Mapping and Scheduling HPC Applications for optimizing I/O

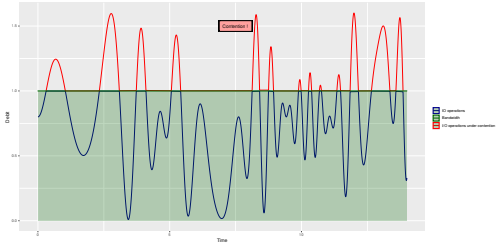
Nicolas Vidal Emmanuel Jeannot Guillaume Pallez

TADaaM - Inria Bordeaux Sud-Ouest

December 8, 2020

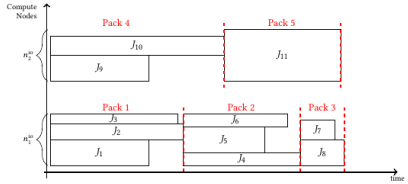
Context

- Concurrent application allocated on different compute nodes
- Racks of compute nodes are associated with I/O resources
- I/O nodes are shared proxies that have to be used to perform I/O
- Bandwidth is shared
- If too many I/O operations are performed, there is contention

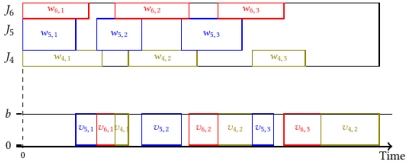


Problems

1. Given a set of jobs which perform I/O on a given node, how can we schedule the operation to optimize performance ?
2. How do we allocate jobs to racks ?



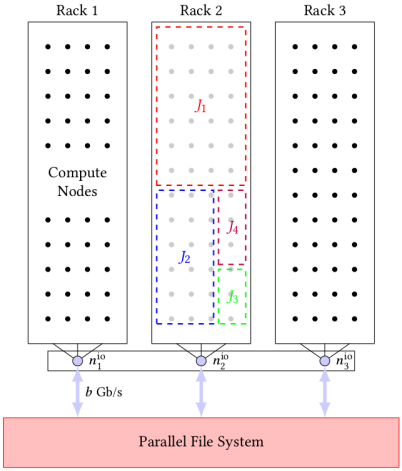
(a) The mapping of 11 applications into packs



(b) The scheduling of I/O operations (bottom) within Pack 2. Computations are allocated on dedicated compute nodes and can start as soon as the I/O is transfered, but I/O operation share the available bandwidth b and can delay applications.

Models proposition

Machine Model



Application Model

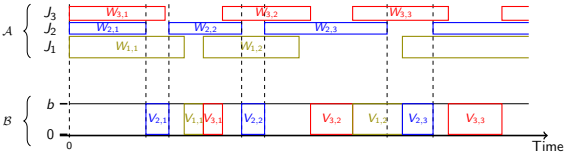


Figure: Schematic overview of three jobs J_1 , J_2 , J_3 scheduled on a bi-resources platform.

Scheduling problem

Problem

Find a schedule that minimizes the total makespan MS-HPC-IO
or the maximum stretch ρ-HPC-IO.

- NP-complete
- Few simple cases: chains of length 1 ; homogeneous tasks
- List-Scheduling (LS) provides a 2-approximation for MS-HPC-IO

$\rightarrow$ LS policies are similar in theory but should be different in practice. How do we evaluate them ?

Evaluation

- Clarisse/FlexMPI[1] with synthetic workloads
- Steady state
- Choose between two "families" of heuristics:
  1. Free the machine as soon as possible
  2. Grant "fairly" accesses to applications

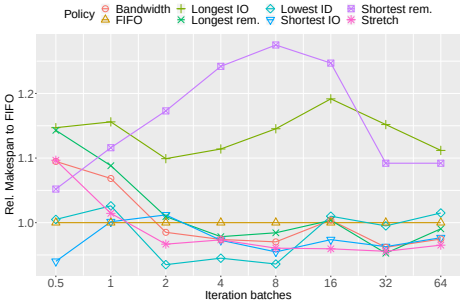
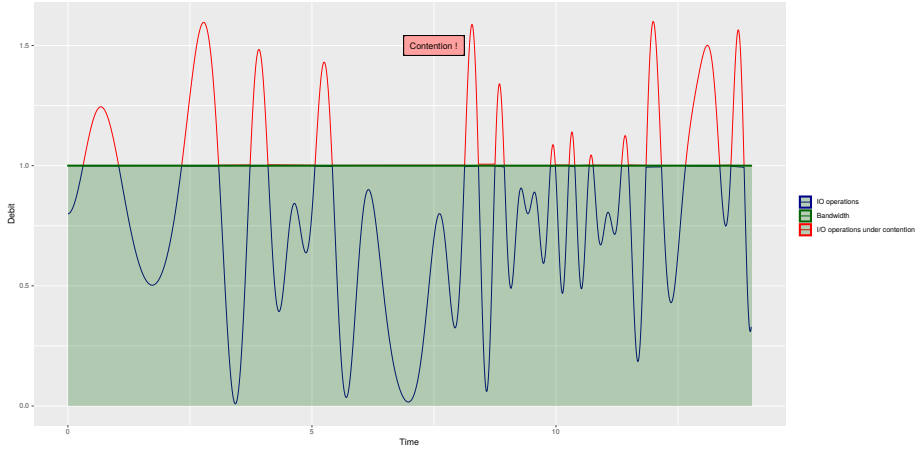


Figure: Relative Makespan to FIFO of the different policies when varying the number of iteration batches.

[1] "CLARISSE is a middleware for enhancing I/O flow coordination and control in the HPC I/O stack software." It aimes to coordinate the parallel I/O on HPC platforms.

Mapping problem



Packing applications

Model change

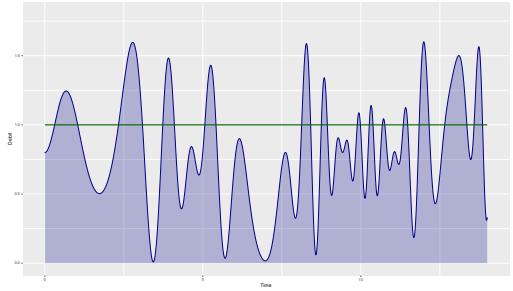


Figure: "Real" behavior

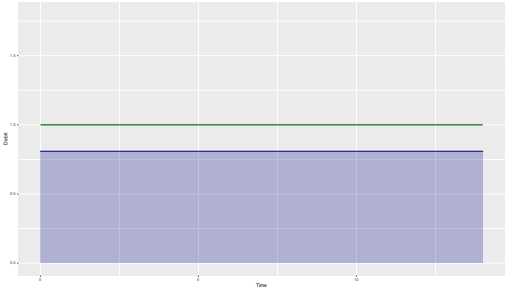


Figure: Continuous model

→ We use this new model to define I/O constraints for pack scheduling

Results

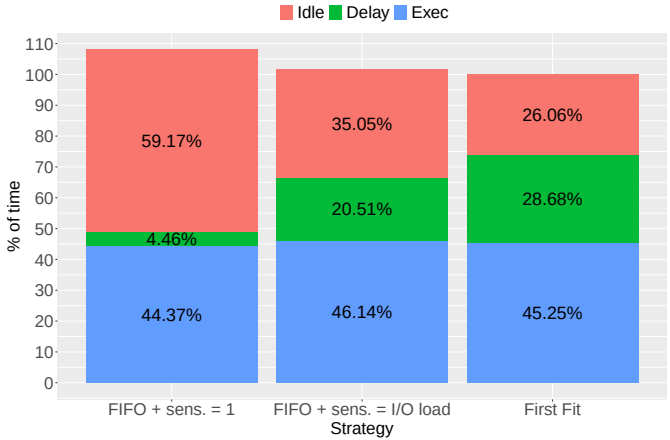
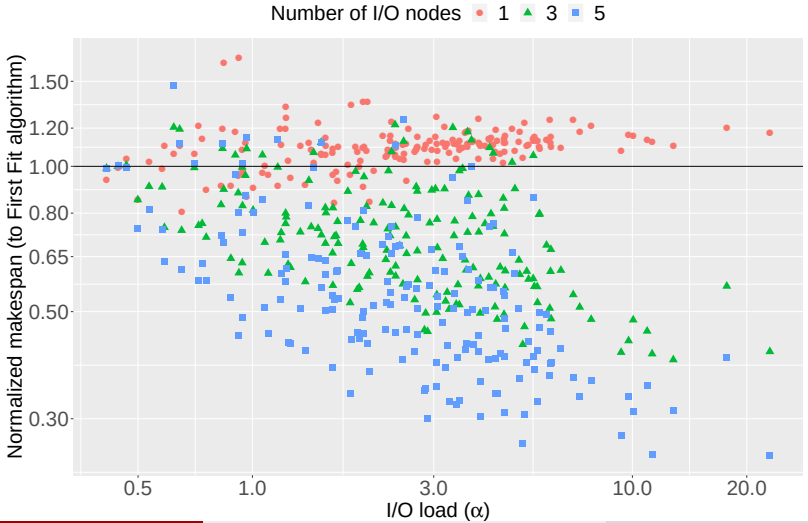


Figure: Average execution, idle and delay time (normalized) for different strategies

Scaling


Number of I/O nodes 1 3 5
Normalized makespan (to First Fit algorithm)
1.50
1.20
1.00
0.80
0.65
0.50
0.30
0.5
1.0
3.0
10.0
20.0
I/O load (α)

Ongoing : LS vs Pack

- Making packs is a big constraint - can we get rid of it ?
- Let's try LS scheduling

Performance

- LS is better when low I/O intensity
- Pack is better when the I/O threshold is tight

Ongoing : LS vs Pack

Impact of I/O threshold on policies

- I/O intensity of packs is an average: total I/O / pack duration
- I/O intensity of LS is at time t: Sum of application I/O intensity at time t
- $\rightarrow$ The constraint is tighter for LS.

What is a good schedule ?

- The point of limiting I/O is to enable a better use of the resources
- How do we quantify if a schedule as good for resource usage
- Can we produce a metric for "suitable for backfilling" ?

Conclusion

Published

- Simple fair heuristics which perform well in the long run.
- A strategy to allocate applications together based on an approximation of their resource usage.
- Interesting results:
  1. Slight degradation of the makespan
  2. Much better control of the waste
  3. Great scaling

Current works

- Remove pack constraint
- Study the impact I/O threshold on policies
- What is a good schedule ?