Scheduling Strategies of Divisible Loads in DIN Networks

Ligang Dong, Lek Heng Ngoh, and Joo Geok Tan

Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
{donglg, lhn, tjg}@i2r.a-star.edu.sg

Abstract. The DIN network is a special network, in which data can "live" by circulating. Circulating data can be rapidly accessed by a large number of hosts connected to the network. Scheduling strategies of processing divisible loads among hosts in DIN networks is studied to minimize the total processing time. We derive the closed-form solution of the processing time. Moreover, our experiment shows that the performance of our scheduling strategy can reduce the process time of a load up to previous 60%.

1 Introduction

In a distributed computing system, a load is often computed/processed among several cooperative computer hosts/processors, so that the processing time of the entire load can be minimized. One of the main methods is to design efficient scheduling strategies [6]. In detail, through the mathematical modelling based on the parameters of networks and hosts, the optimal size and the starting time of load fractions to be distributed among the hosts are calculated. The divisible load may be either arbitrarily divisible or grain-based (i.e., there is a minimum size of data that can be computed independently). We consider only arbitrarily divisible load in this paper. In the past few years, there is a rich development on scheduling divisible loads because of the wide availability of divisible loads, e.g., military and space information, image/vision data, matrix vectors, queries. These data can be treated to be divisible in quite a certain extent. Early study results (from 1988 to 1996) can be found in [1][4]. A compendium of relative papers is available in [5].

In this paper, we study the scheduling strategy of divisible loads in a special network - DIN networks (shown in Fig. 1). The motivation of proposing DIN networks is to make use of the expected proliferation of optical bandwidth and to solve the bottleneck problem of accessing highly popular data from the server I/O. The bottleneck includes "narrow" I/O bandwidth and "long" access time. We let data circulate in a fiber loop, called *DIN loop*. Through a modified router, called *DIN node*, data can be written into or read from the DIN loops. Since servers, which have relatively slow response and small throughput, are not included in the access process, *more* users can *rapidly* access data from DIN networks.

H. Kosch, L. Böszörményi, H. Hellwagner (Eds.): Euro-Par 2003, LNCS 2790, pp. 236–240, 2003.

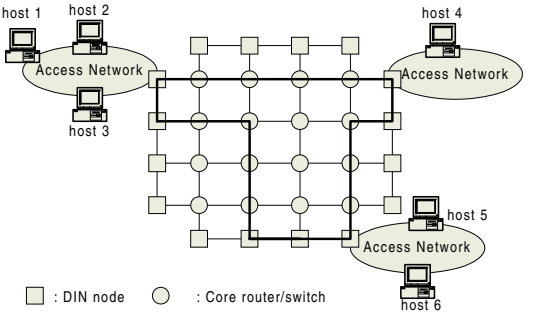

Fig. 1. Structure of DIN networks

Previous scheduling strategy still can be used for DIN networks. However, we propose a novel scheduling strategy that has much better performance than before. Our scheduling strategy has two features. In past scheduling strategy, the allocated fraction of the load is not processed until entire allocated load fraction is received. In other words, when the data is being received, the host does not process data. In our strategy, we let the host process data and receive data *simultaneously*. This strategy can be realized when the hosts are equipped with *front-ends* to communicate [1]. This is the first feature. The second feature is that, unlike traditional scheduling strategy, the originating host of the load does not *directly* transmit the load fractions to hosts. Instead, the load is transmitted to and stored in DIN loops. If one host would like to process the load, then this host can retrieve one load fraction from the DIN loop. Because the originating host of the load *does not* need to partition the load, the originator of the load does not need to know where are the hosts and their parameters (e.g., computing ability of hosts).

2 Analysis of Scheduling Strategy in Processing Arbitrarily Divisible Load

Our scheduling strategy is described as follows. The divisible load is assumed to originate at a host (referred as original host (OH)), which injects the load (in the form of data) into a DIN loop through a DIN node. Thus, the load will *circulate* in the loop. If a host would like to process a part of the load, then this host can retrieve a load fraction via a DIN node, which is in the same access work with the host. In this way, the load is distributed among m hosts, every of which processes a part of the load. For instance, the host p_i processes data with a total size of α_i.

The notation used in this analysis can be summarized as follows. m: Number of hosts that compute/process the load. α_i: Total amount of the load computed/processed in the host p_i. $\sum_{i=1}^{m} \alpha_i = 1$. E_i: Time taken to compute/process a unit load by the host p_i. C: Time taken to transmit a unit load to the loop in DIN by OH. θ: A constant additive communication overhead associated with a communication process by OH.

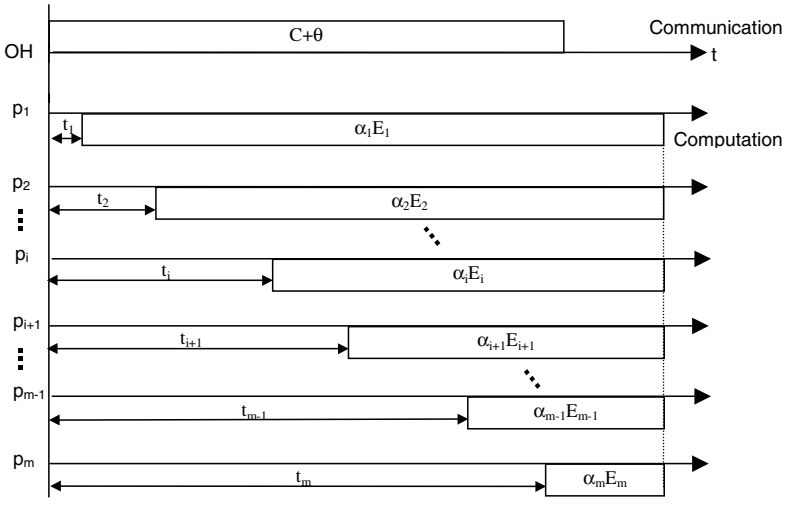

Fig. 2. Timing diagram for processing the arbitrarily divisible load with m hosts

We show the timing diagram for our scheduling strategy in Fig. 2. The traditional scheduling strategy can be found in [6]. In our scheduling strategy, we propose a *starting condition* of hosts: The host p_i cannot start the computation unless there are available data with the size of $i\alpha_0$ in the system. Here, *available data* are referred to the data that have been transferred to the DIN loop and have not computed. Actually, this starting condition is not necessary in a real system. However, with this condition, our scheduling can be analyzable. From Fig. 2, we easily calculate t_i, α_i for $i = 1, ..., m$. The detail is ignored because of the limited space. Finally, we obtain the processing time as $T = t_m + \alpha_m E_m$.

3 Performance Comparison

We now compare the processing time of the traditional scheduling strategy [6] (referred as Strategy I) and the proposed scheduling strategy (Strategy II) through the numerical experiment. Here, our numerical experiments follow the approach in [6]. Input parameters and their default values for the numerical experiment are as follows: $\alpha_0 = 0.01$, $\theta = 0.01$, $C = 0.3$, E_i is uniform distributed in $[0.5, 1.5]$. We carry out an extensive simulation. Owing to limited space, we

Table 1. Comparison between the proposed strategy and the traditional strategy

$C = 0.3$							$C = 0.4$						
m	T^I	T^{II}	T^{II}/T^I	X^I	X^{II}	β	m	T^I	T^{II}	T^{II}/T^I	X^I	X^{II}	β
1	1.539	1.242	80%	0.000	0.000	0.013	1	1.639	1.243	75%	0.000	0.000	0.014
2	0.833	0.598	71%	0.008	0.003	0.004	2	0.914	0.600	65%	0.008	0.005	0.006
3	0.637	0.418	65%	0.027	0.014	0.006	3	0.714	0.422	59%	0.028	0.027	0.013
4	0.542	0.328	60%	0.059	0.044	0.012	4	0.619			0.062	9.923	3.963
5	0.455	0.316	69%	0.106	0.496	0.139	5	0.537			0.115		
6	0.430			0.191	0.464	-0.007	6	0.514			0.217		
7	0.416			0.307			7	0.502			0.362		
8	0.402			0.456			8	0.493			0.559		
9	0.397			0.701			9	0.491			0.911		

only show part results here. Tables 1 is the scheduling result when $C = 0.3$ and $C = 0.4$. From Table 1, firstly, if Strategies I and II adopt the same hosts to compute the load, Strategy II has a $20\% - 40\%$ improvement in comparison with Strategy I. Secondly, when the number of hosts is not limited, Strategy I is apt to make use of more hosts than Strategy II for the computation. Even in this case, Strategy II shows better performances than Strategy I.

4 Conclusion

Our main contribute is to propose an efficient and dynamic scheduling strategy of divisible loads for DIN networks. Although past scheduling strategies can be used, our scheduling strategy has better performances. Two features of our scheduling strategy can be summarized as follows. First, because of simultaneous receiving and processing of data, the processing time is reduced up to 60% of previous scheduling result. Second, we make full use of the feature of DIN networks, i.e., high throughput and short access time. We let DIN loops store the load to be processed, so that the load can be rapidly, dynamically balanced among the hosts.

References

1. V. Bharadwaj, D. Ghose, V.Mani, and T.G. Robertazzi, "Scheduling Divisible Loads in Parallel and Distributed Systems," *IEEE Computer Society Press*, Los Alamitos CA, Sep. 1996.
2. V. Bharadwaj and N. Viswanadham, "Sub-Optimal Solutions Using Integer Approximation Techniques for Scheduling Divisible Loads on Distributed Bus Networks," *IEEE Transactions on Systems, Man, and Cybernatics: Part A*, Vol.30(6), pp.680–691, Nov. 2000.
3. V. Bharadwaj, D. Ghose, and T.G. Robertazzi, "A New Paradigm for Load Scheduling in Distributed Systems," Accepted for publication in a *Special Issue in Cluster Computing*, to appear spring 2003.

4. M. Drozdowski, "Selected Problems of Scheduling Tasks in Multiprocessor Computer Systems," Politechnika Poznanska, Book No.321, Poznan, Poland, 1997.
5. Website about divisible load scheduling, http://www.ee.sunysb.edu/ tom/dlt.html.
6. B. Veeravalli, Xiaolin Li, and Chi Chung Ko, "On the influence of start-up costs in scheduling divisible loads on bus networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol.11(12), Dec. 2000.