International Conference on Control, Engineering & Information Technology (CEIT'13)
Proceedings Engineering & Technology - Vol.2, pp. 45-49, 2013

Scheduling of 2-Steps Graph with UET Tasks in a Heterogeneous Environment

Mounira Belmabrouk[#1], Mounir Marrakchi[*2]

[#] *Faculty of Letters and Human Sciences,*
University of Sousse, Tunisia
[1] Mounira.belmabk@isffs.rnu.tn

[*] *Department of Computer Sciences, Faculty of Sciences,*
University of Sfax, Tunisia
[2]Mounir.Marrakchi@fss.rnu.tn
marrakchimounir@yahoo.fr

***Abstract*— In this paper, we focus on scheduling the 2-steps graph with UET tasks, i.e. tasks with the same length, in a heterogeneous environment where processors are uniform and have different speeds. Given a 2-steps graph of size n and p processors, we determine the lower bound of the execution time of any scheduling by maximizing the activity of each processor. Then, we present an efficiency schedule in execution time and the experimental results without and with communication costs between processors and we compare the performance of our algorithm with existing scheduling scheme through the result of experiments.**

***Keywords*— 2-steps Graph, Scheduling, Communication, Heterogeneous Environment, Optimality.**

I. INTRODUCTION

The 2-steps graph is a precedence graph of several linear algebra algorithms such as the triangularization of a matrix, solving a triangular system ... [4], [5]. Here, we are interested in scheduling of 2-steps graph with UET (Unit Execution Time) tasks, i.e. the execution time for any task by the same processor is constant, which is the precedence graph of the algorithm solving a triangular system. In the general case, the search for an optimal solution of a scheduling problem is NP-complete. But by setting some parameters of the problem, it is possible to resolve this issue. Scheduling in a homogeneous or heterogeneous 2-steps graph has been studied by several authors [1], [2], [3], [4], [5] but for all existing schedules optimality in time parallel execution is not yet reached. This is because, firstly, that the activity of each processor is not maximized and secondly that the communication cost is not minimized. Our goal in this article is to determine firstly the lower bound of the execution time of the 2-steps graph in a heterogeneous environment and secondly to find a scheduling more efficient than the existing. Here, we minimize the schedule length, which is defined as the maximum completion time of all nodes by assuming tasks to processors while respecting the precedence constraints. This paper is organized as follows. In Section 2, we describe the related work. Thereafter, in section 3, we determine for any n the size of graph, the lower bound of the execution time of any scheduling neglecting the communication costs between processors assuming that v_j satisfies $v_p/2 \leq v_j \leq v_p$ $(1 \leq j \leq p-1)$ and p verified $2 \leq p \leq n$. Then, we present in paragraph 4 a parallel algorithm without and with communication costs in the case where $v_1-1=v_2-1= \ldots = v_\alpha-1=v_{\alpha+1=}v_{\alpha+2}=\ldots=v_p=v$, α $(1 \leq \alpha \leq p)$ and v are integers. Finally and before concluding, we compare in section 5 the complexities found experimentally with those of previous work.

II. RELATED WORK

Efficient application scheduling is critical for achieving high performance in homogeneous and heterogeneous multi-processors system. Different schedules, belonging to different classes such as list scheduling, cluster scheduling …, have been published. The schedule (HEFT) «Heterogeneous Earliest-Finish-Time» [5] is a recursive algorithm. It uses the concept of the earliest finish time of each task for ordering the tasks and assigning them to processors later. The (PETS) "Performance Effective Task Scheduling" [3] scheduling has three phases. In the first phase, scheduling (PETS) regroups the independent tasks together. In the second and three phases, (PETS) computes priority and assigns each task to the available processor. The performances of (PETS) are better than the (HEFT). The paper [6] suggest a novel approach called «Constrained Earliest Finish Time» (CEFT) to provide schedules for heterogeneous system using the concept of the constrained critical path. The experimentations results show that the (CEFT) strategy outperforms the well-known (HEFT) and (PETS) strategies.

III. LOWER BOUND OF THE EXECUTION TIME

Before calculating the lower bound of the parallel execution time of 2-steps graph with UET tasks in a heterogeneous environment, we present the following definitions. We consider G(n) the 2-steps graph with UET tasks and p processors p_1, p_2... p_p having respectively speeds

v_1, v_2... v_p. It is assumed that all v_j are integers satisfying $v_1 \le v_2 \le \ldots \le v_{p-1} \le v_p$ where $v_j \ge v_p/2$ for all j satisfying $(1 \le j \le p-1)$. Figure 1 shows G(4). The number of tasks to having the 2-steps graph is equal to $(n^2+n-2)/2$. The graph G(n) is composed of n-1 columns C_k (k = 2, .., n) where k is formed by the tasks T_{1k}, T_{2k}, ..., T_{kk}. A level i of G(n) $(1 \le i \le n-1)$ contains the n-i+1 following tasks $T_{i,i+1}$, $T_{i+1,i+1}$, $T_{i,i+2}$ …, $T_{i,n}$.

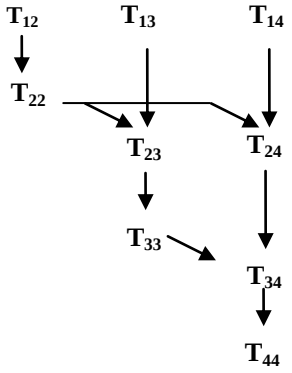

Fig. 1 The 2-steps graph for n=4.

To find the lower bound of the execution time of tasks belonging to G(n), we minimize the processor inactivity time. Our goal is to keep all processors active as long as possible. For this, two processors with two different speeds, one that has the lowest speed is finally released later at the same time than the other. As at the end of the execution of each task $T_{n-p+j,n-p+j}$ $(1 \le j \le p-1)$ a processor must release, the slowest processors become inactive firstly. On the other hand, to minimize the idle time of processors, it is necessary that all processors are active until the execution the task $T_{n-p+1,n-p+2}$ starts because at this time the number of free tasks is lower or equal to p-1. Then, the processor p_1 becomes inactive and each other processor p_j $(2 \le j \le p-1)$ executes tasks of column $C_{n-p+j+1}$for the time when p_p starts the execution of the task $T_{n-p+1,n-p+2}$. The fastest processor p_p executes the tasks of the longest path constituted by $T_{n-p+j-1,\ n-p+j}$, $T_{n-p+j,\ n-p+j}$ $(2 \le j \le p)$. For this, we partition the 2-steps graph into two regions noted R(I) and R(II) as indicated in fig. 2. The boundary curve (C) between R(I) and R(II) is the equipotential curve which contains task $T_{n-p+1,n-p+2}$. We define,

$$ch(T_{i,n-p+j+1}) = (n-p+j-i)/v_j + 2(p-j)/v_p$$

where j verified $(2 \le j \le p-1)$, as the length from the task $T_{i,n-p+j+1} \in R(II)$, to task T_{nn} by the sum of the number of tasks situating between the tasks $T_{i,n-p+j+1}$ and $T_{n-p+j-1,n-p+j+1}$and belonging to $C_{n-p+j+1} \backslash \{T_{n-p+j,n-p+j+1}, T_{n-p+j+1,n-p+j+1}\}$ divided by v_j and the number of tasks of critical path situated between $T_{n-p+j,n-p+j+1}$ and T_{nn} divided by v_p. The curve (C) is defined by the two path identified in bold on the fig. 2 are executed one (the tasks of the longest path) by p_p and the other by p_j (the tasks belonging to the column $C_{n-p+j+1}$) and p_p (the tasks belonging to the longest path) at the same time. It intercepts the column $C_{n-p+j+1}$ on the task $T_{i(j),n-p+j+1}$ where $2 \le j \le p-1$:

$$ch(T_{i(j)+1,n-p+j+1}) < ch(T_{n-p+1,n-p+2}) \le ch(T_{i(j),n-p+j})$$

$$ch(T_{i(j),n-p+j+1}) = (n-p+j-i(j))/v_j + 2(p-j)/v_p$$

$$ch(T_{i(j)+1,n-p+j+1}) = ch(T_{i(j),n-p+j+1}) - 1/v_j$$

$$ch(T_{n-p+1,n-p+2}) = 2(p-1)/v_p$$

Note that the curve (C) depends on the values for the speeds of processors.

In the following, we note S(I) (resp. S(II)) the sum of tasks or part of tasks belonging to R(I) (resp. R(II)) and we determine the lower bound for execution the region R(I) with p processors having different speeds. The value of S(I) is equal to the total number of tasks of G(n)):

$$(n^2+n-2)/2$$

minus the total number of tasks:

$$S(II) = 2\sum_{j=1}^{p}(j-1)v_j/v_p$$

belonging to R(II), then

$$S(I) = (n^2+n-2)/2 - 2\sum_{j=1}^{p}(j-1)v_j/v_p\ .$$

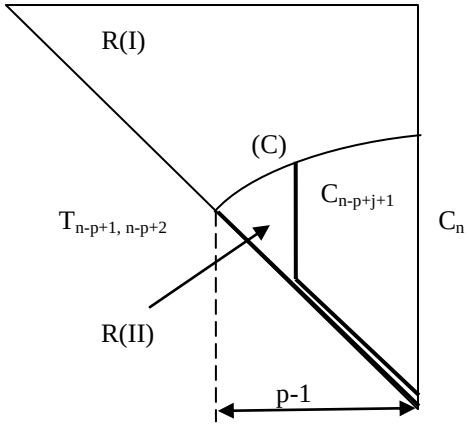

Fig. 2 The partition of the 2-steps graph.

In the following, we have the discussion for determining the lower bound of R(I):

1. If all processors are active without interruption for executing the region R(I). In this case, we can suppose that the

p processors are identical to others p processors q_1, q_2... q_p with the same speed equal to

$$v(m) = \sum_{i=1}^{p} v_i \,/\, p$$

the average of the sum of speeds. Then, S(I)/p is the number of tasks, belonging to R(I), can be executed by each processor q_j and 2(n-p) is the number of tasks constituted the longest path of R(I). Hence, $S(I)/p \geq 2(n-p)$ and then we can consider all the processors are active without interruption for executing R(I) and $LB(I) = S(I)/pv(m)$.

2. If $S(I)/p \leq 2(n-p)$ then it exists at least one processor that is idle for a period of time when executed the tasks of R(I). In this case, for minimizing the idle time of processors, the tasks of the longest path of R(I) must be executed by the faster processor p_p. And each processor p_j ($1 \leq j \leq p-1$) can be executed $2(n-p)v_j / v_p$ tasks of region R(I) private of the tasks constituted the longest path. Then, we have:

$$\Delta = S(I) - \sum_{j=1}^{p} 2(n-p)v_j / v_p$$

$$LB(I) = 2(n-p)/v_p \text{ if } \Delta \leq 0,$$

$$LB(I) = 2(n-p)/v_p + \Delta / \sum_{i=1}^{p} v_i \text{ , otherwise}$$

For the time of parallel execution of region R(II) is equal to

$$LB(II) = ch(T_{n-p+1,n-p+2}) = 2(p-1)/v_p$$

It is necessary that $v_j \geq v_p/2$ ($2 \leq j \leq p-1$). Finally, we have LB=LB(I)+LB(II) and the following results are proved:

THEOREM

The lower bound LB for executing the UET 2-steps graph G(n) with p processors having different speeds $v_1 \leq v_2 \leq \ldots \leq v_{p-1} \leq v_p$ where $v_j \geq v_p/2$ for all j satisfying ($2 \leq j \leq p-1$) is;

1. $LB = [(n^2+n-2)/2 + 2\sum_{j=1}^{p-1}(p-j)v_j / v_p] / \sum_{j=1}^{p} v_j$ if $S(I)/p \geq 2(n-p)$
2. $LB = 2(n-1)/v_p + \Delta / \sum_{i=1}^{p} v_i$ if $\Delta = S(I) - 2\sum_{j=1}^{p}(n-p)v_j / v_p \geq 0$ and if $S(I)/p \leq 2(n-p)$
3. $LB = 2(n-1)/v_p$ if $\Delta \leq 0$ and if $S(I)/p \leq 2(n-p)$

Note that the speed v_1 of processor p_1 can be quite small so that it cannot execute any task of R(I) and thereafter of G (n). In the following, we analyze the first expression of LB in the theorem where $S(I)/p \geq 2(n-p)$ posing these questions:

1. Assuming that the graph size n and the number of processors are invariable, what is the smallest value of LB when we vary the values v_j for j = 1... p and without change the value $V = \sum_{j=1}^{p} v_j$?

2. Let us n fixed and p variable, what is the smallest value of LB when the sum of the speeds V is invariable?

We note firstly that the minimum idle time of processors is equal to:

$$TL(v_1,...,v_p) = \sum_{i=1}^{p-1} 2(p-i)v_i / v_p .$$

Without lose of generality, we note LB par $LB(v_1,v_2,\ldots,v_p)$ and we suppose that $v_1 > 1$. We have then:

$$LB(v_1,v_2,...,v_p) - LB(v_1-1,v_2+1,v_3,...,v_p) = 2/(v_pV) > 0.$$

For the same values of n, p and V, the lower bound decreases if the value of the speed of any processor p_i decreases of a value that will be added to the speeds of the other processors having higher speeds than v_i. This is justified by:

$$TL(v_1,v_2,...,v_p) - TL(v_1-1,v_2+1,v_3,...,v_p) = 2/v_p > 0.$$

i.e. the minimum idle time of processors decreases when the speed of processor p_i decreases of a value added to the speed of another processor p_j where i<j. Hence, we can deduce when n, p and V are invariable that the smallest value of the lower bound is reached when $v_i=1$ for all i verified $1 \leq i \leq p-1$ et $v_p = V-(p-1)$. In this case, the value of lower bound is:

$$LB = [(n^2+n-2)/2 + p(p-1)/v_p]/V \cdot$$

On the other hand, if we vary the value of V without modify p and n, then the value of LB increases. This is because when any speed increased by a certain value the new execution of the graph G becomes faster than the former. Same for the decreasing, the value of LB also decreases if the value of V decreases. Note that when V increases free the minimum idle time TL of processors decreases and vice versa.

Let us now study the second point, by fixing the values of n and V and varying the value of p, the value of LB decreases if the value of p decreases. In fact, if we assume having p-1 processors instead p with speeds $v_1+v_2, v_3, \ldots, v_p$ ($v_1+v_2 \leq v_3$), then we have:

$$LB(v_1,v_2,...,v_p) - LB(v_1+v_2,v_3,...,v_p) = 2v_1/(v_pV) > 0.$$

We infer that if p increases (resp. decreases) then LB increases (resp. decreases). This is explained by the minimum free time of processors decreases if the number of processors p decreases and vice versa while n and V are fixed. As an example of this situation, we assume that we dispose a single processor whose its speed is equal to the sum of speeds of all processors. We find the value of sequential time:

$$LB = (n^2+n-2)/(2V)$$

which is the smallest lower bound among all the lower bounds when n and V fixed and variable p. In this case, the minimum

free time TL=0. In the remaining of this paper, we present a scheduling for executing 2-steps graph with constant tasks.

IV. CRITICAL PATH SCHEDULING (CPS)

In this section, we describe the critical path parallel algorithm which executes with p processors the tasks of 2-steps graph G(n). We assume that the execution time Ex(T) of each task T is the same equal to one time unit. The p processors noted p_1, p_2... p_p having respectively speeds v_1, v_2... v_p. It is assumed that all v_j are integers satisfying: v_1-1= v_2-1= …= v_α-1=$v_{\alpha+1=}v_{\alpha+2}$=…=$v_p$=v where α (1≤α≤p) and v are integers.

We call critical path of a task $T_{i,j}$ the path of G(n) defined by tasks $T_{i,j}$, $T_{i+1,j}$,…, $T_{j,j}$, $T_{j,j+1}$, $T_{j+1,j+1}$,…, $T_{n-1,n}$, $T_{n,n}$. Its length is denoted by cp($T_{i,j}$). It is easy to show that cp($T_{i,j}$)=2n-i-j+1 the sum of Ex(T) where T is the task belonging to critical path of T_{ij}. Generally, at time t, the critical path scheduling (CPS) executes, among independent tasks, (i.e. which predecessors have been executed), the tasks which a critical path are the maximum. (CPS) starts execution at time t=0. The order of task execution is defined by the following property (P):

The execution of an independent task $T_{i,j}$ begins at the latest at the same time as $T_{u,v}$ if c($T_{i,j}$)>cp($T_{u,v}$).

The algorithm affects the tasks of the unique longest path $T_{i,i+1}$, $T_{i+1,i+1}$ (1≤i≤n-1) to the fastest processors p_p. At each time when any processor completes execution of a task, it begins the execution of another independent task having the longest critical path. At the time where the execution of the task $T_{n-p+1,n-p+2}$ begins, the scheduling (CPS) affects at each processor p_j (2≤j≤p-1) the tasks not yet executed of column $C_{n-p+j+1}$without the tasks $T_{n-p+j,n-p+j+1}$, $T_{n-p+j+1,n-p+j+1}$ belonging to the longest path. Fore more details of (CPS), an example of scheduling is given in table 1, where columns Time and Task constitute column s and i,j (resp. t) in column Task (resp. Time) means that the processor p_s starts the execution of the task $T_{i,j}$ at time t. We have n=10, p=3, v_1=v_2=2, v_3=3 and we suppose for simplification that the length of each task T is equal to Ex(T)= $v_1v_2v_3$=12 units time. The time complexity for computation is equal to T(3)=104 units time or the lower bound is LB=696/7≈99.42.

We show that all processors are active until starts the execution of task $T_{n-p+1,n-p+2}$ and at the time the processors are located in the lower part of G(n) bordered upperly (in the large sens) by the tasks $T_{n-p+1,n-p+2}$, $T_{n-p,n-p+3}$ … $T_{n-2p+3,n}$. But at time when the execution of $T_{n-p+1,n-p+2}$ it exists at least one task not yet executed and situated above the curve (C) defined in the previous section. So, there is some gap of time between the computation time and the value of lower bound. The difference increases with the value of the number of processors p. Generally this difference is always lower than 2p/v(v+1). Then, we can deduce that the time of computation of (CPS) is LB+O(p). Remark that we can extend the (CPS) scheduling for any values of v_j that satisfies $v_p/2 \le v_j \le v_p$ (1≤j≤p-1). For evaluate the communication costs, we assume that the processors are fully connected without any regard to the link contention and scheduling of messages, i.e. any number of message passing can take place at any given time: computation can be overlapped with communication. The communication costs between two dependent tasks executed by two different processors (resp. the same processor) is equal to 1 (resp. 0) unit time without overlapped computation-communication.

TABLE 1

EXECUTION OF (CPS) FOR n=10 AND p=3.

1		**2**		**3**	
Task	**Time**	**Task**	**Time**	**Task**	**Time**
1,3	0	1,4	0	1,2	0
1,5	6	1,6	6	2,2	4
2,4	12	2,5	12	2,3	8
1,7	18	3,5	18	3,3	12
3,6	24	2,7	24	2,6	16
4,6	30	1,8	30	3,4	20
1,9	36	2,8	36	4,4	24
3,8	42	4,7	42	4,5	28
4,8	48	5,7	48	5,5	32
2,10	54	3,9	54	3,7	36
3,10	60	4,9	60	5,6	40
5,9	66	6,8	66	2,9	44
6,9	72	5,10	72	1,10	48
7,9	78	6,10	78	6,6	52
		7,10	84	6,7	56
		8,10	90	5,8	60
				7,7	64
				4,10	68
				7,8	72
				8,8	76
				8,9	84
				9,9	88
				9,10	96
				10,10	100

V. COMPARISON

In this section, we experimentally compare the different results. In the fig. 3, we present for v=4 the variation of the values of LB and the execution time of (CPS) with and without communication costs when varying the values of number of processors p. The curves show the difference

between three values the lower bound LB, the execution time without (resp. with) communication costs (CPS(1)) (resp. (CPS(2)). The difference between the execution time of (CPS(1)) and LB is due to the processors are not situated on the curve (C) at the time when the execution of the task $T_{n-p+1,n-p+2}$ starts. The fig. 4 shows that until now it has not reaches the lower bound of the execution time. On the other hand, we remark the superiority of (CPS) with communication costs compared to the existing schedulings.

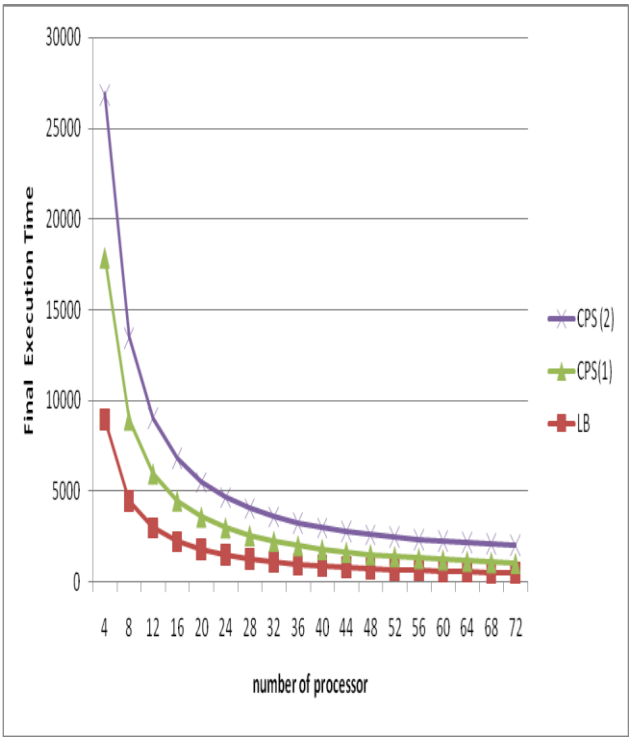

Fig. 3 Comparison of execution time of (CPS) and lower bound.

VI. CONCLUSION

In this paper, we have determined the lower bound of the execution time of any scheduling for 2-steps graph with UET tasks in the case where the speed v_j of each processor p_j verified $v_p/2 \leq v_j \leq v_p$ $(1 \leq j \leq p-1)$. Then, we have presented efficient scheduling (CPS) with p processors having different speeds. The experimental results have showed that the execution time of the (CPS) is better compared to those currently existing. It's necessary to confirm these results theoretically. In addition, it is important to note that theoretical and experimental study of the (CPS) on a cluster of processors is necessary. In other hand, it is possible to push the research to find an optimal scheduling for executing the tasks of 2-steps graph.

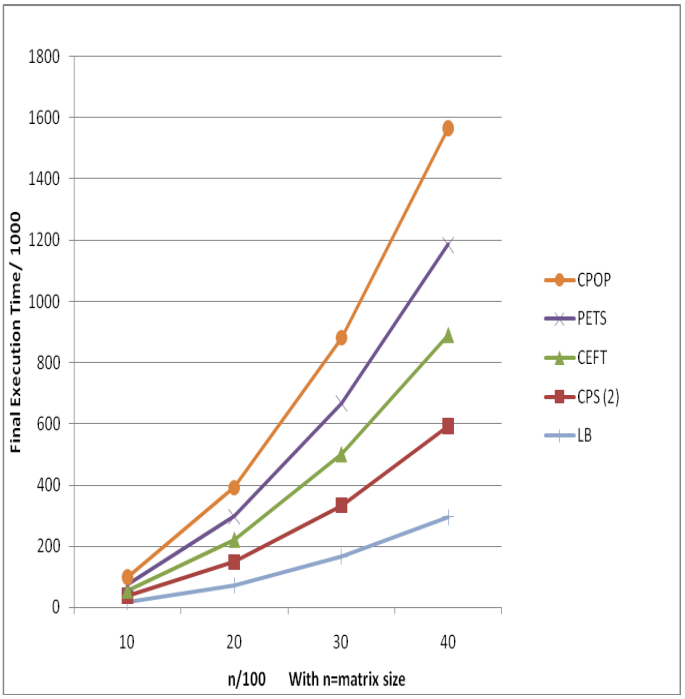

Fig. 4 Comparison of execution time.

REFERENCES

[1] R. Eswari, S. Nickolas, *Expected completion time based scheduling algorithm for heterogeneous processors*, in *IPCSIT'2011,* vol.16, 2011, pp. 72-77

[2] M. I. Daoud, N. Kharma, *A high performance algorithm for static task scheduling in heterogeneous distributed computing systems*, *Parallel Distributed Computing*, 2008, 68, pp.399-409.

[3] E. Illavarasan, P.Thambidurai, *Low complexity performance effective task scheduling algorithm for heterogeneous computing environments*. *J. Computer Science*. 2007, 3(2), pp. 94-103.

[4] S. Mingsheng, S. Shixin, W. Qingxian, *An efficient parallel scheduling algorithm of dependent tasks graphs*,in *PDCAT'2003*, 2003, pp. 595-598

[5] H. Topcuoglu, S. Hariri, M.-Y Wu, *Performance effective and low-complexity task scheduling for heterogeneous computing*, *IEEE tran. on para. and dis. systems*, vol. 13(13), 2002, pp. 260-274

[6] M. A. Khan, *Scheduling for heterogeneous systems using constrained critical paths, Parallel Computing 38*, 2012, pp. 175-193