Complexity of Master-slave Tasking on Heterogeneous Trees

Pierre-François Dutot

Complexity of Master-slave Tasking on Heterogeneous Trees

Pierre-François Dutot
ID-IMAG, 51 avenue Jean Kuntzmann
38330 Montbonnot Saint Martin, France
Pierre-Francois.Dutot@imag.fr

November 14, 2003

Abstract

In this paper, we consider the problem of scheduling independent identical tasks on heterogeneous processors and network, where processing times and communications times are different. We assume that communication-computation overlap is possible for every processor, but only allow one send and one receive at a time. In this model, we prove that scheduling on a tree network is NP-hard in the strong sense, reducing the problem to the well-known 3-partition problem.

1 Introduction

Parallel computation on heterogeneous platform is one of the most important issue in high performance computing nowadays. Famous parallel applications such as SETI@home [9] or the Mersenne prime search [8] are using a wide variety of commodity computing resources to extend as much as possible the pool of volunteers they depend on.

In this paper we deal with the problem of scheduling independent unit execution time (UET) tasks, as used in the previously cited applications, on a sub-class of "grid" computing platform where communication links and computation nodes may be of any kind, and therefore have different speeds.

This work is also related to divisible tasks as first introduced by [5]. Robertazzi et al studied many variations around this topic for divisible tasks. In [1] they first studied the homogeneous tree problem. Then in [10] they looked at the bus problem which is identical to a fork graph with homogeneous communications and heterogeneous computations times. They also recently worked [4] on star graph with heterogeneous communications and computations times. The main difference is that we are working with quanta of workload whereas a divisible task can be split in fractions of any size.

This problem has already been adressed in [2] for the special case of fork graphs. The steady state for trees is also studied in the same paper. Some of the authors of [2] also wrote a research report [3] with a good bibliography covering many similar problems. In some previous work I also adressed other subclasses of trees, such as chains or spiders giving polynomial algorithms [6].

Up to now, the limit between hard problems and polynomial problems has only been looked at from the side of the polynomial problems. This paper is the result of an attempt to look at the limit from the other side. We prove that the problem of scheduling independent identical tasks in the one-port master-slave paradigm is NP-hard in the strong sense on trees.

In the next section we present the model and the definitions used in the proofs. Section 3 recalls the well known 3-Partition problem used in the reduction. The reduction itself is presented in section 4. Some possible future work on this problem is given in the conclusion.

2 Modeling of the platform

The problem studied here is the scheduling of identical independent tasks on a heterogeneous platform where the network and the processors have different speeds. The interconnection graph of the processors here will be a tree. Therefore each processors may have one or more descendant, but only one parent.

The one-port assumption forbids any processors to send two tasks at the same time (even on different communication links) or to receive two tasks at the same time. However a processor can at the same time receive one task while sending another one to one of its children.

The example provided in figure 1 shows how a schedule can be written. In this example, four tasks are scheduled on a simple tree made of three processors.

The tree is shown on the left of the figure. The numbers in the circles are the computation times associated with the nodes. It is the time it takes for a task to be executed on the node. The numbers on the edges are the communication times. They represent the time needed to send one task using the labeled link. The master node is the node without number as it cannot compute.

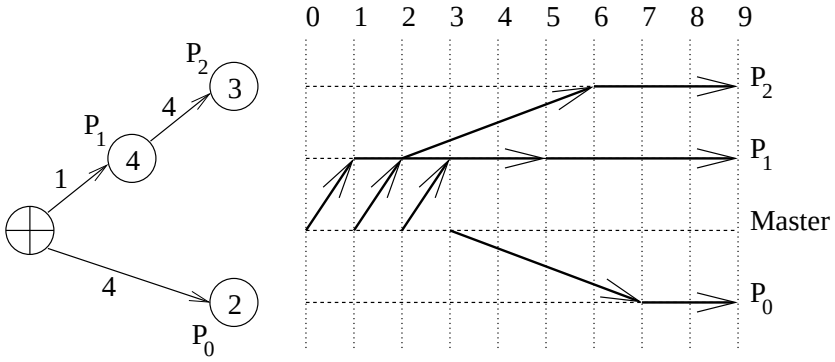

Figure 1: A tree (left) and a schedule (right)

On the diagram, vertical dotted lines are time units, horizontal dashed lines are related to nodes. The horizontal arrows are the execution of the tasks, and the oblique ones are the communications. This kind of settings will be used in the diagrams describing the scheduling in the following sections. Representing

arbitrary trees with such diagrams can be painful, hopefully the trees that will be used in the reduction have readable diagrams.

In this example, one can see that the "one-port" constraint is respected since each communication leaving the master node is not started before the previous one ended.

Between time instant 2 and 3 one can see that processor P_1 receives a task, computes another task and sends a third one. We can also see that the computation of the second task on processor P_1 has been delayed to allow for the completion of the first task.

In this model, it is usually assumed that every node has an unlimited buffer capacity. However the proof is still valid if all the nodes have a buffer of size one.

3 The problems

3.1 3-Partition

The proof in this paper is based on a reduction from the well known problem of 3-partition [7]. In this section we will briefly recall the 3-partition problem and present the modified version that will be used in the rest of the paper.

Definition 1 *3-partition*
Let S and n be integers, and let $(y_i)_{i\in 1..3n}$ be $3n$ integers such that $\sum_{i=1}^{3n} y_i = nS$ and for all i, $\frac{S}{4} < y_i < \frac{S}{2}$.

The question is "Can the set of y_i be partitioned into m disjoint sets S_1, S_2, ..., S_m such that for $1 \leq i \leq m$, $\sum_{y\in S_i} y = S$?".

Theorem 1 *The 3-partition problem is NP-complete in the strong sense.*

In all the following we will use the numbers $x_i = \frac{1}{4}S + \frac{y_i}{8}$ where the $(y_i)_{i\in 1..3n}$ are from an instance of the 3-partition problem. One can easily prove that a solution of the search problem associated with the 3-partition problem for the $(y_i)_{i\in 1..3n}$ yields a partition of the $(x_i)_{i\in 1..3n}$ where the sum of every triplets is exactly $\frac{7}{8}S$. And conversely a partition of the $(x_i)_{i\in 1..3n}$ in triplets of sum $\frac{7}{8}S$ gives a solution to the 3-partition search problem.

The reason behind this transformation is that now for all i we have $\frac{9}{32}S < x_i < \frac{5}{16}S$. This property will be very useful in the reduction.

3.2 MSTHT

Here is the formal presentation of the Master-Slave Tasking on Heterogeneous Trees (MSTHT) problem.

Definition 2 *MSTHT*
Let T=(V,E) be a tree. Let v_0 in V be a special vertex called "Master node". For all the other vertices v_i in V, let w_i be the computation cost. For all edges e_i in E, let c_i be the communication cost. Finally let n be a number of tasks and D be a deadline.

The decision problem MSTHT answers the question "Is it possible to schedule n tasks before the deadline D ?".

4 Reduction

The tree used for this reduction is presented in figure 2. The MSTHT problem is to schedule $4n$ tasks with a makespan smaller or equal to $D = E+nS+\frac{S}{4}$, where E stands for "enormous", typically something greater or equal to $(n+1)S$. The master node of the tree is linked to a distribution node by a link which sends a task in $\frac{S}{4}$ units of time. The distribution node is too slow to compute ($2E > D$) and therefore has to distribute the tasks either to one of the p_i processors with a communication time x_i and a computation time E or to one of the processors Q_i with a communication time $\frac{S}{8}$ and processing time $E + iS$. Another interesting property is that no processor can compute two tasks sequentially as all the computation times are greater or equal to E (and again as $2E > D$).

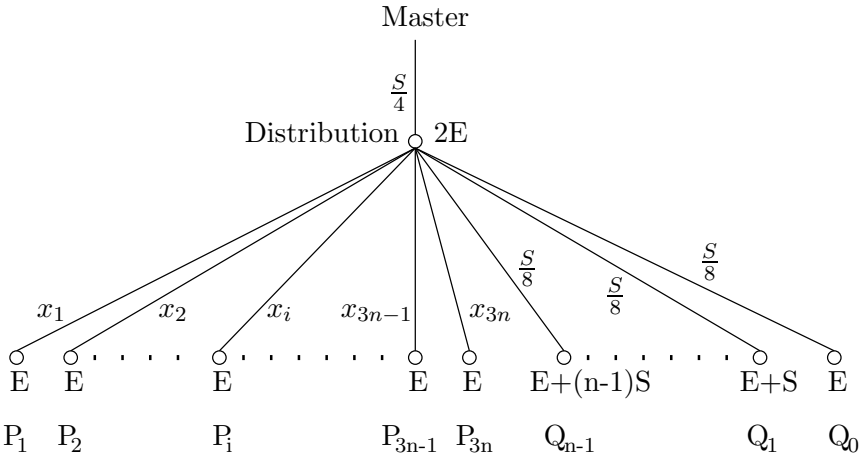

Figure 2: Tree used in the reduction

Here is an introductory explanation of the motivations behind this structure. To use the 3-partition problem in scheduling, usually one needs to create (identical) slots of a resource to be filled with three independent tasks. As the tasks are identical here, the $3n$ different numbers cannot be replaced by tasks with different execution time. However, the time resource is shared by the different communication links. The goal of the reduction was therefore to create a structure where several time slots of the same size were shared among the possible communications. The communication links to the processors Q_i are designed to be the hedges between the time slots.

4.1 From 3-partition to a schedule

Let us first show that if we have a solution to the 3-partition problem, we have a valid scheduling of $4n$ tasks within $\frac{S}{4} + n * S + E$ time units.

For convenience, the x_i will be ordered as in the 3-partition solution (i.e. $x_{3j+1} + x_{3j+2} + x_{3j+3} = \frac{7}{8}S$ for $j \in 0..(n-1)$). Here is how the scheduling is made :

1. Fully load the first link from the master. Sending one task every $\frac{S}{4}$ units of time, the $4n$ tasks are sent in nS units of time.

2. From the distribution node send in order of arrival as soon as possible :

   - Task $4j+1$ on the link x_{3j+1} for $j \in 0..(n-1)$
   - Task $4j+2$ on the link x_{3j+2} for $j \in 0..(n-1)$
   - Task $4j+3$ on the link x_{3j+3} for $j \in 0..(n-1)$
   - Task $4j+4$ on the link $\frac{S}{8}$ going to processor Q_{n-j} of execution time $E+(n-j)S$ for $j \in 0..(n-1)$

The corresponding diagram is given in figure 3 for the example case where $n=2$, $S=15$, $(x_1,x_2,x_3,x_4,x_5,x_6)=(4,4,7,4,5,6)$

As we supposed that each triplet $(x_{3j+1},x_{3j+2},x_{3j+3})$ is exactly of sum $\frac{7}{8}S$, the total communication cost from the distribution node for the tasks $4j+1$ to $4j+4$ is exactly S. Therefore if there is no idle time on the communication link from the distribution node, all $4n$ tasks can be emitted before time $\frac{S}{4}+n*S$.

As $x_i=\frac{S}{4}+\frac{y_i}{8}$ at the end of all the x_i communication, the following task is ready to be sent. The communications going to the processors Q_i end exactly when the following task is ready. Therefore the scheduling described earlier has no idle time on the communication link leaving from the distribution node.

Every processor Q_i receives its task at time $\frac{S}{4}+(n-i)*S$ and starting its computation immediately, finishes just in time.

4.2 From a schedule to 3-Partition

In this section we will show that any schedule of $4n$ tasks within $\frac{S}{4}+nS+E$ units of time is related to a 3-partition and has an execution diagram similar to figure 3.

As stated earlier, the computation times of all the processors only allow one task to be done on each processor (since $2E>D$). So all processors have to be used. Moreover, the communication cost below the distribution node imply that there is no idle time between the $\frac{S}{4}$ time instant and the $\frac{S}{4}+nS$ time instant, as the sum of all the values of the links is exactly nS, the smallest computation time is E and the first task is only available after going through the link $\frac{S}{4}$ from the master.

Let us now focus on the processor Q_{n-1} and prove some properties of the task assigned to this processor.

Lemma 1 *The task assigned to processor Q_{n-1} is one of first four task sent by the master.*

As the computation time of Q_{n-1} is $E+(n-1)S$, if a task is scheduled on this processor, it has to arrive at most at time $\frac{5}{4}S$ on the processor. The ith task cannot arrive before the time instant $i\frac{S}{4}+\frac{S}{8}$, hence the task scheduled on Q_{n-1} has to be one of the first four tasks. □

Lemma 2 *The first three tasks are alloted to three of the P_i processors.*

As stated before, the communication link going out of the distribution node has to be fully occupied in order to schedule $4n$ tasks within the given time bound. We first show that this is possible with the first three tasks alloted to three of the P_i processors. The communication costs x_i are all greater than $\frac{9}{32}S$.

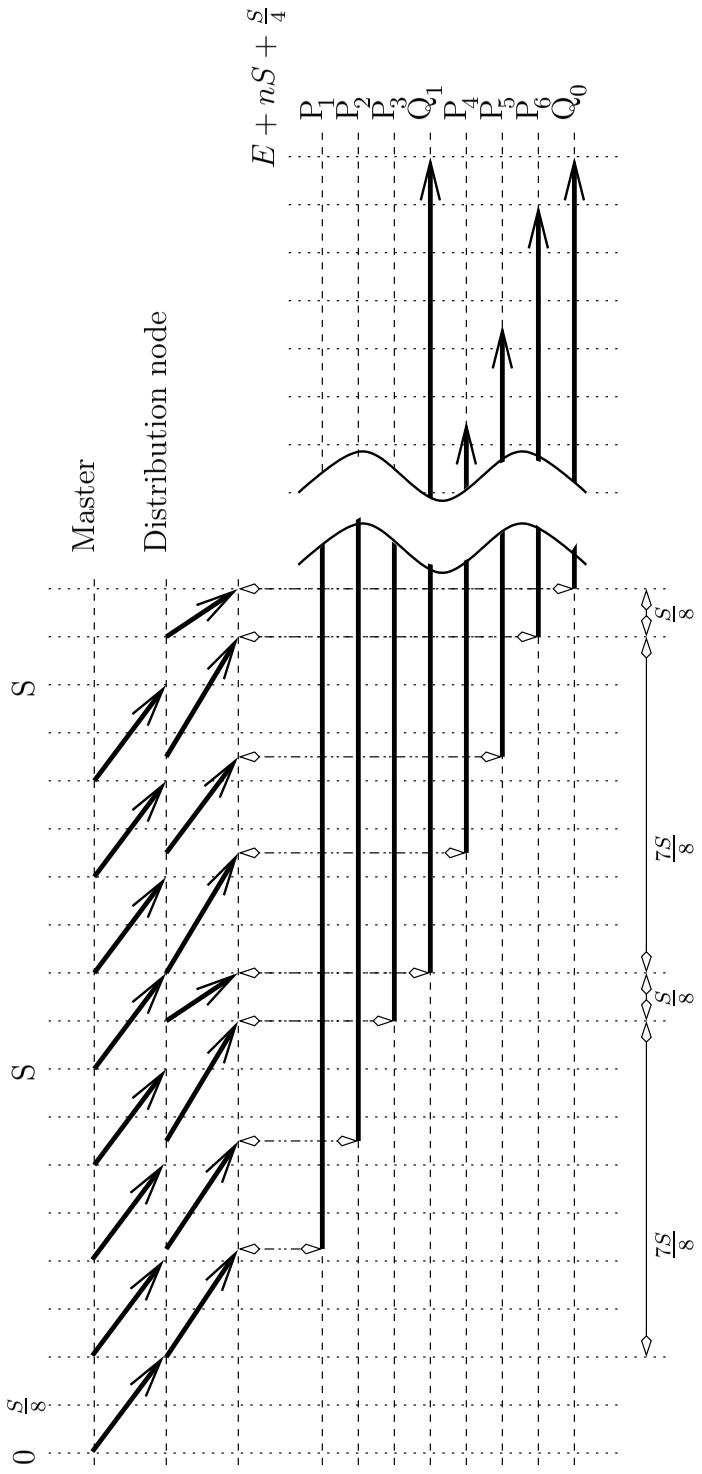

Figure 3: Scheduling 8 tasks using 3-partition

Therefore when a task is sent to a P_i processor, the time needed to send it is bigger than the time needed to get the next one from the master (as $\frac{9}{32}S > \frac{1}{4}S$). Thus there may be no idle time in the communication link if we send the first three tasks to three P_i processors.

However if we send at least one of the first three to one of the Q_i processors, one communication will be of length $\frac{1}{8}S$. The other two communications are strictly smaller than $\frac{5}{16}S$. The communication time of the three tasks is then strictly smaller than $2*(\frac{5}{16}S)+\frac{1}{8}S = \frac{3}{4}S$ which means that either there is some idle time before or between the first three communications, or that the fourth task is not ready when the third communication is completed. □

The combination of these two lemmas states that the task alloted on processor Q_{n-1} is the fourth emitted from the distribution node.

Lemma 3 *The communication time used by the first three tasks is exactly* $\frac{7}{8}S$.

The execution time of processor Q_{n-1} imply that the fourth task arrived before the time instant $\frac{5}{4}S$, which means that the communication time used by the first three tasks is at most $\frac{7}{8}S$ units of time. As the fifth task arrives on the distribution node no sooner than the time instant $\frac{5}{4}S$, and as there is no idle time on the communication link of the distribution node, the first four tasks have a total communication time of at least S. As the communication cost of the link going to Q_{n-1} is $\frac{S}{8}$, the communication time of the first three task is at least $\frac{7}{8}S$. □

Theorem 2 *Scheduling* $4n$ *tasks within* $\frac{S}{4} + nS + E$ *units of time on the tree gives a solution to the associated 3-Partition problem*

We just proved that the first three tasks were associated with a triplet of sum exactly $\frac{7}{8}S$. This property can be recursively proved for rest of the schedule, associating tasks $4j+1$, $4j+2$ and $4j+3$ to three P_i processors with communication times of sum $\frac{7}{8}S$, and task $4j+4$ to processor Q_{n-1-j}. Hence a schedule of $4n$ tasks within $\frac{S}{4} + nS + E$ time units on such tree network of processors is made with n triplets of sum $\frac{7}{8}S$. Those n triplets are the solution to the modified 3-Partition problem. □

5 Conclusion

As we said in the introduction, the sub-problems of the fork-graphs, the chains and the spiders are all polynomial. Proving the general tree structure to be NP difficult closes the complexity study for the problem of identical tasks in the master/slave one-port computation model. We can also note the fact that what makes the problem difficult here is that there is a fork node which is not the master node. The only trees where only the master is allowed to be a fork node are spider graphs. Therefore there is no other subclasses of trees left to consider between the problems known to be polynomial and those known to be NP hard in the strong sense.

Further complexity studies can be done on the special case where the degree of the tree is fixed, as this case is not covered by this proof. Practical problems

of binary trees for example might be polynomial. Another approach related to complexity is the study of inapproximability.

On the practical side the next step is to define efficient heuristics on trees and more general heterogeneous platforms. Some heuristics have already been published as in [2]. With the results on the spider graphs, those heuristics can now be compared to the optimal for the spider structure. This comparison gives an idea of the performance ratio on other structures (at least it gives a lower bound on the ratio).

Finally the model can be modified to be closer to real computation platforms. Two usual modifications are latencies between communication and strictness of the one port model. In the "strictly one port" model a node cannot send and receive at the same time. For both of these modifications, the complexity remains NP-hard in the strong sense (see the proofs in appendix A).

References

[1] Sameer Bataineh, Te-Yu Hsiung, and Thomas G. Robertazzi. Closed form solutions for bus and tree networks of processors load sharing a divisible job. *IEEE Transactions on computers*, 43(10):1184–1196, October 1994.

[2] O. Beaumont, L. Carter, J. Ferrante, A. Legrand, and Y. Robert. Bandwidth-centric allocation of independent tasks on heterogeneous platforms. In *International Parallel and Distributed Processing Symposium*, 2002. Technical report avalaible at `http://www.ens-lyon.fr/~yrobert`.

[3] O. Beaumont, A. Legrand, and Y. Robert. Static scheduling strategies for heterogeneous systems. Technical Report 2002-29, École Normale Supérieure de Lyon, July 2002. Avalaible at `http://www.ens-lyon.fr/~yrobert`.

[4] Saravut Charcranoon, Thomas G. Robertazzi, and Serge Luryi. Optimizing computing costs using divisible load analysis. *IEEE Transactions on computers*, 49(9):987–991, September 2000.

[5] Y.C. Cheng and T.G. Robertazzi. *Distributed computation for a tree network with communication delays*, volume 24, pages 700–712. 1988.

[6] P.F. Dutot. Master-slave tasking on heterogeneous processors. In *International Parallel and Distributed Processing Symposium*. IEEE Computer Society Press, April 2003.

[7] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman and Company, San Francisco, 1979.

[8] Mersenne Prime Search.
URL: `http://www.mersenne.org`.

[9] SETI at home.
URL: `http://setiathome.ssl.berkeley.edu`.

[10] Jeeho Sohn, Thomas G. Robertazzi, and Serge Luryi. Optimizing computing costs using divisible load analysis. *IEEE Transactions on parallel and distributed systems*, 9(3):225–234, March 1998.

A Extending the model

The model presented in this article can be refined in many ways to come closer to a real execution platform. Two usual modifications involves communication latencies and the impossibility to receive and send at the same time (one-port strict). Hopefully the NP-hardness proof can be extended to these two modification to the model. A short summary of the changes needed is given in the two following sections.

A.1 With latencies

In a model where there is a latency between two communications on the same link, it can be interesting to group several communications to pay the latency cost only once.

Let l be the latency before starting a communication. First, we will consider only small latencies ($l < \frac{S}{16}$). We can change the tree presented in figure 2 by removing this value l to all communication costs. This small change makes the scheduling represented in the diagram 3 feasible with latencies (the time needed by the latency plus the communication is the same).

What we need to look at in details is if this scheduling is still the only one possible, i.e. if a solution to MSTHT is still necessarily related to a solution of 3-partition.

Let us prove that grouping communications is not interesting in our problem. First we can remark that after the distribution node, the tasks cannot be grouped as at most one can be done on any node. Therefore the grouping problem arises only for the communications from the master to the distribution node. As before the total communication time needed from the distribution node to the execution nodes is nS if all the $4n$ tasks are scheduled. With the chosen deadline $D = E + nS + S/4$, this is only possible if the first task is available at time instant $S/4$ as the smallest execution time is E. Which proves that the first task is sent alone.

The second task has to be available when the first reemission from the distribution node is over. However this reemission cannot exceed $\frac{5S}{16}$, as the x_i are bounded by this value. Let j be the number of tasks sent with the second one, we have :

$$l + j * (\frac{S}{4} - l) \leq \frac{5S}{16}$$

and

$$l + j * (\frac{S}{4} - l) = j * \frac{S}{4} - (j-1) * l > j * \frac{S}{4} - (j-1)\frac{S}{16} = j * \frac{3S}{16} + \frac{S}{16}$$

which yields:

$$j < \frac{16}{3S} * (\frac{5S}{16} - \frac{S}{16}) = \frac{4}{3}$$

Which means that the second task is also emitted alone.

For the third task, the reemission of the first and second tasks takes at most $\frac{10S}{16}$, so this time we have:

$$\frac{S}{4} + l + j * (\frac{S}{4} - l) \leq \frac{10S}{16}$$

Which leads to $j < 5/3$ and proves that the third task is emitted alone. Again, for the fourth task, the reemission of the first three is no longer than $\frac{15S}{16}$, leading to $j < 6/3 = 2$, proving that the fourth one was also emitted alone.

This is obviously the last task for which this technique can be extended. However, as we proved that the first four tasks were emitted alone, the proofs of the lemmas still hold. Processor Q_{n-1} has to compute the fourth task and the first three are distributed to three P_i processors such that the total communication time is exactly $\frac{7S}{8}$.

As in the proof of Theorem 2, a recursion proves that every communication time x_i belongs to a triplet of sum $\frac{7S}{8}$.

For larger latencies, the trick is to force the grouping of communications in a way which will lead us back to long communications with small latencies. Every computing node can be transformed into a fork with k nodes, and the communications values can be changed, such that all tasks to be computed on one sub fork have to arrive at the same time (to avoid paying twice the latency at the distribution level). Therefore the grouping of size k is imposed and the latency can be made arbitrary small compared to k times the communication time.

A.2 Without communication parallelism

In this modified model, a processor may only send or receive a task. This is often considered a more realistic model as most computers have only one communication card. However this modification does not make our problem easier, as we can also prove the NP-hardness of scheduling on trees.

This proof is very close to the main proof of the paper, but requires a modification of the tree used in the reduction. To replace the communication parallelism we have to introduce some nodes between the master node and the distribution node. These nodes have fast communication links, in order to reduce the impact of their presence on the communications of the master node and of the distribution node.

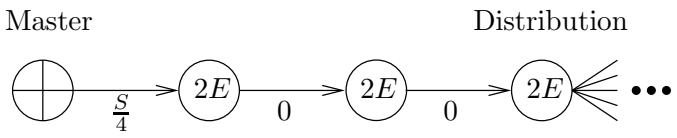

Figure 4: Tree with communications of length zero

Ideally if we can have communications of length zero units of time, the inserted nodes are as shown in figure 4. There is two inserted nodes, as the first link of length zero can only be used between two receptions from the master and the second one can only be used between two emissions from the distribution node. The diagram 3 clearly shows that on the distribution node, the end of a

reception is not always at the same instant as the end of an emission from the distribution node to one of the computing nodes.

This transformation using links with cost zero can be generalized to any node to change the graph representation of a platform where communication parallelism is allowed into a graph representation where communication parallelism is forbidden. This transformation is illustrated in figure 5.

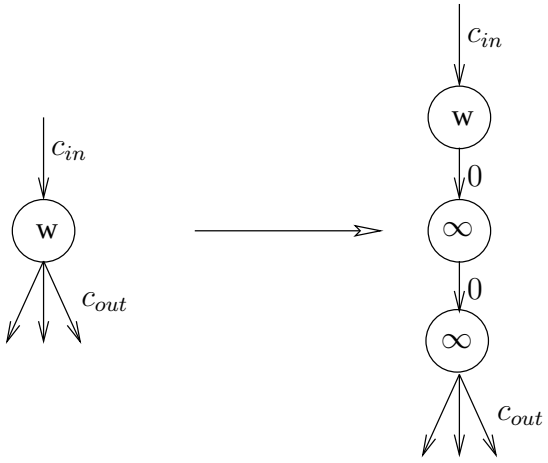

Figure 5: Changing a node into its equivalent in the strict model.

However one could argue that a communication of time zero is in conflict with the assumption of one port communication, as several of those communications could be done one after another and therefore give the impression that several tasks are sent at the same time.

This is also the reason why there is no communications of time zero in the reduction presented in section 4. Hopefully very small communication times are sufficient provided that the communication links from the master node and from the ditribution node are changed accordingly. The new tree is given in figure 6. In this figure the z_i are equal to $x_i - \frac{S}{32}$ for all i.

The modifications on the communication links are the same than in the latency model. In the figure 7, we provided the diagram of a schedule for the example used in the figure 3.

The proof is also very similar, the main difference here is that we need to prove that there are no conflicts between communications arriving to and leaving from the second node inserted after the master.

The communications arriving to this node occupies the node communication capacities during the time instants $i * \frac{S}{4} - \frac{S}{32}$ to $i * \frac{S}{4}$ for all i between 1 and $4n$. The communications leaving from this node are from (for all j between 1 and n):

- $j * \frac{S}{4}$ to $j * \frac{S}{4} + \frac{S}{32}$ for task $4j - 3$
- $j * \frac{S}{4} + x_{k_j} > j * \frac{S}{4} + \frac{9S}{32}$ to $j * \frac{S}{4} + x_{k_j} + \frac{S}{32} < j * \frac{S}{4} + \frac{11S}{32}$ for task $4j - 2$
- $j * \frac{S}{4} + x_{k_j} + x_{k_{j+1}} = j * \frac{S}{4} + (x_{k_j} + x_{k_{j+1}} + x_{k_{j+2}}) - x_{k_{j+2}} > j * \frac{S}{4} + \frac{7S}{8} - \frac{10S}{32} =$

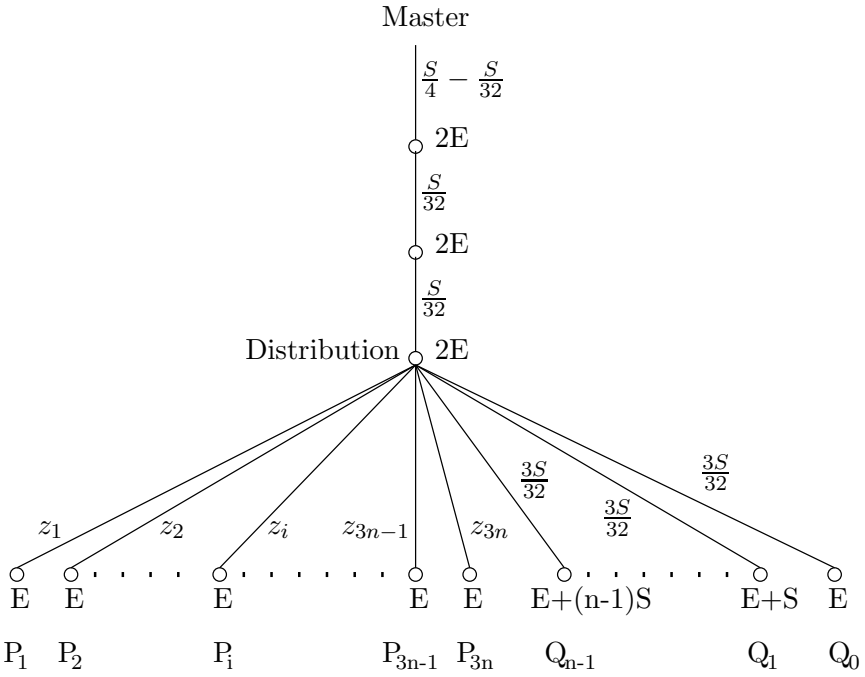

Figure 6: Tree with non zero communications

$j * \frac{S}{4} + \frac{18S}{32}$ to $j * \frac{S}{4} + x_{k_j} + x_{k_{j+1}} + \frac{S}{32} = j * \frac{S}{4} + (x_{k_j} + x_{k_{j+1}} + x_{k_{j+2}}) - x_{k_{j+2}} + \frac{S}{32} < j * \frac{S}{4} + \frac{7S}{8} - \frac{9S}{32} + \frac{S}{32} = j * \frac{S}{4} + \frac{20S}{32}$ for task $4j - 1$

- $j * \frac{S}{4} + x_{k_j} + x_{k_{j+1}} + x_{k_{j+2}} = j * \frac{S}{4} + \frac{7S}{8}$ to $j * \frac{S}{4} + x_{k_j} + x_{k_{j+1}} + x_{k_{j+2}} + \frac{S}{32} = j * \frac{S}{4} + \frac{29S}{32}$ for task $4j$

Let us define the projection in interval $[0; S/4]$ of the real number r as $p = r + k * S/4$ where k is a (possibly negative) integer. Looking at the projections of the bounds we just gave, we have the receptions between $\frac{7S}{32}$ and $\frac{S}{4}$ the emissions between 0 and $\frac{5S}{32}$, which proves that there is no overlap.

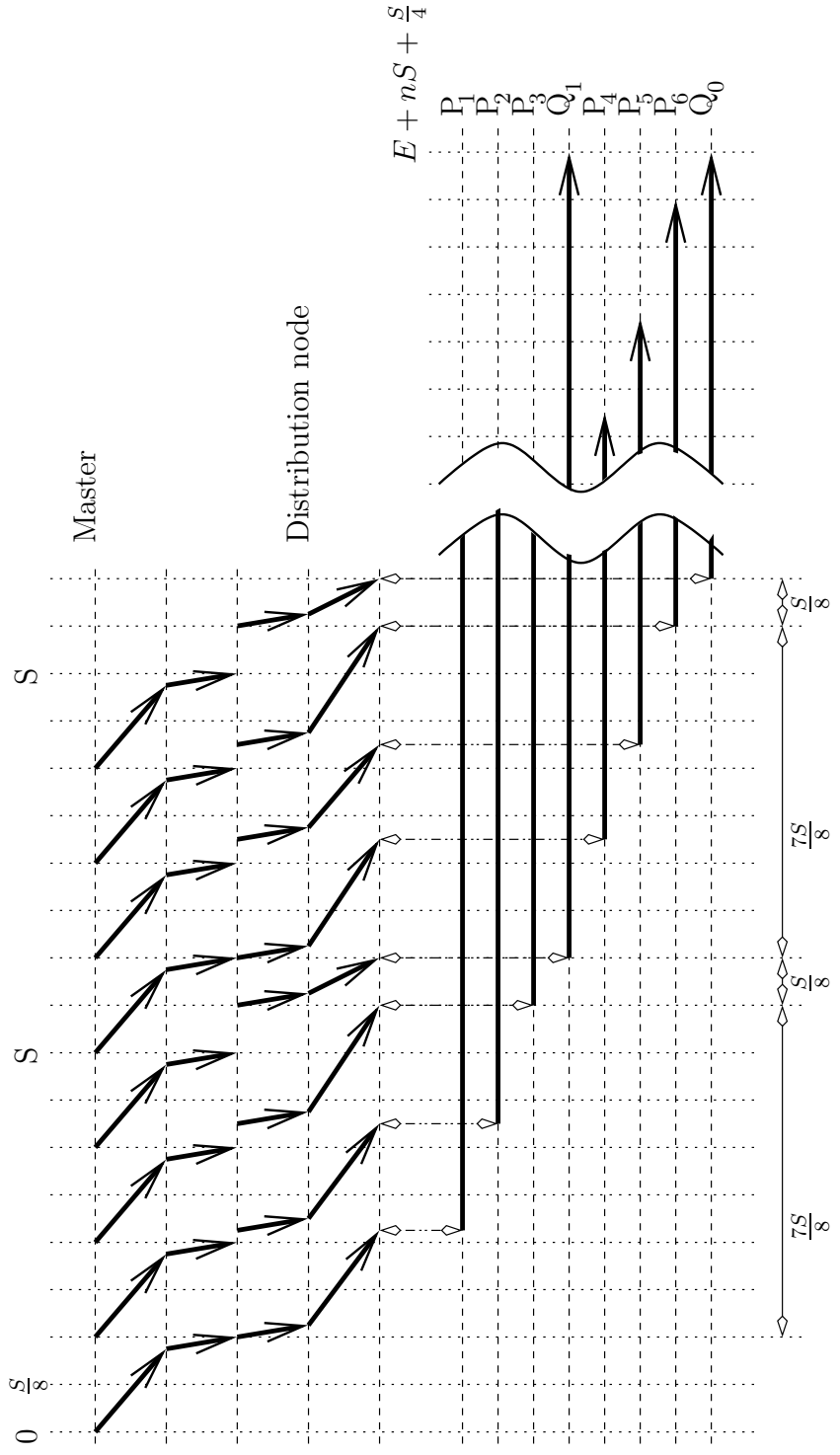

Figure 7: Scheduling 8 tasks using 3-partition with the modified tree.