THE PERFORMANCE LIMITS OF A TWO-DIMENSIONAL NETWORK OF LOAD-SHARING PROCESSORS

Jacek BŁAŻEWICZ[*‡], Maciej DROZDOWSKI[*]

Presented by J.Węglarz

Abstract. A process of computation in a two-dimensional mesh of processors is analysed in this work. A computational task is assumed to be arbitrarily divisible between processors. The load of the job is transferred from the originating processor to other processors by a point to point communication network. Communication delays are taken into consideration. The time costs of computation and communication are assumed to be linear function of the data size. Simple formulae are found to determine the distribution of the task's load and the equivalent speed of the whole network of processors. The performance of the two-dimensional mesh architecture is analysed. It turns out that in realistic situations the performance of this kind of the network is limited.

1.Introduction

Parallel processing is a subject which has been intensively studied for many years. Recently new contexts have been introduced by a rapid progress in technology. A number of parallel architectures has been successfully implemented [9,10,13]. These machines consist of processors with local memories, tied together by a point to point interconnection network. Such an architecture has been considered in many works [7,11,14]. Few researchers, however, while designing scheduling algorithms, took into consideration the fact that a task can be executed by more than one processor at a time [1, 2, 3, 4, 6, 8, 15]. In this work we allow any task not only to be processed by many processors at a time, but we permit it also to be divided arbitrarily among these processors. This is the case of distributed processing of large data files, signal or image

[*] Poznañ University of Technology,Institute of Computing Science,Poznañ, Poland. This work has been partially supported by KBN grant 3P40600106.

‡ Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Gliwice

processing, Kalman filtering, etc. [5]. In such applications and for such the architecture communication delays cannot be neglected.

This work is based on similar assumptions as made in a series of papers [1, 5, 6]. A wide range of interconnection architectures has been considered there: daisy chain of processors, a tree of processors or processors working over a common bus. We extend this model by analysing architecture of a regular two-dimensional mesh of processors (Fig.1), i.e. interconnection network in which each processor is located in corners of four rectangles and has four neighbours. We also use the methodology introduced in [1, 5, 6] to analyse the problem considered.

Let us set up the subject more precisely. Processors form an infinite mesh, i.e. the effects of its limited size will not be considered. The method of processing tasks can be described as follows. At time 0 some processor receives a burst of data (equivalent to one task) to be processed. This processor will be called *originator*. Some part α_0 of the data is processed in this processor while the rest $(1-\alpha_0)$ is sent to the four neighbouring processors. These processors take for local processing some part α_1 of all the data to be processed and retransmit the rest $(1-\alpha_0-\alpha_1)$ of the data to the neighbouring, but still idle, processors. The same process is repeated in the following layers. The computation on all the processors starts immediately after receiving the whole load from the preceding layer (cf. Fig.2). We assume that the network is not limited and is homogenous, that is, all processors and communication channels have the same speed. The task can be processed by a single processor in time equal to wT_{cp}, where w is proportional to the reciprocal of processor speed and T_{cp} is a processing time of the task on the processor with the speed equal to *1*. Transmission time can be computed as a product zT_{cm}, where z is inversely proportional to the speed of communication channel and T_{cm} is a time taken to transmit all the data between two neighbouring processors using link with speed *1*. Although the processors are identical and communication links are identical we will still use w and z (instead of a '1' for example) because explicit expression of the speeds will be useful later on. The aim of the scheduling process is to find the shortest schedule. In the following we are going to present performance analysis of the processor network working according to the above paradigm. The performance will be measured either as a processing time of a task or as an equivalent speed of the whole processor network. The main goal will be to establish the optimistic estimate for the performance of the two-dimensional mesh of processors. Let us note, that unlike in many other works [7,12], the results presented herein are based on a deterministic approach.

2. Performance limit analysis

In this section an optimistic bound for the performance of a regular two-dimensional network of processors will be established. The bound, as a result of the computation model, will be expressed by several parameters such as the equivalent speed of the whole network, the processing time of a standard task, speedup etc.

We start with the analysis of the minimum number of hops necessary to reach certain processor (while jumping from one processor to another). The process of disseminating the data is presented in Fig. 1. Let us call by a *layer* the set of processors that can be accessed in the same number of hops. We see that the number of processors in each layer is equal to the number of processors in the previous layer plus four. There is, however, an exception of the layer *0* consisting of only one processor - the processor from which the processing originates.

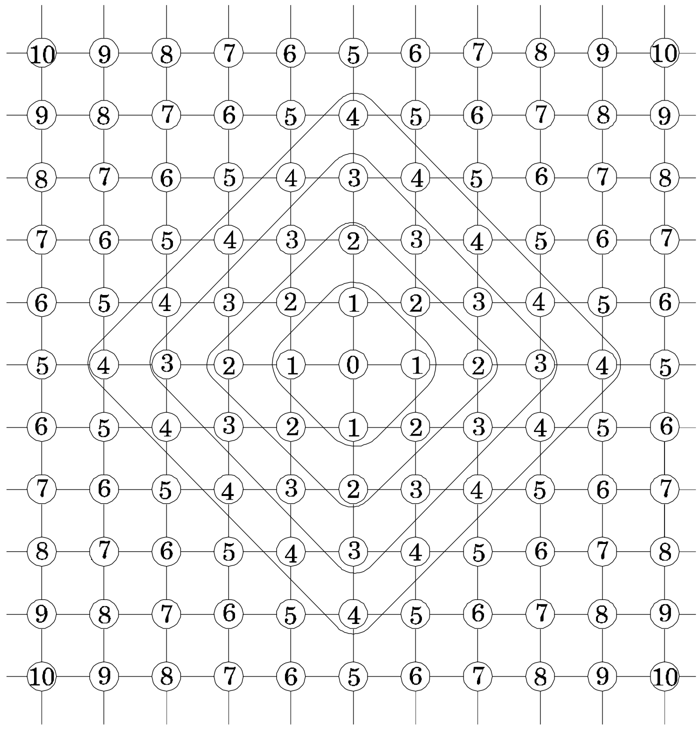

Fig. 1. The process of the data dissemination.

As it was described in Introduction, each layer receives data from the previous layer and immediately after this starts computations and transmission of a part of the data to the descending layers. This is possible for processors equipped with independent communication co-processors (e.g. transputers). Observe, that communication of the results back to the originator is an analogous, but reversed, process. We assume, that the amount of the data returned from the consecutive layers is proportional to the volume of data received for processing. Hence, it is possible to include both broadcasting of the data and the back-propagation of the results in one communication cost related to one transmission between two consecutive layers. For example, it can be done by multiplying all the communication times by constant $(1+\beta)$, where 1 stands for the broadcasting of the data and β represents the volume of the returned results as a fraction of the received data. In such a reduced case the time consumed on communication to some layer plus the time of computing in this layer is the same as in not reduced case where transferring data and results is separated in time. Thus, we can restrict our considerations only to the case where the data is only broadcasted, as it is depicted in Fig.2.

In order to achieve optimality of the schedule, all processors in all layers must stop computations simultaneously. This assumption can be supported by an intuitive argument that when processors are not finishing at the same moment, then some data can be moved from the processor working longer to the processor working shorter and therefore, the completion time can be reduced. Since the computation process is very regular the processing time of some part of the task is proportional to the volume of data constituting it. Thus, the processor working shorter receives all its load earlier and also finishes computations earlier. Hence, it is always feasible to reduce, by some amount, the load transferred to the processor working longer and send it to the processor working in a shorter time. The time diagram illustrating the process of computation and communication between the layers is presented in Fig. 2.

We relax the problem by assuming that in a given layer all the processors start computations at the same moment and that their loads are equal. This is not true in fact. One can easily realize that a layer is not symmetric and it is not possible to transmit the same amount of data in the same time to the processor on the "edge" (accessible via two channels) and to the processor "in the corner" of the layer (accessible via one link only). Thus, from this point of view, results presented further in this paper are optimistic bounds for the performance of the network.

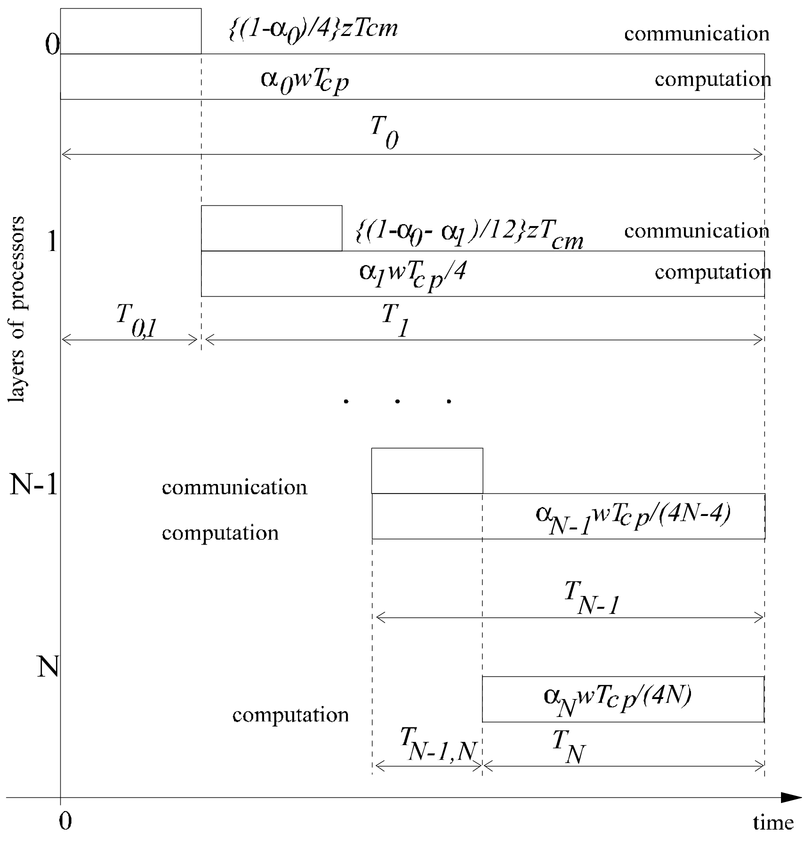

Fig. 2. Computation - communication diagram.

Now, we give the formulae describing the division of the load between layers. The whole layer of processors will be substituted for by one type of equivalent processor. Such equivalent processors will be identical both from the viewpoint of the processing and communication ability. Assuming that the number of layers is limited to *N* and the last layer processes all the data it receives, the part of the load taken by the previous (*N-1*)st layer is calculated. Then, recursively the share of the load for all the layers up to the first one (composed of four processors) can be computed. Finally, the speed of a processor equivalent to the whole network will be calculated.

Let $\hat{\alpha}_i$ denote a part of the data received from the layer i-1 which is processed locally by layer i, i.e.

$$\hat{\alpha}_i(1-\sum_{j=0}^{i-1}\alpha_j)=\alpha_i. \qquad (1)$$

The processing time $T_{N\text{-}1}$ of the processors in the layer N-1 is equal to the communication time from layer N-1 to N (denoted $T_{N\text{-}1,N}$) plus the processing time of processors in layer N (denoted T_N) (cf. Fig.2), i.e.:

$$T_{N-1}=\hat{\alpha}_{N-1}(\alpha_N+\alpha_{N-1})\frac{wT_{cp}}{4(N-1)} \qquad (2)$$

$$T_{N-1,N}+T_N=(1-\hat{\alpha}_{N-1})(\alpha_N+\alpha_{N-1})\left(\frac{w_N^{eq}T_{cp}}{4N}+\frac{zT_{cm}}{8(N-1)+4}\right)$$

$$T_{N-1,N}+T_N=T_{N-1} \qquad (3)$$

where w_N^{eq} is a reciprocal of the equivalent speeds for one processor in the layer N. The notion of the equivalent speed (or its reciprocal) will be used to reduce recursively all the layers in the mesh to a one (the first) layer. For the case of the last and the last but one layers, w_N^{eq} is equal to w. Note, that 4(N-1) processors of layer N-1 are linked to 4N processors of layer N through 8(N-1)+4 links. From equation (3) we can derive that

$$\hat{\alpha}_{N-1}=\frac{1}{1+\dfrac{N(2N-1)wT_{cp}}{(N-1)(NzT_{cm}+(2N-1)w_N^{eq}T_{cp})}} \qquad (4)$$

The above value may not be greater than 1 because no layer can accept more than it receives. A reciprocal of the equivalent speed for the processors in layer N-1 representing both layers N and N-1 is equal to the ratio of the real processing time of processors in layer N-*1*, and the processing time of the same load by the same number of processors as in layer N-*1*, with a standard speed 1. Hence (from (2)),

$$w_{N-1}^{eq}=\frac{T_{N-1}}{\dfrac{(\alpha_N+\alpha_{N-1})}{4(N-1)}T_{cp}}=\frac{\hat{\alpha}_{N-1}(\alpha_N+\alpha_{N-1})wT_{cp}}{(\alpha_N+\alpha_{N-1})T_{cp}}=\hat{\alpha}_{N-1}w \qquad (5)$$

In this way we have reduced layers N-1 and N, to a one equivalent layer with the number of processors equal to the number of processors in layer N-1. Similarly, by a successive substitution of N with values N-2,...,2, in equations (4), (5), one can calculate equivalent speeds of processors in layers N-2 through 1. Now, we will calculate the speed of the whole network from the point of view of the processor at level 0 (originator). Its working time T_0 is equal to the working time T_1 of the first layer plus the communication time $T_{0,1}$ (cf. Fig.2):

$$T_0=\hat{a}_0w_0T_{cp}=a_0w_0T_{cp}$$

$$T_1+T_{0,1}=\frac{1-\hat{a}_0}{4}(zT_{cm}+w_1^{eq}T_{cp})$$

Hence, $\hat{\alpha}_0=\alpha_0$ is equal to

$$\hat{a}_0 = \frac{1}{1+\frac{4wT_{cp}}{w_1^{eq}T_{cp}+zT_{cm}}}$$

and analogously to (5) $w_0^{eq} = \hat{\alpha}_0 w$.

Thus, we have presented a recursive method of finding $\hat{\alpha}_i$. In order to find α_i one has to apply equation (1) for i=1,...N-1. As a result, we have calculated the distribution of the load and equivalent speed of the network from the originator's point of view.

From the model presented above we can derive also such parameters of the network as speedup (S) and average utilization of processors (U) when layers 0 through j take part in computation:

$$S = \frac{wT_{cp}}{\hat{a}_0 wT_{cp}} = 1 + \frac{4wT_{cp}}{w_1^{eq}T_{cp}+zT_{cm}}$$

$$U = \frac{S}{\sum_{i=1}^{j} 4i+1}$$

Now, we are going to comment on the case of infinite mesh of processors. Assuming that the number of layers is infinite, $\hat{\alpha}_{N-1}$ tends to a constant value $\hat{\alpha}_\infty$:

$$\hat{\alpha}_\infty = \lim_{N\to\infty} \hat{\alpha}_{N-1} = \frac{1}{1+\frac{wT_{cp}}{\frac{1}{2}zT_{cm}+w_N^{eq}T_{cp}}}$$

An equivalent reciprocal of the processors' speed in all distant layers tends to $w_\infty^{eq} = \hat{\alpha}_\infty w$ (analogously to (5)). On the other hand, w_N^{eq} in equation describing $\hat{\alpha}_\infty$ is also equal to w_∞^{eq} because layers N and N-1 are identical (N=∞=N-1). Thus,

$$\hat{\alpha}_\infty = \frac{1}{1+\frac{wT_{cp}}{\frac{1}{2}zT_{cm}+\hat{\alpha}_\infty wT_{cp}}}.$$

Solving the above expression for $\hat{\alpha}_\infty$, we obtain the equation:

$$\hat{\alpha}_\infty = \frac{-\frac{1}{2}\rho+\sqrt{(\frac{1}{2}\rho)^2+2\rho}}{2} \qquad (6)$$

where $\rho = \frac{zT_{cm}}{wT_{cp}}$. We see from the above equation that the part of the load intercepted by a layer of processors in a big mesh stabilizes around some fixed value which depends on the amount of computing and communication necessary for a computational task. One can verify, that the value of $\hat{\alpha}_\infty$, in the above equation is never greater or equal to 1. This means that in the infinite mesh no layer intercepts all the load and that the load is processed by all layers.

In the next Section we will present results of modeling the above equations for various values of parameters describing the mesh.

3. Performance modeling

In this section we present results obtained from the model for different values of its parameters. Unless stated otherwise, $z=1$, $w=1$, $T_{cm}=T_{cp}=1$.

In Fig.3 through Fig. 5 the execution time as a function of the number of layers processing a task (or hops in the network, in other words) is depicted. In Fig. 3 one can find a diagram illustrating execution time changes according to changes of z - the reciprocal of the communication channels speed. There are four curves: for z equal to *10, 1, 0.1, 0* (slow, medium, high speed, ideal channels - upper, medium, lower, the lowest curve, respectively). The value $z=0$ can be achieved only in an ideal (unrealistic) network introducing no communication delay. It is presented in the figure as a reference. We see that the curves very quickly level off and three or four layers are almost as effective as twenty or more. What is more Fig. 3 supports the intuitive opinion that the faster the medium is, the shorter the execution time is and the greater the benefit from distributed processing. It can be observed in Fig.3 that for $z\neq0$ execution time stabilizes around some fixed value which is not the case of an ideal network ($z=0$). This is also illustrated in Fig.8 further in this section. Hence, further improving of the network speed may have great impact on performance of the mesh.

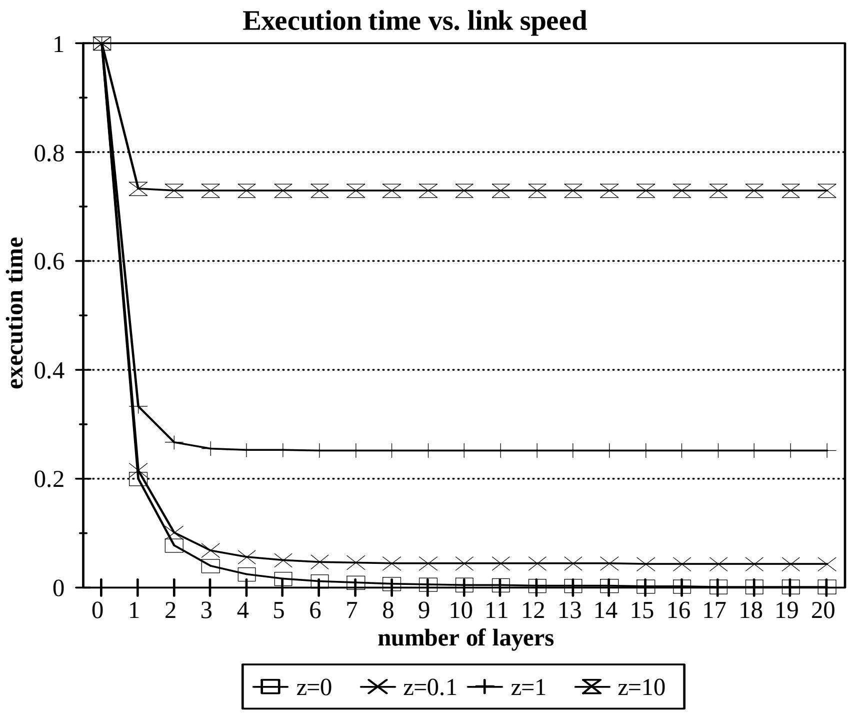

Fig. 3. Execution time vs. the number of layers for different speeds of the communication links.

Fig. 4 illustrates changes of execution time imposed by changes of processor speeds. Three curves illustrate execution time for *w=10, 1, 0.1* respectively (slow, medium speed, fast processors -upper, medium, lower curve, respectively). Of course, the faster the processors are, the faster the whole task can be completed. However, for fast processors (*w=0.1*) reduction of the execution time with the growing number of layers is minimal. For example, adding first layer to the originator reduces execution time by 67% in the case of *w=1*, but only by 26% in the case of *w=0.1*. Thus, meshes with more than 5 layers (for the chosen parameters) seem unjustified.

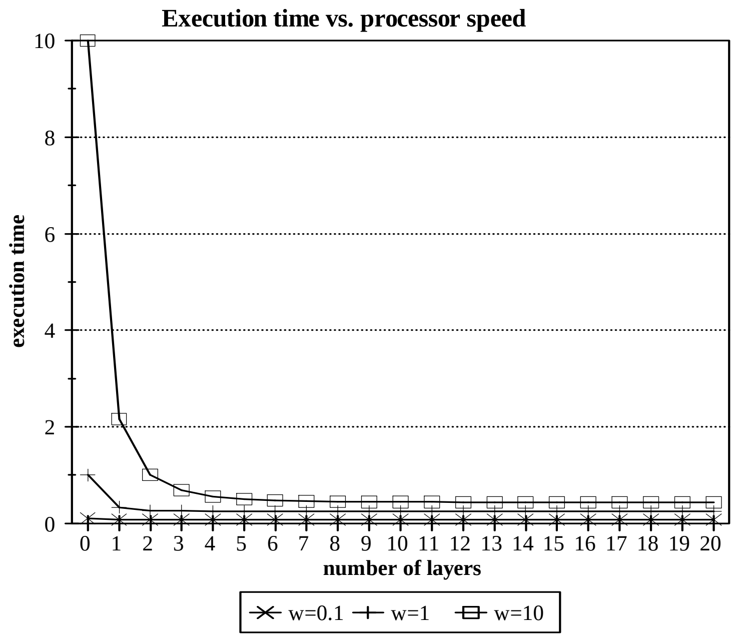

Fig. 4. Execution time vs. the number of layers for different processor speeds.

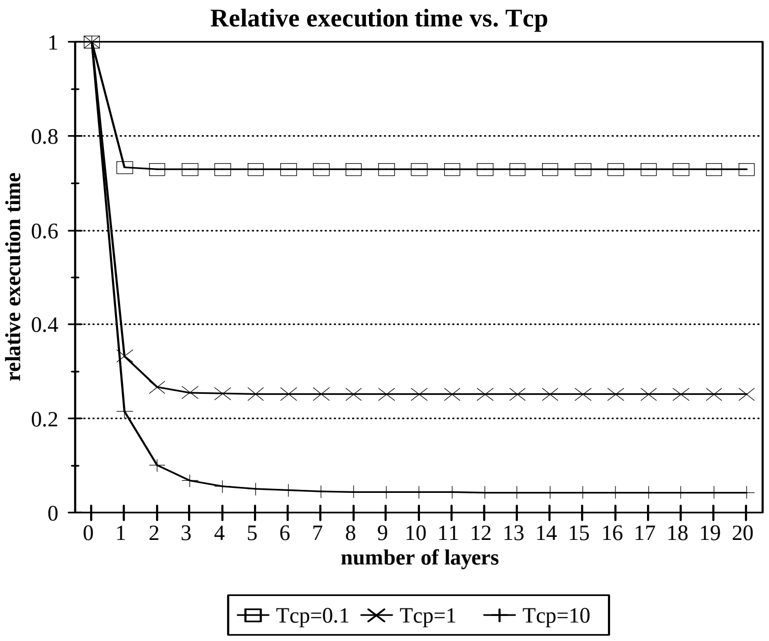

Fig. 5. Relative execution time vs. the number of layers for different task lengths.

Fig. 5 shows the relative execution time of the task as a function of the parameter T_{cp} (i.e. execution time of the task on a single standard speed processor). Relative execution time is the ratio of the real execution time to T_{cp}. Three curves depict relative execution time for T_{cp} *=0.1, 1, 10*, respectively (upper, medium, lower curves, respectively). It can be found that the longer the task is, the greater advantage can be taken from the distributed processing.

In the next chart (Fig. 6) we demonstrate the decreasing contribution of consecutive layers in processing the task for different values of *w*. The curves present distribution of the load of the first ten layers in the network with total twenty layers of processors. As it can be seen the slower the processors are (the bigger *w*) the bigger is the part of the load processed by deeper layers. This means that the maximum of the load moves to the deeper layers with the decrease of the processor speed. The bulk of the data, however, is processed in some intermediate layers. A curve for *z=0* (perfect network) has been added in Fig.6 for reference. In the perfect network the load in a layer is proportional to the number of the layer's processors. Dependence of the load distribution on *z* parameter has a very similar form.

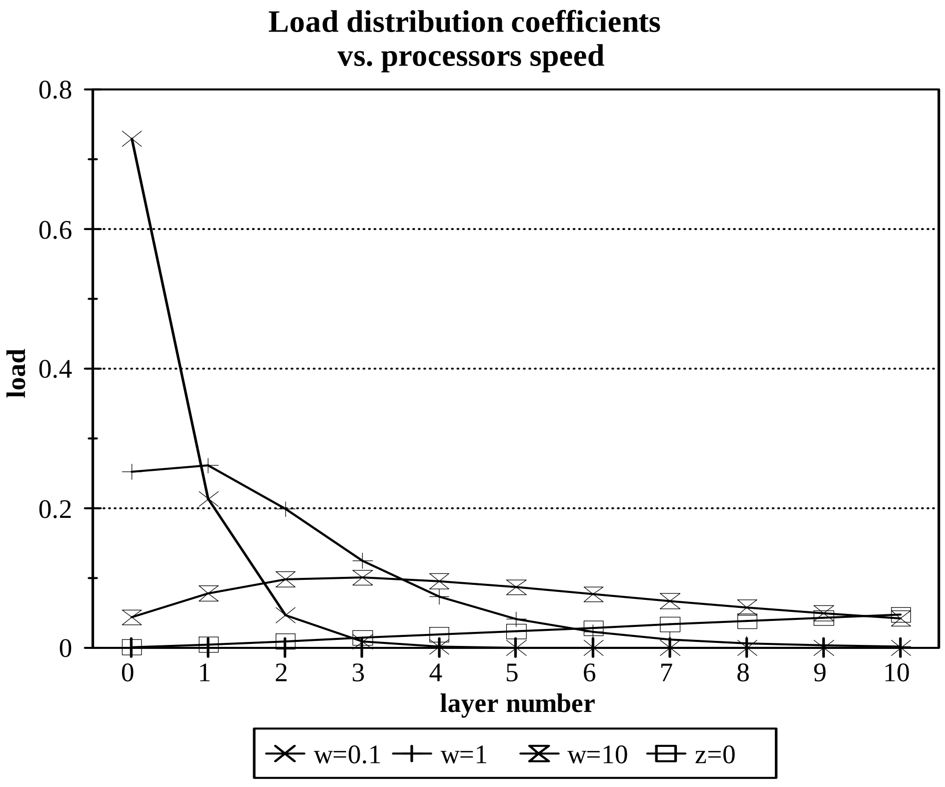

Fig. 6. Values of α_i for different processor speeds.

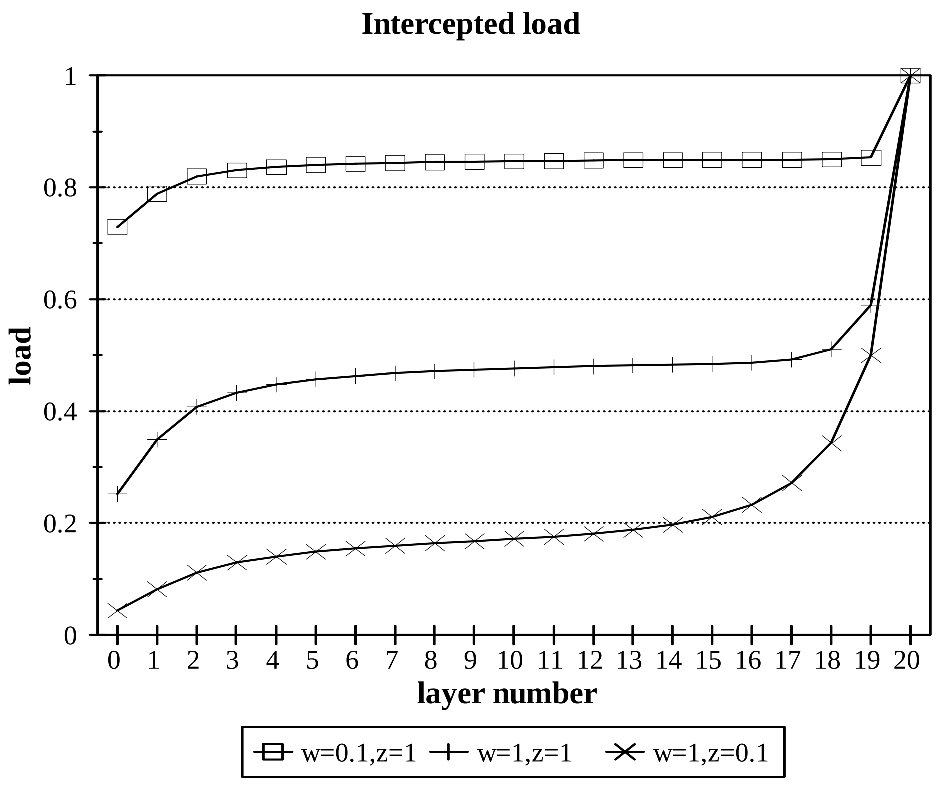

Fig.7. Values of $\hat{\alpha}_i$ - the part of the load intercepted by a layer, for different values of *z, w*.

In Fig. 7 we have presented values $\hat{\alpha}_i$, for a task processed by a mesh consisting of twenty layers, for three example values of *w* and *z*. It can be found that $\hat{\alpha}_i$ stabilizes around some constant value in the intermediate layers. According to theoretical expectations (equation (6)) $\hat{\alpha}_\infty$ takes values 0.854, 0.5, 0.2 for (*w=0.1,z=1*), (*w=1,z=1*), (*w=1,z=0.1*), respectively. Thus, we can say that $\hat{\alpha}_\infty$ is quite well approximated by equation (6).

Finally, in Fig.8 and Fig.9 we present the speedup achieved in the network and the utilization of the processors. As it can be seen, speedup very quickly levels off and at about 40 processors (layer number 4) stops its increase. Only when the system has a perfect communication network (curve *z=0*), can the linear speedup be achieved. The dependence of the speedup on *z* has a very similar form.

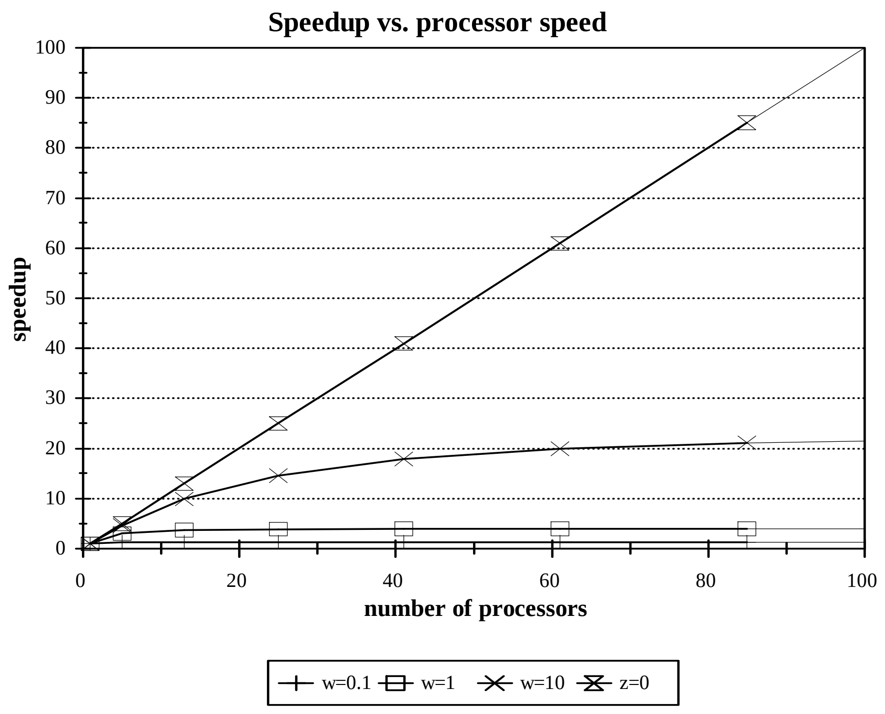

Fig. 8. Speedup for different speeds of processors

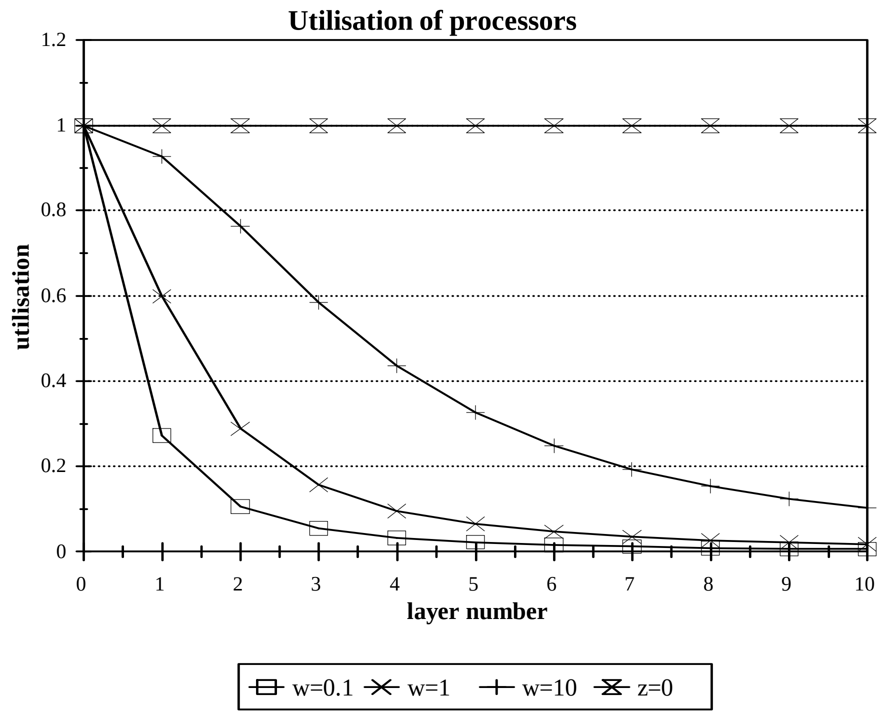

Fig.9. Utilization of processors for different speeds of processors

According to the diagram in Fig. 2 processor in layer 0 is always working (i.e. has utilization 1), while some processors may be idle in some time interval. Fig. 9 illustrates average utilization of all the processors taking part in the process of computation. Only when the communication medium is perfect (*z*=0), can the utilization be equal to 1. In other (realistic) cases the utilization decreases with the speed of processors (i.e. for *w* decreasing) and the number of layers taking part in computation. This means that with the growing speed of processors communication delays are becoming more significant. Distant processors are mainly waiting for their share of the data, while the bulk of computation takes place in the vicinity of the originator.

4. Conclusions

In this work we have presented a simple analytical deterministic model of a distributed computation.

Taking into account the formulae and charts presented in the previous section one can conclude that small numbers of processors are sometimes as efficient as massively parallel systems, of the architecture considered. This leads to a conclusion that very big meshes of processors or massively parallel systems can be efficient only when the locality of the process of computations is exploited. It can be, for example, the case of systolic algorithms exploiting locality in each step of systolic algorithm while the data flows through the network. Intuitive expectations of the dependence of the efficiency of the architecture on the speeds of the processors and the communication medium are justified. The better these two parameters are, the more efficient the network can be. A new observation is that the longer the task is, the bigger are the gains from parallel processing. Another observation is that the speed of the network is a crucial parameter influencing the performance.

Future research in this area may be concerned with other architectures of the network as well as with more than one task present in the network at the same time. The method of finding the distribution of the load among processors of the mesh without allowing for the simplifications present in this work seems to be a challenging task, for future research.

References

1. Bataineh S., Robertazzi T.G.: Ultimate performance limits for networks of load sharing processors". In: *Proceedings of the 1992 Conference on Information Sciences and Systems,* Princeton, NJ, Mar. (1992), 794-799.

2. Błażewicz J., Dell'Olmo P., Drozdowski M., Speranza M.G.: Scheduling multiprocessor tasks on three dedicated processors. *Information Processing Letters* **41** (1992) 275-280.

3. Błażewicz J., Drabowski M., Wêglarz J.: Scheduling multiprocessor tasks to minimize schedule length. *IEEE Transactions on Computers* **C-35** (1986) 389-393.

4. Chen G.I, Lai T.H.: Preemptive scheduling of independent jobs on a hypercube. *Information Processing Letters* **28** (1988) 201-206.

5. Cheng Y.C., Robertazzi T.G.: Distributed computation with communication delays. *IEEE Transactions on Aerospace and Electronic Systems* **24,** No. 6, (1988) 700-712.

6. Cheng Y.C., Robertazzi T.G.: Distributed computation for a tree network with communication delays. *IEEE Transactions on Aerospace and Electronic Systems* **26,** No.3, (1990) 511-516.

7. Dandamundi S.: Performance analysis of a class of hierarchical hypercube multicomputer networks. *Performance Evaluation* **13** (1991) 159-179.

8. Du J., Leung J.Y-T.: Complexity of scheduling parallel task systems. *SIAM J. Discrete Mathemaics* **2** (1989) 473-487.

9. Gehringer E., Siewiorek D.: Segall Z.: *Parallel Processing. The Cm* Experience.* Digital Press, 1987.

10. Maloney M., Olsen P.: Proceeding of the Delta advanced user training class notes. Concurrent Supercomputing Consortium, California University of Technology Report CCSF-25-92, July 1992.

11. Ramamritham K., Stankovic J.A., Zhao W.: Distributed scheduling of tasks with deadlines and resource requirements. *IEEE Transactions on Computers* **38**, No.8, (1989) 1110-1123.

12. Scherson I., Corbett P.: Communications overhead and the expected speedup of multidimensional mesh-connected parallel processors. *Journal of Parallel and Distributed Computing* **11** (1991) 86-96.

13. Seitz C.: The Cosmic Cube. *Communications of the ACM* **28**, No. 1, (1985) 22-23.

14. Stankovic J.A., Ramamritham K., Cheng S.: Evaluation of a flexible task scheduling algorithm for distributed hard real-time systems. *IEEE Transactions on Computers* **34**, No.12, (1985) 1130-143.

15. Veltmann B., Lageweg B.J., Lenstra J.K., Multiprocessor scheduling with communication delays. *Parallel Computing* **16** (1990) 173-182.

Received August 31, 1995