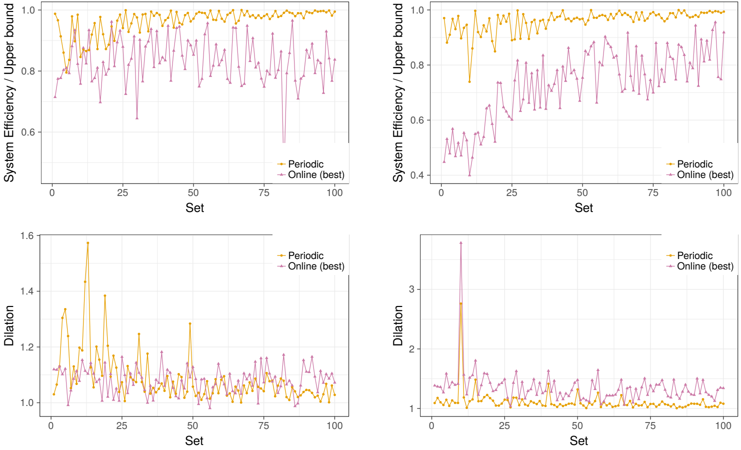


(d) Dilation

The performance estimated by our model is accurate within 3.8% for periodic schedules and 2.3% for online schedules.

More results (simulations)

- We generate more sets of applications
- Simulate on instanciations of Intrepid and Mira.



Intrepid

Mira

The DASH project

- Understanding applications patterns
- Study of robustness: what if w_k and $\mathrm{vol}_k^{\mathrm{io}}$ are not exactly what they were supposed to be?
- Integrating non-periodic application
- Burst-buffers integration/modeling
- Coupling application scheduler to IO scheduler
- Evaluation on real applications

Collaborations needed:

Three steps

1. I/O Modeling
2. Algorithm design
3. Evaluation and Integration

Application/system users:

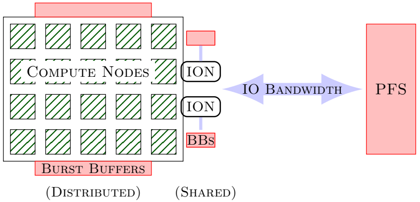
- Data about I/O behaviors: patterns/periodicity, volume etc.
- Discussion about models of I/O scalability

From large-scale IO managers:

- Understanding traces, what kind of properties can we assume on the system?
- Discussion on implementation / integration
- Experiments

Dash

Data-Aware Scheduling at Higher scale



Nicolas Vidal is starting his PhD this October. Talk to him ☺.

Thanks to collab. and co-authors
JT Acquaviva (DDN), O Beaumont (Inria), L Eyraud-Dubois (Inria), E Jeannot (Inria), A Gainaru (Vanderbilt), V Le Fèvre (ENS Lyon), N Vidal (Inria)

Papers, data, code: `https://project.inria.fr/dash/`