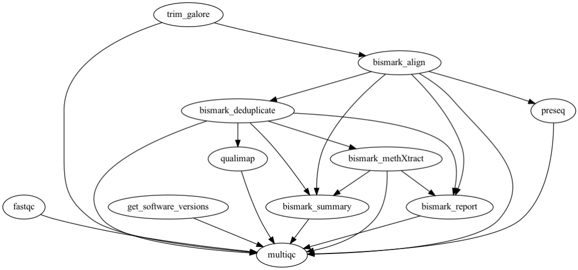
Memory-aware Adaptive Scheduling of Scientific Workflows On Heterogeneous Architectures

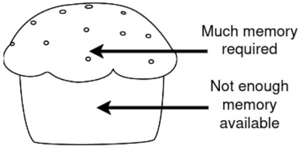
Svetlana Kulagina[1], Anne Benoit[2] and Henning Meyerhenke[1,3]

[1] Humboldt-Universität zu Berlin, Germany
[2] LIP, ENS Lyon and IUF, France
[3] Karlsruhe Institute of Technology (KIT), Germany

Motivation

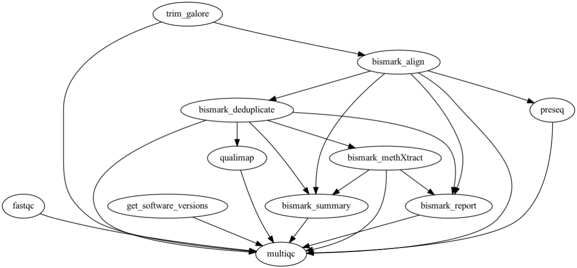
- Large applications represented as DAG-shaped workflows
- Heterogeneous execution environment with limited memories
  - Exceed memory = expensive or even fatal
- Execute the workflow fast

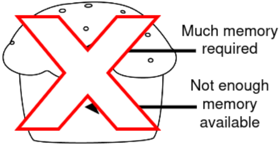




Motivation

- Large applications represented as DAG-shaped workflows
- Heterogeneous execution environment with limited memories
  - Exceed memory = expensive or even fatal
- Execute the workflow fast





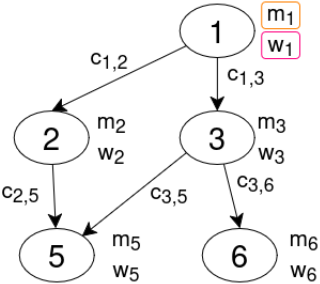
Contributions

3 variants of a HEFT-based scheduler: HEFTM-**BL**, HEFTM-**BLC**, and HEFTM-**MM**

- Respects processor memory sizes
  - HEFTM-**BL** and HEFTM-**BLC** produce valid makespans that are only $10\text{-}35\%$ worse than invalid HEFT makespans
  - HEFTM-**MM** able to schedule all workflows even under extreme memory constraint
- $+$ adaptive strategy for scenario with uncertainty
  - Adapting the schedule leads to $> 20\%$ makespan improvement even on smallest workflows

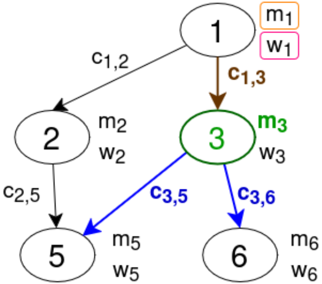
Model: Workflow

- Tasks have memory and workload weights
- Edges have (file) weights



Model: Workflow

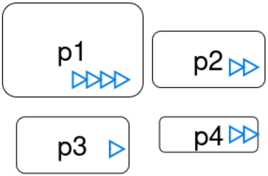
- Tasks have memory and workload weights
- Edges have (file) weights



$$r_u = \max\left\{ \boxed{m_u,} \boxed{\sum_{v:(v,u)\in E} c_{v,u},} \boxed{\sum_{v:(u,v)\in E} c_{u,v}} \right\}$$

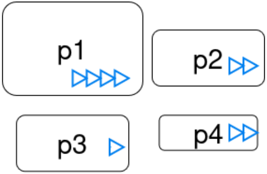
Model: Execution Environment

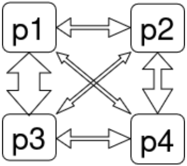
- Each processor has
  - a limited memory of size M_j and a communication buffer MC_j
  - speed s_j



Model: Execution Environment

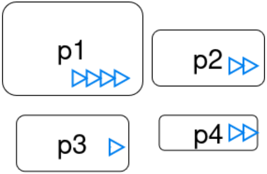
- Each processor has
  - a limited memory of size M_j and a communication buffer MC_j
  - speed s_j

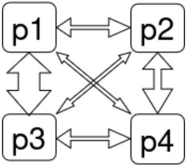




Model: Execution Environment

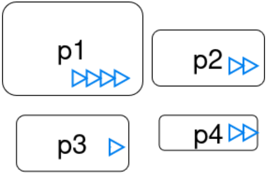
- Each processor has
  - a limited memory of size M_j and a communication buffer MC_j
  - speed s_j
- Keep track of ready times on each
  - Communication channel: $rt_{j,j'}$
  - Communication buffer: rt_j

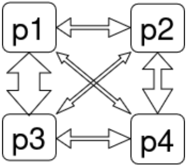




Model: Execution Environment

- Each processor has
  - a limited memory of size M_j and a communication buffer MC_j
  - speed s_j
- Keep track of ready times on each
  - Communication channel: $rt_{j,j'}$
  - Communication buffer: rt_j





Objective: makespan minimization with memory constraints

HEFT (in our model)

HEFT [1]:

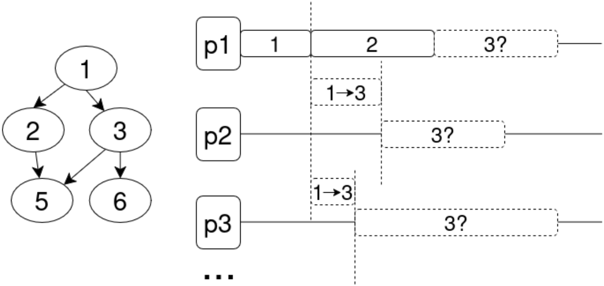
- Assign each task a rank
- Schedule tasks in decreasing rank order

[1] H. Topcuoglu, S. Hariri, M.-Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. 2002.

HEFT (in our model)

HEFT [1]:

- Assign each task a rank
- Schedule tasks in decreasing rank order
  - Tentatively assign the task to each processor
  - Choose the one with shortest finish time



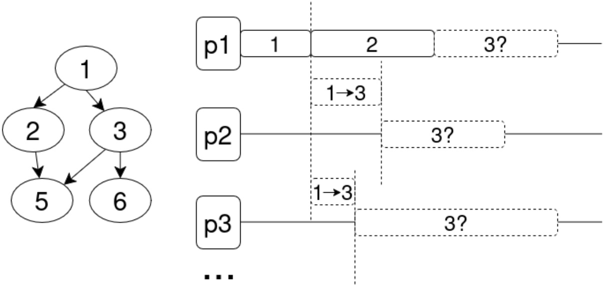
[1] H. Topcuoglu, S. Hariri, M.-Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. 2002.

HEFT (in our model)

HEFT [1]:

- Assign each task a rank
- Schedule tasks in decreasing rank order
  - Tentatively assign the task to each processor
  - Choose the one with shortest finish time

What about memory?



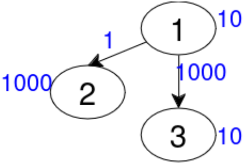
[1] H. Topcuoglu, S. Hariri, M.-Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. 2002.

Memory Traversal

Fit into limited memory $\rightarrow$ use as little memory as possible

Peak memory depends on the traversal

- Traversal $1 \rightarrow 3 \rightarrow 2$
  - Execution of 3: $1000(c_{1,2}) + 10(m_3) = 1011$
  - Execution of 2: $1000(m_2) + 1(c_{1,2}) = 1001$
- Traversal $1 \rightarrow 2 \rightarrow 3$
  - Execution of 2: $1000(m_2) + 1(c_{1,2}) + 1000(c_{1,3}) = 2001$
- Memory-optimal traversal MEMDAG [1]



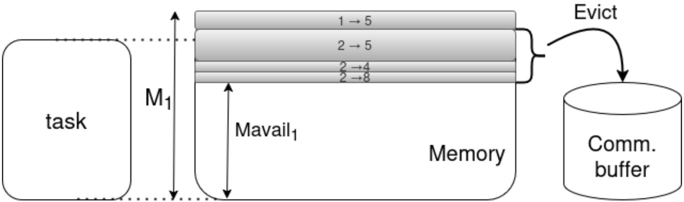
[1] Kayaaslan, E., Lambert, T., Marchal, L., Uçar, B. (2018). Scheduling series-parallel task graphs to minimize peak memory. Theoretical Computer Science, 707, 1-23.

Scenario and Model: Eviction

When a task executes, it

- consumes (deletes from memory) its incoming files,
- creates (inserts into memory) its outgoing files.

To fit a task into memory, we may need to evict pending memories into the communication buffer



No returning tasks back into memory!

S. Kulagina, HU Berlin

HEFTM-* heuristics: Ranking

Similar 2 phases to HEFT: rank and assign.

Implementation of the phases different.

HEFTM-* heuristics: Ranking

Similar 2 phases to HEFT: rank and assign.

Implementation of the phases different.

- HEFTM-**BL**: orders by non-increasing bottom levels
- HEFTM-**BLC**: same, but prioritizes tasks with large incoming communications

$$blc(u) = w_u + \max_{(u,w)\in E}\{c_{u,w} + blc(w)\} + \max_{(v,u)\in E} c_{v,u}$$

- HEFTM-**MM**: orders according to MEMDAG's memory-optimal traversal

HEFTM-* heuristics: Task Assignment

For each task v, **tentatively** try it on each processor p_j

- Check that for all predecessors assigned to p_j the data is still in memory $\rightarrow$ otherwise invalid choice
- Address the memory constraint on the processor

$$Res = availM_j - m_v - \sum_{u \in \Pi(v), u \notin T(p_j)} \{c_{u,v}\} - \sum_{w \in Succ(v)} \{c_{v,w}\}$$

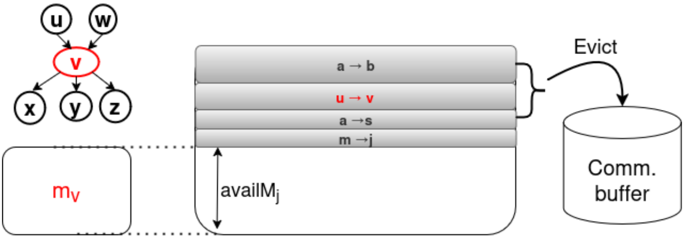
S. Kulagina, HU Berlin

HEFTM-* heuristics: Task Assignment

For each task v, **tentatively** try it on each processor p_j

- Check that for all predecessors assigned to p_j the data is still in memory $\rightarrow$ otherwise invalid choice
- Address the memory constraint on the processor

$$Res = availM_j - m_v - \sum_{u \in \Pi(v), u \notin T(p_j)} \{c_{u,v}\} - \sum_{w \in Succ(v)} \{c_{v,w}\}$$



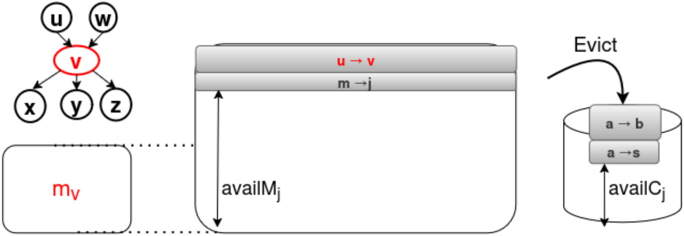
S. Kulagina, HU Berlin

HEFTM-* heuristics: Task Assignment

Greedily assign each task v to processor p_j that minimizes finish time.

- Check that for all predecessors assigned to p_j the data is still in memory $\rightarrow$ otherwise invalid choice
- Address the memory constraint on the processor

$$Res = availM_j - m_v - \sum_{u \in \Pi(v), u \notin T(p_j)} \{c_{u,v}\} - \sum_{w \in Succ(v)} \{c_{v,w}\}$$



S. Kulagina, HU Berlin

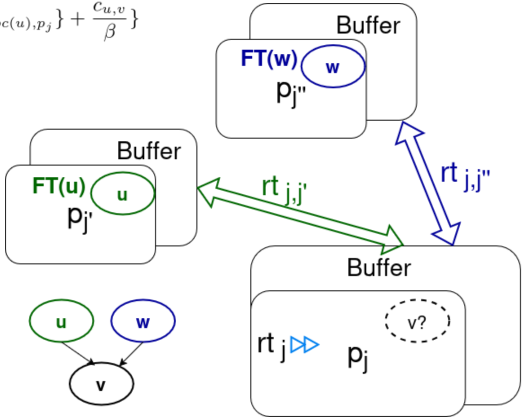
Start and Finish Time Calculation

- Start after all communications have finished

$$ST(v,p_j) = \max\{rt_j, \max_{u \in \text{parent}(v), u \notin T(p_j)} \{FT(u), rt_{proc(u),p_j}\} + \frac{c_{u,v}}{\beta}\}$$

- Finish time

$$FT(v,p_j) = ST(v,p_j) + \frac{w_v}{s_j}$$



S. Kulagina, HU Berlin

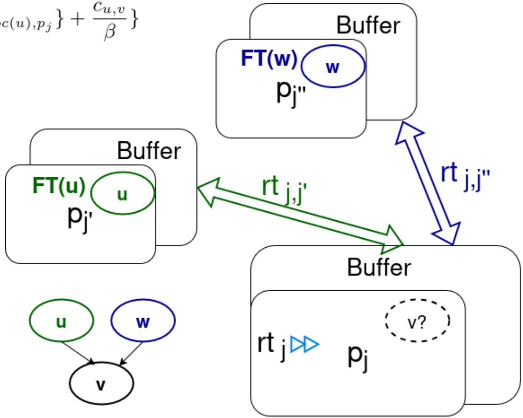
Start and Finish Time Calculation

- Start after all communications have finished

$$ST(v,p_j) = \max\{rt_j, \max_{u \in \text{parent}(v), u \notin T(p_j)} \{FT(u), rt_{proc(u),p_j}\} + \frac{c_{u,v}}{\beta}\}$$

- Finish time

$$FT(v,p_j) = ST(v,p_j) + \frac{w_v}{s_j}$$



S. Kulagina, HU Berlin

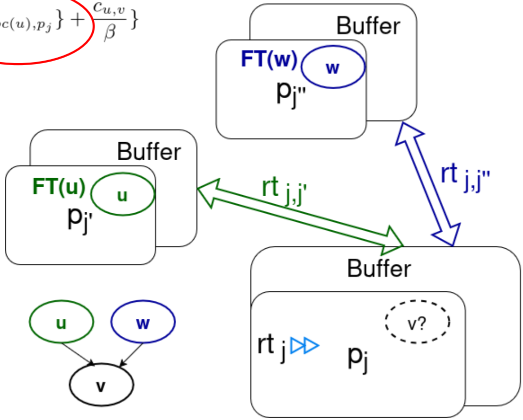
Start and Finish Time Calculation

- Start after all communications have finished

$$ST(v,p_j) = \max\{rt_j, \max_{u \in \text{parent}(v), u \notin T(p_j)} \{FT(u), rt_{proc(u),p_j}\} + \frac{c_{u,v}}{\beta}\}$$

- Finish time

$$FT(v,p_j) = ST(v,p_j) + \frac{w_v}{s_j}$$



S. Kulagina, HU Berlin

Final Assignment

Final assignment to processor

- Evict files that need to be evicted from memory,
  - remove them from pending memory and add to communication buffer
  - reduce buffers sizes accordingly
- New $availM_j$: - eviction, - incoming files, + outgoing files
- For each predecessor on another processor j': ready time on communication buffer $rt_{j,j'}$

Experimental Setup

Schedule real (scientific) workflows

Task and edge weights: historical data from Lotaru [6]

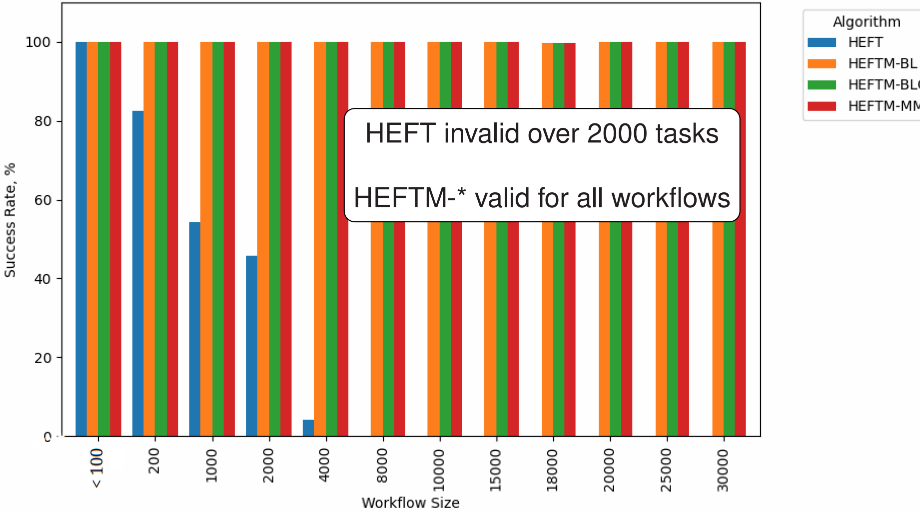
- Workflows
  - Real-world workflows
  - Generated with WFGen [7] from 7 workflow: sizes from 1K to 30K tasks
  - Divided in groups by size
- Execution environments
  - 36 processors: 6 pieces of same 6 kinds of processors as in [6]
  - Memory-constrained: same processors, 10 times less memory

[6] Bader, J., Lehmann, F., Thamsen, L., Will, J., Leser, U., and Kao, O. 2022. Lotaru: Locally Estimating Runtimes of Scientific Workflow Tasks in Heterogeneous Clusters. SSDBM.

[7] Coleman, T., Casanova, H., Pottier, L., Kaushik, M., Deelman, E., and da Silva, R. F. 2022. WfCommons: A framework for enabling scientific workflow research and development. Future Generation Computer Systems.

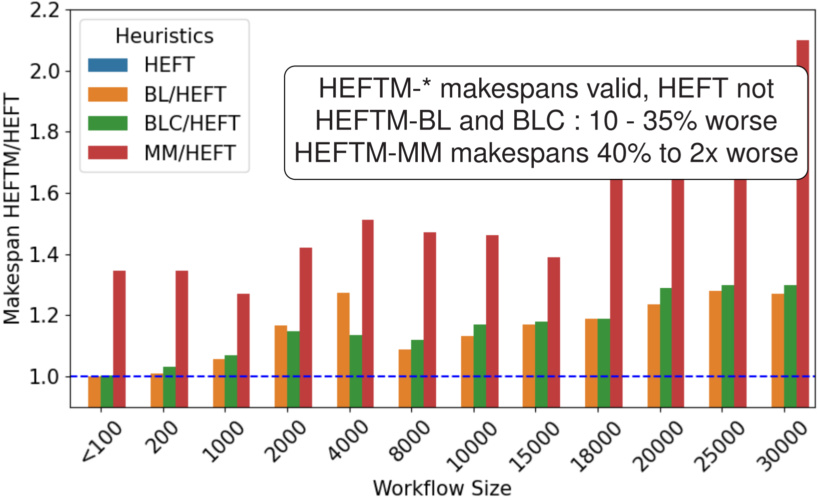
Results

Default cluster - Success Rates



Results

Default cluster - Relative Makespans



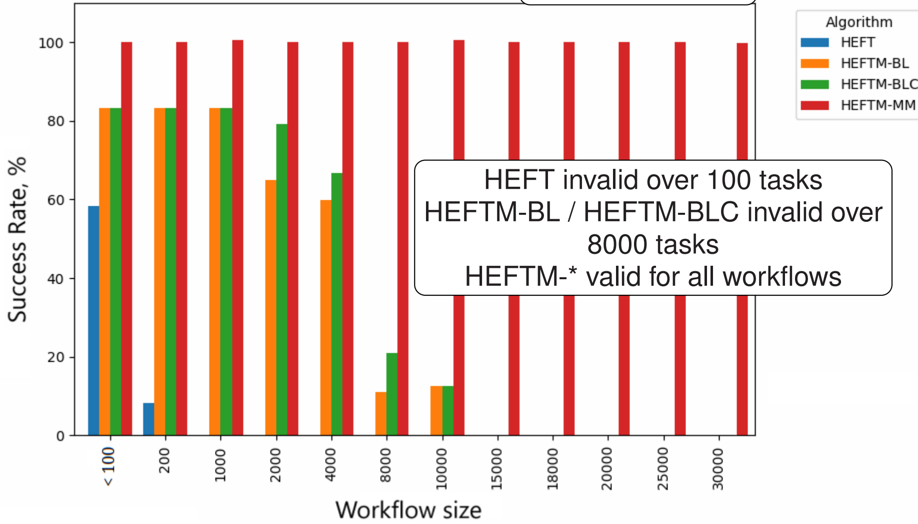
HEFTM-* makespans valid, HEFT not
HEFTM-BL and BLC : 10 - 35% worse
HEFTM-MM makespans 40% to 2x worse

S. Kulagina, HU Berlin

Results

Memory-Constrained Cluster: Success Rates

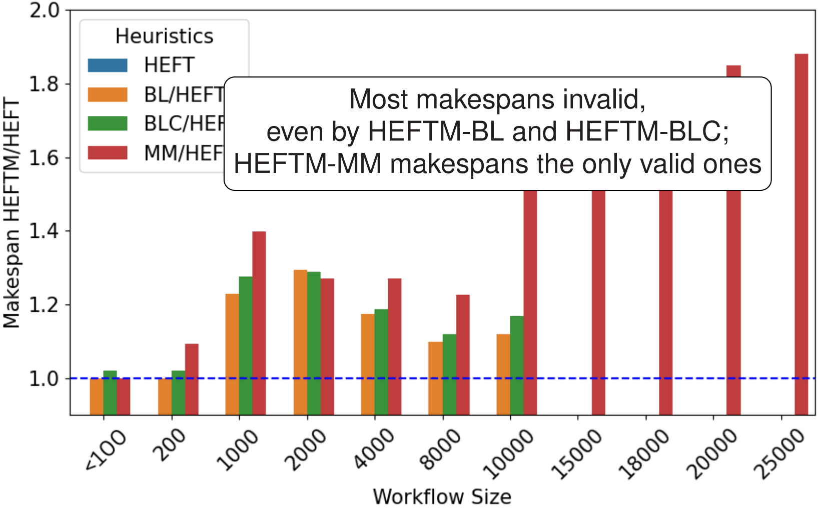
Less realistic, extreme strain for the algorithm



HEFT invalid over 100 tasks
HEFTM-BL / HEFTM-BLC invalid over 8000 tasks
HEFTM-* valid for all workflows

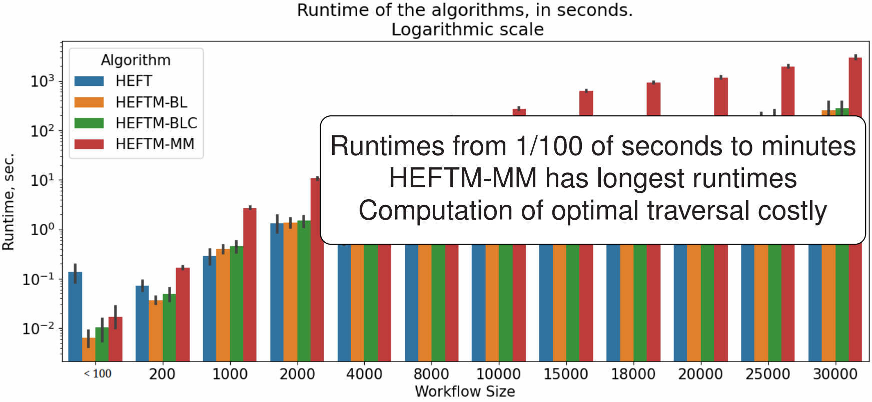
Results

Memory-Constrained Cluster: Relative Makespans



Most makespans invalid,
even by HEFTM-BL and HEFTM-BLC;
HEFTM-MM makespans the only valid ones

Results
Runtime



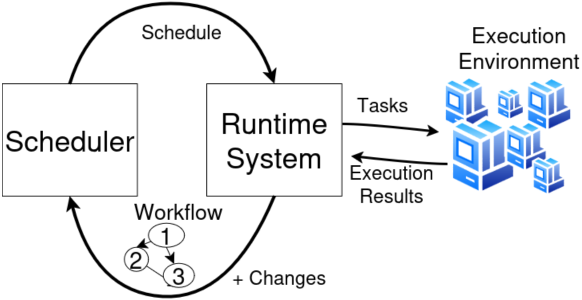
Runtimes from 1/100 of seconds to minutes
HEFTM-MM has longest runtimes
Computation of optimal traversal costly

Adaptive scenario

- No perfect knowledge of task runtimes and memory usage
- Workflow changes: more/less memory, more/less time to execute

Adaptive scenario

- No perfect knowledge of task runtimes and memory usage
- Workflow changes: more/less memory, more/less time to execute

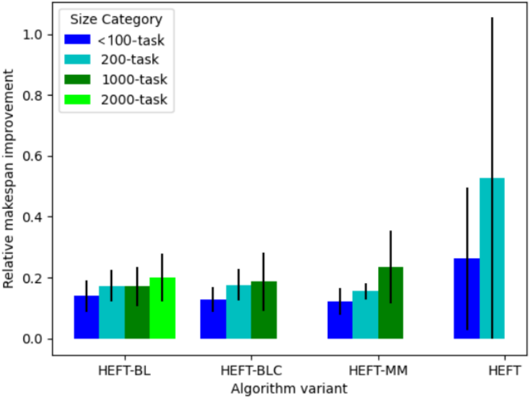


Runtime reports changes

- Can invalidate - not enough memory anymore
- Can lead to later finish time

Adaptive Scenario

No recomputation: the majority of schedules becomes invalid!



Summary

HEFTM-*: memory-valid schedules for each taste!

- Extend an influential predecessor (HEFT)
- Memory-aware, communicate over communication buffers
- Valid makespans, only 10-35% worse than invalid HEFT makespans
- HEFT-**MM** proves unique resilience in memory-constrained scenario

Summary

HEFTM-*: memory-valid schedules for each taste!

- Extend an influential predecessor (HEFT)
- Memory-aware, communicate over communication buffers
- Valid makespans, only 10-35% worse than invalid HEFT makespans
- HEFT-**MM** proves unique resilience in memory-constrained scenario

Scenario	**Winner**
Memory very constrained	HEFT-**MM**
Memory somewhat constrained, need smallest makespan	HEFTM-**BL**
Need balance, willing to trade (very little) makespan for less memory requirement	HEFTM-**BLC**
Lots of memory, need smallest makespan	~~HEFT~~ HEFTM-**BL**

Thank you!

This work is partially supported by Collaborative Research Center (CRC) 1404 FONDA – Foundations of Workflows for Large-Scale Scientific Data Analysis, which is funded by German Research Foundation (DFG).

Future Work

TBD

Previous Work

TBD

- Tree-shaped, then DAG-shaped workflows
- Makespan minimization, memory-aware
- Mapping only, no adaptive scenarios

	Gou et al. [2]	HetPart1 [3]	HetPart2 [4]
Memory	homogeneous	heterogeneous	heterogeneous
Processor speeds	homogeneous	homogeneous	heterogeneous

[2] Gou, C., Benoit, A., Marchal, L.: Partitioning tree-shaped task graphs for distributed platforms with limited memory. IEEE Trans on Par and DistSystems 31(7), (2020)

[3] Kulagina, S., Meyerhenke, H., Benoit, A.: Mapping Tree-shaped Workflows on Memory-heterogeneous Architectures. HeteroPar 2022

[4] Kulagina, S., Meyerhenke, H., Benoit, A.: Mapping Tree-shaped Workflows on Systems with Different Memory Sizes and Processor Speeds. Wiley's Concurrency and Computation Practice and Experience