ELSEVIER

Parallel Computing 25 (1999) 381–404

PARALLEL COMPUTING

Scheduling divisible loads in a three-dimensional mesh of processors

Maciej Drozdowski [a,1], Włodzimierz Głazek [b,*]

[a] *Institute of Computing Science, Poznań University of Technology, ul. Piotrowo 3A, 60-965 Poznań, Poland*
[b] *Department of Informatics, Technical University of Gdańsk, ul. Narutowicza 11/12, 80-952 Gdańsk, Poland*

Received 15 February 1998; received in revised form 16 August 1998

Abstract

We study distributed processing of a divisible load in a three-dimensional mesh of communicating processors. The objective is to find distribution of the load among processors which guarantees minimal processing time. We describe a family of load distribution algorithms and obtain closed-form formulae for optimal load shares allocated to processors in each algorithm. Our model takes into consideration communication delays involved in moving load shares from one processor to another. In large meshes our algorithms attain speedup limit of $1 + p/\rho$, where p is the number of communication ports used simultaneously by each processor in data transfer and ρ is the ratio of processing to communication transfer rate. We also show a matching upper bound on the speedup in this topology.

Keywords: Distributed processing; Divisible load; Mesh network; Scheduling

1. Introduction

Research in parallel computing has been very active in the last decade as indicated by thousands of publications in this area. Rapid progress in the design of parallel computer architectures has been matched by significant refinement of scheduling techniques. A major part of research on scheduling tasks on parallel computers done so far concerned indivisible tasks [6,10]. An *indivisible* task cannot be divided into smaller subtasks and has to be processed in its entirety. Only recently an alternative

* Corresponding author. E-mail: glazek@eti.pg.gda.pl
[1] Partially supported by the project CRIT2 and KBN grant.

0167-8191/99/$ – see front matter
PII: S0167-8191(99)00004-6

model of scheduling divisible loads has been brought under investigation [11]. A *divisible* load is characterized by its fine granularity and large volume, in which each data element requires exactly the same type of processing. Such a load may be partitioned among processors in the system in an arbitrary way, whereby each part is processed independently and separately of the other parts. Numerous applications may be modeled in this way. Processing of long linear data files occurs, for example, in signal and image processing, cryptography and pattern matching. In each of these cases data granularity is small in comparison to the data file volume so the problem of optimal data partitioning may be studied within divisible load paradigm. This leads to a tractable, continuous model of computation which may be analyzed and solved with standard methods of linear algebra and recursive analysis.

Study on the problems of this nature was originated in [3,11,12] for linear, tree and bus-oriented networks of processors. Subsequently, asymptotic results for performance limits in the above architectures were obtained in [7,15,21]. Closed-form solutions for load shares assigned to the processors were derived in [2,17]. Application of pipelining to distributing divisible loads in star networks was investigated in [4]. Multi-load distribution was considered in [5]. Constant communication setup time was first included in communication delay in work [8]. Scheduling divisible loads in networks with the circuit-switched routing was studied for the first time in [9]. The considered network topology was a two-dimensional mesh. A recursive distribution algorithm for a three-dimensional mesh of processors with the same method of routing was proposed recently in [13]. However, in neither of these last two works were closed-form solutions for load shares assigned to particular processors obtained.

In this paper we study processing a divisible load in a three-dimensional mesh of homogeneous processors. We assume message-passing system architecture and circuit-switched routing model. Each processor, equipped with local memory and a network front-end, may compute and communicate at the same time. Moreover, p communication ports can be used simultaneously by each processor. In the circuit-switched routing model the time to communicate a message of length L between two processors located d links apart from each other is usually modeled as $T_{\mathrm{com}} = S + d\delta + LC$, where S denotes communication setup time and C stands for *inverse* transfer rate of a single communication link (e.g. seconds per byte). Parameter δ, representing the time to commute a switch, is usually very small in comparison to setup time and will be neglected in further considerations.

We describe a family of load distribution algorithms and find closed-form solutions for optimal load shares allocated to the processors in each algorithm. We obtain an analytic formula for computation speedup which, in rather general circumstances, is symmetric to Amdahl's Law. It has been observed before [15] that communication delays have a similar effect on the performance of distributed computation in linear and star networks as the sequential portion of a parallel program has on the speedup according to this law. Our result provides theoretical explanation of these phenomena in the case of mesh networks. Next, we use closed-form solutions to demonstrate that the limit of speedup in a mesh network under the considered load distribution scheme is equal to $1 + p/\rho$, where $\rho = C/A$ and A is the

inverse processing rate of one processor (e.g. in seconds per byte). Finally, we prove a matching upper bound on speedup of processing in this topology. Although our investigations concern three-dimensional meshes, we note that the presented solutions also apply directly to the load distribution method in two-dimensional meshes studied in [9].

The rest of this paper is organized as follows. In Section 2 we present a principle for the recursive partitioning of the network and design load distribution algorithms based on this principle. In Section 3 we develop mathematical model of the load partition and obtain closed-form solutions for the load shares assigned to particular processors. In Section 4 we derive performance characteristics of the mesh network executing the considered computation scheme. In Section 5 we prove a general upper bound on computation speedup in the mesh network. Finally, in Section 6 we summarize our results.

2. Load distribution scheme

In this section we describe a family of recursive algorithms SCATTER(p), $1 \leqslant p \leqslant 5$, for distributing a divisible load in three-dimensional meshes and tori. Mesh and torus networks belong to the class of point-to-point networks. Formally, a *three-dimensional mesh* (or *torus*) has $n = k_0k_1k_2$ nodes, k_i along each dimension i, $0 \leqslant i \leqslant 2$, where $k_i \geqslant 2$. Each node x is identified by three coordinates, $c_2(x)$, $c_1(x)$, $c_0(x)$, where $0 \leqslant c_i(x) \leqslant k_i - 1$. In a three-dimensional mesh, two nodes x and y are neighbors if and only if $c_i(x) = c_i(y)$ for all i, $0 \leqslant i \leqslant 2$, except one, j, where $|c_j(x) - c_j(y)| = 1$. Nodes have from 3 to 6 neighbors, depending on their location in the mesh. In a three-dimensional torus, two nodes x and y are neighbors if and only if $c_i(x) = c_i(y)$ for all i, $0 \leqslant i \leqslant 2$, except one, j, where $|c_j(x) - c_j(y)| = 1$ or $|c_j(x) - c_j(y)| = k_j - 1$. Thus, a three-dimensional torus differs from a three-dimensional mesh only in *wrap-around* links in each dimension. Thus, all nodes have exactly six neighbors.

2.1. Principle of recursive network partition

The proposed load distribution algorithms are based on tiling of a mesh and take advantage of the fact that in the circuit-switched routing mode communication delay is virtually independent of the covered distance. This property allows each active processor to transfer chunks of load far away to a set of carefully selected processors, instead of sending data to its immediate neighbors. Subsequently, newly activated processors become load redistributors and together with their activator repeat the same scattering procedure in their non-overlapping vicinities. The common feature of all these algorithms is that each processor receives only one load shipment containing data addressed not only for an immediate receiver but also for all its successors in the scattering.

Each algorithm in the family repetitively executes three types of distribution moves in submeshes of decreasing size. The algorithms differ in the number of

communication ports p used simultaneously and in the way of selecting processors in successive moves. A set of processors accessed in the same move of scattering is called a *layer*. Three consecutive moves, each of different type, are called a *step*. A set of processors accessed in the same step of scattering is called a *basic cube*. Each step activates all processors of a basic cube. Then each of the active processors becomes a source of further distribution in the basic cube of a smaller size. The size of the activated basic cube decreases after each step by a factor of $1/(p+1)$. Also, each move increases the number of active processors p times, which is maximum possible as the number of communication ports used simultaneously is at most p. Initially, only one processor is active, after 1 distribution move there are $p+1$ active processors, etc., until after k distribution moves the number of cooperating processors is $(p+1)^k$. The maximum number of moves h is at most $\log_{p+1} n$, where n is the number of processors in the network.

2.2. *Algorithms* SCATTER(p)

It follows from the above description that in order to completely specify each algorithm it is sufficient to describe in details the three moves of one scattering step in which each load distribution algorithm SCATTER(p), $1 \leqslant p \leqslant 5$, activates a basic cube of processors. Distribution patterns are shown in Figs. 1 and 2. The simplest case is for $p=1$ and $p=2$ when each move of a step activates p processors along a different dimension.

In the 3-port system the originator activates three processors located in the same two-dimensional cross-section of the basic cube (say, along the plane $y0z$). Next, the four active processors send data along the third dimension (in this case dimension x). In the last move each active processor sends data to the neighbors along the hull of the basic cube and to one neighbor to the inside of the basic cube.

For the 4-port system the first move sends data from the originator to four processors located along one-dimension (e.g. z). Then each processor activates other processors located in a two-dimensional cross-section of a basic cube (along the plane $x0y$) [19].

In the 5-port system the originator located at coordinates $(x_0,\ y_0,\ z_0)$ activates processors at coordinates $(x_0+1,\ y_0-1,\ z_0-3)$, $(x_0, y_0,\ z_0-2)$, $(x_0+1,\ y_0-1,\ z_0-1)$, $(x_0+1,\ y_0-1,\ z_0+1)$ and $(x_0,\ y_0,\ z_0+2)$. The communication systems of processors at z coordinate values equal, respectively, to $z_0-3,\ z_0-2$ and $z_0-1,\ z_0$ and $z_0+1,\ z_0+2$, cooperate in pairs in the moves two and three. Each of the three pairs performs the same communications. In the second move each processor activated in the first move sends data to five neighbors with the same value of z coordinate. For that purpose four processors are activated using only links of the processors in one two-dimensional cross-section (the same z coordinate), and one processor is activated using links of the pairing two-dimensional cross-section. Thus, in each two-dimensional cross-section there are five active processors after move two. In the third move each active processor activates four more processors in the same two-dimensional cross-section and one in the pairing two-dimensional cross-section.

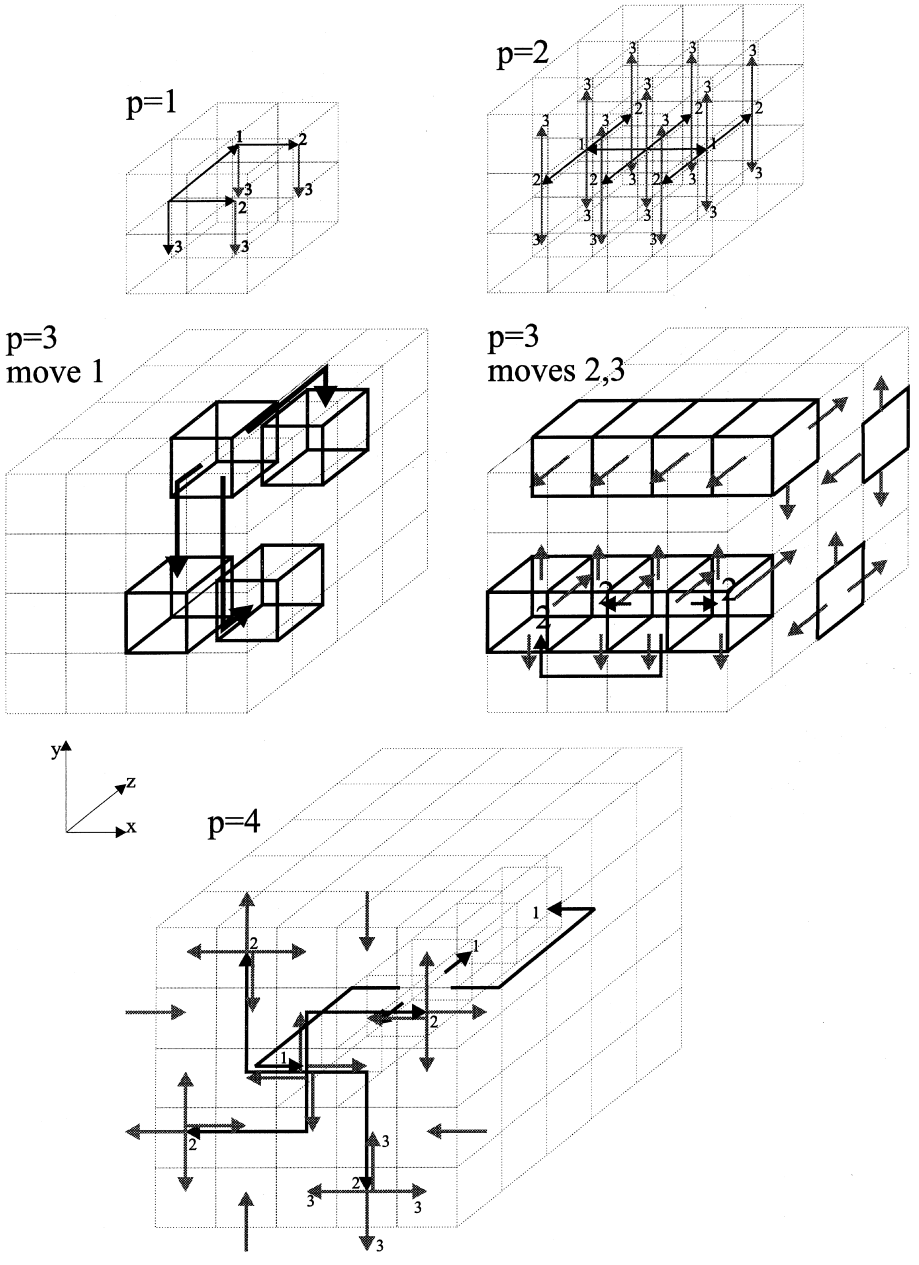

Fig. 1. Load distribution patterns in a basic cube of three-dimensional mesh for $p = 1, 2, 3, 4$.

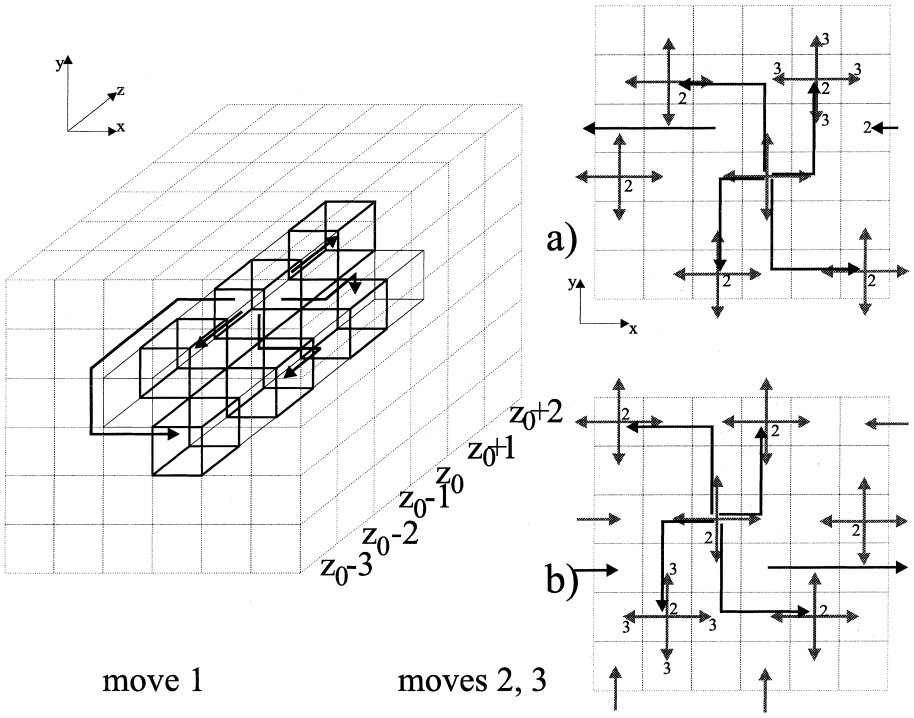

Fig. 2. Load distribution pattern in a basic cube of three-dimensional mesh for $p = 5$.

A method of scattering in a 6-port toroidal three-dimensional mesh has been proposed in [1,20].

Note that in the above scattering algorithms the following equation is satisfied for the basic cube: $(p+1)^k = x^d$, where k is the number of moves per step, x is the length of one edge of the basic cube, and d is the number of dimensions. It was shown in [20] that for meshes not all solutions of this equation lead to feasible tessellations of moves in the basic cube. In general, the existence of scattering algorithms for other values of p, k, x, d and other network topologies remains an open problem.

3. Mathematical model

Before presenting formal model let us introduce some notation. In the sequel symbol A denotes the *inverse* processing rate of a single processor (e.g. in seconds per byte) and V is the volume of the load that is to be processed. Symbol α_i, $0 \leqslant i \leqslant h$, stands for the load share in V assigned to each processor activated in move i.

For simplicity of the presentation we assume that no results of processing are returned to the originator. This assumption by no means restricts applicability of our model as we can always account for the time necessary to complete transfer of results

multiplying transfer rate C by a constant factor $1+\epsilon$, where 1 stands for scattered data and ϵ represents volume of returned results as a fraction of the received data. It was proved in [11] that in the optimal schedule without transfer of results all processors must finish computation at the same moment of time. The time diagram of load scattering and processing in a p-port 4-layer mesh which observes this rule is shown in Fig. 3.

Following the above assumptions, for each i, $0 \leqslant i \leqslant h-1$, the time of computation in layer i must take as long as the time of transmitting to layer $i+1$ plus the time of computation in layer $i+1$. Computation of α_i share of the load takes $A\alpha_i$ units of time. The part of the load communicated to a processor in layer i must accommodate data for all its descendants and be equal to $L = \alpha_i + p\alpha_{i+1} + p(p+1)\alpha_{i+2} + \cdots + p(p+1)^{h-i-1}\alpha_h$. Transmission of this load part takes $S + CL$ units of time. In this way we obtain h non-homogeneous linear recursive equations for $h+1$ variables. The last $(h+1)$th equation is the normalization equation which constrains the size of load shares so that parts of the load assigned to particular processors sum up to V. The share of the load to be processed by each processor can be determined from this model

$$
\begin{aligned}
&A\alpha_{h-1} = S + (A+C)\alpha_h,\\
&A\alpha_{h-i} = S + (A+C)\alpha_{h-i+1} + Cp\sum_{j=2}^{i}\alpha_{h-i+j}(p+1)^{j-2}, \quad i = 2,3,\ldots,h,\\
&V = \alpha_0 + p\sum_{i=1}^{h}\alpha_i(p+1)^{i-1}, \quad \alpha_0,\alpha_1,\ldots,\alpha_h > 0.
\end{aligned} \tag{1}
$$

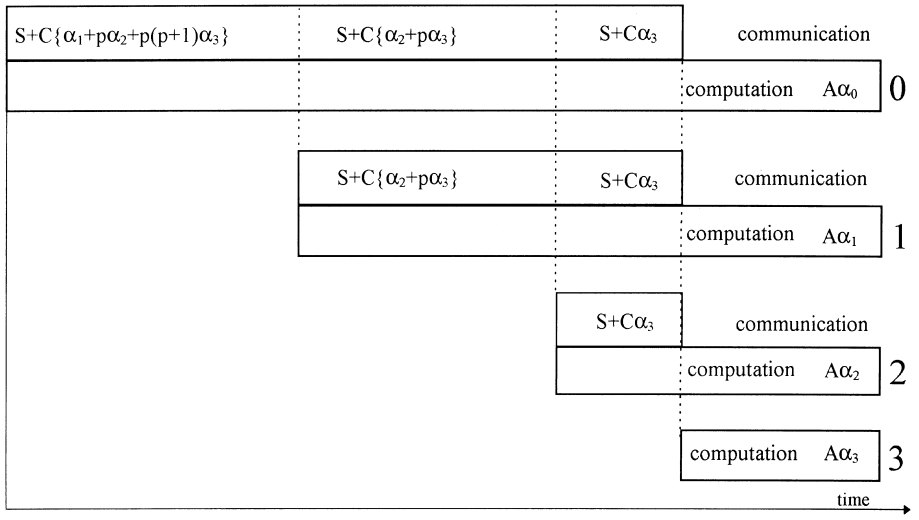

Fig. 3. Time diagram for load distribution in 4-layer three-dimensional mesh.

The generic distribution model (1) has p realizations, one for each scattering algorithm SCATTER(p) described above. Model (1) provides $h+1$ equations for $h+1$ unknowns. It may have no feasible solution for some h. This situation happens if communication speed is low and the front layers finish their assignments before data arrive to the most distant layers. In that case a smaller network should be considered. The maximum size of the usable network can be found by binary search in $O(\log h)$ calls to Eq. (1), or by using closed-form solutions obtained in this work. Note that in the optimal load assignment the originator is never idle. Hence, the whole load is processed by time $A\alpha_0$.

Theorem 1. *The model* (1) *has the following closed-form solutions*:

$$\alpha_{h-i} = \left(V + \frac{\sigma}{p+\rho}\right) \frac{p(p+\rho+1)^{-h} + \rho(p+\rho+1)^{i-h}}{p+\rho} + \frac{\sigma(ip-1)}{p+\rho} \quad \text{for } i = 0,1,\ldots,h, \tag{2}$$

where $\sigma = s/A$. *Hence, fraction* α_0 *of the total load processed by the originator amounts to*

$$\alpha_0 = \left(V + \frac{\sigma}{p+\rho}\right) \frac{p(p+\rho+1)^{-h} + \rho}{p+\rho} + \frac{\sigma(hp-1)}{p+\rho}. \tag{3}$$

Proof. A complete description of the method applied to solve Eq. (1) can be found in Appendix A. □

4. Speedup evaluation

Closed-form solutions obtained in the previous section allow us to evaluate performance of the considered load distribution algorithms with respect to different values of system parameters. In particular, we want to find out under what conditions a three-dimensional mesh of processors executing one of the algorithms SCATTER(p) may significantly outperform a single processor of the same kind as network processors. To this end performance will be measured in terms of computation *speedup* s_h defined as the ratio of sequential computation time on a single processor (e.g. originator) to the working time of the originator embedded in the $(p+1)^h$-node mesh of processors

$$s_h = \frac{AV}{A\alpha_0}. \tag{4}$$

First, we must ensure that the value of α_h given in Eq. (2) is always positive as it denotes load share allocated to each processor in the last layer. This restriction imposes an upper bound on the maximum number of layers $h_{\max}$ that can effectively participate in the computation

$$\alpha_h = \frac{V}{(p+\rho+1)^h} - \frac{\sigma}{p+\rho}\left[1 - \frac{1}{(p+\rho+1)^h}\right] > 0$$
$$\iff V\frac{p+\rho}{\sigma} + 1 > (p+\rho+1)^h$$
$$\iff h \leqslant h_{\max} = \left\lfloor \log_{p+\rho+1}\left(V\frac{p+\rho}{\sigma}+1\right)\right\rfloor. \tag{5}$$

Formula (3) for the originator's load consists of two parts: the first is an exponentially decreasing function of h, the second is a linearly increasing function of h. When h is large, the former converges to the limiting value of $(V+\sigma/(p+\rho))(\rho/(p+\rho))$ while the latter has no limit. Hence, when circuit setup time S is non-negligible then the speedup offered by each of the considered algorithms

$$s_h = \frac{p+\rho}{\left(1+\frac{\sigma}{V(p+\rho)}\right)\left(\rho + p(p+\rho+1)^{-h}\right) + \frac{\sigma}{V}(hp-1)}, \tag{6}$$

does not converge to a finite limit but achieves its maximum for the optimal number of layers h_{opt}. This number can be determined by finding the minimum of denominator in Eq. (6), denoted here by $b(h)$

$$\frac{\mathrm{d}b(h)}{\mathrm{d}h} = 0$$
$$\iff -\left(1+\frac{\sigma}{V(p+\rho)}\right)p\ln(p+\rho+1)(p+\rho+1)^{-h} + \frac{\sigma}{V}p = 0. \tag{7}$$

Consequently, the optimal number of layers h_{opt} participating in computation is equal to the solution of Eq. (7)

$$h_{\text{opt}} = \log_{p+\rho+1}\left(V\frac{p+\rho}{\sigma}+1\right) - \log_{p+\rho+1}\left(\frac{p+\rho}{\ln(p+\rho+1)}\right), \tag{8}$$

rounded to the nearest integer value. The difference between real values of h_{opt} and $h_{\max}$ is at most 0.72 in the range $10^{-3} \leqslant \rho \leqslant 10^3$ and disappears in rounding of Eqs. (5) and (8). Thus, $h_{\max}$ and h_{opt} coincide so it is desirable for the optimal performance to use as many network layers as possible.

The following figures show computer plots of speedup attained by the proposed load distribution algorithms. In Fig. 4 speedup vs number of processors working on a divisible load is depicted. Communication parameters are set to values typical of the commercial three-dimensional mesh system CRAY T3D: $C = 3.3$ ns/byte and $S = 8.57$ μs. It is also assumed that $A = 1$ μs/byte and $V = 10^6$ bytes. There are three speedup curves for $p = 1, 3, 6$. It can be observed that for each p speedup initially grows exponentially with the number of utilized layers. Later, the curves level off which means that adding more processors offers only a diminishing speedup growth. Each speedup growth curve attains a different speedup limit which depends on the value of p. Below we determine the exact value of the speedup limit for each p. We may safely assume that circuit setup time S is negligible in comparison to AV, the time of processing the whole load in a single processor. In that case originator's load share Eq. (3) reduces to

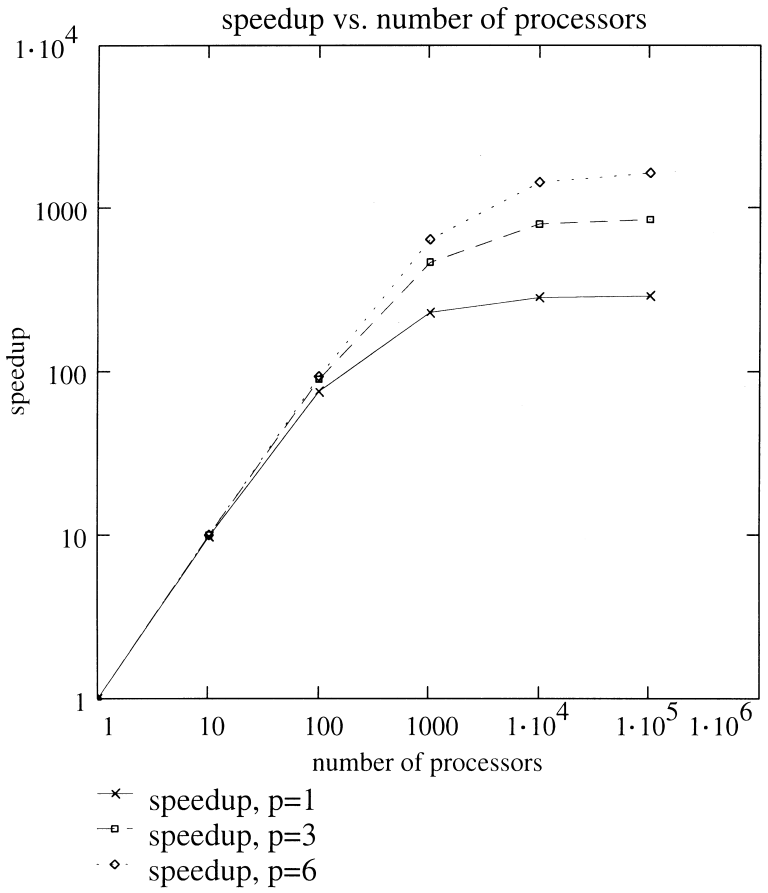

Fig. 4. Speedup vs number of processors for $p = 1, 3, 6$.

$$\alpha_{0|\frac{\sigma}{V}\simeq 0} = V\frac{p(p+\rho+1)^{-h}+\rho}{p+\rho} \tag{9}$$

and the formula for speedup takes a much simpler form

$$s_h = \frac{\rho+p}{\rho+\frac{1}{(p+\rho+1)^h}p}. \tag{10}$$

It is well known that a parallel computer consisting of n processors cannot obtain n-fold speedup because the sequential component of the computation limits the speed of the total process. This effect is usually expressed as Amdahl's Law [16]

$$s_n = \frac{T_s + T_p}{T_s + \frac{1}{n}T_p}, \tag{11}$$

where T_s and T_p are, respectively, the time spent on serial and parallel components of the program in a single processor. It was observed in the earlier study [15] that communication delays cause a similar effect on speedup of distributed computation

in linear and star networks as the sequential portion of a parallel program affects speedup of parallel computing according to Amdahl's Law. The correspondence of Amdahl's Law (11) and the speedup formula (10) provides theoretical explanation of these phenomena in the case of mesh networks.

Formula (10) may be used to derive the limit of speedup attained by our load distribution algorithms in the mesh network. For growing h speedup s_h converges to

$$\lim_{h\to\infty} s_h = \frac{\rho + p}{\rho} = 1 + \frac{p}{\rho}. \tag{12}$$

Corollary 2. *The ultimate speedup attained by load processing scheme* (1) *in the three-dimensional mesh for negligible* σ/V *is equal to* $1 + p/p$.

Note that in this speedup bound the slower processors the higher speedup can be obtained. This is well-known counter-intuitive artifact resulting from the definition of speedup. Intuitively, it can be said that the faster the processors are the worse scalability our algorithms have.

Fig. 5 shows how the speedup is affected by various values of ρ. The middle plot (reference) corresponds to $\rho = 3.3 \times 10^{-3}$ computed for the above parameters. In the lower plot this ratio is an order of magnitude bigger than in the reference system, while in the upper plot it is an order of magnitude smaller. As might be expected, speedup grows with the increase of speed of data transfer over speed of processing because more processors can join computation in shorter time.

In Fig. 6 load shares are plotted against layer number for three values of the number of communication ports used by each processor $p = 1, 3, 6$. A small number of ports puts a greater burden on the front layers which results in poor performance. Conversely, a greater number of available ports facilitates sending more data to deeper layers and permits the originator to take only a tiny fraction of the total work.

Fig. 7 illustrates the impact of ρ on layers' load shares in the computation. Three plots of load shares are drawn in the same settings as discussed in Fig. 5. As the speed of communication increases the bigger part of computation is performed by deeper layers with the maximum share located in the last layer.

The last two results are particularly significant. In the perfect network with no communication delays the load of a layer is proportional to the number of layer processors which, in our case, is exponential in the layer index. Practically, subtle interplay of parameters n, p, A, C and S flattens and reshapes the perfect load distribution curve. Still, our results coincide with the earlier observations for point-to-point networks [7,9]: most of the load is processed in deeper layers while the front layer communicate more. This may be contrasted to the result for bus and tree networks [2], where most of the load stays in the front layers. The difference may be attributed to at least two factors. The prime factor responsible for this difference of behavior is that deeper layers are much easier accessible in the circuit-switched routed networks than in the store-and-forward routed networks. In the circuit-switched routing the transmission time is virtually independent of the covered distance, therefore our algorithms take advantage of this fact and place the bulk of the

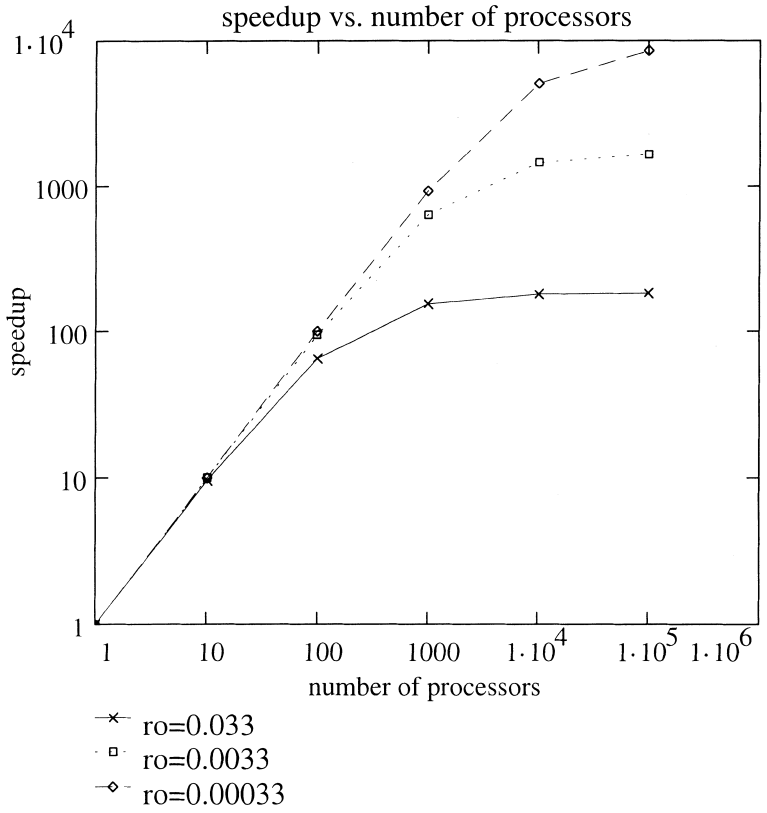

Fig. 5. Speedup vs number of processors for $p = 6$ and different values of ρ.

load in deeper, more powerful layers of exponentially growing size. Another reason is in the definition of a layer. In the former work a layer is a single level of a regular p-ary tree. In our work a layer is defined relative to a different kind of p-ary tree induced by the pattern of load distribution and contains nodes which are successors of all the nodes from the earlier levels of such a tree (this is so-called binomial tree). Due to the structure of the binomial tree the number of processors in a layer grows faster than in the regular p-ary tree.

Comparing Fig. 4 with Fig. 5 and Fig. 6 with Fig. 7 one cannot miss certain similarity of corresponding figures. Clearly, speedup is affected by the number of communication ports p in an inverse way as it is influenced by the ratio ρ (up to the scaling factor). This is not surprising as formula (10) already anticipated this effect.

5. Upper bound on speedup

In this section we obtain an upper bound on the speedup of processing divisible loads in a multiprocessor network with three-dimensional mesh topology. The

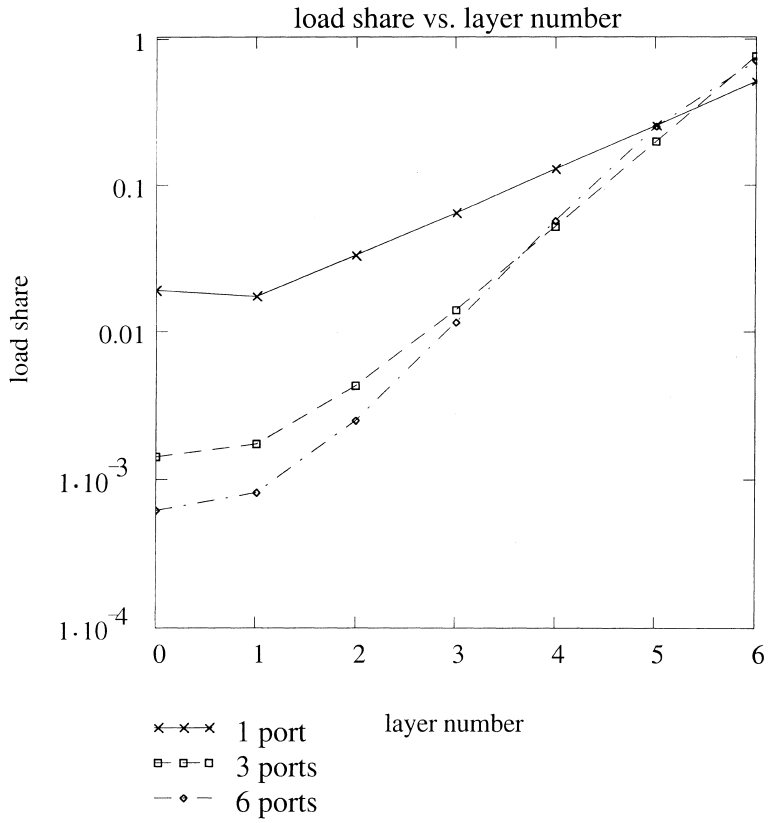

Fig. 6. Load share vs layer number for $p = 1, 3, 6$.

bound is quite general as it makes no assumptions on the type of scattering algorithm. It is inherently related to the cost of communication in a network and reflects the communication 'bottleneck' between the originator and the rest of the network.

Theorem 3. *The maximum speedup offered by an n-processor p-port three-dimensional mesh where load is initially distributed from one node is bounded from above by function* $u_n = \min\{n,\ 1 + (p/\rho)\}$.

Proof. We shall use a dual definition of speedup, that is, we define speedup as the ratio of the load volume that might be processed in a network in a given time interval to the load volume processed in a single processor in the same time. It is not hard to see that both definitions are equivalent. Let us split n-node mesh network N into originator P_0 and the remaining part of mesh M (i.e. $N = \{P_0\} \cup M$). Consider load processing in network N in a fixed time interval $T \gg S$ according to some optimal scheduling algorithm OPT.

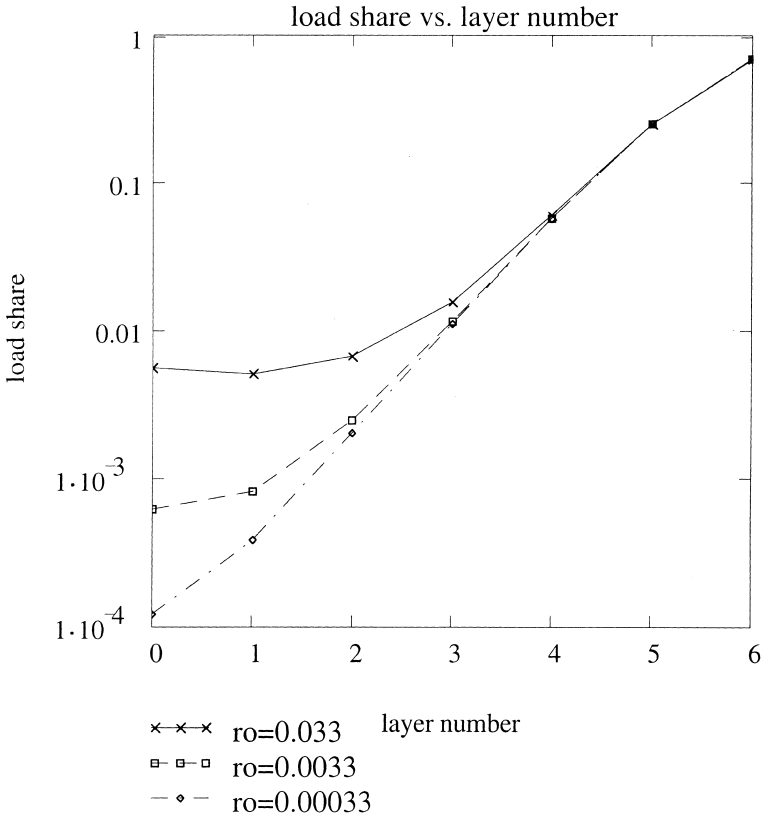

Fig. 7. Load share vs layer number for $p=6$ and different values of ρ.

During this time originator P_0 may process at most $V_0 = T/A$ data units. The amount of data that might be processed in M is limited by two factors: communication bandwidth of links connecting originator P_0 to subnetwork M and processing speed of M. Processor P_0 may communicate data to M over p links simultaneously. Let us represent subnetwork M as a single processor P_M with the 'equivalent' processing rate A_M connected to P_0 by a single link with the 'equivalent' communication rate C_M. Without loss of generality let us assume that algorithm OPT operates in g steps, whereby ith round supplies P_M with amount ϵ_i of data for processing. Computation and communication phases in P_M may be interleaved and/or overlapped in some way. On the other hand, the total amount of load data which may be communicated to P_M in time interval T is bounded from above

$$S + \sum_{i \leqslant g} \epsilon_i C_M \leqslant T. \tag{13}$$

A similar upper bound holds for the amount of data which may be processed in P_M in time interval T

$$S + \sum_{i \leqslant g} \epsilon_i A_M \leqslant T. \tag{14}$$

Denoting by $V_M = \sum_i \epsilon_i$, we conclude from Eqs. (13) and (14) that the amount of data that might be sent to and processed in P_M during time T is bounded from above by

$$V_M \leqslant \frac{T - S}{\max\{A_M,\ C_M\}}. \tag{15}$$

Since we assume that P_0 is the only source of data in a network, therefore $V_0 + V_M$ is the upper bound on the divisible load volume which might be processed in time T by any scheduling algorithm OPT. Thus, the speedup of processing a divisible load in a mesh network is bounded from above by

$$s_n \leqslant \frac{V_0 + V_M}{V_0} = 1 + \left(1 - \frac{S}{T}\right) \min\left\{\frac{A}{A_M},\ \frac{A}{C_M}\right\}. \tag{16}$$

Subnetwork M consists of $n-1$ processors, therefore $A_M \geqslant A/(n-1)$. Similarly, $C_M \geqslant C/p$. Now we consider two cases depending on the network size.

Case 1. Mesh size n is small. With the decrease of n the 'equivalent' processing rate A_M increases but C_M decreases because all nodes are within a close distance from the originator. Hence, for $A_M > C_M$ Eq. (16) reduces to

$$s_n \leqslant 1 + \left(1 - \frac{S}{T}\right) \frac{A}{A_M} \leqslant 1 + \frac{A}{A_M} \leqslant 1 + (n-1) = n. \tag{17}$$

Case 2. Mesh size n is large. With the increase of n the 'equivalent' processing rate A_M decreases but C_M increases because more communication steps are needed to feed further located nodes. Hence, for $A_M \leqslant C_M$ Eq. (16) reduces to

$$s_n \leqslant 1 + \left(1 - \frac{S}{T}\right) \frac{A}{C_M} \leqslant 1 + \frac{A}{C/p} = 1 + \frac{p}{\rho}. \tag{18}$$

Combining both cases (17) and (18) we get

$$s_n \leqslant u_n = \min\left\{n,\ 1 + \frac{p}{\rho}\right\}. \qquad \square \tag{19}$$

6. Summary

In this paper we studied scheduling a divisible load in a three-dimensional mesh of communicating processors. We obtained closed-form solutions for load shares assigned to processors and analyzed various characteristics of distributed computation. We demonstrated that our distribution algorithms attain the speedup limit of $1 + p/\rho$, which is the best possible in the considered model. This value was shown to be determined by architectural constraints of the network (i.e., number of originators, number of communication ports, processing and communication speeds). It is

independent of any particular load distribution method. Another conclusion is that the way in which linear communication delays influence speedup of distributed computation closely resembles the impact of sequential parts of parallel program on the same metric.

The study was done under the assumption that parameter δ, representing the time to commute a switch, is very small in comparison to setup time S and can be safely neglected. In some parallel systems this is not the case (for example, in IPSC/860 parameter δ may be even as high as one sixth of S). Nevertheless, in most current machines, message transmissions are initiated by software and switching is done in hardware, so δ is usually much smaller than S. In fact, experiments with circuit-switched and wormhole-routed machines show that communication latencies are nearly independent of distance [18]. Another important issue is the length of data distribution paths. Currently available systems usually have no more than few thousand of processors, therefore for bigger number of ports p (say, $p = 5$) the length of the longest path remains small. In the very large systems the first step paths may become relatively long but the transmitted load is also large, so both δ and S are small as compared to the linear component. The impact of δ and the length of paths on effectiveness of the proposed algorithms shall be subject to further research.

Note that our analysis concerned distribution model which allowed only one data shipment per processor. Further research shall concentrate on designing more efficient distribution algorithms based on pipelining technique described in [4,13]. Another direction for study is to find out whether scattering of load shares to processors ordered differently than according to the increasing distance from the originator considered here may bring any improvement. This idea is currently under investigation.

Acknowledgements

The authors are grateful to the anonymous referees for providing comments and suggestions which greatly helped to improve this presentation.

Appendix A. Proof of Theorem 1

For notational convenience we relabel our variables so that

$$\beta_0 = \alpha_h, \ldots, \beta_i = \alpha_{h-i}, \ldots, \beta_h = \alpha_0. \tag{A.1}$$

Henceforth, $\rho = C/A$, and $\sigma = S/A$. The transformed equations corresponding to Eqs. (1) and (A.1) are as follows:

$$\beta_1 = \sigma + (1+\rho)\beta_0, \tag{A.2}$$

$$\beta_j = \sigma + (1+\rho)\beta_{j-1} + \rho p \sum_{k=0}^{j-2} \beta_k (p+1)^{j-2-k}, \quad j = 2, 3, \ldots, h, \tag{A.3}$$

$$V = \beta_h + p\sum_{k=0}^{h-1}\beta_k(p+1)^{h-1-k}. \tag{A.4}$$

Each $\beta_j,\ 1 \leqslant j \leqslant h$, can be expressed in terms of β_0, load fraction assigned to each processor in the last layer. From the normalization Eq. (A.4) we can compute the exact value of β_0. Once the value of β_0 is known, all β_j's can be obtained easily. To this end we rewrite Eq. (A.3) so that they do not contain sums of variables. Eq. (A.3) for β_{j-1} and β_j may be rewritten as

$$\beta_{j-1} - \sigma - (1+\rho)\beta_{j-2} = \rho p\sum_{k=0}^{j-3}\beta_k(p+1)^{j-3-k} \tag{A.5}$$

and

$$\beta_j = \sigma + (1+\rho)\beta_{j-1} + \rho p\beta_{j-2} + (p+1)\rho p\sum_{k=0}^{j-3}\beta_k(p+1)^{j-3-k}. \tag{A.6}$$

Eq. (A.6) contains RHS of Eq. (A.5). Replacing it by its LHS equivalent and regrouping terms we obtain

$$\beta_j = (p+\rho+2)\beta_{j-1} - (p+\rho+1)\beta_{j-2} - p\sigma, \quad j = 2,3,\ldots,h. \tag{A.7}$$

It is possible to write each $\beta_j,\ 1 \leqslant j \leqslant h$, as a function of β_0 and σ

$$\beta_j = g_j\sigma + f_j\beta_0, \tag{A.8}$$

where

$$\begin{aligned} f_j &= f(j,\ 0)\rho^0 + f(j,\ 1)\rho^1 + \cdots + f(j,\ j)\rho^j = \sum_{i=0}^{j} f(j,\ i)\rho^i, \\ g_j &= g(j,\ 0)\rho^0 + g(j,\ 1)\rho^1 + \cdots + g(j,\ j)\rho^j = \sum_{i=0}^{j} g(j,\ i)\rho^i. \end{aligned} \tag{A.9}$$

Functions f_j and g_j are polynomials in ρ and each $f(j, i)$ (resp. $g(j, i)$) represents a coefficient of ρ^i in f_j (resp. g_j). They can be determined from Eqs. (A.2), (A.7) and (A.9)

$$\begin{aligned} f(j,\ i) &= (p+1)f(j-1,\ i) + f(j-1,\ i-1) \\ f(j,\ 0) &= 1, \quad f(j,\ j+1) = 0 \\ f(0,\ 0) &= 1, \quad f(1,\ 1) = 1, \qquad \text{for } 1 \leqslant i \leqslant j \leqslant h, \end{aligned} \tag{A.10}$$

and

$$\begin{aligned} g(j,\ i) &= (p+1)g(j-1,\ i) + g(j-1,\ i-1) \\ g(j,\ 0) &= j, \quad g(j,\ j) = 0 \\ g(0,\ 0) &= 0, \quad g(1,\ 1) = 0, \qquad \text{for } 1 \leqslant i \leqslant j \leqslant h. \end{aligned} \tag{A.11}$$

Note that $f(j, i)$ and $g(j, i)$ are given by the same recursive relation but with different boundary conditions, which leads to entirely different function values.

Expanding Eq. (A.10) one can observe that each coefficient $f(j,\ i)$ is, in turn, a polynomial of $p+1$

$$f(j,\ i) = f(j,\ i,\ 0)(p+1)^0 + \cdots + f(j,\ i,\ j-i)(p+1)^{j-i}. \tag{A.12}$$

This allows us to represent each $f(j, i)$ as a number in radix $p+1$ representation with coefficients $f(j, i, k)$ taking place of digits. Similarly we can expand Eq. (A.11) and obtain

$$g(j,\ i) = g(j,\ i,\ 0)(p+1)^0 + \cdots + g(j,\ i,\ j-i-1)(p+1)^{j-i-1}. \tag{A.13}$$

In order to find closed forms of polynomials f_j and g_j we build two tables of coefficients, one for each of the above functions. Table 1 shows $(p+1)$-based expansions of $f(j, i)$ for $0 \leqslant i,\ j \leqslant 6$. Given i and j, radix p+1 representation of $f(j, i)$ is located in jth row and ith column of the table. The cell for $f(j, i)$ contains a sequence of coefficients $f(j,\ i,\ 0),\ f(j,\ i,\ 1),\ \ldots, f(j,\ i,\ j-i-1),\ f(j,\ i,\ j-i)$ in the following order: $f(j, i, 0)$ is in the rightmost position (the least significant bit position), $f(j, i, 1)$ is in the second position from the right, etc. Empty cells contain zeros. To obtain f_j, one takes the whole jth row and plugs each $f(j,\ i),\ 0 \leqslant i \leqslant j$, into Eq. (A.9).

Table 1 is a generalized Pascal's triangle. In the original Pascal's triangle the entry in jth row and ith column is binomial coefficient $\binom{j}{i}$. In Table 1 this entry corresponds to the expansion of $f(j, i)$ Eq. (A.12) and contains a sequence of binomial coefficients. Namely, for each i and j, coefficients $f(j, i, 0),\ldots, f(j, i, j-i)$ can be obtained from the original Pascal's triangle taking a sequence of $j - i + 1$ consecutive binomial coefficients in column $i-1$ from $\binom{i-1}{i-1}$ until $\binom{j-1}{i-1}$. Hence, each $f(j,\ i,\ k),\ 0 \leqslant k \leqslant j-i$, is equal to $\binom{i-1+k}{i-1}$. Moreover, all $f(j, i, k)$'s in the cell $f(j, i)$ sum up to $\binom{j}{i}$. In other words, by setting $p=0$ in Eq. (A.12) one obtains from Table 1 the original Pascal's triangle. Similarly we derive Table 2 of coefficients for $g(j, i)$. Now we can write $f(j, i)$ in a concise way as

Table 1
Coefficients $f(j, i)$ for $0 \leqslant i,\ j \leqslant 6$

i	0	1	2	3	4	5	6
j							
0	1						
1	1	1					
2	1	1,1	1				
3	1	1,1,1	2,1	1			
4	1	1,1,1,1	3,2,1	3,1	1		
5	1	1,1,1,1,1	4,3,2,1	6,3,1	4,1	1	
6	1	1,1,1,1,1,1	5,4,3,2,1	10,6,3,1	10,4,1	5,1	1

Table 2
Coefficients $g(j,\ i)$ for $0 \leqslant i,\ j \leqslant 6$

i	0	1	2	3	4	5	6
j							
0							
1	1						
2	2	1					
3	3	1,2	1				
4	4	1,2,3	2,2	1			
5	5	1,2,3,4	3,4,3	3,2	1		
6	6	1,2,3,4,5	4,6,6,4	6,6,3	4,2	1	

$$f(j,\ i) = \sum_{k=0}^{j-i} \binom{i-1+k}{i-1} (p+1)^k. \tag{A.14}$$

For given j let us consider the difference $f_j - f_{j-1}$. In Table 1 this corresponds to the difference of jth and $(j-1)$th rows. Each column $i,\ 1 \leqslant i \leqslant j$, contributes to the difference the leftmost term $f(j,\ i,\ j-i)$ multiplied by $\rho^i \cdot (p+1)^{j-i}$. Summing all components up we get

$$\begin{aligned} f_j - f_{j-1} &= \sum_{i=1}^{j} f(j,\ i,\ j-i)\rho^i (p+1)^{j-i} \\ &= \sum_{i=1}^{j} \binom{(i-1)+(j-i)}{i-1} \rho^i (p+1)^{j-i} \\ &= \rho \sum_{i=1}^{j} \binom{j-1}{i-1} \rho^{i-1} (p+1)^{j-i} \\ &= \rho \sum_{k=0}^{j-1} \binom{j-1}{k} \rho^k (p+1)^{(j-1)-k} \\ &= \rho(p+\rho+1)^{j-1}. \end{aligned} \tag{A.15}$$

We can add up differences obtained in Eq. (A.15) with the initial condition $f_0 = \beta_0$, which gives us a closed-form for f_j

$$\begin{aligned} f_j &= (f_j - f_{j-1}) + (f_{j-1} - f_{j-2}) + \cdots + (f_2 - f_1) + (f_1 - f_0) + f_0 \\ &= \sum_{i=0}^{j-1} \left(\rho(p+\rho+1)^i\right) + 1 \\ &= \rho \frac{(p+\rho+1)^j - 1}{p+\rho} + 1. \end{aligned} \tag{A.16}$$

In the case of polynomial g_j, the difference $g_j - g_{j-1}$ is equal to the difference of jth and $(j-1)$th row of Table 2. Comparing Tables 1 and 2 one can see that this difference amounts to the $(j-1)$th row of Table 1

$$g_j - g_{j-1} = f_{j-1}. \tag{A.17}$$

Adding up differences from Eq. (A.17) with the initial condition $g_0 = 0$ we obtain

$$\begin{aligned} g_j &= (g_j - g_{j-1}) + (g_{j-1} - g_{j-2}) + \cdots + (g_2 - g_1) + (g_1 - g_0) + g_0 \\ &= \sum_{i=0}^{j-1} f_i = \sum_{i=0}^{j-1} \left(\rho \frac{(p+\rho+1)^i - 1}{p+\rho} + 1 \right) \\ &= \frac{1}{p+\rho} \sum_{i=0}^{j-1} \left(\rho (p+\rho+1)^i + p \right) \\ &= \frac{1}{p+\rho} \left(\rho \frac{(p+\rho+1)^j - 1}{p+\rho} + jp \right) \\ &= \frac{1}{p+\rho} \left(f_j + jp - 1 \right). \end{aligned} \tag{A.18}$$

Closed-form formula (A.18) for g_j facilitates expressing β_j in terms of f_j and j

$$\begin{aligned} \beta_j &= \frac{\sigma}{p+\rho} \left(f_j + jp - 1 \right) + \beta_0 f_j \\ &= \left(\beta_0 + \frac{\sigma}{p+\rho} \right) f_j + \frac{\sigma}{p+\rho} (jp - 1). \end{aligned} \tag{A.19}$$

Given Eq. (A.19) we can resolve normalization Eq. (A.4). Before we do that let us introduce two auxiliary coefficients $c = \beta_0 + \sigma/(p+\rho)$ and $d_j = (\sigma/(p+\rho))(jp-1)$, which allows us to split normalization sum of Eq. (A.4) into two easier manageable parts V_1 and V_2.

$$\begin{aligned} V &= \beta_h + p \sum_{j=0}^{h-1} \beta_j (p+1)^{h-1-j} \\ &= c f_h + d_h + p \sum_{j=0}^{h-1} (c f_j + d_j)(p+1)^{h-1-j} \\ &= c \left(f_h + p \sum_{j=0}^{h-1} f_j (p+1)^{h-1-j} \right) + \left(d_h + p \sum_{j=0}^{h-1} d_j (p+1)^{h-1-j} \right) \\ &= V_1 + V_2, \end{aligned} \tag{A.20}$$

where

$$V_1 = c\left(f_h + p\sum_{j=0}^{h-1} f_j (p+1)^{h-1-j}\right)$$
$$= cf_h + cp\sum_{j=0}^{h-1}\left[\sum_{i=0}^{j}\rho^i\left(\sum_{k=0}^{j-i}\binom{i-1+k}{i-1}(p+1)^k\right)\right](p+1)^{h-1-j} \tag{A.21}$$

and

$$V_2 = d_h + p\sum_{j=0}^{h-1} d_j (p+1)^{h-1-j}$$
$$= \frac{\sigma}{p+\rho}(hp-1) + p\sum_{j=0}^{h-1}\left(\frac{\sigma}{p+\rho}(jp-1)\right)(p+1)^{h-1-j}. \tag{A.22}$$

First, we evaluate V_1 by means of interchanging the order of sums twice so that summation over j is done first

$$V_1 = cf_h + cp\sum_{i=0}^{h-1}\rho^i\left[\sum_{j=i}^{h-1}\left(\sum_{k=0}^{j-i}\binom{i-1+k}{i-1}(p+1)^k\right)(p+1)^{h-1-j}\right]$$
$$= cf_h + cp\sum_{i=0}^{h-1}\rho^i\left[\sum_{l+i=i}^{l+i=h-1}\left(\sum_{k=0}^{(l+i)-i}\binom{i-1+k}{i-1}(p+1)^k\right)(p+1)^{h-1-(l+i)}\right]$$
$$= cf_h + cp\sum_{i=0}^{h-1}\rho^i\left[\sum_{l=0}^{h-1-i}\sum_{k=0}^{l}\binom{i-1+k}{i-1}(p+1)^{h-1-i-l+k}\right]$$
$$= cf_h + cp\sum_{i=0}^{h-1}\rho^i\left[\sum_{k=0}^{h-1-i}\binom{i-1+k}{i-1}(p+1)^k\left(\sum_{l=k}^{h-1-i}(p+1)^{h-1-i-l}\right)\right]$$
$$= cf_h + cp\sum_{i=0}^{h-1}\rho^i\left[\sum_{k=0}^{h-1-i}\binom{i-1+k}{i-1}(p+1)^k\frac{(p+1)^{h-i-k}-1}{p}\right]$$
$$= cf_h + cp\sum_{i=0}^{h-1}\rho^i\left[\sum_{k=0}^{h-1-i}\binom{i-1+k}{i-1}\frac{(p+1)^{h-i}-(p+1)^k}{p}\right]$$
$$= cf_h + cp\sum_{i=0}^{h-1}\frac{\rho^i}{p}\left[(p+1)^{h-i}\sum_{k=0}^{h-1-i}\binom{i-1+k}{i-1} - \sum_{k=0}^{h-1-i}\binom{i-1+k}{i-1}(p+1)^k\right]. \tag{A.23}$$

At that moment we can apply parallel summation law [14] to the first sum of binomial coefficients in Eq. (A.23) and due to Eq. (A.14) we replace the second sum by $f(h-1,\ i)$, so that

$$
\begin{aligned}
V_1 &= cf_h + c\sum_{i=0}^{h-1}\rho^i\left[(p+1)^{h-i}\binom{h-1-i+i-1+1}{h-1-i} - f(h-1,\ i)\right]\\
&= c(p+1)\sum_{i=0}^{h-1}\rho^i(p+1)^{h-i-1}\binom{h-1}{h-1-i} + c\left(f_h - \sum_{i=0}^{h-1}\rho^i f(h-1,\ i)\right)\\
&= c(p+1)\sum_{i=0}^{h-1}\rho^i(p+1)^{h-i-1}\binom{h-1}{i} + c(f_h - f_{h-1})\\
&= c\left[(p+1)(p+\rho+1)^{h-1} + \rho(p+\rho+1)^{h-1}\right]\\
&= c(p+\rho+1)^h. \qquad \text{(A.24)}
\end{aligned}
$$

The second part of Eq. (A.20) denoted as V_2 can be transformed to

$$
\begin{aligned}
V_2 &= \frac{\sigma(hp-1)}{p+\rho} + \frac{p\sigma}{p+\rho}\sum_{h-1-j=0}^{h-1-j=h-1}[(h-1-j)p-1](p+1)^{h-1-(h-1-j)}\\
&= \frac{\sigma}{p+\rho}\left\{hp-1+p\sum_{j=h-1}^{j=0}[(h-1)p-jp-1](p+1)^j\right\}\\
&= \frac{\sigma}{p+\rho}\left\{hp-1+p[(h-1)p-1]\sum_{j=h-1}^{j=0}(p+1)^j - p^2\sum_{j=h-1}^{j=0}j(p+1)^j\right\}\\
&= \frac{\sigma}{p+\rho}\left\{hp-1+p(hp-p-1)\frac{(p+1)^h-1}{p}\right\}\\
&\quad - \frac{\sigma}{p+\rho}\left\{p^2\frac{(p+1)-h(p+1)^h+(h-1)(p+1)^{h+1}}{p^2}\right\}\\
&= -\frac{\sigma}{p+\rho}. \qquad \text{(A.25)}
\end{aligned}
$$

It follows from Eqs. (A.20), (A.24), (A.25) that

$$
V = \left(\frac{\sigma}{p+\rho}+\beta_0\right)(p+\rho+1)^h - \frac{\sigma}{p+\rho},
$$

so β_0 takes the form of

$$
\beta_0 = \alpha_h = \frac{V}{(p+\rho+1)^h} - \frac{\sigma}{p+\rho}\left[1-\frac{1}{(p+\rho+1)^h}\right]. \qquad \text{(A.26)}
$$

Finally, by the use of Eqs. (A.16), (A.19) and (A.26) the closed-form for $\beta_j, 1 \leqslant j \leqslant h$, can be derived

$$\begin{aligned}\beta_j = \alpha_{h-j} &= \left(\beta_0 + \frac{\sigma}{p+\rho}\right) f_j + \frac{\sigma}{p+\rho}(jp-1) \\ &= \left[\frac{V}{(p+\rho+1)^h} - \frac{\sigma}{p+\rho}\left(1 - \frac{1}{(p+\rho+1)^h}\right) + \frac{\sigma}{p+\rho}\right] f_j + \frac{\sigma}{p+\rho}(jp-1) \\ &= \frac{\left(V + \frac{\sigma}{p+\rho}\right)}{(p+\rho+1)^h} \frac{\left(\rho(p+\rho+1)^j + p\right)}{p+\rho} + \frac{\sigma}{p+\rho}(jp-1) \\ &= \frac{V + \frac{\sigma}{p+\rho}}{p+\rho}\left(\rho(p+\rho+1)^{j-h} + p(p+\rho+1)^{-h}\right) + \frac{\sigma(jp-1)}{p+\rho}.\end{aligned}$$

References

[1] T. Armitage, J.G. Peters, Circuit-switched broadcasting in 3-dimensional toroidal meshes, manuscript, 1995.

[2] S. Bataineh, T.-Y. Hsiung, T.G. Robertazzi, Closed form solutions for bus and tree networks of processors load sharing a divisible job, IEEE Transactions on Computers 43 (1994) 1184–1196.

[3] S. Bataineh, T.G. Robertazzi, Bus oriented load sharing for a network of sensor driven processors, IEEE Transactions on Systems, Man and Cybernetics 21 (1991) 1202–1205.

[4] V. Bharadwaj, D. Ghose, V. Mani, Multi-installment load distribution in tree networks with delays, IEEE Transactions on Aerospace and Electronic Systems 31 (1995) 555–567.

[5] V. Bharadwaj, D. Ghose, V. Mani, T.G. Robertazzi, Scheduling Divisible Loads in Parallel and Distributed Systems, IEEE Computer Society Press, Los Alamitos, 1996.

[6] J. Blazewicz, M. Drabowski, J. Weglarz, Scheduling multiprocessor tasks to minimize schedule length, IEEE Transactions on Computers 35 (1986) 389–398.

[7] J. Blazewicz, M. Drozdowski, The performance limits of a two-dimensional network of load-sharing processors, Foundations of Computing and Decision Sciences 21 (1996) 3–15.

[8] J. Blazewicz, M. Drozdowski, Distributed processing of divisible jobs with communication startup costs, Discrete Applied Mathematics 76 (1997) 21–41.

[9] J. Blazewicz, M. Drozdowski, F. Guinand, D. Trystram, Scheduling under architectural constraints, Technical Report RA-003/95, Institute of Computing Science, Poznań University of Technology, 1995.

[10] S. H. Bokhari, Assignment Problems in Parallel and Distributed Computing, Kluwer Academic Publishers, Boston, 1987.

[11] Y.C. Cheng, T.G. Robertazzi, Distributed computation with communication delay, IEEE Transactions on Aerospace and Electronic Systems 24 (1988) 700–712.

[12] Y.C. Cheng, T.G. Robertazzi, Distributed computation for a tree network with communication delay, IEEE Transactions on Aerospace and Electronic Systems 26 (1990) 511–516.

[13] M. Drozdowski, Selected Problems of Scheduling Tasks in Multiprocessor Computer Systems, Poznań University of Technology Press, Poznań, 1997.

[14] R.L. Graham, D.E. Knuth, O. Patashnik, Concrete Mathematics, Addison-Wesley, Reading, 1989.

[15] D. Ghose, V. Mani, Distributed computation with communication delays: asymptotic performance analysis, Journal of Parallel and Distributed Computing 23 (1994) 293–305.

[16] J.L. Gustafson, Re-evaluating Amdahl's law, Communications of ACM 31 (1988) 532–533.

[17] V. Mani, D. Ghose, Distributed computation in a linear network: closed-form solutions, IEEE Transactions on Aerospace and Electronic Systems 30 (1994) 471–483.

[18] L.M. Ni, P.K. McKinley, A survey of wormhole routing techniques in direct networks, Computer 26 (1993) 62–76.

[19] J.G. Peters, M. Syska, Circuit-switched broadcasting in torus networks, IEEE Transactions on Parallel and Distributed Systems 7 (1996) 246–255.
[20] J.L. Park, H. Choi, Circuit switched broadcasting in torus mesh networks, IEEE Transactions on Parallel and Distributed Systems 7 (1996) 184–190.
[21] T.G. Robertazzi, Processor equivalence for a linear daisy chain of load sharing processors, IEEE Transactions on Aerospace and Electronic Systems 29 (1993) 1216–1221.