

# Scheduling Strategies for Master-Slave Tasking on Heterogeneous Processor Platforms

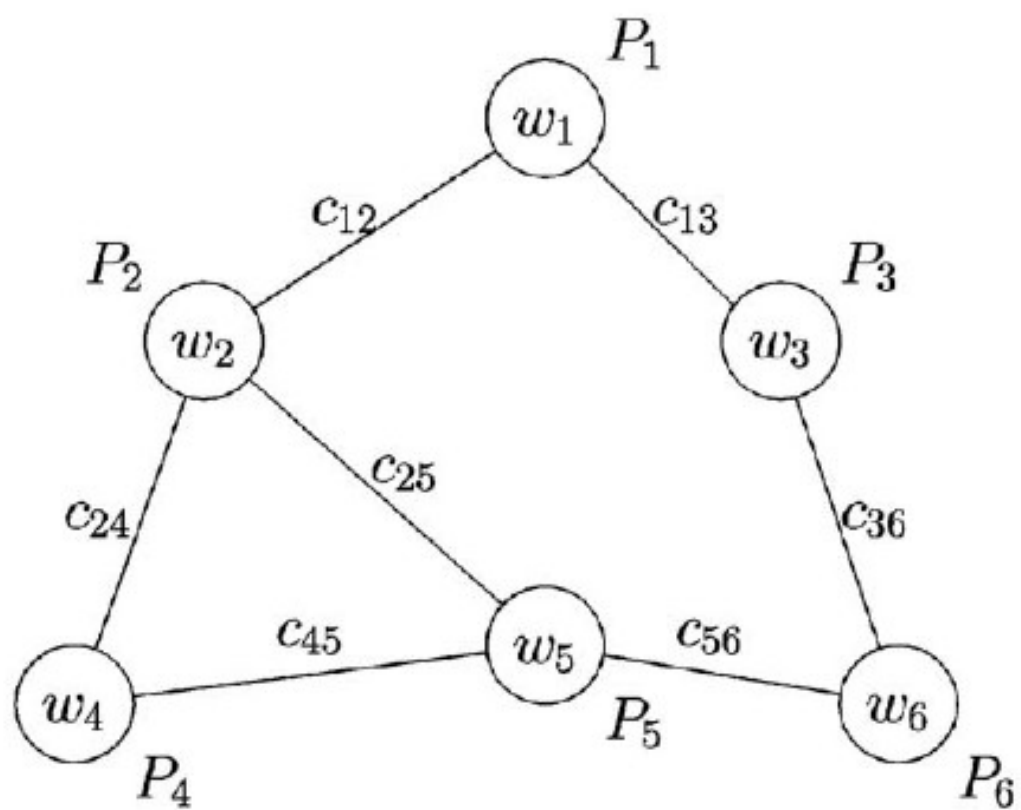
Cyril Banino, Olivier Beaumont, Larry Carter, Fellow, IEEE, Jeanne Ferrante, Senior Member, IEEE, Arnaud Legrand, and Yves Robert, Member, IEEE

IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS,  
VOL. 15, NO. 4, APRIL 2004

- master-slave paradigm on a heterogeneous platform
- resources have different speeds of computation and communication
- large number of independent, equal-sized tasks

# Optimal Steady State

- For each processor, determine the fraction of time spent computing and the fraction of time spent sending or receiving tasks along each communication link, so that the (averaged) overall number of tasks processed at each time-step is maximum.



# Architectural Model

- Node-weighted edge-weighted graph  $G(V, E, w, c)$
- $p = |V|$ , number of nodes
- Each  $P_i$  in  $V$  represents a computing resource of weight  $w_i$ , meaning that node  $P_i$  requires  $w_i$  units of time to process a task
- Each edge  $e_{ij} : P_i \rightarrow P_j$  is labeled by a value  $c_{ij}$  which represents the time needed to communicate one (data file associated with a) task between  $P_i$  and  $P_j$

- A master processor node  $P_m$ , initially holds the data for a large collection of independent tasks to be executed.
- Each task as being associated with a file that contains all the data required for the execution of the task.
- Tasks are atomic; their computation or communication cannot be preempted.

# Communication Model

- Full overlap, single-port
- A processor node can simultaneously receive data from one of its neighbors, perform some (independent) computation, and send data to one of its neighbors.
- At any given time-step, there are at most two communications involving a given processor, one sent and the other received.

# Results

- General Graph, asymptotically optimal, using linear programming
- Tree, optimal in linear time, using a bottom-up algorithm bandwidth-centric scheduling

# Optimal Steady State For General Platform Graph

- $\alpha_i$  is the fraction of time spent by  $P_i$  computing.
- $s_{ij}$  is the fraction of time spent by  $P_i$  sending tasks to each neighbor processor  $P_j$ , for  $j \in n(i)$ , i.e., for each  $e_{ij} \in E$ .
- $r_{ij}$  is the fraction of time spent by  $P_i$  receiving tasks from each neighbor processor  $P_j$ , for  $j \in n(i)$ , i.e., for each  $e_{ij} \in E$ .

$$\forall i, 0 \leq \alpha_i \leq 1, \quad (1)$$

$$\forall i, \forall j \in n(i), 0 \leq s_{ij} \leq 1, \quad (2)$$

$$\forall i, \forall j \in n(i), 0 \leq r_{ij} \leq 1. \quad (3)$$

**One-port model for outgoing communications.** Because send operations to the neighbors of  $P_i$  are assumed to be sequential, we have the equation

$$\forall i, \sum_{j \in n(i)} s_{ij} \leq 1. \quad (5)$$

**One-port model for incoming communications.** Because receive operations from the neighbors of  $P_i$  are assumed to be sequential, we have the equation

$$\forall i, \sum_{j \in n(i)} r_{ij} \leq 1. \quad (6)$$

**Full overlap.** Because of the full overlap hypothesis, there is no further constraint on  $\alpha_i$ :  $0 \leq \alpha_i \leq 1$  and  $\alpha_i = 1$  would mean that  $P_i$  is kept busy processing tasks all the time.

**Limited bandwidth.** This constraint is due to our hypothesis that the same link  $e_{ij}$  may be used in both directions simultaneously. We have to guarantee that the link bandwidth is not exceeded. The constraint translates into:

$$\forall e_{ij} \in E, s_{ij} + r_{ij} \leq 1. \quad (7)$$

We can slightly reformulate (7) by introducing  $b_{ij}$ , the link bandwidth, expressed in tasks per second (i.e.,  $b_{ij} = 1/c_{ij}$ ). In each time unit, there are  $\frac{s_{ij}}{c_{ij}}$  tasks sent by  $P_i$  to  $P_j$ , and  $\frac{r_{ij}}{c_{ij}}$  tasks received by  $P_i$  to  $P_j$ , so constraint (7) can be written

$$\forall e_{ij} \in E, \frac{s_{ij}}{c_{ij}} + \frac{r_{ij}}{c_{ij}} \leq b_{ij}.$$

**Conservation laws.** The last constraints deal with *conservation laws*: For every processor  $P_i$  which is not the master, the number of tasks received by  $P_i$ , i.e.,  $\sum_{j \in n(i)} \frac{r_{ij}}{c_{ij}}$ , should be equal to the number of tasks that  $P_i$  consumes itself, i.e.,  $\frac{\alpha_i}{w_i}$ , plus the number of tasks forwarded to its neighbors, i.e.,  $\sum_{j \in n(i)} \frac{s_{ij}}{c_{ij}}$ . We derive the equation:

$$\forall i \neq m, \sum_{j \in n(i)} \frac{r_{ij}}{c_{ij}} = \frac{\alpha_i}{w_i} + \sum_{j \in n(i)} \frac{s_{ij}}{c_{ij}}. \quad (8)$$

MASTER SLAVE SCHEDULING PROBLEM MSSG( $G$ )

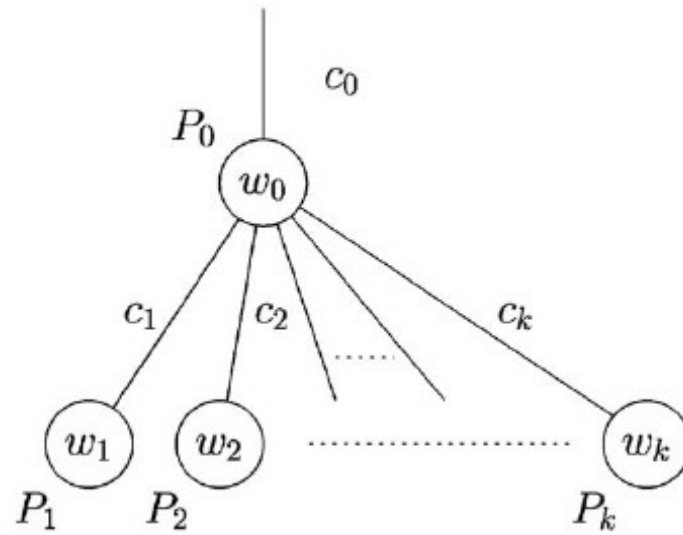
$$\text{Maximize } n_{\text{task}}(G) = \sum_{i=1}^p \frac{\alpha_i}{w_i},$$

subject to

$$\left\{ \begin{array}{ll} \forall i, & 0 \leq \alpha_i \leq 1 \\ \forall i, \forall j \in n(i), & 0 \leq s_{ij} \leq 1 \\ \forall i, \forall j \in n(i), & 0 \leq r_{ij} \leq 1 \\ \forall e_{ij} \in E, & s_{ij} = r_{ji} \end{array} \right. \left| \begin{array}{ll} \forall i, & \sum_{j \in n(i)} s_{ij} \leq 1 \\ \forall i, & \sum_{j \in n(i)} r_{ij} \leq 1 \\ \forall e_{ij} \in E, & s_{ij} + r_{ij} \leq 1 \\ \forall i \neq m, & \sum_{j \in n(i)} \frac{r_{ij}}{c_{ij}} = \frac{\alpha_i}{w_i} + \sum_{j \in n(i)} \frac{s_{ij}}{c_{ij}} \\ \forall j \in n(m), & r_{mj} = 0 \end{array} \right.$$

# Bandwidth-centric Strategy for Tree Shaped Platforms

# Star Graph



1. Sort the children by increasing communication times. Renumber them so that  $c_1 \leq c_2 \dots \leq c_k$ .
2. Let  $p$  be the largest index so that  $\sum_{i=1}^p \frac{c_i}{w_i} \leq 1$ . If  $p < k$ , let  $\varepsilon = 1 - \sum_{i=1}^p \frac{c_i}{w_i}$ ; otherwise, let  $\varepsilon = 0$ .
3. Then,  $n_{\text{task}}(F) = \min\left(\frac{1}{c_0}, \frac{1}{w_0} + \sum_{i=1}^p \frac{1}{w_i} + \frac{\varepsilon}{c_{p+1}}\right)$ .

# Arbitrary Tree Graphs

1. Consider any subtree  $F$  consisting of several leaves and their parent. Replace this tree with a single node whose weight is  $w = \frac{1}{n_{\text{task}}(F)}$ , where  $n_{\text{task}}(F)$  is given by Proposition 1.
2. Iterate the process until there remains a single node.  
Then, the minimal value is equal to the weight of the single node.

