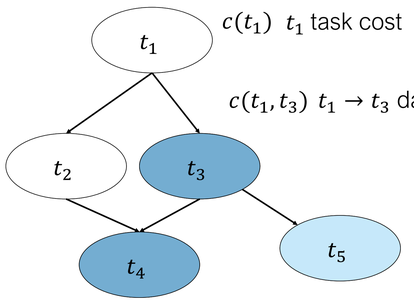
Task Scheduling Analysis: Strategic Task Duplication Integrated in SAGA

The Task Scheduling Problem

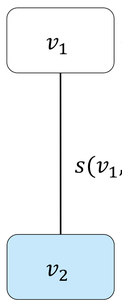
Task Graph



$c(t_1)$ t_1 task cost

$c(t_1, t_3)$ $t_1 \rightarrow t_3$ data size

Compute Network



$s(v_1)$ compute speed for node v_1

$s(v_1, v_3)$ $v_1 \leftrightarrow v_3$ comm strength

$\frac{c(t_1)}{s(v_1)}$ Time for task t_1 to execute on compute node v_1

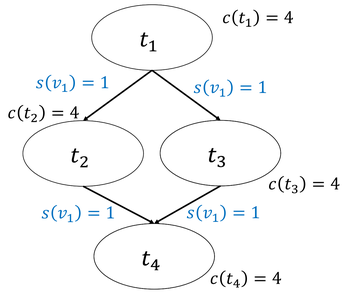
Schedule

compute node v_1	task t_1	task t_2		task t_4
compute node v_2			task t_3	task t_5

$\frac{c(t_1, t_3)}{s(v_1, v_2)}$ Time for compute node v_1 to send task t_1 outputs to compute node v_2

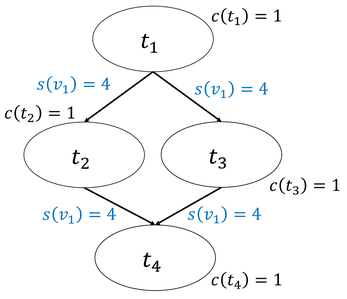
Makespan:
Time from beginning of first task to ending of last task

A Motivating Example



Computation-Heavy Workflow:

- Task execution dominates total time
- Inter-processor communication $s(v_1)$ is fast in relation to task size $c(t_1)$
- Average Computation Cost > Average Communication Cost
- Low Communication-to-Computation Ratio (CCR)



Communication-Heavy Workflow:

- Communication dominates total time
- Inter-processor communication $s(v_1)$ is slow in relation to task size $c(t_1)$
- Average Communication Cost > Average Computation Cost
- High Communication-to-Computation Ratio (CCR)

SAGA: Scheduling Algorithms Gathered

Abbreviation	Algorithm	
BIL	Best Imaginary Level	[26]
BruteForce	Brute Force	-
CPoP	Critical Path on Processor	[32]
Duplex	Duplex	[4]
ETF	Earliest Task First	[19]
FastestNode	Fastest Node	-
FCP	Fast Critical Path	[27]
FLB	Fast Load Balancing	[27]
GDL	Generalized Dynamic Level	[31]
HEFT	Heterogeneous Earliest Finish Time	[32]
MaxMin	MaxMin	[4]
MCT	Minimum Completion Time	[4]
MET	Minimum Execution Time	[4]
MinMin	MinMin	[4]
OLB	Opportunistic Load Balancing	[4]
SMT	SMT-driven Binary Search	-
WBA	Workflow-Based application	[1]

Idea:

- Extend and optimize traditional SAGA schedulers by allowing task duplication and relaxing the constraint that each task must run exactly once.
- If communication delays dominate execution time, then replicating strategically selected tasks on multiple processors should allow their successors to access data locally, eliminating costly inter-processor data transfers.

Duplication Strategy Explained

Concepts

- **Duplication Factor**: Maximum number of times a task is allowed to duplicate across processors
- **Task Out-Edges:** Number of successor tasks for each given node
- **Average Inter-node Link:** Average communication speed between two processors
- **Average Computation Time:** Task Cost $c(t_1)$ / Node Speed $s(v_1)$
- **Average Communication Time:** Data Size $c(t_1, t_3)$ / Inter-node Link

Our Solution: The should_duplicate Heuristic

1. Get the task object from the task graph
2. Compute average computation time across all nodes
3. If task has no out-edges, return False (duplication only makes sense if a node has multiple “children” that access data locally)
4. Compute average inter-node link speed
5. If inter-node link speed = 0, return False (no additional processors to allow duplication)
6. Compute the average communication time over all outgoing dependencies.
7. If **Average communication time > Average computation time,** return True (only duplicate on communication-heavy instances)

Results: Tests on CPoP and HEFT

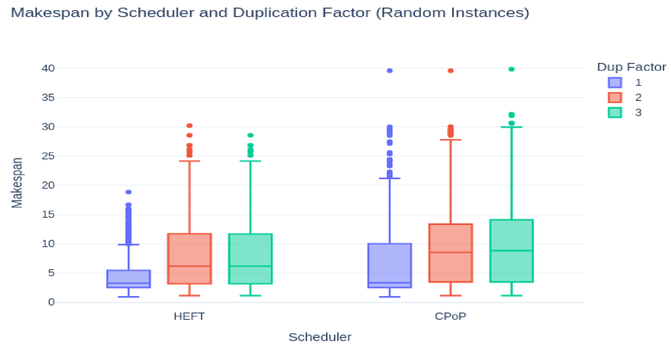


Figure 1: Makespan across different duplication factors

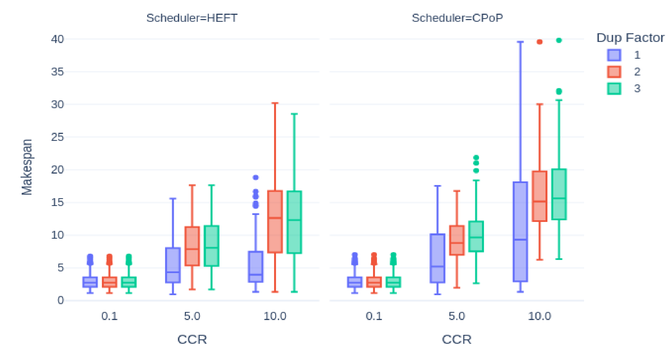


Figure 2: Makespan across different duplication factors and changes in CCR

Conclusions & Future Work

- Contrary to our thesis, task duplication is increasing the makespan by duplicating even when there's no communication benefit.
- Our should_duplicate heuristic seems to be incompatible with the previously developed schedulers in SAGA like HEFT and CPoP.
- Related work shows duplication directly optimizes the schedulers, and we’re researching for ways to adapt it to our framework to see similar results.
- Future work includes refinining the **should_duplicate** heuristic to see positive results and extending the implementation to *all* the available schedulers in SAGA.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Award No. 2451267.