Parcella '90

Proceedings of the V. International Workshop on Parallel Processing by Cellular Automata and Arrays
held in Berlin, September 17 - 21, 1990

edited by

Gottfried Wolf
Tamás Legendi
Udo Schendel

Akademie-Verlag Berlin

An approach of automatic scheduling for the ZKI-Multiprocessor System

A.Jaekel, O.Taraszow

Central Institute of Cybernetics and Information Processes (ZKI), GDR academy of Sciences, Berlin

1.Introduction

The multiprocessor system MPS was developed at the ZKI. It has been described in /Ber90/. It consists of a control computer and signal processors as processing elements. There exists different possibilities to program the MPS.
In this paper an approach for semiautomatic parallelization of sequential programs for the MPS is presented. The kernel of parallelization is a multiprocessor scheduler. The criterion for parallelization is to minimize the total processing time of the program with respect to the cost of data transport.
One can find a wide range of methods for automated program mapping among others in /Sal88/.
Fig.1. shows the principal paradigma of our investigations:
First a dataflow analyzer generates the dataflow graph (DFG) of the sequential program. Then the program is partitioned into tasks corresponding to the dataflow and resulting data dependencies. Next the DFG has to be weighted. An evaluator assesses expected running time of tasks and data transport requirements. Last the target program package is generated by the multiprocessor scheduler.

2.Multiprocessor model

The model **Q** of the MPS is a wheel-like model. The control computer is located in the center of the wheel. The processing elements are configured around the control computer like a ring. Each processor is coupled to its neighbor by a local communication memory. For each processing element exists local resources. They consist of a data memory, a program memory and the communication memory. They are controlled by the control computer. The MPS is symmetrically,homogenously and scalable. See fig.2.
The tasks are processed on a processor in time slices. During a time slice all processing elements work on their assigned task and wait until the last processor has finished its work (barrier synchronization).

A processor can accept two states: active or non-active. If a processor is active, it can process either a task or a communication routine to provide data transport. All processing elements are able to work in SPMD-mode (single program on multiple data) or MPMD-mode (multiple program on multiple data). To start a task on a processing element, the local environment of that processor must contain all data needed for the execution of the task. The processing is non-preemptive.
Following restrictions are relevant for the MPS:

- capacities of control computer memory,
  program memory,
  data memory
  and communication memory,
- overhead time for the control computer,
- necessary time for data transport between local meories per memory unit,
- necessary time for data transport between control computer and local memories per memory unit.

3.The data flow graph

The dataflow graph is a finite, acyclic, weighted digraph $G=(V,E)$. $V \subset N$, $E \subseteq V^2$.
The set of nodes V represent tasks and the set of edges E represent the dependencies between tasks. The DFG contains one start node and one end node. They are considered as start routine and final routine of the control computer.
Following functions are given to assign weights to the DFG:

$g: V \dashrightarrow R$, memory requirement of a task (node weight),

$f: E \dashrightarrow R$, amount of data to be transported from start node to end node of an edge (edge weight),

$l: V \dashrightarrow R$, expected runtime of tasks (node weight).

4.The multiprocessor scheduler

Suppose that the DFG is given.
We construct a static, non-preemptive schedule with three parts: The first one is the decomposition of the DFG, the second one consists of the transformation of the graph into a MPS-like structure and the last part realizes the allocation of tasks onto processors.
Let Φ be a mapping

$\Phi: G \dashrightarrow Q$ with

$\Phi = (\alpha,\beta)$, $\alpha: V(G) \dashrightarrow V(Q)$ and
$\beta: E(G) \dashrightarrow \Omega(Q)$.

$\Omega(Q)$ is the set of all pathes in Q.

Then is

$\Phi = \tau \bullet \mu \bullet \sigma$, with τ representing the allocation,
μ representing the transformation,
σ representing the decomposition.

4.1. The decomposition

Given the graph **G=(V,E)** and the functions **f** and **g**.

Def.1. A node weight constraint is a real number $c \in R$ with

$$\sum_{v \in V} g(v) > c \quad \text{and} \quad g(v) \le c \text{ for all } v \in V.$$

Def.2. A family $D=\{d\}$ of non-empty subsets of **V** are said to be a decomposition of V if and only if

a) $\bigcup_{d \in D} d = V$

b) for all $d, d' \in D$: $d \neq d' \Longrightarrow d \cap d' = \emptyset$.

Def.3. A decomposition is said to be admissible if and only if for all $d \in D$:

$$\sum_{v \in V} g(v) \le c.$$

Assume that $G'=(D,E')$ is a graph and **D** is a decomposition of **V**. For any $d, d' \in D$:

$$(d,d') \in E' \; :\Longleftrightarrow \; (\exists v \in d, \exists v' \in d' \mid (v,v') \in E).$$

G′ is the clustergraph of G. The nodes of G′ are said to be clusters.

Def.4. The weight w(d) of a cluster **d** is the sum of weights of all inner edges of **d**.
The weight of the decomposition D is the function

$$z(D) = \sum_{d \in D} w(d).$$

Def.5. An optimal decomposition is an admissible decomposition with a maximal value **z(D)**.

The problem of the first part of the scheduler is to locking for a mapping σ from **G** onto G′ so that **D** becomes an optimal decomposition of **V(G)**.

We can give an interpretation of the formal task:
The criterion for optimization is to minimize data transport between clusters of DFG-tasks. It is possible to process all nodes of a cluster with small communication rate to its outside world in one processor. The decomposition concentrates tasks with high communication rate inside the clusters. The communication cost are zero in a cluster.
The condition for an admissible decomposition is a constraint equivalent to the possible cluster size. It is the sum of program memory demand of tasks belonging to the cluster. The sum is restricted by the program memory capacity of a processor.
We solved the problem of decomposition with an adapted branch-and-bound algorithm given by H.Widjaja /Wid82/.

4.2.Transformation

Each transformation **μ** corresponds to a specific multiprocessor model and vice versa.
Assume that
a) the number of clusters is less than the number of processors,
b) data transport will not be realized by the control computer due to its high communication cost.
Now the cluster graph **G´** is transformed into a MPS-like structure.
First we construct a circle containing the maximum possible number of resulting clusters; (It may be a Hamiltonian circle): Therefore we have to find a basis circle. Then all suitable clusters are inserted in the circle. A suitable cluster outside the circle is either adjacent to exact any two neighboring clusters of the circle or there exists a path between any two circle clusters containing this cluster.
In a second step we insert all remaining clusters into the circle:
Let there be a cluster **d** outside the current circle. **d** is adjoined to non-neighboring circle cluster. **d** changes to a circle cluster regard to small total data transport cost of the current circle. Copy nodes, representing data transport, will be inserted to all circle clusters placed between **d** and it´s neighbors in the circle. The copy nodes will be connected in the correct order by edges. The original edges will be deleted. Their weights are assigned to the corresponding edges between copy nodes.
The third step provides the insertion of all remaining, non-processed edges of circle clusters considering small cost for data transport too.

4.3. Allocation

The arising circle of 4.2. is assigned to the processor ring, taken into account the utilization of the MPS and the requirement of small data transport cost.
Obviously its either necessary to insert additional clusters consisting of copy nodes. They are allocated to available processors without tasks assigned to them. Or its necessary to split some clusters to utilize more or all processing elements. We have to find a good compromise between the two aspects mentioned above. If the number of clusters is equal to the number of processors the allocation is trivial.
If every cluster is assigned to a processor, we determine the order of task processing of any cluster. The method of searching nodes without predecessors in the DFG expanding by copy nodes is the most promising approach.

References

/Ber90/ B.Berlin, et. al., "A Signal-processor-based Multiprocessor System with Partial Shared Memory", J. New Gener. Comput.Syst. 3 (1990) , 3-19

/Sal88/ J.Saltz, "Methods for automated program mapping", In Numerical algorithms for modern parallel computer architectures, Springer-Verlag, Berlin, Heidelberg, N.Y., Tokyo, 1988, pp. 173-195

/Wid82/ H.Widjaja, "An effective structured approach to finding optimal partitions of networks", Computing 29 (1982), 241-262

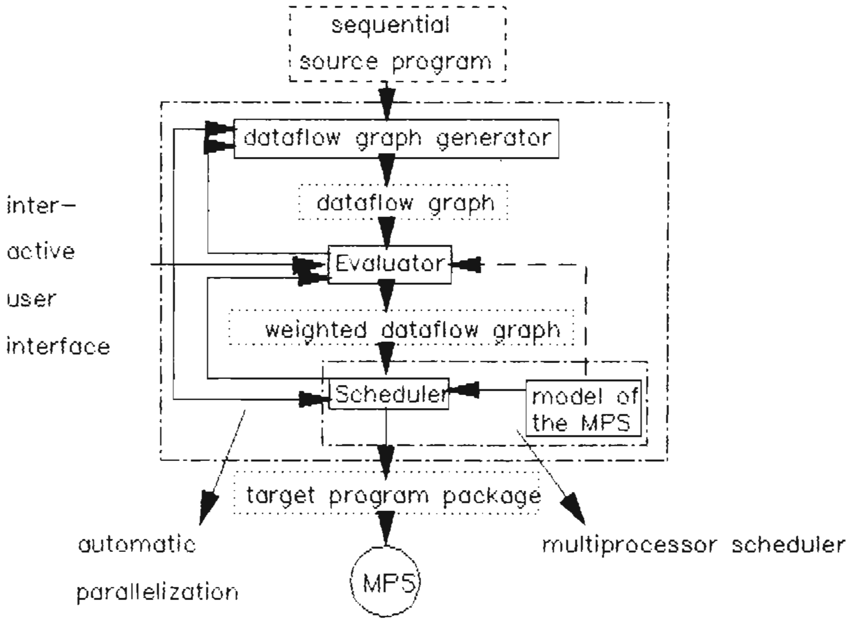

fig.1. process of automatic parallelization

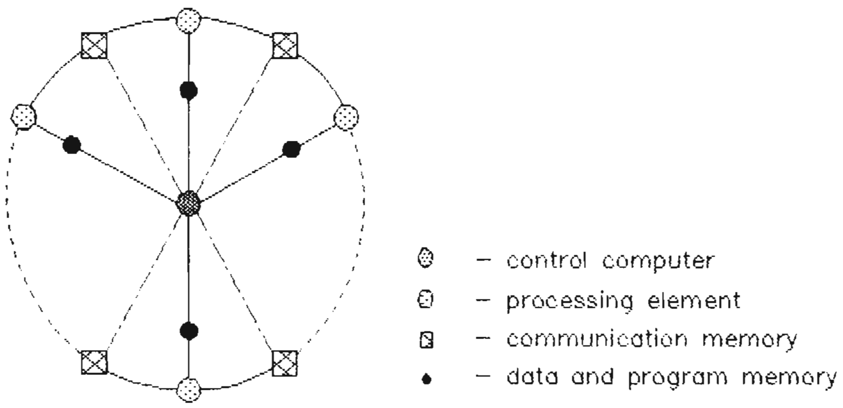

fig.2. model of the MPS