Scheduling Divisible Loads to Optimize the Computation Time and Cost

Natalia V. Shakhlevich

School of Computing, University of Leeds, Leeds LS2 9JT, U.K.
n.shakhlevich@leeds.ac.uk

Abstract. Efficient load distribution plays an important role in grid and cloud applications. In a typical problem, a divisible load should be split into parts and allocated to several processors, with one processor responsible for the data transfer. Since processors have different speed and cost characteristics, selecting the processor order for the transmission and defining the chunk sizes affect the computation time and cost. We perform a systematic study of the model analysing the properties of Pareto optimal solutions. We demonstrate that the earlier research has a number of limitations. In particular, it is generally assumed that the load should be distributed so that all processors have equal completion times, while in fact this property is satisfied only for some deadlines; for many optimal schedules this property does not hold. Moreover, fixing the processor sequence in the non-decreasing order of the cost-characteristic may be appropriate only for Pareto-optimal solutions with relatively large deadlines; optimal schedules for tight deadlines may have a different order of processors. We conclude with an efficient algorithm for finding the time-cost trade-off.

Keywords: Scheduling, Divisible Load, Time/cost Optimization.

1 Introduction

Parallel computer systems have given rise to new scheduling models that go beyond the classical scheduling theory. While in a traditional scheduling model a task can be processed by one machine at a time, a new feature of multiprocessor computations is the ability to split tasks into several parts and to process them simultaneously by different processors, see, e.g., [8,14]. An additional feature of modern Grid computing and cloud computing systems is the introduction of the cost factor, see, e.g. [4,11,16]. This study is motivated by the lack of theoretical research in the area and some inaccuracies which can be found in the earlier research.

We consider the network model described in [13]. There is a set $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$ of m processors connected via a bus type communication medium. One processor of the set $\mathcal{P}$ is selected as a master processor to receive a divisible load of size τ and to divide it into portions of size $\alpha_1\tau$, $\alpha_2\tau$,..., $\alpha_m\tau$, $\sum_{k=1}^{m} \alpha_k = 1$, which are then transmitted to slave processors from $\mathcal{P}$ to perform required computations.

J. Altmann, K. Vanmechelen, and O.F. Rana (Eds.): GECON 2013, LNCS 8193, pp. 138–148, 2013.

The processors have different computation speeds and for each processor $P_k \in \mathcal{P}$ the inverse of the speed w_k is given. This implies that the load of size $\alpha_k \tau$ allocated to processor P_k requires computation time $\alpha_k w_k \tau$.

If P_1 is selected as a master processor and the transmission sequence is P_2, P_3, ..., P_m, then P_1 can start processing its own load of size $\alpha_1 \tau$ at time 0 and at the same time it can start transmitting the relevant portions of the load first to P_2, then to P_3, etc., until the last portion is transmitted to P_m, see Fig. 1. If z is the time needed to transmit the whole load of size τ, then the communication time for transmitting the portion $\alpha_k \tau$ to processor P_k is $\alpha_k z$.

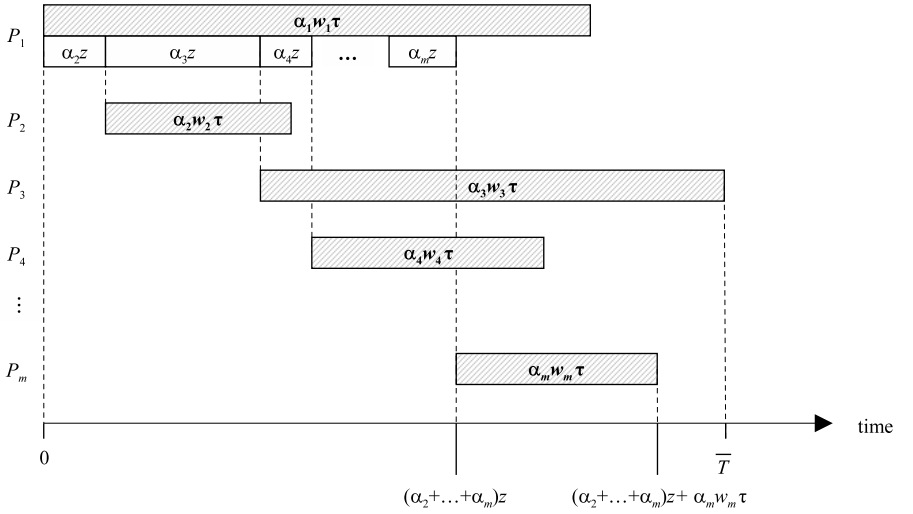

Fig. 1. An example of a schedule with master processor P_1 and transmission sequence $P_2, \ldots, P_m$

With the selected transmission order, processor P_1 completes its portion of computation at time

$$T_1 = \alpha_1 w_1 \tau. \tag{1}$$

Processor P_k, $2 \leq k \leq m$, receives its portion of the load at time $\sum_{i=2}^{k} \alpha_i z$ and immediately after that it can start computation, which takes $\alpha_k w_k \tau$ time. Thus processor P_k completes its portion of the load at time

$$T_k = \sum_{i=2}^{k} \alpha_i z + \alpha_k w_k \tau.$$

The finish time T of the load is defined as the *makespan* of the schedule; it is equal to the maximum completion time among all processors,

$$T = \max_{1 \leq k \leq m} \{T_k\}. \tag{2}$$