Scheduling Data-Intensive Scientific Workflows with Reduced Communication

Ilia Pietri
University of Athens, Greece

Rizos Sakellariou
The University of Manchester, UK

ABSTRACT

Data-intensive scientific workflows, typically modelled by directed acyclic graphs, consist of inter-dependent tasks that exchange significant amounts of data and are executed on parallel/distributed clusters. However, the energy or monetary costs associated with large data transfers between tasks executing on different nodes may be significant. As a result, there is scope to explore the possibility of trading some communication for computation, aiming to reduce overall communication costs. In this work, we propose a scheduling approach that scales the weight of communication to increase its impact when building the schedule of a scientific workflow; the aim is to assign pairs of tasks with significant data transfers to the same computational node so that the overall communication cost is minimized. The proposed approach is evaluated using simulation and three real-world scientific workflows. The tradeoff between scientific workflow execution time and the size of data transfers is assessed for different weights and a different number of computational nodes.

CCS CONCEPTS

• **Computing methodologies** → *Distributed algorithms*;

KEYWORDS

Data-intensive workflows, workflow scheduling, communication

ACM Reference Format:
Ilia Pietri and Rizos Sakellariou. 2018. Scheduling Data-Intensive Scientific Workflows with Reduced Communication. In *SSDBM '18: 30th International Conference on Scientific and Statistical Database Management, July 9–11, 2018, Bozen-Bolzano, Italy.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3221269.3221298

1 INTRODUCTION

Scientific workflows [12], typically modelled by Directed Acyclic Graphs (DAGs), consist of inter-dependent tasks that communicate through the use of files; the output data of a task may be used as input from a subsequent task. Such workflows may consist of several thousands of tasks and are extensively executed on large-scale parallel/distributed platforms, including clusters or clouds, so that independent tasks can be processed in parallel and the workflow makespan (overall execution time) can be reduced. When tasks are executed on different computational nodes, intermediate files generated during the workflow execution can be either stored on a shared storage system to which all nodes have access or transferred directly between the nodes. If the files required for the execution of a task are not already available on the node where the task will run, the files need to be transferred. Such data transfers may be significantly costly not only in terms of execution time, but most importantly in terms of energy and money [3, 7].

Scientific workflow scheduling [10] as well as DAG scheduling [2] have traditionally focused on minimizing overall execution time or cost, using communication costs simply to determine the impact of data transfers on the execution of tasks. Data-related aspects have been considered but as part of the overall scheduling optimization process. For instance, the approach in [7] also accounts for data transfer costs in the model used, while the work in [1] tries to minimize the number of data transfers for workflow ensembles by taking advantage of data caching and file locality. Considering storage constraints and storage usage minimization is another direction of workflow scheduling approaches [8]. Finally, there is a body of work that focuses on data placement to distribute the files across the sites used [3, 4, 16], addressing the problems of task assignment and data placement separately or in an integrated manner.

On the quest to exascale and as scientific applications rely on an ever larger set of data, it becomes important to focus on the increased communication complexity and its associated costs. As noted in [9], there is a need "to design algorithms that communicate as little as possible". In turn, this suggests that, in order to minimize communication costs, one can envision that there might be scope to trade computation with communication, whereby some extra effort in computation may be justified if it can result in significant communication savings. This paper explores this premise further by proposing a scientific workflow scheduling algorithm, which extends the widely used Heterogeneous Earliest Finish Time (HEFT) algorithm [13], to obtain different tradeoffs between execution time and communication costs. The idea is to boost the weight of communication when building a schedule so that the chance of placing pairs of inter-dependent tasks which are communication-demanding on different nodes is minimized. The benefits of the proposed algorithm are investigated using synthetic data from three real-world scientific workflows.

2 THE PROPOSED APPROACH

This work considers scientific workflows, which can be modelled as a Directed Acyclic Graph (DAG), where the nodes of the graph represent computational tasks and the edges the flow of data transfers between tasks. We mainly focus on data-intensive workflows where computation and communication costs are comparable, as in [7]. In the model considered, the user is interested in achieving a good tradeoff between workflow execution time and communication costs.

SSDBM '18, July 9–11, 2018, Bozen-Bolzano, Italy

ACM ISBN 978-1-4503-6505-5/18/07…$15.00
https://doi.org/10.1145/3221269.3221298

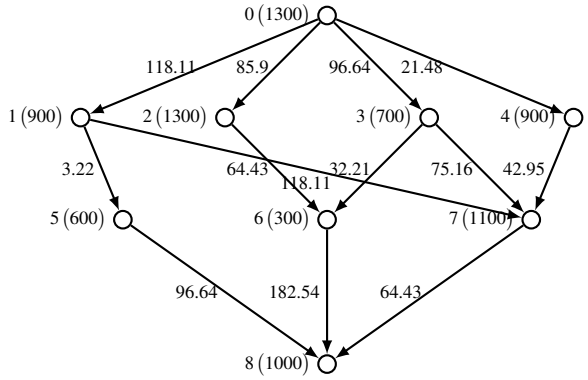


Figure 1: Example DAG with nine tasks.

It is assumed that information on task runtimes and data sizes is provided. Each workflow task can only start after data transfer from all of its predecessors finishes. The communication cost, $commCost_{p \to t}$, from a task p to a subsequent task t is given by the sum of datasizes that need to be transferred divided by the bandwidth, bw, when the two tasks run on different computational nodes, while it is considered to be zero when the two tasks run on the same node. Finally, the workflow makespan is given by the maximum completion time of all tasks.

As mentioned earlier, the idea of this work is to schedule the workflow tasks to the available nodes scaling the communication weight to increase the impact of data transfer overhead on the assignment of tasks and favor the placement of communication demanding interdependent tasks on the same node. To do so, the network bandwidth is skewed using Equation 1:

$$bw = w \cdot bw_{actual}, \tag{1}$$

where w is the given data communication weight and bw_{actual} is the actual network bandwidth. Note that w takes values less than 1 to increase the impact of communication costs on task placement; clearly, smaller weights (close to zero) increase the impact of communication further.

An example with the execution of a DAG with nine tasks on two nodes is used to demonstrate the motivation of this work and describe how the proposed approach works. The example DAG used is shown in Figure 1. Each node in the graph is annotated with the task id and its size in million instructions, *task id (task size)*, and each edge is labeled with the size of data communication (in GB) between the tasks. Figure 2a shows the execution schedule of the DAG obtained using HEFT and assuming a bandwidth between the two nodes of 1 Gbps. For simplicity, two homogeneous nodes with the ability to execute 1 million instructions per second (MIPS) are assumed. The communication time between tasks that run on the same node is considered to be zero. The assignment of tasks onto each of the two nodes (*n0, n1*) shows the task id and its start and finish time (sec): *task id (start_time-finish_time)*. The makespan achieved is 6144.89 seconds, while the overall size of data communication transferred between the two nodes is about 497.14 GBytes.

In order to generate solutions with a lower communication cost, a communication weight ($w = 0.1$) is used to skew the bandwidth

node *n0*	node *n1*
0 (0.0-1300.0)	
2 (1300.0-2600.0)	1 (2224.89-3144.89)
3 (2600.0-3300.0)	4 (3144.89-4044.89)
6 (3300.0-3600.0)	7 (4044.89-5144.89)
5 (3600.0-4200.0)	8 (5144.89-6144.89)
makespan: 6144.89 sec commCost: 497.14 GB	

(a) HEFT schedule for actual bandwidth ($w = 1.0$).

node *n0*	node *n1*
0 (0.0-1300.0)	
2 (1300.0-2600.0)	
3 (2600.0-3300.0)	4 (3017.99-3917.99)
1 (3300.0-4200.0)	
6 (4200.0-4500.0)	5 (4457.7-5057.7)
7 (7353.96.0-8453.96)	
8 (12788.64-13788.64)	

(b) HEFT schedule for skewed bandwidth ($w = 0.1$).

node *n0*	node *n1*
0 (0.0-1300.0)	
2 (1300.0-2600.0)	4 (1471.8-2371.8)
3 (2600.0-3300.0)	
1 (3300.0-4200.0)	
6 (4200.0-4500.0)	5 (4225.77-4825.77)
7 (4500.0-5600.0)	
8 (5600.0-6600.0)	
makespan: 6600.0 sec commCost: 164.28 GB	

(c) Updated schedule for the actual bandwidth.

Figure 2: Execution schedules of the example DAG.

and increase the impact of communication cost on the assignment of the tasks. The obtained schedule for the skewed bandwidth using HEFT is shown in Figure 2b. Based on the placement of the tasks to the nodes, the schedule created is updated for the actual bandwidth (1Gbps) to calculate the start and finish time of each task. The final schedule is shown in Figure 2c. The resulting makespan is 6600 seconds incurring a communication data transfer size of about 164.28 GBytes. As can be seen, the proposed scheduling approach of skewing the bandwidth used for the assignment of the tasks to the nodes allows us to obtain a schedule where overall communication cost is significantly improved (by about 67%) at the expense of a comparatively small increase in workflow makespan (by about 7%).

Algorithm 1 Workflow scheduling for reduced communication

```
1: procedure PARETO-EFFICIENT SCHEDULER(dag, nodeConfs, weights, bw_actual)
2:     solutions ← ∅                          ▷ makespan and communication cost pairs
3:     for ∀conf ∈ nodeConfs do                          ▷ configuration of nodes
4:         for ∀w ∈ weights do
5:             Compute bandwidth bw for weight w using Equation 1
                                                              ▷ Skewed bandwidth
6:             plan_cur ← generateHEFTPlan(dag, conf, bw)
                    ▷ Generate HEFT plan using bandwidth bw for data transfers
7:             plan_new ← update plan_cur for bw_actual
                ▷ Update start and finish times of tasks using the actual bandwidth
8:             Compute makespan and size of communication costs of plan_new
9:             solutions.add(plan_new)
10:        end for
11:    end for
12:    Compute and return the Pareto front of solutions
13: end procedure
```

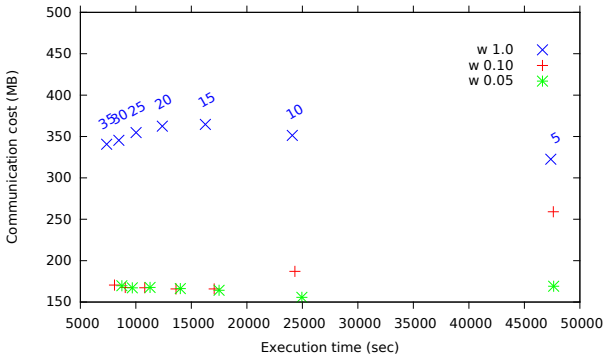
Our proposed scheduling approach to reduce communication is presented in Algorithm 1. The algorithm assigns the workflow tasks to the available nodes using HEFT [13] computing the communication costs based on the skewed bandwidth. The execution schedule built is then updated using the actual bandwidth bw_{actual} to recalculate the start and finish times of the workflow tasks on the assigned nodes. The algorithm computes the workflow makespan in the updated execution schedule and the size of all data transfers during the workflow execution. The procedure described may be repeated for different values of communication weights and node configurations (number of nodes); each time, the algorithm returns workflow makespan and total communication cost. As there are multiple such pairs, one may choose to keep the non-dominated solutions only (exhibiting different tradeoffs between computation and communication), thereby building a Pareto front of optimal solutions.

We note that HEFT [13] is a widely used list scheduling algorithm for DAGs that works in two phases. It initially assigns ranks to the workflow tasks based on an upward ranking to determine the execution order in which the tasks will be scheduled to the available set of nodes. The upward rank of a task is the length of the longest path from the task to an exit task of the workflow taking into account the computation and communication costs of the tasks. Following the ordering of the tasks, the algorithm schedules each task to the slot with the earliest finish time, considering all the possible slots on the available nodes which meet the data dependency constraints.

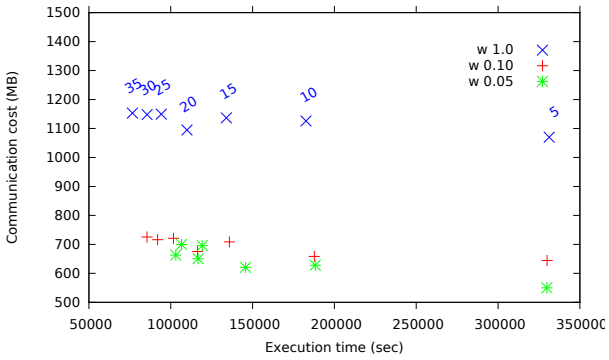
3 EXPERIMENTAL EVALUATION

The simulator in [11] was used to implement and evaluate the proposed scheduling approach. Synthetic data for three real-world scientific workflows, namely LIGO (detection of gravitational waves in the universe) [6], Epigenome (mapping of the epigenetic state of human cells) [14] and Montage (generation of image mosaics of the sky) [5], were generated using the workflow generator in [15] to obtain a workflow of 1000 tasks for each application. A small bandwidth of 1KB/sec is assumed for LIGO and Epigenome and 1MB/sec for Montage to consider data-intensive workflows that exhibit a Communication to Computation Ratio (CCR) larger than 1 (comparable sizes), achieving a CCR of 1.77 for LIGO, 4.19 for Epigenome and 1.27 for Montage. For simplicity, homogeneous nodes with a processing capacity of 1 MIPS are assumed. The number of nodes used for the assignment of the tasks varied between 5 to 35 (using a step of 5) and three different values of communication weights ($w = 1.0, 0.5, 0.01$) were used to investigate the achieved tradeoffs between workflow execution time (sec) and overall size of communication (MB). Clearly, the value $w = 1.0$ corresponds to the schedule built by the original HEFT algorithm.

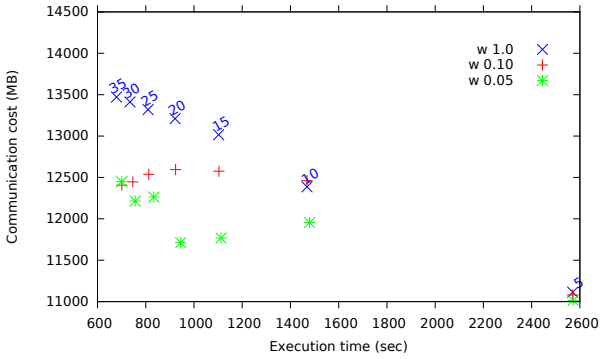
Figures 3a, 3b and 3c show the results for the LIGO, Epigenome and Montage workflows of 1000 tasks, respectively. The tradeoffs achieved for the case of $w = 1.0$ are the results using HEFT and the labels show the number of nodes used at each run. As can be seen, schedules with significantly lower communication costs can be achieved for lower values of w, without greatly affecting workflow makespan. This may be because of the overlap between communication and computation, especially for the case of configurations with a small number of nodes. In several cases, better solutions in terms of both execution time and communication costs may be achieved



(a) A LIGO workflow of 1000 tasks.



(b) An Epigenome workflow of 1000 tasks.



(c) A Montage workflow of 1000 tasks.

Figure 3: Computation vs communication using different scheduling approaches.

by decreasing w like in the case of Epigenome for the schedules with 5 nodes. Also, it can be seen that communication costs do not necessarily increase for schedules with a larger number of nodes. Hence, in several cases a combination of a larger number of nodes and a smaller value of w may be preferred to achieve a good tradeoff between workflow makespan and communication costs.

Table 1 summarizes the solutions achieved using HEFT (labeled as HEFT solutions) and *non-dominated solutions* achieved using our proposed scheduling approach (labeled as *Pareto solutions*). The

Table 1: HEFT vs Pareto-optimal solutions.

(a) LIGO

nodes	HEFT solutions	Pareto solutions	Time	ComCost
	time (sec), comCost (MB)		savings (%)	
5	47380.93, 322.51			
10	24090.39, 351.26			
15	16261.31, 364.65	17497.77, 164.23	−7.60	54.96
20	12384.79, 362.44	13616.24, 165.85	−9.94	54.24
25	10018.23, 354.71			
30	8465.75, 345.42	9044.67, 167.56	−6.84	51.49
		9677.95, 167.14	−14.32	51.61
35	7371.83, 340.676	8738.92, 169.68	−18.54	50.19
		8074.22, 170.47	−9.53	49.96

(b) Epigenome

nodes	HEFT solutions	Pareto solutions	Time	ComCost
	time (sec), comCost (MB)		savings (%)	
5	331271.93, 1070.01	329805.56, 550.03	0.44	48.60
10	182615.74, 1126.06			
15	134011.90, 1136.67	145700.98, 620.73	−8.72	45.39
20	109881.86, 1094.81			
25	94212.61, 1149.43	116737.13, 650.48	−23.91	43.41
30	85539.91, 1147.92	91968.80, 716.26	−7.52	37.60
35	76567.91, 1153.18	103054.69, 663.21	−34.59	42.49
		85498.13, 725.52	−11.66	37.09

(c) Montage

nodes	HEFT solutions	Pareto solutions	Time	ComCost
	time (sec), comCost (MB)		savings (%)	
5	2570.38, 11113.82	2570.02, 11079.89	0.01	0.31
10	1468.55, 12385.99			
15	1102.11, 13015.85			
20	921.74, 13209.75	945.15, 11712.01	−2.54	11.34
25	809.87, 13319.46			
30	734.64, 13412.45	756.27, 12214.75	−2.94	8.93
35	678.79, 13471.16	700.75, 12406.85	−3.24	7.90
		700.40 12448.56	−3.18	7.59

number of computational nodes used, the makespan achieved (sec) and the communication cost required (MB), annotated as *time, comCost*, are given for each solution. The results from the comparison of each Pareto solution with the HEFT solution of the same number of nodes (time and communication cost % savings) are also presented. For example, in the case of Epigenome the proposed approach achieves a Pareto-optimal solution of 145700.98 sec and 620.73 MB for configurations on 15 nodes compared to the HEFT solution of 134011.90 sec and 1136.67 MB on 15 nodes, resulting in 45.39% savings in communication cost at the expense of an increase of 8.72% in execution time. For some configurations, for a given number of nodes, different Pareto-optimal solutions may exist. For example, in the case of LIGO for 30 or 35 nodes, there are two different solutions in each case: the second solution is listed in the row below the first solution keeping the corresponding HEFT part empty. At the same time, if, for a given number of nodes, our proposed scheduling approach gives solutions that are dominated by another solution (even if this is obtained with a different number of nodes or the original HEFT), the corresponding entry of the 'Pareto solutions' column in the table is left blank.

Overall, it can be seen that the % savings in communication cost outweigh the % increase in makespan in all cases; in two cases the makespan is also improved by the Pareto-optimal solution obtained by our proposed scheduling approach. Finally, higher savings in communication cost are achieved for Epigenome and LIGO, while in the case of Montage the savings are much smaller. This may be because although Montage is I/O bound spending most of its runtime on I/O operations, several job classes that consist of a large number of parallel tasks read and write small files.

4 CONCLUSION

This paper addressed the problem of workflow/DAG scheduling in a way that minimizes communication without overly increasing computation costs. The motivation stems from the envisioned high-end computing challenges where energy due to communication may come at a premium. A scheduling approach that attempts to increase the impact of communication on the placement of tasks to generate schedules with lower data transfers has been proposed. Preliminary results are very promising giving high % savings in communication for comparatively lower % increases in computation. Future work could investigate how to refine the set of configurations (weights and number of nodes) that the proposed algorithm considers to generate good execution time and communication cost tradeoffs for different types of real-world scientific workflows and execution environments.

REFERENCES

[1] Piotr Bryk, Maciej Malawski, Gideon Juve, and Ewa Deelman. 2016. Storage-aware Algorithms for Scheduling of Workflow Ensembles in Clouds. *Journal of Grid Computing* 14, 2 (2016), 359–378.

[2] Louis-Claude Canon, Emmanuel Jeannot, Rizos Sakellariou, and Wei Zheng. 2008. Comparative evaluation of the robustness of DAG scheduling heuristics. In *Grid Computing: Achievements and Prospects*. Springer.

[3] Ümit V. Çatalyürek, Kamer Kaya, and Bora Uçar. 2011. Integrated Data Placement and Task Assignment for Scientific Workflows in Clouds. In *Proceedings of the 4th International Workshop on Data-intensive Distributed Computing*. ACM, 45–54.

[4] Ann Chervenak, Ewa Deelman, Miron Livny, Mei-Hui Su, Rob Schuler, Shishir Bharathi, Gaurang Mehta, and Karan Vahi. 2007. Data placement for scientific applications in distributed environments. In *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*. IEEE, 267–274.

[5] D. S. Katz, J. C. Jacob, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, G. B. Berriman, J. Good, A. C. Laity, and T. A. Prince. 2005. A Comparison of Two Methods for Building Astronomical Image Mosaics on a Grid. In *Proceedings of the IEEE International Conference on Parallel Processing Workshops*. IEEE, 85–94.

[6] LIGO project, Laser interferometer gravitational wave observatory. 2017. (2017). http://www.ligo.caltech.edu/

[7] S. Pandey, L. Wu, S. M. Guru, and R. Buyya. 2010. A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments. In *Proceedings of the 24th International Conference on Advanced Information Networking and Applications*. IEEE, 400–407.

[8] A. Ramakrishnan, G. Singh, H. Zhao, E. Deelman, R. Sakellariou, K. Vahi, K. Blackburn, D. Meyers, and M. Samidi. 2007. Scheduling Data-Intensive Workflows onto Storage-Constrained Distributed Resources. In *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid*. IEEE, 401–409.

[9] Daniel A. Reed and Jack Dongarra. 2015. Exascale Computing and Big Data. *Commun. ACM* 58, 7 (2015), 56–68.

[10] M. A. Rodriguez and R. Buyya. 2017. A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. *Concurrency and Computation: Practice and Experience* 29, 8 (2017).

[11] Cloud Workflow Simulator. 2013. (2013). https://github.com/malawski/cloudworkflowsimulator

[12] Ian J. Taylor, Ewa Deelman, Dennis Gannon, and Matthew Shields. 2007. *Workflows for e-Science*. Springer.

[13] H. Topcuoglu, S. Hariri, and Min-You Wu. 2002. Performance-Effective and Low-complexity Task Scheduling for Heterogeneous Computing. *IEEE Transactions on Parallel and Distributed Systems* 13, 3 (2002), 260–274.

[14] USC Epigenome Center. 2017. (2017). http://epigenome.usc.edu

[15] Workflow Generator. 2013. (2013). https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator

[16] D. Yuan, Y. Yang, X. Liu, and J. Chen. 2010. A Data Placement Strategy in Scientific Cloud Workflows. *Future Generation Computer Systems* 26, 8 (2010).