Scheduling for gathering multitype data

Joanna Berlińska (Speaker) * Bartłomiej Przybylski †

1 Introduction

Scheduling in data gathering networks has been attracting an increasing interest in recent years. Algorithms were proposed, e.g., for maximizing network lifetime [1], minimizing data compression cost [2, 6] and minimizing maximum dataset lateness [3]. In this work, we depart from two common assumptions: that all required datasets are located on remote computers, and that the whole available data have to be collected. We analyze gathering required amounts of data of different types, with the possibility of generating additional datasets.

Our research is motivated by the following scenario. A number of datasets containing randomized simulation results are available in distributed locations. Each dataset is characterized by a *type*, which describes the input and algorithm parameters used for simulations that produced the data. In order to provide a meaningful analysis of the impact of parameter values on the results, we have to collect appropriate numbers of datasets of certain types on a single computer. It is possible that the number of datasets of some type generated so far is insufficient. The remote nodes are no longer available for computations, but additional results can be produced locally. It is assumed that all datasets have equal sizes, and hence, the time required to download a dataset depends only on its initial location. However, using different parameter values may result in varying simulation time, and hence, the time necessary to produce a dataset depends on its type. Our goal is to gather the required data (either by downloading them from remote nodes, or by running computations on a local processor) in the shortest possible time.

2 Problem formulation

We analyze a star network consisting of a central computer P_0 and m nodes $P_1, \ldots, P_m$ holding data. There exist n types of datasets, denoted by numbers $j = 1, \ldots, n$. For each j, we have to gather K_j datasets of type j on P_0. The

*Joanna.Berlinska@amu.edu.pl. Faculty of Mathematics and Computer Science, Adam Mickiewicz University in Poznań, Umultowska 87, 61-614 Poznań, Poland

†bap@amu.edu.pl. Faculty of Mathematics and Computer Science, Adam Mickiewicz University in Poznań, Umultowska 87, 61-614 Poznań, Poland

number of datasets of type j located on node P_i, where $i = 1, \ldots, m$, is k_{ij}. Downloading one dataset from node P_i to P_0 takes time C_i. Only one dataset can be transferred at a time. The time required to generate one dataset of type j on the server is denoted by A_j. Note that the order in which datasets are downloaded or produced does not affect the total time of communication or computation. Thus, the scheduling problem is to decide which datasets should be downloaded from which nodes, and how many datasets of each type should be generated on P_0, in order to minimize the total data gathering time T.

In other words, we have to execute n jobs, where job j consists in obtaining K_j datasets of type j on P_0, for $j = 1, \ldots, n$. Since only one dataset can be downloaded at a time, the communication network can be seen as the first available machine, while processor P_0 is the second machine. Naturally, it is possible to generate and download datasets of the same type in parallel. Thus, our problem resembles scheduling preemptive work-preserving malleable jobs [4] on two machines. However, there are substantial differences between these two problems. Firstly, our first machine does not have a constant speed, because it consists of many nodes with possibly different communication capabilities. Secondly, in our problem preemptions are possible only after an integer number of datasets have been downloaded (on the first machine) or generated (on the second machine).

3 Results

We first show that our problem is NP-hard. Indeed, suppose that $m = n$, $K_j = 1$, $k_{jj} = 1$ for $j = 1, \ldots, n$, and $k_{ij} = 0$ for $i \neq j$. Let $A_j = C_j = p_j$ for each $j = 1, \ldots, n$. In this case, each job j consists in obtaining just one dataset, and hence, it is non-preemptable. Its execution time is p_j independently of the chosen machine. Thus, we have to solve the NP-hard problem $P2||C_{max}$ [5]. Hence, our problem is also NP-hard, as its generalization.

Let x_{ij} be integer variables representing the number of datasets of type j downloaded from node P_i, for $i = 1, \ldots, m$, $j = 1, \ldots, n$. Our problem can be formulated as an integer linear program in the following way.

$$\text{minimize} \quad T \tag{1}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} C_i x_{ij} \leq T \tag{2}$$

$$\sum_{j=1}^{n} A_j (K_j - \sum_{i=1}^{m} x_{ij}) \leq T \tag{3}$$

$$0 \leq x_{ij} \leq k_{ij} \quad \text{for} \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \tag{4}$$

In the above program, we minimize the schedule length T. Constraints (2) guarantee that all data transfers fit in time T, and by (3) the required, yet not downloaded, datasets are generated on P_0 within time T. Inequalities (4) ensure that we do not download from P_i more datasets of type j than available.

Program (1)-(4) can also be used to obtain a polynomial 2-approximation algorithm for our problem. To this end, we solve its relaxed version with rational variables x_{ij}, and then round the obtained values of x_{ij} to the nearest integers.

We propose the following algorithm for the case when $C_i = C$ for $i = 1, \ldots, m$. Note that in this case we can replace nodes $P_1, \ldots, P_m$ by a single node holding $k_j = \sum_{i=1}^{m} k_{ij}$ datasets of each type j. First, we construct a schedule with the maximum number of downloaded datasets, i.e. we transfer $\min\{K_j, k_j\}$ datasets of type j and generate $\max\{0, K_j - k_j\}$ datasets of this type. Then, we go through a list of dataset types sorted according to nondecreasing computation costs A_j, and check how many datasets of a given type should be produced instead of downloading them, in order to decrease the makespan as much as possible. We change the schedule accordingly, and move to the next type. This procedure stops when no improvement is possible, yielding an optimum solution in $O(mn+n\log n)$ time.

A similar greedy approach can be used to solve the problem if C_i are arbitrary, but $A_j = A$ for all $j = 1, \ldots, n$. Again, we start with a schedule that maximizes the number of downloaded datasets. Obviously, if datasets of a given type can be downloaded from many nodes, we prefer the nodes with lower communication costs. Afterwards, we iterate over the nodes in the order of nonincreasing communication costs C_i, and replace downloading datasets they hold by producing them on P_0, as long as this shortens the schedule. This algorithm runs in $O(mn + m\log m)$ time.

As an outlook for future work, we analyze the properties of a more general problem, where datasets of the same type require additional merging and processing after being gathered.

References

[1] J. Berlińska (2014). Communication scheduling in data gathering networks with limited memory. *Applied Mathematics and Computation* 235, 530-537.

[2] J. Berlińska (2015). Scheduling for data gathering networks with data compression. *European Journal of Operational Research* 246, 744-749.

[3] J. Berlińska (2018). Scheduling Data Gathering with Maximum Lateness Objective. In: R. Wyrzykowski et al. (ed.), *Parallel Processing and Applied Mathematics: 12th International Conference PPAM 2017, Part II*, LNCS 10778, 135-144. Springer, Cham.

[4] M. Drozdowski (2009). *Scheduling for Parallel Processing*. Springer, London.

[5] M.R. Garey and D.S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.

[6] W. Luo, Y. Xu, B. Gu, W. Tong, R. Goebel and G. Lin (2018). Algorithms for Communication Scheduling in Data Gathering Network with Data Compression. *Algorithmica* 80(11), 3158-3176.