ELSEVIER

Discrete Applied Mathematics 76 (1997) 21–41

DISCRETE APPLIED MATHEMATICS

Distributed processing of divisible jobs with communication startup costs☆

Jacek Błażewicz,* Maciej Drozdowski

Institute of Computing Science, Poznań University of Technology, ul. Piotrowo 3a, 60-965 Poznań, Poland

Received 3 October 1995; revised 18 December 1995

Abstract

In this work we analyze the problem of an optimal distribution of a computational task among a set of processors. We assume that the task can be arbitrarily divided and its parts can be processed in parallel on different processors. A wide range of interconnection architectures of distributed computer systems is taken into consideration: a chain, a loop, a tree, a star of processors, a set of processors using shared buses, and a hypercube of processors. It is assumed that the communication time is equal to some startup value plus some amount proportional to the volume of transferred data. Using a uniform methodology we present a method to find the distribution of the load so that the minimum completion time is achieved for the considered data distribution scheme. The results can also be used to find such parameters of the processor network as equivalent speed, speedup and utilization. Moreover, the methodology presented here can be a model of the application roll-in time, and can be applied in load balancing in a heterogeneous multiprocessor system.

Keywords: Distributed processing; Scheduling; Communication delays; Performance evaluation

1. Introduction

Parallel processing is the focus of research these days. As the classical computer systems approach their physical limitations in processing speed, the distributed computer systems are becoming the only known way to increase the speed of computations further. One of important problems related to this field is scheduling in parallel computer systems. Scheduling algorithms always require some assumptions on the computer system and the computation process. Hence, a good model of

☆ The research has been partially supported by the KBN grant and CRIT2 project.

*Corresponding author. Tel.: + 48 61 782-375; fax: + 48 61 771-525;
e-mail: blazewic@sol.put.poznan.pl

PII S0166-218X(96)00115-1

parallel computation is required. The models, on the other hand, are closely related to the estimations of the system performance. There exists a great variety of scheduling algorithms [12, 8, 9] based on different assumptions. A new branch related to the scheduling of parallel applications has appeared recently in the scheduling theory. It is called *multiprocessor* task scheduling (or *parallel* task scheduling or *coscheduling*) [6, 15, 19, 16, 14, 7]. It is assumed that a task may require more than one processor at a time. Also, in this work we assume that a task may be executed by more than one processor at a time. The performance models of computer systems are often based on statistic approaches. A disadvantage of the stochastic analysis is that certain probability distribution parameters, sometimes without experimental justification, are assumed arbitrarily. In this work we adapt a strictly deterministic approach.

Contemporary supercomputers are very often multicomputers, i.e. a set of processing elements (PEs) connected by a high-speed network e.g. CSCC Paragon, CM-5, CRAY-T3D and many others. A cluster of workstations or computers in a wide area network can be harnessed to work in the above way. In such distributed systems communication delays cannot be neglected. In this work we assume that a task not only can be executed by more than one processor at a time, but can also be arbitrarily divided between the cooperating processors. This model is useful both in practical and theoretical considerations. In the computations on large data files like searching for a pattern, fast Fourier transformation, filtering, etc., the volume of data can be divided into parts of different sizes analyzed separately. It is also the case of modelling behavior of a large number of particles because particles interact mainly locally and the whole set of particles can be divided into areas considered separately. Similarly, parallel implementations of metaheuristics (tabu search, genetic search) based on a master–slave model of computations can be analyzed in this way because analysis of potential new solutions can be done independently in parallel. Furthermore, some problems of linear algebra can be analyzed in this way [3]. What is more, it can be a model of an application roll-in in a multicomputer system, i.e. of the "unstable state" due to the application startup: unfolding of the code or distributing data on PEs. This is an important element of computer system efficiency as I/O operations become a bottleneck in contemporary supercomputers. The model is also interesting due to its simplicity and the results obtained.

Before going into details we will present roughly the model of the computational process (more detailed discussion of implementation issues takes place at the end of Section 2.1). At the beginning of the computation the data to be processed are stored in one processing element (PE). We will call this first PE *originator*. The originator intercepts some part α_1 of all the data for local processing and transmits the rest to its neighbors. Then, each neighbor i intercepts for local processing some part α_i of the whole data volume from the part it receives, and then retransmits the rest to its still idle neighbors. The process is repeated until the last processor N. The interceptions are instantaneous, i.e. the time for interception is negligible. As it can be observed, it is a model of store-and-forward communication. We assume that the processes of computation and communication are independent and can take place in the same

moment of time. It is justified for PEs with independent communication channels (e.g. transputer family of processors, or computers with Direct Memory Access). The whole computational task can be executed on one standard processor in time T_{cp}. When processor i has different than standard processing speed, the processing time of the whole task would last $w_i T_{cp}$, where w_i is a reciprocal of the processor's speed. The communication channel j is described by two parameters [17] the startup time r_j and the transfer rate $(1/z_j)$. Thus, the time needed to transfer b bytes is equal to $r_j + z_j b$. We assume that the total amount of the data that can be transferred is equal to T_{cm}. Hence, the whole data would be transferred through the link with the transfer rate equal to 1 and $r = 0$ in time T_{cm}. The main objective of the further analysis is to distribute the computation in such a way that it is completed in the minimal time.

The ideas are presented in this work stem from the series of works [10, 11, 1, 2]. In [10] the problem of optimal distributing computation on a chain of processors has been addressed. The processing elements could have communication co-processors or not, and their speeds were assumed to be different. In [11] this model has been applied to solve the problem of a job distributing on a tree network of processors, and in [1] – on the processors interconnected through a bus type medium. Finally, in [2] the performance limits are given for infinite chain and tree networks of processors. The same methodology has been applied to analyze two-dimensional mesh of processors [4] and hypercube of processors [5]. In all the above works it was assumed that communication startup time is zero. In this work we differ in a more sophisticated model of communication cost, a wider set of interconnection architectures is considered, and finally, more general treatment has been applied.

The rest of the paper is divided on the basis of analyzed interconnection architectures. In Section 2 various cases of the linear network (chain) architecture is analyzed. A tree of processors is considered in Section 3. In Section 4 a set of processors connected by shared buses is considered. The hypercube architecture is analyzed in Section 5.

2. Linear networks

2.1. Chain – originator at the end

In this section we will analyze a simple interconnection architecture – the chain of processors (Fig. 1). The methodology introduced here will be used for other interconnection types.

Assume that the originator is positioned at one of the chain's ends. For the time being, let us ignore the process of the solutions consolidation because, as it will be explained later, it can be incorporated into our model. The process of distributing the data and processing it is presented in Fig. 2. The originator intercepts α_1 of the whole data volume and sends $(1 - \alpha_1)$ to its neighbor. The second processor processes locally α_2 of the whole volume of data and retransmits $(1 - \alpha_1 - \alpha_2)$. Processor

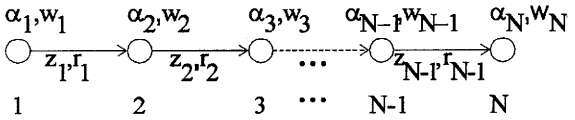

Fig. 1. Chain interconnection architecture.

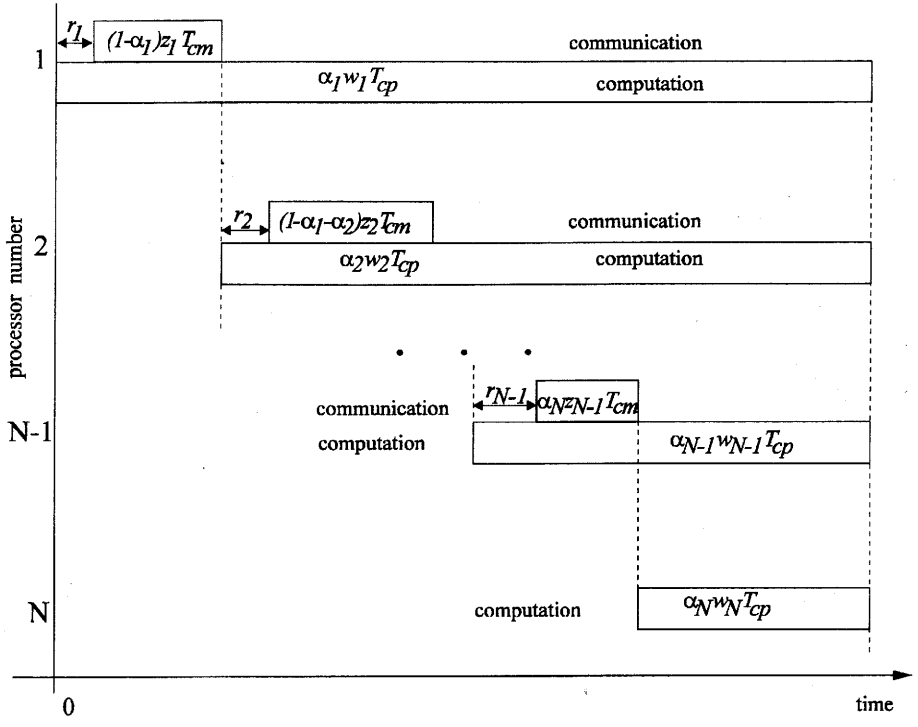

Fig. 2. Communication-computation diagram for a chain.

i intercepts α_i of the whole data volume. The interceptions are instantaneous. The process is continued until the last Nth processor. The communication from the originator to the second PE takes $r_1 + (1 - \alpha_1)z_1T_{cm}$ time, from the second PE to the third $r_2 + (1 - \alpha_1 - \alpha_2)z_2T_{cm}$, etc. The computations on the first processor are executed in $\alpha_1w_1T_{cp}$ time, on the second processor $\alpha_2w_2T_{cp}$, etc. The computations on all processors must finish at the same moment of time. This very basic observation can be explained as follows. When one of the PEs finishes earlier, one can find a better (shorter) schedule in which this PE is more loaded and computes some part of the data which was processed by the processors finishing later. This argument will be widely applied in this paper (for more formal proof see the appendix). Now, we are ready to compute the α_i's. Let us observe that for each of the communication links the communication time is equal to the difference of computing times on the transmitter

and the receiver. Hence, we can write the following set of equations and inequalities:

$$\begin{aligned}
&\alpha_1 w_1 T_{cp} = r_1 + (\alpha_2 + \alpha_3 + \cdots + \alpha_N) z_1 T_{cm} + \alpha_2 w_2 T_{cp},\\
&\alpha_2 w_2 T_{cp} = r_2 + (\alpha_3 + \alpha_4 + \cdots + \alpha_N) z_2 T_{cm} + \alpha_3 w_3 T_{cp},\\
&\vdots\\
&\alpha_{N-2} w_{N-2} T_{cp} = r_{N-2} + (\alpha_{N-1} + \alpha_N) z_{N-2} T_{cm} + \alpha_{N-1} w_{N-1} T_{cp},\\
&\alpha_{N-1} w_{N-1} T_{cp} = r_{N-1} + \alpha_N (z_{N-1} T_{cm} + w_N T_{cp}),\\
&\alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_N = 1\\
&\alpha_1, \ldots, \alpha_N \geqslant 0.
\end{aligned} \tag{1}$$

There are N equations and N unknowns in the above formulation. By solving it we obtain a set of coefficients $\alpha_I, \ldots, \alpha_N$ describing the division of the load among the PEs. Let us note that Eqs. (1) can be solved recursively in linear time (if the solution exists). It is possible to express α_{N-i}, for $i = 1, \ldots, N-1$, as a linear function of α_N. Hence, from the first $N-1$ equations of (1), we have

$$\alpha_{N-i} = l_{N-i} + \alpha_N k_{N-i} \quad \text{for } i = 1, \ldots, N-1,$$

where

$$l_{N-1} = \frac{r_{N-1}}{w_{N-1} T_{cp}}, \qquad k_{N-1} = \frac{z_{N-1} T_{cm} + w_N T_{cp}}{w_{N-1} T_{cp}},$$

$$l_{N-i} = \frac{r_{N-i}}{w_{N-i} T_{cp}} + \frac{z_{N-i} T_{cm}}{w_{N-i} T_{cp}} \sum_{j=N-i+1}^{N} l_j + l_{N-i+1} \frac{w_{N-i+1}}{w_{N-i}} \quad \text{for } i = 2, \ldots, N-1$$

$$k_{N-i} = \frac{z_{N-i} T_{cm}}{w_{N-i} T_{cp}} \sum_{j=N-i+1}^{N} k_j + k_{N-i+1} \frac{w_{N-i+1}}{w_{N-i}} \quad \text{for } i = 2, \ldots, N-1.$$

Of course, $k_N = 1, l_N = 0$. From the last equation of (1) we obtain $\sum_{j=1}^{N} (k_j \alpha_N + l_j) = 1$, and

$$\alpha_N = \frac{1 - \sum_{j=1}^{N} l_j}{\sum_{j=1}^{N} k_j}.$$

The rest of the load distribution coefficients can be found recursively from the above equations.

The set of equations (1) may not always be solvable. It is the case when the number of processors one wants to use is too big and it is impossible to transfer data to the last processor(s) during the time of computation on the first processor. In such a case α_N calculated recursively becomes negative. Thus, instead of formulation (1) it would

be more precise to solve for $\alpha_1, \ldots, \alpha_N$, implicitly stated formulation:

$$\alpha_i w_i T_{cp} \begin{cases} = r_i + z_i T_{cm} \sum_{j=i+1}^{N} \alpha_j + \alpha_{i+1} w_{i+1} T_{cp} \\ \quad \text{when } \alpha_{i+1} \geqslant 0 \quad (\text{for } i = 1, \ldots, N-1) \\ \geqslant 0 \\ \quad \text{when } \alpha_{i+1} = \cdots = \alpha_N = 0 \end{cases} \tag{2}$$

$$\alpha_i = 0 \Rightarrow \alpha_{i+1} = \cdots = \alpha_N = 0 \quad \text{for } i = 1, \ldots, N-1$$

$$\alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_N = 1, \qquad \alpha_1, \ldots, \alpha_N \geqslant 0.$$

A set of processors that can take part in minimal-time execution of a task will be called *usable*. Now the key question is how to determine the usable set of processors. When (1) is not solvable we propose to apply an iterative method as follows. Set $N = 1$. Solve (1) for the current value of N. If a feasible solution exists, check if the following inequality holds for the current value of N:

$$\sum_{i=1}^{N-1} \left(r_i + z_i T_{cm} \sum_{j=i+1}^{N} \alpha_j \right) + r_N < \alpha_1 w_1 T_{cp}. \tag{3}$$

In the above inequality we check whether it is possible to add to the current sequence of communications at least a startup time for the communication to processor $N + 1$. If it is not possible ((3) does not hold) N is the maximal number of processors that can be used for the given parameters. If yes (i.e. (3) holds), then set $N = N + 1$ and repeat this procedure. Keep on increasing N until (1) is not solvable, or the maximum number of available processors is reached. We can further improve this procedure by applying a binary search. Hence, in $O(\log N)$ trials and $O(N \log N)$ time we can identify the maximal usable set of processors.

We are now able to derive several parameters describing the whole computer network. The equivalent speed of the whole network is the ratio of the execution time on a standard processor to the execution time on the chain. It is equal to (cf. Fig. 2)

$$ES_N = \frac{T_{cp}}{\alpha_1 w_1 T_{cp}} = \frac{1}{\alpha_1 w_1}.$$

The speedup and utilization of the processors achieved in the network are, respectively, equal to

$$S_N = \frac{w_1 T_{cp}}{\alpha_1 w_1 T_{cp}} = \frac{1}{\alpha_1},$$

$$U_N = \frac{S_N}{N} = \frac{1}{\alpha_1 N}.$$

Let us present a simple example.

Example 1. Consider a chain of five processors: $w_1 = w_2 = w_3 = 1$, $w_4 = w_5 = 0.5$, $z_1 = z_2 = 1, z_3 = z_4 = 0.5, r_1 = r_2 = 0.2, r_3 = r_4 = 0.1$. The parameters of the task are $T_{cp} = 2$, $T_{cm} = 1$. This means that the fourth and the fifth processors are twice as fast as the first three processors. The communication links to the second and the third processor have only half of the speed of the remaining two links. The solution of the equation set (1) for $N = 5$ does not exist. For $N = 4$, the solution is: $\alpha_1 \approx 0.5794$, $\alpha_2 \approx 0.2692$, $\alpha_3 \approx 0.0934$, $\alpha_4 \approx 0.0579$, and since the last processor does not take part in computations $\alpha_5 = 0$. The whole computation is executed in 1.159 units of time. The equivalent speed of the whole network is $ES_4 = ES_5 \approx 1.73$ (compare with $1/w_1 = 1$), speedup is $S_4 = S_5 \approx 1.73$ and utilization $U_4 \approx 0.431$. Fig. 3 presents the momentary utilization of the five processor chain during the computation.

In the following paragraph we will illustrate that the above model can be used to analyze the performance of the computer system. Namely, we show the influence of the model parameters on the execution time, speedup and utilization in a homogeneous network. If not stated otherwise, parameters for the network were $w_i = 1$, $z_i = 1$, $r_1 = 0.0001$ for $i = 1, \ldots, N$, $T_{cp} = T_{cm} = 1$. In Fig. 4 the execution time of the whole computational task versus the number of PEs, r and z is depicted. The upper set of curves is for $z_i = 1$, and the lower for $z_i = 0.1$. As it can be seen, small values of r do not influence significantly the execution time. On the other hand, if r is close to z ($z = r = 0.1$), then it does contribute to the execution time. What is more, in slow network (e.g. $z = 1$), r influences mainly the number of processors that can be efficiently used. In Fig. 5 the number of usable processors is depicted as a function of r.

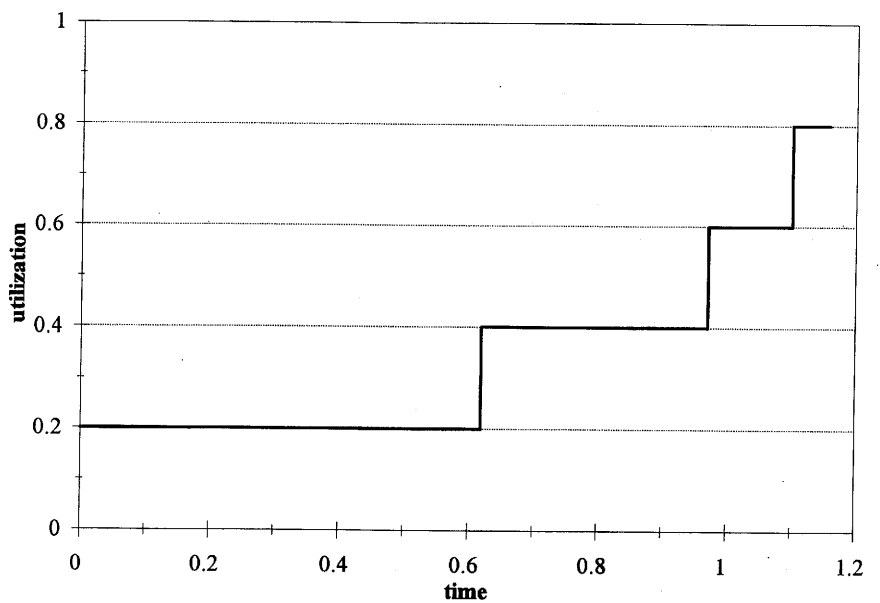

Fig. 3. Example 1: momentary utilization of the processor set.

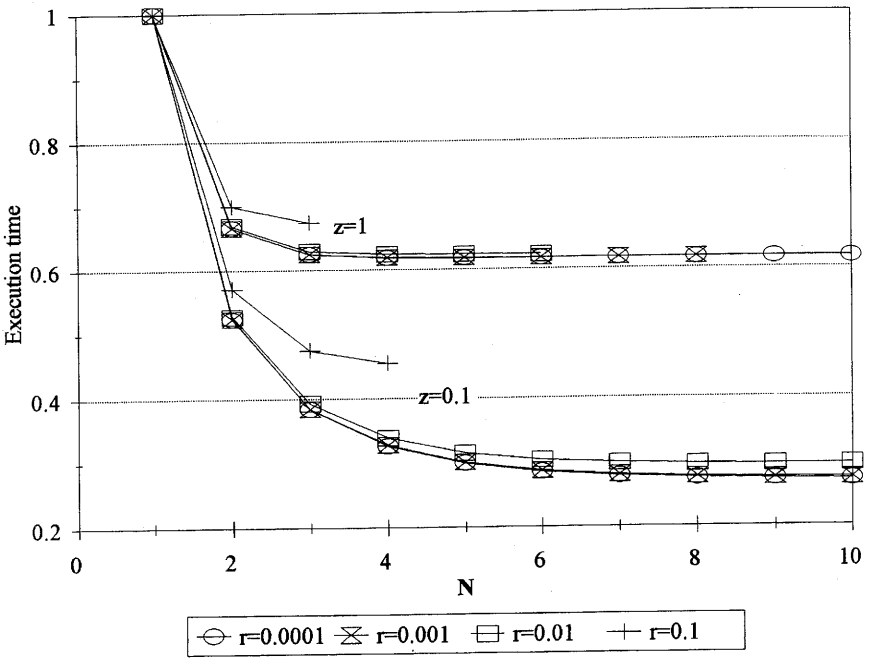

Fig. 4. Chain architecture: execution time vs. N, r, z.

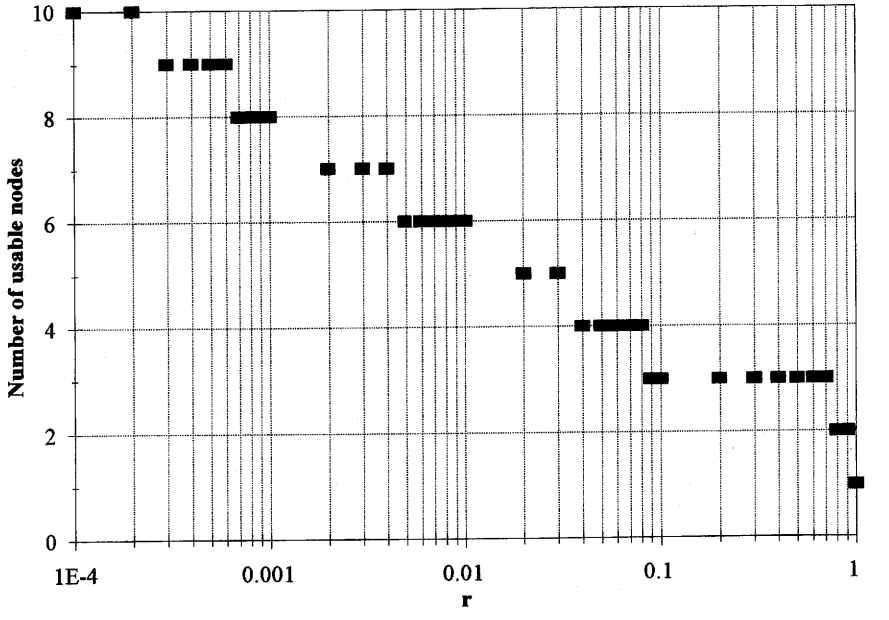

Fig. 5. Chain architecture: number of usable processors vs r.

In Fig. 6 we have presented the speedup achieved in the network versus N, r and z. The upper set of curves represents $z_i = 0.1$, and the lower for $z_i = 1$. The same conclusions as for Fig. 4 can be derived from Fig. 6. z is a parameter which has a significant influence on the speedup; it is illustrated for a wider set of z values in Fig. 7.

Now, we are going to comment on including the results consolidation process in our model. To our knowledge there are two main forms of returning the results. Either each of the PEs returns a message of a constant length, like saying "yes", "found", returning an identifier of some object, or (otherwise) the PE returns some amount of data proportional to the data volume it received for processing, e.g. after sorting, filtering, etc. The first case can be handled in this way: the constant time of the return communication can be included in r parameter, while the second case can be handled by adding the return communication time coefficient to the z parameters describing distribution of unprocessed data. Thus, in the second case, the total communication cost per unit of data in link i would be $(1 + \beta)z_i$, where β is a fraction of the received data returned in form of the results. Furthermore, if merging of the results on a higher level PE is necessary we can include it in a similar way by adding, on the right-hand side of the first $N - 2$ equations in (1), the merging time as a function of the load. Let us note that it would not be difficult to incorporate in the proposed model a startup time related not only to using some communication link, but also a startup time related to starting computation on a PE. In such a case formulation (1) must be

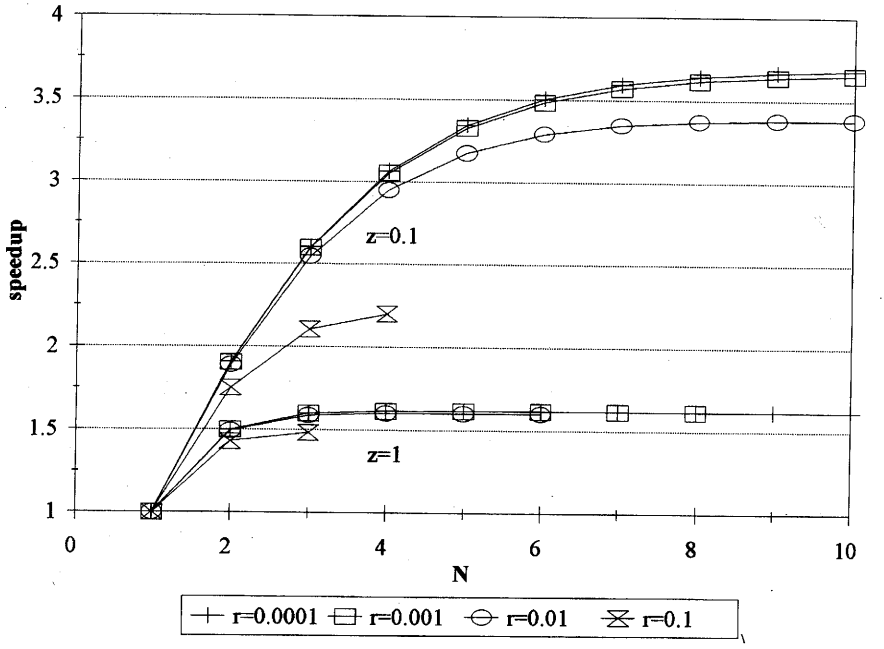

Fig. 6. Chain architecture: speedup vs. N, r and z.

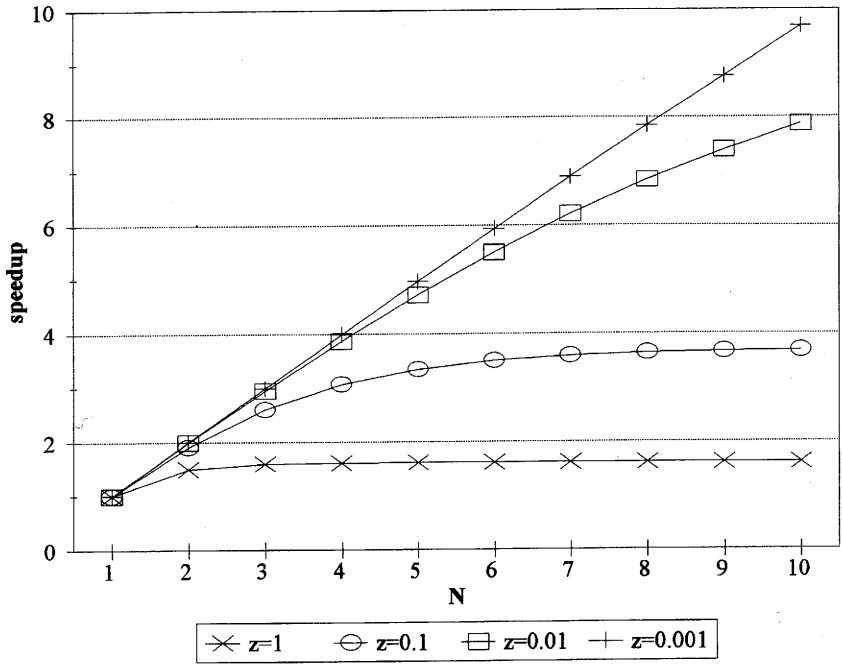

Fig. 7. Chain architecture: Speedup vs. z.

appropriately adjusted. Computation startup and the computation itself on the transmitter takes place while transmitting to, starting computation on the receiver and computing on the receiver. Hence, we should add computing startup time of the transmitter on the left-hand side and the computation startup time of the receiver on the right-hand side of the first $N-1$ equations in formulation (1). The case when the interception of the load is not instantaneous can be dealt analogously.

Finally, let us remark on the way a PE can split the received load into parts for local processing and for its neighbors. The distribution of the load can be calculated by the originator or, when the volume of data is fixed, by the compiler. The distribution information i.e. how to split the received volumes can be transmitted together with the data volume. The PE strips the information directed to itself, intercepts the required amount of the load and sends everything else further on. Hence, the division can be done in constant time. To include the above in formulation (1) it is necessary to increase the right-hand sides of the first $N-1$ equations by the time spent on transmission of the additional distribution information.

2.2. *Originator inside chain*

In this case the originator is not positioned at one of the chain's ends, but it is some PE number j. The communication–computation diagram for this interconnection

type is analogous to the one in Fig. 2. The set of equations (1) can be reformulated as follows:

$$
\begin{aligned}
&\alpha_j w_j T_{cp} = r_j + (\alpha_{j+1} + \alpha_{j+2} + \cdots + \alpha_N) z_j T_{cm} + \alpha_{j+1} w_{j+1} T_{cp},\\
&\alpha_{j+1} w_{j+1} T_{cp} = r_{j+1} + (\alpha_{j+2} + \alpha_{j+3} + \cdots + \alpha_N) z_{j+1} T_{cm} + \alpha_{j+2} w_{j+2} T_{cp},\\
&\vdots\\
&\alpha_{N-1} w_{N-1} T_{cp} = r_{N-1} + \alpha_N (z_{N-1} T_{cm} + w_N T_{cp}),\\
&\alpha_j w_j T_{cp} = r_{j-1} + (\alpha_1 + \alpha_2 + \cdots + \alpha_{j-1}) z_{j-1} T_{cm} + \alpha_{j-1} w_{j-1} T_{cp},\\
&\alpha_{j-1} w_{j-1} T_{cp} = r_{j-2} + (\alpha_1 + \alpha_2 + \cdots + \alpha_{j-2}) z_{j-2} T_{cm} + \alpha_{j-2} w_{j-2} T_{cp},\\
&\vdots\\
&\alpha_2 w_2 T_{cp} = r_1 + \alpha_1 (z_1 T_{cm} + w_1 T_{cp}),\\
&\alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_N = 1, \qquad \alpha_1, \ldots, \alpha_N \geqslant 0.
\end{aligned}
\tag{4}
$$

It can be solved in a similar way as (1). Again, the above set of equations and inequalities is not always solvable. Let us consider an example.

Example 2. Consider a chain consisting of $N = 11$ identical processors. The originator is the central PE ($j = 6$). $T_{cp} = 1$, $T_{cm} = 1$, $w_i = 1$ for $i = 1, \ldots, 11$and $z_i = 0.1$, $r_i = 0.05$ for $i = 1, \ldots, 10$. Formulation (4) cannot be solved for all possible configurations of the processor numbers in the left and in the right branch of the chain. In Fig. 8 the area of usable PEs configurations is presented. The piecewise line separates the

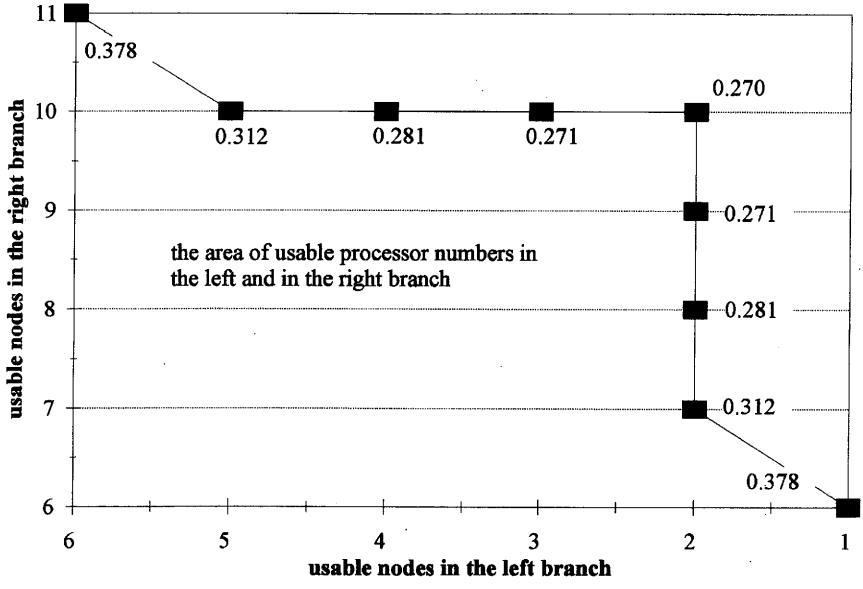

Fig. 8. Example 2: the area of usable configurations of processors in the chain branches and related execution times.

area of usable configurations from the area where (4) is not solvable. Each configurations on this line has attached a number describing the processing time for the task in a given configuration of processors in the left and in the right branch.

Obviously, the shortest execution times are achieved by configurations where the numbers of usable processors in branches of the chain are maximal. From Example 2, however, we conclude that the shape of the line separating unsable configurations (where (4) is solvable) from configurations where some processors are idle ((4) is not solvable) can be rather complex even for homogeneous network (cf. Fig. 8). Moreover, if communication links and PEs have different speeds then shape of the line, and the position of the best configuration, can be less predictable. Hence, when (4) is not solvable, we suggest to find the best distribution of the load (and PEs configuration) on the basis of try-and-error search along the line separating usable and not usable configurations. This can be implemented in $O(N)$ tries. The parameters like equivalent speed, speedup and utilization can be derived analogously to the case analyzed previously.

2.3. A loop

The loop interconnection can be dealt with analogously to the chain interconnection. The situation is slightly different because the last processor j (the one furthest from the originator which has number 1) can receive data from two links (cf. Fig. 9).

Assume that it is possible to communicate to processor j on two paths: using processors $1, 2, \ldots, j-1$ and $1, N, N-1, \ldots, j+1$. Let us denote by α'_j the amount of data received by PE j from PE number $j-1$, and by α''_j the amount of data received from PE number $j+1$. The set of equations (1) can be reformulated as follows:

$$
\begin{aligned}
&\alpha_1 w_1 T_{cp} = r_1 + (\alpha_2 + \alpha_3 + \cdots + \alpha_{j-1} + \alpha'_j) z_1 T_{cm} + \alpha_2 w_2 T_{cp},\\
&\alpha_2 w_2 T_{cp} = r_2 + (\alpha_3 + \alpha_4 + \cdots + \alpha_{j-1} + \alpha'_j) z_2 T_{cm} + \alpha_3 w_3 T_{cp},\\
&\vdots\\
&\alpha_{j-1} w_{j-1} T_{cp} = r_{j-1} + \alpha'_j z_{j-1} T_{cm} + (\alpha'_j + \alpha''_j) w_j T_{cp},\\
&\alpha_{j+1} w_{j+1} T_{cp} = r_j + \alpha''_j z_j T_{cm} + (\alpha'_j + \alpha''_j) w_j T_{cp},\\
&\alpha_{j+2} w_{j+2} T_{cp} = r_{j+1} + (\alpha_{j+1} + \alpha''_j) z_{j+1} T_{cm} + \alpha_{j+1} w_{j+1} T_{cp},\\
&\vdots\\
&\alpha_N w_N T_{cp} = r_{N-1} + (\alpha_{N-1} + \cdots + \alpha_{j+1} + \alpha''_j) z_{N-1} T_{cm} + \alpha_{N-1} w_{N-1} T_{cp},\\
&\alpha_1 w_1 T_{cp} = r_N + (\alpha_N + \alpha_{N-1} + \cdots + \alpha_{j+1} + \alpha''_j) z_N T_{cm} + \alpha_N w_N T_{cp},\\
&\alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_{j-1} + \alpha'_j + \alpha''_j + \alpha_{j+1} + \cdots + \alpha_N = 1,\\
&\alpha_1, \ldots, \alpha'_j, \alpha''_j, \ldots, \alpha_N \geqslant 0.
\end{aligned}
\tag{5}
$$

In the above formulation we have assumed that the communication to PE j from both directions must finish at the same time. Only after receiving its whole load can processor j start computations. Inequalities (5) can be solved in linear time similar

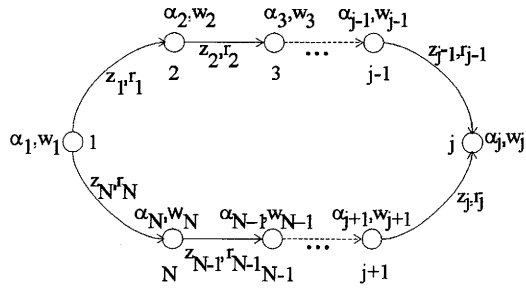

Fig. 9. Loop interconnection.

to the solution of (1) (one has to use the first $N-1$ equations to express α_i ($i=1, \ldots, j-1, j+1, \ldots, N$) as a linear function of α'_j and α''_j, then the first and Nth equation describing $\alpha_1 w_1 T_{cp}$ can be used to express α'_j as a function of α''_j, finally from the last equation we obtain α''_j). Analogous to the case of a chain network, formulation (5) may not always have a feasible solution. In this case it is not possible to communicate to the last j-th PE which results in "opening" of the loop. Such an "open" loop is equivalent to a chain with the originator inside the chain. Thus, we can find optimal distribution of the load solving (5) $O(N^2)$ times by analyzing at most N cases of different j locations, and if it results in "opening" of the loop then each of these cases can be dealt with $O(N)$ trials analogous to a chain with the originator inside the chain.

3. An arbitrary interconnection graph, a tree and a star

In this section we tackle the problem of modelling computations in an arbitrary interconnection graph. Unfortunately, this problem is NP-hard in general as the problem of finding the maximum flow in a network with a fixed charging (i.e. when some cost is imposed if flow on some arc is positive) and Steiner r-branching are NP-hard [18]. A similar NP-hard problem arises while scheduling file transfers [13]. Thus, finding the optimal schedule of computations and data transfers is computationally hard.

Suppose, we know which processors in the computer networks are to be used and by which communication links we have to communicate. We assume that all processors are accessed by one link, which means that communication in the computer network takes place over branches of a spanning tree (however, this constraint can be dealt with analogously to the case of a loop where one processor receives data from two directions). Without loss of generality, we assume that the originator is placed in the root of the tree while the other PEs are located in the internal nodes and levels of the tree. Moreover, for each node which distributes some volume of data we still assume that it is possible to transmit simultaneously in each link and compute on the node. The observation that all the processors must stop computation at the same moment of time is still valid. Hence, for each of the communication links the computation time of

a transmitting PE must be equal to the computation time of a receiver plus the transmission time. We can formulate a set of constraints on the division of the load among the PEs:

$$\alpha_i w_i T_{cp} = r_{ij} + z_{ij} T_{cm} \sum_{k \in S(j)} \alpha_k + \alpha_j w_j T_{cp} \quad \text{for } j \in O(i), \quad i = 1, \ldots, N$$

$$\alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_N = 1, \qquad \alpha_1, \ldots, \alpha_N \geqslant 0 \tag{6}$$

where $O(i)$ is a set of PE i immediate descendants, $S(j)$ is a set of all j descendants (not necessarily immediate), and r_{ij}, z_{ij} are parameters describing link (i,j) in which i is a transmitter and j is a receiver.

In the above formulation we have as many equations and unknowns as the number of the communication links plus 1. When formulation (6) cannot be solved it means that the given set of PEs and links is infeasible. As it was mentioned at the beginning of this section, finding the optimum is NP-hard in general.

At the end of this section, let us analyze a simple version of the tree – a star network. The star network is a very attractive high-level representation of the master – slave computation model. The communication link parameters can describe not only the physical layer properties but it can accommodate all the intermediate layers of the software and hardware between the master running in the center of the star and the slave running on a different PE in the arm of the star. Then, solving such a model can be useful to balance the load between heterogeneous slaves. The complexity of this problem remains open in general because it is not known whether finding the optimal set of usable processors and the order of transmissions to the slaves can be determined in polynomial time. However, special cases can be identified where polynomial-time solutions exist. Analysis of these cases is based on considering a *dual* problem to the one we have considered before. Such a dual problem is finding the maximum volume of data that can be computed for a given computation time. We assume that the master intercepts part α_1 of the load. The communication links are numbered as the slaves using them.

First, let us analyze the case of the star in which the master can transmit to all the slaves simultaneously. A part α_1 of the load is computed by the master in time T. Note that the load already resides at the master and in the case of the master T consists of computation only. The volume of data that can be computed in time T by PE i, different from the master can be calculated from the observation that T is equal to the communication time to that PE plus computation time for that PE. Therefore, we have

$$T = \begin{cases} \alpha_1 w_1 T_{cp} & \text{for } i = 1, \text{ i.e. for the master,} \\ r_i + \alpha_i (z_i T_{cm} + w_i T_{cp}) & \text{for } i = 2, \ldots, N. \end{cases} \tag{7}$$

Hence, from Eqs. (7) the capacity of the star, i.e. the amount of the data that can be computed as a function of the allowed computation time T is

$$capacity_{star}(T) = \frac{T}{w_1 T_{cp}} + \sum_{i=2}^{N} \max\left\{0, \frac{T - r_i}{w_i T_{cp} + z_i T_{cm}}\right\}.$$

The above function is piecewise linear and its slope changes at $T = r_i$ for $i = 2, \ldots, N$. Hence, for a given load (equal 1) one has to sort r_i values in $O(N \log N)$ time, and then it is possible to identify in $O(\log N)$ tries the proper range of T in relation to r_i's. The optimal processing time of the computational task can be found by linear interpolation, and α_i can be found from Eq. (7).

In the next two cases we assume that master can transmit to only one slave PE at a time. Consider two PEs in a star network: PE_i and PE_j described by parameters w_i, z_i, r_i, w_j, z_j, r_j, respectively. Assume that these two PEs receive load one after another (but not necessarily as the first and the second in the star). When PE_i receives its load first, then we will denote the computation time for the couple by T_1. Furthermore, $\alpha_i + \alpha_j = k$ because the two PEs must process some amount $k \in [0, 1]$ of data assigned to them. Then, the following equations describe the transmission time to both PEs.

$$T_1 = r_i + \alpha_i(z_i T_{cm} + w_i T_{cp}),$$

$$\alpha_i w_i T_{cp} = r_j + \alpha_j(z_j T_{cm} + w_j T_{cp}),$$

$$\alpha_i + \alpha_j = k.$$

From the above set of equations we get

$$T_1 = r_i + (z_i T_{cm} + w_i T_{cp}) \frac{k z_j T_{cm} + k w_j T_{cp} + r_j}{z_j T_{cm} + w_j T_{cp} + w_i T_{cp}}. \tag{8}$$

Analogously, for PE_j receiving its load first, we have processing time T_2 equal to

$$T_2 = r_j + (z_j T_{cm} + w_j T_{cp}) \frac{k z_i T_{cm} + k w_i T_{cp} + r_i}{z_i T_{cm} + w_i T_{cp} + w_j T_{cp}}. \tag{9}$$

From Eqs. (8) and (9) it can be inferred that in two cases one of the two processing times is smaller than the second one. Namely, for $r_i = r_j = r$ and $z_i = z_j = z$ we get

$$\begin{aligned} T_1 - T_2 &= z T_{cm}\left(\frac{k z T_{cm} + k w_j T_{cp} + r}{z T_{cm} + w_i T_{cp} + w_j T_{cp}} - \frac{k z T_{cm} + k w_i T_{cp} + r}{z T_{cm} + w_i T_{cp} + w_j T_{cp}}\right) \\ &\quad + T_{cp}\left(w_i \frac{k z T_{cm} + k w_j T_{cp} + r}{z T_{cm} + w_i T_{cp} + w_j T_{cp}} - w_j \frac{k z T_{cm} + k w_i T_{cp} + r}{z T_{cm} + w_i T_{cp} + w_j T_{cp}}\right) \\ &= \frac{k z T_{cm} T_{cp}(w_j - w_i) + (k z T_{cm} + r) T_{cp}(w_i - w_j)}{z T_{cm} + w_i T_{cp} + w_j T_{cp}} = \frac{r T_{cp}(w_i - w_j)}{z T_{cm} + w_i T_{cp} + w_j T_{cp}}. \end{aligned}$$

When communication links to both PEs are identical then communicating to the faster first produces shorter schedule than for the inverted order. Thus, in any schedule in which communication links are identical and slower PE receives its load

earlier inverting the order reduces processing time of the two and creates free time space for processing more load. Hence, we conclude that for the case of identical communication links faster PEs should receive their data earlier. This order of communications can be fixed in $O(N \log N)$ time. We assume now that PEs are labelled according to the above order. The optimal distribution of the load can be found from the observation that the computation time on PE i lasts as long as sending to PE $i+1$ and computing on PE $i+1$. Thus, we have

$$\alpha_i w_i T_{cp} = r_{i+1} + \alpha_{i+1}(z_{i+1} T_{cm} + w_{i+1} T_{cp}) \quad \text{for } i = 1, \ldots, N-1,$$

$$\alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_N = 1. \tag{10}$$

The above equation set can be solved in $O(N)$ time analogously to (1). Then, in $O(\log N)$ tries the maximum set of usable processors that can be identified (where a try is solving (10)) and then the optimal distribution of the load can be found.

For the case where $r_i = r_j = 0$, expressions for T_1 and T_2 become identical except for the denominators:

$$T_1 = k(z_i T_{cm} + w_i T_{cp}) \frac{z_j T_{cm} + w_j T_{cp}}{z_j T_{cm} + w_j T_{cp} + w_i T_{cp}},$$

$$T_2 = k(z_j T_{cm} + w_j T_{cp}) \frac{z_i T_{cm} + w_i T_{cp}}{z_i T_{cm} + w_i T_{cp} + w_j T_{cp}}.$$

when $z_j > z_i$ then PE_i has faster communication channel and should be served first. The rest of the argument is the same as in the case of $r_i = r_j = r$, $z_i = z_j = z$. Thus, we have identified three cases when the distribution of the load for the star architecture can be found in polynomial time.

4. Bus architecture

The bus interconnection gives to a designer of the computer system a great variety of possible ways of communicating. We can characterize roughly this flexibility by answering several questions: (1) how many buses are available? (2) which buses are available to which PEs? (3) which buses can be simultaneously listened to by a given PE? and (4) which PEs can simultaneously listen to a given bus? The flexibility can be even greater if we have a system of hierarchical buses. It can be shown that this problem is NP-hard in a strong sense.

Theorem 1. *Finding an optimal schedule of computations and communications in a bus interconnection system is* NP*-hard in general.*

Proof. The problem is obviously in NP. To show its NP-hardness we refer to 3-D MATCHING problem.

3-D MATCHING

Instance. Disjoint sets W, X, Z such that $|W| = |X| = |Z| = q$, set $M \subseteq W \times X \times Z$.

Question. Is there a three-dimensional match in M, i.e. set $M' \subseteq M$ such that $|M'| = q$ and each element of W, X, Z appears in M' exactly once?

Any instance of 3-D MATCHING can be transformed into the instance of decision version of our problem. Define $3q + 1$ PEs and $|M|$ buses. PEs with numbers $1, \ldots, q$ correspond to elements in set W, PEs with numbers $q + 1, \ldots, 2q$ correspond to elements in the set X, PEs numbered $2q + 1, \ldots, 3q$ correspond to the elements of set Z. PE number $3q + 1$ is an originator. Each bus corresponds to an element of set M. We assume that each PE can listen to only one bus at a time, while the originator can transmit on at most q buses simultaneously. No more than three PEs can simultaneously listen to the same bus. PEs which correspond to elements of W, X, Z, which are together in the same element $e \in M$ can simultaneously listen to a bus corresponding to e. Furthermore, PEs in range $1, \ldots, 3q$ compute with identical speed equal to 1. PE number $3q + 1$ has $w_{3q+1} = \infty$ and cannot compute. Communication on each bus lasts always a unit of time (i.e. $r = 1$ and $z = 0$). We ask if it is possible to compute the $T_{cp} = T_{cm} = 3q$ units of data in two units of time.

Suppose the answer to 3-D MATCHING is positive. Then, we use buses corresponding to M' to communicate to all $3q$ PEs in the first unit of time. Each PE computes its share of data in the second unit of time. Now, suppose the answer to our scheduling problem is positive. This means that all PEs are computing in interval $[1, 2]$. What is more in the first unit of time there were exactly q groups of three PEs receiving data, which is equivalent to positive answer to 3-D MATCHING. Otherwise, some PE does not receive its share of load in time. □

From the above theorem we conclude that finding optimal distribution of the load is computationally hard. However, we can still use our methodology provided that we know which PEs receive on a given bus. Assume that the number of buses is k. The parameters are $r_1, z_1, r_2, z_2, \ldots, r_k, z_k$ for buses $1, 2, \ldots, k$, respectively. On bus i the following PEs can receive: $i1, i2, \ldots, il_i$. PE number ij has reciprocal of speed equal to w_{ij} (cf. Fig. 10 for notation). Originator has reciprocal of speed w_1. It can transmit on all k buses simultaneously. PEs can listen to only one bus and only one PE on a given bus at a time. We will call such an organization of transfers *basic* organization.

Note that if we know which PEs use which buses we can establish the order of receiving data among the PEs on the same bus. It is always advantageous to communicate to a faster PE first (cf. Eqs. (8) and (9); PEs on bus i have the same r_i and z_i). In what follows, we assume that PEs on each bus are ordered according to their speed (the first is the fastest one). As for the previous interconnection types we can

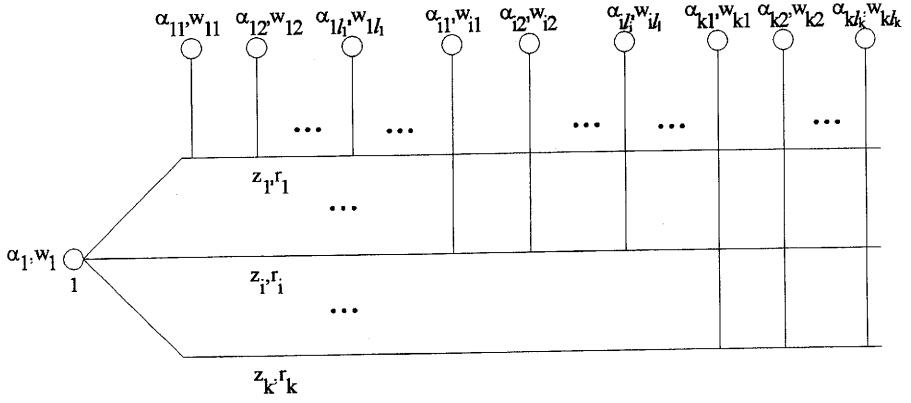

Fig. 10. The bus interconnection.

formulate a set of constraints:

$$
\begin{aligned}
&\alpha_1 w_1 T_{\mathrm{cp}} = r_1 + \alpha_{11} z_1 T_{\mathrm{cm}} + \alpha_{11} w_{11} T_{\mathrm{cp}},\\
&\alpha_1 w_1 T_{\mathrm{cp}} = 2r_1 + (\alpha_{11} + \alpha_{12}) z_1 T_{\mathrm{cm}} + \alpha_{12} w_{12} T_{\mathrm{cp}},\\
&\vdots\\
&\alpha_1 w_1 T_{\mathrm{cp}} = l_1 r_1 + z_1 T_{\mathrm{cm}} \sum_{i=1}^{l_1} \alpha_{1i} + \alpha_{1l_1} w_{1l_1} T_{\mathrm{cp}},\\
&\alpha_1 w_1 T_{\mathrm{cp}} = r_2 + \alpha_{21} z_2 T_{\mathrm{cm}} + \alpha_{21} w_{21} T_{\mathrm{cp}},\\
&\vdots\\
&\alpha_1 w_1 T_{\mathrm{cp}} = l_2 r_2 + z_2 T_{\mathrm{cm}} \sum_{i=1}^{l_2} \alpha_{2i} + \alpha_{2l_2} w_{2l_2} T_{\mathrm{cp}},\\
&\vdots\\
&\alpha_1 w_1 T_{\mathrm{cp}} = l_k r_k + z_k T_{\mathrm{cm}} \sum_{i=1}^{l_k} \alpha_{ki} + \alpha_{kl_k} w_{kl_k} T_{\mathrm{cp}},\\
&\alpha_1 + \sum_{i=1}^{k} \sum_{j=1}^{l_i} \alpha_{ij} = 1, \qquad \alpha_1, \alpha_{11}, \ldots, \alpha_{kl_k} \geqslant 0.
\end{aligned}
\tag{11}
$$

When the above formulation has no solution then the information about the feasible assignment of PEs to buses is invalid. Note that there are $O(k^N)$ different assignments of N PEs to k buses in a basic bus organization. Thus, the optimal solution can be found in polynomial time provided N is fixed. Finding the optimal solution is NP-hard in general (especially if we admit other bus organizations).

5. Hypercube

A hypercube interconnection of dimension d consists of 2^d PEs. Each PE in such a network can be uniquely labelled by a binary number in the range $[0, 2^d - 1]$.

A communication link exists between two PEs with labels differing on exactly one position. PEs and communication networks are homogeneous in hypercube multicomputers. Let us name by a *layer* the set of PEs accessed in the same number of hops (the path to the originator has the same number of intermediate processors). We assume that the communication–computation process in a hypercube proceeds as follows. The originator computes locally volume α_0 of data and sends to its d neighbors equal parts of the remaining volume. Neighbors of the originator form layer number 1. Each of the originator's neighbors transmits to its $d-1$ still idle neighbors (in layer 2) by $d-1$ links. Each of the PEs in layer 2 receives data by 2 links. We can generalize this and say that in layer i, PEs receive data from i links and retransmits the rest (not computed locally) in equal shares to $d-i$ still idle neighbors. Due to symmetric structure of interconnection and the homogeneity of its elements all PEs in a given layer receive the same amount of data to process. The number of PEs in layer i is equal to $\binom{d}{i}$. Hence, to find the distribution of the load among the PEs we can formulate the following set of constraints:

$$\sum_{i=0}^{d}\binom{d}{i}\alpha_i = 1,$$

$$\alpha_0 w T_{cp} = r + \frac{(1-\alpha_0) z T_{cm}}{d} + \alpha_1 w T_{cp},$$

$$\alpha_1 w T_{cp} = r + \frac{(1-\alpha_0 - d\alpha_1) z T_{cm}}{d(d-1)} + \alpha_2 w T_{cp},$$

$$\vdots$$

$$\alpha_i w T_{cp} = r + \frac{\left(1-\alpha_0 - \cdots - \binom{d}{i}\alpha_i\right) z T_{cm}}{\binom{d}{i} d - i} + \alpha_{i+1} w T_{cp}, \tag{12}$$

$$\vdots$$

$$\alpha_{d-1} w T_{cp} = r + \left(1-\alpha_0 - \cdots - \binom{d}{d-1}\alpha_{d-1}\right)\left(\frac{z T_{cm}}{d} + w T_{cp}\right)$$

$$= r + \alpha_d\left(\frac{z T_{cm}}{d} + w T_{cp}\right),$$

$$\alpha_1, \ldots, \alpha_d \geqslant 0.$$

There are d equations and d unknowns in (12). If it is unsolvable then not all PEs in a hypercube can be used for processing of a task because it can be finished in a shorter time than the communication time needed to reach the furthest processor. By use of

a binary search and dropping of further layers one can find the number of usable layers in $O(\log d)$ tries.

6. Conclusions

In this paper we have presented a simple method which has enabled us to find a distribution of a computational task among PEs such that the completion time is minimized. This method allows one to also describe the process of data distribution during the application execution initialization as well as gives some insight into the efficiency of a PE's network. Moreover, it can be applied to balance the load in a heterogeneous computer network. This method uses a simple linear formulation to give optimal solution. The main part of the linear formulation consists of equations describing for each communication link the following relation: the computation time of the transmitter is equal to the computation time of the receiver plus the communication time on the link.

Further research in this area may include, for example, considering other interconnection networks (e.g. meshes and tori), other communication paradigms (e.g. other broadcasting methods), as well as a deeper analysis of the computational complexity issues.

Appendix

Lemma A.1. *The computer network which has the greatest capacity for a given value of computation time has also the minimal computation time for a given capacity.*

Proof. Consider the function $capacity(T)$. If for some computer system i value of $capacity_i(T)$ is greater than for any other system for any T, then the reciprocal function $capacity_i^{-1}(load)$ is also always smaller than for any other system. □

Lemma A.2. *Computer system in which PEs during the whole computation time T either receive their load or compute on it, has maximal capacity, provided that communication and computation times are non-decreasing functions of the load.*

Proof. Transmission time from originator to PE number i is non-decreasing function of the transferred load. Computation time on this PE is also non-decreasing function of the load. Hence, also their sum and reciprocal function of this sum are non-decreasing functions. Thus, PE i receives and computes maximal load if T is maximal possible. This rules out idle times between transmission, computation, the beginning and the end of the schedule. □

Corollary A.3. *If there is no return of the results and no results consolidations all PEs finish computations at the same moment of time.*

References

[1] S. Bataineh and T.G. Robertazzi, Bus-oriented load sharing for network of sensor driven processors, IEEE Trans. Systems Man Cybernet. 21 (1991) 1202–1205.
[2] S. Bataineh and T.G. Robertazzi, Ultimate performance limits for networks of load sharing processors, CEAS Technical Report 623, State University of New York at Stony Brook (1992).
[3] J.-Y. Blanc and D. Trystram, Implementation of parallel numerical routines using broadcast communication schemes, in: H. Burkhart, ed., Proceedings of CONPAR 90-VAPP IV, Joint International Conference on Vector and Parallel Processing, Lecture Notes in Computer Science, Vol. 457 (Springer, Berlin, 1990) 467–478.
[4] J. Błażewicz and M. Drozdowski, Performance limits of two-dimensional network of load-sharing processors, Foundations Comput. Decision Sci., 21 (1996) 3–15.
[5] J. Błażewicz and M. Drozdowski, Scheduling divisible jobs on hypercubes, Parallel Comput. 21 (1995) 1945–1956.
[6] J. Błażewicz, M. Drabowski and J. Węglarz, Scheduling multiprocessor tasks to minimize schedule length, IEEE Trans. Comput. C-35 (1986) 389–393.
[7] J. Błażewicz, M. Drozdowski and J. Węglarz, Scheduling multiprocessor tasks – a survey, Internat. J. Microcomput. Appl. 13, (1994) 89–97.
[8] J. Błażewicz, K. Ecker, G. Schmidt and J. Węglarz, Scheduling in Computer and Manufacturing Systems (Springer, New York, 2nd Ed., 1994).
[9] T. Casavant and J. Kuhl, A taxonomy of scheduling in general-purpose distributed computing systems, IEEE Trans. Software Eng. 14 (1988) 141–154.
[10] Y.C. Cheng and T.G. Robertazzi, Distributed computation with communication delays, IEEE Trans. Aerospace Electronic Systems 24 (1988) 700–712.
[11] Y.C. Cheng and T.G. Robertazzi, Distributed computation for a tree network with communication delays, IEEE Trans. Aerospace Electronic Systems 26 (1990) 511–516.
[12] E.G. Coffman Jr., Computer and Job-Shop Scheduling Theory (Wiley, New York, 1976).
[13] E.G. Coffman Jr., M.R. Garey, D.S. Johnson and A.S. LaPaugh, Scheduling file transfers, SIAM J. Comput. 14 (1985) 744–780.
[14] M. Drozdowski, Scheduling multiprocessor tasks on hypercubes, Bull. Polish Academy Sci. Tech. Sci. 42 (1994) 437–445.
[15] J. Du and J.Y.-T. Leung, Complexity of scheduling parallel task systems, SIAM J. Discrete Math. 2 (1989) 473–487.
[16] D. Feitelson and L. Rudolph, Gang scheduling performance benefits for fine-grain synchronization, J. Parallel Distributed Comput. 16 (1992) 308–318.
[17] R. Hockney, Performance parameters and benchmarking of supercomputers, in: J.J. Dongarra and W. Gentzsch, Eds., Computer Benchmarks (Elsevier, Amsterdam, 1993) 41–64.
[18] G.L. Nemhauser and L.A. Wolsey, Integer and Combinatorial Optimization (Wiley, New York, 1988).
[19] B. Veltmann, B.J. Lageweg and J.K. Lenstra, Multiprocessor scheduling with communication delays, Parallel Comput. 16 (1990) 173–182.