Bull. Nov. Comp. Center, Comp.Science, 14 (2001), 79–84
© 2001 NCC Publisher

Parallel algorithm for data array input into distributed computer system

Mikhail S. Tarkov, Alexey V. Tsarenko

A parallel algorithm is presented for the input of data arrays (partly, of images) into processors of a distributed computer system (CS). The algorithm minimizes the input time by means of effective routing of array fragments in the CS interconnection network. The algorithm modeling shows that it is optimum for the CS with topology of the E_2-graph (torus) and the optimum D_2-graph (circulant). The time of the array input is determined by the number N of the graph nodes and by the degree of the CS root node which contains the input array initially, and is not dependent upon the graph type (torus or circulant) for $N > 20$.

1. Introduction

A distributed computer system (DCS) is a set of elementary computers (EC), each EC consisting of a processor, memory and hardware for data input/output to channels connected to other ECs [1–5]. Application of the DCS in image processing is conditioned by advantages gained from the parallel processing (high performance, system scalability, robustness). In the case of the DCS usage for computations in real time environment, a problem of large data array transmissions between the input/output devices and every EC arises because of the absence of direct interconnections between them. Transit message exchanges are necessary with decrease the data input speed. For this reason, a problem of data input speed maximization is set. The problem can be solved by means of the optimum choice of data packet routes and the sequence of their passing to communication links.

2. Problem statement

Let DCS be given by a graph $G_s = (V_s, E_s)$, where V_s is a set of EC, E_s is a set of bidirected (duplex) communication links between the EC. Every component of E_s is considered to be edge of the graph G_s or a pair of oppositely directed arcs. A parallel program is given by a graph $G_p = (V_p, E_p)$, where V_p is a set of branches of the parallel program and E_p is a set of communications between branches, and, in addition, $|V_s| = |V_p| = N$. The program graph is mapped on the DCS graph so that every node of the graph G_p correspond to only one node of the graph G_s (numeration of nodes on the graph G_p gives the numeration of nodes on the graph G_s). We suggest that the parallel program have the following two properties:

1. Geometrical parallelism, i.e., decomposition of the program on parallel branches is determined by data distribution on the DCS processors.
2. All data fragments are equal in size.

By Property 1, the graph G_p determines the data distribution on the DCS processors. The node of the graph G_s containing initially all input data array will be called the root node. The data array (an image array, for example) is split to fragments each being transmitted for processing to the suitable EC. Let l_i be a path length of the i fragment transition from the root node to the goal node of the graph G_s, s_{ij} be the number of data fragments which are predecessors of the i fragment in the node j ($i \in \{1, \dots, N-1\}$, $j \in \{0, \dots, N-1\}$). We suggest that $|V_s| = N$ and the fragment with $i = 0$ remain in the root node. We also assume that the fragment transmissions to different arcs outgoing from the node can be implemented independently and simultaneously. Therefore the number of fragments passing to one arc does not influence on the number of predecessors of any other fragment passing to another arc.

If the time of the data fragment transmission between two nodes is considered as unit, then the time of the i-th fragment loading is equal to $t_i = l_i + \sum_{j=1}^{l_i} s_{ij}$. Since the time of the data array processing initiation is determined by the maximum of all the fragment loading times and all the fragment loadings are initiated simultaneously, then a minimum of the loading time is provided by minimization of the criterion

$$T = \max_i \Big(l_i + \sum_{j=1}^{l_i} s_{ij} \Big). \tag{1}$$

It is necessary to develop an algorithm providing the minimum time T. The time T minimization is a complex sorting problem. To simplify the problem we assume that all the fragments be transmitted across the shortest paths. Then the minimization of value (1) can be realized on the basis of the following notions:

1. Since in a common case there are several shortest paths between two nodes (partly because there are transit nodes with several outgoing arcs), then the minimization of the predecessor number can be reached by the even distribution of transmitted fragments onto outgoing arcs.
2. The longer is the shortest path of any fragment loading, the smaller the number of predecessors a fragment has. Therefore the fragments transmitted through a certain arc must be ordered by decreasing the length of suitable shortest paths from a given node to a target to provide minimum of criterion (1).

Hence follows the strategy to construct the plan of data loading from the root node:

1. Construct an oriented graph $G(\mathrm{SP})$ on the DCS graph assigning a set of shortest paths from the root node to all other graph nodes.
2. Assign a correspondence between each orgraph arc and a set of target nodes, to which the arcs belong, to a suitable set of the shortest paths (the initial plan of loading).
3. For any node i having outgoing arcs on the constructed orgraph $G(\mathrm{SP})$, create a list OUT_p of transmitted fragments for any outgoing arc $p \in \{0, \ldots, v_i^{\mathrm{out}}\}$ (v_i^{out} is the number of outgoing arcs of the node i) so that all these arcs are loaded evenly, and fragments are ordered in the list by a decrease in distance between the node i and the target node (the final plan of loading).

The final plan determines the implementation of the data loading algorithm in each node: any data fragment coming to the node either stays in it for processing (if the node is the goal for the fragment, i.e., the fragment number is equal to the node number) or transmitted to the neighboring node according to the plan determined by the list OUT_p containing the fragment number.

All the steps of constructing the plan and the data loading are realized by a parallel algorithm implemented by all elementary computers of the DCS.

3. Construction of the shortest path orgraph

The root node $i = 0$ sends to all outgoing arcs a message SP (the shortest path), containing the mark $d(\mathrm{SP}) = 1$.

Any non-root node $i \neq 0$ having received the message SP for the first time:

1) marks incoming arc $p \in \{0, \ldots, v_i - 1\}$ with $d_{ip} = d(\mathrm{SP})$, this mark being equal to the distance from the node i to the root node (v_i is the node degree);
2) increments $d(\mathrm{SP})$ by 1;
3) transmits the message SP to all outgoing arcs including an arc opposite to the incoming arc p the message was received from.

Any outgoing arc is marked only if the mark $d(\mathrm{SP})$ of the message transmitted to a given arc is less than the mark of the message received from the suitable incoming arc. It is easy to verify that one and only one message SP will be transmitted to each arc of the graph and the resulted marking of outgoing arcs determines the orgraph $G(\mathrm{SP})$ of the shortest paths from the root node to all other nodes.

4. Primary plan construction

Construction of the primary plan for data array loading is based on the usage of the orgraph $G(\mathrm{PP}) = G^{-1}(\mathrm{SP})$, which is obtained from the orgraph $G(\mathrm{SP})$ by inversion its arc directions. The construction process of the primary data loading plan is initiated by the orgraph nodes that have no outgoing arcs on $G(\mathrm{SP})$. Every node sends to its every outgoing arc on $G(\mathrm{PP})$ the message PP (the primary plan) containing: 1) the node i number; 2) the value $d(\mathrm{PP}) = 1$ giving the distance from the node i to the neighboring node where the message PP is transmitted to.

Any node i having outgoing arcs on the graph $G(\mathrm{PP})$ after receipt of the message PP includes into constructed message R_i the number j of the message R_j source and the distance d_j from the source of R_j to the node i. After receipt of messages R_j from all incoming arcs on the graph $G(\mathrm{PP})$, the node i: 1) includes into the message R_i its number i and the distance $d_i = 1$; 2) sends the message R_i to outgoing arcs of the orgraph $G(\mathrm{PP})$. In the root node of the orgraph $G(\mathrm{SP})$, the described process terminates because this node has no outgoing arcs on the orgraph $G(\mathrm{PP})$.

Any input arc $p \in \{0, \ldots, v_i^{\mathrm{in}} - 1\}$ (v_i^{in} is an incoming semidegree) of the orgraph $G(\mathrm{PP})$, node i corresponding to the set IN_{ip} of the pairs (j, d_j), where j is a source node number and d_j is the distance to the source. In the set IN_{ip}, numbers j of source nodes are ordered by a decrease of the value d_j. A set of the sets IN_{ip} forms the primary plan of allowed directions for transmitting fragments. We suggest that a data fragment be transmitted from the node i to the arc $p \in \{0, \ldots, v_i^{\mathrm{in}} - 1\}$ if the number of its destination belongs to the set IN_{ip}.

5. Final plan construction

Construction of the final plan of the data array loading is initiated by the root node $i = 0$ of the orgraph $G(\mathrm{SP})$. Any outgoing arc p of the root node corresponds to the set OUT_p of numbers of data fragments transmitted to the arc $p \in \{0, \ldots, v_i^{\mathrm{in}} - 1\}$. A number j of the goal node is included into the set OUT_p of the arc p which corresponds to the minimum of the criterion $l_{jp} + s_{jp}$ where l_{jp} is the distance to the goal node for transmitting the fragment j to the arc p. Here s_{jp} is the number of predecessors of the fragment j in the ordered set OUT_p (data fragments in the set OUT_p are ordered by a decrease in the distance l_{jp} to the goal).

After completion of the construction of sets OUT_p to all outgoing arcs the messages FP (the final plan) are transmitted. Each message FP contains the set OUT_p of numbers of the goal nodes, which correspond to the arc p.

In the non-root node $i \neq 0$, the final plan construction is implemented in the same way as in the root one, but the construction begins only after receipt of the messages FP from all the incoming arcs of the node in the orgraph $G(\mathrm{SP})$.

6. Modeling of data array input to distributed computer system

The above parallel algorithm for the input of data arrays has been simulated for the distributed CS with the two-measured toroidal graphs (E_2) and the optimal two-measured circulant graphs (D_2) [5]. The simulation shows that for such a DCS, the time of the system loading is equal to $T = \frac{N}{4} \pm 1$, where $N = |V|$. For $N > 20$, the time T does not depend upon the graph type, i.e., the time is determined by the number of the graph nodes and by the degree $v_0 = 4$ of the root node.

The minimum time $T_{\min} \leq T$ of the data input cannot be less than the time T_q of the previous data fragment transmission from the root node to the neighboring one. If in the root node the data fragments are distributed approximately evenly among the sets OUT_p (i.e., the difference of the set OUT_p cardinalities is no more than 1), then $T_q = T$. Hence $T_{\min} \geq T_q$, it follows that

$$T = T_{\min} = \frac{N}{4} \pm 1, \tag{2}$$

i.e., for the above graphs the proposed algorithm of the data array input is optimal.

The time (2) is determined by the time T_q because the data fragments in the sets OUT_p are ordered by a decrease in distance to the goal node. As a result, the time of the data fragment transmission to the goal node which is most distant from the root coincides with T_q.

The further development of the presented algorithm suggests a consideration of several input nodes in the DCS graph. It is assumed to realize an optimum partitioning of the DCS graph into connected subgraphs [4] and to use own input (root) node for each subgraph (subsystem). In addition, it is interesting to study the influence of the DCS component (processing units and interconnections) failures on the organization of the data array input.

References

[1] Korneev V.V. Parallel Computing Systems. – Moscow: Znanie, 1999 (in Russian).

[2] Tarkov M.S. Parallel fault-tolerant image processing on transputer system MICROS-T // Nauchnoe priborostroenie. – 1995. – Vol. 5, № 3–4. – P. 74–79 (in Russian).

[3] Tarkov M.S. Mapping parallel programs onto distributed robust computer systems // Proc. of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics / Ed. Achim Sydow. – Berlin, August 1997. – Vol. 6: Applications in Modelling and Simulation. – P. 365–370.

[4] Tarkov M.S. Parallel algorithm for structure bisection of a distributed computer system // Proc. of the 16th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation / Eds. Michel Devil and Robert Owens. – Lausanne, Switzerland, August 21–25, 2000. – Parallel Computing Session, 217-6 (CD-ROM, 217-6.pdf).

[5] Monakhova E.A. Synthesys of optimum circulant networks for computer systems // Parallel Algorithms and Structures. – Novosibirsk, 1991. – P. 20–29 (in Russian).